Universidad Veracruzana
Centro de Investigación en Inteligencia Artificial

Discretización de series temporales usando un algoritmo multi-objetivo y árboles de decisión

Tesis

Para obtener el título de
Doctor en Inteligencia Artificial

Presentada por:
Aldo Márquez Grajales

Director:
Dr. Héctor Gabriel Acosta Mesa

Co-director:
Dr. Efrén Mezura Montes

Xalapa-Enríquez, Veracruz, México. 2019

Agradecimientos

En estos párrafos quiero plasmar mis más sinceros agradecimientos a todas aquellas personas que estuvieron conmigo y me apoyaron incondicionalmente para que esta meta llegara a buen término.

En primer lugar, quiero agradecer al Dr. Héctor Gabriel Acosta Mesa y al Dr. Efrén Mezura Montes, por su paciencia y acertada dirección para cumplir con los objetivos de este trabajo.

Al Dr. Mario Graff Guerrero, a sus colaboradores y a los estudiantes del Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación (INFOTEC) Aguascalientes por sus consejos, apoyo y confianza al recibirme en ese instituto durante mi estancia doctoral.

Al Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación (INFOTEC) Aguascalientes, por facilitar su infraestructura tecnológica para la ejecución del diseño experimental presentado en este documento.

Al Centro de Investigación en Inteligencia Artificial (CIIA) y al Consejo Nacional de Ciencia y Tecnología (CONACYT), por las facilidades y recursos económicos brindados para la realización de esta investigación.

A mis padres, que sin su apoyo jamás hubiera concretado ninguno de los objetivos de vida que me he planteado.

No puedo dejar de lado a mis amigos, tanto mis amigos y compañeros del doctorado como a mis amigos y familiares de mi pueblo natal. Especialmente quiero agradecer a Nancy, quien siempre me apoyó y ayudó en todas las dudas que me surgían a lo largo del desarrollo de este proyecto.

A Fernando, quien siempre estuvo ahí para levantar los ánimos cuando éstos iban en picada.

A todos y cada una de las personas que han estado a mi lado brindándome su apoyo incondicionalmente, GRACIAS. No me alcanzan las líneas para mencionar a todos, pero desde el fondo de mi corazón *Gracias*.

Abstract

In this document, a symbolic discretization algorithm for time series is presented based on three functions: classification, complexity and information loss. This algorithm is called *enhanced multi-objective approach for the symbolic discretization of time series (eMODiTS)*. eMODiTS defines a flexible or adaptive discretization scheme for the data; that is, it defines different alphabets schemes for each word segment, allowing adaptation to the temporal information found in each one. The mechanism used by eMODiTS to find the optimal value of each of the cuts is the well-known multi-objective evolutionary algorithm called *Non-dominated Sorting Genetic Algorithm II (NSGA-II)*.

Since NSGA-II is a multi-objective algorithm, a set of potential solutions (called *Pareto Set*) to a problem is found instead of just one, so it is necessary to define a preference selection method to obtain the final solution. In this work, four methods were proposed to perform this task: the knee method, the CV method, the k1 method (k-means with $k = 1$) and the K20 method (a hybrid between CV and k-means). Final solutions were evaluated based on the misclassification rate calculated through the decision tree classifier. Subsequently, nine methods (based on the well-known SAX discretization algorithm) were compared against eMODiTS: EP, SAX, *alpha* SAX, ESAX, ESAXKMeans, 1D-SAX, RKMeans, SAXKMeans and rSAX.

Every comparison was performed using the 85 temporary databases of the UCR repository. The results suggest that our proposal finds an adequate discretization scheme with respect to the classification, reduction of dimensionality and information loss, even outperforming some SAX-based methods against which it was compared.

Moreover, eMODiTS provides a graphical form to visualize the relevant regions, relationships or patterns within the temporary databases, through the decision tree used to evaluate the final solution selected from the Pareto Set. This feature is useful for the information owner in making the right decisions in a timely manner.

Resumen

En el presente documento se presenta un algoritmo de discretización de series de tiempo, llamado **Discretización Simbólica Multi-objetivo Mejorada de Series Temporales** (***e**nhanced **M**ulti-objective symb**O**lic **D**iscretization for **T**ime **S**eries, eMODiTS* en inglés), basado en tres características esenciales para la discretización de bases de datos temporales: clasificación, complejidad y pérdida de la información. El método eMODiTS define un esquema de discretización flexible y adaptativo a los datos, es decir, define esquemas de cortes de alfabetos diferentes por cada segmento o corte de palabra, permitiendo adaptarse a la información temporal encontrada en cada corte de palabra. El mecanismo empleado por eMODiTS para encontrar los valores óptimos de cada uno de los cortes, es el algoritmo evolutivo multi-objetivo llamado *Non-dominated Sorting Genetic Algorithm II (NSGA-II)*.

NSGA-II, al ser un algoritmo multi-objetivo, encuentra un conjunto de soluciones potenciales (llamado *Conjunto de Óptimos de Pareto*) a un problema en lugar de una sola, por lo que, es necesario definir un método de selección de preferencias para obtener la solución final de dicho conjunto. En este trabajo, se proponen cuatro métodos para realizar dicha tarea: el método de la rodilla (knee), el método CV, el método k1 (k-means con $k = 1$) y el método K20 (híbrido entre CV y k-means). Las soluciones finales fueron evaluadas en función de la tasa de clasificación errónea calculada a través del clasificador de árbol de decisión. Posteriormente, eMODiTS fue comparado contra nueve métodos basados en el proceso de discretización definido por el algoritmo llamado *Symbolic Aggregate Approximation (SAX)*. Dichos métodos fueron: EP, αSAX, ESAX, ESAXKMeans, 1D-SAX, RKMeans, SAXKMeans, rSAX y el propio SAX.

Cada una de las comparaciones fueron efectuadas usando las 85 bases de datos temporales del repositorio UCR. Los resultados sugieren que nuestra propuesta encuentra un esquema de discretización competitivo con respecto a la clasificación, la reducción de la dimensionalidad y la pérdida de información, superando incluso a varios métodos basados en SAX con los que fue comparado.

Adicionalmente, eMODiTS proporciona una forma gráfica de visualizar las regiones, relaciones o patrones relevantes dentro de las bases de datos temporales, mediante el árbol de decisión usado para evaluar las solución final seleccionada del conjunto de óptimos de Pareto. Dicha característica es de utilidad para entender el comportamiento de los datos y, por consiguiente, ayudar al usuario en la toma de decisiones oportunas ante eventos inesperados.

Índice general

Índice de figuras xv

Índice de tablas xxxi

Lista de Símbolos xxxiii

1. Introducción 1
- 1.1. Introducción . . . 1
- 1.2. Definición del Problema . . . 2
- 1.3. Motivación . . . 4
- 1.4. Hipótesis . . . 6
- 1.5. Objetivos . . . 7
  - 1.5.1. General . . . 7
  - 1.5.2. Específicos . . . 7
- 1.6. Justificación . . . 8
- 1.7. Trabajo Relacionado . . . 9
  - 1.7.1. Calibración del Tamaño del Alfabeto y/o Palabra . . . 10
  - 1.7.2. Pérdida de Información . . . 12
- 1.8. Contribuciones . . . 14
- 1.9. Organización del Documento . . . 15

2. Minería de Datos en Series Temporales 17
- 2.1. Introducción . . . 17
  - 2.1.1. Tipos de Datos . . . 19
  - 2.1.2. Procesamiento de la Información . . . 21
  - 2.1.3. Tareas de Minería de Datos . . . 23
- 2.2. Discretización de Series de Tiempo . . . 25
  - 2.2.1. Reducción de Dimensionalidad . . . 25
  - 2.2.2. Discretización Simbólica . . . 26
- 2.3. Medidas de Similitud o Distancia . . . 30
  - 2.3.1. Distancia Euclidiana . . . 31
  - 2.3.2. Alineamiento Temporal Dinámico (*Dynamic Time Warping, DTW*) . . . 31

2.3.3. Subsecuencia Común más Larga (*The Longest Common Subsequence*) . . . 33
2.4. Clasificación en Series de Tiempo . . . 34
2.4.1. k-NN: Vecinos más Cercanos . . . 35
2.4.2. Clasificador Bayesiano Ingenuo . . . 36
2.4.3. Árboles de Decisión . . . 37
2.5. Agrupamiento (*Clustering*) en Series de Tiempo . . . 41
2.5.1. K-Medias . . . 41

3. Optimización Multi-Objetivo 43
3.1. Introducción . . . 43
3.1.1. Dominancia de Pareto . . . 45
3.1.2. Frente y Conjunto Óptimo de Pareto . . . 46
3.1.3. Selección de Preferencias . . . 47
3.2. Métodos Clásicos para Optimización Multi-objetivo . . . 48
3.2.1. Suma Ponderada de Funciones (SPF) . . . 49
3.2.2. Método ϵ-Constraint . . . 49
3.2.3. Método de Programación por Metas . . . 50
3.2.4. Método Lexicográfico . . . 51
3.2.5. Método de Obtención de Metas . . . 52
3.3. Algoritmos evolutivos para optimización multi-objetivo . . . 52
3.3.1. Basados en Pareto . . . 52
3.3.2. Basados en Descomposición . . . 55
3.3.3. Basados en Métricas . . . 56
3.4. Non-dominated Sorting Genetic Algorithm II (NSGA-II) . . . 56

4. Propuesta 61
4.1. Introducción . . . 61
4.2. Esquemas de Discretización con Múltiples Alfabetos . . . 63
4.3. Algoritmo Evolutivo Multi-objetivo . . . 65
4.3.1. Codificación de Esquemas de Discretización con Múltiples Alfabetos . . . 65
4.3.2. Funciones de Evaluación . . . 66
4.3.3. Operador de Cruza Adaptado . . . 68
4.4. Selección de Preferencias . . . 69
4.5. Evaluación del Esquema de Discretización Final . . . 70

5. Experimentación **73**
5.1. Introducción . . . 73
5.2. Bases de Datos Temporales . . . 75
5.3. Evaluación de Métodos de Selección de una Solución del Frente de Pareto . . . 78
5.3.1. Resultados . . . 78
5.3.2. Discusión . . . 80
5.4. Comparación Contra Otros Métodos Simbólicos . . . 81
5.4.1. Introducción . . . 81
5.4.2. Resultados . . . 82
5.4.3. Discusión . . . 95
5.5. Interpretación Gráfica . . . 99
5.5.1. Introducción . . . 99
5.5.2. Resultados . . . 100
5.5.3. Discusión . . . 100

6. Conclusiones y Trabajo Futuro **103**

Apéndices **106**

A. Árboles de decisión **109**
A.1. Adiac . . . 111
A.2. ArrowHead . . . 111
A.3. Beef . . . 112
A.4. BeetleFly . . . 112
A.5. BirdChicken . . . 113
A.6. Car . . . 113
A.7. CBF . . . 114
A.8. ChlorineConcentration . . . 114
A.9. CinCECGtorso . . . 115
A.10. Coffee . . . 115
A.11. Computers . . . 116
A.12. CricketX . . . 116
A.13. CricketY . . . 116
A.14. CricketZ . . . 116
A.15. DiatomSizeReduction . . . 117
A.16. DistalPhalanxOutlineAgeGroup . . . 117
A.17. DistalPhalanxOutlineCorrect . . . 118
A.18. DistalPhalanxTW . . . 118
A.19. Earthquakes . . . 118

A.20.ECG200 119
A.21.ECG5000 119
A.22.ECGFiveDays 120
A.23.ElectricDevices 120
A.24.FaceAll 120
A.25.FaceFour 121
A.26.FacesUCR 121
A.27.FiftyWords 121
A.28.Fish 121
A.29.FordA 122
A.30.FordB 122
A.31.GunPoint 122
A.32.Ham 123
A.33.HandOutlines 123
A.34.Haptics 124
A.35.Herring 124
A.36.InlineSkate 125
A.37.InsectWingbeatSound 125
A.38.ItalyPowerDemand 125
A.39.LargeKitchenAppliances 126
A.40.Lighting2 126
A.41.Lighting7 127
A.42.Mallat 127
A.43.Meat 128
A.44.MedicalImages 128
A.45.MiddlePhalanxOutlineAgeGroup 129
A.46.MiddlePhalanxOutlineCorrect 129
A.47.MiddlePhalanxTW 130
A.48.MoteStrain 130
A.49.NonInvasiveFetalECGThorax1 130
A.50.NonInvasiveFetalECGThorax2 130
A.51.OliveOil 131
A.52.OSULeaf 131
A.53.PhalangesOutlinesCorrect 131
A.54.Phoneme 132
A.55.Plane 132
A.56.ProximalPhalanxOutlineAgeGroup 133
A.57.ProximalPhalanxOutlineCorrect 133
A.58.ProximalPhalanxTW 134

A.59.RefrigerationDevices 134
A.60.ScreenType 135
A.61.ShapeletSim 135
A.62.ShapesAll 135
A.63.SmallKitchenAppliances 136
A.64.SonyAIBORobotSurface1 136
A.65.SonyAIBORobotSurface2 137
A.66.StarLightCurves 137
A.67.Strawberry 138
A.68.SwedishLeaf 138
A.69.Symbols 138
A.70.SyntheticControl 139
A.71.ToeSegmentation1 139
A.72.ToeSegmentation2 139
A.73.Trace 140
A.74.TwoLeadECG 140
A.75.TwoPatterns 141
A.76.UWaveGestureLibraryAll 141
A.77.UWaveGestureLibraryX 141
A.78.UWaveGestureLibraryY 141
A.79.UWaveGestureLibraryZ 141
A.80.Wafer 142
A.81.Wine 142
A.82.WordSynonyms 143
A.83.Worms 143
A.84.WormsTwoClass 143
A.85.Yoga 143

B. Distribución de clases **145**
B.1. Adiac 147
B.2. ArrowHead 148
B.3. Beef 148
B.4. BeetleFly 149
B.5. BirdChicken 149
B.6. Car 150
B.7. CBF 150
B.8. ChlorineConcentration 151
B.9. CinCECGtorso 151
B.10.Coffee 152

B.11.Computers 152
B.12.CricketX 153
B.13.CricketY 153
B.14.CricketZ 154
B.15.DiatomSizeReduction 154
B.16.DistalPhalanxOutlineAgeGroup 155
B.17.DistalPhalanxOutlineCorrect 155
B.18.DistalPhalanxTW 156
B.19.Earthquakes 156
B.20.ECG200 157
B.21.ECG5000 157
B.22.ECGFiveDays 158
B.23.ElectricDevices 158
B.24.FaceAll 159
B.25.FaceFour 159
B.26.FacesUCR 160
B.27.FiftyWords 160
B.28.Fish 161
B.29.FordA 161
B.30.FordB 162
B.31.GunPoint 162
B.32.Ham 163
B.33.HandOutlines 163
B.34.Haptics 164
B.35.Herring 164
B.36.InlineSkate 165
B.37.InsectWingbeatSound 165
B.38.ItalyPowerDemand 166
B.39.LargeKitchenAppliances 166
B.40.Lighting2 167
B.41.Lighting7 167
B.42.Mallat 168
B.43.Meat 168
B.44.MedicalImages 169
B.45.MiddlePhalanxOutlineAgeGroup 169
B.46.MiddlePhalanxOutlineCorrect 170
B.47.MiddlePhalanxTW 170
B.48.MoteStrain 171
B.49.NonInvasiveFetalECGThorax1 171

B.50.NonInvasiveFetalECGThorax2 172
B.51.OliveOil 172
B.52.OSULeaf 173
B.53.PhalangesOutlinesCorrect 173
B.54.Phoneme 174
B.55.Plane 174
B.56.ProximalPhalanxOutlineAgeGroup 175
B.57.ProximalPhalanxOutlineCorrect 175
B.58.ProximalPhalanxTW 176
B.59.RefrigerationDevices 176
B.60.ScreenType 177
B.61.ShapeletSim 177
B.62.ShapesAll 178
B.63.SmallKitchenAppliances 178
B.64.SonyAIBORobotSurface1 179
B.65.SonyAIBORobotSurface2 179
B.66.StarLightCurves 180
B.67.Strawberry 180
B.68.SwedishLeaf 181
B.69.Symbols 181
B.70.SyntheticControl 182
B.71.ToeSegmentation1 182
B.72.ToeSegmentation2 183
B.73.Trace 183
B.74.TwoLeadECG 184
B.75.TwoPatterns 184
B.76.UWaveGestureLibraryAll 185
B.77.UWaveGestureLibraryX 185
B.78.UWaveGestureLibraryY 186
B.79.UWaveGestureLibraryZ 186
B.80.Wafer 187
B.81.Wine 187
B.82.WordSynonyms 188
B.83.Worms 188
B.84.WormsTwoClass 189
B.85.Yoga 189

Bibliografía **191**

Índice de figuras

1.1. (a) Esquema de discretización del componente temporal. (b) Discretización simbólica de series de tiempo. 3
1.2. Clasificación general de los métodos de representación simbólica basada en las limitaciones del algoritmo SAX 9

2.1. Proceso de descubrimiento de conocimiento 19
2.2. Clasificación de los diferentes tipos de datos. 20
2.3. Categorización de las tareas de la Minería de Datos (MD). 25
2.4. Ejemplo de reducción de dimensionalidad usando el método PAA . 27
2.5. Ejemplo de discretización de series de tiempo continuas a series discretas simbólicas mediante el método de SAX 29
2.6. Comparación de alineación de series de tiempo usando las medidas de similitud: (a) distancia euclidiana, (b) DTW, y (c) LCS. Las lineas entre las dos series temporales representan la correspondencia encontrada entre cada serie. 33
2.7. Ejemplo del funcionamiento del algoritmo k-NN, para $k = 5$. La forma blanca en la Figura (a) representa el caso desconocido, $d_i, i = \{1, \ldots, 5\}$ representan las distancias de ese punto a cada caso conocido. La Figura (b) muestra como el caso desconocido tomó la forma de la figura más cercana encontrada. 35
2.8. Elementos de un árbol de decisión. 38
2.9. Ejemplificación del proceso de agrupamiento de instancias, (a) datos antes de la generación de los grupos, y (b) datos agrupados de acuerdo al parecido entre ellos. 41

3.1. Ejemplo de dominancia de Pareto en un problema con dos funciones objetivo, donde x_2, x_3, y x_6 dominan a las demás soluciones en todos los objetivos. 46
3.2. Representación del conjunto y frente de Pareto. Los círculos representan las mejores soluciones vistas tanto en el conjunto de Pareto, llamadas soluciones óptimas de Pareto (a), como en el frente de Pareto conocidas como soluciones no dominadas (b). 47

3.3. Breve clasificación de los algoritmos evolutivos multi-objetivo existentes en la literatura 53

4.1. Metodología general de eMODiTS. Rectángulos con líneas discontinuas representan las modificaciones realizadas al algoritmo evolutivo multi-objetivo para lidiar con esquemas de discretización con múltiples alfabetos. 63
4.2. Ejemplo de un esquema de discretización generado por eMODiTS para la base temporal llamada Beef obtenida de [27]. 64
4.3. Codificación de un esquema de discretización donde cada segmento de palabra (w_j) es incluido seguido de su correspondiente esquema de alfabetos (a_{w_j}). 66
4.4. Operador de cruza implementado en eMODiTS y basado en el operador de cruza de un sólo punto. Las posiciones de los cortes en cada padre son representados por una línea punteada. 69
4.5. Métodos se selección de preferencias propuestos para escoger una solución dentro del frente de Pareto encontrado por eMODiTS. (a) Método de la rodilla, donde el punto $\vec{x}_2$ es la solución más cercana al punto de referencia, y por lo tanto es el punto seleccionado. (b) Método CV, donde el punto $\vec{x}_2$ es la solución seleccionada debido a que presenta el menor error en clasificación E. (c) Método KM1, donde el punto $\vec{x}_2$ es la solución más cercana al punto medio (centroide) C del grupo formado por k-medias con $k = 1$ y es la elegida para ser reportada por eMODiTS. (d) Método KM20, donde la solución $\vec{x}_2$ es la que menor error en clasificación E obtuvo de las soluciones cercanas a sus respectivos centroides. 71

5.1. (a) Rango promedio obtenido por cada método de selección de preferencias. (b) Resultado de aplicar la prueba post hoc de Nemenyi usando la tasa de error en clasificación mínima obtenida en cada método. 80
5.2. Rango promedio obtenido al clasificar las 85 bases de datos temporales de prueba por cada método basado en SAX. 82
5.3. Resultados obtenidos al aplicar la prueba estadística post hoc de Nemenyi usando los mejores valores encontrados por cada método en cada base de datos. 88
5.4. Resultados estadísticos obtenidos al aplicar la prueba de suma de rangos por pares de Wilcoxon con un 95 % de confianza, comparando eMODiTS en contra de EP, SAX, ESAX, rSAX, 1D-SAX y αSAX. 89

5.5. Tasas de compresión obtenidas por eMODiTS y los métodos basados en SAX en cada una de las bases de datos del repositorio UCR. . . 90
5.6. Resultados obtenidos por cada método al analizar la pérdida de información incurrida en cada base de datos temporal, presentando (a) los rangos promedios obtenidos por cada enfoque, y (b) los resultados estadísticos al aplicar la prueba post hoc de Nemenyi. 93
5.7. Rangos promedios logrados por cada método basados en SAX en 85 bases de datos temporales para (a) clases balanceadas y no balanceadas, (b) clases separadas y traslapadas, (c) intensidad de ruido. 94
5.8. Tiempo empleado por eMODiTS en cada base de datos temporal. La dimensionalidad de las series de tiempo es calculada multiplicando el tamaño de las series de tiempo por el número de series en el conjunto de datos. 95
5.9. (a) Árbol de decisión obtenido por eMODiTS para la base de datos BeetleFly. (b) Distribución de las clases para la bases de datos BeetleFly donde cada rectángulo representa un nodo hoja del árbol de decisión. 100
5.10. (a) Árbol de decisión obtenido por eMODiTS para la base de datos GunPoint. (b) Distribución de las clases para la base de datos GunPoint donde cada rectángulo representa un nodo hoja del árbol de decisión. 100
5.11. Árbol de decisión obtenido por eMODiTS para la base de datos ItalyPowerDemand. (b) Distribución de las clases para la base de datos ItalyPowerDemand donde cada rectángulo representa un nodo hoja del árbol de decisión. 101

A.1. Árbol de decisión obtenido por eMODiTS para la base de datos Adiac. 111
A.2. Árbol de decisión obtenido por eMODiTS para la base de datos ArrowHead. 111
A.3. Árbol de decisión obtenido por eMODiTS para la base de datos Beef. 112
A.4. Árbol de decisión obtenido por eMODiTS para la base de datos BeetleFly. 112
A.5. Árbol de decisión obtenido por eMODiTS para la base de datos BirdChicken. 113
A.6. Árbol de decisión obtenido por eMODiTS para la base de datos Car. 113
A.7. Árbol de decisión obtenido por eMODiTS para la base de datos CBF. 114
A.8. Árbol de decisión obtenido por eMODiTS para la base de datos ChlorineConcentration. 114

A.9. Árbol de decisión obtenido por eMODiTS para la base de datos CinCECGtorso. 115
A.10. Árbol de decisión obtenido por eMODiTS para la base de datos Coffee. 115
A.11. Árbol de decisión obtenido por eMODiTS para la base de datos Computers. 116
A.12. Árbol de decisión obtenido por eMODiTS para la base de datos CricketX. 116
A.13. Árbol de decisión obtenido por eMODiTS para la base de datos CricketY. 116
A.14. Árbol de decisión obtenido por eMODiTS para la base de datos CricketZ. 116
A.15. Árbol de decisión obtenido por eMODiTS para la base de datos DiatomSizeReduction. 117
A.16. Árbol de decisión obtenido por eMODiTS para la base de datos DistalPhalanxOutlineAgeGroup. 117
A.17. Árbol de decisión obtenido por eMODiTS para la base de datos DistalPhalanxOutlineCorrect. 118
A.18. Árbol de decisión obtenido por eMODiTS para la base de datos DistalPhalanxTW. 118
A.19. Árbol de decisión obtenido por eMODiTS para la base de datos Earthquakes. 118
A.20. Árbol de decisión obtenido por eMODiTS para la base de datos ECG200. 119
A.21. Árbol de decisión obtenido por eMODiTS para la base de datos ECG5000. 119
A.22. Árbol de decisión obtenido por eMODiTS para la base de datos ECGFiveDays. 120
A.23. Árbol de decisión obtenido por eMODiTS para la base de datos FaceFour. 121
A.24. Árbol de decisión obtenido por eMODiTS para la base de datos Fish. 121
A.25. Árbol de decisión obtenido por eMODiTS para la base de datos GunPoint. 122
A.26. Árbol de decisión obtenido por eMODiTS para la base de datos Ham. 123
A.27. Árbol de decisión obtenido por eMODiTS para la base de datos HandOutlines. 123
A.28. Árbol de decisión obtenido por eMODiTS para la base de datos Haptics. 124
A.29. Árbol de decisión obtenido por eMODiTS para la base de datos Herring. 124

A.30. Árbol de decisión obtenido por eMODiTS para la base de datos InlineSkate. 125
A.31. Árbol de decisión obtenido por eMODiTS para la base de datos InsectWingbeatSound. 125
A.32. Árbol de decisión obtenido por eMODiTS para la base de datos ItalyPowerDemand. 125
A.33. Árbol de decisión obtenido por eMODiTS para la base de datos LargeKitchenAppliances. 126
A.34. Árbol de decisión obtenido por eMODiTS para la base de datos Lighting2. 126
A.35. Árbol de decisión obtenido por eMODiTS para la base de datos Lighting7. 127
A.36. Árbol de decisión obtenido por eMODiTS para la base de datos Mallat. 127
A.37. Árbol de decisión obtenido por eMODiTS para la base de datos Meat. 128
A.38. Árbol de decisión obtenido por eMODiTS para la base de datos MedicalImages. 128
A.39. Árbol de decisión obtenido por eMODiTS para la base de datos MiddlePhalanxOutlineAgeGroup. 129
A.40. Árbol de decisión obtenido por eMODiTS para la base de datos MiddlePhalanxOutlineCorrect. 129
A.41. Árbol de decisión obtenido por eMODiTS para la base de datos MiddlePhalanxTW. 130
A.42. Árbol de decisión obtenido por eMODiTS para la base de datos MoteStrain. 130
A.43. Árbol de decisión obtenido por eMODiTS para la base de datos OliveOil. 131
A.44. Árbol de decisión obtenido por eMODiTS para la base de datos OSULeaf. 131
A.45. Árbol de decisión obtenido por eMODiTS para la base de datos PhalangesOutlinesCorrect. 131
A.46. Árbol de decisión obtenido por eMODiTS para la base de datos Plane. 132
A.47. Árbol de decisión obtenido por eMODiTS para la base de datos ProximalPhalanxOutlineAgeGroup. 133
A.48. Árbol de decisión obtenido por eMODiTS para la base de datos ProximalPhalanxOutlineCorrect. 133
A.49. Árbol de decisión obtenido por eMODiTS para la base de datos ProximalPhalanxTW. 134
A.50. Árbol de decisión obtenido por eMODiTS para la base de datos RefrigerationDevices. 134

A.51. Árbol de decisión obtenido por eMODiTS para la base de datos ScreenType. 135
A.52. Árbol de decisión obtenido por eMODiTS para la base de datos ShapeletSim. 135
A.53. Árbol de decisión obtenido por eMODiTS para la base de datos SmallKitchenAppliances. 136
A.54. Árbol de decisión obtenido por eMODiTS para la base de datos SonyAIBORobotSurface1. 136
A.55. Árbol de decisión obtenido por eMODiTS para la base de datos SonyAIBORobotSurface2. 137
A.56. Árbol de decisión obtenido por eMODiTS para la base de datos StarLightCurves. 137
A.57. Árbol de decisión obtenido por eMODiTS para la base de datos Strawberry. 138
A.58. Árbol de decisión obtenido por eMODiTS para la base de datos Symbols. 138
A.59. Árbol de decisión obtenido por eMODiTS para la base de datos SyntheticControl. 139
A.60. Árbol de decisión obtenido por eMODiTS para la base de datos ToeSegmentation1. 139
A.61. Árbol de decisión obtenido por eMODiTS para la base de datos ToeSegmentation2. 139
A.62. Árbol de decisión obtenido por eMODiTS para la base de datos Trace. 140
A.63. Árbol de decisión obtenido por eMODiTS para la base de datos TwoLeadECG. 140
A.64. Árbol de decisión obtenido por eMODiTS para la base de datos UWaveGestureLibraryAll. 141
A.65. Árbol de decisión obtenido por eMODiTS para la base de datos UWaveGestureLibraryY. 141
A.66. Árbol de decisión obtenido por eMODiTS para la base de datos Wafer. 142
A.67. Árbol de decisión obtenido por eMODiTS para la base de datos Wine. 142
A.68. Árbol de decisión obtenido por eMODiTS para la base de datos WordSynonyms. 143
A.69. Árbol de decisión obtenido por eMODiTS para la base de datos Worms. 143
A.70. Árbol de decisión obtenido por eMODiTS para la base de datos WormsTwoClass. 143

B.1. Distribución de las clases para la base de datos Adiac extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 147

B.2. Distribución de las clases para la base de datos ArrowHead extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 148
B.3. Distribución de las clases para la base de datos Beef extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 148
B.4. Distribución de las clases para la base de datos BeetleFly extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 149
B.5. Distribución de las clases para la base de datos BirdChicken extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 149
B.6. Distribución de las clases para la base de datos Car extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 150
B.7. Distribución de las clases para la base de datos CBF extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 150
B.8. Distribución de las clases para la base de datos ChlorineConcentration extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 151
B.9. Distribución de las clases para la base de datos CinCECGtorso extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 151
B.10. Distribución de las clases para la base de datos Coffee extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 152
B.11. Distribución de las clases para la base de datos Computers extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 152

B.12. Distribución de las clases para la base de datos CricketX extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 153
B.13. Distribución de las clases para la base de datos CricketY extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 153
B.14. Distribución de las clases para la base de datos CricketZ extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 154
B.15. Distribución de las clases para la base de datos DiatomSizeReduction extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 154
B.16. Distribución de las clases para la base de datos DistalPhalanxOutlineAgeGroup extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 155
B.17. Distribución de las clases para la base de datos DistalPhalanxOutlineCorrect extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 155
B.18. Distribución de las clases para la base de datos DistalPhalanxTW extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 156
B.19. Distribución de las clases para la base de datos Earthquakes extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 156
B.20. Distribución de las clases para la base de datos ECG200 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 157
B.21. Distribución de las clases para la base de datos ECG5000 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 157

B.22. Distribución de las clases para la base de datos ECGFiveDays extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 158
B.23. Distribución de las clases para la base de datos ElectricDevices extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 158
B.24. Distribución de las clases para la base de datos FaceAll extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 159
B.25. Distribución de las clases para la base de datos FaceFour extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 159
B.26. Distribución de las clases para la base de datos FacesUCR extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 160
B.27. Distribución de las clases para la base de datos FiftyWords extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 160
B.28. Distribución de las clases para la base de datos Fish extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 161
B.29. Distribución de las clases para la base de datos FordA extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 161
B.30. Distribución de las clases para la base de datos FordB extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 162
B.31. Distribución de las clases para la base de datos GunPoint extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 162

B.32. Distribución de las clases para la base de datos Ham extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 163
B.33. Distribución de las clases para la base de datos HandOutlines extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 163
B.34. Distribución de las clases para la base de datos Haptics extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 164
B.35. Distribución de las clases para la base de datos Herring extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 164
B.36. Distribución de las clases para la base de datos InlineSkate extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 165
B.37. Distribución de las clases para la base de datos InsectWingbeatSound extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 165
B.38. Distribución de las clases para la base de datos ItalyPowerDemand extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 166
B.39. Distribución de las clases para la base de datos LargeKitchenAppliances extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 166
B.40. Distribución de las clases para la base de datos Lighting2 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 167
B.41. Distribución de las clases para la base de datos Lighting7 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 167

B.42. Distribución de las clases para la base de datos Mallat extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 168
B.43. Distribución de las clases para la base de datos Meat extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 168
B.44. Distribución de las clases para la base de datos MedicalImages extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 169
B.45. Distribución de las clases para la base de datos MiddlePhalanxOutlineAgeGroup extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 169
B.46. Distribución de las clases para la base de datos MiddlePhalanxOutlineCorrect extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 170
B.47. Distribución de las clases para la base de datos MiddlePhalanxTW extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 170
B.48. Distribución de las clases para la base de datos MoteStrain extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 171
B.49. Distribución de las clases para la base de datos NonInvasiveFetalECGThorax1 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 171
B.50. Distribución de las clases para la base de datos NonInvasiveFetalECGThorax2 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 172
B.51. Distribución de las clases para la base de datos OliveOil extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 172

B.52. Distribución de las clases para la base de datos OSULeaf extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 173
B.53. Distribución de las clases para la base de datos PhalangesOutlinesCorrect extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 173
B.54. Distribución de las clases para la base de datos Phoneme extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 174
B.55. Distribución de las clases para la base de datos Plane extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 174
B.56. Distribución de las clases para la base de datos ProximalPhalanxOutlineAgeGroup extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 175
B.57. Distribución de las clases para la base de datos ProximalPhalanxOutlineCorrect extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 175
B.58. Distribución de las clases para la base de datos ProximalPhalanxTW extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 176
B.59. Distribución de las clases para la base de datos RefrigerationDevices extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 176
B.60. Distribución de las clases para la base de datos ScreenType extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 177
B.61. Distribución de las clases para la base de datos ShapeletSim extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 177

B.62. Distribución de las clases para la base de datos ShapesAll extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 178
B.63. Distribución de las clases para la base de datos SmallKitchenAppliances extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 178
B.64. Distribución de las clases para la base de datos SonyAIBORobotSurface1 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 179
B.65. Distribución de las clases para la base de datos SonyAIBORobotSurface2 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 179
B.66. Distribución de las clases para la base de datos StarLightCurves extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 180
B.67. Distribución de las clases para la base de datos Strawberry extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 180
B.68. Distribución de las clases para la base de datos SwedishLeaf extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 181
B.69. Distribución de las clases para la base de datos Symbols extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 181
B.70. Distribución de las clases para la base de datos SyntheticControl extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 182
B.71. Distribución de las clases para la base de datos ToeSegmentation1 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 182

B.72. Distribución de las clases para la base de datos ToeSegmentation2 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 183
B.73. Distribución de las clases para la base de datos Trace extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 183
B.74. Distribución de las clases para la base de datos TwoLeadECG extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 184
B.75. Distribución de las clases para la base de datos TwoPatterns extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 184
B.76. Distribución de las clases para la base de datos UWaveGestureLibraryAll extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 185
B.77. Distribución de las clases para la base de datos UWaveGestureLibraryX extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 185
B.78. Distribución de las clases para la base de datos UWaveGestureLibraryY extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 186
B.79. Distribución de las clases para la base de datos UWaveGestureLibraryZ extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 186
B.80. Distribución de las clases para la base de datos Wafer extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 187
B.81. Distribución de las clases para la base de datos Wine extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 187

B.82. Distribución de las clases para la base de datos WordSynonyms extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 188
B.83. Distribución de las clases para la base de datos Worms extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 188
B.84. Distribución de las clases para la base de datos WormsTwoClass extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 189
B.85. Distribución de las clases para la base de datos Yoga extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja. 189

xxx

Índice de tablas

5.1. Entorno de parámetros usado en eMODiTS 74
5.2. Características de las bases de datos temporales usadas en cada experimento realizado en este trabajo. # STE representa el número de series de tiempo en el conjunto de entrenamiento, # STP es el número de series de tiempo en el conjunto de prueba, C es el número de clases, T es el tamaño de cada serie temporal, NI representa si la base fue inicialmente normalizada o no, BC es si la base de datos esta balanceado, PC es la proximidad entre las clases y R es la cantidad de ruido presente en la base de datos temporal. 76
5.3. Error en clasificación obtenido por cada método de selección de preferencias a través del clasificador de Árbol. Los valores posteriores al símbolo ± representan la desviación estándar de los datos, el número entre paréntesis representa el rango calculado por la prueba estadística multi-comparador, y los números en negrita representan los mejores valores obtenidos por cada base de datos temporal. . . . 78
5.4. Porcentajes de error en clasificación obtenidos por cada método basado en SAX usando el clasificador de árboles de decisión. Los valores que aparecen después del símbolo ± representan la desviación estándar de los datos, los números entre paréntesis representan el rango calculado por la prueba estadística usada para comparar múltiples métodos a la vez, y los valores resaltados en negrita representan los mejores valores obtenidos en cada base de datos. . . 83
5.5. Pérdida de la información calculada mediante el método LCSS donde los valores mínimos representan similitud entre la base de datos original y la reconstruida, mientras que valores altos representan que ambos conjuntos son desiguales entre sí. Los números entre paréntesis representan el rango calculado por la prueba estadística multi-comparador y los números en negritas representan los valores mínimos encontrados en cada base de datos. 91

Lista de Símbolos

$\mathbf{I}$	Instancia de una base de datos expresado como $I = \{at_1, at_2, \cdots, at_q\}$.
$\mathbf{q}$	Número de atributos en una base de datos.
$\mathbf{at_i}$	Valor de un atributo de una base de datos, para $1 <= i <= q$
$\mathbf{T_{at}}$	Conjunto de datos para el atributo at, expresado como $T_{at} = \{(v_1, y_1), (v_2, y_2), \cdots, (v_t, y_t)\}$
$\mathbf{v_t}$	Valor de un atributo de la base de datos en la instancia t.
$\mathbf{y_t}$	Valor de la etiqueta de clase de un atributo en la instancia t.
$\mathbf{O}$	Subgrupos en los que se subdivide un conjunto de instancias, expresado como $O = \{o_1, o_2, \cdots, o_k\}$.
$\overline{\mathbf{o_i}}$	Centroide o valor medio del grupo i.
$\mathbf{n}$	Longitud de las series temporales.
$\mathbf{N}$	Número de series temporales encontradas en una base de datos.
$\mathbf{S}$	Esquema de discretización expresado como $S = \{[s_0, s_1], [s_1, s_2], \cdots, [s_{n-1}, s_n]\}$.
$\mathbf{g}$	Nivel de discretización.
$\mathbf{L}$	Número de variables de decisión.
$\mathbf{F}$	Número de funciones objetivo.
$\boldsymbol{\mathcal{F}}$	Región factible.
$\boldsymbol{\mathcal{PF}}$	Frente de Pareto.
$\vec{x}$	Solución potencial al problema conteniendo al conjunto de variables de decisión.
$\vec{f}$	Conjunto de valores obtenidos al evaluar $\vec{x}$ en cada función.
$\mathbf{h}(\vec{x})$	Restricciones de igualdad.
$\mathbf{g}(\vec{x})$	Restricciones de desigualdad.
$\vec{u}$	Conjunto de pesos usados para dar prioridad a las funciones objetivos en los métodos de optimización multi-objetivo clásicos.

$\tau_{\mathbf{i}}$ . . . . . Meta definida para optimización multi-objetivo mediante el método programación por metas.

$\delta_{\mathbf{i}}$ . . . . . Desviación entre la meta y la función objetivo usada en el método programación por metas.

$\mathbf{m}$ . . . . . Tamaño del conjunto de segmentos de palabra.

$\mathbf{w_i}$ . . . . . Valor de corte en la posición i de un segmento de palabra del esquema de discretización.

$\mathbf{W}$ . . . . . Conjunto de segmentos de palabra representados como $W = \{w_1, w_2, \cdots, w_m\}$.

$\mathbf{a_{w_i}^{(j)}}$ . . . . . Valor del corte de alfabeto en la posición j para el segmento de palabra w_i.

$\vec{a}_{\mathbf{w_i}}$ . . . . . Conjunto de esquemas de alfabeto definido en el segmento de palabra w_i y expresado como $\vec{a}_{w_i} = \{a_{w_i}^{(1)}, a_{w_i}^{(2)}, \cdots, a_{w_i}^{(j)}\}$, donde j es el número de corte del alfabeto.

$\mathbf{TS}$ . . . . . Serie de tiempo expresada como $TS = \{ts_1, ts_2 \ldots, ts_n\}$.

$\mathbf{ts_i}$ . . . . . Valor de la serie temporal en el tiempo j.

$\mathbf{n'}$ . . . . . Longitud de la serie temporal discreta.

$\overline{\mathbf{TS}}$ . . . . . Serie de tiempo reducida usando el algoritmo de PAA expresada como $\overline{TS} = \{\overline{ts_1}, \overline{ts_1}, \ldots, \overline{ts_{n'}}\}$.

$\overline{\mathbf{ts_i}}$ . . . . . Coeficiente obtenido por PAA en la posición i.

$\mathbf{D}$ . . . . . Serie de tiempo obtenida después del proceso de discretización simbólica expresada como $D = \{d_1, d_2, \cdots, d_w\}$.

$\mathbf{D'}$ . . . . . Número de series de tiempo discretas no repetidas

$\mathbf{C}$ . . . . . Número de clases.

$\mathbf{TD}$ . . . . . Base de datos temporal expresada como $TD = \{K_1, K_2, \ldots, K_C\}$.

$\mathbf{K_c}$ . . . . . Subconjunto de series temporales de una misma clase c extraído de la base de datos original expresado como $K_c = \{TS_1, TS_2, \ldots, TS_k\}$.

$\mathbf{k}$ . . . . . Número de series de tiempo temporales pertenecientes a una específica etiqueta de clase c.

$\mathbf{c}$ . . . . . Etiqueta de clase.

$\mathbf{CL}$ . . . . . Conjunto de etiquetas de clase.

$\mathbf{r}$ . . . . . Número de bases de datos temporales de prueba.

$\tilde{\mathbf{D}}$ . . . . . Serie de tiempo simbólica reconstruida.

$\tilde{\mathbf{d}}_{\mathbf{i}}$ . . . . . Valor de la serie de tiempo reconstruida en la posición i.

$\tilde{\mathbf{D}}'$ . . . . . Número de series de tiempo simbólicas reconstruidas.

1
Introducción

Contenido

1.1. Introducción **1**
1.2. Definición del Problema **2**
1.3. Motivación **4**
1.4. Hipótesis **6**
1.5. Objetivos **7**
1.5.1. General 7
1.5.2. Específicos 7
1.6. Justificación **8**
1.7. Trabajo Relacionado **9**
1.7.1. Calibración del Tamaño del Alfabeto y/o Palabra 10
1.7.2. Pérdida de Información 12
1.8. Contribuciones **14**
1.9. Organización del Documento **15**

1.1. Introducción

En este trabajo se introduce una forma de discretización simbólica de bases de datos temporales, llamada **Discretización Simbólica Multi-objetivo Mejorada de Series Temporales** (***e****nhanced* ***M****ulti-objective symb****O****lic* ***D****iscretization for* ***T****ime* ***S****eries, eMODiTS* en inglés), la cual toma como base el algoritmo llamado *Symbolic Aggregate aproXimation (SAX)*[64, 65], por ser uno de los más eficientes

y, por consiguiente, más empleados en la literatura especializada. Sin embargo, diversos autores han evidenciado algunas limitaciones de SAX, proponiendo mejoras tanto en el proceso de discretización como en la forma de obtenerlo. Nuestra propuesta impacta en mejorar la discretización de series temporales proponiendo una estructura flexible, donde cada corte en el tiempo contiene su propio esquema de cortes definido en el espacio de valores de la serie temporal. Para encontrar dicho esquema flexible, se incluye un algoritmo evolutivo multi-objetivo basado en tres funciones propias para determinar la eficiencia del esquema encontrado: en términos de clasificación, complejidad del modelo y pérdida de información.

En esta sección se presentan los fundamentos de la propuesta, resaltando el impacto y la contribución en las Secciones 1.2, 1.3, y 1.8. Posteriormente, se presentan la hipótesis a comprobar (Sección 1.4), los objetivos perseguidos (Sección 1.5), la justificación de la propuesta (Sección 1.6), así como un detalle de los trabajos relacionados encontrados en la literatura con respecto a la discretización de series temporales (Sección 1.7).

1.2. Definición del Problema

La minería de datos en series temporales (TSDM, por sus siglas en inglés) ha ganado importancia en diversas áreas como: economía, climatología, medicina, entre otras, debido a la necesidad humana de entender el comportamiento de los datos y tomar decisiones acertadas en el momento oportuno [38].

Para comprender el reto que la TSDM tiene en dichas áreas, es necesario definir los conceptos de serie de tiempo y base de datos temporal. Una *Serie de Tiempo o Serie Temporal*, es una colección de datos, típicamente de números reales $\mathbb{R}$, obtenidos en ciertos periodos de tiempo [38, 26]; mientras que, una *base de datos temporal* es un conjunto de series temporales. Una serie temporal puede ser tan extensa o corta como el mismo problema lo demande; por ello, el manejo de grandes cantidades de este tipo de datos conlleva un costo de cómputo elevado para las organizaciones, haciendo compleja la tarea de entender la variabilidad de la información. Este problema puede definirse como *el problema de dimensionalidad* [99].

Una forma de reducción de la dimensionalidad en series de tiempo es la *Discretización de series temporales*, la cual consiste en convertir valores continuos en valores discretos [88]. Por lo general, este proceso necesita definir intervalos o cortes de valores en el espacio continuo de cada variable.

Dichos intervalos son secuencias o segmentos de la serie de tiempo. Al conjunto de éstos se le conoce como *esquema de discretización* y se expresa como $D = \{[d_0, d_1]_1, [d_1, d_2]_2, \ldots, [d_{n-2}, d_{n-1}]_{g-1}, [d_{n-1}, d_n]_g\}$, donde d_0 y d_n son los valores máximos y mínimos del componente temporal de la serie. Cada intervalo en D representa un segmento donde el conjunto de valores de la serie contenidos dentro de éste son transformados en valores discretos. Al número total de segmentos g se le conoce como *grado de discretización* [88]. La Figura 1.1a muestra gráficamente un esquema de discretización donde cada línea vertical representa un corte en el espacio temporal.

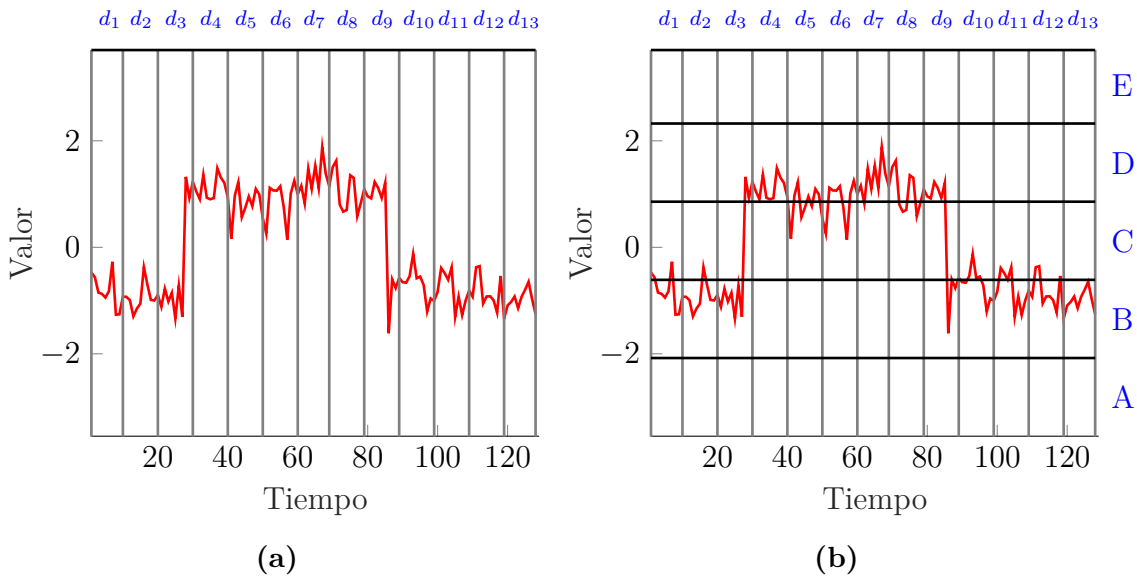


Figura 1.1: (a) Esquema de discretización del componente temporal. (b) Discretización simbólica de series de tiempo.

Los valores que puede tomar la serie discreta son obtenidos mediante diversos métodos propuestos en la literatura especializada [13, 11, 67]. Una de las formas más utilizadas por los científicos es, calcular el promedio de los valores encontrados en cada intervalo de tiempo $\bar{x}^i_{[d_i, d_{i+1}]}$, obteniendo una serie temporal reducida

con valores continuos $\overline{ST} = \{\bar{x}^1, \bar{x}^2, ..., \bar{x}^s\}$; a este método se le conoce como *Piecewise Aggregate Approximation (PAA)*. Para convertir estos promedios en valores discretos, se utilizan representaciones simbólicas donde la serie de tiempo discreta es conocida como *strings o cadenas*.

El enfoque llamado *Symbolic Aggregate Approximation (SAX)* es uno de los métodos simbólicos más populares en la literatura para discretizar series de tiempo, debido a su fácil comprensión y a su rápida ejecución. El método SAX consiste en relacionar cada promedio obtenido por el algoritmo PAA con un símbolo mediante cortes en el espacio de valores (Ecuación 1.1). Las líneas horizontales de la Figura 1.1b representan gráficamente estos cortes. Para definir esta relación, simplemente se observa el intervalo de valores donde el promedio obtenido en cada segmento de tiempo incide, y dependiendo de que intervalo le correspondió es el símbolo asignado. Por ejemplo, en la Figura 1.1b las líneas horizontales pequeñas y de un grosor mayor representan el promedio de los valores en cada segmento $[d_i, d_{i+1}]$. Al observar los intervalos donde están incluidos cada promedio, obtenemos la cadena $CBCDDDDDCBBBB$. A los cortes realizados en el componente temporal, SAX los identifica como *segmentos de palabra* y a los cortes realizados al componente de valores los identifica como *alfabetos*. Un caracter l_i de la cadena discreta *string* puede ser obtenido mediante la Ecuación 1.1, donde $\bar{x}^i$ representa el valor promedio de la serie de tiempo en el intervalo de tiempo $[d_i, d_{i+1}]$, $[a_j, a_{j+1}]$ representa un intervalo de cortes de alfabeto, y τ_j es el caracter o símbolo correspondiente en el corte de alfabeto j.

$$\begin{aligned} string &= \{l_1, l_2, \cdots, l_s\} \\ l_j &= \bar{x}^i \mapsto \tau_j \\ \tau_j &= \{es\ un\ símbolo\ o\ letra\} \end{aligned} \tag{1.1}$$

1.3. Motivación

Como se explicó anteriormente, la discretización simbólica de series de tiempo consiste en definir cortes, tanto en el componente temporal como en el espacio de

valores, llamados segmentos de palabra y alfabetos, respectivamente.

El método SAX es el algoritmo por excelencia para la discretización simbólica temporal; sin embargo, requiere como parámetros iniciales el número de cortes en el tiempo y en el espacio de valores para generar los cortes en ambos componentes. Dichos cortes se crean de manera automática de la siguiente forma: (a) los segmentos de palabra son intervalos del mismo tamaño calculados como $I = \frac{n}{w}, w \neq 0$, donde w es el número de cortes en el tiempo, y n es el tamaño de la serie temporal, y (b) los cortes del alfabeto son segmentos del mismo tamaño definidos siguiendo una distribución normal.

Estos parámetros iniciales afectan el desempeño del algoritmo, dependiendo de la tarea a realizar; por ejemplo, valores bajos conllevan una pérdida de la forma original de la serie temporal, en otras palabras, pérdida de información. Por otro lado, valores altos favorecen la conservación de la información más importantes de la serie, pero dificulta encontrar patrones o modelos que permitan discriminar unas series de tiempo de otras.

El algoritmo SAX, al subdividir cada componente de las series de tiempo (valores y tiempo), forma una malla bidimensional sobre la serie temporal y, mediante los promedios obtenidos por PAA, la transforma en una serie de tiempo simbólica o *strings*. Esta malla puede no ser idónea en conjuntos de series de tiempo complejos, es decir, series de tiempo con formas caprichosas o etiquetadas de forma similar a otras que no guardan ningún parecido entre sí, generando series discretas con baja compresión donde un algoritmo de clasificación no logre encontrar un modelo para predecir el grupo o clase a la que pertenece, o incluso, haciendo una transformación donde datos esenciales de la serie se pierdan.

Adicionalmente, dado que SAX asume una distribución normal de los datos para generar los cortes para la asignación de símbolos o caracteres, necesita un proceso previo de normalización de la base temporal para obtener una media igual a cero y desviación estándar igual a uno; sin embargo, en bases temporales con ruido alto, este proceso puede amplificar significativamente dicho ruido dificultando el análisis de los datos temporales [64].

Nuestra motivación radica en mejorar las limitaciones de SAX mencionadas en párrafos anteriores (calibración de los valores tanto para el tamaño del alfabeto como el número de segmentos de palabra y la pérdida de información). En este documento se presenta un algoritmo que permite definir los cortes en cada componente de la serie de tiempo (el tiempo y los valores) sin asumir ninguna distribución de probabilidad y tratando de ajustarse, de forma más adecuada, a la serie temporal. Es por ello, que se resaltan tres principales contribuciones: 1) un esquema de discretización donde cada segmento en el tiempo contenga su propio conjunto de cortes en el espacio de valores, permitiendo un incremento en el grado de libertad del algoritmo para obtener series simbólicas más parecidas a la original; 2) utilizar la métrica para estimar la pérdida de información propuesta en [93] como guía en el proceso de búsqueda de esquemas de discretización con el menor índice de pérdida de información, y 3) la inclusión de una forma gráfica para obtener los patrones o relaciones entre clases, permitiendo un mejor entendimiento de los datos y por consecuencia una toma de decisiones apropiada por parte del responsable de la información. Esta última, no representa una limitación de SAX, sino que, es una aportación adicional proporcionada por nuestra propuesta que, en la revisión de la literatura especializada realizada, no se ha abordado bajo ningún otro enfoque de discretización simbólica.

1.4. Hipótesis

La discretización de bases de datos temporales usando cortes de alfabeto únicos y diferentes por cada segmento de palabra permitirá encontrar representaciones discretas eficientes en términos de compresión, clasificación y pérdida de información en bases temporales. Adicionalmente, dado que se usarán árboles de decisión para evaluar su desempeño, se podrá evidenciar gráficamente los patrones intrínsecos y/o las relaciones existentes dentro de cada base temporal.

1.5. Objetivos

1.5.1. General

Discretizar bases de datos temporales utilizando representaciones flexibles (alfabetos específicos para cada segmento de palabra en un esquema de discretización), árboles de decisión y algoritmos evolutivos multi-objetivo, con la finalidad de encontrar aquellas representaciones o esquemas que logren reducir la dimensionalidad de la serie de tiempo sin perder información importante y permitiendo la correcta clasificación de las mismas.

1.5.2. Específicos

1. Definir esquemas de discretización donde cada segmento de palabra contenga su propio alfabeto.

2. Estimar la viabilidad del esquema en términos de clasificación, complejidad del modelo y pérdida de la información.

3. Implementar un algoritmo multi-objetivo para encontrar el compromiso entre cada una de las funciones mencionadas en el objetivo anterior.

4. Usar el algoritmo de clasificación mediante árboles de decisión para evaluar la solución encontrada por la propuesta.

5. Probar la propuesta en las 85 bases de datos del repositorio de UCR [27].

6. Comparar la eficiencia de nuestro trabajo midiendo la tasa de clasificación errónea contra nueve algoritmos de discretización simbólicas (SAX, EP, αSAX, ESAX, ESAXKMeans, 1D-SAX, RKMeans, SAXKMeans y rSAX).

7. Analizar los árboles obtenidos por nuestra propuesta para extraer los patrones o relaciones implícitas en las bases temporales, y así, entender el comportamiento de los datos.

Cada uno de los objetivos planteados es definido con la finalidad de comprobar la hipótesis planteada en la Sección 1.4.

1.6. Justificación

En los últimos años, el crecimiento de la información se ha dado de manera exponencial. Por ejemplo, en el sector económico, diariamente se almacenan datos como tasas de interés, precios de cierre, cifras de ventas, índice de precios, entre otros; en climatología, diariamente se almacenan las mediciones de variable climáticas como temperatura, precipitación, velocidad del viento, etc. [29]. En cada una de estas áreas, es de vital importancia mantener el registro de los datos históricos y actuales para entender y generar modelos para la predicción de eventos o fenómenos.

La mayoría de esta información es almacenada como series de tiempo, lo que conlleva la necesidad de analizar al conjunto de datos completo y no por partes, ya que no se puede explicar un dato sin tomar en cuenta el anterior, dado el componente temporal implícito en los datos.

Sin embargo, dentro de los retos actuales generados por la alta dimensionalidad de los datos se encuentran:

1. *La infraestructura de almacenamiento con la que actualmente cuenta la industria.* El incontrolado crecimiento de la información obliga a empresas tecnológicas a desarrollar dispositivos con mayor capacidad de almacenamiento; sin embargo, este crecimiento puede darse a mayor velocidad que el desarrollo de nuevos dispositivos.

2. *El manejo de los datos.* El análisis de la información de alta dimensionalidad puede resultar complejo y lento con la tecnología computacional existente, conllevando al desaprovechamiento de la riqueza de los datos útil para la toma de decisiones, ya sea empresarial como de cualquier otra rama.

Por ello, es necesario el desarrollo de herramientas que permitan resolver cada uno de estos retos para lidiar con datos de alta dimensionalidad. En nuestro caso, la discretización de datos temporales, permite reducir el tamaño del conjunto de datos para que pueda ser almacenado en dispositivos de baja capacidad. Por otra parte, permite realizar un análisis de los datos más eficiente al no tener que contemplar todo el conjunto de datos para la descripción y explicación de la información.

1.7. Trabajo Relacionado

En la literatura especializada se han reportado varios métodos encaminados a mejorar las desventajas de SAX. La mayoría están basados en el mismo procedimiento de SAX, pero modificando ciertas partes esenciales que permiten mejorar su rendimiento, o utilizando heurísticas para encontrar valores adecuados para los parámetros iniciales del algoritmo. En esta sección se presenta una revisión de estos trabajos, basada en una clasificación general elaborada con respecto de las limitaciones de SAX presentadas en apartados anteriores. Ello debido a que nuestra propuesta está basada en dicho algoritmo mejorando sus limitaciones. La Figura 1.2 muestra la clasificación realizada.

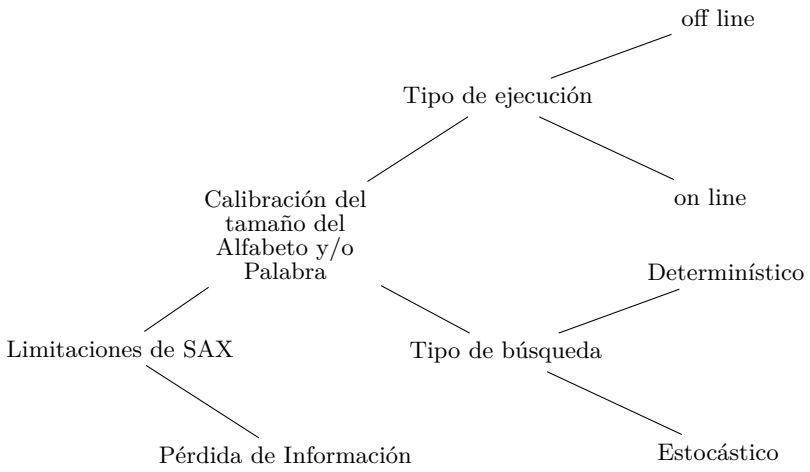


Figura 1.2: Clasificación general de los métodos de representación simbólica basada en las limitaciones del algoritmo SAX

Cada método es agrupado de acuerdo a dos desventajas principalmente: (a) la optimización de los parámetros iniciales de SAX (número de segmentos de palabra y alfabeto), y (b) pérdida de información, la cuál se refiere a la forma en que se obtiene cada valor discreto para cada segmento de palabra; por ejemplo, PAA usa la media aritmética de los valores de la series de tiempo que se encuentran dentro del intervalo de cada segmento de palabra. Sin embargo, este proceso puede ignorar algunos puntos o regiones esenciales de la serie temporal. Con respecto a la primera desventaja mencionada, lo métodos presentados han utilizado dos características principales para abordarla:

1. *Tipo de Procesamiento.* Indica en qué momento del proceso de discretización se optimizan los parámetros del algoritmo, es decir, si los valores para el tamaño de alfabeto y palabra son optimizados antes (Pre-procesamiento) o durante el proceso de discretización.

2. *Tipo de Proceso.* En este rubro se encuentran los enfoques dependiendo del tipo de proceso usado (estocástico o determinístico) en la búsqueda de los valores óptimos para los parámetros iniciales.

Esta clasificación permite establecer las fortalezas y debilidades de cada método simbólico de esta revisión, las cuales son descritas en los siguientes apartados de acuerdo a la Figura 1.2.

1.7.1. Calibración del Tamaño del Alfabeto y/o Palabra

Uno de los primeros enfoques que busca la calibración idónea del alfabeto y la palabra es *Adaptive Symbolic Aggregate approXimation (αSAX).* El algoritmo αSAX es un método determinístico que adapta el vector de alfabetos antes del proceso de discretización mediante el algoritmo de agrupamiento *K-Means* en una dimensión (Algoritmo de Lloyd) [82]. Este algoritmo es evaluado respecto a la característica del límite inferior (*lower-bounding feature*), reducción de dimensionalidad y el número de accesos aleatorios al dispositivo de almacenamiento; es decir, su evaluación no está dada en términos de clasificación. La principal diferencia entre αSAX y SAX es que αSAX usa el tamaño del alfabeto para definir el número de grupos necesarios para el algoritmo de agrupamiento anteriormente mencionado, y los cortes del alfabeto encontrados mediante una distribución normal son usados para inicializar los intervalos de cada grupo.

Otro método determinístico que trata de mejorar la calibración inicial del tamaño de alfabeto y palabra es el enfoque llamado *Symbolic Aggregate approXimation by data-driven Optimization (SAXO)* [13]. El enfoque SAXO optimiza dichos valores y los intervalos de cada uno durante su procesamiento usando un algoritmo de agrupamiento llamado *co-clustering algorithm (MODL)* [14] que utiliza la

distribución típica de los datos para crear cada grupo. Posteriormente, a cada grupo se le asigna un símbolo para formar la cadena discreta final. Sin embargo, el método SAXO esta limitado por la dimensionalidad de la base temporal debido al costo computacional requerido para su ejecución. SAXO fue evaluado mediante la estimación de la ganancia de la información obtenida contra la ganancia de información obtenida por MODL.

Por otra parte, *HSAX* es un método estocástico basado en el algoritmo de búsqueda armónica (*Harmony Search Algorithm*) [6], el cual es implementado para encontrar los valores óptimos tanto para el tamaño del alfabeto como de la palabra. Esta búsqueda es realizada antes del proceso de discretización realizado mediante el algoritmo SAX. El método HSAX fue evaluado en términos de clasificación mediante el algoritmo *1-NN*. En [5] se presenta una versión mejorada de HSAX llamada SAX^{++}. La modificación radicó en mejorar el tamaño del alfabeto mediante la adaptación de sus valores usando el método RF (*Relative Frequency*). Aunque los resultados encontrados por este método fueron competitivos en términos de clasificación, fueron agregados dos parámetros más a la calibración inicial del algoritmo.

Al igual que HSAX y SAX^{++}, otros métodos estocásticos surgidos para la búsqueda de la configuración óptima del tamaño de alfabeto y palabra son *GASAX* (Genetic Algorithms-Based SAX) [77] y *DESAX* (Differential Evolution-Based SAX) [45]. Estos enfoques utilizan dos algoritmos de optimización bien conocidos en la literatura: el algoritmo genético y evolución diferencial, respectivamente. Dicha optimización se realiza antes de ejecutar el proceso de discretización mediante SAX y son evaluados mediante la tasa de error en clasificación usando el algoritmo *1-NN*. Incluso cuando los resultados obtenidos fueron prometedores en términos de reducción de dimensionalidad y clasificación, ambos algoritmos necesitan definir valores iniciales para el número de segmentos del alfabeto y de la palabra.

Acosta *et al.*, en [3], propusieron un método estocástico llamado *EBLA2 (Entropy Based Linear Approximation 2)*, el cual usa un algoritmo de búsqueda local para buscar automáticamente los valores para el tamaño de palabra y alfabeto, así como los cortes de cada uno. La principal desventaja de este método es la susceptibilidad de

quedar atrapados en óptimos locales. *Genetic Entropy Based Linear Approximation (GENEBLA)* [49] fue propuesto para mejorar las deficiencias de EBLA2. El método GENEBLA implementó un Algoritmo Genético basado en la medida de ganancia de la información para encontrar esquemas de discretización competitivos en términos de clasificación. Sin embargo, GENEBLA busca el número de cortes del alfabeto y palabra mediante dos procesos independientes produciendo soluciones incompletas.

Otro enfoque que optimiza automáticamente los valores del tamaño de palabra y alfabeto con sus respectivos cortes es presentado en [88, 2]. Dicho enfoque es llamado *Evolutionary Programming (EP)*. EP usa el algoritmo evolutivo llamado Programación Evolutiva para la búsqueda del esquema de discretización adecuado basado en tres funciones objetivo: Entropía, Complejidad y Compresión. Para evaluar cada esquema se usa el modelo de toma de decisiones multi-objetivo llamado *Suma de Funciones Ponderadas (SPF)*, la cual convierte un problema de múltiples funciones es una sola sumando cada función ponderada mediante pesos para determinar su prioridad. Este método presenta dos principales desventajas: (1) la calibración idónea de los pesos de la función, la cual puede llevar a la búsqueda hacia óptimos locales; y (2) el rendimiento ineficiente en problemas con frentes de Pareto no convexos [70].

1.7.2. Pérdida de Información

Muchos autores [94, 93, 92, 99, 20] consideran que los coeficientes obtenidos por los métodos de reducción de dimensionalidad no garantizan una pérdida de información mínima cuando la serie temporal es transformada en una discreta. Uno de los métodos que trata de minimizar dicha pérdida es *1D_SAX* [69]. 1D_SAX considera las tendencias o pendientes de la serie temporal junto con el valor medio de cada segmento de palabra para representar cada serie de tiempo; es decir, la representación simbólica encontrada por 1D_SAX está dada por dos valores, la pendiente y el valor medio (s, a). La asignación de cada símbolo o letra se realiza intercalando la representación binara de s y a en cada conjunto de segmentos de palabra. Sin embargo, 1D_SAX necesita la calibración inicial de los tamaños de palabra y alfabeto, así como, el número de pendientes necesarias.

ESAX (Extended SAX) [67, 68] surgió para minimizar la pérdida de la información mediante la inclusión de los valores máximo, mínimo y el promedio de cada segmento de palabra como parte de la representación simbólica. Cada valor es relacionado con un símbolo de acuerdo a reglas definidas para determinar el orden de cada uno. Es importante destacar que, la longitud de la serie de tiempo discreta final obtenida por este método, es tres veces mayor al número de segmentos de palabra definido al inicio del proceso; es decir, por cada segmento de palabra se tendrán 3 valores discretos en lugar de uno, lo que incrementa la longitud de la serie discretas $3m$, donde m es el número de segmentos de palabra. Sin embargo, este método también requiere la calibración inicial del número de cortes de palabra y alfabeto.

Otro método propuesto para minimizar la pérdida de la información es *rSAX (Random Shifting based SAX)* [11]. rSAX aplica una perturbación a los valores de corte del alfabeto τ veces mediante una distribución normal; es decir, rSAX transforma la serie temporal en τ representaciones simbólicas diferentes. Este método no fue comparado en términos de clasificación, sino que, su objetivo es mejorar la propiedad de límite inferior. Sin embargo, sigue siendo necesario la calibración inicial del tamaño de alfabeto y palabra.

Dos Santos Passos *et al.*, en [36], presentaron tres métodos estocásticos basados en algoritmos de agrupamiento. El primero de ellos consiste en obtener los cortes de alfabeto mediante el algoritmo KMeans (*RKMeans*), mientras que los otros, son híbridos entre el algoritmo KMeans con SAX (*SAX-KMeans*) y ESAX (*ESAX-KMeans*). Cada algoritmo fue evaluado en términos de clasificación usando bases de datos de electrocardiogramas. Sin embargo, al igual que los métodos mencionados anteriormente, se sigue necesitando la calibración inicial de los parámetros iniciales.

Recientemente, en [71] se propone un algoritmo de discretización multi-objetivo llamado *MODiTS*, el cual surge para mejorar las desventajas de EP y SAX. MODiTS usa el algoritmo evolutivo multi-objetivo *NSGA-II* para manejar el compromiso entre las tres funciones propuestas por EP. La principal contribución de este método es un esquema de discretización donde cada segmento de palabra tiene un conjunto de cortes de alfabetos diferentes, lo que incrementa el grado de libertad en la búsqueda

de esquemas de discretización adecuados. Sin embargo, la solución seleccionada del frente de Pareto fue aquella más cercana al punto llamado rodilla, la cual es ineficiente en frentes no convexos.

Los algoritmos mencionados en esta sección fueron propuestos para mejorar el rendimiento de SAX. Cada uno se centran en encontrar los valores óptimos para el tamaño de alfabeto y palabra, y algunos de ellos, en sus respectivos cortes, dejando de lado el grado de libertad del proceso de búsqueda y la pérdida de información causada por la reducción de la dimensionalidad de las series temporales.

1.8. Contribuciones

En esta sección se presentan las principales contribuciones del presente trabajo de investigación, las cuáles están enfocadas en tres aspectos: a) el esquema de discretización, b) el manejo de la pérdida de información, y c) la interpretación de los datos. A continuación se listan a detalle cada una.

1. *Discretización.* Se utiliza un esquema de discretización flexible y adaptable a la información de cada segmento de palabra, permitiendo un aumento en el grado de libertad de la búsqueda de esquemas de discretización capaces de mejorar los índices de clasificación.

2. *Pérdida de Información.* Dado que se pretende reducir el tamaño de la serie temporal, se incluye una estimación de la pérdida de información propia de dicha reducción, para encontrar esquemas que no sólo realicen una clasificación competitiva de los datos, si no que también, minimicen la cantidad de información importante perdida en el proceso de transformación.

3. *Interpretación de Datos.* Al final del proceso de discretización, es necesario evaluar la propuesta en términos de clasificación; para ello, se utiliza un árbol de decisión que no sólo obtiene los índices de errores en dicha tarea, sino que también permite, de forma gráfica, explicar y entender el comportamiento de los datos con la finalidad de apoyar al usuario en la toma de decisiones importantes.

1.9. Organización del Documento

El presente trabajo de investigación está organizado de la siguiente forma: en el Capítulo 2 se presenta el marco teórico con respecto a la minería de datos en series temporales. En el Capítulo 3 se describen los conceptos necesarios sobre la optimización multi-objetivo, desde los métodos tradicionales hasta las meta-heurísticas más empleadas en la literatura especializada. El Capítulo 4 describe a detalle los elementos propios de la propuesta presentada en este documento, así como su implementación. El Capítulo 5 presenta el ambiente experimental, los resultados obtenidos, así como la discusión derivada de cada análisis realizado. Por último, en el Capítulo 6 se detallan las conclusiones a las que conllevó la investigación presentada en este documento.

2

Minería de Datos en Series Temporales

Contenido

2.1. Introducción 17
2.1.1. Tipos de Datos 19
2.1.2. Procesamiento de la Información 21
2.1.3. Tareas de Minería de Datos 23
2.2. Discretización de Series de Tiempo 25
2.2.1. Reducción de Dimensionalidad 25
2.2.2. Discretización Simbólica 26
2.3. Medidas de Similitud o Distancia 30
2.3.1. Distancia Euclidiana 31
2.3.2. Alineamiento Temporal Dinámico (*Dynamic Time Warping, DTW*) 31
2.3.3. Subsecuencia Común más Larga (*The Longest Common Subsequence*) 33
2.4. Clasificación en Series de Tiempo 34
2.4.1. k-NN: Vecinos más Cercanos 35
2.4.2. Clasificador Bayesiano Ingenuo 36
2.4.3. Árboles de Decisión 37
2.5. Agrupamiento (*Clustering*) en Series de Tiempo . . . 41
2.5.1. K-Medias 41

2.1. Introducción

En la actualidad, la cantidad de información generada por el sector empresarial va en constante aumento, y con ella, la necesidad de entender el conocimiento

oculto dentro de los datos.

Sin embargo, la extracción de conocimiento a partir de datos puede verse afectada por ciertos problemas que limitan el correcto análisis de la información. Algunos de dichos problemas son:

- *La naturaleza de los datos.* A medida que la tecnología avanza, la forma en que los datos son generados y almacenados está en un cambio constante, llevando a representaciones caprichosas que son difíciles de analizar de forma tradicional.

- *Inconsistencia en el almacenamiento de los datos.* Conforme los datos son colectados y almacenados en dispositivos de almacenamiento, pueden ocurrir problemas externos que provoquen un almacenamiento incorrecto de los mismos, resultando en análisis erróneos de la información.

Para solventar estas situaciones, han surgido técnicas o métodos capaces de encontrar patrones dentro de grandes volúmenes de información lidiando con inconsistencias de datos. Al conjunto de estas técnicas se le conoce como *minería de datos (MD).*

La **minería de datos** consiste en coleccionar, limpiar, procesar, analizar y encontrar información útil en grandes cantidades de información [56]. Forma parte del proceso de *descubrimiento de la información* descrito en la Figura 2.1, el cual consiste de una serie de etapas para convertir los datos originales en datos útiles para las organizaciones [97]. Cada etapa es descrita a continuación.

1. *Preprocesamiento de la información.* Esta etapa se refiere al tratamiento de los datos (eliminación de ruido, datos duplicados, datos vacíos, entre otros) antes de ser analizados por los métodos de extracción de conocimiento, con la finalidad de que sean manipulables (con el formato correcto) por éstos y los resultados sean consistentes con lo requerido por el usuario final.

2. *Minería de Datos.* Como ya se mencionó, ésta corresponde al conjunto de técnicas empleadas para la extracción del conocimiento en grandes volúmenes de información.

3. *Post-procesamiento de resultados.* Esta última se refiere a la visualización de los resultados, para darle la interpretación necesaria para las organizaciones y por consiguiente usar el conocimiento en la tarea requerida.

Figura 2.1: Proceso de descubrimiento de conocimiento

2.1.1. Tipos de Datos

Como todo proceso informático, el descubrimiento de la información requiere de datos de entrada para la ejecución de los procesos posteriores. A cada grupo de datos que representa una característica en especial, se le conoce como *Atributo*; mientras que a la colección de atributos de un caso particular se le conoce como *Instancias* y al conjunto de instancias se le llama *Base de Datos*.

Existen varios tipos de datos en los que se puede representar la información de las organizaciones. Estos pueden ser obtenidos de diversas formas y almacenados en diferentes formatos dependiendo del problema que se esté abordando. La Figura 2.2 muestra una categorización de los tipos de datos realizada en [4], donde son divididos en 2 grupos: los datos sin dependencia y datos con dependencia.

Datos independientes. Éstos se refieren a aquellos datos simples recolectados de forma independiente, es decir, sin que se necesite conocer otro valor para ser interpretados.

Datos dependientes. Son aquellos datos que dependen de algún otro factor como el tiempo o el espacio para ser utilizados, es decir, contienen relaciones implícitas con otros datos.

A continuación se detallan los diferentes tipos datos dependientes e independientes mostrados en la Figura 2.2.

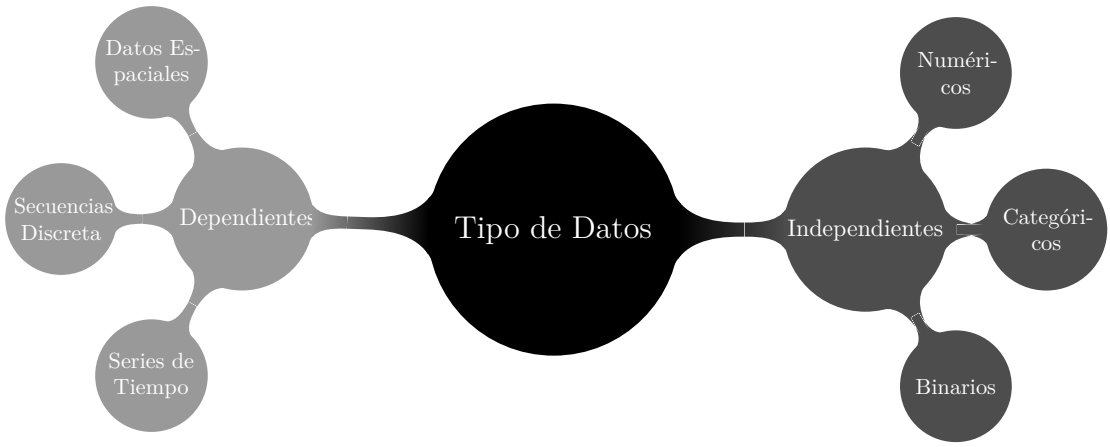


Figura 2.2: Clasificación de los diferentes tipos de datos.

Datos Numéricos. Este tipo de datos es uno de los más comunes en el mundo empresarial. Expresan una medida *cuantitativa* de alguna característica mediante el conjunto de números reales. También son conocidos como *datos continuos*. Ejemplo de datos numéricos: precio de un producto, edad, estatura, etc.

Datos Categóricos. Expresan una categoría en forma desordenada; es decir, son datos cualitativos discretos sin un orden natural entre ellos. Ejemplos de este tipo son: género, raza y código postal.

Datos Binarios. Un dato binario es un dato cuantitativo que solo puede tomar uno o dos valores discretos. Por ejemplo, si una persona sabe o no manejar un vehículo, si presentó una patología o no, etc.

Series de Tiempo. Una serie de tiempo es una colección de valores, típicamente pertenecientes al conjunto de números reales $\mathbb{R}$, recolectados durante un periodo de tiempo específico [38, 26]. Formalmente, una serie de tiempo se puede expresar como $S = \{s_1, s_2, \cdots, s_T\}$ donde s_t es el valor obtenido en el instante t. Es importante mencionar que en este tipo de datos ya no encontramos un único valor para representar una característica, sino que se

tienen 2 valores que viven en dos espacios diferentes:*espacio de valores*, que es el conjunto de valores resultante de las mediciones; y *espacio temporal*, que son los valores de los momentos en que se tomó cada medición. Un ejemplo de serie temporal sería el resultado de medir la actividad cardíaca de un paciente o la medición de los niveles de precipitación para una zona geográfica específica.

Secuencias Discretas o Cadenas de Caracteres. Este es un tipo especial de series de tiempo; es decir, es una serie de tiempo pero en lugar de valores continuos o reales, son almacenados valores discretos o categóricos.

Datos Espaciales. Los datos espaciales son mediciones de las características de objetos obtenidos en ubicaciones espaciales, es decir, características geográficas usualmente almacenadas como coordenadas. Por ejemplo, la medición de la temperatura y presión de la superficie del mar para prevenir la creación de huracanes, donde se puede conocer dicho valor mediante las coordenadas de longitud y latitud.

2.1.2. Procesamiento de la Información

Dentro del proceso de descubrimiento de la información es usual encontrarnos con problemas de almacenamiento de los datos, dificultando el análisis de la información. Este fenómeno se presenta de forma recurrente en la mayoría de bases de datos a nivel mundial, donde los datos se generan a ritmos incontrolables y no existe un filtro eficiente en el almacenamiento de la información. Dentro de los problemas que podemos enfrentarnos al momento de aplicar técnicas de descubrimiento de la información se tienen [97]:

1. *Datos atípicos.* Son valores que sobresalen del modelo general de los datos. Generalmente, tienen un comportamiento aleatorio por lo que son difíciles de detectar y por consiguiente eliminar o evitar en los procesos de extracción de conocimiento.

2. *Datos incompletos.* Este problema se presenta cuando existen valores faltantes en las bases de datos debido a fallas en el almacenamiento o simplemente a la falta de medición para alguna característica en particular.

3. *Datos ruidosos.* Son datos que no pertenecen al modelo generador de la información del problema, sino que se ajustan a otro modelo diferente al buscado, generalmente aleatorio. Conlleva una distorsión de los valores de los atributos. Algunos autores consideran los datos atípicos también como datos ruidosos.

4. *Datos no estandarizados.* Son datos que se presentan en diferentes escalas de referencia de valores, es decir, se pueden presentar características medidas en intervalos pequeños ($(0, 1)$ por ejemplo) y otros medidos con magnitudes muy altas. Esto puede conllevar a que los resultados estarán dominados por los atributos de mayor magnitud [101].

5. *Datos sin formato.* Son datos que no fueron almacenados en formatos idóneos para su manejo, por lo que no son utilizables por la mayoría de las técnicas de descubrimiento de información.

6. *Datos inconsistentes.* Son datos que no pertenecen al dominio específico de un atributo en particular. Por ejemplo, suponga que se tiene una base de datos con un atributo edad, el cual tiene valores numéricos (son los valores esperados) y caracteres (valores que no corresponderían al atributo), lo que ocasiona que los algoritmos de descubrimiento de patrones se confundan o simplemente arrojen resultados erróneos.

7. *Datos extensos.* Este problema se deriva de la necesidad de almacenar y mantener vastas cantidades de información. El análisis de grandes cantidades de datos pueden ocupar mucho tiempo, haciéndolo poco práctico o inviable.

Con la finalidad de solventar cada uno de estos problemas, se han desarrollado técnicas o métodos especiales agrupadas en cada una de las siguientes fases [53]:

- *Limpieza de datos.* En esta fase, se busca eliminar las inconsistencia de los datos, como son: datos faltantes, inconsistentes, con ruido y/o atípicos. Para lidiar con datos faltantes, por ejemplo, se aplican técnicas de estimación o imputación, dependiendo los valores del atributo, para rellenar los huecos existentes. Por su parte, para la detección de datos atípicos o ruido, se utilizan métodos estadísticos para analizar el comportamiento de los datos e identificar aquellos que no se ajusten al modelo de los mismos.

- *Transformación de datos.* Consiste en cambiar el tipo de dato actual por algún otro reconocible por las técnicas tradicionales de minería de datos. Se utiliza para el manejo de datos sin formato reconocible.

- *Normalización.* En esta fase, los datos son escalados a un mismo rango de valores usando las medidas de tendencia central y dispersión de la estadística tradicional.

- *Reducción de la dimensionalidad.* Es el proceso de reducir el número de variables aleatorias o atributos, con la finalidad de obtener representaciones más compactas del conjunto original.

La fase de preprocesamiento de la información, aunque no realiza tareas de descubrimiento de conocimiento, es una de las tareas más importantes, debido a que si los datos de entrada son inconsistentes, con una alta tasa de ruido y con datos faltantes, los resultados obtenidos serán inútiles y fuera de la realidad del problema a resolver. Desafortunadamente, la información obtenida de la mayoría de los problemas actuales siempre presentan una o más de estas inconsistencias, por lo que es indispensable realizar este proceso cada vez que se desee aplicar técnicas de minería de datos.

2.1.3. Tareas de Minería de Datos

Dentro del proceso de minería de datos en general, se han definido diferentes tipos de tareas para descubrir patrones y/o información útil a partir de un conjunto

de datos. De acuerdo a Pang Nin et al., en [97], estas tareas pueden ser agrupadas como se muestra en la Figura 2.3.

Tareas predictivas Las técnicas agrupadas bajo este rubro son diseñadas para predecir valores futuros dependiendo del comportamiento de los datos actuales. El atributo a predecir se le conoce como *variable dependiente* mientras que los otros son conocidos como *variables independientes*.

Tareas descriptivas Estas tareas tratan de descubrir patrones dentro del conjunto de datos, los cuales describen las relaciones importantes de la información. Por lo regular, se necesita un post-procesamiento de los resultados para visualizarlos, validarlos y explicarlos. Utilizan un aprendizaje previo de los datos para su funcionamiento.

Estas tareas utilizan procesos de aprendizaje que le permiten entender la estructura de los datos para la predicción o extracción de conocimiento. A continuación se presentan dos tipos de aprendizajes dependiendo el tipo de tarea de minería de datos a realizar.

- **Aprendizaje Supervisado**. En este rubro se encuentran las técnicas donde el aprendizaje se realiza conociendo la clase de cada instancia de la base de datos. La idea general es aprender de los datos para predecir la categoría de futuros casos desconocidos. Dentro de estas técnicas están: el clasificador *bayesiano*, *Vecinos Más Cercanos*, *Árboles de Decisión*, entre otros. Este tipo de aprendizaje es usado para tareas predictivas.

- **Aprendizaje No Supervisado**. Contrario al aprendizaje supervisado, las técnicas bajo este grupo desconocen la categoría de la información, por lo que su principal función es la separación y asignación de clases dependiendo del modelo de los datos. Una de las técnicas más usadas de aprendizaje no supervisado es el algoritmo *K-Medias*. Este tipo de aprendizaje es usado para tareas descriptivas.

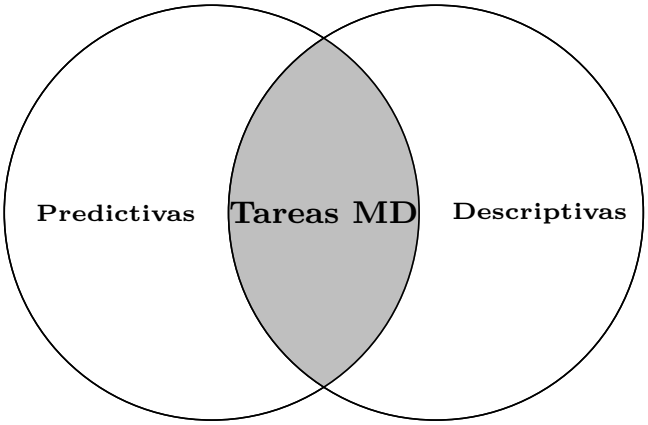


Figura 2.3: Categorización de las tareas de la Minería de Datos (MD).

En este capítulo se abordarán las tareas de clasificación y agrupamiento para un tipo de dato dependiente: las series de tiempo. Dado que este tipo de dato suele ser extenso, es necesario aplicar técnicas de reducción de la dimensionalidad y transformación de la información para realizar un análisis adecuado del conjunto temporal. En el siguiente capítulo se abordarán estas fases de preprocesamiento antes de aplicar alguna de las tareas de minería de datos previamente mencionadas.

2.2. Discretización de Series de Tiempo

El proceso de discretización consiste en transformar un tipo de dato en valores discretos [8]. Particularmente, en series de tiempo, éste consiste en transformar la serie continua en secuencias discretas o cadenas de caracteres.

En esta sección abordaremos una forma de discretización que ha demostrado obtener resultados competitivos en las diversas tareas de minerías de datos en los que se ha probado: *discretización simbólica*. Sin embargo, todo proceso de discretización requiere un proceso previo para reducir la dimensionalidad de la serie antes de la transformación, es por ello que a continuación abordaremos algunos de los métodos más conocidos.

2.2.1. Reducción de Dimensionalidad

La reducción de la dimensionalidad consiste en disminuir el tamaño de la serie temporal. Diversos métodos han sido propuestos para tal tarea, entre los

que encontramos: *Transformada Discreta de Fourier*, *Análisis de Componente Principales*, *Piecewise Aggregate Aproximation (PAA)*, entre otros.

El método PAA es uno de los métodos más recurridos en la literatura, debido a su simplicidad y rápida ejecución. Dicho algoritmo consiste en subdividir la serie temporal $TS = \{ts_1, ts_2, \cdots, ts_n\}$ de tamaño n en m intervalos de igual longitud. Dichos intervalos son expresados como $W = \{w_1, w_2, \ldots, w_m\}$. La longitud de cada intervalo es calculada dividiendo el tamaño de la serie de tiempo n y el número de cortes o intervalos temporales m. Por cada intervalo obtenido, se calcula el valor promedio $\overline{ts}_i$ de los valores de la serie temporal ts_j que se encuentran dentro de los mismos. El conjunto de valores promedios conforman la *serie temporal reducida*. La Ecuación 2.1 expresa matemáticamente dicho procedimiento, mientras que el Algoritmo 1 detalla los pasos a seguir para obtener la serie de tiempo reducida.

$$\overline{ts}_i = \frac{m}{n} \sum_{j=\frac{n((i-1)+1)}{m}}^{\frac{n}{m}(i)} ts_j \tag{2.1}$$

Algoritmo 1 Proceso de reducción de dimensionalidad mediante el proceso PAA

Entrada: $W = \{w_1, w_2, \ldots, w_m\}$: Cortes en el espacio temporal, TS: Serie de tiempo

1: **para** $w_i \in W$, $i = 0, 1, 2, \ldots, m$ **hacer**
2: NoValores = 0
3: SumaValores = 0
4: **para** $j = w_i, j \leq w_{i+1}$ **hacer**
5: $SumaValores = SumaValores + TS(j)$
6: $NoValores = NoValores + 1$
7: $\overline{TS}(i) = SumaValores/NoValores$
devolver $\overline{TS}$

La Figura 2.4 muestra un ejemplo de una serie de tiempo reducida mediante el proceso de PAA.

2.2.2. Discretización Simbólica

Uno de los métodos de discretización simbólica que ha demostrado ser competitivo en la literatura referente a minería de datos en series temporales, es el bien conocido método *Symbolic Aggregate Approximation (SAX)*.

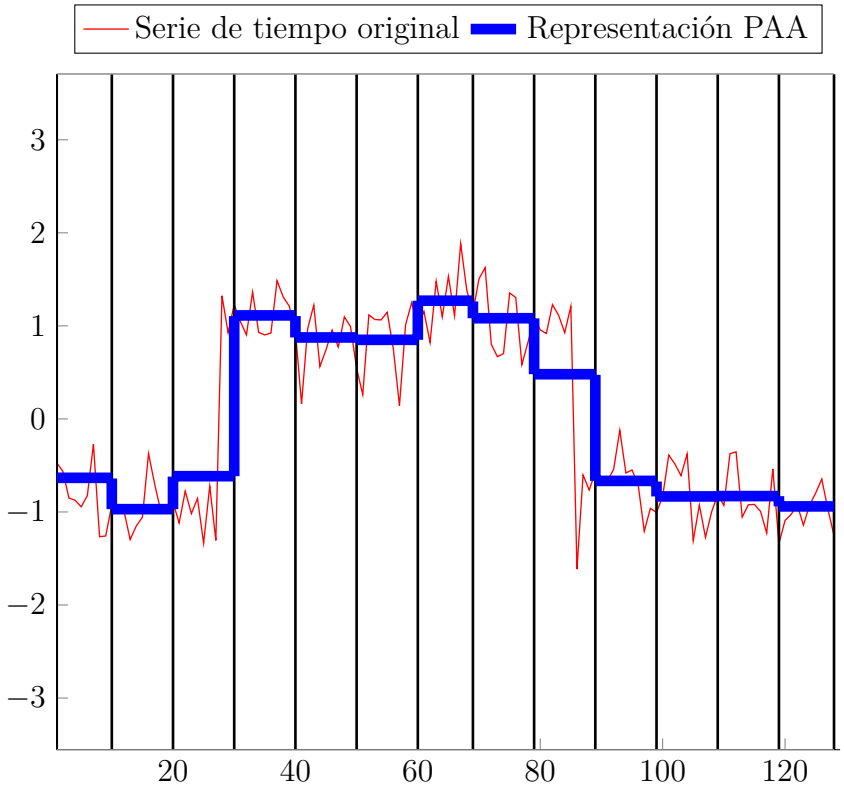


Figura 2.4: Ejemplo de reducción de dimensionalidad usando el método PAA

El método SAX inicia su proceso normalizando cada serie temporal de la base de datos para ajustarla a una distribución normal con media de cero y desviación estándar de uno. Posteriormente, se aplica el método PAA para obtener la serie temporal reducida con los valores promedios de cada intervalo calculado en el tiempo. SAX, llama a cada intervalo *segmento de palabra*, ya que por cada valor promedio se obtendrá un símbolo o letra que formará una palabra, que es la serie temporal discreta.

Definición 2.2.1. Segmentos de Palabra. *Conjunto de números que representan los cortes o intervalos en que se subdividirá la serie temporal.*

Para la asignación de los símbolos correspondientes, el algoritmo de SAX subdivide el espacio de valores en cortes generados de forma automática siguiendo una distribución normal. Cada corte representa una letra o símbolo en particular y son conocidos como *alfabeto*.

Definición 2.2.2. Alfabetos. *Valores ordenados de números obtenidos mediante distribución normal en el espacio de valores de la serie temporal y que representarán los símbolos de la serie temporal discreta.*

Una vez que se han definido los cortes del alfabeto, se realiza una comparación de cada promedio, obtenido por el método PAA, para verificar si se encuentra entre un intervalo determinado en el alfabeto, de ser así, el símbolo correspondiente en ese segmento será asignado a la serie discreta. Este proceso se realiza por cada segmento de palabra hasta formar la *palabra* o la *serie temporal discreta*. Este proceso es mostrado mediante el Algoritmo 2.

Definición 2.2.3. Palabra. *Conjunto de símbolos obtenidos en cada segmento de palabra y que forman la serie temporal discreta.*

Algoritmo 2 Proceso de discretización mediante el algoritmo SAX

Entrada: AS: Tamaño del alfabeto, WS: Numero de segmentos de palabra, TS: Serie de tiempo

1: $TS' = Normalizar(TS)$ ▷ *Normalizar* traslada la serie de tiempo para que tenga media cero y desviación estándar uno.
2: $W = ObtenerSegmentosPalabra(WS)$ ▷ *ObtenerSegmentosPalabra* obtiene WS segmentos de palabra equidistantes entre sí
3: $A = ObtenerAlfabetos(AS)$ ▷ *ObtenerAlfabetos* obtiene AS alfabetos mediante distribución normal
4: $\overline{TS} = PAA(W, TS')$ ▷ *PAA* obtiene la serie de tiempo reducida mediante el Algoritmo 1
5: **para** $i = 0, i \leq$ longitud$(\overline{TS})$ **hacer** ▷ *longitud* determina el número de valores en $\overline{TS}$
6: **para** $a = 0, a \leq$ longitud(A) **hacer**
7: **si** $\overline{TS}(i) > A(a)$ y $\overline{TS}(i) <= A(a+1)$ **entonces**
8: $D = AsignaSimbolo(i)$ ▷ *AsignaSimbolo* asigna el símbolo correspondiente al valor dado
devolver D ▷ Regresa la serie de tiempo simbólica encontrada

La Figura 2.5 muestra un ejemplo de discretización simbólica usando el procedimiento de SAX, donde se ilustran los cortes en el espacio temporal y el de valores, así como la asignación de los símbolos correspondientes.

Aunque SAX es uno de los métodos más utilizados dada su fácil implementación y sus buenos resultados, éste tiene varias desventajas enumeradas a continuación.

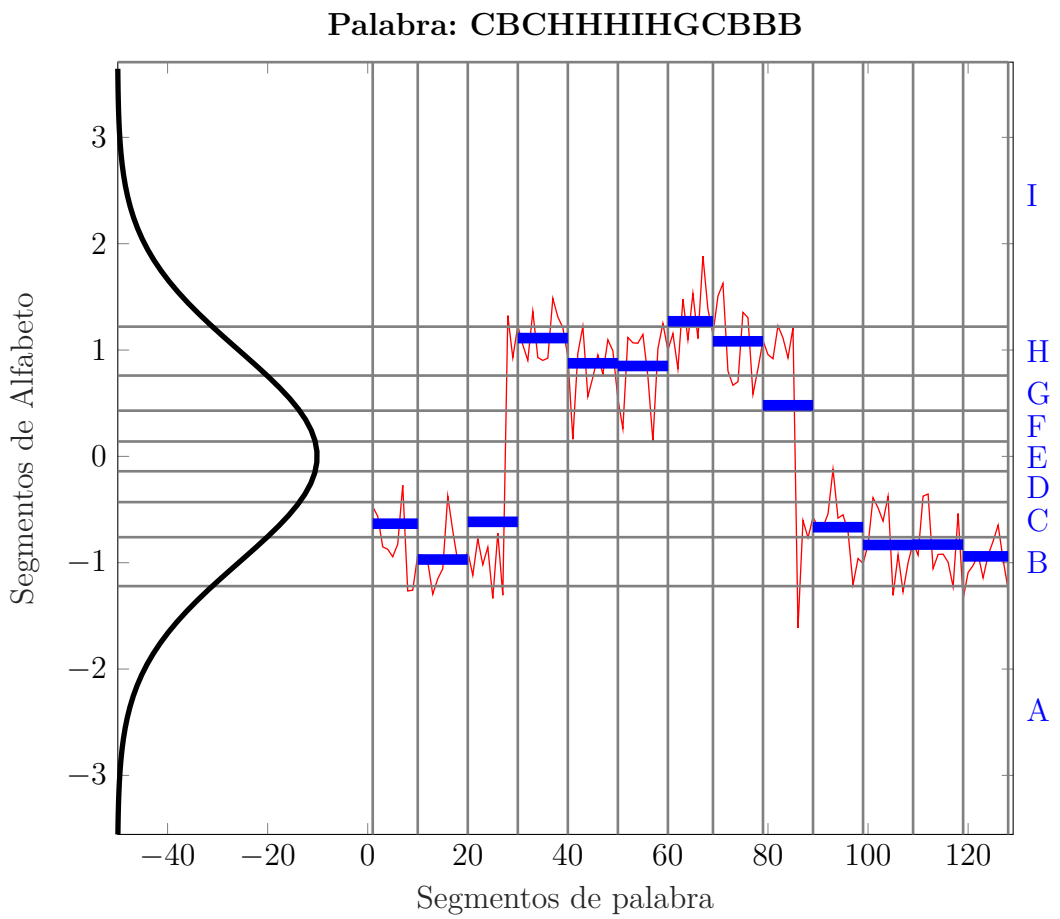


Figura 2.5: Ejemplo de discretización de series de tiempo continuas a series discretas simbólicas mediante el método de SAX

1. *Asunción de normalidad.* Asume que todas las series temporales a discretizar se ajustan a una distribución normal, aunque algunas no cumplan con esta condición.

2. *Número de segmentos de palabra y alfabeto.* Para poder realizar el proceso de discretización, SAX necesita los valores para el tamaño de la palabra y el número de cortes del alfabeto para la conversión. Sin embargo, encontrar los valores adecuados de estos parámetros no es tarea fácil, esto debido a que un número pequeño puede incurrir en la deformación de la serie de tiempo y valores altos minimiza el grado de discretización. Los valores de estas características depende del conocimiento a priori del conjunto de datos.

3. *Distribución de cortes.* SAX, genera los cortes para la palabra de forma

equidistante, es decir, con una misma distancia entre ellos. Mientras que para el alfabeto, como ya se mencionó, utiliza una distribución normal, lo que puede provocar que la serie de tiempo discreta no se ajuste a la serie original.

2.3. Medidas de Similitud o Distancia

Tener una medida de comparación entre dos series de tiempo es de vital importancia para las tareas de minería de datos y el análisis de series temporales [44]. Esta comparación puede ser vista como una medida de similitud entre dos series de tiempo, es decir, qué tanto se parecen una de la otra. Al hablar de similitud, se sobreentiende que se compararan dos series con diferentes características (forma, tamaño, entre otras), siendo un problema definir una medida que aproxime una "similitud" entre dos series de tiempo.

Existen diversos enfoques en los que se han agrupado las técnicas de comparación de series de tiempo, principalmente en este capítulo nos centraremos en tres medidas de similitud organizadas en dos categorías descritas a continuación [34].

Medidas punto a punto (Lock-step measures). En este tipo de medida se compara el i-ésimo punto de la serie temporal A con el i-ésimo punto de la serie temporal B; es decir, se comparan ambas series punto a punto. Se asume que ambas series son de mismo tamaño, de lo contrario, este tipo de métricas no pueden ser usadas. Una de las medidas de similitud más usadas de este tipo es la *distancia euclidiana*.

Medidas elasticas (Elastics measures). Por su parte, este grupo de medidas permite comparar segmentos de las series de tiempo para encontrar subsecuencias comunes en ambas series; es decir, no se hace una comparación punto a punto, sino que se comparan partes de ambas series para estimar la similitud. Estas medidas también permiten la comparación de series de tiempo de diferentes tamaños. Dentro de las más usadas se encuentran: *Dynamic Time Warping (DTW)* y *Subsecuencia Común más Larga (The Longest Common Subsequence)*.

A continuación se detallan cada una de las medidas anteriormente mencionadas.

2.3.1. Distancia Euclidiana

Es la medida de similitud más simple para series de tiempo [85]. Una de sus principales ventajas radica en la complejidad lineal que presenta, permitiendo su fácil implementación y rápida ejecución. Además, es un método que no conlleva ningún tipo de parámetro adicional, evadiendo la calibración inicial [34]. Sin embargo, asume que ambas series son del mismo tamaño, escala, base y no presentan huecos o segmentos vacíos [74]. Ello aunado a su sensibilidad al ruido presente en las series, hacen que en casos reales no sea la métrica idónea de usar para comparar dos o mas series de tiempo.

La Ecuación 2.2 representa la forma en que se calcula esta métrica, donde TS_1 y TS_2 son las series de tiempo a comparar y n es el tamaño de ambas series.

$$DistanciaEuclidiana(TS_1, TS_2) = \sqrt{\sum_{i=1}^{n}(TS_1(i) - TS_2(i))^2} \tag{2.2}$$

2.3.2. Alineamiento Temporal Dinámico (*Dynamic Time Warping, DTW*)

A diferencia de la distancia euclidiana, el método de Alineamiento Temporal Dinámico (*Dynamic Time Warping, DTW*) es una medida de similitud que considera la alineación óptima de dos series temporales obtenidas en períodos de tiempo distintos, siendo más robusto que las medidas punto a punto [1].

Para obtener la distancia entre dos series temporales, DTW necesita generar una *matriz de distancias (MD)* con el cálculo de las distancias entre todas las posibles subsecuencias encontradas en cada serie. Dadas dos series de tiempo $TS_1 = \{ts_1^{(1)}, ts_1^{(2)}, \cdots, ts_1^{(n)}\}$ y $TS_2 = \{ts_2^{(1)}, ts_2^{(2)}, \cdots, ts_2^{(m)}\}$; la MD es calculada mediante la Ecuación 2.3, donde $d(ts_1^{(i)}, ts_2^{(j)})$ es la distancia entre las subsecuencias de las series de tiempo TS_1 y TS_2.

$$MD = \begin{bmatrix} d(ts_1^{(1)}, ts_2^{(1)}) & d(ts_1^{(1)}, ts_2^{(2)}) & \cdots & d(ts_1^{(1)}, ts_2^{(m)}) \\ d(ts_1^{(2)}, ts_2^{(1)}) & d(ts_1^{(2)}, ts_2^{(2)}) & & \\ \vdots & & \ddots & \\ d(ts_1^{(n)}, ts_2^{(1)}) & & & d(ts_1^{(n)}, ts_2^{(m)}) \end{bmatrix} \tag{2.3}$$

La principal idea de DTW es calcular la ruta óptima en MD que minimice la Ecuación 2.4, donde k es el número total de distancias encontradas y $w_i = MD(a, b),\ 1 \leq a \leq n,\ 1 \leq b \leq m$.

$$DTW(TS_1, TS_2) = min\left(\sqrt{\sum_{i=1}^{k} w_i}\right) \tag{2.4}$$

Sin embargo, la ruta óptima esta sujeta a varias limitaciones que aseguran una alineación idónea entre las series de tiempo [58]. (1) *Límites*, se debe asegurar que w_1 y w_k correspondan al primer y último elemento de MD y no se superen esas posiciones. (2) *Continuidad*, se debe cumplir que dos distancias dentro de la ruta sean subsecuentes y no existan brechas grandes entre ellas. (3) *Monotonía*, cada distancia de la ruta deben estar ordenados en el tiempo, es decir, la primera distancia debe ser menor que la segunda, y así sucesivamente. Una forma para cumplir con cada una de las restricciones es utilizar la programación dinámica [23].

La programación dinámica requiere el cálculo de una matriz de costo acumulado (MCA) del mismo tamaño que MD [1]. MCA sustituirá a MD en la minimización de la Ecuación 2.4, es decir ahora $w_i = MCA(a, b)$. Dicha matriz se calcula de acuerdo a la Ecuación 2.5.

$$MCA(a, b) = d(ts_1^a, ts_2^b) + min\left\{MCA(a-1, b-1), MCA(a-1, b), MCA(a, b-1)\right\} \tag{2.5}$$

En la literatura existen diversas modificaciones a esta medida de similitud, pero en general, cada enfoque busca la alineación correcta y por consiguiente una estimación de similitud correcta dependiendo de las características de las series temporales.

2.3.3. Subsecuencia Común más Larga (*The Longest Common Subsequence*)

Otra medida de similitud entre dos series de tiempo que busca la alineación de ambas mediante la comparación de subsecuencias o partes de ellas, es el método conocido como la *Subsecuencia Común más Larga (*The Longest Common Subsequence, LCS*)*.

Una de las principales ventajas de este método es la robustez al ruido o datos atípicos, a diferencia de DTW, que si ve afectado su funcionamiento con series de tiempo con este tipo de problemas [23]. Ello se debe a que este método permite alinear series de tiempo con subsecuencias no encontradas en otra serie, es decir, que no se pueden relacionar [85]. La Figura 2.6 muestra una comparación gráfica entre la distancia euclidiana, DTW y LCS, donde se puede apreciar que LCS no relaciona segmentos de la segunda serie de tiempo que no se muestran en la primera.

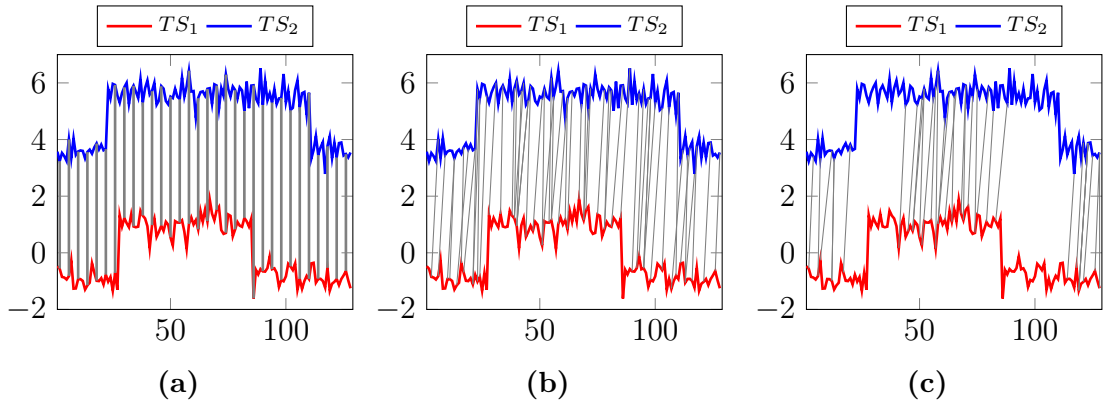


Figura 2.6: Comparación de alineación de series de tiempo usando las medidas de similitud: (a) distancia euclidiana, (b) DTW, y (c) LCS. Las lineas entre las dos series temporales representan la correspondencia encontrada entre cada serie.

Al igual que DTW, LCS construye una matriz muy parecida a MCA. Sin embargo, en lugar de considerar distancias, dicha matriz (mLCS) considera similitudes entre subsecuencias. Dicha similitud es la longitud de la subsecuencia común mas larga. Por ejemplo, imaginemos que tenemos dos series de tiempo $X = \{3, 2, 4, 6, 7, 5, 1\}$ y $Y = \{9, 3, 4, 8, 7, 5, 1\}$, la subsecuencia común mas larga es $Z = \{3, 4, 7, 5, 1\}$, la cual es la secuencia de elementos que consecutivamente se repiten en ambas series [85].

Sean $TS_1 = \{ts_1^{(1)}, ts_1^{(2)}, \cdots, ts_1^{(n)}\}$ y $TS_2 = \{ts_2^{(1)}, ts_2^{(2)}, \cdots, ts_2^{(m)}\}$ dos series de tiempo, mLCS se calcula mediante la Ecuación 2.6. Cada elemento de la matriz es la subsecuencia común más larga de todos los posibles segmentos de ambas series de tiempo.

$$mLCS(a,b) = \left\{ \begin{array}{cc} 0 & a = 0 \\ 0 & b = 0 \\ 1 + mLCS[a-1, b-1] & Si\ ts_1^a = ts_2^b \\ max(mLCS[a-1, b], mLCS[a, b-1]) & caso\ contrario \end{array} \right\} \quad (2.6)$$

Una vez obtenida mLCS, la distancia entre dos series temporales TS_1 y TS_2 se obtiene con la Ecuación 2.7, donde n y m son las longitudes de TS_1 y TS_2, respectivamente, y l es la subsecuencia común más larga encontrada.

$$LCS(TS_1, TS_2) = \frac{n + m + 2l}{m + n} \quad (2.7)$$

La complejidad computacional de cada uno de esto métodos depende del tamaño de las series temporales. Para DTW y LCS la complejidad es medida como $O(mn)$ [23], donde m y n son los tamaños de las series comparadas. Para el caso de la distancia euclidiana, la complejidad es lineal, es decir $O(n)$ [34], debido a que hace comparaciones directas en series de tiempo de mismo tamaño.

2.4. Clasificación en Series de Tiempo

La *clasificación* es una tarea de minería de datos encargada de asignar una categoría o *etiqueta de clase*, mediante la construcción de modelos, objetos o casos a través del aprendizaje que obtiene de los mismos datos (aprendizaje supervisado) [17, 97, 56]. En esta tarea se asume conocimiento previo del problema a resolver [74], el cual constituye un conjunto de datos llamado *conjunto de entrenamiento*, con el que cada técnica aprende para realizar la clasificación. Las técnicas probablemente más conocidas de clasificación en minería de datos son: *k-NN: Vecinos más Cercanos*, *Clasificador Bayesiano* y *Árboles de Decisión*. Cada una de estas técnicas fue creada para bases de datos no temporales, sin embargo pueden ser extendidas para el manejo de series de tiempo.

2.4.1. k-NN: Vecinos más Cercanos

Es una de las técnicas más simples de clasificación de datos. La idea básica consiste en asignar a un caso sin clase, la categoría del caso más cercano de un conjunto de k instancias, llamadas *vecinos*. Para estimar la cercanía de dos puntos se usan medidas de distancia como la distancia euclidiana. k es un parámetro inicial del algoritmo y su funcionamiento varía dependiendo del valor que tome dicho parámetro.

La Figura 2.7 muestra un ejemplo del funcionamiento del algoritmo K-NN usando un conjunto de 5 casos conocidos ($K = 5$), donde el objeto con menor distancia fue el triángulo, asignando dicha forma al caso desconocido (Figura 2.7b).

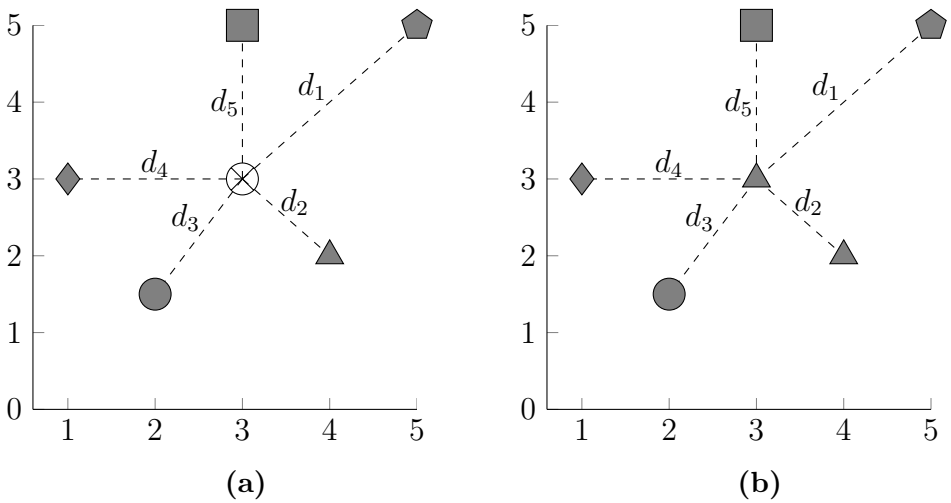


Figura 2.7: Ejemplo del funcionamiento del algoritmo k-NN, para $k = 5$. La forma blanca en la Figura (a) representa el caso desconocido, $d_i, i = \{1, \ldots, 5\}$ representan las distancias de ese punto a cada caso conocido. La Figura (b) muestra como el caso desconocido tomó la forma de la figura más cercana encontrada.

A pesar de ser una técnica simple y sencilla de aplicar, tiene varias consideraciones que deben tomarse en cuenta para su uso [74]:

- *Valor adecuado para k.* Como ya se mencionó anteriormente, el algoritmo es sensible al número de vecinos donde se buscará la clase correspondiente al caso desconocido. A valores más grandes de k, pueden encontrarse más de una clase; mientras que, a valores pequeños, el algoritmo puede ser afectado por ruido.

- *Medida de similitud.* El uso de alguna técnica de estimación de distancia puede afectar al rendimiento del algoritmo, debido a que también éstas dependen del tipo de datos, su distribución y su dimensionalidad.

- *Error.* El error del algoritmo K-NN se aproxima de forma asíntotica al error de Bayes [74].

El Algoritmo 3 detalla el procedimiento general para clasificar bases de datos usando el procedimiento de vecinos más cercanos.

Algoritmo 3 Algoritmo de clasificación del k-NN vecino más cercano

Entrada: k: Número de vecinos más cercanos, E: Conjunto de instancias de entrenamiento, F: Conjunto de instancias de prueba

1: **para** $f = (x', y') \in F$ **hacer**
2: Calcular la distancia $d(x', x)$ entre f y $e = (x, y) \in E$.
3: Seleccionar las instancias de E más cercanas a las instancias de F.
4: Elegir la etiqueta para y' de la etiqueta más frecuente del conjunto de vecinos más cercanos.

Para clasificar bases de datos temporales mediante esta técnica, basta usar una de las medidas de similitud mencionadas en la Sección 2.3 como estimador de la cercanía entre dos series de tiempo.

2.4.2. Clasificador Bayesiano Ingenuo

El clasificador Bayesiano Ingenuo es una técnica de clasificación estadística basada en el Teorema de Bayes, el cuál, calcula las probabilidades condicionales de que una instancia desconocida pertenezca a otra con clase definida. Para ello, se asume que todas las características de la base de datos son condicionalmente independientes, es decir, cada una cambia su valor independientemente de las otras [56].

Supongamos que tenemos una instancia $I = \{at_1, at_2, \cdots, at_q\}$ de clase desconocida y un conjunto de posibles clases para cada instancia de ese conjunto $CL = \{c_1, c_2, \cdots, c_m\}$. El clasificador bayesiano ingenuo consiste en calcular las probabilidades condicionales (distribución posterior) de que I pertenezca a un elemento de C. Dicha probabilidad condicional se calcula mediante la Ecuación 2.8, donde $P(c_j)$ y $P(I)$ son las probabilidades de que la clase c_j y la instancia I se

presenten en el conjunto de entrenamiento respectivamente, siendo este último sólo es un factor de normalización, por lo tanto puede no ser ocupado. Para calcular $P(I|c_j)$ se utiliza la Ecuación 2.9, que es la multiplicación de las probabilidades de que los q elementos de I pertenezcan a la clase c_j.

$$P(c_j|I) = \frac{P(I|c_j)P(c_j)}{P(I)} \tag{2.8}$$

$$P(I|c_j) = \prod_{p=1}^{q} P(at_p|c_j) \tag{2.9}$$

La idea del método es calcular todas las posibles probabilidades para cada $c_i \in C$, de tal forma que aquella con la mayor probabilidad determina la clase a la que corresponde la instancia de muestra.

2.4.3. Árboles de Decisión

El clasificador por árboles de decisión es una herramienta fuertemente usada en problemas del mundo real [59]. La idea básica es construir un conjunto de reglas que permitan inferir la clase a nuevas instancias con clase desconocida. Ese conjunto de reglas es presentada mediante nodos y aristas que permiten asignar una etiqueta de clase a una instancia. A este conjunto de nodos y aristas se le llaman *árboles de decisión*.

Un árbol de decisión es un modelo jerárquico no paramétrico donde las regiones para una clase en particular son identificadas mediante subdivisiones recursivas a través de nodos de decisión [56]. La Figura 2.8 muestra los elementos de un árbol de decisión, los cuales se detallan a continuación.

- *Nodos raíz.* Es el nodo principal, con el que se inicia el proceso de búsqueda en el árbol. No contiene aristas entrantes y de él pueden o no partir aristas, en el peor de los escenarios.

- *Nodos internos.* Son nodos que reciben solo una entrada y de él se derivan una o más aristas dependiendo del dominio del atributo.

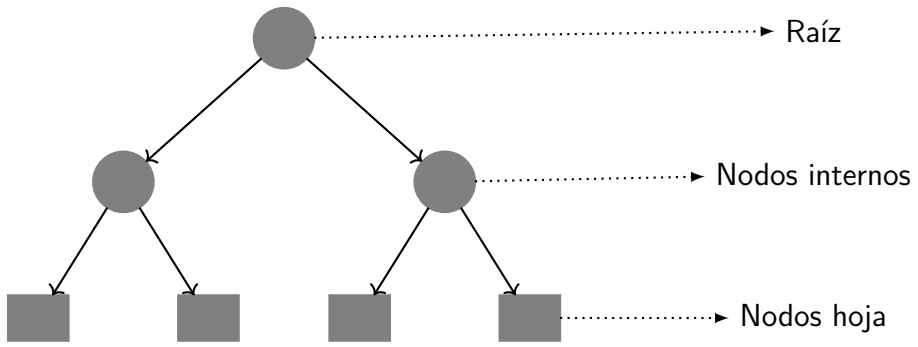


Figura 2.8: Elementos de un árbol de decisión.

- *Nodos hoja o terminales.* Son nodos que reciben solo una arista o entrada y no sale ningún dato de ellos. Son el último nivel del árbol y contienen el valor de clase que se está buscando.

En clasificación, los nodos que no son terminales contienen los atributos, las aristas contienen las reglas de separación de los atributos y los nodos hojas contienen las clases que, dada las reglas, deben contener.

La construcción de árboles de decisión a partir de los datos, es usualmente generada mediante el algoritmo de Hunt [97], el cual es la base para los métodos de clasificación de árboles de decisión ID3, C4.5 y CART. El algoritmo de Hunt consiste en dividir, en forma recursiva, los registros de entrenamiento en subconjuntos más puros, es decir, en subconjuntos que sólo contengan una clase.

Definición 2.4.1. *Sea* $CL = \{c_1, c_2, \cdots, c_C\}$ *un conjunto de clases y* $T_{at} = \{(v_1, y_1), (v_2, y_2), \cdots, (v_t, y_t)\}$ *un conjunto de registros de entrenamiento para el atributo at, donde* v_t *es un dato de la base y* y_t *es la etiqueta del registro, un árbol de decisión es generado por el algoritmo de Hunt siguiendo los pasos que a continuación se describen [97]:*

1. $\forall y \in T_{at}$*, si "y" pertenece a la misma clase entonces el atributo "at" es un nodo hoja del árbol etiquetado como "c".*

2. *Si* $T_{at} \in \varnothing$ *entonces se crea un nodo hoja asignándole la clase mas frecuente del padre [56].*

3. *$\forall y \in T_{at}$, si "y", contiene diferente etiqueta de clase, entonces se crea un nodo interno con el atributo "at" y se selecciona una condición que divida a T_{at} en subconjuntos. Cada subconjunto forma un nodo hijo del nodo interno creado y los datos de T_{at} se distribuyen de acuerdo a la condición creada. Por cada nodo hijo se repite el proceso desde el paso 1.*

Sin embargo, seleccionar el atributo "*at*" que será el nodo raíz del árbol y cada condición de separación de los datos no es tarea fácil. Para ello, se ha propuesto el uso de la ganancia de la información para hacer las divisiones e ir seleccionando los nodos internos y hojas. Dicha métrica de separación está basada en el cálculo de entropía, la cual es una medida de aleatoriedad o caos dentro de un conjunto de datos.

Definición 2.4.2. *Sea $T_{at} = \{(v_1, y_1), (v_2, y_2), \cdots, (v_t, y_t)\}$ un conjunto de registros de entrenamiento para el atributo "at" y $CL = \{c_1, c_2, \cdots, c_C\}$ el conjunto de clases, la entropía se calcula mediante la Ecuación 2.10, donde $p(T_{at}|c_i)$ es la frecuencia relativa de la clase c_i del nodo T_{at} [56].*

$$E(T_{at}) = -\sum_{i=1}^{C} p(T_{at}|c_i) \cdot \log_2(p(T_{at}|c_i)) \tag{2.10}$$

Definición 2.4.3. *Sea $T_{at} = \{(v_1, y_1), (v_2, y_2), \cdots, (v_t, y_t)\}$ un conjunto de registros de entrenamiento para el atributo "at" y $CL = \{c_1, c_2, \cdots, c_C\}$ el conjunto de clases, la ganancia de la información se calcula usando la Ecuación 2.11, donde k es el número de particiones realizadas, t_i es el tamaño de la partición i [84].*

$$IG(T_{at}) = E(T_{at}) - (\sum_{i=1}^{k} \frac{t_i}{t} E(i)) \tag{2.11}$$

La ganancia de la información calcula la impuridad de cada atributo, mediante la resta de la Entropía de dicho atributo antes del proceso de separación y el promedio de las entropía de los grupos formados después de la separación. El atributo con el valor más alto de ganancia de la información es seleccionado para ser el nodo raíz o interno dependiendo del nivel del árbol en construcción. El número de divisiones por cada atributo depende del tipo de dato que contiene,

es decir, si es categórico, el número de divisiones será el número de categorías encontradas, y si es numérico, se calcula un umbral de separación donde la entropía sea menor al generar dos nodos hijos.

Aún cuando la ganancia de la información obtiene buenos resultados con árboles compactos, los árboles con muchos nodos hijos son preferidos a diferencia de nodos con pocas salidas [56]. Para solucionar esta situación se agrega un término de normalización, el cual es la entropía de separar un nodo en p particiones. A la ganancia de la información con dicho término se le conoce como *porcentaje de ganancia de información.*

Definición 2.4.4. *El porcentaje de ganancia de información es la ganancia de la información del atributo "at" dividida entre la entropía del conjunto de separaciones creadas para "at". La Ecuación 2.12 muestra cómo se calcula esta métrica.*

$$PorcentajeIG(T_a) = \frac{IG(T_a)}{-\sum_{i=1}^{k} \frac{|T_{a,i}|}{|T|} \cdot \log_2\left(\frac{|T_{a,i}|}{|T|}\right)} \tag{2.12}$$

El porcentaje de ganancia de información refleja la proporción de datos generados por la división que es útil, es decir, que parece útil en la clasificación [56].

El Algoritmo 4 muestra una estructura general para la construcción de un árbol de decisión basado en un conjunto de instancias de entrenamiento I.

Algoritmo 4 Proceso de construcción de un árbol de decisión

Entrada: E: Conjunto de instancias de entrenamiento, A: Conjunto de atributos

1: **si** E contiene la misma etiqueta de clase o los mismos valores de atributo **entonces**
2: Se crea nodo hoja con la etiqueta de la clase más frecuente en el conjunto de entrenamiento **devolver** nodo hoja
3: **si no**
4: Se crea nodo raíz con el atributo con la mayor ganancia de la información calculada con la Ecuación 2.11.
5: **para** cada valor del nodo raiz **hacer**
6: Se crea un nodo hijo del nodo raíz
7: Se repite el paso 1
devolver nodo raíz

Generalmente, después de la creación de los árboles de decisión, se realiza un post-procesamiento para eliminar ramas que no aportan información al árbol y solo

aumentan la complejidad del mismo. Esta etapa es considerada como adicional. Sin embargo, es importante verificar que los árboles generados no entorpezcan el proceso de clasificación al ser de profundidad alta y complejos.

2.5. Agrupamiento (*Clustering*) en Series de Tiempo

El proceso de agrupamiento (o *clustering* en inglés) consiste en subdividir un conjunto de instancias u observaciones en grupos (*clusters* en inglés) dependiendo de la similitudes entre las instancias, es decir, cada grupo contiene observaciones similares entre sí [53]. La Figura 2.9 ejemplifica cómo se realizaría el proceso de agrupamiento. Este tipo de método es considerado como *aprendizaje no supervisado*, dado que se desconoce una clasificación de los datos y lo que se pretende es establecer una mediante la separación de las observaciones. Dentro de los métodos de agrupamiento más usados está el método llamado *K-Medias*.

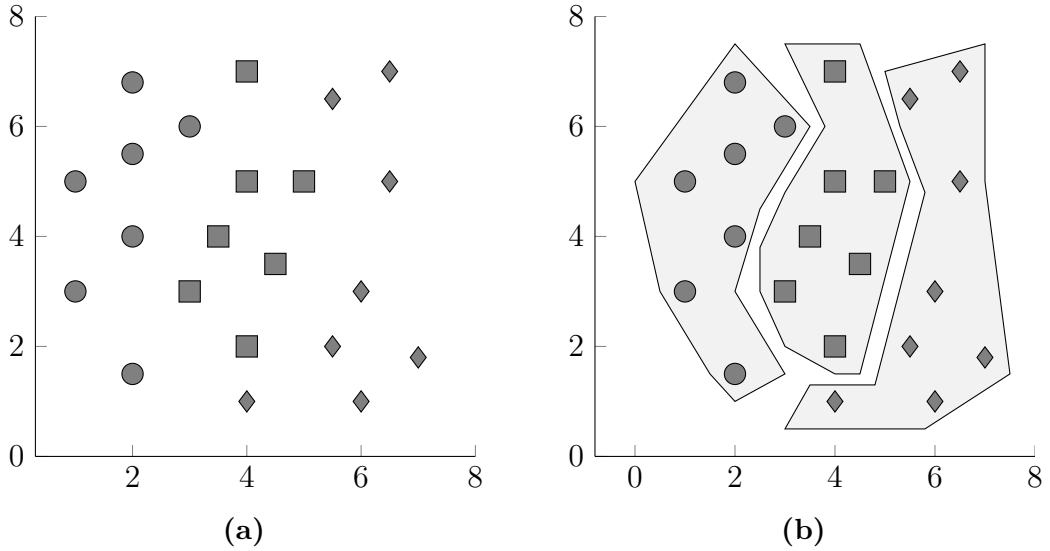


Figura 2.9: Ejemplificación del proceso de agrupamiento de instancias, (a) datos antes de la generación de los grupos, y (b) datos agrupados de acuerdo al parecido entre ellos.

2.5.1. K-Medias

El método de agrupación llamado *K-Medias* es el algoritmo más popular dentro de esta tarea de minería de datos [74]. K-Medias consiste en encontrar los valores medios

o centroides de las observaciones como guías para crear los grupos y definir que instancias corresponden a cada uno. Los centroides se van modificando de acuerdo a las observaciones existentes en cada grupo. Una vez modificados los centroides se recalculan las distancias y se vuelven a asignar las instancias que deben contener cada grupo. Este proceso es iterativo hasta que se minimice el error definido en la Ecuación 2.13, donde I es una observación o instancia, $O = \{o_1, o_2, \cdots, o_k\}$ es el conjunto de grupos de instancias, $\overline{o_i}$ es el centroide del grupo i y $dist$ es una función de similitud entre dos objetos. Dado que la búsqueda realizada es una heurística voraz, K-Medias es bueno encontrando valores locales óptimos pero no el óptimo global [74]. El Algoritmo 5 detalla el procedimiento del método K-Medias.

$$Error = \sum_{i=1}^{k} \sum_{I \in O_i} dist(I, \overline{o_i})^2 \tag{2.13}$$

Algoritmo 5 Proceso de agrupación mediante el algoritmo K-Medias

Entrada: k: Número de grupos, D: Conjunto de observaciones o instancias

1: Escoger aleatoriamente k centroides de D
2: $E = Inf()$ ▷ $Inf()$ es una función que devuelve un valor alto
3: **mientras** $E \leq \delta$ **hacer** ▷ δ es un umbral definido para determinar la condición de paro
4: Reasignar las observaciones a cada grupo que sean más cercanas al centroide de cada grupo.
5: Actualizar los valores de los centroides de acuerdo a las observaciones que pertenecen a cada grupo.
devolver El conjunto de k grupos

Este algoritmo solo se puede utilizar cuando es posible determinar los centroides dentro de un conjunto de observaciones. Por ejemplo, para tipos de datos categóricos, se utiliza el método *k-modas*, donde en lugar de calcular medias, se calculan las modas de los datos para generar los grupos.

Otra limitación es el valor óptimo para k. Si k es alto, el número de observaciones por grupo disminuye, generando soluciones parecidas al conjunto original. Por el contrario, si k es bajo se tienen grupos con observaciones poco similares entre ellas, acarreando malos resultados al análisis.

3

Optimización Multi-Objetivo

Contenido

3.1. Introducción 43
3.1.1. Dominancia de Pareto 45
3.1.2. Frente y Conjunto Óptimo de Pareto 46
3.1.3. Selección de Preferencias 47
3.2. Métodos Clásicos para Optimización Multi-objetivo . 48
3.2.1. Suma Ponderada de Funciones (SPF) 49
3.2.2. Método ϵ-Constraint 49
3.2.3. Método de Programación por Metas 50
3.2.4. Método Lexicográfico 51
3.2.5. Método de Obtención de Metas 52
3.3. Algoritmos evolutivos para optimización multi-objetivo 52
3.3.1. Basados en Pareto 52
3.3.2. Basados en Descomposición 55
3.3.3. Basados en Métricas 56
3.4. Non-dominated Sorting Genetic Algorithm II (NSGA-II) 56

3.1. Introducción

Optimizar significa encontrar valores ideales a variables para la solución de un problema en particular [86]. Habitualmente, se suele utilizar este proceso para encontrar la mejor solución de un problema basados en una sola función objetivo,

a este tipo de problema se le conoce como *optimización de un solo objetivo o mono-objetivo (SOOP, en sus siglas en inglés)*. Una función objetivo es una métrica utilizada para medir la calidad de las soluciones potenciales a un problema [18].

Sin embargo, existen otro tipo de situaciones donde se tienen dos o mas objetivos que se requieren optimizar para resolver un problema; y por lo general, estos objetivos suelen estar en conflicto unos con otros, lo que significa que, al mejorar un objetivo, el(los) otro(s) suele(n) empeorar. Por tal motivo es necesario buscar soluciones que optimicen ambas funciones sin que ninguna se vea afectada por la otra, para ello se utiliza la *optimización multi-objetivo*.

El problema de optimización multi-objetivo (MOOP, en sus siglas en inglés) consiste en buscar soluciones que maximicen o minimicen dos o más funciones objetivo simultáneamente [31]. El MOOP es expresado mediante la Ecuación 3.1, donde L es el número de variables de decisión, F es el número de funciones objetivo, $\vec{x}$ es una solución potencial al problema conteniendo al conjunto de variables de decisión, $\vec{f}$ es el conjunto de valores obtenidos al evaluar $\vec{x}$ en cada función objetivo, y por último, $h(\vec{x})$ y $g(\vec{x})$ son funciones de igualdad y desigualdad, respectivamente. Estas funciones determinan la región factible $\mathcal{F}$ de donde se encuentran las soluciones potenciales al problema [31].

$$
\begin{aligned}
&\text{Hallar } \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_L \end{bmatrix} \in \mathbb{R}^L \text{ tal que} \\
&\text{optimice } \vec{f}(\vec{x}) = \begin{bmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_F(\vec{x}) \end{bmatrix} \\
&\text{Sujeto a:} \\
&h_n(\vec{x}) = 0 \\
&g_n(\vec{x}) \leq 0
\end{aligned}
\tag{3.1}
$$

De la Ecuación 3.1, se distinguen dos tipos de espacios o conjuntos utilizados para el proceso de optimización: *el espacio de las variables de decisión* y *el*

espacio de las funciones objetivo. En el primero, se encuentran los valores de cada una de las variables involucradas para resolver el problema, mientras que en el segundo, se encuentran los valores resultantes de evaluar una solución en cada función objetivo [31].

A diferencia de los SOOPs donde el proceso de optimización contempla un único óptimo, en un MOOP el óptimo consiste de un conjunto de soluciones con un buen compromiso entre todas las funciones objetivos.

Para encontrar dicho conjunto es utilizado los conceptos de *optimalidad* expuestos por Edgeworth en 1881 y generalizados por Pareto en 1886 [78].

3.1.1. Dominancia de Pareto

En problemas donde se busca optimizar un único objetivo, encontrar una solución que sea la mejor posible de todo el espacio de búsqueda es fácil de determinar, ya que se puede hacer una comparación directa entre ellas mediante la calidad determinada por la función objetivo. Sin embargo, cuando dos o más funciones son involucradas, la comparación ya no es directa, por lo que es necesario utilizar métodos para discriminar o discernir entre soluciones mediante múltiple objetivos.

Edgeworth y Pareto propusieron una de las técnicas más usadas en los MOOPs llamada *optimalidad de Pareto.* Dicha técnica determina si una solución es mejor que otra mediante el término de dominancia de soluciones, o también llamado *dominancia de Pareto.*

Definición 3.1.1. Dominancia de Pareto. *Una solución $\vec{x}_1$ es dominada por $\vec{x}_2$, si y solo si $f_i(\vec{x}_2) \leq f_i(\vec{x}_1)\ \forall\ i \in \{1, 2, \ldots, F\}$ y $f(\vec{x}_2) < f(\vec{x}_1)$ al menos en un objetivo. Esta dominancia es expresada como $\vec{x}_2 \prec \vec{x}_1$ [70].*

La dominancia de Pareto determina las soluciones que cumplen con el compromiso óptimo en todos los objetivos del conjunto de soluciones encontradas en la región factible $\mathcal{F}$.

La Figura 3.1 muestra un ejemplo de las soluciones que cumplen con un compromiso, asumiendo minimización, en las dos funciones objetivo del problema

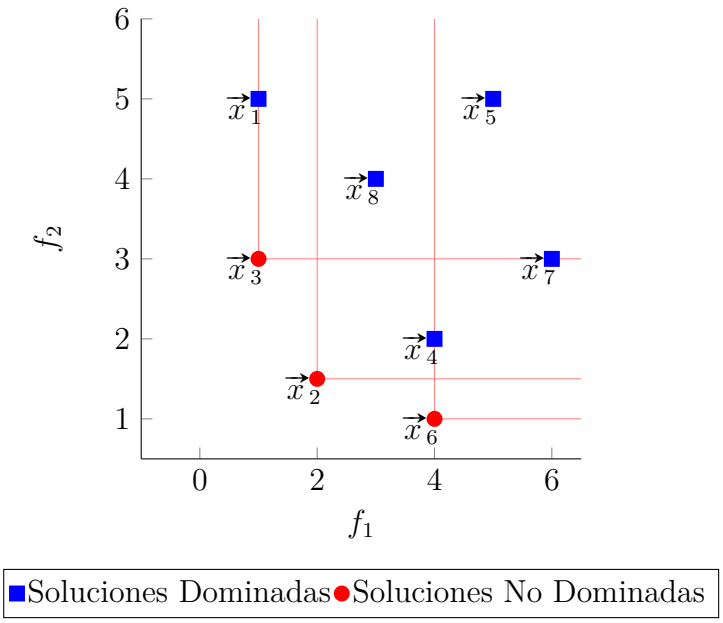


Figura 3.1: Ejemplo de dominancia de Pareto en un problema con dos funciones objetivo, donde x_2, x_3, y x_6 dominan a las demás soluciones en todos los objetivos.

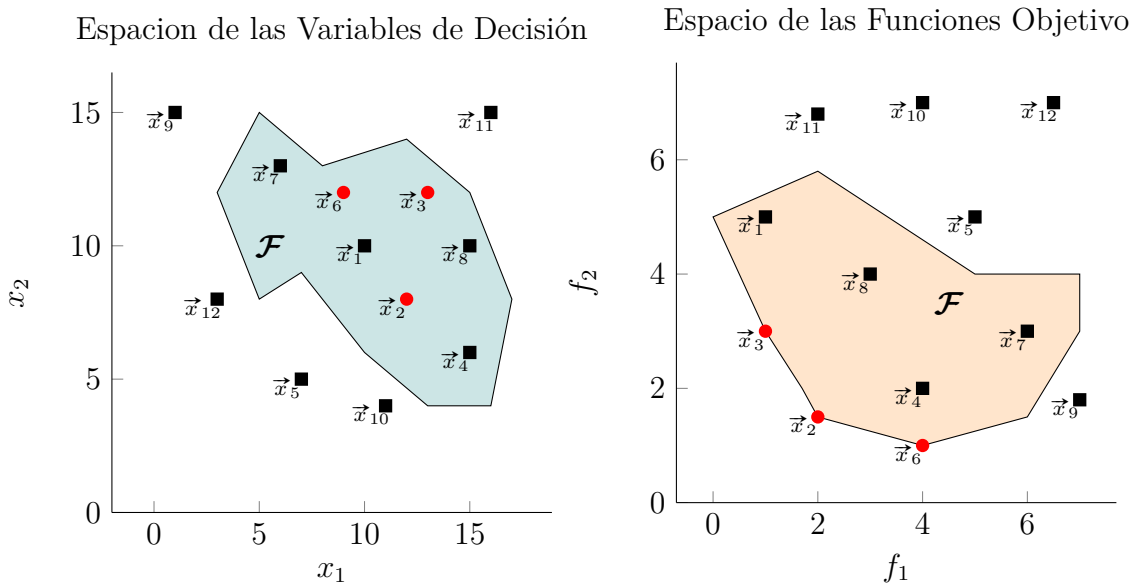
(f_1 y f_2), donde $\vec{x}_3$ es mejor que $\vec{x}_5$ y $\vec{x}_8$ en ambos objetivos y tienen la misma calidad que las soluciones $\vec{x}_1$ y $\vec{x}_7$ en al menos un objetivo. Por otro lado, $\vec{x}_2$ tiene una mejor evaluación que $\vec{x}_4$, $\vec{x}_5$, $\vec{x}_7$, y $\vec{x}_8$ en ambas funciones. Por último, $\vec{x}_6$ tiene el mismo valor de calidad que $\vec{x}_4$ y domina a $\vec{x}_5$ y $\vec{x}_7$ en los dos objetivos. Por lo tanto, $\vec{x}_2$, $\vec{x}_3$ y $\vec{x}_6$ son soluciones que cumplen con la Definición 3.1.1, siendo el conjunto de *soluciones no dominadas* debido a que dominan al resto de soluciones en ambas funciones. Al conjunto de soluciones que son dominadas por al menos una solución se les conoce como *soluciones dominadas*.

3.1.2. Frente y Conjunto Óptimo de Pareto

A la solución dentro del espacio de variables de decisión que representa una solución no dominada en el espacio de las funciones objetivo se le conoce como *Solución Óptima de Pareto*.

Definición 3.1.2. Solución Óptima de Pareto ($\vec{x}^*$). *Una solución $\vec{x}_i \in \mathcal{F}$ se considera solución óptima de Pareto, si no exista otra $\vec{x}_j \in \mathcal{F}$ tal que $\vec{f}(\vec{x}_j) \prec \vec{f}(\vec{x}_i)$*

La Figura 3.2 muestra la distribución de las soluciones, tanto en el espacio de las funciones como el de los valores, en un problema con dos objetivos. La Figura 3.2b muestra el conjunto de soluciones no dominadas en el espacio de las funciones objetivo, el cual se le conoce como *Frente de Pareto ($\mathcal{PF}$)*. Por otro lado la Figura 3.2a muestra a las soluciones no dominadas en el espacio de las variables de decisión, el cual es conocido como conjunto óptimo de Pareto.



(a) Conjunto de Óptimos de Pareto **(b)** Frente Óptimo de Pareto

Figura 3.2: Representación del conjunto y frente de Pareto. Los círculos representan las mejores soluciones vistas tanto en el conjunto de Pareto, llamadas soluciones óptimas de Pareto (a), como en el frente de Pareto conocidas como soluciones no dominadas (b).

Definición 3.1.3. Conjunto Óptimo de Pareto ($\mathcal{POF}$). *El conjunto óptimo de Pareto es definido como* $\mathcal{POF} = \{\vec{x} \in \mathcal{F} \mid \vec{x} = \vec{x}^*\}$ *[72].*

Definición 3.1.4. Frente Óptimo de Pareto ($\mathcal{PF}$). *En los MOOPs, el frente óptimo de Pareto* $\mathcal{PF}$ *es definido como* $PF = \{\vec{f}(\vec{x}) \mid \vec{x} \in \mathcal{POF}$ *[72].*

3.1.3. Selección de Preferencias

Los métodos propuestos en la literatura especializada para la solución de problemas multi-objetivo utilizan dos fases para encontrar los valores óptimos

a las variables del problema a resolver: (a) Encontrar el Frente de Pareto, y (b) Seleccionar una o un subconjunto de soluciones de dicho frente [41].

Esta última fase representa una tarea difícil para el tomador de decisiones [22], puesto que todos los elementos del frente de Pareto representan buenas soluciones a su problema. Este proceso es llamado *Manejo de Preferencias* debido a que el tomador de decisiones incluye información adicional necesaria para decidir sobre los elementos del Frente de Pareto.

Definición 3.1.5. Manejo de Preferencias. *Proceso de seleccionar una solución o un subconjunto de ellas del Frente de Pareto, tomando en cuenta las preferencias del tomador de decisiones [22, 96].*

En problemas reales, es complicado definir un método para la selección de preferencias; es por ello que, existen una alta cantidad de manejadores de preferencias definidas acorde al problema que se está optimizando, y que no necesariamente son adecuados a otros problemas, incluso si optimizan problemas similares.

3.2. Métodos Clásicos para Optimización Multi-objetivo

En la literatura especializada [54, 18, 31, 70, 51, 28], existen varios métodos matemáticos o clásicos para la optimización de problemas con múltiples objetivos. Dichos métodos se centran en convertir el problema multi-objetivo en problemas de optimización de un sólo objetivo.

Una vez que el problema multi-objetivo ha sido transformado, ahora es posible abordarlo con métodos de optimización mono-objetivo clásicos, permitiendo una fácil interpretación de los resultados.

En este apartado nos enfocaremos en describir algunos de estos enfoques que han sido usados para resolver problemas con más de un objetivo.

3.2.1. Suma Ponderada de Funciones (SPF)

Este método es uno de los enfoques más comunes y simples utilizados en la resolución de problemas multi-objetivo [35]. Fue propuesto por Gass et al. en [50], en el año 1955 y consiste en minimizar un problema multi-objetivo mediante la transformación del conjunto de funciones en un solo objetivo. Lo anterior se realiza sumando cada una de las funciones y ponderando su importancia mediante pesos. La Ecuación 3.2 expresa dicha transformación, donde u_i representa el peso o prioridad de cada función dependiendo de las preferencias del tomador de decisiones. Dichos pesos están sujetos a $\sum_{i=1}^{k} u_i = 1$.

$$min \sum_{i=1}^{k} u_i f_i(\vec{x}) \tag{3.2}$$

Este método es el enfoque más simple encontrado para resolver MOOP, dado a que es intuitivo y fácil de implementar; además, en problemas con Frentes de Pareto convexos, la SPF garantiza encontrar soluciones en todo el conjunto óptimo de Pareto [31].

Sin embargo, su principal desventaja radica en su ineficiencia al encontrar soluciones en frentes de Pareto no convexos o en funciones discontinuas, características que suelen ser frecuentes en el modelado de problemas del mundo real. Adicionalmente, requiere una buena calibración de pesos para explorar los puntos del Frente de Pareto que son de interés para el tomador de decisiones.

3.2.2. Método ϵ-Constraint

Este método fue introducido por Haimes et al. en 1971 [102], el cuál mejora las deficiencias mostradas por la SPF; es decir, encuentra soluciones óptimas no importando si el frente de Pareto es cóncavo o convexo.

La idea detrás de ϵ-Constraint es minimizar uno de los objetivos del problema, mientras que los objetivos restantes son transformados en funciones de restricción limitadas por diferentes valores definidos por el usuario (ϵ). La Ecuación 3.3 expresa matemáticamente el funcionamiento de este método.

$$
\begin{aligned}
&\text{minimizar } f_1(\vec{x}) \\
&\text{Sujeto a:} \\
&\quad f_2(\vec{x}) \geq \epsilon_2, \\
&\quad f_3(\vec{x}) \geq \epsilon_3, \\
&\quad \vdots \\
&\quad f_F(\vec{x}) \geq \epsilon_F, \\
&\quad g_n(\vec{x}) \geq 0, \\
&\quad h_n(\vec{x}) = 0, \\
&\quad x \in S
\end{aligned}
\tag{3.3}
$$

Este método es capaz de encontrar diferentes soluciones en el Frente de Pareto dependiendo de los valores asignados a cada ϵ, permitiendo ser robusto a la forma que el mismo frente tenga, es decir, si es cóncavo o convexo. Por el contrario, la elección de los valores óptimos para cada ϵ representa la principal desventaja de este enfoque, siendo visto incluso como otro problema de optimización dentro de los MOOP.

3.2.3. Método de Programación por Metas

El método de programación por metas fue propuesto originalmente para resolver problemas de optimización lineales mono-objetivos por Charnes et al. en 1955, posteriormente, autores como Lee, Ignizio y Romero han demostrado su funcionalidad en aplicaciones de ingeniería [31].

Este método consiste en definir metas τ_j para cada función que se requiera optimizar $F_j(x)$. La idea es minimizar las desviaciones de cada meta hacia las funciones objetivo; esto es, minimizar $\sum_{i=1}^{j} \delta_i$, donde δ_i es la desviación de la meta τ_i para la i-ésima función objetivo [70]. Dicha desviación es separada en dos valores, uno positivo δ_i^+ y otro negativo δ_i^-, con el fin de modelar el bajo y alto rendimiento de cada meta. En resumen, este método, a diferencia del método ϵ-Constraint, transforma un problema multi-objetivo a uno de un sólo objetivo definiendo una función basada en desviaciones de los objetivos del problema a metas definidas para

cada una de ellos, y los objetivos son convertidos en restricciones del problema. La Ecuación 3.4 expresa dicha transformación.

$$\begin{aligned} &\text{minimizar } \sum_{i=1}^{j} u_i\delta_i^+ + u_i\delta_i^- \\ &\text{Sujeto a:} \\ &\qquad f_j(\vec{x}) + \delta_j^+ - \delta_j^- = \tau_j \\ &\qquad p_k, n_k \geq 0 \end{aligned} \tag{3.4}$$

Donde $\vec{u} = \{u_1, u_2, \ldots, u_i\}$ son pesos para controlar hacia donde debe dirigirse la búsqueda en la minimización de los objetivos, teniendo un comportamiento similar a la SPF.

La programación por metas es un método muy recurrido y popular debido a que su idea es intuitiva, comprensible y fácil para el tomador de decisiones [18]. Adicionalmente, este método es capaz de lidiar con problemas de baja y alta dimensionalidad, es decir, gran número de variables de decisión [18]. Sin embargo, este método, al igual que la SPF, el tomador de decisión requiere tener información suficiente para definir los mejores posibles valores para cada $\vec{u}$.

3.2.4. Método Lexicográfico

Una variante al método de programación por metas, es el llamado *programación por metas lexicográfico*. La diferencia radica en que la meta está dada por la mejor aptitud encontrada en la función predecesora. Para ello es necesario ordenar las funciones en términos de importancia, lo que significa que se necesita tener conocimiento a priori de las funciones para definir cuál debe ejecutarse primero. Teniendo ordenadas el conjunto de funciones, se van ejecutando una a una y el resultado de la anterior se convierte ahora en la meta de la función actual, incluyéndola como restricción de igualdad.

De lo anterior se aprecia que es un método simple, pero requiere de un buen manejo de restricciones de igualdad, cuestión que no es simple en la optimización de cualquier tipo de problema.

3.2.5. Método de Obtención de Metas

El último de los métodos presentados en esta sección es *Obtención de metas*. Este método resuelve un MOOP convirtiéndolo en un problema de programación no lineal, tal cual se expresa en la Ecuación 3.5.

$$\begin{aligned} &Minimize\ \lambda \\ &Sujeto\ a: \\ &f_i(\vec{x}) - u_i\lambda \leq \tau_i, i = 1, 2, ..., k \end{aligned} \tag{3.5}$$

Donde u_i, son pesos que indican la relevancia de cada función objetivo, τ_i son cada una de las metas definidas para cada función objetivo y λ es un valor escalar no restringido de signo. La minimización de λ conduce al hallazgo de una solución no dominada que alcanza o supera los objetivos especificados dependiendo el coeficiente $u_i\lambda$ [43].

3.3. Algoritmos evolutivos para optimización multi-objetivo

Cada uno de los métodos presentados en la Sección 3.2 representan una transformación del modelo multi-objetivo a un modelo mono-objetivo, con la finalidad de implementar los métodos tradicionales de optimización. Sin embargo, este tipo de heurísticas mono-objetivo, tienen algunas limitaciones como por ejemplo: suelen tener dificultades en funciones no derivables en segundo grado, la forma del Frente de Pareto, entre otros. Estas desventajas han motivado el surgimiento de meta-heurísticas capaces de encontrar soluciones adecuadas a cualquier tipo de MOOP. Dichos enfoques son descritos de acuerdo a la categorización mostrada en la Figura 3.3.

3.3.1. Basados en Pareto

Dentro de este rubro se encuentran aquellos métodos que basan su funcionamiento en la optimalidad de Pareto. Algunos métodos encontrados en este rubro son:

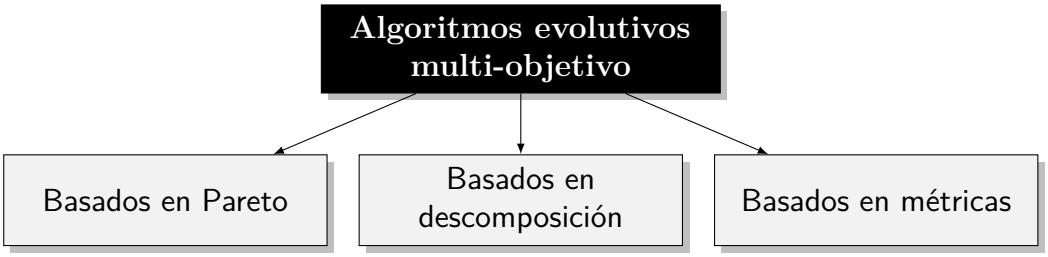


Figura 3.3: Breve clasificación de los algoritmos evolutivos multi-objetivo existentes en la literatura

Multiple Objective Genetic Algorithm (MOGA), *Non-dominated Sorting Genetic Algorithm (NSGA)*, *Strength Pareto Evolutionary Algorithm 2 (SPEA-2)*, y *Multi-Objective Particle Swarm Optimization (MOPSO).*

Multiple Objective Genetic Algorithm (MOGA)

Multiple Objective Genetic Algorithm (MOGA) es uno de los primeros métodos en utilizar la asignación de aptitud basada en la optimalidad de Pareto propuesta por Goldberg [52]. MOGA fue propuesto por Fonseca y Fleming en 1993 [43]. Ellos propusieron una jerarquización modificada a la propuesta por Goldberg, donde las soluciones no dominadas son encontradas en primer lugar, y las soluciones dominadas son penalizadas con una jerarquía calculada de acuerdo a la densidad de la población localizada en la generación g más uno, $r_{\vec{x}}^{(g)} = \gamma(\vec{x}, g) + 1$, donde $\gamma(\vec{x}, g)$ representa la densidad medida como el número de individuos que dominan a $\vec{x}$. Para garantizar una buena distribución de las soluciones a lo largo de todo el Frente de Pareto, MOGA utiliza operadores de diversidad como *crowding* y *fitness sharing*, propuestos por Holland en 1975 y Goldberg et al. en 1987, respectivamente. El mecanismo de diversidad *fitness sharing* utilizado por MOGA calcula su valor en función de los valores máximos y mínimos actuales de los objetivos y el tamaño de la población.

Non-dominated Sorting Genetic Algorithm (NSGA)

Al igual que MOGA, el algoritmo evolutivo llamado *Non-dominated Sorting Genetic Algorithm (NSGA)* utiliza el esquema de jerarquización propuesto por Goldberg. Este método fue propuesto por Srinivas y Deb en 1994 [95]. La idea principal de este método es subdividir a la población en capas de acuerdo a la

no dominancia de Pareto; es decir, en la primera capa o frente se encuentran las soluciones no dominadas de la población, en el segundo frente, se agrupan las soluciones no dominadas de la población sin el primer frente, y así sucesivamente se van clasificando todas las soluciones. Las soluciones encontradas en el primer frente son las que tienen una mayor probabilidad de supervivencia. Una versión mejorada de este algoritmo, llamada *Non-dominated Sorting Genetic Algorithm II (NSGA-II)*, fue propuesta por Deb et al. en 2002 [30] y será discutida a detalle en la Sección 3.4 debido a que es el algoritmo usado en este estudio como guía de búsqueda de esquemas de discretización.

Strength Pareto Evolutionary Algorithm 2 (SPEA2)

Strength Pareto Evolutionary Algorithm 2 (SPEA2) es una versión mejorada del algoritmo llamado *Strength Pareto Evolutionary Algorithm (SPEA)* propuesto por Zitzler et al. [108]. SPEA utiliza un archivo externo donde se almacenan las soluciones no dominadas encontradas durante su ejecución. Dichas soluciones, en cada generación, son eliminadas de la población. Este archivo se utiliza para calcular la aptitud de cada individuo en la población.

SPEA2 fue propuesto por Zitzler en [109]. SPEA2, a diferencia de SPEA, utiliza una forma de asignación de aptitud para un individuo de acuerdo al número de individuos dominados y los que éste domina. Adicionalmente, incorpora una método para truncar el tamaño del archivo externo que garantiza la preservación del limite de las soluciones.

Multi-Objective Particle Swarm Optimization (MOPSO)

La idea de utilizar el algoritmo Particle Swarm Optimization (PSO) para la solución de MOOPs fue introducida por Moore et al., en [75], el cual fue llamado *Multi-Objective Particle Swarm Optimization (MOPSO)*. La adaptación consistió en utilizar un archivo externo para almacenar los líderes del cúmulo de partículas, que son las soluciones no dominadas en el vuelo l. Por cada partícula del cúmulo, se selecciona un líder del archivo externo para actualizar su vuelo. Posteriormente, el archivo externo es actualizado con las soluciones existentes y las nuevas no

dominadas, manteniendo sólo los elementos no dominados del mismo. Al final, el archivo externo es reportado como el Frente de Pareto encontrado por MOPSO.

3.3.2. Basados en Descomposición

Un enfoque que ha demostrado ser una buena estrategia para resolver problemas complejos o de múltiple dimensiones, es el llamado *divide y vencerás*. Básicamente, este enfoque consiste en subdividir un problema difícil en varios más pequeños y fáciles de resolver. Esta es la filosofía que siguen los algoritmos basados en descomposición. En esta apartado se describe uno de los algoritmos de este tipo más recurrido en la literatura, el llamado *Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)*.

Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)

Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) fue propuesto por Zhang et al., en 2007 [107]. MOEA/D subdivide un MOOP en múltiples SOOPs vistos como funciones de agregación. Todos los SOOPs son optimizados simultáneamente evolucionando una población de soluciones.

En este proceso de evolución, es necesario definir el concepto de vecindad entre subproblemas, el cuál se calcula mediante la distancia entre sus vectores de peso de agregación; es decir, se dice que el subproblema i es vecino del subproblema j si sus pesos de agregación son similares a los pesos del subproblema j. Este vecindario es utilizado para optimizar cada subproblema, en otras palabras, un subproblema se optimiza utilizando la información de los subproblemas vecinos.

MOEA/D, al igual que MOPSO o SPEA, utiliza un archivo externo donde se almacena la solución no dominada encontrada hasta el momento t. Posteriormente y mediante operadores genéticos, se generan nuevas soluciones y si alguna es mejor que la encontrada en el archivo, ésta es reemplazada por la nueva solución no dominada. Este algoritmo mantiene un bajo costo computacional, y conserva la diversidad de la población debido a la buena distribución de los vectores de peso que, además, sirven como guías del proceso de la búsqueda.

3.3.3. Basados en Métricas

En este rubro se agrupan los algoritmos que utilizan alguna métrica para medir el desempeño de los algoritmos como medida de selección. Particularmente, en este rubro se describirá el conocido algoritmo llamado *Multi-Objective Selection based on Dominated Hypervolume (SMS-EMOA)* debido a su fácil implementación, idea intuitiva y resultados competitivos.

Multi-Objective Selection based on Dominated Hypervolume (SMS-EMOA)

Multi-Objective Selection based on Dominated Hypervolume (SMS-EMOA) fue propuesto por Beume et al., en [12]. SMS-EMOA es un algoritmo intuitivo y de rápida implementación. Básicamente, SMS-EMOA genera un hijo mediante operadores de variación aleatorios, el cuál es incluido en la población de la generación actual. Posteriormente, la población es subdivida en frentes de Pareto, al igual que NSGA, para eliminar la solución del último frente con la menor medida de hipervolumen; de esta manera, se mantiene uniforme el tamaño de la población inicial. La medida de hipervolumen es un indicador de calidad que premia la convergencia hacia el frente de Pareto, así como la distribución representativa de puntos a lo largo del frente [12]. Ésto lo realiza obteniendo el área formada por cada solución del frente de Pareto y un punto de referencia, el cuál debe ser dominado por todas las soluciones de dicho frente.

Una ventaja de SMS-EMOA es que encuentra frentes de Pareto bien distribuidos con poblaciones pequeñas, lo que conlleva a una complejidad computacional menor.

3.4. Non-dominated Sorting Genetic Algorithm II (NSGA-II)

El algoritmo *Non-dominated Sorting Genetic Algorithm II (NSGA-II)* fue propuesto por Deb et al., en [30]. NSGA-II se distingue por ser un algoritmo evolutivo multi-objetivo y elitista con un procedimiento de ordenamiento por dominancia (NS) y un operador de nichos sin parámetros [30]. El operador de nichos es una

técnica para mantener la diversidad de las soluciones del frente de Pareto [110]. El Algoritmo 6 ilustra los pasos generales de NSGA-II.

Algoritmo 6 Algoritmo de NSGA-II

Entrada: GN: Número de generaciones, SP: Tamaño de la población

1: Inicializa una población aleatoria P de tamaño SP.
2: Evalúa la población P en cada función objetivo.
3: $gn = 1$
4: **mientras** $gn \leq GN$ **hacer**
5: Ordena la población P de acuerdo al algoritmo de ordenación rápida de dominancia 7 y se almacena en $\mathcal{PF}$.
6: Calcula la distancia de amontonamiento de cada elemento en $\mathcal{PF}$ basado en el algoritmo 8.
7: Escoge un conjunto de individuos (padres) R de P mediante torneos estocásticos.
8: Genera O descendientes de R aplicando un operador de cruza.
9: Mutar O mediante operadores de mutación.
10: Evaluar O en cada objetivo.
11: Unir en J la población actual P y la descendencia O.
12: Calcular un nuevo frente $\mathcal{PF}$, como en el paso 5, pero usando el conjunto J.
13: Repetir el paso 6.
14: Reemplazar la población P con $\mathcal{PF}$.

El procedimiento NS fue incluido en el algoritmo NSGA, tal cual se describió en la Sección 3.3.1. Dicho procedimiento consiste en subdividir el conjunto de soluciones potenciales en κ capas de acuerdo al proceso de dominancia de Pareto. Como consecuencia, en la primera capa, se encuentran las soluciones no dominadas de la población actual ($\mathcal{PF}_1$), en la segunda capa se encuentran las soluciones no dominadas del resto de la población ($\mathcal{PF}_2$) y así sucesivamente. El Algoritmo 7 muestra este proceso.

Una característica importante de NSGA-II es el manejo de diversidad entre los miembros de la población. Esta propiedad se logra mediante un operador de comparación de amontonamiento $\prec_n$, el cual prefiere una solución del resto de acuerdo a su rango y a la suma de las distancias de los puntos vecinos al punto actual en cada uno de los objetivos del problema. El Algoritmo 8 describe la forma de favorecer la diversidad en NSGA-II.

Algoritmo 7 Algoritmo de ordenamiento rápido de dominancia

Entrada: P: Población

1: $N = size(P)$
2: **para** $p = 1$ to $N - 1$ **hacer**
3: $\quad S_{P_p} = \emptyset$
4: $\quad n_{P_p} = 0$
5: $\quad$ **para** $q = p + 1$ hasta N **hacer**
6: $\quad\quad$ **si** $P_p \prec P_q$ **entonces**
7: $\quad\quad\quad S_{P_p} = S_{P_p} \cup \{P_q\}$
8: $\quad\quad$ **si no si** $P_q \prec P_p$ **entonces**
9: $\quad\quad\quad n_{P_p} = n_{P_p} + 1$
10: $\quad$ **si** $n_{P_p} = 0$ **entonces**
11: $\quad\quad P_{p_{rank}} = 1$
12: $\quad\quad \mathcal{F}_1 = \mathcal{F}_1 \cup P_p$
13: $i = 1$
14: **mientras** $\mathcal{F}_i \neq \emptyset$ **hacer**
15: $\quad Q = \emptyset$
16: $\quad$ **para** cada $f \in \mathcal{F}_i$ **hacer**
17: $\quad\quad$ **para** cada $s \in S_f$ **hacer**
18: $\quad\quad\quad n_s = n_s - 1$
19: $\quad\quad\quad$ **si** $n_s = 0$ **entonces**
20: $\quad\quad\quad\quad s_{rank} = i + 1$
21: $\quad\quad\quad\quad Q = Q \cup s$
22: $\quad i = i + 1$
23: $\quad \mathcal{F}_i = Q$

Algoritmo 8 Procedimiento para asignar el valor de la distancia de amontonamiento

Entrada: F: Conjunto de soluciones, num_objetivos: Número de funciones objetivo a optimizar

1: $n = |F|$;
2: **para** $i = 0$ hasta $n - 1$ **hacer**
3: $\quad F[i].distancia = 0$
4: **para** $m = 1$ hasta $num_objetivos$ **hacer**
5: $\quad F = sort(F, m)$ $\quad\triangleright$ Ordena de acuerdo a los objetivos
6: $\quad F[0].distancia = \infty$
7: $\quad F[n-1].distancia = \infty$
8: $\quad$ **para** $i = 1$ to $n - 2$ **hacer**
9: $\quad\quad F[i].distancia = F[i].distancia + \frac{F[i+1].m - F[i-1].m}{f_m^{max} - F_m^{min}}$

La selección de padres se realiza mediante torneos estocásticos. Sin embargo, el criterio para decidir un ganador está dado por el operador $\prec_n$.

Por último, NSGA-II genera una nueva población seleccionando los individuos del primer Frente de Pareto encontrado por el procedimiento NS; en caso de que la nueva población sea de menor tamaño a la original, se toman los siguientes individuos de los subsecuentes frentes. Las soluciones usadas para completar la población, serán aquellas con el menor valor obtenido mediante el operador $\prec_n$.

Como se mencionó en párrafos anteriores, NSGA-II es el algoritmo evolutivo multi-objetivo seleccionado como herramienta de búsqueda de esquemas de discretización simbólicas. Sin embargo, para cumplir dicha función es necesario adaptar los elementos descritos en este capítulo para tratar con los problemas derivados de nuestra propuesta. Es por ello, que en el siguiente capítulo se describirán tanto la propuesta como las adaptaciones realizadas a NSGA-II para encontrar esquemas de discretización simbólicas adecuadas en términos de clasificación, reducción de la dimensionalidad y pérdida de información.

4
Propuesta

Contenido

4.1. Introducción 61
4.2. Esquemas de Discretización con Múltiples Alfabetos . 63
4.3. Algoritmo Evolutivo Multi-objetivo 65
4.3.1. Codificación de Esquemas de Discretización con Múltiples Alfabetos 65
4.3.2. Funciones de Evaluación 66
4.3.3. Operador de Cruza Adaptado 68
4.4. Selección de Preferencias 69
4.5. Evaluación del Esquema de Discretización Final 70

4.1. Introducción

En este capítulo se describen los elementos de nuestra propuesta, surgida para mejorar las desventajas de los métodos analizados en la Sección 1.7. Básicamente, nuestra propuesta se centra en proporcionar mayor libertad de búsqueda de esquemas de discretización capaces de reducir considerablemente la dimensionalidad de las bases temporales y con tasas de clasificación buenas. Dicho incremento en el grado de libertad de búsqueda, se realiza mediante la definición de un alfabeto por cada segmento de palabra, es decir, en un esquema de discretización. Para ello se tendrá

un esquema de discretización flexible, donde por cada segmento de palabra se tendrán diferentes cortes de alfabeto.

Esta propuesta no sólo apunta a incrementar la capacidad de búsqueda de esquemas de discretización, sino que también trata de minimizar la pérdida de información presente cuando se reduce la dimensionalidad de la base temporal. Esta desventaja ha sido fuertemente criticada por los expertos en análisis de series temporales. Para lograrlo, se incluye, como parte del conjunto de funciones objetivo, una métrica para estimar la pérdida de información en la que incurre un esquema de discretización, cuestión que no ha sido abordada por ningún método de discretización de series de tiempo en la literatura especializada.

Nuestro método llamado **eMODiTS** (**e**nhanced **M**ulti-objective symb**O**lic **Di**scretization for **T**ime **S**eries), está basado en el método presentado en [71], llamado *MODiTS* (**M**ulti-objective symb**O**lic **Di**scretization for **T**ime **S**eries). MODiTS utiliza un algoritmo evolutivo multi-objetivo para encontrar el número de segmentos de palabras adecuados y un conjunto de cortes de alfabeto acorde a la serie temporal. Las funciones objetivos usadas fueron la estimación de la tasa de clasificación mediante el uso de Entropía, una métrica para estimar la complejidad del modelo, y la tasa de compresión lograda por el esquema de discretización. El algoritmo evolutivo empleado para realizar la búsqueda, fue NSGA-II, debido a los resultados satisfactorios encontrados en las múltiples aplicaciones en que ha sido probado en la literatura especializada [55, 73, 83, 7, 19], siendo considerado como un algoritmo fuertemente competitivo para optimizar problemas con múltiples objetivos. Sin embargo, una de las críticas a MODiTS es la pérdida de información presente en los esquemas de discretización encontrados al final de la ejecución del algoritmo. Es por ello, que en eMODiTS se incluye un objetivo relacionado con esta característica como una de las guías de la búsqueda.

La Figura 4.1 muestra los principales elementos de nuestra propuesta y cómo están conectados unos con otros. Dichas etapas se detallan en las siguientes subsecciones.

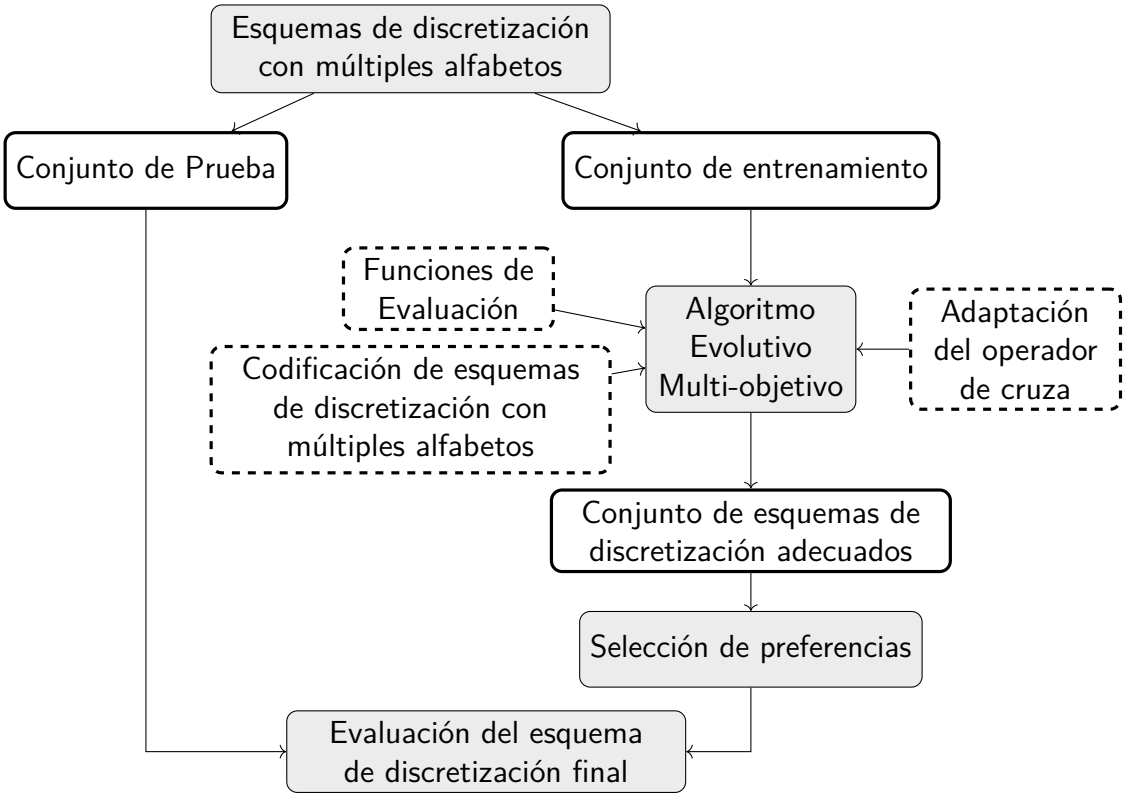


Figura 4.1: Metodología general de eMODiTS. Rectángulos con líneas discontinuas representan las modificaciones realizadas al algoritmo evolutivo multi-objetivo para lidiar con esquemas de discretización con múltiples alfabetos.

4.2. Esquemas de Discretización con Múltiples Alfabetos

En la Sección 1.7 se presentaron un conjunto de enfoques de discretización simbólica de series de tiempo surgidos para mejorar las limitaciones de SAX: el número óptimo de segmentos de palabra y de cortes de alfabetos. La mayoría de ellos asumen un único esquema de alfabetos para todos los segmentos de palabra, lo que limita encontrar un esquema idóneo para tareas de clasificación. El esquema de discretización usado en eMODiTS y propuesto en [71], cambia completamente la estructura que hasta el momento se ha descrito en la literatura especializada, dado que se definen conjuntos de alfabetos acordes a cada segmento de palabra.

Definición 4.2.1. Múltiples conjuntos de alfabetos. *Sea* $W = \{w_1, w_2, \cdots, w_i\}$ *un conjunto de segmentos de palabra, el conjunto de cortes de alfabeto para cada* w_i *es definido como:* $\vec{a}_{w_i} = \{a^{(1)}_{w_i}, a^{(2)}_{w_i}, \cdots, a^{(j)}_{w_i}\}$.

El tamaño de cada esquema de alfabetos varía entre cada segmento de palabra, es decir, el número de cortes en un alfabeto puede ser similar o distinto al resto de los alfabetos en el mismo esquema de discretización. La Figura 4.2 muestra gráficamente un ejemplo del esquema con múltiples alfabetos propuesto, donde se aprecia como la malla bidimensional es transformada por una malla más flexible que se adapta al segmento de serie de tiempo dentro de cada palabra.

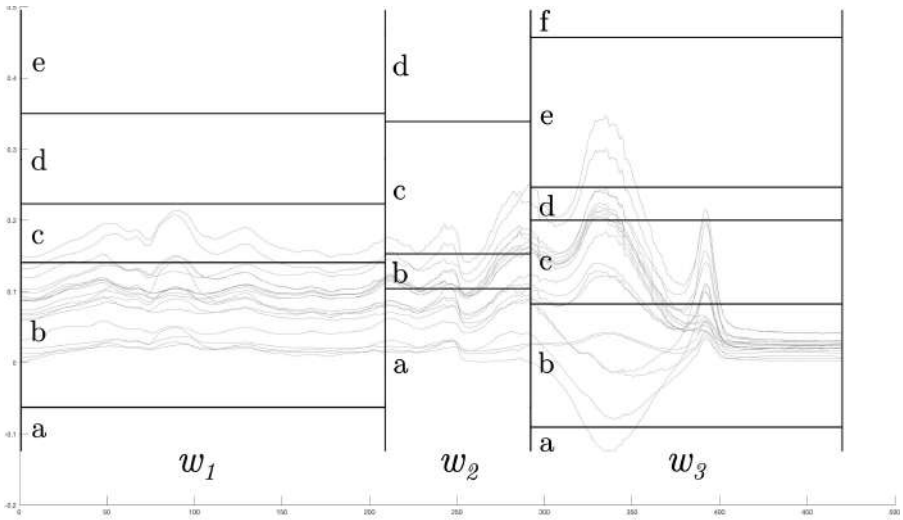


Figura 4.2: Ejemplo de un esquema de discretización generado por eMODiTS para la base temporal llamada Beef obtenida de [27].

En nuestra propuesta, eMODiTS utiliza el método de PAA para reducir la dimensionalidad de la serie temporal. Sin embargo, dado que PAA fue desarrollado para funcionar con cortes equidistantes de segmentos de palabra, se realizó una adaptación del mismo para obtener la reducción bajo cortes no equidistantes.

Definición 4.2.2. Coeficiente de PAA adaptado. *Sea* $TS = \{ts_1, ts_2, \cdots, ts_n\}$ *una serie de tiempo, y* $W = \{w_1, w_2, \cdots, w_m\}$ *el conjunto de segmentos de palabra, un coeficiente calculado con el método de PAA* $\overline{ts_i}$ *es obtenido como sigue:*

$$\overline{ts_i} = \frac{1}{(w_{i+1} - w_i)} \sum_{j=w_i+1}^{w_{i+1}} ts_j$$

Cada coeficiente obtenido por el método PAA es transformado en símbolos o caracteres para obtener la serie de tiempo simbólica completa. Dado que eMODiTS

propone un esquema con múltiples alfabetos, cada coeficiente es transformado de acuerdo a su respectivo esquema de alfabetos.

Definición 4.2.3. Serie de tiempo simbólica. *Un serie de tiempo simbólica es una serie de tiempo discreta construida a partir de símbolos o caracteres. Sea η una función para transformar valores discretos a caracteres, una serie de tiempo simbólica $D = \{d_1, d_2, \cdots, d_w\}$ es obtenida mediante:*

$$d_i = \eta(k), \quad a_{w_i}^{(k-1)} \geq \overline{ts_i} \leq a_{w_i}^{(k)}$$

donde k denota el número de corte de alfabeto.

4.3. Algoritmo Evolutivo Multi-objetivo

Como se ha mencionado anteriormente, nuestra propuesta encuentra esquemas adecuados de discretización basado en tres criterios: clasificación estimada mediante entropía, complejidad del modelo y pérdida de información. Generalmente, los algoritmos tradicionales de toma de decisión multi-criterio suelen atrapar óptimos locales limitando la búsqueda a pequeñas regiones donde es posible que no se encuentren buenas soluciones. Como consecuencia, eMODiTS emplea un algoritmo de búsqueda global multi-objetivo como herramienta de búsqueda de esquemas competitivos de discretización, llamado *Non-dominated sorting genetic algorithm II (NSGA-II)*. NSGA-II fue adoptado debido a ser uno de los algoritmos evolutivos multi-objetivo más competitivo en la literatura y por su simplicidad e intuitividad para su implementación. En los siguientes apartados se presentan las modificaciones realizadas al algoritmo canónico de NSGA-II para la búsqueda de esquemas de discretización con múltiples alfabetos.

4.3.1. Codificación de Esquemas de Discretización con Múltiples Alfabetos

En eMODiTS, una solución representa un esquema de discretización con múltiples alfabetos codificado como un vector de números reales. La estructura del vector es definida posicionando el primer corte del conjunto de segmentos de palabra

seguido de su correspondiente alfabeto, posteriormente se inserta el siguiente corte de palabra con su alfabeto, y así sucesivamente hasta tener el vector completo con todos los cortes de ambos conjuntos. La Figura 4.3 muestra dicha estructura donde w_j es el $j-$ésimo corte de palabra y $\vec{a}^{i}_{w_j}$ es el $i-$ésimo corte del alfabeto para el segmento de palabra w_j.

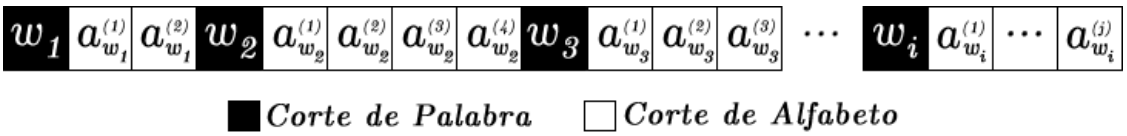


Figura 4.3: Codificación de un esquema de discretización donde cada segmento de palabra (w_j) es incluido seguido de su correspondiente esquema de alfabetos (a_{w_j}).

Es importante resaltar que, como consecuencia de la estructura de una solución en eMODiTS, cada individuo de la población puede tener diferentes longitudes, tanto en el número de segmentos como el número de cortes de cada alfabeto.

4.3.2. Funciones de Evaluación

En la literatura especializada existen diversas funciones objetivo utilizadas para guiar a un algoritmo de discretización hacia esquemas competitivos en términos de clasificación, tales como: criterio de la información [33], estado de persistencia [76], maximización de la entropía de la información (IEM, en sus siglas en inglés) [40, 60], ganancia de la información, longitud de descripción mínima (MDL) [39, 61], entre otros. eMODiTS adopta dos de las funciones propuestas por el método de discretización EP para estimar la precisión en clasificación mediante entropía y la complejidad del modelo. A parte de estas funciones, eMODiTS usa el Error Cuadrático Medio (MSE, en sus siglas en inglés) para estimar la pérdida de la información entre la serie original y la serie simbólica reconstruida [93]. Antes de calcular el valor de cada función objetivo es necesario discretizar las series de tiempo, tal cuál fue descrito en la Sección 4.2. Cada una de las funciones mencionadas son descritas a continuación.

Tasa de Clasificación Estimada Mediante Entropía

La precisión en clasificación es estimada mediante la creación de una matriz de confusión (CM) donde se almacenan cada serie de tiempo simbólica no repetida junto con las diferentes clases con las que originalmente aparecen en la base de datos temporal. La métrica propone estimar si la base de datos discreta de tamaño D' es caótica, es decir, si una serie de tiempo simbólica contiene más de una clase asignada. La Ecuación 4.1 expresa matemáticamente el cálculo de la Entropía, donde $CM(P_{i,j})$ es la matriz de confusión, mencionada líneas arriba. $CM(P_{i,j})$ contiene las probabilidades de que una serie de tiempo D pertenezca a una clase determinada c. Para esta métrica, las soluciones con menor valor son preferidas con respecto al resto de la población.

$$\text{Entropía} = \sum_{i=1}^{D'} - \sum_{c=1}^{C} CM(P_{D^{(i)},c}) * \log_2 CM(P_{D^{(i)},c}) \tag{4.1}$$

Complejidad del Modelo

Esta métrica consiste en medir la complejidad de la discretización resultante de cada esquema o individuo. Para su cálculo, es necesario contar el número de series de tiempo simbólicas distintas generadas por una solución, tal cual se expresa en la Ecuación 4.2, donde N es número total de series de tiempo en la base de datos original, C es el número de etiquetas de clase y D' es la cantidad de series de tiempo simbólicas. Al igual que la función descrita en el Apartado 4.3.2, es preferible tener soluciones con el menor valor en esta métrica.

$$Complejidad = (D' - C)/(N + C) \tag{4.2}$$

Pérdida de Información

Una forma de estimar la pérdida de la información propuesta en [93] es utilizar la bien conocida función llamada Error Cuadrático Medio (MSE), la cual es usada para estimar que tan separados se encuentran dos conjuntos de datos. Para aplicarlo a nuestro problema, es necesario reconstruir la serie de tiempo simbólica sustituyendo

el dato original con su correspondiente símbolo. MSE calcula la separación entre la serie de tiempo original ($\widetilde{TS}$) y la reconstruida ($\widetilde{D}$) como una forma de estimar la pérdida de la forma de la serie de tiempo original. La Ecuación 4.3 expresa esta función, donde $\widetilde{D'}$ es el número de series de tiempo reconstruidas en la base de datos temporal discreta, $\widetilde{d_i}$ es el valor de la serie de tiempo reconstruida en la posición i, y ts_j es el valor de la series temporal en el tiempo j. Las series de tiempo original y reconstruida fueron escaladas a [0,1] para que la comparación entre ambas sea justa. Las soluciones con valores bajos en esta función representan las mejores soluciones al problema.

$$\begin{aligned} \text{Pérdida de Información} &= \frac{1}{\widetilde{D'}} \sum_{i=1}^{\widetilde{D'}} MSE(\widetilde{D_i}, TS_i) \\ MSE(\widetilde{D}, TS) &= \frac{\sum(\widetilde{d_i} - ts_j)^2}{n-1}, \widetilde{d_i} \in \widetilde{D}, ts_j \in TS \end{aligned} \tag{4.3}$$

4.3.3. Operador de Cruza Adaptado

Como se mencionó en la Sección 4.2, en eMODiTS el tamaño de cada solución en una misma población varía entre ellas, por lo tanto, no pueden ser aplicados los operadores de cruza tradicionales propuestos en la literatura especializada. Para generar descendientes en eMODiTS, se propone una adaptación del operador de cruza más simple desarrollado para algoritmos genéticos, el *operador de cruza de un solo punto*. La modificación a dicho operador consistió en generar aleatoriamente puntos de corte en cada uno de los padres; posteriormente, los primeros elementos hasta el punto de corte del primer padre son copiados en las primeras posiciones del primer hijo, y los elementos restantes son copiados en la última parte del segundo; de la misma forma, los primeros elementos del segundo padre antes de su corte son copiados a las primeras posiciones del segundo hijo y los elementos restantes son copiados a la última parte del primer hijo. Este proceso se muestra en la Figura 4.4, donde se puede apreciar que no sólo se copian los segmentos de palabra sino que se copian dichos segmentos junto con sus respectivos alfabetos para mantener la factibilidad de los esquemas de discretización.

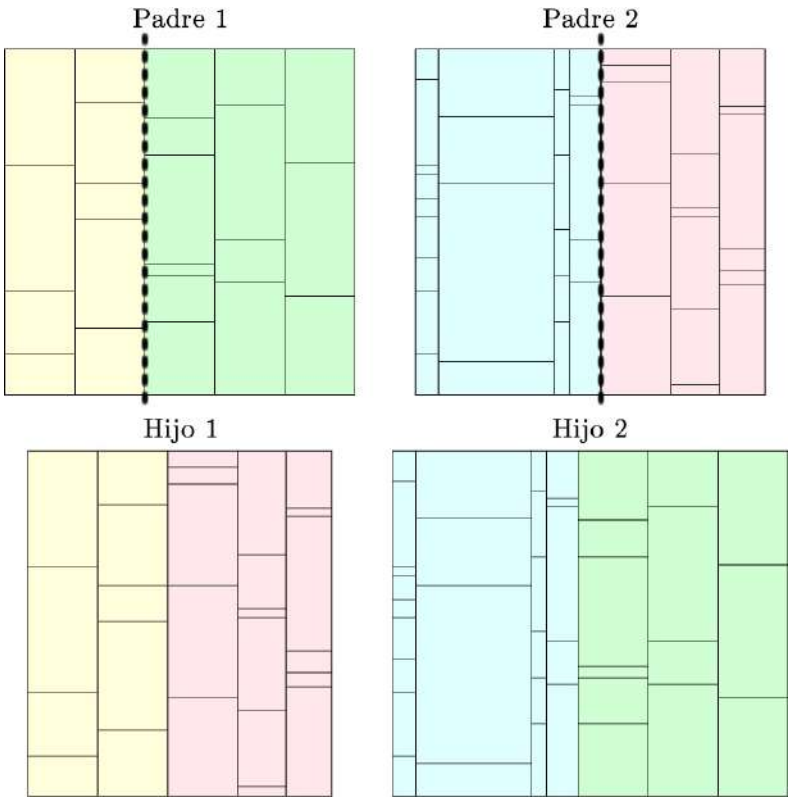


Figura 4.4: Operador de cruza implementado en eMODiTS y basado en el operador de cruza de un sólo punto. Las posiciones de los cortes en cada padre son representados por una línea punteada.

4.4. Selección de Preferencias

Dado que eMODiTS encuentra un conjunto de posibles soluciones en lugar de una, se propusieron cuatro métodos para seleccionar la solución final de dicho conjunto, conocido como Frente de Pareto ($\mathcal{PF}$), descrito en la Sección 3.1.2. Para realizar dicha selección, cada $\mathcal{PF}$ encontrado en cada ejecución del algoritmo fue agrupado en un solo conjunto, llamado $\mathcal{PF}$ *acumulado*, donde sólo se conservan las soluciones no dominadas de este último. Los métodos propuestos y utilizados en esta propuesta son descritos a continuación.

1. **Knee Method (Método de la rodilla)**. En este método se selecciona el punto más cerca al origen (0,0,0). Ver Figura 4.5a.

2. **CV Method (Método CV)**. Este método selecciona la solución con la menor tasa de error en clasificación obtenida a través del método de validación

cruzada usando sólo el conjunto de entrenamiento. Ver Figura 4.5b.

3. **KM1 Method (Método KM1)**. Para escoger una solución del $\mathcal{PF}$ acumulado se utiliza el algoritmo k-medias con $k = 1$, seleccionando aquel punto que se localice más cerca al punto medio del grupo generado. Ver Figura 4.5c.

4. **KM20 Method (Método KM20)**. Este método es una combinación de los métodos CV y KM1. El valor para k en el algoritmo k-medias fue el 20 % del tamaño del $\mathcal{PF}$ acumulado. La selección de la solución final fue realizada tomando aquella con el menor valor de la tasa de error en clasificación de las soluciones más cercanas a los centroides de cada grupo generado por el algoritmo k-medias. Al igual que el método CV, el cálculo de la tasa de error en clasificación se obtuvo usando sólo el conjunto de entrenamiento extraído de la base temporal original. Ver Figura 4.5d.

4.5. Evaluación del Esquema de Discretización Final

El último paso de nuestra propuesta es la evaluación del esquema de discretización final. Dicha evaluación fue realizada usando un nuevo conjunto de datos, llamado conjunto de prueba, y el algoritmo de clasificación mediante árboles de decisión, descrito en la Sección 2.4.3. Dicho clasificador fue seleccionado debido a la representación gráfica en forma de árbol que genera, la cual es usada para analizar y entender las relaciones o patrones encontrados en las bases de datos temporales, relevantes para el tomador de decisión, y que no son fáciles de identificar con los métodos tradicionales.

En el siguiente capítulo se presentarán los resultados y las discusiones logradas al implementar la propuesta descrita en este capítulo. Adicionalmente, se detallará el diseño experimental realizado con la finalidad de medir el desempeño de nuestro algoritmo en la discretización de series temporales.

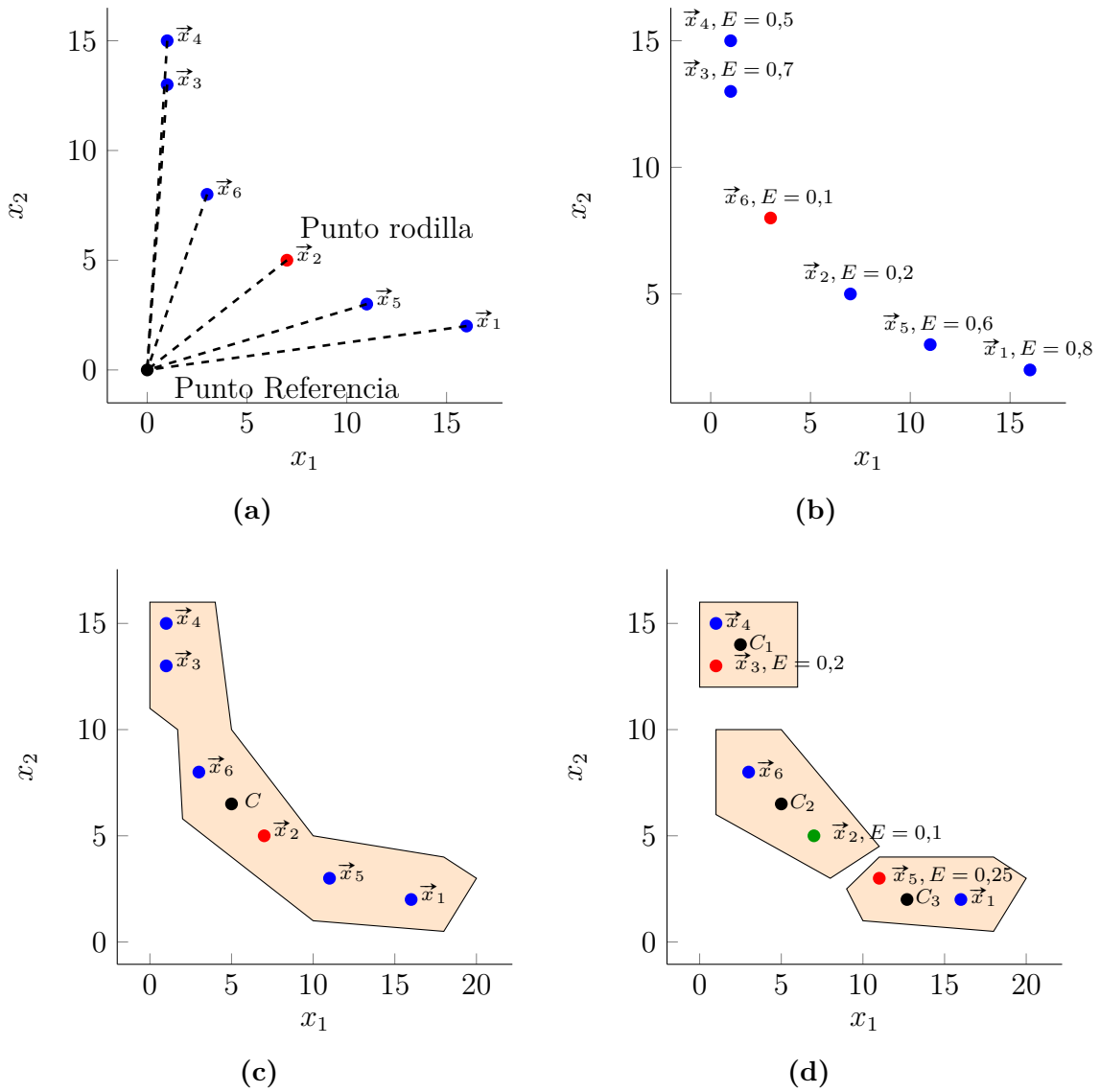


Figura 4.5: Métodos se selección de preferencias propuestos para escoger una solución dentro del frente de Pareto encontrado por eMODiTS. (a) Método de la rodilla, donde el punto $\vec{x}_2$ es la solución más cercana al punto de referencia, y por lo tanto es el punto seleccionado. (b) Método CV, donde el punto $\vec{x}_2$ es la solución seleccionada debido a que presenta el menor error en clasificación E. (c) Método KM1, donde el punto $\vec{x}_2$ es la solución más cercana al punto medio (centroide) C del grupo formado por k-medias con $k = 1$ y es la elegida para ser reportada por eMODiTS. (d) Método KM20, donde la solución $\vec{x}_2$ es la que menor error en clasificación E obtuvo de las soluciones cercanas a sus respectivos centroides.

5
Experimentación

Contenido

5.1. Introducción **73**
5.2. Bases de Datos Temporales **75**
5.3. Evaluación de Métodos de Selección de una Solución del Frente de Pareto **78**
5.3.1. Resultados 78
5.3.2. Discusión 80
5.4. Comparación Contra Otros Métodos Simbólicos **81**
5.4.1. Introducción 81
5.4.2. Resultados 82
5.4.3. Discusión 95
5.5. Interpretación Gráfica **99**
5.5.1. Introducción 99
5.5.2. Resultados 100
5.5.3. Discusión 100

5.1. Introducción

En esta sección se presentan los experimentos y resultados dirigidos a evaluar el desempeño de nuestra propuesta. El primer análisis realizado consistió en comparar los cuatro métodos de selección de preferencias descritos en la Sección 4.4, con la finalidad de escoger una solución con el error en clasificación menor en la mayoría de las bases de datos. Una vez seleccionada una solución, se realizó el segundo

análisis, el cual consistió en comparar varios métodos de discretización simbólicas basados en SAX contra eMODiTS. Cada comparación fue realizada de acuerdo a la tasa de error en clasificación encontrada mediante el clasificador de árboles de decisión bajo el mismo ambiente experimental.

La Tabla 5.1 resume el entorno de parámetros usado en todos los experimentos realizados. Los valores de los parámetros fueron seleccionados basados en un análisis previo donde se compararon otros conjuntos de valores, siendo los reportados en la Tabla 5.1 aquellos con los mejores resultados obtenidos basados en la prueba estadística multi-comparador de Friedman. En dicho análisis se compararon cinco configuraciones de parámetros diferentes, donde se buscó que el algoritmo tuviera una mayor exploración del espacio de búsqueda, incrementando el número de evaluaciones y ejecuciones independientes de nuestro algoritmo, para proporcionarle mayor posibilidad de alcanzar soluciones competitivas.

Tabla 5.1: Entorno de parámetros usado en eMODiTS

Nombre del Parámetro	Valor
Tamaño de la población	100
Número de Generaciones	300
Número de ejecuciones independientes	15
Probabilidad de Cruza	80 %
Probabilidad de Mutación	20 %

Para validar estadísticamente los resultados mostrados en esta sección, se aplicó una prueba de normalidad a los datos para determinar el método idóneo a utilizar. Los resultados de dicha prueba sugirieron que la distribución de los datos no seguía una distribución normal, por lo tanto, se seleccionó la prueba estadística multi-comparador llamada *Prueba de Friedman*, la cuál ha demostrado buenos resultados en datos bajo un distribución no normal. Junto con esta prueba, se utilizó la prueba post hoc de Nemenyi con un 95 % de confianza para reforzar los datos obtenidos por la prueba de Friedman. En nuestro contexto, la hipótesis nula es que todos los métodos comparados obtienen resultados similares sin presentar diferencias significativas entre ellos [32, 46].

Es importante señalar que, la mayoría de las pruebas estadísticas multi-comparador utilizan una categorización para ordenar los métodos de acuerdo al rendimiento alcanzado en cada caso de prueba. Para la asignación de una *categoría* o *rango*, las pruebas asignan el número uno al algoritmo con mejor rendimiento, el valor dos al segundo mejor, y así sucesivamente; en caso de que haya empates (dos o más algoritmos obtuvieron el mismo rendimiento) se les asigna el promedio de las categorías involucradas, por ejemplo, si dos métodos obtuvieron el mejor rendimiento, se les asigna a ambos el promedio del primer y segundo lugar, en otras palabras, ambos obtendrían 1,5 como rango [32, 46].

Los resultados que se presentan en esta sección fueron ejecutados usando un equipo de cómputo con procesadores Intel(R) Xeon(R), CPU E5-2680 v4 a 2.40GHz. Mientras que, la propuesta fue codificada utilizando el lenguaje de programación JAVA por ser un lenguaje de programación libre y se puede ejecutar en cualquier sistema operativo. El código se puede consultar en el repositorio `https://github.com/scoramg/eMODiTS/`.

5.2. Bases de Datos Temporales

Antes de describir cada uno de los experimentos realizados en este trabajo, es necesario describir las características de los conjuntos de datos usados como prueba. Dichos conjuntos de datos fueron tomados de [9], el cual es un repositorio con 85 bases de datos con series de tiempo artificiales y de aplicación real. La Tabla 5.2 resume las principales características de cada base de datos, las cuales son: nombre, tamaño del conjunto de entrenamiento (# STE), tamaño del conjunto de prueba (# STP), número de clases (C), longitud de la serie temporal (T), y si cada base de datos es normalizada o no (NI). Las características de balanceo de clases (BC), proximidad entre clases (PC) y ruido (R) fueron propuestos en este trabajo para medir la complejidad de cada conjunto de datos temporal.

Tabla 5.2: Características de las bases de datos temporales usadas en cada experimento realizado en este trabajo. # STE representa el número de series de tiempo en el conjunto de entrenamiento, # STP es el número de series de tiempo en el conjunto de prueba, C es el número de clases, T es el tamaño de cada serie temporal, NI representa si la base fue inicialmente normalizada o no, BC es si la base de datos esta balanceado, PC es la proximidad entre las clases y R es la cantidad de ruido presente en la base de datos temporal.

Base de datos	# STE	# STP	C	T	NI	BC	PC	R
Adiac	390	391	37	176	Si	No	Traslapado	Bajo
ArrowHead	36	175	3	251	No	No	Separado	Bajo
Beef	30	30	5	470	No	Si	Separado	Bajo
BeetleFly	20	20	2	512	No	Si	Separado	Bajo
BirdChicken	20	20	2	512	No	Si	Separado	Bajo
Car	60	60	4	577	Si	Si	Separado	Bajo
CBF	30	900	3	128	Si	Si	Separado	Bajo
ChlorineConcentration	467	3840	3	166	Si	No	Traslapado	Alto
CinCECGtorso	40	1380	4	1639	Si	Si	Separado	Alto
Coffee	28	28	2	286	No	No	Separado	Bajo
Computers	250	250	2	720	No	Si	Separado	Alto
CricketX	390	390	12	300	Si	No	Traslapado	Alto
CricketY	390	390	12	300	Si	No	Traslapado	Bajo
CricketZ	390	390	12	300	Si	No	Traslapado	Alto
DiatomSizeReduction	16	306	4	345	Si	No	Separado	Bajo
DistalPhalanxOutlineAgeGroup	400	139	3	80	No	No	Separado	Alto
DistalPhalanxOutlineCorrect	600	276	2	80	No	No	Separado	Alto
DistalPhalanxTW	400	139	6	80	No	No	Separado	Alto
Earthquakes	322	139	2	512	No	No	Separado	Alto
ECG200	100	100	2	96	No	No	Separado	Alto
ECG5000	500	4500	5	140	Si	No	Traslapado	Alto
ECGFiveDays	23	861	2	136	Si	Si	Separado	Alto
ElectricDevices	8926	7711	7	96	No	No	Traslapado	Alto
FaceAll	560	1690	14	131	Si	No	Separado	Alto
FaceFour	24	88	4	350	Si	No	Separado	Bajo
FacesUCR	200	2050	14	131	Si	No	Traslapado	Bajo
FiftyWords	450	455	50	270	Si	No	Traslapado	Alto
Fish	175	175	7	463	Si	Si	Separado	Bajo
FordA	3601	1320	2	500	No	No	Separado	Bajo
FordB	3636	810	2	500	No	No	Separado	Bajo
GunPoint	50	150	2	150	Si	Si	Separado	Bajo
Ham	109	105	2	431	No	No	Separado	Bajo
HandOutlines	1000	370	2	2709	Si	No	Separado	Bajo
Haptics	155	308	5	1092	Si	No	Traslapado	Bajo
Herring	64	64	2	512	No	No	Separado	Bajo
InlineSkate	100	550	7	1882	Si	No	Traslapado	Bajo
InsectWingbeatSound	220	1980	11	256	Si	Si	Separado	Alto
ItalyPowerDemand	67	1029	2	24	Si	No	Separado	Alto
LargeKitchenAppliances	375	375	3	720	No	Si	Traslapado	Alto
Lighting2	60	61	2	637	Si	No	Separado	Bajo
Lighting7	70	73	7	319	Si	No	Separado	Alto
Mallat	55	2345	8	1024	Si	Si	Separado	Bajo
Meat	60	60	3	448	No	Si	Separado	Bajo
MedicalImages	381	760	10	99	Si	No	Traslapado	Alto
MiddlePhalanxOutlineAgeGroup	400	154	3	80	No	No	Separado	Alto
MiddlePhalanxOutlineCorrect	600	291	2	80	No	No	Separado	Alto
MiddlePhalanxTW	399	154	6	80	No	No	Separado	Alto
MoteStrain	20	1252	2	84	Si	No	Separado	Alto
NonInvasiveFetalECGThorax1	1800	1965	42	750	Si	No	Separado	Bajo
NonInvasiveFetalECGThorax2	1800	1965	42	750	Si	No	Separado	Bajo
OliveOil	30	30	4	570	No	No	Separado	Bajo
OSULeaf	200	242	6	427	Si	No	Traslapado	Bajo
PhalangesOutlinesCorrect	1800	858	2	80	No	No	Separado	Alto
Phoneme	214	1896	39	1024	Si	No	Traslapado	Bajo
Plane	105	105	7	144	No	Si	Separado	Bajo
ProximalPhalanxOutlineAgeGroup	400	205	3	80	No	No	Separado	Alto
ProximalPhalanxOutlineCorrect	600	291	2	80	No	No	Separado	Alto
ProximalPhalanxTW	400	205	6	80	No	No	Separado	Alto
RefrigerationDevices	375	375	3	720	No	Si	Traslapado	Alto
ScreenType	375	375	3	720	No	Si	Traslapado	Alto
ShapeletSim	20	180	2	500	No	Si	Traslapado	Alto
ShapesAll	600	600	60	512	No	No	Separado	Alto
SmallKitchenAppliances	375	375	3	720	No	Si	Traslapado	Alto
SonyAIBORobotSurface1	20	601	2	70	Si	No	Separado	Alto
SonyAIBORobotSurface2	27	953	2	65	Si	No	Separado	Alto
StarLightCurves	1000	8236	3	1024	Si	No	Separado	Bajo
Strawberry	613	370	2	235	No	No	Separado	Bajo
SwedishLeaf	500	625	15	128	Si	Si	Separado	Bajo
Symbols	25	995	6	398	Si	No	Separado	Bajo
SyntheticControl	300	300	6	60	Si	Si	Separado	Alto
ToeSegmentation1	40	228	2	277	No	No	Separado	Bajo
ToeSegmentation2	36	130	2	343	No	No	Traslapado	Alto
Trace	100	100	4	275	Si	Si	Separado	Alto
TwoLeadECG	23	1139	2	82	Si	Si	Separado	Alto
TwoPatterns	1000	4000	4	128	Si	No	Separado	Alto
UWaveGestureLibraryAll	896	3582	8	945	No	No	Separado	Bajo

(Continua...)

BASE DE DATOS	# STE	# STP	C	T	NI	BC	PC	R
UWaveGestureLibraryX	896	3582	8	315	Si	No	Separado	Bajo
UWaveGestureLibraryY	896	3582	8	315	Si	No	Separado	Bajo
UWaveGestureLibraryZ	896	3582	8	315	Si	No	Separado	Bajo
Wafer	1000	6164	2	152	Si	No	Separado	Alto
Wine	57	54	2	234	No	No	Separado	Bajo
WordSynonyms	267	638	25	270	Si	No	Traslapado	Alto
Worms	181	77	5	900	No	No	Traslapado	Alto
WormsTwoClass	181	77	2	900	No	No	Separado	Bajo
Yoga	300	3000	2	426	Si	No	Separado	Bajo

Para calcular el balanceo entre clases en una base de datos, se utilizó la Ecuación 5.1 [63, 66], donde TD es una base de datos temporal y $P(c)_i$ representa la proporción entre el número de instancias de una clase y el número total de instancias. Como puede observarse, dicha Ecuación utiliza la Entropía para determinar si cada etiqueta de clase contiene el mismo número de instancias.

$$BC(TD) = \begin{cases} No & (-1 * \sum_{i=1}^{C} P(c)_i * \log(P(c)_i))\ mod\ \log(C) \\ Si & \text{caso contrario} \end{cases} \tag{5.1}$$

Por otro lado, la proximidad entre clases se calcula usando el *Coeficiente de Silhouette (SC)*[97]. Dicho coeficiente es una métrica para evaluar la consistencia entre los grupos de datos generados por algún algoritmo de agrupamiento; es decir, mide qué tan bien están formados (separados) los grupos. En nuestro caso, el SC se utiliza para medir si las etiquetas de clase están separadas o traslapadas, tal como se expresa en la Ecuación 5.2.

$$\begin{aligned} PC(TD) &= \begin{cases} Traslapado, & SC(TD) < 0 \\ Separado, & SC(TD) \geq 0 \end{cases} \\ SC(TD) &= \frac{1}{N} \sum_{i=0}^{N} \frac{Separación(TS_i) - Cohesión(TS_i)}{\text{máx}\{Cohesión(TS_i), Separación(TS_i)\}} \end{aligned} \tag{5.2}$$

La función $Cohesión(TS_i)$ es la distancia de una serie de tiempo en la posición i con respecto al conjunto de series de tiempo etiquetadas con la misma clase [97]. Sea $TS_i \in K_c$, donde $K_c = \{TS_1, TS_2, \ldots, TS_k\}$ es un subconjunto de la base temporal de tamaño k perteneciente a la clase c, y $DE(TS_i, TS_j)$ la distancia euclidiana de la serie temporal TS_i a TS_j, la función cohesión se calcula mediante la Ecuación 5.3.

$$Cohesión(TS_i) = \frac{1}{k} \sum_{i=0, j \neq i}^{k} DE(TS_i, TS_j) \tag{5.3}$$

Mientras tanto, la función $Separación(TS_i)$ es la distancia entre la serie de tiempo TS_i a cada serie de tiempo perteneciente al subconjunto de la clase más próxima donde TS_i no es miembro. Sea $L = \{K_{c_1}, K_{c_2}, \ldots, K_{c_{C-1}}\}$ el conjunto de series de tiempo pertenecientes a las etiquetas de clase que no corresponden a la clase de la serie temporal TS_i, la función separación es calculada a través de la Ecuación 5.4.

$$Separación(TS_i) = \min_{\forall K | K \in L} \{\frac{1}{k} \sum_{p=0}^{k} DE(TS_i, TS_p)\} \tag{5.4}$$

Por último, el nivel de ruido presente en cada base de datos es calculado mediante la Ecuación 5.5 [89], donde $RD = \{Ruido(TD_1), Ruido(TD_2), \ldots, Ruido(TD_r)\}$ es el conjunto de medidas de ruido obtenidas para todas las bases de datos temporales, $\widetilde{RD}$ es el valor de la mediana del conjunto RD, $\overline{TS}_i$ es la serie de tiempo promedio para una clase en particular, y $\widetilde{TS}_i$ es la serie de tiempo de la mediana del conjunto para una clase específica. Para el cálculo de la distancia entre estas series temporales se tomó la métrica de la Subsecuencia Común más Larga (LCSS) descrita en la Sección 2.3.3, y calculada mediante la Ecuación 2.7, dado que permite obtener distancias normalizadas dentro del intervalo $[0, 1]$, donde valores cercanos a 0 significa similitud y valores cercanos a 1 disimilitud.

$$\begin{aligned} R(TD) &= \begin{cases} Bajo, & Ruido(TD) <= \widetilde{RD} \\ Alto, & Ruido(TD) > \widetilde{RD} \end{cases} \\ Ruido(TD) &= \frac{1}{C}\sum_{i}^{C} LCSS(\overline{TS}_i, \widetilde{TS}_i) \end{aligned} \tag{5.5}$$

5.3. Evaluación de Métodos de Selección de una Solución del Frente de Pareto

5.3.1. Resultados

Antes de comparar el rendimiento de eMODiTS con respecto a otros enfoques basados en SAX, es necesario decidir la forma de extraer la solución del Frente de Pareto que será utilizada para tal fin. Los cuatro métodos usados en este análisis fueron descritos en la Sección 4.4. Cada uno de ellos fue diseñado de acuerdo al tipo de problema al que está dirigido nuestro enfoque: *Clasificación de series de tiempo*.

La Tabla 5.3 muestra la tasa de error en clasificación obtenida por cada método bajo el mismo ambiente experimental descrito en la Sección 5.1, donde los números entre paréntesis representan la categoría o rango asignado, y los números en negritas representan los métodos con el mejor resultado entre los cuatro, el cuál, en nuestro contexto implicaría aquel con la menor tasa de error en clasificación.

Tabla 5.3: Error en clasificación obtenido por cada método de selección de preferencias a través del clasificador de Árbol. Los valores posteriores al símbolo ± representan la desviación estándar de los datos, el número entre paréntesis representa el rango calculado por la prueba estadística multi-comparador, y los números en negrita representan los mejores valores obtenidos por cada base de datos temporal.

Base de datos	CV	Knee	KM1	KM20
Adiac	0.5358 ± 0.0055 (2)	0.5418 ± 0.0121 (3)	0.5441 ± 0.0148 (4)	**0.5193 ± 0.0155 (1)**
ArrowHead	0.2923 ± 0.0499 (3)	**0.2042 ± 0.0295 (1)**	0.3936 ± 0.0215 (4)	0.2483 ± 0.0442 (2)
Beef	0.4397 ± 0.0371 (2)	**0.4091 ± 0.0342 (1)**	0.5701 ± 0.0302 (4)	0.536 ± 0.084 (3)
BeetleFly	**0.08 ± 0.051 (1)**	0.3388 ± 0.0464 (4)	0.3376 ± 0.0765 (3)	0.2118 ± 0.0839 (2)
BirdChicken	**0.1169 ± 0.07 (1)**	0.4429 ± 0.0519 (4)	0.2766 ± 0.0695 (2)	0.3344 ± 0.0638 (3)
Car	**0.2983 ± 0.0345 (1)**	0.3445 ± 0.0174 (4)	0.2998 ± 0.0631 (2)	0.3042 ± 0.039 (3)
CBF	0.171 ± 0.0122 (3)	**0.1087 ± 0.0201 (1)**	0.2641 ± 0.012 (4)	0.115 ± 0.0158 (2)
ChlorineConcentration	0.2515 ± 0.0077 (2)	**0.2431 ± 0.0076 (1)**	0.436 ± 0.0031 (4)	0.2519 ± 0.0095 (3)
CinCECGtorso	0.3668 ± 0.0112 (2)	**0.3522 ± 0.0121 (1)**	0.3792 ± 0.014 (4)	0.3728 ± 0.0049 (3)
Coffee	**0.1872 ± 0.0868 (1)**	0.5304 ± 0.031 (4)	0.3268 ± 0.0417 (2)	0.3384 ± 0.0628 (3)

(Continua...)

Base de datos	CV	Knee	KM1	KM20
Computers	**0.3334 ± 0.0139 (1)**	0.3561 ± 0.0164 (3)	0.3878 ± 0.0194 (4)	0.3375 ± 0.019 (2)
CricketX	0.643 ± 0.0152 (3)	**0.6021 ± 0.0099 (1)**	0.6351 ± 0.0105 (2)	0.6693 ± 0.0208 (4)
CricketY	0.6123 ± 0.0065 (4)	0.5697 ± 0.0151 (2)	0.5868 ± 0.0287 (3)	**0.531 ± 0.0297 (1)**
CricketZ	0.5766 ± 0.0118 (2)	0.6202 ± 0.0141 (3)	0.6291 ± 0.0111 (4)	**0.5688 ± 0.0166 (1)**
DiatomSizeReduction	0.1297 ± 0.0079 (2)	0.6377 ± 0.0143 (4)	**0.1096 ± 0.0289 (1)**	0.1739 ± 0.0288 (3)
DistalPhalanxOutlineAgeGroup	**0.2028 ± 0.009 (1)**	0.2303 ± 0.0205 (3)	0.2339 ± 0.0112 (4)	0.2044 ± 0.0206 (2)
DistalPhalanxOutlineCorrect	**0.1832 ± 0.0165 (1)**	0.216 ± 0.0047 (3)	0.246 ± 0.0125 (4)	0.1968 ± 0.0116 (2)
DistalPhalanxTW	0.264 ± 0.0108 (2)	0.2983 ± 0.0116 (4)	0.291 ± 0.0091 (3)	**0.2598 ± 0.0065 (1)**
Earthquakes	**0.2014 ± 0.0044 (2)**	**0.2014 ± 0.0044 (2)**	**0.2014 ± 0.0044 (2)**	0.219 ± 0.0117 (4)
ECG200	0.1508 ± 0.0398 (3)	0.2022 ± 0.0435 (4)	**0.1175 ± 0.0308 (1)**	0.1383 ± 0.0122 (2)
ECG5000	0.0767 ± 0.0021 (3)	0.0688 ± 0.0017 (2)	0.0864 ± 0.0045 (4)	**0.0675 ± 0.0036 (1)**
ECGFiveDays	0.2386 ± 0.0223 (2)	0.3077 ± 0.0125 (3)	0.4012 ± 0.0215 (4)	**0.1646 ± 0.0099 (1)**
ElectricDevices	0.2966 ± 0.0016 (2)	0.4665 ± 0.0031 (4)	0.3106 ± 0.0019 (3)	**0.2925 ± 0.0053 (1)**
FaceAll	**0.303 ± 0.0105 (1)**	0.3348 ± 0.0067 (2)	0.349 ± 0.0144 (4)	0.3386 ± 0.0058 (3)
FaceFour	**0.2007 ± 0.0209 (1.5)**	**0.2007 ± 0.0209 (1.5)**	0.5254 ± 0.0532 (4)	0.2568 ± 0.0598 (3)
FacesUCR	0.4429 ± 0.0081 (3)	0.3831 ± 0.0111 (2)	0.5039 ± 0.0101 (4)	**0.336 ± 0.0124 (1)**
FiftyWords	0.6053 ± 0.0178 (4)	0.5544 ± 0.0086 (2)	0.5961 ± 0.011 (3)	**0.546 ± 0.0063 (1)**
Fish	0.3992 ± 0.0143 (3)	0.3744 ± 0.02 (2)	0.4264 ± 0.0213 (4)	**0.3734 ± 0.0328 (1)**
FordA	**0.4139 ± 0.0032 (1)**	0.4517 ± 0.006 (3)	0.487 ± 0.0004 (4)	0.4216 ± 0.005 (2)
FordB	0.4723 ± 0.0033 (4)	0.456 ± 0.0064 (2)	**0.4408 ± 0.0045 (1)**	0.4698 ± 0.0075 (3)
GunPoint	**0.134 ± 0.027 (1)**	0.2784 ± 0.028 (4)	0.1422 ± 0.0217 (2)	0.1775 ± 0.0234 (3)
Ham	**0.2203 ± 0.0297 (1)**	0.2824 ± 0.0148 (2)	0.3173 ± 0.0583 (4)	0.2842 ± 0.0258 (3)
HandOutlines	**0.1107 ± 0.0127 (1.5)**	0.1367 ± 0.0103 (4)	0.1329 ± 0.0053 (3)	**0.1107 ± 0.0127 (1.5)**
Haptics	0.5673 ± 0.029 (2)	0.5684 ± 0.047 (3)	0.6004 ± 0.0381 (4)	**0.5389 ± 0.0493 (1)**
Herring	0.3858 ± 0.0423 (2)	0.3942 ± 0.0312 (3)	0.4081 ± 0.0207 (4)	**0.3517 ± 0.0348 (1)**
InlineSkate	0.7399 ± 0.0138 (4)	**0.6666 ± 0.0099 (1)**	0.7019 ± 0.0128 (3)	0.7003 ± 0.0207 (2)
InsectWingbeatSound	**0.4201 ± 0.0097 (1)**	0.4472 ± 0.0123 (3)	0.4385 ± 0.0146 (2)	0.4815 ± 0.0125 (4)
ItalyPowerDemand	0.059 ± 0.0036 (3)	**0.0415 ± 0.0067 (1)**	0.0874 ± 0.0035 (4)	0.0426 ± 0.0056 (2)
LargeKitchenAppliances	0.4084 ± 0.0099 (2)	0.4442 ± 0.0084 (4)	0.4119 ± 0.0111 (3)	**0.3832 ± 0.0179 (1)**
Lighting2	0.195 ± 0.0163 (3)	0.1926 ± 0.0345 (2)	0.317 ± 0.0171 (4)	**0.1775 ± 0.052 (1)**
Lighting7	**0.296 ± 0.0327 (1)**	0.4137 ± 0.011 (3)	0.4387 ± 0.025 (4)	0.3883 ± 0.0361 (2)
Mallat	0.2418 ± 0.0079 (3)	**0.0732 ± 0.0042 (1)**	0.2766 ± 0.0042 (4)	0.1768 ± 0.0124 (2)
Meat	**0.0173 ± 0.0099 (1)**	0.5962 ± 0.024 (4)	0.2617 ± 0.0219 (3)	0.0175 ± 0.0177 (2)
MedicalImages	**0.3041 ± 0.0104 (1)**	0.3459 ± 0.0144 (3)	0.3462 ± 0.0208 (4)	0.3331 ± 0.0126 (2)
MiddlePhalanxOutlineAgeGroup	**0.2691 ± 0.022 (1)**	0.2721 ± 0.017 (2)	0.3002 ± 0.0147 (4)	0.2776 ± 0.0129 (3)
MiddlePhalanxOutlineCorrect	**0.2149 ± 0.0086 (1)**	0.2768 ± 0.0279 (3)	0.2931 ± 0.0213 (4)	0.246 ± 0.0151 (2)
MiddlePhalanxTW	0.4063 ± 0.0121 (3)	0.3938 ± 0.0233 (2)	0.4088 ± 0.0144 (4)	**0.3932 ± 0.0079 (1)**
MoteStrain	0.2783 ± 0.0058 (4)	0.2006 ± 0.0026 (2)	**0.166 ± 0.0257 (1)**	0.2755 ± 0.0088 (3)
NonInvasiveFetalECGThorax1	0.3451 ± 0.012 (3)	**0.3198 ± 0.0099 (1)**	0.3791 ± 0.015 (4)	0.3374 ± 0.0055 (2)
NonInvasiveFetalECGThorax2	**0.2462 ± 0.0091 (1)**	0.2522 ± 0.0028 (3)	0.3459 ± 0.0069 (4)	0.2492 ± 0.005 (2)
OliveOil	0.1665 ± 0.037 (3)	**0.0969 ± 0.0547 (1)**	0.2011 ± 0.0555 (4)	0.1312 ± 0.075 (2)
OSULeaf	0.5334 ± 0.0245 (2)	0.6291 ± 0.0303 (3)	0.6305 ± 0.0153 (4)	**0.5314 ± 0.0262 (1)**
PhalangesOutlinesCorrect	**0.2418 ± 0.0062 (1)**	0.264 ± 0.0091 (4)	0.2524 ± 0.0057 (3)	0.2495 ± 0.0071 (2)
Phoneme	0.8958 ± 0.0034 (4)	0.8805 ± 0.0053 (2)	**0.8749 ± 0.0079 (1)**	0.8861 ± 0.0094 (3)
Plane	**0.0426 ± 0.0163 (1)**	0.0692 ± 0.0093 (2)	0.3153 ± 0.0371 (4)	0.0954 ± 0.0097 (3)
ProximalPhalanxOutlineAgeGroup	**0.1288 ± 0.0203 (1)**	0.1857 ± 0.0138 (4)	0.1495 ± 0.0151 (2)	0.16 ± 0.0138 (3)
ProximalPhalanxOutlineCorrect	**0.1874 ± 0.0134 (1)**	0.2171 ± 0.0055 (3)	0.2442 ± 0.0074 (4)	0.1954 ± 0.0068 (2)
ProximalPhalanxTW	0.2039 ± 0.0206 (2)	**0.2019 ± 0.0093 (1)**	0.2259 ± 0.0095 (3)	0.2263 ± 0.0107 (4)
RefrigerationDevices	**0.4684 ± 0.0161 (1)**	0.4895 ± 0.0078 (4)	0.4815 ± 0.0286 (2)	0.4884 ± 0.0153 (3)
ScreenType	0.5831 ± 0.0188 (4)	**0.5127 ± 0.0257 (1)**	0.5317 ± 0.0147 (2)	0.547 ± 0.0143 (3)
ShapeletSim	0.4796 ± 0.0352 (3)	0.4971 ± 0.0276 (4)	0.4176 ± 0.0202 (2)	**0.3624 ± 0.035 (1)**
ShapesAll	**0.522 ± 0.0092 (1)**	0.5521 ± 0.0162 (3)	0.6093 ± 0.0044 (4)	0.5312 ± 0.0051 (2)
SmallKitchenAppliances	0.3405 ± 0.0118 (2)	0.4009 ± 0.0228 (3)	0.4555 ± 0.0283 (4)	**0.3288 ± 0.0149 (1)**
SonyAIBORobotSurface1	0.2067 ± 0.0139 (2)	0.2977 ± 0.0143 (3)	**0.1784 ± 0.0115 (1)**	0.4483 ± 0.0045 (4)
SonyAIBORobotSurface2	**0.1907 ± 0.0117 (1)**	0.2611 ± 0.0143 (3)	0.2269 ± 0.0119 (2)	0.2714 ± 0.0218 (4)
StarLightCurves	0.0821 ± 0.0017 (2.5)	0.0821 ± 0.0017 (2.5)	0.1014 ± 0.0024 (4)	**0.0806 ± 0.0016 (1)**
Strawberry	0.0787 ± 0.0042 (2)	0.107 ± 0.0085 (3)	0.1152 ± 0.0122 (4)	**0.0775 ± 0.0101 (1)**
SwedishLeaf	0.368 ± 0.0213 (2)	**0.3361 ± 0.0061 (1)**	0.4004 ± 0.0197 (4)	0.381 ± 0.0111 (3)
Symbols	0.1493 ± 0.0086 (2)	0.2335 ± 0.0057 (4)	**0.1263 ± 0.0069 (1)**	0.2151 ± 0.0073 (3)
SyntheticControl	0.0867 ± 0.0142 (2.5)	0.0867 ± 0.0142 (2.5)	0.1194 ± 0.0147 (4)	**0.0808 ± 0.0178 (1)**
ToeSegmentation1	0.4559 ± 0.0096 (4)	0.397 ± 0.0262 (2)	**0.3807 ± 0.0482 (1)**	0.4202 ± 0.0264 (3)

(Continua...)

Base de datos	CV	Knee	KM1	KM20
ToeSegmentation2	**0.2351 ± 0.0233 (1)**	0.2506 ± 0.0146 (4)	0.2457 ± 0.0092 (2.5)	0.2457 ± 0.0092 (2.5)
Trace	0.0958 ± 0.0047 (2)	0.2155 ± 0.0168 (3)	0.2405 ± 0.0236 (4)	**0.0582 ± 0.0228 (1)**
TwoLeadECG	0.2787 ± 0.0317 (4)	**0.2314 ± 0.0211 (1.5)**	0.2706 ± 0.0238 (3)	**0.2314 ± 0.0211 (1.5)**
TwoPatterns	**0.1813 ± 0.0017 (1)**	0.2438 ± 0.0031 (3)	0.3127 ± 0.0063 (4)	0.2174 ± 0.0059 (2)
UWaveGestureLibraryAll	**0.1454 ± 0.0028 (1)**	0.1623 ± 0.0053 (3)	0.252 ± 0.0068 (4)	0.1601 ± 0.0056 (2)
UWaveGestureLibraryX	**0.3031 ± 0.0061 (1)**	0.3531 ± 0.0052 (4)	0.3402 ± 0.0082 (3)	0.3331 ± 0.0063 (2)
UWaveGestureLibraryY	0.3913 ± 0.0066 (2)	**0.3862 ± 0.007 (1)**	0.4191 ± 0.0061 (4)	0.3997 ± 0.0084 (3)
UWaveGestureLibraryZ	0.3804 ± 0.0081 (3)	**0.3745 ± 0.007 (1)**	0.3782 ± 0.0041 (2)	0.3957 ± 0.006 (4)
Wafer	0.0042 ± 0.0013 (2)	0.0061 ± 0.0006 (3)	0.0254 ± 0.0017 (4)	**0.0035 ± 0.0011 (1)**
Wine	0.3599 ± 0.0303 (4)	**0.1693 ± 0.0147 (1)**	0.2748 ± 0.0579 (3)	0.2203 ± 0.063 (2)
WordSynonyms	**0.5149 ± 0.0264 (1)**	0.548 ± 0.0121 (2)	0.5824 ± 0.0124 (4)	0.5686 ± 0.021 (3)
Worms	0.5504 ± 0.0336 (3)	0.5214 ± 0.0176 (2)	0.604 ± 0.0183 (4)	**0.5173 ± 0.0191 (1)**
WormsTwoClass	0.4589 ± 0.1056 (3)	0.4438 ± 0.0117 (2)	0.4633 ± 0.072 (4)	**0.4376 ± 0.0642 (1)**
Yoga	0.2044 ± 0.0077 (2)	0.2384 ± 0.0031 (3)	**0.1802 ± 0.0055 (1)**	0.2479 ± 0.0053 (4)

La Figura 5.1a muestra la categorización de los cuatro métodos comparados; en ella, se aprecia el rango promedio alcanzado por cada uno en las 85 bases de datos temporales de prueba. Por su parte, la Figura 5.1b muestra el resultado de aplicar la prueba estadística multi-comparador de Friedman usando la prueba post hoc de Nemenyi; el intervalo de color azul representa el método con el cual se esta visualizando si existe diferencia significativa estadística, los de color negro representan aquellos que no presentaron dicha diferencia con el método del intervalo azul, y el intervalo rojo representan los métodos en donde si se presentó una diferencia significativa con aquel tomado como base para la visualización.

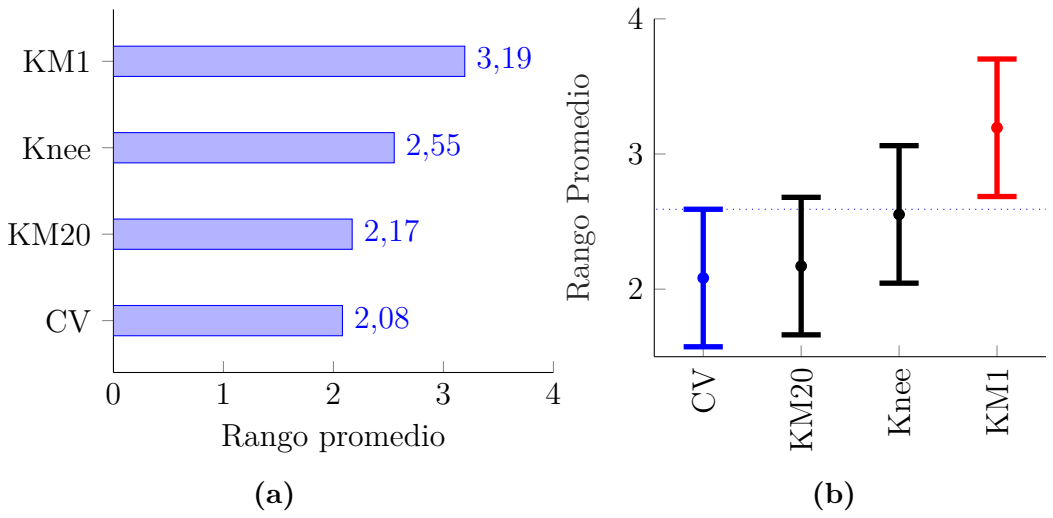


Figura 5.1: (a) Rango promedio obtenido por cada método de selección de preferencias. (b) Resultado de aplicar la prueba post hoc de Nemenyi usando la tasa de error en clasificación mínima obtenida en cada método.

5.3.2. Discusión

En este experimento, se realizó un análisis para escoger el método de selección de preferencias a usar en nuestra propuesta. Para tal tarea, se utilizaron cuatro

métodos: el método CV, el método K20, el método Knee (de la rodilla), y el método K1, cada uno descritos en la Sección 4.4.

Los parámetros utilizados fueron los mismos para cada una de las técnicas comparadas, así como las pruebas estadísticas usadas. La prueba multi-comparador estadística de Friedman junto con la prueba post hoc de Nemenyi fueron aplicadas para validar el comportamiento de cada método en la tarea de seleccionar la solución final del Frente de Pareto obtenido por eMODiTS.

Los resultados sugieren que el método con el menor rango alcanzado, es decir, con la menor tasa de error en clasificación obtenida en la mayoría de las bases de datos temporales, fue el método CV. Dichos resultados fueron confirmados con la prueba estadística aplicada, donde se aprecia que CV supera estadísticamente a K1. Aunque no existe diferencia significativa con los métodos K20 y Knee, CV logra posicionarse mejor en seleccionar soluciones con menor tasa de error en clasificación, por lo que este método fue seleccionado para extraer la solución final del Frente de Pareto que será usada para comparar nuestra propuesta contra otros métodos de discretización simbólica basados en SAX.

5.4. Comparación Contra Otros Métodos Simbólicos

5.4.1. Introducción

Una vez seleccionado el método de selección de preferencias para extraer la solución final del Frente de Pareto obtenido por nuestra propuesta, es conveniente realizar un análisis del rendimiento (en términos de clasificación y discretización) de nuestro enfoque contra métodos de discretización simbólica basados en SAX y encontrados en la literatura especializada.

Los enfoques seleccionados para este análisis, fueron escogidos de acuerdo a las características que poseen y la tarea hacia donde fueron dirigidos, es decir, se escogieron aquellos métodos que tuvieran características similares a nuestra propuesta y que fueran frecuentemente usados en la literatura especializada. Dichos métodos fueron: EP, SAX, αSAX, ESAX, ESAXKMeans, 1D-SAX, RKMeans, SAXKMeans y rSAX. Cada uno de ellos fueron descritos en la Sección 1.7.

Como se mencionó en la Sección 1.7, cada uno de estos métodos ocupan parámetros adicionales a los empleados por eMODiTS, por lo que los valores para éstos fueron seleccionados como sigue: el método EP fue ejecutado con la misma configuración de parámetros empleados por eMODiTS y detallados en la Tabla 5.1. Los métodos restantes usaron el número de cortes de palabra y alfabeto encontrados por EP para cada base de datos, debido a que, originalmente, no fueron probados con las 85 bases de datos del repositorio UCR.

Para el caso de 1d-SAX y rSAX que utilizan parámetros adicionales al tamaño de la palabra y el alfabeto, se utilizaron aquellos valores sugeridos por los autores en las publicaciones donde se propuso cada método. Para 1d-SAX, el número de pendientes (*slopes*) usado fue ocho, y para rSAX, el valor para el parámetro τ fue 10.

Cada método fue comparado en términos de clasificación, reducción de dimensionalidad y pérdida de información. Los resultados son presentados en la siguiente sección.

5.4.2. Resultados

Análisis del error en clasificación

La Tabla 5.4 muestra los resultados obtenidos por cada método en 15 ejecuciones independientes. Los valores resaltados en negritas, representan las tasas de error en clasificación mínimas por cada una de las bases de datos. Se incluyeron las tasas de clasificación obtenidas por el clasificador de árboles de decisión, usando los datos originales o puros de cada base de datos como referencia, es decir, para verificar que nuestro método mejora la clasificación con la discretización obtenida en lugar de usar los datos originales.

La Figura 5.2 ilustra el rango promedio obtenido por cada método basado en SAX, el cual, como se ya se mencionó, es calculado mediante la prueba estadística de Friedman.

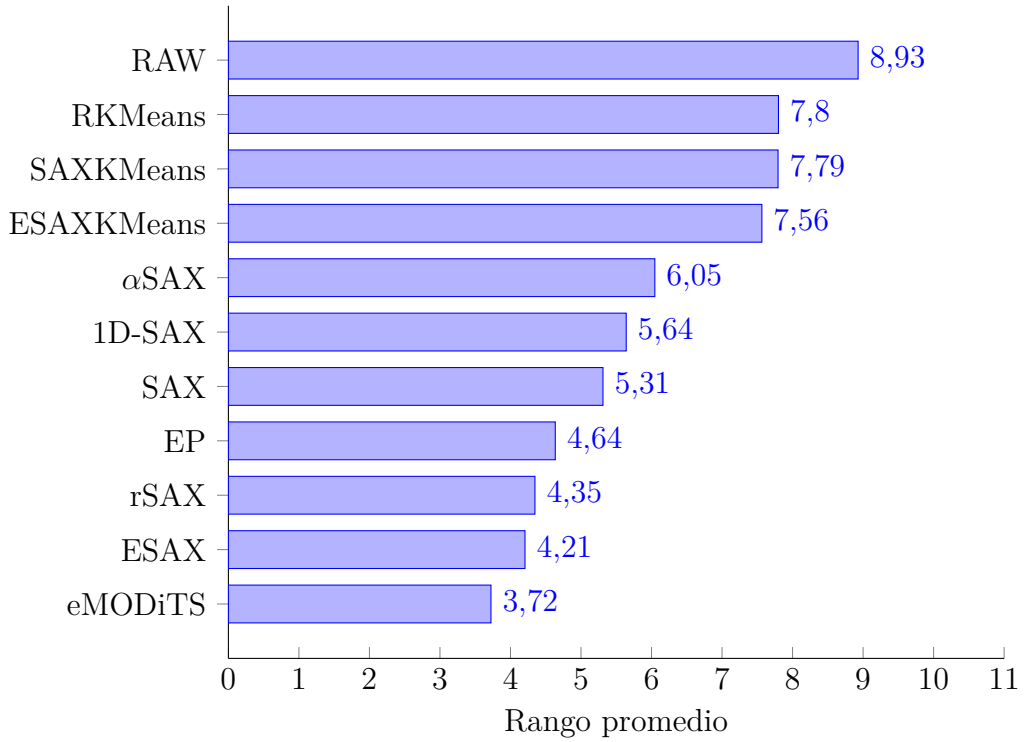


Figura 5.2: Rango promedio obtenido al clasificar las 85 bases de datos temporales de prueba por cada método basado en SAX.

Tabla 5.4: Porcentajes de error en clasificación obtenidos por cada método basado en SAX usando el clasificador de árboles de decisión. Los valores que aparecen después del símbolo ± representan la desviación estándar de los datos, los números entre paréntesis representan el rango calculado por la prueba estadística usada para comparar múltiples métodos a la vez, y los valores resaltados en negrita representan los mejores valores obtenidos en cada base de datos.

Base de datos	eMODiTS	EP	SAX	αSAX	ESAX	ESAXKMeans	1D-SAX	RKMeans	SAXKMeans	rSAX	RAW
Adiac	0.5358 ± 0.0055 (4)	0.5122 ± 0.027 (3)	0.5455 ± 0.0142 (5)	0.7813 ± 0.0052 (8)	0.5008 ± 0.0206 (2)	0.8767 ± 0.0119 (10)	0.5792 ± 0.0103 (7)	0.9002 ± 0.0105 (11)	0.8725 ± 0.0125 (9)	0.5632 ± 0.0209 (6)	**0.4929 (1)**
ArrowHead	0.2923 ± 0.0499 (6)	0.3861 ± 0.028 (9)	0.2359 ± 0.018 (3)	0.3931 ± 0.0207 (10)	**0.2222 ± 0.0217 (1)**	0.3288 ± 0.0675 (8)	0.2251 ± 0.0223 (2)	0.273 ± 0.0323 (5)	0.2961 ± 0.0325 (7)	0.236 ± 0.0123 (4)	0.6384 (11)
Beef	0.4397 ± 0.0371 (4)	0.5886 ± 0.0967 (10)	0.4048 ± 0.0931 (2)	0.4764 ± 0.0257 (6)	0.426 ± 0.097 (3)	0.5862 ± 0.046 (9)	**0.2307 ± 0.0456 (1)**	0.5417 ± 0.083 (8)	0.6601 ± 0.0263 (11)	0.4491 ± 0.0678 (5)	0.4927 (7)
BeetleFly	**0.08 ± 0.051 (1)**	0.3983 ± 0.0442 (9)	0.3041 ± 0.0347 (8)	0.2759 ± 0.0666 (7)	0.2352 ± 0.0406 (6)	0.2234 ± 0.0253 (4)	0.583 ± 0.0691 (10)	0.1945 ± 0.0461 (3)	0.2293 ± 0.0242 (5)	0.1135 ± 0.0445 (2)	0.6875 (11)
BirdChicken	**0.1169 ± 0.07 (1)**	0.4316 ± 0.0557 (10)	0.3907 ± 0.0616 (8)	0.2875 ± 0.083 (3)	0.3463 ± 0.0618 (6)	0.395 ± 0.0452 (9)	0.3112 ± 0.0927 (5)	0.3025 ± 0.1674 (4)	0.3694 ± 0.0515 (7)	0.2077 ± 0.0404 (2)	0.7455 (11)
Car	0.2983 ± 0.0345 (3)	**0.2683 ± 0.0646 (1)**	0.3367 ± 0.0379 (7)	0.2806 ± 0.0833 (2)	0.3081 ± 0.0426 (4)	0.4559 ± 0.026 (10)	0.3296 ± 0.0236 (6)	0.3922 ± 0.0405 (9)	0.3576 ± 0.0249 (8)	0.3128 ± 0.0279 (5)	0.5615 (11)
CBF	0.171 ± 0.0122 (8)	0.1463 ± 0.0079 (7)	**0.0141 ± 0.0043 (1)**	0.2024 ± 0.0118 (9)	0.0839 ± 0.0086 (6)	0.0785 ± 0.0076 (5)	0.0568 ± 0.0123 (4)	0.219 ± 0.0139 (10)	0.0563 ± 0.0052 (3)	0.0238 ± 0.0042 (2)	0.7339 (11)
ChlorineConcentration	0.2515 ± 0.0077 (6)	0.1969 ± 0.0098 (5)	0.1349 ± 0.0055 (2)	0.4066 ± 0.0027 (7)	0.1376 ± 0.0037 (3)	0.4984 ± 0.0075 (9)	**0.119 ± 0.0071 (1)**	0.4576 ± 0.0102 (8)	0.5068 ± 0.0075 (10)	0.1579 ± 0.0072 (4)	0.6737 (11)
CinCECGtorso	0.3668 ± 0.0112 (10)	**0.089 ± 0.0159 (1)**	0.1096 ± 0.0082 (4)	0.1038 ± 0.0065 (3)	0.0935 ± 0.0018 (2)	0.1391 ± 0.0053 (6)	0.1416 ± 0.0211 (8)	0.2074 ± 0.0299 (9)	0.1411 ± 0.0042 (7)	0.111 ± 0.0099 (5)	0.6044 (11)
Coffee	0.1872 ± 0.0868 (6)	0.3017 ± 0.0684 (10)	0.2346 ± 0.0316 (7)	0.2527 ± 0.1142 (8)	0.1058 ± 0.085 (5)	**0.0223 ± 0.0241 (1)**	0.2654 ± 0.0428 (9)	0.0422 ± 0.0274 (2)	0.0914 ± 0.0345 (4)	0.0885 ± 0.0382 (3)	0.8854 (11)
Computers	0.3334 ± 0.0139 (2)	0.4856 ± 0.018 (10)	0.3819 ± 0.0152 (6)	0.4124 ± 0.0242 (8)	**0.329 ± 0.0141 (1)**	0.4297 ± 0.0327 (9)	0.3456 ± 0.0253 (3)	0.3814 ± 0.0087 (5)	0.3525 ± 0.0117 (4)	0.4108 ± 0.0183 (7)	0.5861 (11)
CricketX	0.643 ± 0.0152 (4)	0.6309 ± 0.0156 (3)	0.7545 ± 0.0259 (7)	0.6093 ± 0.0143 (2)	0.7088 ± 0.013 (5)	0.8073 ± 0.0103 (9)	0.7796 ± 0.0084 (8)	0.8478 ± 0.017 (11)	0.8324 ± 0.0119 (10)	0.7442 ± 0.0153 (6)	**0.3349 (1)**
CricketY	0.6123 ± 0.0065 (2)	0.6464 ± 0.03 (4)	0.7224 ± 0.0198 (5)	0.6133 ± 0.0097 (3)	0.7449 ± 0.0215 (7)	0.8264 ± 0.0122 (11)	0.7696 ± 0.0224 (8)	0.7959 ± 0.0116 (10)	0.7785 ± 0.0059 (9)	0.7299 ± 0.0161 (6)	**0.3455 (1)**
CricketZ	0.5766 ± 0.0118 (4)	0.5715 ± 0.0107 (2)	0.6696 ± 0.0254 (5)	0.5736 ± 0.0199 (3)	0.6816 ± 0.0147 (6)	0.7404 ± 0.0262 (9)	0.7393 ± 0.0256 (8)	0.775 ± 0.0105 (10)	0.7798 ± 0.0132 (11)	0.7206 ± 0.0102 (7)	**0.3347 (1)**
DiatomSizeReduction	0.1297 ± 0.0079 (6)	0.0548 ± 0.0103 (2)	0.1583 ± 0.0145 (8)	0.2493 ± 0.0184 (10)	0.0829 ± 0.0109 (4)	**0.0524 ± 0.0115 (1)**	0.1473 ± 0.0104 (7)	0.071 ± 0.0115 (3)	0.1584 ± 0.014 (9)	0.095 ± 0.0172 (5)	0.7329 (11)
DistalPhalanxOutlineAgeGroup	0.2028 ± 0.009 (3)	0.2255 ± 0.0146 (6)	0.221 ± 0.0168 (5)	0.2353 ± 0.0123 (8)	0.1955 ± 0.0132 (2)	0.2658 ± 0.01 (9)	0.2087 ± 0.0068 (4)	0.2741 ± 0.015 (10)	0.2266 ± 0.0185 (7)	**0.1846 ± 0.0164 (1)**	0.7373 (11)

(Continua...)

Base de datos	eMODiTS	EP	SAX	αSAX	ESAX	ESAXKMeans	1D-SAX	RKMeans	SAXKMeans	rSAX	RAW
DistalPhalanxOutlineCorrect	**0.1832 ± 0.0165 (1)**	0.1974 ± 0.0102 (2)	0.2146 ± 0.0176 (4)	0.2785 ± 0.0178 (7)	0.2131 ± 0.0236 (3)	0.3535 ± 0.018 (10)	0.2423 ± 0.0103 (6)	0.3481 ± 0.0104 (9)	0.3102 ± 0.0092 (8)	0.2155 ± 0.0065 (5)	0.7414 (11)
DistalPhalanxTW	**0.264 ± 0.0108 (1)**	0.2985 ± 0.0096 (5)	0.2732 ± 0.0086 (3)	0.3007 ± 0.0075 (6)	0.353 ± 0.0225 (9)	0.3275 ± 0.0125 (8)	0.2698 ± 0.0121 (2)	0.3604 ± 0.0078 (10)	0.304 ± 0.0178 (7)	0.287 ± 0.0167 (4)	0.6314 (11)
Earthquakes	0.2014 ± 0.0044 (4.5)	**0.1879 ± 0.0209 (1)**	0.2014 ± 0.0044 (4.5)	0.2214 ± 0.008 (9)	0.2014 ± 0.0044 (4.5)	0.2113 ± 0.0057 (8)	0.2014 ± 0.0044 (4.5)	0.2249 ± 0.0093 (10)	0.2014 ± 0.0044 (4.5)	0.2014 ± 0.0044 (4.5)	0.697 (11)
ECG200	**0.1508 ± 0.0398 (1)**	0.1919 ± 0.034 (2)	0.2157 ± 0.0139 (4)	0.1992 ± 0.0231 (3)	0.2277 ± 0.0418 (6)	0.2579 ± 0.0243 (8)	0.2983 ± 0.0225 (10)	0.2381 ± 0.017 (7)	0.288 ± 0.0266 (9)	0.2222 ± 0.0192 (5)	0.7714 (11)
ECG5000	0.0767 ± 0.0021 (3)	**0.0697 ± 0.002 (1)**	0.0837 ± 0.003 (7)	0.0744 ± 0.0042 (2)	0.0812 ± 0.0053 (4)	0.0927 ± 0.0059 (9)	0.0829 ± 0.0028 (6)	0.1065 ± 0.003 (10)	0.0921 ± 0.0035 (8)	0.0827 ± 0.0075 (5)	0.9036 (11)
ECGFiveDays	0.2386 ± 0.0223 (9)	0.3074 ± 0.0156 (10)	0.0899 ± 0.0087 (5)	0.2184 ± 0.0197 (8)	**0.0545 ± 0.0051 (1)**	0.0844 ± 0.0123 (4)	0.0795 ± 0.0136 (2)	0.1479 ± 0.0179 (6)	0.2112 ± 0.0066 (7)	0.0796 ± 0.0122 (3)	0.7304 (11)
ElectricDevices	0.2966 ± 0.0016 (2)	0.7273 ± 0.0011 (11)	0.3342 ± 0.0027 (3)	0.4429 ± 0.0022 (9)	**0.2651 ± 0.002 (1)**	0.3346 ± 0.0025 (4)	0.37 ± 0.0038 (7)	0.3649 ± 0.0024 (6)	0.4031 ± 0.0037 (8)	0.3354 ± 0.001 (5)	0.6865 (10)
FaceAll	**0.303 ± 0.0105 (1)**	0.3047 ± 0.03 (2)	0.4086 ± 0.0102 (6)	0.3556 ± 0.0133 (4)	0.3394 ± 0.0136 (3)	0.5519 ± 0.0105 (8)	0.4158 ± 0.0145 (7)	0.6599 ± 0.0113 (11)	0.6393 ± 0.0072 (10)	0.406 ± 0.0116 (5)	0.6364 (9)
FaceFour	0.2007 ± 0.0209 (3)	0.2221 ± 0.0312 (7)	0.2164 ± 0.0293 (6)	**0.1377 ± 0.0237 (1)**	0.1517 ± 0.0559 (2)	0.2918 ± 0.0518 (10)	0.2236 ± 0.0284 (8)	0.2317 ± 0.0279 (9)	0.2128 ± 0.0451 (5)	0.2107 ± 0.0343 (4)	0.6483 (11)
FacesUCR	0.4429 ± 0.0081 (4)	**0.3272 ± 0.0215 (1)**	0.5098 ± 0.0145 (8)	0.382 ± 0.0137 (2)	0.4188 ± 0.0129 (3)	0.629 ± 0.0162 (9)	0.4931 ± 0.011 (6)	0.6557 ± 0.0093 (10)	0.6907 ± 0.0067 (11)	0.4924 ± 0.0101 (5)	0.5069 (7)
FiftyWords	0.6053 ± 0.0178 (4)	0.5415 ± 0.0218 (3)	0.6773 ± 0.0225 (8)	0.523 ± 0.0221 (2)	0.6458 ± 0.0211 (5)	0.745 ± 0.0099 (11)	0.6695 ± 0.0179 (6)	0.7108 ± 0.006 (9)	0.7369 ± 0.0136 (10)	0.6766 ± 0.0079 (7)	**0.4047 (1)**
Fish	0.3992 ± 0.0143 (2)	**0.3348 ± 0.013 (1)**	0.444 ± 0.0178 (4)	0.498 ± 0.0308 (7)	0.4796 ± 0.0319 (6)	0.6628 ± 0.0129 (9)	0.446 ± 0.0379 (5)	0.6632 ± 0.0517 (10)	0.6954 ± 0.0181 (11)	0.421 ± 0.0286 (3)	0.5783 (8)
FordA	0.4139 ± 0.0032 (2)	0.4617 ± 0.0052 (4)	0.4727 ± 0.0051 (5)	**0.4021 ± 0.0096 (1)**	0.4745 ± 0.009 (6)	0.4916 ± 0.0078 (9)	0.4949 ± 0.0077 (10)	0.4834 ± 0.0073 (7)	0.4891 ± 0.0056 (8)	0.4523 ± 0.0061 (3)	0.5302 (11)
FordB	0.4723 ± 0.0033 (6)	0.4788 ± 0.0074 (7)	0.4811 ± 0.0059 (10)	0.4599 ± 0.0064 (4)	0.455 ± 0.0045 (2)	**0.4547 ± 0.0061 (1)**	0.4702 ± 0.0029 (5)	0.4804 ± 0.0128 (9)	0.4795 ± 0.0094 (8)	0.4555 ± 0.0032 (3)	0.5532 (11)
GunPoint	0.134 ± 0.027 (3)	0.2618 ± 0.0279 (7)	0.2949 ± 0.0274 (9)	0.2757 ± 0.015 (8)	0.1565 ± 0.0382 (4)	0.2081 ± 0.0209 (6)	**0.0607 ± 0.0338 (1)**	0.1599 ± 0.0133 (5)	0.3278 ± 0.0288 (10)	0.1145 ± 0.0368 (2)	0.8267 (11)
Ham	**0.2203 ± 0.0297 (1)**	0.2593 ± 0.0125 (2)	0.3258 ± 0.0166 (5)	0.3266 ± 0.0531 (6)	0.3117 ± 0.0397 (4)	0.4542 ± 0.0409 (9)	0.4672 ± 0.041 (10)	0.3967 ± 0.0329 (7)	0.4026 ± 0.0259 (8)	0.3026 ± 0.0189 (3)	0.6964 (11)
HandOutlines	0.1107 ± 0.0127 (2)	0.1142 ± 0.011 (4)	0.1277 ± 0.0045 (7)	0.1121 ± 0.0124 (3)	0.1156 ± 0.0105 (6)	0.137 ± 0.0123 (10)	0.115 ± 0.014 (5)	0.1318 ± 0.0221 (9)	0.1282 ± 0.0084 (8)	**0.1027 ± 0.0104 (1)**	0.8481 (11)
Haptics	0.5673 ± 0.029 (2)	0.6234 ± 0.0424 (4)	0.6761 ± 0.0099 (9)	0.6091 ± 0.0296 (3)	0.6385 ± 0.047 (5)	0.6546 ± 0.0116 (6)	0.6666 ± 0.0209 (7)	0.6821 ± 0.0352 (10)	0.6927 ± 0.0232 (11)	0.6716 ± 0.0164 (8)	**0.3588 (1)**
Herring	0.3858 ± 0.0423 (2)	0.3876 ± 0.0189 (3)	0.4689 ± 0.0171 (7)	0.4268 ± 0.059 (4)	0.5495 ± 0.0536 (10)	0.4932 ± 0.0412 (8)	0.4543 ± 0.0356 (6)	0.4527 ± 0.0282 (5)	**0.3762 ± 0.0337 (1)**	0.5034 ± 0.0255 (9)	0.5538 (11)

(Continua...)

Base de datos	eMODiTS	EP	SAX	αSAX	ESAX	ESAXKMeans	1D-SAX	RKMeans	SAXKMeans	rSAX	RAW
InlineSkate	0.7399 ± 0.0138 (11)	0.7127 ± 0.013 (8)	0.6371 ± 0.0186 (2)	0.668 ± 0.0316 (4)	0.7001 ± 0.0227 (7)	0.6702 ± 0.0133 (5)	0.644 ± 0.0296 (3)	0.7393 ± 0.025 (9)	0.7396 ± 0.0107 (10)	0.6858 ± 0.0225 (6)	**0.2531 (1)**
InsectWingbeatSound	0.4201 ± 0.0097 (2)	0.4471 ± 0.0122 (5)	0.4457 ± 0.012 (3)	**0.3943 ± 0.0108 (1)**	0.4653 ± 0.0069 (8)	0.4581 ± 0.0126 (7)	0.503 ± 0.0065 (10)	0.4894 ± 0.0145 (9)	0.4483 ± 0.007 (6)	0.446 ± 0.0122 (4)	0.5089 (11)
ItalyPowerDemand	0.059 ± 0.0036 (5)	**0.0349 ± 0.005 (1)**	0.0669 ± 0.0081 (7)	0.3598 ± 0.0503 (10)	0.0432 ± 0.0027 (2)	0.0799 ± 0.0055 (8)	0.0456 ± 0.0015 (3)	0.0496 ± 0.0031 (4)	0.1023 ± 0.0111 (9)	0.0626 ± 0.0107 (6)	0.9481 (11)
LargeKitchenAppliances	**0.4084 ± 0.0099 (1)**	0.4293 ± 0.0147 (2)	0.447 ± 0.0205 (4)	0.5529 ± 0.0229 (10)	0.4555 ± 0.0091 (5)	0.5826 ± 0.0124 (11)	0.4783 ± 0.009 (6)	0.5279 ± 0.0244 (7)	0.5478 ± 0.0081 (9)	0.4373 ± 0.0237 (3)	0.5367 (8)
Lighting2	**0.195 ± 0.0163 (1)**	0.2677 ± 0.0295 (2)	0.3778 ± 0.0292 (8)	0.2848 ± 0.0352 (3)	0.3346 ± 0.0291 (5)	0.3442 ± 0.045 (6)	0.3795 ± 0.0153 (9)	0.3744 ± 0.0478 (7)	0.315 ± 0.0209 (4)	0.4402 ± 0.039 (10)	0.6959 (11)
Lighting7	**0.296 ± 0.0327 (1)**	0.5166 ± 0.0252 (5)	0.4706 ± 0.0862 (3)	0.51 ± 0.0379 (4)	0.6437 ± 0.0409 (11)	0.609 ± 0.0335 (10)	0.5969 ± 0.0557 (9)	0.5858 ± 0.0414 (8)	0.5286 ± 0.0258 (6)	0.4262 ± 0.0714 (2)	0.56 (7)
Mallat	0.2418 ± 0.0079 (8)	0.2561 ± 0.0097 (9)	0.1316 ± 0.0056 (7)	0.3311 ± 0.0126 (10)	0.0499 ± 0.0027 (6)	0.0276 ± 0.0019 (2)	**0.0195 ± 0.0063 (1)**	0.031 ± 0.0022 (3)	0.0422 ± 0.0058 (5)	0.0329 ± 0.0066 (4)	0.7008 (11)
Meat	**0.0173 ± 0.0099 (1)**	0.1095 ± 0.0301 (5)	0.151 ± 0.0362 (6)	0.2709 ± 0.0205 (7)	0.0646 ± 0.0274 (3)	0.5525 ± 0.0839 (9)	0.1079 ± 0.0258 (4)	0.3511 ± 0.0665 (8)	0.6701 ± 0.0622 (10)	0.0493 ± 0.0212 (2)	0.9395 (11)
MedicalImages	**0.3041 ± 0.0104 (1)**	0.4015 ± 0.0094 (5)	0.4423 ± 0.0128 (6)	0.3911 ± 0.0119 (4)	0.3581 ± 0.0235 (2)	0.4729 ± 0.0197 (8)	0.4527 ± 0.0168 (7)	0.4932 ± 0.0132 (10)	0.4903 ± 0.0122 (9)	0.3892 ± 0.0086 (3)	0.6227 (11)
MiddlePhalanxOutlineAgeGroup	**0.2691 ± 0.022 (1)**	0.304 ± 0.0251 (4)	0.317 ± 0.0246 (8)	0.3114 ± 0.024 (6)	0.3142 ± 0.0183 (7)	0.3242 ± 0.0186 (9)	0.3091 ± 0.0124 (5)	0.3289 ± 0.0251 (10)	0.3021 ± 0.0101 (3)	0.2994 ± 0.021 (2)	0.5592 (11)
MiddlePhalanxOutlineCorrect	**0.2149 ± 0.0086 (1)**	0.2692 ± 0.0271 (6)	0.2551 ± 0.0133 (3)	0.3443 ± 0.013 (9)	0.2617 ± 0.0179 (4)	0.326 ± 0.014 (7)	0.2687 ± 0.0174 (5)	0.3556 ± 0.0275 (10)	0.3397 ± 0.0081 (8)	0.252 ± 0.0184 (2)	0.7312 (11)
MiddlePhalanxTW	0.4063 ± 0.0121 (2)	**0.4049 ± 0.0068 (1)**	0.4102 ± 0.0203 (4)	0.4291 ± 0.0073 (6)	0.4541 ± 0.0172 (8)	0.4289 ± 0.0083 (5)	0.4654 ± 0.0084 (10)	0.4651 ± 0.0214 (9)	0.4309 ± 0.0147 (7)	0.4066 ± 0.0109 (3)	0.4858 (11)
MoteStrain	0.2783 ± 0.0058 (9)	0.2803 ± 0.008 (10)	0.1696 ± 0.003 (5)	0.1991 ± 0.0036 (6)	**0.123 ± 0.0149 (1)**	0.1585 ± 0.0121 (3)	0.1395 ± 0.0116 (2)	0.2009 ± 0.0074 (7)	0.2065 ± 0.0154 (8)	0.1644 ± 0.0064 (4)	0.7704 (11)
NonInvasiveFetalECGThorax1	**0.3451 ± 0.012 (1)**	0.3907 ± 0.0085 (2)	0.4231 ± 0.0051 (4)	0.477 ± 0.0069 (6)	0.3964 ± 0.0088 (3)	0.8041 ± 0.0038 (11)	0.4808 ± 0.0056 (7)	0.7722 ± 0.0061 (9)	0.7958 ± 0.0081 (10)	0.4312 ± 0.0043 (5)	0.7037 (8)
NonInvasiveFetalECGThorax2	**0.2462 ± 0.0091 (1)**	0.2593 ± 0.0035 (2)	0.3484 ± 0.0069 (5)	0.4287 ± 0.0053 (7)	0.3403 ± 0.002 (4)	0.7165 ± 0.0051 (9)	0.3391 ± 0.0112 (3)	0.6957 ± 0.0057 (8)	0.7504 ± 0.0062 (10)	0.3573 ± 0.0132 (6)	0.791 (11)
OliveOil	0.1665 ± 0.037 (2)	0.321 ± 0.0679 (5)	0.3238 ± 0.0669 (6)	0.5661 ± 0.0478 (8)	**0.0767 ± 0.0594 (1)**	0.7156 ± 0.0392 (9)	0.2646 ± 0.0379 (4)	0.5214 ± 0.0835 (7)	0.8069 ± 0.0577 (11)	0.1863 ± 0.0696 (3)	0.7637 (10)
OSULeaf	0.5334 ± 0.0245 (2)	0.5664 ± 0.0234 (4)	0.5863 ± 0.0637 (5)	0.639 ± 0.0379 (10)	0.6067 ± 0.0259 (7)	0.6363 ± 0.0274 (9)	0.5353 ± 0.0146 (3)	0.614 ± 0.0191 (8)	0.6436 ± 0.032 (11)	0.5894 ± 0.0224 (6)	**0.3798 (1)**
PhalangesOutlinesCorrect	**0.2418 ± 0.0062 (1)**	0.2741 ± 0.0058 (6)	0.2542 ± 0.0091 (4)	0.3274 ± 0.0031 (9)	0.2446 ± 0.0065 (3)	0.3409 ± 0.0058 (10)	0.2422 ± 0.0057 (2)	0.3219 ± 0.0048 (8)	0.319 ± 0.0044 (7)	0.2684 ± 0.0091 (5)	0.7408 (11)
Phoneme	0.8958 ± 0.0034 (3)	0.8755 ± 0.0048 (2)	0.896 ± 0.0078 (4)	0.9423 ± 0.0052 (11)	0.9223 ± 0.0054 (8)	0.9222 ± 0.008 (7)	0.9078 ± 0.004 (6)	0.9278 ± 0.0064 (9)	0.9282 ± 0.0049 (10)	0.9021 ± 0.0051 (5)	**0.067 (1)**

(Continua...)

Base de datos	eMODiTS	EP	SAX	αSAX	ESAX	ESAXKMeans	1D-SAX	RKMeans	SAXKMeans	rSAX	RAW
Plane	0.0426 ± 0.0163 (2)	**0.0247 ± 0.0131 (1)**	0.0929 ± 0.0212 (5)	0.1434 ± 0.01 (10)	0.0939 ± 0.0204 (6)	0.0875 ± 0.0114 (3)	0.089 ± 0.0252 (4)	0.1402 ± 0.0249 (9)	0.1268 ± 0.0149 (8)	0.1155 ± 0.0158 (7)	0.9045 (11)
ProximalPhalanxOutlineAgeGroup	**0.1288 ± 0.0203 (1)**	0.1658 ± 0.021 (3)	0.1735 ± 0.0175 (4)	0.2622 ± 0.0227 (10)	0.162 ± 0.0091 (2)	0.2099 ± 0.0325 (7)	0.1915 ± 0.0158 (5)	0.21 ± 0.0086 (8)	0.229 ± 0.0094 (9)	0.1953 ± 0.0127 (6)	0.7961 (11)
ProximalPhalanxOutlineCorrect	**0.1874 ± 0.0134 (1)**	0.2444 ± 0.0122 (6)	0.2066 ± 0.0046 (4)	0.2856 ± 0.0119 (8)	0.191 ± 0.012 (2)	0.2994 ± 0.0097 (10)	0.23 ± 0.0216 (5)	0.2475 ± 0.0163 (7)	0.2929 ± 0.012 (9)	0.1995 ± 0.0128 (3)	0.8001 (11)
ProximalPhalanxTW	**0.2039 ± 0.0206 (1)**	0.218 ± 0.0067 (3)	0.2315 ± 0.0094 (5)	0.4109 ± 0.0163 (10)	0.2357 ± 0.0121 (6)	0.2577 ± 0.0179 (7)	0.231 ± 0.0135 (4)	0.2662 ± 0.027 (8)	0.2829 ± 0.0083 (9)	0.2043 ± 0.0097 (2)	0.7361 (11)
RefrigerationDevices	0.4684 ± 0.0161 (2)	0.5779 ± 0.0098 (4)	0.5813 ± 0.0288 (5)	0.5698 ± 0.0099 (3)	0.5866 ± 0.017 (6)	0.636 ± 0.0314 (10)	0.6174 ± 0.0184 (8)	0.6347 ± 0.0128 (9)	0.6483 ± 0.0236 (11)	0.5953 ± 0.025 (7)	**0.4559 (1)**
ScreenType	0.5831 ± 0.0188 (9)	0.5433 ± 0.0207 (4)	0.6218 ± 0.0157 (11)	0.5373 ± 0.0177 (3)	0.5287 ± 0.0271 (2)	0.5914 ± 0.0187 (10)	0.5694 ± 0.0081 (6)	0.5809 ± 0.007 (8)	0.5506 ± 0.0145 (5)	0.5806 ± 0.0218 (7)	**0.404 (1)**
ShapeletSim	0.4796 ± 0.0352 (6)	0.4876 ± 0.0153 (7)	0.4586 ± 0.0154 (4)	0.4894 ± 0.0429 (8)	**0.3863 ± 0.066 (1)**	0.4346 ± 0.0239 (2)	0.5046 ± 0.0082 (10)	0.4348 ± 0.0458 (3)	0.5141 ± 0.043 (11)	0.4742 ± 0.0261 (5)	0.4941 (9)
ShapesAll	0.522 ± 0.0092 (7)	0.4996 ± 0.0064 (4)	0.5144 ± 0.0078 (5)	0.4914 ± 0.0093 (3)	0.5184 ± 0.0216 (6)	0.5742 ± 0.0079 (10)	0.5327 ± 0.0079 (8)	0.5822 ± 0.0094 (11)	0.5596 ± 0.0105 (9)	0.4909 ± 0.0164 (2)	**0.4748 (1)**
SmallKitchenAppliances	0.3405 ± 0.0118 (3)	0.4517 ± 0.0303 (7)	0.4104 ± 0.0136 (5)	0.4492 ± 0.0079 (6)	0.3205 ± 0.0168 (2)	0.5896 ± 0.0093 (9)	**0.2978 ± 0.0196 (1)**	0.6003 ± 0.0054 (10)	0.5326 ± 0.0107 (8)	0.3852 ± 0.0162 (4)	0.6139 (11)
SonyAIBORobotSurface1	0.2067 ± 0.0139 (8)	0.145 ± 0.0056 (5)	0.081 ± 0.013 (2)	0.2863 ± 0.0034 (10)	**0.08 ± 0.0053 (1)**	0.1682 ± 0.0149 (7)	0.1211 ± 0.0159 (4)	0.1662 ± 0.028 (6)	0.2246 ± 0.0195 (9)	0.1009 ± 0.0049 (3)	0.7249 (11)
SonyAIBORobotSurface2	0.1907 ± 0.0117 (7)	0.181 ± 0.0166 (4)	0.1701 ± 0.0151 (2)	0.1872 ± 0.0259 (5)	0.1771 ± 0.0158 (3)	0.1937 ± 0.0138 (8)	**0.1604 ± 0.012 (1)**	0.2536 ± 0.0103 (10)	0.2462 ± 0.0279 (9)	0.1899 ± 0.0134 (6)	0.7585 (11)
StarLightCurves	0.0821 ± 0.0017 (8)	0.1085 ± 0.0015 (9)	0.0709 ± 0.0033 (3)	0.1495 ± 0.0011 (10)	0.0783 ± 0.0022 (7)	0.0738 ± 0.0039 (5)	**0.0647 ± 0.0011 (1)**	0.0715 ± 0.0023 (4)	0.0652 ± 0.0018 (2)	0.0769 ± 0.0017 (6)	0.9081 (11)
Strawberry	0.0787 ± 0.0042 (2)	0.1313 ± 0.0112 (6)	0.1041 ± 0.0059 (5)	0.1959 ± 0.0079 (7)	**0.0771 ± 0.008 (1)**	0.2672 ± 0.012 (9)	0.0879 ± 0.0073 (3)	0.2214 ± 0.0104 (8)	0.2756 ± 0.0172 (10)	0.0896 ± 0.0068 (4)	0.9409 (11)
SwedishLeaf	0.368 ± 0.0213 (2)	**0.3528 ± 0.0087 (1)**	0.424 ± 0.0172 (5)	0.5463 ± 0.0039 (7)	0.3884 ± 0.0109 (3)	0.6584 ± 0.0102 (9)	0.4285 ± 0.0036 (6)	0.6718 ± 0.005 (11)	0.6697 ± 0.0099 (10)	0.4213 ± 0.0097 (4)	0.6344 (8)
Symbols	0.1493 ± 0.0086 (8)	0.0995 ± 0.0099 (3)	0.1511 ± 0.0077 (9)	0.1753 ± 0.0113 (10)	0.1346 ± 0.0084 (5)	0.1454 ± 0.0229 (7)	**0.0886 ± 0.0159 (1)**	0.1377 ± 0.0132 (6)	0.1296 ± 0.0098 (4)	0.0936 ± 0.0046 (2)	0.6818 (11)
SyntheticControl	**0.0867 ± 0.0142 (1)**	0.102 ± 0.02 (2)	0.1556 ± 0.0129 (4)	0.3959 ± 0.0168 (8)	0.2426 ± 0.0072 (6)	0.3536 ± 0.0279 (7)	0.1823 ± 0.0306 (5)	0.5436 ± 0.0333 (10)	0.4098 ± 0.0139 (9)	0.1153 ± 0.03 (3)	0.7787 (11)
ToeSegmentation1	0.4559 ± 0.0096 (10)	0.3731 ± 0.0407 (8)	0.3199 ± 0.0294 (3)	0.343 ± 0.0343 (5)	**0.2996 ± 0.0417 (1)**	0.3189 ± 0.0103 (2)	0.3902 ± 0.0351 (9)	0.3278 ± 0.0179 (4)	0.3524 ± 0.0233 (6)	0.3592 ± 0.0283 (7)	0.5556 (11)
ToeSegmentation2	**0.2351 ± 0.0233 (1)**	0.2816 ± 0.012 (7)	0.2742 ± 0.0215 (5)	0.2362 ± 0.0246 (2)	0.2743 ± 0.0204 (6)	0.3685 ± 0.0391 (10)	0.2457 ± 0.0092 (3)	0.3624 ± 0.0317 (9)	0.3081 ± 0.0305 (8)	0.2622 ± 0.0329 (4)	0.5621 (11)
Trace	0.0958 ± 0.0047 (4)	0.1154 ± 0.0265 (6)	0.2118 ± 0.02 (8)	0.4469 ± 0.0272 (10)	**0.0406 ± 0.0176 (1)**	0.0698 ± 0.0101 (2)	0.3195 ± 0.0196 (9)	0.1009 ± 0.0172 (5)	0.0743 ± 0.0153 (3)	0.1608 ± 0.0175 (7)	0.7979 (11)

(Continua...)

Base de datos	eMODiTS	EP	SAX	αSAX	ESAX	ESAXKMeans	1D-SAX	RKMeans	SAXKMeans	rSAX	RAW
TwoLeadECG	0.2787 ± 0.0317 (7)	0.3553 ± 0.0155 (8)	0.1404 ± 0.0115 (4)	0.393 ± 0.0149 (10)	**0.0659 ± 0.0039 (1)**	0.2532 ± 0.012 (5)	0.1311 ± 0.0043 (2)	0.2706 ± 0.0115 (6)	0.3736 ± 0.0084 (9)	0.1366 ± 0.0106 (3)	0.7286 (11)
TwoPatterns	0.1813 ± 0.0017 (2)	**0.1129 ± 0.0026 (1)**	0.4342 ± 0.004 (4)	0.5471 ± 0.0053 (10)	0.4536 ± 0.0133 (5)	0.5127 ± 0.008 (8)	0.5181 ± 0.0061 (9)	0.5065 ± 0.0023 (7)	0.5051 ± 0.0086 (6)	0.378 ± 0.0033 (3)	0.6323 (11)
UWaveGestureLibraryAll	0.1454 ± 0.0028 (5)	0.2295 ± 0.0033 (9)	0.1342 ± 0.0041 (2)	**0.1341 ± 0.0072 (1)**	0.145 ± 0.0038 (4)	0.2137 ± 0.0052 (8)	0.2341 ± 0.0057 (10)	0.2123 ± 0.004 (7)	0.1991 ± 0.005 (6)	0.1347 ± 0.0034 (3)	0.7824 (11)
UWaveGestureLibraryX	**0.3031 ± 0.0061 (1)**	0.3325 ± 0.0086 (3)	0.4739 ± 0.0035 (6)	0.3254 ± 0.0046 (2)	0.472 ± 0.0081 (5)	0.5772 ± 0.0055 (9)	0.514 ± 0.0062 (7)	0.5846 ± 0.0026 (10)	0.571 ± 0.0035 (8)	0.4702 ± 0.0026 (4)	0.622 (11)
UWaveGestureLibraryY	0.3913 ± 0.0066 (2)	**0.3806 ± 0.0052 (1)**	0.5173 ± 0.0082 (5)	0.4149 ± 0.0063 (3)	0.526 ± 0.0078 (6)	0.5815 ± 0.0039 (9)	0.6505 ± 0.0135 (11)	0.5838 ± 0.0095 (10)	0.5748 ± 0.0068 (8)	0.5161 ± 0.0035 (4)	0.5677 (7)
UWaveGestureLibraryZ	0.3804 ± 0.0081 (3)	**0.3758 ± 0.0147 (1)**	0.4958 ± 0.0077 (5)	0.3796 ± 0.003 (2)	0.488 ± 0.0101 (4)	0.5419 ± 0.0105 (8)	0.6128 ± 0.0044 (11)	0.5789 ± 0.0092 (10)	0.5395 ± 0.0111 (7)	0.4968 ± 0.0022 (6)	0.5736 (9)
Wafer	0.0042 ± 0.0013 (2)	**0.0029 ± 0.001 (1)**	0.0132 ± 0.001 (7)	0.0051 ± 0.0002 (3)	0.0131 ± 0.0013 (6)	0.0319 ± 0.0043 (9)	0.0125 ± 0.0012 (5)	0.0209 ± 0.0011 (8)	0.0368 ± 0.0019 (10)	0.0097 ± 0.0007 (4)	0.9757 (11)
Wine	0.3599 ± 0.0303 (7)	0.2111 ± 0.0245 (2)	0.2388 ± 0.0442 (5)	0.3449 ± 0.056 (6)	**0.1375 ± 0.0212 (1)**	0.5189 ± 0.0481 (10)	0.2167 ± 0.061 (4)	0.4865 ± 0.0324 (9)	0.463 ± 0.0482 (8)	0.2153 ± 0.0295 (3)	0.7565 (11)
WordSynonyms	0.5149 ± 0.0264 (3)	0.4992 ± 0.0223 (2)	0.5885 ± 0.0071 (9)	0.5381 ± 0.0152 (5)	0.545 ± 0.0223 (6)	0.5921 ± 0.0106 (10)	0.5842 ± 0.0065 (8)	0.6139 ± 0.0189 (11)	0.5787 ± 0.0123 (7)	0.5275 ± 0.0133 (4)	**0.374 (1)**
Worms	0.5504 ± 0.0336 (8)	0.6282 ± 0.0473 (11)	0.5413 ± 0.0193 (6)	0.5605 ± 0.0423 (9)	0.5224 ± 0.0159 (3)	0.5966 ± 0.0326 (10)	0.5336 ± 0.0218 (4)	0.5198 ± 0.0376 (2)	0.544 ± 0.0336 (7)	0.5378 ± 0.0062 (5)	**0.4069 (1)**
WormsTwoClass	0.4589 ± 0.1056 (9)	0.4336 ± 0.025 (7)	0.4594 ± 0.0876 (10)	0.4122 ± 0.0207 (5)	0.4258 ± 0.0894 (6)	0.3574 ± 0.066 (2)	0.4345 ± 0.0691 (8)	**0.3465 ± 0.0319 (1)**	0.3641 ± 0.0368 (3)	0.3743 ± 0.0273 (4)	0.6004 (11)
Yoga	0.2044 ± 0.0077 (7)	0.1732 ± 0.0049 (5)	0.1638 ± 0.0084 (3)	0.1766 ± 0.0085 (6)	**0.1555 ± 0.0023 (1)**	0.2427 ± 0.0035 (10)	0.167 ± 0.0027 (4)	0.2387 ± 0.0026 (9)	0.2135 ± 0.0081 (8)	0.1603 ± 0.0037 (2)	0.727 (11)

Por su parte, la Figura 5.3 muestra los resultados estadísticos de la comparación de todos los enfoques usando la prueba de Friedman y Nemenyi.

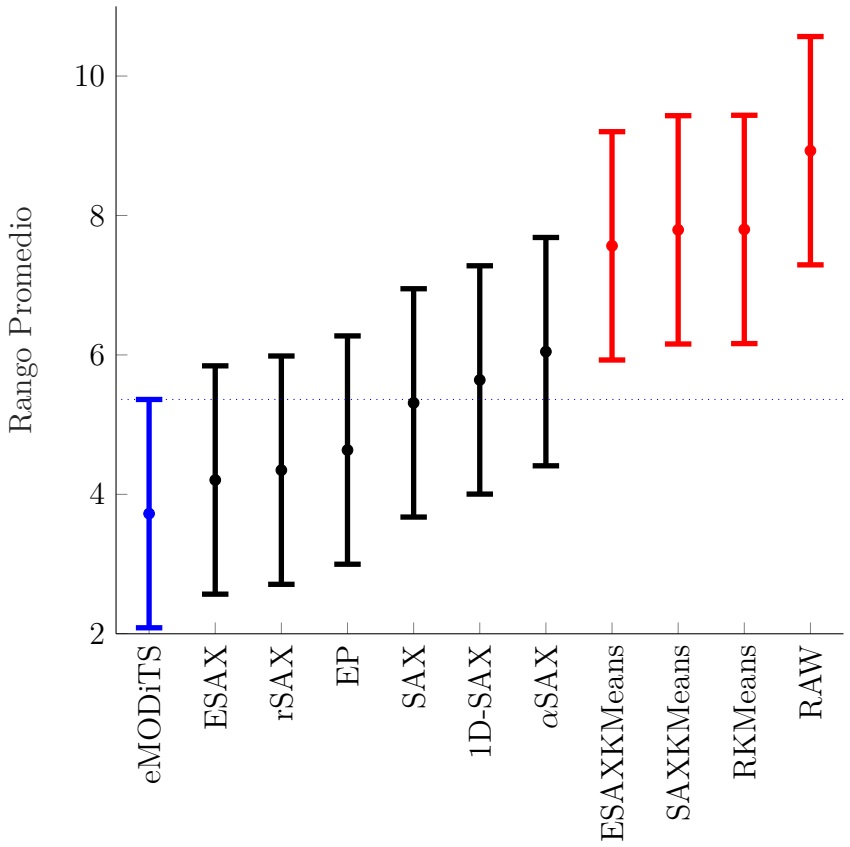


Figura 5.3: Resultados obtenidos al aplicar la prueba estadística post hoc de Nemenyi usando los mejores valores encontrados por cada método en cada base de datos.

Sin embargo, era de nuestro interés analizar a fondo los casos en los que eMODiTS era estadísticamente similar a otros métodos; por ello, se utilizó la prueba estadística de suma de rango por pares de Wilcoxon, para comparar nuestra propuesta con cada uno de los métodos que no arrojaron diferencias significativas en la prueba estadística. Para aplicar la prueba de Wilcoxon, se tomaron las valores mínimos de clasificación por cada base de datos en cada método, es decir, este análisis se hizo a nivel de base de datos y no a nivel general por método. Los resultados se presentan en la Figura 5.4.

Reducción de la dimensionalidad

La principal meta de los algoritmos de discretización de series de tiempo es la *reducción de la dimensionalidad* del conjunto temporal. Por lo tanto, en este apartado se muestra la reducción de la dimensionalidad alcanzada por eMODiTS y los enfoques basados en SAX. Dado que estos últimos utilizan el mismo número de cortes de palabras para discretizar una base de datos temporal, fueron agrupados en un sólo rubro para la comparación.

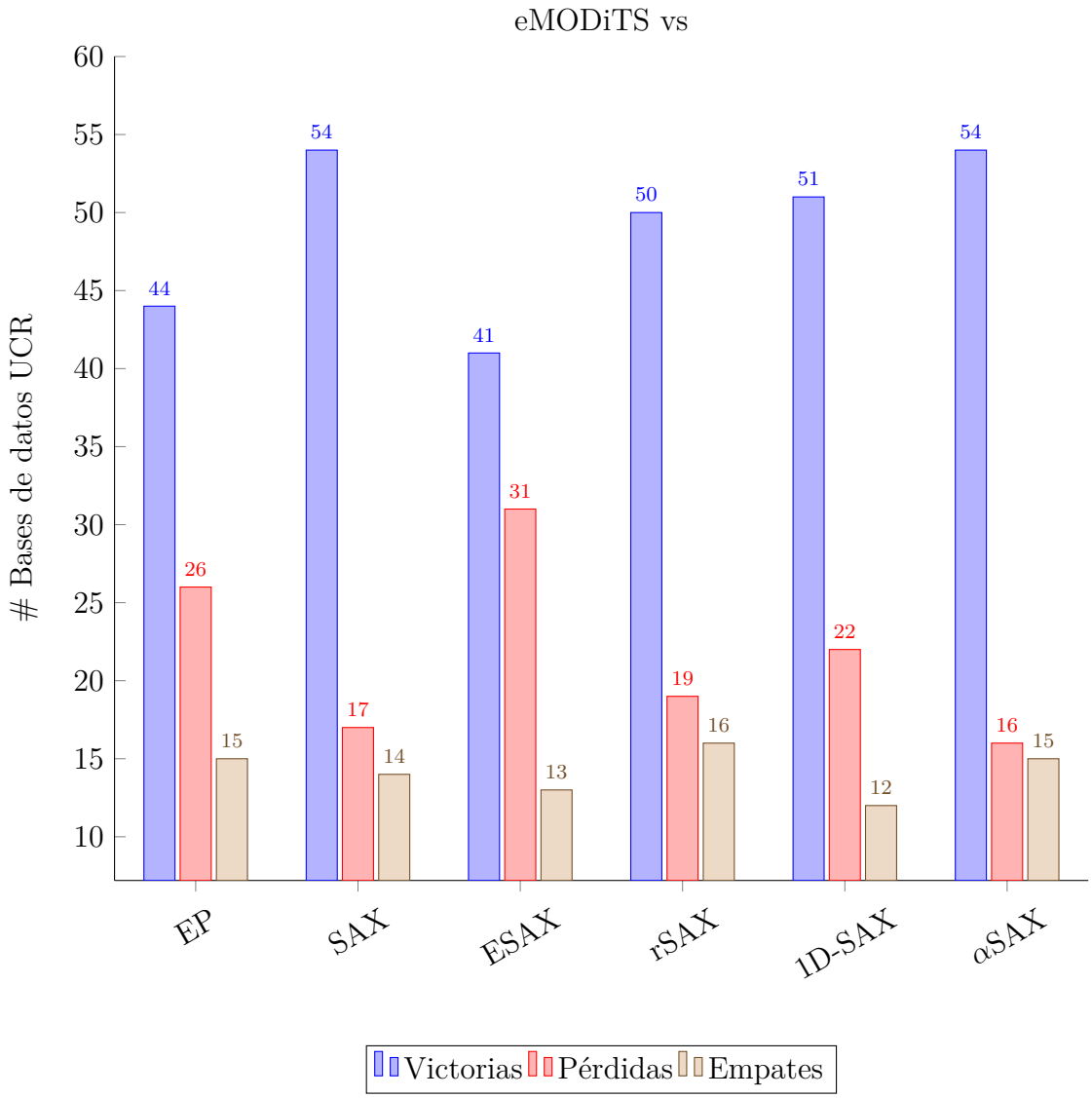


Figura 5.4: Resultados estadísticos obtenidos al aplicar la prueba de suma de rangos por pares de Wilcoxon con un 95 % de confianza, comparando eMODiTS en contra de EP, SAX, ESAX, rSAX, 1D-SAX y αSAX.

La Figura 5.5 muestra la tasa de reducción alcanzada por nuestra propuesta y esos métodos, la cual es calculada usando la Ecuación 5.6, donde m es el número de segmentos de palabra (longitud de la serie de tiempo discreta), y n es el tamaño de la serie de tiempo original. La valores obtenidos por dicha métrica varían entre 0 y 1, donde valores cercanos a cero representan una tasa de reducción alta y valores cercanos a uno representan bajos porcentajes de reducción.

$$TasaCompresion = \frac{m}{n} \tag{5.6}$$

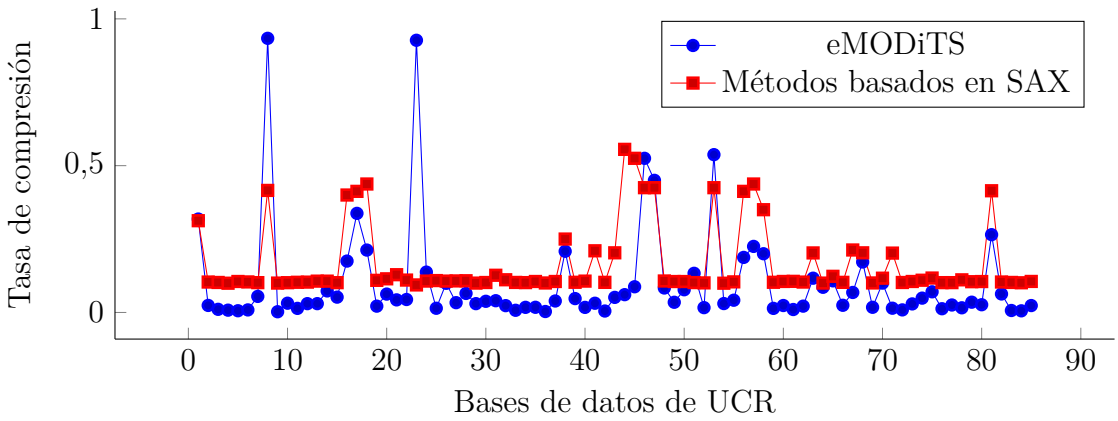


Figura 5.5: Tasas de compresión obtenidas por eMODiTS y los métodos basados en SAX en cada una de las bases de datos del repositorio UCR.

Pérdida de Información

Aún cuando la meta de los algoritmos de discretización es alcanzar una tasa alta de compresión o reducción de la dimensionalidad, ésta suele representar un problema mayor: incurrir en pérdida de información importante.

Por lo tanto, se realizó una estimación de la pérdida de la información incurrida por cada método comparado usando una métrica de similitud de la serie de tiempo original y la serie de tiempo discreta reconstruida. Para realizar la reconstrucción de la serie temporal se sustituyó cada dato continuo de la serie temporal por su correspondiente símbolo [93]. Cabe destacar que, para tener una comparación justa, se escalaron ambas series dentro del intervalo $[0, 1]$.

La métrica de similitud empleada fue la métrica de la *Subsecuencia Común más Larga (Longest Common Subsequence, LCSS)*, la cual fue descrita en la Sección 2.3.3. Este método permite comparaciones entre series de tiempo de diferente tamaño, así como, evitar relaciones de regiones atípicas con regiones de la serie comparada; es decir, LCSS es un método robusto al ruido o datos atípicos [100, 34].

La estimación de la pérdida de la información fue aplicada usando las series de tiempo promedio obtenidas tanto del conjunto temporal original como del conjunto reconstruido. Una serie temporal promedio es el promedio de todas las series de tiempo del conjunto de datos. La Tabla 5.5 muestra los resultados obtenidos de comparar la serie de tiempo promedio original y la serie de tiempo promedio reconstruida, donde los valores iguales o cercanos a cero representan similitud entre ambas series, es decir, pérdida de información mínima. Mientras que, valores cercanos o iguales a uno indican que se presenta una pérdida sustancial de información cuando la serie original fue discretizada por el método correspondiente.

Tabla 5.5: Pérdida de la información calculada mediante el método LCSS donde los valores mínimos representan similitud entre la base de datos original y la reconstruida, mientras que valores altos representan que ambos conjuntos son desiguales entre sí. Los números entre paréntesis representan el rango calculado por la prueba estadística multi-comparador y los números en negritas representan los valores mínimos encontrados en cada base de datos.

Base de datos	eMODiTS	EP	SAX	αSAX	ESAX	ESAXK-Means	1D-SAX	SAXK-Means	rSAX
Adiac	0.0114(4)	0.267(8)	**0.0057(2)**	0.1193(7)	0.0511(5)	0.0568(6)	0.3068(9)	**0.0057(2)**	**0.0057(2)**
ArrowHead	0.4223(8)	0.6614(9)	**0(2.5)**	0.0159(6)	0.004(5)	**0(2.5)**	0.4143(7)	**0(2.5)**	**0(2.5)**
Beef	0.3277(7)	0.8021(9)	0.0638(4)	0.4255(8)	0.066(5)	0.0106(2)	0.0191(3)	**0.0064(1)**	0.1149(6)
BeetleFly	0.4395(5)	1(9)	0.5137(7)	0.4766(6)	0.3691(4)	0.1289(3)	0.0332(2)	**0.0254(1)**	0.5234(8)
BirdChicken	0.7285(8)	1(9)	0.1934(7)	0.1875(5)	0.1914(6)	0.0195(3)	**0.0117(1)**	0.0137(2)	0.0488(4)
Car	0.3934(8)	0.5321(9)	**0.0017(3.5)**	0.0468(7)	**0.0017(3.5)**	**0.0017(3.5)**	**0.0017(3.5)**	**0.0017(3.5)**	**0.0017(3.5)**
CBF	0.1094(3)	0.9531(9)	0.1563(4.5)	0.2422(8)	0.2188(7)	0.2031(6)	0.0156(2)	**0.0078(1)**	0.1563(4.5)
ChlorineConcentration	0.3012(3)	0.9759(9)	0.4639(7)	0.3253(4)	0.4337(6)	0.1084(2)	0.4217(5)	**0.0422(1)**	0.4699(8)
CinCECGtorso	0.9994(3)	1(6.5)	1(6.5)	1(6.5)	1(6.5)	0.7694(2)	**0.2441(1)**	1(6.5)	1(6.5)
Coffee	0.0105(2)	0.6783(9)	0.028(5)	0.4021(8)	0.0315(6)	0.014(3.5)	0.014(3.5)	**0(1)**	0.035(7)
Computers	0.9472(5)	0.8931(4)	1(7.5)	0.6792(3)	1(7.5)	**0.1861(1)**	1(7.5)	0.2083(2)	1(7.5)
CricketX	0.6067(3)	1(7.5)	1(7.5)	0.87(4)	0.9433(5)	0.56(2)	1(7.5)	**0.07(1)**	1(7.5)
CricketY	0.9967(8)	1(9)	0.22(3)	0.6333(5)	0.7167(6)	0.4167(4)	0.7467(7)	**0.0033(1)**	0.06(2)
CricketZ	0.9467(5)	1(8)	1(8)	0.9067(4)	0.99(6)	0.4733(2)	0.8067(3)	**0.25(1)**	1(8)
DiatomSizeReduction	**0.0087(1)**	0.9536(9)	0.1101(3.5)	0.1507(6)	0.1101(3.5)	0.1913(8)	0.0261(2)	0.1826(7)	0.1275(5)
DistalPhalanxOutlineAgeGroup	0.125(7)	0.65(9)	0.0125(2.5)	0.05(4)	0.1(6)	0.0875(5)	0.3875(8)	**0(1)**	0.0125(2.5)
DistalPhalanxOutlineCorrect	0.0375(5)	0.3625(9)	**0(2.5)**	0.0625(6)	0.1125(8)	0.1(7)	**0(2.5)**	**0(2.5)**	**0(2.5)**
DistalPhalanxTW	0.3(7)	0.5375(9)	0.0125(2.5)	0.0375(4)	0.175(6)	0.1625(5)	0.4(8)	**0(1)**	0.0125(2.5)
Earthquakes	0.9121(4)	0.9102(3)	1(7)	1(7)	1(7)	**0.627(1)**	1(7)	0.7402(2)	1(7)
ECG200	0.6667(9)	0.625(8)	0.2708(4)	0.5625(7)	0.2813(5)	0.0833(2)	0.25(3)	**0.0625(1)**	0.2917(6)
ECG5000	0.8786(8)	0.9429(9)	0.3857(5)	0.5143(7)	0.3857(5)	0.0429(2)	0.3857(5)	**0.0357(1)**	0.3429(3)
ECGFiveDays	0.9044(9)	0.7794(5.5)	0.7794(5.5)	0.8088(8)	0.6985(3)	0.0882(2)	0.7353(4)	**0.0809(1)**	0.7868(7)
ElectricDevices	0.8542(3)	1(7)	1(7)	0.8854(4)	1(7)	0.3958(2)	1(7)	**0.0833(1)**	1(7)
FaceAll	0.7405(9)	0.6412(8)	0.3206(3)	0.4809(7)	0.4351(5)	0.4427(6)	0.3206(3)	**0.2214(1)**	0.3206(3)
FaceFour	0.3486(4)	0.9971(9)	0.6086(7)	0.48(5)	0.5429(6)	0.2286(3)	0.2257(2)	**0.1029(1)**	0.8286(8)
FacesUCR	0.626(8)	0.8779(9)	0.229(3)	0.542(7)	0.5267(6)	0.4885(5)	0.4504(4)	0.2137(2)	**0.2061(1)**
FiftyWords	0.4889(4)	0.3778(3)	0.8963(8)	0.7926(5)	0.9037(9)	0.1667(2)	0.8259(6)	**0.0556(1)**	0.863(7)
Fish	0.2354(8)	0.8272(9)	**0.0022(3.5)**	0.0994(7)	**0.0022(3.5)**	**0.0022(3.5)**	**0.0022(3.5)**	**0.0022(3.5)**	**0.0022(3.5)**
FordA	0.53(2)	1(7)	1(7)	1(7)	0.962(4)	1(7)	0.782(3)	**0.176(1)**	1(7)
FordB	0.998(4)	1(7)	1(7)	1(7)	0.878(2)	1(7)	0.942(3)	**0.144(1)**	1(7)
GunPoint	0.04(2)	0.7267(9)	0.14(4)	0.1267(3)	0.1533(5)	0.2467(6)	**0.0067(1)**	0.2933(7)	0.4533(8)
Ham	0.7239(9)	0.5267(4)	0.6984(8)	0.3619(3)	0.6381(6.5)	**0.0719(1)**	0.6102(5)	0.0742(2)	0.6381(6.5)
HandOutlines	0.4614(9)	0.0893(7)	0.0712(3)	0.1462(8)	0.0712(3)	0.0834(6)	**0.0277(1)**	0.0831(5)	0.0712(3)
Haptics	0.4249(6)	0.837(9)	0.1822(2)	0.4643(8)	0.1832(3)	0.4002(5)	0.2179(4)	0.4286(7)	**0.1658(1)**
Herring	0.3867(8)	0.8789(9)	0.0098(5)	0.1289(7)	0.0039(3.5)	**0.002(1.5)**	0.0039(3.5)	**0.002(1.5)**	0.0137(6)
InlineSkate	0.3438(8)	0.9028(9)	0.1722(7)	0.1429(4)	0.1647(5)	**0.0005(2)**	**0.0005(2)**	**0.0005(2)**	0.1663(6)
InsectWingbeatSound	**0.4961(1)**	0.9922(5)	1(7.5)	0.9375(4)	1(7.5)	0.6445(2)	1(7.5)	0.7617(3)	1(7.5)
ItalyPowerDemand	0.2083(8)	0.9167(9)	**0.0417(2.5)**	0.0833(5)	0.125(6.5)	0.125(6.5)	**0.0417(2.5)**	**0.0417(2.5)**	**0.0417(2.5)**
LargeKitchenAppliances	0.7514(2)	1(7)	1(7)	0.9542(4)	1(7)	**0.675(1)**	1(7)	0.8278(3)	1(7)
Lighting2	0.5667(3)	0.9717(5)	1(7.5)	0.5918(4)	1(7.5)	0.0612(2)	1(7.5)	**0.0031(1)**	1(7.5)
Lighting7	0.7335(4)	0.8997(5)	1(8)	0.6207(3)	1(8)	0.2476(2)	0.9875(6)	**0.1787(1)**	1(8)
Mallat	0.3135(8)	0.4609(9)	0.1514(4)	0.2559(5)	0.1152(3)	0.293(7)	0.0449(2)	0.2842(6)	**0.0371(1)**
Meat	0.0022(2)	0.0603(4)	0.1496(7)	0.442(8)	0.1406(6)	0.0134(3)	0.596(9)	**0(1)**	0.1384(5)

(Continua...)

Base de datos	eMODiTS	EP	SAX	αSAX	ESAX	ESAXK-Means	1D-SAX	SAXK-Means	rSAX
MedicalImages	0.3838(4)	0.5455(5)	0.7374(6)	0.1616(2.5)	0.9091(8)	0.1616(2.5)	0.9192(9)	**0.0101(1)**	0.798(7)
MiddlePhalanxOutlineAgeGroup	0.275(5.5)	0.275(5.5)	0.0125(2.5)	0.05(4)	0.375(9)	0.3625(7.5)	0.3625(7.5)	**0(1)**	0.0125(2.5)
MiddlePhalanxOutlineCorrect	0.0125(4)	0.025(5)	**0(2)**	0.05(6)	0.1375(8)	0.125(7)	0.3125(9)	**0(2)**	**0(2)**
MiddlePhalanxTW	0.0625(6)	0.3125(9)	**0(2.5)**	0.05(5)	0.1375(8)	0.125(7)	**0(2.5)**	**0(2.5)**	**0(2.5)**
MoteStrain	0.4524(8)	0.9881(9)	**0.0119(2)**	0.2857(7)	**0.0119(2)**	**0.0119(2)**	0.0357(6)	0.0238(4.5)	0.0238(4.5)
NonInvasiveFetalECGThorax1	0.5347(8)	0.468(7)	0.1547(4)	0.7827(9)	0.1453(3)	0.0093(2)	0.212(6)	**0.008(1)**	0.1693(5)
NonInvasiveFetalECGThorax2	0.3547(6)	0.5453(8)	0.1507(4.5)	0.772(9)	0.1467(3)	0.0093(2)	0.5(7)	**0.008(1)**	0.1507(4.5)
OliveOil	0.0333(3.5)	0.0333(3.5)	0.3175(8)	0.3421(9)	0.2526(5)	**0.0123(1)**	0.2754(7)	0.0175(2)	0.2614(6)
OSULeaf	0.7939(8)	0.8525(9)	0.7377(7)	0.6534(6)	0.5667(5)	0.1452(4)	0.0304(2)	**0.0164(1)**	0.0749(3)
PhalangesOutlinesCorrect	0.075(5)	0.275(8)	**0(2)**	0.05(4)	0.1375(7)	0.125(6)	0.4(9)	**0(2)**	**0(2)**
Phoneme	0.916(6)	1(9)	0.9053(5)	0.5752(2)	0.9932(8)	0.9893(7)	0.7656(4)	**0.0127(1)**	0.6016(3)
Plane	0.4861(8)	0.5903(9)	0.2639(6)	0.25(5)	0.2222(4)	0.0764(2)	0.0972(3)	**0.0556(1)**	0.3056(7)
ProximalPhalanxOutlineAgeGroup	0.3125(8)	0.0125(4)	**0(2)**	0.1125(6.5)	0.1125(6.5)	0.1(5)	0.4625(9)	**0(2)**	**0(2)**
ProximalPhalanxOutlineCorrect	0.0375(4)	0.05(5)	**0(2)**	0.1(6)	0.175(8)	0.15(7)	0.2375(9)	**0(2)**	**0(2)**
ProximalPhalanxTW	0.1125(8)	0.025(5.5)	0.0125(3)	0.0875(7)	0.025(5.5)	0.0125(3)	0.375(9)	**0(1)**	0.0125(3)
RefrigerationDevices	**0.9458(1)**	0.9903(2)	1(6)	1(6)	1(6)	1(6)	1(6)	1(6)	1(6)
ScreenType	0.4167(4)	0.8833(5)	0.9722(8)	0.1(3)	0.9583(6)	0.0139(2)	1(9)	**0.0014(1)**	0.9611(7)
ShapeletSim	0.994(5)	1(8)	0.998(6)	0.85(3)	0.636(2)	0.87(4)	1(8)	**0.56(1)**	1(8)
ShapesAll	0.4023(8)	1(9)	0.0352(4)	0.1328(6)	0.1055(5)	**0.002(1.5)**	0.0059(3)	**0.002(1.5)**	0.2441(7)
SmallKitchenAppliances	**0.9556(1)**	1(5.5)	1(5.5)	1(5.5)	1(5.5)	1(5.5)	1(5.5)	1(5.5)	1(5.5)
SonyAIBORobotSurface1	**0.1571(1.5)**	0.9286(9)	0.1857(4)	0.6(8)	0.2571(5)	0.1714(3)	0.3286(7)	0.3(6)	**0.1571(1.5)**
SonyAIBORobotSurface2	**0.2923(1.5)**	0.9692(9)	0.7538(7.5)	0.3231(3)	0.3385(4)	**0.2923(1.5)**	0.7231(6)	0.7077(5)	0.7538(7.5)
StarLightCurves	0.4053(7)	0.9121(9)	**0.001(3)**	0.0049(6)	**0.001(3)**	**0.001(3)**	0.4268(8)	**0.001(3)**	**0.001(3)**
Strawberry	0.3021(8)	0.1064(3)	0.2553(4)	0.3447(9)	0.2936(6.5)	0.0255(2)	0.2596(5)	**0(1)**	0.2936(6.5)
SwedishLeaf	0.2031(7)	0.4844(9)	0.0391(4.5)	0.1563(6)	0.0391(4.5)	0.0156(2.5)	0.3594(8)	**0.0078(1)**	0.0156(2.5)
Symbols	0.206(4)	0.6683(8)	0.3643(6)	0.4397(7)	0.3492(5)	0.0352(3)	**0.0151(1)**	0.0276(2)	0.7136(9)
SyntheticControl	0.9833(8)	1(9)	0.75(7)	0.3833(2)	0.65(5)	0.6667(6)	0.4(3)	0.4833(4)	**0.3(1)**
ToeSegmentation1	0.6245(2)	1(6.5)	1(6.5)	1(6.5)	1(6.5)	**0.574(1)**	1(6.5)	0.9856(3)	1(6.5)
ToeSegmentation2	0.9971(3)	1(6.5)	1(6.5)	1(6.5)	1(6.5)	**0.3003(1)**	1(6.5)	0.6327(2)	1(6.5)
Trace	0.6291(7)	0.8545(9)	**0.0036(2.5)**	0.3018(6)	**0.0036(2.5)**	**0.0036(2.5)**	0.7527(8)	**0.0036(2.5)**	0.1127(5)
TwoLeadECG	0.1463(7)	0.7439(9)	0.0976(5.5)	0.2439(8)	0.0854(3.5)	**0.061(1)**	0.0976(5.5)	0.0732(2)	0.0854(3.5)
TwoPatterns	0.9063(5)	1(8)	1(8)	1(8)	0.5625(3)	0.9766(6)	**0.25(1)**	0.2578(2)	0.6328(4)
UWaveGestureLibraryAll	0.8053(4)	1(9)	0.9894(7.5)	0.9894(7.5)	0.9746(6)	0.3439(2)	**0.1513(1)**	0.4836(3)	0.9164(5)
UWaveGestureLibraryX	0.9968(8)	1(9)	0.0095(4)	0.0635(5)	0.1714(7)	0.0984(6)	**0.0032(2)**	**0.0032(2)**	**0.0032(2)**
UWaveGestureLibraryY	0.6413(8)	1(9)	0.4667(6)	0.3016(5)	0.2667(4)	0.0571(2)	0.2571(3)	**0.0032(1)**	0.5397(7)
UWaveGestureLibraryZ	0.7556(8)	1(9)	0.2222(6)	0.3524(7)	0.1873(5)	0.019(2)	0.1365(4)	**0.0032(1)**	0.0254(3)
Wafer	0.4605(8)	0.9539(9)	0.2829(6)	0.3882(7)	0.2237(4)	0.0987(2)	0.2171(3)	**0.0395(1)**	0.25(5)
Wine	0.0897(2.5)	0.2137(8)	0.1026(5)	0.8291(9)	0.1496(6)	0.1838(7)	0.0897(2.5)	**0(1)**	0.094(4)
WordSynonyms	**0.2185(1)**	1(8.5)	0.9296(7)	0.8593(4)	0.9259(6)	0.4741(2)	0.9(5)	0.6593(3)	1(8.5)
Worms	0.58(4)	1(8.5)	0.9889(7)	0.9378(6)	0.8911(5)	0.2311(3)	**0.09(1)**	0.1956(2)	1(8.5)
WormsTwoClass	0.9989(7)	1(8.5)	1(8.5)	0.99(5)	0.9956(6)	0.2133(2)	**0.2011(1)**	0.3667(3)	0.9889(4)
Yoga	0.3826(8)	0.9225(9)	**0.0023(3)**	0.0587(7)	0.007(6)	**0.0023(3)**	**0.0023(3)**	**0.0023(3)**	**0.0023(3)**
Rango Promedio	**5.4647**	**7.5294**	**5.1588**	**5.8294**	**5.4176**	**3.5588**	**4.8529**	**2.2353**	**4.9529**

Por su parte, las Figuras 5.6a y 5.6b muestran los resultados gráficamente del rango obtenido por cada enfoque comparado y la aplicación de la prueba estadística de Friedman con la prueba post hoc de Nemenyi, respectivamente.

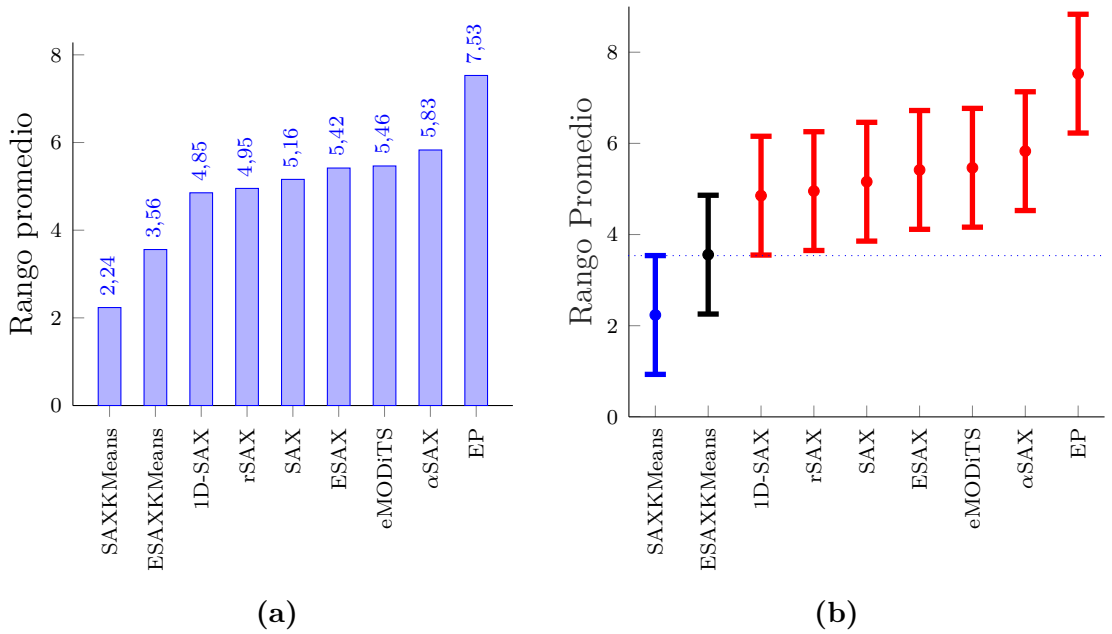


Figura 5.6: Resultados obtenidos por cada método al analizar la pérdida de información incurrida en cada base de datos temporal, presentando (a) los rangos promedios obtenidos por cada enfoque, y (b) los resultados estadísticos al aplicar la prueba post hoc de Nemenyi.

Análisis de las características de las bases de datos temporales

Otro estudio realizado en este trabajo fue el análisis de las características más relevantes de las bases de datos temporales, con la finalidad de descubrir las fortalezas y debilidades de nuestra propuesta.

Para ello, se analizó específicamente la proximidad, el balanceo de clases y la intensidad de ruido presente, cada una de estas características fueron detalladas en la Sección 5.2 y obtenidas de la Tabla 5.2. La Figura 5.7a muestra el rango promedio alcanzado por cada método considerando bases de datos balanceadas y no balanceadas; mientas que, para bases de datos con clases traslapadas o separadas se presenta la Figura 5.7b, y finalmente, la Figura 5.7c ilustra el comportamiento de cada enfoque dependiendo de la intensidad de ruido presente en las bases de datos temporales.

Tiempo computacional

Un aspecto necesario que se debe estudiar cuando se introduce un nuevo enfoque al estado del arte, es el tiempo de cómputo ocupado para obtener el resultado deseado. Este aspecto es de vital importancia cuando se emplean meta-heurísticas como herramientas de búsqueda, ya que, por lo general, suelen ser criticadas por

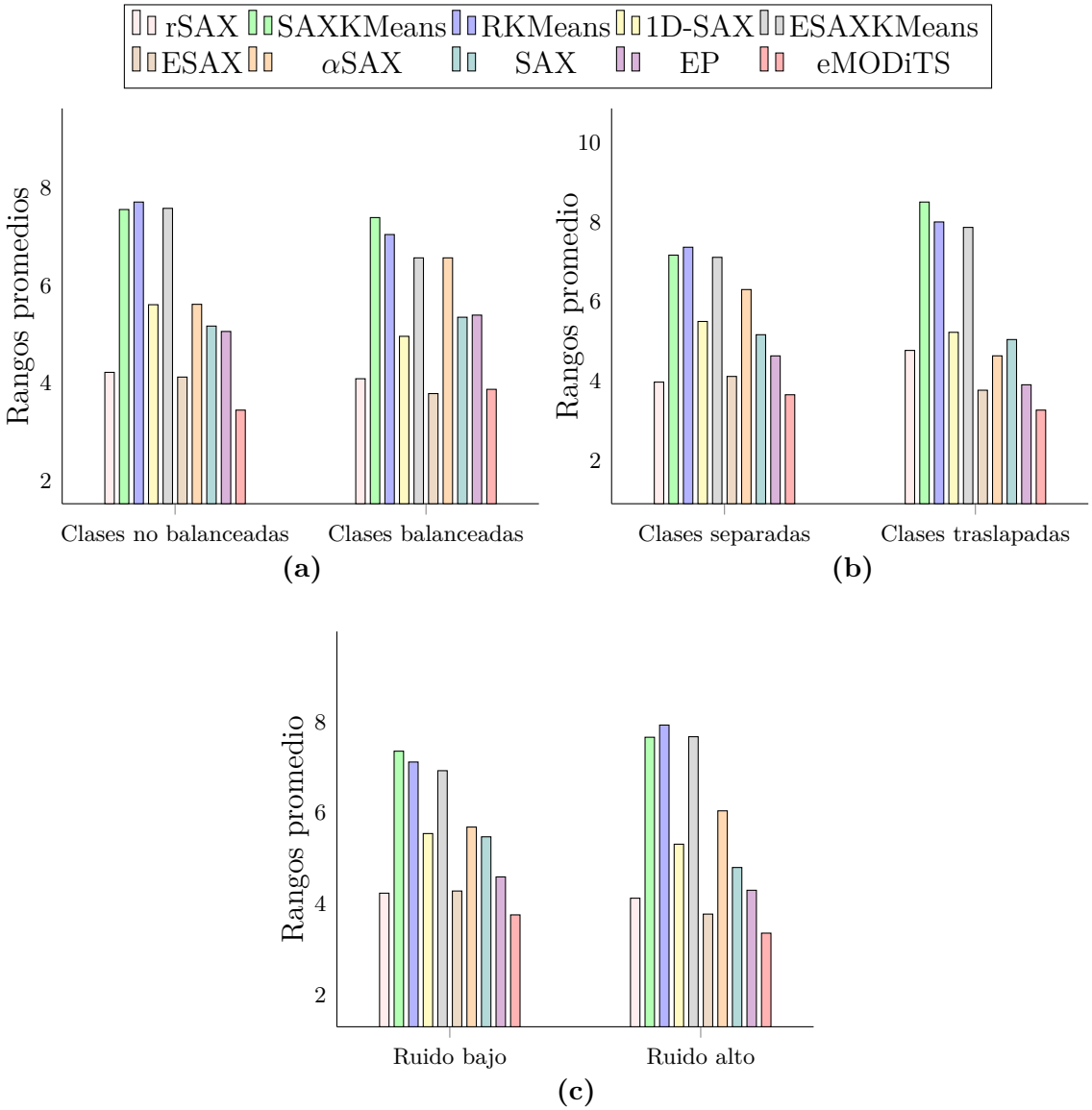


Figura 5.7: Rangos promedios logrados por cada método basados en SAX en 85 bases de datos temporales para (a) clases balanceadas y no balanceadas, (b) clases separadas y traslapadas, (c) intensidad de ruido.

ocupar un tiempo considerable para encontrar soluciones a problemas reales. Esto debido a que son poblacionales y que toman decisiones con elementos aleatorios.

Dado que nuestro enfoque utiliza una meta-heurística como herramienta de búsqueda de esquemas de discretización, fue necesario medir el tiempo ocupado en cada base de datos para encontrar una solución. La Figura 5.8 muestra el comportamiento de eMODiTS cuando la dimensionalidad de los datos aumenta. Dicha dimensionalidad es calculada multiplicando el tamaño de las series temporales por el número de instancias que tiene la base temporal.

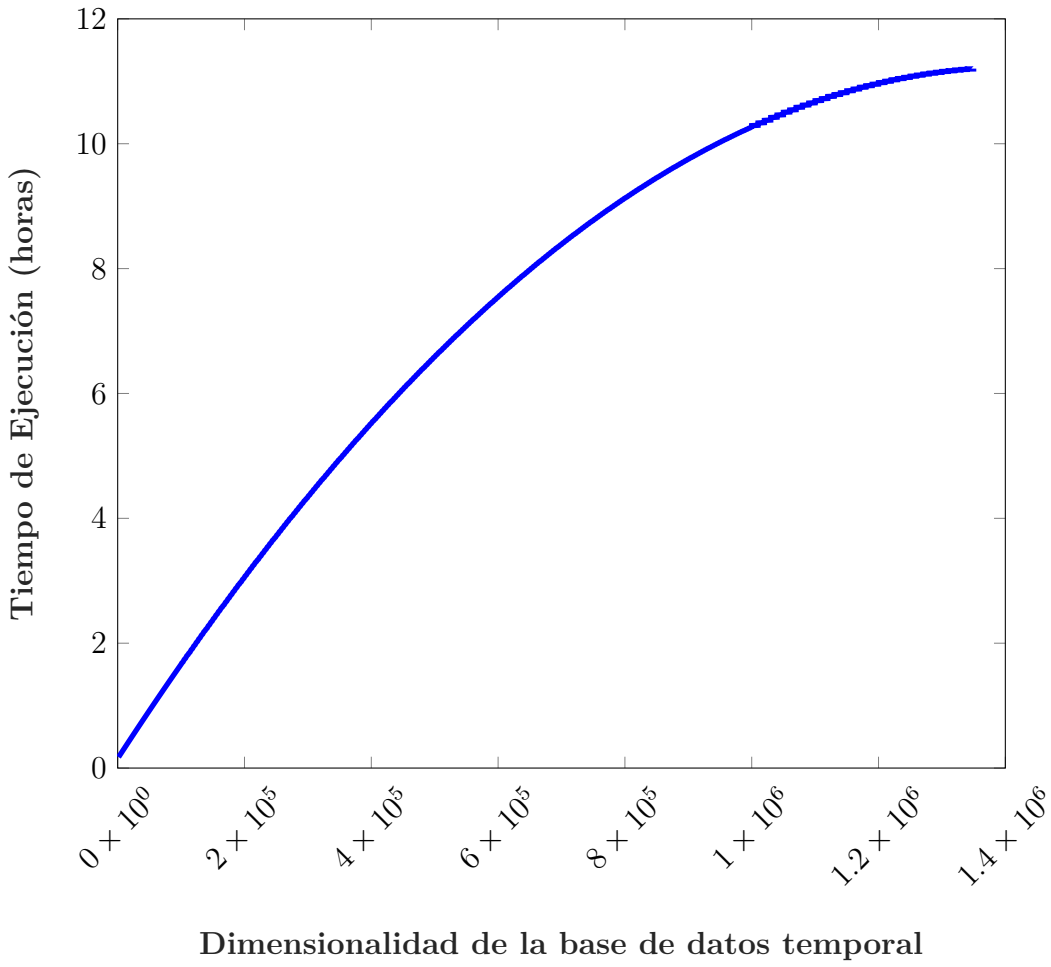


Figura 5.8: Tiempo empleado por eMODiTS en cada base de datos temporal. La dimensionalidad de las series de tiempo es calculada multiplicando el tamaño de las series de tiempo por el número de series en el conjunto de datos.

5.4.3. Discusión

En esta sección se presentaron los resultados obtenidos por cada método de discretización simbólica comparado, los cuales fueron rSAX, SAXKMeans, RKMeans, 1D-SAX, ESAXKMeans, ESAX, αSAX, SAX, EP, y nuestra propuesta llamada eMODiTS. En cada uno de ellos se analizó la tasa de error en clasificación obtenida, la tasa de compresión alcanzada, y la pérdida de información incurrida. Los resultados obtenidos por cada uno fueron ejecutados sin dar ventaja a ninguno de los enfoques, es decir, se ejecutaron bajo circunstancias iguales y justas.

A continuación se analizarán los resultados obtenidos basados en las tres características mencionadas en el párrafo anterior: Evaluación del desempeño de nuestra propuesta, reducción de la dimensionalidad y pérdida de la información.

Evaluación del desempeño de nuestra propuesta

En este rubro se analizará el desempeño de eMODiTS en términos de clasificación de las series temporales. En la Tabla 5.4 se presentaron las tasas de error en clasificación mínimas obtenidas por cada enfoque, así como el rango alcanzado de acuerdo a dichos porcentajes. En esta tabla se puede apreciar como eMODiTS encuentra las tasas de error en clasificación más bajas en comparación con los otros nueve enfoques comparados, lo que sugiere que nuestra propuesta es capaz de mejorar el error en clasificación en bases de datos temporales. Gráficamente, esto puede apreciarse en la Figura 5.2 donde se observa que eMODiTS tiene el rango promedio más bajo de los 10 enfoques de discretización simbólica.

Sin embargo, al aplicar una prueba estadística multi-comparador, en este caso la prueba de Friedman, se observa que eMODiTS logra superar estadísticamente a tres métodos de los 9 contra los que fue comparado (ESAXKMeans, SAXKMeans y RKMeans), así como a la clasificación usando los datos originales (sin ninguna discretización), lo que demuestra que, nuestra propuesta no pierde capacidad de clasificación cuando discretiza series temporales, sino que al contrario, mejora el porcentaje de esta tarea, permitiendo analizar grandes volúmenes de datos en computadoras personales mediante la reducción de la dimensionalidad. Esta prueba se puede apreciar en la Figura 5.3.

En los enfoques donde no hubo diferencia significativa (ESAX, rSAX, EP, SAX y 1D-SAX), se aplicó una prueba estadística por pares (suma de rangos de Wilcoxon), con la finalidad de analizar a detalle el rendimiento de eMODiTS, es decir, observar la tasa de clasificación obtenida en cada base de datos. La Figura 5.4 muestra que eMODiTS es estadísticamente superior en más de la mitad de las bases temporales en comparación con estos enfoques, siendo superada solo en unos cuantos casos. Dicha prueba confirma que nuestro enfoque es una herramienta competitiva para tareas de clasificación de series de tiempo, superando a los enfoques con objetivos similares e incluso mejorando el error en clasificación en la mayoría de las bases de datos de prueba.

Por otro lado, al utilizar una meta-heurística como herramienta de búsqueda de soluciones, eMODiTS adolece de emplear tiempo computacional alto, tal cual lo demuestra la Figura 5.8, donde se aprecia que al aumentar la dimensionalidad de los datos, mayor será el tiempo empleado por nuestro enfoque para encontrar la solución al problema. Desafortunadamente, en el mundo real, es frecuente encontrarse con problemas donde la dimensionalidad de los datos es alta, dejando a criterio del dueño de la información, el uso de herramientas que tengan un bajo costo computacional, pero que los resultados no sean los mejores, o usar herramientas (como la propuesta en este documento) que sean costosas en tiempo pero que encuentren soluciones competitivas o mejores a las que podría encontrar con otros métodos.

Observando el comportamiento de nuestra propuesta en las 85 bases de datos de prueba, se puede destacar que, eMODiTS tiene un rendimiento superior en bases temporales donde las clases se traslapan, ello tomando en cuenta los enfoques comparados en este trabajo (Figura 5.7b). Como se demuestra en las bases de datos *LargeKitchenAppliances* y *MedicalImages*, donde se puede apreciar como nuestra propuesta obtiene una tasa de error significativamente menor a los otros

enfoques de acuerdo a la Tabla 5.4.

Otra problema de las bases de datos en donde nuestra propuesta presenta un rendimiento favorable, es el ruido. En la Figura 5.7c se aprecia como eMODiTS es robusto a este tipo de problema. Esto se refleja en las bases de datos *BeetleFly* y *Lighting7*, donde se presentan niveles de ruido alto y bajo, respectivamente. En ambos conjuntos, nuestra propuesta alcanza una tasa de error en clasificación significativamente menor a las encontradas por los otros métodos independientemente del nivel de ruido. Mismo efecto se presenta en bases de datos no balanceadas (Figura 5.7a), donde nuestra propuesta obtiene las menores tasas de error en clasificación en la mayoría de los conjuntos con esta característica.

Estas ventajas se deben al esquema flexible propuesto, ya que, al utilizar diferentes esquemas de alfabeto por cada segmento de palabra, se incrementa el grado de libertad en la búsqueda, generando esquemas de discretización que permiten: (a) incrementar la separación entre los grupos de series de tiempo etiquetados con clases diferentes, (b) minimizar el nivel de ruido, y (c) mejorar la clasificación en conjuntos de datos con pocas instancias.

Sin embargo, aún cuando nuestra propuesta es robusta a fuentes de dificultad en bases de datos poco complejas (clases separadas, balanceadas y con ruido bajo), el costo computacional de eMODiTS es una limitación que debe tomarse en consideración cuando se requiere aplicar un proceso de discretización de bases temporales.

Reducción de la dimensionalidad

Como se mencionó anteriormente, los métodos de discretización buscan transformar una base de datos de alta dimensional en una de baja dimensionalidad, para poder analizarla de forma más eficiente y en equipos de cómputo convencionales. La mayoría de los métodos comparados en este estudio realizan una reducción de las dimensiones de la serie de tiempo, con excepción de RKMeans que sólo transforma series de tiempo continuas en series de tiempo discretas, es decir, sólo convierte los números reales a datos discretos sin realizar un proceso de reducción.

Como puede observarse en la Figura 5.5, eMODiTS logra reducir la dimensión de los datos significativamente en más del 95 % de las bases de datos temporales de prueba. En tres de las 85 bases de datos, la reducción lograda fue mínima, y en otras dos la compresión fue ligeramente menor a aquella obtenida por los otros métodos.

Estos resultados sugieren que nuestro enfoque es una herramienta competitiva capaz de encontrar soluciones con una tasa de compresión alta, pero con el mayor porcentaje de clasificación, convirtiéndose en una buena opción para clasificar datos de gran tamaño con el menor espacio de almacenamiento en memoria.

Pérdida de la información

Un efecto colateral de una alta compresión de los datos es la pérdida de la información importante de los mismos. Por lo tanto, es necesario comprobar que eMODiTS, al ser una herramienta con una alta capacidad de compresión, no está perdiendo la mayor cantidad de información durante el proceso de discretización.

En la Figura 5.6a se puede observar como eMODiTS tiene un rango promedio similar a la mayoría de los enfoques comparados, siendo superado por SAXKMeans, quien es el método con el porcentaje menor de pérdida de información. La Figura 5.6b demuestra estadísticamente estos resultados, donde SAXKMeans supera a los demás métodos de discretización simbólica excepto a ESAXKMeans quien es estadísticamente similar a éste.

Al analizar el comportamiento de los enfoques comparados en este estudio, se hace evidente el compromiso entre la pérdida de la información y el error en clasificación. Los algoritmos con la menor pérdida de información (SAXKMeans y ESAXKMeans) encuentran altas tasas de error en clasificación en la mayoría de las bases de datos temporales; mientras que, los métodos con el porcentaje de error en clasificación mínimo (eMODiTS, ESAX, rSAX, y EP) tienen una pérdida de información considerable, tal es el caso de EP, que es el método con la mayor cantidad de información perdida. Este fenómeno se presenta porque, con un conjunto de datos muy comprimido, un proceso de clasificación hace una separación de las clases eficientemente, y como consecuencia, una tasa de clasificación más eficiente. Sin embargo, puede presentar pérdida de información de los componentes principales de la serie de tiempo. Por otro lado, con una compresión baja del conjunto temporal, las clases siguen traslapadas o muy juntas unas de otras, presentando tasas de error en clasificación altas, y pérdida de información bajas dado que la series de tiempo discreta es similar a la original.

Nuestra propuesta presenta un buen compromiso entre las medidas de evaluación discutidas en este apartado. Dado que se posicionó como el mejor método para clasificar series de tiempo en comparación con los otros algoritmos comparados, basado en el rango promedio obtenido y los resultados estadísticos de la prueba de Friedman. De la misma forma, eMODiTS alcanza porcentajes de compresión superiores en la mayoría de las bases de datos de prueba. Sin embargo, aún cuando nuestro enfoque no es el método con la menor pérdida de información, presenta una pérdida similar a la mayoría de los enfoques superando a αSAX y EP.

En resumen, eMODiTS es una herramienta competitiva para discretizar series de tiempo reduciendo significativamente su dimensionalidad con la menor tasa de error en clasificación y una aceptable pérdida de información, es decir, eMODiTS reduce lo suficiente una base de datos temporal para eliminar traslapes entre subconjuntos de series de tiempo de cada una de las etiquetas de clase existentes sin perder demasiada información de los datos. Estas características son vitales para el dueño de la información dado que sus datos pueden almacenarse en dispositivos de almacenamiento comerciales y ser analizados de una forma eficiente. Sin embargo, el precio que tiene que pagar por ello es el alto costo computacional del proceso de discretización.

5.5. Interpretación Gráfica

5.5.1. Introducción

Otro aspecto crítico para el dueño de la información es la interpretación de los datos; es decir, conocer los patrones relevantes presentes en las bases de datos temporales que permitirá anticipar posibles eventos y tomar las decisiones acertadas en el tiempo adecuado.

Existen diversas herramientas gráficas y estadísticas para extraer estas relaciones o patrones de los conjuntos temporales. Dado que eMODiTS utiliza árboles de decisión como clasificador de los datos, éste también fue utilizado para entender gráficamente cada base de datos temporal. En nuestra propuesta, los conjuntos de datos temporales, que son datos de un sólo atributo, son transformados en conjuntos de datos con m atributos que representan intervalos de tiempo de la serie, permitiendo graficar la ocurrencia de cada una de las clases de las bases temporales.

En el árbol de decisión obtenido, cada nodo hoja representa una determinada clase; un nodo interno representa un segmento en el tiempo y cada arista representa un intervalo de valores necesarios que se satisfagan para que se presente una clase o evento. Con esta información y usando nuestro esquema de discretización, se puede observar gráficamente el período de tiempo y el rango de valores en los cuales un evento puede ocurrir y entonces, ser capaces de anticipar las repercusiones que pudiera tener a un determinado problema.

Para facilitar la presentación e interpretación de los resultados, sólo se tomaron 3 de las 85 bases de datos temporales utilizadas en este estudio. Dichas bases son: BeetleFly, GunPoint y ItalyPowerDemand, las cuales son representativas del conjunto de problemas encontrados en el repositorio UCR de prueba. Los resultados de las bases de datos restantes pueden consultarse en los Anexos A y B.

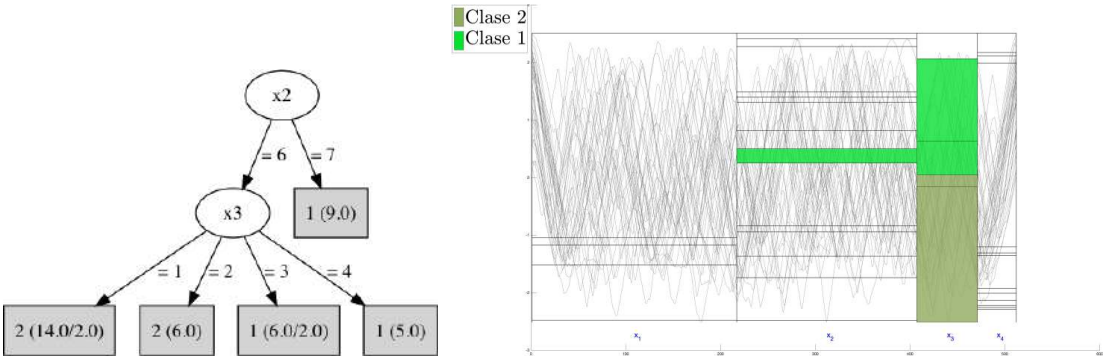
La base de datos BeetleFly consiste de un conjunto de imágenes binarias convertidas en series de tiempo de contornos de insectos en diferentes posiciones. Las clases de este conjunto son dos tipos de insectos: *escarabajo* y *mosca*. Por su parte, la bases de datos GunPoint consiste de un conjunto de series de tiempo con los movimientos de un actor, tanto masculino como femenino, simulando apuntar con una pistola hacia un objetivo particular. Las clases de esta base de datos son: *desenfundar* y *apuntar* la pistola.

Finalmente, la base de datos ItalyPowerDemand contiene series de tiempo que representan el consumo de energía diario en un año en Italia. Dicho consumo es clasificado dependiendo del periodo del año debido a que los calefactores son usados con mas frecuencia en invierno gastando una considerable cantidad de energía eléctrica. Mientras tanto, en verano, el uso de aires acondicionados es poco frecuente, generando una baja demanda de electricidad en dicho país. Por lo tanto, la tarea de clasificación en esta base consiste en determinar si un día particular corresponde a uno de estos dos periodos del año. Las clases de esta base de datos son: *días de invierno* (de Octubre a Marzo) y *días de verano* (Abril a Septiembre).

En la Sección 5.5.2, se muestra la interpretación visual de cada una de las bases de datos mencionadas mediante el esquema encontrado por eMODiTS y los árboles de decisión usados para su evaluación.

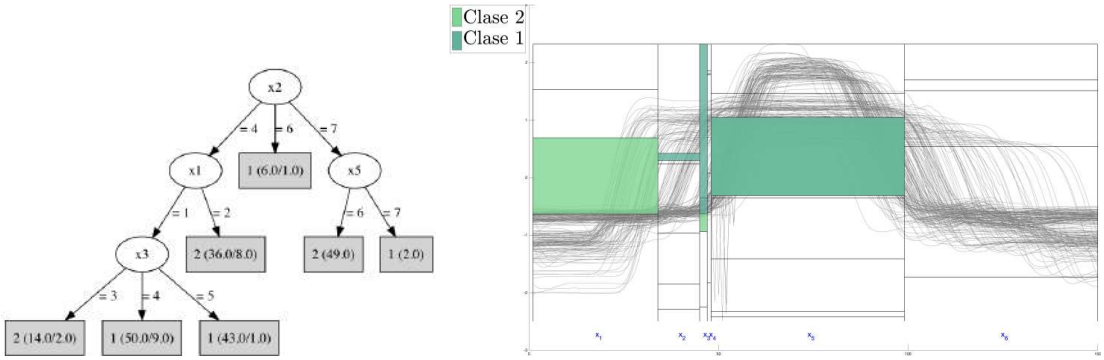
5.5.2. Resultados

En este apartado se presentan los árboles de decisión y el esquema final obtenido por eMODiTS, en donde se marcan las secciones del espacio temporal donde se presenta principalmente una clase. Las Figuras 5.9a, 5.10a, y 5.11a muestran los árboles de decisión para cada base de datos anteriormente mencionadas. Mientras que las Figuras 5.9b, 5.10b, y 5.11b muestran la distribución de las clases de acuerdo a las reglas indicadas en su correspondiente árbol de decisión.



(a) (b)

Figura 5.9: (a) Árbol de decisión obtenido por eMODiTS para la base de datos BeetleFly. (b) Distribución de las clases para la bases de datos BeetleFly donde cada rectángulo representa un nodo hoja del árbol de decisión.

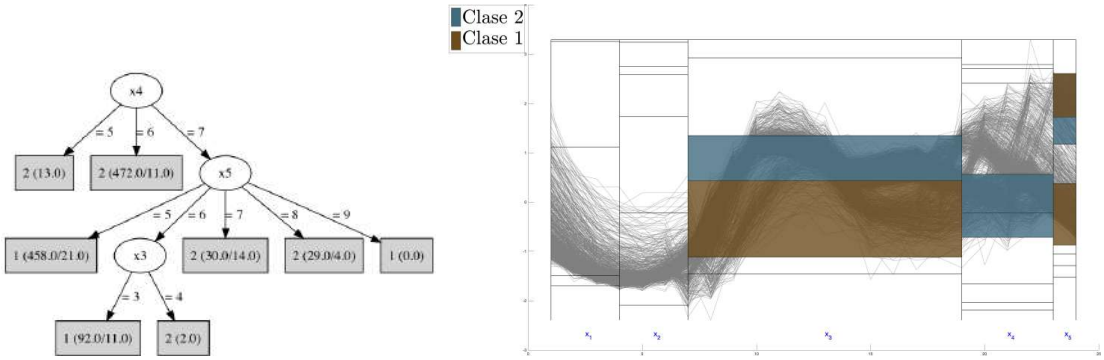


(a) (b)

Figura 5.10: (a) Árbol de decisión obtenido por eMODiTS para la base de datos GunPoint. (b) Distribución de las clases para la base de datos GunPoint donde cada rectángulo representa un nodo hoja del árbol de decisión.

5.5.3. Discusión

En esta sección se presentaron los resultados visuales de algunas de las bases de datos temporales usadas en los experimentos: BeetleFly, GunPoint y ItalyPowerDemand. De cada una de ellas, se obtuvo el árbol de decisión y el esquema gráfico



(a) **(b)**

Figura 5.11: Árbol de decisión obtenido por eMODiTS para la base de datos ItalyPower-Demand. (b) Distribución de las clases para la base de datos ItalyPowerDemand donde cada rectángulo representa un nodo hoja del árbol de decisión.

obtenidos por eMODiTS. En este último se plasmó gráficamente la información del árbol de decisión para analizar cómo se distribuyen o se localizan cada una de las clases de cada base de datos.

La Figura 5.9b muestra la distribución de las clases para la base de datos *BeetleFly*, donde se puede apreciar que la clase *escarabajo* es ubicada de la parte media al final de las series temporales, y la clase *mosca* puede ser identificada al final de la serie de tiempo. Como puede observarse, con esta información es posible desarrollar sistemas inteligentes que, basados en una imagen, encuentre todas las imágenes relacionadas y regrese un conjunto de objetos similares al inicial, sólo con analizar una porción de la serie temporal en lugar de la serie completa. De igual forma, con estos resultados, se podría desarrollar un robot con la habilidad de identificar objetos de acuerdo a su contorno basándose en un conjunto de imágenes previas.

Por otro lado, la Figura 5.10b ilustra la ubicación de cada clase de la base de datos *GunPoint*, donde es claro que del inicio a la mitad de la información temporal es posible determinar si una persona va a apuntar con una pistola hacia un objetivo, o no. Para esta última, es hasta la mitad de la serie temporal cuando es posible determinar que la intención del individuo no es apuntar y sólo esta sacando la pistola para otro fin. Con esta información, se pueden desarrollar aplicaciones o agentes inteligentes que puedan ser entrenados para anticipar eventos dependiendo de las acciones de una persona, evitando situaciones de alto riesgo o incluso salvando vidas humanas.

Por último, la Figura 5.11b muestra las relaciones y ubicaciones de cada etiqueta de clase para la base de datos *ItalyPowerDemand*. En esta figura, un día es considerado de invierno si su demanda desde el medio día hasta la noche es alta, por el contrario, un día es considerado de verano si su demanda es baja en el mismo período de tiempo. Este tipo de información puede ser útil para desarrollar sistemas de potencia que permitan controlar los voltajes de energía para una casa con un importante ahorro energético regional, contribuyendo a la mitigación de problemas severos de índole mundial como el calentamiento global.

Como se mencionó anteriormente, eMODiTS no sólo contribuye con esquemas de discretización competitivos de series temporales en términos de clasificación y

reducción de la dimensionalidad, sino que también permite visualizar gráficamente los componentes importante de los datos, característica útil para desarrollar herramientas capaces de tomar decisiones oportunas en determinadas situaciones de riesgo.

6

Conclusiones y Trabajo Futuro

En este documento, se presentó un algoritmo de discretización de series de tiempo, llamado eMODiTS, basado en tres características esenciales para dicho proceso: clasificación, complejidad, y pérdida de la información. La propuesta define un esquema de discretización flexible o adaptativo a los datos, es decir, define esquemas de cortes de alfabetos diferentes por cada segmento o corte de palabra, permitiendo adaptarse a la información temporal encontrada en cada uno. Para definir los valores de cada uno de los cortes, eMODiTS utiliza una herramienta de búsqueda global capaz de encontrar soluciones en más de dos características conflictuadas entre sí. Dicha herramienta es el algoritmo evolutivo multi-objetivo NSGA-II, implementado por ser uno de los algoritmos que ha demostrado buenos resultados en la mayoría de los problemas donde se ha utilizado. NSGA-II busca los valores competitivos tanto para el número de cortes de palabra y alfabetos, como los valores de dichos cortes, situación que representa una desventaja para el conocido algoritmo de discretización llamado SAX, es decir, eMODiTS trata de solventar dos de las desventajas descritas en la literatura especializada de SAX: (1) número de cortes de palabra y alfabeto, y (2) valores competitivos para cada corte.

Dado que NSGA-II es un algoritmo multi-objetivo, el óptimo es un conjunto de soluciones compromiso llamado conjunto de óptimos de Pareto; por lo cual, fue necesario determinar una estrategia de selección de dicho conjunto para obtener la solución final al problema. Para dicha tarea, se implementaron y analizaron cuatro métodos de selección de preferencias con la finalidad de observar cuál seleccionaba la solución más competitiva. Dichos métodos fueron: el método de la rodilla (Knee), el método CV (validación cruzada), el método KM1, y el método KM20. De acuerdo a los resultados de ese análisis, el método CV fue el que seleccionó la mejor solución de $\mathcal{PF}$, dado que fue el que mejor rango promedio alcanzó en todas las bases de datos de prueba y de todos los algoritmos confrontados.

Las bases temporales de prueba fueron obtenidas de [27], las cuales forman un total de 85 conjuntos de series temporales diseñadas para resolver diferentes

problemas de clasificación de aplicaciones reales. Las características de las bases de datos fueron descritas en la Sección 5.2 y en la Tabla 5.2.

Una vez escogido el método de selección de preferencias, eMODiTS fue comparado contra nueve algoritmos de discretización simbólica basados en SAX, implementados por tener características similares a nuestra propuesta y obtener resultados competitivos en los problemas donde fueron implementados. Dicho algoritmos fueron: EP, SAX, αSAX, ESAX, ESAXKMeans, 1D-SAX, RKMeans, SAXKMeans, y rSAX. Los experimentos fueron guiados para analizar el rendimiento de nuestra propuesta en tres tareas criticadas cuando un proceso de discretización es realizado: porcentaje de clasificación, reducción de la dimensionalidad y pérdida de la información.

Con respecto a la tarea de clasificación, eMODiTS supera estadísticamente a ESAXKMeans, RKMeans, y SAXKMeans basado en los resultados de la prueba estadística multi-comparación de Friedman y la prueba post hoc de Nemenyi. Sin embargo, comparado contra EP, SAX, ESAX, rSAX, αSAX, y 1D-SAX, la prueba de Friedman no arroja diferencias significativas contra eMODiTS, por ello y para analizar a detalle el comportamiento de nuestro algoritmo con respecto a dichos métodos, se realizó una prueba estadística por pares, donde se utilizaron los mejores resultados por base de datos y no en forma general. Dicha prueba es la bien conocida prueba de suma de rangos de Wilcoxon. Los resultados de la prueba demostraron que eMODiTS supera a cada uno de esos métodos donde la prueba de Friedman no encontró diferencias significativas, en más de la mitad de las bases de datos, demostrando que eMODiTS tiene un comportamiento superior con respecto a cada uno de los métodos comparados en cuanto a clasificación de series de tiempo se refiere.

Por otro lado, eMODiTS logró encontrar esquemas con tasas alta de reducción de dimensionalidad en más del 95 % de las bases temporales, ahorrando espacio de almacenamiento y facilitando su procesamiento en recursos convencionales de cómputo.

Finalmente, el último punto evaluado fue la pérdida de la información, la cual es un problema serio en algoritmos de reducción de dimensionalidad. La pérdida de la información fue estimada usando una medida de similitud robusta a datos atípico y/o ruido, llamada la *Subsecuencia Común más Larga (Longest Common Subsequence, LCSS)*. Los resultados sugieren que ESAXKMeans y SAXKMeans conservan más información de la serie de tiempo original comparado con los otros enfoques. Por otra parte, eMODiTS pierde menos información que EP, pero tiene un comportamiento similar al resto de los métodos basados en SAX. Es importante mencionar que los enfoques con un pérdida de información menor obtuvieron una alta tasa de error en clasificación, y viceversa, demostrando el compromiso entre estos aspectos. Sin embargo, eMODiTS logró encontrar un equilibrio entre estas funciones, obteniendo tasas de error en clasificación menores con alta reducción de dimensionalidad y pérdida de información aceptable.

En resumen, el método propuesto es un enfoque novedoso que busca esquemas de discretización con una alta tasa de compresión, sin una pérdida significativa de la información y con una habilidad superior para clasificar datos temporales. Estos aspectos no son considerados en conjunto en los métodos basados en SAX encontrados en la literatura especializada, los cuales sólo se enfocan en la reducción de la dimensionalidad y/o en clasificación de datos, dejando de lado la pérdida de

información resultante de los procesos de compresión de datos.

Adicionalmente, eMODiTS es útil en bases de datos donde es difícil encontrar un modelo para discernir entre etiquetas de clase, así como, en bases de datos no balanceadas y con niveles altos de ruido. Por lo contrario, en bases de datos donde las clases están separadas, balanceadas y con poco ruido, es conveniente utilizar otro método de discretización simbólica basado en SAX, los cuales obtienen resultados competitivos, similares a los obtenidos por eMODiTS, pero con menor costo computacional que el empleado por nuestra propuesta.

Con todo lo mencionado, se puede concluir que nuestra propuesta alcanzó el objetivo general planteado en esta investigación. De la misma forma, se logró comprobar la hipótesis planteada, debido a que nuestra propuesta logró reducir el error en clasificación con una alta tasa de compresión de las series de tiempo y con una pérdida de información aceptable, siendo una propuesta competitiva en el proceso de discretización de bases de datos temporales.

El tiempo computacional representa la principal limitación de nuestro método. Dicho problema depende del tamaño de la base de datos a analizar, es decir, en bases de datos temporales pequeñas (pocas instancias) una ejecución toma alrededor de 30 segundos en encontrar una solución adecuada al problema. Para el caso de bases de datos con un número alto de instancias, una ejecución toma alrededor de una hora o más. Queda a elección del usuario decidir el uso de eMODiTS en sus datos sin importar el tiempo empleado en la búsqueda de la solución idónea.

Como consecuencia, se puede sugerir como trabajo futuro un análisis de la complejidad de nuestro algoritmo para decrementar el tiempo computacional del mismo. Una posible acción para minimizar el tiempo es probar con otros algoritmos evolutivos multi-objetivo que demostraron gastar menos tiempo computacional en los problemas donde fueron implementados, así como el uso de algoritmos subrogados que aproximen el valor de la evaluación de soluciones en un menor tiempo que el empleado por eMODiTS.

Además, podrían incluirse otros mecanismos para reducir la pérdida de información, ya sea cambiando la función objetivo o mejorando el método de reducción de dimensionalidad utilizado.

Apéndices

A

Árboles de decisión

Contenido

A.1. Adiac 111
A.2. ArrowHead 111
A.3. Beef 112
A.4. BeetleFly 112
A.5. BirdChicken 113
A.6. Car 113
A.7. CBF 114
A.8. ChlorineConcentration 114
A.9. CinCECGtorso 115
A.10. Coffee 115
A.11. Computers 116
A.12. CricketX 116
A.13. CricketY 116
A.14. CricketZ 116
A.15. DiatomSizeReduction 117
A.16. DistalPhalanxOutlineAgeGroup 117
A.17. DistalPhalanxOutlineCorrect 118
A.18. DistalPhalanxTW 118
A.19. Earthquakes 118
A.20. ECG200 119
A.21. ECG5000 119
A.22. ECGFiveDays 120
A.23. ElectricDevices 120
A.24. FaceAll 120
A.25. FaceFour 121
A.26. FacesUCR 121
A.27. FiftyWords 121
A.28. Fish 121

A.29.FordA 122
A.30.FordB 122
A.31.GunPoint 122
A.32.Ham 123
A.33.HandOutlines 123
A.34.Haptics 124
A.35.Herring 124
A.36.InlineSkate 125
A.37.InsectWingbeatSound 125
A.38.ItalyPowerDemand 125
A.39.LargeKitchenAppliances 126
A.40.Lighting2 126
A.41.Lighting7 127
A.42.Mallat 127
A.43.Meat 128
A.44.MedicalImages 128
A.45.MiddlePhalanxOutlineAgeGroup 129
A.46.MiddlePhalanxOutlineCorrect 129
A.47.MiddlePhalanxTW 130
A.48.MoteStrain 130
A.49.NonInvasiveFetalECGThorax1 130
A.50.NonInvasiveFetalECGThorax2 130
A.51.OliveOil 131
A.52.OSULeaf 131
A.53.PhalangesOutlinesCorrect 131
A.54.Phoneme 132
A.55.Plane 132
A.56.ProximalPhalanxOutlineAgeGroup 133
A.57.ProximalPhalanxOutlineCorrect 133
A.58.ProximalPhalanxTW 134
A.59.RefrigerationDevices 134
A.60.ScreenType 135
A.61.ShapeletSim 135
A.62.ShapesAll 135
A.63.SmallKitchenAppliances 136
A.64.SonyAIBORobotSurface1 136
A.65.SonyAIBORobotSurface2 137
A.66.StarLightCurves 137
A.67.Strawberry 138
A.68.SwedishLeaf 138
A.69.Symbols 138
A.70.SyntheticControl 139
A.71.ToeSegmentation1 139
A.72.ToeSegmentation2 139
A.73.Trace 140
A.74.TwoLeadECG 140
A.75.TwoPatterns 141
A.76.UWaveGestureLibraryAll 141

A.77.UWaveGestureLibraryX **141**
A.78.UWaveGestureLibraryY **141**
A.79.UWaveGestureLibraryZ **141**
A.80.Wafer **142**
A.81.Wine **142**
A.82.WordSynonyms **143**
A.83.Worms **143**
A.84.WormsTwoClass **143**
A.85.Yoga **143**

Los árboles de decisión presentados en este apartado son aquellos resultantes de la clasificación usada para evaluar el esquema de discretización encontrado por eMODiTS. Cada uno de los árboles de decisión incluidos pueden ser consultados en la página `https://github.com/scoramg/eMODiTS/` para mayor detalle.

A.1. Adiac

Figura A.1: Árbol de decisión obtenido por eMODiTS para la base de datos Adiac.

A.2. ArrowHead

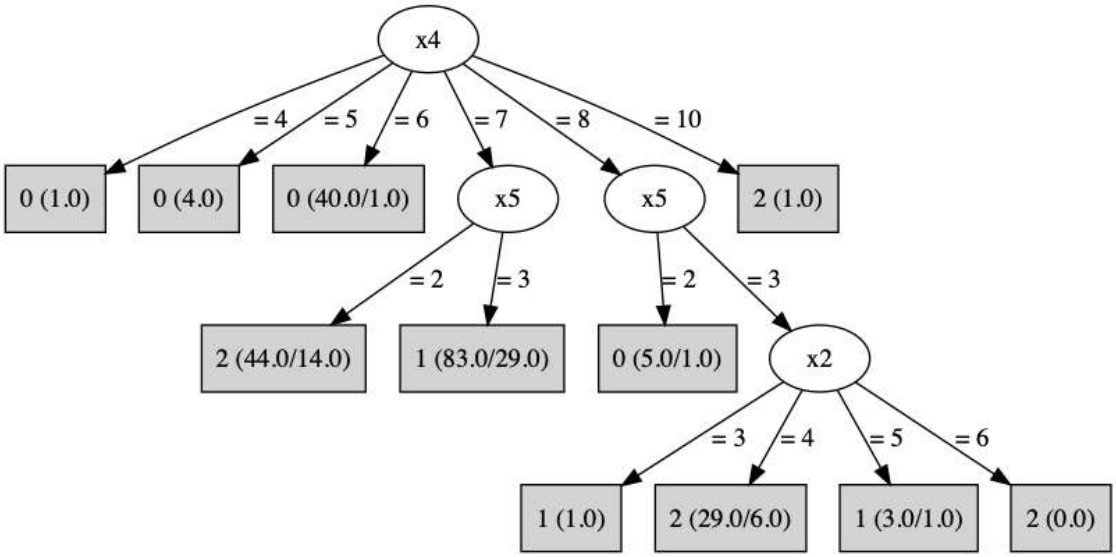


Figura A.2: Árbol de decisión obtenido por eMODiTS para la base de datos ArrowHead.

A.3. Beef

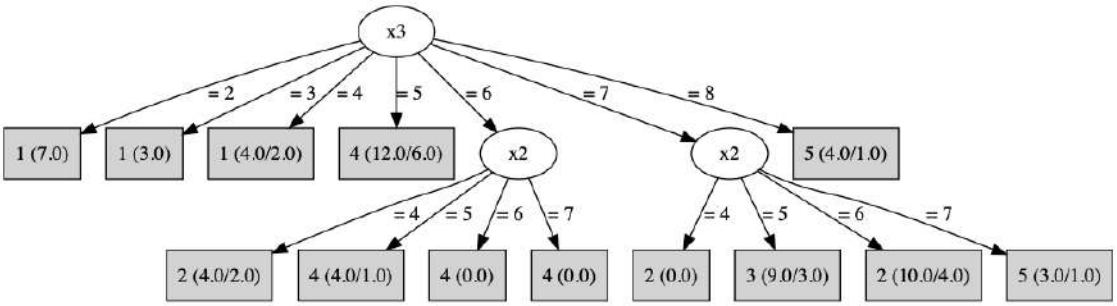


Figura A.3: Árbol de decisión obtenido por eMODiTS para la base de datos Beef.

A.4. BeetleFly

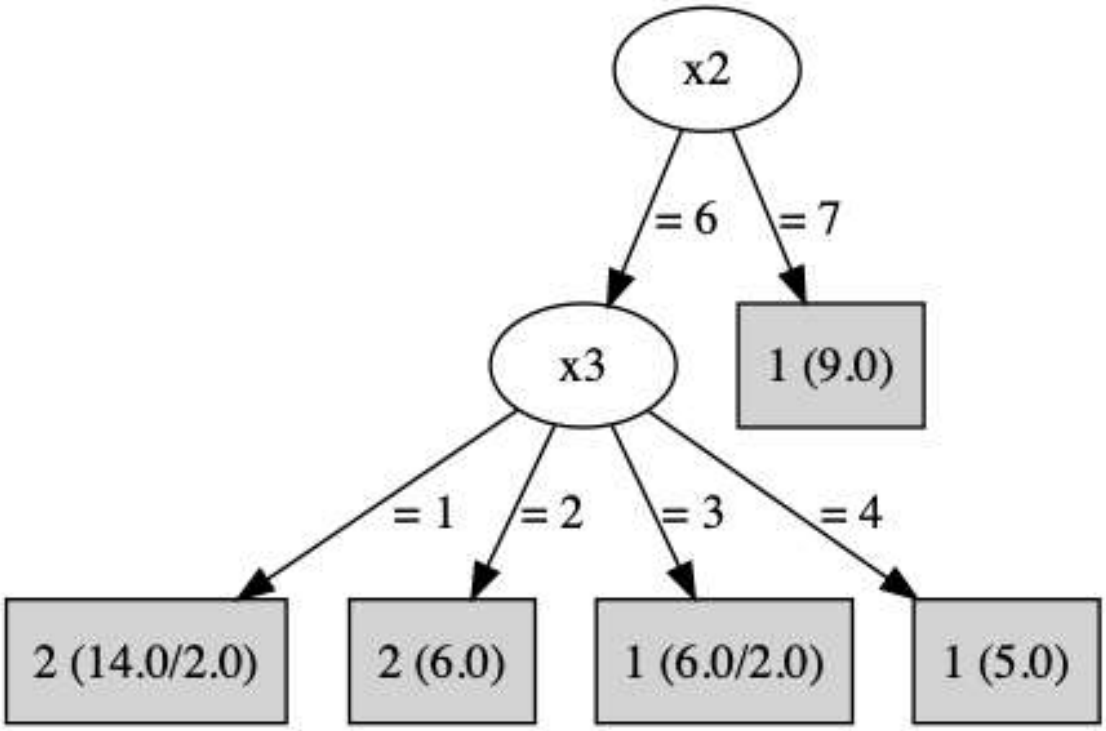


Figura A.4: Árbol de decisión obtenido por eMODiTS para la base de datos BeetleFly.

A.5. BirdChicken

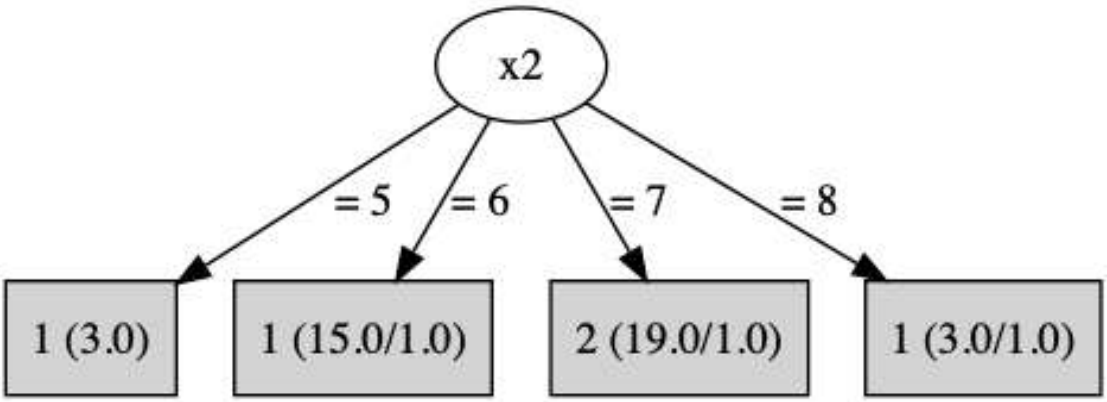


Figura A.5: Árbol de decisión obtenido por eMODiTS para la base de datos BirdChicken.

A.6. Car

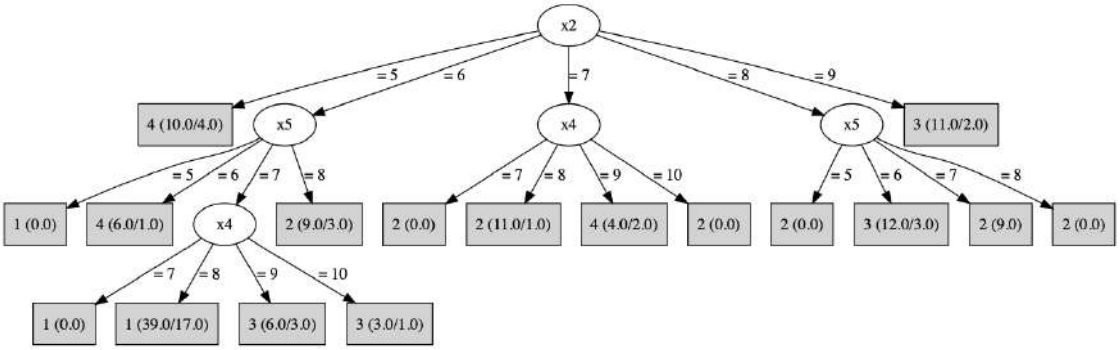


Figura A.6: Árbol de decisión obtenido por eMODiTS para la base de datos Car.

A.7. CBF

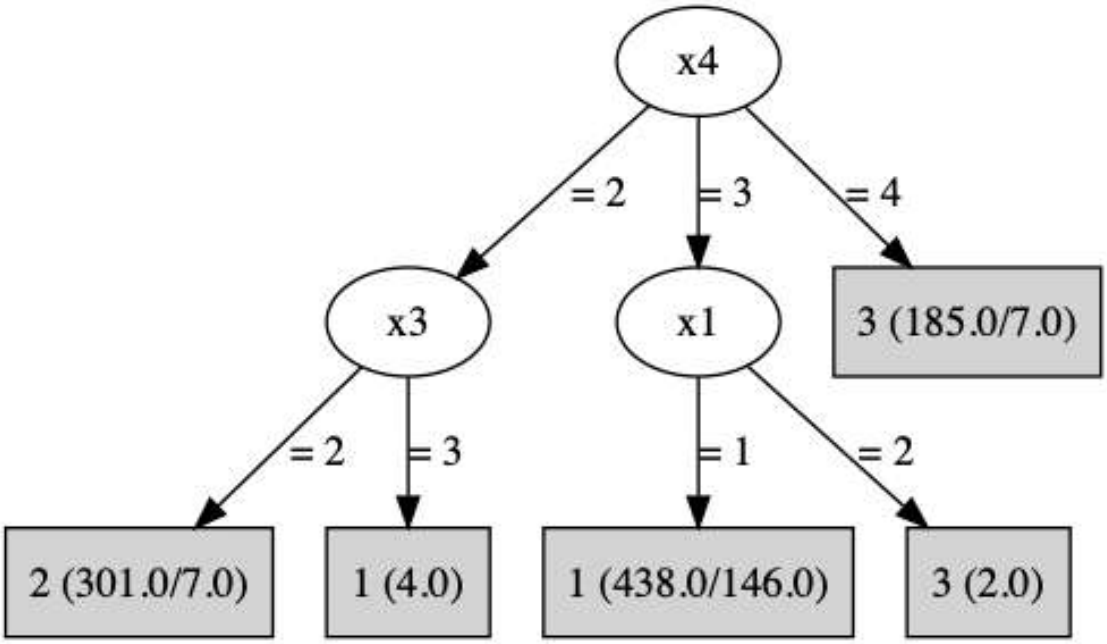


Figura A.7: Árbol de decisión obtenido por eMODiTS para la base de datos CBF.

A.8. ChlorineConcentration

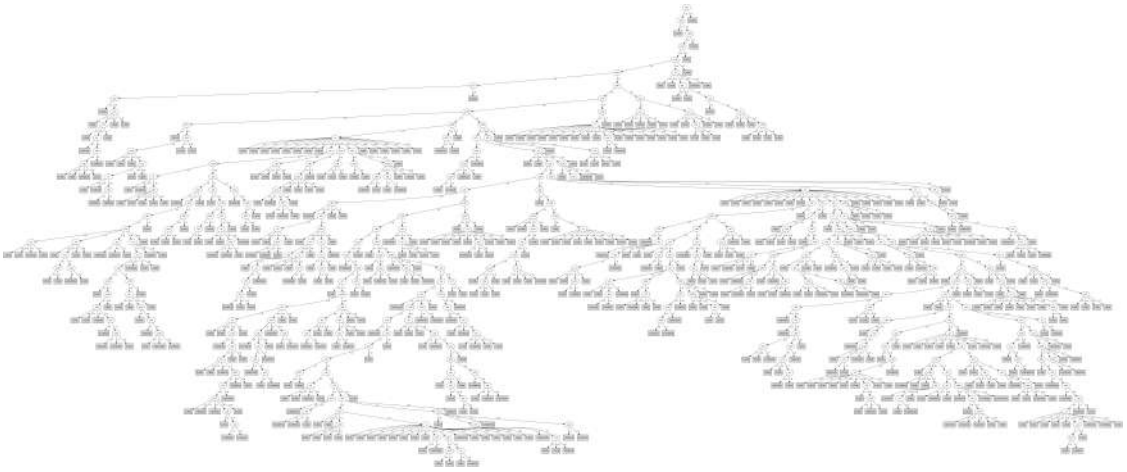


Figura A.8: Árbol de decisión obtenido por eMODiTS para la base de datos ChlorineConcentration.

A.9. CinCECGtorso

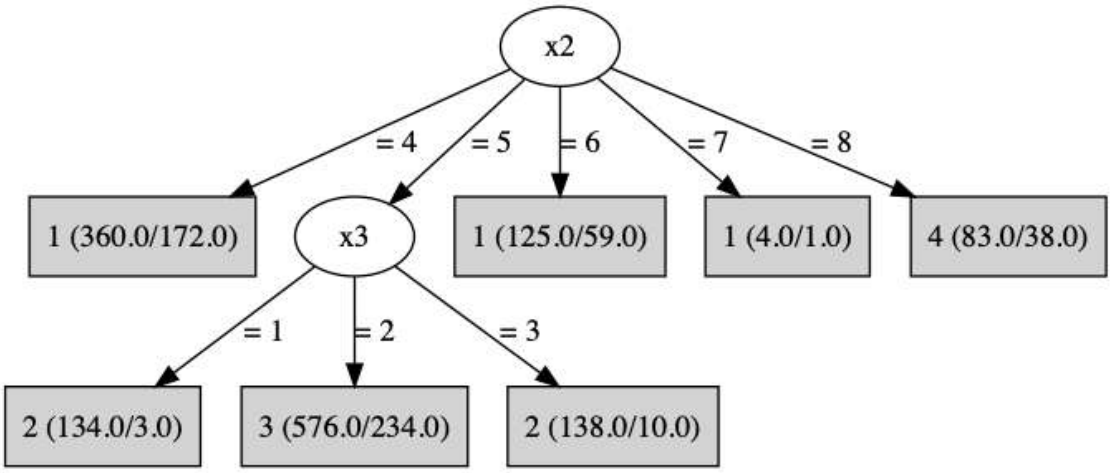


Figura A.9: Árbol de decisión obtenido por eMODiTS para la base de datos CinCECGtorso.

A.10. Coffee

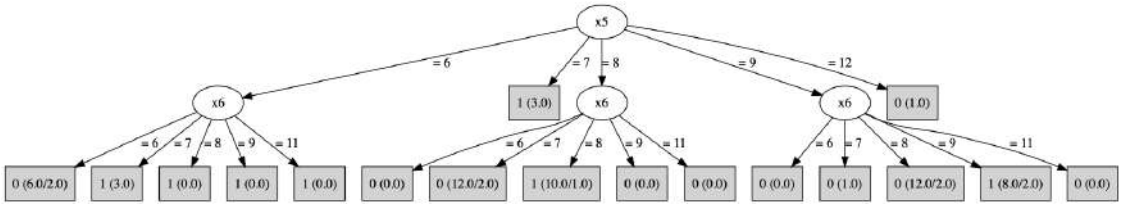


Figura A.10: Árbol de decisión obtenido por eMODiTS para la base de datos Coffee.

A.11. Computers

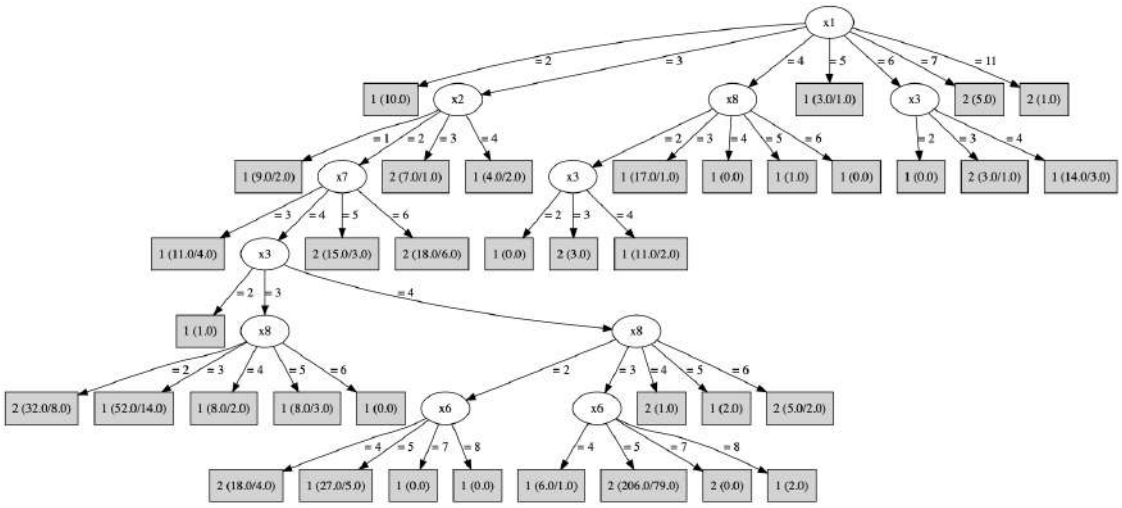


Figura A.11: Árbol de decisión obtenido por eMODiTS para la base de datos Computers.

A.12. CricketX

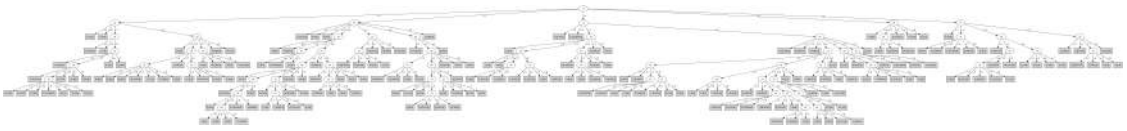


Figura A.12: Árbol de decisión obtenido por eMODiTS para la base de datos CricketX.

A.13. CricketY

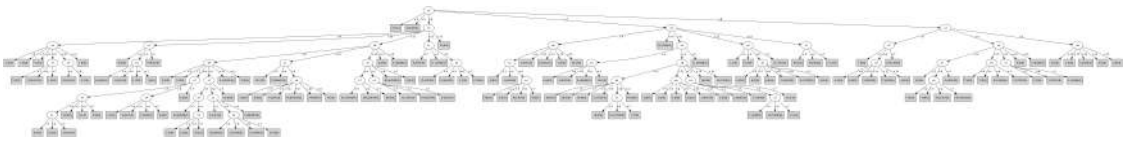


Figura A.13: Árbol de decisión obtenido por eMODiTS para la base de datos CricketY.

A.14. CricketZ

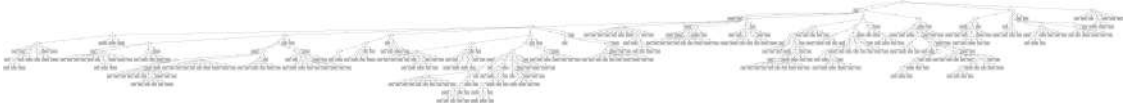


Figura A.14: Árbol de decisión obtenido por eMODiTS para la base de datos CricketZ.

A.15. DiatomSizeReduction

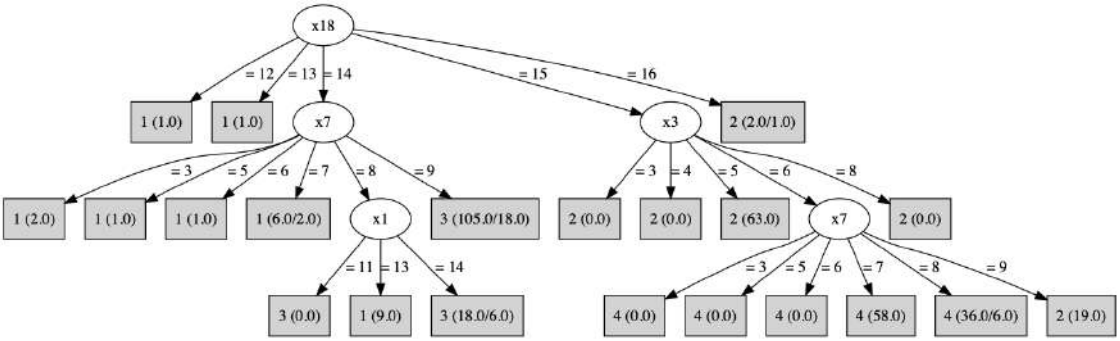


Figura A.15: Árbol de decisión obtenido por eMODiTS para la base de datos DiatomSizeReduction.

A.16. DistalPhalanxOutlineAgeGroup

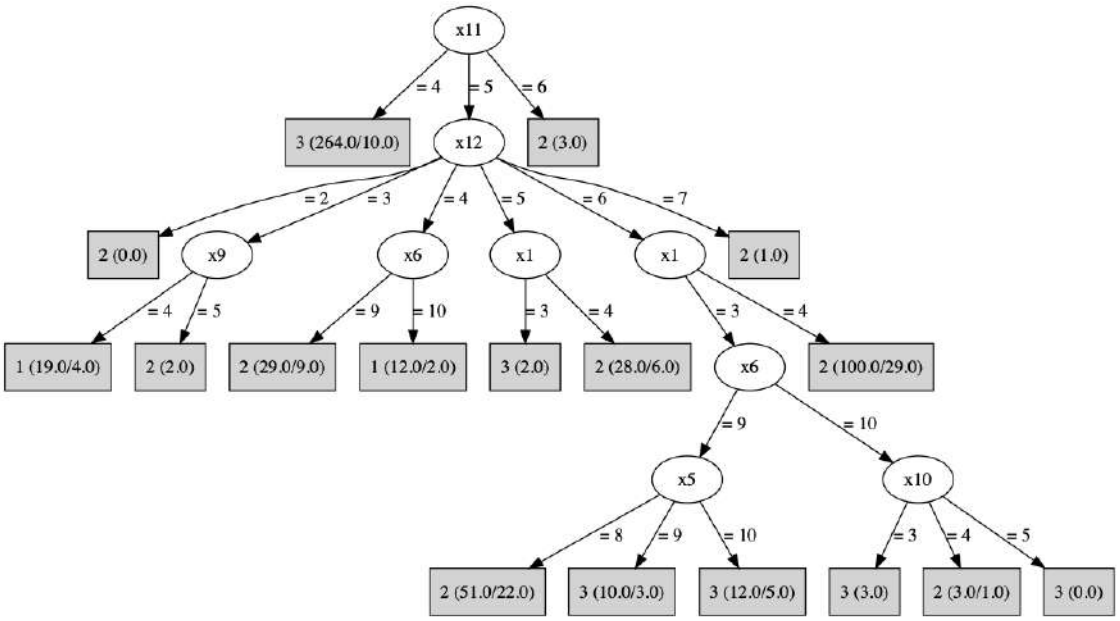


Figura A.16: Árbol de decisión obtenido por eMODiTS para la base de datos DistalPhalanxOutlineAgeGroup.

A.17. DistalPhalanxOutlineCorrect

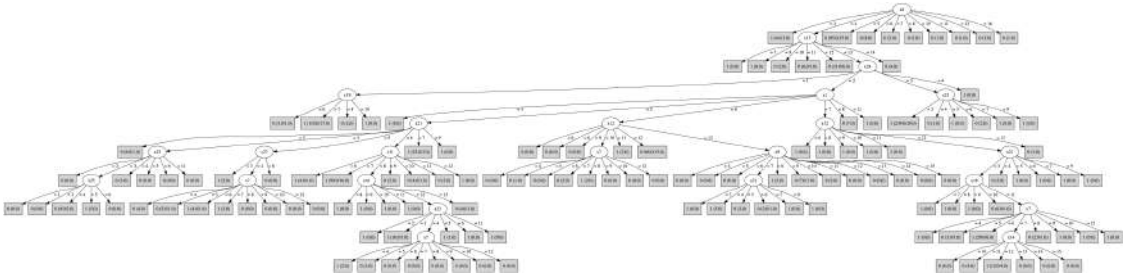


Figura A.17: Árbol de decisión obtenido por eMODiTS para la base de datos DistalPhalanxOutlineCorrect.

A.18. DistalPhalanxTW

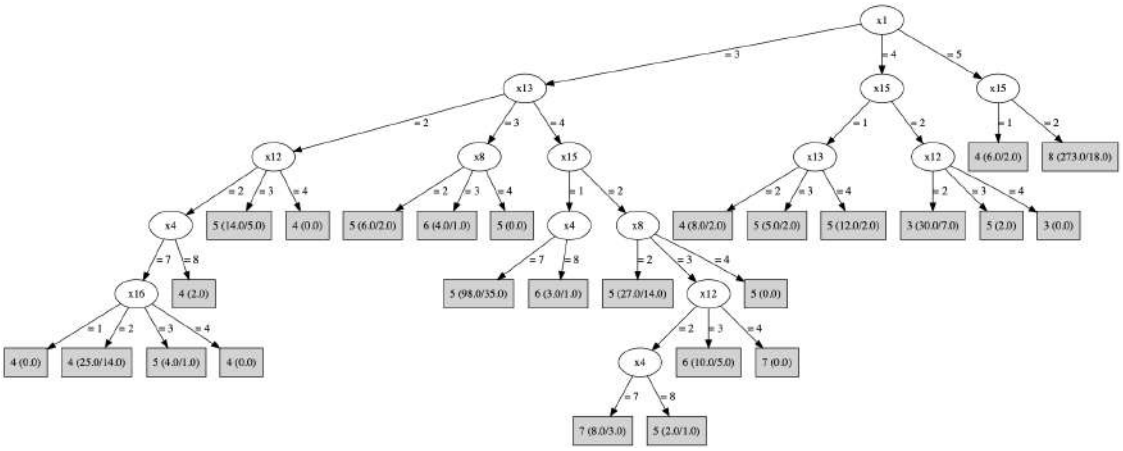


Figura A.18: Árbol de decisión obtenido por eMODiTS para la base de datos DistalPhalanxTW.

A.19. Earthquakes

Figura A.19: Árbol de decisión obtenido por eMODiTS para la base de datos Earthquakes.

A.20. ECG200

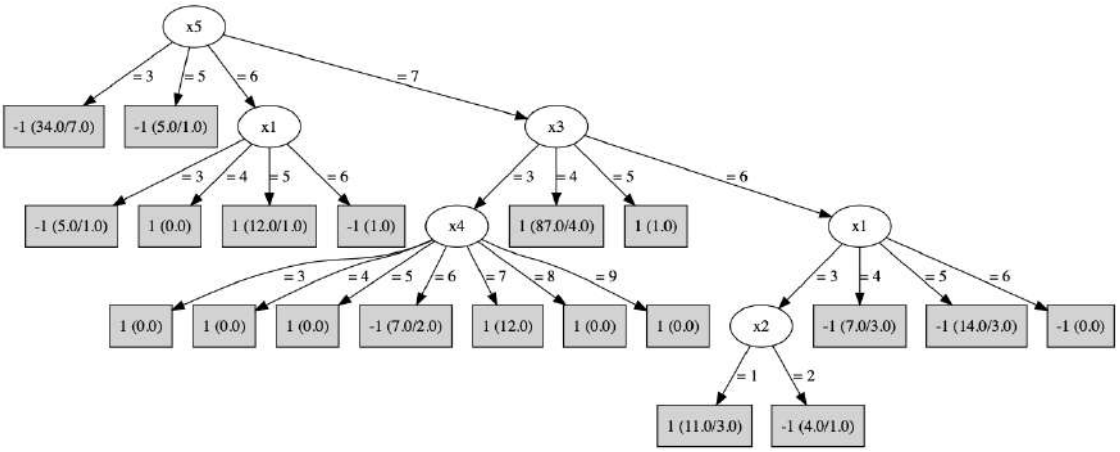


Figura A.20: Árbol de decisión obtenido por eMODiTS para la base de datos ECG200.

A.21. ECG5000

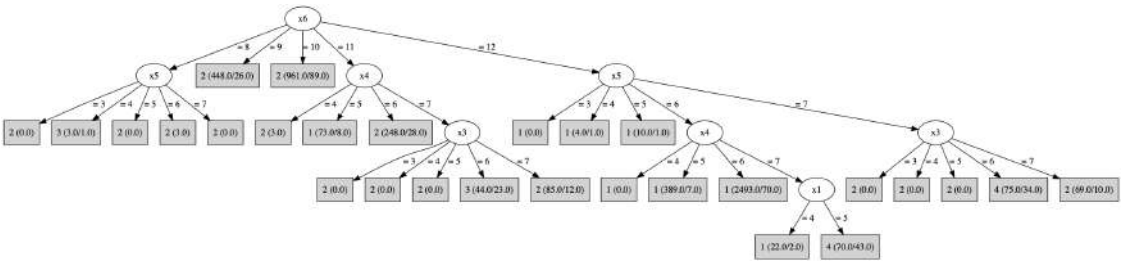


Figura A.21: Árbol de decisión obtenido por eMODiTS para la base de datos ECG5000.

A.22. ECGFiveDays

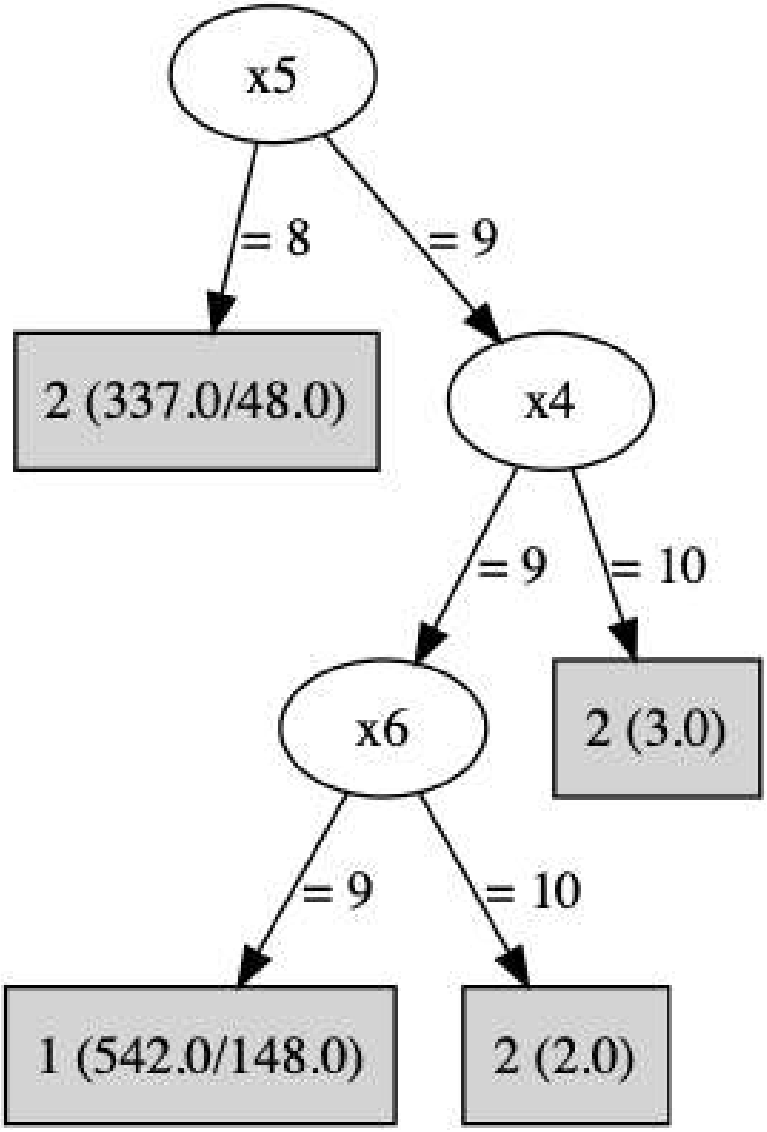


Figura A.22: Árbol de decisión obtenido por eMODiTS para la base de datos ECGFiveDays.

A.23. ElectricDevices

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.24. FaceAll

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.25. FaceFour

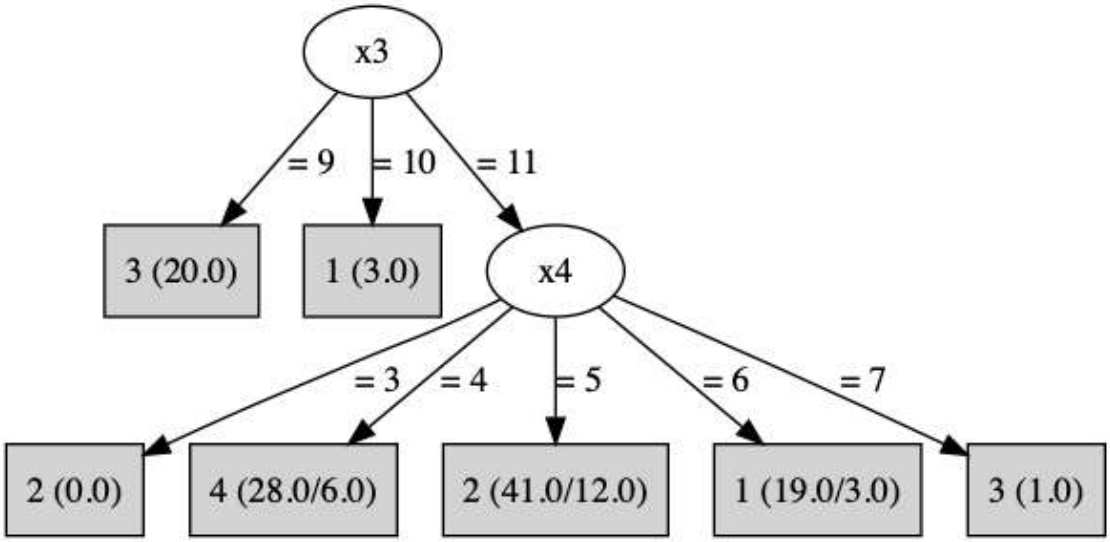


Figura A.23: Árbol de decisión obtenido por eMODiTS para la base de datos FaceFour.

A.26. FacesUCR

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.27. FiftyWords

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.28. Fish

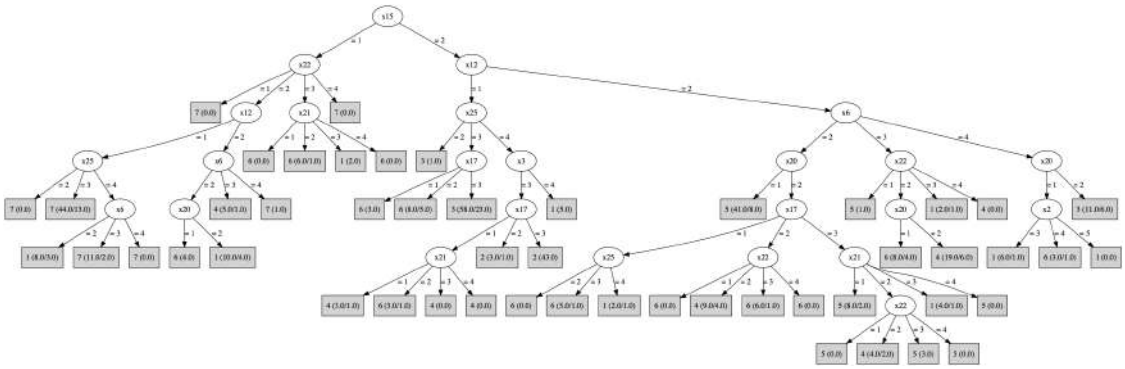


Figura A.24: Árbol de decisión obtenido por eMODiTS para la base de datos Fish.

A.29. FordA

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.30. FordB

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.31. GunPoint

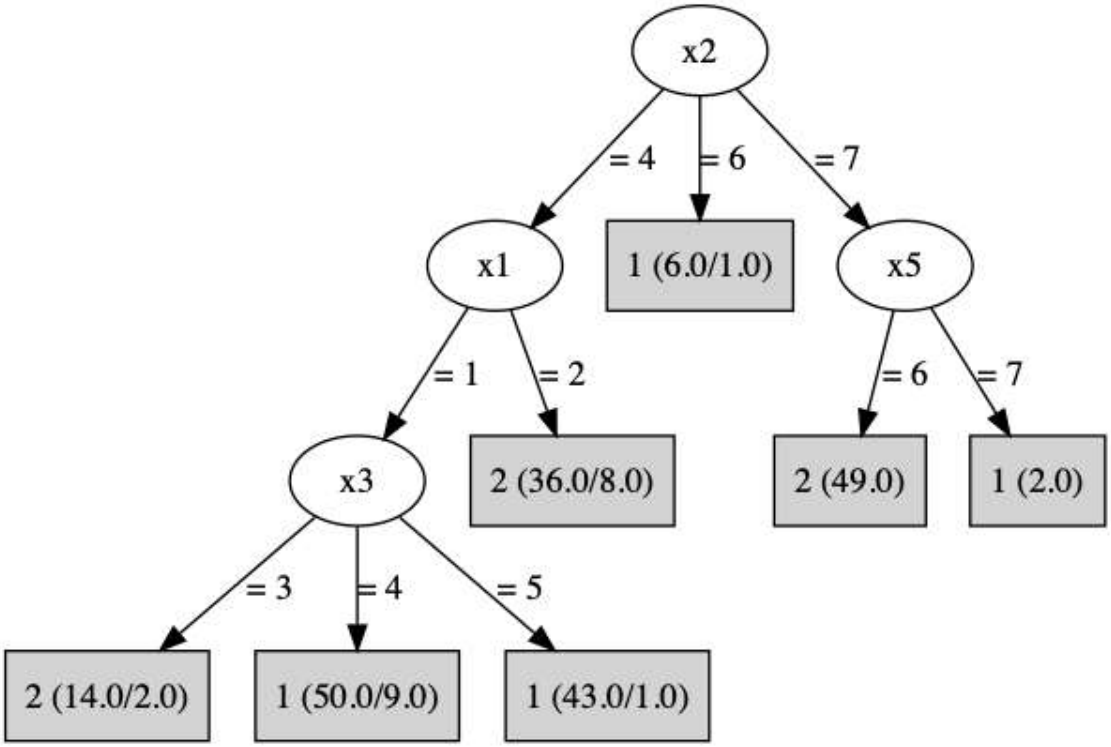


Figura A.25: Árbol de decisión obtenido por eMODiTS para la base de datos GunPoint.

A.32. Ham

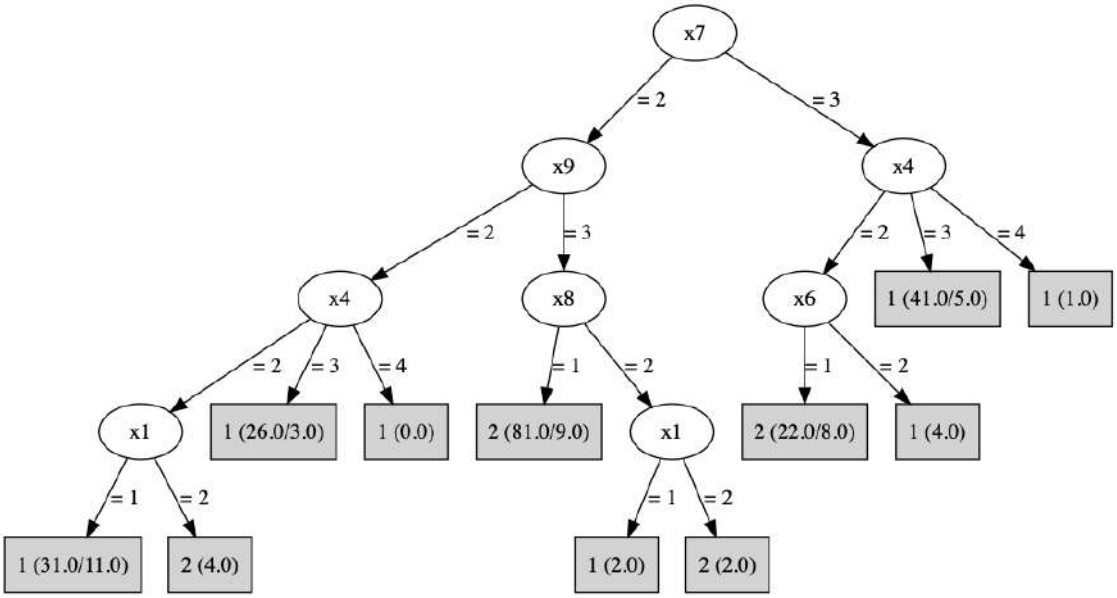


Figura A.26: Árbol de decisión obtenido por eMODiTS para la base de datos Ham.

A.33. HandOutlines

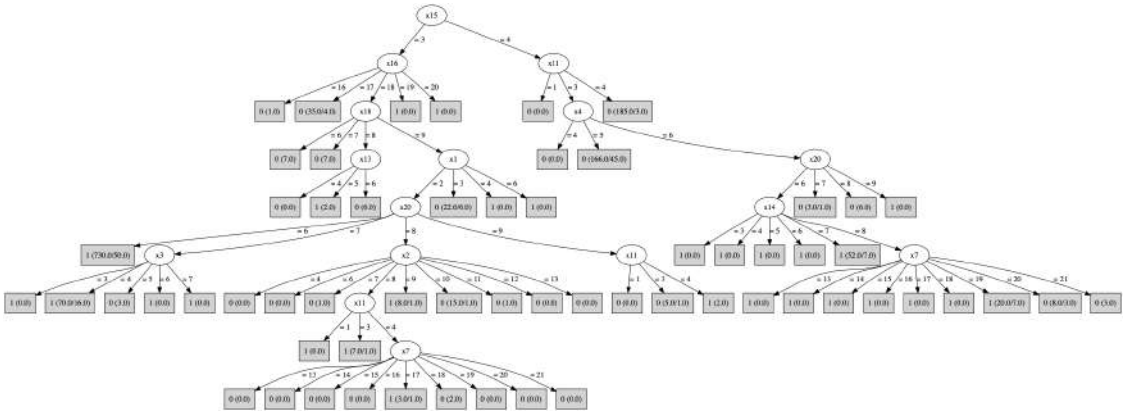


Figura A.27: Árbol de decisión obtenido por eMODiTS para la base de datos HandOutlines.

A.34. Haptics

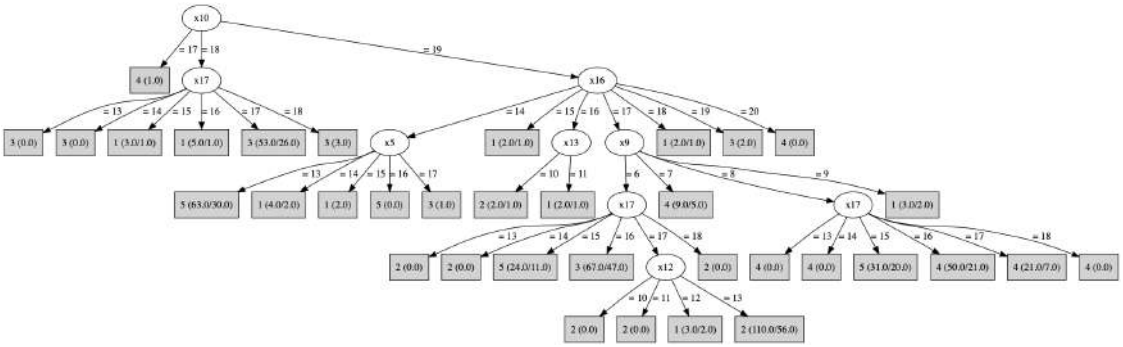


Figura A.28: Árbol de decisión obtenido por eMODiTS para la base de datos Haptics.

A.35. Herring

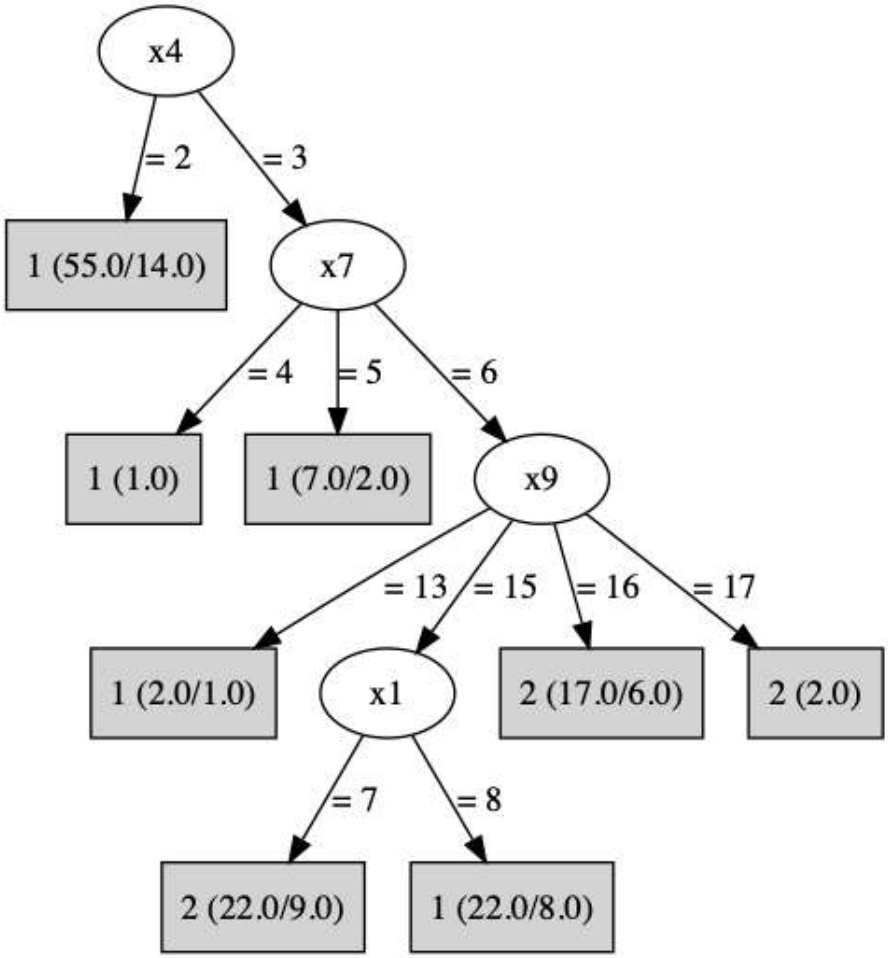


Figura A.29: Árbol de decisión obtenido por eMODiTS para la base de datos Herring.

A.36. InlineSkate

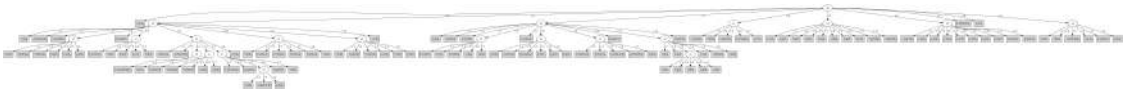


Figura A.30: Árbol de decisión obtenido por eMODiTS para la base de datos InlineSkate.

A.37. InsectWingbeatSound

Figura A.31: Árbol de decisión obtenido por eMODiTS para la base de datos InsectWingbeatSound.

A.38. ItalyPowerDemand

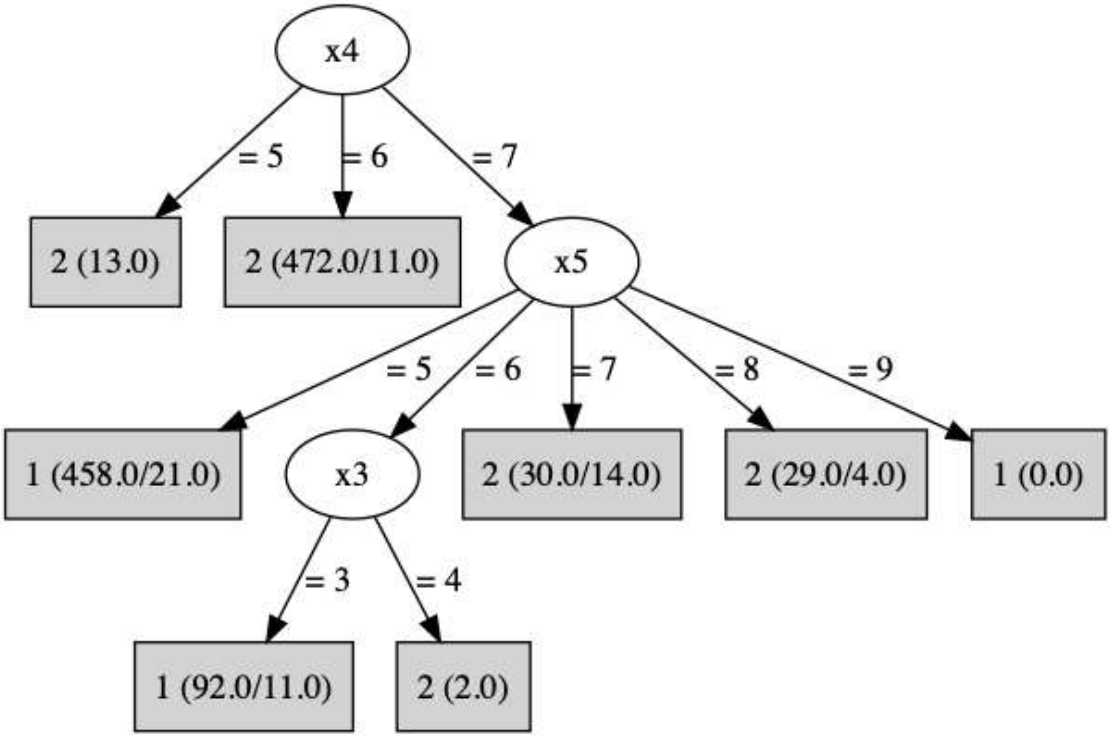


Figura A.32: Árbol de decisión obtenido por eMODiTS para la base de datos ItalyPowerDemand.

A.39. LargeKitchenAppliances

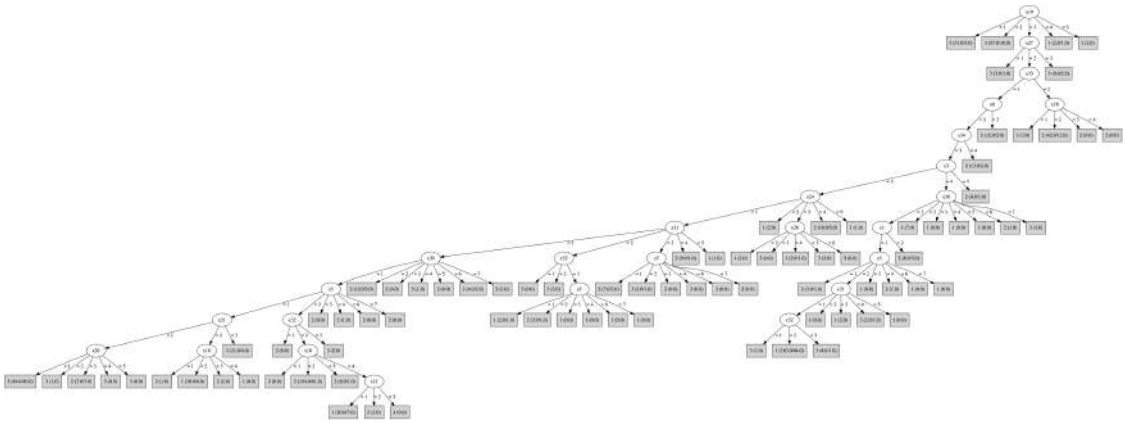


Figura A.33: Árbol de decisión obtenido por eMODiTS para la base de datos LargeKitchenAppliances.

A.40. Lighting2

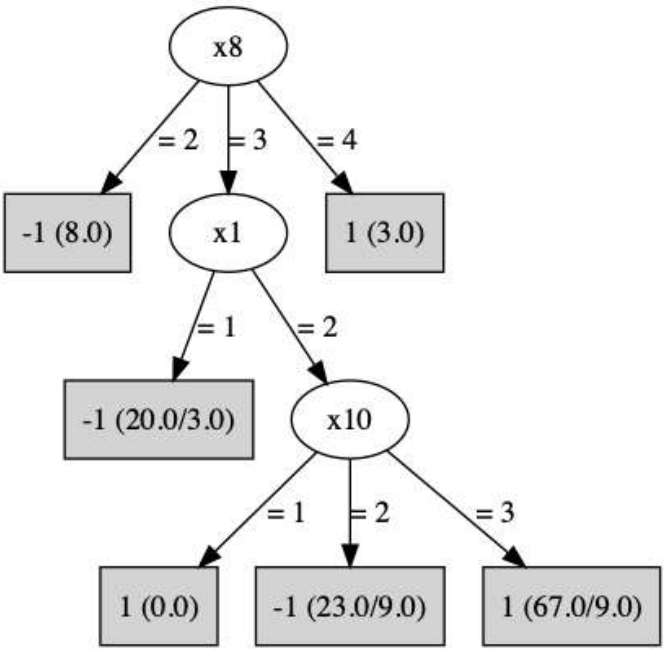


Figura A.34: Árbol de decisión obtenido por eMODiTS para la base de datos Lighting2.

A.41. Lighting7

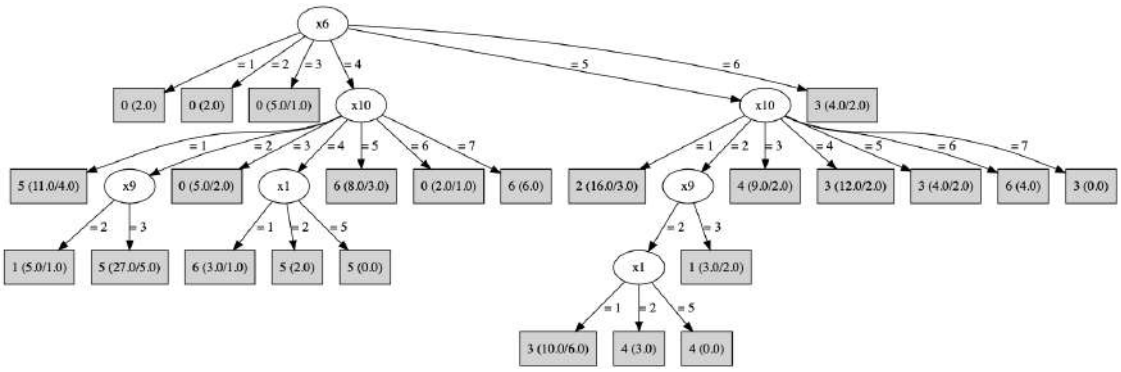


Figura A.35: Árbol de decisión obtenido por eMODiTS para la base de datos Lighting7.

A.42. Mallat

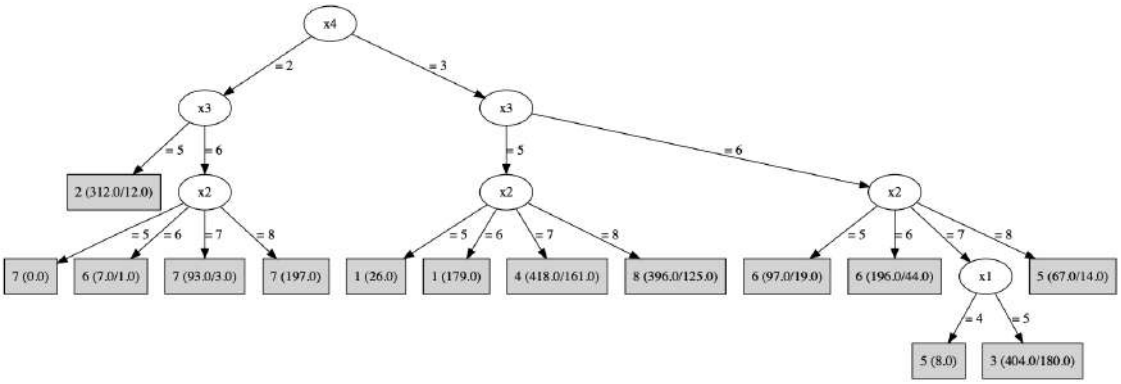


Figura A.36: Árbol de decisión obtenido por eMODiTS para la base de datos Mallat.

A.43. Meat

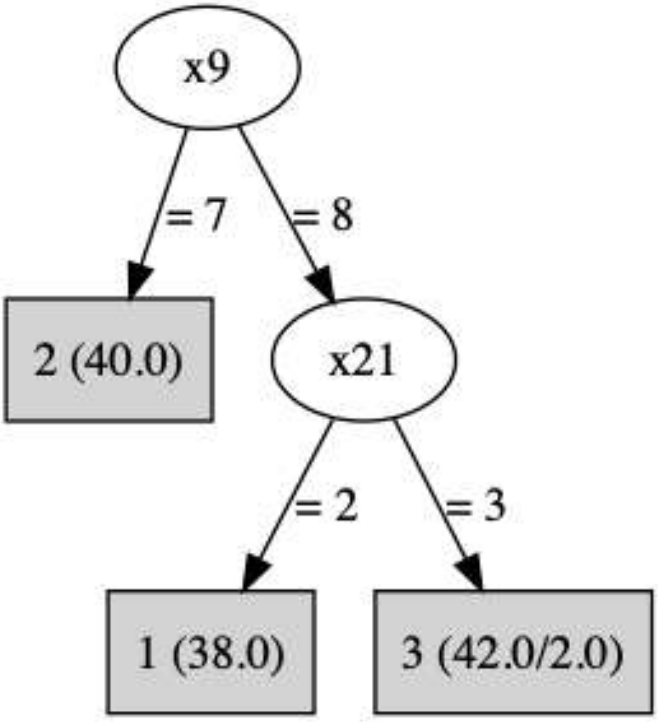


Figura A.37: Árbol de decisión obtenido por eMODiTS para la base de datos Meat.

A.44. MedicalImages

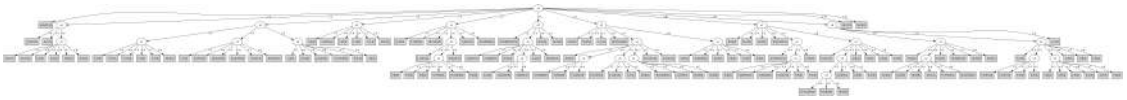


Figura A.38: Árbol de decisión obtenido por eMODiTS para la base de datos MedicalImages.

A.45. MiddlePhalanxOutlineAgeGroup

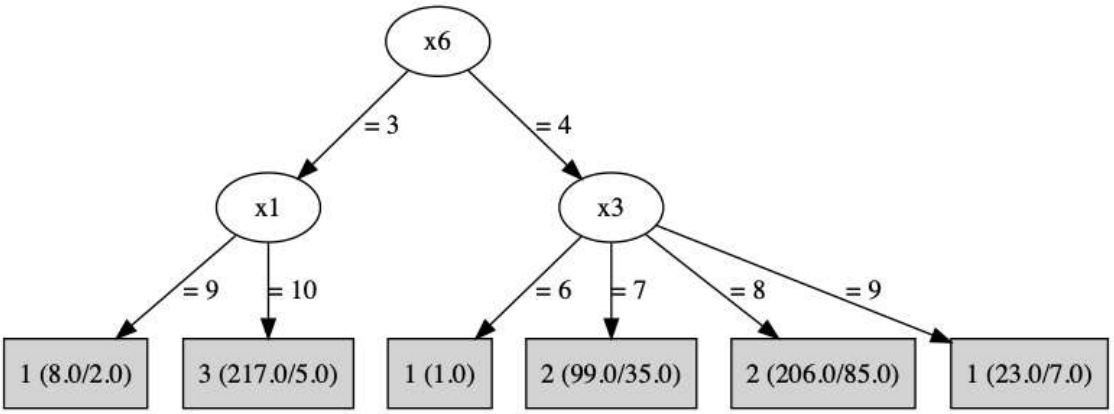


Figura A.39: Árbol de decisión obtenido por eMODiTS para la base de datos MiddlePhalanxOutlineAgeGroup.

A.46. MiddlePhalanxOutlineCorrect

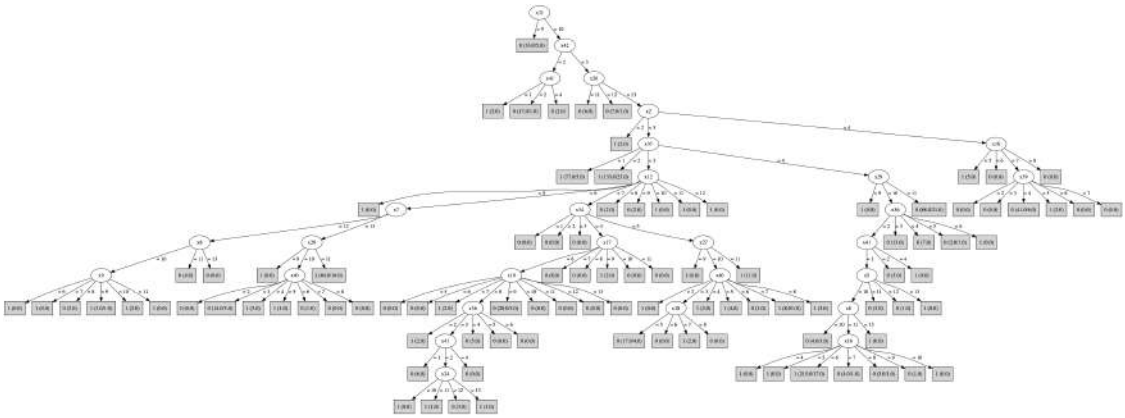


Figura A.40: Árbol de decisión obtenido por eMODiTS para la base de datos MiddlePhalanxOutlineCorrect.

A.47. MiddlePhalanxTW

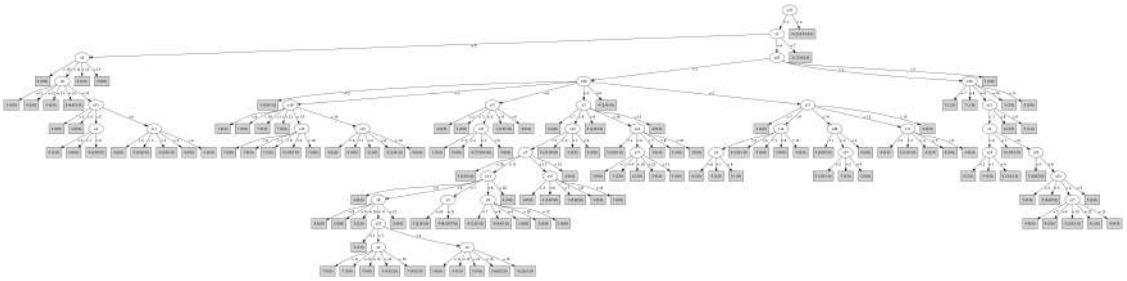


Figura A.41: Árbol de decisión obtenido por eMODiTS para la base de datos MiddlePhalanxTW.

A.48. MoteStrain

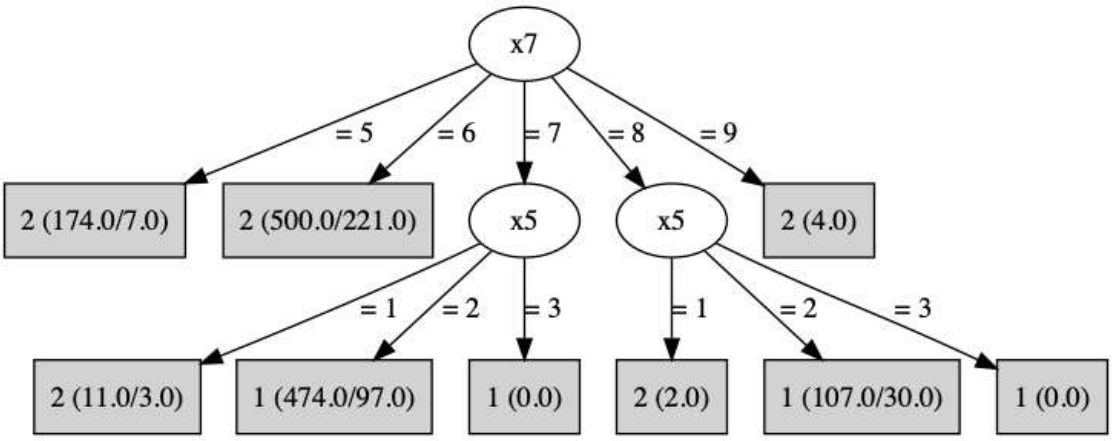


Figura A.42: Árbol de decisión obtenido por eMODiTS para la base de datos MoteStrain.

A.49. NonInvasiveFetalECGThorax1

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.50. NonInvasiveFetalECGThorax2

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.51. OliveOil

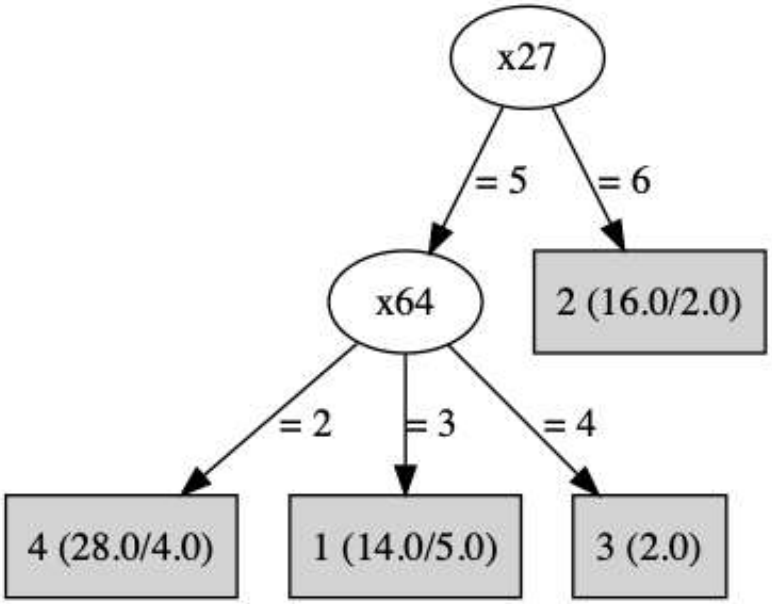


Figura A.43: Árbol de decisión obtenido por eMODiTS para la base de datos OliveOil.

A.52. OSULeaf

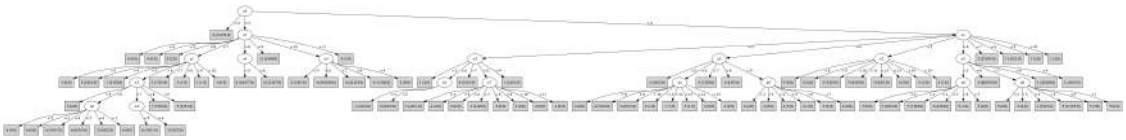


Figura A.44: Árbol de decisión obtenido por eMODiTS para la base de datos OSULeaf.

A.53. PhalangesOutlinesCorrect

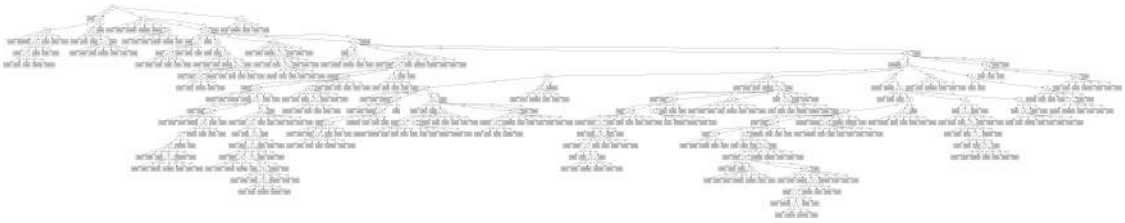


Figura A.45: Árbol de decisión obtenido por eMODiTS para la base de datos PhalangesOutlinesCorrect.

A.54. Phoneme

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.55. Plane

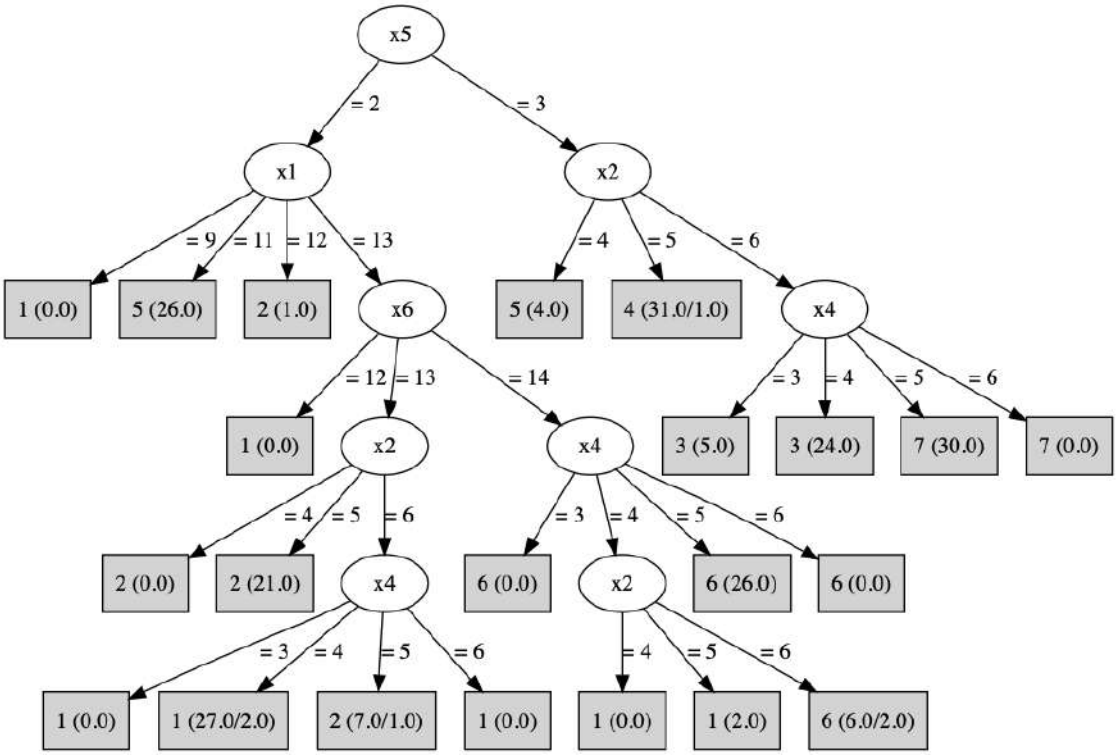


Figura A.46: Árbol de decisión obtenido por eMODiTS para la base de datos Plane.

A.56. ProximalPhalanxOutlineAgeGroup

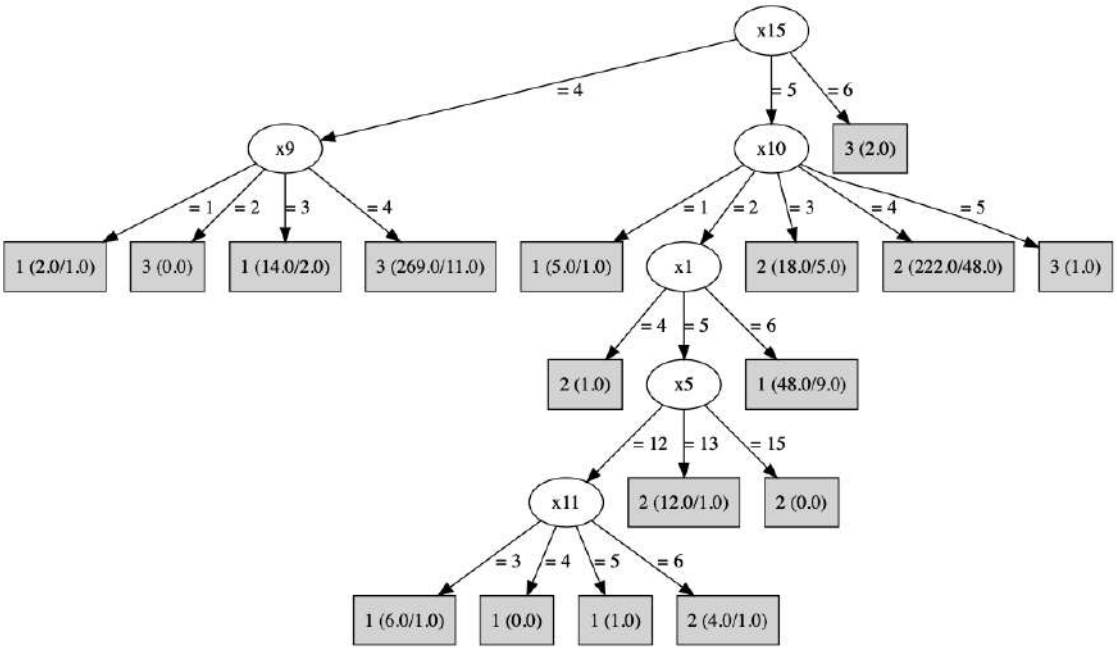


Figura A.47: Árbol de decisión obtenido por eMODiTS para la base de datos ProximalPhalanxOutlineAgeGroup.

A.57. ProximalPhalanxOutlineCorrect

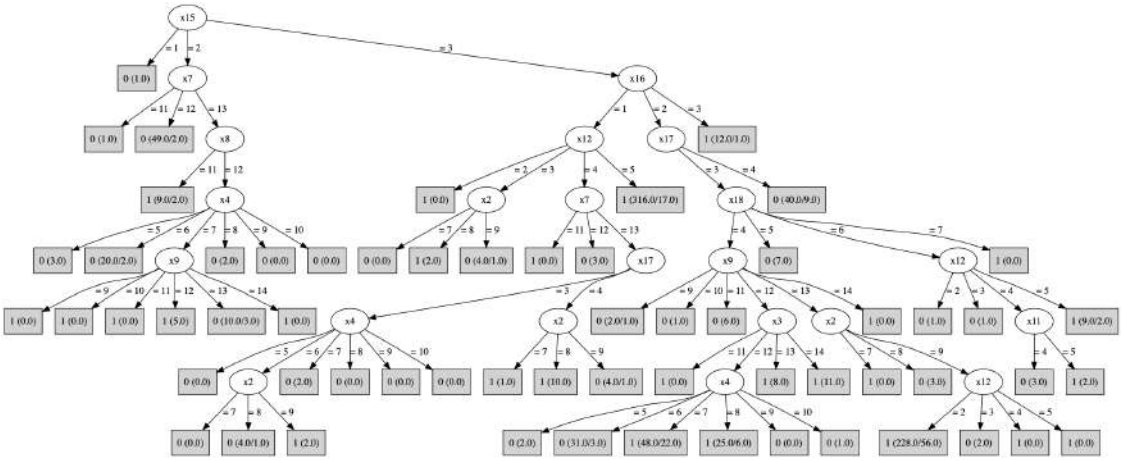


Figura A.48: Árbol de decisión obtenido por eMODiTS para la base de datos ProximalPhalanxOutlineCorrect.

A.58. ProximalPhalanxTW

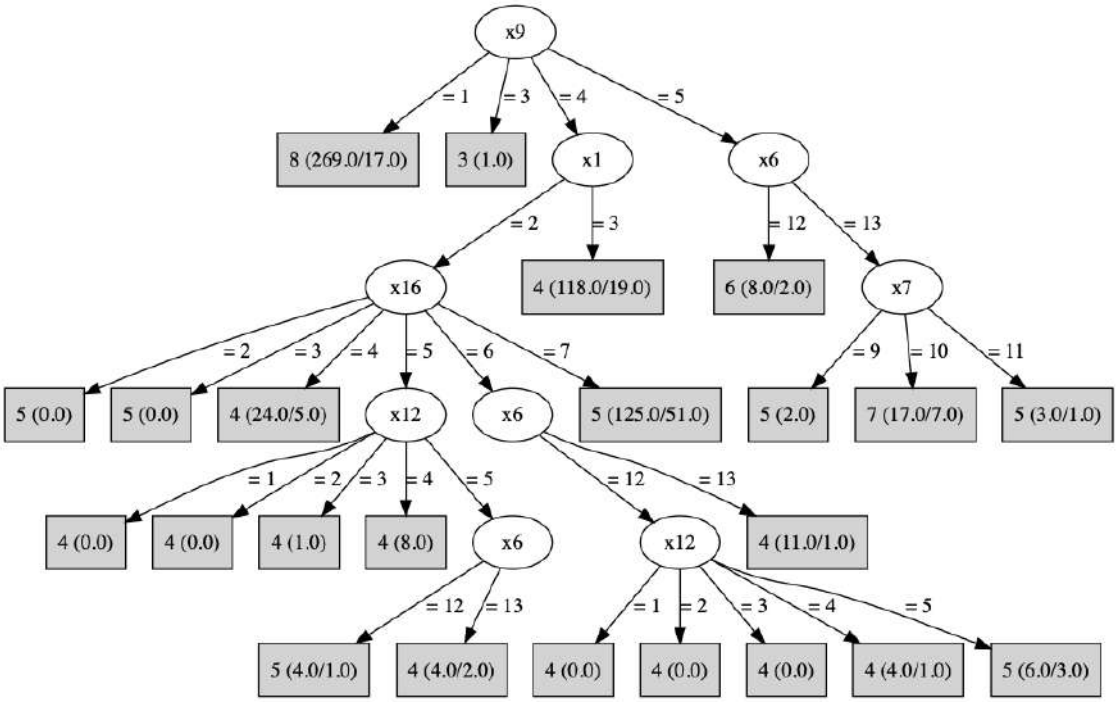


Figura A.49: Árbol de decisión obtenido por eMODiTS para la base de datos ProximalPhalanxTW.

A.59. RefrigerationDevices

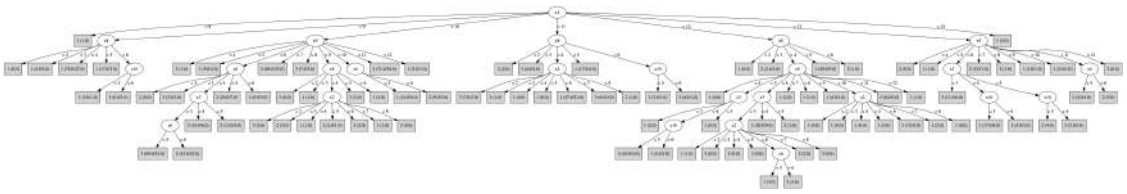


Figura A.50: Árbol de decisión obtenido por eMODiTS para la base de datos RefrigerationDevices.

A.60. ScreenType

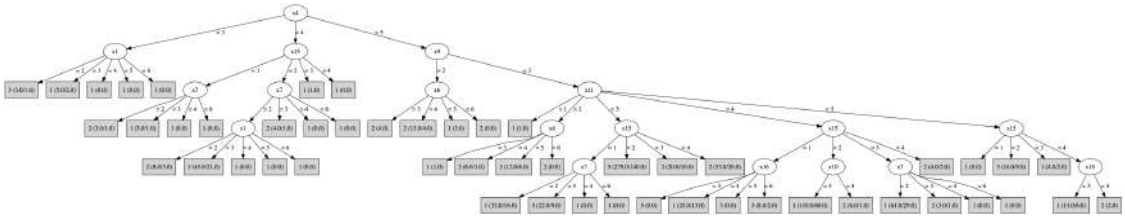


Figura A.51: Árbol de decisión obtenido por eMODiTS para la base de datos ScreenType.

A.61. ShapeletSim

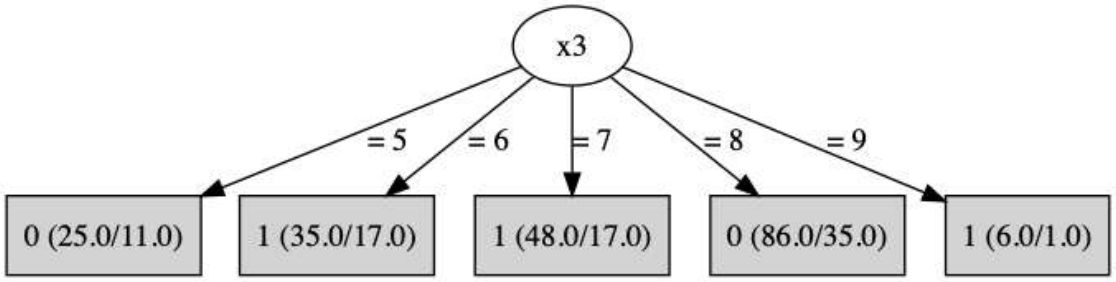


Figura A.52: Árbol de decisión obtenido por eMODiTS para la base de datos ShapeletSim.

A.62. ShapesAll

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.63. SmallKitchenAppliances

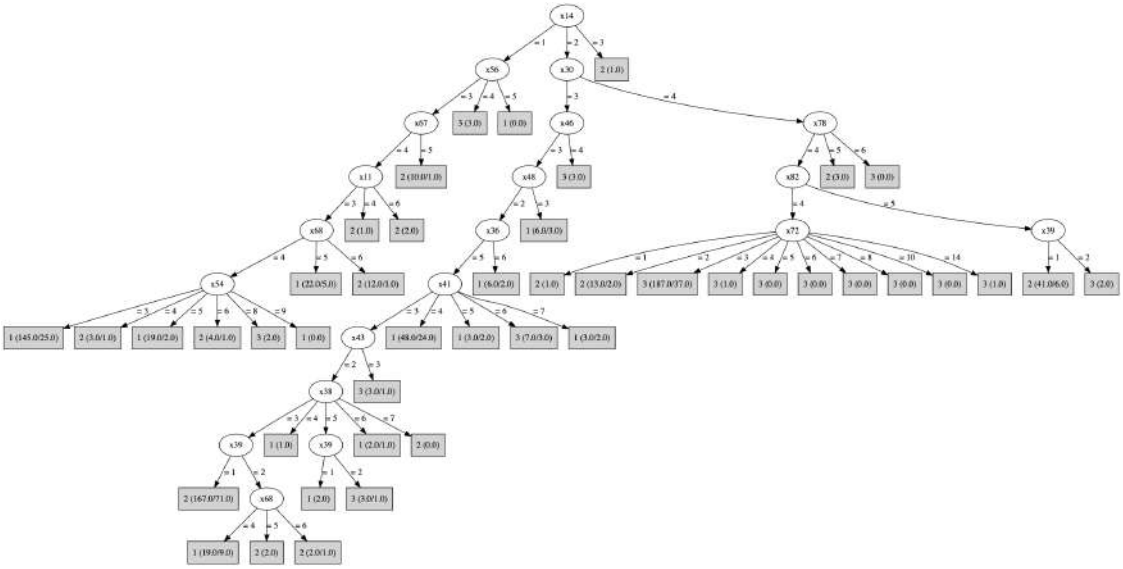


Figura A.53: Árbol de decisión obtenido por eMODiTS para la base de datos SmallKitchenAppliances.

A.64. SonyAIBORobotSurface1

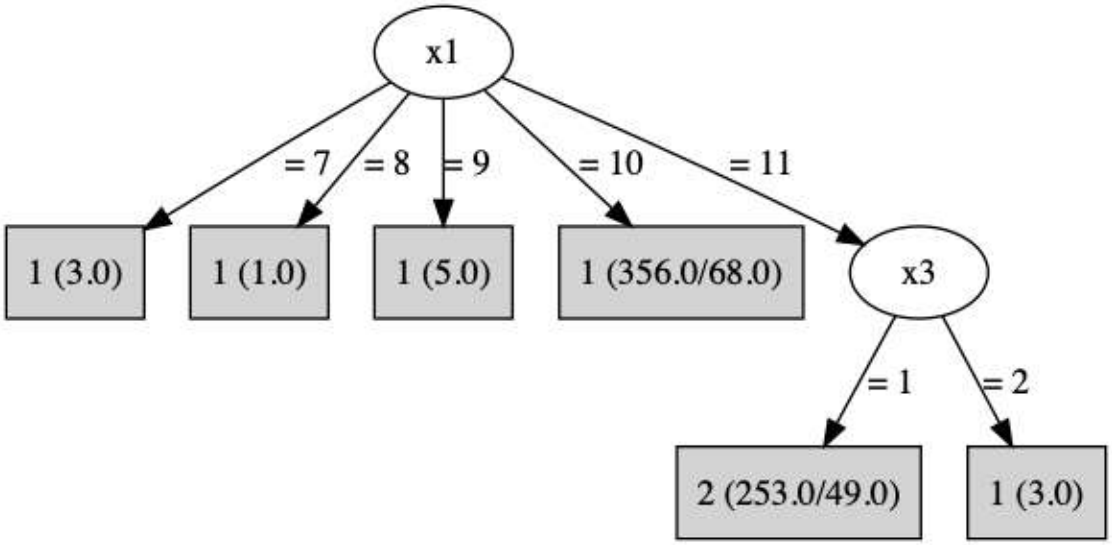


Figura A.54: Árbol de decisión obtenido por eMODiTS para la base de datos SonyAIBORobotSurface1.

A.65. SonyAIBORobotSurface2

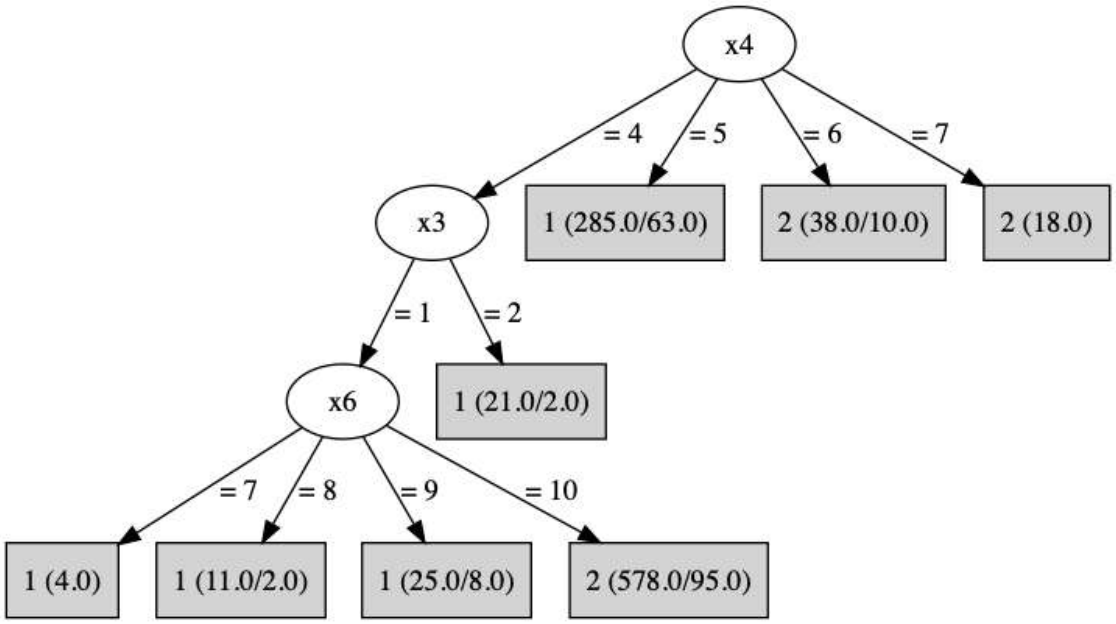


Figura A.55: Árbol de decisión obtenido por eMODiTS para la base de datos SonyAIBORobotSurface2.

A.66. StarLightCurves

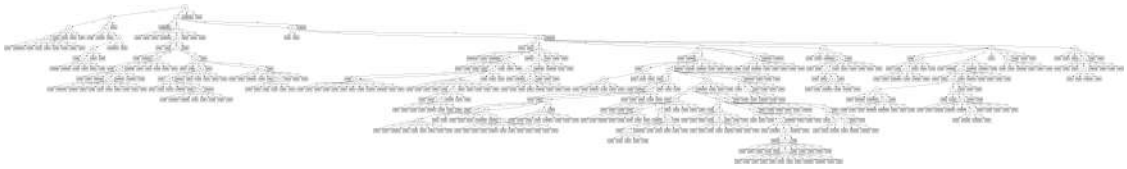


Figura A.56: Árbol de decisión obtenido por eMODiTS para la base de datos StarLightCurves.

A.67. Strawberry

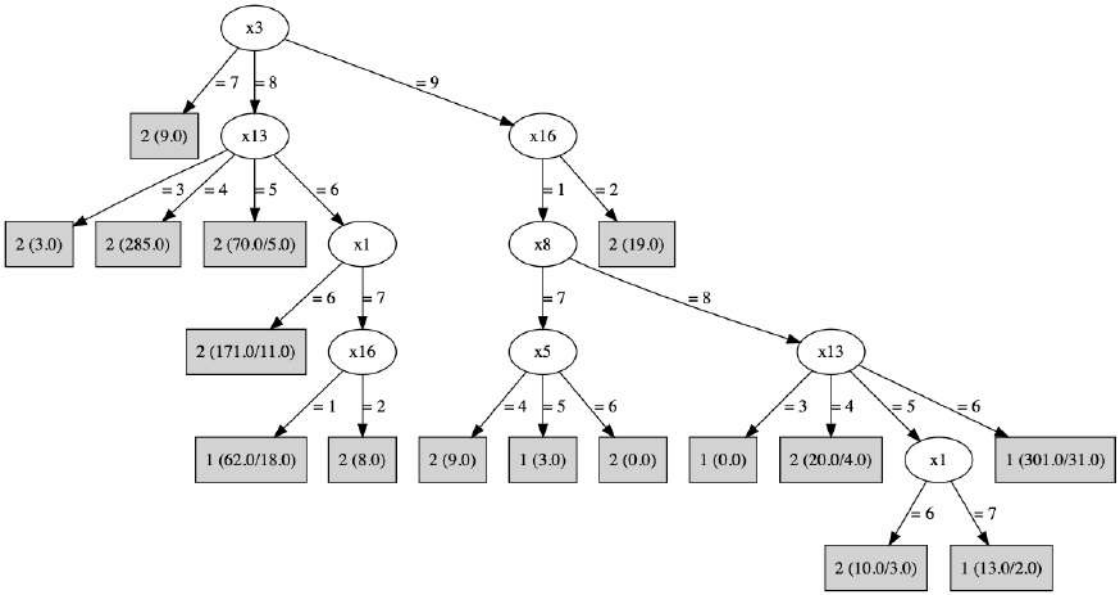


Figura A.57: Árbol de decisión obtenido por eMODiTS para la base de datos Strawberry.

A.68. SwedishLeaf

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.69. Symbols

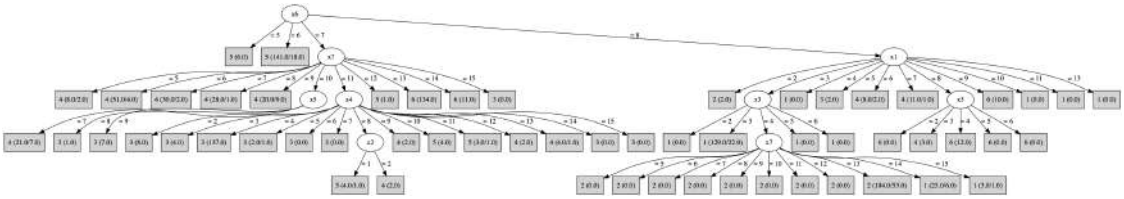


Figura A.58: Árbol de decisión obtenido por eMODiTS para la base de datos Symbols.

A.70. SyntheticControl

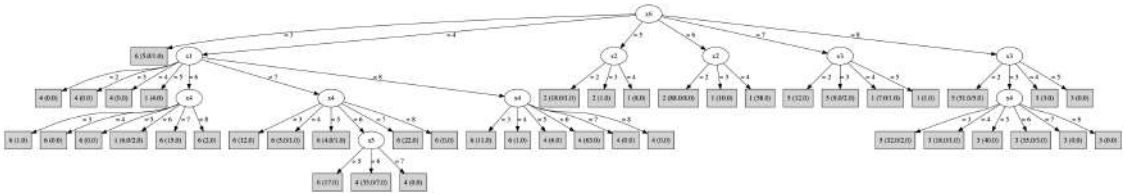


Figura A.59: Árbol de decisión obtenido por eMODiTS para la base de datos SyntheticControl.

A.71. ToeSegmentation1

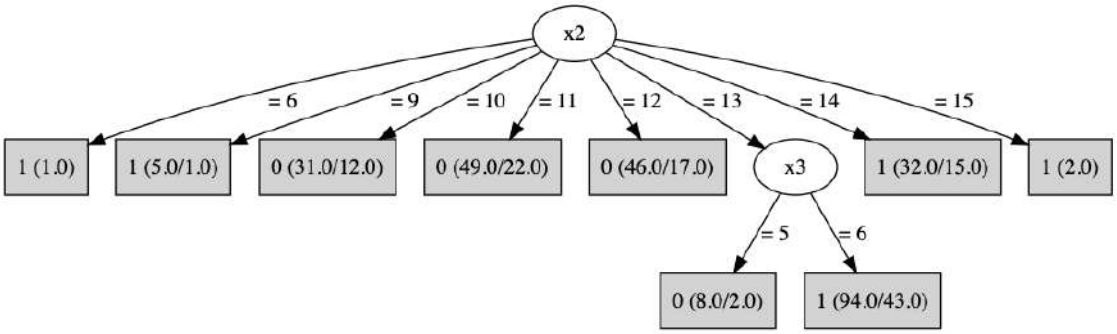


Figura A.60: Árbol de decisión obtenido por eMODiTS para la base de datos ToeSegmentation1.

A.72. ToeSegmentation2

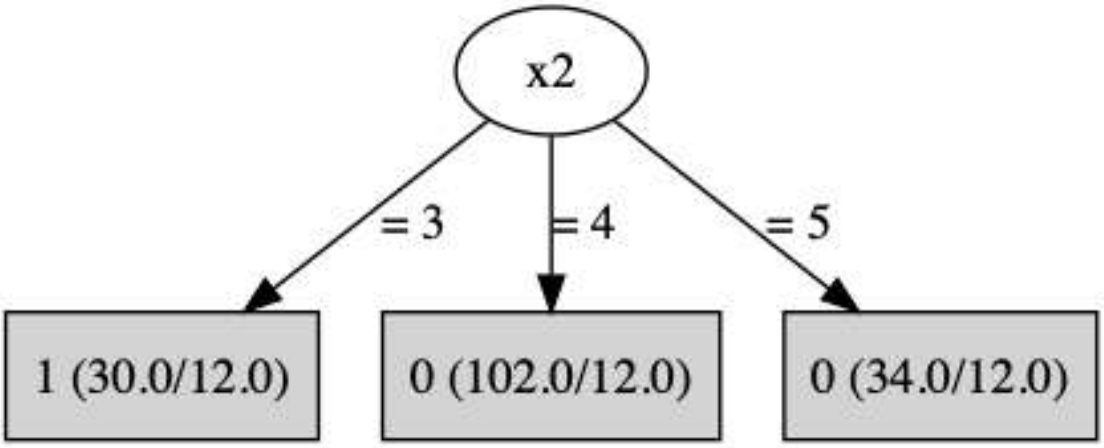


Figura A.61: Árbol de decisión obtenido por eMODiTS para la base de datos ToeSegmentation2.

A.73. Trace

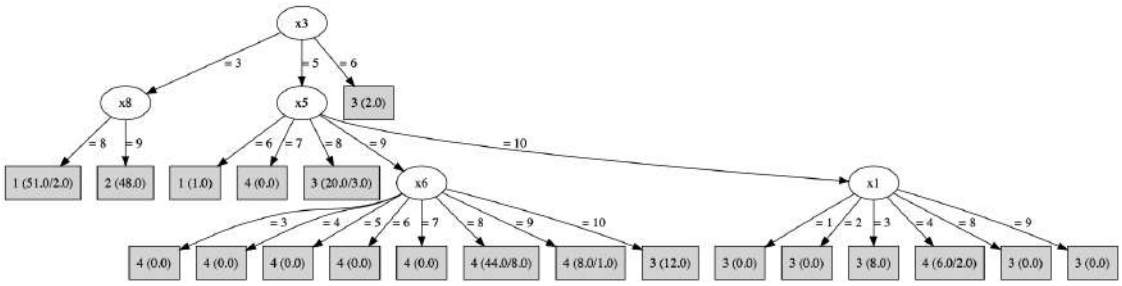


Figura A.62: Árbol de decisión obtenido por eMODiTS para la base de datos Trace.

A.74. TwoLeadECG

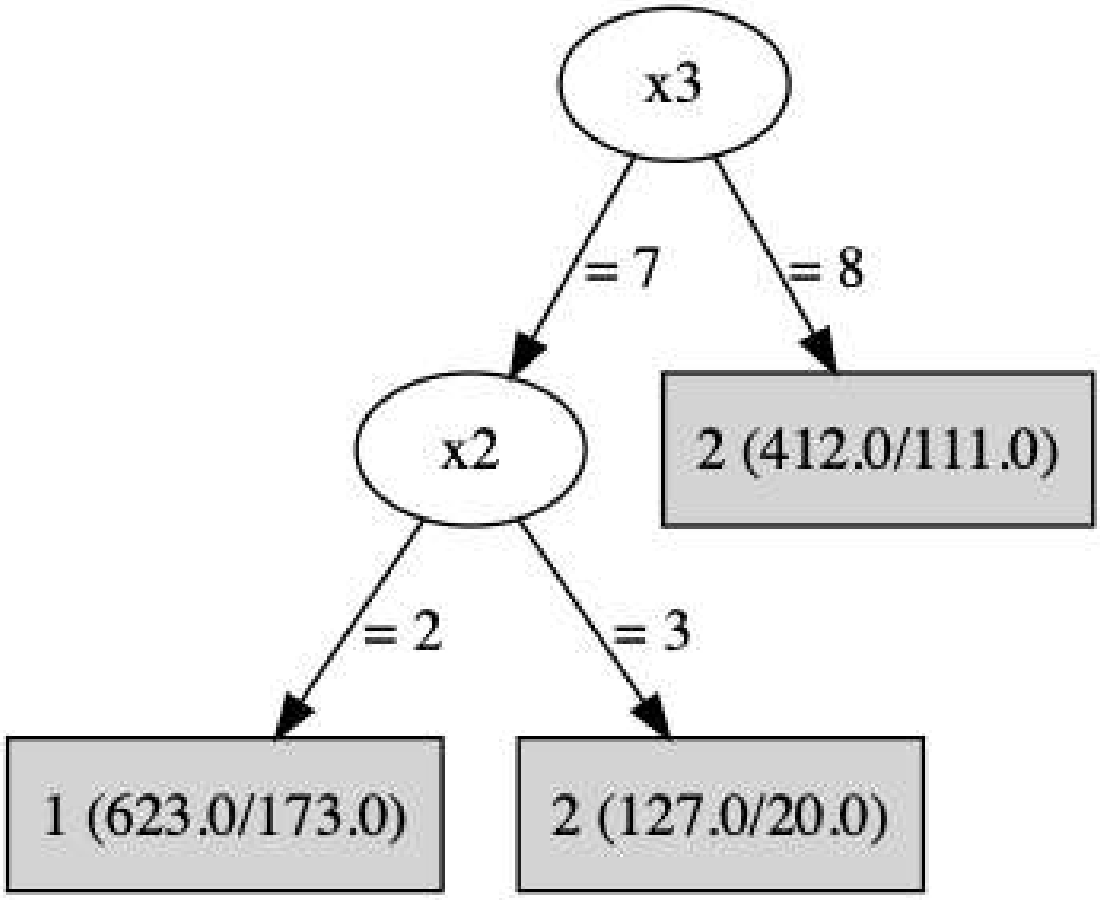


Figura A.63: Árbol de decisión obtenido por eMODiTS para la base de datos TwoLeadECG.

A.75. TwoPatterns

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.76. UWaveGestureLibraryAll

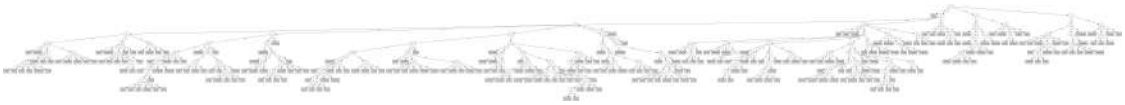


Figura A.64: Árbol de decisión obtenido por eMODiTS para la base de datos UWaveGestureLibraryAll.

A.77. UWaveGestureLibraryX

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.78. UWaveGestureLibraryY

Figura A.65: Árbol de decisión obtenido por eMODiTS para la base de datos UWaveGestureLibraryY.

A.79. UWaveGestureLibraryZ

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

A.80. Wafer

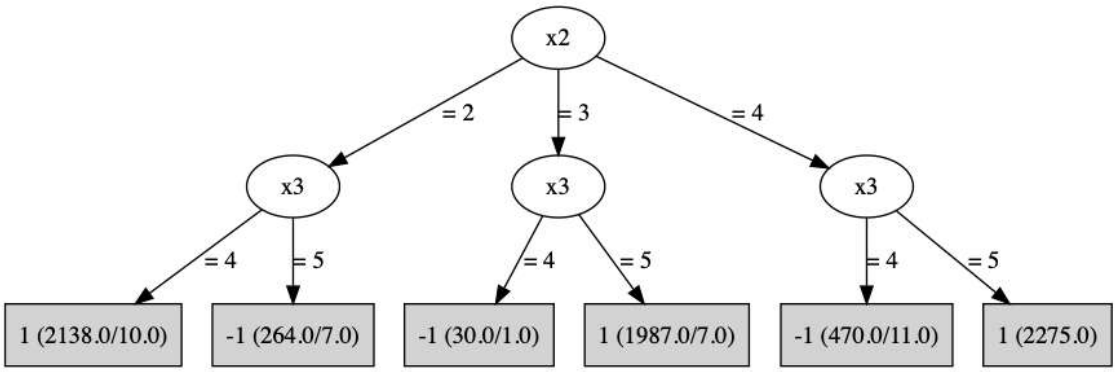


Figura A.66: Árbol de decisión obtenido por eMODiTS para la base de datos Wafer.

A.81. Wine

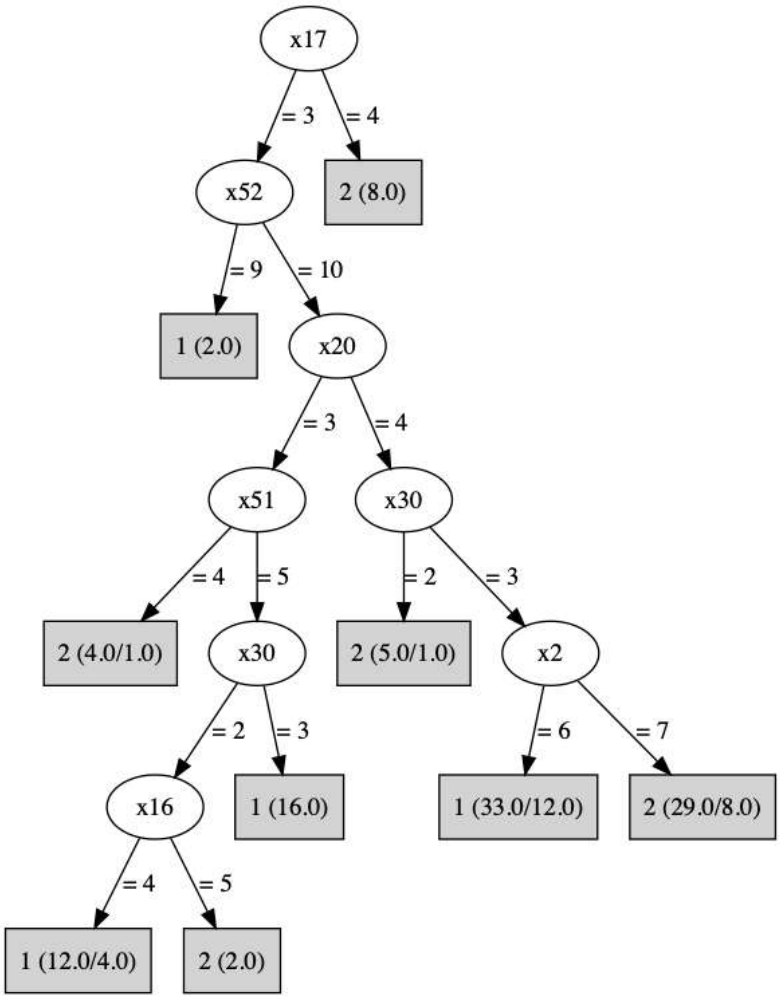


Figura A.67: Árbol de decisión obtenido por eMODiTS para la base de datos Wine.

A.82. WordSynonyms

Figura A.68: Árbol de decisión obtenido por eMODiTS para la base de datos WordSynonyms.

A.83. Worms

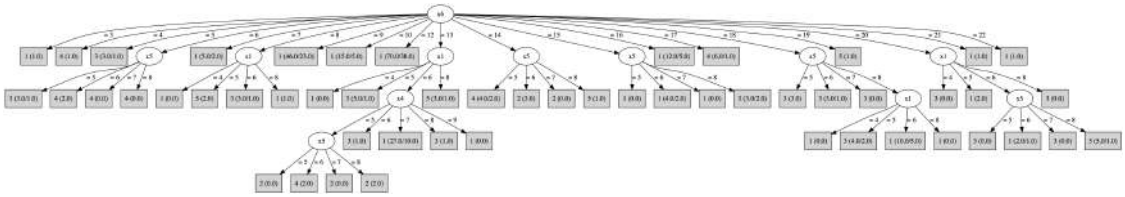


Figura A.69: Árbol de decisión obtenido por eMODiTS para la base de datos Worms.

A.84. WormsTwoClass

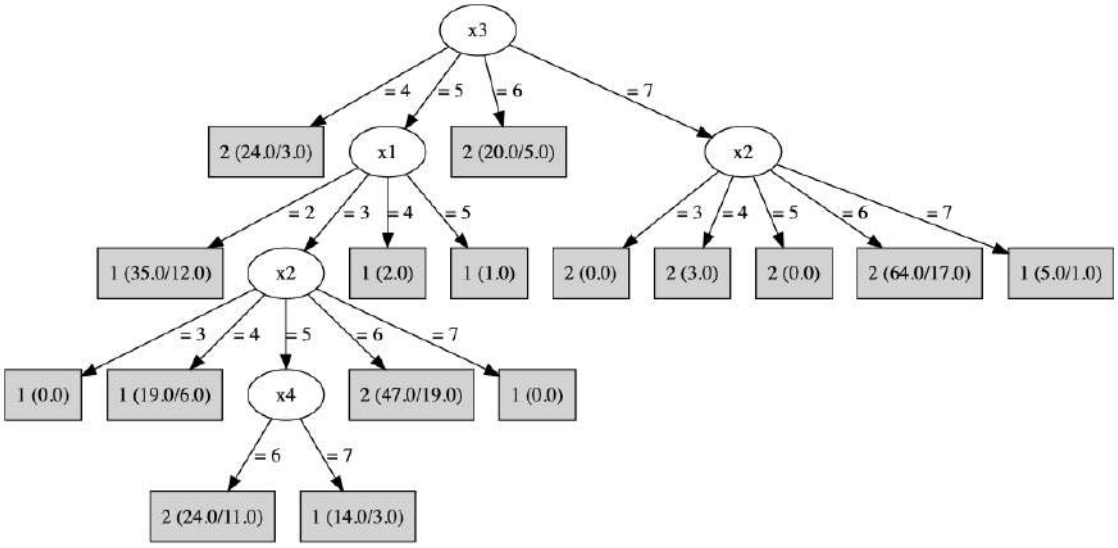


Figura A.70: Árbol de decisión obtenido por eMODiTS para la base de datos WormsTwoClass.

A.85. Yoga

Este árbol, por cuestiones de tamaño, no puede ser desplegado en esta sección por lo que se puede consultar en la página `https://github.com/scoramg/eMODiTS/`.

B

Distribución de clases

Contenido

B.1. Adiac 147
B.2. ArrowHead 148
B.3. Beef 148
B.4. BeetleFly 149
B.5. BirdChicken 149
B.6. Car 150
B.7. CBF 150
B.8. ChlorineConcentration 151
B.9. CinCECGtorso 151
B.10. Coffee 152
B.11. Computers 152
B.12. CricketX 153
B.13. CricketY 153
B.14. CricketZ 154
B.15. DiatomSizeReduction 154
B.16. DistalPhalanxOutlineAgeGroup 155
B.17. DistalPhalanxOutlineCorrect 155
B.18. DistalPhalanxTW 156
B.19. Earthquakes 156
B.20. ECG200 157
B.21. ECG5000 157
B.22. ECGFiveDays 158
B.23. ElectricDevices 158
B.24. FaceAll 159
B.25. FaceFour 159
B.26. FacesUCR 160
B.27. FiftyWords 160
B.28. Fish 161

B.29.FordA 161
B.30.FordB 162
B.31.GunPoint 162
B.32.Ham 163
B.33.HandOutlines 163
B.34.Haptics 164
B.35.Herring 164
B.36.InlineSkate 165
B.37.InsectWingbeatSound 165
B.38.ItalyPowerDemand 166
B.39.LargeKitchenAppliances 166
B.40.Lighting2 167
B.41.Lighting7 167
B.42.Mallat 168
B.43.Meat 168
B.44.MedicalImages 169
B.45.MiddlePhalanxOutlineAgeGroup 169
B.46.MiddlePhalanxOutlineCorrect 170
B.47.MiddlePhalanxTW 170
B.48.MoteStrain 171
B.49.NonInvasiveFetalECGThorax1 171
B.50.NonInvasiveFetalECGThorax2 172
B.51.OliveOil 172
B.52.OSULeaf 173
B.53.PhalangesOutlinesCorrect 173
B.54.Phoneme 174
B.55.Plane 174
B.56.ProximalPhalanxOutlineAgeGroup 175
B.57.ProximalPhalanxOutlineCorrect 175
B.58.ProximalPhalanxTW 176
B.59.RefrigerationDevices 176
B.60.ScreenType 177
B.61.ShapeletSim 177
B.62.ShapesAll 178
B.63.SmallKitchenAppliances 178
B.64.SonyAIBORobotSurface1 179
B.65.SonyAIBORobotSurface2 179
B.66.StarLightCurves 180
B.67.Strawberry 180
B.68.SwedishLeaf 181
B.69.Symbols 181
B.70.SyntheticControl 182
B.71.ToeSegmentation1 182
B.72.ToeSegmentation2 183
B.73.Trace 183
B.74.TwoLeadECG 184
B.75.TwoPatterns 184
B.76.UWaveGestureLibraryAll 185

B.77. UWaveGestureLibraryX **185**
B.78. UWaveGestureLibraryY **186**
B.79. UWaveGestureLibraryZ **186**
B.80. Wafer **187**
B.81. Wine **187**
B.82. WordSynonyms **188**
B.83. Worms **188**
B.84. WormsTwoClass **189**
B.85. Yoga **189**

En este apartado se presentan las distribuciones de las clases por cada base de datos, donde los rectángulos indican un nodo hoja del árbol de decisión, es decir, una clase. Cada uno de los gráficos mostrados pueden ser consultados en la página `https://github.com/scoramg/eMODiTS/` para mayor detalle.

B.1. Adiac

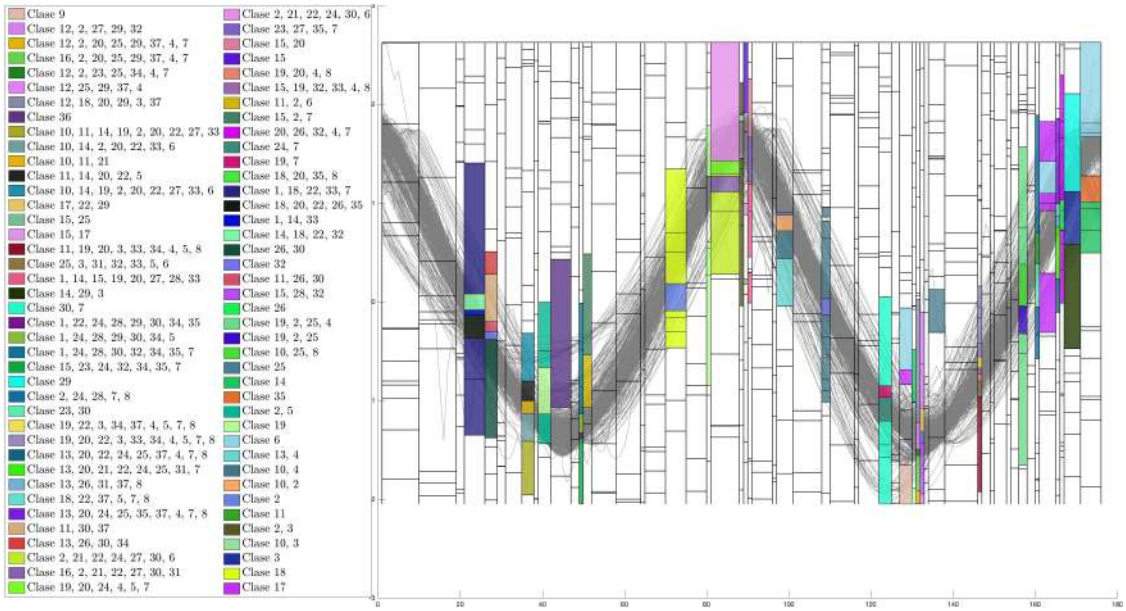


Figura B.1: Distribución de las clases para la base de datos Adiac extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.2. ArrowHead

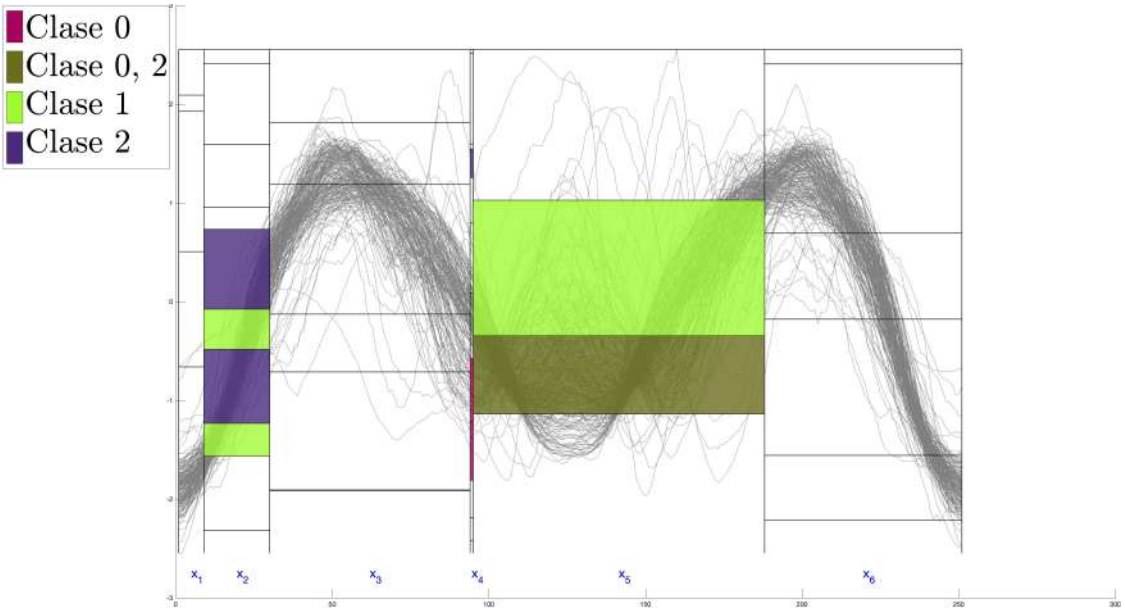


Figura B.2: Distribución de las clases para la base de datos ArrowHead extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.3. Beef

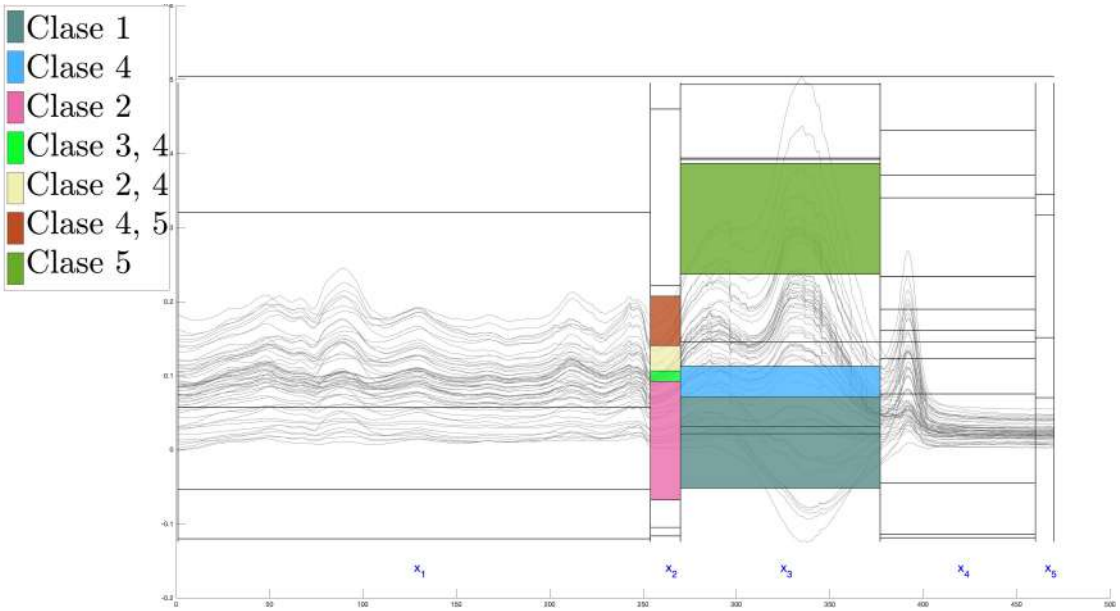


Figura B.3: Distribución de las clases para la base de datos Beef extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.4. BeetleFly

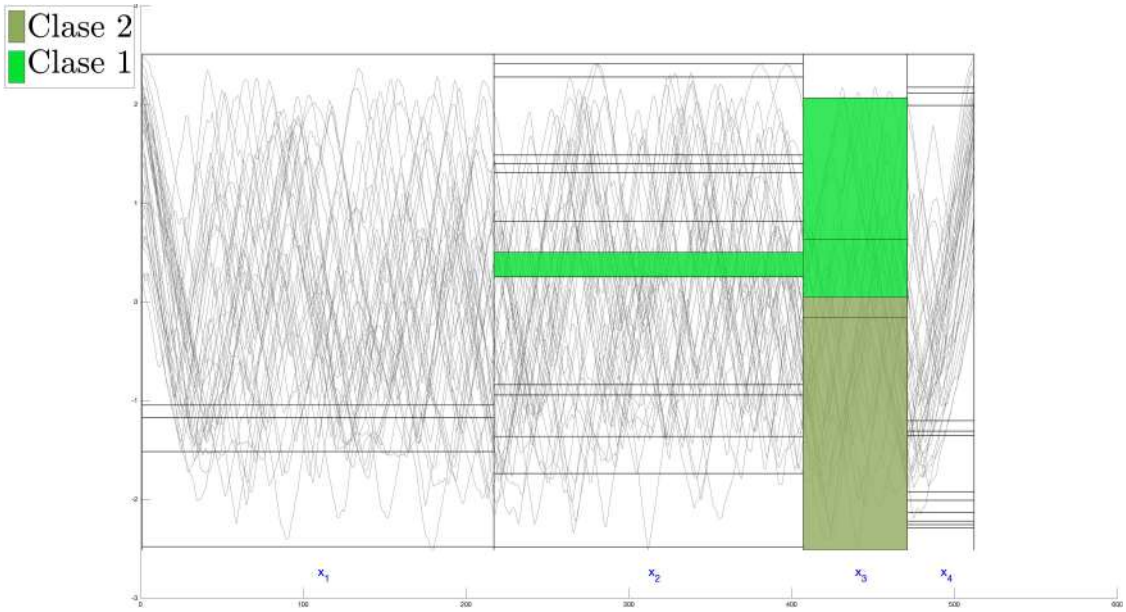


Figura B.4: Distribución de las clases para la base de datos BeetleFly extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.5. BirdChicken

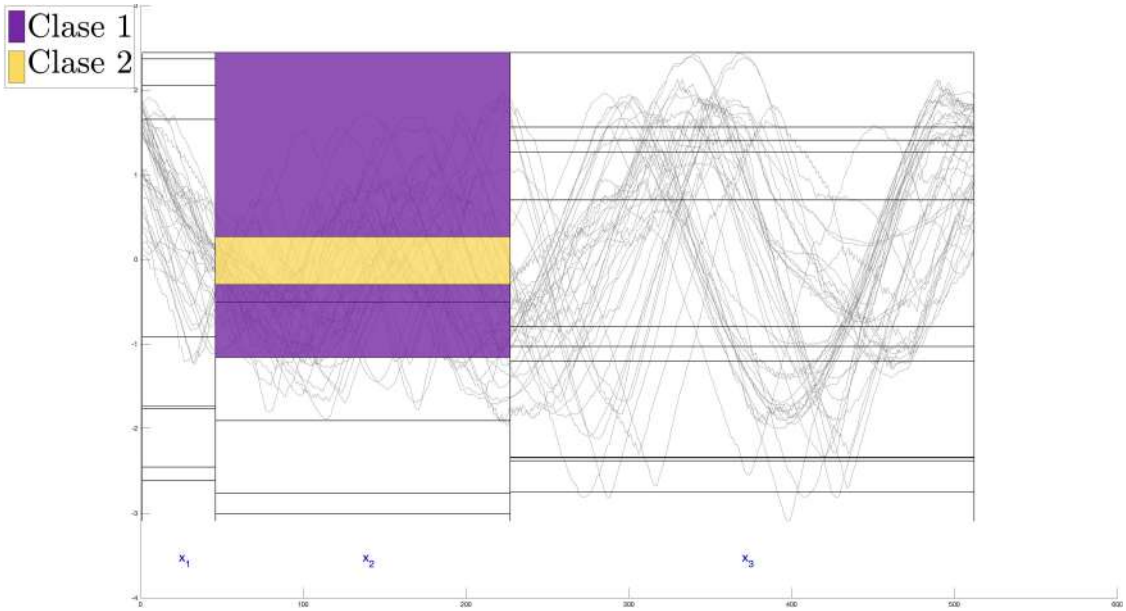


Figura B.5: Distribución de las clases para la base de datos BirdChicken extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.6. Car

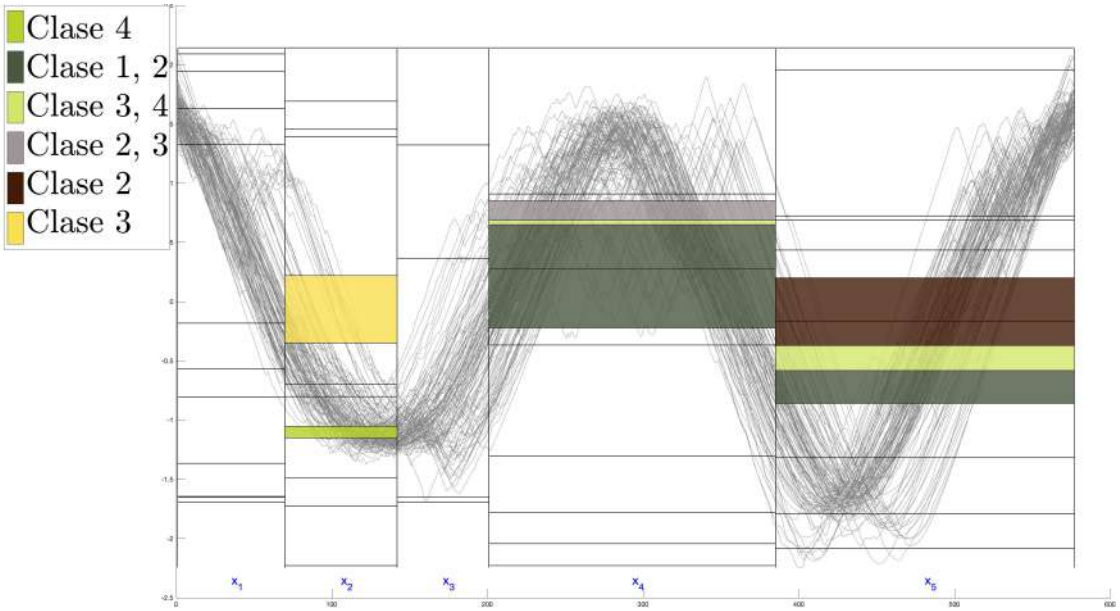


Figura B.6: Distribución de las clases para la base de datos Car extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.7. CBF

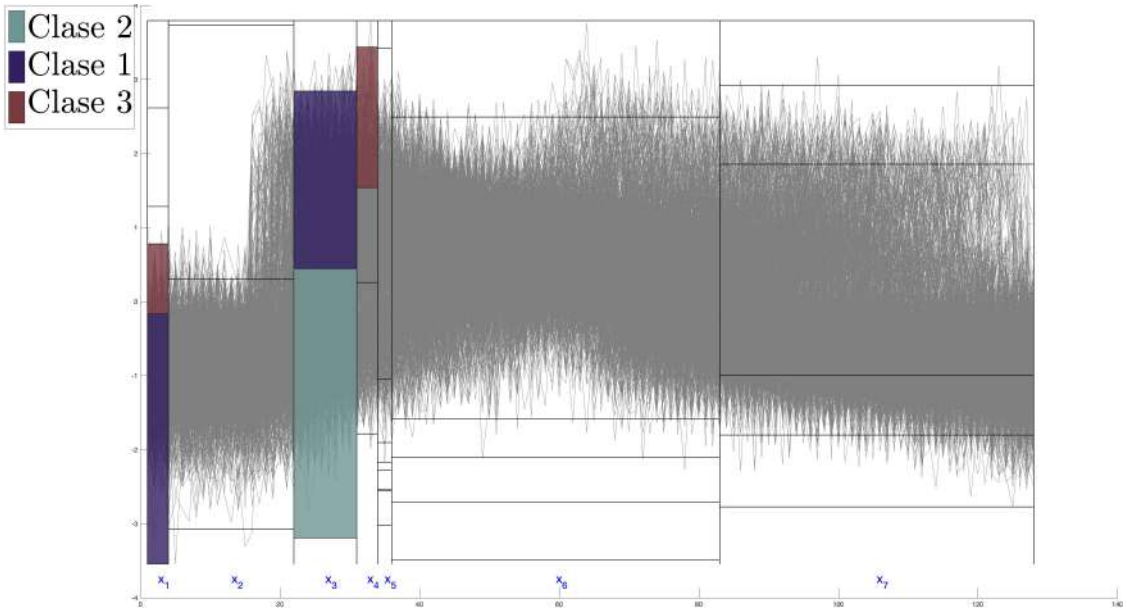


Figura B.7: Distribución de las clases para la base de datos CBF extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.8. ChlorineConcentration

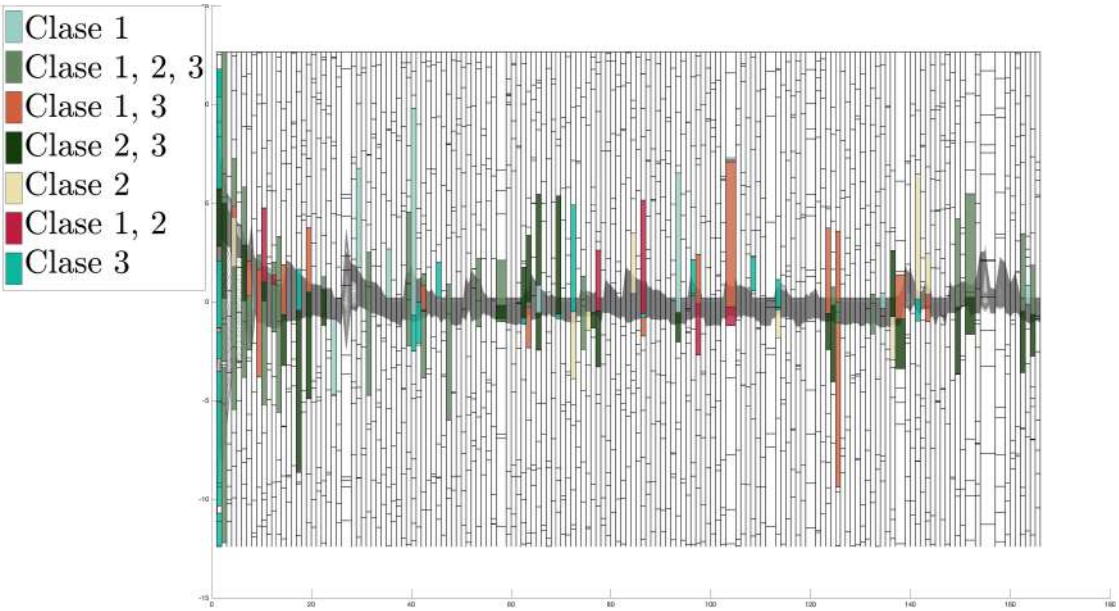


Figura B.8: Distribución de las clases para la base de datos ChlorineConcentration extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.9. CinCECGtorso

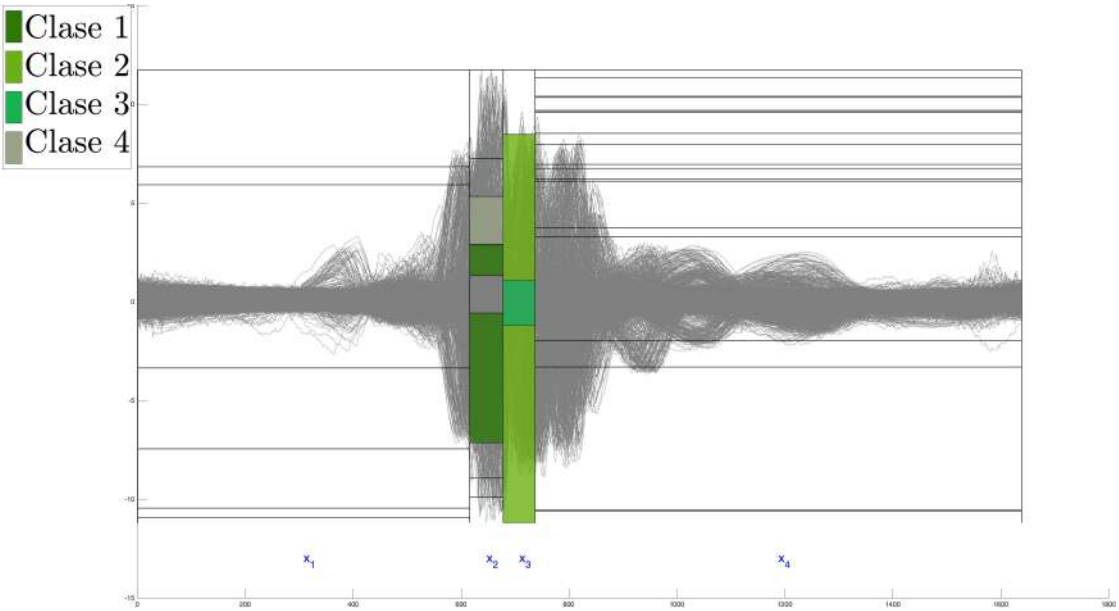


Figura B.9: Distribución de las clases para la base de datos CinCECGtorso extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.10. Coffee

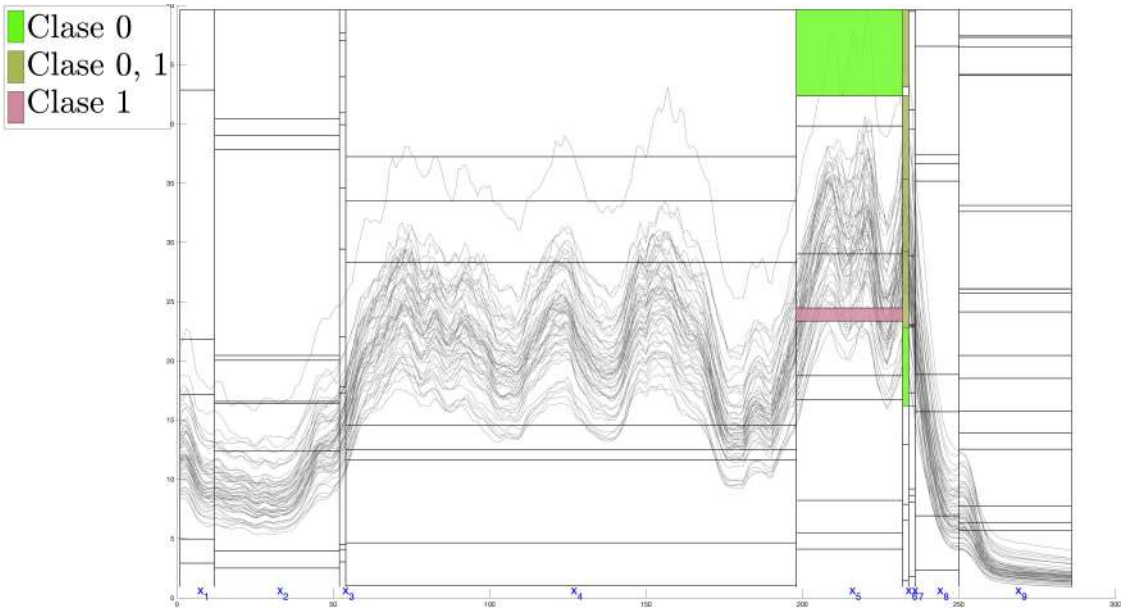


Figura B.10: Distribución de las clases para la base de datos Coffee extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.11. Computers

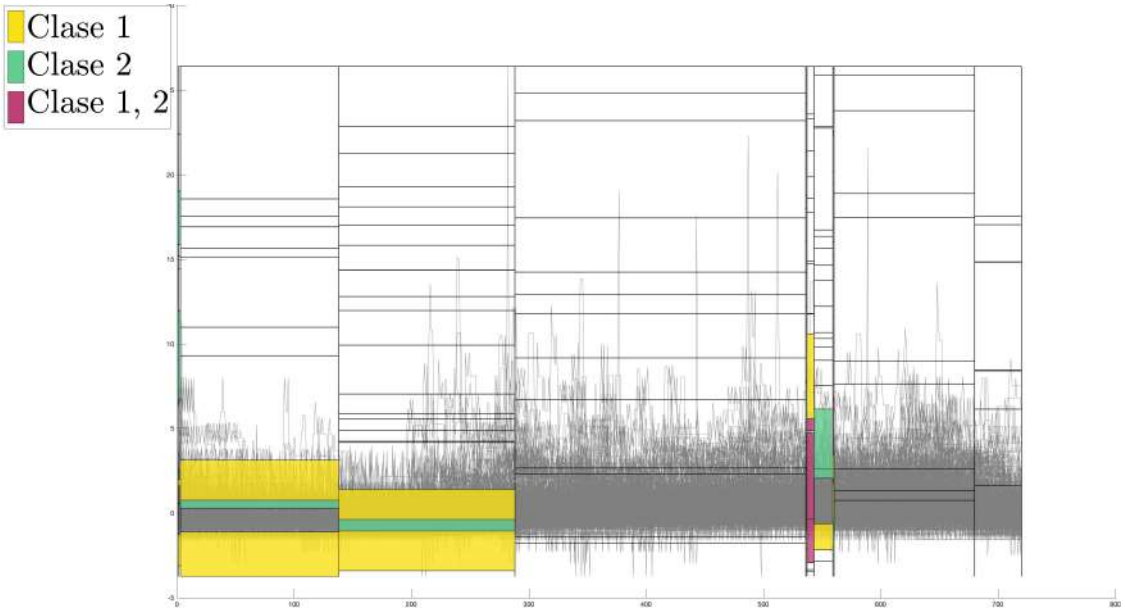


Figura B.11: Distribución de las clases para la base de datos Computers extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.12. CricketX

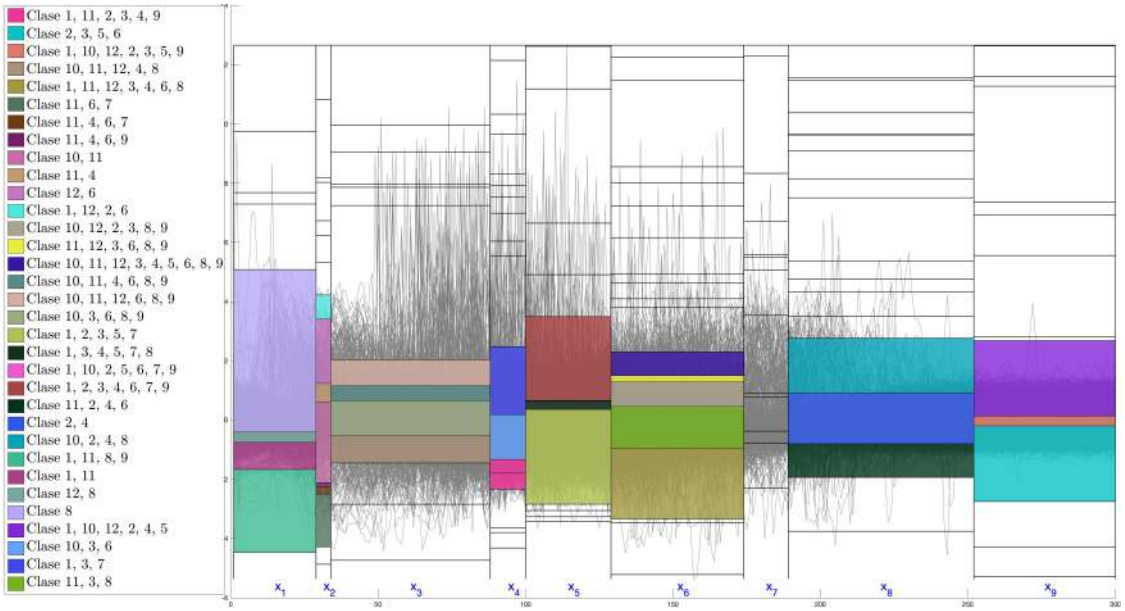


Figura B.12: Distribución de las clases para la base de datos CricketX extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.13. CricketY

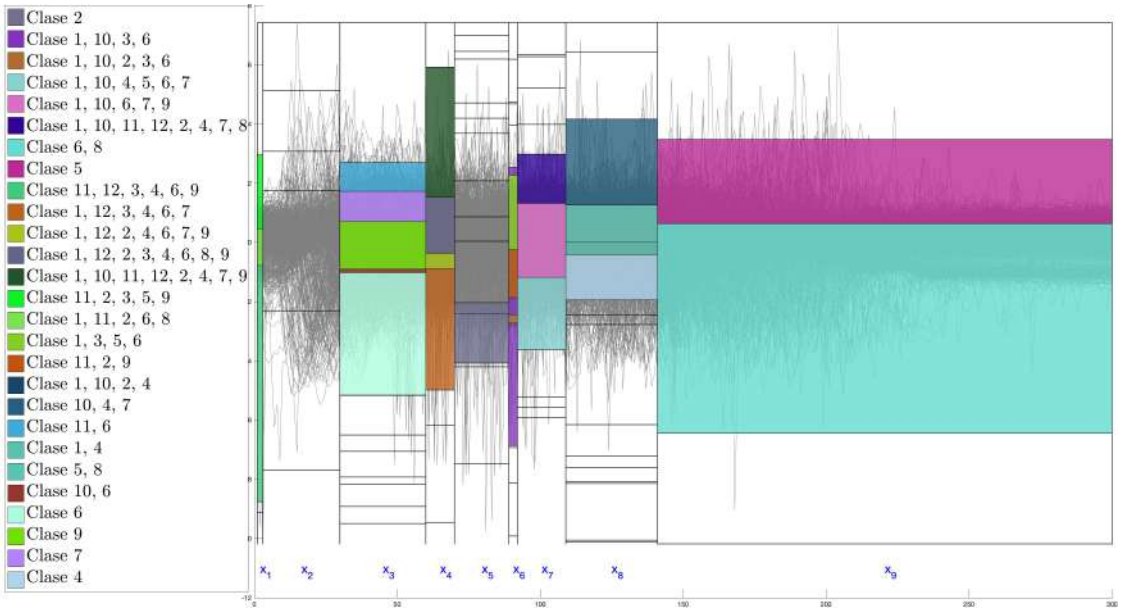


Figura B.13: Distribución de las clases para la base de datos CricketY extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.14. CricketZ

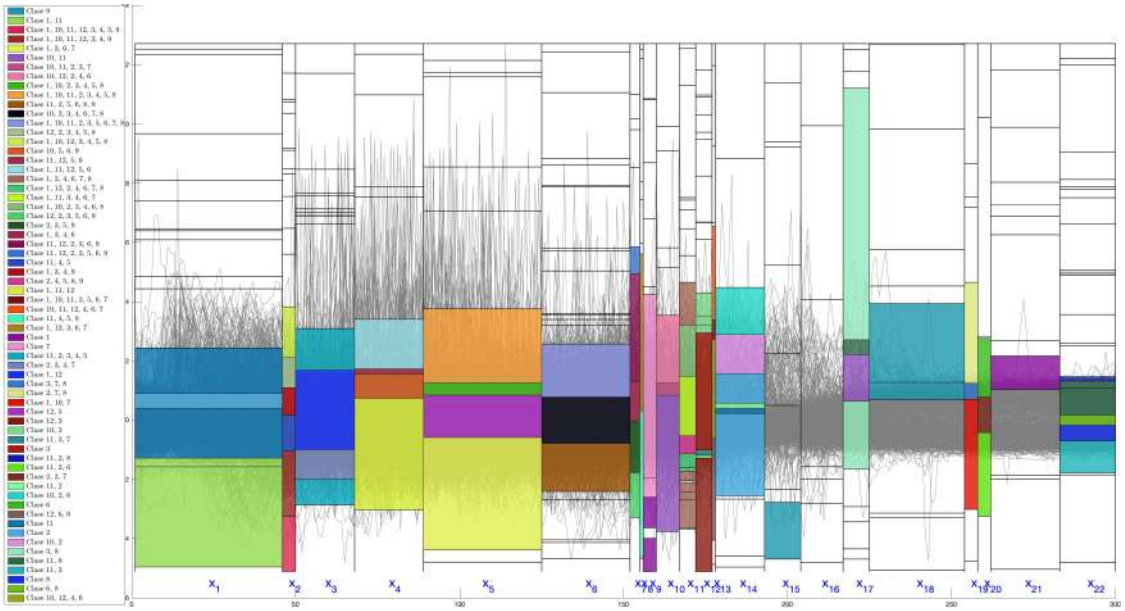


Figura B.14: Distribución de las clases para la base de datos CricketZ extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.15. DiatomSizeReduction

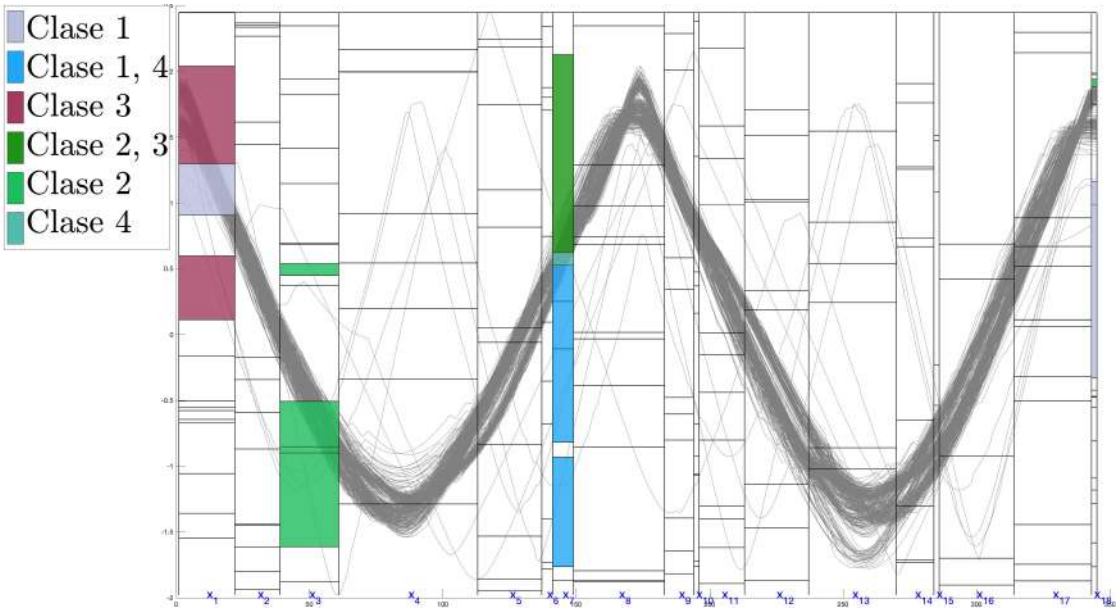


Figura B.15: Distribución de las clases para la base de datos DiatomSizeReduction extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.16. DistalPhalanxOutlineAgeGroup

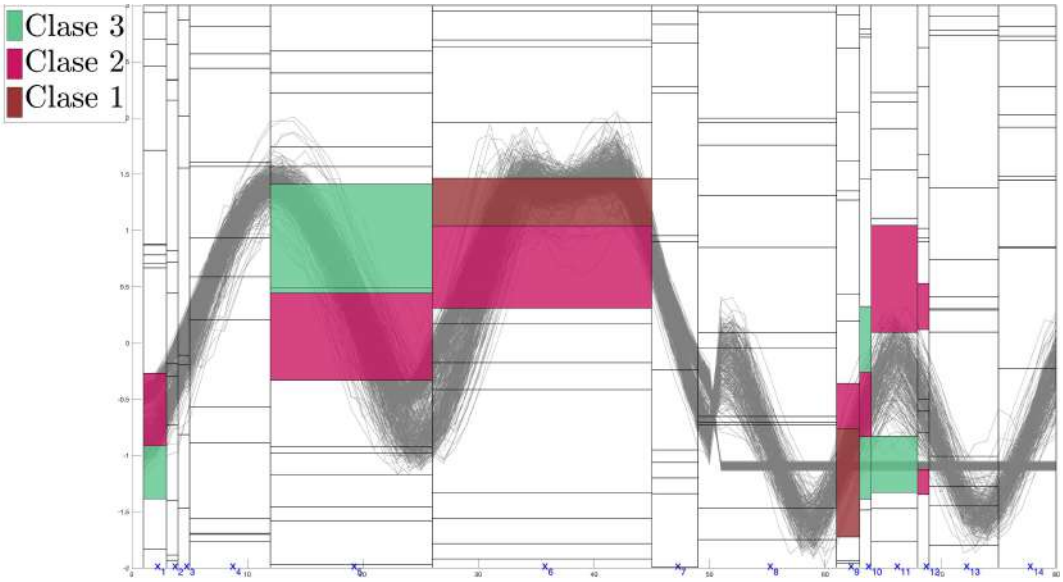


Figura B.16: Distribución de las clases para la base de datos DistalPhalanxOutlineAge-Group extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.17. DistalPhalanxOutlineCorrect

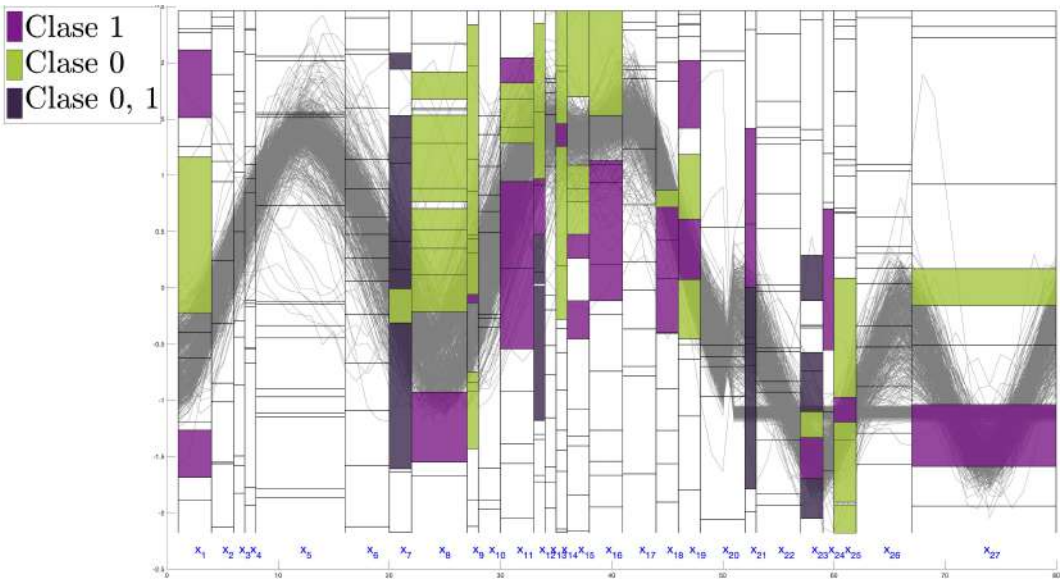


Figura B.17: Distribución de las clases para la base de datos DistalPhalanxOutline-Correct extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.18. DistalPhalanxTW

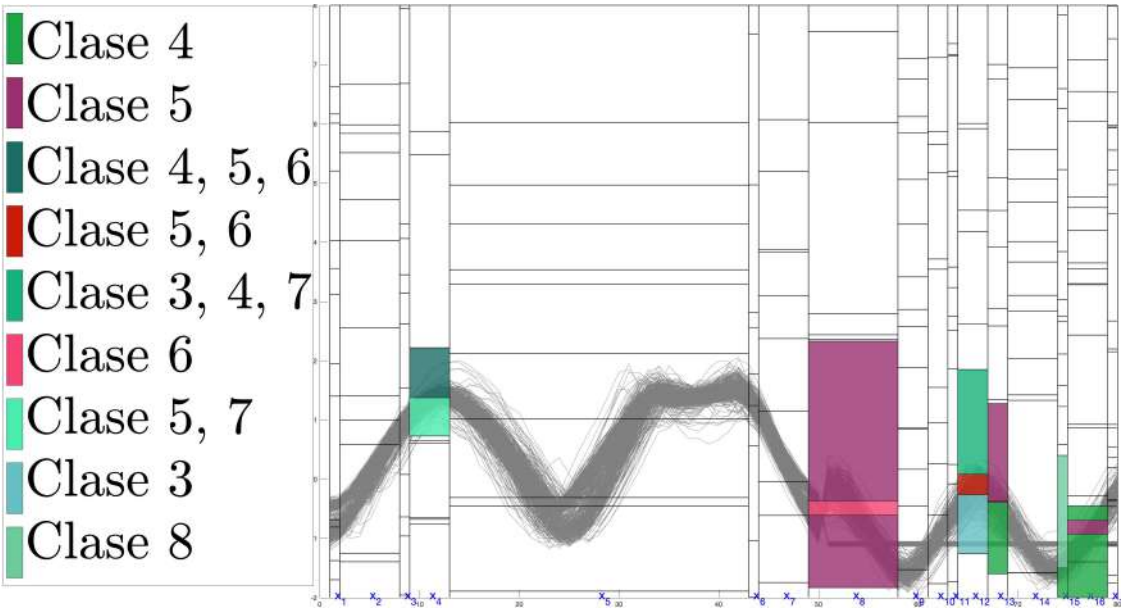


Figura B.18: Distribución de las clases para la base de datos DistalPhalanxTW extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.19. Earthquakes

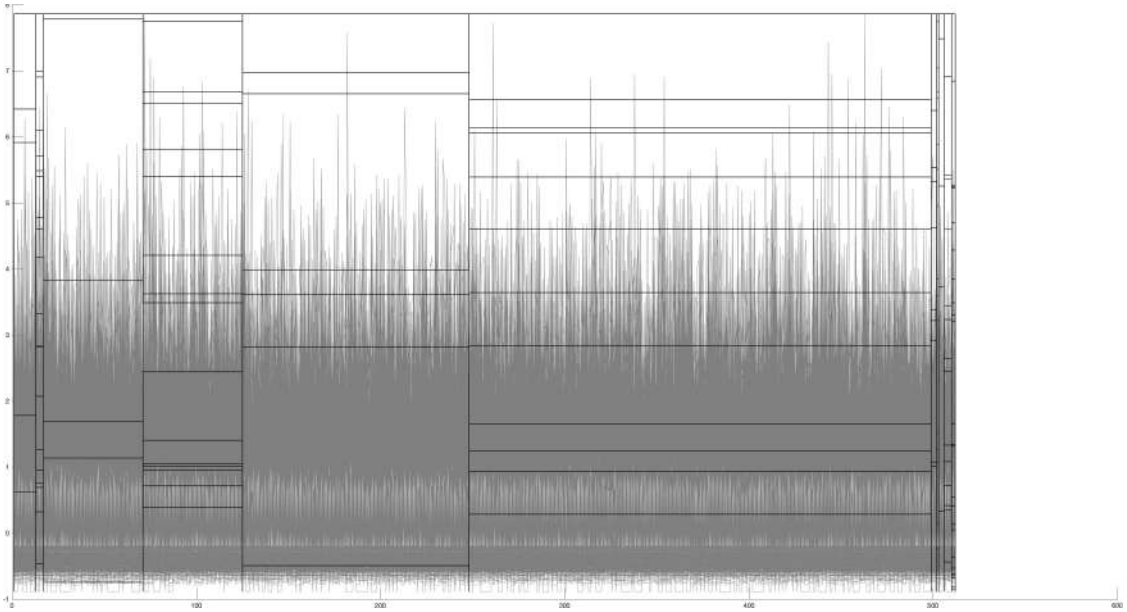


Figura B.19: Distribución de las clases para la base de datos Earthquakes extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.20. ECG200

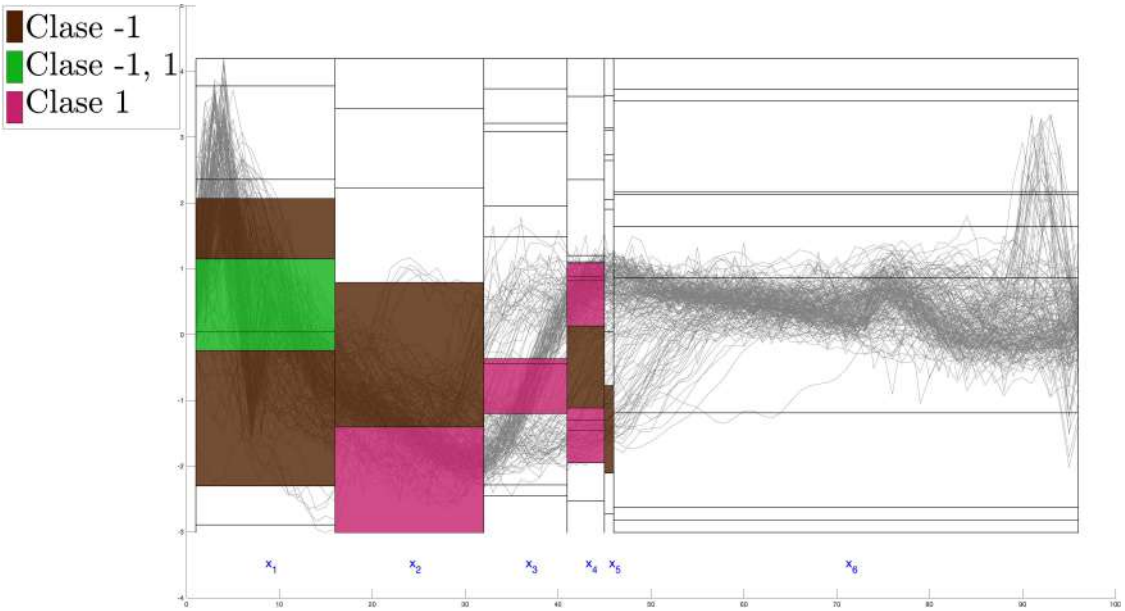


Figura B.20: Distribución de las clases para la base de datos ECG200 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.21. ECG5000

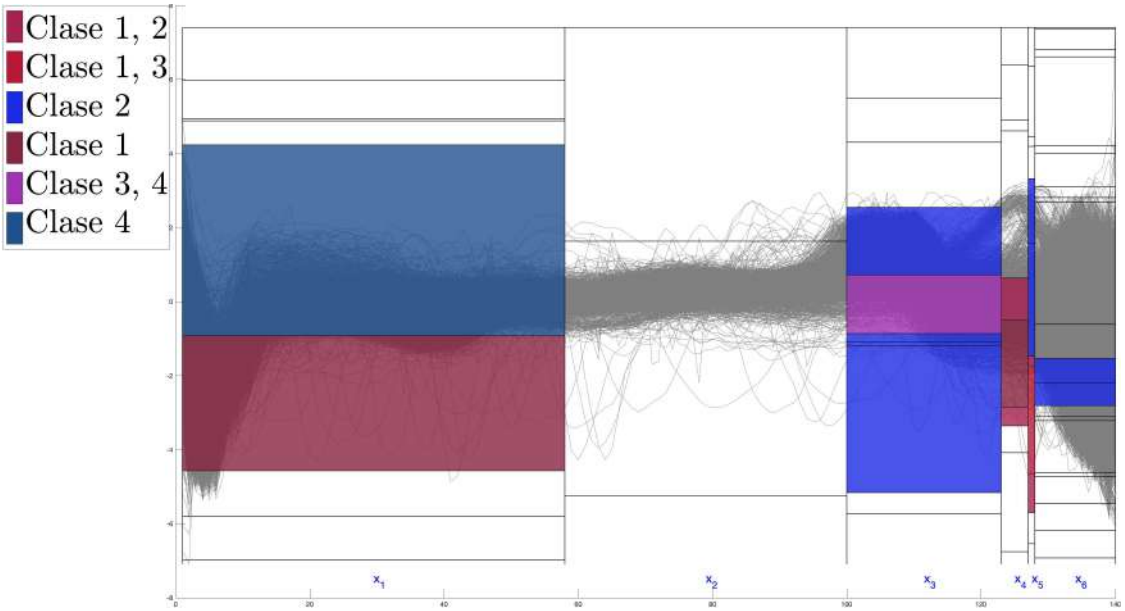


Figura B.21: Distribución de las clases para la base de datos ECG5000 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.22. ECGFiveDays

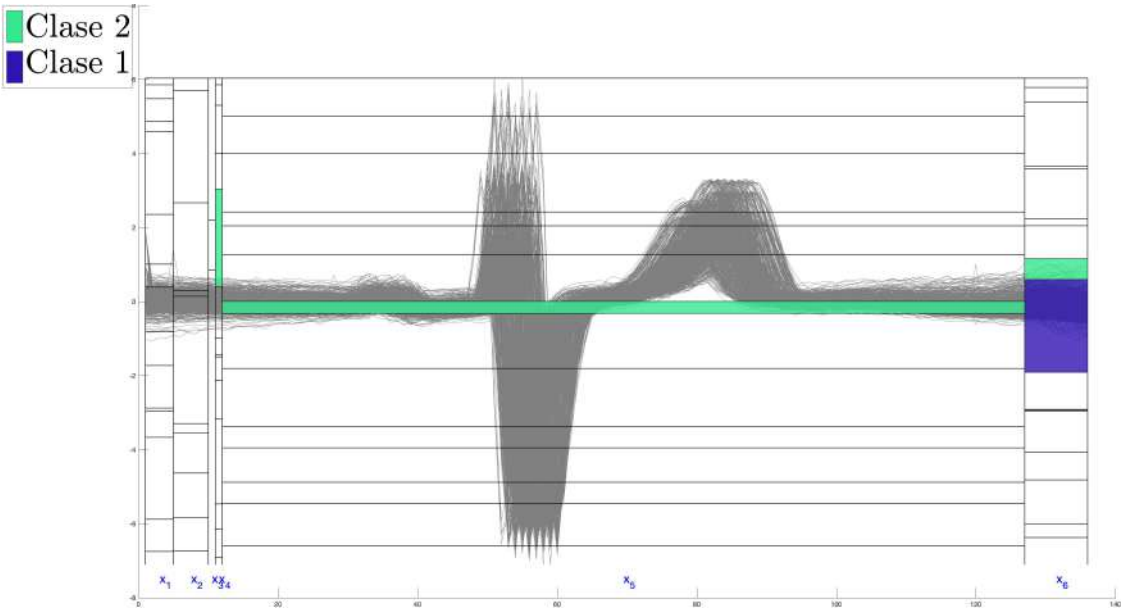


Figura B.22: Distribución de las clases para la base de datos ECGFiveDays extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.23. ElectricDevices

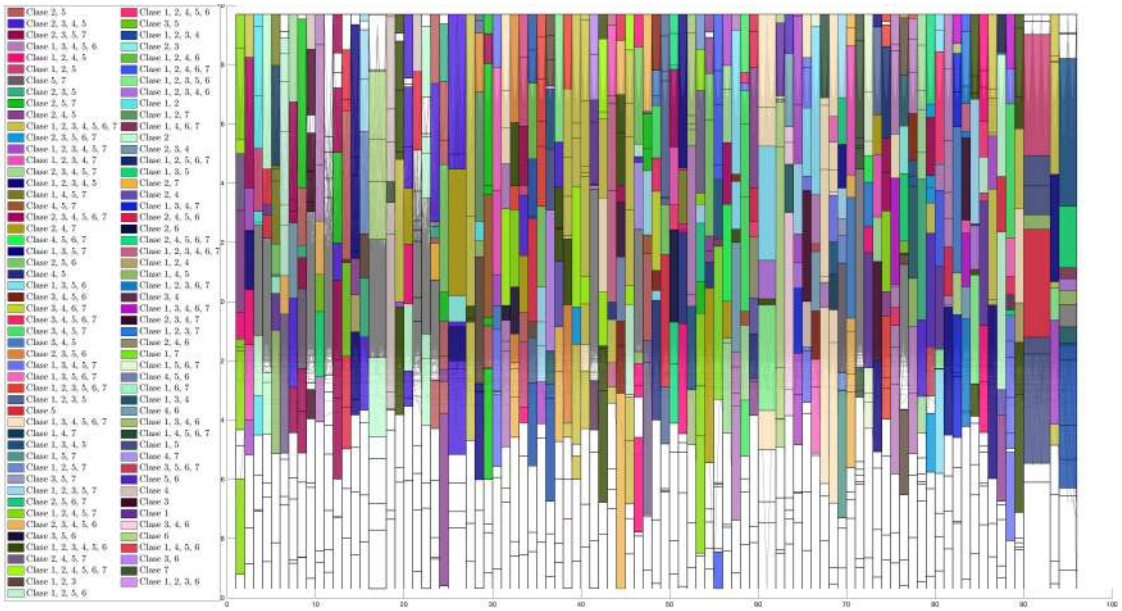


Figura B.23: Distribución de las clases para la base de datos ElectricDevices extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.24. FaceAll

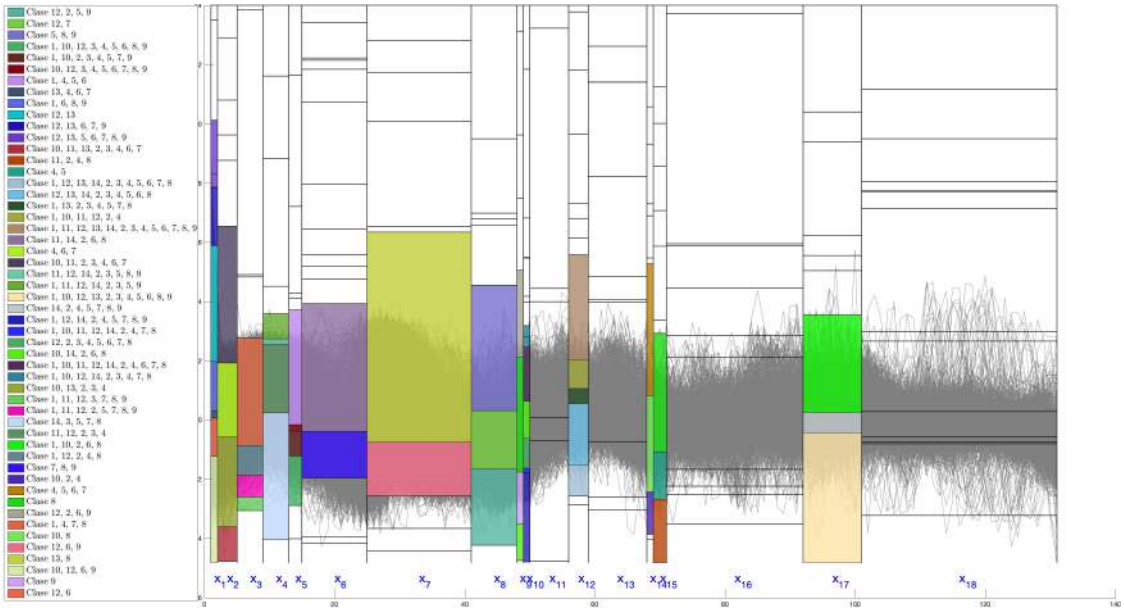


Figura B.24: Distribución de las clases para la base de datos FaceAll extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.25. FaceFour

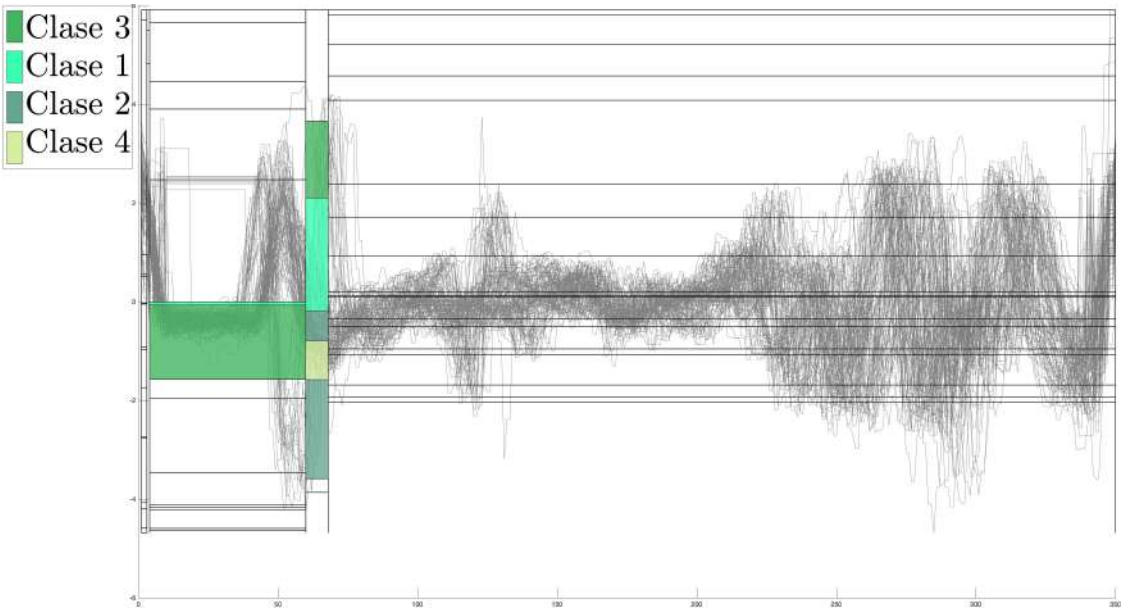


Figura B.25: Distribución de las clases para la base de datos FaceFour extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.26. FacesUCR

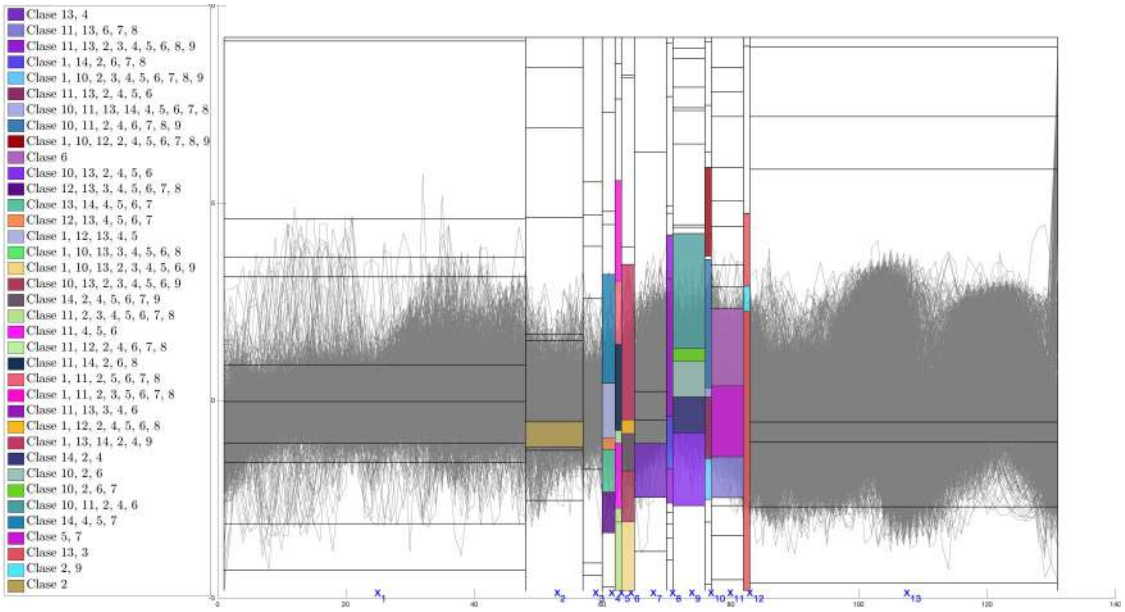


Figura B.26: Distribución de las clases para la base de datos FacesUCR extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.27. FiftyWords

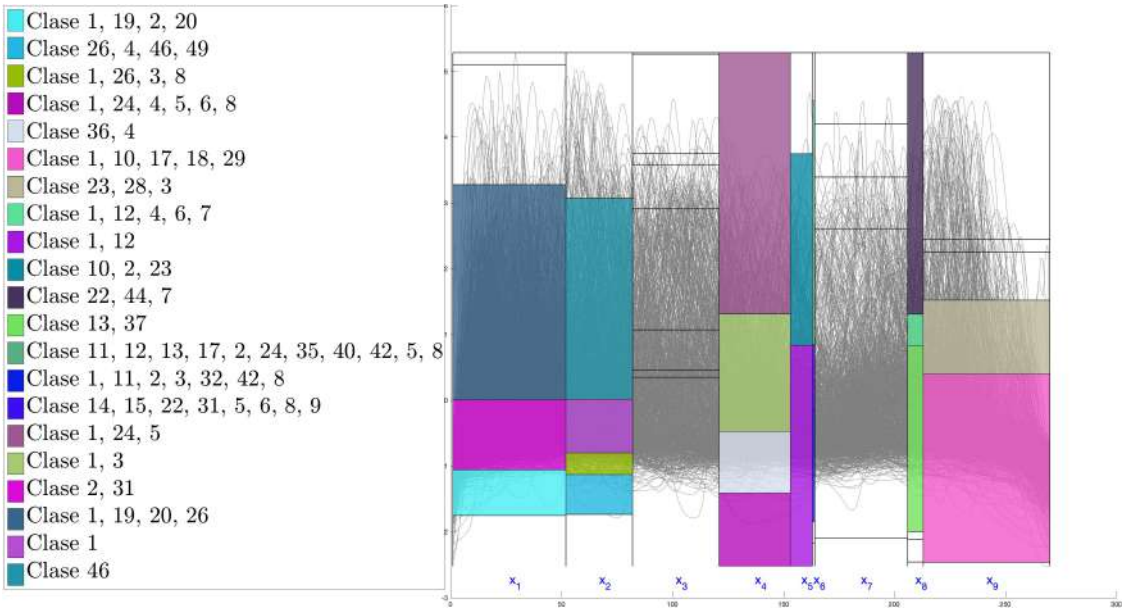


Figura B.27: Distribución de las clases para la base de datos FiftyWords extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.28. Fish

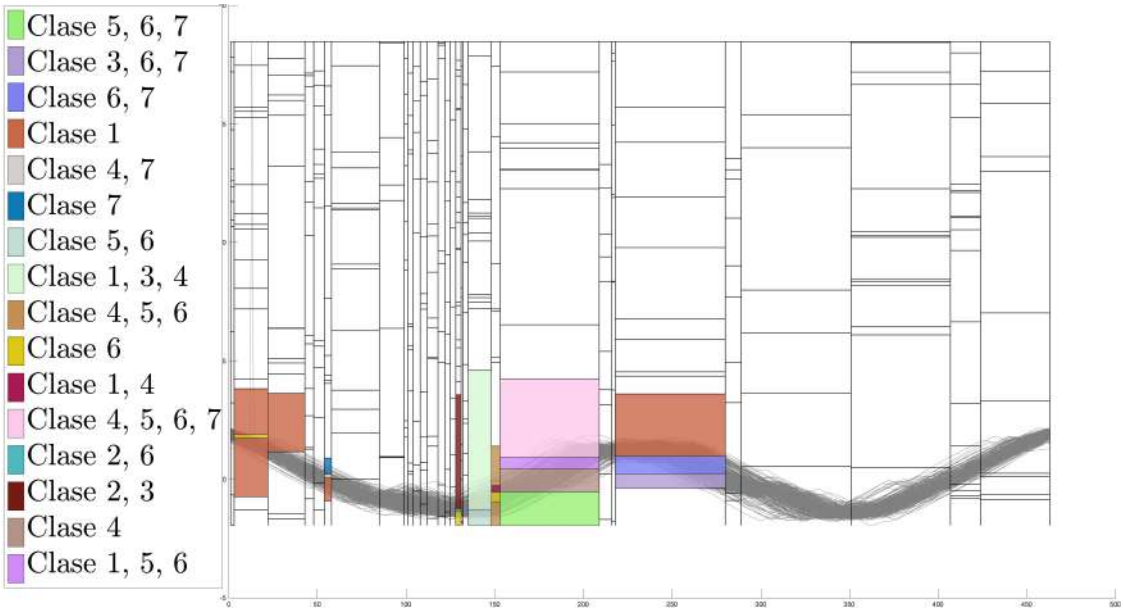


Figura B.28: Distribución de las clases para la base de datos Fish extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.29. FordA

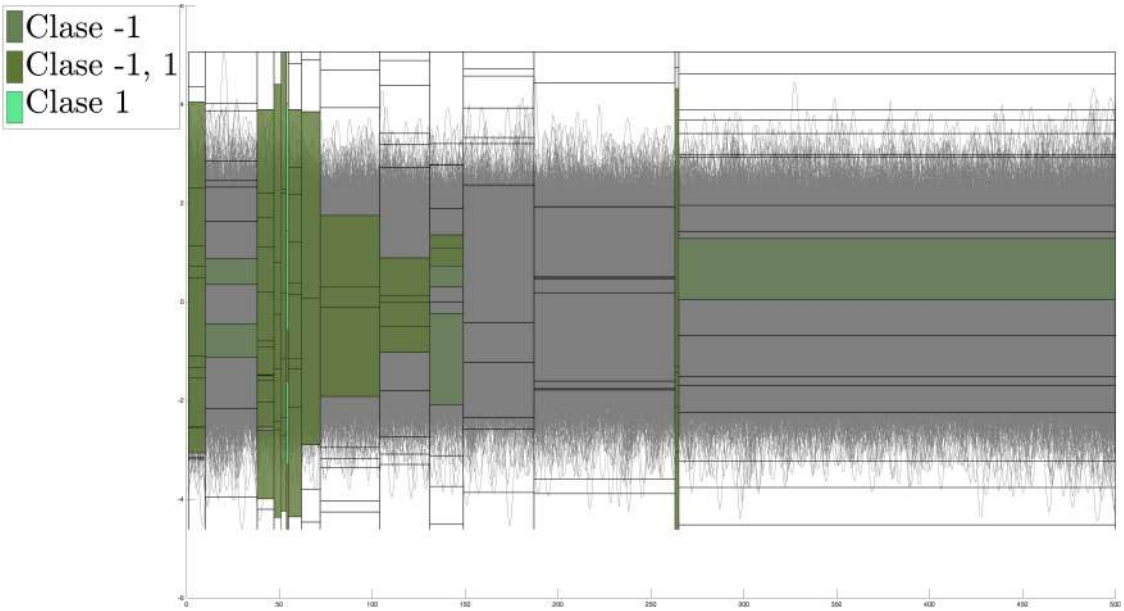


Figura B.29: Distribución de las clases para la base de datos FordA extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.30. FordB

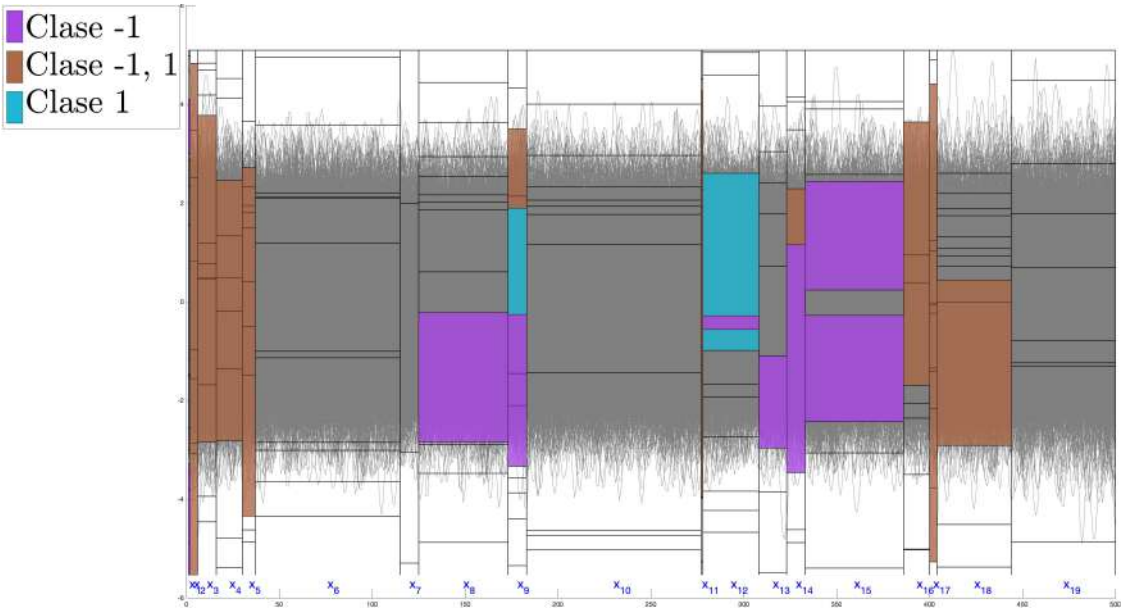


Figura B.30: Distribución de las clases para la base de datos FordB extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.31. GunPoint

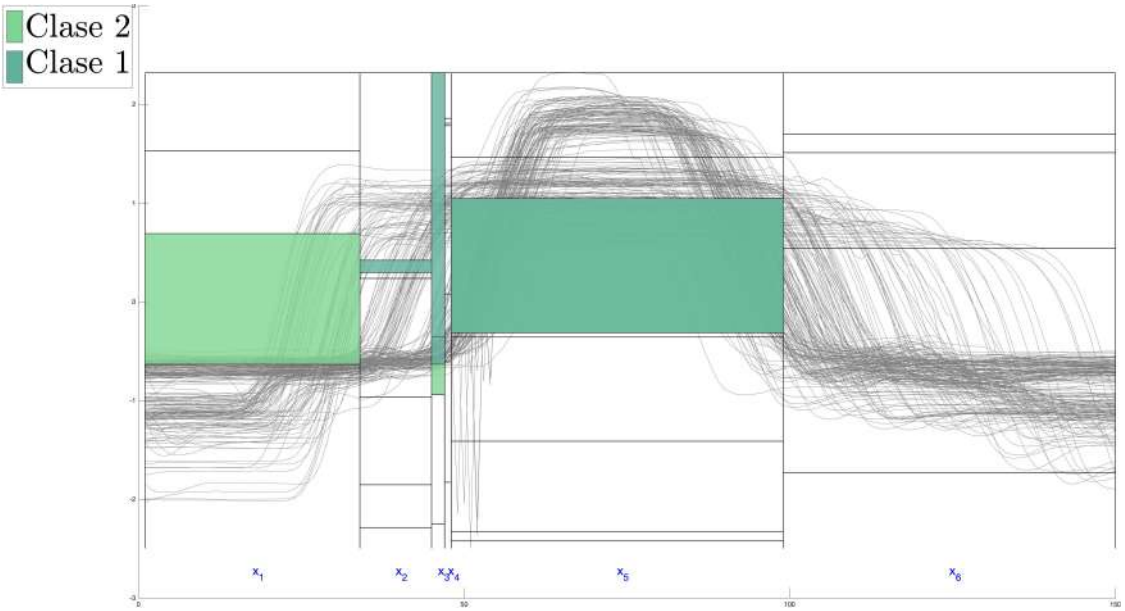


Figura B.31: Distribución de las clases para la base de datos GunPoint extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.32. Ham

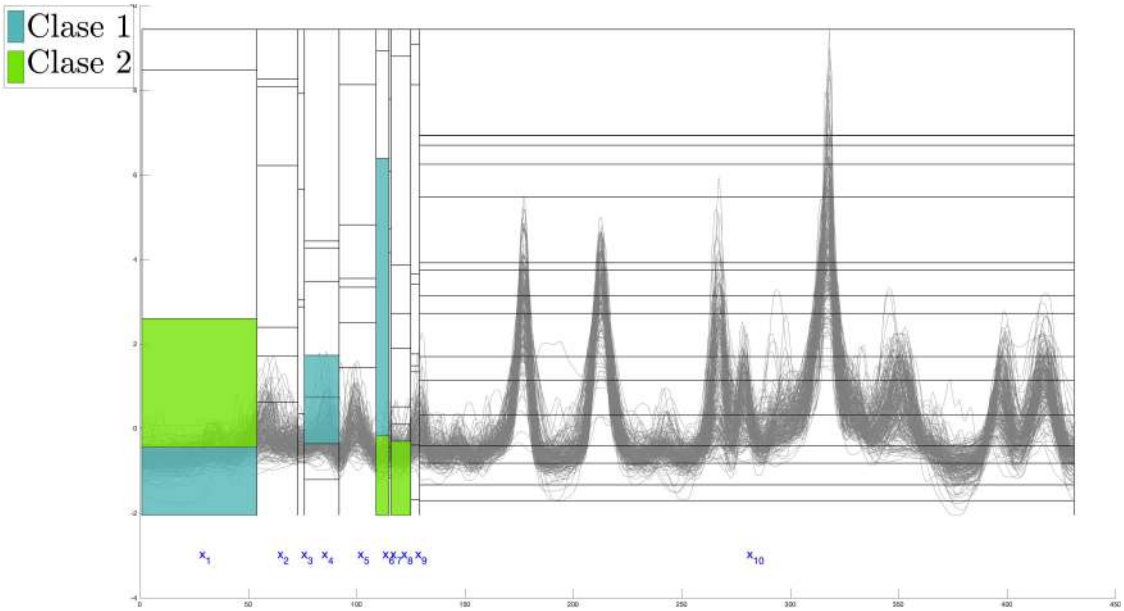


Figura B.32: Distribución de las clases para la base de datos Ham extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.33. HandOutlines

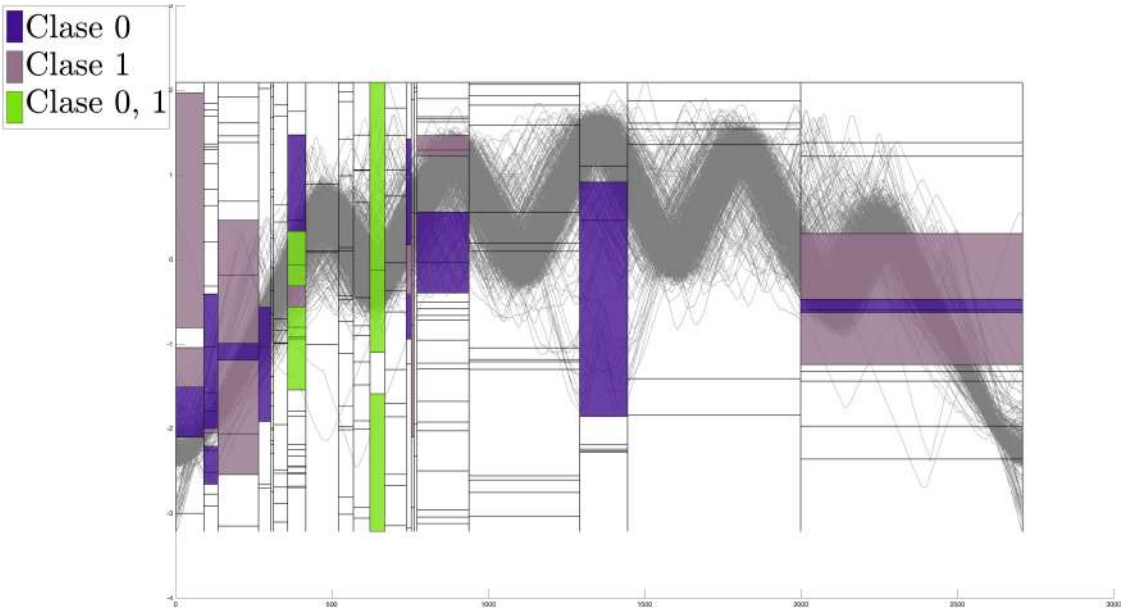


Figura B.33: Distribución de las clases para la base de datos HandOutlines extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.34. Haptics

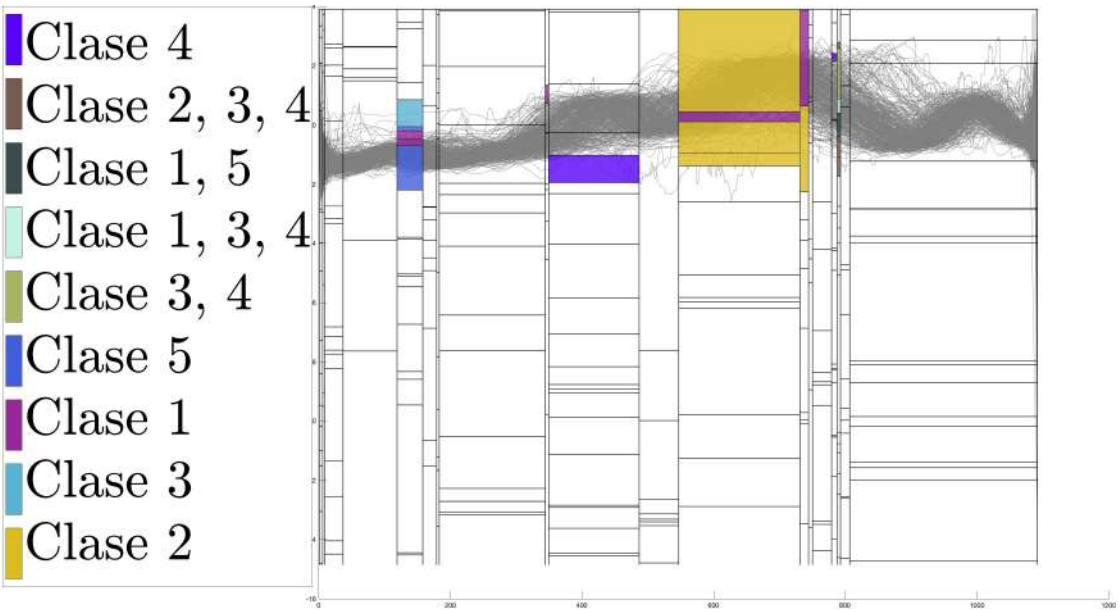


Figura B.34: Distribución de las clases para la base de datos Haptics extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.35. Herring

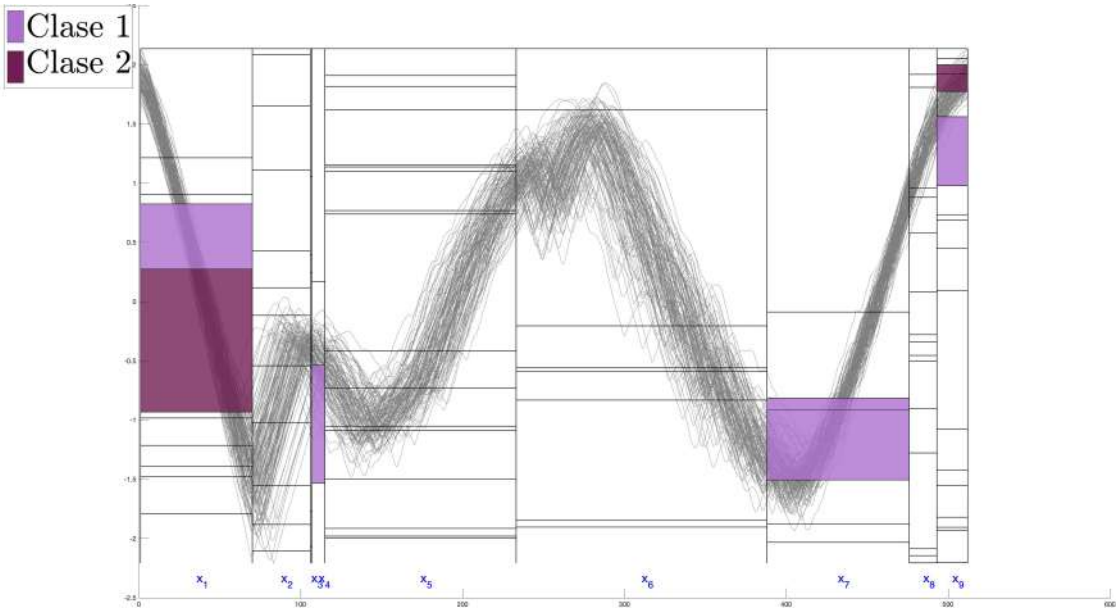


Figura B.35: Distribución de las clases para la base de datos Herring extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.36. InlineSkate

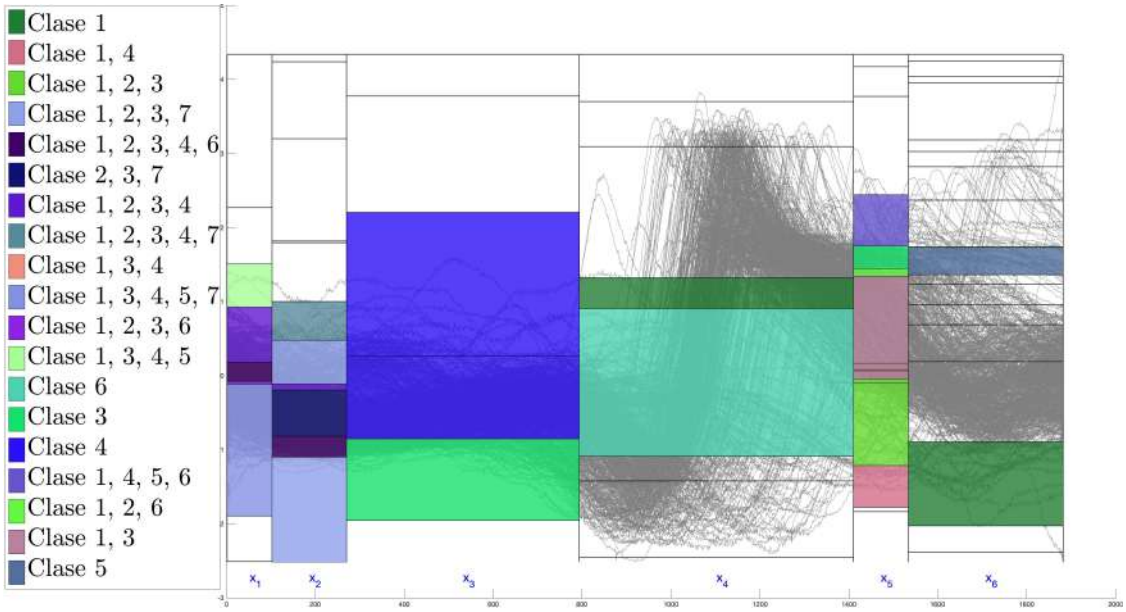


Figura B.36: Distribución de las clases para la base de datos InlineSkate extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.37. InsectWingbeatSound

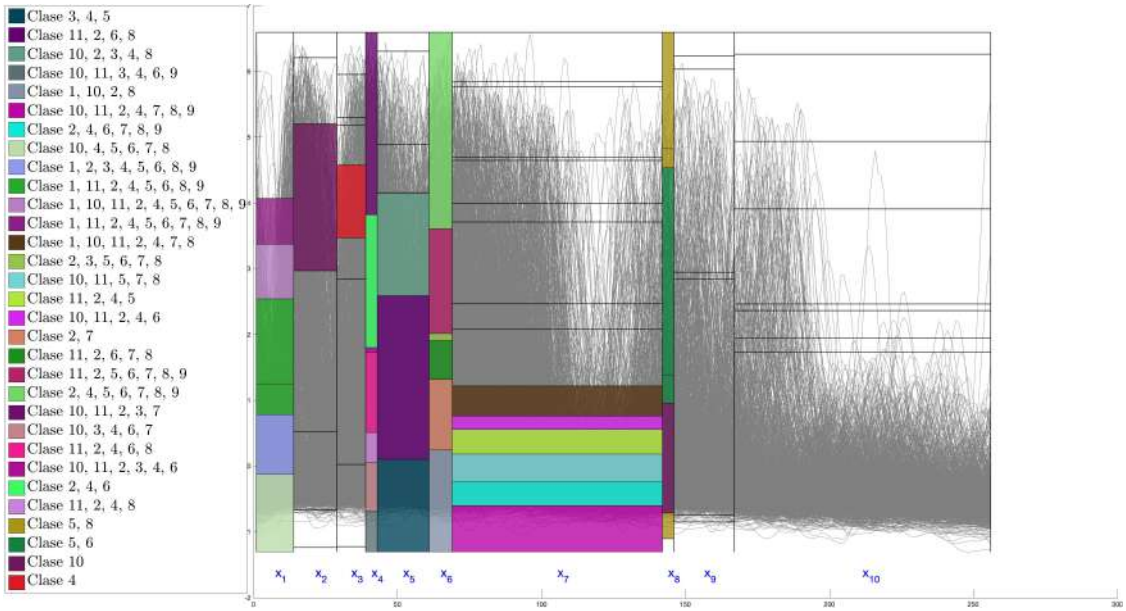


Figura B.37: Distribución de las clases para la base de datos InsectWingbeatSound extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.38. ItalyPowerDemand

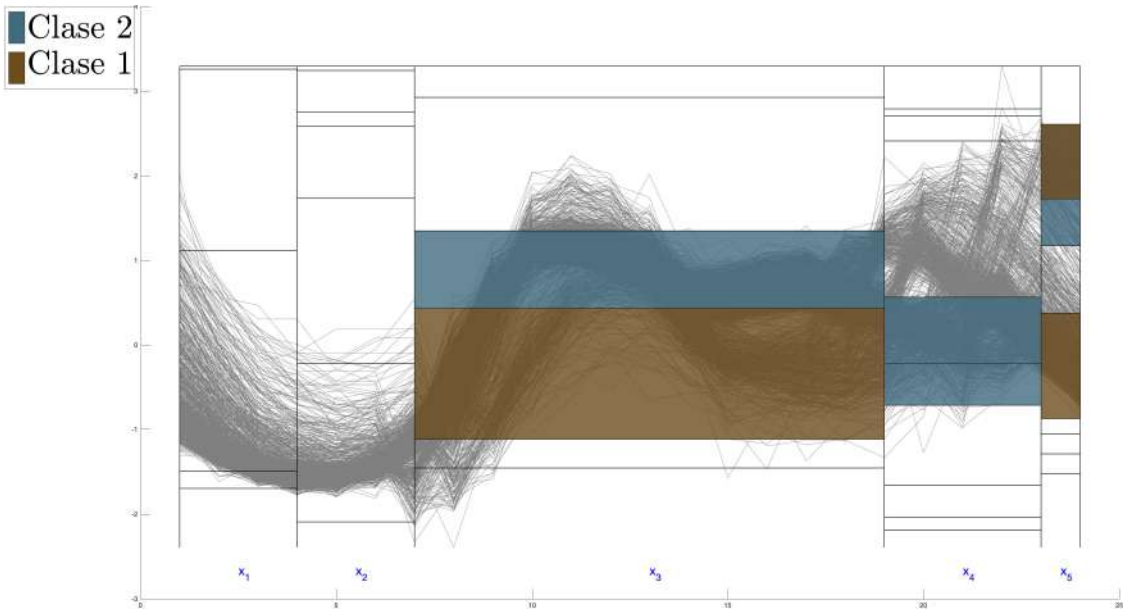


Figura B.38: Distribución de las clases para la base de datos ItalyPowerDemand extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.39. LargeKitchenAppliances

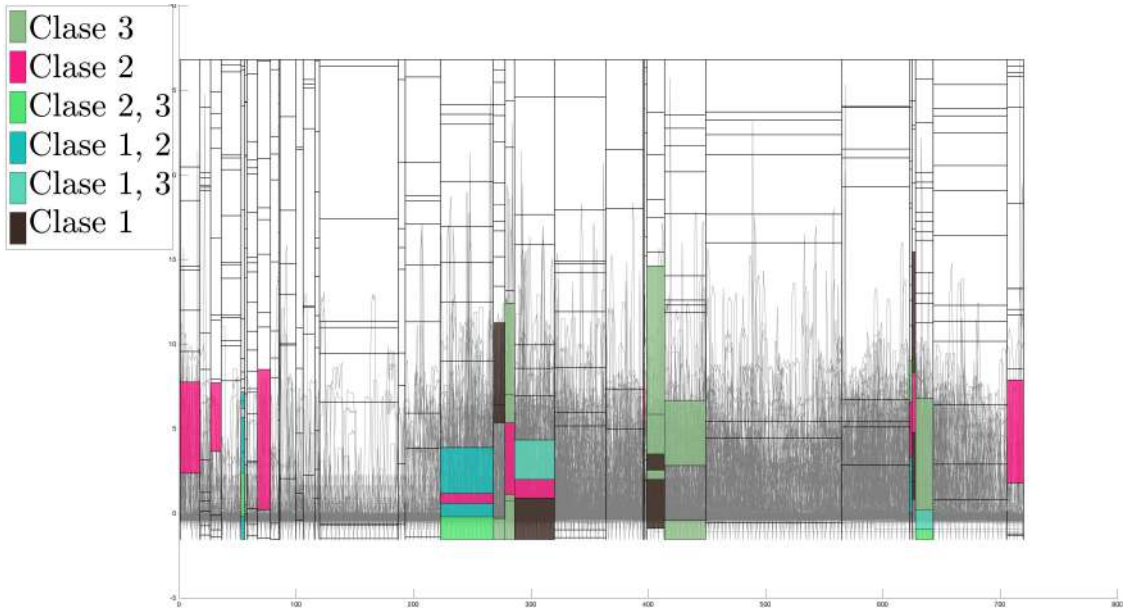


Figura B.39: Distribución de las clases para la base de datos LargeKitchenAppliances extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.40. Lighting2

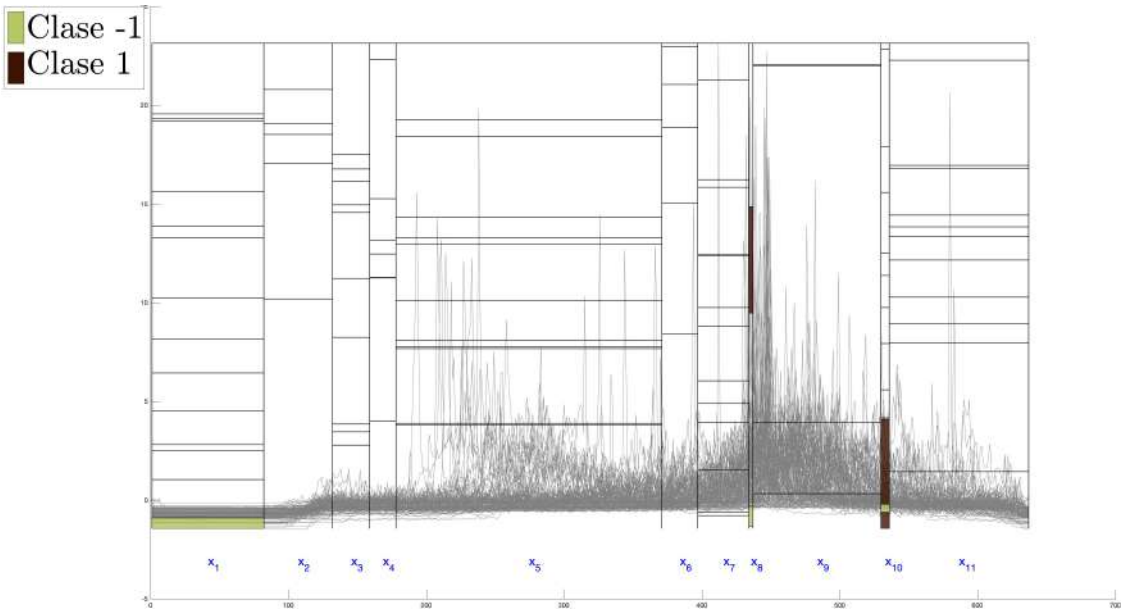


Figura B.40: Distribución de las clases para la base de datos Lighting2 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.41. Lighting7

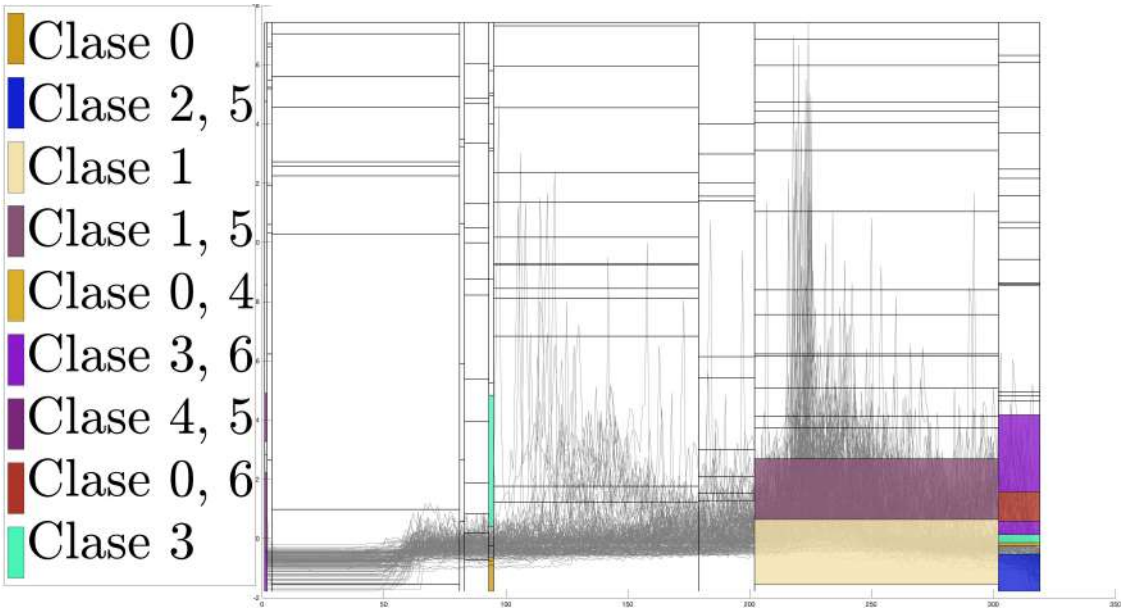


Figura B.41: Distribución de las clases para la base de datos Lighting7 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.42. Mallat

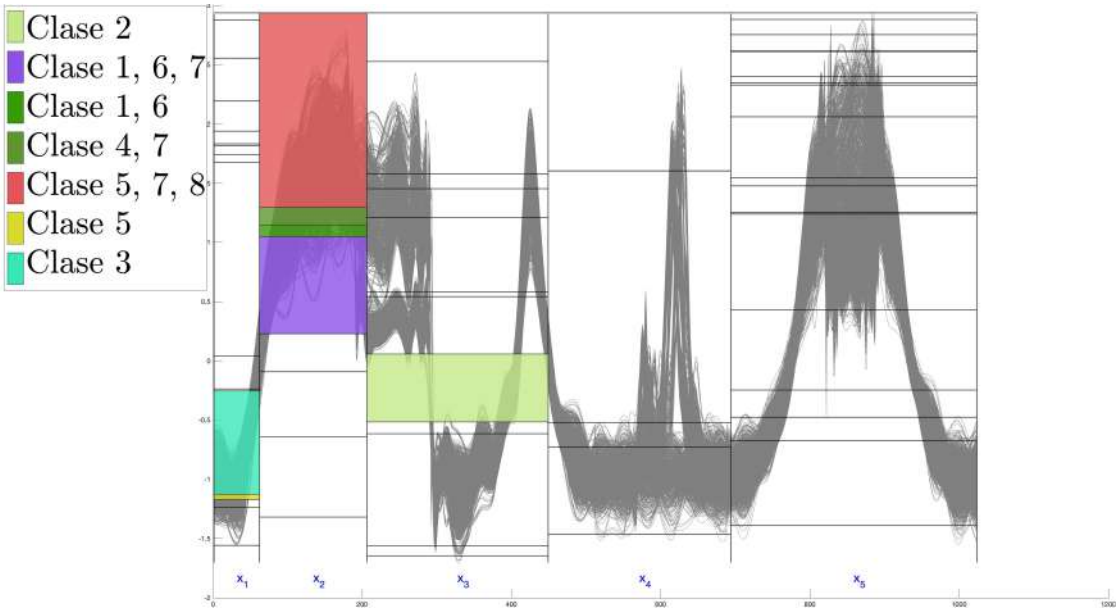


Figura B.42: Distribución de las clases para la base de datos Mallat extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.43. Meat

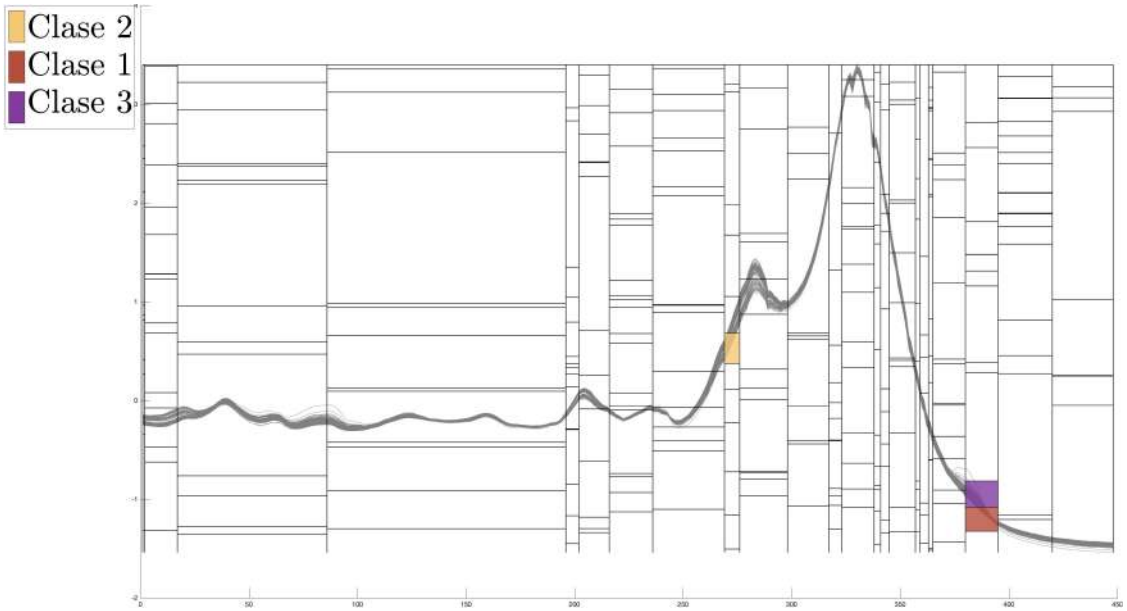


Figura B.43: Distribución de las clases para la base de datos Meat extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.44. MedicalImages

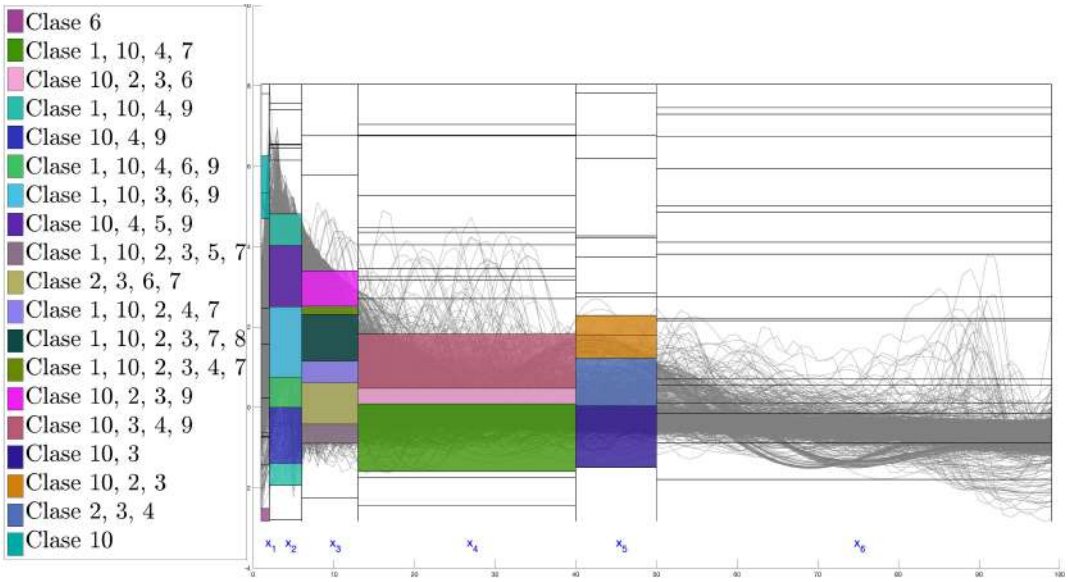


Figura B.44: Distribución de las clases para la base de datos MedicalImages extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.45. MiddlePhalanxOutlineAgeGroup

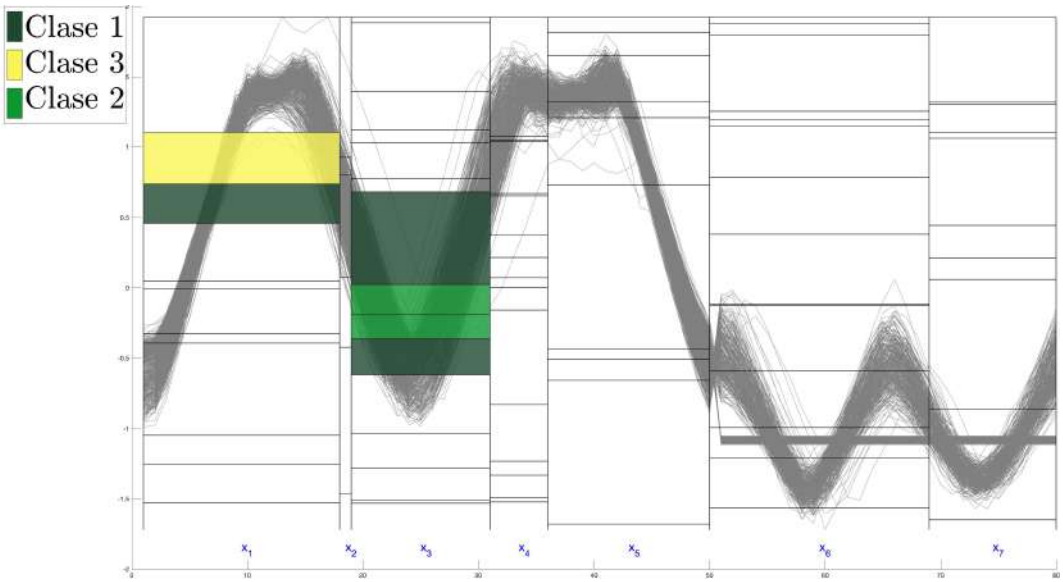


Figura B.45: Distribución de las clases para la base de datos MiddlePhalanxOutlineAgeGroup extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.46. MiddlePhalanxOutlineCorrect

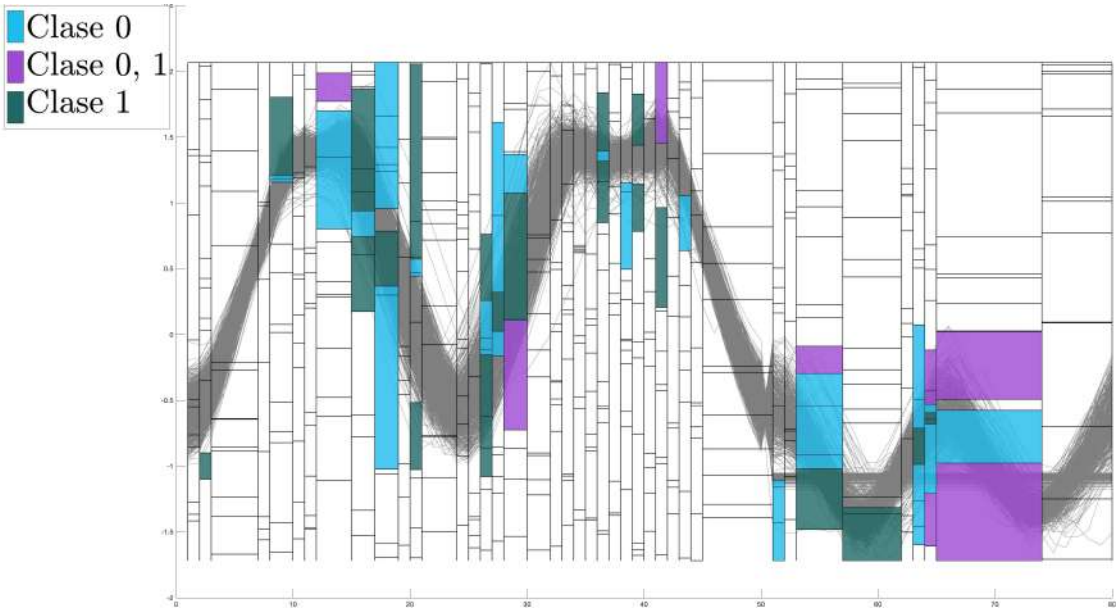


Figura B.46: Distribución de las clases para la base de datos MiddlePhalanxOutline-Correct extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.47. MiddlePhalanxTW

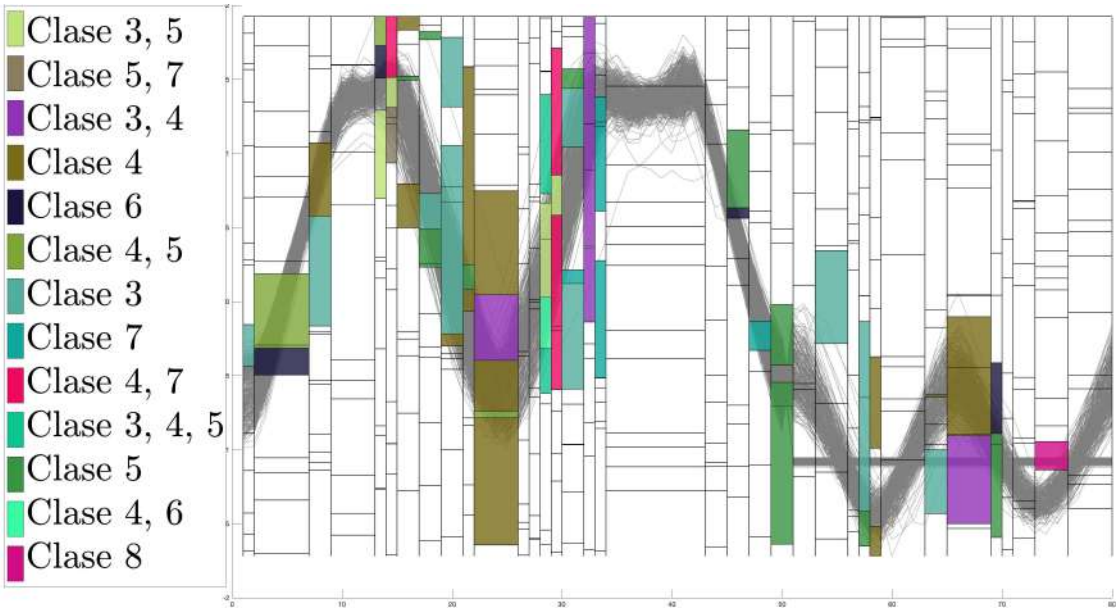


Figura B.47: Distribución de las clases para la base de datos MiddlePhalanxTW extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.48. MoteStrain

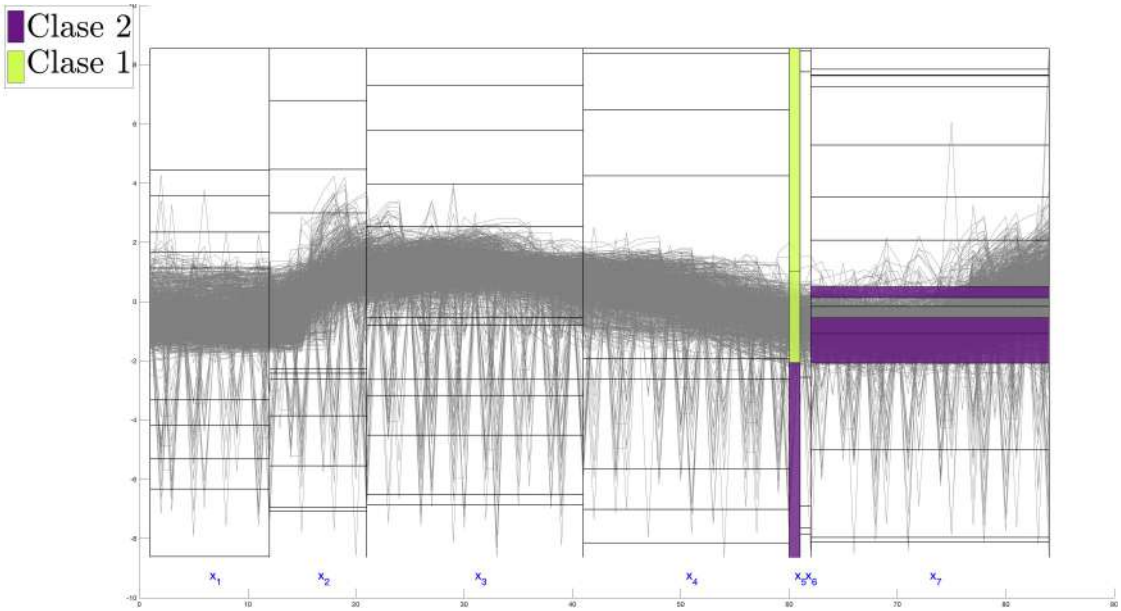


Figura B.48: Distribución de las clases para la base de datos MoteStrain extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.49. NonInvasiveFetalECGThorax1

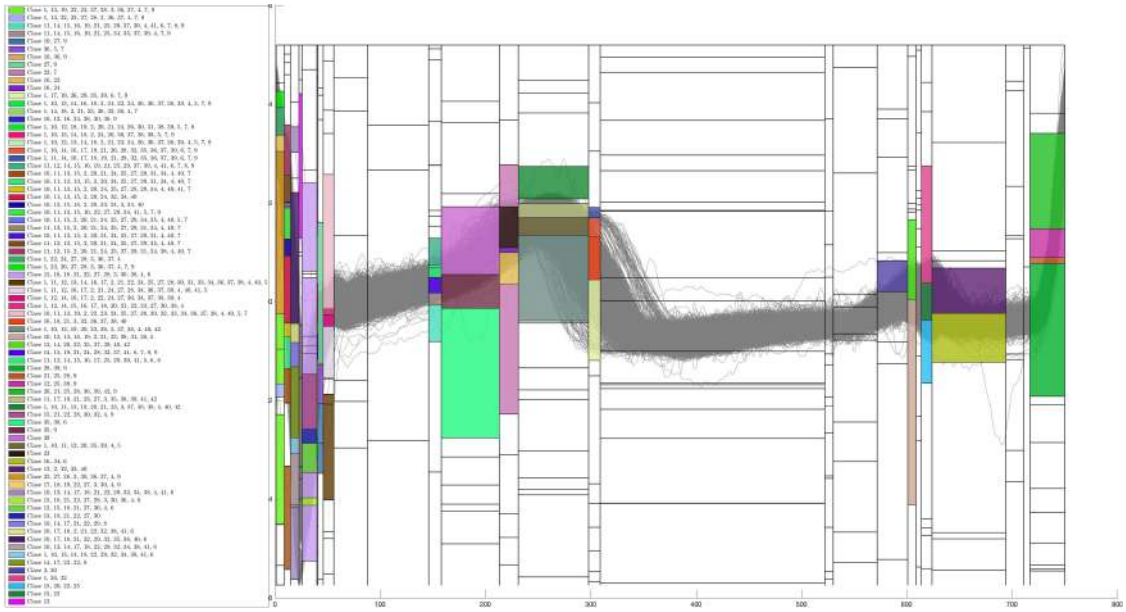


Figura B.49: Distribución de las clases para la base de datos NonInvasiveFetalECGThorax1 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.50. NonInvasiveFetalECGThorax2

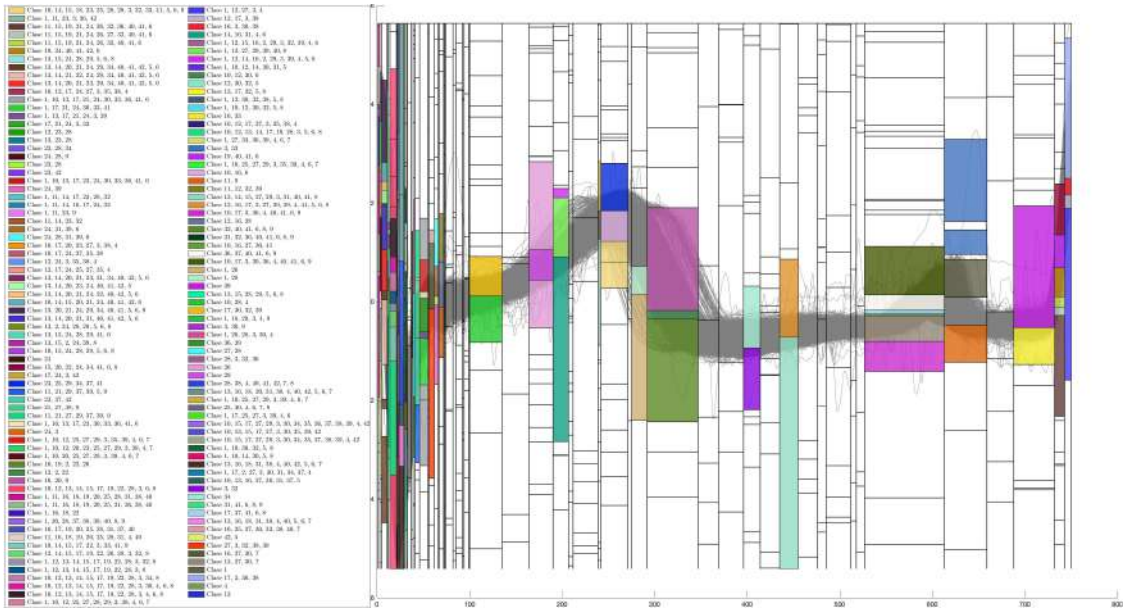


Figura B.50: Distribución de las clases para la base de datos NonInvasiveFetalECGThorax2 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.51. OliveOil

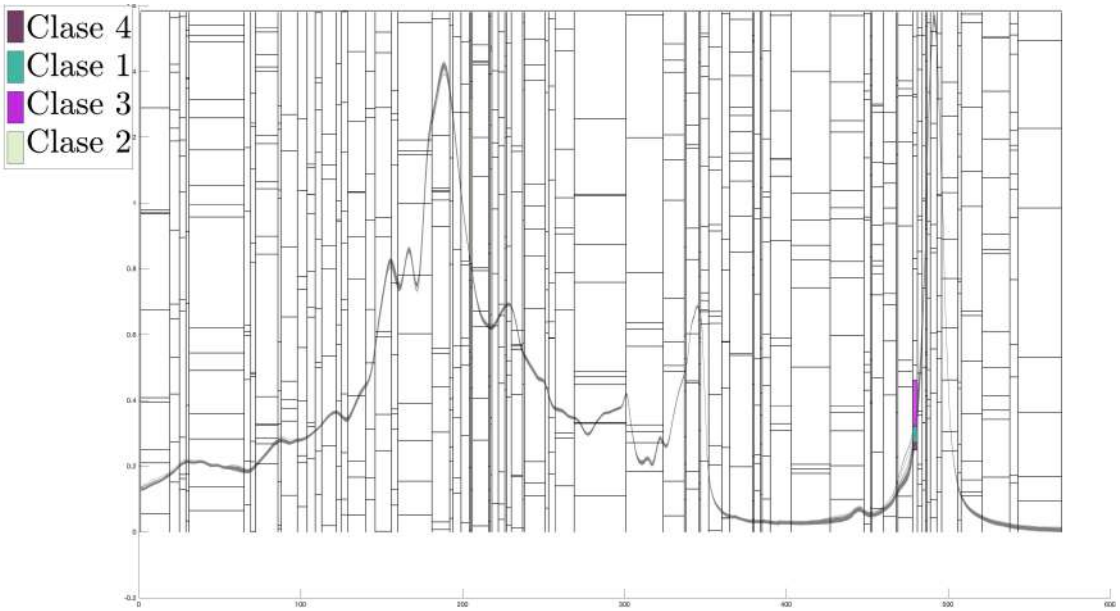


Figura B.51: Distribución de las clases para la base de datos OliveOil extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.52. OSULeaf

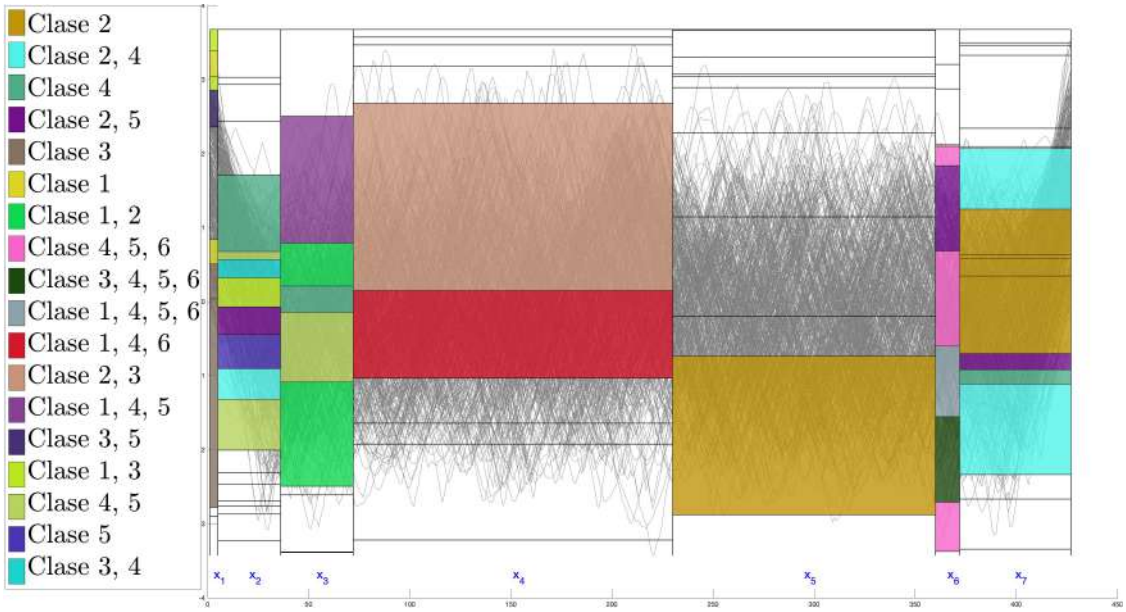


Figura B.52: Distribución de las clases para la base de datos OSULeaf extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.53. PhalangesOutlinesCorrect

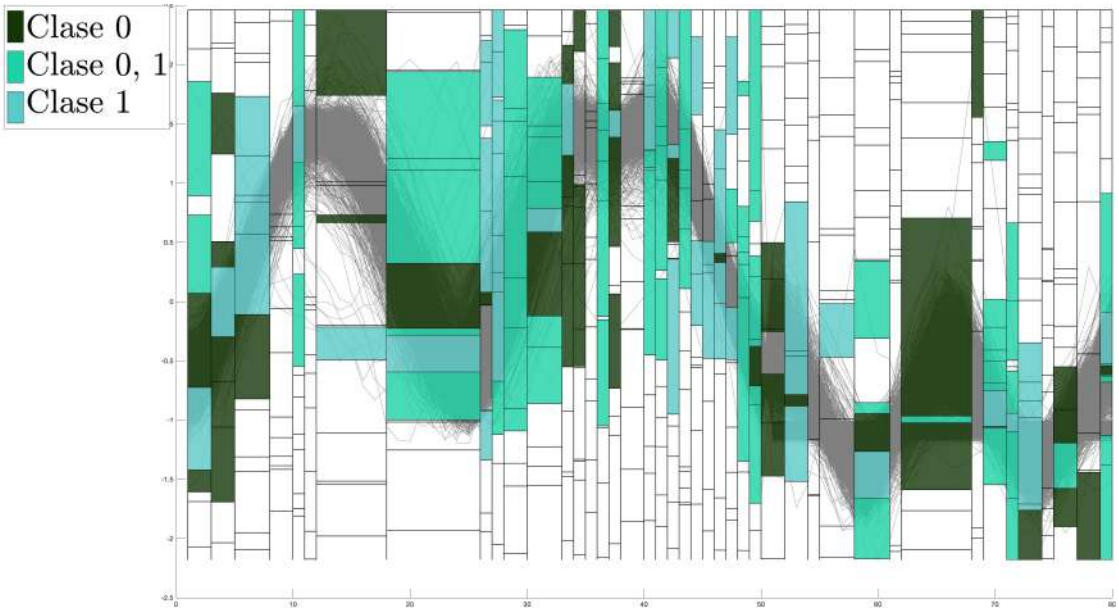


Figura B.53: Distribución de las clases para la base de datos PhalangesOutlinesCorrect extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.54. Phoneme

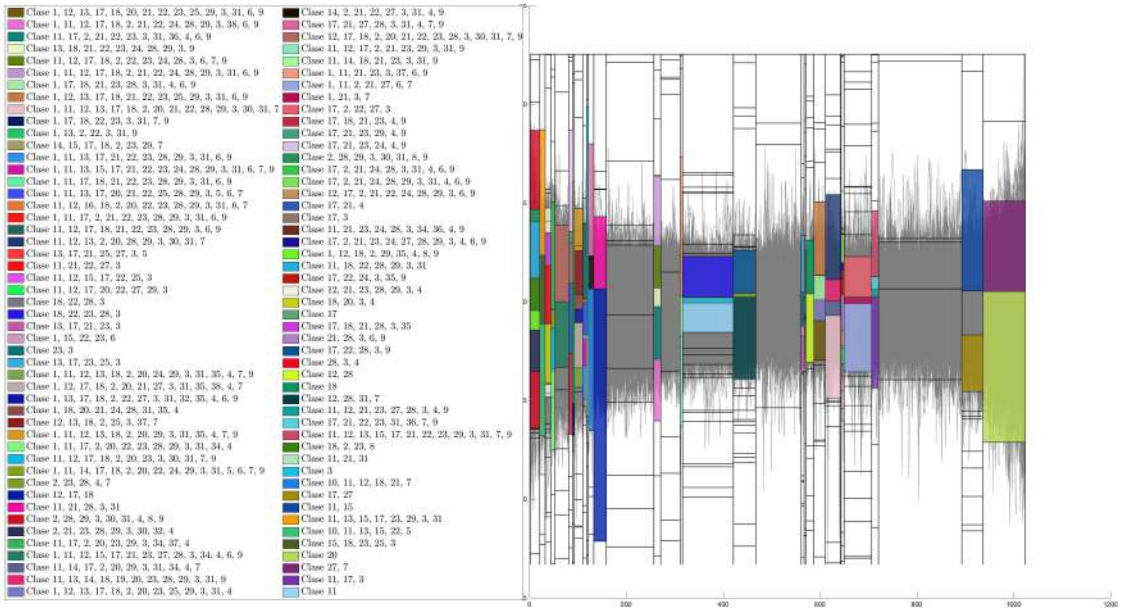


Figura B.54: Distribución de las clases para la base de datos Phoneme extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.55. Plane

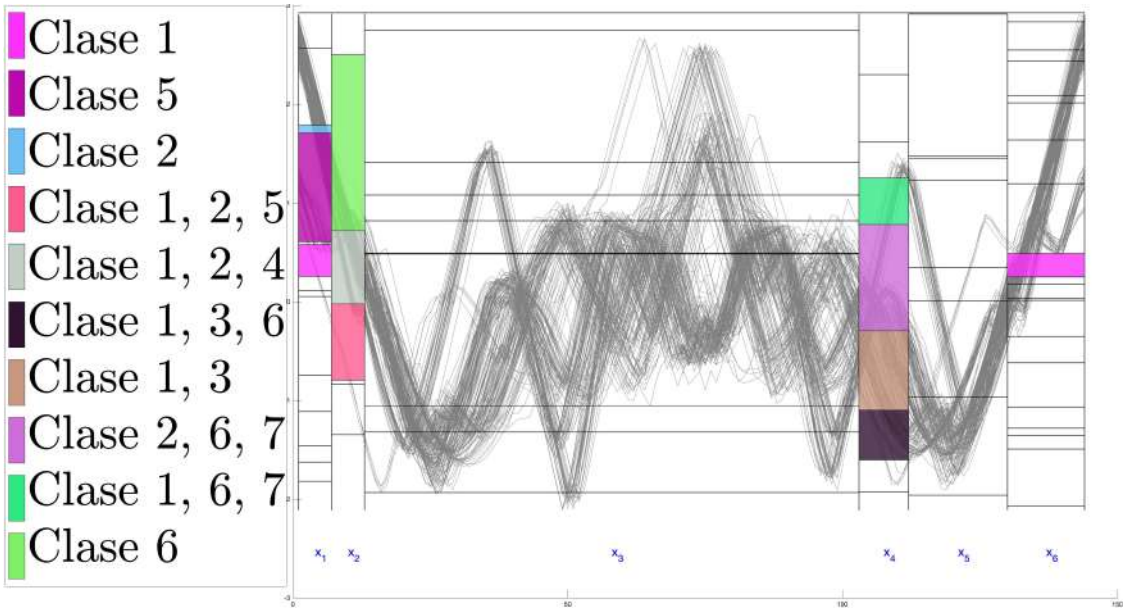


Figura B.55: Distribución de las clases para la base de datos Plane extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.56. ProximalPhalanxOutlineAgeGroup

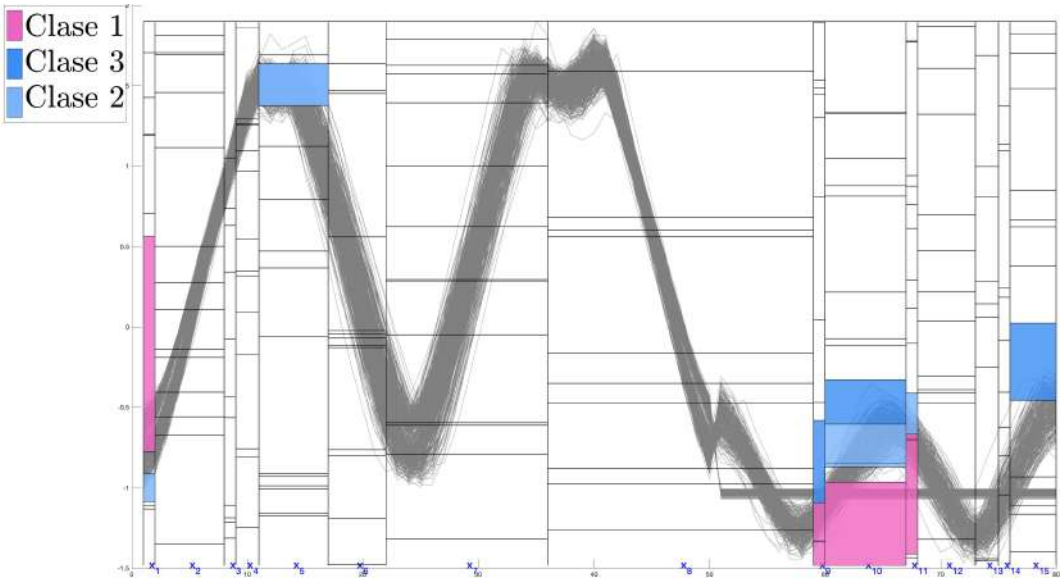


Figura B.56: Distribución de las clases para la base de datos ProximalPhalanxOutlineAgeGroup extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.57. ProximalPhalanxOutlineCorrect

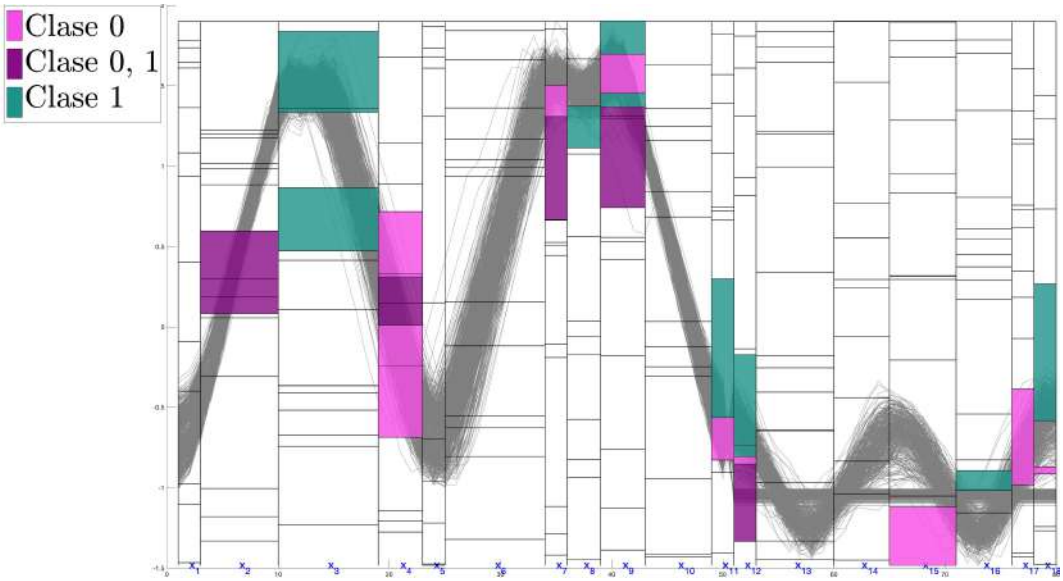


Figura B.57: Distribución de las clases para la base de datos ProximalPhalanxOutlineCorrect extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.58. ProximalPhalanxTW

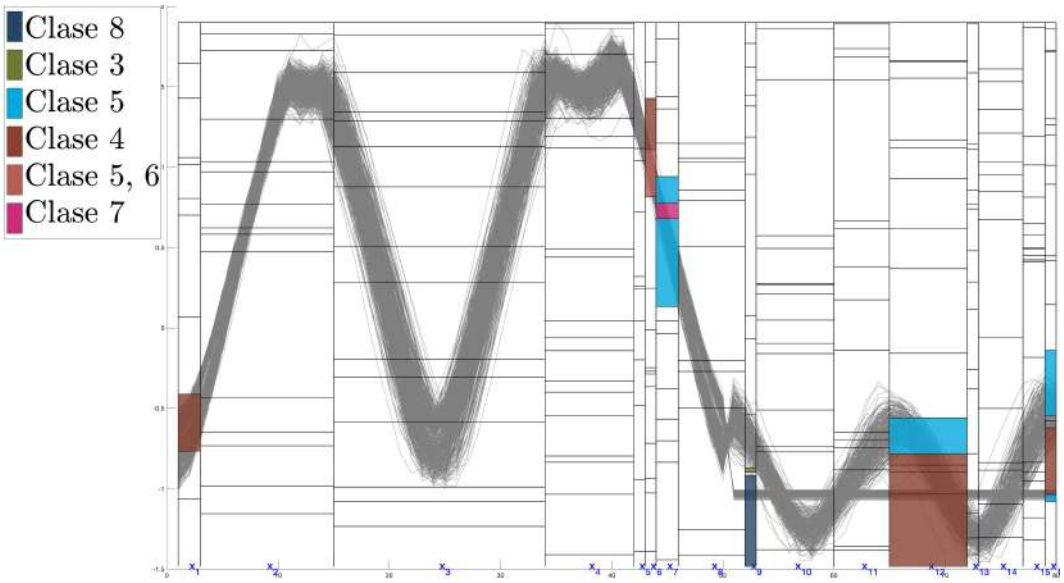


Figura B.58: Distribución de las clases para la base de datos ProximalPhalanxTW extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.59. RefrigerationDevices

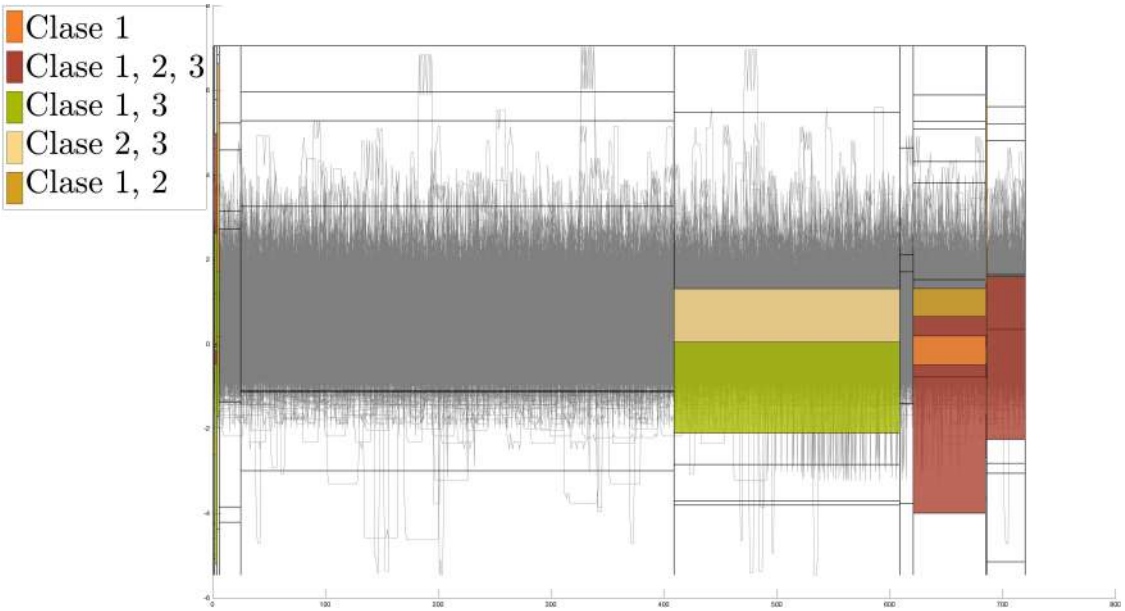


Figura B.59: Distribución de las clases para la base de datos RefrigerationDevices extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.60. ScreenType

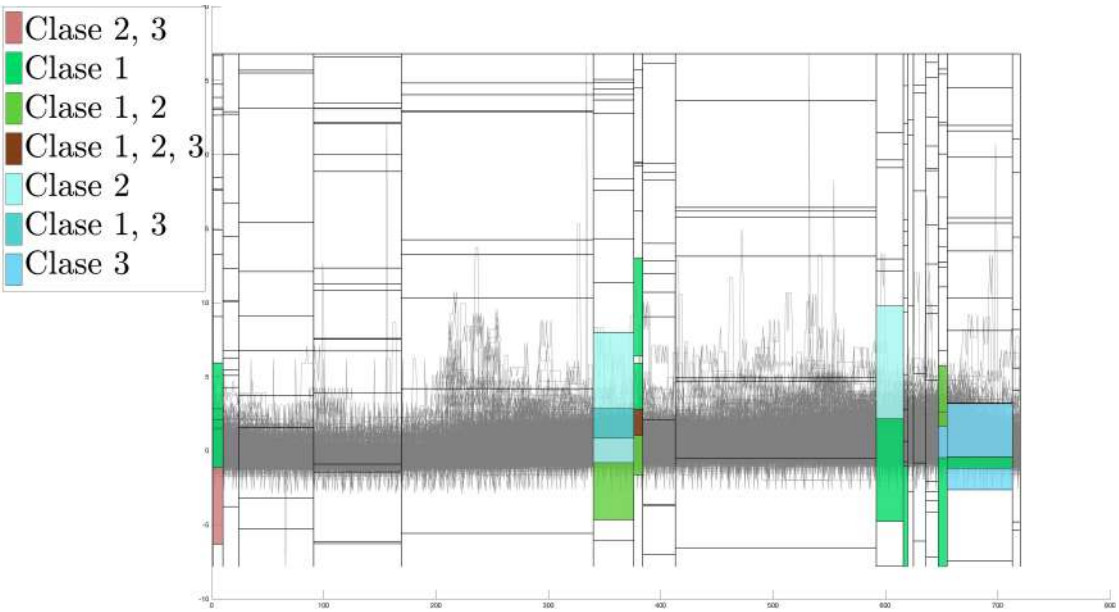


Figura B.60: Distribución de las clases para la base de datos ScreenType extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.61. ShapeletSim

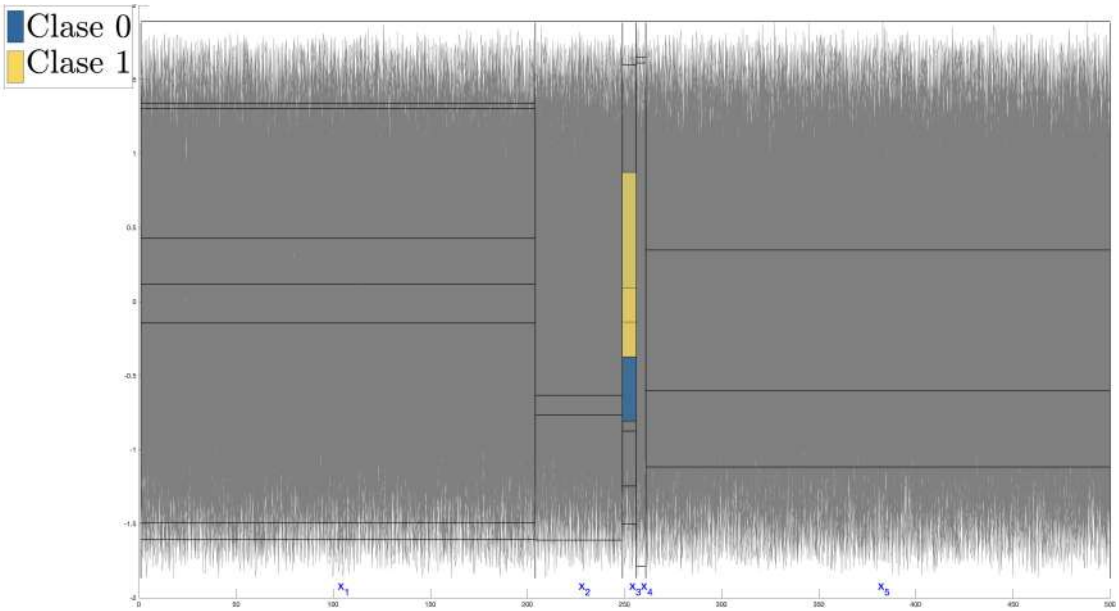


Figura B.61: Distribución de las clases para la base de datos ShapeletSim extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.62. ShapesAll

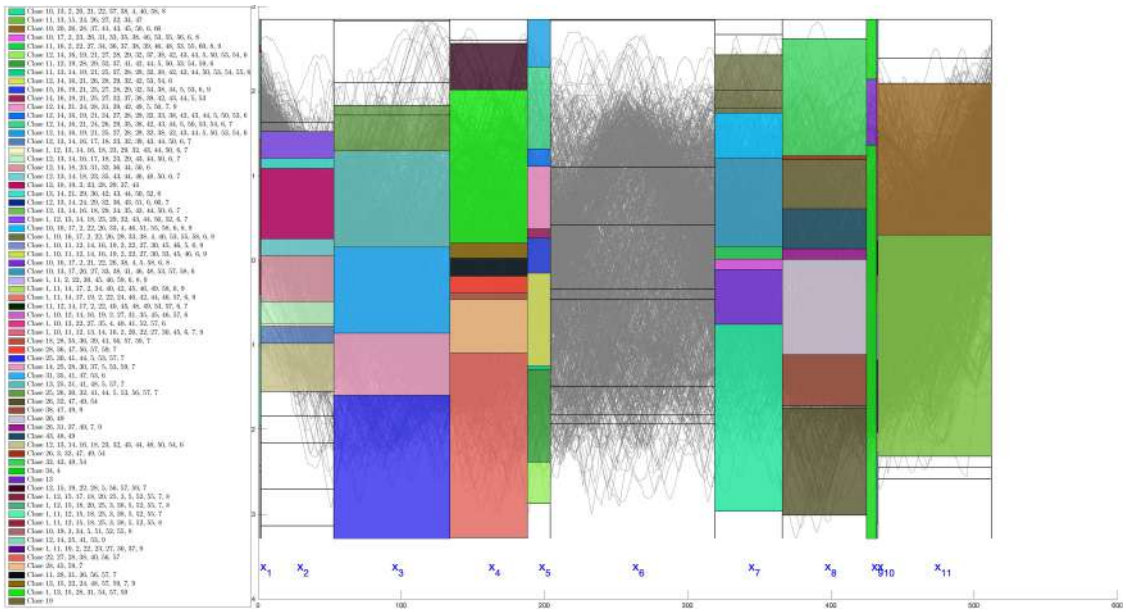


Figura B.62: Distribución de las clases para la base de datos ShapesAll extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.63. SmallKitchenAppliances

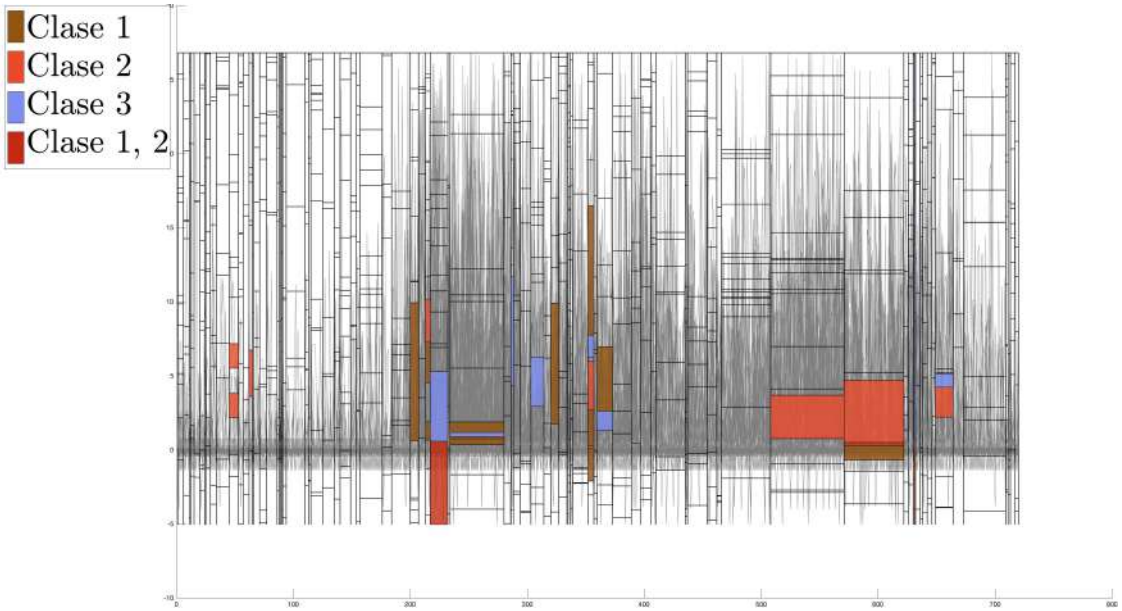


Figura B.63: Distribución de las clases para la base de datos SmallKitchenAppliances extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.64. SonyAIBORobotSurface1

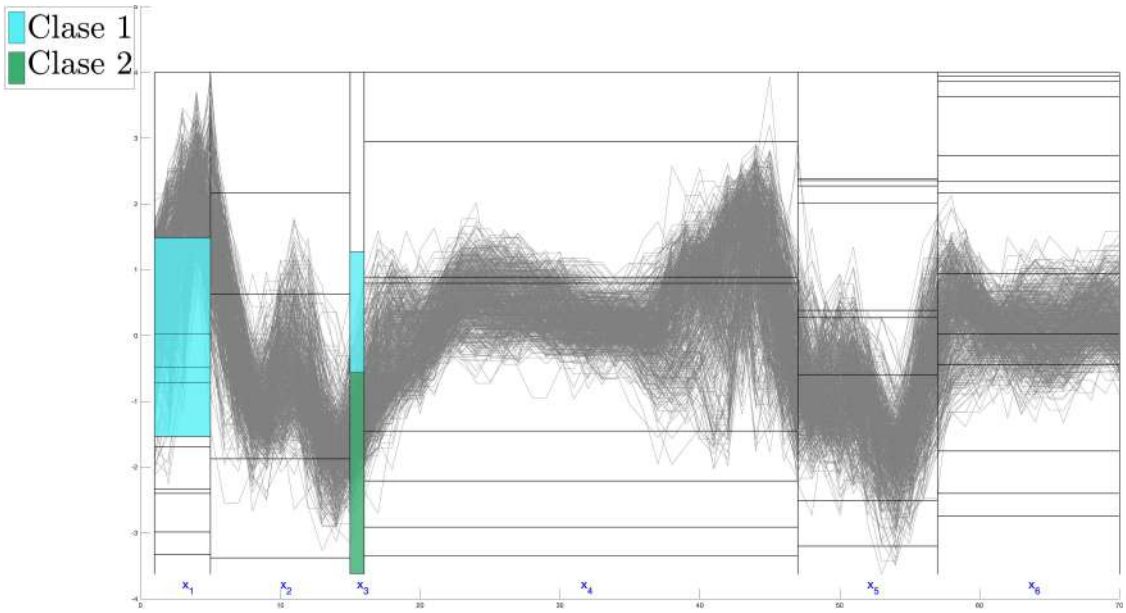


Figura B.64: Distribución de las clases para la base de datos SonyAIBORobotSurface1 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.65. SonyAIBORobotSurface2

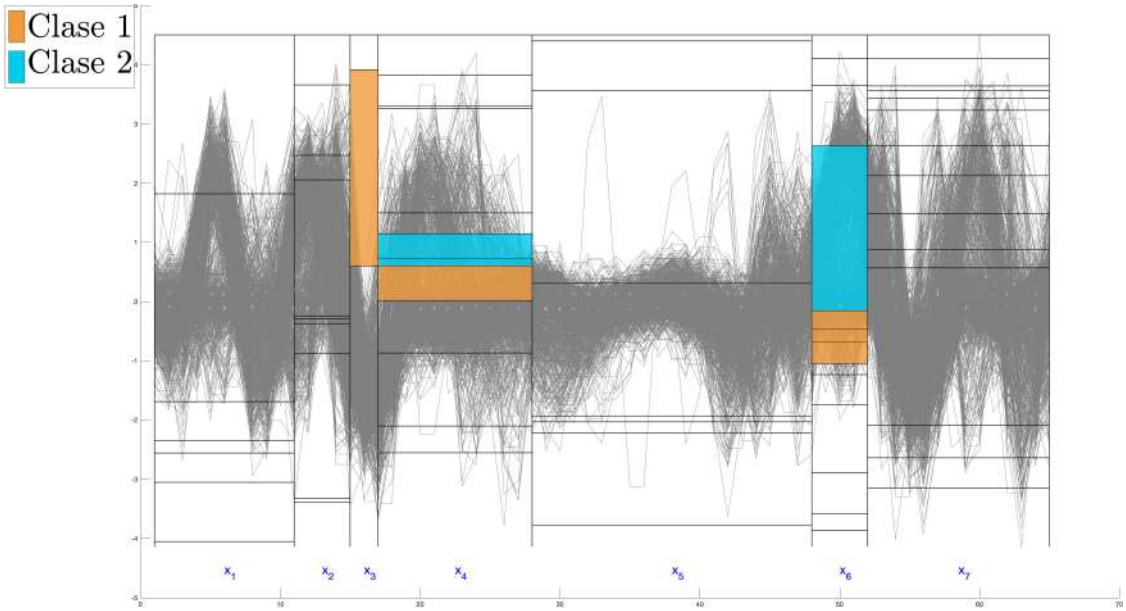


Figura B.65: Distribución de las clases para la base de datos SonyAIBORobotSurface2 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.66. StarLightCurves

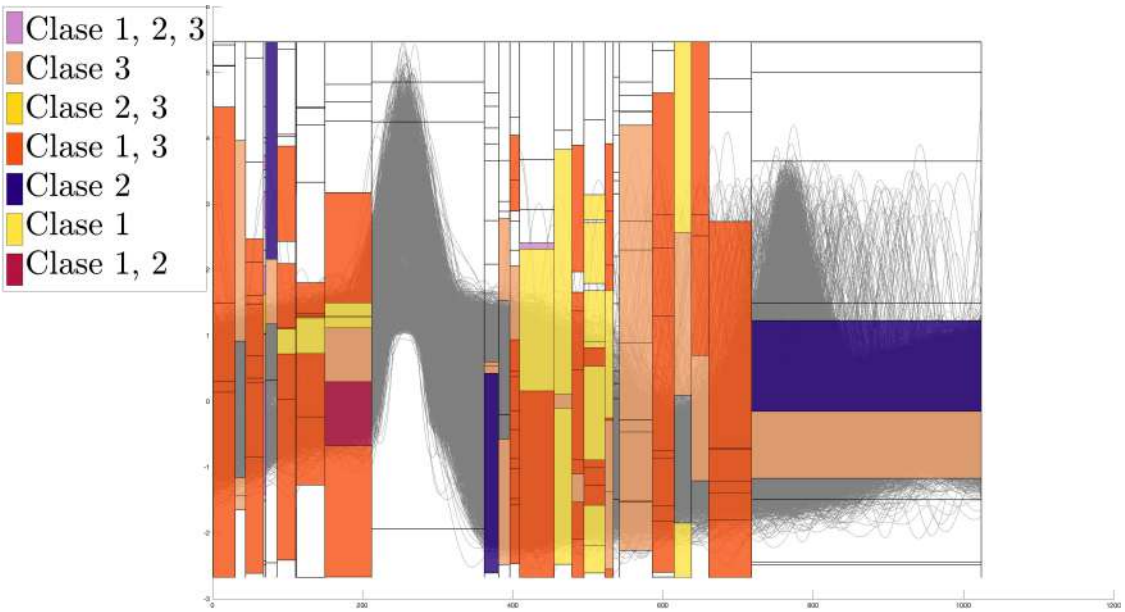


Figura B.66: Distribución de las clases para la base de datos StarLightCurves extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.67. Strawberry

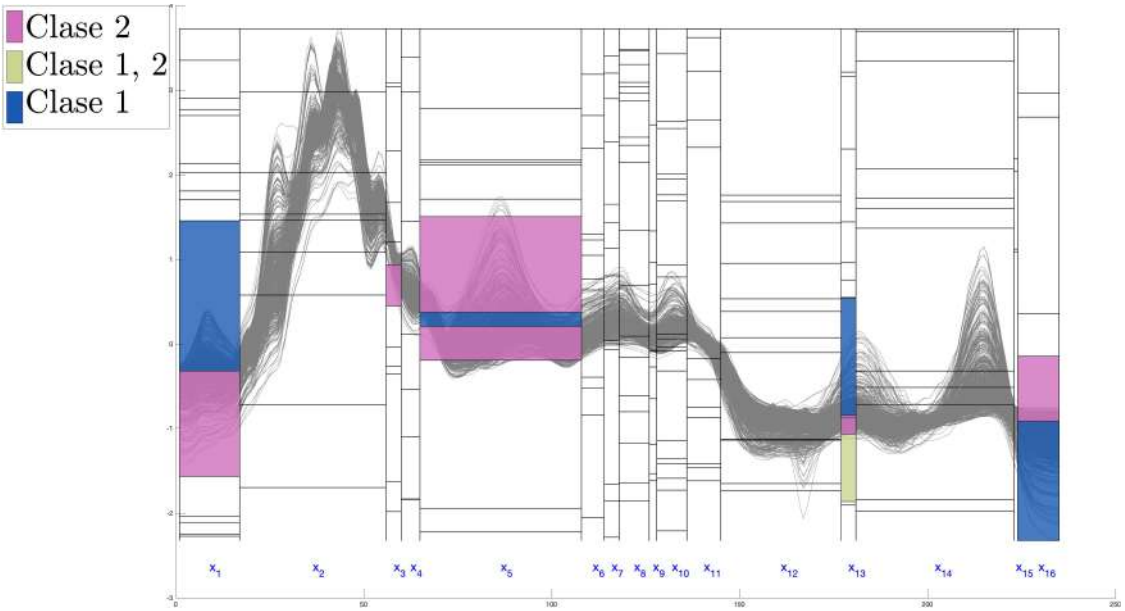


Figura B.67: Distribución de las clases para la base de datos Strawberry extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.68. SwedishLeaf

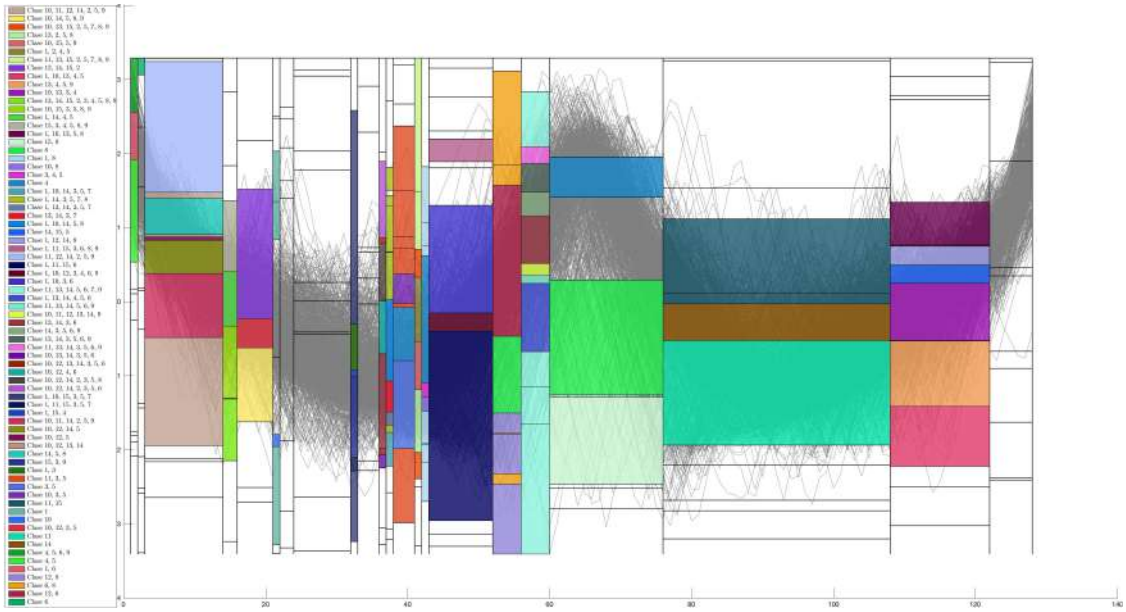


Figura B.68: Distribución de las clases para la base de datos SwedishLeaf extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.69. Symbols

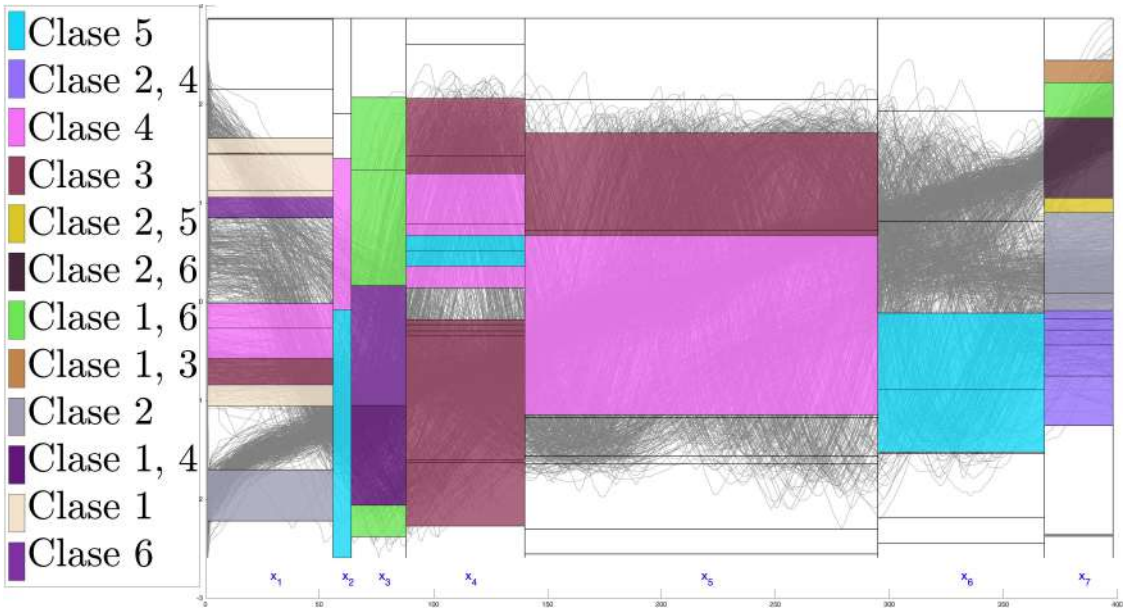


Figura B.69: Distribución de las clases para la base de datos Symbols extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.70. SyntheticControl

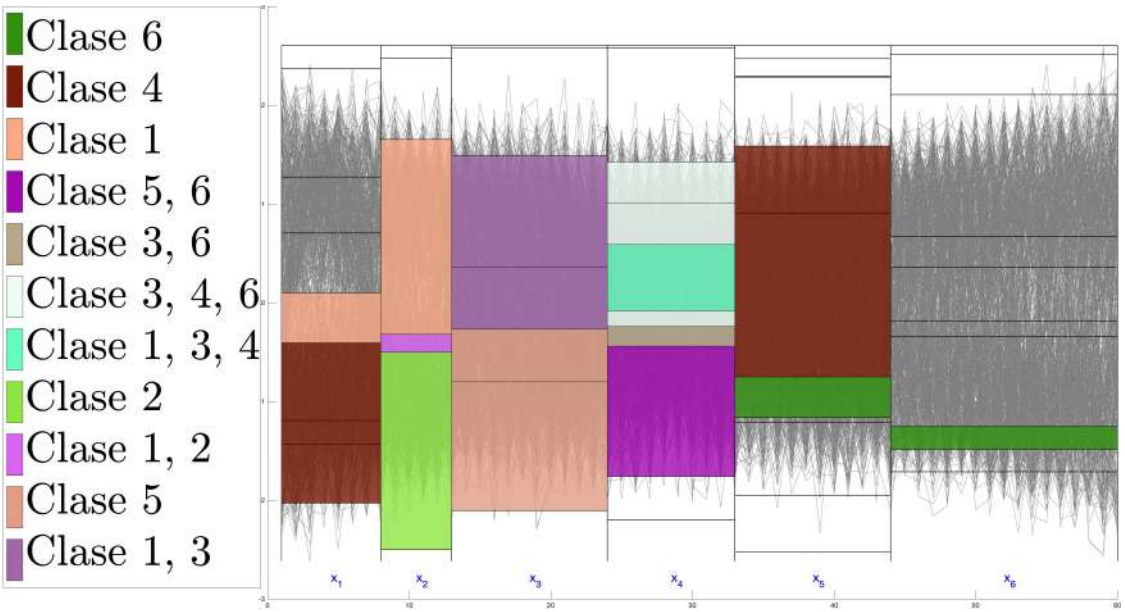


Figura B.70: Distribución de las clases para la base de datos SyntheticControl extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.71. ToeSegmentation1

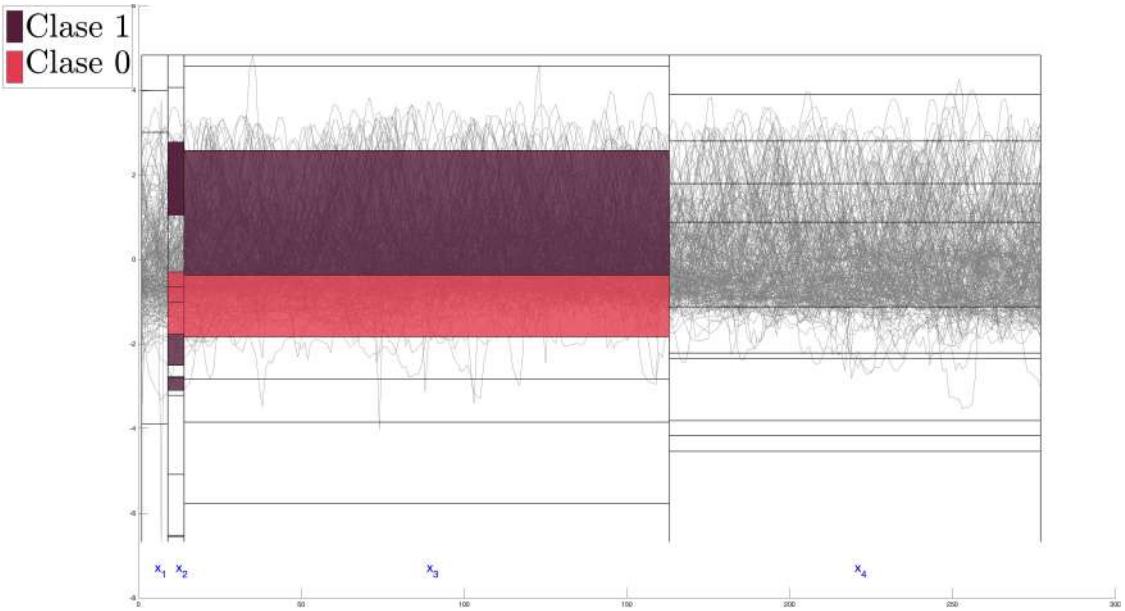


Figura B.71: Distribución de las clases para la base de datos ToeSegmentation1 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.72. ToeSegmentation2

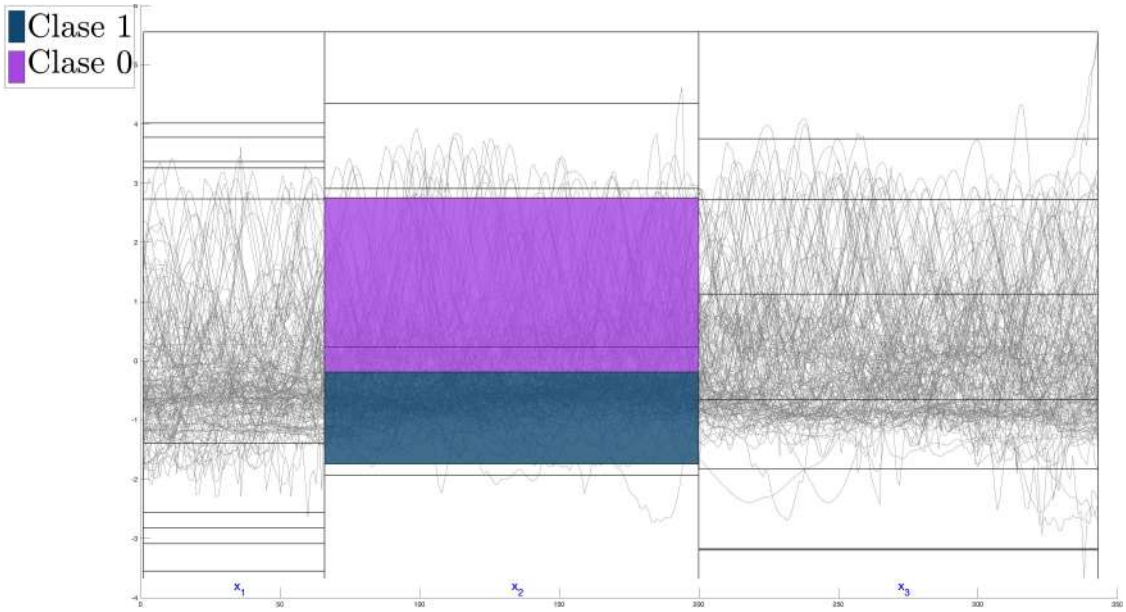


Figura B.72: Distribución de las clases para la base de datos ToeSegmentation2 extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.73. Trace

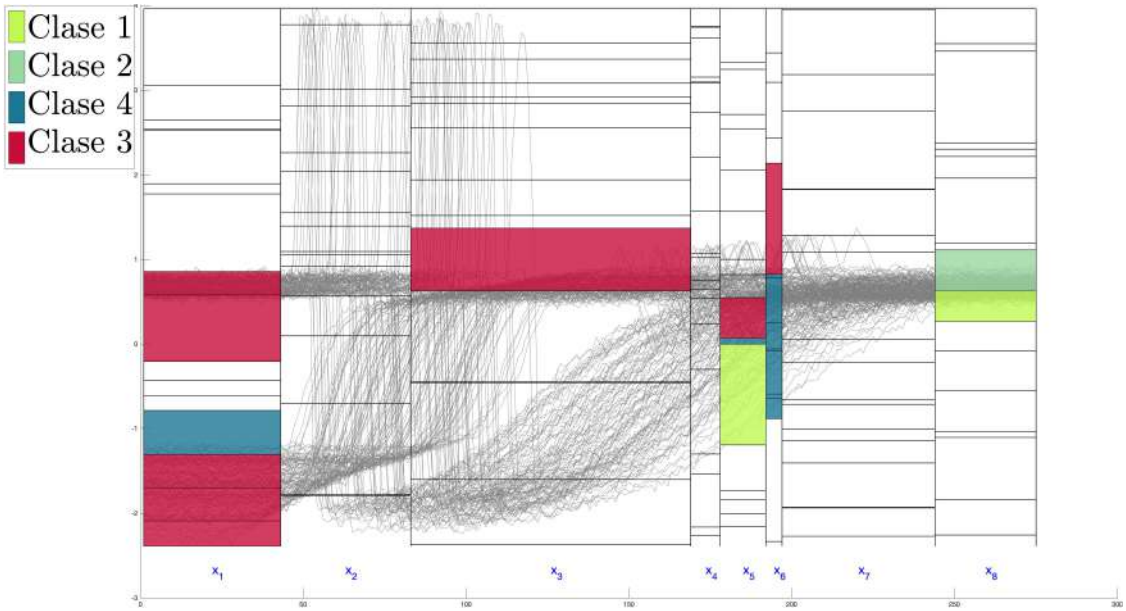


Figura B.73: Distribución de las clases para la base de datos Trace extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.74. TwoLeadECG

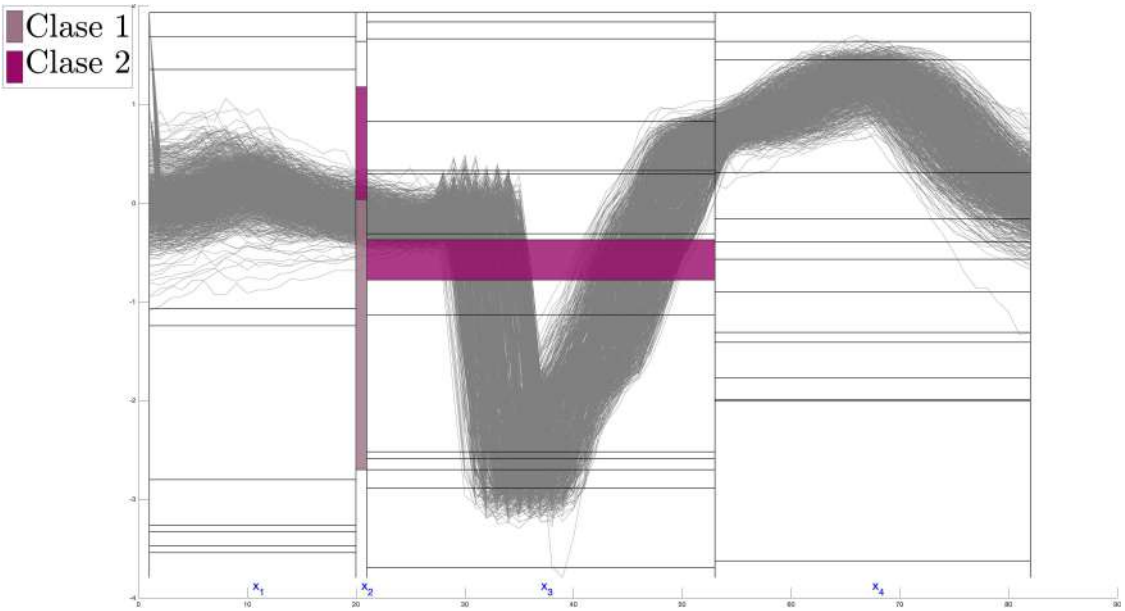


Figura B.74: Distribución de las clases para la base de datos TwoLeadECG extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.75. TwoPatterns

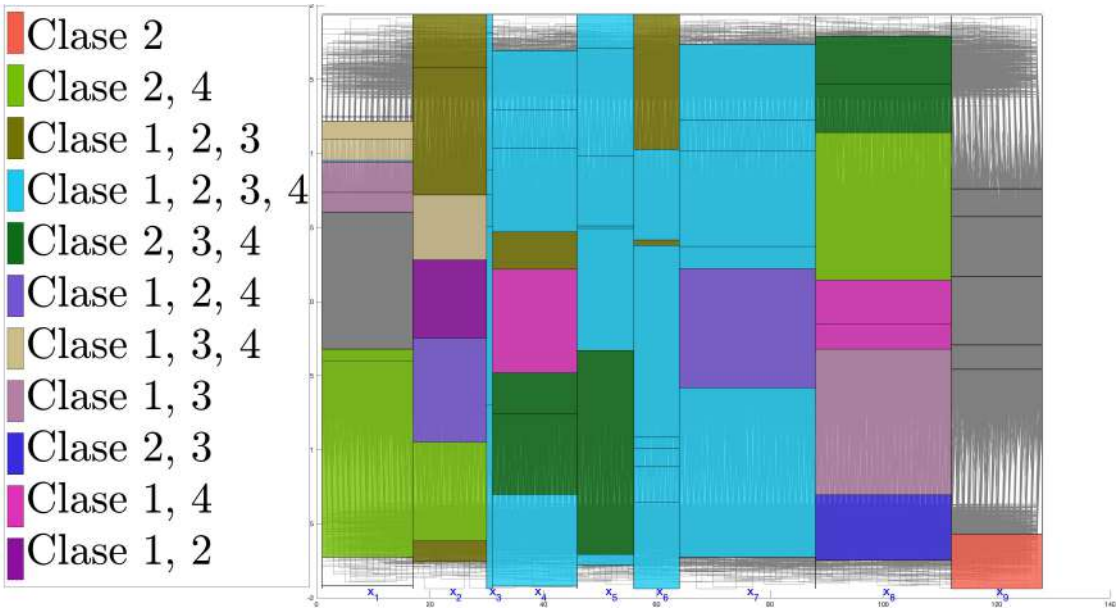


Figura B.75: Distribución de las clases para la base de datos TwoPatterns extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.76. UWaveGestureLibraryAll

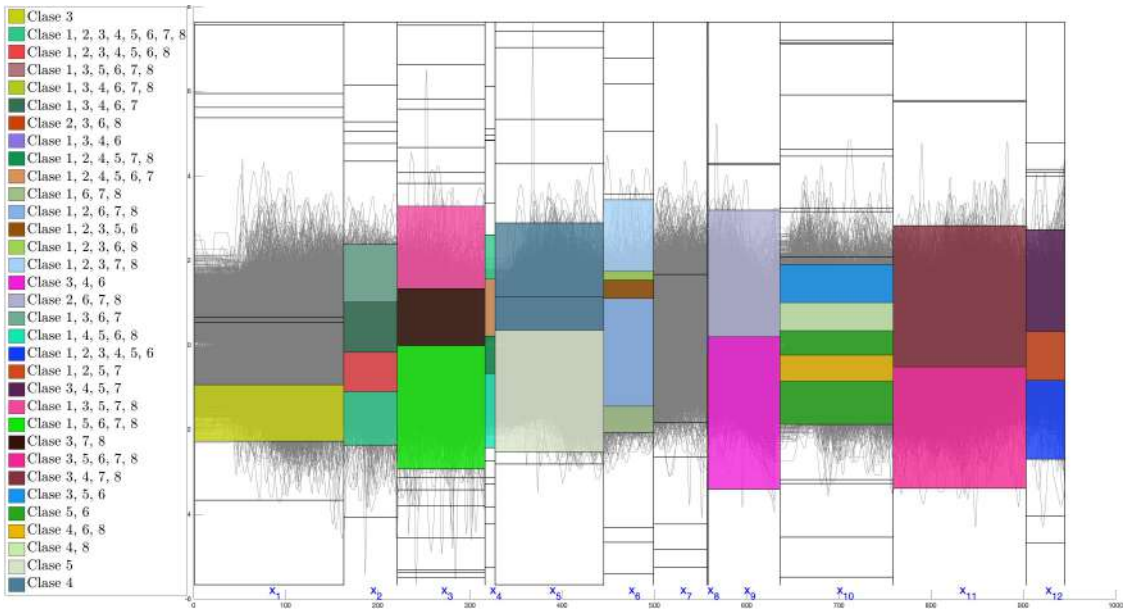


Figura B.76: Distribución de las clases para la base de datos UWaveGestureLibraryAll extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.77. UWaveGestureLibraryX

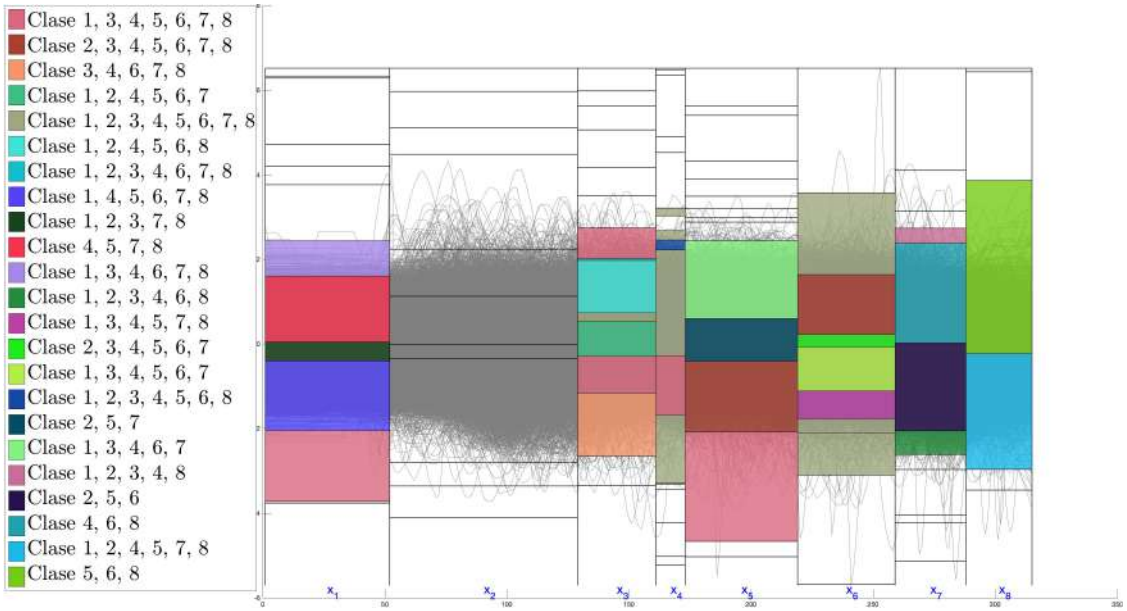


Figura B.77: Distribución de las clases para la base de datos UWaveGestureLibraryX extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.78. UWaveGestureLibraryY

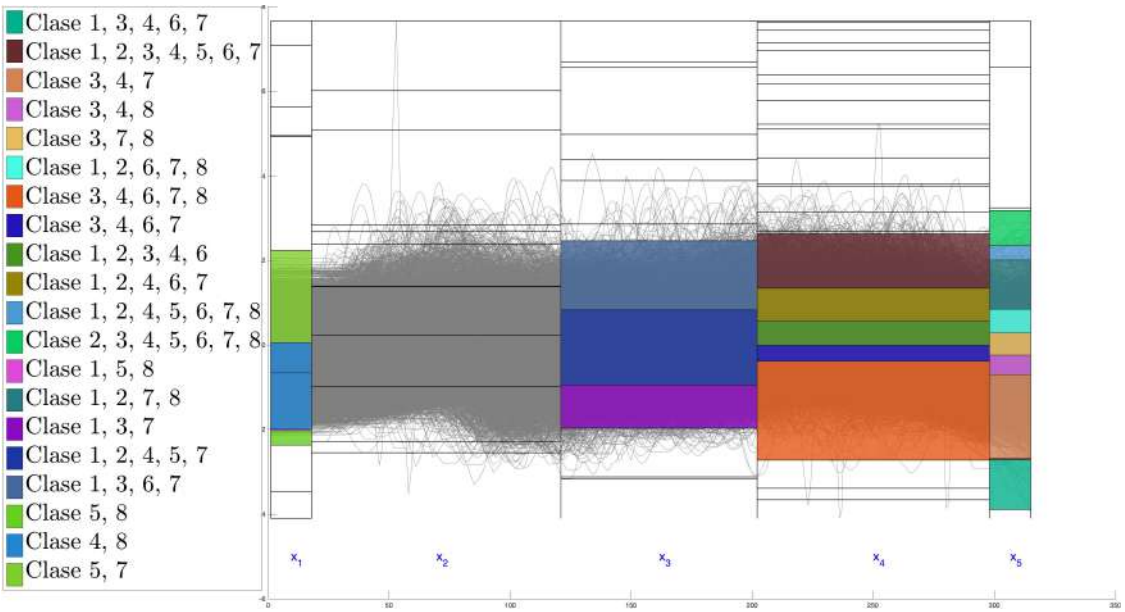


Figura B.78: Distribución de las clases para la base de datos UWaveGestureLibraryY extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.79. UWaveGestureLibraryZ

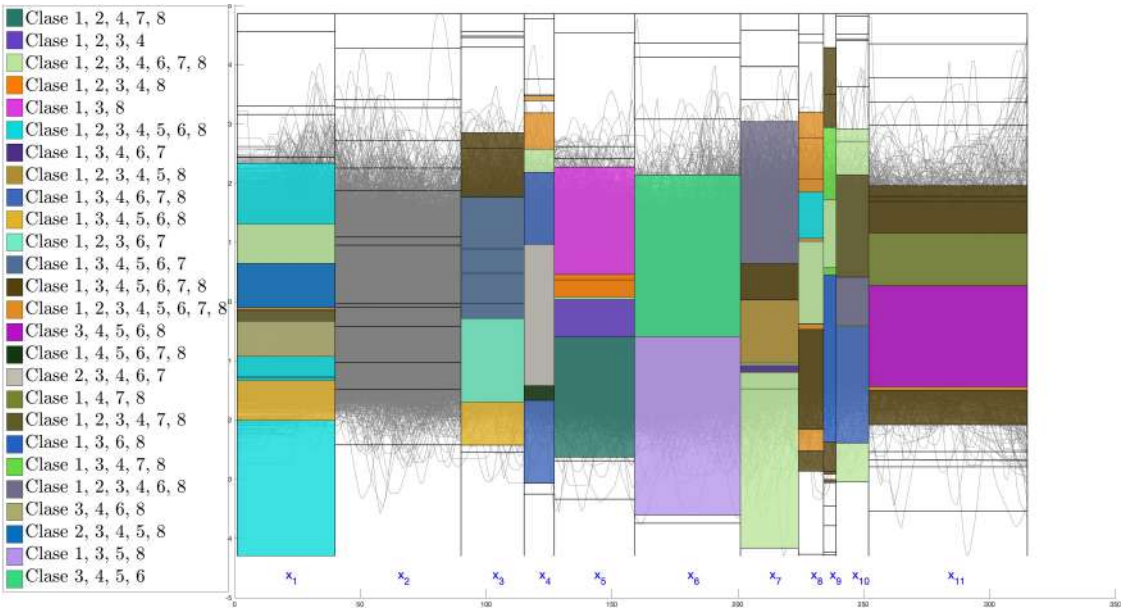


Figura B.79: Distribución de las clases para la base de datos UWaveGestureLibraryZ extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.80. Wafer

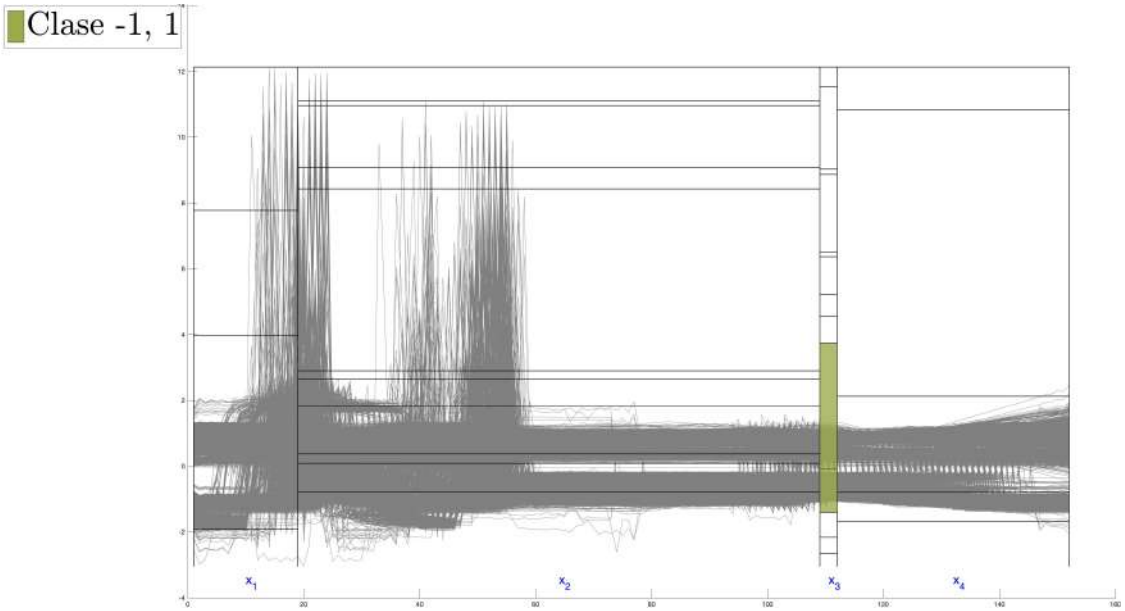


Figura B.80: Distribución de las clases para la base de datos Wafer extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.81. Wine

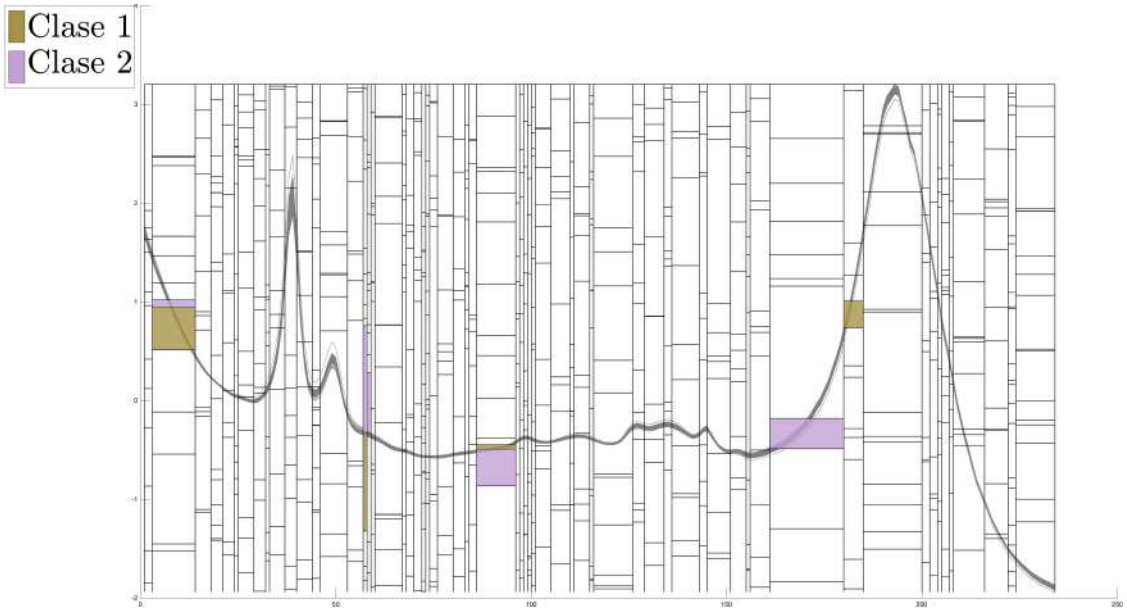


Figura B.81: Distribución de las clases para la base de datos Wine extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.82. WordSynonyms

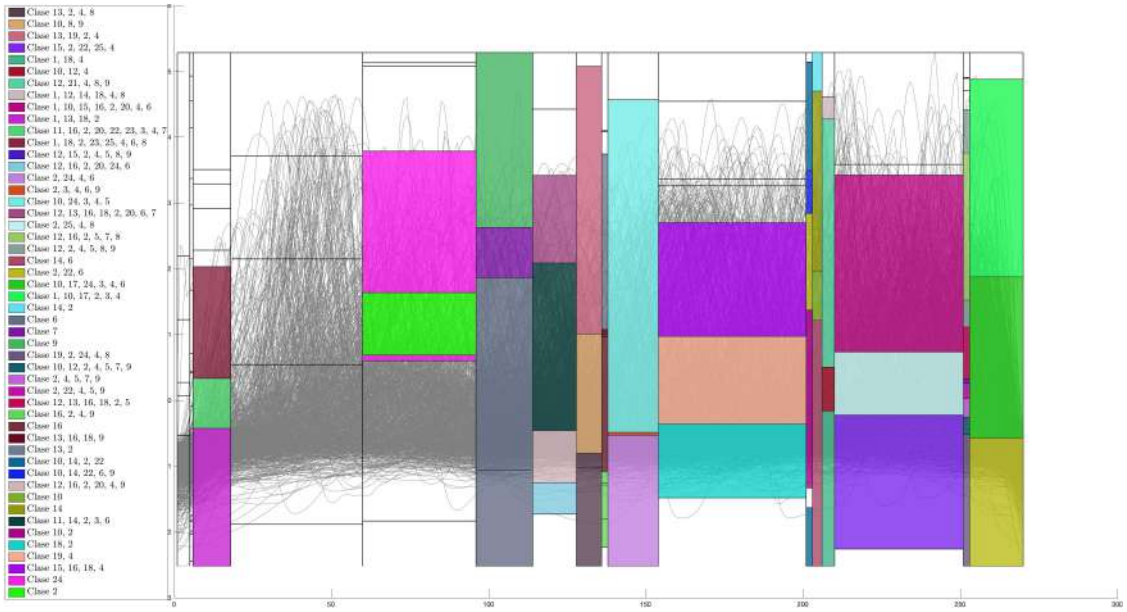


Figura B.82: Distribución de las clases para la base de datos WordSynonyms extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.83. Worms

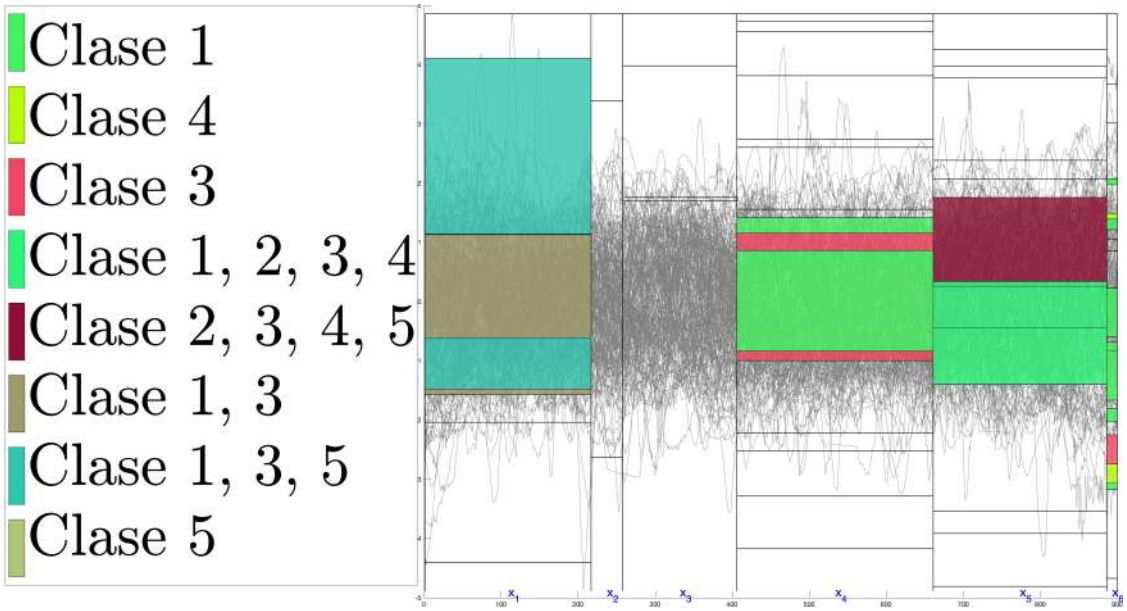


Figura B.83: Distribución de las clases para la base de datos Worms extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.84. WormsTwoClass

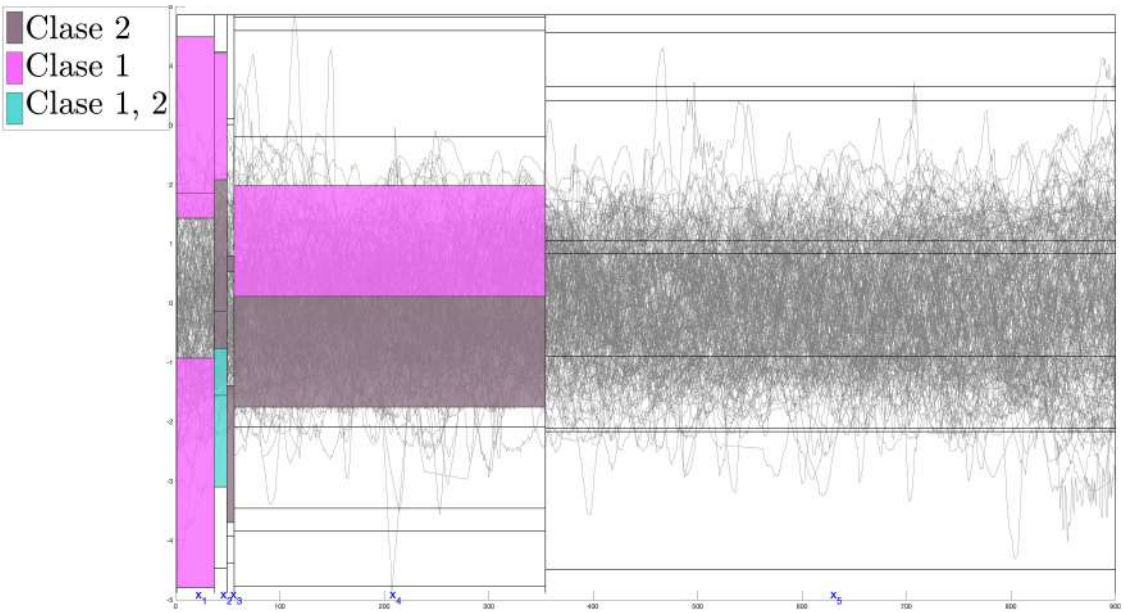


Figura B.84: Distribución de las clases para la base de datos WormsTwoClass extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

B.85. Yoga

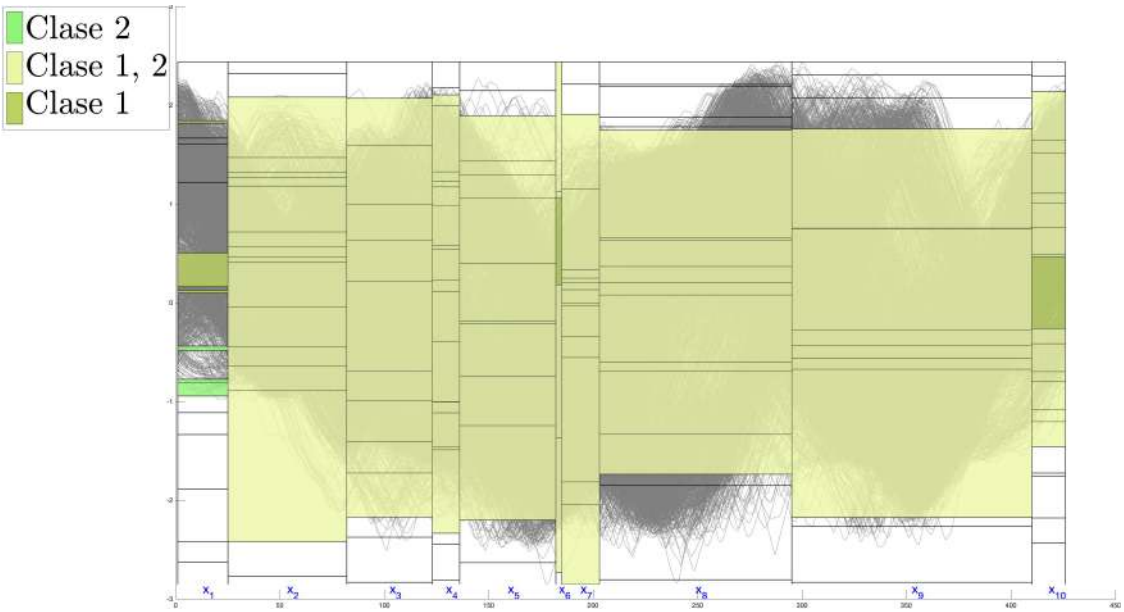


Figura B.85: Distribución de las clases para la base de datos Yoga extraída del árbol obtenido por eMODiTS. Cada segmento en el eje del tiempo representa un nodo interno (x_i) del árbol y cada rectángulo de color representa un nodo hoja.

Bibliografía

[1] *Dynamic Time Warping*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[2] H. G. Acosta-Mesa, F. Rechy-Ramírez, E. Mezura-Montes, N. Cruz-Ramírez, and R. Jiménez-Hernández. Application of time series discretization using evolutionary programming for classification of precancerous cervical lesions. *J Biomed Inform*, 2014.

[3] H.G. Acosta-Mesa, N. Cruz-Ramírez, and D.A. García-López. Entropy based linear approximation algorithm for time series discretization. *Advances in Artificial Intelligence and Applications*, 32:214–224, 2008.

[4] Charu C. Aggarwal. *Data Mining: The Textbook*. Springer Publishing Company, Incorporated, 2015.

[5] Almahdi M Ahmed, Azuraliza Abu Bakar, and Abdul Razak Hamdan. A harmony search algorithm with multi-pitch adjustment rate for symbolic time series data representation. *International Journal of Modern Education and Computer Science*, 6(6):58, 2014.

[6] A.M. Ahmed, A.A. Bakar, and A.R. Hamdan. Harmony search algorithm for optimal word size in symbolic time series representation. In *Data Mining and Optimization (DMO), 2011 3rd Conference on*, pages 57–62, June 2011.

[7] Najmeh Alikar, Seyed Mohsen Mousavi, Raja Ariffin Raja Ghazilla, Madjid Tavana, and Ezutah Udoncy Olugu. Application of the nsga-ii algorithm to a multi-period inventory-redundancy allocation problem in a series-parallel system. *Reliability Engineering & System Safety*, 160:1–10, 2017.

[8] R Azulay, R. Moskovitch, D. Stopel, M. Verduijn, E. De Jonge, and Y. Shahar. Temporal discretization of medical time series: A comparative study. In *Workshop on Intelligent Data Analysis in Biomedicine and Pharmacology*. Amsterdam, The Netherlands, 2007.

[9] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, Online First, 2016.

[10] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent

algorithmic advances. *Data Mining and Knowledge Discovery*, Online First, 2016.

[11] Xue Bai, Yun Xiong, Yangyong Zhu, and Hengshu Zhu. Time series representation: a random shifting perspective. In *International Conference on Web-Age Information Management*, pages 37–50. Springer, 2013.

[12] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multi-objective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[13] A. Bondu, M. Boullé, and B. Grossin. Saxo: An optimized data-driven symbolic representation of time series. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9, Aug 2013.

[14] Alexis Bondu, Marc Boullé, and Antoine Cornuéjols. *Symbolic Representation of Time Series: A Hierarchical Coclustering Formalization*, pages 3–16. Springer International Publishing, Cham, 2016.

[15] U. Boryczka and J. Kozak. An adaptive discretization in the acdt algorithm for continuous attributes. In Piotr Jedrzejowicz, Ngoc Thanh Nguyen, and Kiem Hoang, editors, *ICCCI (2)*, volume 6923 of *Lecture Notes in Computer Science*, pages 475–484. Springer, 2011.

[16] L.P Braga, L. Ortíz, and S.S. Ramirez. *Introducción a la Minería de Datos*. E-PAPERS.

[17] Max Bramer. *Principles of data mining*. Springer Publishing Company, Incorporated, 2013.

[18] Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski, editors. *Multiobjective Optimization, Interactive and Evolutionary Approaches [outcome of Dagstuhl seminars].*, volume 5252 of *Lecture Notes in Computer Science*. Springer, 2008.

[19] Andrea Brunello, Enrico Marzano, Angelo Montanari, and Guido Sciavicco. A novel decision tree approach for the handling of time series. In *International Conference on Mining Intelligence and Knowledge Exploration*, pages 351–368. Springer, 2018.

[20] LJ Cao, Kok Seng Chua, WK Chong, HP Lee, and QM Gu. A comparison of pca, kpca and ica for dimensionality reduction in support vector machine. *Neurocomputing*, 55(1-2):321–336, 2003.

[21] R. I.;Leaver D.S. Cape, J.N.;Smith. Cleaned uk rainfall chemistry data (1986 - 2011), 2014.

[22] Victor M. Carrillo and Heidi Taboada. A post-pareto approach for multi-objective decision making using a non-uniform weight generator method.

Procedia Computer Science, 12:116 – 121, 2012. Complex Adaptive Systems 2012.

[23] Carmelo Cassisi, Placido Montalto, Marco Aliotta, Andrea Cannata, and Alfredo Pulvirenti. Similarity measures and dimensionality reduction techniques for time series data mining. In *Advances in data mining knowledge discovery and applications*. InTech, 2012.

[24] Yair Censor. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization*, 4(1):41–59, Mar 1977.

[25] L Chambers, editor. *The Practical Handbook of Genetic Algorithms: Applications, 2nd edition*. Chapman and Hall CRC, 2001.

[26] P. Chaudhari, Dipti P. Rana, Rupa G. Mehta, Narendra. J. Mistry, and Mukesh M. Raghuwanshi. Discretization of temporal data: A survey. *CoRR*, abs/1402.4283, 2014.

[27] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

[28] Jared L Cohon and David H Marks. A review and evaluation of multiobjective programing techniques. *Water Resources Research*, 11(2):208–220, 1975.

[29] J.D. Cryer and K.S. Chan. *Time Series Analysis: With Applications in R*. Springer Texts in Statistics. Springer, 2008.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Trans. Evol. Comp*, 6(2):182–197, April 2002.

[31] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[32] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[33] Elena S Dimitrova, M Paola Vera Licona, John McGee, and Reinhard Laubenbacher. Discretization of time series data. *Journal of Computational Biology*, 17(6):853–868, 2010.

[34] Hui Ding, Goce Trajcevski, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. In *In Proc of the 34 th VLDB*, pages 1542–1552, 2008.

[35] Yezid Donoso and Ramon Fabregat. *Multi-objective optimization in computer networks using metaheuristics*. Auerbach Publications, 2016.

[36] H. dos Santos Passos, F. G. S. Teodoro, B. M. Duru, E. L. de Oliveira, S. M. Peres, and C. A. M. Lima. Symbolic representations of time series applied to biometric recognition based on ecg signals. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3199–3207, May 2017.

[37] F. Y. Edgeworth. Mathematical psychics : An essay on the application of mathematics to the moral sciences. london : Kegan paul, 1881.

[38] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1):12:1–12:34, December 2012.

[39] Usama Fayyad and Keki Irani. Multi-interval discretization of continuous-valued attributes for classification learning. 1993.

[40] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.

[41] J. C. Ferreira, C. M. Fonseca, and A. Gaspar-Cunha. Methodology to select solutions from the pareto-optimal set: A comparative study. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 789–796, New York, NY, USA, 2007. ACM.

[42] Peter Ffoulkes. *insideBIGDATA Guide to Use of Big Data on an Industrial Scale*, 2017 (accessed May 24, 2018).

[43] Carlos M Fonseca, Peter J Fleming, et al. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Icga*, volume 93, pages 416–423, 1993.

[44] Tak-Chung Fu. A review on time series data mining. *Eng. Appl. Artif. Intell.*, 24(1):164–181, February 2011.

[45] Muhammad Marwan Muhammad Fuad. Differential evolution versus genetic algorithms: Towards symbolic aggregate approximation of non-normalized time series. In *Proceedings of the 16th International Database Engineering & Applications Sysmposium*, pages 205–210. ACM, 2012.

[46] Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044 – 2064, 2010. Special Issue on Intelligent Distributed Information Systems.

[47] D. García-López. Algoritmo de discretización de series de tiempo basado en entropía y su aplicación en datos colposcópicos. Master's thesis, Departamento de Inteligencia Artificial de la Universidad Veracruzana, 2007.

[48] D. A. García-López, H. G. Acosta-Mesa, and N. Cruz-Ramírez. Entropy based linear approximation algorithm for time series discretization. *Advances in Artificial Inteligent and Applications*, 32:214–224, 2008.

[49] D.A. García-López and Héctor-Gabriel A.M. Discretization of time series dataset with a genetic search. In *Proceedings of the 8th Mexican International Conference on Artificial Intelligence*, MICAI '09, pages 201–212, Berlin, Heidelberg, 2009. Springer-Verlag.

[50] Saul Gass and Thomas Saaty. The computational algorithm for the parametric objective function. *Naval Research Logistics Quarterly*, 2, 03 1955.

[51] F Gembicki and Y Haimes. Approach to performance and sensitivity multiobjective optimization: The goal attainment method. *IEEE Transactions on Automatic control*, 20(6):769–771, 1975.

[52] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[53] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining concepts and techniques, third edition, 2012.

[54] Miettinen Kaisa. *Nonlinear Multiobjective Optimization*, volume 12 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Boston, USA, 1999.

[55] S. Kannan, S. Baskar, J. D. McCalley, and P. Murugan. Application of nsga-ii algorithm to generation expansion planning. *IEEE Transactions on Power Systems*, 24(1):454–461, Feb 2009.

[56] Mehmed Kantardzic. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

[57] E. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 206–215, New York, NY, USA, 2004. ACM.

[58] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[59] S. B. Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283, Apr 2013.

[60] L. A. Kurgan and K. J. Cios. Caim discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):145–153, Feb 2004.

[61] Lukasz A Kurgan and Krzysztof J Cios. Caim discretization algorithm. *IEEE transactions on Knowledge and Data Engineering*, 16(2):145–153, 2004.

[62] M. Last, A. Kandel, and H. Bunke. *Data Mining in Time Series Databases*. Series in machine perception and artificial intelligence. World Scientific, 2004.

[63] Philippe Lenca, Stéphane Lallich, Thanh-Nghi Do, and Nguyen-Khang Pham. A comparison of different off-centered entropies to deal with class imbalance for decision trees. In Takashi Washio, Einoshin Suzuki, Kai Ming Ting, and Akihiro Inokuchi, editors, *Advances in Knowledge Discovery and Data Mining*, pages 634–643, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[64] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, pages 2–11, New York, NY, USA, 2003. ACM.

[65] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, Oct 2007.

[66] Jinfu Liu, Qinghua Hu, and Daren Yu. A weighted rough set based method developed for class imbalance learning. *Information Sciences*, 178(4):1235 – 1256, 2008.

[67] Battuguldur Lkhagva, Yu Suzuki, and Kyoji Kawagoe. Extended sax: Extension of symbolic aggregate approximation for financial time series data representation. *DEWS2006 4A-i8*, 7, 2006.

[68] Battuguldur Lkhagva, Yu Suzuki, and Kyoji Kawagoe. New time series data representation esax for financial applications. In *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, pages x115–x115. IEEE, 2006.

[69] Simon Malinowski, Thomas Guyet, René Quiniou, and Romain Tavenard. *1d-SAX: A Novel Symbolic Representation for Time Series*, pages 273–284. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[70] R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 2004.

[71] Aldo Márquez-Grajales, Héctor Gabriel Acosta-Mesa, and Efren Mezura-Montes. An adaptive symbolic discretization scheme for the classification of temporal datasets using nsga-ii. In *2017 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pages 1–8, Nov 2017.

[72] María-Guadalupe Martínez-Peñaloza, Efrén Mezura-Montes, Nicandro Cruz-Ramírez, Héctor-Gabriel Acosta-Mesa, and Homero-Vladimir Ríos-Figueroa.

Improved multi-objective clustering with automatic determination of the number of clusters. *Neural Computing and Applications*, 28(8):2255–2275, 2017.

[73] Anabel Martínez-Vargas, Josué Domínguez-Guerrero, Ángel G Andrade, Roberto Sepúlveda, and Oscar Montiel-Ross. Application of nsga-ii algorithm to the spectrum assignment problem in spectrum sharing networks. *Applied Soft Computing*, 39:188–198, 2016.

[74] Theophano Mitsa. *Temporal Data Mining*. Chapman & Hall/CRC, 1st edition, 2010.

[75] Jacqueline Moore, Richard Chapman, and Gerry Dozier. Multiobjective particle swarm optimization. In *Proceedings of the 38th annual on Southeast regional conference*, pages 56–57. ACM, 2000.

[76] Fabian Mörchen and Alfred Ultsch. Optimizing time series discretization for knowledge discovery. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 660–665. ACM, 2005.

[77] Muhammad Marwan Muhammad Fuad. *Genetic Algorithms-Based Symbolic Aggregate Approximation*, pages 105–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[78] Vilfredo Pareto. *Cours d'Economie Politique*. Droz, Genève, 1896.

[79] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing*, SAC '02, pages 603–607, New York, NY, USA, 2002. ACM.

[80] Nancy Pérez-Castro, Héctor Gabriel Acosta-Mesa, Efrén Mezura-Montes, and Hugo Jair Escalante. Multi-objective full model selection in temporal databases: Optimizing time and performance. In *Power, Electronics and Computing (ROPEC), 2016 IEEE International Autumn Meeting on*, pages 1–6. IEEE, 2016.

[81] N. D. Pham, Q. L. Le, and T. K. Dang. Hot αsax: A novel adaptive symbolic representation for time series discords discovery. In Ngoc Thanh Nguyen, Manh Thanh Le, and Jerzy Swiatek, editors, *ACIIDS (1)*, volume 5990 of *Lecture Notes in Computer Science*, pages 113–121. Springer, 2010.

[82] N. D. Pham, Q. L. Le, and T. K. Dang. Two novel adaptive symbolic representations for similarity search in time series databases. In *2010 12th International Asia-Pacific Web Conference*, pages 181–187, April 2010.

[83] Masoud Rabbani, Hamed Farrokhi-Asl, and Bahare Asgarian. Solving a bi-objective location routing problem by a nsga-ii combined with clustering

approach: application in waste collection problem. *Journal of Industrial Engineering International*, 13(1):13–27, 2017.

[84] Kajal Rai, M Syamala Devi, and Ajay Guleria. Decision tree based algorithm for intrusion detection. *International Journal of Advanced Networking and Applications*, 7(4):2828, 2016.

[85] Chotirat Ann Ralanamahatana, Jessica Lin, Dimitrios Gunopulos, Eamonn Keogh, Michail Vlachos, and Gautam Das. *Mining Time Series Data*, pages 1069–1103. Springer US, Boston, MA, 2005.

[86] Gade Pandu Rangaiah. *Multi-objective optimization: techniques and applications in chemical engineering*, volume 1. World Scientific, 2009.

[87] G.P. Rangaiah. *Multi-Objective Optimization: Techniques and Applications in Chemical Engineering*. Advances in process systems engineering. World Scientific, 2009.

[88] F. Rechy-Ramírez, H. G. Acosta-Mesa, E. Mezura-Montes, and N. Cruz-Ramírez. Times series discretization using evolutionary programming. In Ildar Z. Batyrshin and Grigori Sidorov, editors, *MICAI (2)*, volume 7095 of *Lecture Notes in Computer Science*, pages 225–234. Springer, 2011.

[89] Fernando Rechy-Ramírez. Discretización de series de tiempo usando un algoritmo genético multiobjetivo. Master's thesis, Universidad Veracruzana, Xalapa, Veracruz, México, 2010.

[90] Claude Sammut and Geoffrey I. Webb, editors. *Encyclopedia of Machine Learning and Data Mining*. Springer, 2017.

[91] A. Sant'Anna and N. Wickström. Symbolization of time-series: An evaluation of sax, persist, and aca. In *2011 4th International Congress on Image and Signal Processing*, volume 4, pages 2223–2228, Oct 2011.

[92] Anita Sant'Anna and Nicholas Wickström. Symbolization of time-series: An evaluation of sax, persist, and aca. In *Image and Signal Processing (CISP), 2011 4th International Congress on*, volume 4, pages 2223–2228. IEEE, 2011.

[93] Wei Song, Zhiguang Wang, Fan Zhang, Yangdong Ye, and Ming Fan. Empirical study of symbolic aggregate approximation for time series classification. *Intelligent Data Analysis*, 21(1):135–150, 2017.

[94] Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano. A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*, 03 2014.

[95] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, Sept 1994.

[96] H Taboada and D Coit. Post-pareto optimality analysis to efficiently identify promising solutions for multi-objective problems. In *Rutgers University ISE Working Paper*, pages 05–15, 2005.

[97] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[98] Mark Velasquez and Patrick Hester. An analysis of multi-criteria decision making methods, 05 2013.

[99] Michel Verleysen and Damien François. The curse of dimensionality in data mining and time series prediction. In *International Work-Conference on Artificial Neural Networks*, pages 758–770. Springer, 2005.

[100] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 673–684. IEEE, 2002.

[101] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, Amsterdam, 3 edition, 2011.

[102] Y Y. Haimes, Leon Lasdon, and D A. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics - TSMC*, 1:296–297, 07 1971.

[103] Y. Yamada, H. Yokoi, and K. Takabayashi. Decision-tree induction from time-series data based on standard-example split test. In *In Proceedings of the 20th International Conference on Machine Learning (ICML03)*, pages 840–847. Morgan Kaufmann, 2003.

[104] M. Yin, S. Tangsripairoj, and B. Pupacdi. Variable length motif-based time series classification. In *Recent Advances in Information and Communication Technology*, volume 265 of *Advances in Intelligent Systems and Computing*, pages 73–82. Springer International Publishing, 2014.

[105] S. Yin and O. Kaynak. Big data for modern industry: Challenges and trends [point of view]. *Proceedings of the IEEE*, 103(2):143–146, Feb 2015.

[106] Willian Zalewski, Fabiano Silva, Feng Chung Wu, Huei Diana Lee, and André Gustavo Maletzke. *A Symbolic Representation Method to Preserve the Characteristic Slope of Time Series*, pages 132–141. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[107] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, Dec 2007.

[108] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov 1999.

[109] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-Report*, 103, 07 2001.

[110] Eckart Zitzler and Lothar Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach, 1998.