Reconocimiento de patrones

Proyecto

Determinar el tipo de fenómeno natural

Profesor:

Dr. Ramón Soto de la Cruz

Alumno:

Raúl Francisco Pérez Rodríguez

Diciembre 2017

Contenido

Introducción

En este documento se verán los resultados de la implementación de técnicas de reconocimiento de patrones en un conjunto de datos que contienen mediciones del clima. Estas mediciones contienen variables tales como temperatura, humedad, viento, precipitación y el evento o fenómeno natural que ocurre en el día. Cuando ocurre un fenómeno natural estos afectan a las variables de tal manera que suben o bajan algunas de estas, entonces ¿Es posible determinar que fenómeno natural ocurrió en el día con solo con algunas de estas variables climáticas?.

Descripción de los datos

Los datos fueron conseguidos de la página web kaggle.

- https://www.kaggle.com/juliansimon/weather_madrid_lemd_1997_2015.csv
- https://www.kaggle.com/grubenm/austin-weather

La unión de ambos conjuntos de datos después de hacer una conversión de medidas, da una cantidad de 8131 registros y 10 variables de información las cuales son:

1. TemperaturaMaxC
2. TemperaturaPromC
3. TemperaturaMinC
4. HumedadMax
5. HumedadProm
6. HumedadMin
7. VientoMaxKMXH
8. VientoPromKMXH
9. PrecipitacionMM
10. Eventos

TemperaturaMaxC: Describe la temperatura máxima en grados centígrados.

TemperaturaPromC: Describe la temperatura promedio en grados centígrados.

TemperaturaMinC: Describe la temperatura mínima en grados centígrados.

HumedadMax: Describe el porcentaje de humedad máxima.

HumedadProm: Describe el porcentaje de humedad promedio.

HumedadMin: Describe el porcentaje de humedad mínima.

VientoMaxKMXH: Describe la velocidad del viento máximo dado por kilómetros por hora.

VientoPromKMXH: Describe la velocidad del viento promedio dado por kilómetros por hora.

PrecipitacionMM: Describe la precipitación está dado por milímetros.

Eventos: Describe el evento o eventos sucedidos en el día, se representa con un valor entero. (Nada = 0, Neblina = 1, Nieve = 2, Lluvia = 3, Neblina-Lluvia = 4, Tornado = 5, Lluvia-Nieve = 6, Lluvia-Tormenta = 7, Neblina-Lluvia-Tormenta = 8, Lluvia-Granizo-Tormenta).

Limpieza de los datos

*El código relacionado a esta parte está contenido en el archivo **limpieza.py** dentro de la carpeta del proyecto.*

Valores faltantes

Después de analizar los datos se determinó la cantidad de valores faltantes para cada variable en el conjunto de datos, los cuáles son siguiente:

```
Contabilidad de valores faltantes por columna
TemperaturaMaxC        2
TemperaturaPromC       3
TemperaturaMinC        2
HumedadMax             4
HumedadProm            4
HumedadMin             4
VientoMaxKMXH          3
VientoPromKMXH         8
PrecipitacionMM      124
```

Una vez teniendo el número de valores faltantes se determinó el porcentaje del mismo.

```
Porcentaje de valores faltantes por columna
TemperaturaMaxC 0.0245972205141
TemperaturaPromC 0.0368958307711
TemperaturaMinC 0.0245972205141
HumedadMax 0.0491944410282
HumedadProm 0.0491944410282
HumedadMin 0.0491944410282
VientoMaxKMXH 0.0368958307711
VientoPromKMXH 0.0983888820563
PrecipitacionMM 1.52502767187
```

Únicamente la variable de **PrecipitacionMM** tiene un grado de impacto manejable, las otras variables tienen un grado de impacto trivial.

Imputación

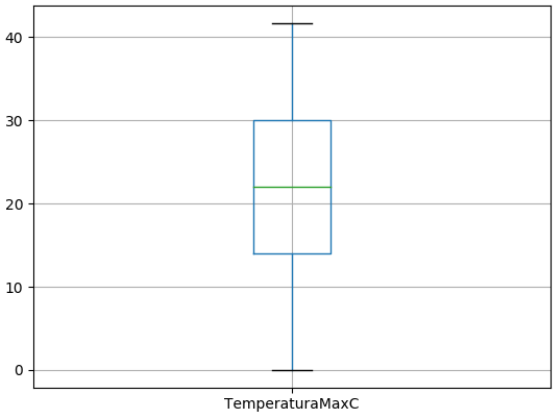
Después se determinó usar imputación con la mediana para el rellenado de los datos faltantes ya que interpolación no tendría sentido ya que los datos ni siquiera tiene una variable de fecha.

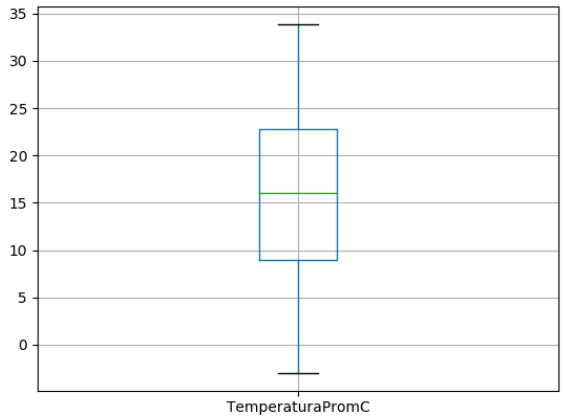
	TemperaturaMaxC	TemperaturaPromC	TemperaturaMinC	HumedadMax
count	8131.000000	8131.000000	8131.000000	8131.000000
mean	22.030446	15.763826	9.753727	82.232813
std	9.045966	8.018298	7.458588	16.831881
min	0.000000	-3.000000	-10.000000	16.000000
25%	14.000000	9.000000	4.000000	71.000000
50%	22.000000	16.000000	9.444444	87.000000
75%	30.000000	22.777778	16.000000	94.000000
max	41.666667	33.888889	28.000000	100.000000

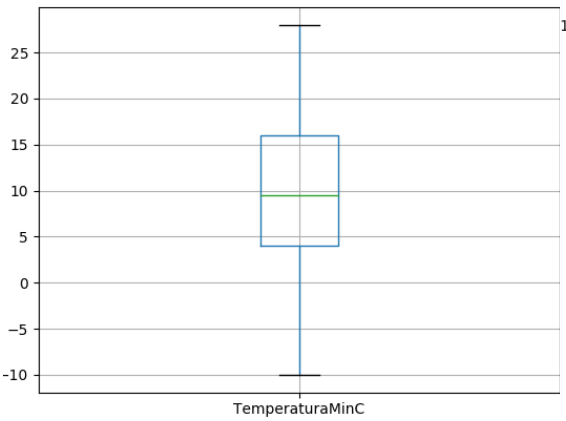
	HumedadProm	HumedadMin	VientoMaxKMXH	VientoPromKMXH	PrecipitacionMM
count	8131.000000	8131.000000	8131.000000	8131.000000	8131.000000
mean	59.382118	36.385930	21.848780	8.996462	0.564503
std	18.964535	19.325448	9.332345	4.879508	4.584764
min	15.000000	4.000000	2.000000	1.609344	0.000000
25%	44.000000	21.000000	14.000000	6.000000	0.000000
50%	61.000000	34.000000	21.000000	8.000000	0.000000
75%	74.000000	49.000000	27.000000	11.000000	0.000000
max	100.000000	100.000000	182.000000	39.000000	132.080000

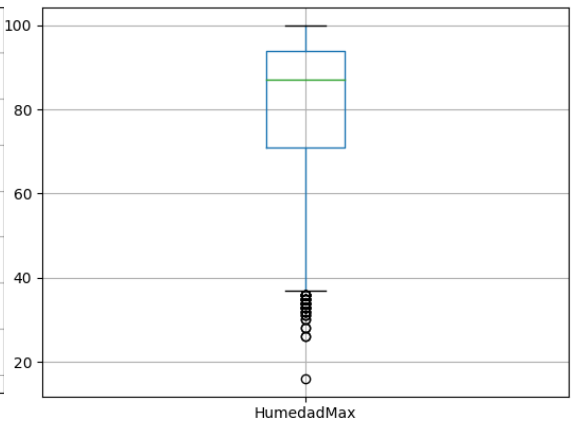
Valores atípicos

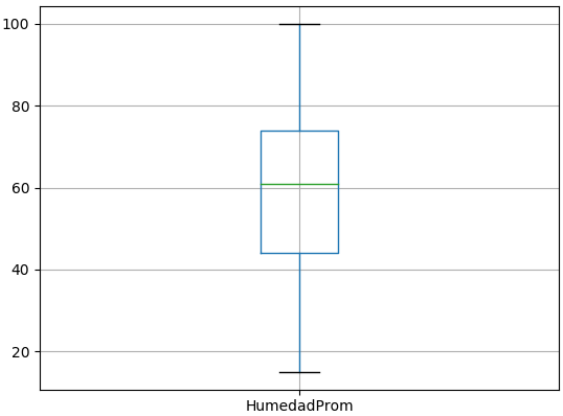
Para el análisis de los valores atípicos se usó diagramas de cajas para cada variable numérica en el conjunto de datos.

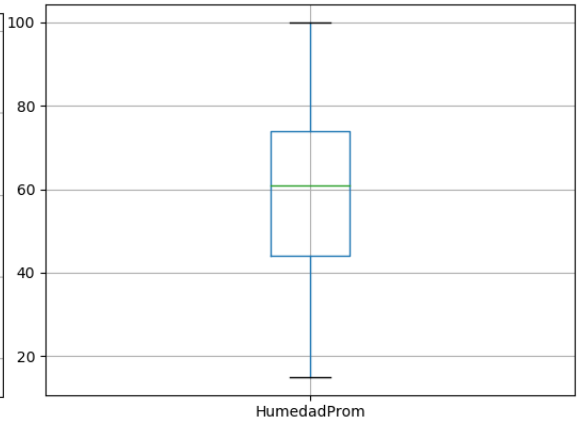


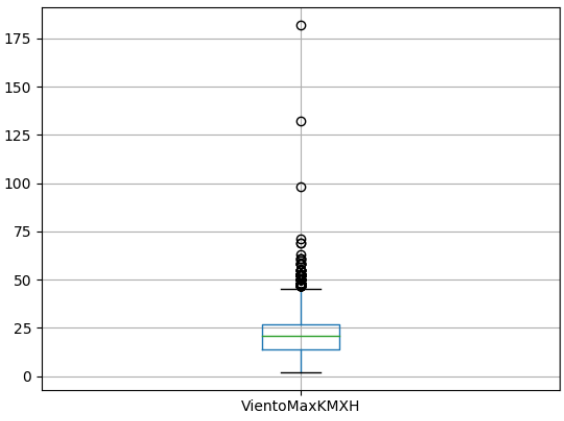


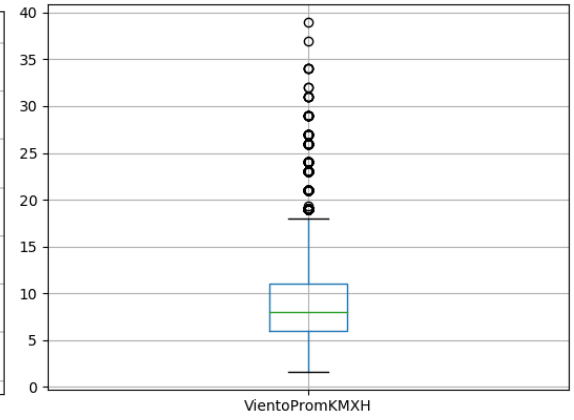

25
20
15
10
5
0
−5
−10
TemperaturaMinC

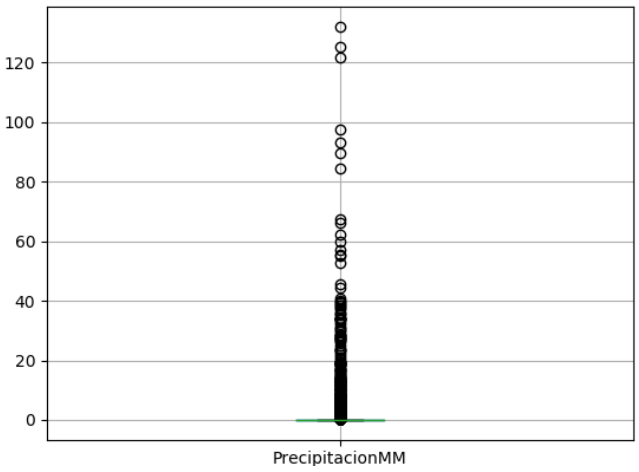

100
80
60
40
20
HumedadMax


100
80
60
40
20
HumedadProm


100
80
60
40
20
HumedadProm


175
150
125
100
75
50
25
0
VientoMaxKMXH


40
35
30
25
20
15
10
5
0
VientoPromKMXH


120
100
80
60
40
20
0
PrecipitacionMM

Los valores atípicos dados por las variables se consideran normales, ya que el conjunto de datos es la unión de mediciones del clima de dos ciudades diferentes y por lo tanto **HumedadMin** contiene valores atípicos. En el caso de las otras variables las de **VientoMaxKMXH**, **VientoPromKMXH** y **PrecipitacionMM**. También son valores atípicos previstos, ya que son las situaciones de las lluvias posiblemente provocadas por tormentas, huracanes, tornados, etc.

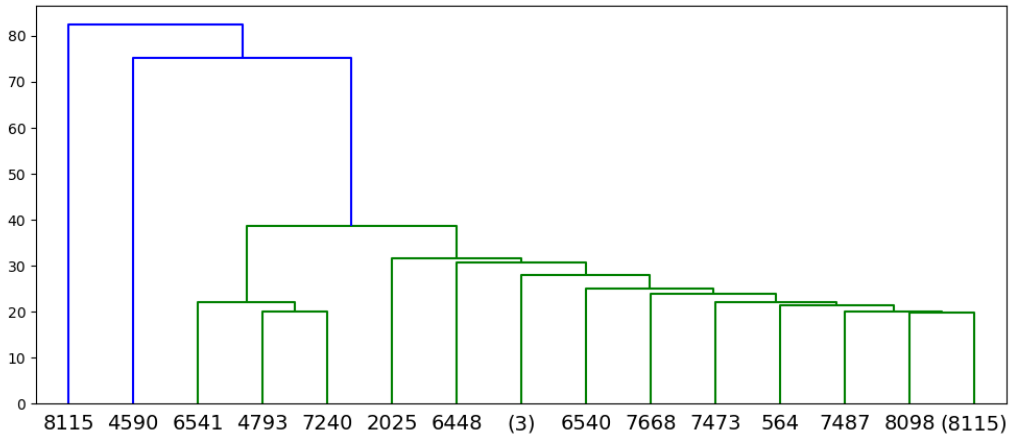
Clustering

Dendrogramas

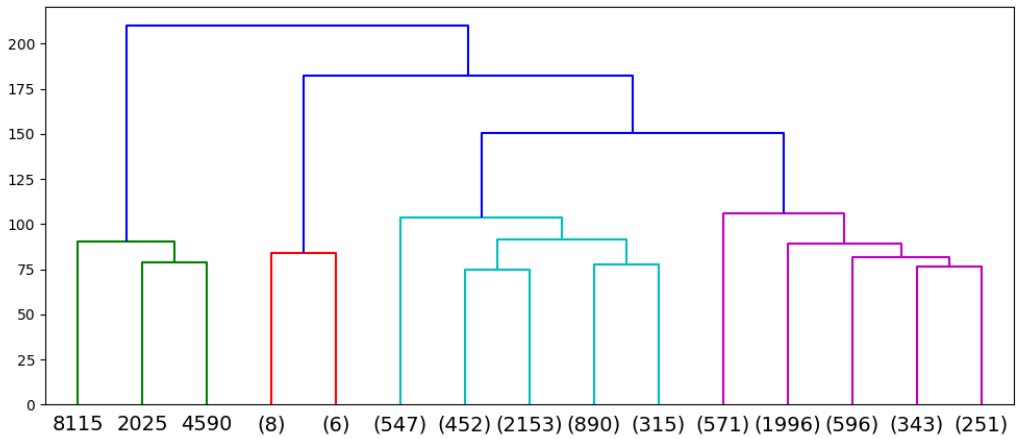
*El código relacionado a esta parte está contenido en el archivo **dendrograma.py** dentro de la carpeta del proyecto.*

Los dendrogramas son útiles para una primera impresión para saber el número de grupos que puede haber en el conjunto de datos. Se usaran los métodos de singler, complete y centroid para visualizar los últimos 15 clusters formados del conjunto de datos.

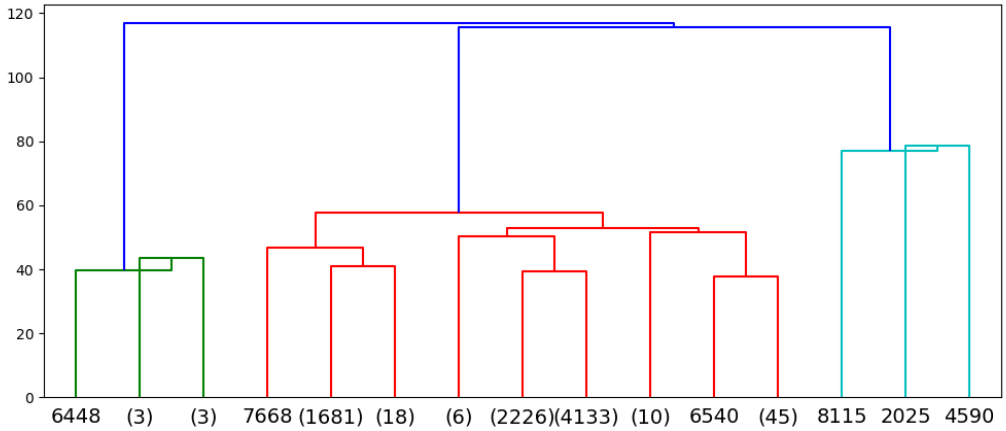
Dendrograma método singler



Dendrograma método complete



Dendrograma método centroid



Usando dendrogramas y analizando los resultados con los tres métodos se pueden observar la formación de tres o cuatro grupos en el conjunto de datos. Como los datos son de un conjunto de mediciones del clima, se puede llegar a concluir que existen uno o dos grupos contiene a gran parte de las mediciones que son posiblemente de los tiempos normales del día, es decir, cuando no ocurre un fenómeno natural de gran intensidad ya se por ejemplo un huracán. Los otros grupos que contienen pocas mediciones pueden ser de mediciones de cuando está ocurriendo un fenómeno natural de gran intensidad.

K-Means

*El código relacionado a esta parte está contenido en el archivo **kmeans.py** dentro de la carpeta del proyecto.*

K-Means es un método de agrupamiento que tiene como objetivo la división de un conjunto de n observaciones en k grupos en el que cada observación pertenece al grupo cuyo valor medio es más cercano. Como utilizando dendrogramas en el conjunto de datos se concluyó que posiblemente existan cuatro diferentes agrupaciones de datos, se usara k-Means con k igual a 3 y 4 con una forma de muestreo aleatoria.

Primer resultado k = 3

```
Cluster 0: 3404 miembros.
[ 21.57  15.06   8.77  86.8   59.72  32.89  22.77   9.36   0.21]

Cluster 1: 2484 miembros.
[ 15.04  10.67   6.54  96.53  80.94  59.61  19.92   8.16   1.55]

Cluster 2: 2243 miembros.
[ 30.47  22.47  14.81  59.47  35.    15.97  22.59   9.37   0.01]
```

Segundo resultado k = 3

```
Cluster 0: 2480 miembros.
[ 15.04  10.67   6.55  96.54  80.96  59.63  19.93   8.16   1.55]

Cluster 1: 2243 miembros.
[ 30.47  22.47  14.81  59.47  35.    15.97  22.59   9.37   0.01]

Cluster 2: 3408 miembros.
[ 21.57  15.05   8.76  86.8   59.73  32.91  22.76   9.36   0.21]
```

Tercer resultado k = 3

```
Cluster 0: 3404 miembros.
[ 21.57  15.06   8.77  86.8   59.72  32.89  22.77   9.36   0.21]

Cluster 1: 2484 miembros.
[ 15.04  10.67   6.54  96.53  80.94  59.61  19.92   8.16   1.55]

Cluster 2: 2243 miembros.
[ 30.47  22.47  14.81  59.47  35.    15.97  22.59   9.37   0.01]
```

Primer resultado k = 4

```
Cluster 0: 1457 miembros.
[ 13.5    9.73   6.23  97.56  85.19  67.06  19.23   8.02   2.21]

Cluster 1: 2280 miembros.
[ 23.65  16.6    9.78  80.62  52.19  26.63  22.93   9.58   0.08]

Cluster 2: 2590 miembros.
[ 18.71  13.02   7.53  92.97  69.67  42.83  21.83   8.81   0.46]

Cluster 3: 1804 miembros.
[ 31.63  23.52  15.76  56.48  32.86  14.7   22.62   9.32   0.01]
```

Segundo resultado k = 4

```
Cluster 0: 2590 miembros.
[ 18.71  13.02   7.53  92.97  69.67  42.83  21.83   8.81   0.46]

Cluster 1: 2280 miembros.
[ 23.65  16.6    9.78  80.62  52.19  26.63  22.93   9.58   0.08]

Cluster 2: 1804 miembros.
[ 31.63  23.52  15.76  56.48  32.86  14.7   22.62   9.32   0.01]

Cluster 3: 1457 miembros.
[ 13.5    9.73   6.23  97.56  85.19  67.06  19.23   8.02   2.21]
```

Tercer resultado k = 4

```
Cluster 0: 1804 miembros.
[ 31.63  23.52  15.76  56.48  32.86  14.7   22.62   9.32   0.01]

Cluster 1: 2590 miembros.
[ 18.71  13.02   7.53  92.97  69.67  42.83  21.83   8.81   0.46]

Cluster 2: 2280 miembros.
[ 23.65  16.6    9.78  80.62  52.19  26.63  22.93   9.58   0.08]

Cluster 3: 1457 miembros.
[ 13.5    9.73   6.23  97.56  85.19  67.06  19.23   8.02   2.21]
```

Se puede observar tanto con k igual a 3 como k igual a 4, que los grupos que se forman son iguales (k = 4) o casi iguales (k = 3) en los 3 resultados que se hicieron de cada uno. Por lo tanto, se puede pensar que es posible que existan tres o cuatro grupos diferentes como se había concluido en los dendrogramas.

ISODATA

*El código relacionado a esta parte está contenido en el archivo **isodata.py** dentro de la carpeta del proyecto.*

El método ISODATA es un algoritmo similar al k-means. Su objetivo es particionar un conjunto de datos en subconjuntos. Sin embargo, a diferencia de k-means, el método ISODATA maneja una serie de heurísticas con tres objetivos que le permiten optimizar el número de clusters:

- Eliminar clusters con poco ejemplares.
- Unir clusters muy cercanos.
- Dividir clusters dispersos

Centroides iniciales resultado uno

```
Centroides inicializados en:
[12.0, 8.0, 4.0, 100.0, 86.0, 67.0, 14.0, 5.0, 0.0]
[30.0, 20.0, 10.0, 62.0, 35.0, 9.0, 13.0, 8.0, 0.0]
[33.0, 23.0, 14.0, 48.0, 31.0, 15.0, 16.0, 6.0, 0.0]
[17.0, 12.0, 8.0, 100.0, 84.0, 67.0, 21.0, 10.0, 0.0]
[22.0, 17.0, 12.0, 72.0, 48.0, 24.0, 21.0, 11.0, 0.0]
[34.444444439999998, 27.222222219999999, 19.444444440000002, 87.0, 63.0, 38.0, 16.093440000000001, 3.2186880000000002, 0.0]
[27.0, 19.0, 12.0, 58.0, 33.0, 12.0, 24.0, 14.0, 0.0]
[7.0, 2.0, -2.0, 100.0, 92.0, 78.0, 11.0, 5.0, 0.0]

Actualizando clusters
El cluster  0  incluye  992 miembros.
El cluster  1  incluye  373 miembros.
El cluster  2  incluye  687 miembros.
El cluster  3  incluye  1382 miembros.
El cluster  4  incluye  2363 miembros.
El cluster  5  incluye  1515 miembros.
El cluster  6  incluye  613 miembros.
El cluster  7  incluye  206 miembros.
```

Centroides finales resultado uno

```
Actualizando clusters
El cluster  0  incluye  1507 miembros.
El cluster  1  incluye  2237 miembros.
El cluster  2  incluye  1785 miembros.
El cluster  3  incluye  2602 miembros.

Los nuevos centroids son:
[13.543537565436628, 9.7287473273211695, 6.1863894418566669, 97.511612475116124, 84.986065029860654, 66.577305905773059, 19.231256928998022,
8.0060289104180526, 2.1509621765096232]
[23.696071126991963, 16.639050315484582, 9.8150300501654009, 80.294143942780508, 51.864103710326333, 26.367009387572644, 22.960612877961612,
9.6099674349575235, 0.068119803308001825]
[31.697541238711491, 23.57348272641849, 15.800871459691875, 56.350140056022411, 32.764145658263303, 14.659943977591036, 22.609047699719888, 9
.3014484616246449, 0.0064145658263305315]
[18.882099239891641, 13.149201468969256, 7.6187121017498081, 92.806302843966179, 69.276710222905464, 42.41775557263643, 21.887349423520462, 8
.8334222905457445, 0.45528132205995331]
```

Centroides iniciales resultado dos

```
Centroides inicializados en:
[11.0, 8.0, 6.0, 100.0, 88.0, 67.0, 19.0, 3.0, 0.0]
[16.0, 6.0, -3.0, 86.0, 63.0, 23.0, 14.0, 6.0, 0.0]
[38.333333330000002, 31.666666670000001, 25.0, 88.0, 60.0, 31.0, 28.968191999999998, 8.0467200000000005, 0.0]
[17.0, 12.0, 8.0, 100.0, 84.0, 67.0, 21.0, 10.0, 0.0]
[10.0, 3.0, -4.0, 80.0, 54.0, 17.0, 21.0, 10.0, 0.0]
[12.0, 9.0, 6.0, 81.0, 67.0, 54.0, 37.0, 21.0, 0.0]
[35.0, 29.444444440000002, 23.88888889, 94.0, 69.0, 44.0, 14.484095999999999, 3.2186880000000002, 0.0]
[16.0, 8.0, 1.0, 93.0, 75.0, 45.0, 14.0, 6.0, 0.0]

Actualizando clusters
El cluster  0  incluye  625 miembros.
El cluster  1  incluye  975 miembros.
El cluster  2  incluye  1923 miembros.
El cluster  3  incluye  677 miembros.
El cluster  4  incluye  1357 miembros.
El cluster  5  incluye  640 miembros.
El cluster  6  incluye  515 miembros.
El cluster  7  incluye  1419 miembros.
```

Centroides finales resultado dos

```
Actualizando clusters
El cluster  0  incluye  2480 miembros.
El cluster  1  incluye  3408 miembros.
El cluster  2  incluye  2243 miembros.

Los nuevos centroids son:
[15.037186379950404, 10.671818996433066, 6.5460573476463688, 96.540322580645167, 80.956451612903223, 59.628629032258061, 19.926927303225874,
8.1617280774193564, 1.5548040322580656]
[21.566966614473895, 15.054414449723298, 8.762878195066607, 86.803697183098592, 59.728579812206576, 32.90698356807512, 22.762201333333458, 9.
3570479248826413, 0.2087570422535209]
[30.466835091880522, 22.471739238125732, 14.805815623919305, 59.468568880962998, 35.001783325902807, 15.973250111457869, 22.58585229068213, 9
.3715238805171612, 0.010083816317432014]
```

Realizando varias ejecuciones del ISODATA con el conjunto de datos, se observó que con una k inicial de 8 y con una forma de muestreo aleatorio, al final de las iteraciones se formaban grupos de 3 y 4 clusters. Mismos que se observaban en los demás técnicas de clustering.

Clasificación

K-Vecinos Próximos

*El código relacionado a esta parte está contenido en el archivo **kvecinos.py** dentro de la carpeta del proyecto.*

Debido a que el conjunto de datos contiene varios eventos dependiendo de lo sucedido en el día, se cambiara sus valores a sucedió o no sucedió un evento en el día, para que de esa manera haya dos clases.

El método de k-vecinos próximos es un método de aprendizaje vago (lazy learning). Esto significa que el aprendizaje no conduce a una generalización: no existe una fase de entrenamiento (o es muy breve) y en su lugar el método mantiene todos los datos disponibles y los emplea para realizar la clasificación.

Para la ejecución del método se tomaron 45 debido a que correspondía con la cantidad sugerida: $\frac{1}{2}\sqrt{n}$ donde n es la cantidad de datos. Se tomó la última medición del conjunto de datos, correspondiente a un dato del tipo de evento con valor 1 (sucedió) y la primera medición que corresponde a un evento de no sucedió.

Última medición

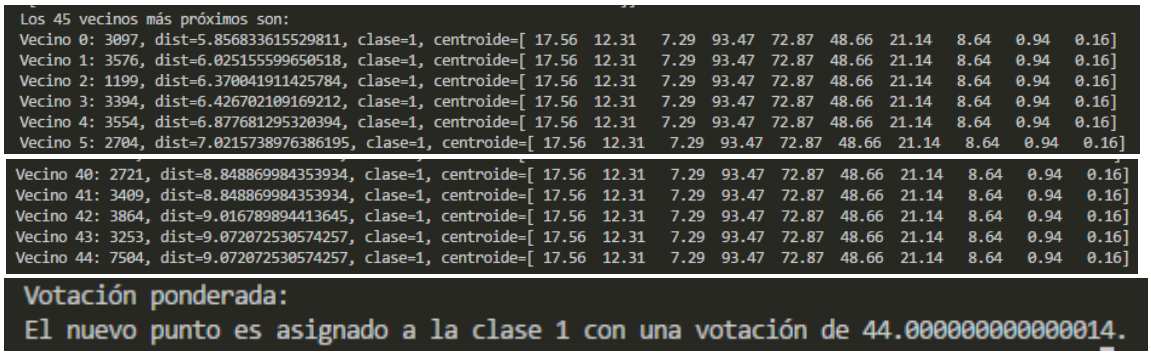
```
Datos de entrenaiento:
[[ 15.    10.     5.   ...,  11.     3.05   0.  ]
 [ 19.    14.    10.   ...,  26.     2.03   0.  ]
 [  4.44   0.56  -3.33 ...,   8.05   1.27   0.  ]
 ...,
 [ 31.67  26.67  21.67 ...,   8.05  12.45   1.  ]
 [ 33.89  28.89  23.33 ...,   6.44   6.1    1.  ]
 [ 21.11  11.11   1.11 ...,   9.66   3.81   1.  ]]

Dato de prueba:
[  32.78   25.56   18.33  100.     79.     57.     20.92    8.05    0.51
    1.  ]

Prototipos de clase (centroides):
 [[ 28.26  20.57  13.18  66.58  40.6   19.3   22.82   9.5    0.03   0.01]
 [ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]]
```

```
Los 45 vecinos más próximos son:
Vecino 0: 7074, dist=3.2235480435923933, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
Vecino 1: 1472, dist=5.792387752326234, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
Vecino 2: 7865, dist=5.96264490257908, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
Vecino 3: 8050, dist=6.1236052594659, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
Vecino 4: 1663, dist=6.296965321007993, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
Vecino 5: 972, dist=6.960631142958684, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
```

```
Vecino 40: 1076, dist=10.768491735449604, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
Vecino 41: 1599, dist=10.924392484205008, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
Vecino 42: 7864, dist=10.969157753588611, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
Vecino 43: 1370, dist=11.045804495944116, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
Vecino 44: 942, dist=11.133694948184315, clase=1, centroide=[ 17.55  12.31   7.29  93.47  72.87  48.66  21.14   8.64   0.94   0.16]
```

```
Votación ponderada:
El nuevo punto es asignado a la clase 1 con una votación de 44.0.
```

Primera medición



Se puede observar que los dos grupos formados por el método, se encuentran mesclados mediciones de eventos de sucedió o no sucedió. Ya que tanto la primera como la segunda medición caen en el mismo centroide.

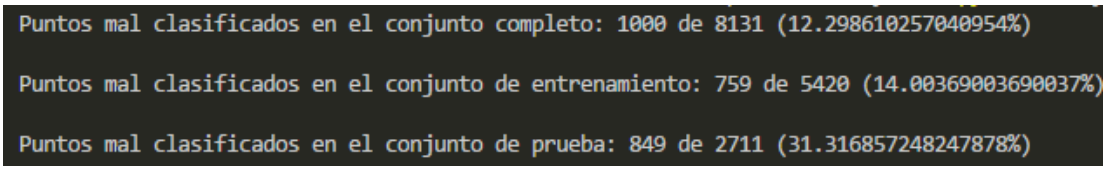
Teorema de Bayes

*El código relacionado a esta parte está contenido en el archivo **bayes.py** dentro de la carpeta del proyecto.*

En teoría de la probabilidad y minería de datos, un clasificador Bayesiano ingenuo es un clasificador probabilístico fundamentado en el teorema de y algunas hipótesis simplificadoras adicionales. Es a causa de estas simplificaciones, que se suelen resumir en la hipótesis de independencia entre las variables predictoras, que recibe el apelativo de ingenuo.

Para el clasificador de Bayes se dividieron los datos en datos de entrenamiento y datos de prueba para poder hacer las pruebas con la distribución Gaussiana.

Modelo gaussiano



Con la distribución Gaussiana se obtuvo un error de 12.29% para el conjunto de datos completos, es un error razonable teniendo en cuanta que hay 9 tipos de clases en el conjunto de datos.

Árbol de decisión

El código relacionado a esta parte está contenido en el archivo ***arbol.py*** *dentro de la carpeta del proyecto.*

Los árboles de decisión son uno de los métodos de clasificación más antiguos y robustos y, por lo tanto, de las más utilizados. Los árboles de decisión son un método no paramétrico de aprendizaje supervisado utilizado para clasificación y regresión. El objetivo es generar un conjunto de reglas de decisión simples a partir de las características de los datos de entrenamiento. En un árbol de decisión, los datos de entrenamiento son divididos en cada nodo utilizando reglas de decisión para obtener grupos de elementos similares. Este agrupamiento equivale a una operación de decisión tipo *switch*(caso). Cada nodo interno en el árbol representa una evaluación sobre un atributo, los arcos representan los posibles resultados de la evaluación y las hojas representan una decisión. Las trayectorias, desde el nodo raíz a las hojas son las reglas de decisión.

El árbol de decisión generado es tan grande que no se vería en el documento, si se desea ver se encuentra en la carpeta de ***image*** *con el nombre de* ***arbol.png.***

Primera ejecución

```
Clases de la muestra de prueba:  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 3, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 4, 0, 0, 0, 1, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 1, 0, 0, 3, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 7, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0,
 3, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 6, 0, 8, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 3, 8, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 3, 0, 0, 0, 3, 0, 0, 0, 0, 3, 0, 1, 0, 0, 2, 0, 3, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 7, 0, 6, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3,
 0, 3, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 1, 0, 8, 0, 3, 0, 0, 0,
 0, 0, 0, 0, 0, 8, 0, 3, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
Objetivos:  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 1, 7,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 3, 3, 0, 0,
 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 3, 3, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 7, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0,
 0, 1, 1, 0, 0, 0, 0, 6, 0, 8, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 3, 8, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 3,
 0, 0, 0, 0, 3, 0, 1, 0, 0, 2, 0, 3, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 7, 0, 6, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 0, 3, 4, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 1, 0, 8, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0,
 3, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
Resultados:  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 3
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 0, 0
, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 4, 4, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 7, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 3, 0, 0, 0, 0, 0, 0, 1, 0, 4, 0, 0, 0, 0, 0, 0
, 0, 1, 0, 0, 0, 0, 0, 3, 0, 7, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 4, 3, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 7
, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0
, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 9, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 3, 3, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 3, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 4, 0, 0, 0, 0, 0, 1, 0, 0, 0, 3, 0, 0, 0, 0, 1, 0, 8, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0
, 3, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
Puntos mal clasificados en el conjunto de prueba: 66 de 814 (8.108108108108109%)
```

Segunda ejecución

```
Resultados:  [1, 0, 0, 0, 7, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1
, 7, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 1, 0
, 0, 0, 0, 0, 4, 3, 0, 3, 0, 7, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 3, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 8, 0, 0, 0, 0, 0, 7, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 1, 0, 0, 7, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 3, 3, 0, 0, 0, 0, 0, 0, 3, 3, 0, 0, 0, 0, 0, 0, 7, 3, 3, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 1, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 1, 0, 4, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 7, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 2, 7, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 3, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0
, 8, 0, 0, 0, 0, 0, 1, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 1, 0, 3, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0
, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4, 0
, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 3, 1, 0]
Puntos mal clasificados en el conjunto de prueba: 54 de 814 (6.6339066339066335%)
```

Tercera ejecución

```
Resultados:  [8, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0
, 0, 0, 0, 0, 0, 3, 0, 7, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 4, 0, 0, 0
, 0, 0, 0, 3, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 3, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 3, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 7, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0
, 7, 0, 1, 0, 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 7, 0, 0, 0, 0, 0, 0, 3, 0
, 9, 0, 0, 0, 6, 0, 0, 0, 1, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 7, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 3, 0, 0, 0, 0, 0, 7, 0, 1, 0, 0, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 1, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 1, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 7, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 6, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 1, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 3, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Puntos mal clasificados en el conjunto de prueba: 63 de 814 (7.73955773955774%)
```

Los resultados después de tres ejecuciones del árbol de decisión fueron distintos pero todos dieron mejor resultado que el Bayes que fue de 12.29%. El mejor resultado del árbol de decisión fue de 6.63% mucho mejor que el de Bayes.

Máquinas de vectores de soporte

*El código relacionado a esta parte está contenido en el archivo **vectores.py** dentro de la carpeta del proyecto.*

Las máquinas de vectores de soporte (SVM) se basan en la idea de definir planos de decisión que separen a los objetos pertenecientes a diferentes clases. El objetivo de clasificación es encontrar el hiperplano que separe mejor las regiones en el espacio de características ocupadas por cada una de las clases.

SVM lineal

A continuación, se presentan los resultados de la aplicación de SVM lineal al conjunto de datos utilizando diferentes valores de penalización.

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	0.1	7.57%	6.51%
2	0.1	10.90%	12.77%
3	0.1	9.81%	10.68%
4	0.1	8.41%	7.98%
5	0.1	10.08%	8.47%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	0.5	8.40%	7.86%
2	0.5	10.91%	12.89%
3	0.5	8.10%	8.10%
4	0.5	8.20%	8.10%
5	0.5	9.85%	9.70%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	1	11.35%	10.07%
2	1	6.98%	8.35%
3	1	7.47%	7.73%
4	1	7.83%	8.23%
5	1	9.15%	9.09%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	20	18.62%	18.79%
2	20	7.99%	9.45%
3	20	7.73%	8.10%
4	20	7.38%	7.24%
5	20	7.58%	7.61%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	100	7.92%	7.49%
2	100	8.59%	9.95%
3	100	7.61%	7.61%
4	100	7.83%	7.86%
5	100	8.86%	8.59%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	500	8.41%	7.49%
2	500	14.78%	16.95%
3	500	9.17%	10.44%
4	500	8.52%	8.10%
5	500	8.26%	7.61%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	1000	7.50%	7.12%
2	1000	6.97%	8.47%
3	1000	9.74%	9.70%
4	1000	13.20%	11.17%
5	1000	7.50%	7.24%

A partir de estos resultados podemos observar que, en este caso, no hay un valor de C claramente superior, el más constante fue con C=100, pero este no obtuvo el mejor porcentaje. El mejor porcentaje obtenido para el conjunto de prueba fue de un 6.51%, con un valor de C=0.1, el peor porcentaje obtenido fue un 18.79% obtenido con un valor de C=20.

SVM kernel RBF

A continuación, mostramos los resultados de varias corridas utilizando una máquina de vectores de soporte con función kernel RBF (función de base radial)

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	0.1	0.01%	10.93%
2	0.1	0.01%	9.95%
3	0.1	0.01%	11.54%
4	0.1	0.01%	12.77%
5	0.1	0.01%	10.81%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	0.5	0.01%	10.93%
2	0.5	0.01%	9.95%
3	0.5	0.01%	11.54%
4	0.5	0.01%	12.77%
5	0.5	0.01%	10.81%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	1	0.01%	10.68%
2	1	0.01%	9.95%
3	1	0.01%	11.42%
4	1	0.01%	12.65%
5	1	0.01%	10.68%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	20	0.01%	10.93%
2	20	0.01%	10.19%
3	20	0.01%	12.03%
4	20	0.01%	12.40%
5	20	0.01%	10.81%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	100	0.01%	10.93%
2	100	0.01%	10.19%
3	100	0.01%	11.91%
4	100	0.01%	12.40%
5	100	0.01%	10.81%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	500	0.01%	10.93%
2	500	0.01%	10.19%
3	500	0.01%	11.91%
4	500	0.01%	12.40%
5	500	0.01%	10.81%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	1000	0.01%	10.93%
2	1000	0.01%	10.19%
3	1000	0.01%	11.91%
4	1000	0.01%	12.40%
5	1000	0.01%	10.81%

A partir de estos resultados podemos observar que en cuanto a los datos de entrenamiento todos tuvieron el mismo resultado, pero con los de prueba el valor de C con mejores resultados fue C=1, pero cabe recalcar que independientemente del valor de C los porcentajes son muy similares. El mejor resultado obtenido fue 9.95% con valor de C= 0.1, 0.5, 1.

SVM kernel RBF (Gamma)

El otro parámetro de la función kernel RBF es γ. A continuación, mostramos variaciones del valor de γ en ejercicios de clasificación sobre los datos, utilizando C=1.

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	0.1	2.30%	9.33%
2	0.1	2.14%	11.42%
3	0.1	2.26%	9.45%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	0.5	0.17%	9.58%
2	0.5	0.17%	11.42%
3	0.5	0.20%	9.45%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	1	0.04%	9.58%
2	1	0.04%	11.42%
3	1	0.05%	9.45%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	20	0.01%	9.58%
2	20	0.01%	11.42%
3	20	0.01%	9.45%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	100	0.00%	9.58%
2	100	0.00%	11.42%
3	100	0.00%	9.45%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	500	0.00%	9.58%
2	500	0.00%	11.42%
3	500	0.00%	9.45%

Corrida	Valor de C	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	1000	0.00%	9.58%
2	1000	0.00%	11.42%
3	1000	0.00%	9.45%

A partir de estos resultados podemos observar que en cuanto a los datos de entrenamiento con una C mayor a 100 no se obtiene un porcentaje de error, en cuanto a los de prueba los resultados fueron iguales o casi iguales independientemente del valor de C y del porcentaje de error en el conjunto de entrenamiento. El mejor resultado fue obtenido con un valor de C=0.1, con un porcentaje de 9.33%.

SVM Sigmoide con C=1

Corrida	Puntos de entrenamiento mal clasificados	Puntos de prueba mal clasificados
1	10.55%	11.79%
2	10.82%	9.33%
3	10.56%	11.67%

Conclusión

Con el SVM Lineal se obtuvieron los mejores resultados y el mejor porcentaje con un 6.51% con los datos de prueba. En cuanto al conjunto de entrenamiento con un SVM Kernel, se obtuvieron porcentajes de error de 0.00%, pero no obtuvieron el mejor porcentaje en el conjunto de prueba.

Redes Neuronales

*El código relacionado a esta parte está contenido en el archivo **redes.py** dentro de la carpeta del proyecto.*

Perceptrón

El perceptrón es un modelo de neurona obtenido al combinar el modelo de la neurona de McCulloch-Pitts con una extensión de la regla de Hebb. La regla de entrenamiento consta de los siguientes pasos:

1. Inicializar los pesos con valores aleatorios pequeños.
2. Para cada vector de entrenamiento Xj con valor de clase Yj (el valor de salida esperado):
   A. Calcular el valor de salida Y*j de la función de activación
   B. Actualizar los pesos de acuerdo a la regla de Hebb modificada:

$$\Delta \mathbf{w} = \eta(y_j - y_j^*)\mathbf{x}_j$$

Como en el conjunto de datos existen 9 tipos de clases, se usara el conjunto de datos usado para el k vecinos próximos, donde solo existen dos clases: sucedió un evento o no.

A continuación se presentan los resultados del perceptrón para la mitad del conjunto de datos.

Corrida	Resultado
1	368 vectores mal clasificados de 4064 (9.06%)
2	363 vectores mal clasificados de 4064 (8.93%)
3	2030 vectores mal clasificados de 4064 (49.95%)
4	367 vectores mal clasificados de 4064 (9.03%)
5	357 vectores mal clasificados de 4064 (8.78%)

Como se puede observarse, el porcentaje de error en las corridas presentadas, va desde 8.78% hasta 49.95%. Esta variabilidad se debe a que el perceptrón solo toma en cuenta si la clasificación fue correcta o no, pero no se toman en cuenta los aciertos que estuvieron cerca de fallar y se magnifican los errores cometidos por un margen pequeño o corrige los pesos cada que encuentra un vector que se clasifica erróneamente, lo que lo hace susceptible a valores atípicos.

A continuación se realizaron pruebas bajo tres rondas de entrenamiento. Los resultados fueron los siguientes.

Corrida 1:

- **Ronda 1:**
  - 383 vectores mal clasificados de 4064 (9.42%)
- **Segunda ronda:**
  - 612 vectores de entrenamiento mal clasificados de 4064 (15.06%)
- **Tercera ronda con todos los datos:**
  - 480 vectores de entrenamiento mal clasificados de 8128 (5.91%)
  - 1671 vectores mal clasificados de 4064 (41.12%)

Corrida 2:

- **Ronda 1:**
  - 723 vectores mal clasificados de 4064 (17.79%)
- **Segunda ronda:**
  - 639 vectores de entrenamiento mal clasificados de 4064 (15.72%)
- **Tercera ronda con todos los datos:**
  - 480 vectores de entrenamiento mal clasificados de 8128 (5.91%)
  - 1691 vectores mal clasificados de 4064 (41.61%)

Corrida 3:

- **Ronda 1:**
  - 321 vectores mal clasificados de 4064 (7.90%)
- **Segunda ronda:**
  - 597 vectores de entrenamiento mal clasificados de 4064 (14.69%)
- **Tercera ronda con todos los datos:**
  - 480 vectores de entrenamiento mal clasificados de 8128 (5.91%)
  - 1698 vectores mal clasificados de 4064 (41.78%)

Al observar los resultados, el mejor resultado de las tres corridas con todos los datos fue de 41.12%. Este resultado es muy inferior a otros métodos ya vistos.

Adaline y la regla delta

A continuación se presentan los resultados de las ejecuciones de la implementación de este.

Corrida	**Iteraciones**	**Resultado**
1	5	7261 vectores mal clasificados de 8078 (89.89%)
2	10	817 vectores mal clasificados de 8078 (10.11%)
3	15	817 vectores mal clasificados de 8078 (10.11%)

Como se observa, a partir de las 10 iteraciones ya no hay una mejora en el porcentaje de aciertos. El mejor resultado fue un 10.11%, mucho mejor que el perceptrón.

Redes *feed-forward* y el algoritmo *backpropagation*

A continuación se presentan los resultados de las ejecuciones de la implementación de este con números diferentes de neuronas en las capas ocultas.

Capas ocultas = (2, 2, 1)

Corrida	**Resultado**
1	138 vectores mal clasificados de 4064 (3.39%)
2	429 vectores mal clasificados de 4064 (10.55%)
3	377 vectores mal clasificados de 4064 (9.27%)
4	129 vectores mal clasificados de 4064 (3.17%)
5	410 vectores mal clasificados de 4064 (10.088%)

Capas ocultas = (3, 3, 1)

Corrida	Resultado
1	140 vectores mal clasificados de 4064 (3.44%)
2	136 vectores mal clasificados de 4064 (3.34%)
3	141 vectores mal clasificados de 4064 (3.46%)
4	135 vectores mal clasificados de 4064 (3.32%)
5	129 vectores mal clasificados de 4064 (3.17%)

Capas ocultas = (4, 4, 2)

Corrida	Resultado
1	406 vectores mal clasificados de 4064 (9.99%)
2	112 vectores mal clasificados de 4064 (2.75%)
3	415 vectores mal clasificados de 4064 (10.21%)
4	127 vectores mal clasificados de 4064 (3.125%)
5	416 vectores mal clasificados de 4064 (10.23%)

Al observar los resultados de los tres tipos de capas ocultas, la capa con una estructura (4, 4, 2) fue la que tuvo el mejor resultado al obtener un 2.75%, pero sus resultados fueron vareados. El que tuvo menos diferencia en sus resultados fue una capa con (3, 3, 1). Los resultados obtenidos con redes feed-forward han sido los más bajos en comparación de las otras técnicas.

Conclusión

El conjunto de datos con el que se trabajó venían con valores faltantes, los cuales se trataron en la parte de limpieza de datos. En las técnicas de Clustering se concluyó que existen tres o cuatro grupos de datos en el conjunto, pero en realidad el conjunto de datos tiene nueve clases, por lo tanto, no acertaron en el número de las clases.

Para poder trabajar con las técnicas de clasificación, los tipos de clases del conjunto de datos se tuvieron que cambiar de tipo cadena a tipo entero. En algunas técnicas tales como K-Vecinos o el perceptrón se tuvo que hacer otro conjunto de datos donde las nuevas clases fueron sucedió o no sucedió un evento (0 y 1). La mejor técnica de clasificación fue la de una red feed-fordward donde se obtuvo un porcentaje de error de 2.75% lo cual es bastante bueno al ver el número de clases que hay en el conjunto de datos.

En conclusión se puede decir que es relativamente posible determinar el tipo de evento que sucedió en el día, con las variables climáticas tratadas en el proyecto.

Referencias

Las definiciones de las técnicas tratadas en el proyecto se encuentran en el repositorio del Dr. Ramón Soto de la Cruz el cual es el siguiente:

- https://github.com/rsotoc/pattern-recognition

Los conjuntos de datos que se usaron en el proyecto se pueden conseguir en la página web Kaggle:

- https://www.kaggle.com/juliansimon/weather_madrid_lemd_1997_2015.csv
- https://www.kaggle.com/grubenm/austin-weather