INF-354-1P-P6

December 11, 2023

1 Pimer Parcial de Inteligencia Artificial

1.0.1 Nombre: Steve Brandom Nina Huacani

1.0.2 Pregunta 6. El dataset elegido en PYTHON, realice tres tecnicas de preprocesamiento. Explique la razón de aplicar estas técnicas.

Para la visualizacion del dataset utilizaremos las librerias de pandas

[1]:
```
#importamos la libreria pandas
import pandas as pd
#importamos la libreria numpy
import numpy as np
#importamos el modulo drive
#from google.colab import drive
#montamos la carpeta de drive que contiene el dataset
#drive.mount("/content/drive")
#asignamos la ruta del dataset
#archivo="/content/drive/MyDrive/data/wineQuality.csv"
#abrimos el archivo
archivo = "wine_quality.csv"
df = pd.read_csv(archivo)
#mostramos el dataset en un dataframe
df
```

[1]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	\
0	white	7.0	0.270	0.36	20.7	
1	white	6.3	0.300	0.34	1.6	
2	white	8.1	0.280	0.40	6.9	
3	white	7.2	0.230	0.32	8.5	
4	white	7.2	0.230	0.32	8.5	
…	…	…	…	…	…	
6493	red	5.9	0.550	0.10	2.2	
6494	red	6.3	0.510	0.13	2.3	
6495	red	5.9	0.645	0.12	2.0	
6496	red	6.0	0.310	0.47	3.6	
6497	red	6.0	0.310	0.47	3.6	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	\

```
0         0.045                 45.0                 170.0  1.00100  3.00
1         0.049                 14.0                 132.0  0.99400  3.30
2         0.050                 30.0                  97.0  0.99510  3.26
3         0.058                 47.0                 186.0  0.99560  3.19
4         0.058                 47.0                 186.0  0.99560  3.19
…         …                     …                    …      …        …
6493      0.062                 39.0                  51.0  0.99512  3.52
6494      0.076                 29.0                  40.0  0.99574  3.42
6495      0.075                 32.0                  44.0  0.99547  3.57
6496      0.067                 18.0                  42.0  0.99549  3.39
6497      0.067                 18.0                  42.0  0.99549  3.39

      sulphates  alcohol  quality
0          0.45      8.8        6
1          0.49      9.5        6
2          0.44     10.1        6
3          0.40      9.9        6
4          0.40      9.9        6
…          …       …        …
6493        NaN     11.2        6
6494       0.75     11.0        6
6495       0.71     10.2        5
6496       0.66     11.0        6
6497       0.66     11.0        6

[6498 rows x 13 columns]
```

1.1 Primera tecnica de preprocesamiento

1.1.1 Missing Values

La imputacion de valores faltantes es una tecnica de preprocesamiento muy importante ya que los algoritmos de aprendiza automatico no pueden manejar valores faltantes.

Habitualmente se asocian tres tipos de problemas con los valores faltantes:

Perdida de eficiencia

Complicaciones en el manejo y analisis de los datos

Sesgo resultante de las diferencias entre los datos faltantes y los que estan completos

Existe una amplia variedad de metodos de imputacion como por ejemplo: la sustitucion por medias, medianas o modas, el vecindario K mas cercano, procedimientos de maxima verosimilitud entre otras. Algunos de los beneficios de utilizar la imputacion de valores faltantes son las siguientes.

la preservación de los datos: Ya que si se eliminara una tupla que contiene el valor faltante se pierde la información de las otras caracteristicas de la fila o columna, la imputacion de valores faltantes nos permite mantener la maxima cantidad de informacion posible.

Consistencia de los datos: Ayuda a mantener la consistencia de un conjunto de datos ya que es importante mantener el mismo numero de muestras en todas las categorias para que el analisis sea

el mas optimo posible.

Scikit-Learn nos ofrece la clase SimpleImputer que proporciona estrategias basicas para la imputacion de valores faltantes, como ser con una constante provista o con metodos estadisticos como ser la media, la mediana y la moda

Procederemos a realizar la imputacion de las columnas de nuestro dataset con la estrategia "mean" de scikit-learn

[2]:
```
#importamos el modulo preprocessing de sklearn
from sklearn import preprocessing
#importamos el modulo simpleimputer de sklearn
from sklearn.impute import SimpleImputer

#extraemos en x1 los registros de las columnas
x1 = df[["fixed acidity","volatile acidity","citric acid","residual␣
 ↪sugar","chlorides","free sulfur dioxide","total sulfur␣
 ↪dioxide","density","pH","sulphates","alcohol"]]
#realizamos una instancia de la clase SimpleImputer definiendo la estragegia de␣
 ↪imputacion "mean" (media)
imputter = SimpleImputer(strategy="mean")
#en el array x2 ajustaremos y transformaremos el array con datos faltantes
x2 = imputter.fit_transform(x1)
#ocnvertimos a dataframe el array anterior
df[["fixed acidity","volatile acidity","citric acid","residual␣
 ↪sugar","chlorides","free sulfur dioxide","total sulfur␣
 ↪dioxide","density","pH","sulphates","alcohol"]] = x2
#mostramos el dataframe resultante
df
```

[2]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	\
0	white	7.0	0.270	0.36	20.7	
1	white	6.3	0.300	0.34	1.6	
2	white	8.1	0.280	0.40	6.9	
3	white	7.2	0.230	0.32	8.5	
4	white	7.2	0.230	0.32	8.5	
…	…	…	…	…	…	
6493	red	5.9	0.550	0.10	2.2	
6494	red	6.3	0.510	0.13	2.3	
6495	red	5.9	0.645	0.12	2.0	
6496	red	6.0	0.310	0.47	3.6	
6497	red	6.0	0.310	0.47	3.6	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	\
0	0.045	45.0	170.0	1.00100	3.00	
1	0.049	14.0	132.0	0.99400	3.30	
2	0.050	30.0	97.0	0.99510	3.26	
3	0.058	47.0	186.0	0.99560	3.19	

```
4          0.058                  47.0                  186.0  0.99560  3.19
…          …                    …                     …      …        …
6493       0.062                  39.0                   51.0  0.99512  3.52
6494       0.076                  29.0                   40.0  0.99574  3.42
6495       0.075                  32.0                   44.0  0.99547  3.57
6496       0.067                  18.0                   42.0  0.99549  3.39
6497       0.067                  18.0                   42.0  0.99549  3.39

      sulphates  alcohol  quality
0      0.450000      8.8        6
1      0.490000      9.5        6
2      0.440000     10.1        6
3      0.400000      9.9        6
4      0.400000      9.9        6
…           …      …       …
6493   0.531235     11.2        6
6494   0.750000     11.0        6
6495   0.710000     10.2        5
6496   0.660000     11.0        6
6497   0.660000     11.0        6

[6498 rows x 13 columns]
```

1.2 Segunda tecnica de preprocesamiento

1.2.1 Remove Duplicates

Esta técnica de preprocesamiento se aplica para poder eliminar instancias duplicadas. Esto es especialmente útil cuando se trabaja con conjuntos de datos grandes y se quiere asegurar de que cada entrada sea única. La presencia de duplicados puede afectar negativamente a los análisis y modelos de aprendizaje automatico, ya que podría introducir sesgos o distorsiones en los resultados. Algunas razones para utilizarlo: Datos de baja calidad: Los conjuntos de datos pueden contener registros duplicados debido a errores de entrada o problemas en la recopilación de datos. Eliminar duplicados ayuda a limpiar el conjunto de datos y mejorar la calidad de los datos. Modelos de aprendizaje automatico: Algunos algoritmos de aprendizaje automatico pueden ser sensibles a la presencia de duplicados. La existencia de registros duplicados puede conducir a un sobreajuste y afectar la capacidad del modelo para generalizar correctamente a nuevos datos.

[4]:
```
#MOstramos el dataset original y verificaremos con cuantas instancias cuenta
df
#plicaremos la tecnica de preprocesaieto con la libreria pandas
df = df.drop_duplicates()
#ahora verificaremos con cuantas instancias cuenta nuestro dataset
df
```

[4]:
```
       type  fixed acidity  volatile acidity  citric acid  residual sugar  \
0     white            7.0             0.270         0.36            20.7
1     white            6.3             0.300         0.34             1.6
```

```
2     white            8.1             0.280         0.40             6.9
3     white            7.2             0.230         0.32             8.5
6     white            6.2             0.320         0.16             7.0
…     …              …               …             …              …
6491    red            6.8             0.620         0.08             1.9
6492    red            6.2             0.600         0.08             2.0
6493    red            5.9             0.550         0.10             2.2
6495    red            5.9             0.645         0.12             2.0
6496    red            6.0             0.310         0.47             3.6

      chlorides  free sulfur dioxide  total sulfur dioxide  density    pH  \
0         0.045                 45.0                 170.0  1.00100  3.00
1         0.049                 14.0                 132.0  0.99400  3.30
2         0.050                 30.0                  97.0  0.99510  3.26
3         0.058                 47.0                 186.0  0.99560  3.19
6         0.045                 30.0                 136.0  0.99490  3.18
…         …                   …                    …      …      …
6491      0.068                 28.0                  38.0  0.99651  3.42
6492      0.090                 32.0                  44.0  0.99490  3.45
6493      0.062                 39.0                  51.0  0.99512  3.52
6495      0.075                 32.0                  44.0  0.99547  3.57
6496      0.067                 18.0                  42.0  0.99549  3.39

      sulphates  alcohol  quality
0      0.450000      8.8        6
1      0.490000      9.5        6
2      0.440000     10.1        6
3      0.400000      9.9        6
6      0.470000      9.6        6
…         …       …       …
6491   0.820000      9.5        6
6492   0.580000     10.5        5
6493   0.531235     11.2        6
6495   0.710000     10.2        5
6496   0.660000     11.0        6

[5329 rows x 13 columns]
```

1.3 Tercera tecnica de preprocesamiento

1.3.1 Estandarizacion

La estandarizacion es una tecnica de preprocesamiento de datos que se utiliza para transformar las caracteristicas de un conjunto de datos de forma que tengan una media en 0 y una desviacion estandar de 1.

Algunas de las desventajas de trabajar con caracteristicas no estandarizadas son las siguientes:

Mayor sensibilidad a los valores atipicoos, ya que pueden tener mayor impacto si es que la carac-

teristica no esta estandarizada.

Problemas de interpretacion, puede ser dificil interpretar la importancia de una caracteristica si no estan en la misma escala.

Dificultades en la visualizacion de los datos, Al graficar datos no estandarizados las diferencias en las escalas pueden dar origen a datos dificiles de interpretar

Algunos de los beneficios de estandarizar los datos son:

Los algoritmos basados en la distancia como el k-NN reducen la desproporcionalidad que puedan existir en las caracteristicas

Convergencia rapida, algunos algoritmos de optimizacion convergen mucho mas rapido cuando las caracteristicas estan en la misma escala, como en los algoritmos de descenso de gradiente.

Scikit-learn proporciona la clase StandardScaler y lo empleamos de la siguiente forma:

[9]:
```
#importamos el modulo StandartScaler
from sklearn.preprocessing import StandardScaler
#extraemos en array los registros de las columnas
array = df[["fixed acidity","volatile acidity","citric acid","residual␣
 ↪sugar","chlorides","free sulfur dioxide","total sulfur␣
 ↪dioxide","density","pH","sulphates","alcohol"]]
#definimos una instancia de la clase StandartScaler
scaler = StandardScaler()
#ajustamos y transformamos el array
array_scaled = scaler.fit_transform(array)
#convertimos a dataframe para la visualizacion
df[["fixed acidity","volatile acidity","citric acid","residual␣
 ↪sugar","chlorides","free sulfur dioxide","total sulfur␣
 ↪dioxide","density","pH","sulphates","alcohol"]] = array_scaled
#mostramos el dataframe
df
```

```
C:\Users\TOSHIBA\AppData\Local\Temp\ipykernel_12048\3513895584.py:10:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[["fixed acidity","volatile acidity","citric acid","residual
sugar","chlorides","free sulfur dioxide","total sulfur
dioxide","density","pH","sulphates","alcohol"]] = array_scaled
```

[9]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	\
0	white	-0.164340	-0.440944	0.280575	3.474773	
1	white	-0.695516	-0.262464	0.144578	-0.767085	
2	white	0.670365	-0.381451	0.552571	0.409975	
3	white	-0.012575	-0.678917	0.008580	0.765314	

```
6     white      -0.771398         -0.143478    -1.079403         0.432184
…     …            …               …           …               …
6491    red      -0.316104          1.641321    -1.623395        -0.700459
6492    red      -0.771398          1.522334    -1.623395        -0.678250
6493    red      -0.999045          1.224868    -1.487397        -0.633832
6495    red      -0.999045          1.790054    -1.351399        -0.678250
6496    red      -0.923162         -0.202971     1.028564        -0.322911

      chlorides  free sulfur dioxide  total sulfur dioxide   density  \
0     -0.316773             0.838662              0.982996  2.179883
1     -0.208178            -0.901550              0.313700 -0.180714
2     -0.181029            -0.003376             -0.302757  0.190237
3      0.036160             0.950934              1.264804  0.358851
6     -0.316773            -0.003376              0.384152  0.122791
…         …                  …                    …        …
6491   0.307648            -0.115648             -1.341926  0.665729
6492   0.904919             0.108896             -1.236248  0.122791
6493   0.144755             0.501847             -1.112957  0.196981
6495   0.497689             0.108896             -1.236248  0.315011
6496   0.280499            -0.677007             -1.271474  0.321756

            pH  sulphates   alcohol  quality
0    -1.401558  -0.556392 -1.474288        6
1     0.471982  -0.289108 -0.884080        6
2     0.222176  -0.623213 -0.378187        6
3    -0.214983  -0.890496 -0.546818        6
6    -0.277434  -0.422750 -0.799765        6
…        …         …       …      …
6491  1.221397   1.915982 -0.884080        6
6492  1.408751   0.312280 -0.040926        5
6493  1.845910  -0.013572  0.549282        6
6495  2.158167   1.180952 -0.293872        5
6496  1.034043   0.846848  0.380652        6

[5329 rows x 13 columns]
```