INF-354-1P-P6

October 13, 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

[3]:
```
#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
```

[3]:

	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	
…	…	…	…	…	…	
6492	red	6.2	0.600	0.08	2.0	
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	

	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
…         …                     …                    …      …        …
6492      0.090                 32.0                  44.0  0.99490  3.45
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

      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
…          …      …       …
6492       0.58     10.5        5
6493        NaN     11.2        6
6494       0.75     11.0        6
6495       0.71     10.2        5
6496       0.66     11.0        6

[6497 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

[4]:
```
#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
```

[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	
4	white	7.2	0.230	0.32	8.5	
…	…	…	…	…	…	
6492	red	6.2	0.600	0.08	2.0	
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	

	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
…          …                    …                     …      …        …
6492       0.090                  32.0                   44.0  0.99490  3.45
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

      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
…         …       …       …
6492   0.580000     10.5        5
6493   0.531215     11.2        6
6494   0.750000     11.0        6
6495   0.710000     10.2        5
6496   0.660000     11.0        6

[6497 rows x 13 columns]
```

1.2 Segunda tecnica de preprocesamiento

1.2.1 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 caracteristica 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:

[5]:
```
#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
```

[5]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	\
0	white	-0.167159	-0.423564	0.284245	3.206977	
1	white	-0.707427	-0.241232	0.146523	-0.808136	
2	white	0.681835	-0.362786	0.559689	0.306005	
3	white	-0.012796	-0.666673	0.008801	0.642350	
4	white	-0.012796	-0.666673	0.008801	0.642350	
…	…	…	…	…	…	
6492	red	-0.784609	1.582089	-1.643863	-0.724050	
6493	red	-1.016152	1.278203	-1.506141	-0.682007	
6494	red	-0.707427	1.035093	-1.299558	-0.660986	
6495	red	-1.016152	1.855587	-1.368419	-0.724050	
6496	red	-0.938971	-0.180454	1.041716	-0.387706	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
0	-0.315222	0.815565	0.959976	2.102214	
1	-0.201027	-0.931107	0.287618	-0.232332	
2	-0.172479	-0.029599	-0.331660	0.134525	
3	0.055911	0.928254	1.243074	0.301278	
4	0.055911	0.928254	1.243074	0.301278	
…	…	…	…	…	
6492	0.969467	0.083090	-1.269422	0.067824	
6493	0.170105	0.477500	-1.145567	0.141195	
6494	0.569786	-0.085943	-1.340197	0.347969	
6495	0.541237	0.083090	-1.269422	0.257923	
6496	0.312848	-0.705730	-1.304809	0.264593	

	pH	sulphates	alcohol	quality
0	-1.359665	-0.545959	-1.418558	6
1	0.508045	-0.277064	-0.831615	6
2	0.259017	-0.613183	-0.328521	6

```
3    -0.176782  -0.882078 -0.496219        6
4    -0.176782  -0.882078 -0.496219        6
…          …          …         …        …
6492  1.441900   0.327950  0.006875        5
6493  1.877699   0.000000  0.593818        6
6494  1.255129   1.470755  0.426120        6
6495  2.188984   1.201860 -0.244672        5
6496  1.068358   0.865741  0.426120        6

[6497 rows x 13 columns]
```

1.3 Tercera tecnica de preprocesamiento

1.3.1 Normalizacion

La normalizacion es una tecnica de preprocesamiento que ajusta los valores de las caracteristicas de un conjunto de datos para que se encuentren dentro de un rango especifico, generalmente este es de 0 y 1.

Los problemas que se tienen con caracteristicas no normalizadas son las siguientes:

Diferencia de magnitudes y rangos de las caracteristicas, ya que puede ocasionar que algunos algoritmos sean menos efectivos y eficientes.

Problemas de interpretacion, resulta mas dificiles interpretar caracteristicas no normalizadas

Algunos de los beneficios de aplicar la normalizacion son los siguientes:

Reduccion de valores atipicos, ya que estos llegaran a tener menor influencia en el modelo despues de haberse realizada la normalizacion

Aceleracion del tiempo de entrenamiento, algunos algoritmos pueden converger mas rapido cuando las caracteristicas estan normalizadas

scikit-learn tiene la clase MinMaxScaler La normalización min-max tiene como objetivo escalar todos los valores numéricos v de un atributo numérico A a un rango específico indicado por [nuevo − min A, nuevo − maxA]. Así, se obtiene un valor transformado en este caso [0-1]

[8]:
```
#inportmos el modulo nomalize
from sklearn.preprocessing import MinMaxScaler
#extraemos en array1 los registros de las columnas
array1 = df[["fixed acidity","volatile acidity","citric acid","residual␣
 ↪sugar","chlorides","free sulfur dioxide","total sulfur␣
 ↪dioxide","density","pH","sulphates","alcohol"]]
#definimos una instancia de MinMaxScaler
scaler = MinMaxScaler()
#aplicamos la normalizacion a array1
array1 = scaler.fit_transform(array1)
#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"]] = array1
#mostramos el dataframe
df
```

[8]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	\
0	white	0.264463	0.126667	0.216867	0.308282	
1	white	0.206612	0.146667	0.204819	0.015337	
2	white	0.355372	0.133333	0.240964	0.096626	
3	white	0.280992	0.100000	0.192771	0.121166	
4	white	0.280992	0.100000	0.192771	0.121166	
…	…	…	…	…	…	
6492	red	0.198347	0.346667	0.048193	0.021472	
6493	red	0.173554	0.313333	0.060241	0.024540	
6494	red	0.206612	0.286667	0.078313	0.026074	
6495	red	0.173554	0.376667	0.072289	0.021472	
6496	red	0.181818	0.153333	0.283133	0.046012	

	chlorides	free sulfur dioxide	total sulfur dioxide	density	\
0	0.059801	0.152778	0.377880	0.267785	
1	0.066445	0.045139	0.290323	0.132832	
2	0.068106	0.100694	0.209677	0.154039	
3	0.081395	0.159722	0.414747	0.163678	
4	0.081395	0.159722	0.414747	0.163678	
…	…	…	…	…	
6492	0.134551	0.107639	0.087558	0.150183	
6493	0.088040	0.131944	0.103687	0.154425	
6494	0.111296	0.097222	0.078341	0.166377	
6495	0.109635	0.107639	0.087558	0.161172	
6496	0.096346	0.059028	0.082949	0.161558	

	pH	sulphates	alcohol	quality
0	0.217054	0.129213	0.115942	6
1	0.449612	0.151685	0.217391	6
2	0.418605	0.123596	0.304348	6
3	0.364341	0.101124	0.275362	6
4	0.364341	0.101124	0.275362	6
…	…	…	…	…
6492	0.565891	0.202247	0.362319	5
6493	0.620155	0.174840	0.463768	6
6494	0.542636	0.297753	0.434783	6
6495	0.658915	0.275281	0.318841	5
6496	0.519380	0.247191	0.434783	6

[6497 rows x 13 columns]