INF_354_1P_P3

October 13, 2023

1 Pimer Parcial de Inteligencia Artificial

1.0.1 Nombre: Steve Brandom Nina Huacani

1.0.2 Pregunta 3. Del dataset elegido realice la imputación por columnas sin scikit-learn.

Importamos el dataset elegido en este caso "Wine Quality", relacionado con las variantes tinta y blanca del vino portugués "Vinho Verde". EL dataset lo guardaremos en Drive y lo importaremos de la siguiente forma:

[1]:
```
#importamos el modulo de drive
#from google.colab import drive
#montamos la carpeta donde se encuentra nuestro dataset
#drive.mount("/content/drive")
#especificamos la ruta donde se encuentra el archivo csv
#archivo="/content/drive/MyDrive/data/wineQuality.csv"
archivo = "wine_quality.csv"
```

Definimos la funcion obtenerDatos() que tendra como parametro la ruta donde se encuentra el dataset en este caso esta almacenado en Drive. La funcion devolvera dos arrays, el primero unidemensional que sera un array que contenga las cabeceras con el nombre de los atributos del dataset y el segundo sera un array bidimensional donde se almacenaran unicamente los registros del dataset.

[2]:
```
#Funcion para obtener los datos en forma de arrays directamente del archivo csv
def obtenerDatos(ruta):
  #Array que almacenara las cabeceras del dataset
  cabecera = []
  #Array que almacenara los registros del dataset
  registros = []
  #lectura del archivo csv
  with open(ruta, 'r') as archivo:
    #almacena las lineas del archivo csv
    filas = archivo.readlines()
    #lectura linea por linea del archivo
    for i, fila in enumerate(filas):
      #separamos los valores con el patron de comas
      valores = fila.strip().split(',')
      #almacenamos las cabeceras
```

```
      if i == 0:
        cabeceras = valores
      #almacenamos los registros
      else:
        registros.append(valores)
    #devolvemos los arrays encontrados
    return cabeceras, registros
```

Solamente para visualizar en un dataframe el dataset podriamos hacer uso de la libreria pandas de la siguiente forma sin embargo podriamos utilizar la funcion mostrarDataset() pero no seria apreciado de buena forma

[3]:
```
#obtenemos las cabeceras y los registros del archivo csv
cabeceras, registros = obtenerDatos(archivo)

#SOLAMENTE UTILIZAREMOS PANDAS PARA QUE EL DATASET SEA MAS APRECIABLE
import pandas as pd
#abrimos el dataset con pandas
datos=pd.read_csv(archivo)
#mostramos el dataset con dataframe
datos
```

[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]
```

Como podemos observar en el dataframe se cuenta con datos NaN por lo que definimos la funcion verificarDatosFaltantes() que tendra como parametro el array de registros obtenido previamente. Esta funcion nos permitira mostrar la posicion en la que se encuentran los datos NaN para posteriormente realizar la imputacion ya sea por media o por la mediana.

[4]:
```
#definimos la funcion que verificara los datos faltantes en el array de␣
 ↪registros
def verificarDatosFaltantes(registros):
  #el swicth nos idicara si existe datos faltantes
  sw = 0
  #iteramos la matriz buscando datos faltantes
  for i in range(len(registros)):
    for j in range(13):
      #si encontramos uno mostramos su posicion en i y en j
      if(registros[i][j] == ""):
        print("Posicion i: ",i," j: ",j)
        #cambiamos el valor del swicht
        sw = 1
  #verificamos el estado del swicht
  if(sw == 0):
    print("No existen campos vacios")
  else:
    print("Existen campos vacios")
```

Definimos la funcion getMedia() que tendra como parametros el array de cabeceras, el array de registros y el nombre de la columna que debera de ser de tipo cadena. Ademas definiremos la funcion getMediana() que llevara los mismos parametros que la anterior funcion. Ambas funciones nos ayudaran para elegir la estrategia de imputacion mas adelante.

[5]:
```
#definimos la funcion para obtener la media que nos servira para elegir la␣
 ↪estrategia
#funcion para hallar la media
def getMedia(cabeceras, registros, columna):
```

```
  suma = 0
  cont = 0
  #buscara la posicion del nombre de la columna introducida esto con el fin de␣
 ↪luego de encontrar la columna en el array de registros
  indice = cabeceras.index(columna)
  #iteramos sumando los valores de la columna
  for i in range(len(registros)):
    if(registros[i][indice] != ""):
      #convertimos a float ya que con la funcion anterior los datos eran cadenas
      suma += float(registros[i][indice])
      cont+=1
  #devolvemos la media
  return suma/cont

#funcion para hallar la mediana
def getMediana(cabeceras, registros, columna):
  #creamos un array unidimensional
  array = []
  #buscara la posicion del nombre de la columna introducida esto con el fin de␣
 ↪luego de encontrar la columna en el array de registros
  indice = cabeceras.index(columna)
  #almacenamos los datos de la columna en el array
  for i in range(len(registros)):
    if(registros[i][indice] != ""):
      array.append(float(registros[i][indice]))
  #ordenamos el array
  array.sort()
  #obtenemos la mediana
  if(len(array)%2 == 0):
    mediana = array[int(len(array)/2)]
  else:
    mediana = (array[int(len(array)//2)] + array[int((len(array)//2) + 1)])/2
  #devolvemos la mediana
  return mediana

#funcion para obtener la moda
def getModa(cabeceras, registros, columna):
  #creamos un diccionario que almacenara las frecuencias
  frecuencias = {}
  #buscara la posicion del nombre de la columna introducida esto con el fin de␣
 ↪luego de encontrar la columna en el array de registros
  indice = cabeceras.index(columna)
  #iteramos la columna seleccionada
  for i in range(len(registros)):
    if(registros[i][indice] != ""):
      dato = float(registros[i][indice])
      #si el dato esta en el diccionario sumamos una frecuencia
```

```
      if dato in frecuencias:
        frecuencias[dato] += 1
      #si no esta creamos un registro con el valor de 1
      else:
        frecuencias[dato] = 1
  #obtenemos el dato con el mayor numero de repeticiones
  moda = max(frecuencias, key=frecuencias.get)
  #devolvemos la moda
  return moda
```

Definimos la funcion imputacion() que recibira como parametros el array de cabeceras, el array de registros, el nombre de la columna a imputar este debe ser de tipo cadena y la estrategia la cual podra ser la media o mediana. La funcion recorrera la columna introducida verificando si se tiene datos vacios para que posteriormente sean llenados con la estrategia elegida.

[6]:
```
#definimos la funcion para la imputacion
def imputacion(cabeceras, registros, columna, estrategia):
  #buscara la posicion del nombre de la columna introducida esto con el fin de␣
 ↪luego de encontrar la columna en el array de registros
  indice = cabeceras.index(columna)
  #si la estrategia es media rellenara los campos vacios con la media
  if(estrategia == "media"):
    media = getMedia(cabeceras, registros, columna)
    for i in range(len(registros)):
      if(registros[i][indice] == ""):
        registros[i][indice] = media
  #si la estrategia es media rellenara los campos vacios con la mmediana
  elif(estrategia == "mediana"):
    mediana = getMediana(cabeceras, registros, columna)
    for i in range(len(registros)):
      if(registros[i][indice] == ""):
        registros[i][indice] = mediana
  #si la estrategia es media rellenara los campos vacios con la moda
  elif(estrategia == "moda"):
    moda = getModa(cabeceras, registros, columna)
    for i in range(len(registros)):
      if(registros[i][indice] == ""):
        registros[i][indice] = moda
  #devolvemos el array ya imputado
  return registros
```

[7]:
```
print("Verificamos datos faltantes si existe uno imprimiremos su posicion\n")
#verificamos datos faltantes
verificarDatosFaltantes(registros)
#mostramos la moda de la columna fixed acidity
print("La media de la columna fixed acidity es: ",getModa(cabeceras, registros,␣
 ↪"fixed acidity"))
```

```
#aplicamos la estrategia de moda en la columna de fixed acidity
mat1 = imputacion(cabeceras, registros, "fixed acidity", estrategia="moda")
#volvemos a verificar datos faltantes
verificarDatosFaltantes(registros)
#mostramos la mediana de la columna volatile acidity
print("La mediana de la columna volatile acidity es: ",getMediana(cabeceras,␣
 ↪registros, "volatile acidity"))
#aplicamos la estrategia de la mediana en la columna volatile acidity
mat1 = imputacion(cabeceras, registros, "volatile acidity",␣
 ↪estrategia="mediana")
#volvemos a verificar los datos faltantes
verificarDatosFaltantes(registros)
#mostramos la media de la columna  citric acid
print("La media de la columna citric acid es: ",getMedia(cabeceras, registros,␣
 ↪"citric acid"))
#aplicamos la estrategia de la media en la columna citric acid
mat1 = imputacion(cabeceras, registros, "citric acid", estrategia="media")
#aplicamos la estrategia de la media en la columna residual sugar
mat1 = imputacion(cabeceras, registros, "residual sugar", estrategia="media")
#aplicamos la estrategia de la media en la columna chlorides
mat1 = imputacion(cabeceras, registros, "chlorides", estrategia="media")
#aplicamos la estrategia de media en la columna de sulfur dioxide
mat1 = imputacion(cabeceras, registros, "free sulfur dioxide",␣
 ↪estrategia="media")
#aplicamos la estrategia de media en la columna de total sulfur dioxide
mat1 = imputacion(cabeceras, registros, "total sulfur dioxide",␣
 ↪estrategia="media")
#aplicamos la estrategia de media en la columna de density
mat1 = imputacion(cabeceras, registros, "density", estrategia="media")
#aplicamos la estrategia de media en la columna de pH
mat1 = imputacion(cabeceras, registros, "pH", estrategia="media")
#aplicamos la estrategia de media en la columna de sulphates
mat1 = imputacion(cabeceras, registros, "sulphates", estrategia="media")
print()
#volvemos a verificar los datos faltantes
verificarDatosFaltantes(mat1)
```

```
Verificamos datos faltantes si existe uno imprimiremos su posicion

Posicion i:  17  j:  1
Posicion i:  33  j:  4
Posicion i:  54  j:  9
Posicion i:  86  j:  2
Posicion i:  98  j:  5
Posicion i:  139  j:  9
Posicion i:  174  j:  1
Posicion i:  224  j:  10
```

```
Posicion i:  249  j:  1
Posicion i:  267  j:  1
Posicion i:  268  j:  3
Posicion i:  368  j:  1
Posicion i:  438  j:  4
Posicion i:  440  j:  9
Posicion i:  518  j:  1
Posicion i:  521  j:  2
Posicion i:  587  j:  9
Posicion i:  621  j:  2
Posicion i:  697  j:  9
Posicion i:  747  j:  5
Posicion i:  812  j:  2
Posicion i:  909  j:  3
Posicion i:  972  j:  9
Posicion i:  1079  j:  1
Posicion i:  1079  j:  2
Posicion i:  2894  j:  2
Posicion i:  2902  j:  1
Posicion i:  2902  j:  10
Posicion i:  4892  j:  9
Posicion i:  4895  j:  2
Posicion i:  6320  j:  3
Posicion i:  6321  j:  10
Posicion i:  6428  j:  1
Posicion i:  6428  j:  9
Posicion i:  6429  j:  1
Posicion i:  6429  j:  9
Posicion i:  6486  j:  2
Posicion i:  6493  j:  10
Existen campos vacios
La media de la columna fixed acidity es:  6.8
Posicion i:  33  j:  4
Posicion i:  54  j:  9
Posicion i:  86  j:  2
Posicion i:  98  j:  5
Posicion i:  139  j:  9
Posicion i:  224  j:  10
Posicion i:  268  j:  3
Posicion i:  438  j:  4
Posicion i:  440  j:  9
Posicion i:  521  j:  2
Posicion i:  587  j:  9
Posicion i:  621  j:  2
Posicion i:  697  j:  9
Posicion i:  747  j:  5
Posicion i:  812  j:  2
Posicion i:  909  j:  3
```

```
Posicion i:  972  j:  9
Posicion i:  1079  j:  2
Posicion i:  2894  j:  2
Posicion i:  2902  j:  10
Posicion i:  4892  j:  9
Posicion i:  4895  j:  2
Posicion i:  6320  j:  3
Posicion i:  6321  j:  10
Posicion i:  6428  j:  9
Posicion i:  6429  j:  9
Posicion i:  6486  j:  2
Posicion i:  6493  j:  10
Existen campos vacios
La mediana de la columna volatile acidity es:  0.29
Posicion i:  33  j:  4
Posicion i:  54  j:  9
Posicion i:  98  j:  5
Posicion i:  139  j:  9
Posicion i:  224  j:  10
Posicion i:  268  j:  3
Posicion i:  438  j:  4
Posicion i:  440  j:  9
Posicion i:  587  j:  9
Posicion i:  697  j:  9
Posicion i:  747  j:  5
Posicion i:  909  j:  3
Posicion i:  972  j:  9
Posicion i:  2902  j:  10
Posicion i:  4892  j:  9
Posicion i:  6320  j:  3
Posicion i:  6321  j:  10
Posicion i:  6428  j:  9
Posicion i:  6429  j:  9
Posicion i:  6493  j:  10
Existen campos vacios
La media de la columna citric acid es:  0.3187218971358124

No existen campos vacios
```

Solamente para verificar los cambios utilizaremos pandas ya que nos permite una visualizacion mas amigable.

[8]:
```
df = pd.DataFrame(registros, columns=cabeceras)
df
```

[8]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	\
0	white	7	0.27	0.36	20.7	
1	white	6.3	0.3	0.34	1.6	

```
2     white           8.1             0.28         0.4            6.9
3     white           7.2             0.23        0.32            8.5
4     white           7.2             0.23        0.32            8.5
…     …             …               …          …             …
6492    red           6.2              0.6        0.08              2
6493    red           5.9             0.55         0.1            2.2
6494    red           6.3             0.51        0.13            2.3
6495    red           5.9            0.645        0.12              2
6496    red             6             0.31        0.47            3.6

     chlorides free sulfur dioxide total sulfur dioxide  density    pH  \
0        0.045                  45                  170    1.001     3
1        0.049                  14                  132    0.994   3.3
2         0.05                  30                   97   0.9951  3.26
3        0.058                  47                  186   0.9956  3.19
4        0.058                  47                  186   0.9956  3.19
…        …                    …                   …       …   …
6492      0.09                  32                   44   0.9949  3.45
6493     0.062                  39                   51  0.99512  3.52
6494     0.076                  29                   40  0.99574  3.42
6495     0.075                  32                   44  0.99547  3.57
6496     0.067                  18                   42  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          0.4     9.9       6
4          0.4     9.9       6
…        …     …      …
6492      0.58    10.5       5
6493  0.531215    11.2       6
6494      0.75      11       6
6495      0.71    10.2       5
6496      0.66      11       6

[6497 rows x 13 columns]
```