**LAB EXERCISE – 2**

**Data Preprocessing**

**Aim of the Experiment.**

The main aim of this experiment is to preprocess the given dataset. The database is created and is available in the file sample.csv.

Sample Dataset

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | first | last | gender | Marks | selected |
| 1 | Leone | Debrick | Female | 50 | TRUE |
| 2 | Romola | Phinnessy | Female | 60 | FALSE |
| 3 | Geri | Prium | Male | 65 | FALSE |
| 4 | Sandy | Doveston | Female | 95 | FALSE |
| 5 | Jacenta | Jansik | Female | 31 | TRUE |
| 6 | Diane-marie | Medhurst | Female | 45 | TRUE |
| 7 | Austen | Pool | Male | 45 | TRUE |
| 8 | Vanya | Teffrey | Male | 70 | FALSE |
| 9 | Giordano | Elloy | Male | 36 | FALSE |
| 10 | Rozele | Fawcett | Female | 50 | FALSE |

The objectives of this experiment are

1. Explore Label Encoder
2. Explore Scikit Preprocessing routines like Scaling
3. Explore Scikit Preprocessing routines like Binarizer

**Reference to the Textbook and Explanation**

All the fundamentals are given in Chapter 2 and Appendix 2.

The variable in the dataset Female and Male can be changed to 0 or 1 using Label Encoder. It is done as given below:

df\_gender\_encode=LabelEncoder()

df.gender=df\_gender\_encode.fit\_transform(df.gender)

Scaling can be done as follows:

df.Marks = preprocessing.scale(df.Marks)

scaled\_df= preprocessing.scale(df.Marks)

Scaling removes the mean

Binarization uses threshold and converts values to binary as shown below:

scaled\_df\_bin = preprocessing.Binarizer(threshold=0.5).transform(newarr)

Duplicates can be removed as follows:

df\_duplicates\_removed = pd.DataFrame.drop\_duplicates(df\_duplicated)

The NaN of a column can be removed as shown below:

df['m5']=df['m5'].fillna(0)

This removes all the NaN to zero.

The command,

df=df.dropna(axis=1)

removes all the columns that has NaN.

**Listing 1**

import pandas as pd

col\_list=["id","first","last","gender","Marks","selected"]

df = pd.read\_csv("sample.csv",usecols=col\_list)

print(df)

print("End of Listing\n\n\n")

# Let us convert the in Gender column, make Female as 0 and

# male as 1 using LabelEncoder in scikitlearn method

from sklearn.preprocessing import LabelEncoder

df\_gender\_encode=LabelEncoder()

df.gender=df\_gender\_encode.fit\_transform(df.gender)

# One can observe that female is coded as 0 and Male as 1

print(df)

print("End of Listing\n\n\n")

# Now one can scale the marks to remove mean

from sklearn import preprocessing

df.Marks = preprocessing.scale(df.Marks)

scaled\_df= preprocessing.scale(df.Marks)

print(df)

print("Scaling of marks is completed\n\n\n\n")

newarr = scaled\_df.reshape(-1,1)

scaled\_df\_bin = preprocessing.Binarizer(threshold=0.5).transform(newarr)

df['Marks']=scaled\_df\_bin

print(df)

print("Binarizarion of marks is completed\n\n\n\n")

**Output**

**A screen shot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**Listing 2**

import pandas as pd

col\_list=["id","first","last","gender","Marks","selected"]

df = pd.read\_csv("sample.csv",usecols=col\_list)

print(df)

print("End of Listing\n\n\n")

# Let us create duplicate elements in the given dataset

# This is done using the command concate 2 times as given below

df\_duplicated = pd.concat([df]\*2, ignore\_index=True)

print(df\_duplicated)

print("Display before duplication\n\n\n\n")

df\_duplicates\_removed = pd.DataFrame.drop\_duplicates(df\_duplicated)

print(df\_duplicates\_removed)

print("Display after duplication\n\n\n\n")

Output

A picture containing text, scoreboard, plaque

Description automatically generated

**A close up of text on a black background

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**Listing 3**

import pandas as pd

df = pd.DataFrame({

'm1':[50,'A',60,'A',80],

'm2':[60,'A','60','A',80],

'm3':[50,70,'A','A',60],

'm4':[60,'A','A','A',60],

'm5':['A','A','A',10,20]

})

df = df.apply(pd.to\_numeric,errors='coerce')

print(df)

print('Dataframe with NaN\n\n\n')

# Make all the NaN in Mark5 as zero

df['m5']=df['m5'].fillna(0)

print(df)

print('Making m5 NaN as 0 using fillna() function\n\n\n\n')

df1 = df.copy()

df1['m2'].fillna(df1['m2'].mean(),inplace=True)

print(df1)

print('Making m5 NaN as mean using fillna() function\n\n\n\n')

df2 = df.copy()

df1['m3'].fillna(df1['m2'].median(),inplace=True)

print(df2)

print('Making m5 NaN as median using fillna() function\n\n\n\n')

# Dropping all columns having NaN

df=df.dropna(axis=1)

print(df)

print('Dropping all columns having NaN\n\n\n\n')

**Output**

**A close up of a screen

Description automatically generated**

**A picture containing calendar

Description automatically generated**

**A picture containing calendar

Description automatically generated**

**A close up of a screen

Description automatically generated**

**Listing 4**

This listing illustrates the use of MinMax scaling and Standard scaling for finding Z-scores.

from numpy import asarray

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import StandardScaler

data = asarray([[1,3],[8,5],[6,7],[8,9]])

print("\n Original Data")

print(data)

scaler1 = MinMaxScaler()

scaler2 = StandardScaler()

scaled1 = scaler1.fit\_transform(data)

scaled2 = scaler2.fit\_transform(data)

print("\n\nThe output of MinMax Scaling")

print(scaled1)

print("\n\nThe output of Standard scaling as z-score")

print(scaled2)

**Output**

**Text

Description automatically generated**