missxmuyi

August 14, 2023

CONTEXT

To apply for a master's degree is a very expensive and intensive work. With this kernel, students will guess their capacities and they will decide whetherto apply for a master's degree or not.So, basically this set is about the Graduate Admissions data i.e. Given a set of standardized scores like GRE, TOEFL, SOP standard scores, LOR standard scores. All those folks who are preparing for Master's, might point out this question, from where did you get SOP & LOR scores. This project aimed to assist prospective students in making informed decisions by providing them with insights into their potential for admission, it's worth considering that universities might utilize internal rating systems to standardize these scores and enhance their predictive accuracy.

[1]:
```
#Used for Data Processing and Analysis
import pandas as pd
#Used for Mathematical Computations
import numpy as np
#Used for data visualizations
import matplotlib.pyplot as plt
#used for data Statistical visualizations
import seaborn as sns
#For calculate accuracy score from data
from sklearn.metrics import accuracy_score,confusion_matrix
```

[2]:
```
# Load the dataset using Pandas
graduate_ad=pd.read_csv(r"C:\Python310\practice23\machine learning1\New␣
↪folder\Admission_Predict_Ver1.1.csv")
```

[3]:
```
#Extract first rows from datasets
graduate_ad.head()
```

[3]:
```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
0           1        337          118                  4  4.5   4.5  9.65
1           2        324          107                  4  4.0   4.5  8.87
2           3        316          104                  3  3.0   3.5  8.00
3           4        322          110                  3  3.5   2.5  8.67
4           5        314          103                  2  2.0   3.0  8.21

   Research  Chance of Admit
```

```
0          1              0.92
1          1              0.76
2          1              0.72
3          1              0.80
4          0              0.65
```

[4]:
```
#Extract last five rows from datasets
graduate_ad.tail()
```

[4]:
```
     Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  \
495         496        332          108                  5  4.5   4.0  9.02
496         497        337          117                  5  5.0   5.0  9.87
497         498        330          120                  5  4.5   5.0  9.56
498         499        312          103                  4  4.0   5.0  8.43
499         500        327          113                  4  4.5   4.5  9.04

     Research  Chance of Admit
495         1             0.87
496         1             0.96
497         1             0.93
498         0             0.73
499         0             0.84
```

[5]:
```
#Total number of rows and columns
graduate_ad.shape
```

[5]:
```
(500, 9)
```

[6]:
```
#Information about the dataset like total no.of. rows,total no.of␣
↪columns,datatypes of each columns and memory management
graduate_ad.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         500 non-null    int64
 1   GRE Score          500 non-null    int64
 2   TOEFL Score        500 non-null    int64
 3   University Rating  500 non-null    int64
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
```

```
memory usage: 35.3 KB
```

[7]:
```
#Checking for any empty/null values present in dataset
graduate_ad.isnull().sum()
```

[7]:
```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

[8]:
```
#Statistical measures of a dataframe
graduate_ad.describe()
```

[8]:
```
       Serial No.   GRE Score  TOEFL Score  University Rating         SOP  \
count  500.000000  500.000000   500.000000         500.000000  500.000000
mean   250.500000  316.472000   107.192000           3.114000    3.374000
std    144.481833   11.295148     6.081868           1.143512    0.991004
min      1.000000  290.000000    92.000000           1.000000    1.000000
25%    125.750000  308.000000   103.000000           2.000000    2.500000
50%    250.500000  317.000000   107.000000           3.000000    3.500000
75%    375.250000  325.000000   112.000000           4.000000    4.000000
max    500.000000  340.000000   120.000000           5.000000    5.000000

            LOR         CGPA    Research  Chance of Admit
count  500.00000  500.000000  500.000000        500.00000
mean     3.48400    8.576440    0.560000          0.72174
std      0.92545    0.604813    0.496884          0.14114
min      1.00000    6.800000    0.000000          0.34000
25%      3.00000    8.127500    0.000000          0.63000
50%      3.50000    8.560000    1.000000          0.72000
75%      4.00000    9.040000    1.000000          0.82000
max      5.00000    9.920000    1.000000          0.97000
```

[9]:
```
graduate_ad.columns
```

[9]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')
```

PREPROCESSING THE DATASETS

[10]:
```
X=graduate_ad.drop(['Serial No.','Chance of Admit '],axis=1)
```

[11]:
```
print(X)
```

```
     GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  Research
0          337          118                  4  4.5   4.5  9.65         1
1          324          107                  4  4.0   4.5  8.87         1
2          316          104                  3  3.0   3.5  8.00         1
3          322          110                  3  3.5   2.5  8.67         1
4          314          103                  2  2.0   3.0  8.21         0
..         …            …                 …    …    …    …         …
495        332          108                  5  4.5   4.0  9.02         1
496        337          117                  5  5.0   5.0  9.87         1
497        330          120                  5  4.5   5.0  9.56         1
498        312          103                  4  4.0   5.0  8.43         0
499        327          113                  4  4.5   4.5  9.04         0

[500 rows x 7 columns]
```

[12]:
```
y=graduate_ad['Chance of Admit ']
y
```

[12]:
```
0      0.92
1      0.76
2      0.72
3      0.80
4      0.65
       …
495    0.87
496    0.96
497    0.93
498    0.73
499    0.84
Name: Chance of Admit , Length: 500, dtype: float64
```

SPLITTING DATASET INTO TRAINING AND TEST DATA

[13]:
```
from sklearn.model_selection import train_test_split
```

[14]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
 ↪20,random_state=42)
```

[15]:
```
X_train,X_test,y_train,y_test
```

[15]:
```
(      GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  Research
 249        321          111                  3  3.5   4.0  8.83         1
 433        316          111                  4  4.0   5.0  8.54         0
 19         303          102                  3  3.5   3.0  8.50         0
 322        314          107                  2  2.5   4.0  8.27         0
 332        308          106                  3  3.5   2.5  8.21         1
```

```
..         …          …                 …  …   …   …      …
106        329          111                 4  4.5   4.5  9.18         1
270        306          105                 2  2.5   3.0  8.22         1
348        302           99                 1  2.0   2.0  7.25         0
435        309          105                 2  2.5   4.0  7.68         0
102        314          106                 2  4.0   3.5  8.25         0

[400 rows x 7 columns],
     GRE Score  TOEFL Score  University Rating  SOP  LOR   CGPA  Research
361        334          116                 4  4.0   3.5  9.54         1
73         314          108                 4  4.5   4.0  9.04         1
374        315          105                 2  2.0   2.5  7.65         0
155        312          109                 3  3.0   3.0  8.69         0
104        326          112                 3  3.5   3.0  9.05         1
..         …          …                 …  …   …   …      …
347        299           94                 1  1.0   1.0  7.34         0
86         315          106                 3  4.5   3.5  8.42         0
75         329          114                 2  2.0   4.0  8.56         1
438        318          110                 1  2.5   3.5  8.54         1
15         314          105                 3  3.5   2.5  8.30         0

[100 rows x 7 columns],
249    0.77
433    0.71
19     0.62
322    0.72
332    0.75
       …
106    0.87
270    0.72
348    0.57
435    0.55
102    0.62
Name: Chance of Admit , Length: 400, dtype: float64,
361    0.93
73     0.84
374    0.39
155    0.77
104    0.74
       …
347    0.42
86     0.72
75     0.72
438    0.67
15     0.54
Name: Chance of Admit , Length: 100, dtype: float64)
```

[16]:
```python
print(X_train.shape,X_test.shape,X.shape)
```

```
(400, 7) (100, 7) (500, 7)
```

[17]:
```python
y_train=[1 if value>0.8 else 0 for value in y_train]
y_test=[1 if value>0.8 else 0 for value in y_test]
y_train=np.array(y_train)
y_test=np.array(y_test)
```

[18]:
```python
print(y_train)
```

```
[0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 0 1
 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1 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 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0 1 0
 0 0 1 0 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 1
 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 1 0 1
 1 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
 1 1 0 1 0 1 1 0 0 0 1 0 1 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0
 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 0 1 0 0 0 1 0
 1 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0]
```

[19]:
```python
print(y_test)
```

```
[1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0
 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0]
```

STANDARDSCALER:

The StandardScaler is a preprocessing technique commonly used in machine learning to transform numerical features in a dataset. It is used to standardize the scale of the features, making them have a mean of 0 and a standard deviation of 1.

[20]:
```python
from sklearn.preprocessing import StandardScaler
```

[21]:
```python
SC=StandardScaler()
```

[22]:
```python
X_train=SC.fit_transform(X_train)
X_test=SC.fit_transform(X_test)
```

[23]:
```python
print(X_train)
```

```
[[ 0.38998634  0.6024183  -0.09829757 …  0.56498381  0.4150183
   0.89543386]
 [-0.06640493  0.6024183   0.7754586  …  1.65149114 -0.06785154
  -1.11677706]
 [-1.25302222 -0.87691722 -0.09829757 … -0.52152352 -0.13445427
```

```
 -1.11677706]
…
[-1.34430047 -1.37002906 -1.8458099  … -1.60803084 -2.2157898
 -1.11677706]
[-0.7053527  -0.38380538 -0.97205374 …  0.56498381 -1.49981038
 -1.11677706]
[-0.24896144 -0.21943477 -0.97205374 …  0.02173015 -0.55072138
 -1.11677706]]
```

[24]:
```
X_test
```

[24]:
```
array([[ 1.48887961e+00,  1.56082934e+00,  7.76121546e-01,
         6.25913605e-01,  0.00000000e+00,  1.58648736e+00,
         8.50962943e-01],
       [-1.16381425e-01,  2.29631609e-01,  7.76121546e-01,
         1.12267043e+00,  5.31494003e-01,  7.77715255e-01,
         8.50962943e-01],
       [-3.61183733e-02, -2.69567541e-01, -9.87791059e-01,
        -1.36111371e+00, -1.06298801e+00, -1.47067119e+00,
        -1.17513930e+00],
       [-2.76907529e-01,  3.96031326e-01, -1.05834756e-01,
        -3.67600054e-01, -5.31494003e-01,  2.11574782e-01,
        -1.17513930e+00],
       [ 8.46775196e-01,  8.95230476e-01, -1.05834756e-01,
         1.29156776e-01, -5.31494003e-01,  7.93890697e-01,
         8.50962943e-01],
       [ 1.08756435e+00,  7.28830760e-01,  7.76121546e-01,
         1.12267043e+00,  5.31494003e-01,  1.08504865e+00,
         8.50962943e-01],
       [-2.04269467e+00, -1.10156613e+00, -1.86974736e+00,
        -1.85787054e+00, -1.59448201e+00, -1.61625017e+00,
        -1.17513930e+00],
       [-1.15980110e+00, -1.03167824e-01,  7.76121546e-01,
        -8.64356883e-01, -5.31494003e-01, -1.44284943e-01,
        -1.17513930e+00],
       [ 2.04670782e-01,  3.96031326e-01, -1.05834756e-01,
         1.29156776e-01,  5.31494003e-01,  1.06887321e+00,
         8.50962943e-01],
       [ 3.65196886e-01,  8.95230476e-01,  7.76121546e-01,
        -3.67600054e-01,  1.06298801e+00,  4.86557297e-01,
         8.50962943e-01],
       [ 6.05986041e-01,  2.29631609e-01, -1.05834756e-01,
         1.29156776e-01, -5.31494003e-01,  6.59958036e-02,
        -1.17513930e+00],
       [ 4.41446785e-02,  3.96031326e-01, -1.05834756e-01,
         1.29156776e-01, -5.31494003e-01,  3.24802877e-01,
        -1.17513930e+00],
```

```
[ 5.25722989e-01, -6.02366975e-01,  7.76121546e-01,
 -3.67600054e-01, -1.06298801e+00, -8.72179836e-01,
  8.50962943e-01],
[ 1.97045792e+00,  1.39442963e+00,  1.65807785e+00,
  1.12267043e+00,  1.06298801e+00,  1.44090838e+00,
  8.50962943e-01],
[ 6.86249093e-01,  5.62431043e-01, -1.05834756e-01,
  1.29156776e-01, -5.31494003e-01,  1.06887321e+00,
  8.50962943e-01],
[-1.56111636e+00, -1.60076528e+00, -9.87791059e-01,
 -1.85787054e+00, -1.59448201e+00, -1.22803956e+00,
 -1.17513930e+00],
[ 1.32835351e+00,  2.29631609e-01,  1.65807785e+00,
  1.12267043e+00,  5.31494003e-01,  7.45364370e-01,
  8.50962943e-01],
[-1.24006415e+00, -1.60076528e+00, -9.87791059e-01,
 -3.67600054e-01, -5.31494003e-01, -7.42776300e-01,
  8.50962943e-01],
[-1.40059025e+00, -9.35166408e-01, -9.87791059e-01,
 -1.85787054e+00, -1.59448201e+00, -1.13098691e+00,
 -1.17513930e+00],
[-1.48085331e+00, -9.35166408e-01, -1.05834756e-01,
 -1.36111371e+00,  5.31494003e-01, -1.43832031e+00,
  8.50962943e-01],
[-6.78222787e-01, -2.69567541e-01, -9.87791059e-01,
 -8.64356883e-01,  1.06298801e+00, -7.10425416e-01,
  8.50962943e-01],
[-1.48085331e+00, -1.26796584e+00,  7.76121546e-01,
 -3.67600054e-01,  0.00000000e+00, -1.21186412e+00,
 -1.17513930e+00],
[-3.57170580e-01, -7.68766692e-01, -1.05834756e-01,
  1.12267043e+00,  5.31494003e-01,  1.30697572e-01,
  8.50962943e-01],
[ 9.27038248e-01,  3.96031326e-01, -1.05834756e-01,
  1.29156776e-01,  5.31494003e-01,  3.40978319e-01,
  8.50962943e-01],
[ 9.27038248e-01,  8.95230476e-01, -1.05834756e-01,
 -3.67600054e-01, -5.31494003e-01,  2.60101108e-01,
  8.50962943e-01],
[-1.15980110e+00, -7.68766692e-01, -1.05834756e-01,
 -8.64356883e-01, -1.59448201e+00, -6.94249974e-01,
  8.50962943e-01],
[ 1.16782740e+00,  2.22642821e+00,  1.65807785e+00,
  1.12267043e+00,  1.59448201e+00,  1.61883824e+00,
  8.50962943e-01],
[ 1.00730130e+00,  5.62431043e-01,  7.76121546e-01,
  1.61942726e+00,  5.31494003e-01,  9.39469675e-01,
```

```
 8.50962943e-01],
[-2.76907529e-01, -2.69567541e-01, -9.87791059e-01,
 -8.64356883e-01, -5.31494003e-01, -7.10425416e-01,
 -1.17513930e+00],
[-2.76907529e-01,  6.32318924e-02,  7.76121546e-01,
  1.12267043e+00,  5.31494003e-01,  1.46873014e-01,
  8.50962943e-01],
[-1.24006415e+00, -9.35166408e-01, -1.05834756e-01,
  1.29156776e-01, -1.06298801e+00, -1.09863603e+00,
 -1.17513930e+00],
[ 2.84933834e-01,  5.62431043e-01, -1.05834756e-01,
 -3.67600054e-01, -1.06298801e+00,  3.73329203e-01,
 -1.17513930e+00],
[-8.38748891e-01, -4.35967258e-01, -9.87791059e-01,
 -8.64356883e-01, -2.12597601e+00, -1.24421500e+00,
 -1.17513930e+00],
[ 6.05986041e-01,  1.06163019e+00,  7.76121546e-01,
  6.25913605e-01,  1.06298801e+00,  1.08504865e+00,
  8.50962943e-01],
[ 4.41446785e-02, -4.35967258e-01, -1.05834756e-01,
 -3.67600054e-01,  0.00000000e+00, -9.04530720e-01,
  8.50962943e-01],
[ 2.84933834e-01,  2.29631609e-01, -1.05834756e-01,
 -3.67600054e-01,  0.00000000e+00, -3.10568488e-02,
  8.50962943e-01],
[-1.24006415e+00, -1.43436556e+00, -1.86974736e+00,
 -1.36111371e+00, -1.06298801e+00, -8.72179836e-01,
 -1.17513930e+00],
[ 1.32835351e+00,  1.89362878e+00,  1.65807785e+00,
  1.61942726e+00,  1.59448201e+00,  1.74824178e+00,
  8.50962943e-01],
[-1.15980110e+00, -4.35967258e-01, -1.05834756e-01,
  1.29156776e-01,  5.31494003e-01, -7.10425416e-01,
  8.50962943e-01],
[-3.61183733e-02,  6.32318924e-02, -9.87791059e-01,
  6.25913605e-01, -5.31494003e-01, -9.57586170e-02,
  8.50962943e-01],
[ 1.97045792e+00,  1.06163019e+00,  7.76121546e-01,
  1.61942726e+00,  1.59448201e+00,  1.90999620e+00,
  8.50962943e-01],
[-1.15980110e+00, -1.60076528e+00, -9.87791059e-01,
 -3.67600054e-01, -5.31494003e-01, -1.09863603e+00,
  8.50962943e-01],
[-3.57170580e-01, -9.35166408e-01, -9.87791059e-01,
 -8.64356883e-01,  0.00000000e+00, -3.54565690e-01,
  8.50962943e-01],
[ 9.27038248e-01,  1.06163019e+00,  7.76121546e-01,
```

```
  1.12267043e+00,  1.06298801e+00,  8.90943349e-01,
  8.50962943e-01],
[ 1.97045792e+00,  1.22802991e+00,  1.65807785e+00,
  6.25913605e-01,  5.31494003e-01,  1.68354001e+00,
  8.50962943e-01],
[-1.15980110e+00, -1.26796584e+00, -9.87791059e-01,
 -3.67600054e-01, -1.59448201e+00, -5.48670995e-01,
 -1.17513930e+00],
[ 1.72966877e+00,  1.89362878e+00,  7.76121546e-01,
  1.12267043e+00,  1.06298801e+00,  1.76441722e+00,
  8.50962943e-01],
[ 9.27038248e-01,  7.28830760e-01,  7.76121546e-01,
  6.25913605e-01,  1.06298801e+00,  7.13013486e-01,
  8.50962943e-01],
[ 9.27038248e-01, -1.03167824e-01,  7.76121546e-01,
  6.25913605e-01,  1.06298801e+00,  3.08627435e-01,
  8.50962943e-01],
[ 1.00730130e+00,  1.56082934e+00,  1.65807785e+00,
  1.61942726e+00,  1.59448201e+00,  1.52178559e+00,
  8.50962943e-01],
[ 4.45459937e-01,  7.28830760e-01,  1.65807785e+00,
  1.61942726e+00,  1.59448201e+00,  1.44090838e+00,
  8.50962943e-01],
[-1.32032720e+00, -1.10156613e+00, -9.87791059e-01,
 -3.67600054e-01,  0.00000000e+00, -1.09863603e+00,
 -1.17513930e+00],
[-9.99274994e-01, -2.69567541e-01,  1.65807785e+00,
  1.61942726e+00,  1.06298801e+00,  1.46873014e-01,
 -1.17513930e+00],
[-1.15980110e+00, -1.76716499e+00, -1.86974736e+00,
 -3.67600054e-01,  5.31494003e-01, -1.61625017e+00,
 -1.17513930e+00],
[ 1.64940571e+00,  1.89362878e+00,  1.65807785e+00,
  1.12267043e+00,  1.59448201e+00,  1.57031192e+00,
  8.50962943e-01],
[ 3.65196886e-01, -6.02366975e-01, -1.05834756e-01,
 -3.67600054e-01, -5.31494003e-01, -1.38979398e+00,
 -1.17513930e+00],
[ 1.89019487e+00,  1.56082934e+00,  7.76121546e-01,
  6.25913605e-01,  0.00000000e+00,  2.00704885e+00,
  8.50962943e-01],
[ 2.04670782e-01,  5.62431043e-01, -1.05834756e-01,
  6.25913605e-01, -5.31494003e-01,  3.89504645e-01,
 -1.17513930e+00],
[-1.40059025e+00, -2.69567541e-01, -1.05834756e-01,
  1.29156776e-01,  5.31494003e-01, -3.10568488e-02,
 -1.17513930e+00],
```

```
[-1.56111636e+00, -1.93356471e+00, -9.87791059e-01,
 -3.67600054e-01, -1.59448201e+00, -1.64860106e+00,
  8.50962943e-01],
[-1.15980110e+00, -1.26796584e+00, -1.05834756e-01,
 -8.64356883e-01, -1.59448201e+00, -1.76635827e-01,
  8.50962943e-01],
[-5.97959736e-01, -6.02366975e-01, -9.87791059e-01,
 -8.64356883e-01,  5.31494003e-01, -3.22214806e-01,
  8.50962943e-01],
[-8.38748891e-01, -7.68766692e-01, -9.87791059e-01,
 -1.36111371e+00, -1.06298801e+00, -6.13372763e-01,
 -1.17513930e+00],
[-9.19011943e-01, -6.02366975e-01,  1.65807785e+00,
  1.61942726e+00, -5.31494003e-01, -1.03393426e+00,
 -1.17513930e+00],
[-1.72164246e+00, -2.26636414e+00, -1.86974736e+00,
 -1.85787054e+00, -1.59448201e+00, -1.93975901e+00,
 -1.17513930e+00],
[-6.78222787e-01,  2.29631609e-01, -9.87791059e-01,
  6.25913605e-01,  0.00000000e+00, -1.38979398e+00,
 -1.17513930e+00],
[ 6.86249093e-01,  1.06163019e+00,  7.76121546e-01,
  1.12267043e+00,  1.06298801e+00,  1.11739954e+00,
  8.50962943e-01],
[ 1.08756435e+00,  1.22802991e+00,  1.65807785e+00,
  6.25913605e-01,  1.59448201e+00,  1.19827675e+00,
  8.50962943e-01],
[-4.37433632e-01, -4.35967258e-01, -1.05834756e-01,
 -1.36111371e+00,  0.00000000e+00, -3.06039364e-01,
 -1.17513930e+00],
[ 2.04670782e-01, -1.03167824e-01, -1.05834756e-01,
 -1.36111371e+00, -5.31494003e-01,  1.46873014e-01,
 -1.17513930e+00],
[-9.99274994e-01, -1.10156613e+00, -9.87791059e-01,
 -3.67600054e-01,  0.00000000e+00, -8.07478068e-01,
  8.50962943e-01],
[ 8.46775196e-01,  5.62431043e-01, -1.05834756e-01,
  1.29156776e-01,  0.00000000e+00,  3.24802877e-01,
  8.50962943e-01],
[ 3.65196886e-01, -9.35166408e-01, -9.87791059e-01,
 -8.64356883e-01, -5.31494003e-01,  9.83466878e-02,
 -1.17513930e+00],
[-6.78222787e-01, -2.69567541e-01, -9.87791059e-01,
 -8.64356883e-01, -5.31494003e-01, -1.47067119e+00,
 -1.17513930e+00],
[-1.24006415e+00, -1.26796584e+00, -1.86974736e+00,
 -2.35462737e+00, -1.06298801e+00, -8.88355278e-01,
```

```
 -1.17513930e+00],
[-1.96644477e-01, -1.03167824e-01, -9.87791059e-01,
 -8.64356883e-01, -1.59448201e+00, -2.08986712e-01,
 -1.17513930e+00],
[ 9.27038248e-01,  1.06163019e+00,  7.76121546e-01,
  1.12267043e+00,  1.59448201e+00,  9.39469675e-01,
 -1.17513930e+00],
[ 1.00730130e+00,  1.39442963e+00,  7.76121546e-01,
  1.12267043e+00,  5.31494003e-01,  9.71820559e-01,
  8.50962943e-01],
[-8.38748891e-01, -7.68766692e-01, -9.87791059e-01,
 -1.85787054e+00, -1.06298801e+00, -1.48684663e+00,
 -1.17513930e+00],
[-8.38748891e-01, -1.76716499e+00,  7.76121546e-01,
 -3.67600054e-01,  1.06298801e+00, -4.83969227e-01,
 -1.17513930e+00],
[ 8.46775196e-01,  2.29631609e-01, -1.05834756e-01,
 -3.67600054e-01,  0.00000000e+00,  5.35083624e-01,
 -1.17513930e+00],
[ 3.65196886e-01,  5.62431043e-01,  1.65807785e+00,
  1.61942726e+00,  1.06298801e+00,  1.06887321e+00,
  8.50962943e-01],
[-5.17696684e-01, -2.69567541e-01,  7.76121546e-01,
  1.29156776e-01, -1.59448201e+00, -6.13372763e-01,
 -1.17513930e+00],
[ 5.25722989e-01,  5.62431043e-01,  7.76121546e-01,
  6.25913605e-01,  1.59448201e+00,  9.23294233e-01,
  8.50962943e-01],
[ 6.05986041e-01, -4.35967258e-01, -1.05834756e-01,
  6.25913605e-01,  5.31494003e-01, -1.92811269e-01,
  8.50962943e-01],
[ 2.04670782e-01, -1.03167824e-01, -9.87791059e-01,
  6.25913605e-01,  5.31494003e-01, -1.03393426e+00,
  8.50962943e-01],
[-1.24006415e+00, -4.35967258e-01, -1.05834756e-01,
  1.29156776e-01, -5.31494003e-01, -6.45723647e-01,
 -1.17513930e+00],
[ 4.45459937e-01,  5.62431043e-01,  7.76121546e-01,
  1.29156776e-01,  5.31494003e-01, -3.38390248e-01,
  8.50962943e-01],
[-3.57170580e-01,  6.32318924e-02,  7.76121546e-01,
  1.12267043e+00,  1.06298801e+00,  7.13013486e-01,
  8.50962943e-01],
[-5.97959736e-01, -6.02366975e-01, -9.87791059e-01,
 -3.67600054e-01,  0.00000000e+00, -1.11934059e-01,
 -1.17513930e+00],
[ 2.84933834e-01, -2.69567541e-01, -1.05834756e-01,
```

```
 -3.67600054e-01,  0.00000000e+00,  1.79223898e-01,
  8.50962943e-01],
[ 9.27038248e-01,  1.56082934e+00,  7.76121546e-01,
  6.25913605e-01,  1.06298801e+00,  1.48943471e+00,
  8.50962943e-01],
[ 1.80993182e+00,  1.39442963e+00,  1.65807785e+00,
  1.12267043e+00,  1.59448201e+00,  1.08504865e+00,
  8.50962943e-01],
[-3.61183733e-02, -4.35967258e-01, -1.05834756e-01,
 -3.67600054e-01, -1.06298801e+00, -3.70741132e-01,
 -1.17513930e+00],
[ 3.65196886e-01,  8.95230476e-01, -9.87791059e-01,
  1.29156776e-01,  0.00000000e+00,  3.57153761e-01,
  8.50962943e-01],
[-1.32032720e+00, -2.09996443e+00, -1.86974736e+00,
 -2.35462737e+00, -2.65747002e+00, -1.97210990e+00,
 -1.17513930e+00],
[-3.61183733e-02, -1.03167824e-01, -1.05834756e-01,
  1.12267043e+00,  0.00000000e+00, -2.25162154e-01,
 -1.17513930e+00],
[ 1.08756435e+00,  1.22802991e+00, -9.87791059e-01,
 -1.36111371e+00,  5.31494003e-01,  1.29403537e-03,
  8.50962943e-01],
[ 2.04670782e-01,  5.62431043e-01, -1.86974736e+00,
 -8.64356883e-01,  0.00000000e+00, -3.10568488e-02,
  8.50962943e-01],
[-1.16381425e-01, -2.69567541e-01, -1.05834756e-01,
  1.29156776e-01, -1.06298801e+00, -4.19267458e-01,
 -1.17513930e+00]])
```

[25]:
```python
#Importing LogisticRegression model
from sklearn.linear_model import LogisticRegression
```

MODEL TRAINING AND EVALUATION

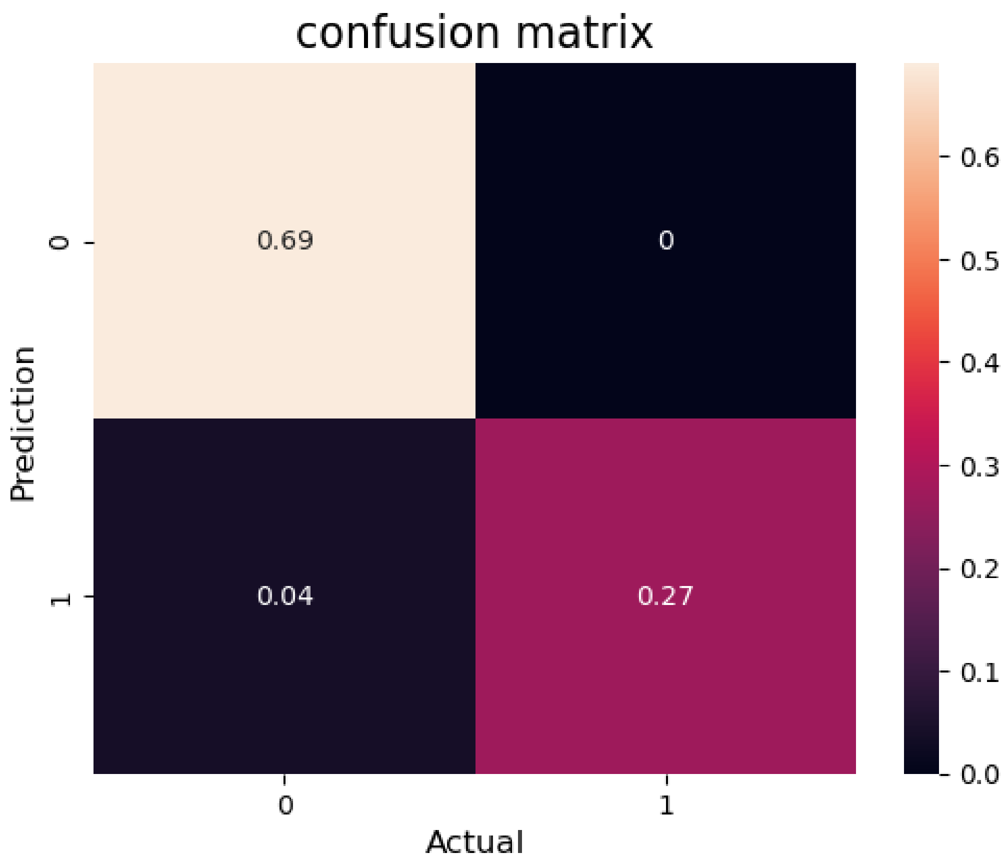
[26]:
```python
logr=LogisticRegression()
logr.fit(X_train,y_train)
y_pred1=logr.predict(X_test)
print(accuracy_score(y_pred1,y_test))
```

```
0.96
```

[27]:
```python
matrix=confusion_matrix(y_pred1,y_test)
```

[28]:
```python
sns.heatmap(matrix/np.sum(matrix),
            fmt='g',
            annot= True)
plt.ylabel('Prediction', fontsize = 12)
```

```python
plt.xlabel('Actual', fontsize = 12)
plt.title('confusion matrix',fontsize=16)
plt.show()
```



[29]:
```python
#Importing SupportVectorClassifier model
from sklearn import svm
```

MODEL TRAINING AND EVALUATION

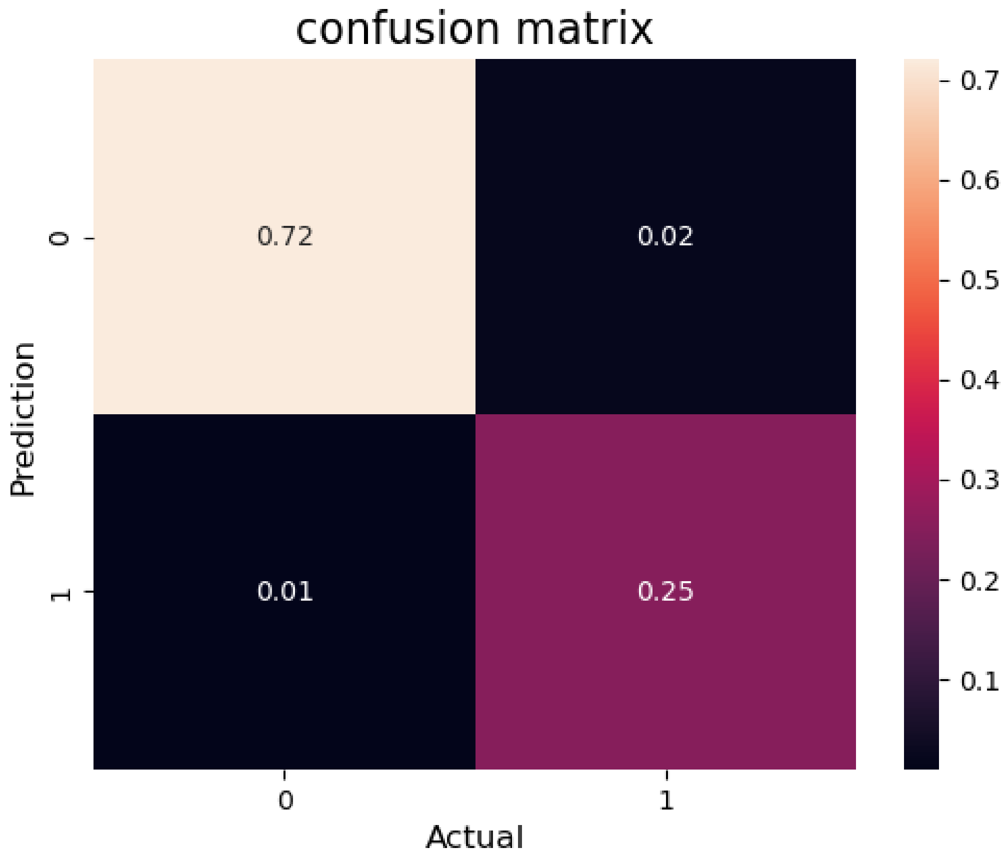
[30]:
```python
svm=svm.SVC()
svm.fit(X_train,y_train)
y_pred2=svm.predict(X_test)
print(accuracy_score(y_pred2,y_test))
```

```
0.97
```

[31]:
```python
matrix=confusion_matrix(y_pred2,y_test)
```

[32]:
```python
sns.heatmap(matrix/np.sum(matrix),
            fmt='g',
```

```
            annot= True)
plt.ylabel('Prediction', fontsize = 12)
plt.xlabel('Actual', fontsize = 12)
plt.title('confusion matrix',fontsize=16)
plt.show()
```



[33]:
```
##Importing KNN model
from sklearn.neighbors import KNeighborsClassifier
```

MODEL TRAINING AND EVALUATION

[34]:
```
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
y_pred3=knn.predict(X_test)
print(accuracy_score(y_pred3,y_test))
```

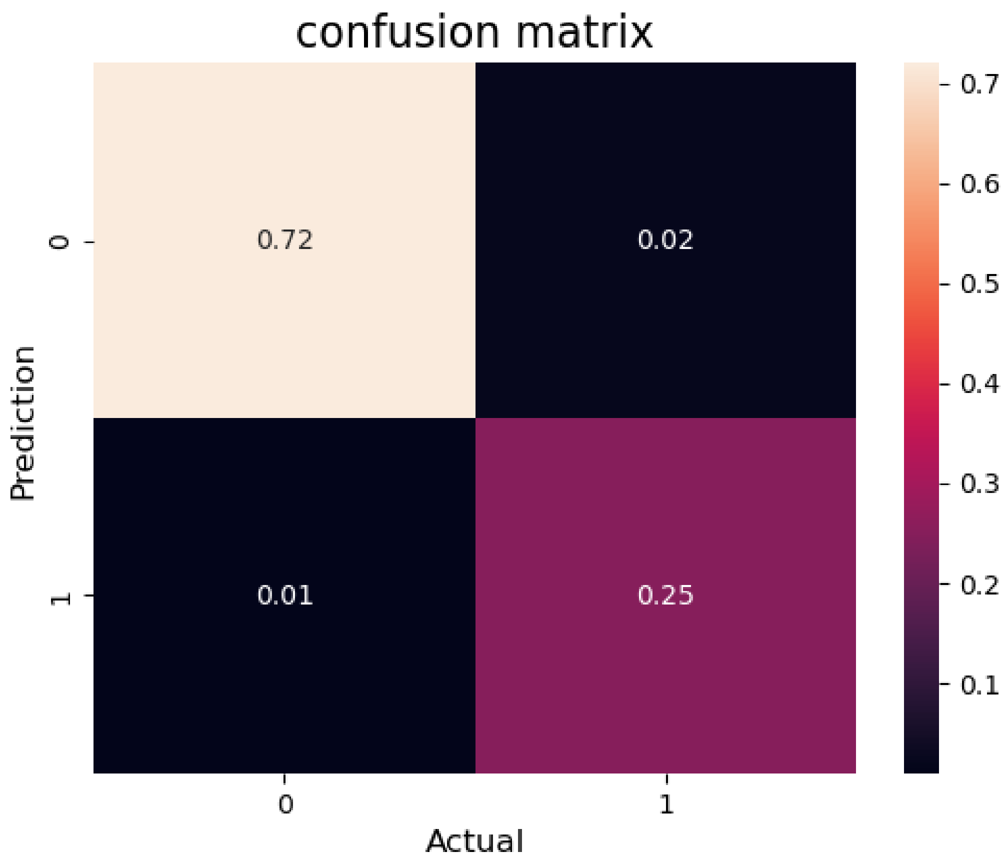
```
0.97
```

```
C:\Users\Varshini\anaconda3\lib\site-
packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other
reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode`
```

```
typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will
change: the default value of `keepdims` will become False, the `axis` over which
the statistic is taken will be eliminated, and the value None will no longer be
accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

[35]:
```
matrix=confusion_matrix(y_pred3,y_test)
```

[36]:
```
sns.heatmap(matrix/np.sum(matrix),
            fmt='g',
            annot= True)
plt.ylabel('Prediction', fontsize = 12)
plt.xlabel('Actual', fontsize = 12)
plt.title('confusion matrix',fontsize=16)
plt.show()
```



[37]:
```
#Importing RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
```

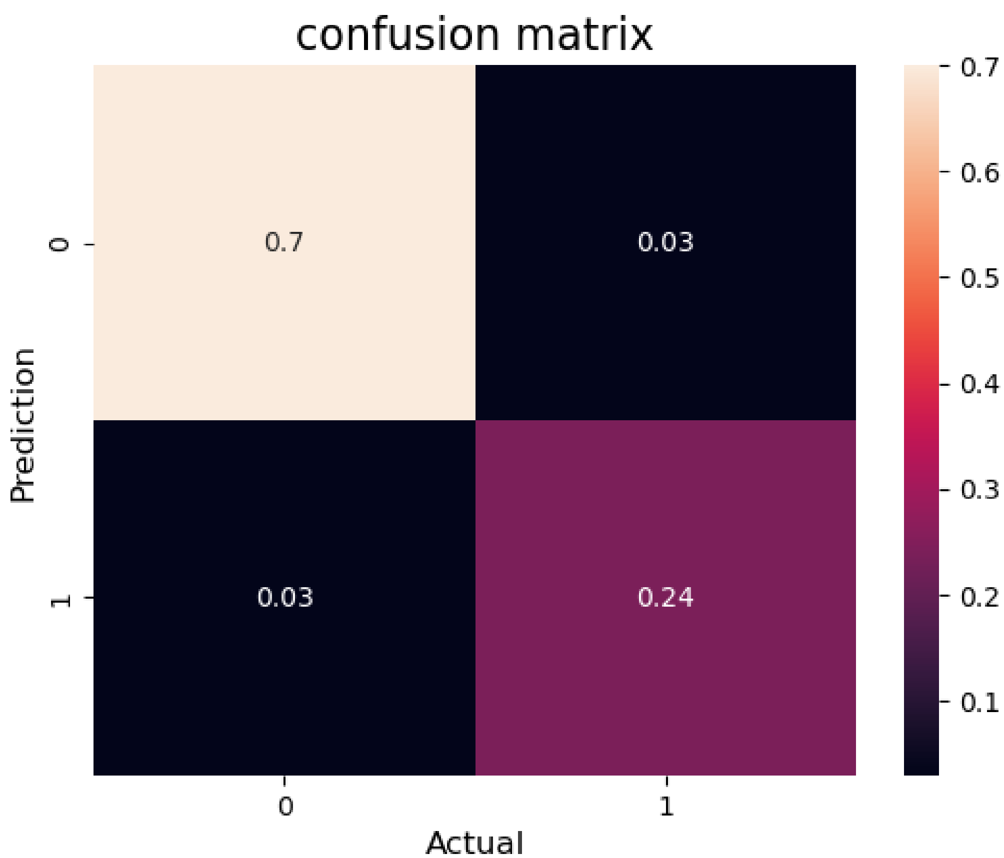
MODEL TRAINING AND EVALUATION

[38]:
```
RDF=RandomForestClassifier()
RDF.fit(X_train,y_train)
y_pred4=RDF.predict(X_test)
print(accuracy_score(y_pred4,y_test))
```

```
0.94
```

[39]:
```
matrix=confusion_matrix(y_pred4,y_test)
```

[40]:
```
sns.heatmap(matrix/np.sum(matrix),
            fmt='g',
            annot= True)
plt.ylabel('Prediction', fontsize = 12)
plt.xlabel('Actual', fontsize = 12)
plt.title('confusion matrix',fontsize=16)
plt.show()
```



[41]:
```
final_output=pd.DataFrame({'Model':['LR','SVC','KNN','RDF'],'ACCURACY_SCORE':
    [accuracy_score(y_pred1,y_test),
```

```
↪    accuracy_score(y_pred2,y_test),
↪    accuracy_score(y_pred3,y_test),
↪    accuracy_score(y_pred4,y_test)]})
```

[42]:
```
final_output
```

[42]:

	Model	ACCURACY_SCORE
0	LR	0.96
1	SVC	0.97
2	KNN	0.97
3	RDF	0.94

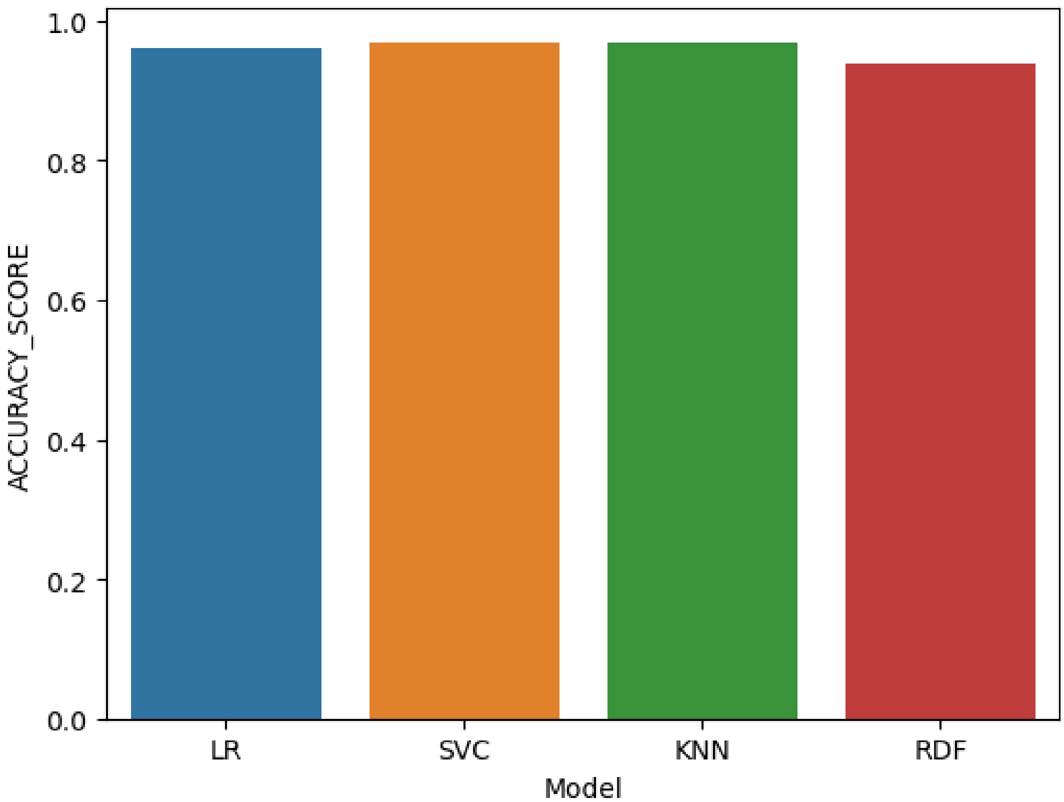
ACCURACY BARPLOT

Accuracy is a common metric used in machine learning and statistics to measure the performance of a classification model. It provides a straightforward way to understand how well the model is predicting the correct classes compared to the total number of instances

[43]:
```
sns.barplot(final_output['Model'],final_output['ACCURACY_SCORE'])
```

```
C:\Users\Varshini\anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variables as keyword args: x, y. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[43]:
```
<AxesSubplot:xlabel='Model', ylabel='ACCURACY_SCORE'>
```



Hence, SVM and KNearestNeighbours has more accuracy than other models, while comparing to the confusion matrix it has less **FN** values according to the classification model.