**Name : Mohit Manish Bhavsar**

**Roll No : 20U437**

**Div : 4**

import pandas as pd

df=pd.read\_csv('https://raw.githubusercontent.com/shivang98/Social-Network-ads-Boost/maste df.sample(15)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **User ID** | **Gender** | **Age** | **EstimatedSalary** | **Purchased** |
| **336** | 15664907 | Male | 58 | 144000 | 1 |
| **235** | 15646227 | Male | 46 | 79000 | 1 |
| **311** | 15622585 | Male | 39 | 96000 | 1 |
| **204** | 15660866 | Female | 58 | 101000 | 1 |
| **389** | 15668521 | Female | 48 | 35000 | 1 |
| **325** | 15695679 | Female | 41 | 60000 | 0 |
| **379** | 15749381 | Female | 58 | 23000 | 1 |
| **275** | 15727467 | Male | 57 | 74000 | 1 |
| **126** | 15610801 | Male | 42 | 65000 | 0 |
| **279** | 15759684 | Female | 50 | 36000 | 1 |
| **287** | 15761950 | Female | 48 | 138000 | 1 |
| **61** | 15673619 | Male | 25 | 87000 | 0 |
| **269** | 15583137 | Male | 40 | 61000 | 0 |
| **398** | 15755018 | Male | 36 | 33000 | 0 |
| **158** | 15762605 | Male | 26 | 30000 | 0 |

df.drop(columns=['User ID'],inplace=True) df.sample(15)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Gender** | **Age** | **EstimatedSalary** | **Purchased** |
| **95** | Female | 35 | 44000 | 0 |
| **87** | Female | 28 | 85000 | 0 |
| **279** | Female | 50 | 36000 | 1 |
| **26** | Male | 49 | 28000 | 1 |
| df.dtypes |  |  |  |  |

Gender object

**249** Female 35 97000 1

**88** Male 26 81000 0

**262** Female 55 125000 1

**161** Male 25 90000 0

**394** Female 39 59000 0

Age int64

EstimatedSalary int64 Purchased int64 dtype: object

df['Gender']=df['Gender'].astype('category') df.dtypes

**184** Female 33 60000 0

**293** Male 37 77000 0

**341** Male 35 75000 0

**141** Female 18 68000 0

**129** Female 26 84000 0

**192** Male 29 43000 0

Gender category Age int64

EstimatedSalary int64

Purchased int64 dtype: object

df['Gender']=df['Gender'].cat.codes df.sample(10)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Gender** | **Age** | **EstimatedSalary** | **Purchased** |
| **100** | 1 | 27 | 88000 | 0 |
| **182** | 0 | 32 | 117000 | 1 |
| **329** | 0 | 47 | 107000 | 1 |
| **88** | 1 | 26 | 81000 | 0 |
| **91** | 0 | 30 | 116000 | 0 |
| **151** | 1 | 41 | 45000 | 0 |
| **278** | 0 | 52 | 38000 | 1 |
| **80** | 1 | 30 | 80000 | 0 |
| **67** | 0 | 23 | 82000 | 0 |
| **383** | 1 | 49 | 28000 | 1 |

df['Gender'].value\_counts() 0 204

1 196

Name: Gender, dtype: int64

def DetectOutlier(df,var):

high, low = df[var].mean() + 3\* df[var].std() , df[var].mean() - 3\* df[var].std()

print("Highest allowed in variable:", var, high) print("lowest allowed in variable:", var, low)

count = df[(df[var] > high) | (df[var] < low)][var].count() print('Total outliers in:',var,':',count)

DetectOutlier(df,'Age')

Highest allowed in variable: Age 69.10362979192377 lowest allowed in variable: Age 6.206370208076244 Total outliers in: Age : 0

DetectOutlier(df,'EstimatedSalary')

Highest allowed in variable: EstimatedSalary 172033.38084727435 lowest allowed in variable: EstimatedSalary -32548.380847274355 Total outliers in: EstimatedSalary : 0

df.isna().sum()

Gender 0

Age 0

EstimatedSalary 0

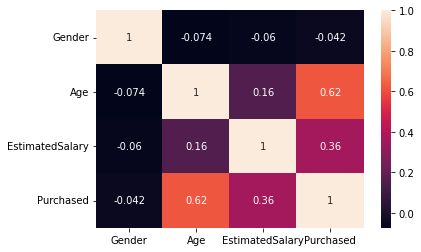
Purchased 0

dtype: int64

import seaborn as sns

sns.heatmap(df.corr(),annot=True)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f3464353d10>



x=df[['Age','EstimatedSalary']] y=df['Purchased']

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

from sklearn.linear\_model import LogisticRegression model=LogisticRegression()

model.fit(x\_train,y\_train)

print ('Model Score:',model.score(x\_test,y\_test))

Model Score: 0.65

x=df[['Age','Gender','EstimatedSalary']] y=df['Purchased']

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

from sklearn.linear\_model import LogisticRegression model=LogisticRegression()

model.fit(x\_train,y\_train)

print ('Model Score:',model.score(x\_test,y\_test)) Model Score: 0.65

df.describe()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Gender** | **Age** | **EstimatedSalary** | **Purchased** |
| **count** | 400.000000 | 400.000000 | 400.000000 | 400.000000 |
| **mean** | 0.490000 | 37.655000 | 69742.500000 | 0.357500 |
| **std** | 0.500526 | 10.482877 | 34096.960282 | 0.479864 |
| **min** | 0.000000 | 18.000000 | 15000.000000 | 0.000000 |
| **25%** | 0.000000 | 29.750000 | 43000.000000 | 0.000000 |
| **50%** | 0.000000 | 37.000000 | 70000.000000 | 0.000000 |
| **75%** | 1.000000 | 46.000000 | 88000.000000 | 1.000000 |
| **max** | 1.000000 | 60.000000 | 150000.000000 | 1.000000 |

x=df[['Age','EstimatedSalary']] y=df['Purchased']

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

from sklearn.preprocessing import MinMaxScaler norm=MinMaxScaler().fit(x\_train)

x\_train=norm.transform(x\_train) norm=MinMaxScaler().fit(x\_test) x\_test=norm.transform(x\_test)

from sklearn.linear\_model import LogisticRegression model=LogisticRegression()

model.fit(x\_train,y\_train)

print ('Model Score:',model.score(x\_test,y\_test))

Model Score: 0.875

from sklearn.linear\_model import LogisticRegression model=LogisticRegression()

model.fit(x\_train,y\_train)

y\_pred=model.predict(x\_test)

print('model score:',model.score(x\_test,y\_test)) from sklearn.metrics import confusion\_matrix

cf\_matrix=confusion\_matrix(y\_test,y\_pred)#actual o/p and predicted output print(cf\_matrix)

model score: 0.875

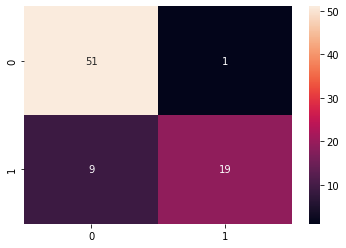
[[51 1]

[ 9 19]]

import seaborn as sns

sns.heatmap(cf\_matrix,annot=True)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f34619b0810>



from sklearn.metrics import precision\_recall\_fscore\_support

precision\_recall\_fscore\_support(y\_test,y\_pred,average='macro')

(0.8999999999999999, 0.8296703296703296, 0.8511904761904763, None)

precision\_recall\_fscore\_support(y\_test,y\_pred,average='micro') (0.875, 0.875, 0.875, None)

precision\_recall\_fscore\_support(y\_test,y\_pred,average='weighted') (0.885, 0.875, 0.869047619047619, None)

from sklearn.metrics import precision\_recall\_fscore\_support

score=precision\_recall\_fscore\_support(y\_test,y\_pred,average='micro') print('Precision of Model:',score[0])

print('Recall of Model:',score[1]) print('F-score of Model:',score[2])

Precision of Model: 0.875 Recall of Model: 0.875

F-score of Model: 0.875

