import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

df= pd.read\_csv('Social\_Network\_Ads.csv')

df.shape

df.head()

#drop the column UserId

df.drop(['User ID'],axis=1,inplace=True)

df.head()

df.Purchased.value\_counts()

df.Gender.value\_counts()

df.dtypes

#data preprocessing

df.isnull().sum()

df.describe()

g = sns.catplot(x = "Gender",y = "Purchased",data = df,kind = "bar",height = 4)

g.set\_ylabels("Purchased Probability")

plt.show

M2 = pd.crosstab(df.Gender, df.Purchased, normalize='index')

print(M2)

M2.plot.bar(figsize=(6,4),stacked=True)

plt.legend(title='Gender vs Purchased', loc='upper right')

plt.show()

X=df.drop(['Gender','Purchased'],axis=1)

Y= df['Purchased']

X.head()

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

# Shuffle and split the data into training and testing subsets

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

# Success

print("Training and testing split was successful.")

#building the model

from sklearn.linear\_model import LogisticRegression

basemodel= LogisticRegression()

basemodel.fit(X\_train,y\_train)

print("Training accuracy:", basemodel.score(X\_train,y\_train)\*100)

#Make predictions on test data

y\_predict= basemodel.predict(X\_test)

print("Testing accuracy:", basemodel.score(X\_test,y\_test)\*100)

#Normalize the data using Min Max Normalization or any other technique

from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler()

X=df[['Age','EstimatedSalary']]

X\_scaled= scaler.fit\_transform(X)

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

print("Training and testing split was successful.")

model= LogisticRegression()

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

y\_predict= model.predict(X\_test)

print("Training accuracy:", model.score(X\_train,y\_train)\*100)

print("Testing accuracy:", model.score(X\_test,y\_test)\*100)

#Measure the performance of the model

from sklearn.metrics import accuracy\_score

Acc=accuracy\_score(y\_test,y\_predict)

print(Acc)

from sklearn.metrics import confusion\_matrix

cm= confusion\_matrix(y\_test,y\_predict)

print(cm)

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

prf= precision\_recall\_fscore\_support(y\_test,y\_predict)

print('precision:',prf[0])

print('Recall:',prf[1])

print('fscore:',prf[2])

print('support:',prf[3])

from sklearn.metrics import classification\_report

cr= classification\_report(y\_test,y\_predict)

print(cr)