ASSIGNMENT 2

LOGISTIC REGRESSION

INTRODUCTION TO INTELLIGENT SYSTEMS

PROJECT REPORT

SUBMITTED BY:- ANANYA KANSAL
ROLL NO:- 2019458

LOGISTIC REGRESSION HAS MANY ASPECTS TO IT. FOLLOWING ARE THE STEPS THAT I FOLLOWED TO PREPARE MY MODEL.

1. COLLECTING DATA: THE DATA WAS ALREADY PROVIDED TO US IN AN EXCEL FILE. I EXTRACTED THE DATA IN EXCEL USING PANDAS LIBRARY TO LOAD IT INTO PYTHON AND PERFORM VARIOUS ACTIONS ON IT.

2. ANALYSING THE DATA: I WENT THROUGH THE VARIOUS COLUMNS AND HEADINGS WHICH WERE PROVIDED TO US. THEY ALL WERE THE INDEPENDENT VARIABLES THAT CONTRIBUTED TO THE OUTPUT ‘HIGH-SALARY’ WHICH WAS THE DEPENDENT VARIABLE.

3. WRANGLING THE DATA: IN IT I WAS BASICALLY EXPERIMENTING WITH THE DATA AS TO WHICH COLUMNS WERE USEFUL FOR THE OUTPUT AND ELIMINATING THE UNNECESSARY COLUMNS LIKE THE ID OF THE PERSON WHICH WOULDN’T CONTRIBUTE TO THE PERSON GETTING LOW SALARY OR HIGH SALARY.
   MOREOVER, THERE WERE SOME COLUMNS LIKE GENDER, SPECIALIZATION, EDUCATION BOARD ETC. IN WHICH THE DATA WAS STORED AS TYPE ‘STRING’ AND FOR LOGISTIC REGRESSION TO BE APPLIED ONTO THESE VALUES IT WAS NECESSARY TO CONVERT THEM INTO NUMERICAL VALUES WHICH I AGAIN DID USING PANDAS LIBRARY FUNCTIONS (USED ONE-HOT ENCODING).

4. TRAINING AND TESTING: THE DATA HAD TO BE DIVIDED INTO SETS- TRAINING AND TESTING. I EXPERIMENTED WITH THIS RATIO AND VARIED IT TO GET DIFFERENT VALUES OF ACCURACIES.

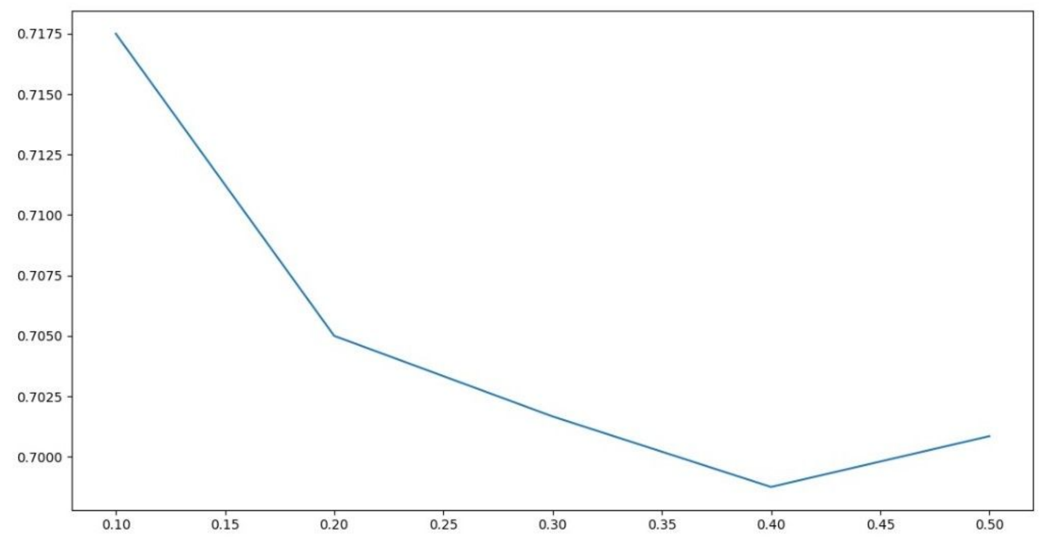

FIG 1.1.
THE GRAPH PLOTS THE ACCURACIES ON THE Y-AXIS WITH THE TRAIN-TEST RATIO ON THE X-AXIS WHEN THE DATA IS **NOT** BEING SHUFFLED.

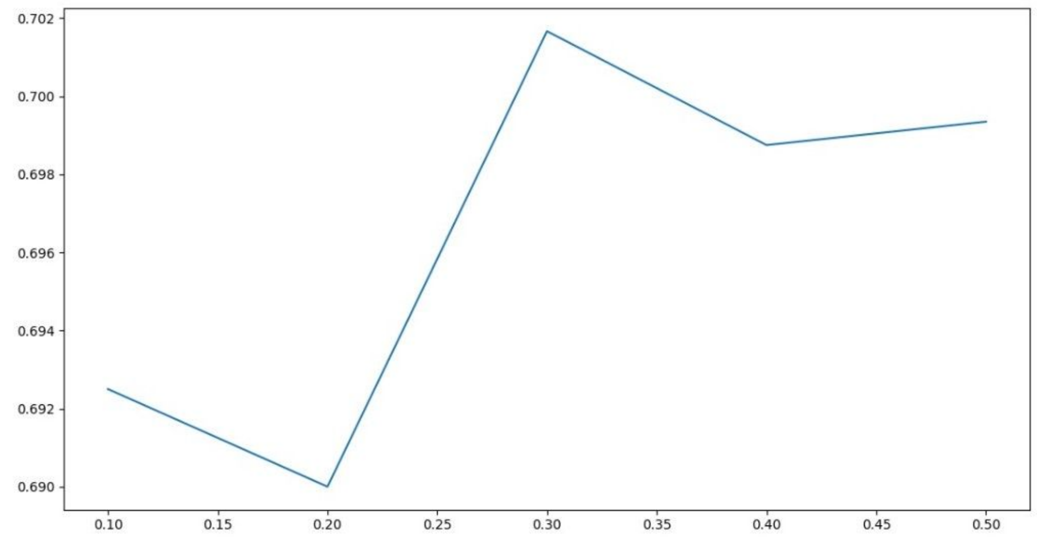

FIG 1.2.
THE GRAPH PLOTS THE ACCURACIES ON THE Y-AXIS WITH THE TRAIN-TEST RATIO ON THE X-AXIS WHEN THE DATA IS BEING **SHUFFLED**.

1. CONFUSION MATRICES OF **UN**SHUFFLED DATA :

- Ratio of train-test data = 0.1
```
[[132  54]
[ 59 155]]
```

- Ratio of train-test data = 0.2
```
[[247 128]
[108 317]]
```

- Ratio of train-test data = 0.3
```
[[366 205]
[153 476]]
```

- Ratio of train-test data = 0.4
```
[[493 268]
[214 625]]
```

- Ratio of train-test data = 0.5
```
[[615 329]
[269 786]]
```

2. CONFUSION MATRICES OF **SHUFFLED** DATA

- Ratio of train-test data = 0.1
```
[[115  67]
 [ 56 162]]
```

- Ratio of train-test data = 0.2
```
[[244 131]
 [117 308]]
```

- Ratio of train-test data = 0.3
```
[[358 187]
 [171 484]]
```

- Ratio of train-test data = 0.4
```
[[487 246]
 [236 631]]
```

- Ratio of train-test data = 0.5
```
[[619 310]
 [291 779]]
```

DROPPING SOME COLUMNS

I ALSO EXPERIMENTED BY DROPPING SOME OF THE COLUMNS WHICH WERE REDUNDANT IN THE DATA AND HELPED IN INCREASING THE ACCURACY.
INITIALLY I DROPPED DOB AND ID AND FOUND ALL THE ACCURACIES DEPICTED IN THE GRAPHS.
LATER I DROPPED SOME MORE COLUMNS LIKE COLLEGE-ID, COLLEGE-CITY AND COLLEGE STATE TO PROCURE **0.71** ACCURACY WITH THE FOLLOWING CONFUSION MATRIX :

```
[[118  64]
[ 52 166]]
```

RESULTS : A SLIGHT DECLINE IN ACCURACY WAS OBSERVED WHEN THE DATA WAS SHUFFLED AND A SLIGHT INCREASE WHEN SOME OF THE COLUMNS WERE DROPPED.
MAJORLY, THE ACCURACY WAS AROUND **70%**.
THE PEAK ACCURACY IN THE **UNSHUFFLED** DATA WAS OBSERVED FOR THE RATIO **0.1.**
THE PEAK ACCURACY IN THE **SHUFFLED** DATA WAS OBSERVED FOR THE RATIO **0.3.**

# PYTHON CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import scipy
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

#Preprocessing start
df_xlsx=pd.read_excel('Data.xlsx')
#print(df_xlsx["Logical"].plot.hist())
#print(df_xlsx.head(3))
gender=(pd.get_dummies(df_xlsx['Gender'],drop_first=True))
board10=(pd.get_dummies(df_xlsx['10board']))
board12=(pd.get_dummies(df_xlsx['12board']))
specialization=(pd.get_dummies(df_xlsx['Specialization']))
degree=(pd.get_dummies(df_xlsx['Degree']))
collegestate=(pd.get_dummies(df_xlsx['CollegeState']))
df_xlsx=pd.concat([df_xlsx,gender,board10,specialization,degree,collegestate,board12],
axis=1)
#print(df_xlsx.head(4))
df_xlsx.drop(['DOB','Gender','10board','CollegeState','Specialization','Degree','ID','12boar
d'],axis=1,inplace=True)
#print(df_xlsx.head(4))
X=df_xlsx.drop('High-Salary',axis=1)
y=df_xlsx['High-Salary']
#Preprocessing over
```

```
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.3,random_state=1)
logmodel=LogisticRegression(max_iter=10000)
logmodel.fit(X_train,y_train)
print(logmodel.score(X_test,y_test))

li=[0.1,0.2,0.3,0.4,0.5]
list1=[]
for cnt in range(0,5):
        X_train,X_test,y_train,y_test=
      train_test_split(X,y,test_size=li[cnt],random_state=1,shuffle=False)
        logmodel=LogisticRegression(max_iter=10000)
        logmodel.fit(X_train,y_train)
        list1.append(logmodel.score(X_test,y_test))
        print(confusion_matrix(y_test,logmodel.predict(X_test)))

xs = np.array(li)
ys = np.array(list1)
plt.plot(xs, ys)
plt.show()

list2=[]
for cnt in range(0,5):
  X_train,X_test,y_train,y_test=
train_test_split(X,y,test_size=li[cnt],random_state=1,shuffle=True)
  logmodel=LogisticRegression(max_iter=10000)
  logmodel.fit(X_train,y_train)
  list2.append(logmodel.score(X_test,y_test))
      print(confusion_matrix(y_test,logmodel.predict(X_test)))

xs = np.array(li)
ys = np.array(list2)
plt.plot(xs, ys)
plt.show()

df_xlsx.drop(['CollegeID','CollegeCityID','collegestate'])
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.1,random_state=1)
logmodel=LogisticRegression(max_iter=10000)
logmodel.fit(X_train,y_train,shuffle=True)
print(logmodel.score(X_test,y_test))
print(confusion_matrix(y_test,logmodel.predict(X_test)))
```