REPORT ASSIGNMENT 3 - ROUNAK SARAF

Steps for Code (Includes modifications and preparations) -

Step 1 - Importing the header files

Step 2 - Manipulating the columns which contained strings as their data. This was done so as to fit the data into the model. This has been done for the columns - ‘Gender’, ‘DOB’, ‘10board’, ‘12board’, ‘Degree’, ‘Specialization’, ‘CollegeState’.

Step 3 - All possible combinations of features of length 31,32,33 features were generated using a function. This was done so as to test which possible combination would give the maximum accuracy.

Step 4 - The logistic regression model was applied to these combinations and the accuracy computed was stored in a dictionary. It was of the form - [feature_combination_index:accuracy].

Step 5 - The top 3 accuracies predicted and the bottom 3 accuracies were predicted and their classification report containing accuracy, class-wise accuracies,confusion matrix, precision, recall, f1 score, macro avg, weighted average were calculated. This was done so that further analysis could be done on these feature combinations and methods used.

Experiments -

This was done with test sizes of 5%,10%,15%,20%,25%,30%,40%. Also, shuffling was done by varying the random state which gave the **best result of 79% accuracy** at random_state of 40, with test_size of 10%. This has been reported in the reports.

Modifications -

1 - Standard Scalar operation was applied on the training data. This was done because all the features had different ranges. If we don't scale the features, then some features will dominate the others when the model finds the nearest neighbor to a data point in data space.

2 - The feature columns including strings as data were also manipulated so they could fit into the model.

Analysis of the program -

The feature selection is done in such a way that it model finds the nearest neighbor to a data point in data space. By performing logistic regression in all possible combinations of features, we saw that there were certain features which provided better accuracy, time taken to train and results as compared to other features.

The maximum accuracy achieved without shuffling was 75.89% and the features that provided so were -

['ID', 'Gender', 'DOB', '10board', '12graduation', '12percentage', '12board', 'CollegeID', 'CollegeTier', 'Degree', 'Specialization', 'CollegeCityID', 'CollegeCityTier', 'CollegeState', 'GraduationYear', 'English', 'Logical', 'Quant', 'Domain', 'ComputerProgramming', 'ElectronicsAndSemicon', 'ComputerScience', 'MechanicalEngg', 'ElectricalEngg', 'TelecomEngg', 'CivilEngg', 'conscientiousness', 'agreeableness', 'extraversion', 'neuroticism', 'openess_to_experience']

The maximum accuracy achieved using shuffling was 79% and the features that provided so were -

['ID', 'Gender', 'DOB', '10board','10percentage', '12graduation', '12percentage', '12board', 'CollegeID', 'CollegeTier', 'Degree', 'Specialization', 'CollegeCityID', 'CollegeCityTier', 'GraduationYear', 'English', 'Logical', 'Quant', 'Domain', 'ComputerProgramming', 'ElectronicsAndSemicon', 'ComputerScience', 'MechanicalEngg',

'ElectricalEngg', 'TelecomEngg', 'CivilEngg', 'conscientiousness',
'agreeableness', 'extraversion', 'neuroticism', 'openess_to_experience']

Listing -

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from itertools import combinations

feature_dictionary={0:'ID',1:'Gender',2:'DOB',3:'10percentage',4:'10board',5
:'12graduation',6:'12percentage',7:'12board',8:'CollegeID',9:'CollegeTier',10:
'Degree',11:'Specialization',12:'collegeGPA',13:'CollegeCityID',14:'CollegeC
ityTier',15:'CollegeState',16:'GraduationYear',17:'English',18:'Logical',19:'Q
uant',20:'Domain',21:'ComputerProgramming',22:'ElectronicsAndSemicon',
23:'ComputerScience',24:'MechanicalEngg',25:'ElectricalEngg',26:'Telecom
Engg',27:'CivilEngg',28:'conscientiousness',29:'agreeableness',30:'extraver
sion',31:'nueroticism',32:'openess_to_experience',33:'High-Salary'}

def logisticregressionfunc(ll):
    x=file_data.iloc[:,ll].values
    y=file_data.iloc[:,33].values

    xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.15)

    sc_x=StandardScaler()
    xtrain=sc_x.fit_transform(xtrain)
    xtest=sc_x.transform(xtest)
```

```
    classifier=LogisticRegression(random_state=40)
    classifier.fit(xtrain,ytrain)

    y_pred=classifier.predict(xtest)

    cm=confusion_matrix(ytest, y_pred)
    print ("Accuracy : ", accuracy_score(ytest, y_pred))
    print ("Confusion Matrix : \n", cm)
    print("Classification Report: ")
    print(classification_report(ytest, y_pred,))

    print("The feature selection is: ")
    feature_list=[]
    for ele in ll:
         feature_list.append(feature_dictionary[ele])
    print(feature_list)

    return accuracy_score(ytest,y_pred)

from itertools import combinations
def func(n,k):
        def dfs(offset, chosen, ret):
        nonlocal k
        if len(chosen) == k:
        ret.append(chosen.copy())
        return

        needed = k - len(chosen)
        for i in range(offset, n - needed + 2):
        chosen.append(i)
        dfs(i + 1, chosen, ret)
        chosen.pop()
        return

        result = []
```

```
        dfs(1, [], result)
        return result

file_data=pd.read_csv('Rounak Saraf - Data for AI-Assignment-3.csv')

gender_col=file_data.iloc[:,1].values

for i in range(len(gender_col)):
    gender=gender_col[i]
    if gender=='f':
        gender_col[i]=1
    else:
        gender_col[i]=2

tenboard_col=file_data.iloc[:,4].values

tenboard_col_set=set()
tenboard_col_set_dic={}
for i in range(len(tenboard_col)):
    tenboard=tenboard_col[i]
    tenboard_col_set.add(tenboard)
    if tenboard not in tenboard_col_set_dic.keys():
        tenboard_col_set_dic[tenboard]=len(tenboard_col_set)

for i in range(len(tenboard_col)):
    tenboard=tenboard_col[i]
    tenboard_col[i]=tenboard_col_set_dic[tenboard]

twelveboard_col=file_data.iloc[:,7].values

twelveboard_col_set=set()
twelveboard_col_set_dic={}
for i in range(len(tenboard_col)):
    twelveboard=twelveboard_col[i]
    twelveboard_col_set.add(twelveboard)
```

```
    if twelveboard not in twelveboard_col_set_dic.keys():
        twelveboard_col_set_dic[twelveboard]=len(twelveboard_col_set)

for i in range(len(tenboard_col)):
    twelveboard=twelveboard_col[i]
    twelveboard_col[i]=twelveboard_col_set_dic[twelveboard]

degree_col=file_data.iloc[:,10].values

degree_col_set=set()
degree_col_set_dic={}
for i in range(len(degree_col)):
    degree=degree_col[i]
    degree_col_set.add(degree)
    if degree not in degree_col_set_dic.keys():
        degree_col_set_dic[degree]=len(degree_col_set)

for i in range(len(degree_col)):
    degree=degree_col[i]
    degree_col[i]=degree_col_set_dic[degree]

specialization_col=file_data.iloc[:,11].values

specialization_col_set=set()
specialization_col_set_dic={}
for i in range(len(specialization_col)):
    specialization=specialization_col[i]
    specialization_col_set.add(specialization)
    if specialization not in specialization_col_set_dic.keys():
        specialization_col_set_dic[specialization]=len(specialization_col_set)

for i in range(len(specialization_col)):
    specialization=specialization_col[i]
    specialization_col[i]=specialization_col_set_dic[specialization]
```

```
state_col=file_data.iloc[:,15].values

state_col_set=set()
state_col_set_dic={}
for i in range(len(state_col)):
    state=state_col[i]
    state_col_set.add(state)
    if state not in state_col_set_dic.keys():
        state_col_set_dic[state]=len(state_col_set)

for i in range(len(state_col)):
    state=state_col[i]
    state_col[i]=state_col_set_dic[state]

dob_col=file_data.iloc[:,2].values

dob_col_set=set()
dob_col_set_dic={}
for i in range(len(dob_col)):
    dob=dob_col[i]
    dob_col_set.add(dob)
    if dob not in dob_col_set_dic.keys():
        dob_col_set_dic[dob]=len(dob_col_set)

for i in range(len(dob_col)):
    dob=dob_col[i]
    dob_col[i]=dob_col_set_dic[dob]

feature_combinations=[]
for k in range(31,34):
    feature_combinations.extend(func(33,k))

for i in range(len(feature_combinations)):
    ll=feature_combinations[i]
    for j in range(len(ll)):
```

```
            ll[j]-=1

#print(feature_combinations)

feature_combination_accuracy_cm_dictionary={}
for i in range(len(feature_combinations)):
    feature_combination=feature_combinations[i]
    accuracy=logisticregressionfunc(feature_combination)
    feature_combination_accuracy_cm_dictionary[i]=accuracy

print(feature_combination_accuracy_cm_dictionary)

keys=feature_combination_accuracy_cm_dictionary.keys()
values=[]
for key in keys:
    values.append(feature_combination_accuracy_cm_dictionary[key])

values.sort()
values2=[]
for value in values:
    if value not in values2:
            values2.append(value)

values=values2
check_values=values[:3]+values[len(values)-3:]
print("Max and Min accuracies achieved: ")
print(check_values)

for key in keys:
    if (feature_combination_accuracy_cm_dictionary[key] in check_values):
            feature_combination=feature_combinations[key]
            logisticregressionfunc(feature_combination)
```