Assignment 4

ML Model For A Career Prediction System

Name: Mohammad Sufyan Azam
Roll No: 2020312

Steps Taken for Making And Analyzing The Model -

1. Imported all the required modules

2. Read data From CSV using pandas

3. Pre-processed data
    a. Clubbed multiple similar job roles into one category.
    b. Did label encoding on columns which had a string value for each row.

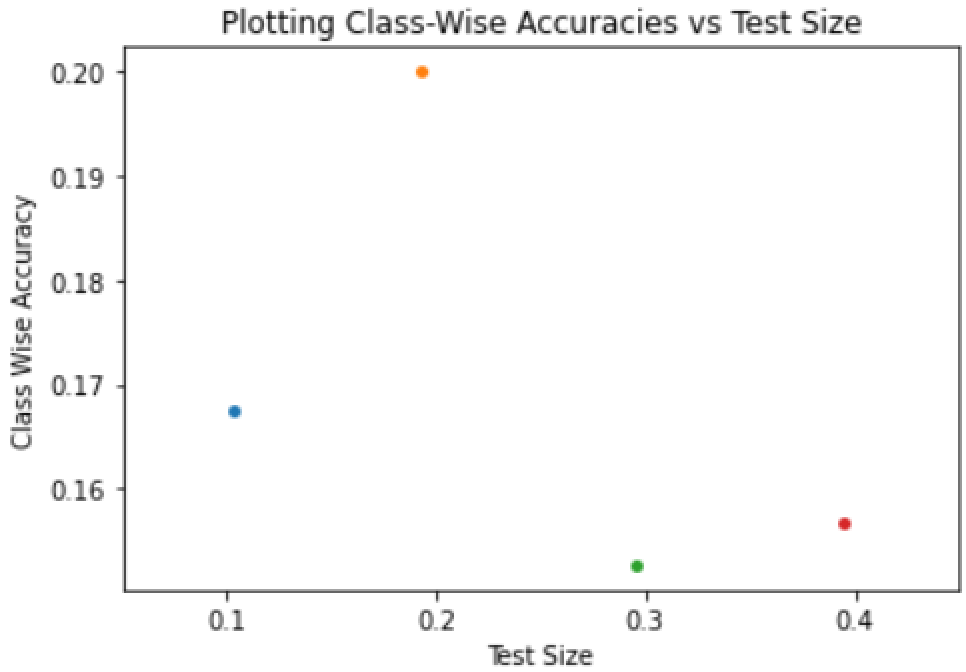
4. Prepared the data for training and testing
    a. Dropped the irrelevant columns like '*processing job role*' and stored it in variable X.
    b. Stored the new column '*processing job role*' in a variable Y for our model to predict.

5. Trained and tested our ML model on 90:10, 80:20, 70:30, 60:40 ratio splits
    a. Used `train_test_split()` to split the data into X_train and Y_train for our model.
    b. Used `MLPClassifier()` *to train our model.*
    c. Got the class-wise accuracy and the overall accuracy for each ratio of training data using `model.score()` and `accuracy_score()` respectively.

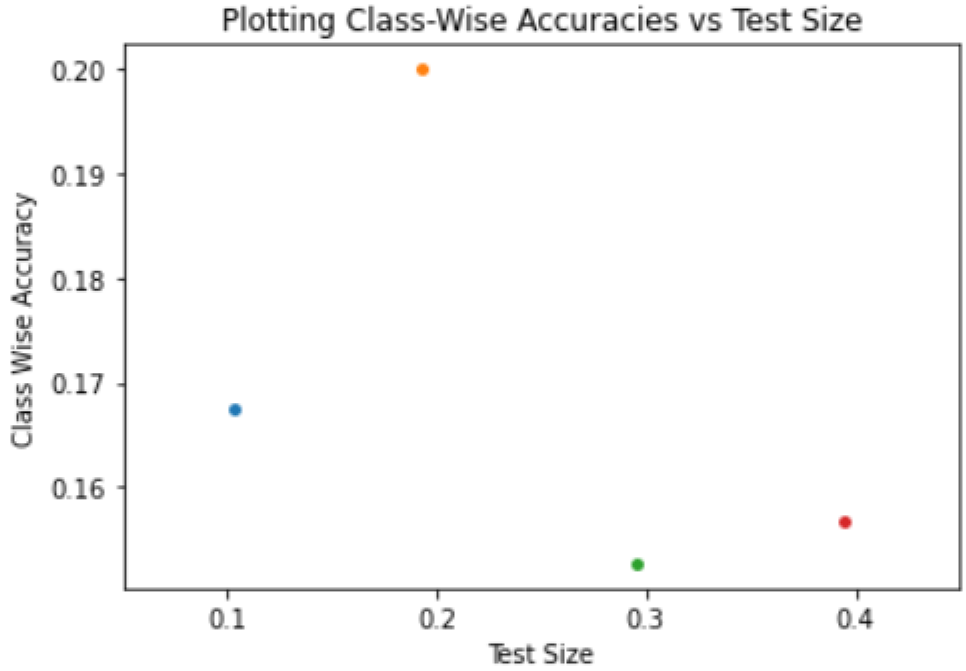
6. Got the results and plotted accuracies for different ratio of training dataset
    a. Used `multilabel_confusion_matrix()` to get the confusion matrix for each class (60-40, 70-30, etc).
    b. Visualised the classwise accuracies vs test size for each train test ratio using *seaborn*.

Observations-

Class Wise Accuracy (using `model.score()`) vs Test Size -

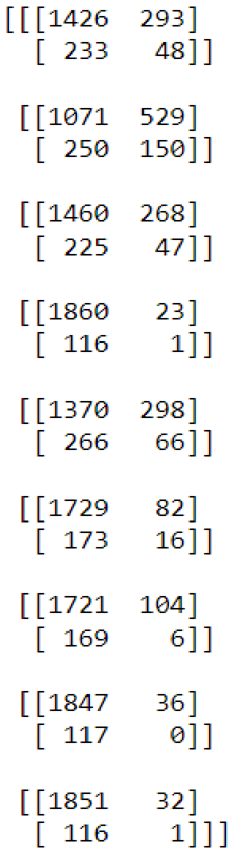


Accuracy (using `accuracy_score()`) vs Test Size -



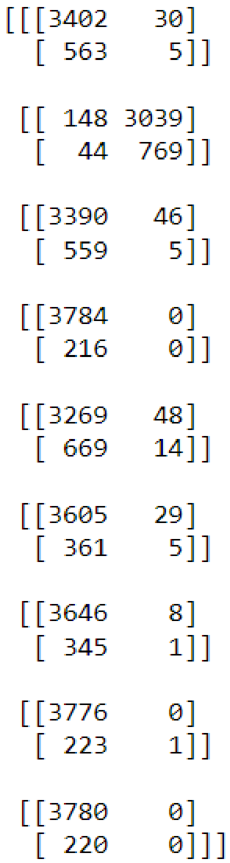
Confusion Matrix -

90:10



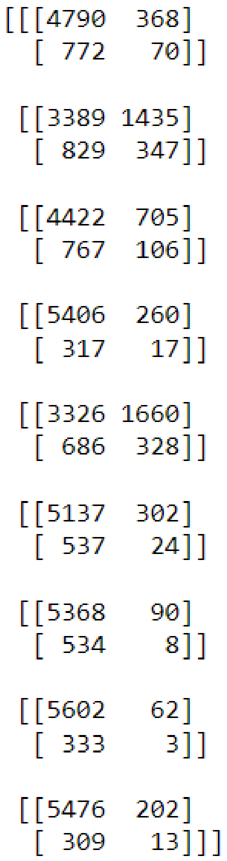
```
[[[1426  293]
  [ 233   48]]

 [[1071  529]
  [ 250  150]]

 [[1460  268]
  [ 225   47]]

 [[1860   23]
  [ 116    1]]

 [[1370  298]
  [ 266   66]]

 [[1729   82]
  [ 173   16]]

 [[1721  104]
  [ 169    6]]

 [[1847   36]
  [ 117    0]]

 [[1851   32]
  [ 116    1]]]
```

80:20



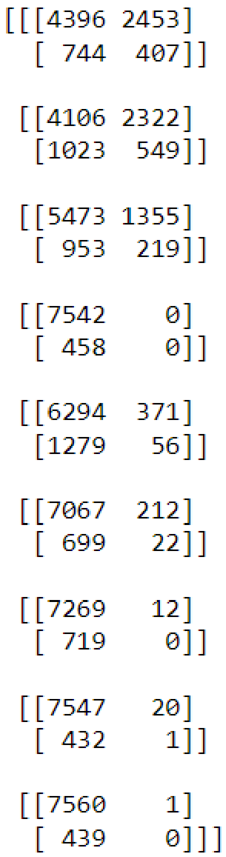
```
[[[3402   30]
  [ 563    5]]

 [[ 148 3039]
  [  44  769]]

 [[3390   46]
  [ 559    5]]

 [[3784    0]
  [ 216    0]]

 [[3269   48]
  [ 669   14]]

 [[3605   29]
  [ 361    5]]

 [[3646    8]
  [ 345    1]]

 [[3776    0]
  [ 223    1]]

 [[3780    0]
  [ 220    0]]]
```

70:30



```
[[[4790  368]
  [ 772   70]]

 [[3389 1435]
  [ 829  347]]

 [[4422  705]
  [ 767  106]]

 [[5406  260]
  [ 317   17]]

 [[3326 1660]
  [ 686  328]]

 [[5137  302]
  [ 537   24]]

 [[5368   90]
  [ 534    8]]

 [[5602   62]
  [ 333    3]]

 [[5476  202]
  [ 309   13]]]
```

60:40



```
[[[4396 2453]
  [ 744  407]]

 [[4106 2322]
  [1023  549]]

 [[5473 1355]
  [ 953  219]]

 [[7542    0]
  [ 458    0]]

 [[6294  371]
  [1279   56]]

 [[7067  212]
  [ 699   22]]

 [[7269   12]
  [ 719    0]]

 [[7547   20]
  [ 432    1]]

 [[7560    1]
  [ 439    0]]]
```

Accuracy for each dataset ratio of train split -

S.No	Ratio of Train Test Split	Accuracy
1.	**90:10**	16.75%
2.	**80:20**	20.0%
3.	**70:30**	15.27%
4.	**60:40**	15.675%

Analysis -

None of the changes to data choice and train test split leads to good accuracy. This may be due to enough different scenarios not being tried for data modifications or may be due to the training data set being small, thus the model is not able to train itself properly.

Code -

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score,
multilabel_confusion_matrix
from sklearn.neural_network import MLPClassifier
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

data = pd.read_csv('roo_data.csv')
data.head()

# All coloumn names
i = 0
for col in data.columns:
    i += 1
    print(f"{i}.", col)

# All rows present for Suggested Job Role
Xa = set(data['Suggested Job Role'])
cnt = 0
for i in Xa:
    cnt += 1
    print(f"{cnt}.", i)

def club(group_roles, group_name):
    for role_name in group_roles:
        data.loc[data['Suggested Job Role'] == role_name, 'Suggested
Job Role' ] = group_name

def club_admin_roles():
    group_roles = ['Network Security Administrator', 'Portal
Administrator', 'Database Administrator', 'Systems Security
Administrator']
    group_name = 'Administrator'
    club(group_roles, group_name)

def club_analyst_roles():
    group_roles = ['Business Intelligence Analyst', 'Business Systems
Analyst', 'CRM Business Analyst', 'Information Security Analyst',
'Programmer Analyst', 'Systems Analyst', 'E-Commerce Analyst']
    group_name = 'Analyst'
    club(group_roles, group_name)

def club_architect_roles():
    group_roles = ['Data Architect', 'Solutions Architect']
    group_name = 'Data Science'
    club(group_roles, group_name)

def club_assosciate_roles():
    group_roles = ['UX Designer', 'Design & UX', 'Software Quality
Assurance (QA) / Testing', 'Quality Assurance Associate',
'Information Technology Auditor']
    group_name = 'Assosciate Roles'
    club(group_roles, group_name)

def club_developer_roles():
    group_roles = ['Applications Developer', 'CRM Technical
Developer', 'Database Developer', 'Mobile Applications Developer',
'Software Developer', 'Web Developer']
    group_name = 'Developer'
    club(group_roles, group_name)

def club_engineer_roles():
    group_roles = ['Network Engineer', 'Network Security Engineer',
'Technical Engineer']
    group_name = 'Engineer'
    club(group_roles, group_name)
    group_roles = ['Software Engineer', 'Software Systems Engineer']
    group_name = 'Software Engineer'
    club(group_roles, group_name)

def club_manager_roles():
    group_roles = ['Database Manager', 'Information Technology
Manager', 'Project Manager']
    group_name = 'IT Manager'
    club(group_roles, group_name)

def club_technical_support_roles():
    group_roles = ['Technical Services/Help Desk/Tech Support',
'Technical Support']
    group_name = 'Technical Support'
    club(group_roles, group_name)

def create_groups():
    club_admin_roles()
    club_analyst_roles()
    club_architect_roles()
    club_assosciate_roles()
    club_developer_roles()
    club_engineer_roles()
    club_manager_roles()
    club_technical_support_roles()

create_groups()
# Data Science, AI Engineer, Software Engineer - Assignment 1

def encoding_label_strings():
    encoder_label = LabelEncoder()
    string_columns = ['can work long time before system?',
'self-learning capability?', 'Extra-courses did', 'certifications',
'workshops', 'talenttests taken?', 'olympiads', 'reading and writing
skills', 'memory capability score', 'Interested subjects',
'Job/Higher Studies?', 'Type of company want to settle in?', 'Taken
inputs from seniors or elders', 'interested in games', 'Interested
Type of Books', 'Salary Range Expected', 'In a Realtionship?',
'Gentle or Tuff behaviour?', 'Management or Technical',
'Salary/work', 'hard/smart worker', 'worked in teams ever?',
'Introvert', 'Suggested Job Role', 'interested career area ']
    for col_name in string_columns:
        data[col_name] = encoder_label.fit_transform(data[col_name])

encoding_label_strings()

X = data.drop(['Suggested Job Role'], axis='columns')
Y = data['Suggested Job Role']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
random_state=3654, test_size=0.1)

model = MLPClassifier(hidden_layer_sizes=(128, 64),
activation="relu", max_iter=1000)
model.fit(X_train, Y_train)
Y_prediction = model.predict(X_test)
```

```
classwise_accuracy1 = model.score(X_test, Y_test)
print(classwise_accuracy1)

accuracy_90_10 = accuracy_score(Y_test, Y_prediction)
print(accuracy_90_10)

confusion_matrix_90_10 = multilabel_confusion_matrix(Y_test,
Y_prediction, labels=np.unique(Y))
print(confusion_matrix_90_10)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
random_state=3654, test_size=0.2)

model = MLPClassifier(hidden_layer_sizes=(128, 64),
activation="relu", max_iter=1000)
model.fit(X_train, Y_train)
Y_prediction = model.predict(X_test)

classwise_accuracy2 = model.score(X_test, Y_test)
print(classwise_accuracy2)

accuracy_80_20 = accuracy_score(Y_test, Y_prediction)
print(accuracy_80_20)

confusion_matrix_80_20 = multilabel_confusion_matrix(Y_test,
Y_prediction, labels=np.unique(Y))
print(confusion_matrix_80_20)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
random_state=3654, test_size=0.3)

model = MLPClassifier(hidden_layer_sizes=(128, 64),
activation="relu", max_iter=1000)
model.fit(X_train, Y_train)
Y_prediction = model.predict(X_test)

classwise_accuracy3 = model.score(X_test, Y_test)
print(classwise_accuracy3)

accuracy_70_30 = accuracy_score(Y_test, Y_prediction)
print(accuracy_70_30)

confusion_matrix_70_30 = multilabel_confusion_matrix(Y_test,
Y_prediction, labels=np.unique(Y))
```

```
print(confusion_matrix_70_30)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
random_state=3654, test_size=0.4)

model = MLPClassifier(hidden_layer_sizes=(128, 64),
activation="relu", max_iter=1000)
model.fit(X_train, Y_train)
Y_prediction = model.predict(X_test)

classwise_accuracy4 = model.score(X_test, Y_test)
print(classwise_accuracy4)

accuracy_60_40 = accuracy_score(Y_test, Y_prediction)
print(accuracy_60_40)

confusion_matrix_60_40 = multilabel_confusion_matrix(Y_test,
Y_prediction, labels=np.unique(Y))
print(confusion_matrix_60_40)

def visualize_class_wise_accuracy():
    Y = [classwise_accuracy1, classwise_accuracy2,
classwise_accuracy3, classwise_accuracy4]
    X = [0.1, 0.2, 0.3, 0.4]

    graph = sns.stripplot(X, Y)
    graph.set(xlabel ='Test Size', ylabel ='Class Wise Accuracy')
    plt.title('Plotting Class-Wise Accuracies vs Test Size')
    plt.show()

visualize_class_wise_accuracy()

def visualize_accuracy():
    Y = [accuracy_90_10, accuracy_80_20, accuracy_70_30,
accuracy_60_40]
    X = [0.1, 0.2, 0.3, 0.4]

    graph = sns.stripplot(X, Y)
    graph.set(xlabel ='Test Size', ylabel ='Accuracy')
    plt.title('Plotting Accuracies vs Test Size')
    plt.show()

visualize_accuracy()
```