CSE643: Artificial Intelligence
Sehban Fazili
M Tech CSE: MT21143

ML-based prediction system to predict the job role for a new graduate.

Steps:

1. **Importing required Libraries**

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

2. **Reading the dataset**

```
dataset = pd.read_csv("roo_data.csv")
```

3. **Exploratory Data Analysis**

In the given dataset:
Total entries = 20000 from range 0 to 19999
Null Values = None
`dtypes: int64(14), object(25)` i.e.,
Categorical Data count: 25
Numerical Data count: 14

3.1 Dropping irrelevant columns using drop command

```
final = dataset.drop(drop_columns, axis = 1)
```

3.2 Since the first 9 columns are Percentages of a student in different subjects so we take the mean of that and print a single percentage column with the name ‘Academic Percentage’.

In this step, first percentage columns were only taken using the ‘iloc’ function call, then mean was calculated using `df.mean(axis=1)` command.

Finally from the created data frame, only the column with mean values was kept and the rest were dropped. The data frame was named df1.

3.3 A separate data frame was created (not with percentages) df3.

Then df1 and df3 were concatenated using the command

```
final_df = pd.concat([df1,df3],axis=1)
```

3.4 Reduced the classes to a total of 6 classes based on some similarities between them.

3.5 Encoding was done on categorical data points using `LabelEncoder & OneHotEncoder.`

```
labelencoder = LabelEncoder()
for i in range(4,9):
    data[:,i] = labelencoder.fit_transform(data[:,i])
```

3.6 Normalization was done to scale down the data using the command

```
normalized_data = Normalizer().fit_transform(data1)
```

Now dataset can be worked on.

4. Classification Task

4.1 Splitting the dataset into train and test using the command:

```
X_train,X_test,y_train,y_test=train_test_split(X1,y,test_size=
0.2,stratify=y,random_state=10)
```

4.2 Applying Multi-layer Perceptron classifier on the train and test data and fitting the model using the command:

```
classifier = MLPClassifier(hidden_layer_sizes=(40,10,2),
max_iter=300,activation = 'relu',solver='adam',random_state=1)
classifier.fit(X_train, y_train)
```

4.3 Predicting the model and calculating accuracy, confusion matrix, etc.

5. Results:

Train-Test Splits	Accuracy
80-20	31.275
70-30	31.2
60-40	31.1875
90-10	31.25

Classification Report:

1) 80-20 split

```
confusion matrics= [[   0    0    0    0    0  571]
 [   0    0    0    0    0  461]
 [   0    0    0    0    0  235]
 [   0    0    0    0    0  694]
 [   0    0    0    0    0  788]
 [   0    0    0    0    0 1251]]

accuracy= 31.275
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	571
1	0.00	0.00	0.00	461
2	0.00	0.00	0.00	235
3	0.00	0.00	0.00	694
4	0.00	0.00	0.00	788
5	0.31	1.00	0.48	1251
accuracy			0.31	4000
macro avg	0.05	0.17	0.08	4000
weighted avg	0.10	0.31	0.15	4000

2) 70-30 split

```
confusion matrics= [[   2    0    0    0    0  854]
 [   2    0    0    0    0  690]
 [   2    0    0    0    0  351]
 [   5    0    0    0    0 1036]
 [   2    0    0    0    0 1180]
 [   3    0    0    0    3 1870]]

accuracy= 31.2
```

	precision	recall	f1-score	support
0	0.12	0.00	0.00	856
1	0.00	0.00	0.00	692
2	0.00	0.00	0.00	353
3	0.00	0.00	0.00	1041
4	0.00	0.00	0.00	1182
5	0.31	1.00	0.48	1876
accuracy			0.31	6000
macro avg	0.07	0.17	0.08	6000
weighted avg	0.12	0.31	0.15	6000

3) 60-40 split

```
confusion matrics= [[   1    0    0    0    4 1137]
 [   2    0    0    0    1  920]
 [   2    0    0    0    0  469]
 [   4    0    0    0    1 1382]
 [   2    0    0    0    3 1571]
 [   4    0    0    0    6 2491]]

accuracy= 31.1875
```

	precision	recall	f1-score	support
0	0.07	0.00	0.00	1142
1	0.00	0.00	0.00	923
2	0.00	0.00	0.00	471
3	0.00	0.00	0.00	1387
4	0.20	0.00	0.00	1576
5	0.31	1.00	0.48	2501
accuracy			0.31	8000
macro avg	0.10	0.17	0.08	8000
weighted avg	0.15	0.31	0.15	8000

4) 90-10 split

```
confusion matrics= [[  0   0   0   0   0 285]
 [  0   0   0   0   0 231]
 [  0   0   0   0   0 118]
 [  0   0   0   0   0 347]
 [  0   0   0   0   0 394]
 [  0   0   0   0   0 625]]

accuracy= 31.25
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	285
1	0.00	0.00	0.00	231
2	0.00	0.00	0.00	118
3	0.00	0.00	0.00	347
4	0.00	0.00	0.00	394
5	0.31	1.00	0.48	625
accuracy			0.31	2000
macro avg	0.05	0.17	0.08	2000
weighted avg	0.10	0.31	0.15	2000

6. Code Listing:

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

dataset = pd.read_csv("roo_data.csv")

dataset.info()

dataset.describe()

drop_columns = ['Hours working per day','Management or Technical','hackathons','can work long time before system?','Extra-courses did','workshops','worked in teams ever?','self-learning capability?','memory capability score','Job/Higher Studies?','hard/smart worker','talenttests taken?','olympiads','reading and writing skills','Type of company want to settle in?','interested in games','Taken inputs from seniors or elders','Interested Type of Books','In a Realtionship?','Gentle or Tuff behaviour?','Introvert']
final = dataset.drop(drop_columns, axis = 1)

final.head()

df = final.iloc[:,:9]
```

```
df.head()

df['Average Academic Percentage'] = df.mean(axis=1)

df.head()

drop_list = ['Acedamic percentage in Operating Systems',  'percentage in Algorithms'
,'Percentage in Programming Concepts','Percentage in Software Engineering',  'Percentage in Computer Networks',  'Percentage in Electronics Subjects'  ,'Percentage in Computer Architecture',  'Percentage in Mathematics', 'Percentage in Communication skills']

df1 = df.drop(drop_list, axis = 1)

df1.info()

df1.head()

df2 = df1['Average Academic Percentage'].nunique()
df2

df3 = final.iloc[:,9:]

df3.head()

final_df = pd.concat([df1,df3],axis=1)

final_df.head()

final_df.shape

Suggested_DF = final_df.iloc[:,9:]

Suggested_DF.head()

Suggested_DF['Suggested Job Role'].replace({"Database Developer": 'Database Profile','Database Manager':'Database Profile','Database Administrator':'Database Profile','Data Architect': "Database Profile", 'Technical Support':'Technical/Support Profile','Technical
```

```
Services/Help Desk/Tech Support':'Technical/Support Profile','Information Technology Auditor':'Technical/Support Profile','Software Quality Assurance (QA) / Testing':'Technical/Support Profile','Information Technology Manager':'Technical/Support Profile','Technical Engineer':'Technical/Support Profile','Portal Administrator':'Technical/Support Profile','Quality Assurance Associate':'Technical/Support Profile','Systems Analyst':'Technical/Support Profile','Solutions Architect':'Technical/Support Profile','Systems Security Administrator':'Networking Profile','Network Security Administrator':'Networking Profile','Network Engineer':'Networking Profile','Network Security Engineer':'Networking Profile','Information Security Analyst':'Networking Profile','UX Designer':'Design Profile','Design & UX':'Design Profile',"Business Systems Analyst": "Business Profile", "Business Intelligence Analyst": "Business Profile", "Project Manager":"Business Profile","E-Commerce Analyst": "Business Profile","CRM Technical Developer": 'Technical/Support Profile', "CRM Business Analyst": "Business Profile","Software Systems Engineer": "Software Profile","Programmer Analyst':'Software Profile","Mobile Applications Developer": "Software Profile", "Web Developer": "Software Profile", "Software Developer": "Software Profile", "Applications Developer": "Software Profile", "Software Engineer": "Software Profile"}, inplace=True)

Suggested_DF.head()

Suggested_DF['Suggested Job Role'].nunique()

first_half=final_df.iloc[:,:9]

first_half.head()

roo_datset = pd.concat([first_half,Suggested_DF], axis=1)

roo_datset.shape

data = roo_datset.iloc[:,:-1].values
label = roo_datset.iloc[:,-1].values
len(data[0])

roo_datset.iloc[:,4:9]

roo_datset.iloc[:,:4]

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
labelencoder = LabelEncoder()

for i in range(4,9):
    data[:,i] = labelencoder.fit_transform(data[:,i])
data[:5]

data[:5,4:]

from sklearn.preprocessing import Normalizer

data1=data[:,:4]

normalized_data = Normalizer().fit_transform(data1)
print(normalized_data.shape)

normalized_data

data2=data[:,4:]
data2.shape

df11 = np.append(normalized_data,data2,axis=1)

df11.shape

X1 = pd.DataFrame(df11,columns=['Average Academic Percentage','Logical quotient rating','coding skills rating','public speaking points','certifications','Interested subjects',
'interested career area','Salary Range Expected',     'Salary/work'])

X1.head()

label = labelencoder.fit_transform(label)
print(len(label))

y=pd.DataFrame(label,columns=["Suggested Job Role"])
y.head()

from sklearn import tree
```

```
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

X_train,X_test,y_train,y_test=train_test_split(X1,y,test_size=0.2,stratify=y,random_state=10)

"""MLP Classifier"""

#Importing MLPClassifier
from sklearn.neural_network import MLPClassifier

#Initializing the MLPClassifier
classifier = MLPClassifier(hidden_layer_sizes=(40,10,2), max_iter=300,activation = 'relu',solver='adam',random_state=1)

classifier.fit(X_train, y_train)

#Predicting y for X_val
y_pred = classifier.predict(X_test)

cm = confusion_matrix(y_test,y_pred)
accuracy = accuracy_score(y_test,y_pred)

print("confusion matrics=",cm)
print("  ")
print("accuracy=",accuracy*100)

from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```