Assignment 4

[Ritisha Gupta, MT22056]

a) **Steps for data preparation/modification.**

Initially, I read the data using pd.read_csv() function of the pandas library. After reading the data, I determined the internal structure/ characteristics of the dataset like the shape of the dataset using the dataset.shape function, checked duplicate values and null values using dataset.isnull().sum() and dataset.duplicated().sum().

```
[52] dataset.shape

[53] dataset.info()

[54] dataset.duplicated().sum()

[55] dataset.isnull().sum()
```

Checking the datatypes of all the columns using dataset.info(). It will give all the datatypes values plus the no of non-null values in each column.

Lately, I found the no of unique values in each column by iterating over every column and using .unique().size. Found all the unique values using .unique() function.

```
lst = dataset.columns

for i in lst:
  print('No of unique values in {}: {}'.format(i,(dataset[i].unique().size)))
```

```
[57] lst = dataset.columns

     for i in lst:
       print('No of unique values in {}: {}'.format(i,(dataset[i].unique())))
```

After seeing the values of each column, I decided to merge the similar values associated with the target value i.e. “Suggested Job Role”. I made 6 groups like

'Administrator', 'Developer', 'Analyst', 'Architects/Designers', 'Engineers', 'Help/Support' and replace original values with these values in our main dataset.

```
[58] #combining values of target variable into 6
     job_roles = {
         'Administrator' : ['Portal Administrator','Project Manager','Information Technology Manager','Network Security Administrator','Information Technology Auditor','Database Manager','
         'Developer' : ['Database Developer','CRM Technical Developer','Mobile Applications Developer','Web Developer','Software Developer','Applications Developer'],
         'Analyst' : ['Business Systems Analyst','Business Intelligence Analyst','Programmer Analyst','Systems Analyst','E-Commerce Analyst','Information Security Analyst','CRM Business Ar
         'Architects/Designers' : ['Design & UX','Solutions Architect','Data Architect','UX Designer'],
         'Engineers' : ['Software Systems Engineer','Network Engineer', 'Software Engineer', 'Technical Engineer', 'Network Security Engineer'],
         'Help/Support': ['Software Quality Assurance (QA) / Testing','Technical Services/Help Desk/Tech Support','Technical Support']
     }

     for i in job_roles:
         dataset.loc[dataset['Suggested Job Role'].isin(job_roles[i]), 'Suggested Job Role'] = i
```

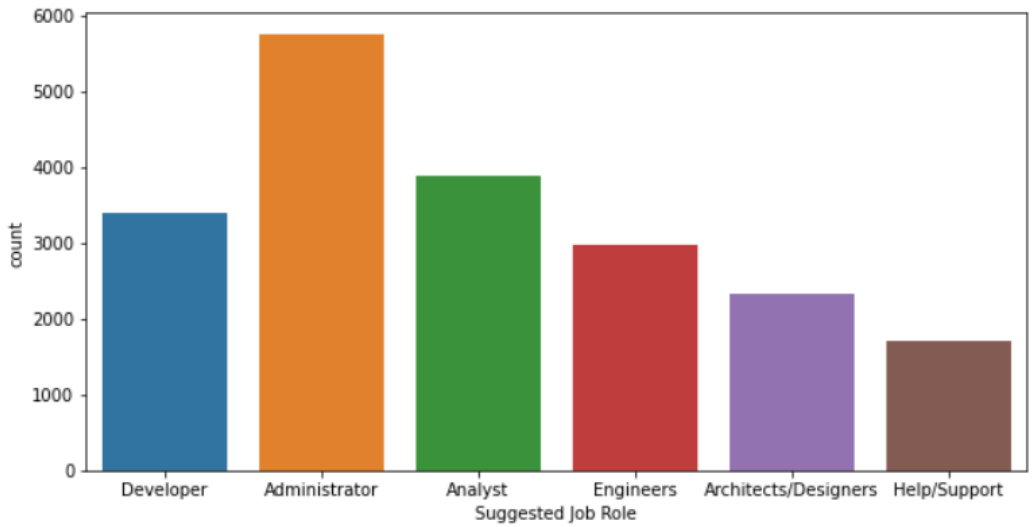
After changing these values we printed unique values of Suggested Job Roles and we got this as our output.

```
[59] #combining similar values in target variable.
     print('Unique Values in Target Variable (Suggested Job Role): {}'.format(dataset['Suggested Job Role'].unique()))

     Unique Values in Target Variable (Suggested Job Role): ['Developer' 'Administrator' 'Analyst' 'Engineers' 'Architects/Designers'
      'Help/Support']
```

All the values are replaced. I plot the count plot between job roles and the count.

```
plt.figure(figsize=(10, 5))
sns.countplot(dataset['Suggested Job Role'])
plt.show()
```



Then for features containing yes/no type values, I have converted them into 0/1. Mapping => 0-no and 1-yes by iterating on all such binary categorical columns.

```
[60] #making list of columns which has only no/yes entries and encoding them to 0/1.
     lst = ['can work long time before system?','worked in teams ever?','Introvert','In a Realtionship?','interested in games',

     for i in lst:
       dataset[i].replace({'yes': 1, 'no': 0}, inplace=True)
```

Later, I did label encoding of all the other categorical variables and we got our final dataset.

```
[63] to_encode = ['certifications','workshops', 'reading and writing skills', 'memory capability score','Interested subjects',
                  'interested career area', 'Job/Higher Studies?','Type of company want to settle in?','Interested Type of Books',
                  'Salary Range Expected', 'Gentle or Tuff behaviour?', 'Management or Technical', 'Salary/work', 'hard/smart worker',
                  'Suggested Job Role']

     for i in to_encode:
       dataset[i] = dataset[i].astype('category').cat.codes
```

b) The experiments performed

Tried and Performed standardization in order to get more accurate results. Later tried different train test splits to see the results of the model.

```
[66] scx = StandardScaler()
     X_scaled = scx.fit_transform(X)
```

```
[ ] lst=['60-40','70-30','90-10']
    for i in range(3):
      print("Accuracy of model with {} train-test split: {}%".format(lst[i],(acc[i]*100)))
```

```
Accuracy of model with 60-40 train-test split: 29.175%
Accuracy of model with 70-30 train-test split: 29.099999999999998%
Accuracy of model with 90-10 train-test split: 29.299999999999997%
```

c) Results

Accuracy of model:

```
[71] acc= accuracy_score(Y_test, test_pred)
     print('Value of Accuracy: {}%'.format(acc*100))
```

```
Value of Accuracy: 29.099999999999998%
```

Confusion Matrix:

```
matrix = confusion_matrix(Y_test, test_pred)
print('Confusion Matrix: \n\n',matrix )
```

```
Confusion Matrix:

 [[1746    0    0    0    0    0]
 [1177    0    0    0    0    0]
 [ 661    0    0    0    0    0]
 [1036    0    0    0    0    0]
 [ 868    0    0    0    0    0]
 [ 512    0    0    0    0    0]]
```

Classwise accuracy:

```
print("Classwise Accuracies: ")
print()
arr = matrix.diagonal()/matrix.sum(axis=1)
for i in range(len(arr)):
  x = round(arr[i]*100, 3)
  print('Accuracy of class {}: {}%'.format(i,x))
```

```
Classwise Accuracies:

Accuracy of class 0: 100.0%
Accuracy of class 1: 0.0%
Accuracy of class 2: 0.0%
Accuracy of class 3: 0.0%
Accuracy of class 4: 0.0%
Accuracy of class 5: 0.0%
```

d) CODE:

```
import numpy as np

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import sklearn as sk
from sklearn.metrics import accuracy_score,confusion_matrix
import warnings
warnings.filterwarnings('ignore')

dataset = pd.read_csv('/content/roo_data - roo_data (1).csv', sep=',')
dataset.head()
```

```
dataset.shape

dataset.info()

dataset.duplicated().sum()

dataset.isnull().sum()

lst = dataset.columns

for i in lst:
  print('No of unique values in {}: {}'.format(i,(dataset[i].unique().size)))

lst = dataset.columns

for i in lst:
  print('No of unique values in {}: {}'.format(i,(dataset[i].unique())))

#combining values of target variable into 6
job_roles = {
    'Administrator' : ['Portal Administrator','Project Manager','Information
Technology Manager','Network Security Administrator','Information Technology
Auditor','Database Manager','Database Administrator','Quality Assurance
Associate','Systems Security Administrator'],
    'Developer' : ['Database Developer','CRM Technical Developer','Mobile
Applications Developer','Web Developer','Software Developer','Applications
Developer'],
    'Analyst' : ['Business Systems Analyst','Business Intelligence
Analyst','Programmer Analyst','Systems Analyst','E-Commerce Analyst','Information
Security Analyst','CRM Business Analyst'],
    'Architects/Designers' : ['Design & UX','Solutions Architect','Data
Architect','UX Designer'],
    'Engineers' : ['Software Systems Engineer','Network Engineer', 'Software
Engineer', 'Technical Engineer', 'Network Security Engineer'],
    'Help/Support': ['Software Quality Assurance (QA) / Testing','Technical
Services/Help Desk/Tech Support','Technical Support']
}

for i in job_roles:
    dataset.loc[dataset['Suggested Job Role'].isin(job_roles[i]), 'Suggested Job
Role'] = i

#combining similar values in target variable.
print('Unique Values in Target Variable (Suggested Job Role):
{}'.format(dataset['Suggested Job Role'].unique()))
```

```
#making list of columns which has only no/yes entries and encoding them to 0/1.
lst = ['can work long time before system?','worked in teams
ever?','Introvert','In a Realtionship?','interested in games','Taken inputs from
seniors or elders','olympiads','talenttests taken?','Extra-courses did','self-
learning capability?']

for i in lst:
  dataset[i].replace({'yes': 1, 'no': 0}, inplace=True)

dataset.head()

plt.figure(figsize=(10, 5))
sns.countplot(dataset['Suggested Job Role'])
plt.show()

to_encode = ['certifications','workshops', 'reading and writing skills', 'memory
capability score','Interested subjects',
             'interested career area', 'Job/Higher Studies?','Type of company
want to settle in?','Interested Type of Books',
             'Salary Range Expected', 'Gentle or Tuff behaviour?', 'Management or
Technical', 'Salary/work', 'hard/smart worker',
             'Suggested Job Role']

for i in to_encode:
  dataset[i] = dataset[i].astype('category').cat.codes

dataset.head()

"""#b)  Divide it into training and testing sets.."""

#seperating dependent and independent variable and making X dataframe having all
independent variables and Y dataframe having all dependent variables.
X=dataset.drop(['Suggested Job Role'], axis=1)
Y=dataset['Suggested Job Role']
print(X.shape)
print(Y.shape)

scx = StandardScaler()
X_scaled = scx.fit_transform(X)

#splitting the dataset into training dataset and test dataset
#0.3 i.e. 30% of the data is test data. and 0.7 i.e. 70% of the data = training
data
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X_scaled, Y, test_size=0.3,
random_state=0)

print("Shape of X_train and X_test: {}, {}".format(X_train.shape,X_test.shape))
print("Shape of Y_train and Y_test: {}, {}".format(Y_train.shape,Y_test.shape))

#Importing MLPClassifier
from sklearn.neural_network import MLPClassifier

#Initializing the MLPClassifier
model = MLPClassifier(hidden_layer_sizes=(150,100,50), max_iter=300,activation =
'logistic',solver='sgd',random_state=1)
model.fit(X_train, Y_train)

train_pred = model.predict(X_train)
test_pred = model.predict(X_test)

acc= accuracy_score(Y_test, test_pred)
print('Value of Accuracy: {}%'.format(acc*100))

matrix = confusion_matrix(Y_test, test_pred)
print('Confusion Matrix: \n\n',matrix )

print("Classwise Accuracies: ")
print()
arr = matrix.diagonal()/matrix.sum(axis=1)
for i in range(len(arr)):
  x = round(arr[i]*100, 3)
  print('Accuracy of class {}: {}%'.format(i,x))

split = [0.4,0.3,0.1]
acc = []

for i in split:
  X_train_analyse, X_test_analyse, Y_train_analyse, Y_test_analyse =
train_test_split(X_scaled, Y, test_size=i, random_state=0)
  model_analyse = MLPClassifier(hidden_layer_sizes=(150,100,50),
max_iter=300,activation = 'logistic',solver='sgd',random_state=1)
  model_analyse.fit(X_train_analyse, Y_train_analyse)
  pred = model_analyse.predict(X_test_analyse)
  acc_score= accuracy_score(Y_test_analyse, pred)
  acc.append(acc_score)

lst=['60-40','70-30','90-10']
for i in range(3):
```

```
  print("Accuracy of model with {} train-test split:
{}%".format(lst[i],(acc[i]*100)))
```

e)