Assignment 4(AI)

About

It's a report file which consists of all the developed models for predict the job role for a new graduate.Different roles are given and we have to perform classification tasks for the job role of the newly graduates.

Usage

I have used google colab as a platform to implement our models. Code mentioned in file AI-A4-Sakshi_kumari-MT21141.ipynb,AI-A4-Sakshi_kumari-MT21141_ANN.ipynb,ai_a4_sakshi_kumari_mt21141.py and ai_a4_sakshi_kumari_mt21141_ann.py.

Dataset:

Given dataset:-

- roo_data (1).csv: Contain all the required data then we just divide it in different ratios for training and testing.

Data preprocessing:

- Firstly club some of the roles and reduce it to 6 class classification tasks.
- Checked for duplicate data in dataset(given dataset has no duplicates).
- Checked for total number of data for each class.
- Checked for the nan values in each column of the dataset.
- First 14 features are categorical, so before applying any kind of ML model I have converted them to numerical data using a label encoder using sklearn library.
- Then normalized the first 14 features of the dataset.
- After that combined all features together and made a new dataframe.
- Splits the dataset in different ratios for training and testing.

Models applied:

I have tried various models like decision tree, svm, adaboost and mlp.

Once I have done with preprocessing of data I moved to next step and divide the dataset in the test and train dataset of different ratio below are the result of 10% test dataset and 90% train dataset and start applying model on it,

- **Decision Tree Classifier:** First model we applied is decision tree classifier which is classifying the test dataset and giving accuracy of 17% . Then we moved to apply the next model.

- **SVM Classifier:** The second model I have applied is svm classifier which is classifying the test dataset and giving accuracy of 25% ,which is slightly better than the previous one but it is not giving good results because it classified all test dataset to one class. So, I have decided to try some other models.

- **Adaboost classifier:** The third model I have tried is the ensemble classifiers i.e adaboost classifier which is classifying the test dataset and giving accuracy of 21% ,which is slightly better than the previous one. So, I have decided to try some other models.

- **MLP classifier:** The fourth model I have tried is the neural network classifiers i.e multi - layer perceptron classifier which is classifying the test dataset and giving accuracy of 23% ,which is slightly better than the previous one. So far it has achieved the best accuracy.

- **ANN classifier:** Also tried to apply the ann classifier but it is not giving good results because it is classified all test dataset to one class.so, it is not so good.Even though its accuracy is 20%.

Result achieved:

Result/accuracy achieved by different model(classifier) at different training and testing percentage are summarized below in a table :

Classifier/Training and Testing ratio	90-10%	80-20%	70-30%	60-40%
Decision Tree	18%	19%	19%	18%
SVM	25%	25%	25%	25%
Adaboost	22%	21%	22%	22%
MPL	23%	20%	22%	21%
ANN	20%	25%	25%	25%

Table 1.1

Accuracy of the models are not good because:-

- The dataset is imbalanced.
- There are too many classes to classify .
- There are also some irrelevant features in the dataset.

Neural network based classification MPL classifier is performing well on this dataset.

Important References:

- Data Preprocessing
- Sklearn_decision_tree
- Sklearn_svm
- Sklearn_adaboost
- Sklearn_mlp
- Sklearn_confusion_matrix
- Sklearn_classification_report

Code :-

```
from google.colab import drive
drive.mount('/content/drive')

# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
from sklearn.neural_network import MLPClassifier
from sklearn.neural_network import MLPRegressor
import keras
from keras.models import Sequential
from keras.layers import Dense
import tensorflow as tf

dataset = pd.read_csv("/content/drive/My Drive/AI_ASSIGNMENT/roo_data (1).csv")

dataset.loc[dataset['Suggested Job Role'].str.contains('Engineer', case=False), 'Suggested Job Role'] = 'Engineer'
dataset.loc[dataset['Suggested Job Role'].str.contains('Analyst', case=False), 'Suggested Job Role'] = 'Analyst'
dataset.loc[dataset['Suggested Job Role'].str.contains('Administrator', case=False), 'Suggested Job Role'] = 'Manager'
dataset.loc[dataset['Suggested Job Role'].str.contains('Manager', case=False), 'Suggested Job Role'] = 'Manager'
dataset.loc[dataset['Suggested Job Role'].str.contains('Designer', case=False), 'Suggested Job Role'] = 'Designer'
dataset.loc[dataset['Suggested Job Role'].str.contains('Developer', case=False), 'Suggested Job Role'] = 'Developer'
dataset.loc[dataset['Suggested Job Role'].str.contains('Quality Assurance', case=False), 'Suggested Job Role'] = 'Technical Support'
dataset.loc[dataset['Suggested Job Role'].str.contains('Support', case=False), 'Suggested Job Role'] = 'Technical Support'
dataset.loc[dataset['Suggested Job Role'].str.contains('Architect', case=False), 'Suggested Job Role'] = 'Designer'
```

```
dataset.loc[dataset['Suggested Job Role'].str.contains('Design & UX', case=False), 'Suggested Job Role'] = 'Designer'
dataset.loc[dataset['Suggested Job Role'].str.contains('Information Technology Auditor', case=False), 'Suggested Job Role'] = 'Manager'

print(dataset['Suggested Job Role'].unique())

dataset[dataset.duplicated()]

dataset['Suggested Job Role'].value_counts()

print(dataset.isna().sum())

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

dataset.iloc[:,14:38]

dataset.iloc[:,:14]

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

labelencoder = LabelEncoder()

for i in range(14,38):
    data[:,i] = labelencoder.fit_transform(data[:,i])
data[:5]

data[:5,14:]

from sklearn.preprocessing import Normalizer

data1=data[:,:14]

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

normalized_data

data2=data[:,14:]
data2.shape

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

df.shape

X1 = pd.DataFrame(df,columns=['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', 'Hours working per day',
    'Logical quotient rating', 'hackathons', 'coding skills rating',
    'public speaking points', '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', 'interested career area ', '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'])

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

X_train,X_test,y_train,y_test=train_test_split(X1,y,test_size=0.1,random_state=10)

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)

from sklearn.metrics import confusion_matrix,accuracy_score

y_pred = clf.predict(X_test)

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

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

from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_pred))
```

```
print(confusion_matrix(y_test,y_pred))

from sklearn import svm

clf = svm.SVC()
clf.fit(X_train, y_train)

svm_y_pred = clf.predict(X_test)

svm_cm = confusion_matrix(y_test,svm_y_pred)
svm_accuracy = accuracy_score(y_test,svm_y_pred)

print("confusion matrics=",svm_cm)
print("  ")
print("accuracy=",svm_accuracy*100)

from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,svm_y_pred))
print(confusion_matrix(y_test,svm_y_pred))

from sklearn.ensemble import AdaBoostClassifier
ada = AdaBoostClassifier(learning_rate=1.75,n_estimators=300)
ada.fit(X_train, y_train)
y_ada = ada.predict(X_test)
ada_cm = confusion_matrix(y_test,y_ada)
ada_accuracy = accuracy_score(y_test,y_ada)

print("confusion matrics=",ada_cm)
print("  ")
print("accuracy=",ada_accuracy*100)

from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_ada))
print(confusion_matrix(y_test,y_ada))

from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(random_state=1, max_iter=300).fit(X_train, y_train)
y_mlp=clf.predict(X_test)
mlp_cm = confusion_matrix(y_test,y_mlp)
mlp_accuracy = accuracy_score(y_test,y_mlp)

print("confusion matrics=",mlp_cm)
print("  ")
print("accuracy=",mlp_accuracy*100)

from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_mlp))
print(confusion_matrix(y_test,y_mlp))
```

Classification metrics for MLP:-

```
[315] from sklearn.metrics import classification_report,confusion_matrix
      print(classification_report(y_test,y_mlp))
      print(confusion_matrix(y_test,y_mlp))
```

```
              precision    recall  f1-score   support

           0       0.18      0.13      0.15       392
           1       0.15      0.03      0.05       248
           2       0.16      0.05      0.08       338
           3       0.22      0.06      0.09       300
           4       0.25      0.70      0.37       505
           5       0.20      0.07      0.10       217

    accuracy                           0.23      2000
   macro avg       0.19      0.17      0.14      2000
weighted avg       0.20      0.23      0.17      2000

[[ 52  13  22  14 277  14]
 [ 37   7  10  12 174   8]
 [ 46   8  17  12 242  13]
 [ 38   5  23  17 206  11]
 [ 87   9  28  15 353  13]
 [ 31   5   8   8 150  15]]
```

Code for ANN:-

```
from google.colab import drive
drive.mount('/content/drive')
```

```
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import keras
from keras.models import Sequential
from keras.layers import Dense
```

```
import tensorflow as tf

dataset = pd.read_csv("/content/drive/My Drive/AI_ASSIGNMENT/roo_data (1).csv")

dataset.loc[dataset['Suggested Job Role'].str.contains('Engineer', case=False), 'Suggested Job Role'] = 'Engineer'
dataset.loc[dataset['Suggested Job Role'].str.contains('Analyst', case=False), 'Suggested Job Role'] = 'Analyst'
dataset.loc[dataset['Suggested Job Role'].str.contains('Administrator', case=False), 'Suggested Job Role'] = 'Manager'
dataset.loc[dataset['Suggested Job Role'].str.contains('Manager', case=False), 'Suggested Job Role'] = 'Manager'
dataset.loc[dataset['Suggested Job Role'].str.contains('Designer', case=False), 'Suggested Job Role'] = 'Designer'
dataset.loc[dataset['Suggested Job Role'].str.contains('Developer', case=False), 'Suggested Job Role'] = 'Developer'
dataset.loc[dataset['Suggested Job Role'].str.contains('Quality Assurance', case=False), 'Suggested Job Role'] = 'Technical Support'
dataset.loc[dataset['Suggested Job Role'].str.contains('Support', case=False), 'Suggested Job Role'] = 'Technical Support'
dataset.loc[dataset['Suggested Job Role'].str.contains('Architect', case=False), 'Suggested Job Role'] = 'Designer'
dataset.loc[dataset['Suggested Job Role'].str.contains('Design & UX', case=False), 'Suggested Job Role'] = 'Designer'
dataset.loc[dataset['Suggested Job Role'].str.contains('Information Technology Auditor', case=False), 'Suggested Job Role'] = 'Manager'

print(dataset['Suggested Job Role'].unique())

dataset['Suggested Job Role'].value_counts()

print(dataset.isna().sum())

from sklearn.preprocessing import LabelEncoder

lb_make = LabelEncoder()
for column in dataset.columns[14:]:
    dataset[column] = lb_make.fit_transform(dataset[column])

X= dataset.iloc[:,0:38]
y= dataset.iloc[:,38]

#Label Encoding k lia
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()

y1=encoder.fit_transform(y)
Y=pd.get_dummies(y1).values
```

```
from sklearn.preprocessing import StandardScaler
normalized_data = StandardScaler().fit_transform(X)
print(normalized_data.shape)
print(X)

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.1,random_state=10)

X_train

model=Sequential()
model.add(Dense(38,input_shape=(38,),activation='relu'))
model.add(Dense(6,activation='softmax'))
model.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(X_train,y_train, batch_size=10, epochs=30)

y_pred= model.predict(X_test)
y_test_class=np.argmax(y_test,axis=1)
y_pred_class=np.argmax(y_pred,axis=1)

from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test_class,y_pred_class))
print(confusion_matrix(y_test_class,y_pred_class))
```

Classification metrics for ANN :-

```
[76] from sklearn.metrics import classification_report,confusion_matrix
     print(classification_report(y_test_class,y_pred_class))
     print(confusion_matrix(y_test_class,y_pred_class))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       392
           1       0.00      0.00      0.00       248
           2       0.00      0.00      0.00       338
           3       0.00      0.00      0.00       300
           4       0.25      1.00      0.40       505
           5       0.00      0.00      0.00       217

    accuracy                           0.25      2000
   macro avg       0.04      0.17      0.07      2000
weighted avg       0.06      0.25      0.10      2000

[[  0   0   0   0 392   0]
 [  0   0   0   0 248   0]
 [  0   0   0   0 338   0]
 [  0   0   0   0 300   0]
 [  0   0   0   0 505   0]
 [  0   0   0   0 217   0]]
```