NEURAL NETWORK & DEEP LEARNING(CS-5720)
(CRN:31196)
ASSIGNMENT -1

Name : Vamsi Krishna Mekala
ID : 700742751
Date : 07/11/2023
Github : https://github.com/vamsi-mekala/Neural-networks-icp-1
Google Drive: https://drive.google.com/file/d/1kjdIfqinS28wKtbMtXumKr5AbwEXL4O9/view?usp=sharing

Question 1: Implement Naïve Bayes method using scikit-learn library
a) Use dataset available with name glass
b) Use train_test_split to create training and testing part
c) Evaluate the model on test part using score
Code:

Importing all the required libraries

In [107]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC, LinearSVC
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error
```

Here we are importing all the required libraries in the python for reading the data into the program and also various other libraries to apply neural network techniques on the dataset we considered.

In [108]:
```
df=pd.read_csv('glass.csv')
df.head()
```

Out[108]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1

Here in the above screenshot we loaded the dataset into the program and tried to see the first 5 rows of the tabular data.

```
In [110]: df.shape
Out[110]: (214, 10)

In [111]: df.isnull()
Out[111]:
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
209	False	False	False	False	False	False	False	False	False	False
210	False	False	False	False	False	False	False	False	False	False
211	False	False	False	False	False	False	False	False	False	False
212	False	False	False	False	False	False	False	False	False	False
213	False	False	False	False	False	False	False	False	False	False

214 rows × 10 columns

In the above screenshot we printed the dimensions of our dataset and also checked if there were any null values in the dataset to perform the data[reprocessing to avoid the errors in model building.

```
In [151]: X=df.iloc[:,:-1]
          y=df.iloc[:,9]
```

Here separated the dependent variables and the feature variables from the dataset.

```
In [115]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)

In [116]: X_train.shape, X_test.shape
Out[116]: ((149, 9), (65, 9))

In [117]: y_train.shape, y_test.shape
Out[117]: ((149,), (65,))
```

Now, the dataset is split into a test set and training set named X_train, X_test, Y_train and Y_test.

```
In [157]: gnb = GaussianNB()
          gnb.fit(X_train, y_train)
Out[157]: GaussianNB(priors=None, var_smoothing=1e-09)

In [158]: y_pred = gnb.predict(X_test)

In [159]: print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
          Model accuracy score: 0.4615

In [160]: print(classification_report(y_test, y_pred))
                        precision    recall  f1-score   support

                     1       0.39      0.86      0.54        21
                     2       0.50      0.12      0.19        26
                     3       0.00      0.00      0.00         7
                     5       0.00      0.00      0.00         2
                     6       0.67      1.00      0.80         2
                     7       0.88      1.00      0.93         7

              accuracy                           0.46        65
             macro avg       0.41      0.50      0.41        65
          weighted avg       0.44      0.46      0.37        65
```

Finally the Naive Bayes method is implemented on the dataset and the model has show an accuracy of around 45% and then the classification report is also printed.

Question 2: Implement linear SVM method using scikit-learn

a) Use the same dataset above
b) Use train_test_split to create training and testing part
c) Evaluate the model on test part using score

Code:

```
In [122]: df2=pd.read_csv('glass.csv')

In [123]: X=df2.iloc[:,:-1]
          y=df2.iloc[:,-1]

In [124]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

In order to use the same dataset we loaded it in as another dataframe and splitted it into a test set and training set.

```
In [125]: svc = SVC()
          svc.fit(X_train, y_train)
          Y_pred = svc.predict(X_test)

In [126]: print(classification_report(y_test, Y_pred))
```

	precision	recall	f1-score	support
1	0.00	0.00	0.00	21
2	0.40	1.00	0.57	26
3	0.00	0.00	0.00	7
5	0.00	0.00	0.00	2
6	0.00	0.00	0.00	2
7	0.00	0.00	0.00	7
accuracy			0.40	65
macro avg	0.07	0.17	0.10	65
weighted avg	0.16	0.40	0.23	65

Here in this screenshot we applied the SVM model to the dataset and printed the classification report after the prediction of the test data on the trained model.

```
In [167]: print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, Y_pred)))
          Model accuracy score: 0.4000
```

Here the SVM model is giving out the model accuracy score as 40%

Question 3: Implement Linear Regression using scikit-learn

a) Import the given "Salary_Data.csv"
b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.
c) Train and predict the model.
d) Calculate the mean_squared error.
e) Visualize both train and test data using scatter plot

Code:

```
In [127]: df3=pd.read_csv("Salary_Data.csv")
```

```
In [128]: df3.head()
```

Out[128]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
In [129]: X=df3.iloc[:,0].values
          y=df3.iloc[:,1].values
```

Here for this 3rd task we are considering another dataset and the dataset is having 2 columns of data and it is loaded into the program and then the dependent variable and independent variables are separated into X and y respectively.

```
In [130]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
```

```
In [131]: regressor = LinearRegression()
          regressor.fit(X_train.reshape(-1,1), y_train.reshape(-1,1))
```

```
Out[131]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
In [132]: Y_Pred = regressor.predict(X_test.reshape(-1,1))
```

```
In [133]: Y_Pred
```

```
Out[133]: array([[ 40835.10590871],
                 [123079.39940819],
                 [ 65134.55626083],
                 [ 63265.36777221],
                 [115602.64545369],
                 [108125.8914992 ],
                 [116537.23969801],
                 [ 64199.96201652],
                 [ 76349.68719258],
                 [100649.1375447 ]])
```

Here in this code snippet we split the dataset to ⅓ ratio to test and train and the linear regressor is fit to the training data and the x_test data is predicted using the trained regressor.

```
In [134]: import math
          mse1 = (mean_squared_error(y_test.reshape(-1,1), Y_Pred,squared=False))
          print(mse1)
```

```
4585.4157204675885
```

The mean squared error is calculated and printed for the above regressor.

```
In [139]: plt.scatter(X_train, y_train, color = "red")
          plt.plot(X_train, regressor.predict(X_train.reshape(-1,1)), color = "green")
          plt.title("Salary vs Experience (Training set)")
          plt.xlabel("Years of Experience")
          plt.ylabel("Salary")
          plt.show()
```

Now we plotted the training set and also the predicted values by the regressor using matplotlib scatter plot to visualize the results clearly.

```
In [144]: plt.scatter(X_test, y_test, color = "red")
          plt.plot(X_train, regressor.predict(X_train.reshape(-1,1)), color = "green")
          plt.title("Salary vs Experience (Testing set)")
          plt.xlabel("Years of Experience")
          plt.ylabel("Salary")
          plt.show()
```

Now we plotted the test set and also the predicted values by the regressor using matplotlib scatter plot to visualize the results clearly.