Neural Networks & Deep Learning

Assignment- 1

Gayathri Keshamoni

ID: 700742488

Video link: https://youtu.be/TiF_QVgw3oU

1.Implement Naïve Bayes method using scikit-learn library Use dataset available with name glass Use train_test_split to create training and testing part. Evaluate the model on test part using score and classification_report(y_true, y_pred)

In [3]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score
import warnings
from sklearn import metrics
```

In [4]:
```
dst_Data = pd.read_csv("glass.csv")
dst_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

- Imported all the required packages.
- Read the data from glass.csv file.
- Extracted the information from the csv file using info() function.

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

In [10]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

In [11]: gn = GaussianNB()

In [12]: gn.fit(X_train, y_train)
Out[12]: GaussianNB()

In [13]: y_pred = gn.predict(X_test)

In [14]: print("Accuracy: ", accuracy_score(y_test, y_pred)*100)
         Accuracy:  37.2093023255814
```

- Using iloc function extracted the selected rows and columns from the data framework.
- Prepared the train data for testing.
- Used GaussianNB() as It helps you to the know the performance of the classification model on a set of test data for that the true values and false are known.
- Calculated the accuracy of the data.

```
In [15]: print("Classification Report:", classification_report(y_test, y_pred))
```

```
Classification Report:              precision    recall  f1-score   support

           1       0.19      0.44      0.27         9
           2       0.33      0.16      0.21        19
           3       0.33      0.20      0.25         5
           5       0.00      0.00      0.00         2
           6       0.67      1.00      0.80         2
           7       1.00      1.00      1.00         6

    accuracy                           0.37        43
   macro avg       0.42      0.47      0.42        43
weighted avg       0.40      0.37      0.36        43
```

- Performance has been evaluated by using classification report.

2. Implement linear SVM method using scikit-learn Use the same dataset above Use train_test_split to create training and testing part Evaluate the model on test part using score and classification_report(y_true, y_pred)

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
```

In [16]:
```
dst_Data = pd.read_csv("glass.csv")
dst_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

- Imported all the required packages.
- Read the data from glass.csv file.
- Extracted the information from the csv file using info() function.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
print("Accuracy: ", accuracy_score(y_test, y_pred)*100)
print("Classification Report: ", classification_report(y_test, y_pred))
```

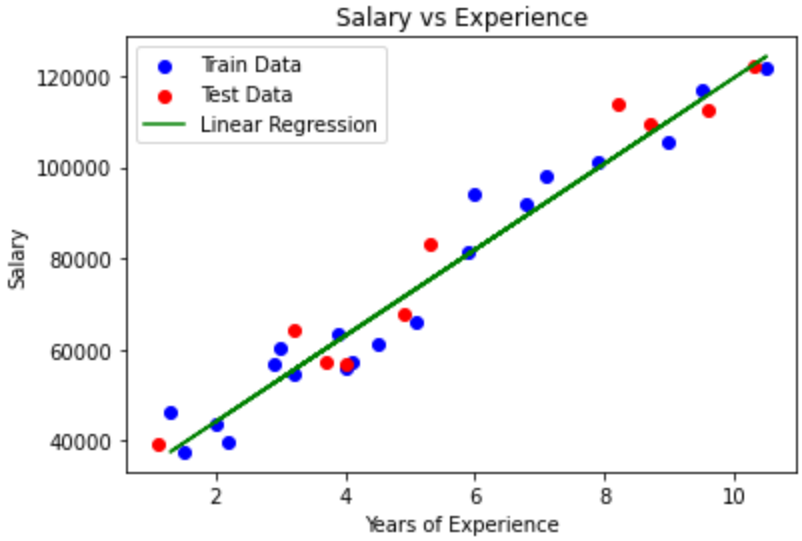
```
Accuracy:  51.162790697674424
Classification Report:                precision    recall  f1-score   support

           1       0.36      0.89      0.52         9
           2       0.58      0.37      0.45        19
           3       0.00      0.00      0.00         5
           5       0.50      0.50      0.50         2
           6       0.00      0.00      0.00         2
           7       0.86      1.00      0.92         6

    accuracy                           0.51        43
   macro avg       0.38      0.46      0.40        43
weighted avg       0.48      0.51      0.46        43
```

- Prepared the training data to test.
- fit() is implemented by every estimator and it accepts an input for the sample data (X) and for supervised models it also accepts an argument for labels (i.e. target data y).
- predict() : given a trained model, predict the label of a new set of data.
- Printed the accuracy and classification report.

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.

In [25]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Step a: Import the dataset
data = pd.read_csv('Salary_Data.csv')
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Step b: Split the data into train and test subsets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=42)

# Step c: Train and predict the model
regressor = LinearRegression()
regressor.fit(X_train, y_train)
y_pred_train = regressor.predict(X_train)
y_pred_test = regressor.predict(X_test)

# Step d: Calculate the mean squared error
mse_train = mean_squared_error(y_train, y_pred_train)
mse_test = mean_squared_error(y_test, y_pred_test)

print("Mean Squared Error (Train):", mse_train)
print("Mean Squared Error (Test):", mse_test)

# Step e: Visualize the data using scatter plot
plt.scatter(X_train, y_train, color='blue', label='Train Data')
plt.scatter(X_test, y_test, color='red', label='Test Data')
plt.plot(X_train, y_pred_train, color='green', label='Linear Regression')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.legend()
plt.show()
```

```
Mean Squared Error (Train): 29793161.082422983
Mean Squared Error (Test): 35301898.887134895
```



- Imported all the required packages.
- Read the data from glass.csv file.
- Extracted the information from the csv file using info() function.
- linearRegression() uses the relationship between the data-points to draw a straight line through all them.
- Predict the label of a new set of data
- a metric to determine the performance of an algorithm .
- The scatter() method in the matplotlib library is used to draw a scatter plot.
- legend() is used to describe elements for a particular area of a graph.