Name:konyala sai vardhan reddy

Student id:700745733

Video link:

https://drive.google.com/file/d/1wWlvCokcwLuKHN6HasbZkrb-PCHjZBvn/view?usp=sharing

Neural Networks & Deep Learning: ICP1

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

First we read the data using pandas package and read csv method

And use describe method to describe the dataset

In [40]:
```
# importing pandas,sklearn packages to read data and use models
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.svm import SVC

df=pd.read_csv('glass.csv')
df.describe() #Basic statistical description of the data
```

Out[40]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009	2.780374
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439	2.103739
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	1.000000
25%	1.516522	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000	1.000000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000	2.000000
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000	3.000000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	7.000000

In [41]:
```
df.head() #  returns the first five rows
```

Out[41]:

	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

Using df.head() to get the top five rows of the dataset

```
In [42]: df.info() # prints the dtype of columns and non null count

<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
```

using the info method to get the non null values and datatypes of columns

```
]: df.columns.values# column names

]: array(['RI', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe', 'Type'],
      dtype=object)
```

Print the column names

```
In [27]: # storing target variable to Y and predictor variable to X
         X = df.drop("Type", axis=1)
         Y = df["Type"]
```

```
In [28]: # Splitting the dataset into the Training set and Test set
         X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,random_state = 42)
```

Storing the target variable and predictor variable separately

Splitting the data into train and test in the ratio of 80:20

```
In [29]: #Initializing the Naive Bayes classifier
         gnb = GaussianNB()

         #Training the model with the train set
         gnb.fit(X_train, Y_train)

         #Using the trained model on the testing data
         Y_pred = gnb.predict(X_test)

         #Evaluating the model using accuracy_score function and predicted output for train set
         acc_knn = round(gnb.score(X_train, Y_train) * 100, 2)
         print('Train Accuracy: ', acc_knn)

         #Evaluating the model using accuracy_score function and predicted output for test set
         acc_knn = round(gnb.score(X_test, Y_test) * 100, 2)
         print('Test Accuracy: ', acc_knn)


         #Getting the classification report of the data set
         print('\nClassification Report: \n', classification_report(Y_test, Y_pred))
```

Using naïve bayes to fit on the training data and using the classifier to predict the test data and calculate the accuracy and classification report

```
Train Accuracy:  56.14
Test Accuracy:  55.81

Classification Report:
              precision    recall  f1-score   support

           1       0.41      0.64      0.50        11
           2       0.43      0.21      0.29        14
           3       0.40      0.67      0.50         3
           5       0.50      0.25      0.33         4
           6       1.00      1.00      1.00         3
           7       0.89      1.00      0.94         8

    accuracy                           0.56        43
   macro avg       0.60      0.63      0.59        43
weighted avg       0.55      0.56      0.53        43
```

In [8]:
```
#Initializing the SVM classifier with linear kernel
svm = SVC(kernel = 'linear')

#Training the model with the training set
svm.fit(X_train, Y_train)

#Predicting the target variable for the test set
Y_pred = svm.predict(X_test)

#Evaluating the model using accuracy_score function and predicted output for train set
acc_knn = round(svm.score(X_train, Y_train) * 100, 2)
print('Train Accuracy: ', acc_knn)

#Evaluating the model using accuracy_score function and predicted output for test set
acc_knn = round(svm.score(X_test, Y_test) * 100, 2)
print('Test Accuracy: ', acc_knn)

#Getting the accuracy report from classification_report
print('Classification Report: \n', classification_report(Y_test, Y_pred,zero_division=1))
```

Using svm to fit on the training data and using the classifier to predict the test data and calculate the accuracy and classification report.

```
Train Accuracy:  66.67
Test Accuracy:  74.42
Classification Report:
              precision    recall  f1-score   support

           1       0.69      0.82      0.75        11
           2       0.67      0.71      0.69        14
           3       1.00      0.00      0.00         3
           5       0.80      1.00      0.89         4
           6       1.00      0.67      0.80         3
           7       0.88      0.88      0.88         8

    accuracy                           0.74        43
   macro avg       0.84      0.68      0.67        43
weighted avg       0.77      0.74      0.72        43
```

Which algorithm you got better accuracy? Can you justify why?

Here the svm performed better when compared to naive bayes as svm works good when the dimensions are more and it is not prone to outliers whereas Naïve Bayes, on the other hand, assumes that features are conditionally independent, and outliers or noise might violate this assumption and negatively impact its performance

Implement Linear Regression using scikit-learn a) Import the given "Salary_Data.csv"

In [35]:
```
# reading salary data csv using pandas
df=pd.read_csv('Salary_Data.csv')
df.head() # printing the first five rows
```

Out[35]:

	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

Reading the salary data using pandas

Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset

In [36]:
```
# storing target variable to Y and predictor variable to X
from sklearn.model_selection import train_test_split
X_train, X_test,Y_train,Y_test = train_test_split(df.iloc[:, :-1].values,df.iloc[:,1].values,test_size =0.33)
X_train#checking train data
```

Out[36]:
```
array([[ 7.9],
       [ 5.9],
       [ 6. ],
       [ 1.3],
       [ 2.9],
       [ 3.2],
       [10.5],
       [ 1.5],
       [ 3.9],
       [ 4.5],
       [ 5.3],
       [ 1.1],
       [ 8.2],
       [ 4.9],
       [ 7.1],
       [ 6.8],
       [ 2. ],
```

Splitting the data into train and test with test size being 33%

In [37]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, Y_train)

# Predicting the Test set result

Y_Pred = regressor.predict(X_test)
```

Calculate the mean_squared error.

In [38]:
```
from sklearn.metrics import mean_squared_error
mean_squared_error(Y_test,Y_Pred)
```

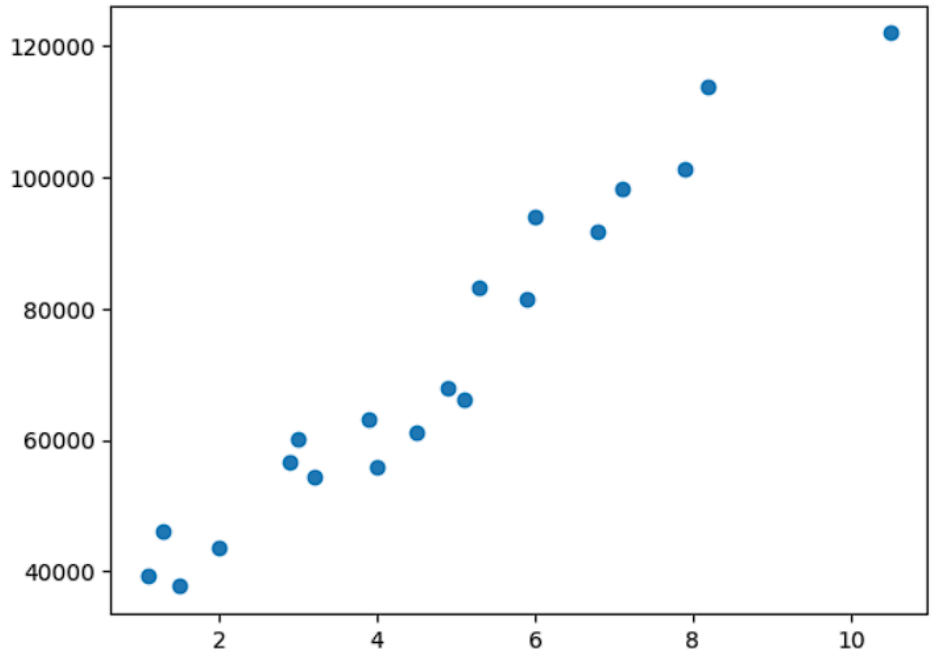
Out[38]:
```
35172616.20754742
```

training the dataset on linear regression and calculating the mean score error

Visualize both train and test data using scatter plot.

[39]:
```
import matplotlib.pyplot as plt
plt.scatter(X_train,Y_train)
```

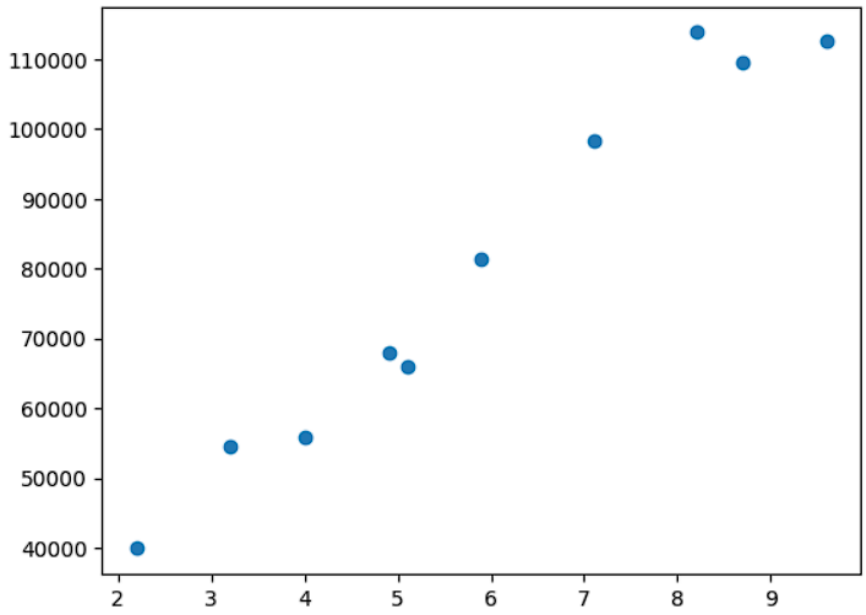
[39]:
```
<matplotlib.collections.PathCollection at 0x17be973c610>
```



Visualizing the train data using scatter plot

In [21]:
```
import matplotlib.pyplot as plt
plt.scatter(X_test,Y_test)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x17be96bf790>
```



visualizing the test data