Neural Networks & Deep Learning

SUSHMA KASOJU

700747358

GitHub Link: https://github.com/Sushma8530/NN_Assignment1.git

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.

```
In [130]: import pandas as pd
          glass_data = pd.read_csv("glass.csv")
          glass_data
```

Out[130]:

	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	0.00	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
209	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 10 columns

I have imported Pandas Library as pd and read the data from "glass.csv" and stored it in data frame named.

```
: from sklearn.model_selection import train_test_split
  X = glass_data.iloc[:,:-1]
  Y = glass_data.iloc[:,-1]
  X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.42,random_state=4)
```

Arrays of input and output are created as X and Y. Scikit-learn is used to divide the test and train data. To divide that we will need to import the train_test_split function from the model_selection module of scikit_learn library.The test size is given as 42 percent of the data. The random state is given as 4.

```
: print(X)
```

```
          RI     Na    Mg    Al     Si     K    Ca    Ba   Fe
0    1.52101  13.64  4.49  1.10  71.78  0.06  8.75  0.00  0.0
1    1.51761  13.89  3.60  1.36  72.73  0.48  7.83  0.00  0.0
2    1.51618  13.53  3.55  1.54  72.99  0.39  7.78  0.00  0.0
3    1.51766  13.21  3.69  1.29  72.61  0.57  8.22  0.00  0.0
4    1.51742  13.27  3.62  1.24  73.08  0.55  8.07  0.00  0.0
..       ...    ...   ...   ...    ...   ...   ...   ...  ...
209  1.51623  14.14  0.00  2.88  72.61  0.08  9.18  1.06  0.0
210  1.51685  14.92  0.00  1.99  73.06  0.00  8.40  1.59  0.0
211  1.52065  14.36  0.00  2.02  73.42  0.00  8.44  1.64  0.0
212  1.51651  14.38  0.00  1.94  73.61  0.00  8.48  1.57  0.0
213  1.51711  14.23  0.00  2.08  73.36  0.00  8.62  1.67  0.0

[214 rows x 9 columns]
```

```
: print(Y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
209    7
210    7
211    7
212    7
213    7
Name: Type, Length: 214, dtype: int64
```

```
: from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
model = gnb.fit(X_train,Y_train)
y_pred = model.predict(X_test)
```

Imported GaussianNB from the sklearn naive bayes model. An instance is created to the GaussianNB and assigned it to the variable gnb. Fit method of the scikit-learn is used to train the model on the train data. Predicted values are found for the test values and are stored in Y_predict variable.

```
In [422]: from sklearn.metrics import classification_report
print(classification_report(Y_test,y_pred))
```

	precision	recall	f1-score	support
1	0.48	0.96	0.64	28
2	0.89	0.22	0.35	37
3	0.00	0.00	0.00	4
5	0.80	0.57	0.67	7
6	1.00	0.67	0.80	3
7	0.69	1.00	0.81	11
accuracy			0.58	90
macro avg	0.64	0.57	0.55	90
weighted avg	0.70	0.58	0.52	90

Imported classification report from metrics of the sklearn library and found the accuracy of the model. The Accuracy I got for this model is 58.0 with test size (42) and random state (4).

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.

In [136]:
```
from sklearn.svm import SVC
svm_linear = SVC(kernel = 'linear')
model = svm_linear.fit(X_train,Y_train)
y_pred = model.predict(X_test)
```

I have imported Pandas Library as pd and read the data from "glass.csv" and stored it in data frame. Imported Support Vector Classifier from the sklearn Support Vector Machine model.An instance is created to the SVC and assigned it to the variable svm_linear.Kernel is given as linear that divides the data and get the linear hyperplane. Fit method of the scikit-learn is used to train the model on the train data. Predicted values are found for the test values and are stored in Y_predict variable.

In [142]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_test,y_pred))
```

```
              precision    recall  f1-score   support

           1       0.61      0.71      0.66        28
           2       0.71      0.65      0.68        37
           3       0.00      0.00      0.00         4
           5       0.50      0.43      0.46         7
           6       1.00      0.67      0.80         3
           7       0.73      1.00      0.85        11

    accuracy                           0.67        90
   macro avg       0.59      0.58      0.57        90
weighted avg       0.64      0.67      0.65        90
```

Imported classification report from metrics of the sklearn library and found the accuracy of the model. The Accuracy I got for this model is 58.0 with test size(42) and random state(4).

The performance of Support Vector Classifier of SVM model is better compared to GaussianNB classifier of Naive Bayes model. SVM is Geometric in nature where as Naive Bayes is Probabilistic in nature. And in SVM we have given a specific kernel as linear so it divides the data as a hyperplane which helps to predict the accurate output values.

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 [137]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
salary_data = pd.read_csv('Salary_Data.csv')
salary_data
```

Out[137]:

	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
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0

Imported Pandas Library and renamed it as pd, Data is read from "Salary_Data.csv" file and stored it in another Data Frame named salary_data.

In [138]:
```
X = salary_data["YearsExperience"]
Y = salary_data["Salary"]
#X1 = [[i,x] for i, x in enumerate(X)]
#Y1 = [[i,y] for i, y in enumerate(Y)]
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.33,random_state=0)
```

Arrays of "YearsExpeience","Salary" are created as X and Y are created. Scikit-learn is used to divide the test and train data. To divide that we will need to import the train_test_split function from the model_selection module of scikit_learn library. As given in the question the test size is given as 1/3 of the data.

In [139]:
```
regressor = LinearRegression()
model = regressor.fit(X_train.values.reshape(-1, 1),Y_train.values.reshape(-1, 1))
```

Imported Linear Regression from the sklearn linear model.An instance is created to the LinearRegression and assigned it to the variable regressor. Fit method of the scikit-learn is used to train the model on the train data. As fit method accepts only the array values, the training values are reshaped.

In [140]:
```
print(model.coef_)
print(model.intercept_)
```

```
[[9345.94244312]]
[26816.19224403]
```

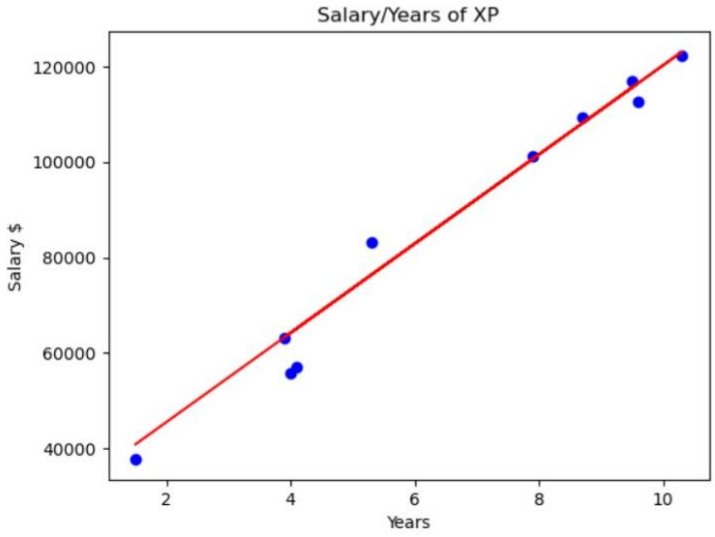
Coefficient and intercept values of the Regression equation are found.

In [141]:
```
Y_predict = model.predict(X_test.values.reshape(-1,1))
mean_squared_error(Y_test, Y_predict)
```

Out[141]:
```
21026037.329511296
```

Predicted values are found for the test values and are stored in Y_predict variable. Mean square error between the original output values and predicted values is found using mean_squared_error method from metrics.

In [142]:
```
plt.title("Salary/Years of XP")
plt.ylabel("Salary $")
plt.xlabel("Years")
plt.scatter(X_test,Y_test,color="blue",label="real data")
plt.plot(X_test,Y_predict,color="red",label="linear model")
plt.show()
```



The predicted values of the model and the original values are compared using scatter and plot methods.