Neural Networks and Deep Learning

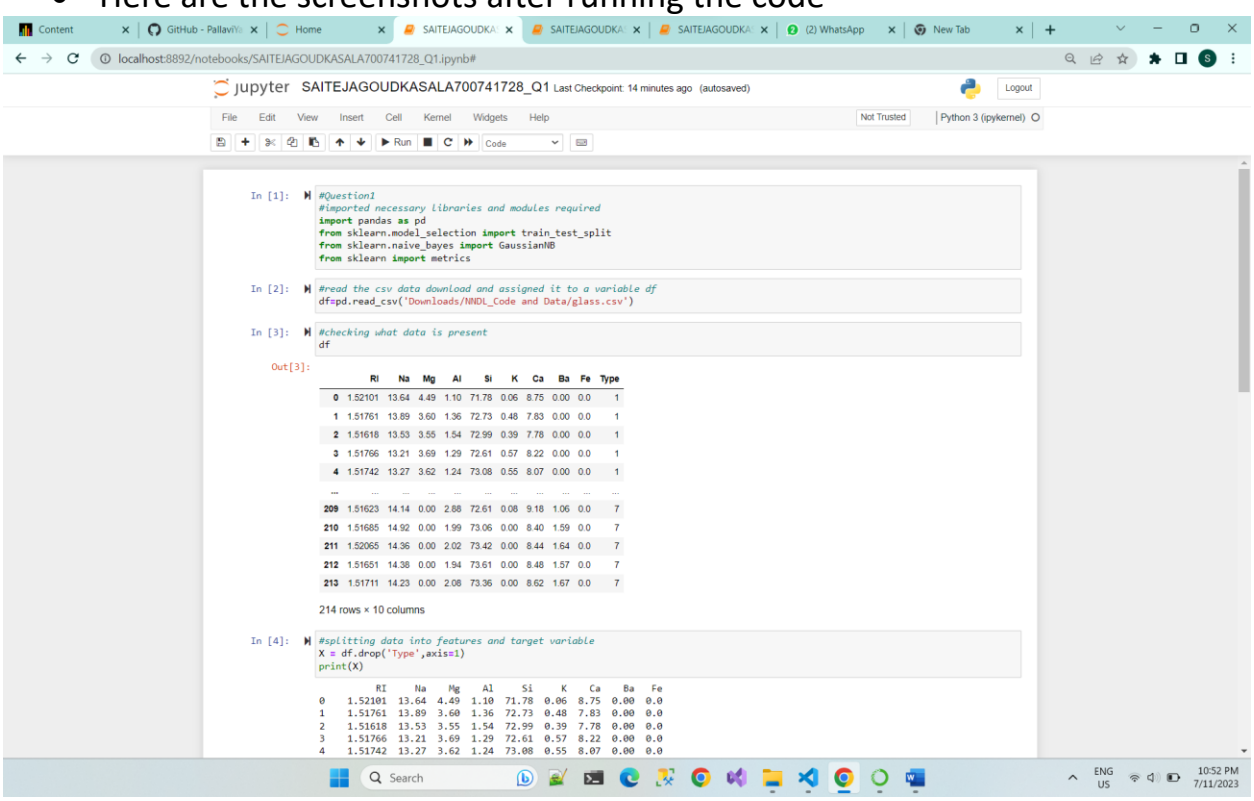
Assignment 1

SAI TEJA GOUD KASALA
700741728

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

In this question,

- The related modules has been imported and reading csv file.
- After that data has been divided into features and target set.
- Later, the data is divided into train and test data sets.
- The data is trained with the model GaussianNB.
- After training the data, need to predict using test data and calculated classification report using y test and y pred.
- Here are the screenshots after running the code



jupyter SAITEJAGOUDKASALA700741728_Q1 Last Checkpoint: 15 minutes ago (autosaved)

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

In [4]:
```
#splitting data into features and target variable
X = df.drop('Type',axis=1)
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]
```

In [5]:
```
#splitting data into features and target variable
y = df['Type']
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
```

In [6]:
```
#splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
```

In [7]:
```
#training the model on training set
model = GaussianNB()
model.fit(X_train, y_train)
```

jupyter SAITEJAGOUDKASALA700741728_Q1 Last Checkpoint: 15 minutes ago (autosaved)

In [7]:
```
#training the model on training set
model = GaussianNB()
model.fit(X_train, y_train)
```

Out[7]:
```
▾ GaussianNB
GaussianNB()
```

In [8]:
```
#predicting based on test set
y_pred = model.predict(X_test)
```

In [9]:
```
#measuring accuracy using test data and predicted data
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Score:", accuracy)
```

```
Score: 0.27906976744186046
```

In [10]:
```
#printing classification report
print("Classification Report:\n", metrics.classification_report(y_test, y_pred))
```

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

           1       0.00      0.00      0.00        33
           2       0.44      0.15      0.23        26
           3       0.11      0.75      0.19         8
           5       0.25      0.50      0.33         4
           6       0.50      0.50      0.50         2
           7       1.00      0.85      0.92        13

    accuracy                           0.28        86
   macro avg       0.38      0.46      0.36        86
weighted avg       0.32      0.28      0.25        86
```

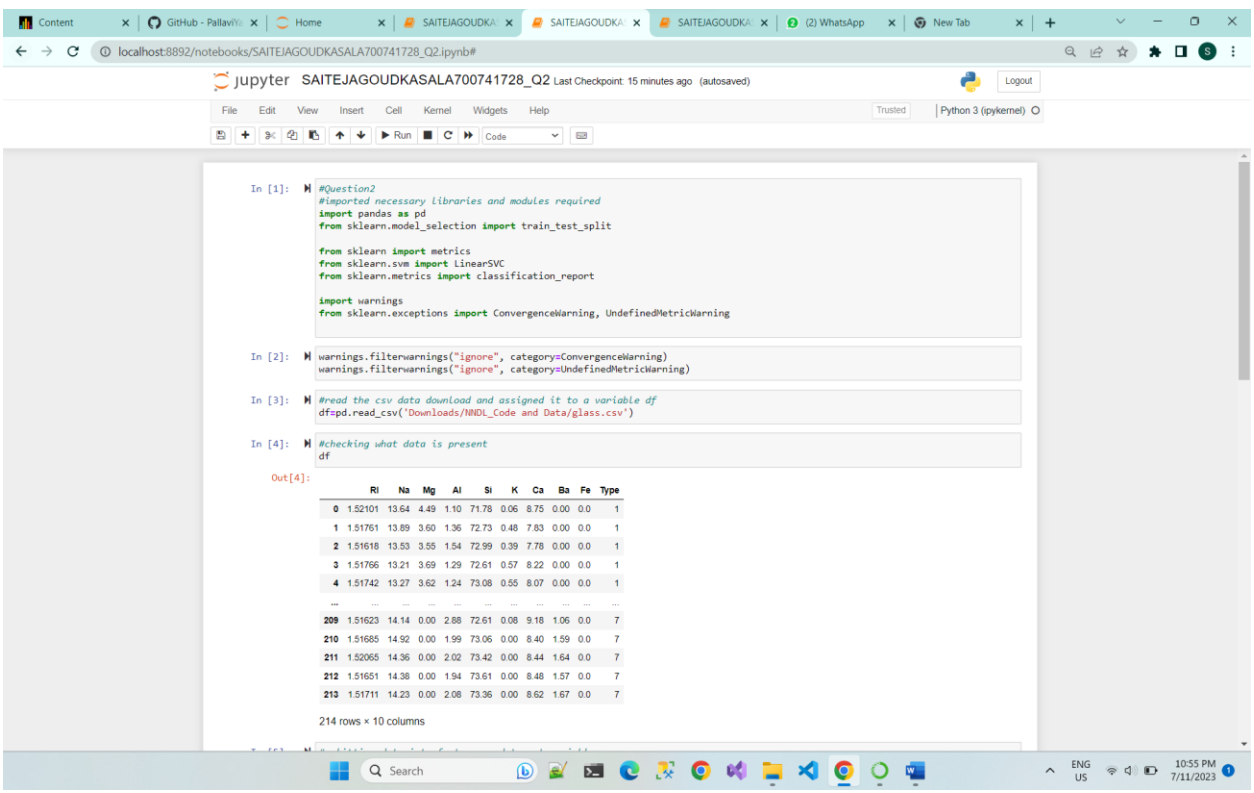
In [ ]:

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

In this question,

- The related modules has been imported and read csv file.
- After that data has been divided into features and target set.
- Later, the data is divided into train and test data sets. The data is trained with the model Linear SVM.
- After training the data, need to predict using test data. And, calculated classification report using y test and y pred.
- It is observed that this has better accuracy score compared to Naïve Bayes method.

Here are the screenshots below running the code.



Content | GitHub - PallaviYe | Home | SAITEJAGOUDKA | SAITEJAGOUDKA | SAITEJAGOUDKA | (2) WhatsApp | New Tab

localhost:8892/notebooks/SAITEJAGOUDKASALA700741728_Q2.ipynb#

jupyter SAITEJAGOUDKASALA700741728_Q2 Last Checkpoint: 17 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

214 rows × 10 columns

In [5]:
```
#splitting data into features and target variable
X = df.drop('Type',axis=1)
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]
```

In [6]:
```
#splitting data into features and target variable
y = df['Type']
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
```

In [7]:
```
#splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
```

In [8]:
```
svm = LinearSVC(random_state=38)
svm.fit(X_train, y_train)
```

Out[8]:
```
LinearSVC
```

Search | ENG US | 10:55 PM 7/11/2023

Content | GitHub - PallaviYe | Home | SAITEJAGOUDKA | SAITEJAGOUDKA | SAITEJAGOUDKA | (2) WhatsApp | New Tab

localhost:8892/notebooks/SAITEJAGOUDKASALA700741728_Q2.ipynb#

jupyter SAITEJAGOUDKASALA700741728_Q2 Last Checkpoint: 17 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

```
211    7
212    7
213    7
Name: Type, Length: 214, dtype: int64
```

In [7]:
```
#splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)
```

In [8]:
```
svm = LinearSVC(random_state=38)
svm.fit(X_train, y_train)
```

Out[8]:
```
LinearSVC
```

In [9]:
```
score = svm.score(X_test, y_test)
y_pred = svm.predict(X_test)
report = classification_report(y_test, y_pred)
```

In [10]:
```
print(f"Accuracy score: ", score)
print(f"Classification report:\n", report)
```

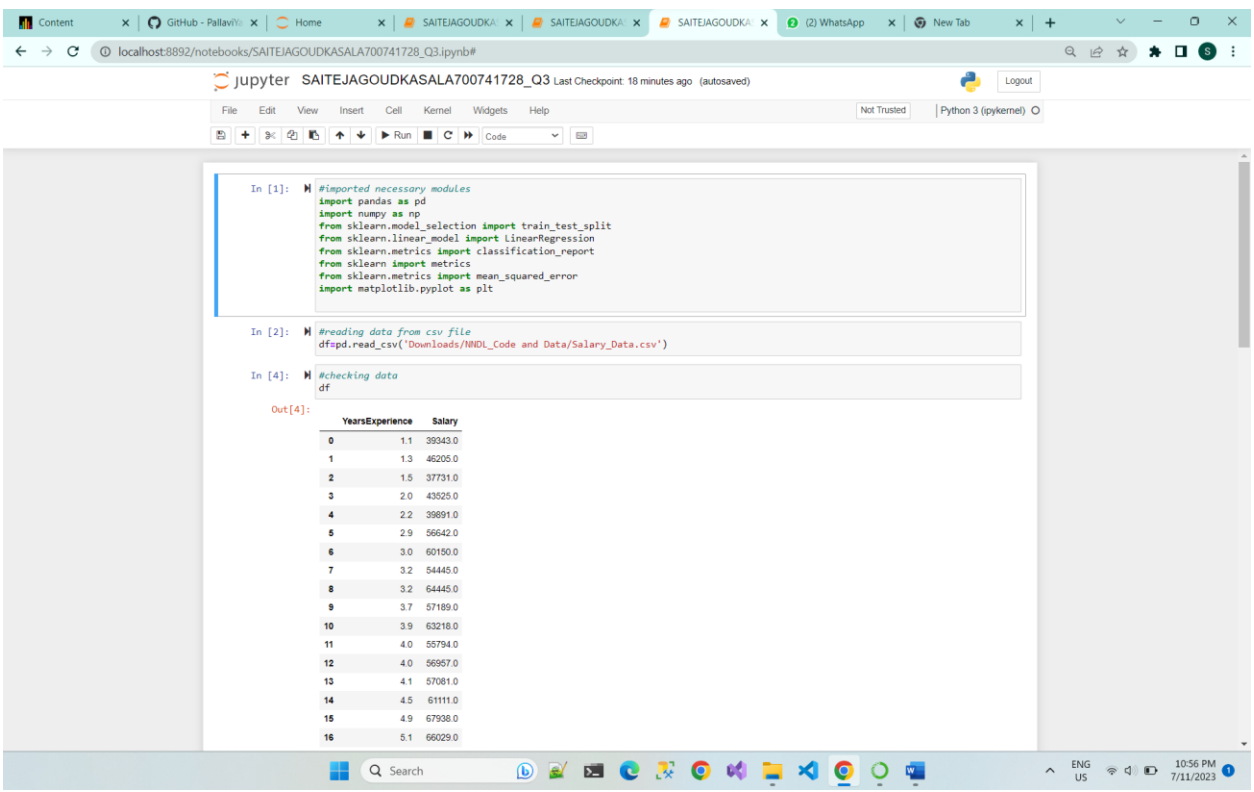
```
Accuracy score:  0.5
Classification report:
                 precision    recall  f1-score   support

           1       0.79      0.33      0.47        33
           2       0.38      0.65      0.48        26
           3       0.00      0.00      0.00         8
           5       0.43      0.75      0.55         4
           6       0.00      0.00      0.00         2
           7       0.63      0.92      0.75        13

    accuracy                           0.50        86
   macro avg       0.37      0.44      0.37        86
weighted avg       0.53      0.50      0.46        86
```

Search | ENG US | 10:55 PM 7/11/2023

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.

- Here salary data has been used.
- In the first step imported all necessary modules and splitted data according to the test partition given.
- After that using Linear Regression trained the model and calculated mean squared error.
- For visualization used matplotlib module and through scatter plot the visualization displayed.

Here are the screenshots below



localhost:8892/notebooks/SAITEJAGOUDKASALA700741728_Q3.ipynb#

jupyter SAITEJAGOUDKASALA700741728_Q3 Last Checkpoint: 18 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

29	10.5	121872.0

In [6]:
```
#splitting data into feature and target set
X=df[['YearsExperience']];
```

In [7]:
```
#splitting data into feature and target set
y=df[['Salary']];
```

In [8]:
```
#splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=1)
```

In [9]:
```
#using linear Regression model on training set
from sklearn.linear_model import LinearRegression
r=LinearRegression()
r.fit(X_train,y_train)
```

Out[9]:
```
▾ LinearRegression
LinearRegression()
```

In [10]:
```
#predicting based on test data
y_pred=r.predict(X_test)
```

In [11]:
```
#calculating mean_squared_error
mean_squared_error=metrics.mean_squared_error(y_test,y_pred)
```

In [12]:
```
#printing mean_squared_error
print(mean_squared_error)
```

```
37496296.6187984
```

In [13]:
```
plt.scatter(X_train, y_train, color='Blue')
plt.scatter(X_test, y_test, color='Green')
plt.plot(X_train, r.predict(X_train), color='Red')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

localhost:8892/notebooks/SAITEJAGOUDKASALA700741728_Q3.ipynb#

jupyter SAITEJAGOUDKASALA700741728_Q3 Last Checkpoint: 18 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

In [11]:
```
#calculating mean_squared_error
mean_squared_error=metrics.mean_squared_error(y_test,y_pred)
```

In [12]:
```
#printing mean_squared_error
print(mean_squared_error)
```

```
37496296.6187984
```

In [13]:
```
plt.scatter(X_train, y_train, color='Blue')
plt.scatter(X_test, y_test, color='Green')
plt.plot(X_train, r.predict(X_train), color='Red')
plt.title('Salary vs Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

In [ ]:

Github link:

https://github.com/sxk17280/NeuralAssignment1