ICP – 5

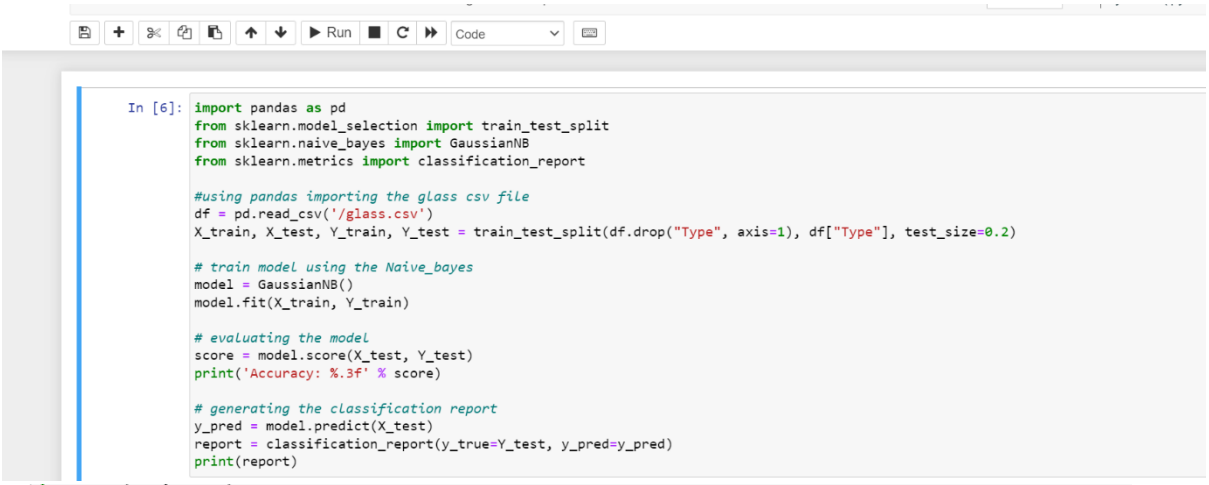
Naga Phaneendra Kumara Gupta Mogili

700757977

GitHub link: https://github.com/nagaphaneendra2001/Deep_Learning_Neural_Networks.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 and classification_report(y_true, y_pred)

Source Code:



```
In [6]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.naive_bayes import GaussianNB
        from sklearn.metrics import classification_report

        #using pandas importing the glass csv file
        df = pd.read_csv('/glass.csv')
        X_train, X_test, Y_train, Y_test = train_test_split(df.drop("Type", axis=1), df["Type"], test_size=0.2)

        # train model using the Naive_bayes
        model = GaussianNB()
        model.fit(X_train, Y_train)

        # evaluating the model
        score = model.score(X_test, Y_test)
        print('Accuracy: %.3f' % score)

        # generating the classification report
        y_pred = model.predict(X_test)
        report = classification_report(y_true=Y_test, y_pred=y_pred)
        print(report)
```

```
Accuracy: 0.372
              precision    recall  f1-score   support

           1       0.50      0.20      0.29        10
           2       0.83      0.25      0.38        20
           3       0.16      0.67      0.26         6
           5       0.00      0.00      0.00         2
           6       1.00      1.00      1.00         2
           7       0.75      1.00      0.86         3

    accuracy                           0.37        43
   macro avg       0.54      0.52      0.46        43
weighted avg       0.63      0.37      0.39        43
```

2. Implement linear SVM method using scikit library
   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)

Source Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report


# using pandas importing the dataset
df = pd.read_csv('/glass.csv')
X_train, X_test, Y_train, Y_test = train_test_split(df.drop("Type", axis=1), df["Type"], test_size=0.2)

# train model using linear support vector machine
model = LinearSVC(dual=False)
model.fit(X_train, Y_train)

# evaluating the model
score = model.score(X_test, Y_test)
print('Accuracy: %.3f' % score)

# generating the classification report
y_pred = model.predict(X_test)
report = classification_report(y_true=Y_test, y_pred=y_pred)
print(report)
```

```
Accuracy: 0.628
              precision    recall  f1-score   support

           1       0.67      0.63      0.65        19
           2       0.53      0.56      0.55        16
           3       0.00      0.00      0.00         1
           5       0.00      0.00      0.00         0
           6       1.00      0.67      0.80         3
           7       0.80      1.00      0.89         4

    accuracy                           0.63        43
   macro avg       0.50      0.48      0.48        43
weighted avg       0.64      0.63      0.63        43
```

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

Linear SVM algorithm got more accuracy than NaiveBayes algorithm. This is because there are several reasons for this. Those are Linear SVM tends to capture more complex relationships between features and labels compared to Naive Bayes, which

assumes independence between features. Also, Naive Bayes might suffer from overfitting with smaller datasets due to its simpler model. However, If there is a significant class imbalance in the dataset, Linear SVM might handle it better than Naive Bayes, which could be sensitive to the distribution of classes.