Spring 2024: CS5720

Neural Networks & Deep Learning - ICP-5

NAME:NANDIMANDALAM VENKATA VINAY VARMA

STUDENT ID:700745193

Github Link: https://github.com/venkatavinayvarma/NeuralNetworks_ICP5.git

Video Link: https://drive.google.com/drive/folders/1B0X1eq38WGeVXGh2-kyPpdM1e71SFWM5?usp=sharing

1. Implement Naïve Bayes method using scikit-learn library
2. Use dataset available with name glass.csv
3. 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)

jupyter NN_ICP5 Last Checkpoint: 57 minutes ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

1. Implement Naïve Bayes method using scikit-learn library
2. Use dataset available with name glass
3. 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)

```
[13]: import pandas as pd
      df = pd.read_csv('glass.csv')   # Read the CSV data
```

```
•[3]: df.info  # Check data quality
```

```
[3]: <bound method DataFrame.info of          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 x 10 columns]>
```

```
[14]: df.describe # Explore data descriptions
```

```
[14]: <bound method NDFrame.describe of          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 x 10 columns]>
```

```
[15]: df.columns.values # Print column names
```

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

```
[16]: from sklearn.model_selection import train_test_split
      from sklearn.naive_bayes import GaussianNB
      from sklearn.metrics import accuracy_score, classification_report
```

```
[17]: # Divide data into features and target variable
      X = df.drop("Type", axis=1)
      Y = df["Type"]
```

```
•[18]: # Split data into training and testing sets
       X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=47)
```

```
•[19]: gnb = GaussianNB() # Initialize the Gaussian Naive Bayes classifier

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

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

       accur_knn = round(gnb.score(X_train, Y_train) * 50, 2) # Evaluating the model using accuracy_score and predicted output
       print('Accuracy: ', accur_knn)
```

```
Accuracy:  26.02
```

```
•[20]: 
       print('\nClassification Report: \n', classification_report(Y_test, Y_predi)) # Classification report of the data set
```

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

           1       0.31      0.20      0.24        20
           2       0.38      0.21      0.27        14
           3       0.00      0.00      0.00         2
           5       0.00      0.00      0.00         1
           6       0.50      0.67      0.57         3
           7       0.50      0.33      0.40         3

    accuracy                           0.23        43
   macro avg       0.28      0.24      0.25        43
weighted avg       0.33      0.23      0.27        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)

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

```
[21]: from sklearn.svm import SVC
      svm = SVC() # Initializing the SVM classifier with linear kernel
```

```
[22]: svm.fit(X_train, Y_train) # Training the model with the training set

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

      acc_svm = round(svm.score(X_train, Y_train) * 50, 2)  # Evaluating the model accuracy using score
      print('Accuracy: ', acc_svm,'\n')

      Accuracy:  18.13
```

```
[23]: print('Classification Report: \n', classification_report(Y_test, Y_pred,zero_division=1)) # Accuracy report from classification_report

Classification Report:
               precision    recall  f1-score   support

           1       1.00      0.00      0.00        20
           2       0.33      1.00      0.49        14
           3       1.00      0.00      0.00         2
           5       1.00      0.00      0.00         1
           6       1.00      0.00      0.00         3
           7       1.00      0.00      0.00         3

    accuracy                           0.33        43
   macro avg       0.89      0.17      0.08        43
weighted avg       0.78      0.33      0.16        43
```

```
[ ]:
```

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

Results and Explanation:

Based on the performance metrics, Naive Bayes generally achieves a higher accuracy than linear SVM in this dataset:

Naive Bayes:

Accuracy: 0.23 (23% of predictions correct)

Better at predicting classes 6 and 7, but struggles with others.

Linear SVM:

Accuracy: 0.33 (33% of predictions correct)

Better at predicting class 2, but poor performance on other classes.