

Spring 2024: CS5720

Neural Networks & Deep Learning - ICP-5

Name: Srikanth Kolluri

Student ID: 700741300

Git Hub Link: <https://github.com/kollurisri/NN-ICP-5>

Video Link: <https://1drv.ms/v/c/914380031521af45/EQUQjtdfZCJCK9uGaGqfV8Bb77LxTvzh5Tw9oVBN9ySA?e=0VRJY6>

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)

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

data = pd.read_csv('glass.csv')

x = data.drop('Type', axis=1)
y = data['Type']

x_train, x_true, y_train, y_true = train_test_split(x, y, test_size=0.3, random_state=42)

nb_classifier = GaussianNB()

nb_classifier.fit(x_train, y_train)

y_pred = nb_classifier.predict(x_true)

accuracy = nb_classifier.score(x_true, y_true)

classification_rep = classification_report(y_true, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)
```

Accuracy: 0.3076923076923077

Classification Report:

	precision	recall	f1-score	support
1	0.00	0.00	0.00	19
2	0.40	0.17	0.24	23
3	0.08	0.75	0.15	4
5	0.33	0.17	0.22	6
6	0.75	1.00	0.86	3
7	0.90	0.90	0.90	10
accuracy			0.31	65
macro avg	0.41	0.50	0.40	65
weighted avg	0.35	0.31	0.29	65

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)

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

data = pd.read_csv('glass.csv')

x = data.drop('Type', axis=1)
y = data['Type']

x_train, x_true, y_train, y_true = train_test_split(x, y, test_size=1/3, random_state=42)

svm_classifier = SVC(kernel='linear')

svm_classifier.fit(x_train, y_train)

y_pred = svm_classifier.predict(x_true)

accuracy = svm_classifier.score(x_true, y_true)

classification_rep = classification_report(y_true, y_pred, zero_division=0)

print("Accuracy:", accuracy)
print("Classification Report:\n", classification_rep)
```

```
Accuracy: 0.6805555555555556
Classification Report:
              precision    recall  f1-score   support

     1              0.69       0.78       0.73         23
     2              0.59       0.68       0.63         25
     3              0.00       0.00       0.00          4
     5              1.00       0.50       0.67          6
     6              0.50       0.50       0.50          4
     7              0.90       0.90       0.90         10

 accuracy                   0.68         72
 macro avg              0.61       0.56       0.57         72
 weighted avg           0.66       0.68       0.66         72
```

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

The Linear SVM algorithm achieved a higher accuracy of approximately 0.68, which is better than the accuracy of approximately 0.31 obtained with Naïve Bayes. Therefore, the Linear SVM algorithm has the best accuracy in this comparison. The Linear SVM model outperforms Naïve Bayes with higher accuracy and better balance across performance metrics. Naïve Bayes assumes feature independence, which may not hold here. The Linear SVM's ability to handle complex relationships makes it a better choice for this task, achieving higher accuracy and balanced performance across classes in the 'glass' dataset.