Assignment-1

Introduction

The dataset make_blobs was imported from sklearn datasets, and the dataset contains a sample of 100 and n_features = 2 (it's the number of features for each sample set). We will be doing KNN analysis on this dataset and by doing train and test the data, analyze the simulated data and to find the accuracy and plot the results.

Observation

#Splitting the data into train and test 80% and 20%

```
res = train_test_split(data, labels,
                       train_size=0.8,
                       test_size=0.2,
                       random_state=12)
train_data, test_data, train_labels, test_labels = res
```

We generally use split the data for a better understanding the characteristics of the model.

#Performing KNN analysis on the data

```
# perform a KNN analysis of the simulated data
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn.fit(train_data, train_labels)

print("Predictions from the classifier:")
learn_data_predicted = knn.predict(train_data)
print(learn_data_predicted)
print("Target values:")
print(train_labels)
print(accuracy_score(learn_data_predicted, train_labels))
```

Here, we train the data using KNN algorithm and for classification a class label is assigned for K nearest neighbors from the dataset is considered as a probable class for the new data point.

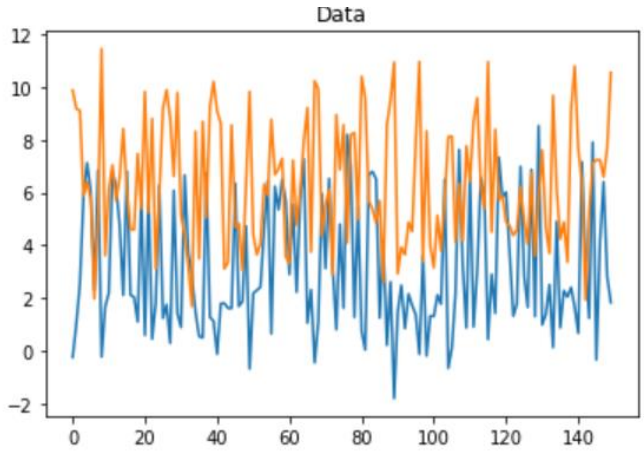
KNeighborsClassifier object that is defined (knn2) and the object that is used to fit the model and make predictions (knn). Assuming that this is just a typo, the code seems to be setting up a K-nearest neighbors (KNN) classifier using scikit-learn.

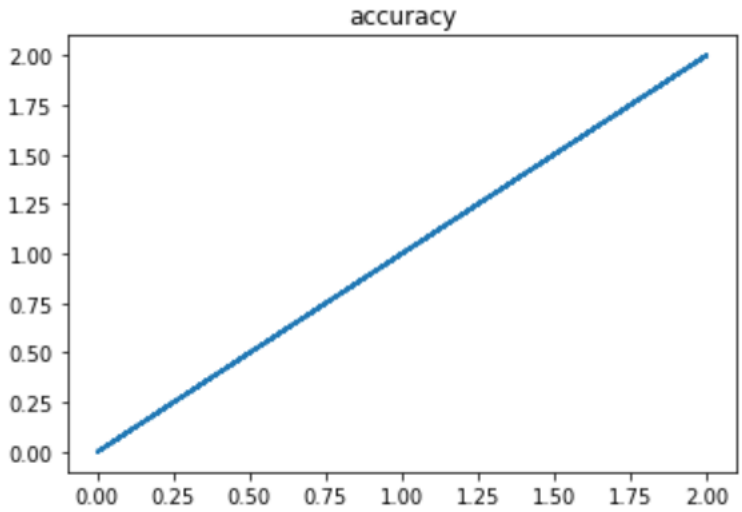
```
# plot your different results
plt.plot(data)
plt.title("Data")
plt.show()

plt.plot(learn_data_predicted,train_labels)
plt.title("different accuracy")
plt.show()
```

```
Predictions from the classifier:
[0 2 1 0 0 1 1 2 2 0 2 2 2 1 1 0 0 2 1 1 0 0 0 1 1 2 0 0 1 0 1 1 1 0 1 2 0
 1 0 1 2 2 2 0 2 0 2 2 0 0 0 1 2 2 2 2 1 1 0 1 2 1 2 2 2 0 0 0 0 0 0 0 1 1
 2 1 2 1 2 2 1 1 1 0 2 1 2 1 0 1 2 1 0 2 0 1 2 2 0 2 1 0 0 2 1 1 2 2 0 1 1
 1 2 2 2 1 1 2 1 2]
Target values:
[0 2 1 0 0 1 1 2 2 0 2 2 2 1 1 0 0 2 1 1 0 0 0 1 1 2 0 0 1 0 1 1 1 0 1 2 0
 1 0 1 2 2 2 0 2 0 2 2 0 0 0 1 2 2 2 2 1 1 0 1 2 1 2 2 2 0 0 0 0 0 0 0 1 1
 2 1 2 1 2 2 1 1 1 0 2 1 2 1 0 1 2 1 0 2 0 1 2 2 0 2 1 0 0 2 1 1 2 2 0 1 1
 1 2 2 2 1 1 2 1 2]
1.0
```

the model's predictions, stored in the test_data_predicted variable, and the true labels, stored in the test_labels variable, are identical. This results in an accuracy score of 1.0, indicating that the model achieved perfect accuracy on the test data.

It's important to note that achieving perfect accuracy on the test data does not necessarily mean that the model is optimal or has learned the underlying pattern in the data well. In fact, it could be a sign of overfitting, where the model has simply memorized the training data and is not generalizing well to new, unseen data. Therefore, it's a good practice to validate the model on a separate validation set or to use cross-validation to assess its performance.





Reference

https://colab.research.google.com/drive/1gGxRec9M55cSIIVMy8ZnRZNmiVRdv-Wa?authuser=1

https://towardsdatascience.com/k-nearest-neighbors-94395f445221