KNN Classifier Summary Report

On

Artificially Generated Make_Blogs Dataset

Professor

CJ Wu

Submitted by

Pavan Chaitanya Bommadevara

Summary:

KNN analysis to the artificial make_blobs dataset and reporting the accuracy.

Procedure:

- There are 4 steps in solving the given problem.
- Step 1: Importing the Libraries.
- Step 2: Spliting the data.
- Step 3: KNN Analysis on Split data.
- Step 4: Accuracy Scores

Importing the Libraries

- My approach towards the problem is initially learned the importance of numpy ,matplotlib libraries.

  These are the required imports

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import numpy as np
```

  We need to use the 3 classes where are a part of make_blobs package.

```
centers = [[2, 4], [6, 6], [1, 9]]
n_classes = len(centers)
```

  Creating the data variables along with the data labels is shown below

```
data, labels = make_blobs(n_samples=150,
                          centers=np.array(centers),
                          random_state=1)
```

  Here we are making the 150 samples with the random state of 1 with the centers as length of the centers.

Spliting the data

- Dividing the Artificial blobs dataset into 2 parts
  - 80 % Training Data
  - 20 % Test Data

So that we can see how the model is going to predict the output. And then assigning the splited data into different variables like train_data,test_data,tain_labels,test_labels.

```
# do a 80-20 split of the data
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
```

KNN Analysis on Split data

The k-nearest neighbors' algorithm, often known as KNN, is a non-parametric, supervised learning classifier that utilizes proximity to create classifications or predictions about an individual data point's grouping.

We now us the KNN classifier with different parameters and then we can analyses the accuracy scores that is generated with them.

Knn classifier with no parameters:

```
from sklearn.neighbors import KNeighborsClassifier
# classifier "out of the box", no parameters
knn = KNeighborsClassifier()
knn.fit(train_data, train_labels)
```

In this we doesn't use any parameters like algorithm,leafsize,weights,metric etc.

Knn classifier with Specific parameters:

```
knn2 = KNeighborsClassifier(algorithm='auto',
                            leaf_size=30,
                            metric='minkowski',
                            p=2,         # p=2 is equivalent to euclidian distance
                            metric_params=None,
                            n_jobs=1,
                            n_neighbors=5,
                            weights='uniform')
```

In this we use any parameters like algorithm,leafsize,weights,metric etc.

Accuracy Scores

We now identify the accuracy scores of different KNN classifier that we have used till now.

```
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 of KNN Classifier with No Parameters: ",accuracy_score(learn_data_predicted, train_labels))
```

The Accuracy values that is generated for the above KNN classifier code are:

```
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]
Accuracy Score of KNN Classifier with No Parameters:  1.0
Accuracy Score of KNN Classifier with Specific Parameters:  1.0
```

We have seen that the accuracy score with the KNN classifier is 1.0 which is quite good.

Summary of results:

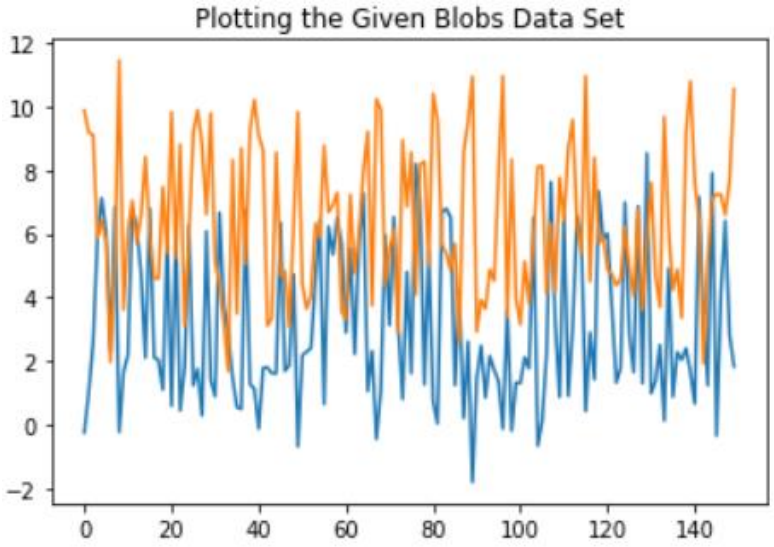
Knn Classifier	Accuracy Score
`knn = KNeighborsClassifier()`	1.0
`Knn2 = KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', p=2, metric_params=None, n_jobs=1, n_neighbors=5, weights='uniform')`	1.0

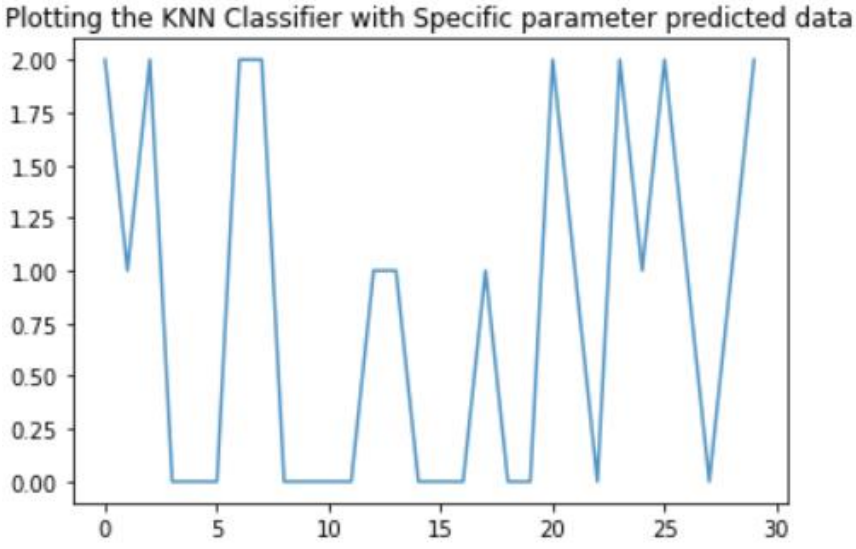
Plotting the Graphs

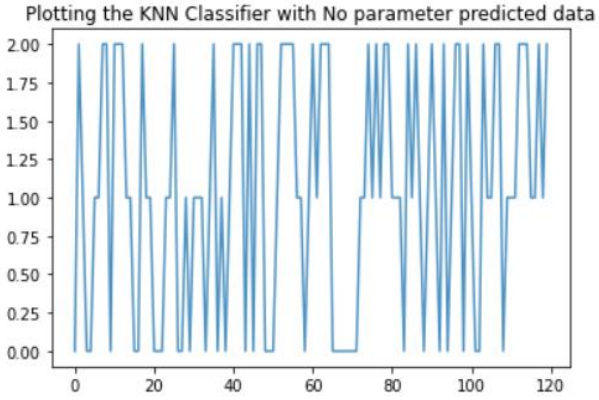
We now see the graphs that are generated from written KNN classifier code are:

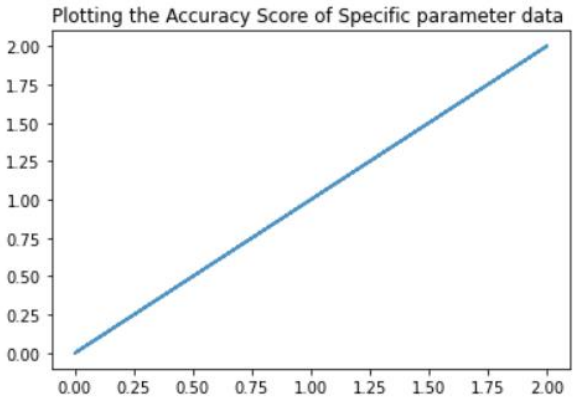
```
# plot your different results
import matplotlib.pyplot as plt

plt.plot(data)
plt.title("Plotting the Given Blobs Data Set")
plt.show()

plt.plot(test_data_predicted,'-',alpha=0.8)
plt.title("Plotting the KNN Classifier with Specific parameter predicted data")
plt.show()

plt.plot(learn_data_predicted,'-',alpha=0.8)
plt.title("Plotting the KNN Classifier with No parameter predicted data")
plt.show()

plt.plot(test_data_predicted, test_labels,'-',alpha=0.8)
plt.title("Plotting the Accuracy Score of Specific parameter data ")
plt.show()

plt.plot(learn_data_predicted, train_labels,'-',alpha=0.8)
plt.title("Plotting the Accuracy Score of No parameter data  ")
plt.show()
```

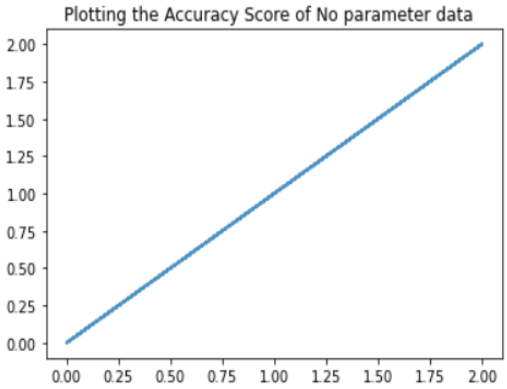
Like this we can plot many data variables and the graphs looks like the below











References:

https://www.tutorialspoint.com/google_colab/google_colab_graphical_outputs.htm

https://colab.research.google.com/notebooks/charts.ipynb

https://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm