Machine Learning

Olivetti Face Data – Gaussian Naive Bayes Classifier (Single Link Clustering)

Advanced Gaussian Naive Bayes classifier (NBC) - with single-link clustering algorithm
Download the Olivetti faces dataset.
Visit https://scikit-learn.org/0.19/datasets/olivetti_faces.html

There are 40 classes (corresponding to 40 people), each class having 10 faces of the individual; so there are a total of 400 images.
Here each face is viewed as an image of size 64 × 64 (= 4096) pixels; each pixel having values 0 to 255 which are ultimately converted into floating numbers in the range [0,1].

Split the dataset into train and test parts such that there are 320 images, 8 images per person (8 X 40) for training and 80 images, 2 images per person, (2 X 40) for testing.

Repeat the experiment using 10 different random splits having 320 training faces and 80 test faces as specified above and report the average accuracy.

Cluster the 4096 features into K = 1200; 1600; 2000; 3000 clusters using the single-link algorithm and represent each cluster by its centroid.
So,320 X 4096 training data matrix is reduced to 320 X K matrix and 80 X 4096 test data matrix is reduced to 80 X K, for a given K. Classify the test data, of size 80 X K using the Gaussian NBC and report the test accuracy
Use the Gaussian Naive Bayes classifier (NBC) to classify the test data and report the results

CODE:

Please find the code committed for Gaussian NBC using single-link clustering as *NaiveBayesClassifier_SingleLinkClustering_OlivettiFaceData_Impl.py*

- **Agglomerative Clustering** from sklearn is used for getting **single link** clustering to reduce the feature set from 4096 to different ranges of 1200, 1600, 2000 and 3000.
- For each of these features, **labels_** is obtained that contains the cluster label for each feature.
- Looping through the **labels_,** the centroid of each cluster is found and represented as the new feature value.
- In this manner, 320 * K train samples are created and 80*K test samples are created.
- **Gaussian NBC** is then applied over this modified dataset to find the accuracy.
- 10 iterations are performed to provide the average accuracy in 10 iterations.

RESULT:

```
accuracy_score for iteration 1 and K value 1200 =  0.225
accuracy_score for iteration 2 and K value 1200 =  0.2625
accuracy_score for iteration 3 and K value 1200 =  0.2375
accuracy_score for iteration 4 and K value 1200 =  0.1875
accuracy_score for iteration 5 and K value 1200 =  0.2125
accuracy_score for iteration 6 and K value 1200 =  0.1625
```

accuracy_score for iteration 7 and K value 1200 = 0.175
accuracy_score for iteration 8 and K value 1200 = 0.1375
accuracy_score for iteration 9 and K value 1200 = 0.2125
accuracy_score for iteration 10 and K value 1200 = 0.15

Average accuracy for K value 1200 = 0.19624999999999998

accuracy_score for iteration 1 and K value 1600 = 0.2
accuracy_score for iteration 2 and K value 1600 = 0.1875
accuracy_score for iteration 3 and K value 1600 = 0.1
accuracy_score for iteration 4 and K value 1600 = 0.1875
accuracy_score for iteration 5 and K value 1600 = 0.175
accuracy_score for iteration 6 and K value 1600 = 0.1875
accuracy_score for iteration 7 and K value 1600 = 0.2125
accuracy_score for iteration 8 and K value 1600 = 0.25
accuracy_score for iteration 9 and K value 1600 = 0.2375
accuracy_score for iteration 10 and K value 1600 = 0.1625

Average accuracy for K value 1600 = 0.19

accuracy_score for iteration 1 and K value 2000 = 0.2375
accuracy_score for iteration 2 and K value 2000 = 0.1875
accuracy_score for iteration 3 and K value 2000 = 0.275
accuracy_score for iteration 4 and K value 2000 = 0.25
accuracy_score for iteration 5 and K value 2000 = 0.15
accuracy_score for iteration 6 and K value 2000 = 0.1625
accuracy_score for iteration 7 and K value 2000 = 0.2375
accuracy_score for iteration 8 and K value 2000 = 0.275
accuracy_score for iteration 9 and K value 2000 = 0.175
accuracy_score for iteration 10 and K value 2000 = 0.175

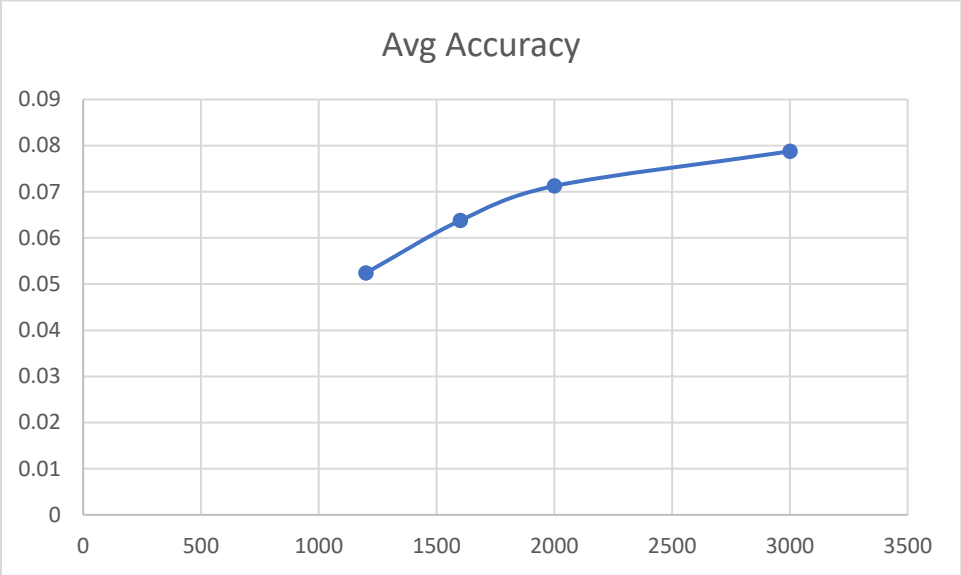
Average accuracy for K value 2000 = 0.2125

accuracy_score for iteration 1 and K value 3000 = 0.2375
accuracy_score for iteration 2 and K value 3000 = 0.175
accuracy_score for iteration 3 and K value 3000 = 0.1875
accuracy_score for iteration 4 and K value 3000 = 0.2125
accuracy_score for iteration 5 and K value 3000 = 0.1375
accuracy_score for iteration 6 and K value 3000 = 0.2375
accuracy_score for iteration 7 and K value 3000 = 0.2125
accuracy_score for iteration 8 and K value 3000 = 0.2
accuracy_score for iteration 9 and K value 3000 = 0.225
accuracy_score for iteration 10 and K value 3000 = 0.2

Average accuracy for K value 3000 = 0.20249999999999999

PLOT:

Here is a plot of K (number of clusters) vs Avg Accuracy.



INFERENCE/ANALYSIS:

- **Single link Clustering** takes the minimum distance between members of the two clusters.
- The dataset has 4096 features and using single link clustering, in this problem, we are reducing the feature set to different K values of 1200, 1600, 2000 and 3000. In each of these clusters, the value used to represent the features is not the feature itself, but its centroid.
- **The features are themselves independent and clustering does not help in this scenario as it simply identifies the centroid of each cluster and that does not help to represent the dataset completely.**
- **Gaussian NBC does not work well in this context and hence we see accuracy of classification is really low.**
- **As K increases, we see there is slight increase in overall avg accuracy.**

As Gaussian or Bernoulli NBC is applied over Olivetti dataset completely, the average accuracy of classification is very high. However, when clustering is applied, whether its single link, complete link or K-means++, the feature set is reduced and the representation is modified. Complete link-based clustering provides better accuracy than K means++ and Single link accuracy. This impacts the overall avg accuracy of the Gaussian NBC model.

RESOURCES USED FOR THE ASSIGNMENT:

- **Environment:**
  Anaconda, Jupyter notebook
- **Software :**
  Python
  Python libraries/modules: Pandas, Numpy, SkLearn etc