CPSC 340 Assignment 2 (due Friday January 26th at 9:00pm)

Ting Hao Ng, Beverly Low

1 Naive Bayes

1.1 Naive Bayes by Hand

Consider the dataset below, which has 10 training examples and 2 features:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Suppose you believe that a naive Bayes model would be appropriate for this dataset, and you want to classify the following test example:

$$\hat{x} = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

(a) Compute the estimates of the class prior probabilities (you don't need to show any work):

- $p(y = 1)$: 0.6
- $p(y = 0)$: 0.4

(b) Compute the estimates of the 4 conditional probabilities required by naive Bayes for this example (you don't need to show any work):

- $p(x_1 = 1|y = 1)$: 0.5
- $p(x_2 = 0|y = 1)$: $\frac{1}{3}$
- $p(x_1 = 1|y = 0)$: 1
- $p(x_2 = 0|y = 0)$: $\frac{3}{4}$

(c) Under the naive Bayes model and your estimates of the above probabilities, what is the most likely label for the test example? (Show your work.)

- $p(x_1 = 1, x_2 = 0|y = 1)p(y = 1) = p(x_1 = 1|y = 1)p(x_2 = 0|y = 1)p(y = 1) = 0.5 * \frac{2}{6} * 0.6 = 0.1$
- $p(x_1 = 1, x_2 = 0|y = 0)p(y = 0) = p(x_1 = 1|y = 0)p(x_2 = 0|y = 0)p(y = 0) = 1 * \frac{3}{4} * 0.4 = 0.3$
- Since the second probability is larger than the first, the most likely label for this test example is 0

1.2 Bag of Words

1. Which word corresponds to column 50 of X? **'lunar'**
2. Which words are present in training example 500? **'car', 'fact', 'gun', 'video'**
3. Which newsgroup name does training example 500 come from? **'talk.*'**

1.3 Naive Bayes Implementation

Completed in *naive_bayes.py*.
Our model generally performs better than the random forest, producing a validation error of 0.188 compared to the validation error produced by the random forest which is often closer to 0.200. Our naive bayes model also produces similar results to scikit-learn's implementation, obtaining a comparable validation error of 0.188 versus scikit-learn's 0.187.

1.4 Laplace smoothing

1. Completed in *naive_bayes.py*.
2. We needed to modify the fit function in order to perform the smoothening and storing the smoothened conditional probabilities into *p_xy*.The predict function remained unchanged.
3. The default valued used for laplace smoothening in scikit-learn's implementation is 1.0. Using this same value, our naive bayes model produce exactly the same validation error of 0.187

1.5 Runtime of Naive Bayes for Discrete Data

$O(tdk)$. For each test examples in t, all we have to do is to calculate the product of the conditional probability of d features, for each label. The probabilities obtained for the k labels are then compared and the label with the highest value is chosen.

2 Random Forests

2.1 Implementation

1. The random tree model does not have a training error of 0 as compared to the decision tree. This is because the random tree model does not use the entire training data, but rather a subset of it to train on. Furthermore, it only takes into account a random subset of $\sqrt{d}$ features. Hence, when evaluating its error against the training set, it would be unlikely for the random tree to overfit on examples that it wasn't trained on.
2. Completed in $random_forest.py$

3. We obtained a training error of 0.000 and a testing error of 0.167.The training error of 0 is much lower than the training error of a single random tree. In a random forest, each training example would have been fitted to a significant proportion of its trees. Since the mode is used during predictions, the trees which are trained on a particular example would likely have the same predictions. Hence, it is possible for the random forest to have a training error of 0, unlike a single random tree. The testing error of the random forest is also significantly lower than both the decision tree and the random tree. By training 50 different random trees on a subset of the training data, the overfitting on training data is reduced. As the error from each tree which is independent, the overall error by using the mode of each tree's prediction will be lowered as well, giving rise to a lower error on the test set.
4. The accuracy of our implementation and scikit-learn's implementation is similar. While testing error varies, the difference between each implementation is often less than 0.005. However, our implementation is much slower, taking 17.623s to fit the training data and make predictions, as compared to scikit-learn's implementation which took 0.111s.This is likely because scikit-learn uses joblib which allows for building of trees in parallel to optimize for speed. Our implementation builds each tree sequentially which is much slower.

2.2 Very-Short Answer Questions

1. One disadvantage would be the longer time required to fit the model and make predictions when using a random forest with a larger number of trees.
2. Your random forest classifier has a training error of 0 and a very high test error. Which ones of the following could help performance?
   (a) Decrease the maximum depth of the trees in your forest.
   (b) Increase the amount of data you consider for each tree (Collect more data and use $2n$ objects instead of n).
   (c) Decrease the number of features you consider for each tree.
3. We could train the model on data from various languages and also perform data augmentation by shifting the pitch of the audio.

3 Clustering

3.1 Selecting among k-means Initializations

1. Completed in *kmeans.py*.
2. The error tend to decrease at a decreasing rate before reaching a plateau.

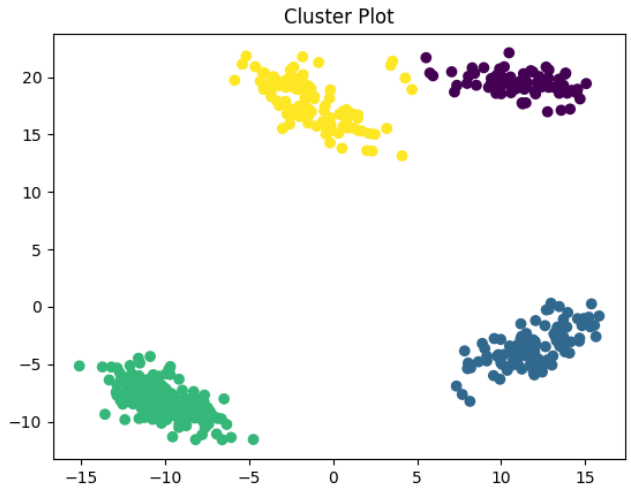


Figure 1: Clustering with lowest error

3.

4. (a) n_clusters: this hyperparameter determines the number of groupings which the clustering algorithm will form.

   (b) init: this hyperparameter determines the method used to initialize the algorithm. KMeans++ can be selected to speed up convergence. Random can also be selected to use random observations for initial centroids.

   (c) n_init: hyperparameter that determines the number of times the algorithm will run with different centroids seed. The final results will be selected based the best output of n_init consecutive runs.

   (d) max_iter: hyperparameter that determines the maximum number of iterations the algorithm will perform if it does not fully converge.

3.2 Selecting k in k-means

We now turn to the task of choosing the number of clusters k.

1. The *error* function should not be used to choose k as the main goal of a clustering algorithm is to group similar objects together in a meaningful way. If the *error* function is used, the ideal number of k would hence be the number of examples in the dataset which would result in an error of 0. However, these clusters would not give us any meaningful insight into the data.

2. Evaluating the *error* function on test data would not be useful. The error can be reduced by increasing k, but reduce the compression of data in a single cluster. The optimal choice of k will strike a balance between maximum compression of the data using a single cluster, and maximum accuracy by assigning each data point to its own cluster, which cannot be found by simply using test error.

3.

4. From this plot, I would select the value of k to be 3. The "elbow" is the most obvious at $k = 3$. This allows us to have a smaller number of clusters and still have a relatively small error. any k larger than 3 would lead to a diminishing decrease in the error value.

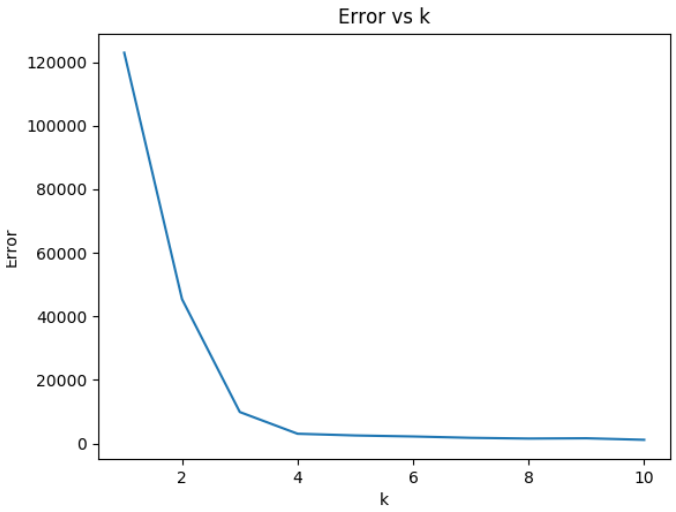


Figure 2: Error vs k-value

3.3 k-medians

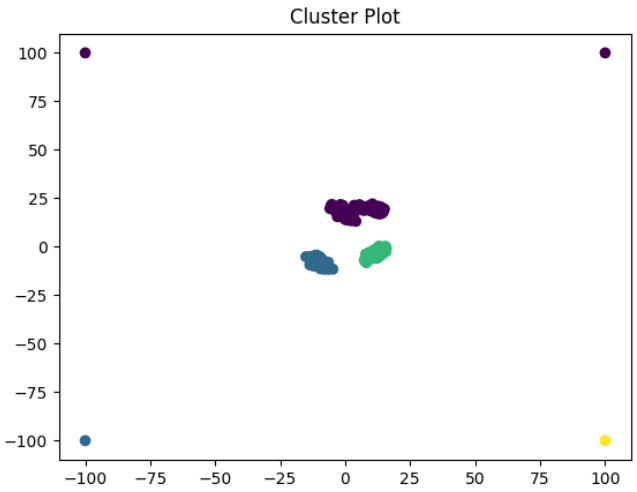


Figure 3: Clustering with lowest error

1. I am not satisfied with the results as the clusters are not meaningful and have extremely different sizes. The purple cluster seems like 2 separate clusters molded together, and the yellow cluster at the bottom only includes one example.

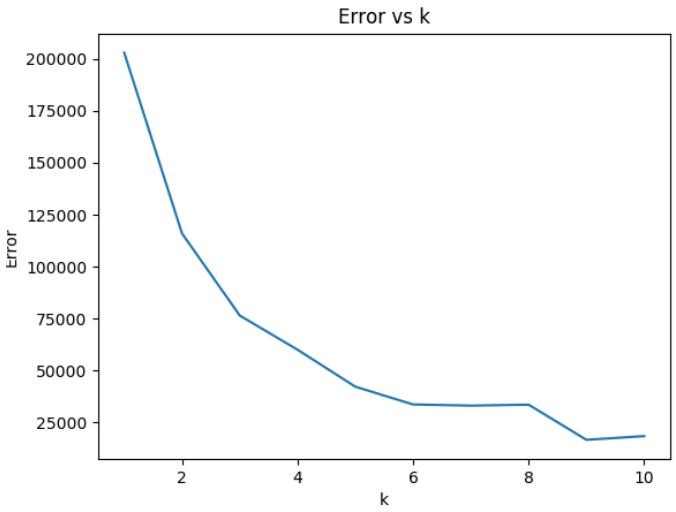


Figure 4: Error vs k value

2. From this plot, there is no obvious elbow in which we can select the value of k. Possible values that can be selected based of this graph are 3, 5, 6 or 9.

Figure 5: Clustering of K Medians

3. Code completed in *kmedians.py*.

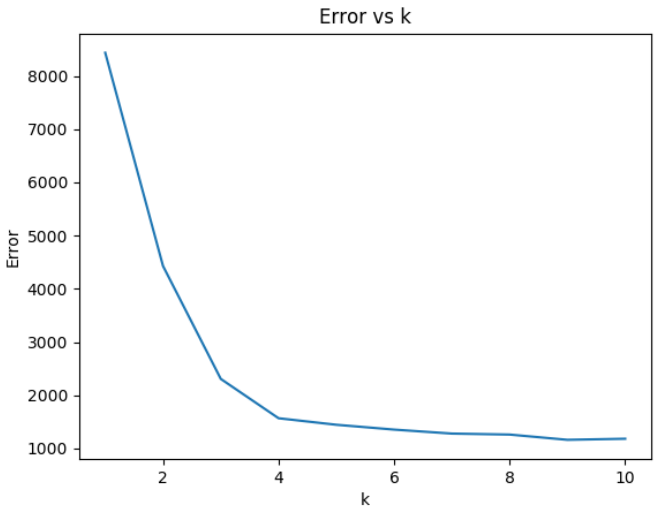
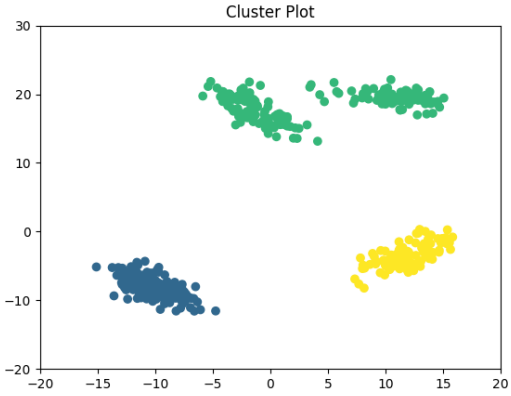


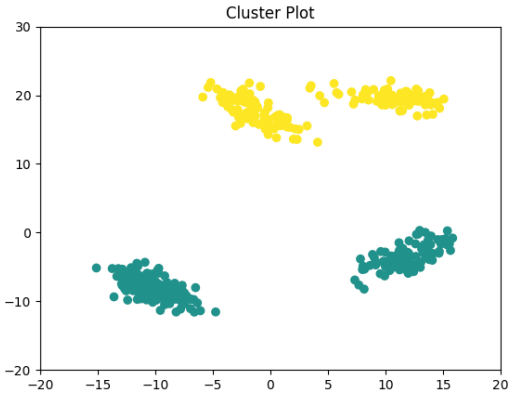
Figure 6: Error vs k (kmedians)

4. Using the elbow strategy, I would select $k = 3$, as the elbow is the most obvious. We are satisfied with the result as this k value strikes a balance between minimizing the number of clusters and minimizing the error function as well.Furthermore, from Figure 5, we can see that the clusters are more balanced and the outliers do not affect the clusters as much as the kmeans.

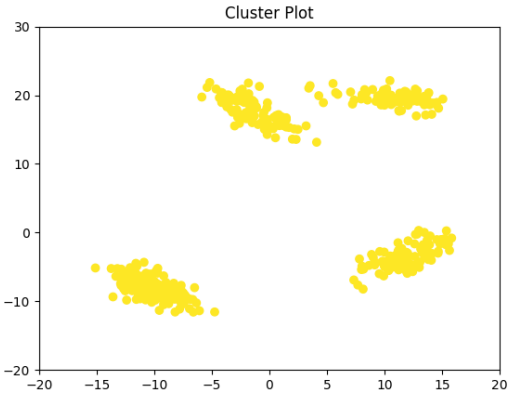
1. $eps = 3$, $min_samples = 3$



2. $eps = 5$, $min_samples = 3$



3. $eps = 15$, $min_samples = 3$



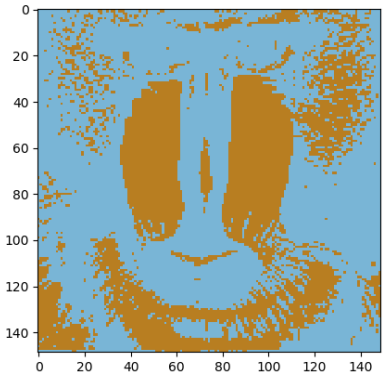
4. $eps = 20$, $min_samples = 3$

Clusters with various eps and min_points

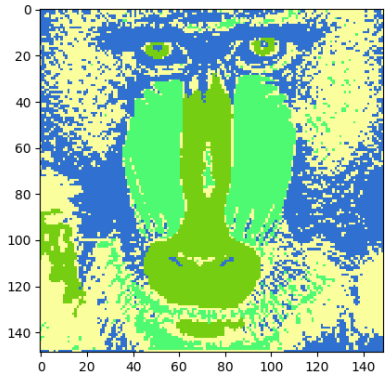
3.4 Density-Based Clustering

3.5 Very-Short Answer Questions

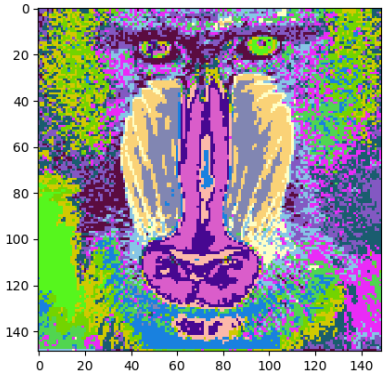
1. No, it depends on the initial values of the centroids that the algorithm is initialized to. It is possible for the algorithm to converge to a local minima that is not the most optimal.
2. The value of k would be equal to the number of points. The means would then be the point itself which will yield a total sum of zero distance. However, such a value of k will not give us any meaningful grouping of the variables and we are no better off than before the algorithm was performed.
3. k-means would not be able to find the true clusters if the clusters have non-convex boundaries.
4. Without performing normalization, the clusters would likely be formed along the axis with the smaller scale. There will unlikely be any clusters formed across the axis with the much larger scale.
5. The key advantage is that we will likely be able to detect outliers with a higher success rate. However, the disadvantage is that we have to commit resources to obtaining or labeling the outliers.



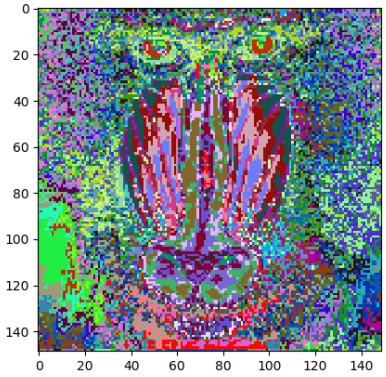
1-bit per pixel



2-bit per pixel



4-bit per pixel



6-bit per pixel

Images formed by different bits per pixel

4 Vector Quantization

1. Completed in *quantize_image.py*.
2.
3. When using lesser bits per pixel, the colours learned tend to be fewer and more neutral as it has to cover a wider range of colours. As the number of bits per pixel increases, the colours learnt becomes more distinct from the others and vibrant.