CPSC 340 Assignment 2

Henry Deng
Student Number: 41584103

1 Naive Bayes

1.1 Naive Bayes by Hand

Rubric: {reasoning:3}

(a)

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

(b)

- $p(x_1 = 1|y = 1) = \frac{1}{2}$
- $p(x_2 = 0|y = 1) = \frac{1}{3}$
- $p(x_1 = 1|y = 0) = 1$
- $p(x_2 = 0|y = 0) = \frac{3}{4}$

(c)

- $p(\hat{y} = 0|\hat{x}) \propto p(\hat{x_1} = 1|\hat{y} = 0) * p(\hat{x_2} = 0|\hat{y} = 0) * p(\hat{y} = 0) = 1 * \frac{3}{4} * 0.4 = 0.3$
- $p(\hat{y} = 1|\hat{x}) \propto p(\hat{x_1} = 1|\hat{y} = 1) * p(\hat{x_2} = 0|\hat{y} = 1) * p(\hat{y} = 1) = \frac{1}{2} * \frac{1}{3} * 0.6 = 0.1$

Naive bayes predicts $\hat{y} = 0$ since $p(\hat{y} = 0|\hat{x})$ is bigger than $p(\hat{y} = 1|\hat{x})$

1.2 Bag of Words

1. league
2. car, engine, evidence, problem, system
3. 2

1.3 Naive Bayes Implementation

`https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_a2/blob/master/code/naive_bayes.py`

Test error after the fix is 0.188
The accuracy of the validation error is close to sklearn's value of 0.187, but a little off from the random forest value of 0.197.

1.4 Laplace smoothing

1. Refer to:

`https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_a2/blob/master/code/naive_bayes.py`

2 I had to modify my fit function

3 The scikit version uses the default value of $\alpha = 1$ for Laplace smoothing. When I passed in $\beta = 1$, I got the same result as the scikit version.

1.5 Runtime of Naive Bayes for Discrete Data

The cost of classifying t test examples is $O(tdk)$

2 Random Forests

2.1 Implementation

1. The random tree does not have a training error of 0 because we only randomly sample a small number of features, which may not cover all possible number of features, so we will never overfit the data regardless of the depth of the tree.
2. Refer to: `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_a2/blob/master/code/random_forest.py`
3. Fig 1. Decision Tree vs. Random Tree vs. Random Forest.

```
Our implementations:
  Decision tree info gain
    Training error: 0.000
    Testing error: 0.367
  Random tree info gain
    Training error: 0.201
    Testing error: 0.527
  Random forest info gain
    Training error: 0.000
    Testing error: 0.197
```

   Yes. These results are expected because decision trees overfit and therefore, have a training error of 0 and a high test error. A random tree cannot cover all features so the training and test errors are high. When we increase the number of random trees into a forest, we cover more features and avoid overfitting; therefore, train error and test error are reduced dramatically.
4. Our implementation was much slower than scikits RandomForestClassifier, but accuracy was similar with our testing error clocked at 0.197 and scikit's testing error at 0.186.

2.2 Very-Short Answer Questions

1. One disadvantage is the increase in calls to the random forest classifier, which could lead to a bad running time.
2. (a) Decrease the maximum depth of the trees in your forest.

   (b) Decrease the amount of data you consider for each tree.

   (c) Decrease the number of features you consider for each tree.

3. Add transformed data during training, ie. warping the audio to create multiple versions of the audio

3 Clustering

3.1 Selecting among k-means Initializations

1. Refer to: `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_a2/blob/master/code/kmeans.py`
2. The value of the k-means error decreases and then stabilizes
3. Plot is provided here:

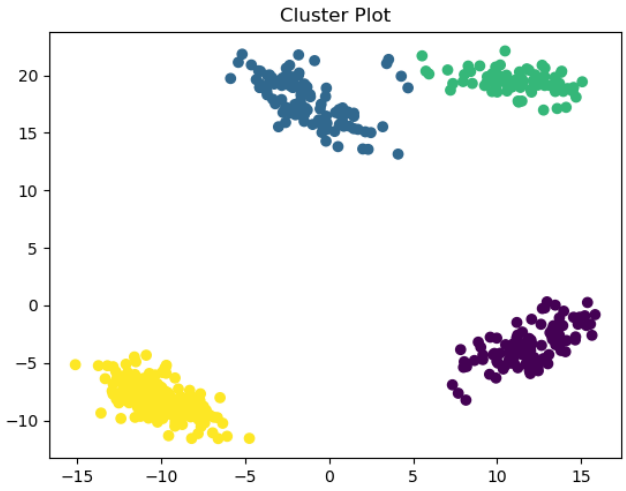


Fig. 2: Plot of the clustering obtained by running k-means 50 times and taking the lowest error

4. n-clusters = number of clusters, init = used to increase values, n-init = number of initializations, max.iterations = how many times you run kmeans

3.2 Selecting k in k-means

1. The error function will keep decreasing as k increases. Therefore, we will always select the largest value of k, which is not a suitable method to select k.
2. If we increase the value of k, we will get more clusters in our test data. As we increase in clusters, the mean of each cluster is going to be closer to the true value of each point. This can lead to overfitting of our data, so the error function is not suitable for selecting k even with test data.
3. Attached Plot:

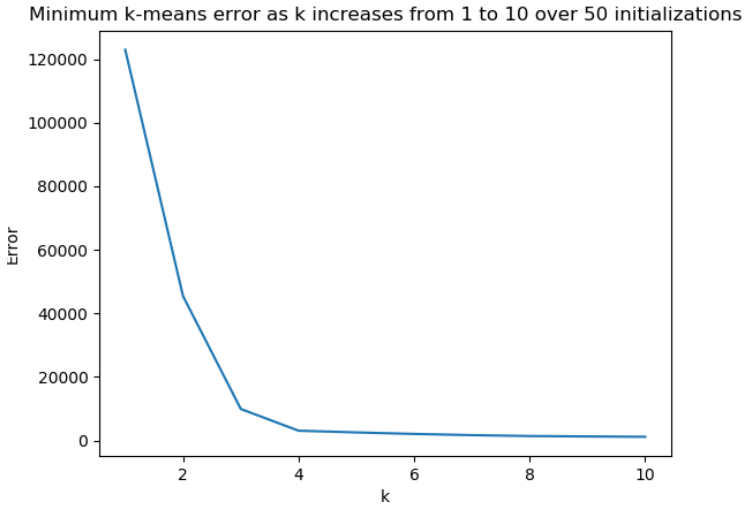
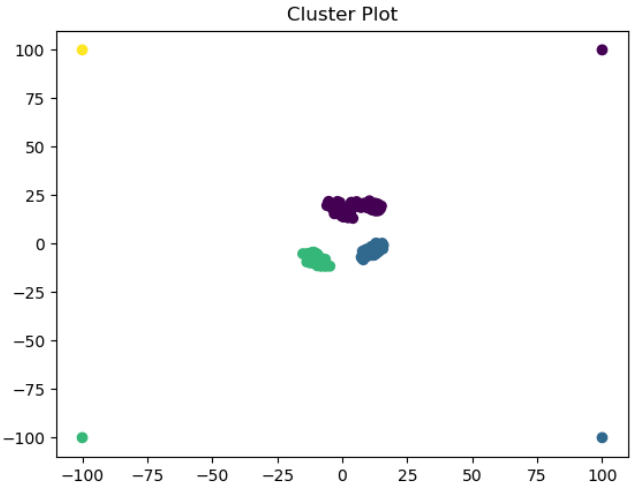


Fig. 3: Plot of the minimum error found across 50 random initializations

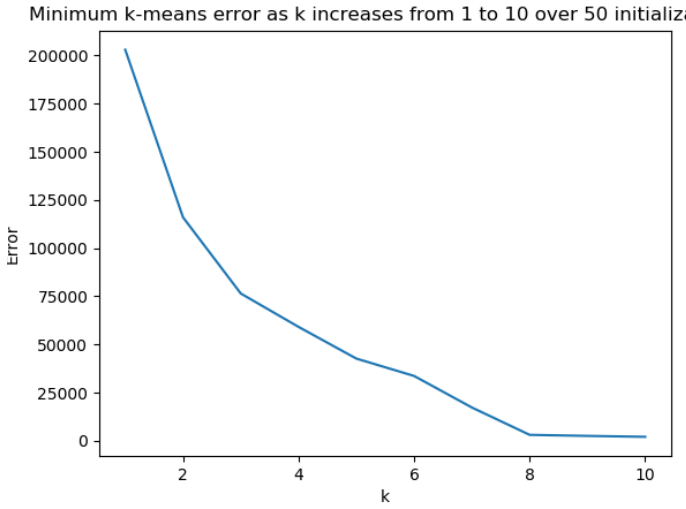
4. Looking at the plot, the elbow method would select k = [2, 3, 4]. I would personally choose k = 3 due to the decreased error.

3.3 k-medians

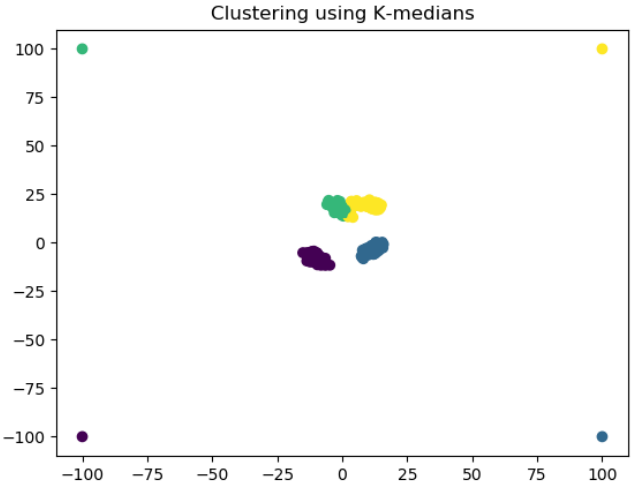
1. The result is not satisfactory because the plot is an example of bad clustering since the outliers are not detected; thus, resulting in high error



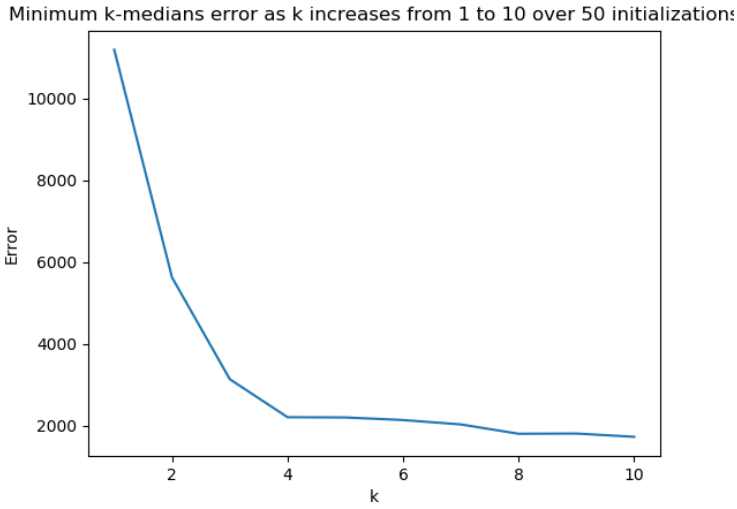
2. The elbow method would select k = [2, 4, 8] since they have the steepest slopes.



3. Refer to: `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_a2/blob/master/code/kmedians.py` Plot:



4. This time, the elbow method would select k = [2, 3, 4] since they have the steepest slopes. Yes, I am satisfied with this result since it is similar to the k-means result back in subsection 3.2. The variance in selecting k has decreased dramatically, which is a good thing.



3.4 Density-Based Clustering

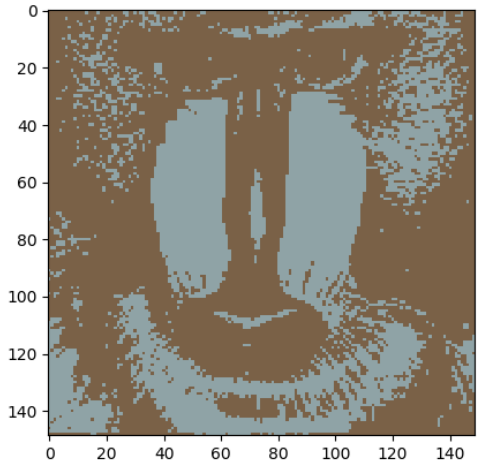
1. eps = 2, minPts = 2
2. eps = 4, minPts = 2
3. eps = 15, minPts = 2
4. eps = 20, minPts = 2

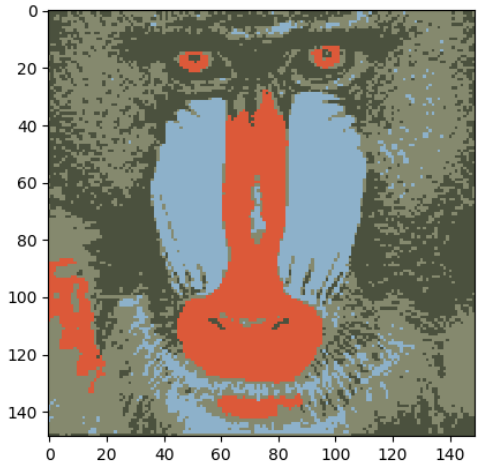
3.5 Very-Short Answer Questions

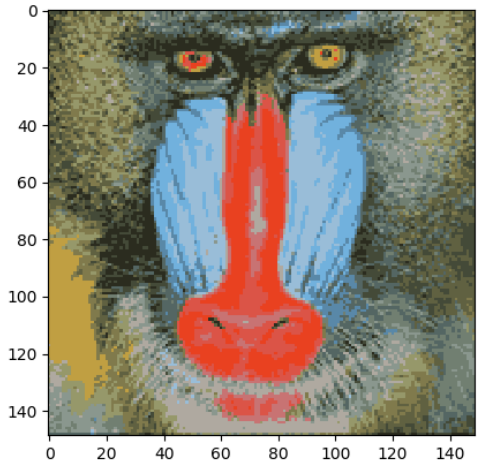
1. No, depends on the initial clusters.
2. When k = n. This value is not useful because we could have imperfect clustering but perfect test data.
3. Hierarchical clustering would make it hard for k-means to find the true clusters
4. This would affect the core assignment because there's different weights associated with kilograms vs. milligrams
5. The key advantage of supervised outlier detection is that we know what the outliers could be, so we can detect them. The disadvantage is that we might not be able to detect new outliers.

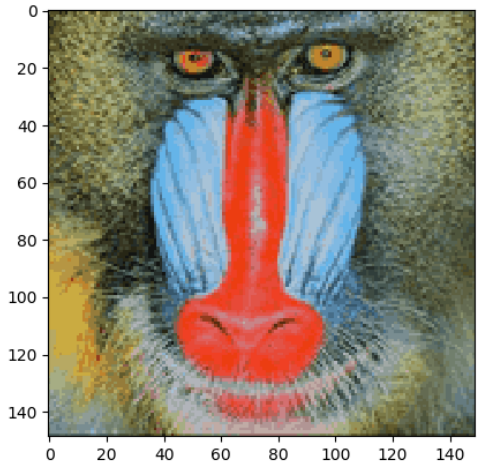
4 Vector Quantization

1. Refer to: `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/c1z8_a2/blob/master/code/quantize_image.py`
2. Quantization bit = 1, 2, 4, and 6 respectively shown below in order:


0
20
40
60
80
100
120
140
0 20 40 60 80 100 120 140


0
20
40
60
80
100
120
140
0 20 40 60 80 100 120 140





3. At each phase, we learn 2^b colours – where b is the number of bits. When b = 6, we have learned most of the colours which is why the image looks very similar to the original image.