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

1 Naive Bayes

1.1 Naive Bayes by Hand

Rubric: {reasoning:3}

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):

- $p(y = 1) = \frac{3}{5}$.
- $p(y = 0 = \frac{2}{5}$.

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

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

(c) 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.)

Ans: $y = 0$ is the most likely.
Sol:

$$p(y = 1|x_1 = 1, x_2 = 0) = \frac{p(x_1 = 1, x_2 = 0|y = 1)}{p(x_1 = 1, x_2 = 0)} \propto p(x_1 = 1|y = 1)p(x_2 = 0|y = 1)p(y = 1) = 1/2\times1/3\times3/5 = 1/10$$

$$p(y=0|x_1=1,x_2=0) = \frac{p(x_1=1,x_2=0|y=0)}{p(x_1=1,x_2=0)} \propto p(x_1=1|y=0)p(x_2=0|y=0)p(y=0) = 1\times 3/4\times 2/5 = 3/10$$

1.2 Bag of Words

Rubric: {reasoning:3}

Answer the following:

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

1.3 Naive Bayes Implementation

Rubric: {code:5}

Modify this function so that `p_xy` correctly computes the conditional probabilities of these values based on the frequencies in the data set. Hand in your code and the validation error that you obtain. Also, briefly comment on the accuracy as compared to the random forest and scikit-learn's naive Bayes implementation.
d = 100
n = 8121
t = 8121
Num classes = 4
Random Forest (sklearn) validation error: 0.202
Naive Bayes (ours) validation error: 0.188
Naive Bayes (sklearn) validation error: 0.187
Naive Bayes validation error is similar to sklearn bayes models.

1.4 Laplace smoothing

Rubric: {code:1}

Do the following:

1. Modify your code so that it uses Laplace smoothing, with β as a parameter taken in by the constructor.
   `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/k6y1b\_a2/blob/master/code/naive_bayes.py`
2. Did you need to modify your fit function, predict function, or both?
   I need to modify fit function. predict function will be automatically modified once we modify fit function because we predict using fit function.
3. Take a look at the documentation for the scikit-learn version of naive Bayes the code is using. How much Laplace smoothing does it use by default? Using the same amount of smoothing with your code, do you get the same results?
   Default is $\alpha = 1$. Using $\beta = 1$ in my Naive, I get the same result as follows.
   d = 100
   n = 8121

```
t = 8121
Num classes = 4
Random Forest (sklearn) validation error: 0.196
Naive Bayes (ours) validation error: 0.187
Naive Bayes (sklearn) validation error: 0.187
```

1.5 Runtime of Naive Bayes for Discrete Data

Rubric: {reasoning:3}

What is the cost of classifying t test examples with the model?
O(tkd).This is because we need 3 for loops. We should go through t objects, and for each object we go through d features, and for each elements of X matrix we go through k lables to classify.

2 Random Forests

2.1 Implementation

Rubric: {reasoning:7}

The file *vowels.pkl* contains a supervised learning dataset where we are trying to predict which of the 11 "steady-state" English vowels that a speaker is trying to pronounce.

You are provided with a `RandomStump` class that differs from `DecisionStump` in two ways: it uses the information gain splitting criterion (instead of classification accuracy), and it only considers $\lfloor\sqrt{d}\rfloor$ randomly-chosen features.[1] You are also provided with a `RandomTree` class that is exactly the same as `DecisionTree` except that it uses `RandomStump` instead of `DecisionStump` and it takes a bootstrap sample of the data before fitting. In other words, `RandomTree` is the entity we discussed in class, which makes up a random forest.

If you run `python main.py -q 2` it will fit a deep `DecisionTree` using the information gain splitting criterion. You will notice that the model overfits badly.

1. Why doesn't the random tree model have a training error of 0?
   This is because the random tree model is using bootstrap method, and the method is replicating same data. That is, some of the training examples are not included in the bootstrap sample.

2. Create a class `RandomForest` in a file called `random_forest.py` that takes in hyperparameters `num_trees` and `max_depth` and fits `num_trees` random trees each with maximum depth `max_depth`. For prediction, have all trees predict and then take the mode.
   `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/k6y1b_a2/blob/master/code/random_forest.py`

3. Using 50 trees, and a max depth of ∞, report the training and testing error. Compare this to what we got with a single `DecisionTree` and with a single `RandomTree`. Are the results what you expected? Discuss.
   Decision tree info gain
   Training error: 0.000
   Testing error: 0.367

[1] The notation $\lfloor x\rfloor$ means the "floor" of x, or "x rounded down". You can compute this with `np.floor(x)` or `math.floor(x)`.

Random tree info gain
Training error: 0.163
Testing error: 0.492
Random forest info gain
Training error: 0.000
Testing error: 0.159
Training error is 0.000, Testing error is 0.159, which are smallest among the three models. I expected that.

4. Compare your implementation with scikit-learn's `RandomForestClassifier` for both speed and accuracy, and briefly discuss. You can use all default hyperparameters if you wish, or you can try changing them.
Our implementations: Random forest info gain
Training error: 0.000
Testing error: 0.193
My random forst took 13.747773 seconds
Random forest info gain, more trees
Training error: 0.000
Testing error: 0.174
sikit learn random forst took 0.112223 seconds
So my random forest is slower and has higher testing error.

2.2 Very-Short Answer Questions

Rubric: {reasoning:3}

1. What is a a disadvantage of using a very-large number of trees in a random forest classifier?
It's slow. You may die before you see the result.
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) Increase the maximum depth of the trees in your forest.
   (b) Decrease the maximum depth of the trees in your forest.
   (c) Increase the amout of data you consider for each tree (Collect more data and use $2n$ objects instead of n).
   (d) Decrease the amount of data you consider for each tree (Use $0.8n$ objects instead of n).
   (e) Increase the number of features you consider for each tree.
   (f) Decrease the number of features you consider for each tree.

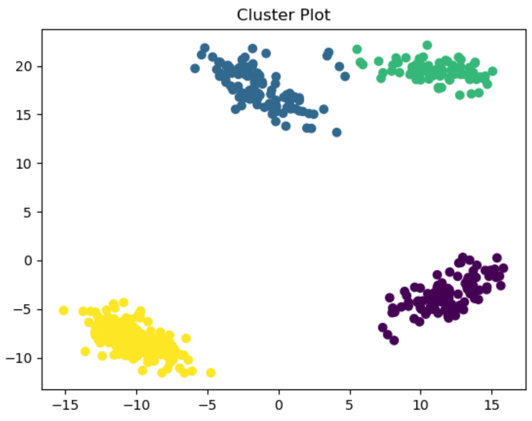
   (b), (c),(f)
3. Suppose that you were training on raw audio segments and trying to recognize vowel sounds. What could you do to encourage the final classifier to be invariant to translation?
We should add all of the translated version of data to the training data set.

3 Clustering

3.1 Selecting among k-means Initializations

Rubric: {reasoning:5}

1. In the *kmeans.py* file, add a new function called *error* that takes the same input as the *predict* function but that returns the value of this above objective function. Hand in your code.
   `https://github.ugrad.cs.ubc.ca/CPSC340-2017W-T2/k6y1b_a2/blob/master/code/kmeans.py`
2. What trend do you observe if you print the value of *error* after each iteration of the k-means algorithm?
   It is decreasing!!!
3. Using `plot_2dclustering`, output the clustering obtained by running k-means 50 times (with $k = 4$) and taking the one with the lowest error.



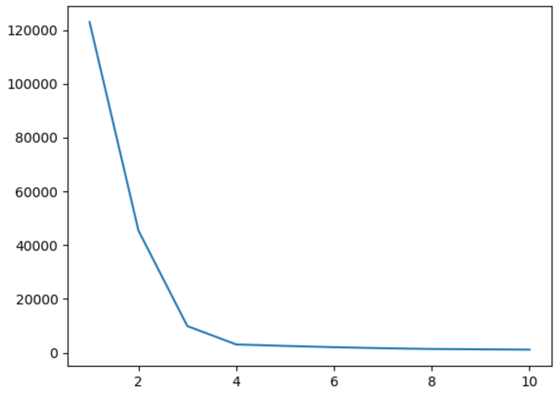
4. Looking at the hyperparameters of scikit-learn's `KMeans`, explain the first four (`n_clusters`, `init`, `n_init`, `max_iter`) very briefly.
   n_clusters=8, the number of clusters to form
   init : methods to choose initial points of means, either randomly and faster version(k-means++) k-means++, random or an ndarray
   n_init : int number of times that Kmeans algorithm will run, default: 10
   max_iter : maximum number of interations in a single run of Kmeans. int, default: 300

3.2 Selecting k in k-means

Rubric: {reasoning:5}

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

1. Explain why the *error* function should not be used to choose k.
   When k is larger, the error will be smaller. So this way will make K only bigger and bigger.
2. Explain why even evaluating the *error* function on test data still wouldn't be a suitable approach to choosing k. **Same reason. Even if this is the test data set, if k is a lot, then the evaluated error will be small.**
3. Hand in a plot of the minimum error found across 50 random initializations, as a function of k, taking k from 1 to 10.



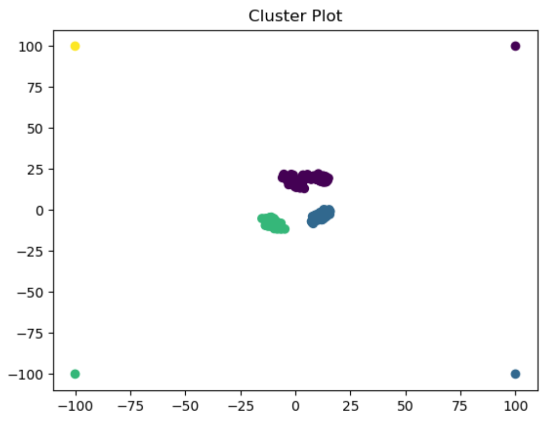
4. The *elbow method* for choosing k consists of looking at the above plot and visually trying to choose the k that makes the sharpest "elbow" (the biggest change in slope). What values of k might be reasonable according to this method?
   I would choose 3.

3.3 k-medians

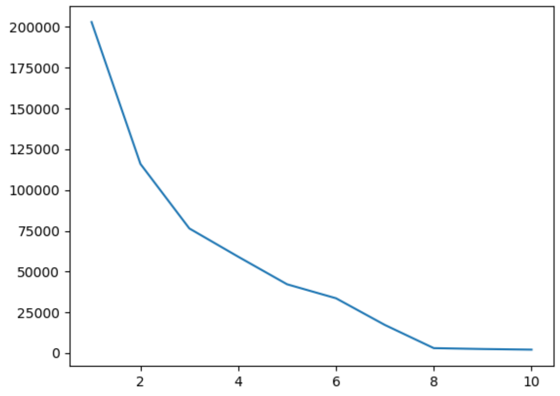
Rubric: {reasoning:5}

The data in *clusterData2.pkl* is the exact same as the above data, except it has 4 outliers that are very far away from the data.

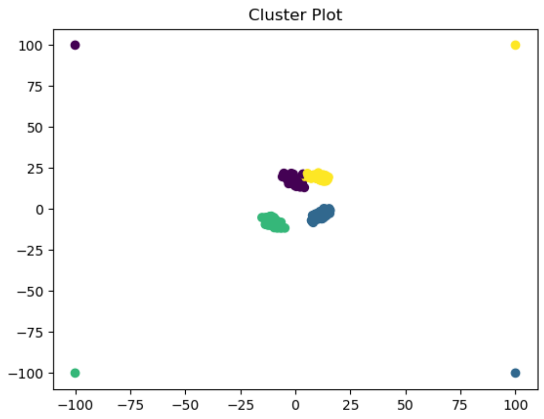
1. Using the `plot_2dclustering` function, output the clustering obtained by running k-means 50 times (with $k = 4$) and taking the one with the lowest error. Are you satisfied with the result?
   NO!!



2. What values of k might be chosen by the elbow method for this dataset?
   Seems to choose 8



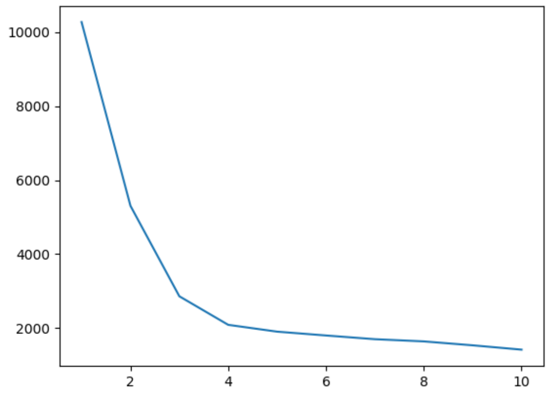
3. Implement the *k-medians* algorithm, which assigns examples to the nearest w_c in the L1-norm and to updates the w_c by setting them to the “median” of the points assigned to the cluster (we define the d-dimensional median as the concatenation of the median of the points along each dimension). Hand in your code and plot obtained with 50 random initializations for $k = 4$.



4. Using the L1-norm version of the error (where y_i now represents the closest median in the L1-norm),

$$f(w_1, w_2, \dots, w_k, y_1, y_2, \dots, y_n) = \sum_{i=1}^{n} \|x_i - w_{y_i}\|_1 = \sum_{i=1}^{n} \sum_{j=1}^{d} |x_{ij} - w_{y_i j}|,$$

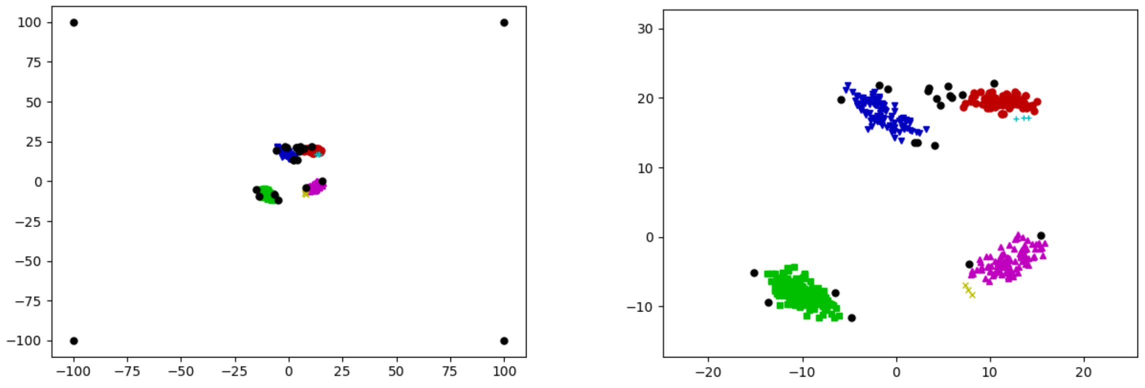
what value of k would be chosen by the elbow method under this strategy? Are you satisfied with this result?
k can be 4. I am fine for that.



3.4 Density-Based Clustering

Rubric: {reasoning:2}

If you run `python main.py -q 3.4`, it will apply the basic density-based clustering algorithm to the dataset from the previous part. The final output should look somewhat like this:



(The right plot is zoomed in to show the non-outlier part of the data.) Even though we know that each object was generated from one of four clusters (and we have 4 outliers), the algorithm finds 6 clusters and does not assign some of the original non-outlier objects to any cluster. However, the clusters will change if we change the parameters of the algorithm. Find and report values for the two parameters, `eps` (which we called the "radius" in class) and `minPts`, such that the density-based clustering method finds:

1. The 4 "true" clusters.
   eps=2 minsamples=3
2. 3 clusters (merging the top two, which also seems like a reasonable interpretaition).
   eps=4 minsamples=3
3. 2 clusters.
   eps=13 minsamples=3
4. 1 cluster (consisting of the non-outlier points).
   eps=16 minsamples=3

3.5 Very-Short Answer Questions

Rubric: {reasoning:3}

1. Does the standard k-means clustering algorithm always yield the optimal clustering solution for a given k?
   No. The initialization dominates this method. And it can converge to sub-optimal solution too.

2. If your set out to minimize the distance between each point and its mean in a k-means clustering, what value of k minimizes this cost? Is this value useful?
   If your k goes larger and larger till k gets n, then it is totally useless and it totally minimizes the cost.But USELESS

3. Describe a dataset with k clusters where k-means would not be able to find the true clusters.
   If true clusters are non-convex, and if the clusters are so close to each other, then k means will not be appropriate.

4. Suppose that you had only two features and that they have very-different scales (like kilograms vs. milligrams). How would this affect the result of density-based clustering?
   If the radius is set to be small, then it might only detect the cluster with small scales(more concentrated points.)If the radius is too large, then all points would be classified as one cluster.

5. Name a key advantage and drawback of using a supervised outlier detection method rather than an unsupervised method?
   You can detect outliers that are very-close to normal points, but you may not be able to detect abnormal outliers such as points which are very far away from clustered points

4 Vector Quantization

Rubric: {reasoning:6}

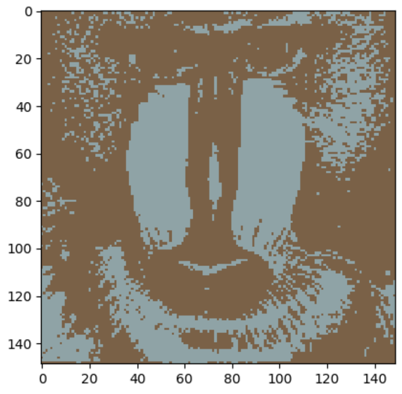
Discovering object groups is one motivation for clustering. Another motivation is *vector quantization*, where we find a prototype point for each cluster and replace points in the cluster by their prototype. If our inputs are images, vector quantization gives us a rudimentary image compression algorithm.

Your task is to implement image quantization in *quantize_image.py* with `quantize` and `dequantize` functions. The `quantize` function should take in an image and, using the pixels as examples and the 3 colour channels as features, run k-means clustering on the data with 2^b clusters for some hyperparameter b. The code should store the cluster means and return the cluster assignments. The `dequantize` function should return a version of the image (the same size as the original) where each pixel's orignal colour is replaced with the nearest prototype colour.

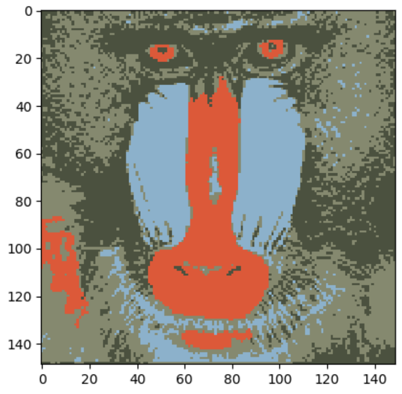
To understand why this is compression, consider the original image space. Say the image can take on the values $0, 1, \ldots, 254, 255$ in each colour channel. Since $2^8 = 256$ this means we need 8 bits to represent each colour channel, for a total of 24 bits per pixel. Using our method, we are restricting each pixel to only take on one of 2^b colour values. In other words, we are compressing each pixel from a 24-bit colour representation to a b-bit colour representation by picking the 2^b prototype colours that are "most representative" given the content of the image. So, for example, if $b = 6$ then we have 4x compression.

The loaded image contains a 3D-array representing the RGB values of a picture. Implement the *quantize* and *dequantize* functions and show the image obtained if you encode the colours using 1, 2, 4, and 6 bits with the provided image. You are welcome to use the provided implementation of k-means or the scikit-learn version.

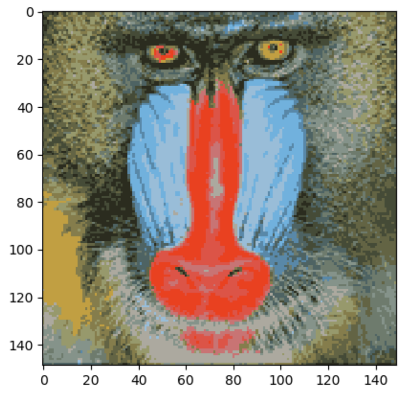
1. Hand in your *quantizeImage* and *deQuantizeImage* functions.
2. Show the image obtained if you encode the colours using 1, 2, 4, and 6 bits per pixel (instead of the original 24-bits).
   for 1



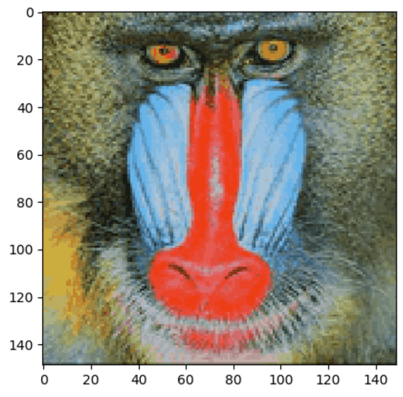
for 2



for 4



for 6



3. Briefly comment on the prototype colours learned in case each, which are saved by the code.
   The dominant color is the first two colors that we got, and then the prototype colors are getting colorfull detailing the picture.