CPSC 340 Assignment 2 (due 2019-01-25 at 11:55pm)

Student name: José Abraham Torres Juárez
Student id: 79507828

Instructions

Rubric: {mechanics:5}

IMPORTANT!!! Before proceeding, please carefully read the general homework instructions at `https://www.cs.ubc.ca/~fwood/CS340/homework/`. The above 5 points are for following the submission instructions. You can ignore the words "mechanics", "reasoning", etc.

We use blue to highlight the deliverables that you must answer/do/submit with the assignment.

1 Training and Testing

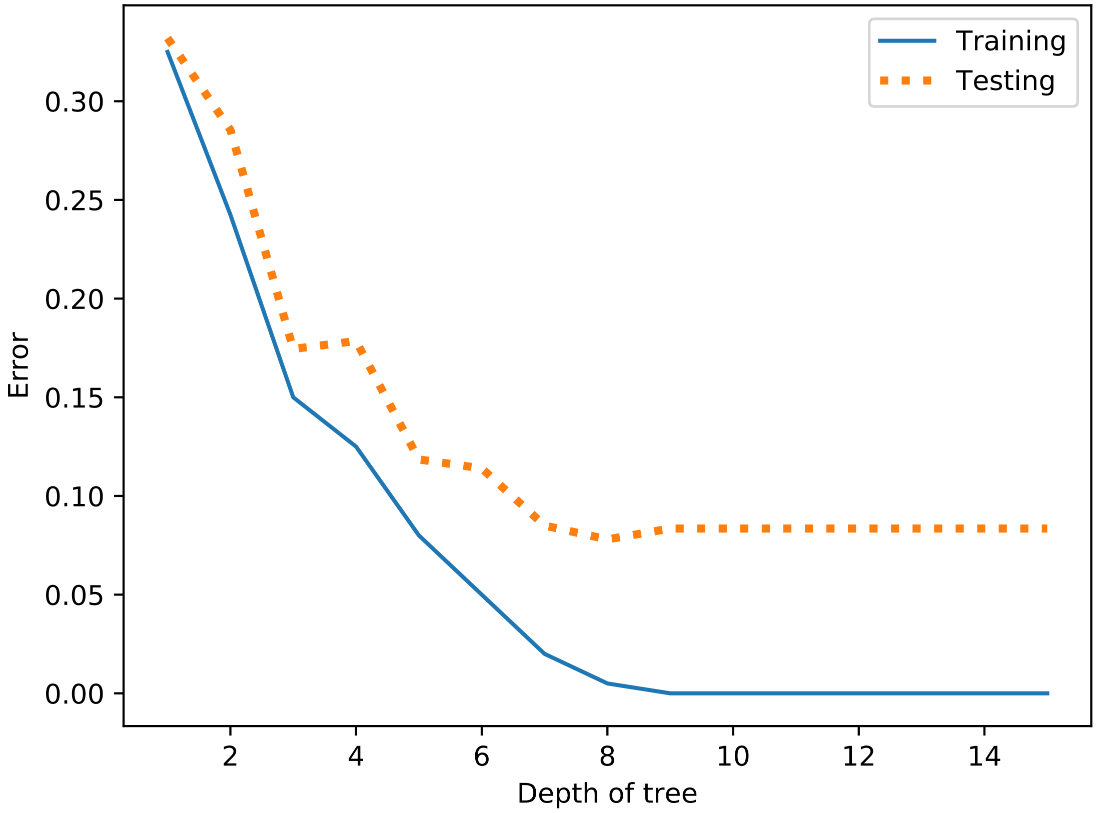
If you run `python main.py -q 1`, it will load the *citiesSmall.pkl* data set from Assignment 1. Note that this file contains not only training data, but also test data, `X_test` and `y_test`. After training a depth-2 decision tree with the information gain splitting rule, it will evaluate the performance of the classifier on the test data. With a depth-2 decision tree, the training and test error are fairly close, so the model hasn't overfit much.

1.1 Training and Testing Error Curves

Rubric: {reasoning:2}

Make a plot that contains the training error and testing error as you vary the depth from 1 through 15. How do each of these errors change with the decision tree depth?

Note: it's OK to reuse code from Assignment 1.

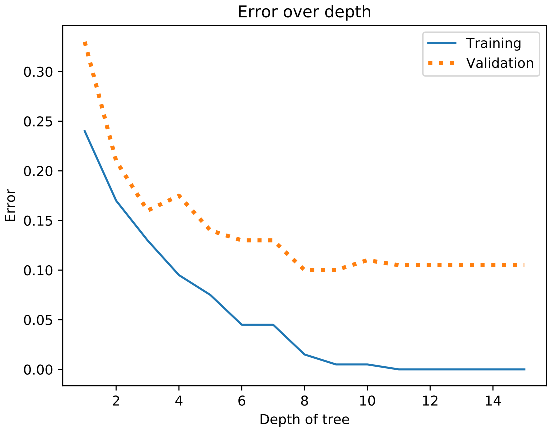


The training error starts to reach 0 (Overfitting) around depth 8. But the testing error doesn't pass the 0.078% error.

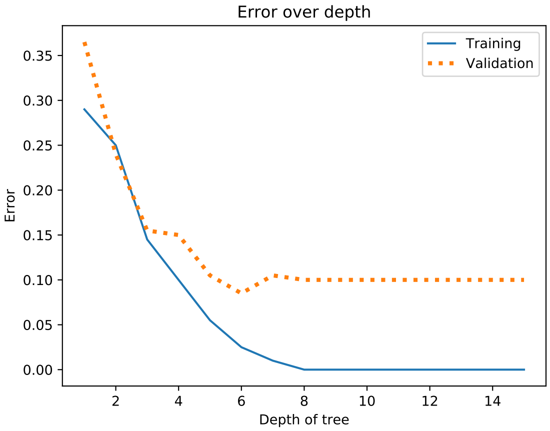
1.2 Validation Set

Rubric: {reasoning:3}

Suppose that we didn't have an explicit test set available. In this case, we might instead use a *validation* set. Split the training set into two equal-sized parts: use the first $n/2$ examples as a training set and the second $n/2$ examples as a validation set (we're assuming that the examples are already in a random order). What depth of decision tree would we pick to minimize the validation set error? Does the answer change if you switch the training and validation set? How could use more of our data to estimate the depth more reliably?



1*st* half as training set and 2*nd* as validation



2*nd* half as training set and 1*st* as validation

By looking at the left plot, we can minimize the validation error by choosing either 8 or 9 as depth of the tree. But if we swap the training and validation sets we minimize the validation error by choosing a depth of 6.

To estimate the depth more reliable we can do cross-validation with the entire training set.

2 Naive Bayes

In this section we'll implement naive Bayes, a very fast classification method that is often surprisingly accurate for text data with simple representations like bag of words.

2.1 Naive Bayes by Hand

Consider the dataset below, which has 10 training examples and 3 features:

$$X = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} \text{spam} \\ \text{spam} \\ \text{spam} \\ \text{spam} \\ \text{spam} \\ \text{spam} \\ \text{not spam} \\ \text{not spam} \\ \text{not spam} \\ \text{not spam} \end{bmatrix}.$$

The feature in the first column is <your name> (whether the e-mail contained your name), in the second column is "pharmaceutical" (whether the e-mail contained this word), and the third column is "PayPal" (whether the e-mail contained this word). Suppose you believe that a naive Bayes model would be appropriate for this dataset, and you want to classify the following test example:

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

2.1.1 Prior probabilities

Rubric: {reasoning:1} Compute the estimates of the class prior probabilities (you don't need to show any work):

- $p(\text{spam})$. $\frac{6}{10} = \boxed{0.6}$
- $p(\text{not spam})$. $\frac{4}{10} = \boxed{0.4}$

2.1.2 Conditional probabilities

Rubric: {reasoning:1}

Compute the estimates of the 6 conditional probabilities required by naive Bayes for this example (you don't need to show any work):

- $p(<\text{your name}> = 1 \mid \text{spam})$. $\frac{1}{6} \approx \boxed{0.16667}$
- $p(\text{pharmaceutical} = 1 \mid \text{spam})$. $\frac{5}{6} \approx \boxed{0.83333}$
- $p(\text{PayPal} = 0 \mid \text{spam})$. $\frac{2}{6} \approx \boxed{0.33333}$
- $p(<\text{your name}> = 1 \mid \text{not spam})$. $\frac{4}{4} = \boxed{1}$
- $p(\text{pharmaceutical} = 1 \mid \text{not spam})$. $\frac{1}{4} = \boxed{0.25}$
- $p(\text{PayPal} = 0 \mid \text{not spam})$. $\frac{3}{4} = \boxed{0.75}$

2.1.3 Prediction

Rubric: {reasoning:1}

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.)

$$p(\text{spam} \mid \hat{x}) = p(<\text{your name}> = 1,\ \text{pharmaceutical} = 1,\ \text{PayPal} = 0 \mid \hat{y} = \text{spam})\ p(\hat{y} = \text{spam})$$

$$=$$

$$p(<\text{your name}> = 1\ |\hat{y} = \text{spam})\ p(\text{pharmaceutical} = 1\ |\hat{y} = \text{spam})\ p(\text{PayPal} = 0\ |\hat{y} = \text{spam})\ p(\hat{y} = \text{spam})$$

$$=$$

$$(\tfrac{1}{6})(\tfrac{5}{6})(\tfrac{2}{6})(0.6) = \tfrac{60}{2160} = \tfrac{1}{36}$$

$$\boxed{p(\text{spam} \mid \hat{x}) = \tfrac{1}{36} \approx 0.02778}$$

$$p(\text{not spam} \mid \hat{x}) = p(<\text{your name}> = 1,\ \text{pharmaceutical} = 1,\ \text{PayPal} = 0 \mid \hat{y} = \text{not spam})\ p(\hat{y} = \text{not spam})$$

$$=$$

$$p(<\text{your name}> = 1\ |\hat{y} = \text{not spam})\ p(\text{pharmaceutical} = 1\ |\hat{y} = \text{not spam})\ p(\text{PayPal} = 0\ |\hat{y} = \text{not spam})\ p(\hat{y} = \text{not spam})$$

$$=$$

$$(1)(0.25)(0.75)(0.6) = 0.1125$$

$$\boxed{p(\text{not spam} \mid \hat{x}) = 0.1125}$$

$p(\text{not spam} \mid \hat{x}) > p(\text{spam} \mid \hat{x})$, therefore the most likely label for the test example $\hat{x}$ is ”not spam”

2.1.4 Laplace smoothing

Rubric: {reasoning:2}

One way to think of Laplace smoothing is that you're augmenting the training set with extra counts. Consider the estimates of the conditional probabilities in this dataset when we use Laplace smoothing (with $\beta = 1$). Give a set of extra training examples that we could add to the original training set that would make the basic estimates give us the estimates with Laplace smoothing (in other words give a set of extra training examples that, if they were included in the training set and we didn't use Laplace smoothing, would give the same estimates of the conditional probabilities as using the original dataset with Laplace smoothing). Present your answer in a reasonably easy-to-read format, for example the same format as the data set at the start of this question.

$$X_{laplace} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad y_{laplace} = \begin{bmatrix} \text{spam} \\ \text{spam} \\ \text{not spam} \\ \text{not spam} \end{bmatrix}.$$

2.2 Bag of Words

Rubric: {reasoning:3}

If you run `python main.py -q 2.2`, it will load the following dataset:

1. X: A binary matrix. Each row corresponds to a newsgroup post, and each column corresponds to whether a particular word was used in the post. A value of 1 means that the word occured in the post.
2. *wordlist*: The set of words that correspond to each column.
3. y: A vector with values 0 through 3, with the value corresponding to the newsgroup that the post came from.
4. *groupnames*: The names of the four newsgroups.
5. *Xvalidate* and *yvalidate*: the word lists and newsgroup labels for additional newsgroup posts.

Answer the following:

1. Which word corresponds to column 51 of X? (This is column 50 in Python.)
   The word is $\boxed{\text{lunar}}$
2. Which words are present in training example 501?
   The example 501 has the words: $\boxed{\text{'car' 'fact' 'gun' 'video'}}$
3. Which newsgroup name does training example 501 come from?
   The groupname is: $\boxed{\text{talk.*}}$

2.3 Naive Bayes Implementation

Rubric: {code:5}

If you run `python main.py -q 2.3` it will load the newsgroups dataset, fit a basic naive Bayes model and report the validation error.

The `predict()` function of the naive Bayes classifier is already implemented. However, in `fit()` the calculation of the variable `p_xy` is incorrect (right now, it just sets all values to 1/2). Modify this function so that `p_xy` correctly computes the conditional probabilities of these values based on the frequencies in the data set. Submit your code and the validation error that you obtain. Also, compare your validation error to what you obtain with scikit-learn's implementation, `BernoulliNB`.

The Naive Bayes classifier implemented has a validation error of 0.188, the Naive Bayes classifier from scikit-learn (`BernoulliNB`) has a validation error of 0.187.

See the following code for the implementation of `Naive Bayes fit()`.

```
def fit(self, X, y):
    N, D = X.shape

    # Compute the number of class labels
    C = self.num_classes

    # Compute the probability of each class i.e p(y==c)
    counts = np.bincount(y)
    p_y = counts / N

    # Compute the conditional probabilities i.e.
    # p(x(i,j)=1 | y(i)==c) as p_xy

    p_xy = np.zeros((D, C))
    for i in range(D):
        for j in range(C):
            xi_given_class_j = X[y==j][:,i]
            freq = xi_given_class_j.sum()
            p_xy[i][j] = freq/counts[j]

    self.p_y = p_y
    self.p_xy = p_xy
```

2.4 Runtime of Naive Bayes for Discrete Data

Rubric: {reasoning:3}

For a given training example i, the predict function in the provided code computes the quantity

$$p(y_i \mid x_i) \propto p(y_i) \prod_{j=1}^{d} p(x_{ij} \mid y_i),$$

for each class y_i (and where the proportionality constant is not relevant). For many problems, a lot of the $p(x_{ij} \mid y_i)$ values may be very small. This can cause the above product to underflow. The standard fix for this is to compute the logarithm of this quantity and use that $\log(ab) = \log(a) + \log(b)$,

$$\log p(y_i \mid x_i) = \log p(y_i) + \sum_{j=1}^{d} \log p(x_{ij} \mid y_i) + \text{(irrelevant propportionality constant)}\,.$$

This turns the multiplications into additions and thus typically would not underflow.

Assume you have the following setup:

- The training set has n objects each with d features.
- The test set has t objects with d features.
- Each feature can have up to c discrete values (you can assume $c \leq n$).
- There are k class labels (you can assume $k \leq n$)

You can implement the training phase of a naive Bayes classifier in this setup in $O(nd)$, since you only need to do a constant amount of work for each $X(i, j)$ value. (You do not have to actually implement it in this way for the previous question, but you should think about how this could be done.) What is the cost of classifying t test examples with the model and this way of computing the predictions?

The cost of classifying t test examples is $O(tdck)$, because for each test sample we have to get $p(x_{ij}$ which will be done in d time and then c to calculate the feature with all of its possible values, finally we will repeat this process k times for the number of classes. So the whole cost of the prediction will be $O(tdck)$.

3 K-Nearest Neighbours

Rubric: {code:3, reasoning:4}

In the *citiesSmall* dataset, nearby points tend to receive the same class label because they are part of the same U.S. state. For this problem, perhaps a k-nearest neighbours classifier might be a better choice than a decision tree. The file *knn.py* has implemented the training function for a k-nearest neighbour classifier (which is to just memorize the data).

Fill in the `predict` function in `knn.py` so that the model file implements the k-nearest neighbour prediction rule. You should Euclidean distance, and may numpy's `sort` and/or `argsort` functions useful. You can also use `utils.euclidean_dist_squared`, which computes the squared Euclidean distances between all pairs of points in two matrices.

1. Write the `predict` function.
   See the following code for the implementation of `K-Nearest Neighbours predict()`.

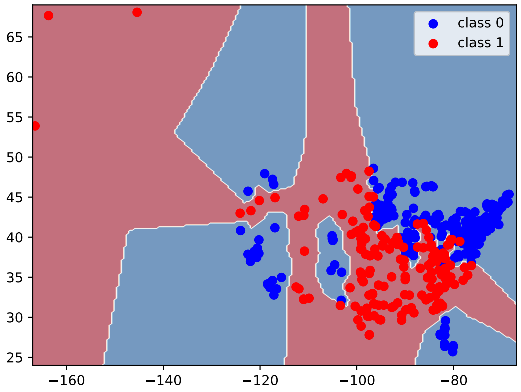
```
def predict(self, Xtest):
    D = utils.euclidean_dist_squared(self.X, Xtest).transpose()
    D_ = np.argsort(D)
    DKNN_indices = D_[:,range(0,self.k)]

    t, k = DKNN_indices.shape
    y_ = np.zeros(t)
    for i in range(t):
        y_[i] = utils.mode(np.take(self.y, DKNN_indices[i]))

    return y_
```

2. Report the training and test error obtained on the *citiesSmall* dataset for $k = 1$, $k = 3$, and $k = 10$. How do these numbers compare to what you got with the decision tree?

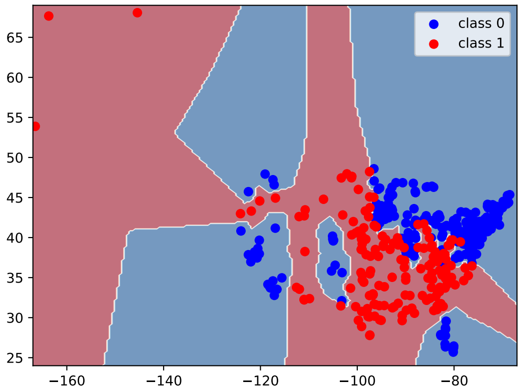
	$k = 1$	$k = 3$	$k = 10$
Training error (ours)	0.000	0.028	0.072
Test error (ours)	0.065	0.066	0.097
Training error (scikit)	0.000	0.028	0.072
Test error (scikit)	0.065	0.066	0.097

The raining and testing errors using KNN are much smaller than with the decision tree

3. Hand in the plot generated by `utils.plotClassifier` on the *citiesSmall* dataset for $k = 1$, using both your implementation of KNN and the KNeighborsClassifier from scikit-learn.



My implementation of KNN



Scikit-Learn KNN

4. Why is the training error 0 for $k = 1$?
   Because the `predict()` function is giving the exact values of the closest neighbour and not averaging between several neighbors.

5. If you didn't have an explicit test set, how would you choose k?
   Using cross-validation

4 Random Forests

4.1 Implementation

Rubric: {code:4,reasoning:3}

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 `DecisionStumpInfoGain` in that 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 4` 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?
   Because the RandomStump doesn't 'considers all d features it just considers $\lfloor \sqrt{d} \rfloor$ randomly-chosen features. Therefore it is pretty odd that the RandomTree overfits.

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.
   See the following code for the implementation of `RandomForest`.

```
class RandomForest():

    def __init__(self, max_depth, num_trees):
        self.max_depth = max_depth
        self.num_trees = num_trees

    def fit(self, X, y):
        N, D = X.shape
        trees = [None] * self.num_trees
        for i in range(self.num_trees):
            model = RandomTree(max_depth=self.max_depth)
            model.fit(X, y)
            trees[i] = model
        self.trees = trees

    def predict(self, Xtest):
        T, D = Xtest.shape
```

[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)`.

```
        predictions = np.zeros([T, self.num_trees])
        for j in range(self.num_trees):
            predictions[:, j] = self.trees[j].predict(Xtest)
        predictions_mode = np.zeros(T)
        for i in range(T):
            predictions_mode[i] = utils.mode(predictions[i,:])
        return predictions_mode
```

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.

	DecisionTree	RandomTree	RandomForest
Training error	0.000	0.133	0.000
Test error	0.367	0.451	0.193

By seeing that the `RandomTree` doesn't have a training error of 0 I expected the `RandomForest` to don't have a training error of 0, but it turned out that our `RandomForest` have a training error of 0.

This may be because of the hyperparameter `num_trees`. The different `RandomTrees` generated may complement with each other and make the whole classifier considers all the features of the training data so the model will eventually overfit.

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.

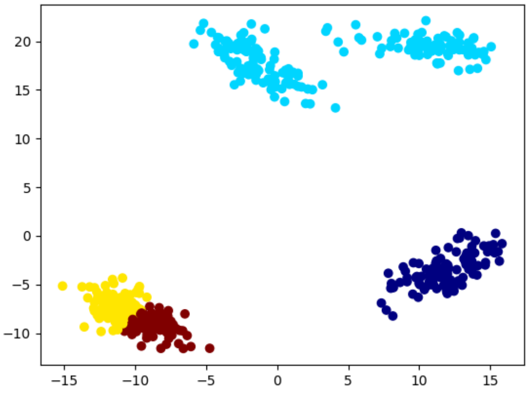
After running a few times our `RandomForest` and the `RandomForestClassifier` from scikit-learn with the same number of trees in the forests (50). We noticed that the scikit-learn classifier does the work much faster than the `RandomForest` we implemented. And regarding to the accuracy, we obtained the following data.

No. run	RandomForest test error	RandomForestClassifier test error
1	0.186	0.178
2	0.216	0.125
3	0.208	0.152
4	0.178	0.136
5	0.152	0.186
5	0.208	0.163

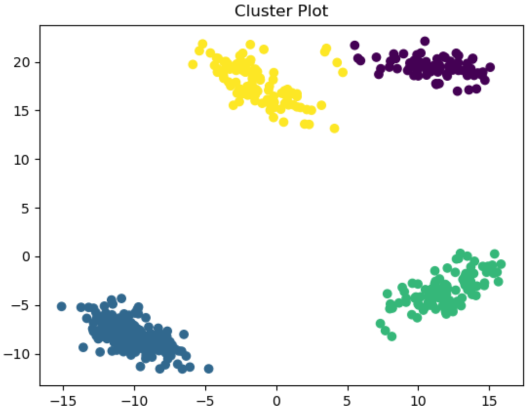
The accuracy difference between both classifier is not remarkably big, it is around 0.09 and 0.008. In most of the cases the `RandomForestClassifier` is the one with the best accuracy.

5 Clustering

If you run `python main.py -q 5`, it will load a dataset with two features and a very obvious clustering structure. It will then apply the k-means algorithm with a random initialization. The result of applying the algorithm will thus depend on the randomization, but a typical run might look like this:



(Note that the colours are arbitrary – this is the label switching issue.) But the 'correct' clustering (that was used to make the data) is this:



5.1 Selecting among k-means Initializations

Rubric: {reasoning:5}

If you run the demo several times, it will find different clusterings. To select among clusterings for a *fixed* value of k, one strategy is to minimize the sum of squared distances between examples x_i and their means w_{y_i},

$$f(w_1, w_2, \dots, w_k, y_1, y_2, \dots, y_n) = \sum_{i=1}^{n} \|x_i - w_{y_i}\|_2^2 = \sum_{i=1}^{n}\sum_{j=1}^{d}(x_{ij} - w_{y_i j})^2.$$

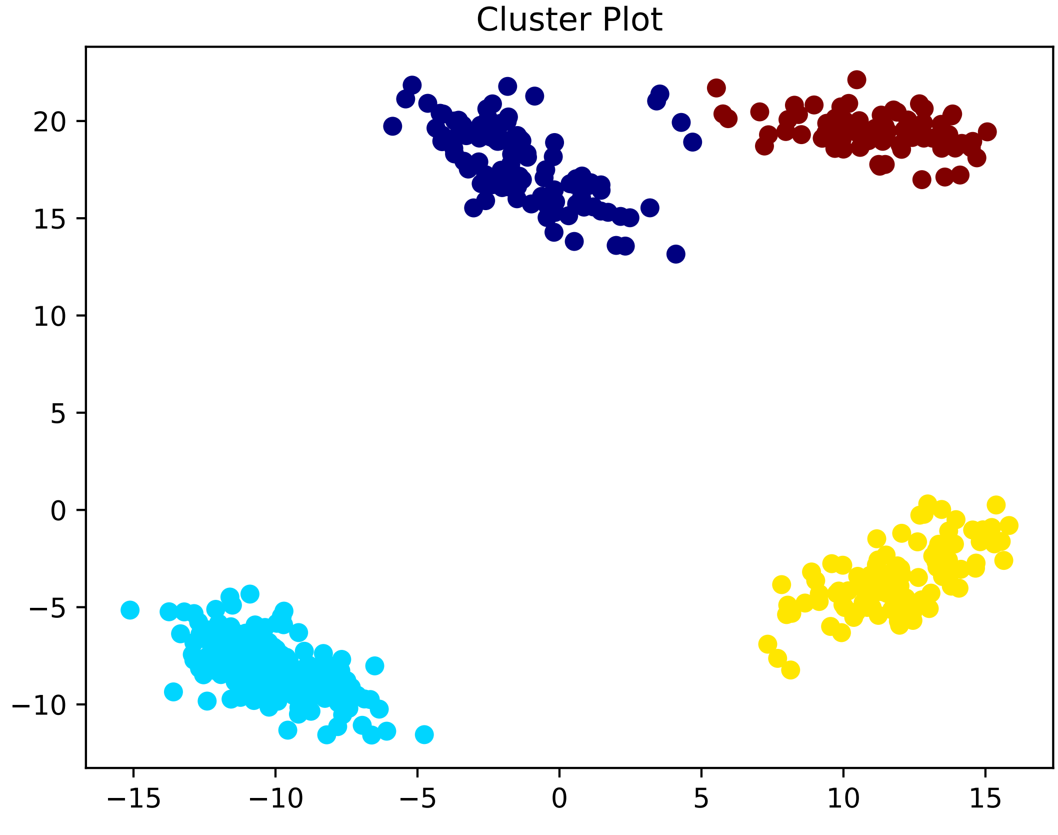
where y_i is the index of the closest mean to x_i. This is a natural criterion because the steps of k-means alternately optimize this objective function in terms of the w_c and the y_i values.

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.
   See the following code for the implementation of K-`Means error()`.

```
def error(self, X):
    n, d = X.shape
    means = self.means
    y_pred = self.predict(X)
    pred_means = means[y_pred]
    dist2 = euclidean_dist_squared(X, pred_means)
    dist2[np.isnan(dist2)] = np.inf
    return np.min(dist2, axis=1).sum()
```

2. What trend do you observe if you print the value of this error after each iteration of the k-means algorithm?
   The error decreases after each iteration, as well as the number of changes in the cluster assignment.

3. Using the code from question 5 in `main.py` (modify if needed), output the clustering obtained by running k-means 50 times (with $k = 4$) and taking the one with the lowest error. Submit your plot.



4. Looking at the hyperparameters of scikit-learn's `KMeans`, explain the first four (`n_clusters`, `init`, `n_init`, `max_iter`) very briefly.

1. n_clusters Number of clusters (classes) to group the data

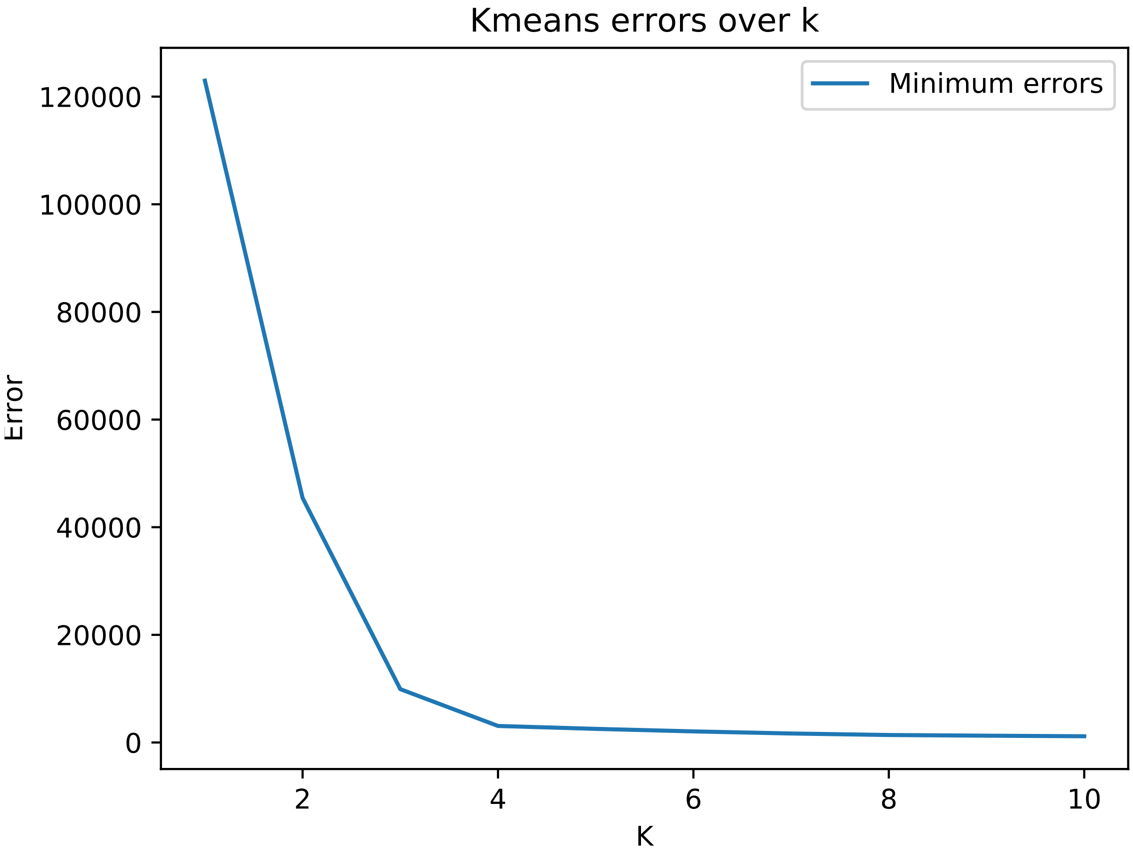
2. init How the means are going to be initialized
3. n_init Number of time KMeans will be running with different seeds
4. max_iter Maximum number of iterations of KMeans, the maximum time that the means will be updated.

5.2 Selecting k in k-means

Rubric: {reasoning:5}

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

1. Explain why we should not choose k by taking the value that minimizes the `error` function.
   Because choosing the k that has the lowest `error` doesn't mean that is the "correct" number of classes. This approach may do the KMeans to group each training point into its own cluster, becoming super specific when predicting labels.
2. Explain why even evaluating the `error` function on test data still wouldn't be a suitable approach to choosing k.
   Even if we use test data to minimize the error and get k. We will just overfit the KMeans to have the nearest means to the testing data.
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.

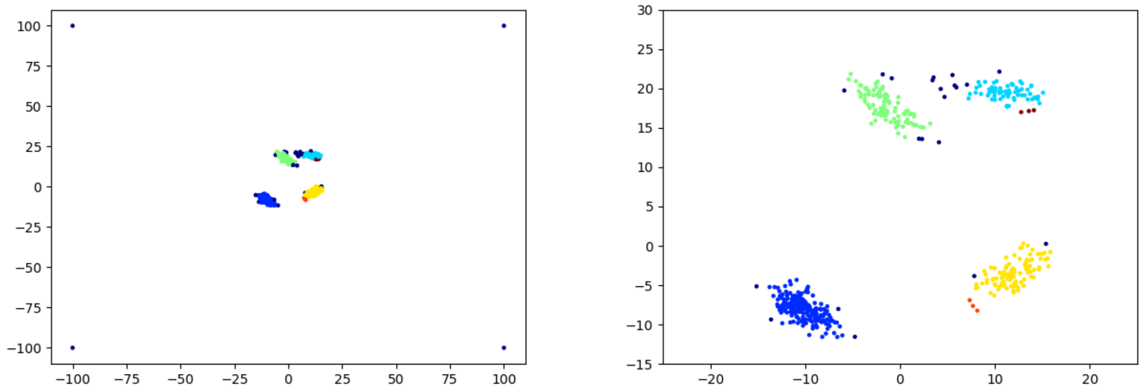


4. The *elbow method* for choosing k consists of looking at the above plot and visually trying to choose the k that makes the sharpest "elbow" (the biggest change in slope). What values of k might be reasonable according to this method? Note: there is not a single correct answer here; it is somewhat open to interpretation and there is a range of reasonable answers.
   The values of k that make the sharpest "elbow" on the error plot is are $k = 3$ or $k = 4$.

5.3 Density-Based Clustering

Rubric: {reasoning:2}

If you run `python main.py -q 5.3`, it will apply the basic density-based clustering algorithm to the dataset from the previous part, but with some outliers added. 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, `minPts` = 3
2. 3 clusters (merging the top two, which also seems like a reasonable interpretaition).
   `eps` = 4, `minPts` = 3
3. 2 clusters.
   `eps` = 13, `minPts` = 3
4. 1 cluster (consisting of the non-outlier points).
   `eps` = 16, `minPts` = 3

6 Very-Short Answer Questions

Rubric: {reasoning:13}

Write a short one or two sentence answer to each of the questions below. Make sure your answer is clear and concise.

1. What is an advantage of using a boxplot to visualize data rather than just computing its mean and variance?
   With a boxplot you can easily identify outliers and see how the data is spread.
2. What is a reason that the the data may not be IID in the email spam filtering example from lecture?
   The bag of words of that email spam filtering example had the words {Hi,CPSC,340,Vicodin,Offer}. The words CPSC and 340 may not be independent.
3. What is the difference between a validation set and a test set?
   The validation set can be used during the training phase, but the test set can't be used at all during the training phase
4. Why can't we (typically) use the training error to select a hyper-parameter?
   Because the classifier will tend to overfit. Therefore if we choose a hyper-parameter using the training

error, our hyper-parameter will help us to overfit quicker and not to be more accurate on the test data.

5. What is the effect of n on the optimization bias (assuming we use a parametric model).
   As n increases, the optimization bias shrinks.

6. What is an advantage and a disadvantage of using a large k value in k-fold cross-validation.
   The advantage is that the cross-validation gets more accurate by using large k, but it gets much more expensive to compute.

7. Why can we ignore $p(x_i)$ when we use naive Bayes?
   Because $p(x_i)$ does not depends on the class that you are classifying, so $p(x_i)$ becomes constant therefore we can ignore that probability.

8. For each of the three values below in a naive Bayes model, say whether it's a parameter or a hyper-parameter:

   (a) Our estimate of $p(y_i)$ for some y_i. parameter

   (b) Our estimate of $p(x_{ij} \mid y_i)$ for some x_{ij} and y_i. parameter

   (c) The value β in Laplace smoothing. hyperparameter

9. What is the effect of k in KNN on the two parts (training error and approximation error) of the fundamental trade-off. Hint: think about the extreme values.
   As k grows the training error gets lower and the approximation error increases.

10. Suppose we want to classify whether segments of raw audio represent words or not. What is an easy way to make our classifier invariant to small translations of the raw audio?
    By adding transformed data to the training set

11. Both supervised learning and clustering models take in an input x_i and produce a label y_i. What is the key difference?
    In supervised learning the classes are known by the model. But with clustering models, the labels are unknown.

12. Suppose you chose k in k-means clustering (using the squared distances to examples) from a validation set instead of a training set. Would this work better than using the training set (which just chooses the largest value of k)?
    It will work better on the validation set but not necessarily on the training set.

13. In k-means clustering the clusters are guaranteed to be convex regions. Are the areas that are given the same label by KNN also convex?
    No, the areas given by KNN not necessarily be convex. Because the data points are matched by distance, and in kmeans they are matched if they are inside an circular area with epsilon value. So the data points in a kluster with KNN can't always create a convex area.