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

We are providing solutions because supervised learning is easier than unsupervised learning, and so we think having solutions available can help you learn. However, the solution file is meant for you alone and we do not give permission to share these solution files with anyone. Both distributing solution files to other people or using solution files provided to you by other people are considered academic misconduct. Please see UBC's policy on this topic if you are not familiar with it:
`http://www.calendar.ubc.ca/vancouver/index.cfm?tree=3,54,111,959`
`http://www.calendar.ubc.ca/vancouver/index.cfm?tree=3,54,111,960`

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.

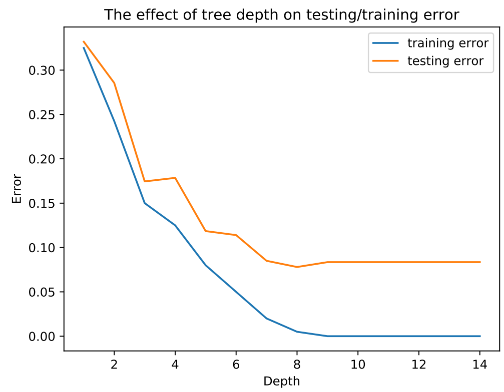
1.1 Training and Testing Error Curves

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.

Answer: The plot should look like this:



The training error goes down monotonically to 0 (quickly at first and then slowly). The test error also starts going down but after a depth of 9 it goes up and then stays flat.

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?

Answer: Using sklearn's implementation with `random_state=1` I got the best depth to be 8 using the first half for training, and 6 when using the second half for training. To make the estimate more reliable, you could use cross-validation.

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):

- $p(\text{spam})$.

  Answer: 6/10.

- $p(\text{not spam})$.

  Answer: 4/10.

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):

- $p(\text{<your name>} = 1 \mid \text{spam})$.

  Answer: 1/6.

- $p(\text{pharmaceutical} = 1 \mid \text{spam})$.

  Answer: 5/6.

- $p(\text{PayPal} = 0 \mid \text{spam})$.

  Answer: 2/6.

- $p(\text{<your name>} = 1 \mid \text{not spam})$.

  Answer: 1.

- $p(\text{pharmaceutical} = 1 \mid \text{not spam})$.

  Answer: 1/4.

- $p(\text{PayPal} = 0 \mid \text{not spam})$.

  Answer: 3/4.

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.)

Answer:

$$\begin{aligned}
p(\text{spam} \mid x_1 = 1, x_2 = 1, x_3 = 0) &\propto p(x_1 = 1, x_2 = 1, x_3 = 0 \mid \text{spam})p(\text{spam}) \\
&= p(x_1 = 1 \mid \text{spam})p(x_2 = 1 \mid \text{spam})p(x_3 = 0 \mid \text{spam})p(\text{spam}) \\
&= (1/6)(5/6)(2/6)(6/10) \\
&\approx 0.028
\end{aligned}$$

$$\begin{aligned}
p(\text{not spam} \mid x_1 = 1, x_2 = 1, x_3 = 0) &\propto p(x_1 = 1, x_2 = 1, x_3 = 0 \mid \text{not spam})p(\text{not spam}) \\
&= p(x_1 = 1 \mid \text{not spam})p(x_2 = 1 \mid \text{not spam})p(x_3 = 0 \mid \text{not spam})p(\text{not spam}) \\
&= (1)(1/4)(3/4)(4/10) \\
&\approx 0.075.
\end{aligned}$$

Since $p(\text{not spam} \mid x_1 = 1, x_2 = 1, x_3 = 0)$ is proportional to a bigger number, and the proportionality constants are the same ($p(x_1 = 1, x_2 = 1, x_3 = 0)$), we would predict "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.

Answer: You could add the following examples:

$$X_\beta = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad y_\beta = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

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.)

   Answer: “lunar”

2. Which words are present in training example 501?

   Answer: “car”, “fact”, “gun”, “video”

3. Which newsgroup name does training example 501 come from?

   Answer: “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`.

Answer: The test error is approximately 0.19. It's the same as scikit-learn's once you add Laplace smoothing.

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?

Answer: For each of the t examples, the dominant cost is computing $p(x_{ij}|y_i)$ for all d values of j and all k class labels. You can do this with three "for" loops (as in the naive Bayes `predict` function in the given code): one looping over the examples t, one looping over the features d, and one looping over the class labels k. Since each of the loops does a constant amount of work, the total time is $O(tdk)$. Note that this is much slower than using a depth m decision tree in the common case that $m << dk$. (However, the training and testing phases can be much faster if the examples are sparse, meaning that most values of x_{ij} are zero.)

3 K-Nearest Neighbours

Rubric: {code:3, reasoning:4}

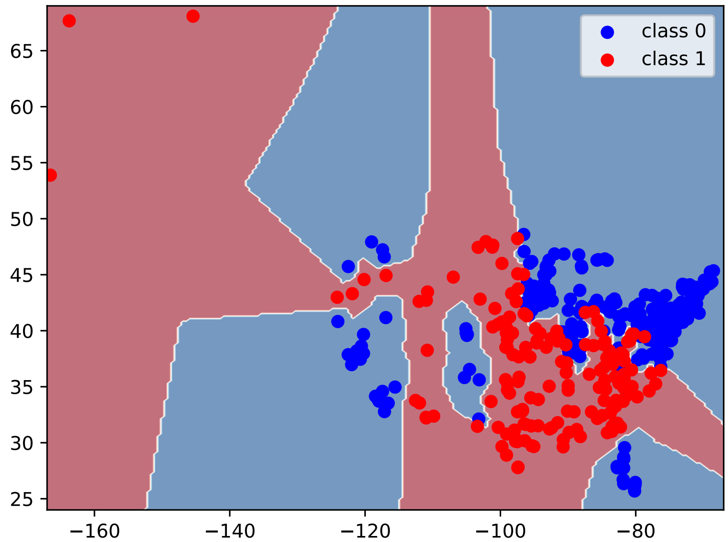
In the *citiesSmall* dataset, nearby points tend to receive the same class label because they are part of the same U.S. state. For this problem, perhaps a k-nearest neighbours classifier might be a better choice than a decision tree. The file *knn.py* has implemented the training function for a k-nearest 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.
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?
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.
4. Why is the training error 0 for $k = 1$?
5. If you didn't have an explicit test set, how would you choose k?

Answer:

1. See knn.py.
2. The training/test errors are:
   - $k = 1$: training is 0, test is 0.065.
   - $k = 3$: training is 0.028, test is 0.066.
   - $k = 10$: training is 0072, test is 0.097.
3. The plot should look like this:



4. The training error is 0 for $k = 1$ because every training example is 1-nearest neighbour of itself, so when you are predicting you just copy the labels from the training data. (Unless you have duplicate training examples, 1-nearest neighbour always obtains a training error of 0: this is why reporting low training errors is meaningless.)
5. The training error strictly goes down as k decreases, so you can't use the training error to choose k. Instead, you should split your data into a training and validation set, or use 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?

   Answer: After a certain depth the tree can perfectly classify the examples it is training on (which is the full training set for the deterministic method and a bootstrap sample for the random tree), so it stops splitting. (Technically, it will also stop at some point because you run out of possible splits, but if you look at the trees they are stopping well before this.)

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.

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.

   Answer: Train error is 0, test error is 0.163. This is better test error than a single tree, as expected.

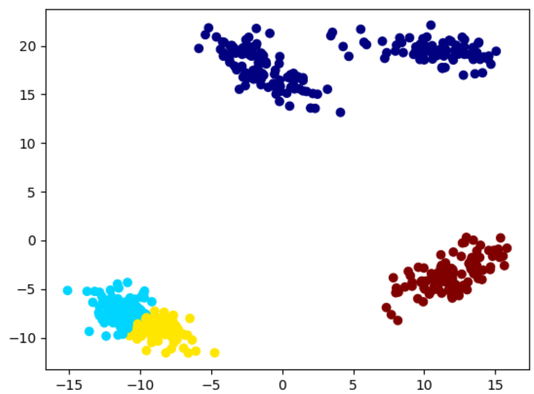
4. Compare your implementation with scikit-learn's RandomForestClassifier for both speed and accuracy, and briefly discuss. You can use all default hyperparameters if you wish, or you can try changing them.

   Answer: With default hyperparameters the accuracy is similar, but sklearn's implementation is much faster. For one thing, we aren't using the $n \log n$ algorithm.

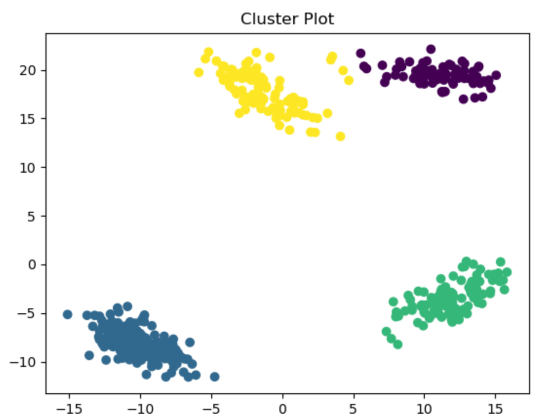
5 Clustering

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:

[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)`.



(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},

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

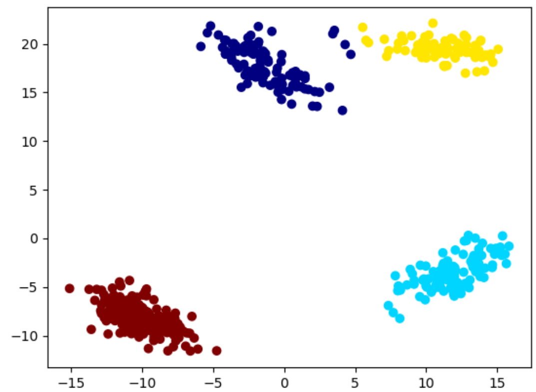
where y_i is the index of the closest mean to x_i. This is a natural criterion because the steps of k-means alternately optimize this objective function in terms of the w_c and the y_i values.

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.
2. What trend do you observe if you print the value of this error after each iteration of the k-means algorithm?

Answer: It decreases monotonically.

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.

   Answer:



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

   Answer: `n_clusters` is the number of clusters, k. `init` is the initialization strategy. `n_init` is the number of initializations to try. `max_iter` is the maximum number of iterations to perform; if you reach this value, you stop before convergence.

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.

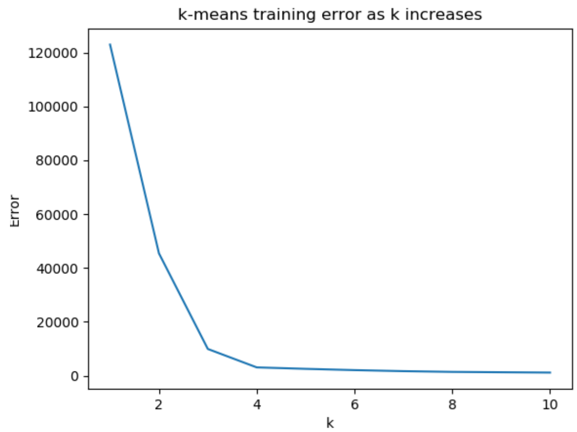
   Answer: The error function always prefers the largest possible k, sort of like choosing the depth of a decision tree with training error.

2. Explain why even evaluating the `error` function on test data still wouldn't be a suitable approach to choosing k.

   Answer: Since you have more clusters as k increases, the closest mean is still likely to be closer on new data with large values of k. So even on test data this objective function would likely prefer the largst value of k.

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.

   Answer:



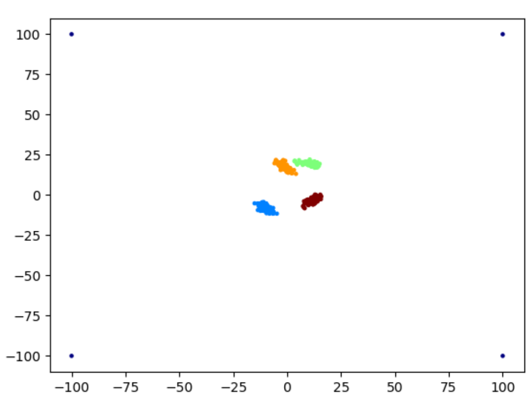
4. The *elbow method* for choosing k consists of looking at the above plot and visually trying to choose the k that makes the sharpest "elbow" (the biggest change in slope). What values of k might be reasonable according to this method? Note: there is not a single correct answer here; it is somewhat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

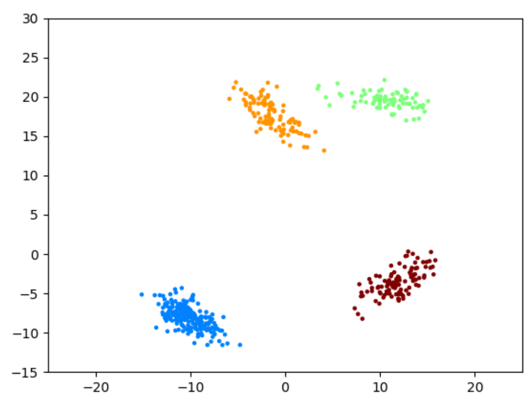
Answer: This will change based on a person's interpretation, but reasonable value might be 3 or 4.

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.

2. 3 clusters (merging the top two, which also seems like a reasonable interpretaition).

3. 2 clusters.

4. 1 cluster (consisting of the non-outlier points).

Answer: There are many possible parameter settings that would work. We found that fixing `minPts` to 3 we could get these clusters by setting the radius to:

- $\sqrt{10}$.
- $\sqrt{20}$.
- $\sqrt{175}$.
- $\sqrt{1000}$.

Note: the reason for the square roots is that we previously formulated the question in terms of the radius squared. Any numbers that satisfy the criteria are acceptable answers.

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?

   Answer: The boxplot gives you more information (e.g., multiple measures of location and spread of the data). Alternately, boxplots aren't sensitive to outliers.

2. What is a reason that the the data may not be IID in the email spam filtering example from lecture?

   Answer: Many answers are possible. For example, the spammers might modify their e-mail over time (in response to getting filtered out).

3. What is the difference between a validation set and a test set?

   Answer: A validation set is part of your training set that is set aside to approximate the test error while training, while the test set is a different dataset that cannot influence training in any way (and that you may not have labels for).

4. Why can't we (typically) use the training error to select a hyper-parameter?

   Answer: Hyper-parameters (typically) control model complexity, and more complex models (typically) have lower training error (so this just leads to picking the most complex model we try rather than one that is likely to have a small test error).

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

   Answer: As n increase the optimization bias decreases.

6. What is an advantage and a disadvantage of using a large k value in k-fold cross-validation.

   Answer: Large k values let you use more data (to fit the model), but are more expensive (since you need to fit k models).

7. Why can we ignore $p(x_i)$ when we use naive Bayes?

   Answer: It's the same for all classes, so doesn't affect our decision.

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.
   (b) Our estimate of $p(x_{ij} \mid y_i)$ for some x_{ij} and y_i.
   (c) The value β in Laplace smoothing.

   Answer: The probabilities are parameters and β is a hyper-parameter.
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.

   Answer: As k grows our training error goes up but our approximation error goes down.
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?

    Answer: Add versions of the training data with small translations.
11. Both supervised learning and clustering models take in an input x_i and produce a label y_i. What is the key difference?

    Answer: In supervised learning we're given specific y_i values during training (so they have a meaning in the real world).
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)?

    Answer: Nope, still chooses the largest value (the distances get smaller as k increases).
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?

    Answer: No (you could have two points with the same label that have other points in between with different labels).