The first assignment is meant to exercise your probability and programming abilities. For both parts, you will need to produce a report which you will submit online. In addition, for the programming portion you will need to submit your code in a separate ZIP file. Both parts are due **Wednesday, Feburary 1 at 11:59pm**. Late submissions will not be accepted. Submissions will be handled through Moodle.

Basic Probability

1. Consider a test which detects if a person has a disease. Let R denote the outcome of the test on a person, D denote whether the person actually has the disease and θ be the likelihood that the test gives the correct result. That is, the probability that it reports that someone has the disease ($R = 1$) when they actually do ($D = 1$), is θ, and the probability that it reports that someone doesn't have the disease when they don't is θ. Formally:

$$p(R{=}1 \,|\, D{=}1) \;=\; p(R{=}0 \,|\, D{=}0) \;=\; \theta$$

Finally, the prior probability of a person having this disease is $p(D) = \alpha$.

a) A patient goes to the doctor, has the test performed and it comes back positive. Derive the posterior probability that the person actually has the disease, and simplify it in terms of θ and α.

b) After the results of the first test come back positive, the doctor runs it a second time. This time it comes back negative. Derive the posterior probability that the person actually has the disease after this second round of testing and simplify in terms of θ and α.

c) Suppose $\theta = .99$ and $\alpha = 0.001$. Suppose 1000 patients get tested, and they're all negative. On average, how many of these patients actually have the disease? I.E., what is the expected number of false negatives?

2. A bag contains four dice: a 4-sided die, two 8-sided dice and a 12-sided die. A person chooses a die from the bag at random and rolls it. The number of faces that the selected die has is denoted by the random variable S and the number rolled by the random variable X.

a) Derive the probability distribution of the number rolled with the selected die. Use your expression for $p(X)$ to determine the probability that the first number rolled is 3. That is, what is $p(X = 3)$?

b) Given that the roll was a 3, what's the posterior probability that the selected die had 12 sides? That is, what is $p(S = 12|X = 3)$?

c) Suppose the person rolls the same die a second time. Denote the random variable of this second roll by Y. If the second roll comes up a 7, how does this change the posterior probability that the selected die has 12 sides? Specifically, what is $p(S = 12|X = 3, Y = 7)$?

d) Suppose the person rolls the same die a third time after telling you that the first two rolls were 3 and 7. Denote the third roll by Z. What is the predictive probability distribution of seeing 2 on this final roll given the knowledge of the first two rolls? I.E., what is $p(Z = 2|X = 3, Y = 7)$?

3. On a game show, a contestant is told the rules as follows:

> There are three boxes, labelled 1, 2, 3. A single prize has been hidden inside one of them. You get to select one box. Initially your chosen box will not be opened. Instead, the gameshow host will open one of the other two boxes, and will do so in such a way as not to reveal the prize. For example, if you first choose box 1, the host will then open one of boxes 2 and 3, and it is guaranteed that the opened box will not reveal the prize. At this point, you will be given a fresh choice of box: you can either stick with your first choice, or you can switch to the other closed box. All the boxes will then be opened and you will receive whatever is inside your final choice of box.

Imagine that the contestant chooses box 1 first; then the gameshow host opens box 3, revealing nothing behind the door, as promised. Should the contestant (a) stick with box 1, or (b) switch to box 2, or (c) does it make no difference? You may assume that initially, the prize is equally likely to be inside any of the 3 boxes.

Hint: Start by defining three hypotheses for which box the prize is inside and a random variable indicating which box the host opens. Then compute the probabilities of the three hypotheses given the observation of this random variable.

4. Given two random variables X and Y, prove that

$$var[X+Y] = var[X] + var[Y] + 2cov[X,Y]$$

where $cov[X,Y]$ is the covariance between X and Y.

Naive Bayes

In this assignment we will consider the problem classification from discrete input features, in particular using the Naive Bayes classifier.

The dataset we will consider contains over 8000 varieties of mushrooms. For each mushroom it has been indicated whether it is edible or not (meaning poisonous or unknown). Further, each mushroom has been classified based on a number of features. For instance, mushrooms are classified by the shape, colour and texture of their caps, their odours, whether they bruise, what type of stalk they have and so on. In total there are 22 of these features, many of which have multiple possible settings.

Your task in this assignment is to build a classifier which will use these features to predict whether a mushroom is edible. As part of this assignment you will learn to load data, explore it by making plots and computing statistics, train classifiers, test their performance and analyze the results. Finally, at the end you will be given an opportunity to extend the basic model we've built in this course for extra credit.

You will submit a report which describes your results and includes the outputs. You should also submit a ZIP file of your source code. Details on how to submit your assignment will be provided separately.

Step 0: Download and load the data.

This data is available from the course website and once unpacked will contain the files MUSHROOMS.CSV and README.TXT. The README file contains a longer description of the dataset and can be ignored unless you're curious. The data file is a text file where each line corresponds to a single entry (or mushroom type) and contains a set of comma separated entries. For instance, the first line is:

```
p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u
```

The first column indicates whether the mushroom is edible ('e') or poisonous ('p'). The remaining 22 columns contain a single character for each field, the meanings of which can be found in the README. To start building a classifier we need to transform this data into a numberical format that is easier to work with.

For each column, construct a mapping from characters to numbers. For instance, the first column contains either the character 'e' or 'p'. We can map 'e' to 0 and 'p' to 1. The second column contains the characters 'x', 'b', 's', 'f', 'k', and 'c' which we can map to the numbers 0 through 5. Do this for all the columns. Note that the specific mapping isn't important, only that you are consistent. Also, for Matlab users it may be more convenient to start the numbers at 1 rather than 0.

Now, use the mapping you established to create a numeric representation of the data in the file. At the end of it, you should have a matrix with 8124 rows and 23 columns. For instance, the first two rows of this matrix might look like:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 1
```

Finally, you should randomly split the dataset into a training set and a validation set. Put 6499 in the training set and the remaining 1625 in the validation set.

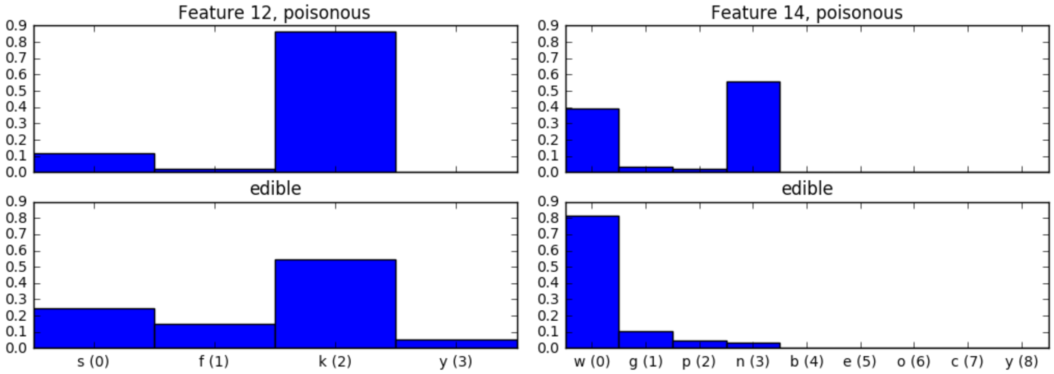
I will provide some simple code to do this for you in Matlab and Python so that you don't get tripped up on this part, however I strongly encourage you to at least try to implement this yourself. In many ML applications, loading the data and getting it into the right format is half the battle.

What to include in your report: Nothing for this section.

Step 1: Analyze the data.

With the data loaded and split up, we're going to do some simple data exploration. It's always helpful and often critical to understand your data. This understanding can inform your understanding of how hard a problem is, what kinds of models may work well, what might be going wrong and many other things. Good visualizations and analysis is the primary way to build this understanding.

First, take the training set and compute what fraction are poisonous and what fraction are edible. Print this out. Next for each feature dimension (i.e., each dimension other than edible or poisonous) we're going to compare the distributions of that feature for edible and poisonous cases. To do this you need to produce two plots. In the first plot, take all training examples which are edible and produce a histogram of the feature values. In the second plot, take all training examples which are poisonous and produce a histogram of the values. Be sure each plot is clearly labelled in terms of what dimension it came from and whether it was the poisonous or edible. For instance, two sets of these plots might look like the following:



Based on this, which features do you think will ultimately be most important and least important for determining edibility? Why?

What to include in your report: Your report should include what fraction of the training data was edible/poisonous and your analysis of which features will be most important and least important for determining edibility. You should also include histograms for two features that you think should be important for determining edibility and histograms for two features that you think won't be important. For the features that you determine to be important, do you think you could use only those features? This means your report for this part should include 8 histograms in total.

Step 2: Fit a Naive Bayes classifier to the data.

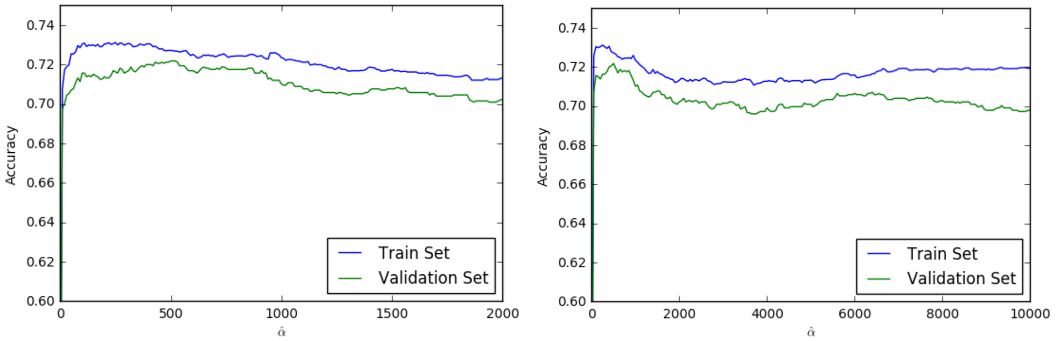
In this section you're going to implement a regularized Naive Bayes classifier. In particular, let E be the random variable indicating edibility and $F_1, \ldots, F_{22}$ be the random variables indicating the 22 features. Then

$$\begin{aligned}
p(E|F_1 = f_1, \ldots, F_{22} = f_{22}) &= \frac{p(E)p(F_1 = f_1, \ldots, F_{22} = f_{22}|E)}{p(F_1 = f_1, \ldots, F_{22} = f_{22})} \\
&= \frac{p(E)\prod_{i=1}^{22} p(F_i = f_i|E)}{\sum_{e\in\{0,1\}} p(F_1 = f_1, \ldots, F_{22} = f_{22}|E = e)p(E = e)} \\
&= \frac{p(E)\prod_{i=1}^{22} p(F_i = f_i|E)}{\sum_{e\in\{0,1\}} \prod_{i=1}^{22} p(F_i = f_i|E = e)p(E = e)}
\end{aligned}$$

is the posterior predictive distribution for edibility given the features. To use this model, we need to fit the prior distribution of mushroom edibility, i.e., $p(E = 1)$, and the likelihood distribution of each feature when the mushrooms are edible, i.e., $p(F_i = f_i|E = 1)$, and when the mushrooms are not edible, i.e., $p(F_i = f_i|E = 0)$. For this you're going to use a Dirichlet-multinomial model with a simplified form of the Dirichlet prior. This model is detailed in Section 3.4 of the textbook. The simplification we will make is that the parameters of the Dirichlet prior, $\alpha = (\alpha_1, \alpha_2, \ldots)$, will be simplified to a single parameter $\alpha_i = \hat{\alpha}$ where $\hat{\alpha}$ is a single scalar parameter.

Your task is to implement the estimation of these distributions using a MAP estimate of the parameters given the training data. (See Equation 3.47 from the textbook.) Then implement the ability to make predictions with this model for new training examples using the equation above. Be sure to do the computations in log space as discussed in class and using the LOG-SUM-EXP trick discussed in Section 3.5.3 to ensure numerical stability. Finally, threshold the predicted probability at 0.5 to make a classification and compute the accuracy (fraction of examples correct) of the fit model on both the training and validation sets.

Once you have implemented the above, fit a variety of models with values of $\hat{\alpha}$ between 0 and 10000. For each fit model, compute and record the accuracy on both the training and validation datasets and plot it. The plots should look something like this:



In addition, take note of the train and validation accuracy with $\hat{\alpha} = 0$, $\hat{\alpha} = 0.1$ and the best value of $\hat{\alpha}$ based on the plots.

What to include in your report: Include your version of the accuracy plot. In addition provide answers to the following questions:

1) Report the accuracy of the classifier with $\hat{\alpha} = 0$ and $\hat{\alpha} = 0.1$. The accuracy with $\hat{\alpha} = 0$ is actually worse than chance performance, meaning if we randomly made a decision for each example it would work better. Increasing $\hat{\alpha}$ to 0.1 dramatically changes this. Why is this?

2) The training set accuracy and validation set accuracy are different. Why?

3) Accuracy increases as $\hat{\alpha}$ increases from 0 until it reaches a peak and then decreases. This peak occurs at different places for the training and validation sets. Which one would you use to set the value of $\hat{\alpha}$.

4) After the peak, accuracy decreases as $\hat{\alpha}$ increases. Explain why this is happening. Considering the estimation of the feature distributions, what happens as $\hat{\alpha} \to \infty$?

Step 3: Inspect the model.

For models like this we can determine which features have the biggest impact on the classifier. To do this we want to look at the value of

$$\log p(F_i = f_i | E = 1) - \log p(F_i = f_i | E = 0)$$

Using the model fit with the best value of $\hat{\alpha}$ as determined previously, compute this quantity for all values of i and f_i. Print them out, sorted by the absolute value. Values of i, f_i for which this quantity is large in absolute value corresponds to features which are highly discriminative. If the value is large and positive, that means if $F_i = f_i$ then it is more likely that the mushroom is edible. If the value is large and negative, it is more likely that the mushroom is poisonous.

What to include in your report: Include the sorted list with the corresponding values of i and f_i. What features and feature values are most indicative of edible mushrooms? What features and feature values are most indicative of poisonous mushrooms? For each feature you discuss here, include the histograms that you generated in Step 1 above and talk about how the histogram relates to the discriminativeness of the feature.

Bonus

One of the limitations of Naive Bayes is that it doesn't capture dependent features. Unfortunately, it's impractical to do a full Bayes model even for this relatively low dimensional case. It would have over eight million parameters and require many times that amount of data! However, we can try to jointly model selected subsets of features. For instance, if we wanted to capture the dependence between F_1 and F_2 we could use

$$p(F_1 = f_1, \ldots, F_{22} = f_{22} | E) = p(F_1 = f_1, F_2 = f_2 | E) \prod_{i=3}^{22} p(F_i = f_i | E)$$

so we only need to estimate the much more tractable joint distributions $p(F_1 = f_1, F_2 = f_2|E = 0)$ and $p(F_1 = f_1, F_2 = f_2|E = 1)$.

For this bonus, compute the mutual information between all pairs of features. That is, compute

$$I(F_i, F_j) = \sum_e \sum_{f_i} \sum_{f_j} p(F_i = f_i, F_j = f_j|E = e) \log \frac{p(F_i = f_i, F_j = f_j|E = e)}{p(F_i = f_i|E = e)p(F_j = f_j|E = e)}$$

for all values of i and j. To do this you will need to estimate all the pairwise joint distributions. Select the pair of features which has maximal mutual information and include the joint distribution instead. Look at the README file and see which features they are. Does it make sense that these two features may be dependent?

Use the new model in a classifier and compare its performance to the original Naive Bayes classifier. Does it perform better, worse or about the same? Can you give an explanation as to why?