CPSC 340 Assignment 1 (due Friday September 29 ATE)

The assignment instructions are the same as Assignment 0, except you have the option to work in a group of 2. If you work in a group, please only hand in *one* assignment. It is recommended that you work in groups as the assignment is quite long, but please only submit one assignment for the group and make sure that everyone's name/ID is on the front page.

1 Summary Statistics and Data Visualization

Download and expand the file *a1.zip*, which contains estimates of the influenza-like illness percentage over 52 weeks on 2005-06 by Google Flu Trends in a comma-separated values (CSV) file. You can open this with Excel or other spreadsheet programs; the first row gives the abbreviation of the region names for each column, and each row gives the estimate for a week. After you change to the a0 directory, you can load this data in Julia using:

```
dataTable = readcsv("fluTrends.csv")
```

This creates an two-dimensional array of type "Any" populated with all the information in the CSV file.

1.1 Summary Statistics

Report the following statistics: the minimum, maximum, mean, median, and mode of all values across the dataset. In light of thea above, is the mode a reliable estimate of the most "common" value? Describe another way we could give a meaningful "mode" measurement for this (continuous) data.

Hint: Since the first row of the CSV file is just the names of the columns, we can create a matrix X containing the data stored as real numbers using:

```
X = real(dataTable[2:end,:])
```

You can make Julia dispaly the matrix X using

```
@show X
```

The *show* macro can be used to display the result of any expression, like showing the tenth row of X:

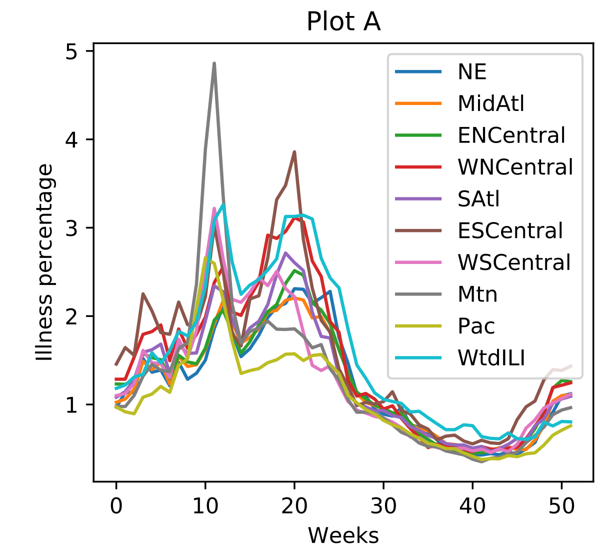
```
@show X[10,:]
```

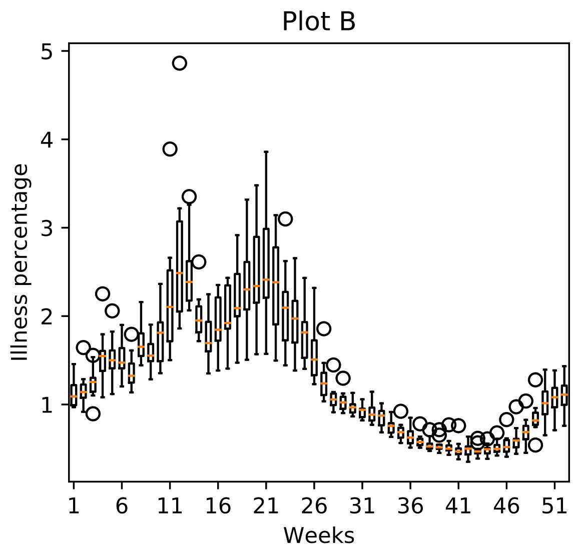
Note that this can be run inside functions, so it's helpful for debugging.

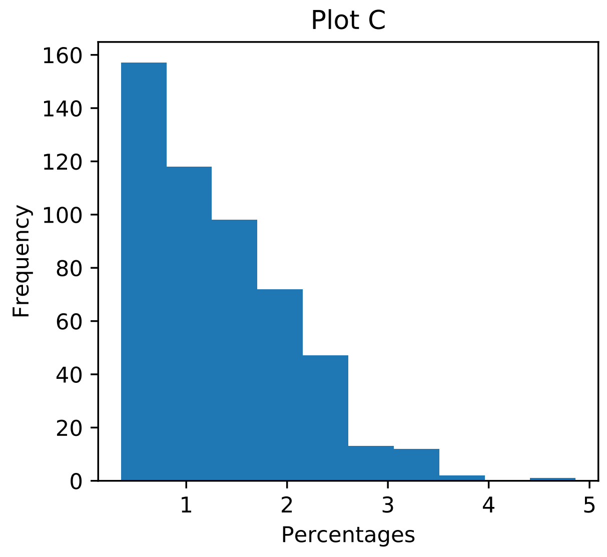
Also, Julia has no mode command so I've included a mode function in 'misc.jl'.

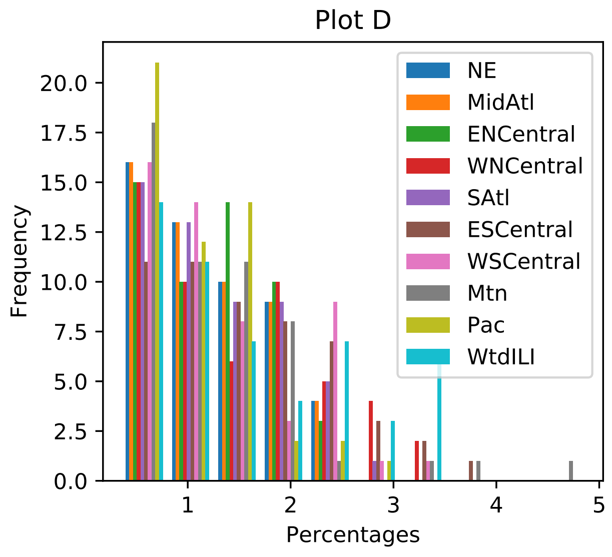
1.2 Data Visualization

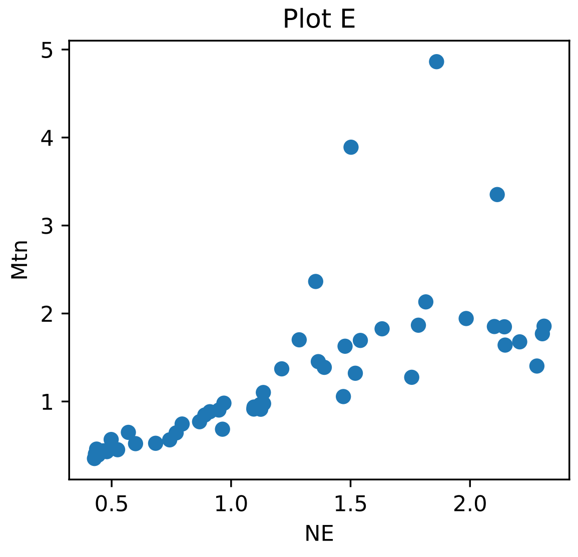
Consider the figure on the next page.

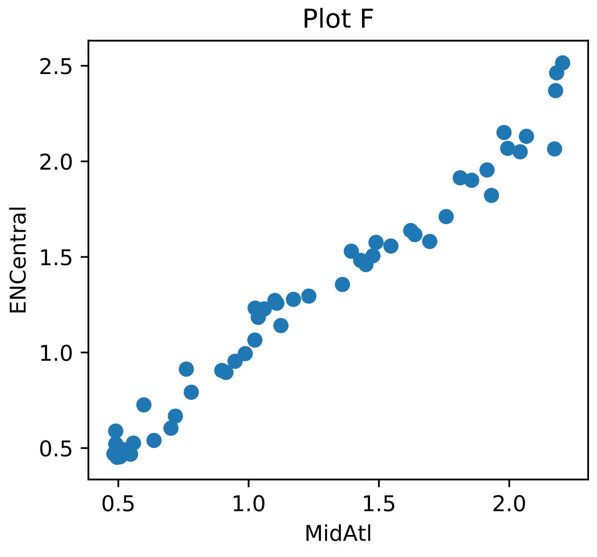

Plot A
Illness percentage
Weeks
NE
MidAtl
ENCentral
WNCentral
SAtl
ESCentral
WSCentral
Mtn
Pac
WtdILI


Plot B
Illness percentage
Weeks


Plot C
Frequency
Percentages


Plot D
Frequency
Percentages
NE
MidAtl
ENCentral
WNCentral
SAtl
ESCentral
WSCentral
Mtn
Pac
WtdILI


Plot E
Mtn
NE


Plot F
ENCentral
MidAtl

The figure contains the following plots, in a shuffled order:

1. A histogram showing the distribution of all values in the matrix X.
2. A boxplot grouping data by weeks, showing the distribution across regions for each week.
3. A scatterplot between the two regions with highest correlation.
4. A single histogram showing the distribution of *each* column in X.
5. A scatterplot between the two regions with lowest correlation.
6. A plot containing the weeks on the x-axis and the percentages for each region on the y-axis.

Match the plots (labeled A-F) with the descriptions above (labeled 1-6), with an extremely brief (a few words is fine) explanation for each decision.

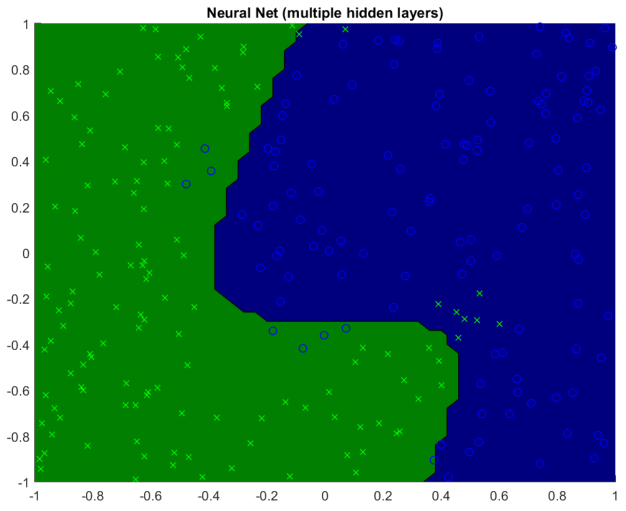
Hint: you can generate similar plots by adding the PyPlot package. To add this package use:

```
Pkg.add("PyPlot") # Do this once per computer
using PyPlot # Do this once per session
plot(1:52,X[:,1]) # Plot the first row
```

To generate similar-looking plots you can use the functions 'plot', 'boxplot', 'plt[:hist]', and 'scatter'.

1.3 Decision Surfaces

Consider the figure below, which plots a set of two-dimensional training examples and the decision surface produced by a "neural network" classifier (a model we'll see later in the course).



How many training examples has the neural network mis-classified? (This figure is best viewed in colour.)

2 Decision Trees

If you run the file *example_decisionStump.jl*, it will load a dataset containing longtitude and latitude data for 400 cities in the US, along with a class label indicating whether they were a "red" state or a "blue" state in the 2012 election.[1] Specifically, the first column of the variable X contains the longitude and the second variable contains the latitude, while the variable y is set to 1 for blue states and 2 for red states. After it loads the data, it plots the data and then fits two simple classifiers: a classifier that always predicts the most common label (1 in this case) and a decision stump that discretizes the features (by rounding to the nearest integer) and then finds the best equality-based rule (i.e., check if a feature is *equal* to some value). It reports the training error with these two classifiers, then plots the decision areas made by the decision stump.

Note that these functions use the "JLD" package for loading the data and the "PyPlot" package to do the plotting. You can install these packages using:

```
Pkg.add("JLD")
Pkg.add("PyPlot")
```

2.1 Equality vs. Inequality Splitting Rules

In class we discussed splitting rules based on inequalities rather than equalities. Is there a type of feature where it makes sense to use an equality-based splitting rule?

2.2 Decision Stump Implementation

The file *decisionstump.jl* contains a function that finds the best decision stump using the equality rule ("decisionStumpEquality"), and then returns a function that can apply this decision stump to new data. Instead of discretizing the data and using a rule based on testing an equality for a single feature, we want to check whether a feature is above a threshold and split the data accordingly (this is the more sane approach, which we discussed in class). Add a new function "decisionStump" to *decision_stump.jl* that finds the best inequality-based rule, and report the updated error you obtain by using inequalities instead of discretizing and testing equality.

Hint: you may want to start by copy/pasting the contents of of the "decisionStumpEquality" function and then make modifications from there. Note that you should remove the calls to the "round" function for the inequality case. Make sure that you maintain the same input/output format in your function, since otherwise subsequent questions will not work (it should produce a plot that divides the US into a northern blue and a southern red area). If you are new to Julia, you may also want to look at*majorityPredictor.jl* to get an idea of the syntax in a simpler case.

2.3 Constructing Decision Trees

Once your *decisionStump* function is finished, the script *example_decisionTree* will be able to fit a decision tree of depth 2 to the same dataset (which results in a lower training error). Look at how the decision tree is stored and how the (recursive) *predict* function works. Using the same splits as the fitted depth-2 decision tree, write out what an alternate version of the predict function would be for classifying one training example as a simple program using if/else statements (as in slide 9 of L3).

Hint: you may find the "@show" macro really helpful.

[1]The cities data was sampled from `http://simplemaps.com/static/demos/resources/us-cities/cities.csv`. The election information was collected from Wikipedia.

2.4 Cost of Fitting Decision Trees

In class, we discussed how in general the decision stump minimizing the classification error can be found in $O(nd \log n)$ time. Using the greedy recursive splitting procedure, what is the total cost of fitting a decision tree of depth m in terms of n, d, and m?

Hint: even thought there could be $(2^m - 1)$ decision stumps, keep in mind not every stump will need to go through every example. Note also that we stop growing the decision tree if a node has no examples, so we may not even need to do anything for many of the $(2^m - 1)$ decision stumps.

3 Training and Testing

3.1 Traning Error

Running `example_train.jl` fits decision trees of different depths using two different implementations: first, our own implementation using your "decisionStump" function, and also using a variant using a more sophisticated splitting criterion called the information gain. Describe what you observe. Can you explain the results?

3.2 Training and Testing Error Curves

Notice that the *citiesSmall.mat* file also contains test data, "Xtest" and "ytest". Running *example_trainTest* trains a depth-2 decision tree and evaluates its performance 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.

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: use the provided infomax-based decision tree code from the previous subsection.

3.3 Validation Set

Suppose we're in the typical case where we don't have the labels for the test data. 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 if we minimized the validation set error? Does the answer change if you switch the training and validation set? How could we use more of our data to estimate the depth more reliably?

Note: use the provided infomax-based decision tree code from the previous subsection.

4 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.

4.1 Naive Bayes by Hand

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):

- $p(y = 1)$.
- $p(y = 0)$.

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

- $p(x_1 = 1|y = 1)$.
- $p(x_2 = 0|y = 1)$.
- $p(x_1 = 1|y = 0)$.
- $p(x_2 = 0|y = 0)$.

(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.)

4.2 Bag of Words

If you run the script *example_bagOfWods.jl*, it will load the following dataset:

1. X: A sparse 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 1 through 4, 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 is present in the newsgroup post if there is a 1 in column 50 of X?

2. Which words are present in training example 500?
3. Which newsgroup name does training example 500 come from?

4.3 Naive Bayes Implementation

If you run the function *example_decisionTree_newsgroups.jl* it will load the newsgroups dataset and report the test error for decision trees of different sizes (it may take a while for the deeper trees, as this is a sub-optimal implementation). On other other hand, if you run the function *example_naiveBayes.jl* it will fit the basic naive Bayes model and report the test error.

While the *predict* function of the naive Bayes classifier is already implemented, 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. Hand in your code and report the test error that you obtain.

4.4 Runtime of Naive Bayes for Discrete Data

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?

5 K-Nearest Neighbours

In *citiesSmall* dataset, nearby points tend to receive the same class label because they are part of the same state. This indicates that a $k-$nearest neighbours classifier might be a better choice than a decision tree (while naive Bayes would probably work really badly on this dataset). The file *knn.jl* has implemented the training function for a $k-$nearest neighbour classifier (which is to just memorize the data) but the predict function always just predicts 1.

5.1 KNN Prediction

Fill in the *predict* function in *knn.jl* so that the model file implements the k-nearest neighbour prediction rule. You should use Euclidean distance, and you may find it useful to pre-compute all the distances and then use the "sortperm" command.

1. Hand in the predict function.
2. Report the training and test error obtained on the *citiesSmall.mat* dataset for $k = 1$, $k = 3$, and $k = 10$. (You can use *example_knn.jl* to get started.)

3. Hand in the plot generatied by classifier2Dplot on the *citiesSmall.mat* dataset for $k = 1$ on the training data.
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?

5.2 Condensed Nearest Neighbours

The file *citiesBig1.mat* contains a version of this dataset with more than 30 times as many cities. KNN can obtain a lower test error if it's trained on this dataset, but the prediction time will be very slow. A common strategy for applying KNN to huge datasets is called *condensed nearest neighbours*, and the main idea is to only store a *subset* of the training examples (and to only compare to these examples when making predictions). A simple variation of this algorithm would be:

```
initialize subset with first training example;
for each training example do
    if the example is incorrectly classified by the KNN classifier using the current subset then
        add the current example to the subset;
    else
        do not add the current example to the subset (do nothing);
    end
end
```

Algorithm 1: Condensed Nearest Neighbours

You are provided with an implementation of this *condensed nearest neighbours* algorithm in *knn.jl*.

1. The point of this algorithm is to be faster than KNN. Try running the condensed nearest neighbours (called "cknn" in the code) on the *citiesBig1* dataset and report how long it takes to make a prediction. What about if you try to use KNN for this dataset?
2. Report the training and testing errors for condensed NN, as well as the number of training examples in the subset, on the *citiesBig1* dataset with $k = 1$.
3. Why is the training error with $k = 1$ now greater than 0?
4. If you have s examples in the subset, what is the cost of running the predict function on t test examples in terms of n, d, t, and s?
5. Try out your function on the dataset *citiesBig2*. Why are the test error *and* training error so high (even for $k = 1$) for this method on this dataset?

6 Very-Short Answer Questions

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 one reason we would want to look at scatterplots of the data before doing supervised learning?
2. What is a reason that the examples in a training and test set might not be IID?
3. What is the difference between a validation set and a test set?
4. Why is naive Bayes called "naive"?
5. What is a situation where the naive Bayes assumption could lead to poor performance?

6. What is the main advantage of non-parametric models?
7. A standard pre-processing step is "standardization" of the features: for each column of X we subtract its mean and divide by its variance. Would this pre-processing change the accuracy of a decision tree classifier? Would it change the accuracy of a KNN classifier?
8. Does increasing k in KNN affect the training or prediction asymptotic runtimes?
9. How does increase the parameter k in k-nearest neighbours affect the two parts (training error and approximation error) of the fundamental trade-off (hint: think of the extreme values).
10. For any parametric model, how does increasing number of training examples n affect the two parts of the fundamental trade-off.