CPSC 340 Assignment 1 (due 2019-01-11 at 11:55pm)

Commentary on Assignment 1: CPSC 340 is tough because it combines knowledge and skills across several disciplines. To succeed in the course, you will need to know or very quickly get up to speed on:

- Basic Python programming, including NumPy and plotting with matplotlib.

- Math to the level of the course prerequisites: linear algebra, multivariable calculus, some probability.

- Statistics, algorithms and data structures to the level of the course prerequisites.

- Some basic LaTeX and git skills so that you can typeset equations and submit your assignments.

This assignment will help you assess whether you are prepared for this course. We anticipate that each of you will have different strengths and weaknesses, so don't be worried if you struggle with *some* aspects of the assignment. But if you find this assignment to be very difficult overall, that is a warning sign that you may not be prepared to take CPSC 340 at this time. Future assignments will be more difficult than this one (and probably around the same length).

Questions 1-4 are on review material, that we expect you to know coming into the course. The rest is new CPSC 340 material from the first few lectures.

A note on the provided code: in the `code` directory we provide you with a file called `main.py`. This file, when run with different arguments, runs the code for different parts of the assignment. For example,

```
python main.py -q 6.2
```

runs the code for Question 6.2. At present, this should do nothing (throws a `NotImplementedError`), because the code for Question 6.2 still needs to be written (by you). But we do provide some of the bits and pieces to save you time, so that you can focus on the machine learning aspects. For example, you'll see that the provided code already loads the datasets for you. The file `utils.py` contains some helper functions. You don't need to read or modify the code in there. To complete your assignment, you will need to modify `grads.py`, `main.py`, `decision_stump.py` and `simple_decision.py` (which you'll need to create).

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 Linear Algebra Review

For these questions you may find it helpful to review these notes on linear algebra:
`http://www.cs.ubc.ca/~schmidtm/Documents/2009_Notes_LinearAlgebra.pdf`

1.1 Basic Operations

Use the definitions below,

$$\alpha = 2, \quad x = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}, \quad z = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}, \quad A = \begin{bmatrix} 3 & 2 & 2 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix},$$

and use x_i to denote element i of vector x. Evaluate the following expressions (you do not need to show your work).

1. $\sum_{i=1}^{n} x_i y_i$ (inner product). 14

2. $\sum_{i=1}^{n} x_i z_i$ (inner product between orthogonal vectors). 0

3. $\alpha(x + z)$ (vector addition and scalar multiplication). $\begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix}$

4. $x^T z + \|x\|$ (inner product in matrix notation and Euclidean norm of x). $\sqrt{5}$

5. Ax (matrix-vector multiplication). $\begin{bmatrix} 6 \\ 5 \\ 7 \end{bmatrix}$

6. $x^T Ax$ (quadratic form). 19

7. $A^T A$ (matrix tranpose and matrix multiplication). $\begin{bmatrix} 11 & 10 & 10 \\ 10 & 14 & 10 \\ 10 & 10 & 14 \end{bmatrix}$

1.2 Matrix Algebra Rules

Assume that $\{x, y, z\}$ are $n \times 1$ column vectors, $\{A, B, C\}$ are $n \times n$ real-valued matrices, 0 is the zero matrix of appropriate size, and I is the identity matrix of appropriate size. State whether each of the below is true in general (you do not need to show your work).

1. $x^T y = \sum_{i=1}^{n} x_i y_i$. True

2. $x^T x = \|x\|^2$. True

3. $x^T x = xx^T$. False

4. $(x - y)^T (x - y) = \|x\|^2 - 2x^T y + \|y\|^2$. True

5. $AB = BA$. False

6. $A^T (B + IC) = A^T B + A^T C$. True

7. $(A + BC)^T = A^T + B^T C^T$. False

8. $x^T Ay = y^T A^T x$. False

9. $A^T A = AA^T$ if A is a symmetric matrix. True

10. $A^T A = 0$ if the columns of A are orthonormal. False

2 Probability Review

For these questions you may find it helpful to review these notes on probability:
http://www.cs.ubc.ca/~schmidtm/Courses/Notes/probability.pdf
And here are some slides giving visual representations of the ideas as well as some simple examples:
http://www.cs.ubc.ca/~schmidtm/Courses/Notes/probabilitySlides.pdf

2.1 Rules of probability

Rubric: {reasoning:6}

Answer the following questions. You do not need to show your work.

1. You are offered the opportunity to play the following game: your opponent rolls 2 regular 6-sided dice. If the difference between the two rolls is at least 3, you win \$15. Otherwise, you get nothing. What is a fair price for a ticket to play this game once? In other words, what is the expected value of playing the game? $5

2. Consider two events A and B such that $\Pr(A, B) = 0$ (they are mutually exclusive). If $\Pr(A) = 0.4$ and $\Pr(A \cup B) = 0.95$, what is $\Pr(B)$? Note: $p(A, B)$ means "probability of A and B" while $p(A \cup B)$ means "probability of A or B". It may be helpful to draw a Venn diagram. 0.55

3. Instead of assuming that A and B are mutually exclusive ($\Pr(A, B) = 0$), what is the answer to the previous question if we assume that A and B are independent? $\frac{11}{12}$

2.2 Bayes Rule and Conditional Probability

Rubric: {reasoning:10}

Answer the following questions. You do not need to show your work.

Suppose a drug test produces a positive result with probability 0.97 for drug users, $P(T = 1 \mid D = 1) = 0.97$. It also produces a negative result with probability 0.99 for non-drug users, $P(T = 0 \mid D = 0) = 0.99$. The probability that a random person uses the drug is 0.0001, so $P(D = 1) = 0.0001$.

1. What is the probability that a random person would test positive, $P(T = 1)$? 0.010096

2. In the above, do most of these positive tests come from true positives or from false positives? False positives

3. What is the probability that a random person who tests positive is a user, $P(D = 1 \mid T = 1)$? $\frac{97}{10096}$

4. Suppose you have given this test to a random person and it came back positive, are they likely to be a drug user? No, the probability is too low

5. What is one factor you could change to make this a more useful test? Assuming the drug tests are independent, I would increase the times the tests are ran. If both the tests are positive, the probability of him or her being a user is higher than just running one test on the individual. That is, $P(user|++) > P(user|+)$

3 Calculus Review

3.1 One-variable derivatives

Answer the following questions. You do not need to show your work.

1. Find the derivative of the function $f(x) = 3x^2 - 2x + 5$. $6x - 2$

2. Find the derivative of the function $f(x) = x(1-x)$. $1 - 2x$

3. Let $p(x) = \frac{1}{1+\exp(-x)}$ for $x \in \mathbb{R}$. Compute the derivative of the function $f(x) = x - \log(p(x))$ and simplify it by using the function $p(x)$. $p(x)$

Remember that in this course we will $\log(x)$ to mean the "natural" logarithm of x, so that $\log(\exp(1)) = 1$. Also, obseve that $p(x) = 1 - p(-x)$ for the final part.

3.2 Multi-variable derivatives

Compute the gradient vector $\nabla f(x)$ of each of the following functions. You do not need to show your work.

1. $f(x) = x_1^2 + \exp(x_1 + 2x_2)$ where $x \in \mathbb{R}^2$. $(2x_1 + \exp(x_1 + 2x_2), 2\exp(x_1 + 2x_2))$

2. $f(x) = \log\left(\sum_{i=1}^{3} \exp(x_i)\right)$ where $x \in \mathbb{R}^3$ (simplify the gradient by defining $Z = \sum_{i=1}^{3} \exp(x_i)$). $\left(\frac{exp(x_1)}{Z}, \frac{exp(x_2)}{Z}, \frac{exp(x_3)}{Z}\right)$

3. $f(x) = a^T x + b$ where $x \in \mathbb{R}^3$ and $a \in \mathbb{R}^3$ and $b \in \mathbb{R}$. a^T

4. $f(x) = \frac{1}{2}x^\top A x$ where $A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$ and $x \in \mathbb{R}^2$. Ax

5. $f(x) = \frac{1}{2}\|x\|^2$ where $x \in \mathbb{R}^d$. x

Hint: it is helpful to write out the linear algebra expressions in terms of summations.

3.3 Optimization

Find the following quantities. You do not need to show your work. You can/should use your results from parts 3.1 and 3.2 as part of the process.

1. $\min 3x^2 - 2x + 5$, or, in words, the minimum value of the function $f(x) = 3x^2 - 2x + 5$ for $x \in \mathbb{R}$. $\frac{14}{3}$

2. $\max x(1-x)$ for $x \in [0,1]$. $\frac{1}{4}$

3. $\min x(1-x)$ for $x \in [0,1]$. 0

4. $\arg\max x(1-x)$ for $x \in [0,1]$. $\frac{1}{2}$

5. $\min x_1^2 + \exp(x_2)$ where $x \in [0,1]^2$, or in other words $x_1 \in [0,1]$ and $x_2 \in [0,1]$. 1

6. $\arg\min x_1^2 + \exp(x_2)$ where $x \in [0,1]^2$. $x_1 = 0, x_2 = 0$

Note: the notation $x \in [0,1]$ means "x is in the interval $[0,1]$", or, also equivalently, $0 \le x \le 1$.

Note: the notation "$\max f(x)$" means "the value of $f(x)$ where $f(x)$ is maximized", whereas "$\arg\max f(x)$" means "the value of x such that $f(x)$ is maximized". Likewise for min and $\arg\min$. For example, the min of the function $f(x) = (x-1)^2$ is 0 because the smallest possible value is $f(x) = 0$, whereas the arg min is 1 because this smallest value occurs at $x = 1$. The min is always a scalar but the arg min is a value of x, so it's a vector if x is vector-valued.

3.4 Derivatives of code

Note: for info on installing and using Python, see
`https://www.cs.ubc.ca/~fwood/CS340/python/`.

The homework zip file contains a file named `grads.py` which defines several Python functions that take in
an input variable x, which we assume to be a 1-d array (in math terms, a vector). It also includes (blank)
functions that return the corresponding gradients. For each function, write code that computes the gradient
of the function in Python. You should do this directly in `grads.py`; no need to make a fresh copy of the file.
However, per the homework instructions, you should include it in the latex report as described so that the
TA can assess it easily. When finished, you can run `python main.py -q 3.4` to test out your code.

Hint: it's probably easiest to first understand on paper what the code is doing, then compute the gradient,
and then translate this gradient back into code.

Note: do not worry about the distinction between row vectors and column vectors here. For example, if the
correct answer is a vector of length 5, we'll accept numpy arrays of shape `(5,)` (a 1-d array) or `(5,1)` (a
column vector) or `(1,5)` (a row vector). In future assignments we will start to be more careful about this.

Warning: Python uses whitespace instead of curly braces to delimit blocks of code. Some people use tabs
and other people use spaces. My text editor (Atom) inserts 4 spaces (rather than tabs) when I press the Tab
key, so the file `grads.py` is indented in this manner. If your text editor inserts tabs, Python will complain
and you might get mysterious errors... this is one of the most annoying aspects of Python, especially when
starting out. So, please be aware of this issue! And if in doubt you can just manually indent with 4 spaces, or
convert everything to tabs. For more information see `https://www.youtube.com/watch?v=SsoOG6ZeyUI`.

```python
import numpy as np

def example(x):
    return np.sum(x**2)


def example_grad(x):
    return 2*x

def foo(x):
    result = 1
       = 4 # this is here to make sure you're using Python 3
    for x_i in x:
        result += x_i**
    return result

def foo_grad(x):
    a = []
    for x_i in x:
        a.append(4*(x_i**3))
    a = np.array(a)
    return a

def bar(x):
    return np.prod(x)

def bar_grad(x):
    a=[]
    for x_i in x:
```

```
        a.append(float(np.prod(x))/x_i)
    a = np.array(a)
    return a
```

4 Algorithms and Data Structures Review

4.1 Trees

Answer the following questions. You do not need to show your work.

1. What is the minimum depth of a binary tree with 64 leaf nodes? 6

2. What is the minimum depth of binary tree with 64 nodes (includes leaves and all other nodes)? 6

Note: we'll use the standard convention that the leaves are not included in the depth, so a tree with depth 1 has 3 nodes with 2 leaves.

4.2 Common Runtimes

Answer the following questions using big-O notation You do not need to show your work.

1. What is the cost of finding the largest number in an unsorted list of n numbers? $O(n)$

2. What is the cost of finding the smallest element greater than 0 in a *sorted* list with n numbers. $O(log(n))$

3. What is the cost of finding the value associated with a key in a hash table with n numbers? (Assume the values and keys are both scalars.) $O(1)$

4. What is the cost of computing the inner product $a^T x$, where a is $d \times 1$ and x is $d \times 1$? $O(d)$

5. What is the cost of computing the quadratic form $x^T A x$ when A is $d \times d$ and x is $d \times 1$. $O(d^2)$

4.3 Running times of code

Your repository contains a file named `bigO.py`, which defines several functions that take an integer argument N. For each function, state the running time as a function of N, using big-O notation. Please include your answers in your report. Do not write your answers inside `bigO.py`.

func1(N): $O(N)$
func2(N): $O(N)$
func3(N): $O(1)$
func4(N): $O(N^2)$

5 Data Exploration

Your repository contains the file `fluTrends.csv`, which contains estimates of the influenza-like illness percentage over 52 weeks on 2005-06 by Google Flu Trends. Your `main.py` loads this data for you and stores it in a pandas DataFrame `X`, where each row corresponds to a week and each column corresponds to a different region. If desired, you can convert from a DataFrame to a raw numpy array with `X.values()`.

5.1 Summary Statistics

Rubric: {reasoning:2}

Report the following statistics:

1. The minimum, maximum, mean, median, and mode of all values across the dataset.
   minimum: 0.35200000000000004
   maximum: 4.862
   mean: 1.3246250000000002
   median: 1.1589999999999998
   mode: 0.77

2. The 5%, 25%, 50%, 75%, and 95% quantiles of all values across the dataset.
   5% quantile: 0.46495000000000003
   25% quantile: 0.718
   50% quantile: 1.1589999999999998
   75% quantile: 1.81325
   95% quantile: 2.624049999999999

3. The names of the regions with the highest and lowest means, and the highest and lowest variances.
   Highest mean: WtdILI
   Lowest mean: Pac
   Highest variance: Mtn
   Lowest variance: Pac

In light of the 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. Note: the function `utils.mode()` will compute the mode value of an array for you. In case of continuous data, mode cannot be a reliable estimate of the most "common" value. This is because it is incredibly rare that more than one data are exactly the same in a continuous range. It would be better to assign the values to intervals of equal distance, make a histogram out of it, and then assign the mode to the interval which has the highest peak.

5.2 Data Visualization

Rubric: {reasoning:3}

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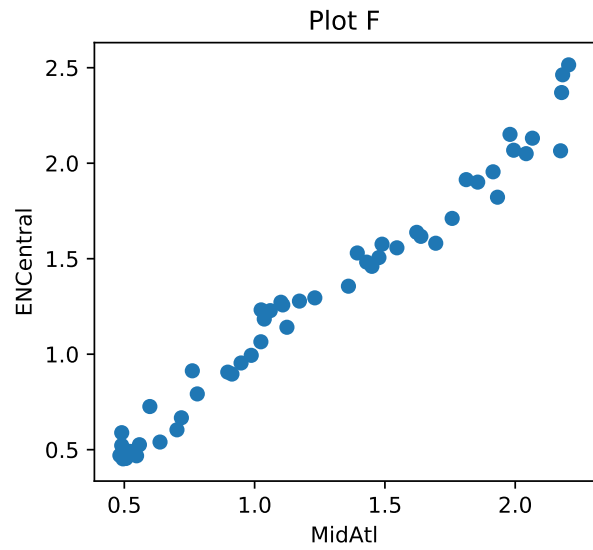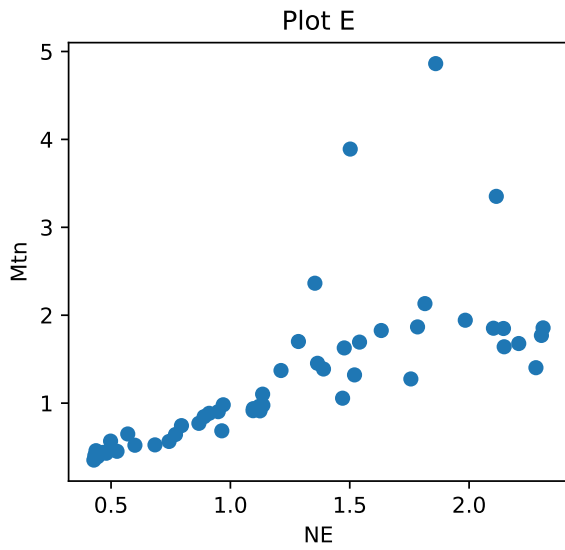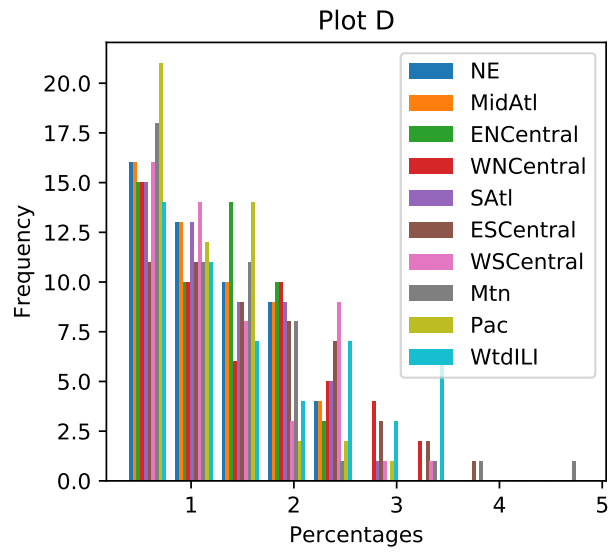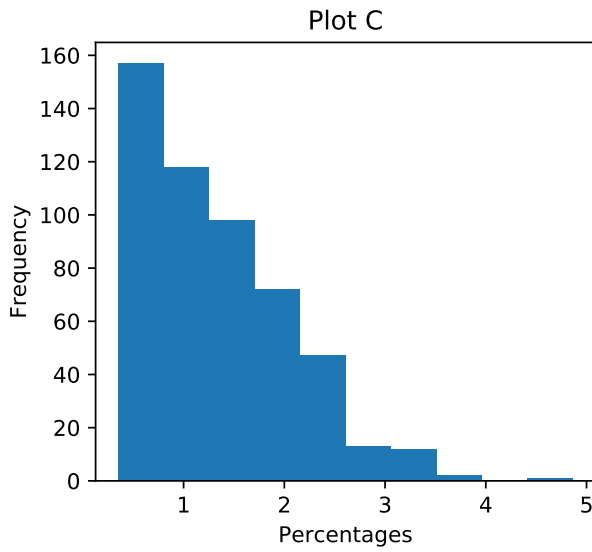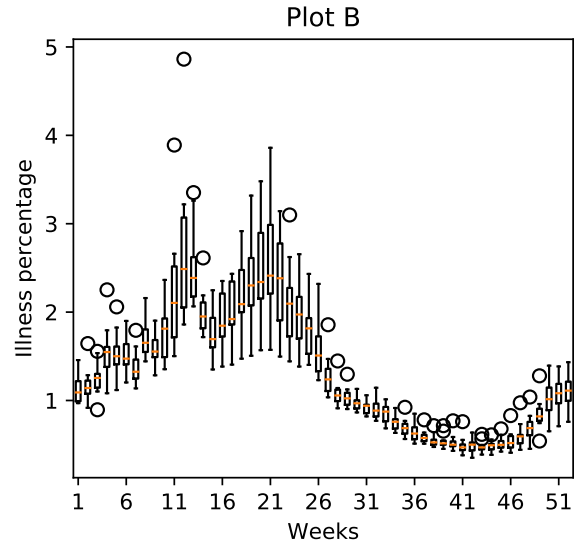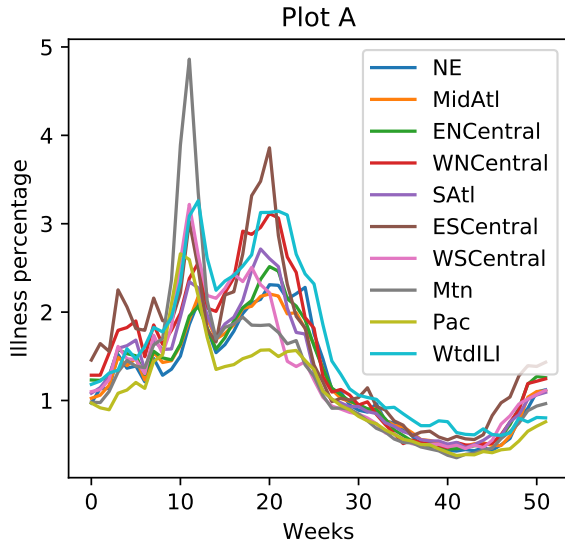
The figure contains the following plots, in a shuffled order:

1. A single histogram showing the distribution of *each* column in X. D. Each column of X represents the frequency data of a region, which is shown in the graph

2. A histogram showing the distribution of each the values in the matrix X. C. This can be seen from x-axis. The data does not go higher than 5 percent, which is exactly how the data is.

3. A boxplot grouping data by weeks, showing the distribution across regions for each week. B. Literally this is the only boxplot.

4. A plot showing the illness percentages over time. A. This literally is a plot showing the illness percentages(y-axis) over time(x-axis)

5. A scatterplot between the two regions with highest correlation. F. Linear correlation is strong

6. A scatterplot between the two regions with lowest correlation. E. Linear correlation is weak due to outliers

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.

6 Decision Trees

If you run `python main.py -q 6`, it will load a dataset containing longitude 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 0 for blue states and 1 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 (0 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. The plot is shown below:

---

[1]The cities data was sampled from `http://simplemaps.com/static/demos/resources/us-cities/cities.csv`. The election information was collected from Wikipedia.
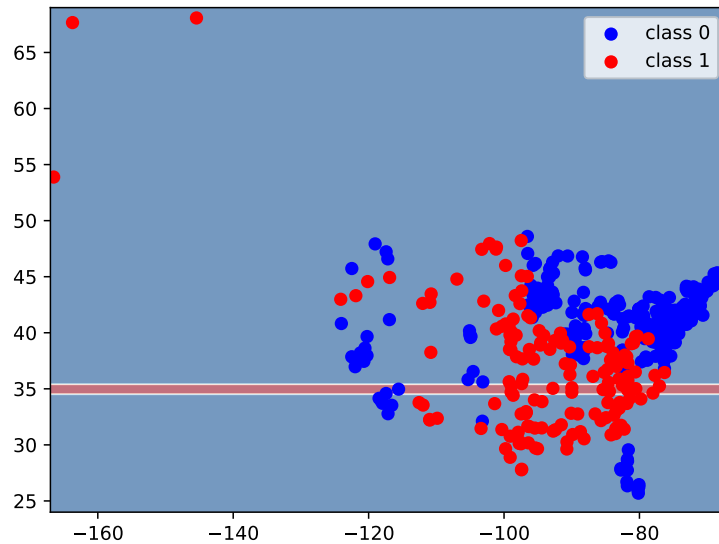
As you can see, it is just checking whether the latitude equals 35 and, if so, predicting red (Republican). This is not a very good classifier.

6.1 Splitting rule

Is there a particular type of features for which it makes sense to use an equality-based splitting rule rather than the threshold-based splits we discussed in class?

If the data can only have the following two values 0 and 1 like boolean data, the equality-based splitting rule makes more sense over threshold-based splitting.

6.2 Decision Stump Implementation

The file `decision_stump.py` contains the class `DecisionStumpEquality` which finds the best decision stump using the equality rule and then makes predictions using that rule. 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 or below a threshold and split the data accordingly (this is a more sane approach, which we discussed in class). Create a `DecisionStumpErrorRate` class to do this, and report the updated error you obtain by using inequalities instead of discretizing and testing equality. Also submit the generated figure of the classification boundary.

Hint: you may want to start by copy/pasting the contents `DecisionStumpEquality` and then make modifications from there.
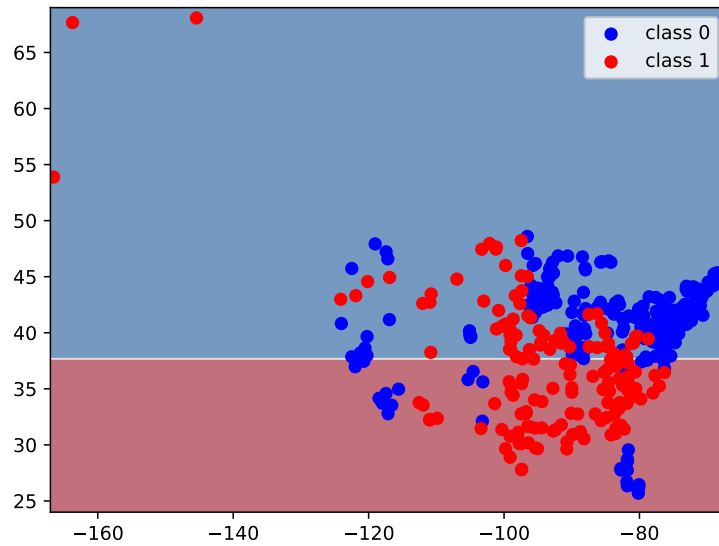
```python
class DecisionStumpErrorRate:

    def __init__(self):
        pass

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

        # Get an array with the number of 0's, number of 1's, etc.
        count = np.bincount(y,minlength = 2)

        # Get the index of the largest value in count.
        # Thus, y_mode is the mode (most popular value) of y
        y_mode = np.argmax(count)

        self.splitSat = y_mode
        self.splitNot = None
        self.splitVariable = None
        self.splitValue = None

        # If all the labels are the same, no need to split further
        if np.unique(y).size <= 1:
            return

        minError = np.sum(y != y_mode)

        # Loop over features looking for the best split
        # X = np.round(X)

        for d in range(D):
            for n in range(N):
                # Choose value to equate to
                value = X[n, d]
```

```
            # Find most likely class for each split
            y_sat = utils.mode(y[X[:,d] > value])
            # y_not = The one that has more between 0 and 1 that's not the same with value
            y_not = utils.mode(y[X[:,d] <= value])

            # Make predictions
            y_pred = y_sat * np.ones(N)
            y_pred[X[:, d] <= value] = y_not

            # Compute error
            errors = np.sum(y_pred != y)

            # Compare to minimum error so far
            if errors < minError:
                # This is the lowest error, store this value
                minError = errors
                self.splitVariable = d
                self.splitValue = value
                self.splitSat = y_sat
                self.splitNot = y_not

    def predict(self, X):

        M, D = X.shape
        # X = np.round(X)

        if self.splitVariable is None:
            return self.splitSat * np.ones(M)

        yhat = np.zeros(M)

        for m in range(M):
            if X[m, self.splitVariable] > self.splitValue:
                yhat[m] = self.splitSat
            else:
                yhat[m] = self.splitNot

        return yhat
```

Error rate: 0.253

6.3   Decision Stump Info Gain Implementation

Rubric: {code:3}

In decision_stump.py, create a DecisionStumpInfoGain class that fits using the information gain criterion discussed in lecture. Report the updated error you obtain, and submit the classification boundary figure.

Notice how the error rate changed. Are you surprised? If so, hang on until the end of this question!

Note: even though this data set only has 2 classes (red and blue), your implementation should work for any number of classes, just like DecisionStumpEquality and DecisionStumpErrorRate.

Hint: take a look at the documentation for np.bincount, at
https://docs.scipy.org/doc/numpy/reference/generated/numpy.bincount.html. The minlength ar-

gument comes in handy here to deal with a tricky corner case: when you consider a split, you might not have any examples of a certain class, like class 1, going to one side of the split. Thus, when you call `np.bincount`, you'll get a shorter array by default, which is not what you want. Setting `minlength` to the number of classes solves this problem.
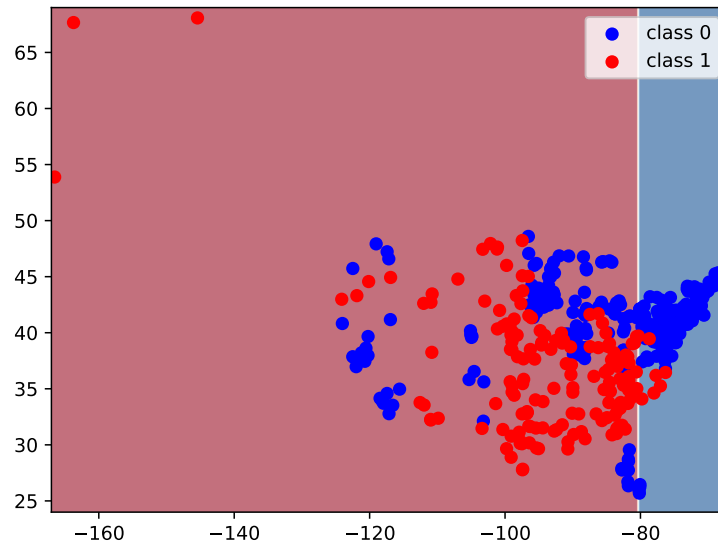

```
class DecisionStumpInfoGain(DecisionStumpErrorRate):
    def fit(self, X, y):
        N, D = X.shape

        # Get an array with the number of 0's, number of 1's, etc.
        count = np.bincount(y, minlength = 2)

        # Get the index of the largest value in count.
        # Thus, y_mode is the mode (most popular value) of y
        y_mode = np.argmax(count)

        self.splitSat = y_mode
        self.splitNot = None
        self.splitVariable = None
        self.splitValue = None

        # If all the labels are the same, no need to split further
        if np.unique(y).size <= 1:
            return

        maxIG = 0

        # Loop over features looking for the best split
        # X = np.round(X)

        for d in range(D):
            for n in range(N):
                # Choose value to equate to
```

```python
            value = X[n, d]

            y_normalize = np.divide(y,float(y.size))
            # Find most likely class for each split
            y_yes = y[X[:,d] > value]
            # y_no = The one that has more between 0 and 1 that's not the same with value
            y_no = y[X[:,d] <= value]

            #print(y_yes)
            #print(y_no)
            # Find most likely class for each split
            y_sat = utils.mode(y[X[:,d] > value])
            # y_not = The one that has more between 0 and 1 that's not the same with value
            y_not = utils.mode(y[X[:,d] <= value])

            y_yes_bincount = np.bincount(y_yes, minlength = 2)
            y_no_bincount = np.bincount(y_no, minlength = 2)

            # print(y_yes.size)
            y_yes_size = y_yes.size
            if (y_yes_size == 0):
                y_yes_size = 1

            # print(y_no.size)
            y_no_size = y_no.size
            if (y_no_size == 0):
                y_no_size = 1

            y_yes_bincount = np.divide(y_yes_bincount, float(y_yes_size))

            y_no_bincount = np.divide(y_no_bincount, float(y_no_size))

            # Compute IG
            IG = entropy(y_normalize) - (y_yes.size/float(N)*entropy(y_yes_bincount)) - (y_n

            # Compare to minimum error so far
            if IG > maxIG:
                # This is the lowest error, store this value
                maxIG = IG
                self.splitVariable = d
                self.splitValue = value
                self.splitSat = y_sat
                self.splitNot = y_not
```

Error: 0.325

6.4 Constructing Decision Trees

Once your DecisionStumpInfoGain class is finished, running python main.py -q 6.4 will 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 splits from the fitted depth-2 decision tree, write a hard-coded version of the predict function that classifies one example using simple if/else

statements (see the Decision Trees lecture). Save your code in a new file called `simple_decision.py` (in the `code` directory) and make sure you include the text of this file in your LaTeX report.

Note: this code should implement the specific, fixed decision tree which was learned after calling `fit` on this particular data set. It does not need to be a learnable model. You should just hard-code the split values directly into the code. Only the `predict` function is needed.

Hint: if you plot the decision boundary you can do a visual sanity check to see if your code is consistent with the plot.
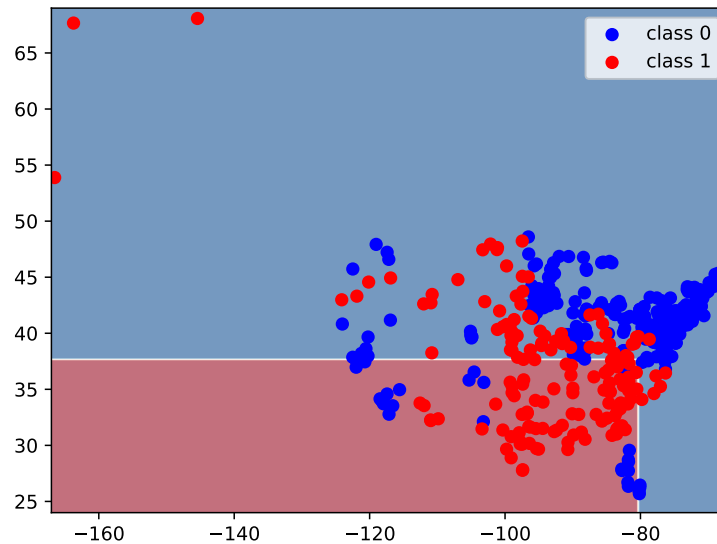
Error: 0.242

```python
def predict(self, X):
        R, C = X.shape
        yhat = np.zeros(R)
        for m in range(R):
            if X[m, 0] > -80.305106:
                yhat[m] = 0
            else:
                if X[m, 1] > 37.669007:
                    yhat[m] = 0
                else:
                    yhat[m] = 1
        return yhat
```

6.5 Decision Tree Training Error

Rubric: {reasoning:2}

Running `python main.py -q 6.5` fits decision trees of different depths using the following different implementations:
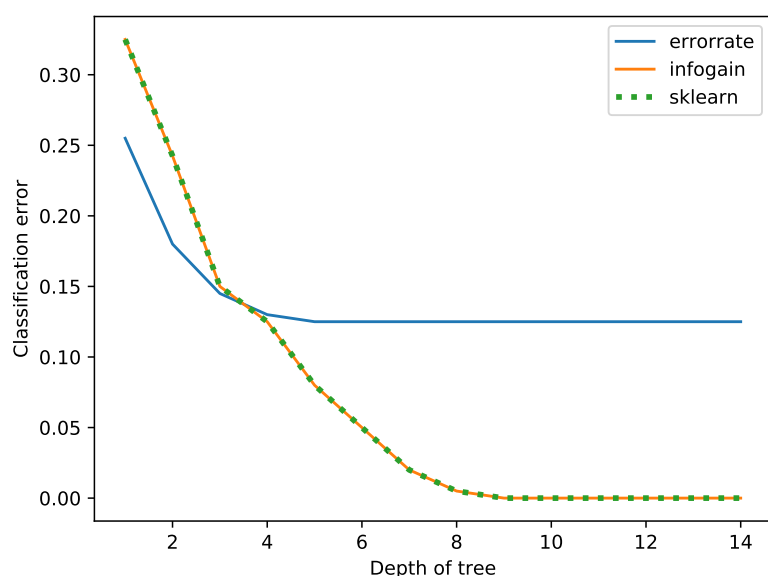
1. A decision tree using `DecisionStump`

2. A decision tree using `DecisionStumpInfoGain`

3. The `DecisionTreeClassifier` from the popular Python ML library *scikit-learn*

Run the code and look at the figure. Describe what you observe. Can you explain the results? Why is approach (1) so disappointing? Also, submit a classification boundary plot of the model with the lowest training error.

Note: we set the `random_state` because sklearn's `DecisionTreeClassifier` is non-deterministic. This is probably because it breaks ties randomly.

Note: the code also prints out the amount of time spent. You'll notice that sklearn's implementation is substantially faster. This is because our implementation is based on the $O(n^2 d)$ decision stump learning algorithm and sklearn's implementation presumably uses the faster $O(nd \log n)$ decision stump learning algorithm that we discussed in lecture.

The classification error that came from DecisionStump, in which we used accuracy as our primary goal for prediction, was the highest as the depth of decision tree got deeper. However, the infogain method and the sklearn method showed the lowest errors as the depth got deeper. This is an example where accuracy as a criteria for splitting fails. Because the data is already quite clustered per class, simply dividing it using a horizontal or vertical line will not increase accuracy that much. However, according to the InfoGain method this is considered ideal since it makes labels less random. As a result, as the tree goes deeper(i.e. Having more horizontal or vertical lines), the InfoGain method eventually leads to high accuracy
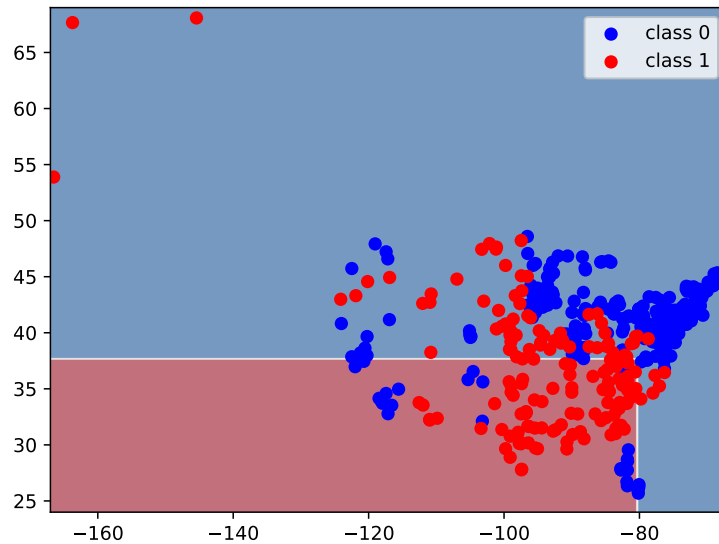
6.6 Comparing implementations

Rubric: {reasoning:2}

In the previous section you compared different implementations of a machine learning algorithm. Let's say that two approaches produce the exact same curve of classification error rate vs. tree depth. Does this conclusively demonstrate that the two implementations are the same? If so, why? If not, what other experiment might you perform to build confidence that the implementations are probably equivalent?

No, that would not demonstrate that the two implementations are the same. It could be mere coincidence. For example, say that the data we have is perfectly symmetrical to a point on a plot. If the data is classified by a straight line that passes that point, regardless of the classification algorithm having the criteria based on accuracy or information gain, the classification error curve and the tree depth could be the same. So if possible, more data would be required to be tested.

6.7 Cost of Fitting Decision Trees

Rubric: {reasoning:3}

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.

$O(mnd \log n)$