Basic One Dimensional Clustering and Classification, Fall 2024

See Dropbox for Due Date

Thomas B. Kinsman

Homework is to be programmed in Python, R, Matlab, or Java.

When coding, assume that the grader has no knowledge of the language or API calls but can read comments. Use prolific comments before each section of code, or function call to explain what the code does, and why you are using it.

Hand in your results, and the commented code, in the same associated dropbox.

Create a directory named HW_NN_<LASTNAME>_<Firstname>_dir.
Do all your work in that directory. Zip up the entire directory and submit in one zip file.
The zip file should expand to one directory that contains: your code, your results, and so forth.

Your writeup should be called HW_NN_<LASTNAME>_<Firstname>.pdf, and other files should follow the same patterns so we can tell them apart. Substitute the homework number for NN.

Feel free to look over each other's shoulders, at each other's work, but do you own work.
Let me know whom you worked with. Do not hand in copies of each other's code.

1D Classification using a threshold classifier:

The homework assignments build on each other. You can probably start with what you did for the last homework and modify it. Just be careful to watch for changes.

Read this:

- Please, read the entire assignment before you code anything.
- There is only one attribute to try to set.
- There will be questions about this homework later, on quizzes and exams.
- You will grow the skeleton of this homework later in other assignments.
- **CAUTION:** You will use the basis of this homework, to do the next homework as well.
  Make sure you make notes to yourself about what you did, so you can copy and reuse the code.

Overview:

- Find the speed that splits the cars into two clusters.
  In other words, find the speed at which the minimum mixed variance occurs.
  This means:
  - That the two clusters are as compact as possible.
  - The intra-cluster distance is as small as possible.
  - You have found **argmin of { the mixed variance }**

- Create a Receiver Operator Characteristics curve of your data.
  Label the critical points. Label the axes.

- Put <u>your</u> best threshold point on the ROC curve

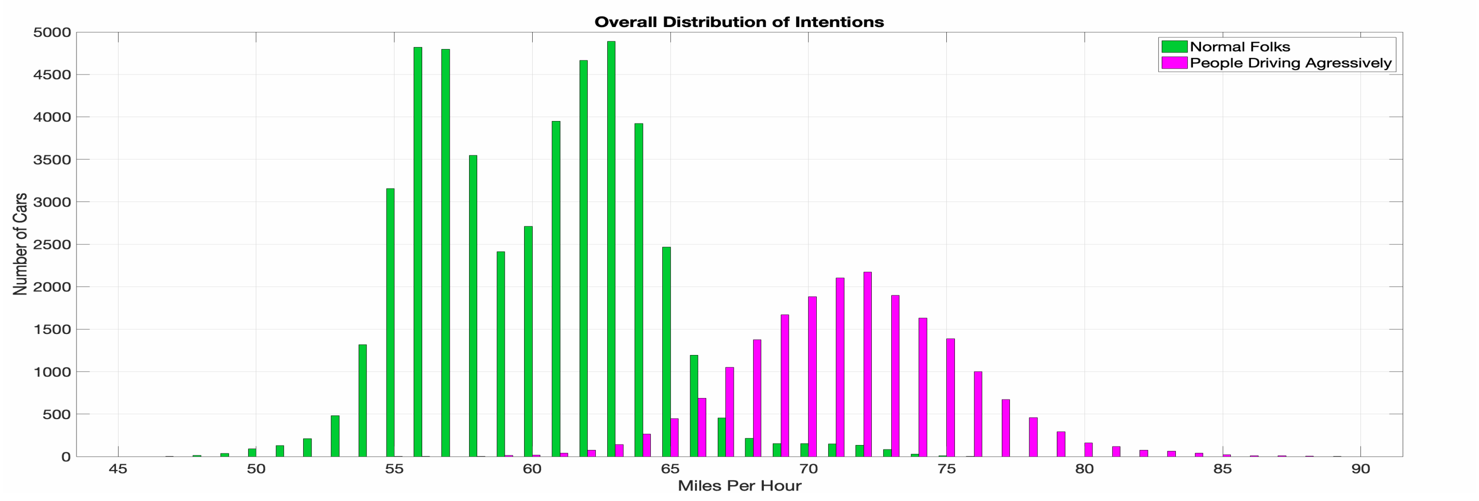
- Compare to the point which minimizes the mixed variance of the data.

Given:
You are provided with a new data set, or sets.
The data includes the recorded speed, and the driver's intention:

The intentions for this homework are:

0	for not in any hurry at all, totally safe drivers
1	for being a normal driver, a bit anxious, but not being aggressive.
2	for drivers who are driving aggressively. **For this homework, this is your target variable.** **Your goal is to predict when the intention is 2.**

Exploratory Data Analysis:
A bar graph of the data might look like this:



How to proceed:

1. Create a program named HW_NN_Lastname_FirstName.py, or .m, or .r, …
2. Read in all of the data.

3. Truncate the data to the nearest mile per hour. ← This is different!
   numpy.floor( ) will do the trick. That is, the bin size is 1 mile per hour for this assignment.

4. For each and every possible threshold speed, from low to high, using a linear search:
   a. Find the number of non-aggressive drivers who are > this threshold speed
      This is the number of false alarms.

      Compute the False Alarm Rate, the FAR:
      i. for each speed, the number of false alarms divided by the total number of non-aggressive drivers.
      ii. in other words, the number of drivers you caught, but did not want to, divided by the number of drivers you never wanted to catch.

      Track the FAR for each speed.

   b. Find the number of aggressive drivers who are > this speed
      This is the number of hits, or true positives.

      Compute the True Positive Rate, the TPR:
      i. for each speed, the number of true positives divided by the total number of aggressive drivers.
      ii. in other words, the number of drivers you caught, and wanted to catch,
          divided by the number of drivers you would ever want to catch.

      Track the TPR for each speed.

   c. Find the number of mistakes total.
      i. If this total number of mistakes is less than or equal to (≤) any number seen yet,
         Keep this threshold around, and the number of mistakes found at this speed.
      ii. Realize that the previous "if statement" just broke a possible tie situation.

5. Have your program open another file named "HW_02_LastName_FirstName_Classifier.py"
   In that file, write a valid piece of code, a one-rule, that only depends on only one variable,
   that records the threshold you found.

   A one-rule is an "if-then-else statement". You need to have both parts of the statement.

   This is an "if statement" that tells your rule. It *might* look something like this, or it *might not*.
   Or it might be ≤, or >, or ≥. You choose the appropriate sign for the intent of 2.

   ```
   if ( speed ≤ the_threshold_you_found ) :
           intent = 0
   else :
           intent = 2
   ```

   This is an example of a one-rule.

   This rule should break ties the same way your original code worked, or it will fail.
   In future assignments, this piece of code must parse and be runnable.

   (continued)

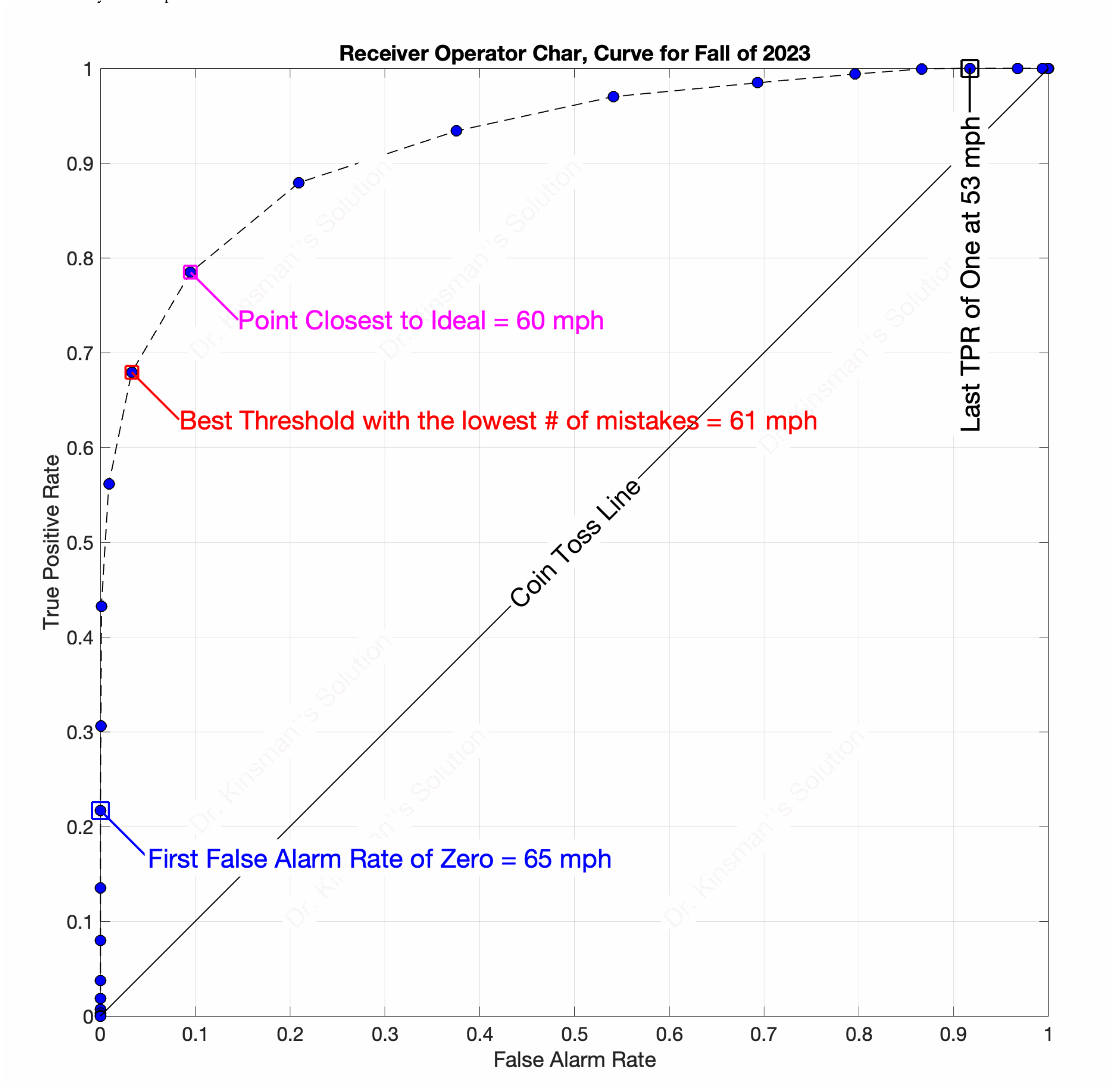
6. Create a ROC curve for your classifier choice, with dots for each threshold, connected by lines.
(Use matplotlib as an initial suggestion if using python.)
Remember recording the (FAR, and TPR) for each speed?
Plot the FAR on the X axis, and the TPR on the Y axis.

Put a magenta square around the point that is closest to a FAR of 0, and a TPR of 1.
This is the "ideal point", which is only an ideal.

Put a red square around the point on the graph with the lowest number of mistakes.

Put a green circle around the point which has the minimum mixed variance. (The clustering solution).

Here is my example from ***older*** data:



(continued)

Write-Up and Grading:

A. Create an output file called HW_NN_LastName_FirstName.docx
Make the obvious substitutions.

B. Put your name at the top, HW_NN_, and the course number.

C. <u>COPY</u> THE FOLLOWING QUESTIONS and ANSWER THEM: (2)
Copy your bar graph of the data in.
Looking at the associated graph of the data, how can we describe this data?
What kind of model is it?

D. In your PDF, show the resulting threshold classifier code.
What was your one-rule? It is just a piece of code. (2)
(Copy and paste from your output classifier file.)

E. Plot the ROC curve, as specified. (4)
Show your results in your write up.
Compare the different speeds found:
   a. the "ideal",
   b. the one with the lowest mistakes, and
   c. the one found which finds the minimum mixed variance (Otsu's method).

What do you observer *for this data*?

F. **Conclusion:** Write up what you learned here using at least three paragraphs. (2)

What did you discover? Were the results what you expected? What was surprising?

Was there anything particularly challenging? Did anything go wrong?

Provide strong evidence of learning.

Write a conclusion that describes what you learned in this homework.
Points are taken off for writing with bullet points or checkmarks.

Penalty Rubric:

We expect that you can do all the things above. Getting the code in, in one directory, with the correct name on it, is what we call "table stakes". If you fail that, you lose points for that. You start with 10 points, and your grade goes down from there.

You do not submit supporting code.	(8)
Your code is handed in with a zip file that creates more than one directory when unzipped.	(2)
Your main program must create the classifier program, or you lose 5 points out of 10. (This is the main goal of this homework.)	(5)
Your main program must run.	(1)
Your main program is commented well.	(2)

Appendix:

Some pseudo-code to get you started. It is not complete code. It *might* not even work.
This is just an outline of the process of finding the best threshold.
It does not use threads.

```
best_n_total_mistakes         = infinite;
all possible thresholds       = floor(min(speeds)) to ceiling(max(speeds))
% round the data points to the nearest bin, based on the rounding.
data = round(data/bin_size)* bin_size;

for threshold = all possible thresholds
     % Try this threshold out, to see what you would get if you used it.
     % This involves temporarily sorting the data.
       LEFT__NODE = Data points with attribute <= Threshold
       RIGHT_NODE = Data points with attribute >  Threshold  % All other data points

     % Guess that each class matches it's majority class:
     temp_left__class_guess   =  majority class of the LEFT__NODE
     temp_right_class_guess   =  majority class of the RIGHT_NODE
       % How many mistakes would these guesses make?
     n_mistakes_on_left       = number of the minority class in the LEFT__NODE
     n_mistakes_on_right      = number of the minority class in the RIGHT_NODE

     n_mistakes_for_this_threshold = n_mistakes_on_left + n_mistakes_on_right
       cost_func       = FN + FP;     ← the usual cost function
       if ( n_mistakes_for_this_threshold <= best_n_total_mistakes )
              best_n_total_mistakes         = n_mistakes_for_this_threshold;
              best_threshold_to_use         = threshold;
              best_left__class              = temp_left__class_guess;
              best_right_class              = temp_right_class_guess;


open a file and print out the rule:
open the output file, and write to it:
if ( attribute_value <= best_threshold_to_use ) :
    class_id = best_left__class ;
else :
    class_id = best_right_class ;
```