DS2500
Fall 2023
Homework 3
Assigned: March 15, 2024
Deadline: March 29, 2024 @ 9pm

We recommend using the scikit learn library (which we'll also explore in lab) for this homework.

Submit your solution on gradescope: https://www.gradescope.com/courses/679045 This homework is due at **9pm on Friday, March 29th.** If you miss the deadline, you can submit up until April 16 at 9pm -- look for the "HW3 - Late Submission" link on gradescope.

Homeworks are broken into three components:

1. **Accuracy**. You'll answer quantitative questions about the dataset on gradescope. These answers are auto-graded. Gradescope can be a little picky, so make sure you don't put extraneous characters or whitespace in your answers -- and double-check the "correct/incorrect" confirmation!
2. **Visualization**. You'll be asked to submit screenshots/downloads of at least one Python plot on every homework. We expect these plots to be labeled, easy to read and understand, and appropriate for the data.
3. **Code Quality.** You'll submit your code as well, which we will review and grade based on its modularity, readability, and reusability.

Please see the DS2500 Homework Grading Guidelines for more on grading.

You are welcome to work with classmates, but you may not share code with each other, and you may not post code on Piazza. If you find a code snippet online, you're welcome to use it, but (1) you must provide a citation, and (2) you may not ask us to debug it for you.

The 30-minute Guideline
If you get stuck on a homework problem, come by office hours or post on Ed Discussion! We recommend you spend about 30 minutes trying to figure out a problem, and then ask for help. Enough time that you can try a few things to get unstuck, but not SO much time that you're banging your head against the wall. Try for 30 minutes, then ask us. :)

For this homework, you'll build a KNN Classifier to predict how a given state would vote in a specific year based on its demographic similarity to other states.

- **Features**: demographic data for each state
- **Label**: the party each state votes for in a presidential election ("party_detailed" column)

We recommend using **scikit learn** for this homework. We'll explore some of its functionality in lab (and in class), but it'll be helpful for you to read up on the documentation. Some useful functions:

- `KNeighborsClassifier(), knn.fit(), knn.predict()`
- `KFold(), cross_validate()`
- `train_test_split()`
- `metrics.classification_report()`

High-Level Steps

- Read in, clean, organize data. Make sure your dataframe has the states in alphabetical order.
- For the year 2000[1], use cross-fold validation to choose the best value of *k* for a KNN Classifier
  - There could be many "best" values of *k* depending on whether you care about accuracy, precision, or recall
- Now that you've chosen a good *k*, set up a new classifier for 2004. This is your "real" one. Your classifier should predict the party a given state voted for in 2004.
- Run the "real" classifier and see how it does!

Data for this Homework

- 1976-2020-president.tab (right-click and choose "save link as" to download)
  - We'll spend some time breaking down and cleaning up this file in class on Tuesday, 3/19!
- demographics.csv

The first file contains all presidential election years from 1976 through 2020. For each year, it lists every state and how many votes that state cast for each candidate on the ballot. The second file contains some demographic information gathered about every state from the 2000 census[2].

Some ideas for data cleaning you'll need for this homework -- these are starting points to consider; you might need more or different things to make it all work!

- We need to know which party ("party_detailed" column) won each state in a given year; currently, it has the raw number of votes per candidate. This will be our classifier label.[3]
- Use this demographic data as features for each state:
  - Percent of population identified as male (relevant columns: `TOT_MALE, TOT_POP`)
  - Percent of population identified as female (relevant columns: `TOT_FEMALE, TOT_POP`)

[1] Keep reusability in mind! We're using years 2000 and 2004 in this homework, but make sure you're flexible enough to switch to any other year without modifying your code.
[2] Ethical concerns: Who is left out of this dataset? Notice there is a gender binary used here, and notice the limited number of racial/ethnic categories.
[3] A party named DEMOCRATIC-FARMER-LABOR appears in one state; you can replace it with DEMOCRAT

    - Percent of population identified as White Alone (relevant columns: `WA_MALE, WA_FEMALE, TOT_POP`)
    - Percent of population identified as Black (relevant columns: `Black, TOT_POP`)
    - Percent of population identified as Hispanic (relevant column: `Hispanic, TOT_POP`)
- You'll probably want to merge the two datasets into one dataframe. State names are what they have in common, but be careful with upper/lowercase, and make sure you put them in alphabetical order before creating your model.

Choose a Value of *k* (Year: 2000)

To be sure we all get the same answers for the auto-graded questions, please make sure that you...

- Set the optional parameter **random_state** to **0** in `train_test_split()`
- Set the optional parameter **random_state** to **0** and **shuffle** to **True** in `KFold()`
- Put your merged dataframe in alphabetical order by state

For the first four questions below, your job is to find the best value of *k* (number of neighbors) for a given year, depending on whether we care most about **accuracy**, **precision**, or **recall**.

How will you find the best *k*? By setting *k = 4, 5, 6, ..., 10* and using Cross Fold Validation to evaluate how good that value of *k* is. Set the number of splits to 5 (using the **n_splits** parameter). Scikit-learn will split the data into 5 groups and use four of them as training and one of them as testing. Each group is used as the testing group once, and scikit-learn can compute the mean accuracy, precision, and recall (among other things) across all 5 folds.

Create the "Real" Classifier (Year: 2004)

Now that you've found good *k* values, set a up a new classifier, but using data from 2004 instead of 2000.

Part 1 - Questions about the Data

This part of your solution will be auto-graded. When you find this assignment on gradescope, the first part will ask you the following questions; type or select your answers. You must compute the answers to these questions programmatically. Gradescope will confirm your answers are correct/incorrect.

Check the output! Gradescope can be a little picky about formatting, and we don't want you to lose points for putting extra characters or whitespace in an answer. Make sure you've got the correct answer to each question for full credit.

Answer these questions (make sure you compute these answers in your Python solution):

1. For year 2000: What is the optimal value of *k* if we care most about recall?
2. For year 2000: What is the lowest mean recall for any value of *k*?
3. For year 2000: What is the optimal value of *k* if we care most about overall precision[4]?

[4] "Overall" here means, not focused on one specific label/class. Look into scikit learn's function for metrics and scoring, which is built for cross-validation.

4. For year 2000: What is the lowest mean precision for any value of *k*?
5. Switch the year to 2004, using only data from that year. Create a new classifier, and set the value of *k* to the optimal value if we care most about precision. Then...
    a. What is the f1 score for just one label -- the states that voted republican?
    b. How many states did your model predict to vote Republican in 2004, and in fact did?
    c. Does your model correctly predict how Ohio voted in 2004?

Part 2 - Visualization

Create two Python plots and upload them as screenshots/downloads.

- **Plot #1**: A heatmap showing the confusion matrix when the value of *k* is optimal if we care most about recall, for the year 2004.

- **Plot #2**: A plot showing why you picked those optimal values of *k* for the year 2000.

Part 3 - Code Quality

Submit on gradescope the code you developed to compute your answers to the Part 1 questions, and to generate the plots for Part 2.

Your code will be graded on modularity, readability, and reusability.