Project 1

For this project you'll have a partner.

Groups (13)	Group members
Group A	Rob Berini (rmberin2@ncsu.edu), Nellie Leddy (heleddy@ncsu.edu)
Group B	David Grant (dgrant@ncsu.edu), Eric Warren (erwarren@ncsu.edu)
Group C	Kevin Lewis (kplewis@ncsu.edu), Yixuan Yang (yyang55@ncsu.edu)
Group D	Ary Bautista Hernandez (jbautis@ncsu.edu), Brian Higginbotham (bhiggin@ncsu.edu)
Group E	Ethan Marburger (elmarbur@ncsu.edu), Emily Winters (ewinter@ncsu.edu)
Group F	Chris Goodwin (cdgoodw3@ncsu.edu), Paige O'Connell (psoconne@ncsu.edu)
Group G	Mary Brown (mbrown29@ncsu.edu), Kyra Kapsaskis (kekapsas@ncsu.edu)
Group H	Supro Debnath (sdebnat@ncsu.edu), Angelice Floyd (arfloyd2@ncsu.edu)
Group I	Alex Figura (asfigura@ncsu.edu), Austin Semmel (adsemmel@ncsu.edu)
Group J	Arti Gupta (agupta67@ncsu.edu), Saurabh Gupta (sgupta52@ncsu.edu)
Group K	Upendra Joshi (ujoshi@ncsu.edu), Smit Miyani (smiyani@ncsu.edu)
Group L	Kejun Qian (kqian2@ncsu.edu), Tyler White (twhite7@ncsu.edu)
Group M	Jacob Neff (jneff@ncsu.edu), Xi Yang (xyang46@ncsu.edu)

You should not discuss the project with anyone outside your group. If you are stuck please contact Dr. Post or the TA. If you run into issues with your partner, let me know immediately. I can't help if it is the day before the due date and you tell me you haven't heard from them. You'll be stuck doing it yourself if you want to obtain full credit! Create a plan with dates for parts to be completed with time to spare.

Overall Goal

Our goal is to write functions that will manipulate and process data sets that come in a certain form. You'll then create fit some basic linear regression models and implement a cross-validation algorithm to judge the models.

This project is meant to assess your ability to program in python. You'll write up your project in a google colab notebook. A link to the notebook should be submitted in the assignment link - please update your sharing settings so we can view the file. The result should be a report with a narrative throughout, section headings, graphs outputted in appropriate places, etc. To be clear **be sure to include markdown text describing what you are doing and your thought process (not just the question prompts), even when not explicitly asked for!** The audience you are writing for is someone that understands programming and very basic statistics, but would need you to provide details and explanation of what you are doing to understand it.

Both partners should submit think link.

Data

We'll use a number of `.csv` files that contain information from the census bureau surveys. This data is a little older (2010 is the newest data). Our first goal will be to read in one of these `.csv` files and parse the data using functions we've written. Then we'll combine some parsed data and deal with it from there.

Part 1: Data Processing

First Steps

First I'll just explain what you'll do to parse the data you read in. Then I'll give you requirements on how to parse it afterward. (I find it easier to first do all of the things below without writing functions first. Then convert your code into the required functions.) You only need to include your final work/functions with narrative in your submitted document.

Read in the data set available here: https://www4.stat.ncsu.edu/~online/datasets/EDU01a.csv

Now the format of the data is kind of weird. There is a column for `Area_name` (US, NC, or a county), a column called `STCOU`, and then four columns corresponding to each question on the survey. This survey was about school enrollment. Let's process this data!

1. Select only the following columns:
    - `Area_name`
    - `STCOU`
    - Any column that ends in "D"
    - To do the above, use the `.loc[]` method. Note that the `.str.endswith()` method can be used on the column names for the columns that end with "D".
2. The data you'll have is in **wide** format. That is, there are multiple observations in a given row (each column that ends in "D" corresponds to a particular year's measurement). We want to convert the data into **long** format where each row has only one enrollment value for that `Area_name`. Do this operation using the `pd.melt()` function (info here).
3. One of the new columns should now correspond to the old column names that ended with a "D". (All columns in these census data files will have this similar format - For more about the variables see the data information sheet)
   The first three characters of the former column names represent the survey and the next four characters represent the type of value you have from that survey. The last two characters prior to the "D" represent the year of the measurement. For this part:
    - Create a loop that loops over the rows of the data frame
    - At each iteration,
        - parse the string from your new column in order to pull out the year and convert it into a **numeric** value such as `1997` or `2002`. Add this new `year` variable to your data frame. Note: the data set above only has data from the 1900's but the next data set we read in has data from the 2000's. Handle that appropriately!
        - grab the first three characters and following four digits to create a new variable representing which measurement was grabbed. Add this new `measurement` variable to your data frame as well.
    - Drop the original column name variable.
4. Split the data frame into two data frames:
    - one data frame should contain only non-county data

- the other should contain only county level data

Note that all county measurements have the format "County Name, DD" where "DD" represents the state. Use the `.apply()` method with a `lambda` function to create an indexing vector you use to do the subsetting. `np.logical_not()` comes in handy!

5. For the county level data frame, create a new variable that describes which state one of these county measurements corresponds to (the two digit abbreviation is fine!).
6. For the non-county level data frame, create a new variable called `division` corresponding to the state's classification of division here (the Census Bureau-designated regions and divisions). If the row corresponds to a non-state (i.e. `UNITED STATES`), return `ERROR` for the division. The code for this part will not be a ton of fun to write! Write a function with basic `if/elif` for a single value of `Area_name`. Then, use `np.vectorize()` to make it work for a full vector of values.

Requirements

Now we want to repeat the above process for the 2nd component of the data set. This is available at the link below.

- https://www4.stat.ncsu.edu/~online/datasets/EDU01b.csv

Rather than copying and pasting a bunch of stuff and changing things here and there, we want to write functions that do the above pieces and one function that we can call to do it all!

- Write one function that takes care of steps 1 & 2 above. Give an optional argument (that is it has a default value) that allows the user to specify the name of the column representing the value ('enrollment' for these two data sets).
- Write another function that takes in the output of step 2 and does step 3 above.
- Write a function to do step 5
- Write a function to do step 6
- Write another function that takes in the output from step 3 and creates the two data frames in step 4, calls the above two functions (to perform steps 5 and 6), and returns two final data frames

Now last thing, put it all into one function call! This is called creating a wrapper function. Create a function that takes in the URL of a `.csv` file in this format and the optional argument for the variable name, calls the functions you wrote above, and then returns the two data frames in a list.

Call It and Combine Your Data

Call the function you made two times to read in and parse the two `.csv` files mentioned so far. Be sure to call the new `value` column the same in both function calls.

Now we want to join the two county level data sets and the two state level data sets. Write a function that takes in unlimited positional arguments. When you call the function these arguments will be the results of calls to your wrapper function (so each argument will be a list with the two data sets in it).

- Within the function itself, use `map()` and a `lambda` function to obtain just the county level data for every argument. Then use the `reduce()` function from the `functools` module with a `lambda` function that calls `pd.concat()`.
- Repeat for the non county level data.
- Put the two combined data sets into a list and return it (so it will have the same format as the inputs!

Call this function to combine the two data objects into one object (that has two data frames: the combined county level data and the combined non-county level data).

Note: Here is what the combined data should look like.

```
## [            Area_name  STCOU  enrollment    year measurement state
## 2         Autauga, AL   1001        6829  1987.0     EDU0101    AL
## 3         Baldwin, AL   1003       16417  1987.0     EDU0101    AL
## 4         Barbour, AL   1005        5071  1987.0     EDU0101    AL
## 5            Bibb, AL   1007        3557  1987.0     EDU0101    AL
## 6          Blount, AL   1009        7319  1987.0     EDU0101    AL
## ...               ...    ...         ...     ...         ...   ...
## 31975  Sweetwater, WY  56037        6964  2006.0     EDU0152    WY
## 31976       Teton, WY  56039        2264  2006.0     EDU0152    WY
## 31977       Uinta, WY  56041        4298  2006.0     EDU0152    WY
## 31978    Washakie, WY  56043        1410  2006.0     EDU0152    WY
## 31979      Weston, WY  56045        1076  2006.0     EDU0152    WY
##
## [62900 rows x 6 columns],            Area_name  STCOU  enrollment    year measurement    division
## 0      UNITED STATES      0    40024299  1987.0     EDU0101       ERROR
## 1             ALABAMA   1000      733735  1987.0     EDU0101  Division 6
## 69             ALASKA   2000      102872  1987.0     EDU0101  Division 9
## 99            ARIZONA   4000      609411  1987.0     EDU0101  Division 8
## 115          ARKANSAS   5000      429260  1987.0     EDU0101  Division 7
## ...               ...    ...         ...     ...         ...         ...
## 31650        VIRGINIA  51000     1220440  2006.0     EDU0152  Division 5
## 31787      WASHINGTON  53000     1026774  2006.0     EDU0152  Division 9
## 31827   WEST VIRGINIA  54000      281938  2006.0     EDU0152  Division 5
## 31883       WISCONSIN  55000      876700  2006.0     EDU0152  Division 3
## 31956         WYOMING  56000       85193  2006.0     EDU0152  Division 8
##
## [1060 rows x 6 columns]]
```

Double Check It Generalizes!

Read in another similar data set and apply our functions!

- Run your data processing and combination functions on the four data sets at URLs given below:
  - https://www4.stat.ncsu.edu/~online/datasets/PST01a.csv
  - https://www4.stat.ncsu.edu/~online/datasets/PST01b.csv
  - https://www4.stat.ncsu.edu/~online/datasets/PST01c.csv
  - https://www4.stat.ncsu.edu/~online/datasets/PST01d.csv

```
## [            Area_name  STCOU  default    year measurement state
## 2         Autauga, AL   1001    25508  1971.0     PST0151    AL
## 3         Baldwin, AL   1003    60141  1971.0     PST0151    AL
## 4         Barbour, AL   1005    23092  1971.0     PST0151    AL
## 5            Bibb, AL   1007    13919  1971.0     PST0151    AL
## 6          Blount, AL   1009    27817  1971.0     PST0151    AL
## ...               ...    ...      ...     ...         ...   ...
## 31975  Sweetwater, WY  56037    41226  2009.0     PST0452    WY
## 31976       Teton, WY  56039    20710  2009.0     PST0452    WY
## 31977       Uinta, WY  56041    20927  2009.0     PST0452    WY
```

```
## 31978    Washakie, WY  56043     7911  2009.0     PST0452    WY
## 31979      Weston, WY  56045     7009  2009.0     PST0452    WY
##
## [125800 rows x 6 columns],            Area_name  STCOU    default    year measurement    division
## 0      UNITED STATES      0  206827028  1971.0     PST0151       ERROR
## 1            ALABAMA   1000    3497452  1971.0     PST0151  Division 6
## 69            ALASKA   2000     316494  1971.0     PST0151  Division 9
## 99           ARIZONA   4000    1896108  1971.0     PST0151  Division 8
## 115         ARKANSAS   5000    1972028  1971.0     PST0151  Division 7
## ...              ...    ...        ...     ...         ...         ...
## 31650       VIRGINIA  51000    7882590  2009.0     PST0452  Division 5
## 31787     WASHINGTON  53000    6664195  2009.0     PST0452  Division 9
## 31827  WEST VIRGINIA  54000    1819777  2009.0     PST0452  Division 5
## 31883      WISCONSIN  55000    5654774  2009.0     PST0452  Division 3
## 31956        WYOMING  56000     544270  2009.0     PST0452  Division 8
##
## [2120 rows x 6 columns]]
```

Cross-Validation

For the last part of the project we'll fit two linear regression models and judge them using cross-validation (no training/test split, we'll just use CV). However, we won't be able to use standard cross-validation. We'll write our own function to do it!

Subset of Data

Use the `enrollment` data sets from earlier.

- Subset the list object so that we're only looking at the non county level data.
- Remove any rows where the division variable is `ERROR` and select only the `year`, `division`, and `enrollment` variables (or whatever you called the last one!).

Two Models Under Consideration

Well use two competing models:

- A SLR model using `year` to predict `enrollment`
- An MLR model using `year` and `division` to predict `enrollment`

You'll need to create dummy variables for the `division` variable as we did in the notes and add them to the data frame. When adding these columns to the data frame, you shouldn't keep all 9 variables (recall the last column is redundant given all the others). Be sure to leave one of the indicator columns off!

If you'd like, feel free to fit the two models to the data here (this doesn't need to go in the final report).

Cross-Validation

We want to see how well these two competing models do at predicting. However, we can't use the usual cross-validation because our data is over time (`year`).

Instead, what we want to do is train the model/judge it sequentially.

1. Use the first three years of data to fit the model. Use that model to predict the fourth year. Calculate the MSE for those predictions.
2. Use the first four years of data to fit the model. Use that model to predict the fifth year. Calculate the MSE for those predictions.
3. Repeat until you predict for the last year.
4. Sum up the MSE values to get an overall MSE for the model!

Write a function to do the above given a particular `X`, `y`, and starting `year`.

Some guidelines and helpful hints:

- First write a function to get the MSE for one step of the above only (don't worry about combining things yet).
  - Have this function take in a data frame of predictors `X` (this will be used in the `.fit()` method of a `LinearRegression()` object), a 1D response `y`, and a `last_year` argument.
  - Use the `last_year` argument to subset the data into a training `X` and `y` and a testing `X` and `y`. Have your training set include all years up to and including `last_year` and your test set just include the year `last_year` + 1.
  - Do the model fitting and predictions. Return the mean squared error
- That will act as a helper function for our function that find the CV error.
- Now write a function to obtain the CV value over all the years (other than the initial training block)
  - Have this function take in `X`, `y` (both as above), and a `first_year` argument.
  - If the `first_year` is less than 1989, have the function raise an error and return a message
  - Initialize an MSE value at 0
  - If the not, use a loop from `first_year` to the last year in the data set (find that value programmatically)
    - Within the loop, use your previous function and augmented assignment to add the MSE for the given year
  - Return the MSE

Now run your function using the SLR model. Repeat using the MLR model. Discuss the MSE values you see.

That's everything. Check the rubric on the next page to make sure you are comfortable with your submission. Good luck! Have fun!

Rubric for Grading (total = 100 points)

Item	Points	Notes
Introduction to purpose of report	5	Worth either 0, 2, or 5
Data Processing Functions	45	Worth either 0, 5, 10, . . . , 45
Combining Data Function	15	Worth either 0, 3, 5, . . . , 15
Cross Validation Functions	35	Worth either 0, 3, 5, . . . , 35

Notes on grading:

- For each item in the rubric, your grade will be lowered one level for each each error (syntax, logical, or other) in the code and for each required item that is missing or lacking a description.
- **You should use Good Programming Practices when coding (see wolfware). If you do not follow GPP you can lose up to 30 points on the project.**

The reports should include a narrative throughout, section headings, graphs outputted in appropriate places, etc. To be clear **be sure to include markdown text describing what you are doing, even when not explicitly asked for!** Points will be deducted from appropriate sections as appropriate.