Intermediate Python Programming

LAB 8

Description: Data Analysis Using Pandas and MatPlotLib

This lab will focus on some of the key features of Pandas, one of the most popular data analysis libraries for Python.

Instructions

- Please follow all instructions closely and read this in its entirety before writing ANY code. There is no penalty for going "above and beyond" the assignment requirements. However, failure to meet all required parts can affect your grade. Please consult the rubric for each section.
- For this assignment you can create a single "raw" Python file, or Jupyter Notebook file, or multiple files, one for each part. Just ensure that if you use multiple files that both are named appropriately.
    - Note: For best result when using matplotlib with Jupyter Notebook, add the following line just below your imports section:

      ```
      %matplotlib inline
      ```
- You will need to also download the "all_alpha_19.csv" file provided on the assignment page and place it in the same folder as your Python script/notebook file.
    - Data from: https://www.fueleconomy.gov/
- When working with DataFrames we try to avoid "inplace" changes to the data:
    - `df.SomeFunction(inplace=True)`
- Instead, use function chaining. It is ok to either reuse the same DataFrame variable or a new one:
    - `df = df.SomeFunction()`
    - `new_df = df.SomeFunction()`
- It is also a good idea to output the head of the DataFrame after each operation so you can verify that your code, for that step, functioned as expected.

Part I – Preparing the DataFrame (ETL)

This first part of the assignment will focus on "ETL", which stands for Extract, Transform and Load. This is a key part of a data analyst's job.

- Ensure the "all_alpha_19.csv" file is in the same folder as your Python script.
- Use the following setup code at the top of your script:

  ```
  import numpy as np
  import pandas as pd
  import matplotlib.pyplot as plt
  %matplotlib inline  # For Jupyter Notebook only
  ```
- Next, use pd.read_csv() to load the file into a DataFrame called "df".
    - The only argument you need is "header" which should be set to 'infer'.
- We only want records that were generated using a specific fuel efficiency standard (Federal Tier 3 Bin 125) for gasoline and diesel engines. Use either column filters or a query to filter the results to include only rows where:

  - 'Stnd' is equal to 'T3B125'
    - AND
  - 'Fuel' is either 'Gasoline' or 'Diesel'
- Now, create a new DataFrame from the filtered data, but only include the following columns:
  - 'Model' ← Car manufacturer and model
  - 'Displ' ← Engine displacement (size of engine)
  - 'Fuel' ← Type of fuel
  - 'City MPG' ← Number of miles the car gets per gallon of fuel in the city
  - 'Hwy MPG' ← Number of miles the car gets per gallon of fuel on the highway
  - 'Cmb MPG'← Combined number of miles the car gets per gallon of fuel in the city and highway
  - 'Greenhouse Gas Score' ← Calculated score indicating the car's efficiency (higher is better)
    - Note: one of the easiest ways to create a new DataFrame using only some of the columns from another is to:
      - Create a list of strings (called cols) where each value corresponds to a column name you wish to migrate
      - Create the new DataFrame using the following syntax:

        ```
        new_df = old_df[cols]
        ```

      - You will also want to reset the index to avoid an extra un-named column in the new frame that equals the index from the old one. You can chain this to the above call with:

        ```
        .reset_index(drop=True)
        ```

- Use the astype() function of the DataFrame to convert the three MPG columns to float.
- Next, because only a small part of the world uses miles and gallons, we are going to add three new columns ('CityKML', 'HwyKML' and 'CmbKML') to store the fuel efficiency in kilometers per liter. To do this we need a conversion function called mpg_to_kml:

  ```
  def mpg_to_kml(mpg):
  ```

  - This method will multiply the incoming value by 0.42514 and return the result
- Now, using the assign() method of the DataFrame, add each of those new columns to the frame.
  - The assign() method allows you to use a lambda or a named function to calculate the value of the new column's cells. It will automatically apply the given function to each row of the frame so you don't need to write a loop.
- Finally, save the DataFrame to a CSV file called "car_data.csv".
  - Use your favorite CSV editor to verify the contents (Excel, Notepad++, etc.)

Scoring:

Criteria	**Points**
Load the "all_alpha_19.csv" file	5
Filter the results	15
Migrate to new DataFrame with smaller column list	10
Convert MPG columns to float	5
Create new KML columns	20
Save DataFrame to "car_data.csv"	5
TOTAL	**60**

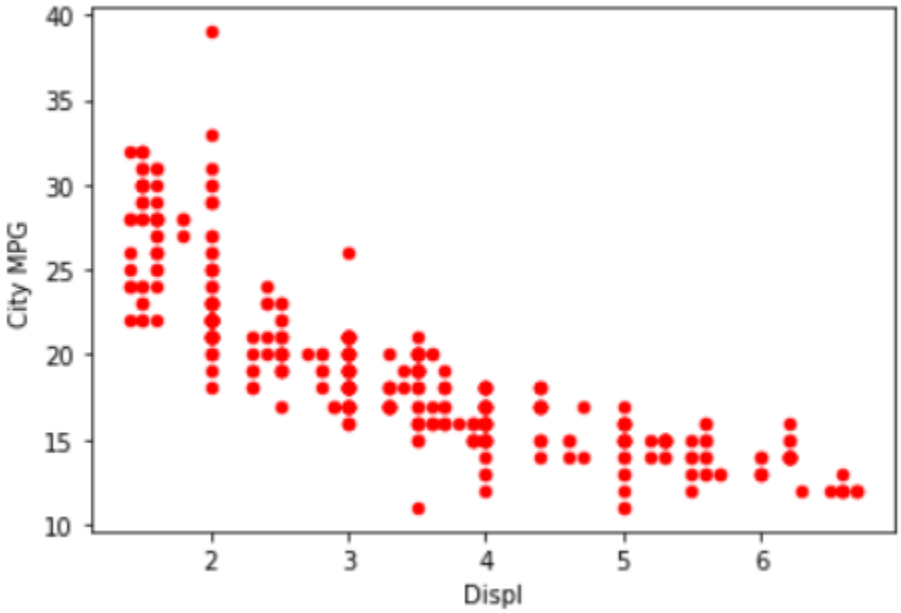
Part II – Visualize the Data

Now you are going to use the CSV file saved in part I to create some graphs using matplotlib.

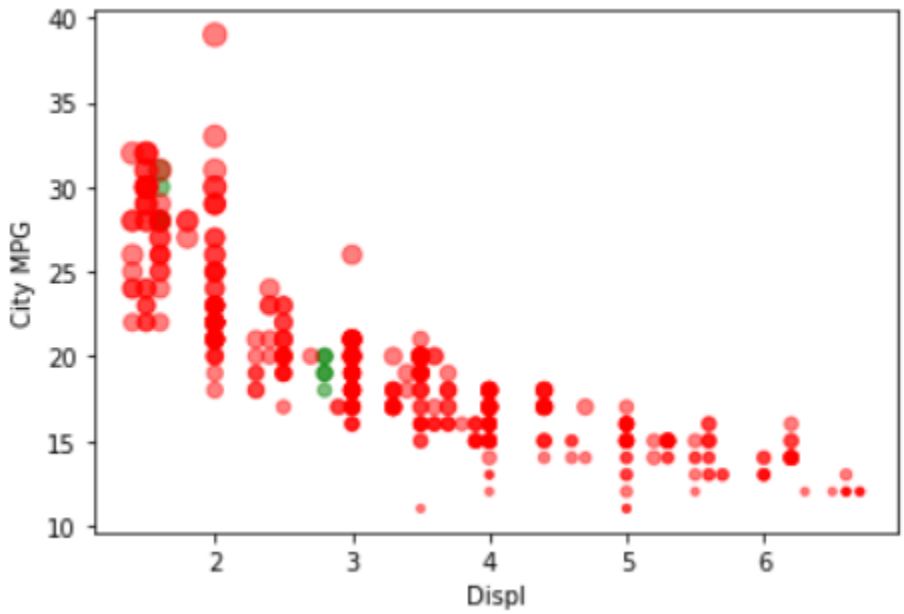
- Use pd.read_csv to load the file into a DataFrame called "df".
  - The header argument should be 'infer' and the index_col argument should be 0.
- Now, use the DataFrame's plot() function to create a scatter plot where 'Displ' is along the x-axis and either 'City MPG' or 'CityKML' is along the y-axis:
  - kind: 'scatter'
  - x: 'Displ'
  - y: 'City MPG' or 'CityKML'
  - color: 'r'
- After calling plot(), be sure to show the chart using:

```
plt.show()
```

- Your graph should look something like:



- Now, we are going to make the graph more compelling.
- Create a new Series/List called 'c' that will either have an 'r' or 'g' string for each element, which is a color corresponding to each vehicle's 'Fuel' type.
  - If the vehicle uses 'Gasoline' the color should be 'r'. If it is 'Diesel' then the color is 'g'.
  - This list should be the exact same length as the DataFrame's record count.
- Create another Series/List called 's' that will correspond to each vehicle's 'Greenhouse Gas Score'.
  - The value should be an integer
  - To make the size more apparent in the chart, multiply each value by at least 3. I like 8 to 10.
- Now, replot the dataframe, this time using the 'c' list for the 'color' argument and 's' for the 's' argument.
  - Also, it helps to make set alpha to 0.5
- Don't forget to re-show the chart!



Extra Credit

- Create a legend and give the axes labels more appropriate names.
- Come up with another compelling chart using the given data.
  - An idea would be to plot the city, highway and combined MPG/KML values using separate colors to prove that city is almost always worse than highway efficiency.
  - Use something other than scatter!

Scoring:

Criteria	**Points**
Load the "car_data.csv" file	5
First plot	10
Prepare color/size lists for second plot	15
Second plot	10
TOTAL	**40**
Extra Credit	
Legend and axes labels	3
New chart	7