Mehtab Randhawa

ECS171

Homework #1:

1. To help split the cars into the even amount of 4 categories ('low','medium', 'high', 'very high'). We apply this categorization by taking the quartile distribution of the cars with **qcut.**
    - Command in terminal to run script: **python3 problem1.py**
        - **Results will be printed to screen with 'mile_category' column as seen bellow**
    - Result:

```
Mehtabs-MacBook-Pro:HW_1 MehtabRandhawa$ python3 problem1.py
      mpg  cylinders  displacement horsepower  weight  acceleration  model_year  origin                  car_name mile_category
0    18.0          8         307.0      130.0  3504.0          12.0          70       1  chevrolet chevelle malibu        medium
1    15.0          8         350.0      165.0  3693.0          11.5          70       1          buick skylark 320           low
2    18.0          8         318.0      150.0  3436.0          11.0          70       1         plymouth satellite        medium
3    16.0          8         304.0      150.0  3433.0          12.0          70       1              amc rebel sst           low
4    17.0          8         302.0      140.0  3449.0          10.5          70       1                ford torino           low
..    ...        ...           ...        ...     ...           ...         ...     ...                        ...           ...
387  27.0          4         140.0       86.00  2790.0          15.6          82       1            ford mustang gl          high
388  44.0          4          97.0       52.00  2130.0          24.6          82       2                  vw pickup     very high
389  32.0          4         135.0       84.00  2295.0          11.6          82       1              dodge rampage     very high
390  28.0          4         120.0       79.00  2625.0          18.6          82       1                ford ranger          high
391  31.0          4         119.0       82.00  2720.0          19.4          82       1                 chevy s-10     very high

[392 rows x 10 columns]
```

2. Out of all the plots in the 2D scatterplot matrix, pair-wise features I found from the pair plot to be the most informative about the mile category was: **weight x model_year** as you can see a clear distinction between the categories. You also notice a trend as you increase through the model years the mpg was also increasing from low to high. Vice versa we can also see from the pair-wise plot that as weight increase the mpg receive would go from a range of high to low. Command in terminal to run script: **python3 problem1.py**
    - Command in terminal to run script: **python3 problem2.py**

- Output will be a 2D scatter plot saved in the directory as **problem2_plot.png** & will plot like the one below:

- This graph was created using **sns.pairplot**

3.Currently mpg is being calculated with **single_feature_solver**(order, feature). Where order is the order to which we want our mpg (dtype: Int) regression, and feature we want to utilize to calculate our expected y value (dtype: Str).

- To run this code simply type: **python3 problem3.py**
  - By default, I have set order = 2 & feature = 'displacement'. If you prefer to change either of those parameters you can by simply. Just change the parameters at line 90 & 91.
- Should get expected output to look something similar to below:

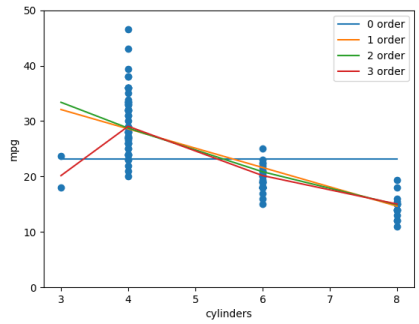
```
Mehtabs-MacBook-Pro:HW_1 MehtabRandhawa$ python3 problem3.py
Calculated MPG for displacement  at row order 2
      mpg    new_mpg
0    18.0  15.594315
1    15.0  14.426097
2    18.0  15.236148
3    16.0  15.699075
4    17.0  15.770601
..    ...        ...
387  27.0  26.041537
388  44.0  30.253326
389  32.0  26.499260
390  28.0  27.922987
391  31.0  28.020598

[392 rows x 2 columns]
```

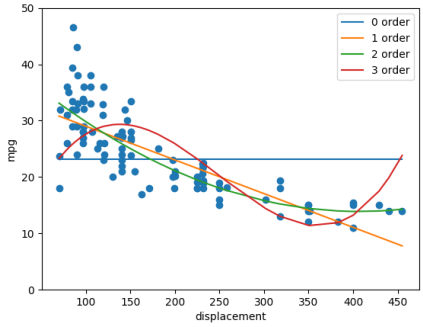
- Better description of implementation of code can be found inside problem3.py script

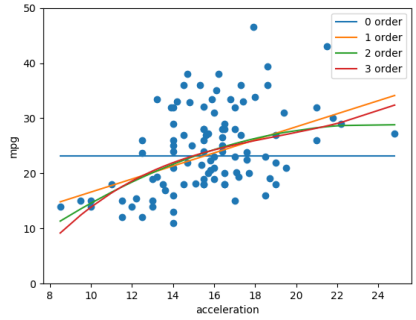
4. For problem 4 we implemented a Mean Square Error with **data_mse(data)** function, data here is shuffled data. Inside the function we split the data into two sets and save the results into a data frame that can be seen below. Furthermore, we also create the plot with function **plt_data(data)** which simply just plots all 7 graphs that can be seen on the next page.
   - To run this script simply type: **python3 problem4.py**
   - Output should look like following:

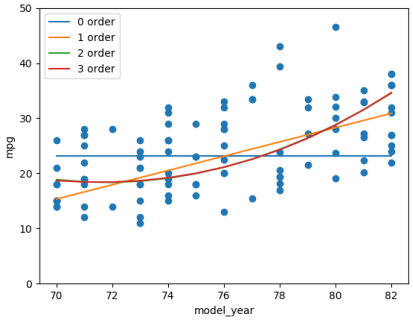
```
Mehtabs-MacBook-Pro:HW_1 MehtabRandhawa$ python3 problem4.py

 Our Mean Score's calculated to the third order for both training and test:
    cylinders           displacement           horsepower              weight           acceleration           model_year              origin
        train      test       train      test      train      test      train      test       train      test      train      test      train      test
0th  61.563104  58.913496   61.563104  58.913496  61.563104  58.913496  61.563104  58.913496   61.563104  58.913496  61.563104  58.913496  61.563104  58.913496
1st  24.403413  22.962689   21.446717  21.189275  23.047221  26.569659  18.137488  20.266918   50.935968  47.080575  39.882540  42.187114  42.232232  39.314117
2nd  24.247534  22.735091   19.096114  18.391954  18.037083  21.765976  35.729230  39.085223   50.245546  45.607361  36.464005  46.019384  41.467290  38.507706
3rd  22.836093  18.913042   32.950224  34.093762  26.795203  29.935852  87.637499  96.942679   50.059615  45.878871  36.507268  46.020390  41.467290  38.507706
plotting problem4cylinders.png
plotting problem4displacement.png
plotting problem4horsepower.png
plotting problem4weight.png
plotting problem4acceleration.png
plotting problem4model_year.png
plotting problem4origin.png
Mehtabs-MacBook-Pro:HW_1 MehtabRandhawa$
```

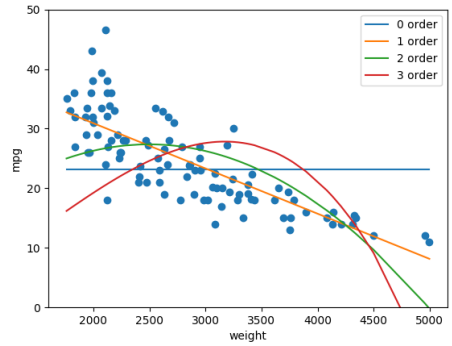
- All 7 plots can be found in the same directory after running the script under their appropriate name.

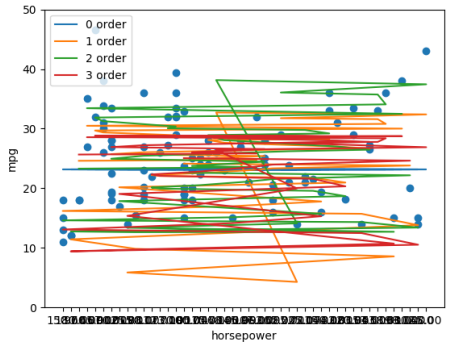












- The plots and mean square error help represent that the second order determines some of the best linear regression models as it has low MSE and relatively fit line according to data.
- Displacement helps accurately represent a cars mpg consumption as it has the lowest MSE as well as one of best fitting line's out of the 7 plots.

5. We calculated the polynomial to the second order with **polynomial_feature_solver(data,order) &** utilized the same testing/training data from the previous script (problem4.py) by importing them into the file.
    - Run **`python3 problem5.py`** `& get following output`

```
Calculed MSE for problem 5:
     Train Polynomial Regression  Test Polynomial Regression
0th                     59.465573                    64.566380
1st                     10.238076                    13.115485
2nd                      7.630078                    10.706343
3rd                      1.000000                     1.000000
Mehtabs-MacBook-Pro:HW_1 MehtabRandhawa$
```

6. **.score()** was used to help calculate the precession for Logistic Regression utilizing the 'lbfgs' solver.
   - **Run: python3 problem6.py**
   - The output returned is as follows:

```
Mehtabs-MacBook-Pro:HW_1 MehtabRandhawa$ python3 problem6.py
The precesion calculated for problem 6 LogisticRegression is:
0.7721518987341772
Mehtabs-MacBook-Pro:HW_1 MehtabRandhawa$
```

7. I implemented min-max normalization with **MinMaxScaler()** and was able to calculate the following precession. Which improved the accuracy of the data by a small amount as the output below shows.

```
The precesion calculated for problem 7 LogisticRegression with min-max normilization is:
0.810126582278481
Mehtabs-MacBook-Pro:HW_1 MehtabRandhawa$
```

   - Run: **python3 problem7.py**

8. For the following problem our second-order, multi-variate polynomial regression expects mpg of **20.446498814732156 mpg** & our logistic regression model predicts our category to be **'low'**.
   - Run: **python3 problem8.py**

```
Problem 8:
second-order, multi-variate polynomial regression expects mpg:
20.446498814732156


The Category predicted is
low
```