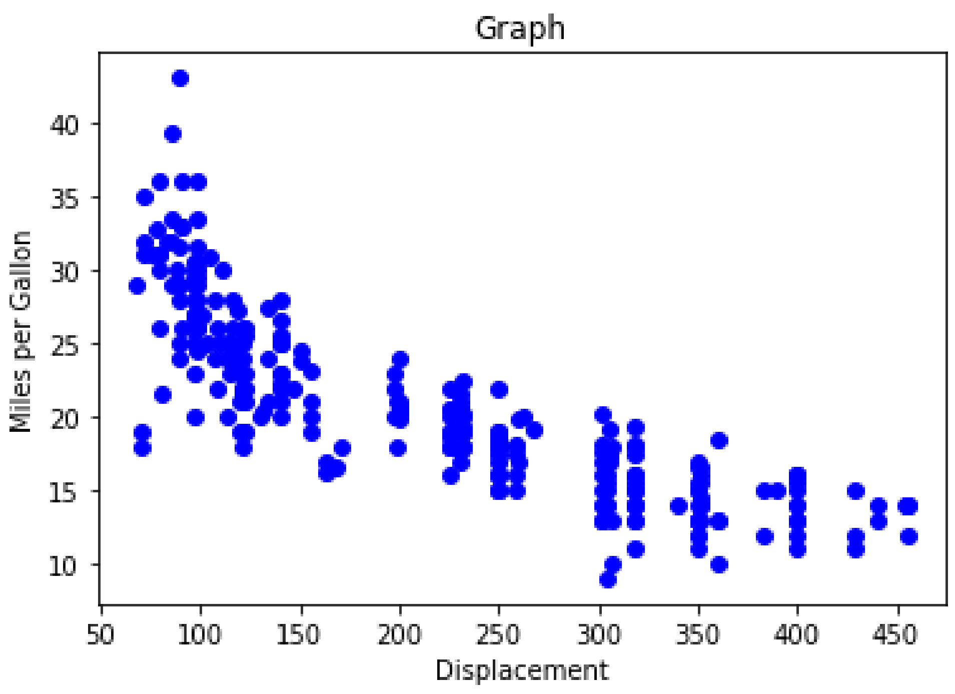
Machine Learning Assignment-2
Part 2

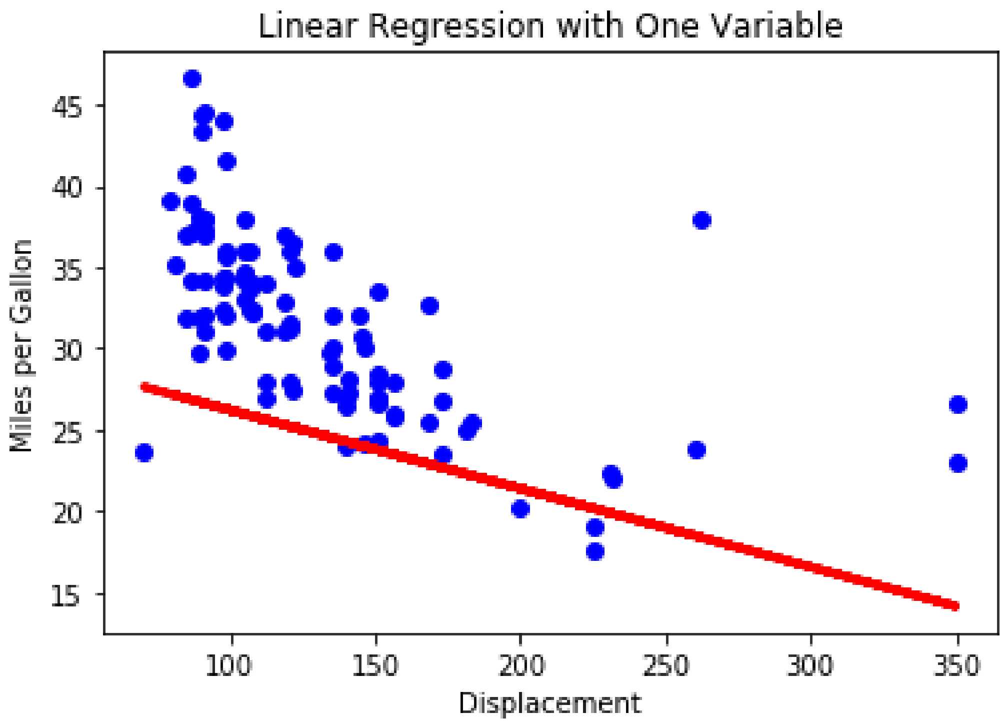
Ilyas Habeeb - mih278
Faiyaz Sundrani- fs1459

1. **For now, ignore the horsepower attribute. Plot the data in the training set (with displacement on the horizontal axis, and mpg on the vertical axis).**



2. **Using simple linear regression, train a linear function to predict mpg based on displacement only, using the training data. Plot the learned line, and report the training and testing error. Use the squared error function Err = ^{1} ((g(x^t) – r^t)^2). 2t Package allowed.**

```
Answer: Training Error: 1557.33
         Testing Error: 3565.76
```



3. **Do polynomial regression: train polynomials of degree 2 and 4 and 6 to predict mpg based on displacement only, using the training data. Plot the 3 resulting curves and report the training and test errors in both cases (squared error Err). Do you see evidence of overfitting? Package allowed.**

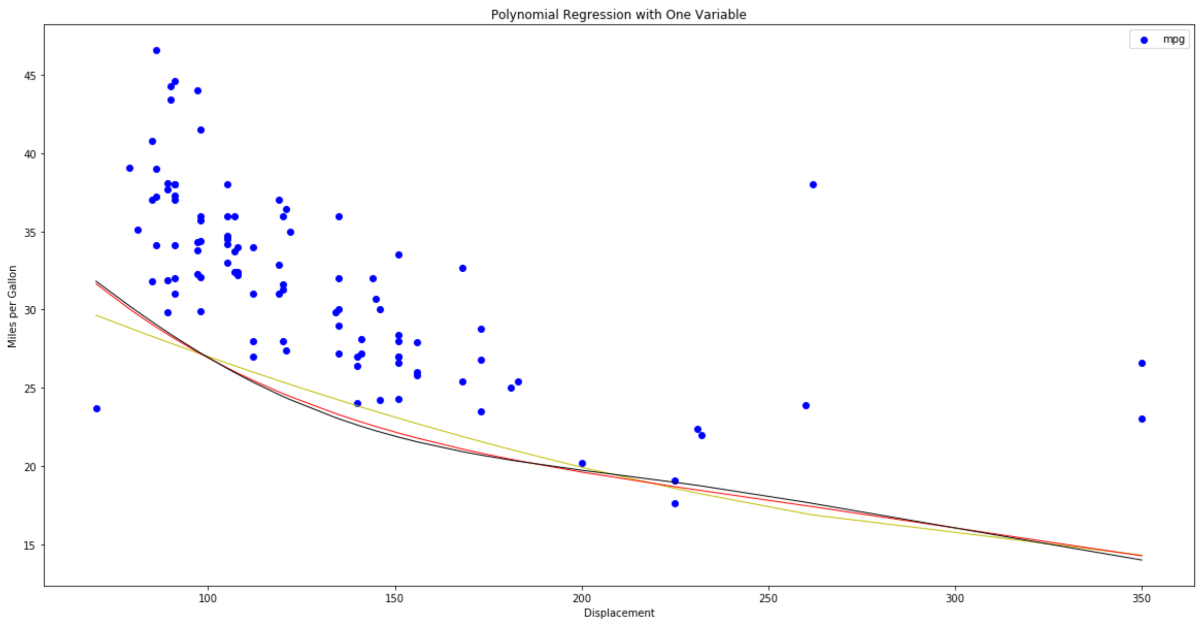
Answer: Training Error for degree 2 is: 1296.01058799
Testing Error for degree 2 is: 3282.09475116
Training Error for degree 4 is: 1242.23889296
Testing Error for degree 4 is: 3411.64000136
Training Error for degree 6 is: 1219.71902014
Testing Error for degree 6 is: 3445.12492549

An Important Point to mention: We implemented the above using scikit-learn's PolynomialFeatures and fit_transform. If we implement it using numpy.polyfit, we get a different value for training error and testing error for degree 6. We get:

When using numpy.polyfit:
Training Error for degree 6 is: 1200.7376
Testing Error for degree 6 is: 3421.2489

This happens because the floating point precision in scikit-learn and numpy are different. The numpy answer is more precise when compared to scikit-learn. Hence, we get different value for k = 6.

Yes, we can see evidence of overfitting. As the degree of polynomial increases, the training error keeps decreasing whereas the testing error keeps increasing. Thus, as degree of polynomial increases, it performs better and better in training dataset but worse in test data set.



4. **Do multiple linear regression: Train a linear function on the training set, using both input attributes, horsepower and displacement. Report the squared error Err on the test set only. Package allowed.**

Answer: Testing Error: 3443.39

5. **Implement k Nearest Neighbour from scratch (do not use a package). Using the data in the training set, predict the output for each example in the test, for k = 1, k = 3, and k = 20 (so you will run the algorithm 3 times, once for each value of k). Report the squared error Err on the test set.**

```
Test Error for Euclidean Distance with K = 1 is: 2868.005
Test Error for Euclidean Distance with K = 3 is: 2794.73
Test Error for Euclidean Distance with K = 20 is: 2746.1914125
```

6. **Did k Nearest Neighbour perform better with k = 3 or with k = 20? Why do you think that happened?**

Answer: K Nearest Neighbour performed better with k=20
Reason:-
When the value of K is too small, it becomes a complex model, exhibiting low bias and high variance. It overfits and is susceptible to noise and outliers.

When the value of K is too large, it becomes a simple model, exhibiting high bias and low variance. It underfits.

In this case, the value of K=20 is neither too large nor too small. Thus, it is resilient to noise and outliers. And that is why, the test error is smaller for K=20.

7. **There are many variations of k Nearest Neighbour, which can sometimes significantly improve its performance.**

- **You can use a different distance function. It could be a custom distance function designed specially for your data, or a standard distance function such as the Manhattan distance or the Chebyshev distance.**
- **Instead of computing an average of the k neighbours, you can compute a weighted average of the neighbours. A common way to do this is to weight each of the neighbours by a factor of 1/d, where d is its distance from the test example. The weighted average of neighbours $x^1, \dots, x^k$ is then $({}^{k}_{t=1}(1/d^t)r^t)/({}^{k}_{t=1}(1/d^t))$, where d^t is the distance of the tth neighbour. Implement one variation of k Nearest Neighbour and choose your k. Report the squared error Err on the test set.**

Answer: Using **Manhattan Distance**

```
Test Error for Manhattan Distance with K = 1 is: 2838.83
Test Error for Manhattan Distance with K = 3 is: 2676.45277778
Test Error for Manhattan Distance with K = 20 is: 2800.3250375
```