Goal: Investigate the relationship between the miles per gallon (mpg) and several features of a car using linear and logistic regression.

Problem 1

Classify the cars into 3 categories: low, medium and high mpg:

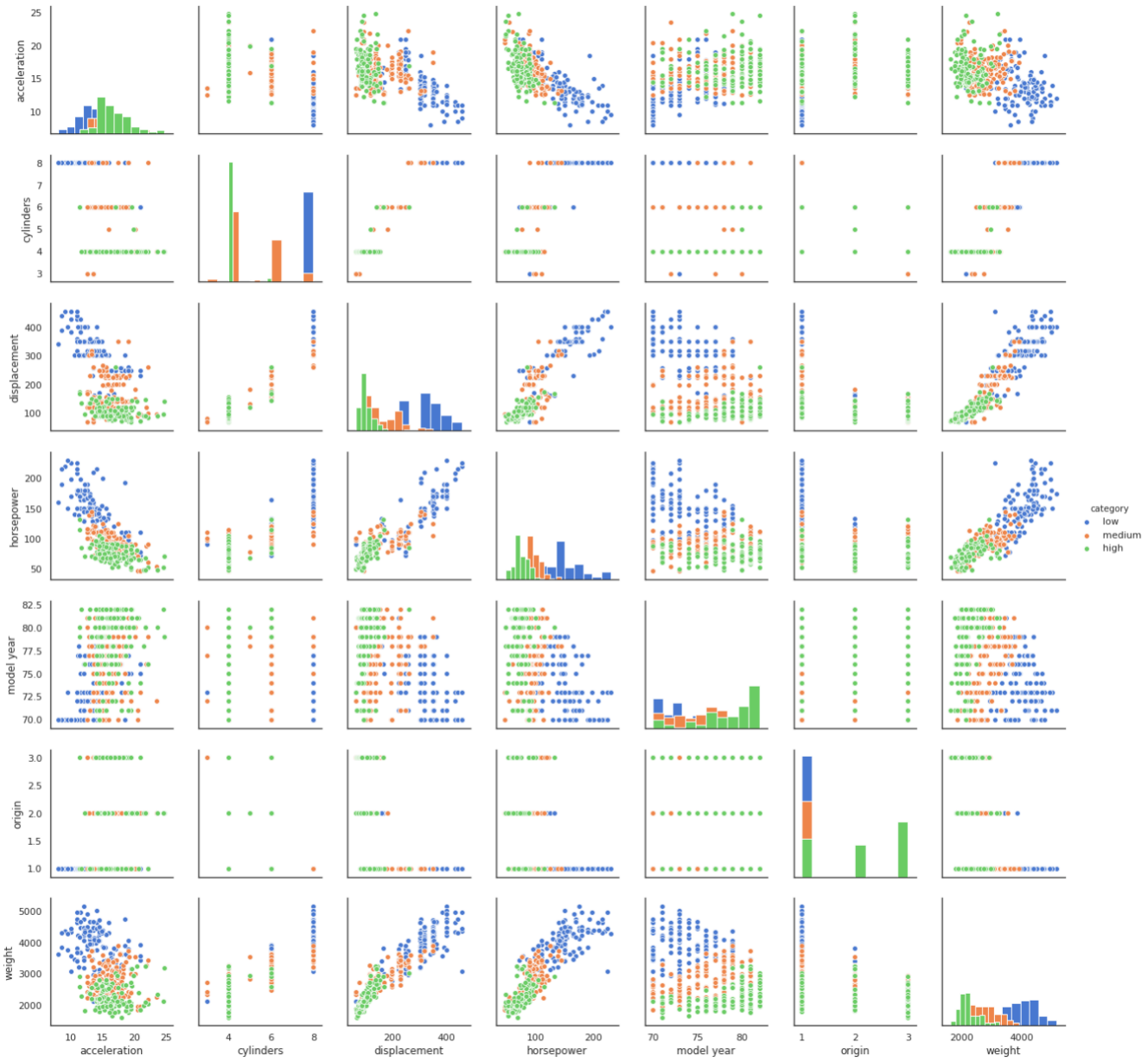
I sorted the dataframe and use np.array_split to split the dataframe into 3 parts. Below is the output that I got after groupby the category.

```
category
high      130
low       131
medium    131
dtype: int64
```

Problem 2

Create a 2D scatterplot matrix to display the correlations between the different attributes of the car.

	acceleration	cylinders	displacement	horsepower	model year	origin	weight
acceleration	1.000000	-0.504683	-0.543800	-0.689196	0.290316	0.212746	-0.416839
cylinders	-0.504683	1.000000	0.950823	0.842983	-0.345647	-0.568932	0.897527
displacement	-0.543800	0.950823	1.000000	0.897257	-0.369855	-0.614535	0.932994
horsepower	-0.689196	0.842983	0.897257	1.000000	-0.416361	-0.455171	0.864538
model year	0.290316	-0.345647	-0.369855	-0.416361	1.000000	0.181528	-0.309120
origin	0.212746	-0.568932	-0.614535	-0.455171	0.181528	1.000000	-0.585005
weight	-0.416839	0.897527	0.932994	0.864538	-0.309120	-0.585005	1.000000



The weight and displacement pair-wise feature combination is the most informative regarding the three mpg categories because it has the highest correlation. Moreover, you can clearly see low weight and displacement correspond to high category. Medium weight and displacement correspond to medium category. High weight and displacement correspond to low category.

Problem 3

Use ordinary least squares estimator to predict the MPG of a car.

```
def solver(dataframe, feature, order):
    """get the weights using OLS formula"""
    # initialize an array with 1
    intercept_value = np.ones((dataframe.shape[0], 1))
    if order == 0:
        X = intercept_value
    else:
        # keep adding the Xs into the array
        order_list = list(range(1, order + 1))
        X = intercept_value
        for order in order_list:
            x1 = np.array(dataframe[feature]**order).reshape(-1, 1)
            X = np.concatenate((X, x1), axis = 1)
    Y = np.array(dataframe['mpg']).T
    #OLS = (X^TX)^-1X^TY
    X_T_Y = np.dot(X.T, Y)
    weight_hat = np.dot(np.linalg.inv(np.dot(X.T, X)), X_T_Y)

    weight_list = weight_hat.tolist()

    return weight_list
```

Problem 4

Split the dataset in the first 200 samples for training and the rest 192 samples for testing. Use OLS solver to regress for 0th to 3rd order polynomial on a single independent variable (feature) each time by using mpg as the dependent variable.

Training Set Mean Squared Error:

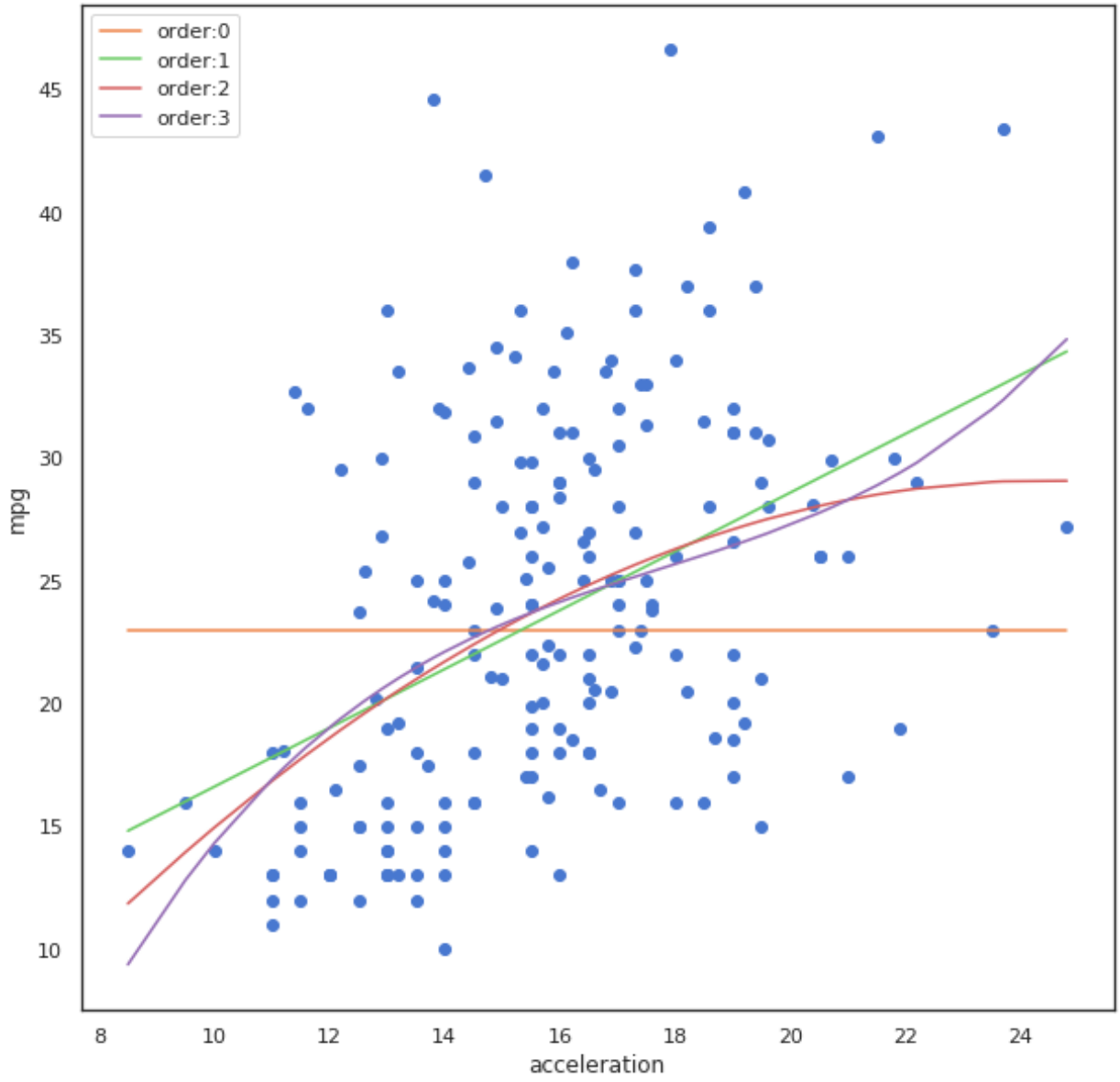
```
acceleration's'0 order's mean squared error is 59.06319
acceleration's'1 order's mean squared error is 49.0194
acceleration's'2 order's mean squared error is 48.42065
acceleration's'3 order's mean squared error is 48.0136


cylinders's'0 order's mean squared error is 59.06319
cylinders's'1 order's mean squared error is 21.31451
cylinders's'2 order's mean squared error is 20.76854
cylinders's'3 order's mean squared error is 19.26468


displacement's'0 order's mean squared error is 59.06319
displacement's'1 order's mean squared error is 19.66003
displacement's'2 order's mean squared error is 16.88157
displacement's'3 order's mean squared error is 16.78953
```

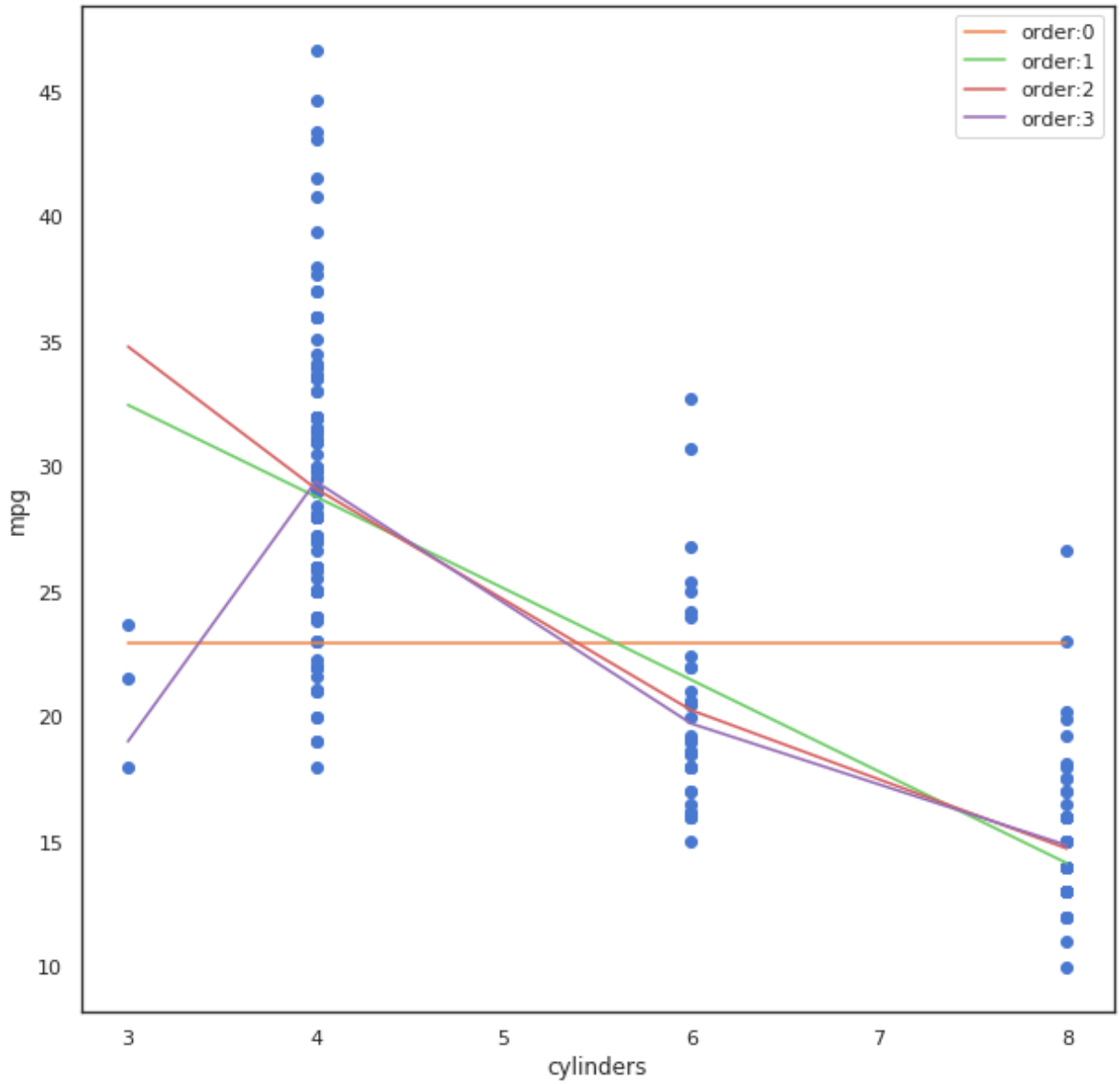
```
horsepower's'0 order's mean squared error is 59.06319
horsepower's'1 order's mean squared error is 22.77429
horsepower's'2 order's mean squared error is 16.8092
horsepower's'3 order's mean squared error is 16.34371


model year's'0 order's mean squared error is 59.06319
model year's'1 order's mean squared error is 40.17046
model year's'2 order's mean squared error is 37.42571
model year's'3 order's mean squared error is 37.41196


origin's'0 order's mean squared error is 59.06319
origin's'1 order's mean squared error is 38.68184
origin's'2 order's mean squared error is 37.60805
origin's'3 order's mean squared error is 1828.35575


weight's'0 order's mean squared error is 59.06319
weight's'1 order's mean squared error is 17.06116
weight's'2 order's mean squared error is 15.12569
weight's'3 order's mean squared error is 15.1216
```

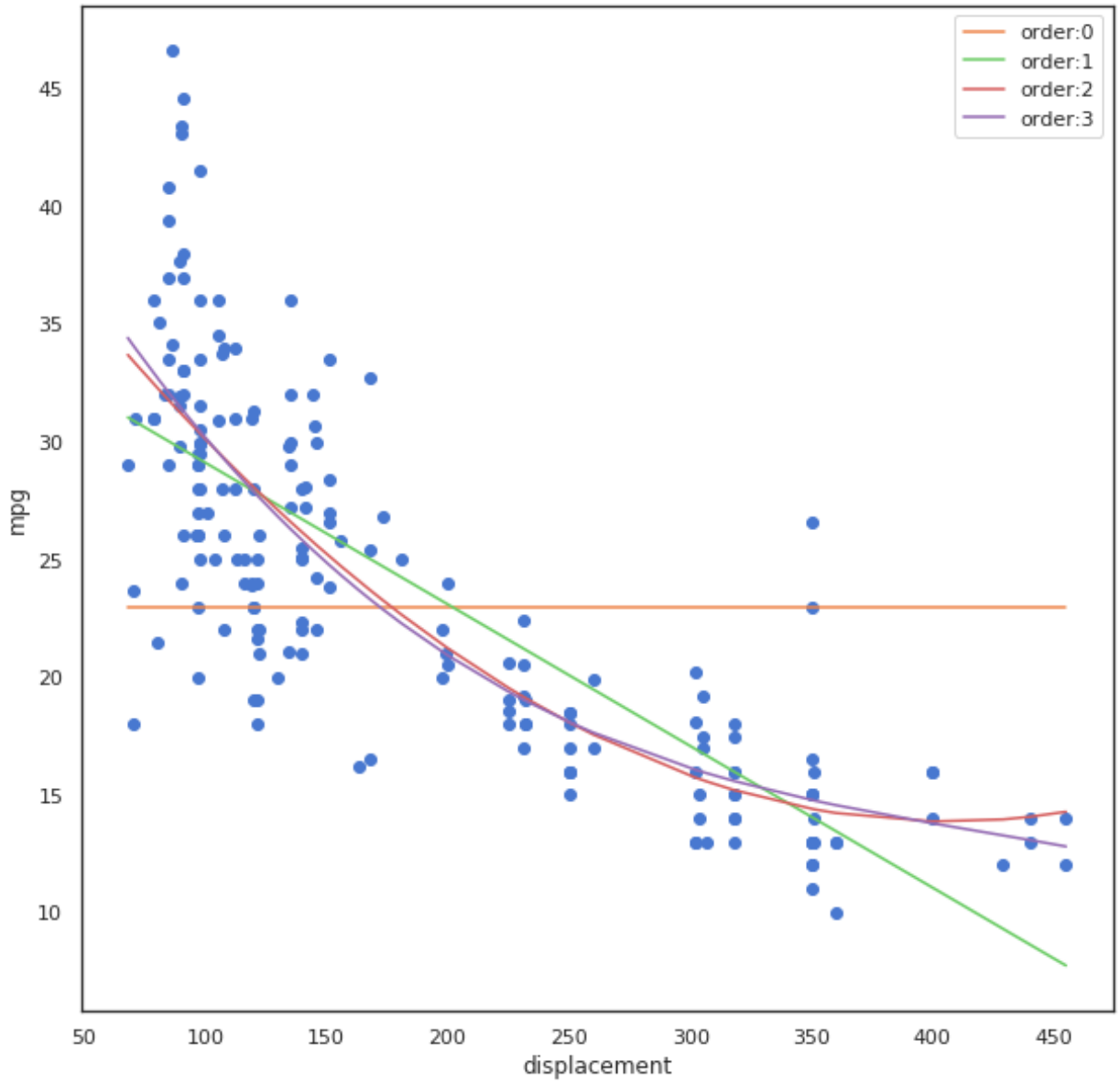
Test_Set Mean Squared Error:

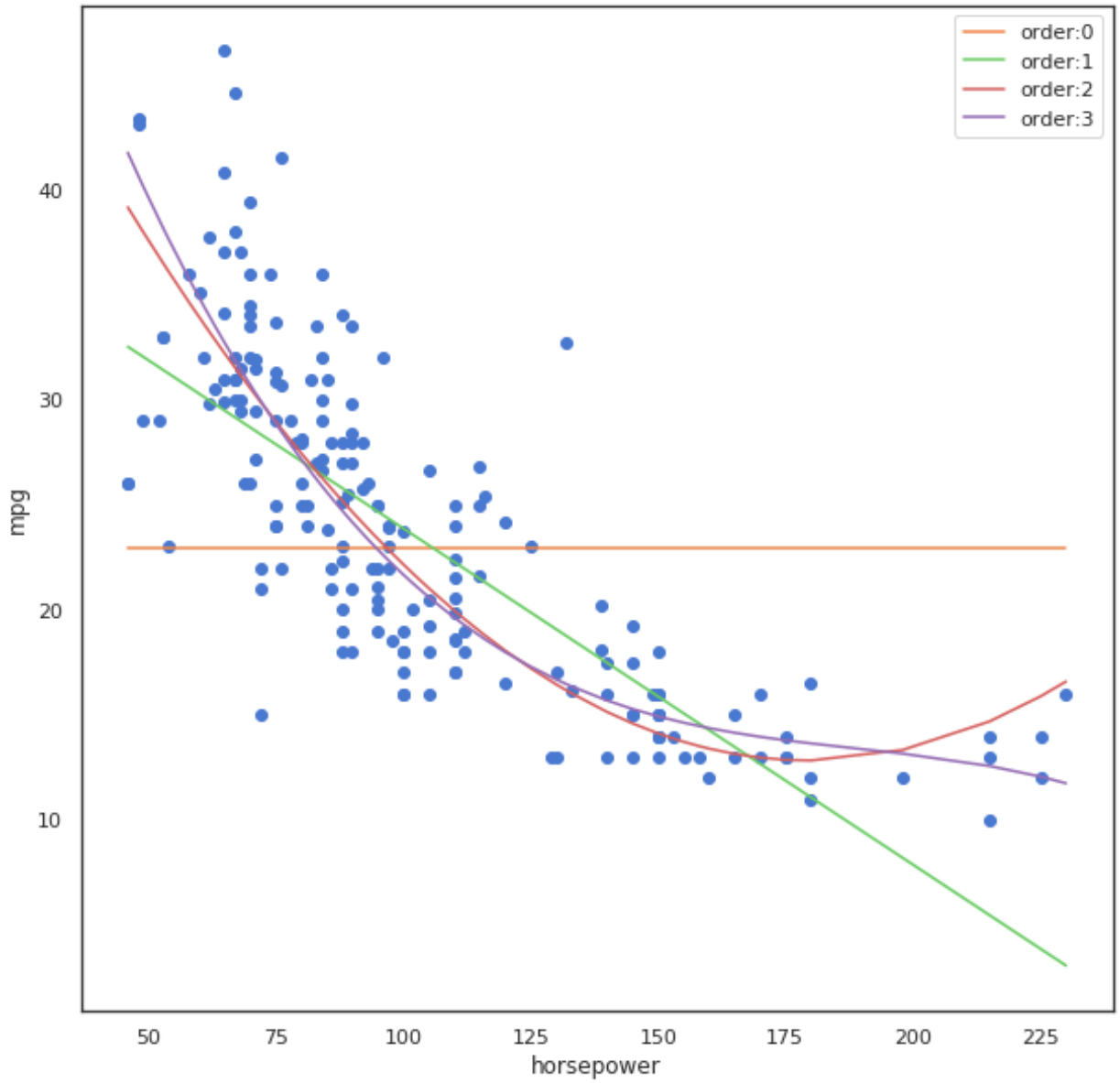
```
acceleration's'0 order's mean squared error is 63.01418
acceleration's'1 order's mean squared error is 50.84666
acceleration's'2 order's mean squared error is 49.75099
acceleration's'3 order's mean squared error is 50.37267


cylinders's'0 order's mean squared error is 63.01418
cylinders's'1 order's mean squared error is 26.91072
cylinders's'2 order's mean squared error is 27.33905
cylinders's'3 order's mean squared error is 24.53442


displacement's'0 order's mean squared error is 63.01418
displacement's'1 order's mean squared error is 23.16152
displacement's'2 order's mean squared error is 21.05027
displacement's'3 order's mean squared error is 21.06452


horsepower's'0 order's mean squared error is 63.01418
horsepower's'1 order's mean squared error is 25.33563
horsepower's'2 order's mean squared error is 21.69581
horsepower's'3 order's mean squared error is 22.75747


model year's'0 order's mean squared error is 63.01418
model year's'1 order's mean squared error is 40.80604
model year's'2 order's mean squared error is 39.97366
```

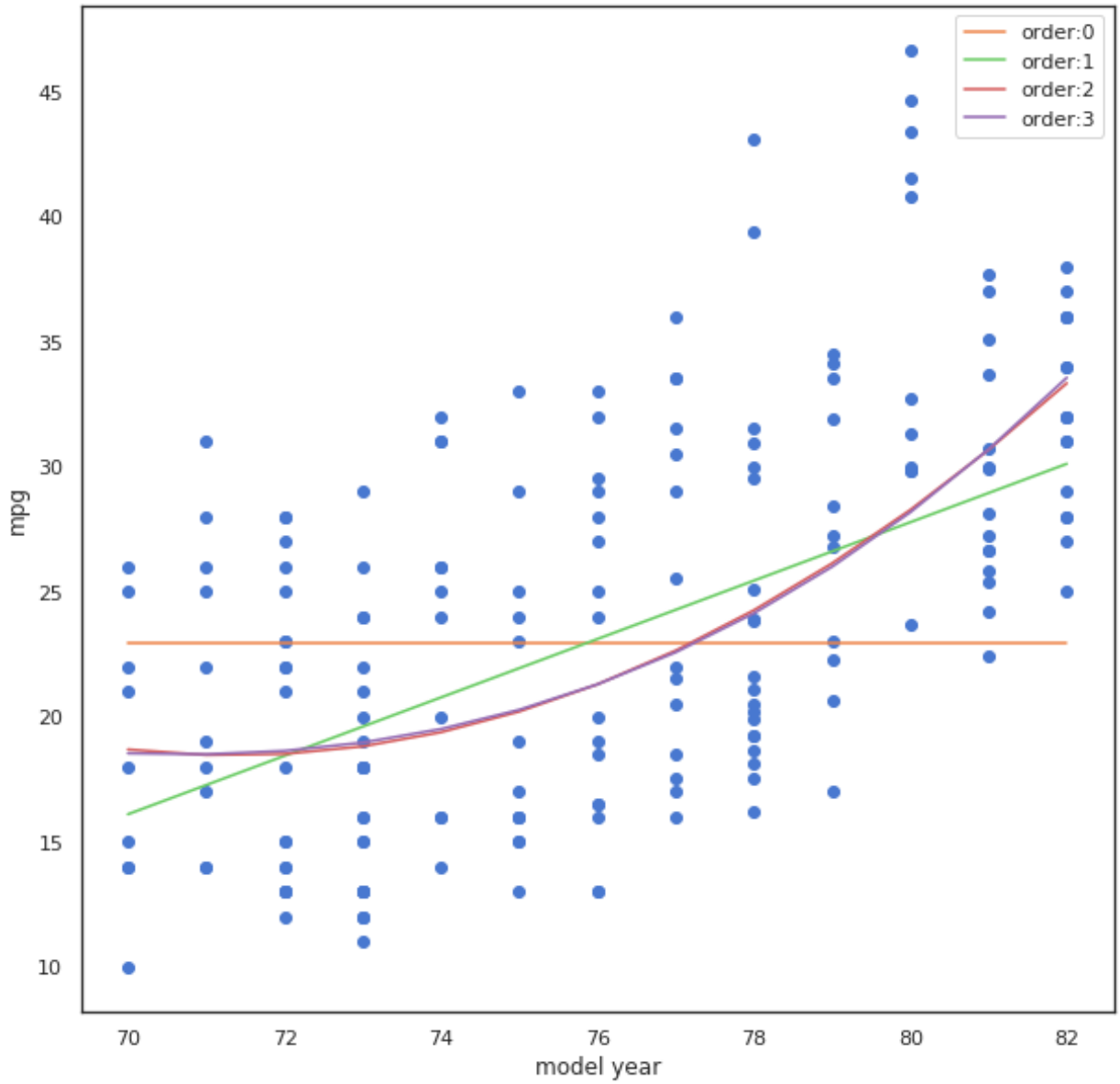
```
model year's'3 order's mean squared error is 40.137


origin's'0 order's mean squared error is 63.01418
origin's'1 order's mean squared error is 44.28936
origin's'2 order's mean squared error is 43.93626
origin's'3 order's mean squared error is 2013.37182


weight's'0 order's mean squared error is 63.01418
weight's'1 order's mean squared error is 20.43057
weight's'2 order's mean squared error is 19.69495
weight's'3 order's mean squared error is 19.71088
```

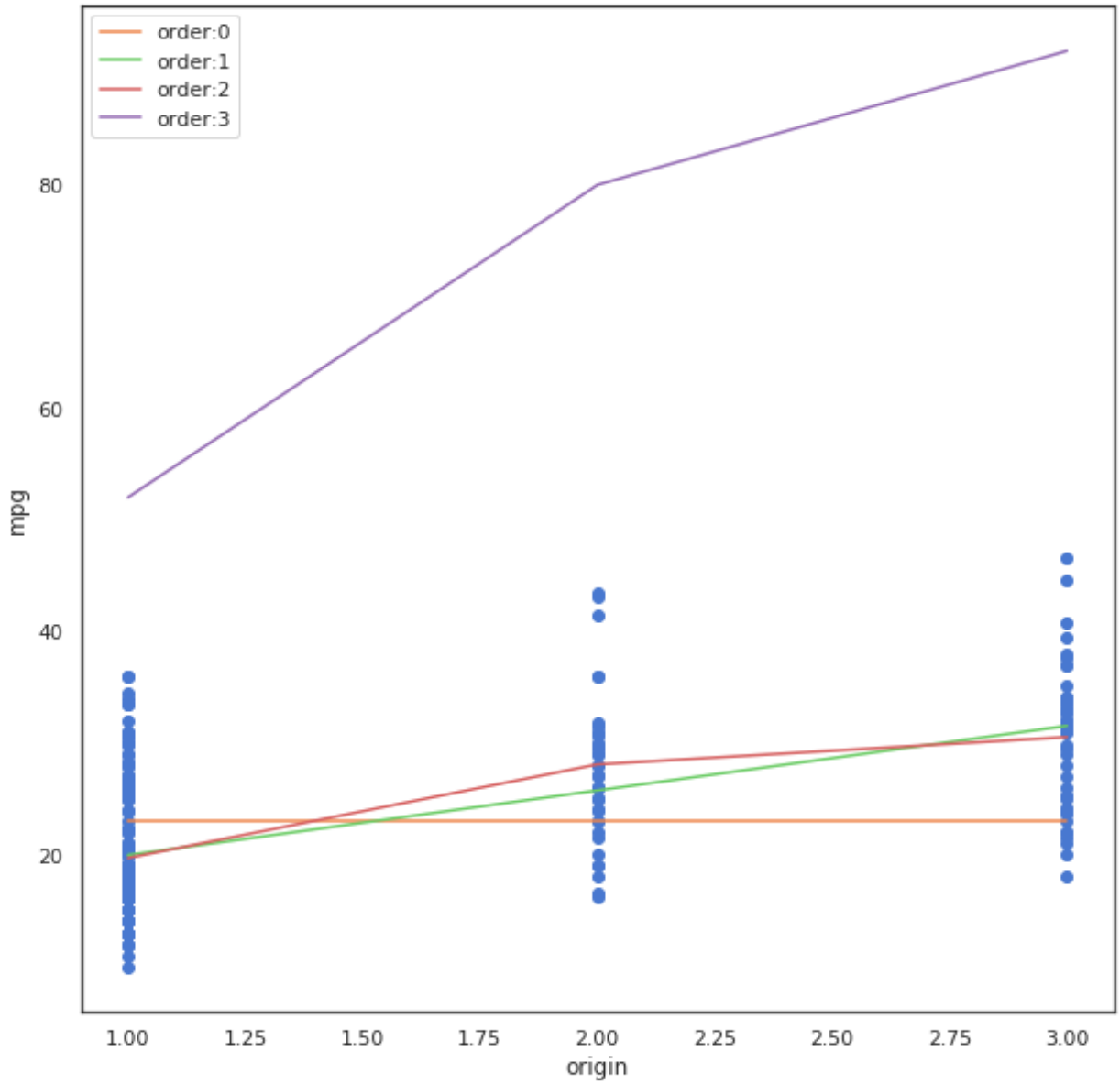


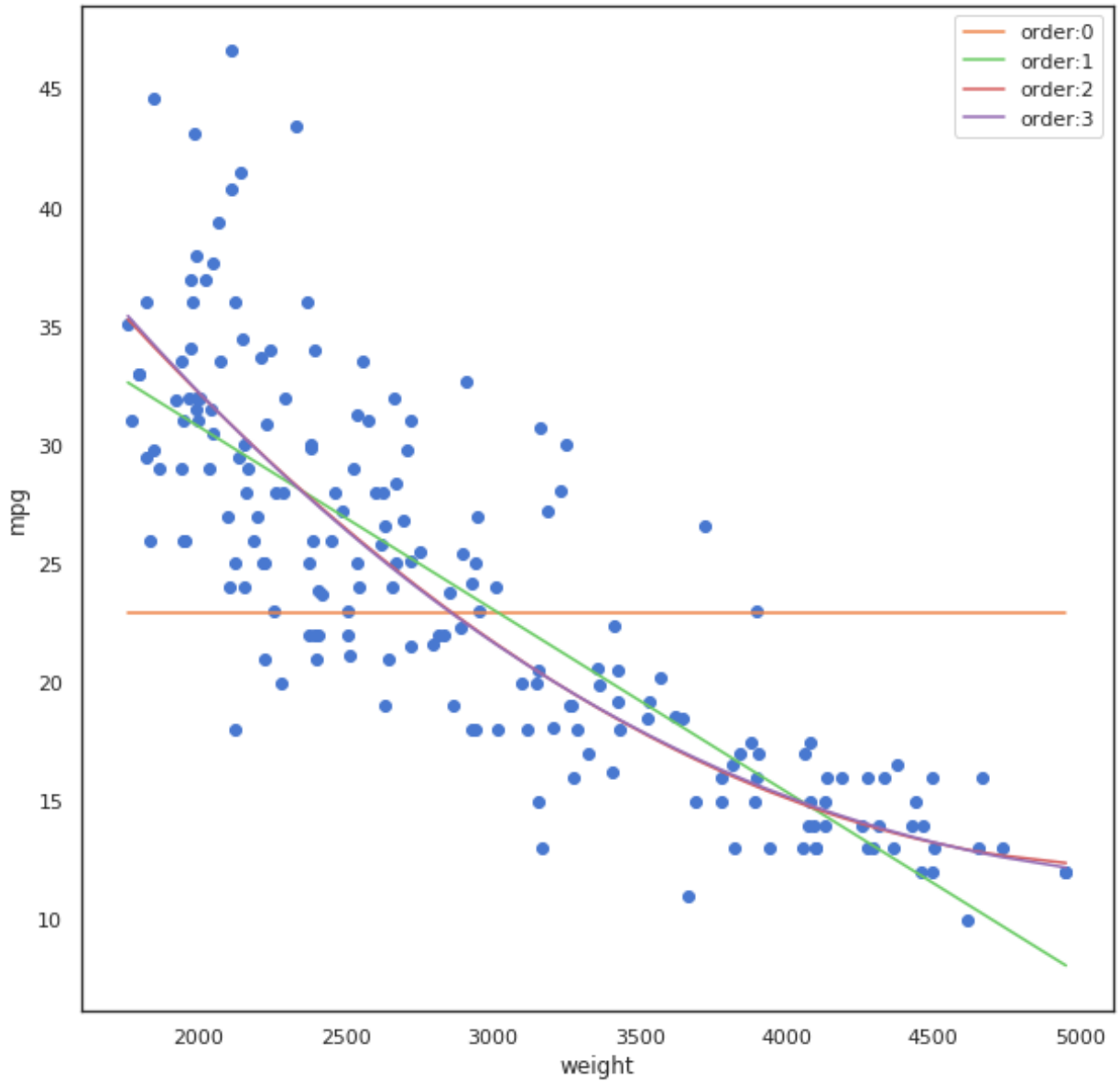

order:0
order:1
order:2
order:3
mpg
cylinders
45
40
35
30
25
20
15
10
3
4
5
6
7
8


order:0
order:1
order:2
order:3
mpg
displacement
45
40
35
30
25
20
15
10
50
100
150
200
250
300
350
400
450


order:0
order:1
order:2
order:3
40
30
20
10
mpg
50
75
100
125
150
175
200
225
horsepower


order:0
order:1
order:2
order:3
mpg
model year
45
40
35
30
25
20
15
10
70
72
74
76
78
80
82


order:0
order:1
order:2
order:3
80
60
40
20
mpg
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
origin



The second order polynomial performs the best in the test set because it has the lowest mean squared error across all the features. The most informative feature for mpg consumption in that case is weight because it has the lowest mean squared error, as the result has shown below.

Second Order Polynomial Mean Squared Error for Test_Set breakdown by features:

```
acceleration's'2 order's mean squared error is 49.75099
cylinders's'2 order's mean squared error is 27.33905
displacement's'2 order's mean squared error is 21.05027
horsepower's'2 order's mean squared error is 21.69581
model year's'2 order's mean squared error is 39.97366
origin's'2 order's mean squared error is 43.93626
weight's'2 order's mean squared error is 19.69495
```

Problem 5

Modify the OLS solver to be able to handle second order polynomials of all 7 independent variables simultaneously (i.e. 15 terms).

```
def solver_multi(dataframe, feature_list, order):
    """
    get the weights using OLS formula for multiple features
    """
    intercept_value = np.ones((dataframe.shape[0], 1))
    if order == 0:
        X = intercept_value
    else:
        X = intercept_value

        order_list = list(range(1, order + 1))
        for order in order_list:
            ## add a for loop for features
            for feature in feature_list:
                x1 = np.array(dataframe[feature]**order).reshape(-1, 1)
                X = np.concatenate((X, x1), axis = 1)
    Y = np.array(dataframe['mpg']).T
    X_T_Y = np.dot(X.T, Y)
    weight_hat = np.dot(np.linalg.inv(np.dot(X.T, X)), X_T_Y)

    weight_list = weight_hat.tolist()

    return weight_list
```

```
0 order's mean squared error for training set is 59.06
1 order's mean squared error for training set is 10.06
2 order's mean squared error for training set is 5.97

0 order's mean squared error for testing set is 63.01
1 order's mean squared error for testing set is 12.09
2 order's mean squared error for testing set is 9.29
```

Problem 6

Using logistic regression (1st order) for low/medium/high classification.

```
column_order_list = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
       'acceleration', 'model year', 'origin']
X_data = np.array(final_df[final_df.columns.difference(['category','car name'])])
Y_data = np.array(final_df['category'])
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, stratify = Y_data, random_state = 42)
log_reg = LogisticRegression(solver = 'liblinear')
clf = log_reg.fit(X_train, Y_train)
Y_predict = log_reg.predict(X_train)
```

```
The accuracy on the training set is 94.22%.
The accuracy on the testing set is 90.82%.
```

Problem 7

If a USA manufacturer (origin1) had considered to introduce a model in 1980 with the following characteristics: 6 cylinders, 350 cc displacement, 180 horsepower, 3700 lb weight, 9 m/sec2 acceleration, the expected MPG rating is 20.71 and the mpg category is low.