06 Model Complexity and Generalization in Supervised Learning

Homework Problems

Nandi Christmas

- Using your own dataset, split it into a training and test set.
- Multiple Linear Regression: Fit a multiple linear regression model with every predictor variable in your data set. Use feature engineering as needed to grossly over fit your model. GROSSLY OVERFIT THIS MODEL! Do this to feel what incorrect maching learning feels like.
- LASSO Regression: Fit a LASSO regression model with every predictor variable in your multiple linear regression model. Compare the coefficients.
- Evaluate the training and test mean squared error for both models. If you grossly overfit your multiple linear regression model, its training mse will be small, but its test mse will be large.

In [47]:
```
import pandas as pd
import numpy as np
df = pd.read_csv(
    filepath_or_buffer = "C:\\Users\\knc5576\\Downloads\\output.csv",
    engine = 'pyarrow',
    dtype = {
    'car_ID': str,
    'symboling': float,
    'doornumber': float,
    'wheelbase': float,
    'carlength': float,
    'carwidth': float,
    'carheight': float,
    'curbweight': float,
    'cylindernumber': float,
    'enginesize': float,
    'boreratio': float,
    'stroke': float,
    'compressionratio': float,
    'horsepower': float,
    'peakrpm': float,
    'citympg': float,
    'highwaympg': float,
    'price': float,
    'TrainTest': str,
    'CarName_1_low': float,
    'CarName_2_moderate': float,
    'CarName_3_high': float,
    'drivewheel_1_low': float,
    'drivewheel_2_moderate': float,
    'drivewheel_3_high': float,
    'enginelocation_2_moderate': float,
    'enginelocation_3_high': float,
    'aspiration_2_moderate': float,
    'aspiration_3_high': float,
    'fuelsystem_1_low': float,
    'fuelsystem_2_moderate': float,
    'fuelsystem_3_high': float,
    'enginetype_2_moderate': float,
    'enginetype_3_high': float,
    'fueltype_2_moderate': float,
    'fueltype_3_high': float,
    'carbody_1_low': float,
    'carbody_2_moderate': float,
    'carbody_3_high': float
    }
)
```

Extreme Overfitting with CarLength Variable

In [48]:
```
df['TrainTest'] = 'Test'
df.loc[~df['carlength'].duplicated(),'TrainTest'] = 'Train'
df.loc[df['price'].isna(),'TrainTest'] = 'Score'
df[['price','carlength','TrainTest']]
```

Out[48]:

	price	carlength	TrainTest
0	9.510075	-0.404369	Train
1	9.711116	-0.404369	Test
2	9.711116	-0.205730	Train
3	9.543235	0.233459	Train
4	9.767095	0.233459	Test
...	...	...	...
405	NaN	1.188740	Score
406	NaN	1.188740	Score
407	NaN	1.188740	Score
408	NaN	1.188740	Score
409	NaN	1.188740	Score

410 rows × 3 columns

In [49]:
```
X_Train = df.loc[df['TrainTest'] == 'Train'][['carlength']]
y_Train = df.loc[df['TrainTest'] == 'Train']['price']
```

In [50]:
```
X_Train
#y_Train
```

Out[50]:

	carlength
0	-0.404369
2	-0.205730
3	0.233459
5	0.289628
6	1.484036
...	...
189	-1.212738
190	-0.664200
191	0.520517
193	0.748552
194	1.188740

75 rows × 1 columns

In [51]:
```
from sklearn.preprocessing import PolynomialFeatures
PolynomialFeatures_OverallQual = PolynomialFeatures(degree=100,include_bias=False).fit(
    X=X_Train,
    y=y_Train
)
```

In [52]:
```
from sklearn.linear_model import LinearRegression
LinearRegression_PolynomialFeatures_OverallQual=LinearRegression().fit(
    X = PolynomialFeatures_OverallQual.transform(
        X=X_Train
    ),
    y = y_Train
)
LinearRegression_PolynomialFeatures_OverallQual.coef_
```

Out[52]:
```
array([-3.90915396e-39,  1.07387219e-43, -7.36904061e-46, -8.55284707e-50,
        0.00000000e+00,  5.21938969e-65,  1.19542972e-64,  2.59723713e-64,
        5.87748762e-64,  1.28357033e-63,  2.88737373e-63,  6.32365713e-63,
        1.41748499e-62,  3.11021341e-62,  6.95508801e-62,  1.52816384e-61,
        3.41123692e-61,  7.50331950e-61,  1.67259574e-60,  3.68234945e-60,
        8.19924609e-60,  1.80650111e-59,  4.01870421e-59,  8.85992074e-59,
        1.96945552e-58,  4.34436376e-58,  9.65091186e-58,  2.12983905e-57,
        4.72893201e-57,  1.04401400e-56,  2.31706547e-56,  5.11700530e-56,
        1.13527268e-55,  2.50774675e-55,  5.56228900e-55,  1.22889770e-54,
        2.72522478e-54,  6.02165765e-54,  1.33521099e-53,  2.95044089e-53,
        6.54183092e-53,  1.44553574e-52,  3.20518844e-52,  7.08174908e-52,
        1.57041983e-51,  3.46911535e-51,  7.69465885e-51,  1.69925200e-50,
        3.77032122e-50,  8.32240958e-50,  1.84751281e-49,  4.07549439e-49,
        9.05362329e-49,  1.99541973e-48,  4.43700784e-48,  9.76759054e-48,
        2.17469937e-47,  4.77977979e-47,  1.06600606e-46,  2.33804251e-46,
        5.22617036e-46,  1.14304356e-45,  2.56262363e-45,  5.58419623e-45,
        1.25683717e-44,  2.72546511e-44,  6.16570150e-44,  1.32849731e-43,
        3.02559421e-43,  6.46441791e-43,  1.48516482e-42,  3.13825565e-42,
        7.29247641e-42,  1.51875262e-41,  3.58171730e-41,  7.31895124e-41,
        1.75939577e-40,  3.50688090e-40,  8.64095178e-40,  1.66725506e-39,
        4.24065548e-39,  7.84209003e-39,  2.07737454e-38,  3.63428528e-38,
        1.01386619e-37,  1.64951513e-37,  4.91321786e-37,  7.26642202e-37,
        2.34986820e-36,  3.06283388e-36,  1.09677665e-35,  1.20588612e-35,
        4.88351546e-35,  4.23877910e-35,  1.96675623e-34,  1.20256653e-34,
        6.01300878e-34,  2.00227046e-34, -9.31286414e-35, -3.21272891e-35])
```

In [53]:
```
import numpy as np
df_Line = pd.DataFrame({
    'carlength' : np.arange(-3, 2.7, 0.01)
})
df_Line['predict'] = LinearRegression_PolynomialFeatures_OverallQual.predict(
    X=PolynomialFeatures_OverallQual.transform(
        X=df_Line
    )
)
```

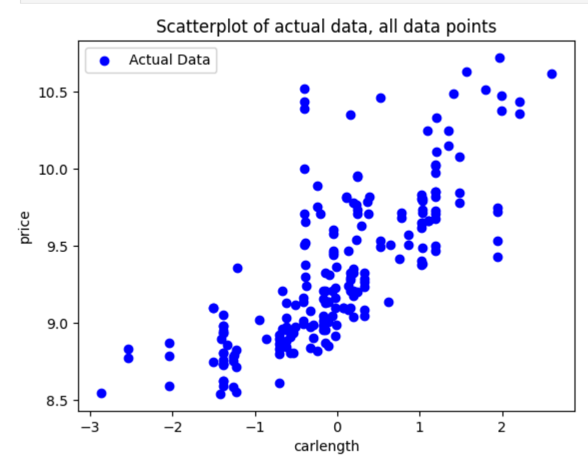
In [54]:
```
df_Line
```

Out[54]:

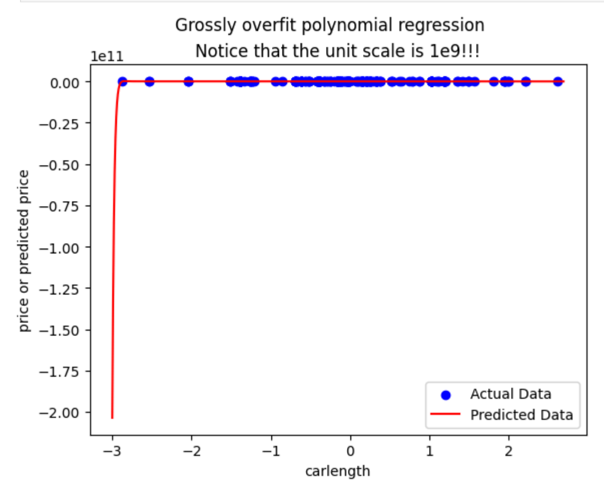
	carlength	predict
0	-3.00	-2.035436e+11
1	-2.99	-1.325862e+11
2	-2.98	-8.561303e+10
3	-2.97	-5.472246e+10
4	-2.96	-3.455919e+10
...	...	...
565	2.65	-4.900110e+06
566	2.66	-8.722403e+06
567	2.67	-1.495796e+07
568	2.68	-2.497581e+07
569	2.69	-4.087022e+07

570 rows × 2 columns

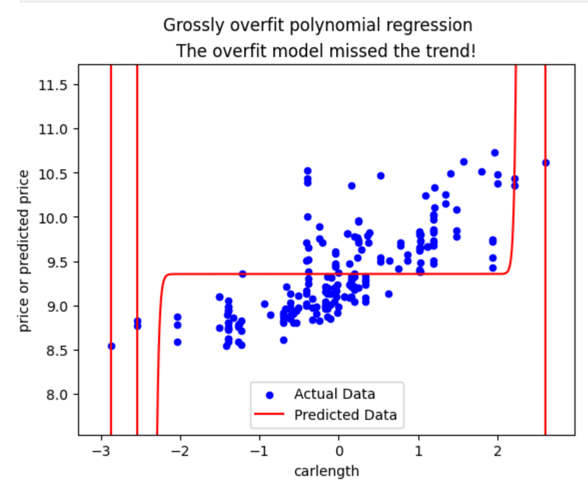
In [55]:
```
import matplotlib.pyplot as plt
plt.scatter(
    x=df['carlength'],
    y=df['price'],
    color='blue',
    label='Actual Data'
)

# Adding labels and title
plt.xlabel('carlength')
plt.ylabel('price')
plt.title('Scatterplot of actual data, all data points')
plt.legend()

# Show plot
plt.show()
```



In [56]:
```
import matplotlib.pyplot as plt
plt.scatter(
    x=df['carlength'],
    y=df['price'],
    color='blue',
    label='Actual Data'
)

# Line plot of predicted data
plt.plot(
    df_Line['carlength'],
    df_Line['predict'],
    color='red',
    label='Predicted Data'
)

# Adding labels and title
plt.xlabel('carlength')
plt.ylabel('price or predicted price')
plt.suptitle('Grossly overfit polynomial regression')
plt.title('Notice that the unit scale is 1e9!!!')
plt.legend()

# Show plot
plt.show()
```



In [57]:
```
import matplotlib.pyplot as plt

# Scatter plot of actual data
plt.scatter(
    x=df['carlength'],
    y=df['price'],
    color='blue',
    label='Actual Data',
    s=20  # Adjust marker size
)
plt.plot(
    df_Line['carlength'],
    df_Line['predict'],
    color='red',
    label='Predicted Data'
)
plt.xlabel('carlength')
plt.ylabel('price or predicted price')
plt.suptitle('Grossly overfit polynomial regression')
plt.title('The overfit model missed the trend!')
plt.legend()
plt.ylim(df['price'].min() - 1, df['price'].max() + 1)

plt.show()
```



In [58]:
```
df['TrainTest'] = np.random.choice(['Train', 'Validation', 'Test'], size=len(df), p=[1/3,1/3,1/3])
df.loc[df['price'].isna(),'TrainTest'] = 'Score'
X_Train = df.loc[df['TrainTest'] == 'Train'][['carlength']]
y_Train = df.loc[df['TrainTest'] == 'Train']['price']

from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
import numpy as np

max_degree = 100

list_PolynomialFeatures_fit = [
    PolynomialFeatures(degree=degree,include_bias=True).fit(
        X=X_Train,
        y=y_Train
    ) for degree in range(max_degree)
]
list_LinearRegression_fit = [
    LinearRegression().fit(
    X = PolynomialFeatures_fit.transform(
        X=X_Train
    ),
    y = y_Train
) for PolynomialFeatures_fit in list_PolynomialFeatures_fit
]
list_squared_error = [
    (
        list_LinearRegression_fit[degree].predict(
            X = list_PolynomialFeatures_fit[degree].transform(
                X = df[['carlength']]
            )
        ) - df['price']
    )**2  for degree in range(max_degree)
]
df_degree = pd.DataFrame(
    data = list_squared_error
).transpose()
df_degree.columns = range(max_degree)
df_degree['TrainTest'] = df['TrainTest']
df_degree = df_degree.groupby('TrainTest').mean().drop('Score').transpose()

print("Degree with least validation MSE: ")
print(np.argmin(df_degree['Validation']))

df_degree.iloc[np.argmin(df_degree['Validation'])]
```

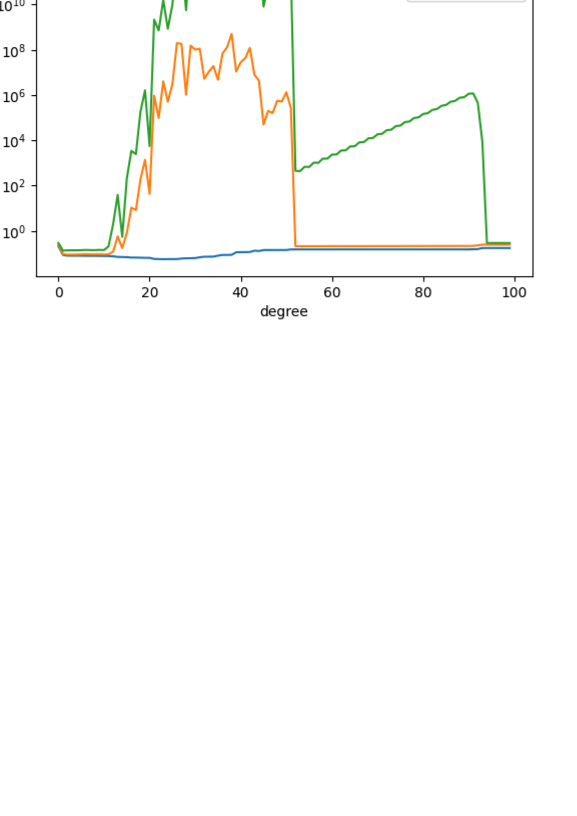
```
Degree with least validation MSE: 
2
```

Out[58]:
```
TrainTest
Test          0.140883
Train         0.081584
Validation    0.090866
Name: 2, dtype: float64
```

Find the optimal degree

my data wouldnt work when it was regular, so i used a log function to help it look a bit better

In [ ]:
```
import matplotlib.pyplot as plt

# Plot the data using a logarithmic scale for the y-axis
df_degree.reset_index().plot(
    x='index',
    y=['Train', 'Validation', 'Test'],
    kind='line',
    logy=True
)

plt.xlabel('degree')
plt.ylabel('mean squared error (log scale)')
plt.title('Line Plot of Train, Validation, and Test')
plt.show()
```



Line Plot of Train, Validation, and Test
mean squared error (log scale)
degree
TrainTest
Train
Validation
Test