08 Lasso, Ridge, and Elastic-Net Regression in Supervised Learning

Linear regression is a fundamental tool in statistics and machine learning. However, when dealing with high-dimensional data, or when multicollinearity (correlation between predictors) is present, plain linear regression can have limitations. To address these challenges we use penalized regression techniques such as Lasso, Ridge, and Elastic-Net regression.

Penalized regression models:

- handle feature selection
- prevent overfitting from multicollinear predictors

Penalized regression's feature selection is not a substitute for subject matter knowledge driven feature selection.

2. Experiment with different values of λ (from very small to very large). What do you observe in terms of coefficient shrinkage?

whenever the lambda seemed to increase, my coefficient numbers would decrease/shrink.

3. Using the same dataset, split it into a training and test set. Evaluate the performance of ridge, lasso, and elastic-net on the test set. Which method performs the best and why? Looking at the last chart to compare all three, it seems that the LassoCV performed the best out of all three. This is because it has the lowest score of all methods, meaning it had better performance

4. Reflect on the importance of feature selection in real-world applications. Discuss scenarios where you'd prefer lasso over ridge regression and vice-versa. Feature selection is important for when you do not need added variables that make the data more 'messy'. This allows for a 'cleaner' seeming dataset so it doesn't get too messy with deleting or changing variables often. I would say you use Lasso for when you would like an easier sort of model that has most coeffients lying at 0, or close to it. You would use Ridge Regression when there is a high amount of predictor variables and that number is bigger than the amount of observations you have.

Load our data

In [32]:
```python
import pandas as pd
import numpy as np
df = pd.read_csv(
    filepath_or_buffer = "D:\output\\cars_prepared.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': int,
        'CarName_2_moderate': int,
        'CarName_3_high': int,
        'drivewheel_1_low': int,
        'drivewheel_2_moderate': int,
        'drivewheel_3_high': int,
        'enginelocation_2_moderate': int,
        'enginelocation_3_high': int,
        'aspiration_2_moderate': int,
        'aspiration_3_high': int,
        'fuelsystem_1_low': int,
        'fuelsystem_2_moderate': int,
        'fuelsystem_3_high': int,
        'enginetype_2_moderate': int,
        'enginetype_3_high': int,
        'fueltype_2_moderate': int,
        'fueltype_3_high': int,
        'carbody_1_low': int,
        'carbody_2_moderate': int,
        'carbody_3_high': int
        }
    )
```

```
<>:4: SyntaxWarning: invalid escape sequence '\o'
<>:4: SyntaxWarning: invalid escape sequence '\o'
C:\Users\nandi\AppData\Local\Temp\ipykernel_20160\2366787299.py:4: SyntaxWarning: invalid escape sequence '\o'
  filepath_or_buffer = "D:\output\\cars_prepared.csv",
```

In [33]:
```python
list_full=[
        'symboling',
       'doornumber',
        'wheelbase',
        'carlength',
        'carwidth',
        'carheight',
        'curbweight',
        'cylindernumber',
        'enginesize',
        'boreratio',
        'stroke',
        'compressionratio',
        'horsepower',
        'peakrpm',
        'citympg',
        'highwaympg',
        'CarName_1_low',
        'CarName_2_moderate',
        'CarName_3_high',
        'drivewheel_1_low',
        'drivewheel_2_moderate',
        'drivewheel_3_high',
        'enginelocation_2_moderate',
        'enginelocation_3_high',
        'aspiration_2_moderate',
        'aspiration_3_high',
        'fuelsystem_1_low',
        'fuelsystem_2_moderate',
        'fuelsystem_3_high',
        'enginetype_2_moderate',
        'enginetype_3_high',
        'fueltype_2_moderate',
        'fueltype_3_high',
        'carbody_1_low',
        'carbody_2_moderate',
        'carbody_3_high'
]

list_reduced = [
'symboling',
'doornumber',
'wheelbase',
'carlength',
'carwidth',
'carheight',
'curbweight',
'cylindernumber',
'enginesize',
'boreratio',
'stroke',
'compressionratio',
'horsepower',
'peakrpm',
'citympg',
'highwaympg',
'CarName_1_low',
'drivewheel_1_low',
'enginelocation_3_high',
'aspiration_3_high',
'fuelsystem_3_high',
'enginetype_2_moderate',
'fueltype_2_moderate',
'carbody_1_low'
]

list_predictors = list_full + list_reduced

list_float = [
        'symboling',
       'doornumber',
        'wheelbase',
        'carlength',
        'carwidth',
        'carheight',
        'curbweight',
        'cylindernumber',
        'enginesize',
        'boreratio',
        'stroke',
        'compressionratio',
        'horsepower',
        'peakrpm',
        'citympg',
        'highwaympg'
]

list_binary = [
        'CarName_1_low',
        'CarName_2_moderate',
        'CarName_3_high',
        'drivewheel_1_low',
        'drivewheel_2_moderate',
        'drivewheel_3_high',
        'enginelocation_2_moderate',
        'enginelocation_3_high',
        'aspiration_2_moderate',
        'aspiration_3_high',
        'fuelsystem_1_low',
        'fuelsystem_2_moderate',
        'fuelsystem_3_high',
        'enginetype_2_moderate',
        'enginetype_3_high',
        'fueltype_2_moderate',
        'fueltype_3_high',
        'carbody_1_low',
        'carbody_2_moderate',
        'carbody_3_high'
]
```

In [34]:
```python
df[list_float].describe().loc[['mean','std']]
```

Out[34]:

	symboling	doornumber	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg
mean	-2.816175e-17	4.332578e-18	0.001478	-4.332578e-17	0.004865	4.332578e-17	0.000309	0.018917	0.00256	-5.199093e-17	-0.006057	-3.899320e-17	-1.733031e-17	-0.000021	6.498866e-18	2.166289e-18
std	1.001222e+00	1.001222e+00	0.996658	1.001222e+00	0.984021	1.001222e+00	1.000284	0.893748	0.99282	1.001222e+00	0.980561	1.001222e+00	1.001222e+00	1.001159	1.001222e+00	1.001222e+00

In [35]:
```python
from sklearn.preprocessing import StandardScaler
StandardScaler_float = StandardScaler().fit(
    X = df[list_float],
    y = df['price']
)
df[list_float] = StandardScaler_float.transform(
    X = df[list_float]
)
df[list_float].describe().loc[['mean','std']]
```

Out[35]:

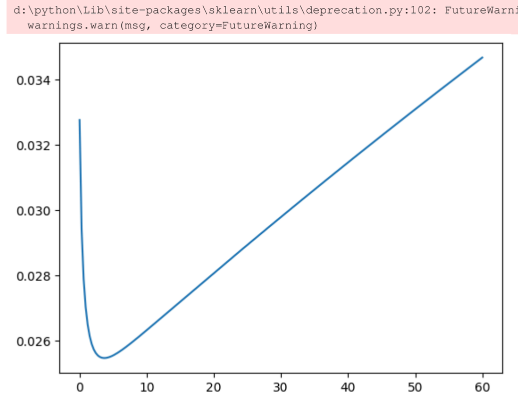
	symboling	doornumber	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio	stroke	compressionratio	horsepower	peakrpm	citympg	highwaympg
mean	-2.599547e-17	4.332578e-18	2.166289e-18	-1.733031e-17	1.299773e-17	1.733031e-17	5.199093e-17	2.707861e-17	4.332578e-18	-5.199093e-17	-1.733031e-17	-1.299773e-17	-6.498866e-18	-8.665155e-18	6.498866e-18	2.166289e-18
std	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00	1.001222e+00

In [36]:
```python
array_alphas = np.linspace(0.01,60,200)
from sklearn.linear_model import RidgeCV
RidgeCV_full = RidgeCV(
    alphas=array_alphas,
    store_cv_results = True
).fit(
    X = df.loc[df['TrainTest'] == 'Train',list_full],
    y = df.loc[df['TrainTest'] == 'Train','price']
)
RidgeCV_full.alpha_
```

Out[36]: `np.float64(3.6274874371859296)`

In [37]:
```python
import matplotlib.pyplot as plt
plt.plot(RidgeCV_full.alphas, RidgeCV_full.cv_values_.mean(axis = 0))
plt.show()
```

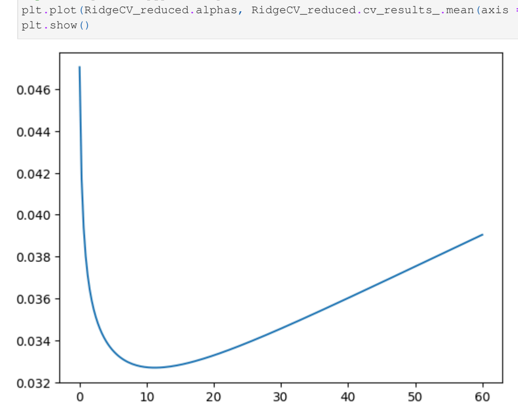
```
d:\python\Lib\site-packages\sklearn\utils\deprecation.py:102: FutureWarning: Attribute `cv_values_` is deprecated in version 1.5 and will be removed in 1.7. Use `cv_results_` instead.
  warnings.warn(msg, category=FutureWarning)
```



In [38]:
```python
RidgeCV_reduced = RidgeCV(
    alphas=array_alphas,
    store_cv_results = True
).fit(
    X = df.loc[df['TrainTest'] == 'Train',list_reduced],
    y = df.loc[df['TrainTest'] == 'Train','price']
)
RidgeCV_reduced.alpha_
```

Out[38]: `np.float64(11.163919597989949)`

In [39]:
```python
import matplotlib.pyplot as plt
plt.plot(RidgeCV_reduced.alphas, RidgeCV_reduced.cv_results_.mean(axis = 0))
plt.show()
```



In [40]:
```python
pd.DataFrame({
    'feature_names_in_' : RidgeCV_reduced.feature_names_in_,
    'coef_' : RidgeCV_reduced.coef_
}).sort_values('coef_')
```

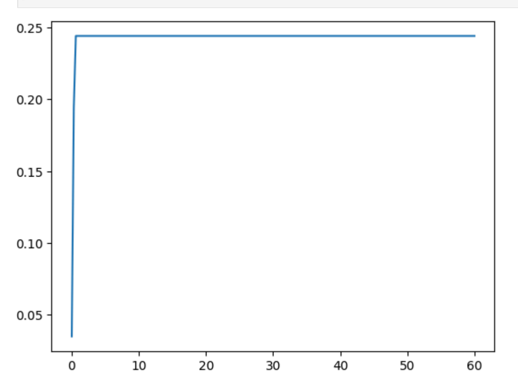
Out[40]:

	feature_names_in_	coef_
14	citympg	-0.088026
16	CarName_1_low	-0.086042
17	drivewheel_1_low	-0.041692
2	wheelbase	-0.038334
15	highwaympg	-0.032904
22	fueltype_2_moderate	-0.031053
23	carbody_1_low	-0.017149
9	boreratio	-0.014830
21	enginetype_2_moderate	-0.001476
0	symboling	-0.001069
10	stroke	-0.000385
20	fuelsystem_3_high	0.003338
1	doornumber	0.003391
3	carlength	0.010208
19	aspiration_3_high	0.013582
5	carheight	0.020659
13	peakrpm	0.023335
18	enginelocation_3_high	0.044824
8	enginesize	0.055351
4	carwidth	0.060468
11	compressionratio	0.069795
12	horsepower	0.089130
6	curbweight	0.101649
7	cylindernumber	0.107096

In [41]:
```python
from sklearn.linear_model import LassoCV
LassoCV_full = LassoCV(
    alphas=array_alphas
).fit(
    X = df.loc[df['TrainTest'] == 'Train',list_full],
    y = df.loc[df['TrainTest'] == 'Train','price']
)
LassoCV_full.alpha_
```

Out[41]: `np.float64(0.01)`

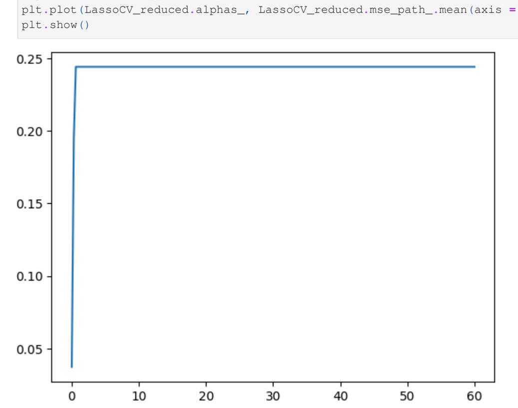
In [42]:
```python
import matplotlib.pyplot as plt
plt.plot(LassoCV_full.alphas_, LassoCV_full.mse_path_.mean(axis = 1))
plt.show()
```



In [43]:
```python
LassoCV_reduced = LassoCV(
    alphas = array_alphas
).fit(
    X = df.loc[df['TrainTest'] == 'Train',list_reduced],
    y = df.loc[df['TrainTest'] == 'Train','price']
)
LassoCV_reduced.alpha_
```

Out[43]: `np.float64(0.01)`

In [44]:
```python
import matplotlib.pyplot as plt
plt.plot(LassoCV_reduced.alphas_, LassoCV_reduced.mse_path_.mean(axis = 1))
plt.show()
```



In [45]:
```python
pd.DataFrame({
    'feature_names_in_' : LassoCV_reduced.feature_names_in_,
    'coef_' : LassoCV_reduced.coef_
}).sort_values('coef_')
```

Out[45]:

	feature_names_in_	coef_
14	citympg	-0.112572
16	CarName_1_low	-0.081983
2	wheelbase	-0.010979
1	doornumber	0.000000
5	carheight	0.000000
10	stroke	-0.000000
8	enginesize	0.000000
0	symboling	-0.000000
15	highwaympg	-0.000000
9	boreratio	-0.000000
21	enginetype_2_moderate	-0.000000
20	fuelsystem_3_high	0.000000
19	aspiration_3_high	0.000000
18	enginelocation_3_high	0.000000
17	drivewheel_1_low	-0.000000
3	carlength	0.000000
22	fueltype_2_moderate	-0.000000
23	carbody_1_low	-0.000000
13	peakrpm	0.006525
4	carwidth	0.019680
11	compressionratio	0.084713
12	horsepower	0.109305
7	cylindernumber	0.130434
6	curbweight	0.176139

In [46]:
```python
array_l1_ratios = np.linspace(0.01, 1, 100)
```

In [47]:
```python
from sklearn.linear_model import ElasticNetCV
import numpy as np
ElasticNetCV_full = ElasticNetCV(
    alphas = array_alphas,
    l1_ratio = array_l1_ratios,
    max_iter = 10000
).fit(
    X = df.loc[df['TrainTest'] == 'Train',list_full],
    y = df.loc[df['TrainTest'] == 'Train','price']
)
```

In [48]:
```python
ElasticNetCV_full.alpha_
```

Out[48]: `np.float64(0.01)`

In [49]:
```python
ElasticNetCV_full.l1_ratio_
```

Out[49]: `np.float64(0.25)`

In [50]:
```python
import seaborn as sns
sns.heatmap(ElasticNetCV_full.mse_path_.mean(axis = 2),cmap="coolwarm")
plt.xlabel('alpha')
plt.ylabel('l1_ratio')
plt.show()
```

In [51]:
```python
ElasticNetCV_reduced = ElasticNetCV(
    alphas = array_alphas,
    l1_ratio = array_l1_ratios,
    max_iter = 10000
).fit(
    X = df.loc[df['TrainTest'] == 'Train',list_reduced],
    y = df.loc[df['TrainTest'] == 'Train','price']
)
```

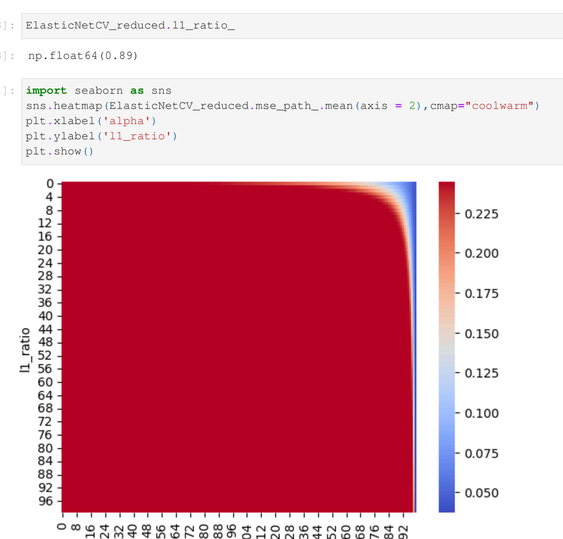
In [52]:
```python
ElasticNetCV_reduced.alpha_
```

Out[52]: `np.float64(0.01)`

In [53]:
```python
ElasticNetCV_reduced.l1_ratio_
```

Out[53]: `np.float64(0.89)`

In [54]:
```python
import seaborn as sns
sns.heatmap(ElasticNetCV_reduced.mse_path_.mean(axis = 2),cmap="coolwarm")
plt.xlabel('alpha')
plt.ylabel('l1_ratio')
plt.show()
```



In [55]:
```python
pd.DataFrame({
    'feature_names_in_' : ElasticNetCV_reduced.feature_names_in_,
    'coef_' : ElasticNetCV_reduced.coef_
}).sort_values('coef_')
```

Out[55]:

	feature_names_in_	coef_
14	citympg	-0.112567
16	CarName_1_low	-0.089057
2	wheelbase	-0.017487
1	doornumber	0.000000
9	boreratio	-0.000000
10	stroke	-0.000000
8	enginesize	0.000000
0	symboling	-0.000000
15	highwaympg	-0.000000
22	fueltype_2_moderate	-0.000000
21	enginetype_2_moderate	0.000000
20	fuelsystem_3_high	-0.000000
19	aspiration_3_high	0.000000
18	enginelocation_3_high	0.000000
17	drivewheel_1_low	-0.000000
3	carlength	0.000000
23	carbody_1_low	-0.000000
5	carheight	0.002777
13	peakrpm	0.008146
4	carwidth	0.023912
11	compressionratio	0.086222
12	horsepower	0.108608
7	cylindernumber	0.130758
6	curbweight	0.175916

In [56]:
```python
list_CV = [RidgeCV_full,RidgeCV_reduced,LassoCV_full,LassoCV_reduced,ElasticNetCV_full,ElasticNetCV_reduced]
df_predict = pd.DataFrame([
    (CV.predict(X = df[CV.feature_names_in_]) - df['price'])**2 for CV in list_CV
]).T
df_predict.columns = [
    'RidgeCV_full','RidgeCV_reduced','LassoCV_full','LassoCV_reduced','ElasticNetCV_full','ElasticNetCV_reduced'
]
df_predict.groupby(df['TrainTest']).mean()
```

Out[56]:

	RidgeCV_full	RidgeCV_reduced	LassoCV_full	LassoCV_reduced	ElasticNetCV_full	ElasticNetCV_reduced
TrainTest						
Score	NaN	NaN	NaN	NaN	NaN	NaN
Test	0.053390	0.056342	0.051899	0.063177	0.053512	0.063135
Train	0.012217	0.019953	0.017819	0.020717	0.012229	0.020282
Validation	0.018458	0.026295	0.018843	0.027990	0.017515	0.028076