CAR PRICE INTERPRETATION

(SUPERVISED MACHINE LEARNING)

Course project – peer review

Wang Xu Apple

21 Dec 2021

Brief description of dataset

- A car company (Geely Auto) is going to enter US market and to gain competitiveness.
- Geely has contracted an automobile consulting company to understand the factors on which the pricing of cars depends.
- Specifically, Geely want to understand the factors affecting the pricing of cars in the American market, which strategies are very different from before.
- Based on various market surveys, the consulting firm has gathered a large data set of different types of cars across the America market.

  - *This is only a part of real-world dataset for analysis.*

Dataset ref: https://www.kaggle.com/hellbuoy/car-price-prediction/download

Project introduction

- This project is mainly focus on interpretation objective from the dataset.
- As an analyst, I'm going to report my findings and insights to Geely's management which they may concerns and help to make business decisions in future:
  - Which variables are significant in predicting the price of a car
  - How well those variables describe the price of a car
  - features interpretation and comparison
- Main contents of report:
  - Data exploration analysis and features engineering
  - Supervised machine learning models (Multi Linear Regression)
  - Conclusion

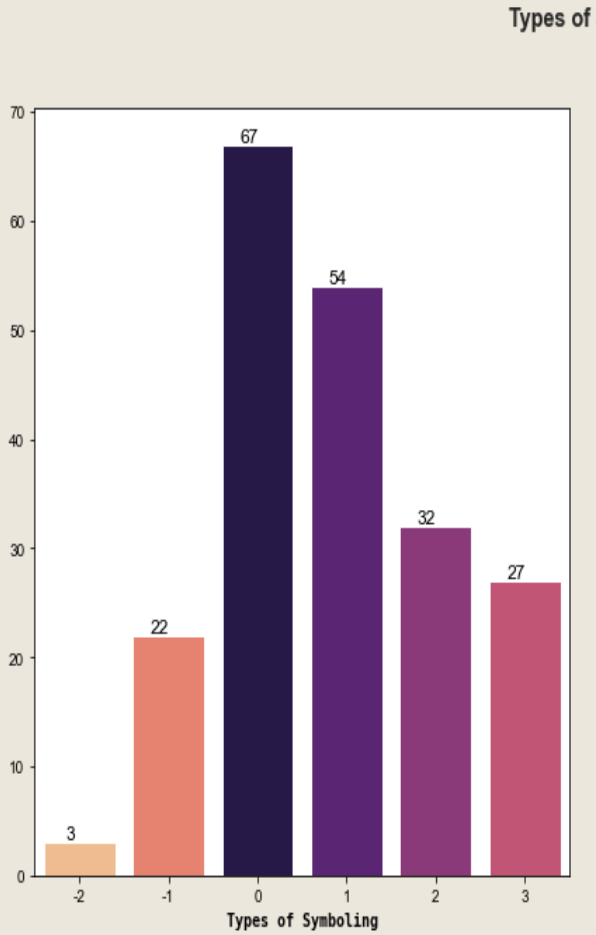
Data Exploration Analysis - dataset

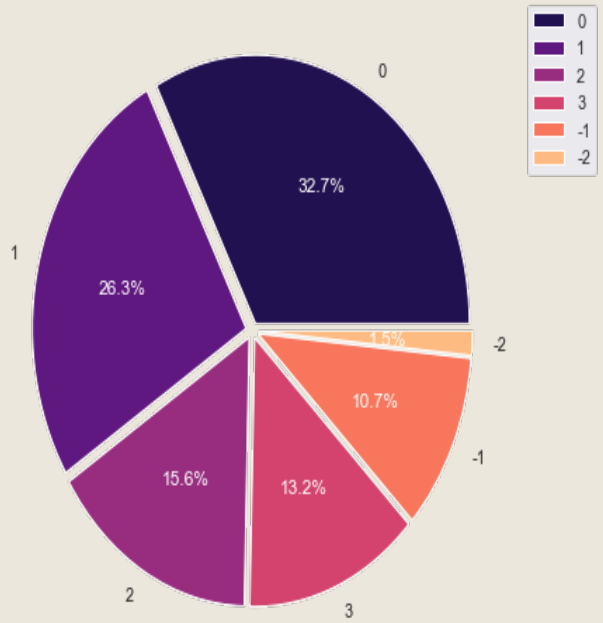
	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
200	201	-1	volvo 145e (sw)	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78
201	202	-1	volvo 144ea	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78
202	203	-1	volvo 244dl	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58
203	204	-1	volvo 246	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01
204	205	-1	volvo 264gl	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78

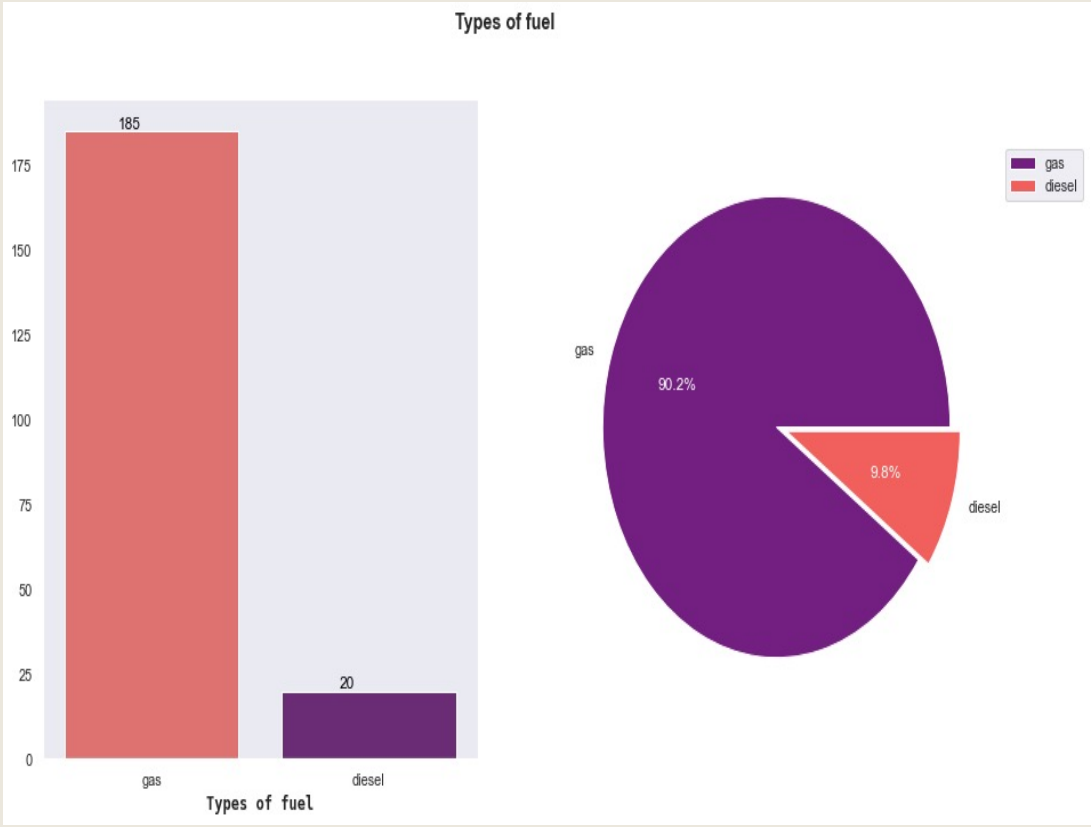
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64
 22  peakrpm           205 non-null    int64
 23  citympg           205 non-null    int64
 24  highwaympg        205 non-null    int64
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

- 205 (rows) observations, 25 (columns) car feature details, 1 target variable – car price
- Includes categorical, numerical features
- Data cleaning:
  - *1. No missing values.*
  - *2. Drop 2 irrelevant columns (car_ID, CarName)*

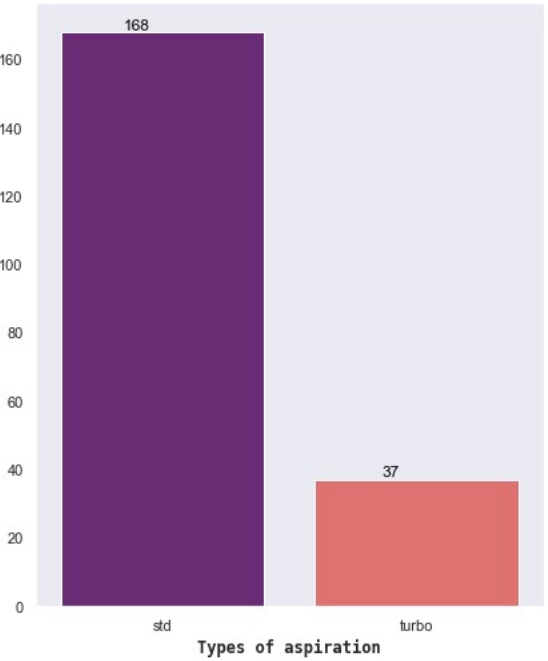
Data Exploration Analysis - Visualization

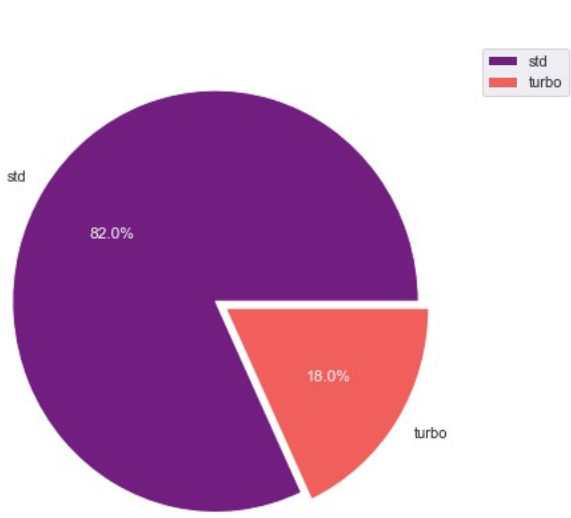




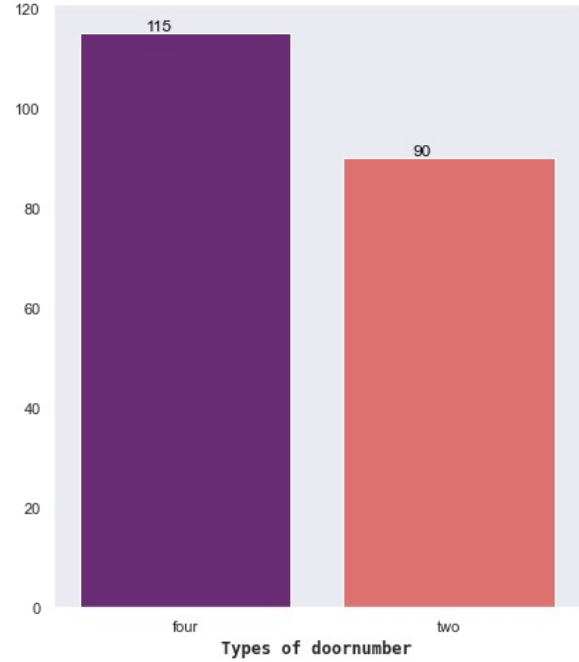


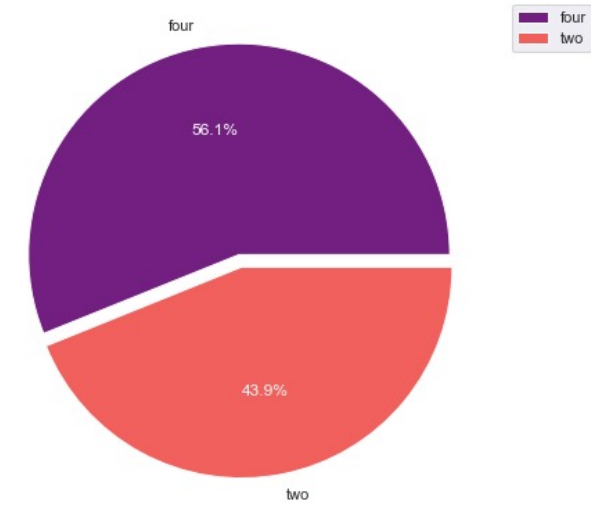
Types of aspiration



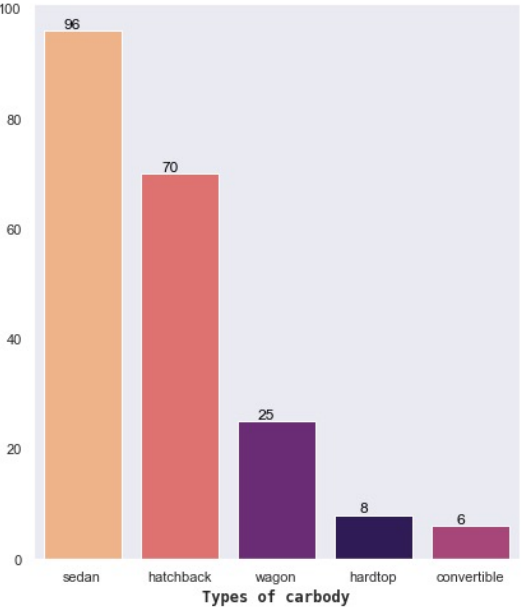


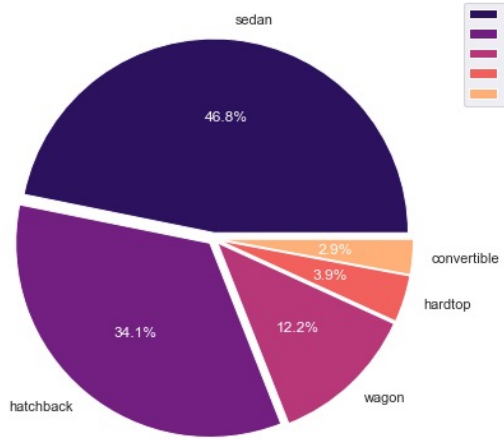
Types of doornumber



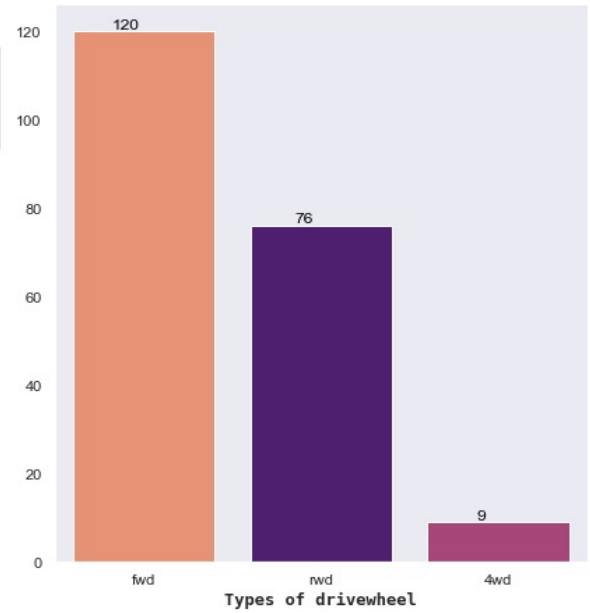


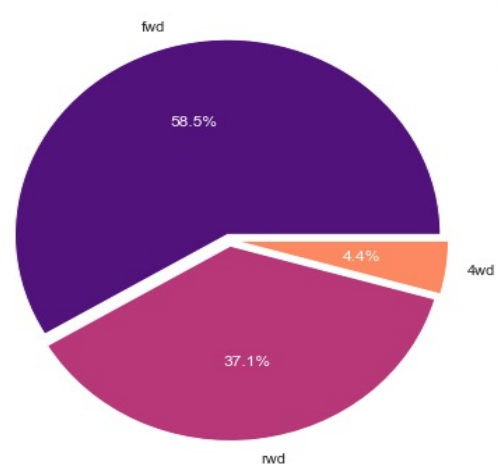
Types of carbody



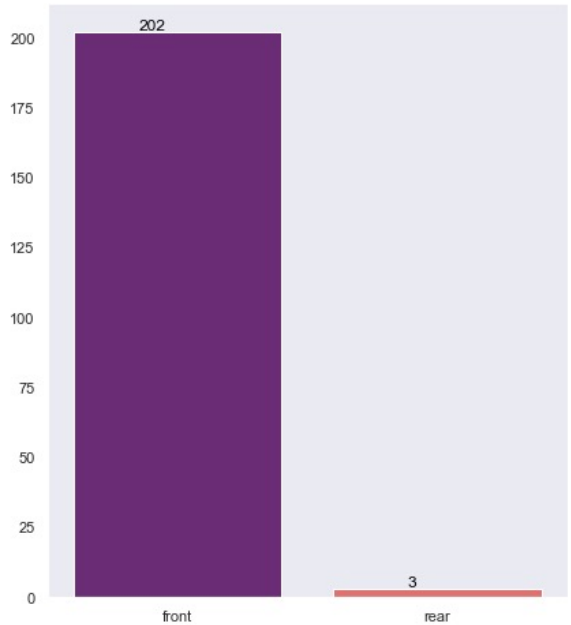


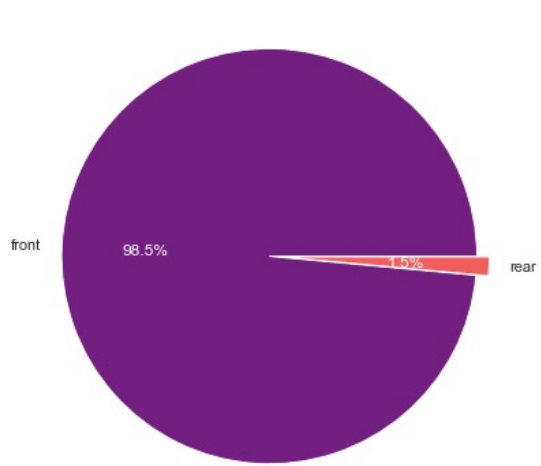
Types of drivewheel



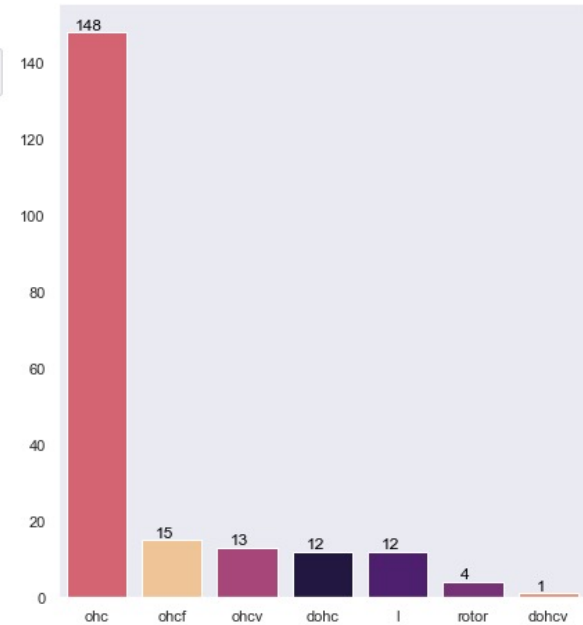


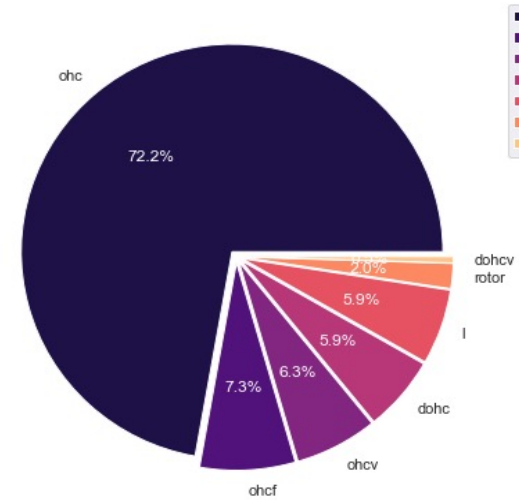
Types of enginelocation



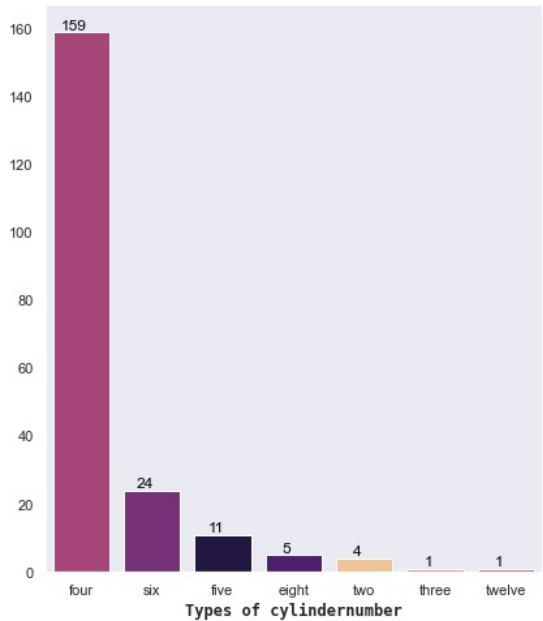


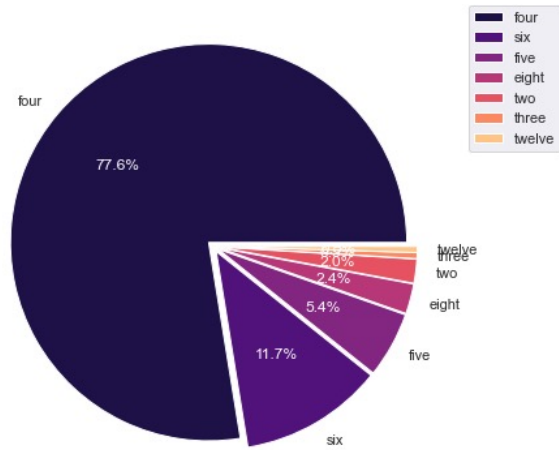
Types of enginetype



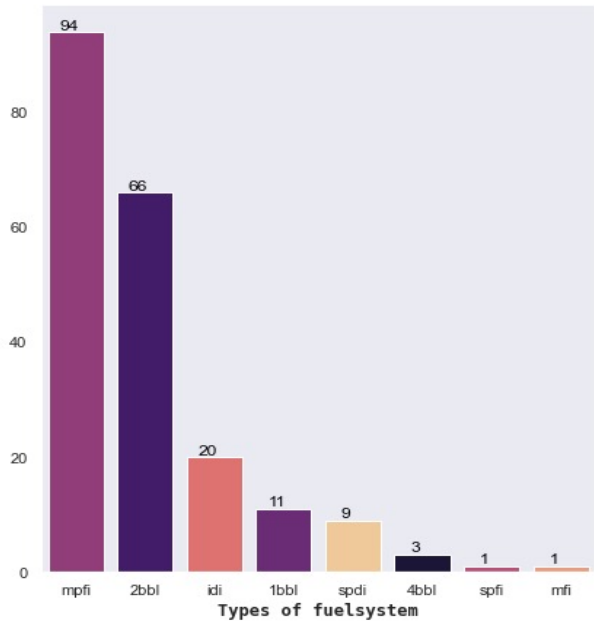


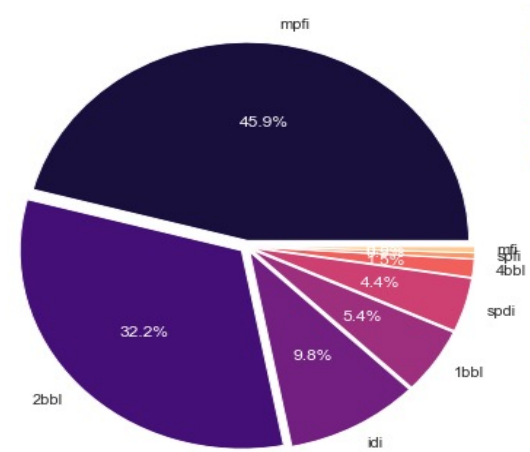
Types of cylindernumber



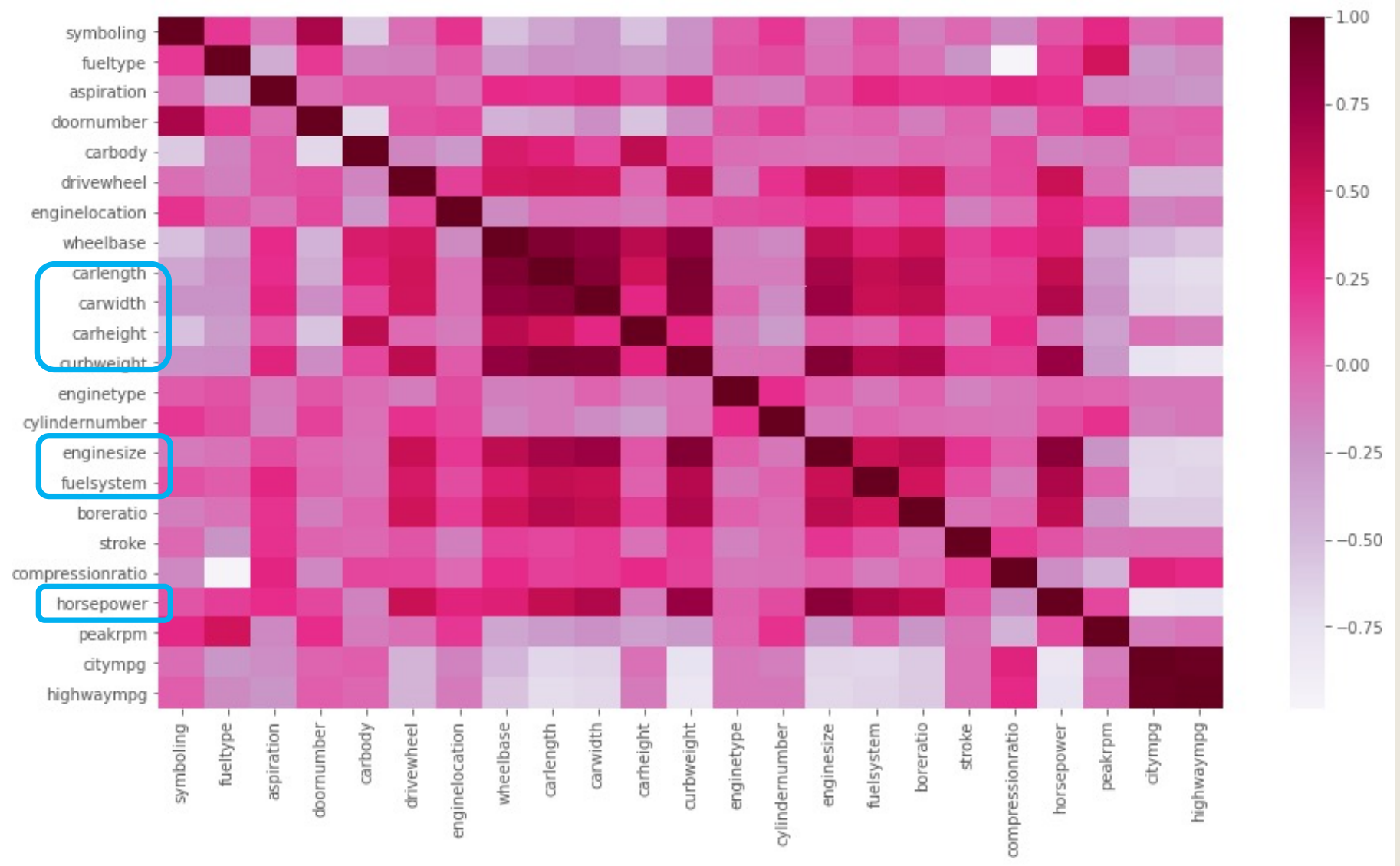


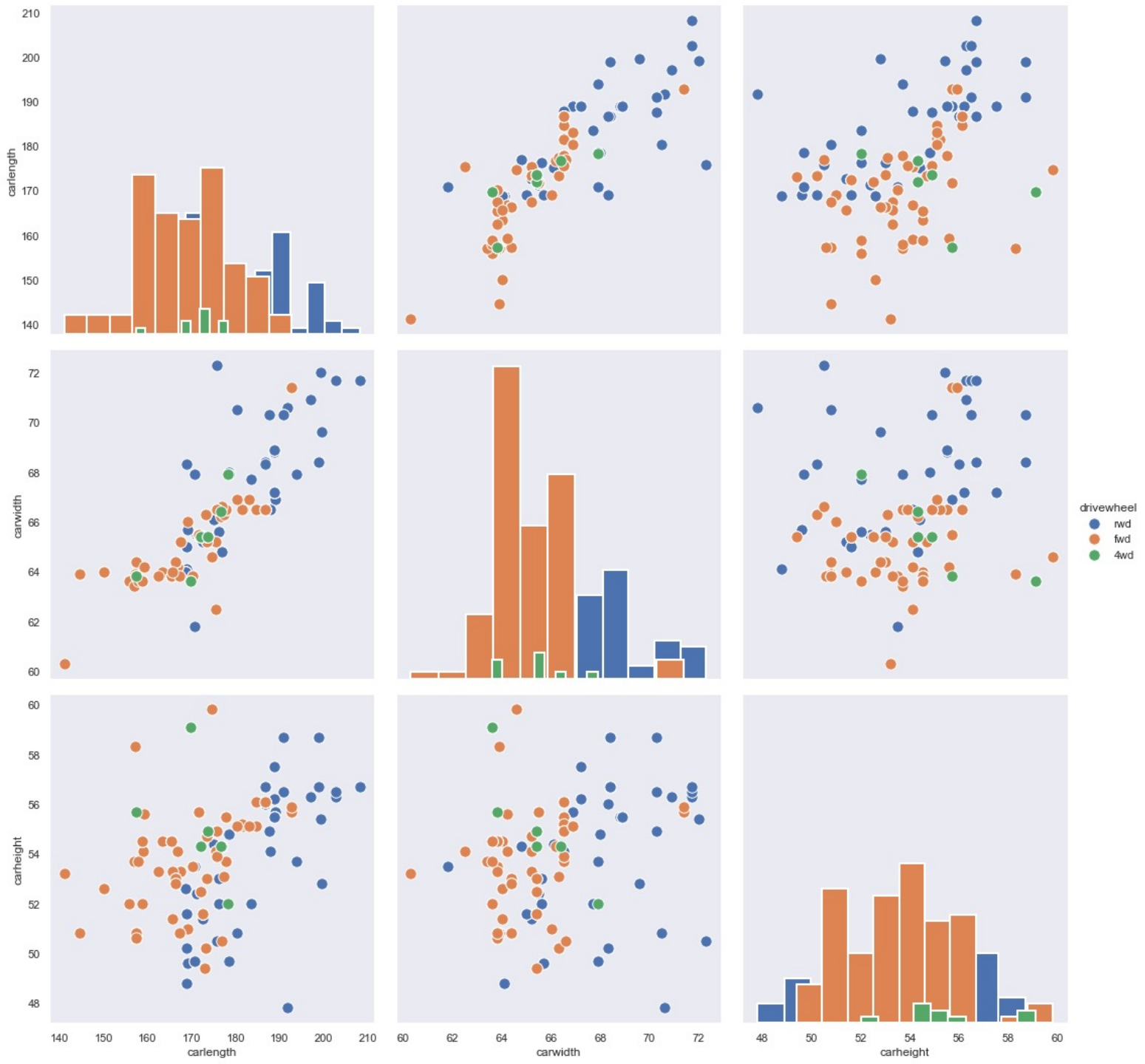
Types of fuelsystem





Correlations between features





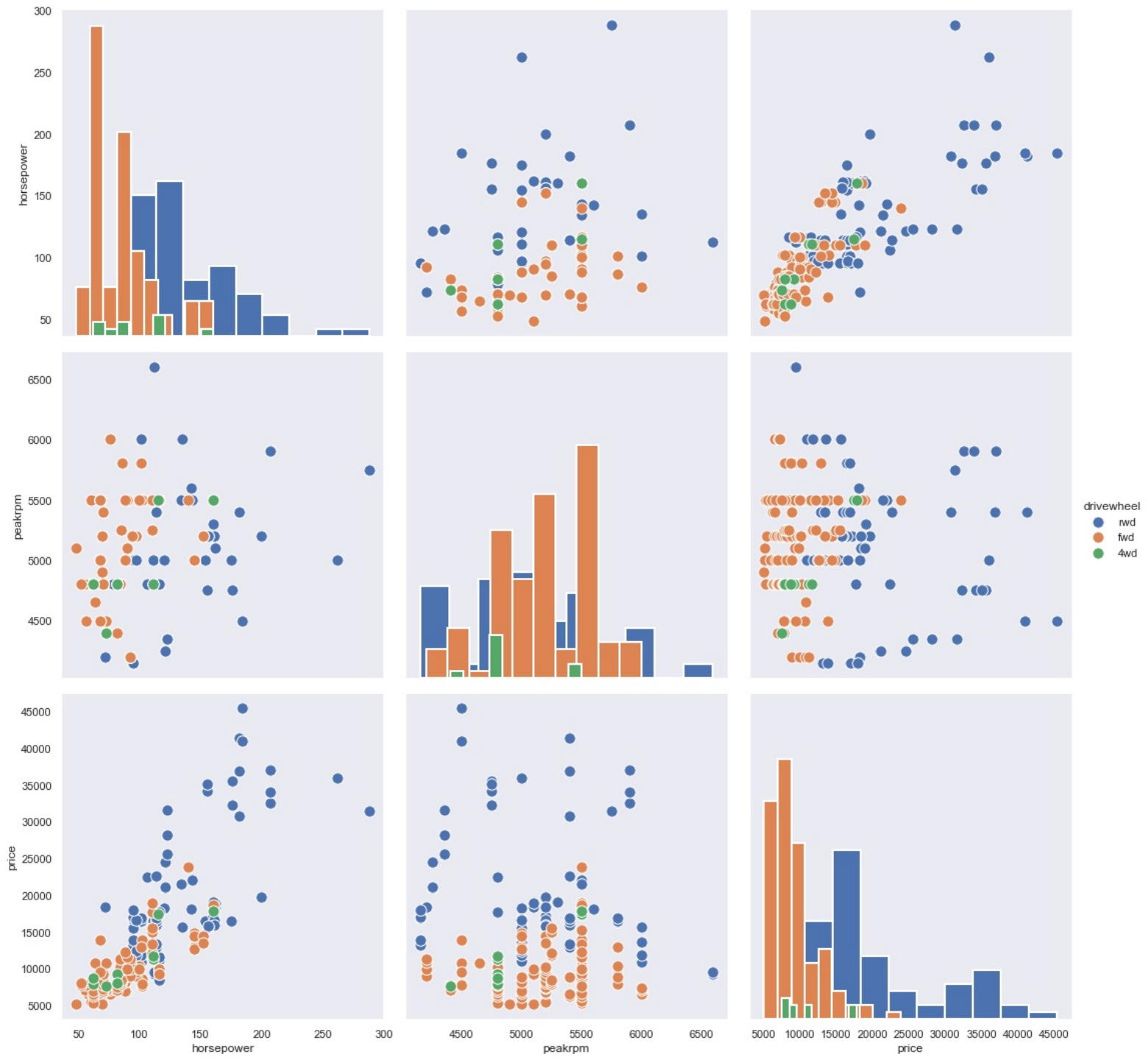
Relationship between

Drive wheel type

vs

car size

(car length, car width, car height)



Distribution between

Drive wheel type

Vs.

Horsepower

Peak rpm

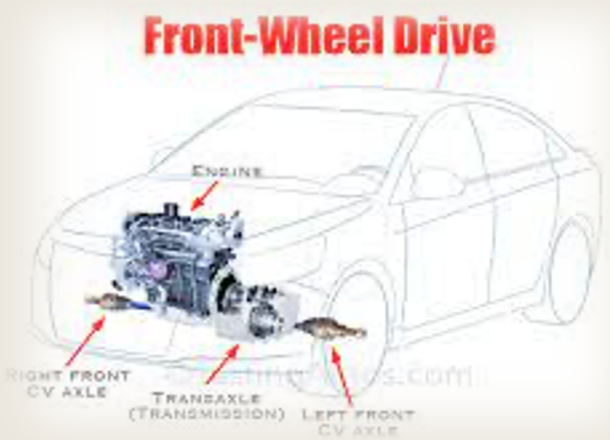
Price

Summary of EDA

- Symboling is close to normal distribution.
- Most fuel type is gas (90.2%), others are diesel.
- Most aspiration of observations are standard (80%), others are turbo.
- Type of door number are four-door (56%), two-door (44%).
- Car body includes :
  - *sedan 47%, hatchback 34%, wagon 12%, hardtop 4%, convertible 3%.*




Summary of EDA



- Drive wheel type :

  fwd (58.5%), rwd (37.1%) , 4wrd (4.4%).

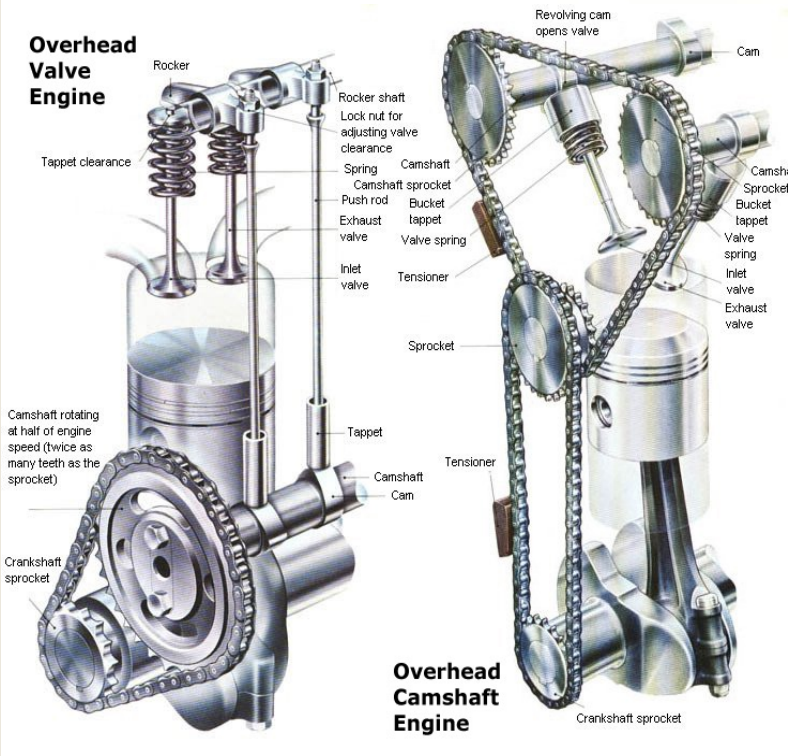
- Most of engine location is front 98.5%.
- Engine type are :

  ohc 72%, ohcf 7%, ohcv 6%, dohc 6%, others.

- Cylinder most distributed in :

  4-cylinders (77%), 5-cylinders (5%) , 6-cylinders (11%).

- Not every feature helps. Need to transform and reduce some, for further steps.



Features engineering (one-hot encoding)

rnumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boreratio	stroke	c
two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	
two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	
two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	
four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	
four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	

drivewheel_1	drivewheel_2	fueltype_0	fueltype_1	aspiration_0
0.0	1.0	0.0	1.0	1.0
0.0	1.0	0.0	1.0	1.0
0.0	1.0	0.0	1.0	1.0
1.0	0.0	0.0	1.0	1.0
0.0	0.0	0.0	1.0	1.0

```
data.columns
```

```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

```
df_ohc.columns
```

```
Index(['symboling', 'wheelbase', 'carlength', 'carwidth', 'carheight',
       'curbweight', 'enginesize', 'boreratio', 'stroke', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price',
       'fuelsystem_0', 'fuelsystem_1', 'fuelsystem_2', 'fuelsystem_3',
       'fuelsystem_4', 'fuelsystem_5', 'fuelsystem_6', 'fuelsystem_7',
       'enginetype_0', 'enginetype_1', 'enginetype_2', 'enginetype_3',
       'enginetype_4', 'enginetype_5', 'enginetype_6', 'cylindernumber_0',
       'cylindernumber_1', 'cylindernumber_2', 'cylindernumber_3',
       'cylindernumber_4', 'cylindernumber_5', 'cylindernumber_6', 'carbody_0',
       'carbody_1', 'carbody_2', 'carbody_3', 'carbody_4', 'drivewheel_0',
       'drivewheel_1', 'drivewheel_2', 'fueltype_0', 'fueltype_1',
       'aspiration_0', 'aspiration_1', 'doornumber_0', 'doornumber_1',
       'enginelocation_0', 'enginelocation_1'],
      dtype='object')
```

	no ohc	ohc
train	0.847043	0.939479
test	0.828886	0.898425

After A/B test, it is not necessary to transform one-hot encoding as it possibly to get overfitting.

Features engineering (label encoding)

carbody	drivewheel
convertible	rwd
convertible	rwd
hatchback	rwd
sedan	fwd
sedan	4wd
...	...
sedan	rwd
sedan	rwd
sedan	rwd
sedan	rwd
sedan	rwd

carbody	drivewheel
0	2
0	2
2	2
3	1
3	0
...	...
3	2
3	2
3	2
3	2
3	2

- As most features of car have just two or no more than 8 categories.
- Label encoding is more easier to interpretate than one-hot encoding.
- After testing in simple linear regression model, it has lower possibilities to overfitting, which is helpful to continue trying other models.

Features engineering – Variance Inflation factors (Stats.)

- *The variance inflation factor (VIF) is a measure for the increase of the variance of the parameter estimates. It is a measure for multicollinearity of the design matrix.*
- *One recommendation is that if VIF is greater than 5, then the explanatory variable is highly collinear with the other explanatory variables, and the parameter estimates will have large standard errors because of this.*

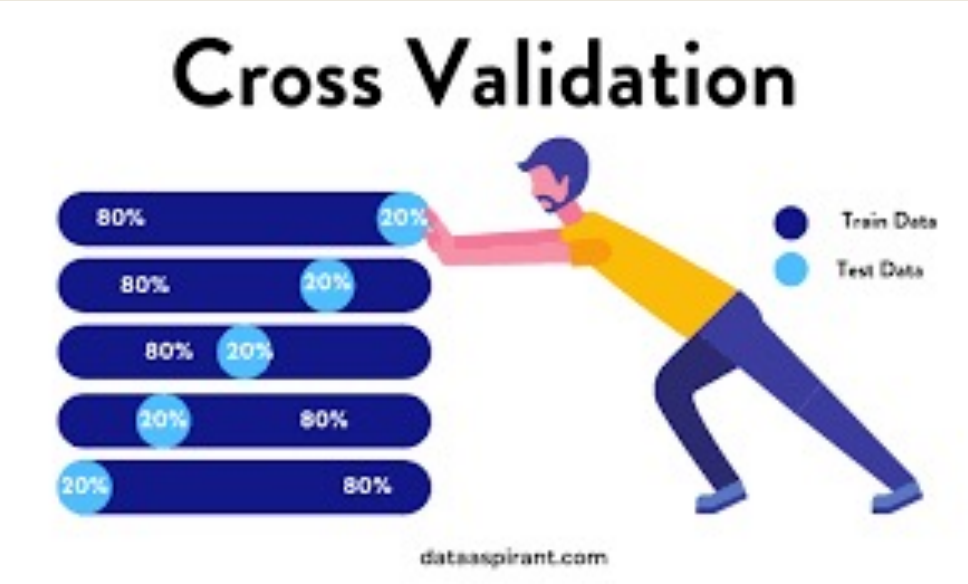
```
symboling              4.088076
fueltype             801.869807
aspiration             3.461189
doornumber             5.000804
carbody               28.635479
drivewheel            17.102140
enginelocation         1.756087
wheelbase           2920.130535
carlength           2360.482629
carwidth            5879.924809
carheight           1255.495605
curbweight           475.771264
enginetype            14.725128
cylindernumber        16.421138
enginesize           137.152566
fuelsystem             8.607494
boreratio            324.326148
stroke               165.096056
compressionratio     575.023405
horsepower           112.264835
peakrpm              303.744416
citympg              522.786019
highwaympg           588.182615
dtype: float64
```

```
vif_data=X_df2
for i in range(10):
    vif_data=MC_remover(vif_data)
vif_data.head()b
```

```
carwidth has been removed
carlength has been removed
wheelbase has been removed
carheight has been removed
highwaympg has been removed
fueltype has been removed
curbweight has been removed
peakrpm has been removed
boreratio has been removed
stroke has been removed
```

- Finally, 10 features whose VIF are greater than 5, and have been removed.

Models - Cross Validation in Linear Regression



5.1 Linear Regression - KFold CV

```
kf = KFold(shuffle=True, random_state=72018, n_splits=3)
kf.split(X_new)
```

```
<generator object _BaseKFold.split at 0x7ff5eda27430>
```

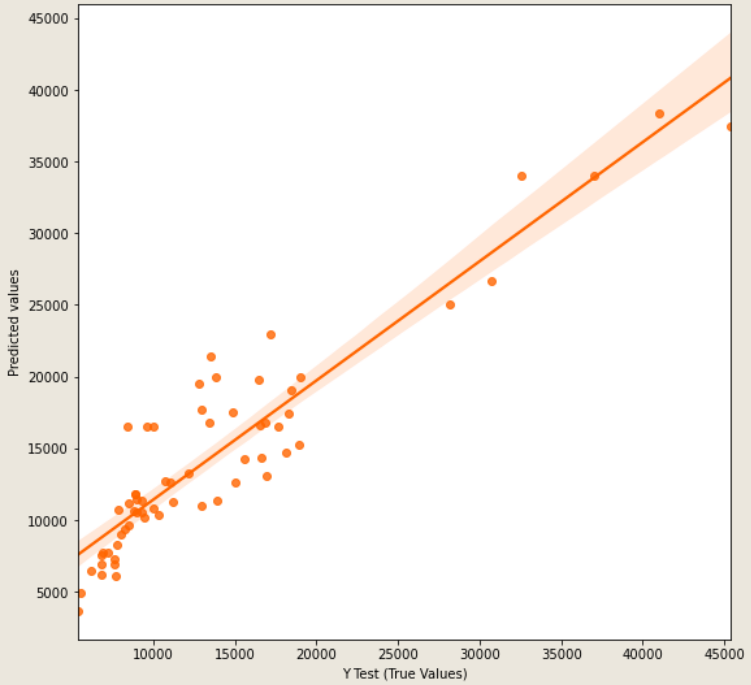
- K-fold = 3 , split train / test dataset into 3 folds.
- Train Linear Regression algorithms.
- Use test-set to predict price, and get a R2 score 0.7499, which is not high.
- Since the data size is pretty small.
- It is not preferable to us K-Fold cross validation instead of directly using Sklearn train_test_split.

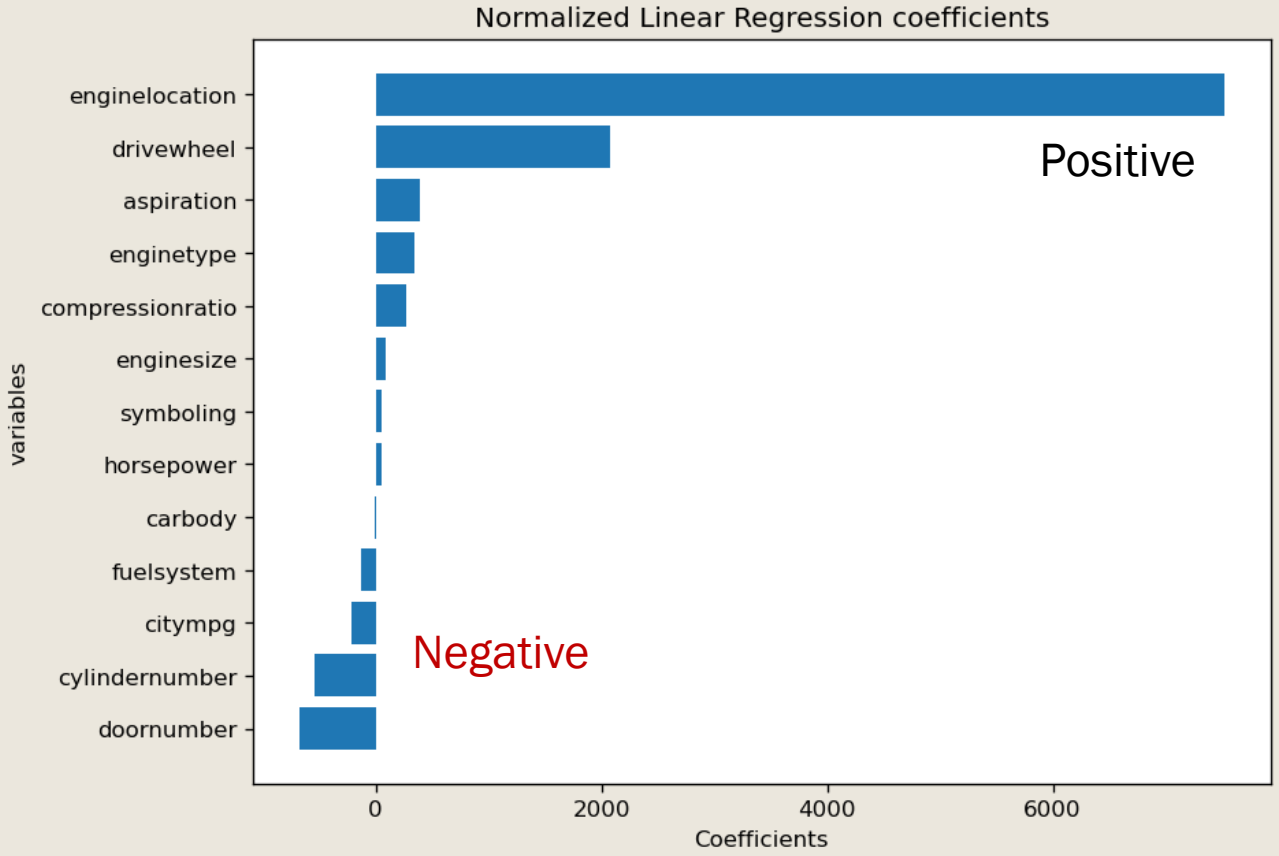
Models - Linear Regression (Baseline)

```
X_train, X_test, y_train, y_test = train_test_split(X_new, y_new, random_state=100, test_size=0.3)
LR = LinearRegression(normalize=True)
LR = LR.fit(X_train, y_train)
y_pred = LR.predict(X_test)
LR_score = r2_score(y_test, y_pred)
print("R2 score:",LR_score)
print("Rmse:",np.sqrt(mean_squared_error(y_test,y_pred)))
```

```
R2 score: 0.8542984272640753
Rmse: 3194.2682245115466
```

- R2 score is much higher (0.8542)
- Its predictability is not bad in fact.
- In this model, features' coefficients:





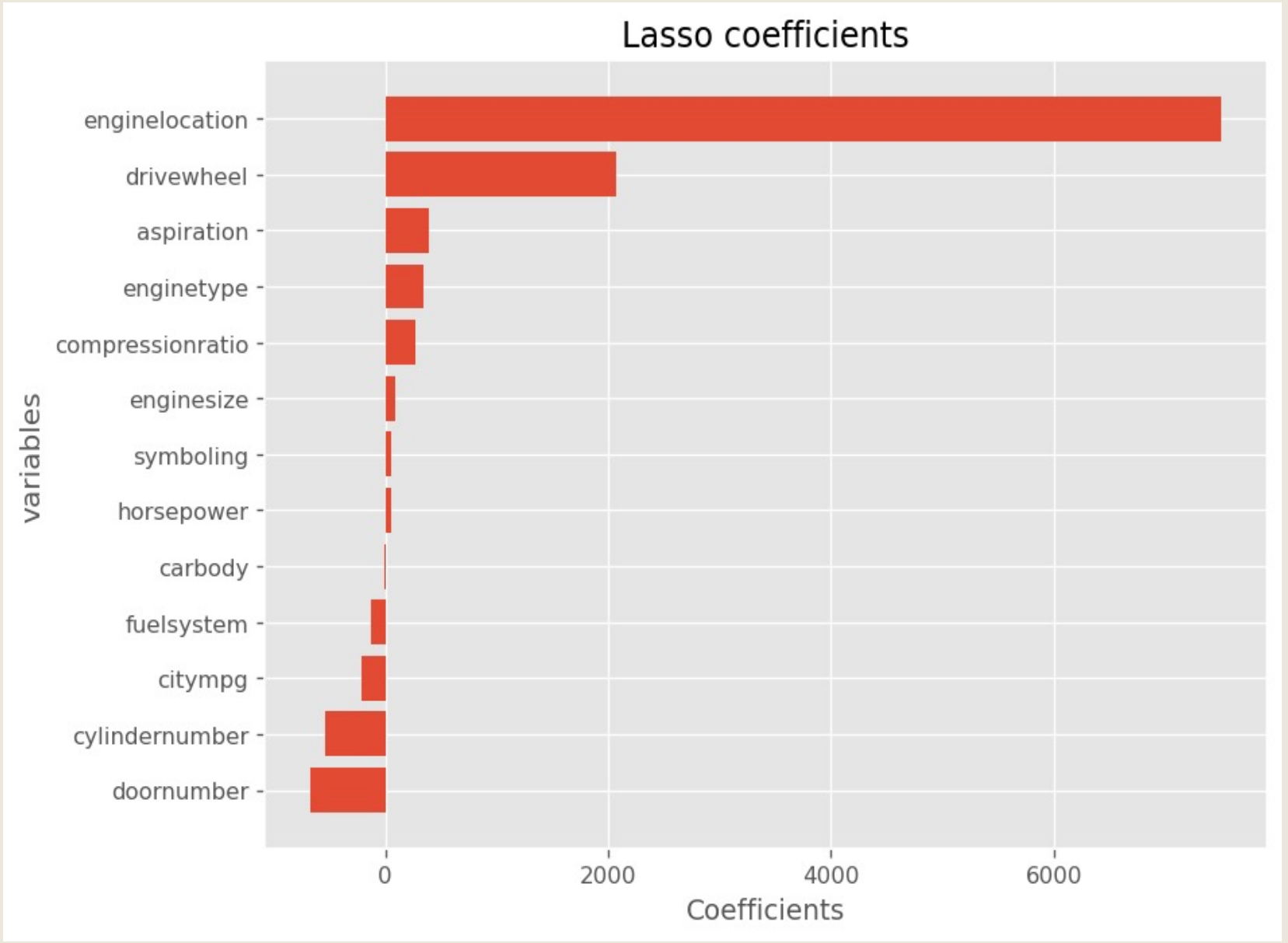
Model – Lasso Regression (L1 regularization penalty)

- After lots of times testing, find a λ (lambda = 0.1), however, score is very close to linear regression 0.85428.
- Higher λ , less complexity of lasso

```
alphas = np.geomspace(1e-5, 2, num=10)

las = Lasso()
scores = []

for alpha in alphas:
    las = Lasso(alpha=alpha, max_iter=100000)

    las = las.fit(X_train, y_train)

    las_pred = las.predict(X_test)

    score = r2_score(y_test, las_pred)

    scores.append(score)
```

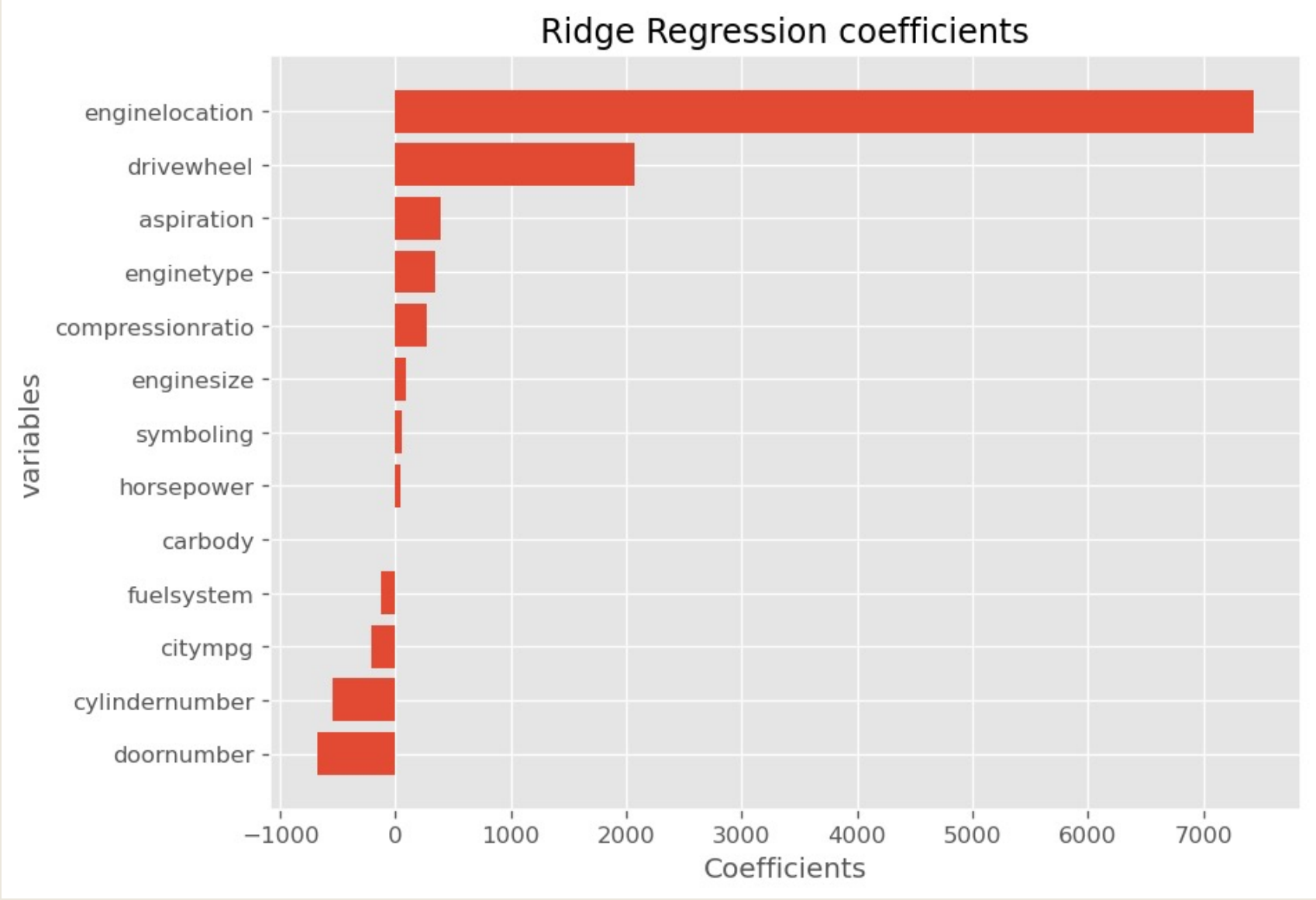
```
list(zip(alphas,scores))
```

```
[(1e-05, 0.8542984262399432),
 (3.881533447356427e-05, 0.8542984232890614),
 (0.0001506630190294667, 0.8542984118341205),
 (0.0005848035476425735, 0.8542983671666096),
 (0.002269345303073466, 0.8542981942045419),
 (0.008810826802697267, 0.8542975221800063),
 (0.034199518933533964, 0.8542949066967883),
 (0.1327465766240115, 0.8542846444588865),
 (0.5152602771881635, 0.8542429842638284),
 (2.0, 0.8540535213159686)]
```



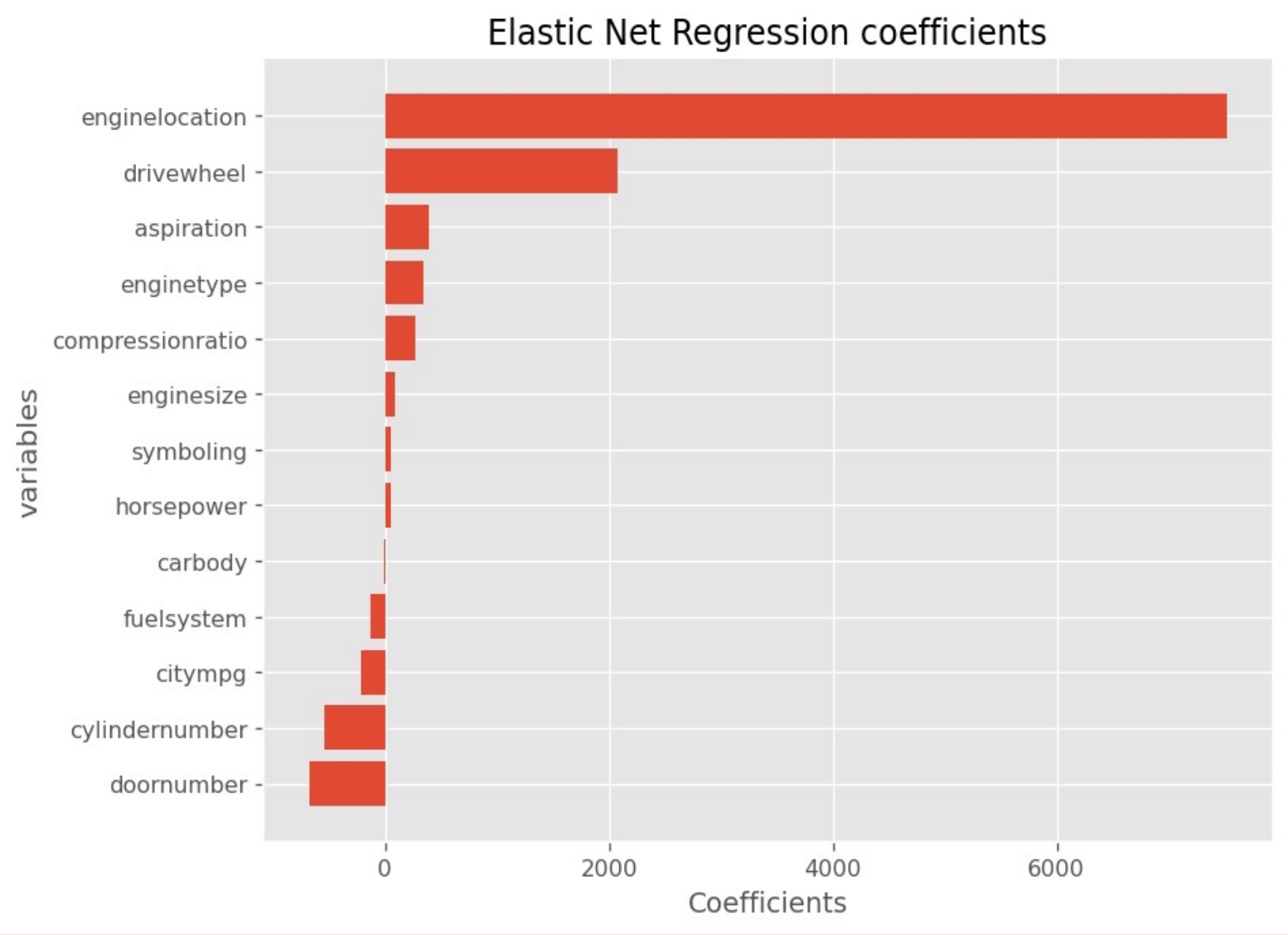
Model –Ridge Regression (L2 regularization penalty)

- Ridge: λ (lambda = 0.01)
- R2 score: 0.854228
- Also very close to baseline (linear)



Model –Elastic Net Regression (Combine L1 & L2 regularization)

- Ridge: λ (lambda = 1e-8)
- R2 score: 0.854298



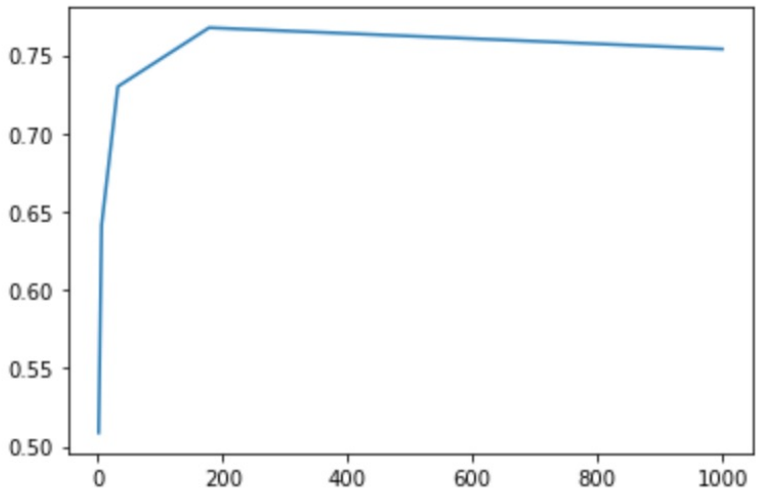
Model –Elastic Net Regression + CV + Polynomial features

```
list(zip(alphas,scores))
```

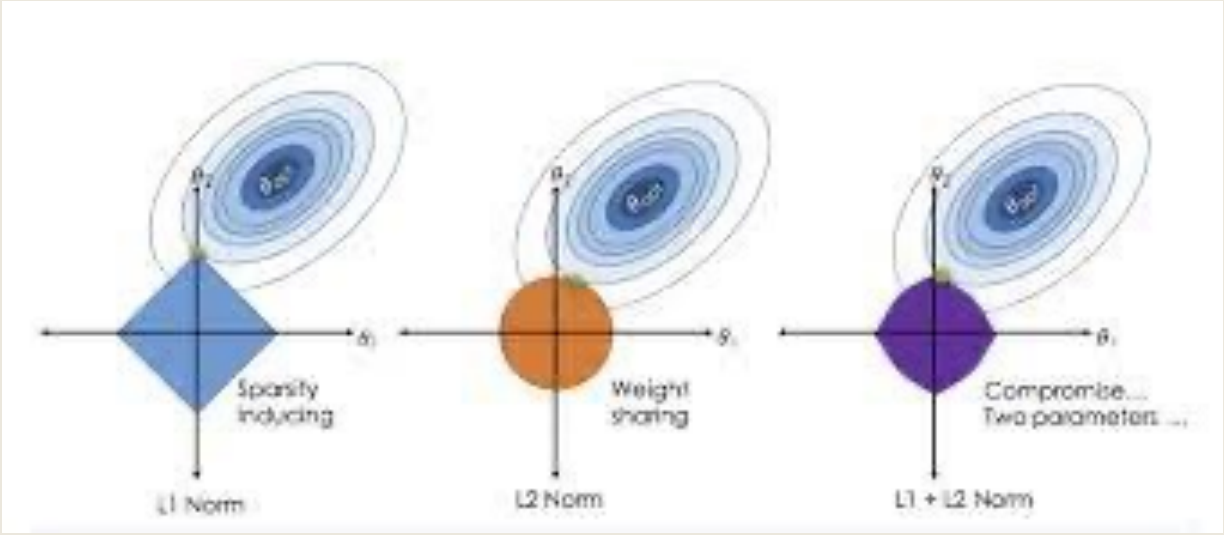
```
[(1.0, 0.5087791679941487),
 (5.623413251903491, 0.6412111704521997),
 (31.622776601683793, 0.7299256491882864),
 (177.82794100389228, 0.7675054311389613),
 (1000.0, 0.7539293320359739)]
```

```
plt.plot(alphas, scores)
```

```
[<matplotlib.lines.Line2D at 0x7ff5ed901af0>]
```

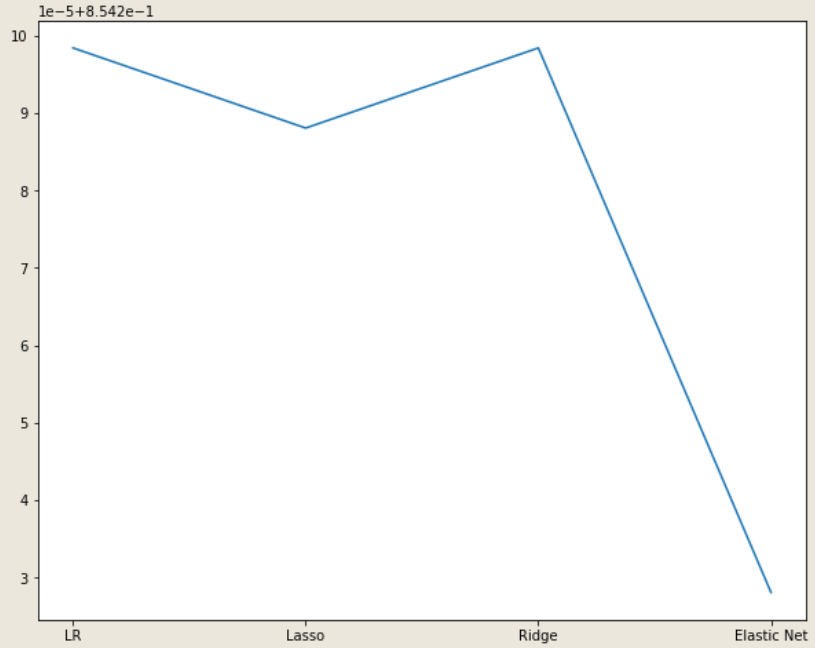


- Try on making higher degrees by adding cross-validation and polynomial features, the scores become lower.
- Increase complexity of model did not help.



Key findings - Models comparison

	test_score
LR	0.854298
Lasso	0.854288
Ridge	0.854298
Elastic Net	0.854228

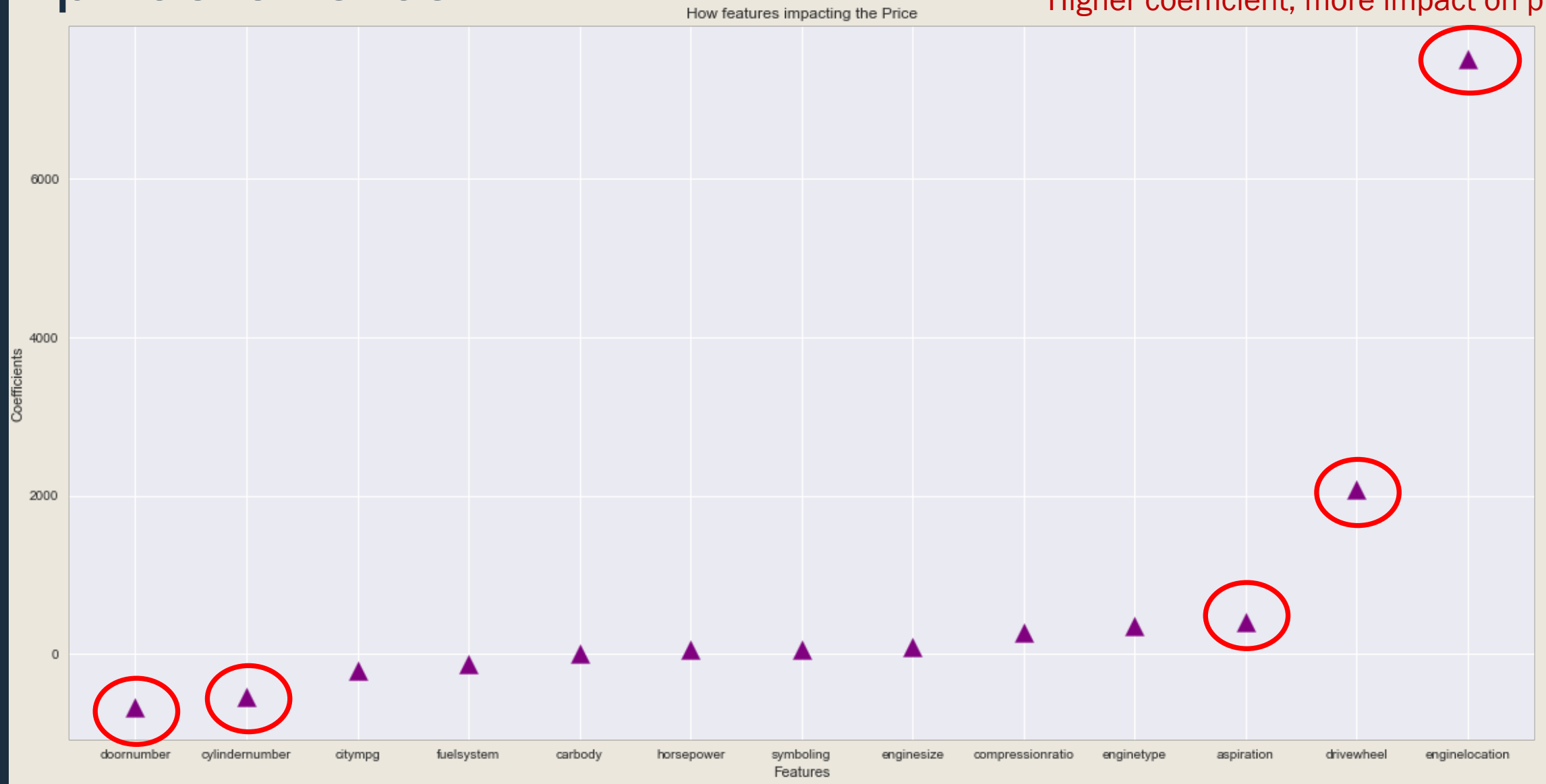


- Comparing to above models, Linear and Ridge regression are more preferable.
- 13 features to train models and data size is pretty small. Due to these reason, increase of complexity, such as adding the L1, L2 regularization penalty, elastic net, polynomial features have very little improvement on testing score.

	Column	Coefficients
2	doornumber	-680.404373
7	cylindernumber	-546.673012
12	citympg	-208.465137
9	fuelsystem	-124.911725
3	carbody	-5.423094
11	horsepower	46.943155
0	symboling	53.597732
8	enginesize	89.928537
10	compressionratio	271.006215
6	enginetype	344.565491
1	aspiration	396.238964
4	drivewheel	2071.816165
5	enginelocation	7518.895684

Key findings – how well variables describe the price of a car

Higher coefficient, more impact on price.



Suggestions for next steps

- Increase the size of dataset, or use similar dataset with bigger size to try.
- More test on regularization problems (Lasso, Ridge, Elastic Net regression algorithms) when have many features to prevent overfitting.
- Using regularization may help to reduce some features directly.
- The features which have high coefficients, I will dig deeper each category of the feature, give more details/insights on helping business-level, factory-level, cost-level decision and suggestions.

THANK YOU FOR WATCHING

Code reference:
https://github.com/apple9855/IBM_ML/blob/main/02f_Course_Project_CarData_Linear.ipynb