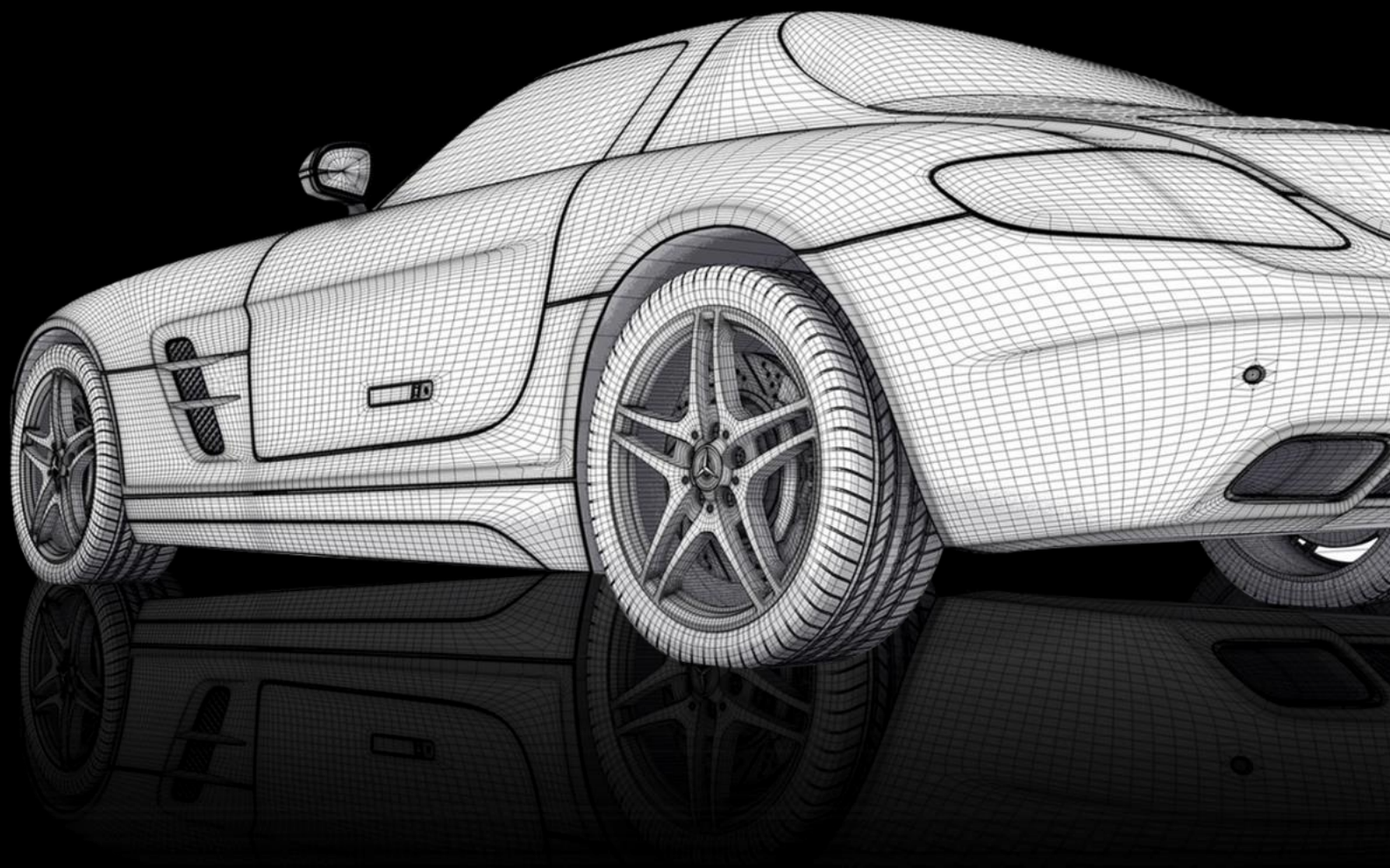
영국 중고차 데이터를 활용한

중고차 가격예측

CONTENTS

1

주제소개




반도체 공급망 마비, 신차 사려면 6개월 대기해야..
중고차 수요 급증
YONHAP NEWS

Kaggle

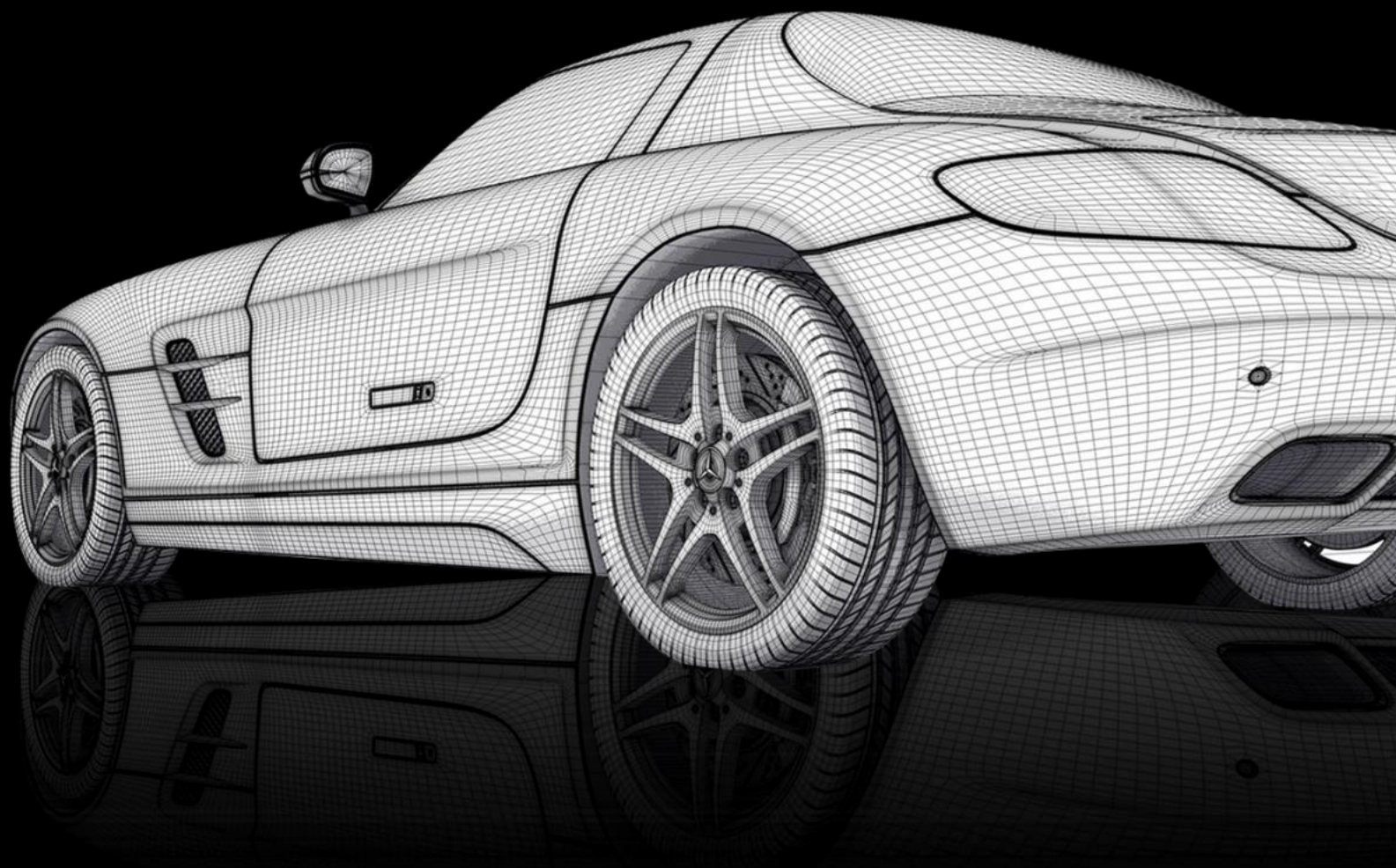
영국 중고차 데이터를 활용한

중고차 가격예측

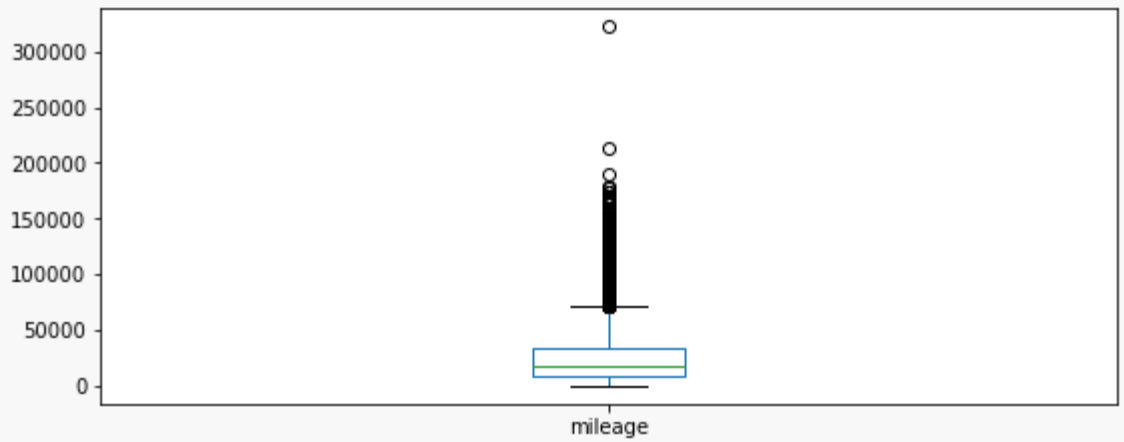
Audi BMW Benz Ford Hyundai Toyota

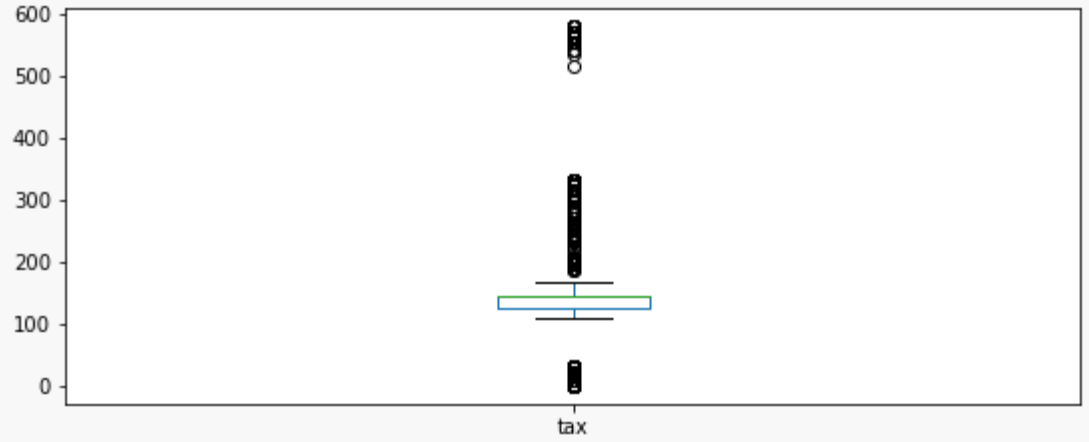
2

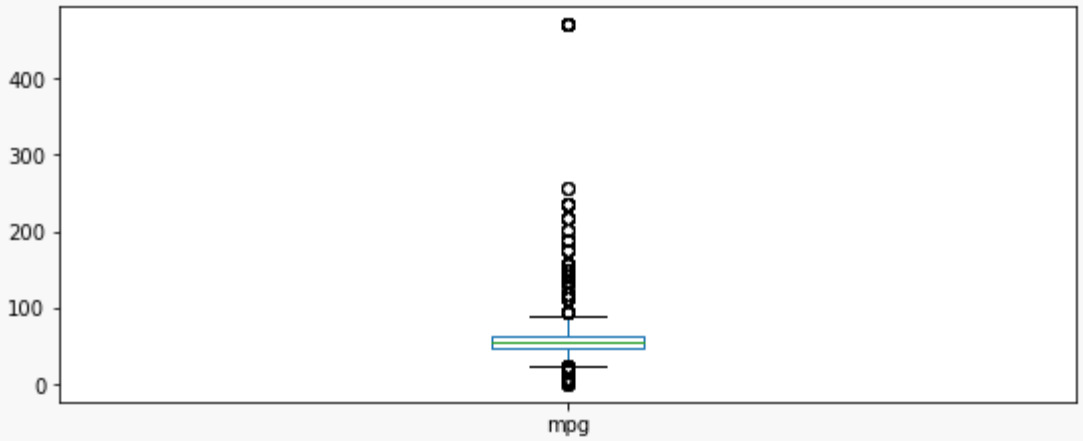
데이터탐색&전처리

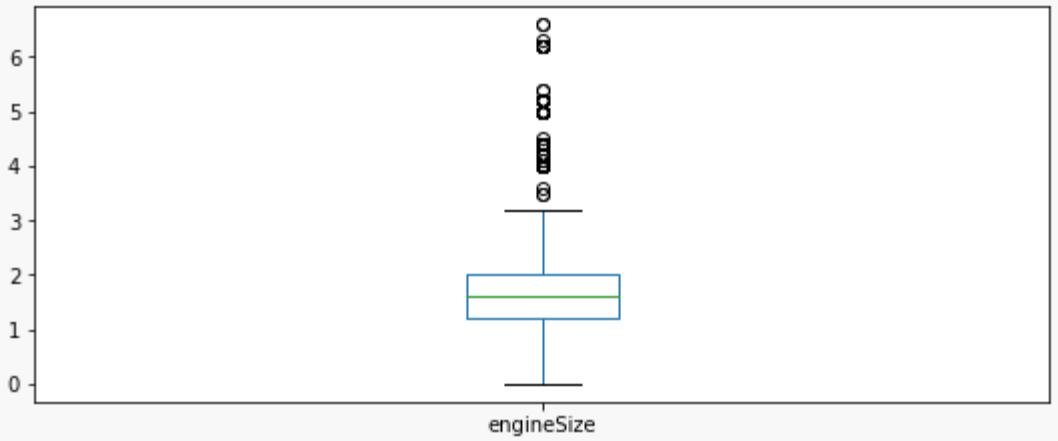


시각화_boxplot


300000
250000
200000
150000
100000
50000
0
mileage

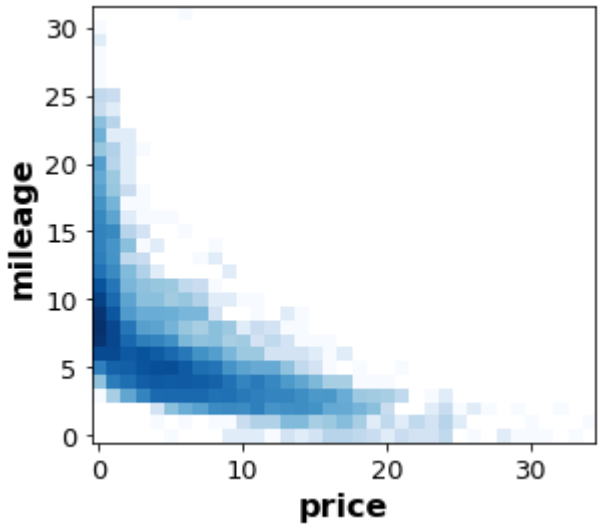

600
500
400
300
200
100
0
tax

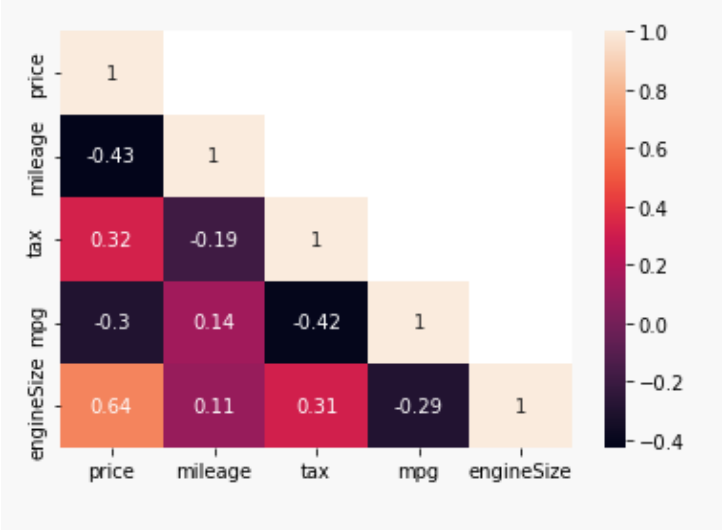

400
300
200
100
0
mpg


6
5
4
3
2
1
0
engineSize

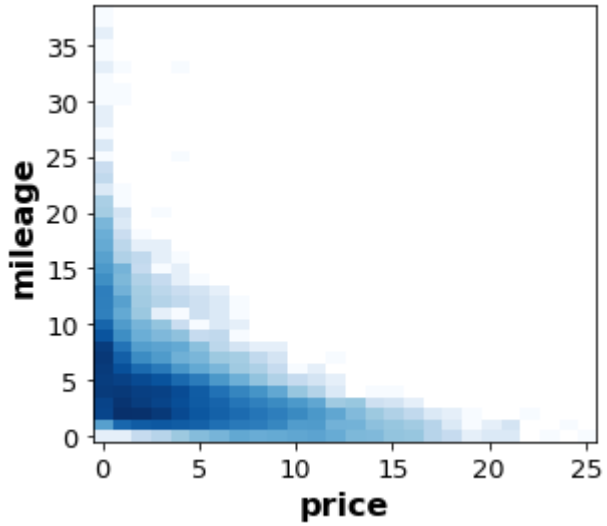
시각화_Lux, Heatmap

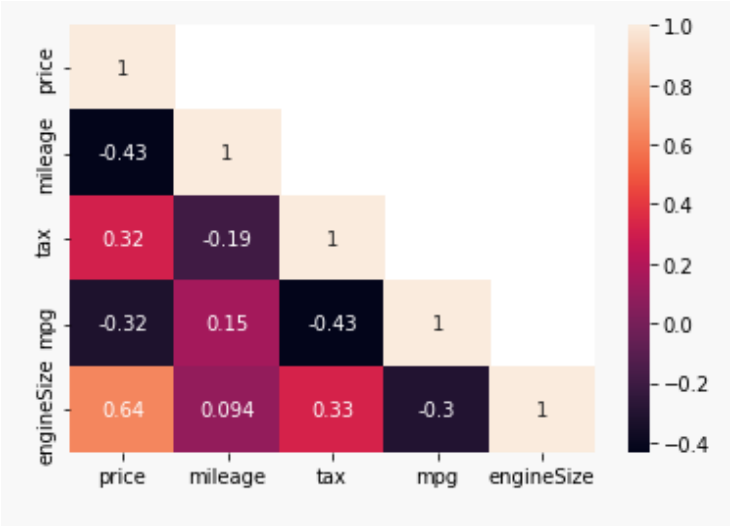
BMW





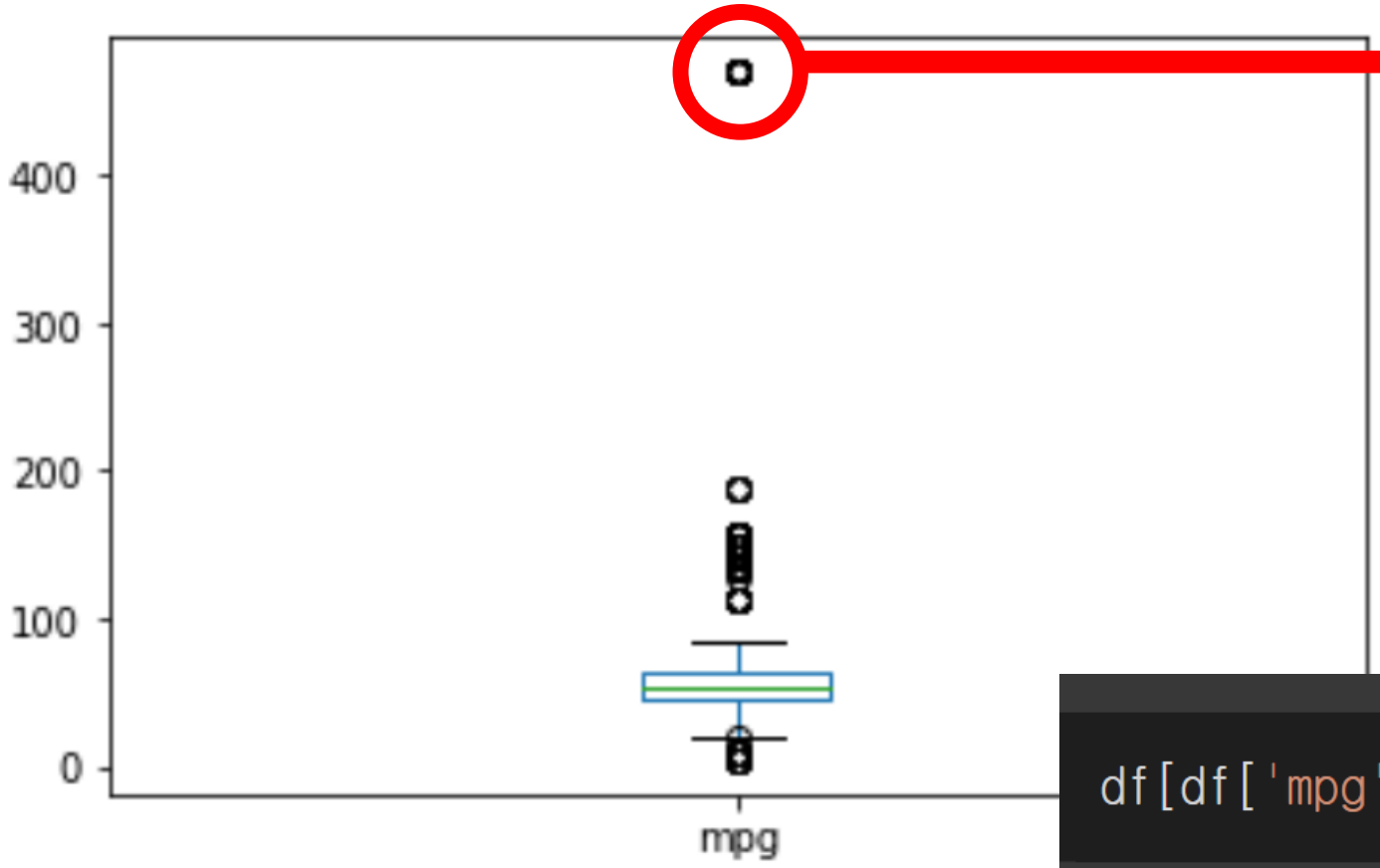
전체데이터





브랜드별로 했을때는 비슷한 클래스이기 때문에 마일이 가장 영향을 주지만, 전체로 놓고 봤을때는 엔진사이즈가 영향을 많이 미침

이상치 처리 _ mpg



470.8

```
df[df['mpg']>400]['model'].value_counts()
```

```
i3      43
Name: model, dtype: int64
```

mpg

w/Range Extender 4dr Hatchback (electric D

111 MPGe*
EPA Combined City/Hwy

N/A City
N/A Highway

* Miles per gallon equivalent. 1 MPGe = 1 mi / 33.7 kWh

4dr Hatchback (electric DD)

118 MPGe*
EPA Combined City/Hwy

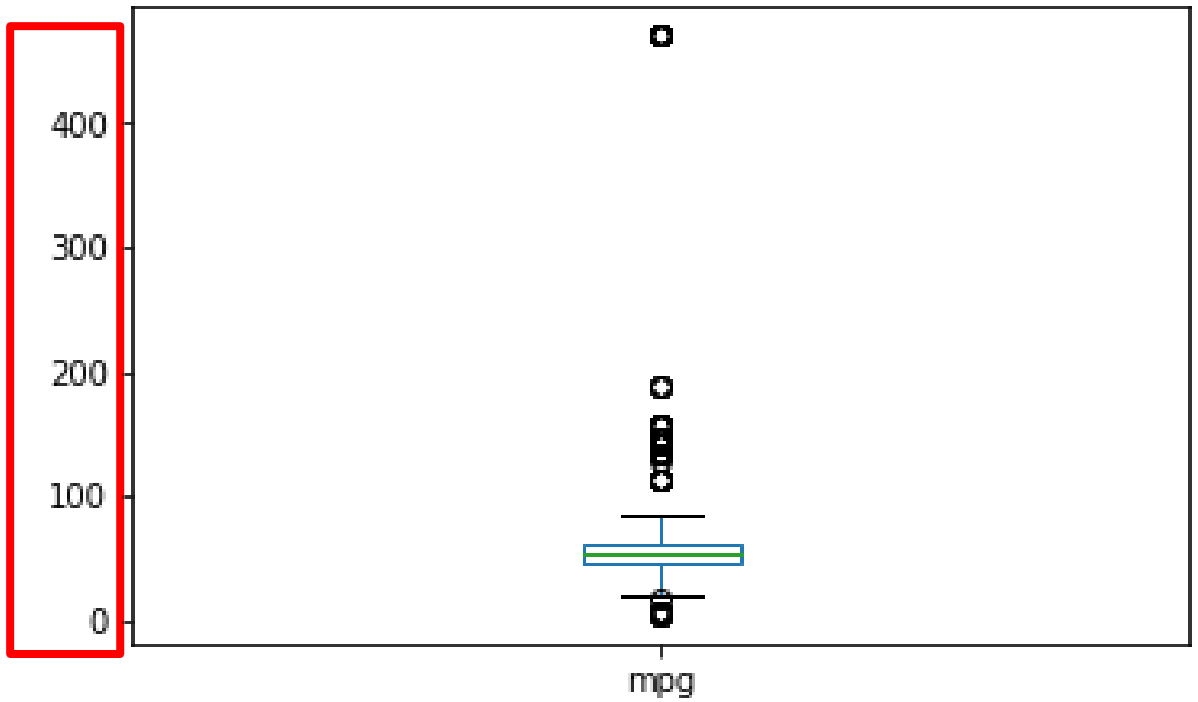
129 City
106 Highway

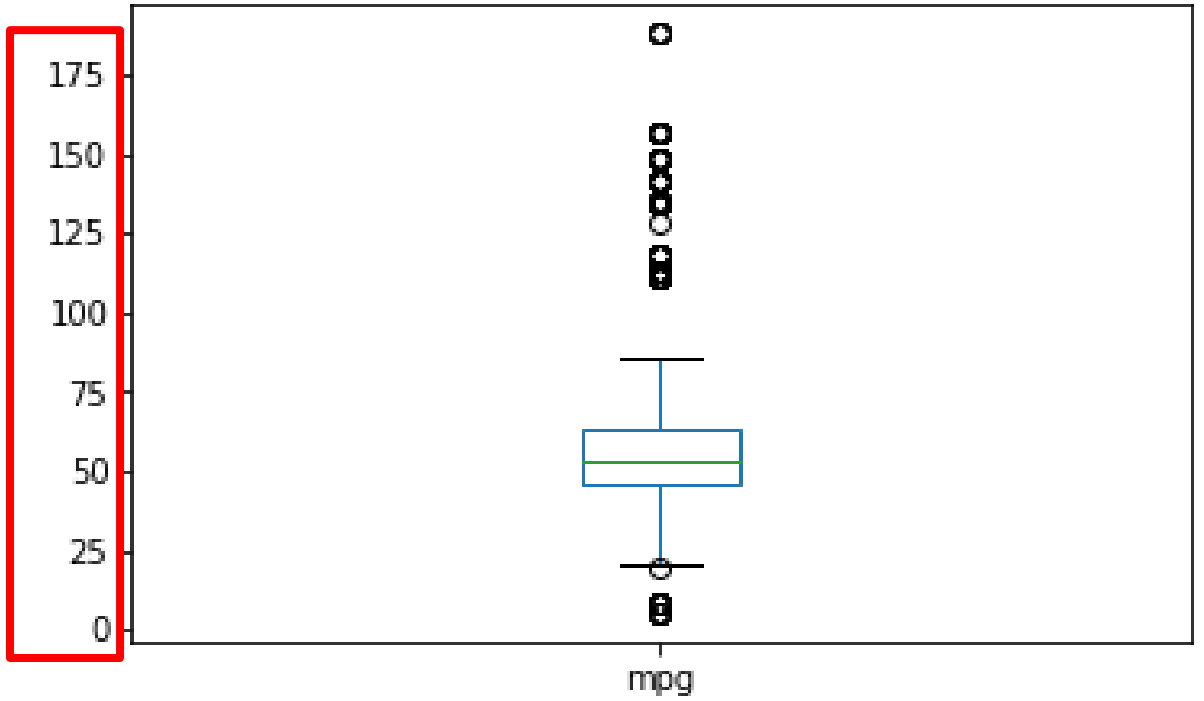
* Miles per gallon equivalent. 1 MPGe = 1 mi / 33.7 kWh

```
[ ]  df['model'] = df[['model']].apply(lambda x: x.str.strip())
```

```
[ ]  df.loc[(df['fuelType']=='Hybrid') & (df['model']=='i3'), ['mpg']] = 111
     df.loc[(df['fuelType']=='Electric') & (df['model']=='i3'), ['mpg']] = 118
     df.loc[(df['fuelType']=='Other') & (df['model']=='i3'), ['mpg']] = 118
```

mpg





결측치 처리

▶ tax, engineSize '0' 으로 된 결측치 존재

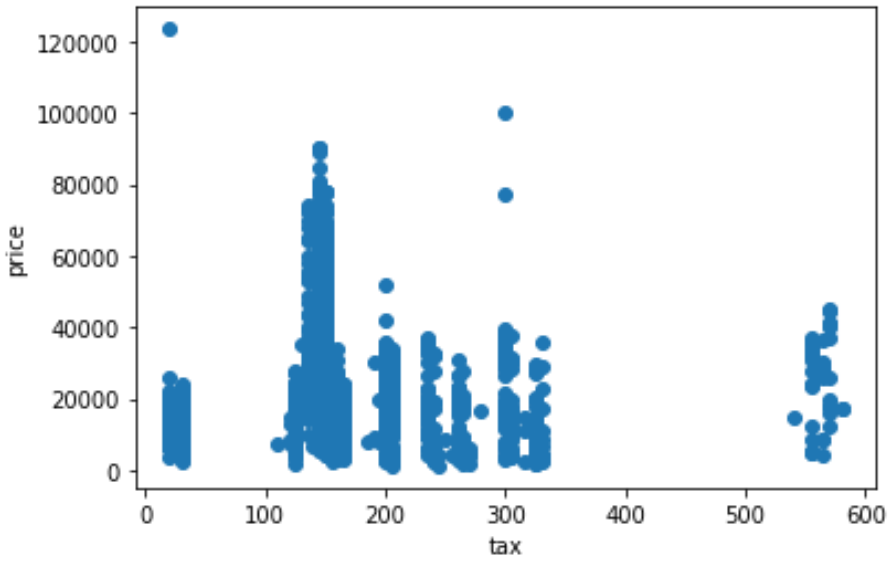
▶ engineSize 결측치를 mode값으로 채워줌 (mode = 2.0)
▶ i3 모델의 engineSize값은 0.6으로 채워줌

```
df['engineSize'] = df[['engineSize']].replace(0,df['engineSize'].mode()[0])
df[df['model'=='i3']]['engineSize'] = 0.6
```

▶ tax의 결측치를 median값으로 채워줌 (median = 145.0)

```
df['tax'] = df['tax'].replace(to_replace=0, value=df['tax'].median())
```

tax 범주화



VED road tax for cars registered 1/3/01-31/3/17

VED Band	CO2 Emissions	Annual rate
A	Up to 100 g/km	£0
B	101-110 g/km	£20
C	111-120 g/km	£30
D	121-130 g/km	£130
E	131-140 g/km	£155
F	141-150 g/km	£170
G	151-165 g/km	£210
H	166-175 g/km	£250
I	176-185 g/km	£275
J	186-200 g/km	£315
K*	201-225 g/km	£340
L	226-255 g/km	£585
M	Over 255 g/km	£600

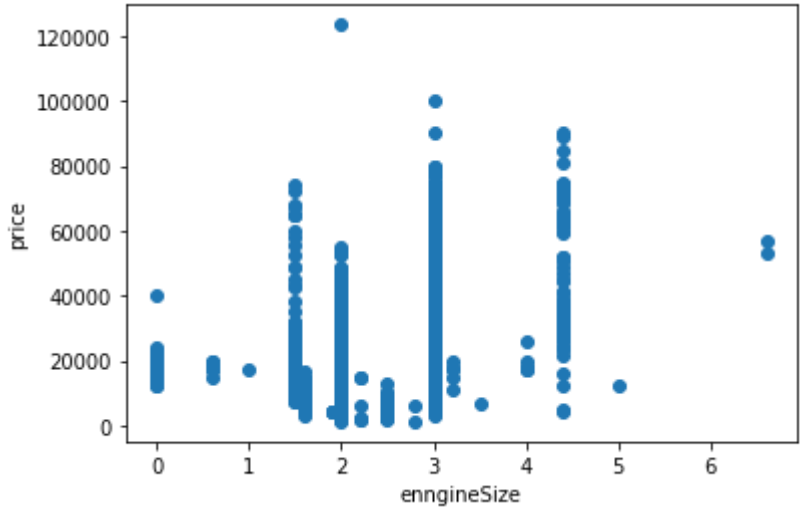
tax 범주화

```
df['tax_cut'] = pd.cut(df.tax,bins=[0,20,30,130,155,170,210,250,275,315,340,600],
                       labels=['A','B','C','D','E','F','G','H','I','J','K'])
```

```
df['tax_cut'].value_counts()
```

```
D    7107
B     989
C     850
E     550
A     485
F     441
I     113
G      87
H      80
K      41
J      38
Name: tax_cut, dtype: int64
```

engineSize 범주화



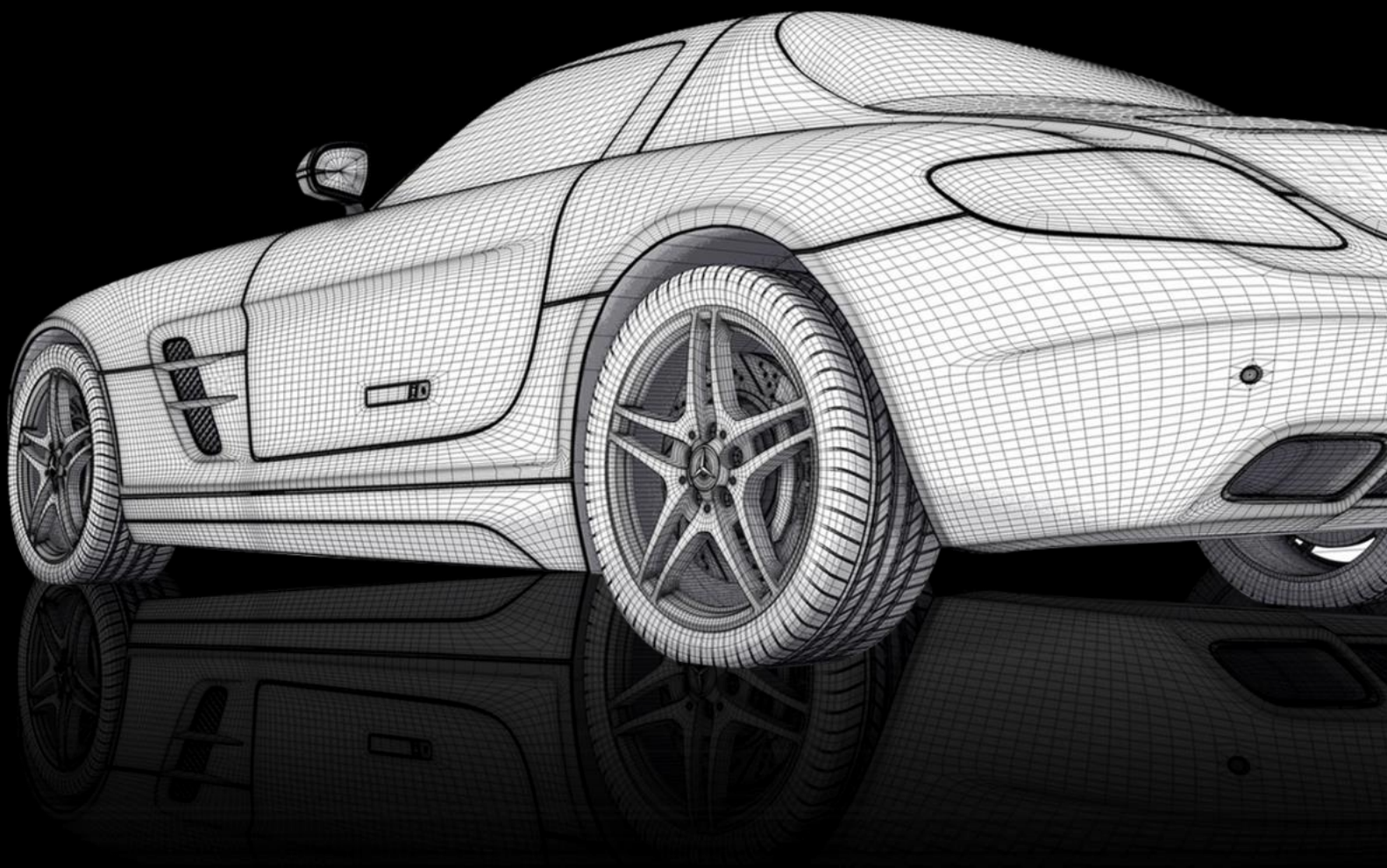

```
df['engineSize'] =  df['engineSize']//1
```

```
df['engineSize'].value_counts()
```

```
2.0    6609
3.0    2464
1.0    1577
4.0      85
0.0      43
6.0       2
5.0       1
Name: engineSize, dtype: int64
```

3

분석기법



pycaret - bmw

```
# Set-up을 바탕으로 각 모델별 성능비교(Hyper parameter defalut 비교하는 것으로 추정 )

top_3_models = compare_models(sort='RMSE', fold=10, n_select=3)
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
0	CatBoost Regressor	1522.7868	5518949.6434	2338.0717	0.9577	0.1183	0.0740	1.4085
1	Light Gradient Boosting Machine	1669.2801	6788256.7837	2595.4119	0.9479	0.1183	0.0808	
2	Extra Trees Regressor	1650.1426	7134279.8798	2657.7005	0.9449	0.1126	0.0779	
3	Random Forest	1669.0292	7207304.4689	2674.1877	0.9448	0.1201	0.0819	
4	Gradient Boosting Regressor	2286.6978	11009394.0653	3309.0082	0.9154	0.1577	0.1161	
5	Extreme Gradient Boosting	2301.6655	11170871.5927	3334.3631	0.9141	0.1593	0.1172	
6	Decision Tree	2156.6657	12654452.0385	3546.6520	0.9027	0.1566	0.1051	
7	Linear Regression	2456.4497	12737814.5184	3554.9189	0.9020	0.1988	0.1227	
8	Bayesian Ridge	2458.7856	12774066.6205	3559.7113	0.9017	0.2213	0.1227	
9	Ridge Regression	2468.7026	12846745.9091	3569.6232	0.9012	0.1993	0.1236	
10	Lasso Regression	2471.8923	12869974.9834	3572.9980	0.9010	0.2178	0.1241	
11	Random Sample Consensus	2416.2088	13277853.0979	3626.9421	0.8980	0.1868	0.1172	
12	Lasso Least Angle Regression	2491.8003	13323723.2343	3634.8811	0.8976	0.1898	0.1244	
13	TheilSen Regressor	3155.0915	33675570.5489	5783.2468	0.7403	0.3292	0.1539	
14	Orthogonal Matching Pursuit	4087.1433	33835380.3546	5803.3239	0.7402	0.3173	0.2085	
15	AdaBoost Regressor	5494.2958	42444842.9675	6512.4294	0.6704	0.3380	0.3335	
16	Elastic Net	4809.9506	51483803.5110	7162.6854	0.6046	0.3414	0.2364	
17	K Neighbors Regressor	5308.2507	62511665.2137	7895.1374	0.5190	0.3104	0.2506	
18	Huber Regressor	5704.4358	82689738.6760	8815.7784	0.3718	0.3929	0.2616	
19	Support Vector Machine	7602.6462	122808383.7082	11064.7667	0.0568	0.4537	0.3872	
20	Passive Aggressive Regressor	45580.1661	7373720794.6146	64667.6323	-61.7725	1.1960	3.5775	
21	Least Angle Regression	117182649608.0783	5183744204506354005363916 8.0000	2276783744830.2344	-353367287305606720.0000	0.3476	13692517.2527	

pycaret - 전체 데이터

```
# Model Compare (Hyper-parameter defalut)

top_3_models = compare_models(sort='RMSE', n_select=3)
```

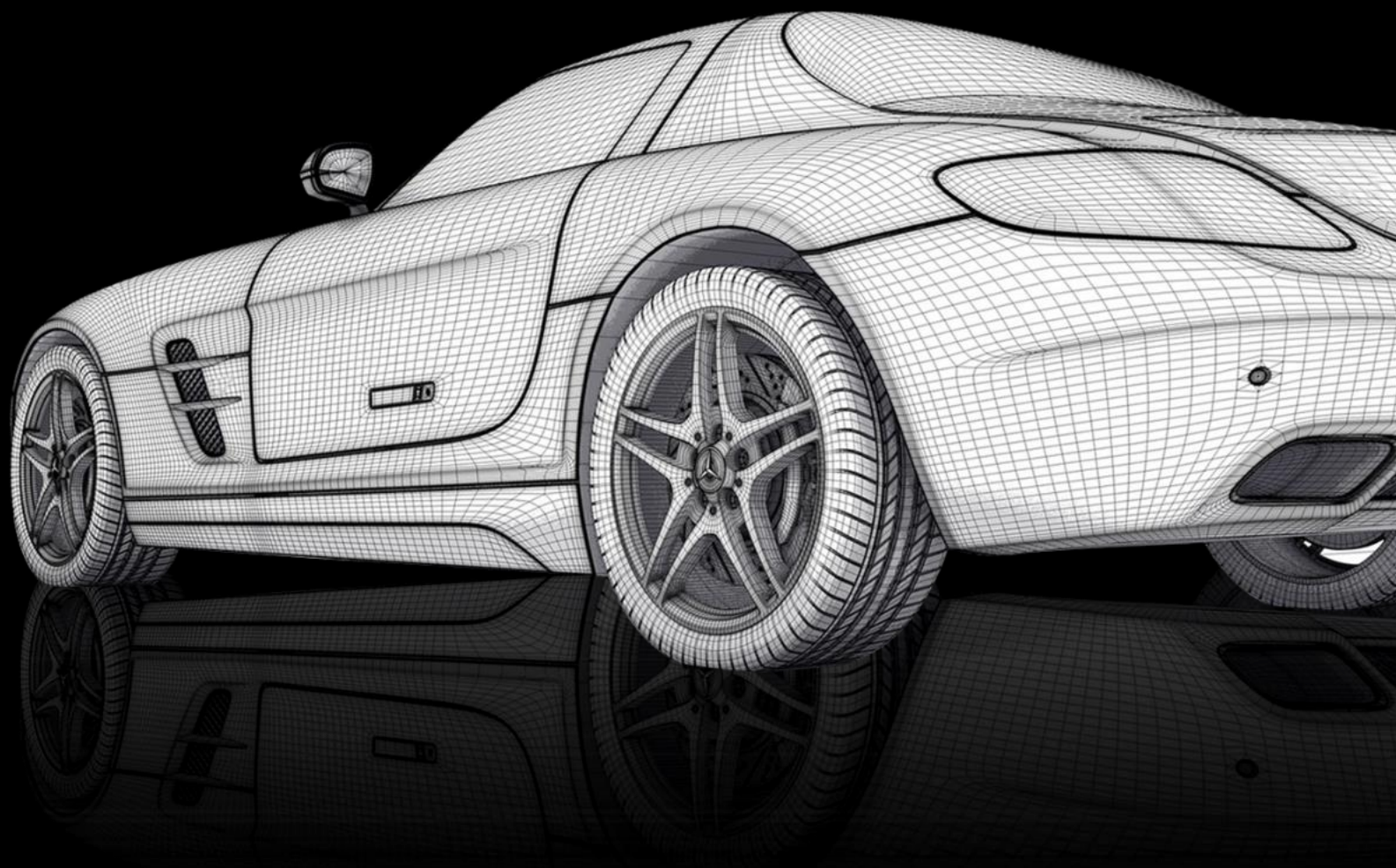
	Model	MAE	MSE	RMSE	R2
0	CatBoost Regressor	1623.0125	8162171.9176	2796.4960	0.9353
1	Random Forest	1763.8195	10031924.5309	3136.0241	0.9197
2	Extra Trees Regressor	1664.8857	11031115.0128	3199.7367	0.9147
3	Light Gradient Boosting Machine	1836.6471	12912170.9411	3515.3256	0.8976
4	Gradient Boosting Regressor	2425.3739	15605514.2896	3915.1124	0.8738
5	Extreme Gradient Boosting	2452.6370	15706839.5729	3932.7854	0.8730
6	Lasso Regression	2490.2739	16528768.0304	3991.0081	0.8706
7	Bayesian Ridge	2490.9954	16698615.4609	4013.4479	0.8691
8	Linear Regression	2487.4461	16709995.1328	4014.3899	0.8690
9	Lasso Least Angle Regression	2542.8747	17252634.9943	4088.1313	0.8644
10	Ridge Regression	2522.4063	17303988.3813	4095.9778	0.8646
11	Random Sample Consensus	2332.9523	17933085.5287	4134.0736	0.8614
12	Decision Tree	2308.5259	17845189.3085	4161.5708	0.8567
13	Orthogonal Matching Pursuit	3130.3979	23118863.5236	4763.6515	0.8159
14	TheilSen Regressor	2637.0146	25307697.5758	4931.1105	0.8051
15	AdaBoost Regressor	4911.9331	39963575.2996	6311.5118	0.6712
16	Elastic Net	4109.2884	48767249.4652	6927.1162	0.6136
17	Huber Regressor	4664.0579	59808271.4226	7504.8630	0.5375
18	K Neighbors Regressor	6422.7054	87620465.7851	9321.0235	0.2970
19	Support Vector Machine	7417.5045	125494429.7446	11148.1057	-0.0032
20	Passive Aggressive Regressor	24122.4406	3067656997.7203	33969.5111	-21.8140

Deep Learning

```
[ ]  # x_train_transformed.shape == (x,157)

     model = models.Sequential()

     model.add(layers.Dense(input_dim=157, units=512, activation=None, kernel_initializer=initializers.he_uniform())) # he-uniform initialization
     model.add(layers.BatchNormalization()) # Use this line as if needed
     model.add(layers.Activation('relu')) # elu or relu (or layers.ELU / layers.LeakyReLU)

     model.add(layers.Dense(units=256, activation=None, kernel_initializer=initializers.he_uniform()))
     model.add(layers.BatchNormalization()) # Use this line as if needed
     model.add(layers.Activation('relu'))

     model.add(layers.Dense(units=32, activation=None, kernel_initializer=initializers.he_uniform()))
     model.add(layers.BatchNormalization()) # Use this line as if needed
     model.add(layers.Activation('relu'))
     model.add(layers.Dropout(rate=0.5)) # Dropout-layer

     model.add(layers.Dense(units=1, activation=None))
```

4

모델링 결과



머신러닝

bmw

	모델링		Hyperparameter Tuning	
	RMSE	R2_Score	RMSE	R2_Score
Catboost	3033.03	0.9297	2978.09	0.932
LightGMB	2986.92	0.93	2950.59	0.95
RandomForest regressor	3121.67	0.93	3085.07	0.93

전체 데이터의 50%

	Hyperparameter Tuning	
	RMSE	R2_Score
Catboost	2610.71	0.9475
RandomForest regressor	3305.47	0.91
ExtraTree regressor	3176.48	0.92
Top 3 모델 Blending	2721.02	0.94

전체데이터

Hyperparameter Tuning

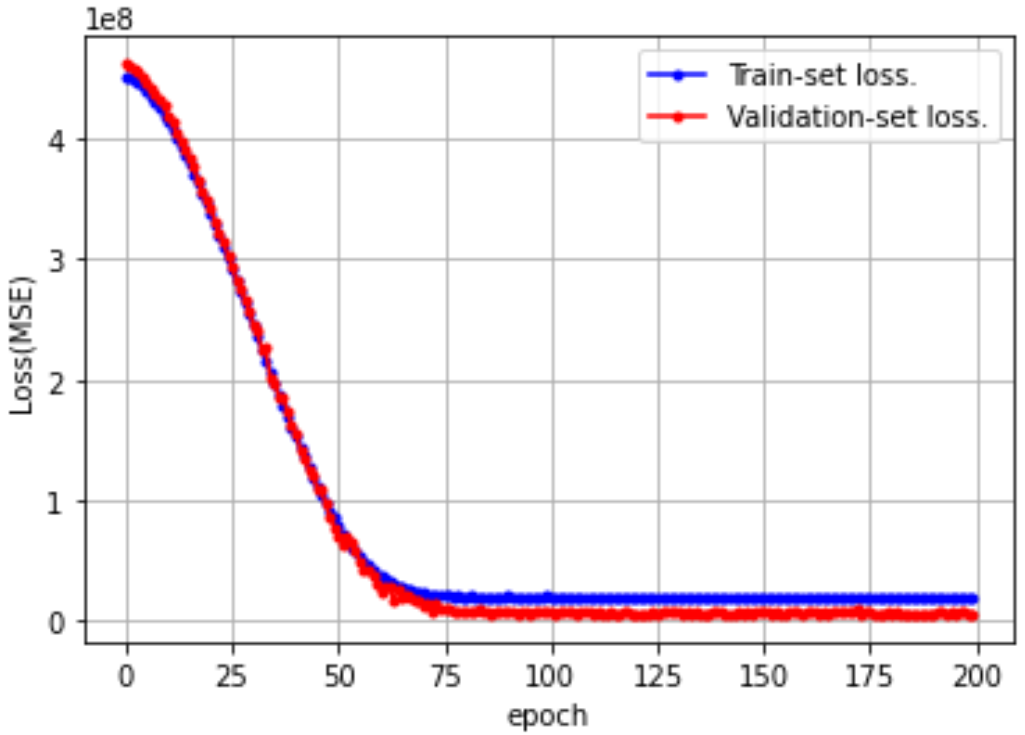
	RMSE	R2_Score
Catboost	2396.98	0.9525

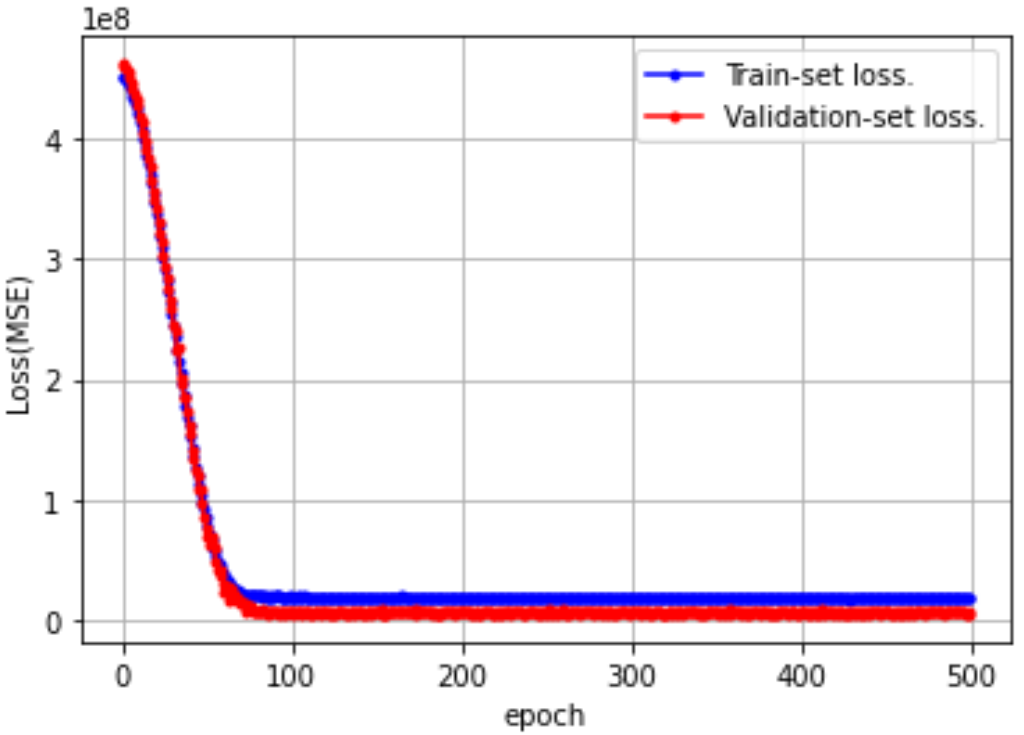
딥러닝

전체데이터

	RMSE
Deep Learning	2781.87

전체데이터


1e8
Train-set loss.
Validation-set loss.
Loss(MSE)
epoch
0
25
50
75
100
125
150
175
200
1
2
3
4


1e8
Train-set loss.
Validation-set loss.
Loss(MSE)
epoch
0
100
200
300
400
500
1
2
3
4

향후과제

- Keras Tuner 적용
- 모든 회사 차에 대한 EDA 진행
- 한국 중고차 데이터 셋 적용

THNAK YOU

감사합니다.