FREE PRESENTATION
TEMPLATE

중고차 가격예측

발표자. 정예지 이수빈

Competition

중고차 가격 예측
경진대회

This is Presentation contents

Competition

중고차 시장 데이터를 이용해
자동차 가격 예측하기

평가 방식

nmae score

NMAE :

(Normalized Mean Absolute Error)

즉, 정규화된 MAE

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$

데이터 확인

- Title : 제조사 모델명
- Odometer : 주행 거리
- Location : 판매처
- Isimported : 현지 사용 여부
- Engine: 엔진 종류
- Transmission : 자동 변속기
- Fuel: 연료 종류
- Paint: 페인트 색상
- Year: 제조 년도
- Target: 자동차 가격

id	title	odometer	location	isimported	engine	transmission	fuel	paint	year	target
0	Toyota RAV 4	18277	Lagos	Foreign Used	4-cylinder(I4)	automatic	petrol	Red	2016	13665000
1	Toyota Land Cruiser	10	Lagos	New	4-cylinder(I4)	automatic	petrol	Black	2019	33015000
2	Land Rover Range Rover Evoque	83091	Lagos	Foreign Used	6-cylinder(V6)	automatic	petrol	Red	2012	9915000
3	Lexus ES 350	91524	Lagos	Foreign Used	4-cylinder(I4)	automatic	petrol	Gray	2007	3815000
4	Toyota Venza	94177	Lagos	Foreign Used	6-cylinder(V6)	automatic	petrol	Red	2010	7385000

데이터 확인

결측치 확인

```
train.isnull().sum()
```

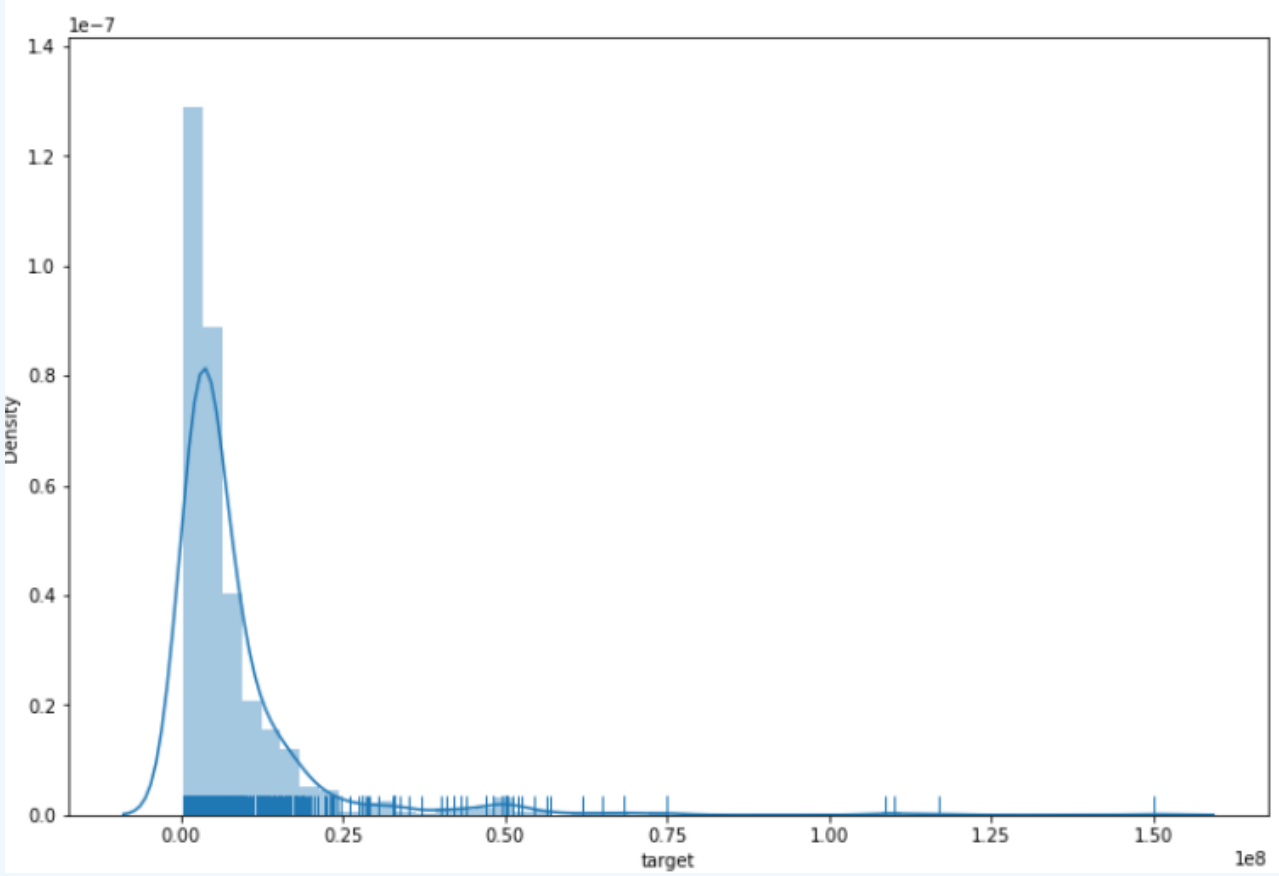
```
id              0
title           0
odometer        0
location        0
isimported      0
engine          0
transmission    0
fuel            0
paint           0
year            0
target          0
dtype: int64
```

```
test.isnull().sum()
```

```
id              0
title           0
odometer        0
location        0
isimported      0
engine          0
transmission    0
fuel            0
paint           0
year            0
dtype: int64
```

Feature Engineering - Target

```
plt.figure(figsize=(12,8))
sns.distplot(train['target'], rug=True)
```

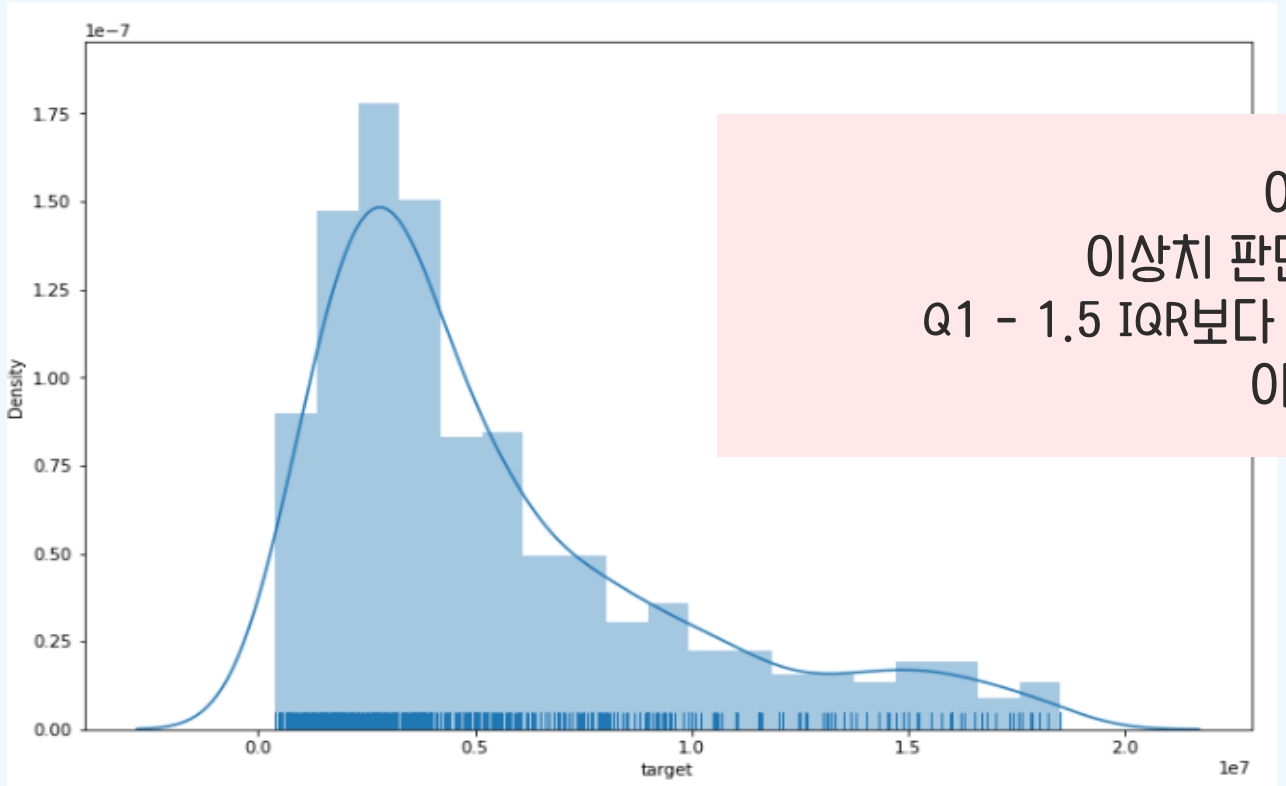


지나치게 큰 값들이 존재

Feature Engineering - Target

```
IQR=abs(train['target'].quantile(0.75)-train['target'].quantile(0.25))
print('큰 이상치:',len(train[train['target']>(train['target'].quantile(0.75)+(1.5*IQR))]))
print('작은 이상치:',len(train[train['target']<(train['target'].quantile(0.25)-(1.5*IQR))]))
train=train[train['target']<=(train['target'].quantile(0.75)+(1.5*IQR))]
```

```
큰 이상치: 82
작은 이상치: 0
```



이상치 확인 후 제거
이상치 판단은 IQR = Q3 - Q1 계산 후
Q1 - 1.5 IQR보다 작거나 Q3 + 1.5 IQR보다 큰 경우를
이상치로 판단하였다.

Feature Engineering - title

```
title = train['title'].unique()
title
```

```
array(['Toyota RAV 4', 'Land Rover Range Rover Evoque', 'Lexus ES 350',
       'Toyota Venza', 'Toyota Corolla', 'Land Rover Range Rover Sport',
       'Pontiac Vibe', 'Toyota Tacoma', 'Ford Escape', 'Honda Civic',
       'Volvo XC90', 'Lexus RX 350', 'BMW 750', 'Infiniti JX',
       'Honda Accord', 'Mercedes-Benz ML 350', 'Toyota Camry',
       'Hyundai Azera', 'Lexus GX 460', 'BMW 325', 'Toyota Sienna',
       'Honda Fit', 'Honda CR-V', 'Hyundai Tucson', 'Ford Transit',
```

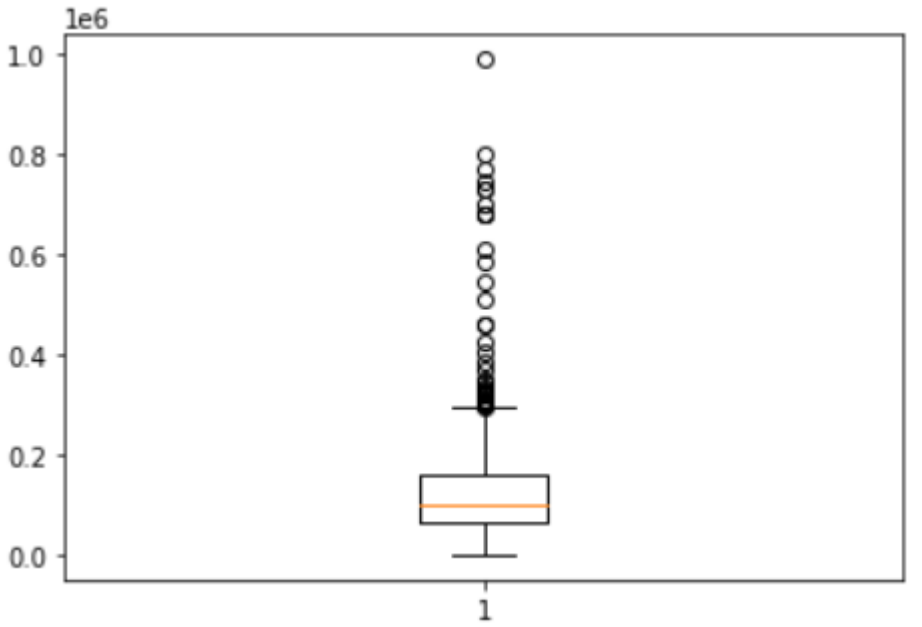
```
train['brand'] = train.title.str.split(' ').str[0]
```

engine	transmission	fuel	paint	year	target	brand
4-cylinder(I4)	automatic	petrol	Red	2016	13665000	Toyota
6-cylinder(V6)	automatic	petrol	Red	2012	9915000	Land
4-cylinder(I4)	automatic	petrol	Gray	2007	3815000	Lexus
6-cylinder(V6)	automatic	petrol	Red	2010	7385000	Toyota
4-cylinder(I4)	automatic	petrol	White	2004	1465000	Toyota

브랜드에 따라 가격이 다를 것이라 예상
-> 브랜드 별 컬럼 생성

Feature Engineering - odometer

```
plt.boxplot(train['odometer'])
```



이상치 확인

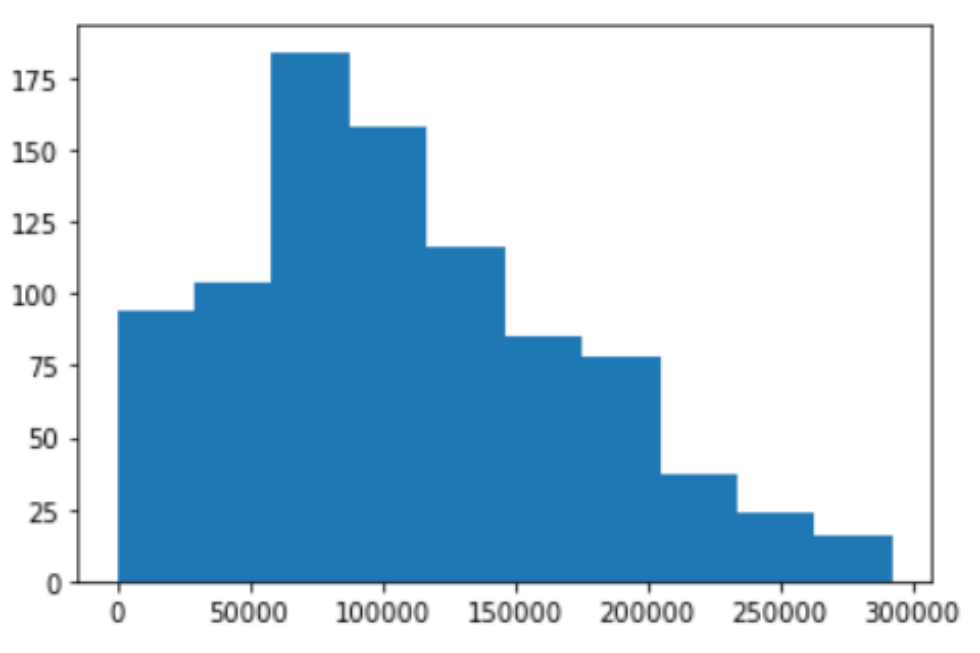
```
IQR=abs(train['odometer'].quantile(0.75)-train['odometer'].quantile(0.25))
print('큰 이상치:',len(train[train['odometer']>(train['odometer'].quantile(0.75)+(1.5*IQR))]))
print('작은 이상치:',len(train[train['odometer']<(train['odometer'].quantile(0.25)-(1.5*IQR))]))
```

```
큰 이상치: 37
작은 이상치: 0
```

Feature Engineering - odometer

이상치 제거

```
train=train[train['odometer']<=(train['odometer'].quantile(0.75)+(1.5*IQR))]
```



Feature Engineering - odometer

```
train['odometer'].describe()
```

```
count       896.000000
mean     108733.260045
std       64321.578619
min           0.000000
25%       64614.250000
50%       97797.000000
75%      151671.750000
max      292547.000000
Name: odometer, dtype: float64
```

주행 거리를 4분위수로
구분하는 컬럼 생성

```
# 구간 나누는 지점 설정
bins = list([0, 64614, 97797, 151671, 292547])
# 구간 이름
bins_label = ['lower', 'low','mid', 'high','higher']
```

```
train["odometer_level"] = pd.cut(train["odometer"],
                                 bins, right=False, labels=bins_label[:-1])
train.head()
```

	id	title	odometer	location	isimported	engine	transmission	fuel	paint	year	target	brand	odometer_level
0	0	Toyota RAV 4	18277	Lagos	Foreign Used	4-cylinder(I4)	automatic	petrol	Red	2016	13665000	Toyota	lower
2	2	Land Rover Range Rover Evoque	83091	Lagos	Foreign Used	6-cylinder(V6)	automatic	petrol	Red	2012	9915000	Land	low
3	3	Lexus ES 350	91524	Lagos	Foreign Used	4-cylinder(I4)	automatic	petrol	Gray	2007	3815000	Lexus	low
4	4	Toyota Venza	94177	Lagos	Foreign Used	6-cylinder(V6)	automatic	petrol	Red	2010	7385000	Toyota	mid
5	5	Toyota Corolla	216375	Abuja	Locally used	4-cylinder(I4)	automatic	petrol	White	2004	1465000	Toyota	high

Feature Engineering - odometer

```
train[['odometer_level', 'target']].groupby(['odometer_level'], as_index=True).describe()
```

	target							
	count	mean	std	min	25%	50%	75%	max
odometer_level								
lower	224.0	9.095989e+06	5.107225e+06	543000.0	4490000.0	8815000.0	13693750.0	18515000.0
low	224.0	5.344386e+06	3.285274e+06	515000.0	3205000.0	4015000.0	6618750.0	18015000.0
mid	224.0	4.366317e+06	2.913646e+06	400000.0	2315000.0	3702500.0	5515000.0	16515000.0
high	223.0	3.041935e+06	2.235217e+06	400000.0	1580000.0	2315000.0	3760000.0	15515000.0

주행거리가 작을 수록가격이 높은 것을 알 수 있음

Feature Engineering - location

```
train['location'].unique()
```

```
array(['Lagos ', 'Lagos', 'Abuja', 'Lagos State', 'Ogun', 'FCT', 'Accra',
       'other', 'Abuja ', 'Abia State', 'Adamawa ', 'Abia', 'Ogun State'],
      dtype=object)
```

```
test['location'].unique()
```

```
array(['Abuja', 'Lagos', 'Lagos ', 'Ogun', 'Mushin', 'Lagos State',
       'other', 'Abuja ', 'Arepo ogun state ', 'Ogun State', 'Abia'],
      dtype=object)
```

확인해본 결과 앞뒤에 공백이 존재하고
같은 주인데 state가 있고 없고의 차이가 보임
-> 공백제거 및 변경

Feature Engineering - location

```
train['location'].unique() #Lagos에서 띄어쓰기 제거해주기
train.replace('Lagos ','Lagos',inplace=True)
train.replace('Abuja ','Abuja',inplace=True)
train.replace('Lagos State','Lagos',inplace=True)
train.replace('Abuja State','Abuja',inplace=True)
train.replace('Abia State','Abia',inplace=True)
train.replace('Ogun State','Ogun',inplace=True)
```

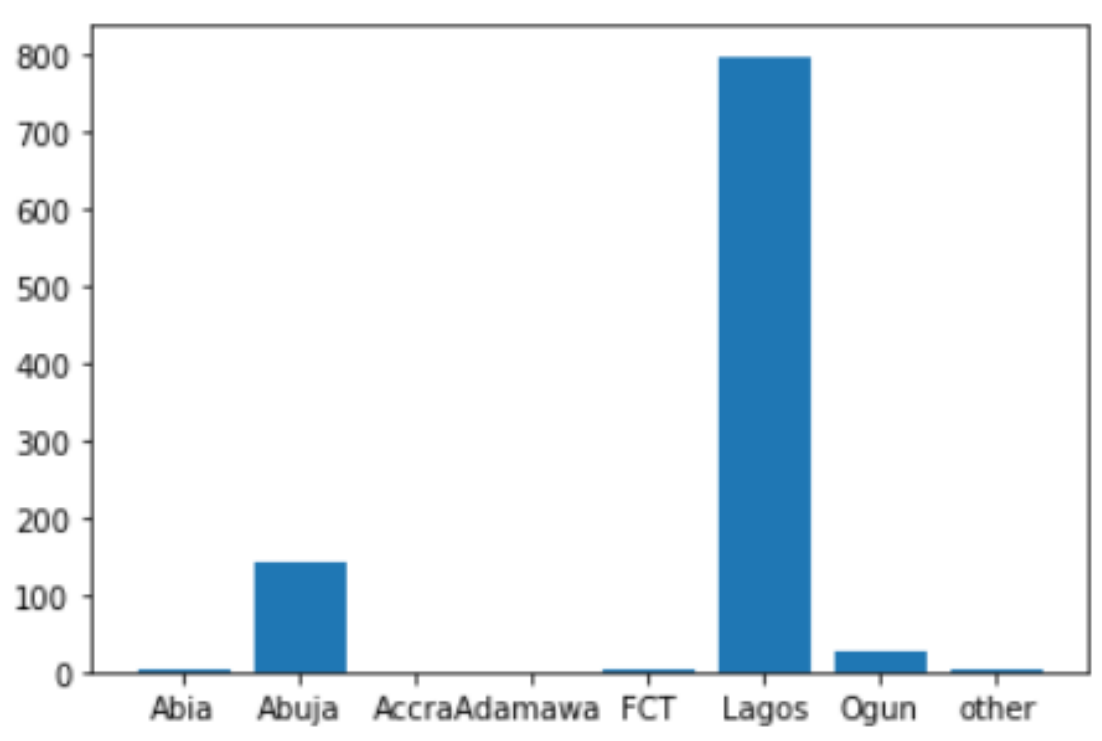
```
test['location'].unique() #Lagos띄어쓰기, Abuja띄어쓰기
test.replace('Lagos ','Lagos',inplace=True)
test.replace('Abuja ','Abuja',inplace=True)
test.replace('Lagos State','Lagos',inplace=True)
test.replace('Abuja State','Abuja',inplace=True)
test.replace('Ogun State','Ogun',inplace=True)
test.replace('Arepo ogun state ','Ogun',inplace=True)
test.replace('Mushin','other',inplace=True) #하나 있길래 그냥 other로
```

```
print('train:',train['location'].unique())
print('test:',test['location'].unique())
```

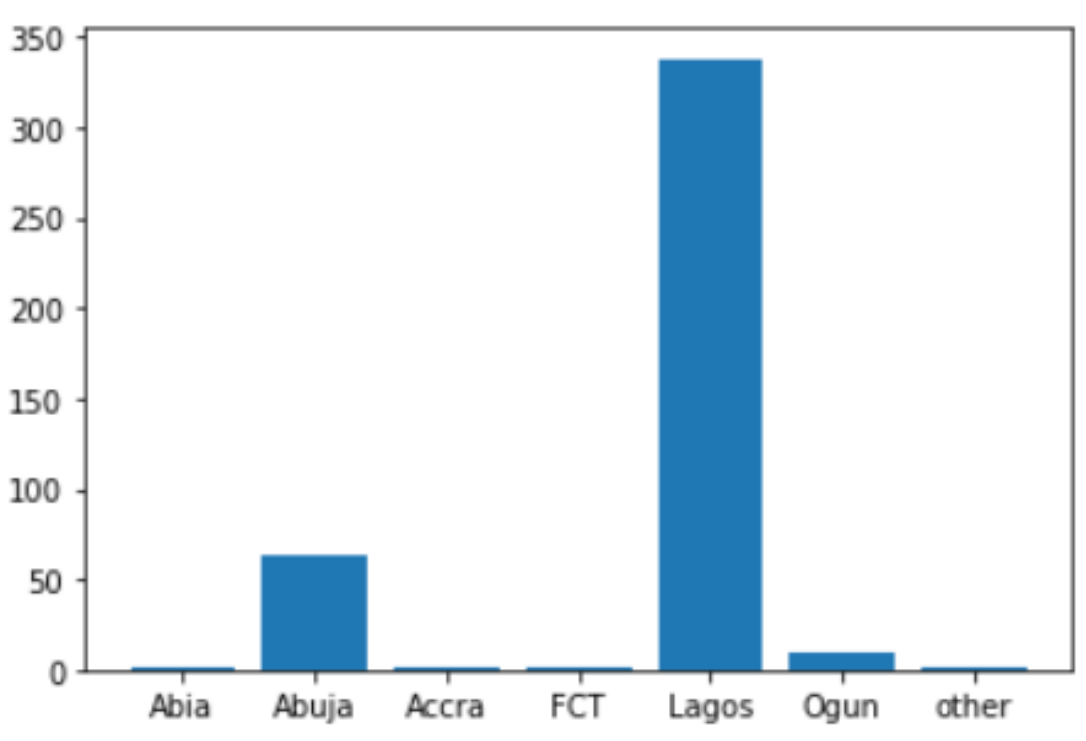
```
train: ['Lagos' 'Abuja' 'Ogun' 'FCT' 'Accra' 'other' 'Abia' 'Adamawa ']
test: ['Abuja' 'Lagos' 'Ogun' 'other' 'Abia']
```

변경 후 값 확인해보니 잘 정돈된 것으로 보임

Feature Engineering - location



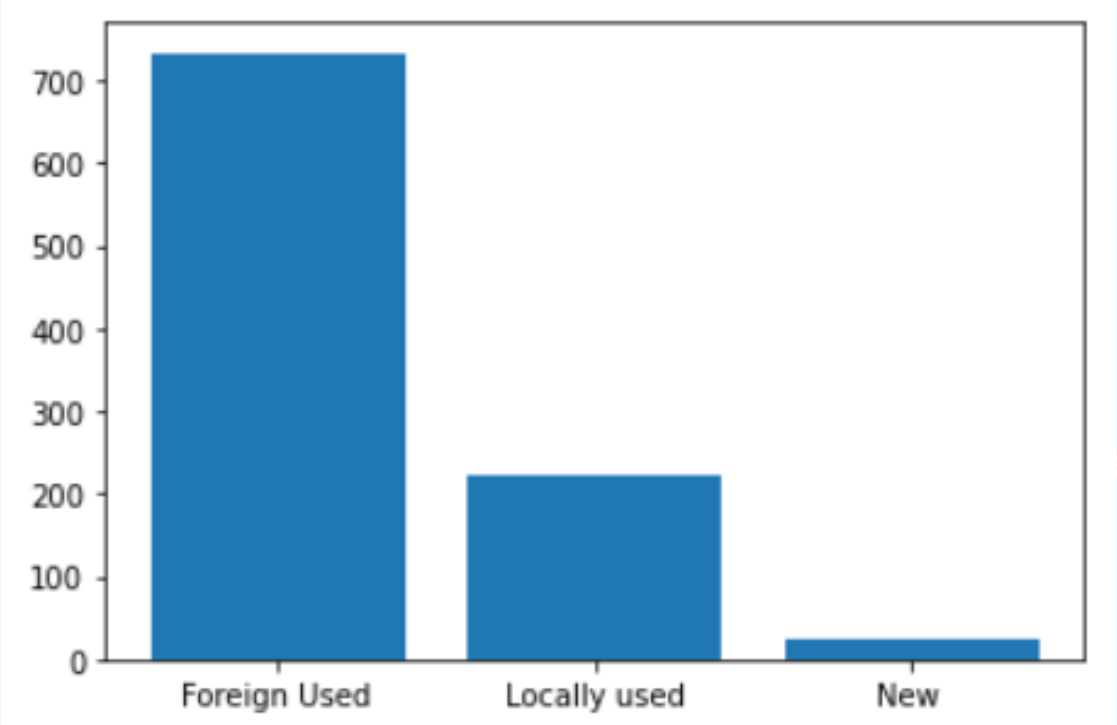
train



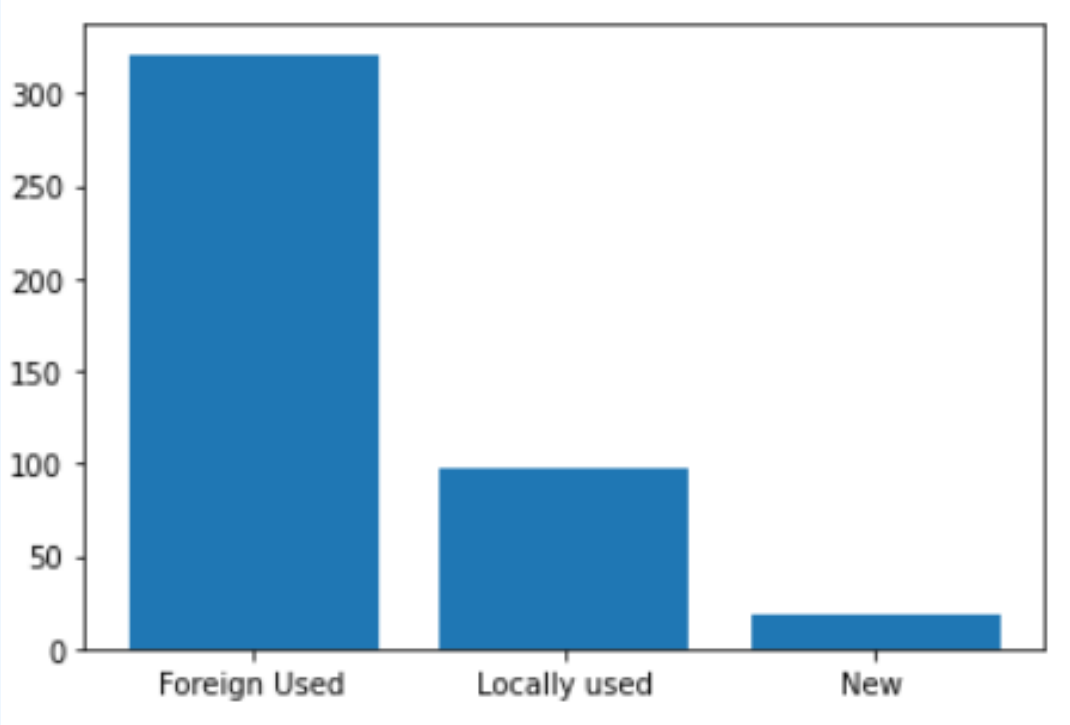
test

분포가 적당히 고르다

Feature Engineering - isimported



train



test

분포가 적당히 고르다

Feature Engineering - engine

```
print('train:',train['engine'].unique())
print('test:',test['engine'].unique())
```

```
train: ['4-cylinder(I4)' '6-cylinder(V6)' '8-cylinder(V8)' '6-cylinder(I6)'
 '4-cylinder(H4)' '5-cylinder(I5)' '3-cylinder(I3)' '2-cylinder(I2)']
test: ['4-cylinder(I4)' '6-cylinder(V6)' '6-cylinder(I6)' '8-cylinder(V8)'
 '5-cylinder(I5)' '2-cylinder(I2)' '12-cylinder(V12)' '3-cylinder(I3)']
```

Train test데이터에 모두 같은 값만 존재하고
이상치가 없음을 확인

Feature Engineering - transmission

```
print(train['transmission'].unique())
print(test['transmission'].unique())
```

```
['automatic' 'manual']
['automatic' 'manual']
```

Train test데이터에 모두 같은 값만 존재하고 이상치가 없음을 확인

Feature Engineering - fuel

```
print(train['fuel'].unique())
print(test['fuel'].unique())
```

```
['petrol' 'diesel']
['petrol' 'diesel']
```

Train test데이터에 모두 같은 값만 존재하고 이상치가 없음을 확인

Feature Engineering - paint

```
print(train['paint'].unique())
print(test['paint'].unique())
```

```
['Red' 'Black' 'Gray' 'White' 'Blue' 'Redl' 'Silver' ' Black/Red'
 'Deep Blue' 'Dark Grey' 'Brown' 'Grey' 'Green' 'Purple' 'Gold'
 'Dark Blue' 'Milk' 'Midnight Black Metal' 'green' 'Beige' 'Blue '
 'Silver ' 'Dark Ash' 'Black ' 'orange' 'Cream' 'blue' 'white' 'Dark gray'
 'White orchild pearl' 'red' 'Dark Green' 'Yellow' 'Sliver' 'White '
 'Wine' 'white-blue' 'Magnetic Gray' 'WHITE' 'yellow' 'Gray '
 'Dark silver ' 'Dark blue ' 'Gold ' 'SILVER' 'Black.' 'WINE'
 'Silver/grey' 'Ink blue' 'Light blue' 'Sky blue' 'Gery' 'Pale brown'
 'Whine ' 'Cream ' 'Black and silver' 'DARK GREY' 'Grey ' 'Dark ash'
 'Light silver ' 'BLACK' 'GOLD' 'Black sand pearl' 'Off white' 'Ash'
 'Maroon' 'Navy blue' 'Super White' ' Black' 'Ash and black' 'Green '
 'Magnetic Gray Metallic' 'Skye blue' 'Off white I']
['White' 'Black' 'Dark Grey' 'Red' 'Silver' 'white' 'Blue' 'Gray' 'Grey'
 'Gold' 'Green' 'Silver ' 'Sliver ' 'Gold ' 'Black ' 'Cream' 'Brown'
 'black' 'Yellow' 'Cream ' 'Dark Green' 'White and green' 'Grey '
 'Light Grey' 'Maroon' 'Wine' 'Ash' 'GOLD' 'Blac' 'Dark Blue' 'Dark Ash'
 'green' 'Sliver' 'Golf' 'BLACK' 'Dark blue ' 'Blue ' 'blue' 'Navy blue'
 'Indigo ink pearl' '  Brown' 'Gre
 'Classic Silver Met(1F7)' 'Beige'
```

띄어쓰기가 앞뒤에 있거나 같은 색을 다양하게 표현,
오타 존재 -> 수정

Feature Engineering - paint

```
for i in range(len(train)):
    train['paint'].iloc[i]=train['paint'].iloc[i].rstrip()
    train['paint'].iloc[i]=train['paint'].iloc[i].lstrip()
    train['paint'].iloc[i]=train['paint'].iloc[i].lower()
for i in range(len(test)):
    test['paint'].iloc[i]=test['paint'].iloc[i].rstrip()
    test['paint'].iloc[i]=test['paint'].iloc[i].lstrip()
    test['paint'].iloc[i]=test['paint'].iloc[i].lower()
```

Rstrip() 우측공백제거
Lstrip() 좌측공백제거
Lower() 소문자로

Feature Engineering - paint

```
train['paint'].replace('grey','gray',inplace=True)
train['paint'].replace('redl','red',inplace=True)
train['paint'].replace('gery','gray',inplace=True)
train['paint'].replace('skye blue','sky blue',inplace=True)
train['paint'].replace('maroon','wine',inplace=True)
train['paint'].replace('whine','wine',inplace=True)
train['paint'].replace('off white l','white',inplace=True)
test['paint'].replace('grey','gray',inplace=True)
test['paint'].replace('redl','red',inplace=True)
test['paint'].replace('gery','gray',inplace=True)
test['paint'].replace('skye blue','sky blue',inplace=True)
test['paint'].replace('maroon','wine',inplace=True)
test['paint'].replace('whine','wine',inplace=True)
test['paint'].replace('off white l','white',inplace=True)
```

Feature Engineering - paint

```
train['paint'].unique()
```

```
array(['red', 'black', 'gray', 'white', 'blue', 'silver', 'black/red',
       'deep blue', 'dark grey', 'brown', 'green', 'purple', 'gold',
       'dark blue', 'milk', 'midnight black metal', 'beige', 'dark ash',
       'orange', 'cream', 'dark gray', 'white orchild pearl',
       'dark green', 'yellow', 'sliver', 'wine', 'white-blue',
       'magnetic gray', 'dark silver', 'black.', 'silver/grey',
       'ink blue', 'light blue', 'sky blue', 'pale brown',
       'black and silver', 'light silver', 'black sand pearl',
       'off white', 'ash', 'navy blue', 'super white', 'ash and black',
       'magnetic gray metallic'], dtype=object)
```

Modeling-전처리

	id	title	location	isimported	engine	transmission	fuel	paint	year	target	comp	odometer_level
0	0	Toyota RAV 4	Lagos	Foreign Used	4-cylinder(I4)	automatic	petrol	red	2016	13665000	Toyota	1
1	2	Land Rover Range Rover Evoque	Lagos	Foreign Used	6-cylinder(V6)	automatic	petrol	red	2012	9915000	Land	2
2	3	Lexus ES 350	Lagos	Foreign Used	4-cylinder(I4)	automatic	petrol	gray	2007	3815000	Lexus	2
3	4	Toyota Venza	Lagos	Foreign Used	6-cylinder(V6)	automatic	petrol	red	2010	7385000	Toyota	3
4	5	Toyota Corolla	Abuja	Locally used	4-cylinder(I4)	automatic	petrol	white	2004	1465000	Toyota	4

Id title은 학습에 사용하지 않을 변수이므로 제거
나머지 변수들은 순서관계가 없다고 판단하여 one hot encoding 사용

```
del train ['id']
del train ['title']
del test ['id']
del test ['title']
```

Modeling-전처리

```
all=pd.concat([train,test])
```

```
All=pd.get_dummies(all)
train=All[All['target'].notnull()]
test=All[All['target'].isnull()]
```

```
Y=train['target']
del train ['target']
X=train
```

```
x_trn,x_val,y_trn,y_val=train_test_split(X,Y,random_state=0, test_size=0.25)
```

Modeling-평가지표

```
from sklearn.metrics import mean_absolute_error
def nmae(y,y_pred):
    import numpy as np
    mae=mean_absolute_error(y,y_pred)
    ab=np.mean(abs(y))
    nmae=mae/ab
    return nmae
```

평가지표 함수를 정의

Modeling-RandomForestRegressor

```
rf = RandomForestRegressor(random_state=0, max_depth=8, min_samples_leaf=2,
                           min_samples_split=2,n_estimators=300)
rf.fit(train_x, train_y)

rf_pred = rf.predict(val_x)
MSE = mean_squared_error(rf_pred, val_y)
RMSE = np.sqrt(MSE)
RMSE
print(nmae(rf_pred, val_y))
print(rf.score(val_x, val_y))
```

```
0.22769394733237874
0.790915802665374 1
```

Modeling-XGBRegressor

```
xgb=XGBRegressor(random_state=0)
xgb.fit(x_trn,y_trn)
y_pred=xgb.predict(x_val)
print(nmae(y_pred,y_val))
print(xgb.score(x_val,y_val))
```

```
0.24575423717660777
0.7763757394737796
```

Modeling-GBRegressor

```
gb=GradientBoostingRegressor(random_state=0)
gb.fit(x_trn,y_trn)
y_pred=gb.predict(x_val)
print(nmae(y_pred,y_val))
print(gb.score(x_val,y_val))
```

```
0.25624724912100283
0.7646290408079663
```

Modeling 결과

1	Jeongsik		0.18685	25	24일 전
2	standing-o		0.21717	14	23일 전
3	c4foryou	c4	0.22527	3	23일 전

```
rf = RandomForestRegressor(random_state=0, max_depth=8, min_samples_leaf=2,
                           min_samples_split=2,n_estimators=300)
rf.fit(train_x, train_y)

rf_pred = rf.predict(val_x)
MSE = mean_squared_error(rf_pred, val_y)
RMSE = np.sqrt(MSE)
RMSE
print(nmae(rf_pred, val_y))
print(rf.score(val_x, val_y))
```

```
0.22769394733237874
0.7909158026653741
```

대회가 끝나서 제출을 하지는 못했지만
1등의 점수와 비교해 보았을 때
꽤 좋은 성능을 냈음을 확인했다

FREE PRESENTATION TEMPLATE

THANK YOU