중고차 가격 예측

데이터 출처 : https://www.kaggle.com/datasets/adityadesai13/used-car-dataset-ford-and-mercedes?select=vw.csv
수정 데이터 출처 : https://www.datamanim.com/dataset/03_dataq/typetwo.html#id16

x_train: https://raw.githubusercontent.com/Datamanim/datarepo/main/carsprice/X_train.csv
y_train: https://raw.githubusercontent.com/Datamanim/datarepo/main/carsprice/y_train.csv
x_test: https://raw.githubusercontent.com/Datamanim/datarepo/main/carsprice/X_test.csv
x_label(평가용) :
https://raw.githubusercontent.com/Datamanim/datarepo/main/carsprice/y_test.csv

project file(.ipynb) : https://github.com/fa-ina-tic/report/blob/main/UsedCar.ipynb

```
import pandas as pd
#데이터 로드
x_train = pd.read_csv("https://raw.githubusercontent.com/Datamanim/datarepo/main/ca
y_train = pd.read_csv("https://raw.githubusercontent.com/Datamanim/datarepo/main/ca
x_test= pd.read_csv("https://raw.githubusercontent.com/Datamanim/datarepo/main/cars

#import data 확인
display(x_train.head())
display(y_train.head())
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engi
0	13207	hyundi	Santa Fe	2019	Semi-Auto	4223	Diesel	145.0	39.8	
1	17314	vauxhall	GTC	2015	Manual	47870	Diesel	125.0	60.1	
2	12342	audi	RS4	2019	Automatic	5151	Petrol	145.0	29.1	
3	13426	vw	Scirocco	2016	Automatic	20423	Diesel	30.0	57.6	
4	16004	skoda	Scala	2020	Semi-Auto	3569	Petrol	145.0	47.1	

	carID	price
0	13207	31995
1	17314	7700
2	12342	58990
3	13426	12999
4	16004	16990

EDA

1. 결측값

2. 이상값
3. 데이터 타입

```
# data 기본 정보 확인
# X : 9 Columns 4960 Rows Null = None
# Y : 2 Columns 4960 Rows Null = None
print(x_train.info())
print(y_train.info())
print(x_test.info())

print(x_train.describe())
print(y_train.describe())
print(x_test.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4960 entries, 0 to 4959
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   carID         4960 non-null   int64  
 1   brand         4960 non-null   object 
 2   model         4960 non-null   object 
 3   year          4960 non-null   int64  
 4   transmission  4960 non-null   object 
 5   mileage       4960 non-null   int64  
 6   fuelType      4960 non-null   object 
 7   tax           4960 non-null   float64
 8   mpg           4960 non-null   float64
 9   engineSize    4960 non-null   float64
dtypes: float64(3), int64(3), object(4)
memory usage: 387.6+ KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4960 entries, 0 to 4959
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   carID   4960 non-null   int64
 1   price   4960 non-null   int64
dtypes: int64(2)
memory usage: 77.6 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2672 entries, 0 to 2671
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   carID         2672 non-null   int64  
 1   brand         2672 non-null   object 
 2   model         2672 non-null   object 
 3   year          2672 non-null   int64  
 4   transmission  2672 non-null   object 
 5   mileage       2672 non-null   int64  
 6   fuelType      2672 non-null   object 
 7   tax           2672 non-null   float64
 8   mpg           2672 non-null   float64
```

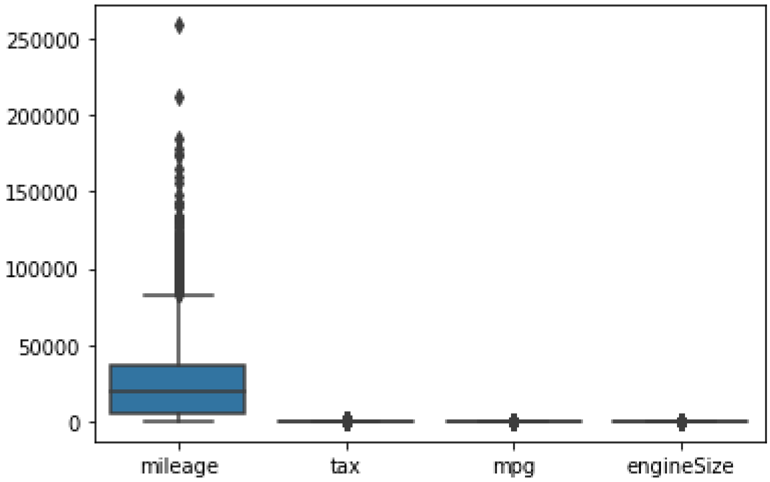
```
     9   engineSize    2672 non-null   float64
    dtypes: float64(3), int64(3), object(4)
    memory usage: 208.9+ KB
    None
                  carID         year        mileage          tax          mpg  \
    count   4960.000000  4960.000000    4960.000000  4960.000000  4960.000000
    mean   15832.446169  2016.737903   24956.286895   152.332661    50.370766
    std     2206.717006     2.884035   24443.333662    82.403844    35.746505
    min    12002.000000  1997.000000       1.000000     0.000000     2.800000
    25%    13929.250000  2016.000000    5641.250000   145.000000    38.700000
    50%    15840.000000  2017.000000   19000.000000   145.000000    47.100000
    75%    17765.750000  2019.000000   36702.000000   150.000000    54.300000
    max    19629.000000  2020.000000  259000.000000   580.000000   470.800000

            engineSize
    count  4960.000000
```

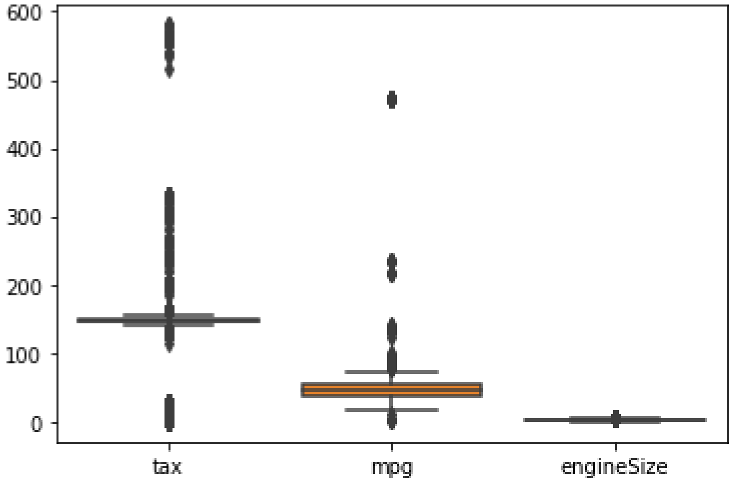
```
# 결측값
print(x_train.isnull().sum())
print(x_train.isnull().sum())
print(x_train.isnull().sum())
```

```
    carID           0
    brand           0
    model           0
    year            0
    transmission    0
    mileage         0
    fuelType        0
    tax             0
    mpg             0
    engineSize      0
    dtype: int64
    carID           0
    brand           0
    model           0
    year            0
    transmission    0
    mileage         0
    fuelType        0
    tax             0
    mpg             0
    engineSize      0
    dtype: int64
    carID           0
    brand           0
    model           0
    year            0
    transmission    0
    mileage         0
    fuelType        0
    tax             0
    mpg             0
    engineSize      0
    dtype: int64
```

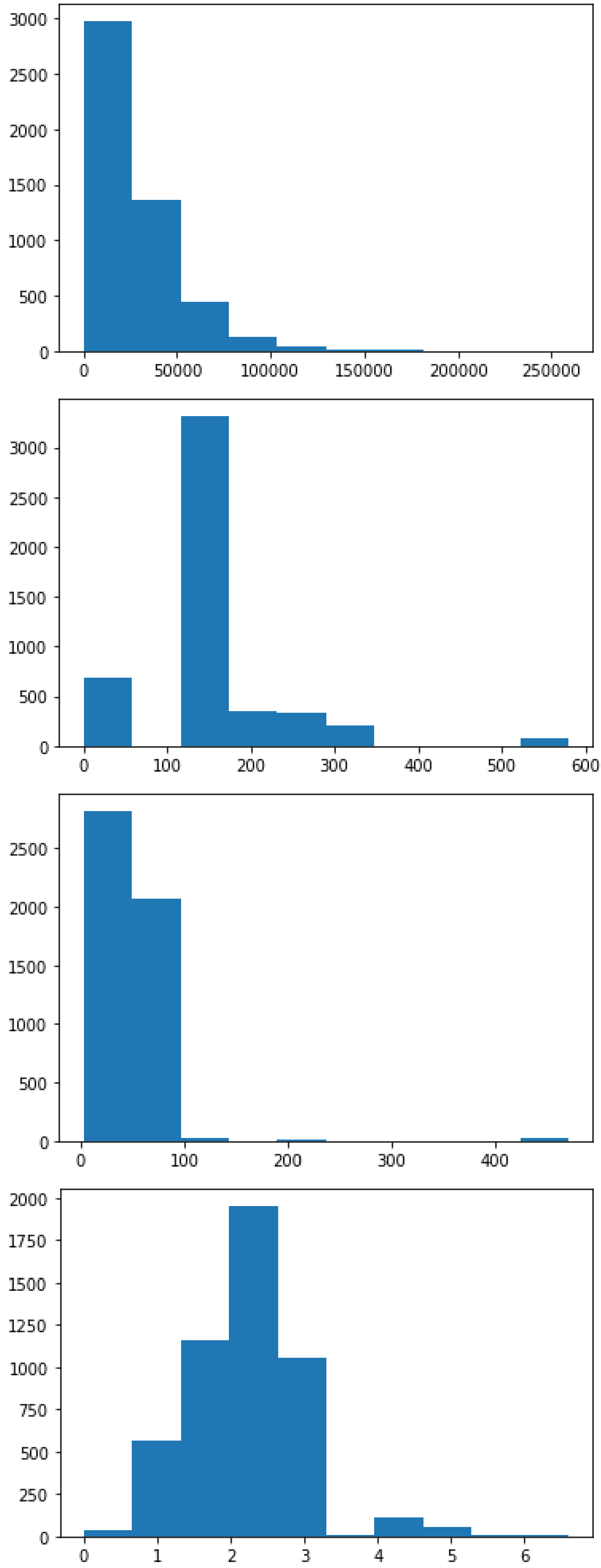
```
# 이상값
# 이상값 제거 함수 정의
def get_outliers(df=None, column=None, weight=1.5):
    per_75 = np.percentile(df[column].values, 75)
    per_25 = np.percentile(df[column].values, 25)

    IQR = (per_75 - per_25) * weight
    high = per_75 + IQR
    low = per_25 - IQR

    outlier_idx = df[(df[column]>high)|(df[column]<low)].index
    return outlier_idx
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# 시각화로 탐색
# 1. 이상치 개수 파악
boxplot = sns.boxplot(data=x_train[['mileage', 'tax', 'mpg', 'engineSize']])
plt.show()

boxplot = sns.boxplot(data=x_train[['tax', 'mpg', 'engineSize']])
plt.show()
```



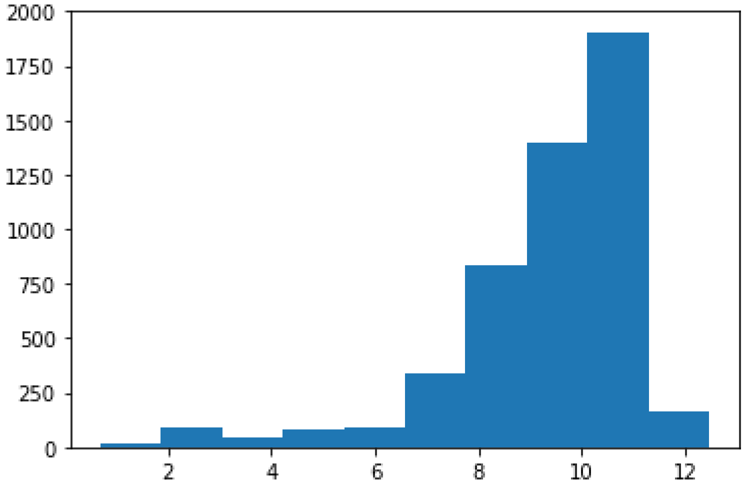


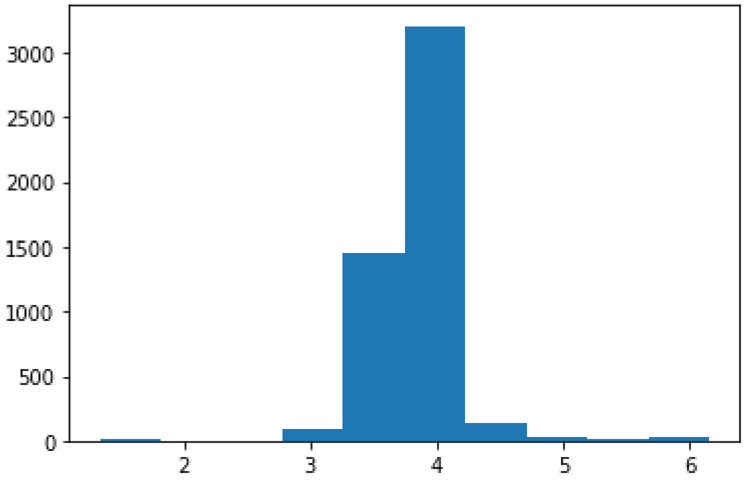
```
# 2. 값들의 분포 확인
for i in ['mileage', 'tax', 'mpg', 'engineSize']:
    plt.hist(x = x_train[[i]])
    plt.show()
```


3000
2500
2000
1500
1000
500
0
0
50000
100000
150000
200000
250000
3000
2500
2000
1500
1000
500
0
0
100
200
300
400
500
600
2500
2000
1500
1000
500
0
0
100
200
300
400
2000
1750
1500
1250
1000
750
500
250
0
0
1
2
3
4
5
6

```
# mileage, mpg column에 로그를 취하여 머신 러닝의 성능을 높인다.
import numpy as np

log_features = ['mileage', 'mpg']

for i in log_features:
    x_train[i] = x_train[i].apply(lambda x: np.log1p(x))
    x_test[i] = x_test[i].apply(lambda x: np.log1p(x))
```

```
for i in log_features:
    plt.hist(x = x_train[[i]])
    plt.show()
```





```
for i in ['mileage', 'tax', 'mpg', 'engineSize']:
    outliers = get_outliers(x_train, i, 2)
    x_train.drop(outliers, axis=0, inplace=True)
    y_train.drop(outliers, axis=0, inplace=True)
```

```
# data type

# 명목변수 분리
cate_col = ['brand', 'model', 'year', 'transmission', 'fuelType']
```

```
# string / float&int
string_col = ['brand', 'model', 'transmission', 'fuelType']
num_col = list(x_train.columns.drop(string_col))
```

```
# 명목 변수 data type category로 변경
for i in cate_col:
    x_train[i] = x_train[i].astype('category')
    x_test[i] = x_test[i].astype('category')
print(x_train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3070 entries, 0 to 4959
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype   
---  ------        --------------  -----   
 0   carID         3070 non-null   int64   
 1   brand         3070 non-null   category
 2   model         3070 non-null   category
 3   year          3070 non-null   category
 4   transmission  3070 non-null   category
 5   mileage       3070 non-null   float64 
 6   fuelType      3070 non-null   category
 7   tax           3070 non-null   float64 
 8   mpg           3070 non-null   float64 
 9   engineSize    3070 non-null   float64 
dtypes: category(5), float64(4), int64(1)
memory usage: 163.0 KB
None
```

```
# train과 test에 명목 변수가 다 같이 있는지 확인
train_dum = pd.get_dummies(x_train[cate_col])
test_dum = pd.get_dummies(x_test[cate_col])

print(set(train_dum.columns) - set(test_dum.columns))
print(set(test_dum.columns) - set(train_dum.columns))
```

```
{'model_ M6'}
{'model_ IQ', 'model_ Amarok', 'model_ CLK', 'model_ Hilux', 'model_ Californi
```

```
# 없는 명목변수를 포함하는 train, test 테이블 생성
X_train_dum = pd.get_dummies(x_train)
X_test_dum = pd.get_dummies(x_test)

train_missing = set(test_dum.columns) - set(train_dum.columns)
test_missing = set(train_dum.columns) - set(test_dum.columns)

for c in train_missing:
    X_train_dum[c] = 0
for c in test_missing:
    X_test_dum[c] = 0
```

Model Selection

```
# package import
from sklearn.model_selection import KFold, train_test_split, GridSearchCV
from sklearn.pipeline import Pipeline
from xgboost import XGBRegressor


X = X_train_dum.iloc[:, 1:]
y = y_train.iloc[:, 1:]

kfold = KFold(n_splits=5, shuffle=True)
pipe = Pipeline(steps=[('model', XGBRegressor(eval_metric='mlogloss', use_label_enc
grid_params = {'model':[XGBRegressor(eval_metric='mlogloss', use_label_encoder=Fals
               'model__max_depth':[3, 5, 7],
               'model__learning_rate':[0.05, 0.1, 0.2, 0.3],
               'model__n_estimators':[50, 100, 200, 300],
               'model__gamma':[0, 0.1, 0.2]}

grid = GridSearchCV(pipe, grid_params, cv=kfold, verbose=0)
grid.fit(X, y)
print(grid.best_params_)
print(grid.best_score_)
```

```
{'model': XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric='mlogloss', gamma=0,
             gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.2, max_bin=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=5,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, n_estimators=300, n_jobs=None,
             num_parallel_tree=None, predictor=None, random_state=None,
             reg_alpha=None, reg_lambda=None, ...), 'model__gamma': 0, 'model_
0.9612029380888971
```

Modeling

```
model = XGBRegressor(eval_metric='mlogloss', use_label_encoder=False,
                      learning_rate=0.2, max_depth=5, n_estimators=200)
model.fit(X, y)
pred = model.predict(X_test_dum.iloc[:, 1:])
pred = pd.Series(pred)

answer = pd.concat([X_test_dum.carID, pred], axis=1)
```

```
answer
```

	carID	0
0	12000	38508.203125
1	12001	33022.828125
2	12004	50232.843750
3	12013	17864.957031
4	12017	80702.773438
...	...	...
2667	19618	72973.968750
2668	19620	16821.384766
2669	19626	16913.240234
2670	19630	25904.589844

```
x_label = pd.read_csv("https://raw.githubusercontent.com/Datamanim/datarepo/main/ca

display(x_label.head())
```

	carID	price
0	12000	38000
1	12001	23495
2	12004	59999
3	12013	16713
4	12017	46000

```
from sklearn.metrics import r2_score

score = r2_score(x_label.price, answer.iloc[:, 1])
print(score)
```

```
0.7104951468890918
```