쿠빅 15기 공도웅, 신윤, 조우영, 최경석

Used Cars Dataset
중고차 가격예측 프로젝트

목차

데이터 개요

데이터 개요

Dataset of Used vehicles for sale in the United States

Model	Manufacturer	Condition	Cylinders	…	Price
…	…	…	…	…	…
…	…	…	…	…	…

중고차의 가격을 예측하는 모델 수립

데이터 EDA

데이터 EDA

2) 변수의 자료형

```
[ ]  vehicles.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 426880 entries, 0 to 426879
Data columns (total 25 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            426880 non-null  int64
 1   url           426880 non-null  object
 2   region        426880 non-null  object
 3   region_url    426880 non-null  object
 4   price         426880 non-null  int64
 5   year          425675 non-null  float64
 6   manufacturer  409234 non-null  object
 7   model         421603 non-null  object
 8   condition     252776 non-null  object
 9   cylinders     249202 non-null  object
 10  fuel          423867 non-null  object
 11  odometer      422480 non-null  float64
 12  title_status  418638 non-null  object
 13  transmission  424324 non-null  object
 14  VIN           265838 non-null  object
 15  drive         296313 non-null  object
 16  size          120519 non-null  object
 17  type          334022 non-null  object
 18  paint_color   296677 non-null  object
 19  image_url     426812 non-null  object
 20  description   426810 non-null  object
 21  state         426880 non-null  object
 22  lat           420331 non-null  float64
 23  long          420331 non-null  float64
 24  posting_date  426812 non-null  object
dtypes: float64(4), int64(2), object(19)
memory usage: 81.4+ MB
```

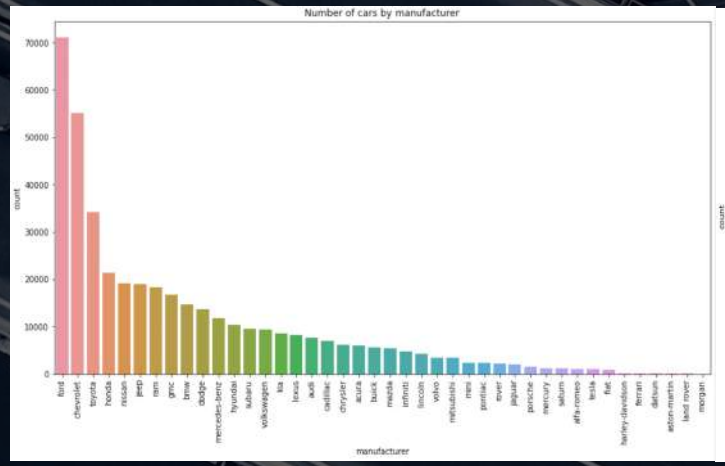
데이터 EDA

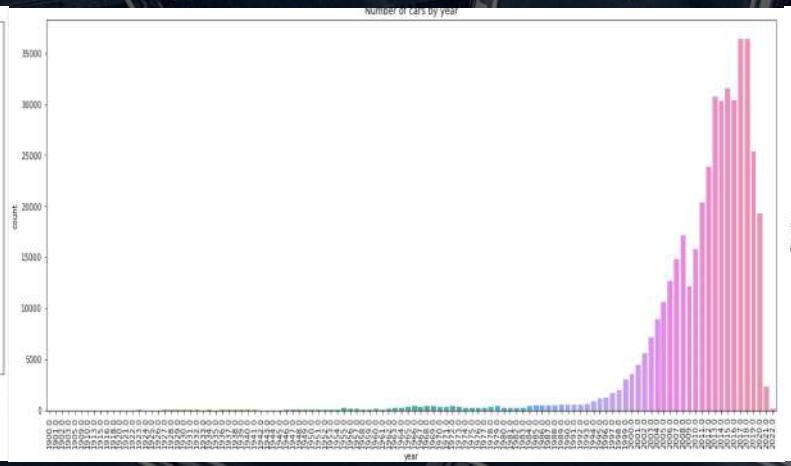
1) 각 column의 의미

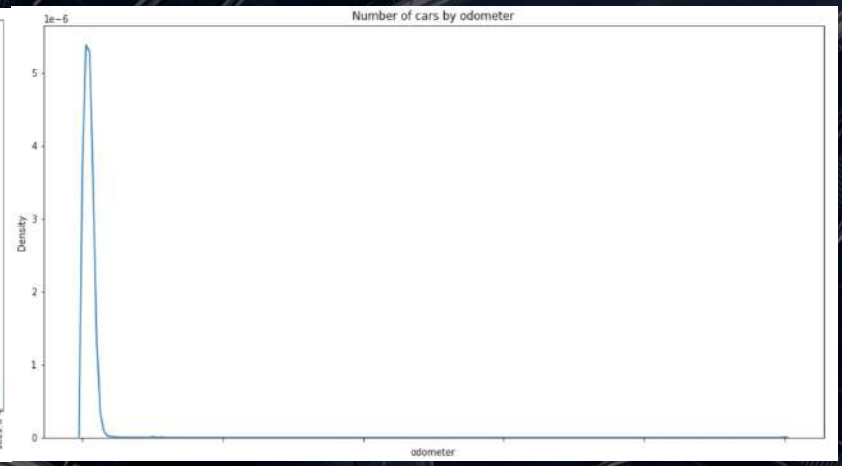
1) id - entry id
2) url - listing URL
3) region - craigslist region
(미국의 지역 온라인 벼룩시장을 말함)
4) region_url: 지역별 온라인 벼룩시장 링크
5) year : 등록연도
6) manufacturer: 자동차 제조사
7) model: 자동차 모델명
8) condition: 자동차의 상태
9) cylinders: 엔진의 기통 수
10) fuel: 연료 종류
11) odometer: 주행거리
12) title_status: 차량등록증 상의 차량 상태
13) transmission: 변속기 종류
14) VIN: 차량의 등록번호
15) drive : 차량의 구동방식
16) size: 차량의 사이즈
17) type: 차량종류
18) paint_color : 차량의 색깔
19) image_URL : 차량 이미지 URL
20) description: 차량에 대한 설명
21) county: useless column by mistake
22) state: 위치하고 있는 state
23) lat: latitude of listing
24) long: longitude of listing
25) posting_date: 업로드된 날짜

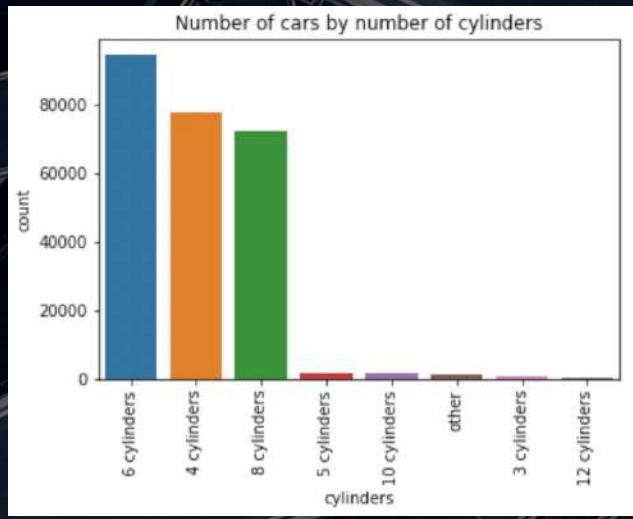
총 25개 중 12개 column 삭제

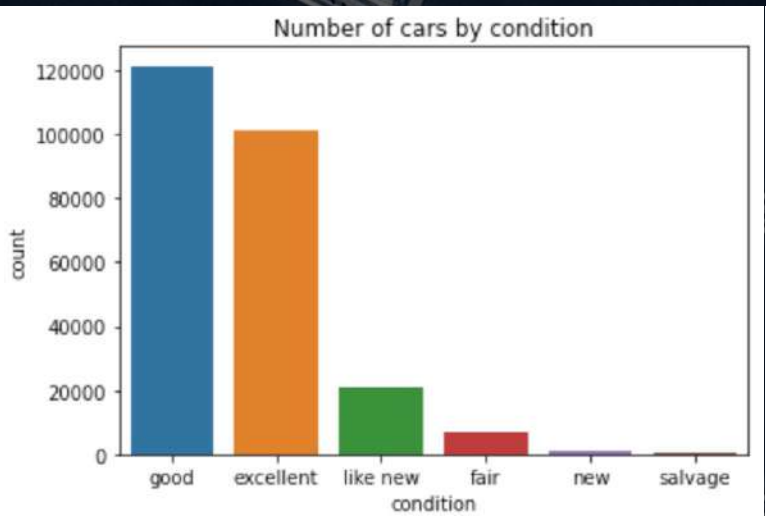
```
[ ]  vehicles.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 426880 entries, 0 to 426879
Data columns (total 25 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            426880 non-null  int64
 1   url           426880 non-null  object
 2   region        426880 non-null  object
 3   region_url    426880 non-null  object
 4   price         426880 non-null  int64
 5   year          425675 non-null  float64
 6   manufacturer  409234 non-null  object
 7   model         421603 non-null  object
 8   condition     252776 non-null  object
 9   cylinders     249202 non-null  object
 10  fuel          423867 non-null  object
 11  odometer      422480 non-null  float64
 12  title_status  418638 non-null  object
 13  transmission  424324 non-null  object
 14  VIN           265838 non-null  object
 15  drive         296313 non-null  object
 16  size          120519 non-null  object
 17  type          334022 non-null  object
 18  paint_color   296677 non-null  object
 19  image_url     426812 non-null  object
 20  description   426810 non-null  object
 21  state         426880 non-null  object
 22  lat           420331 non-null  float64
 23  long          420331 non-null  float64
 24  posting_date  426812 non-null  object
dtypes: float64(4), int64(2), object(19)
memory usage: 81.4+ MB
```

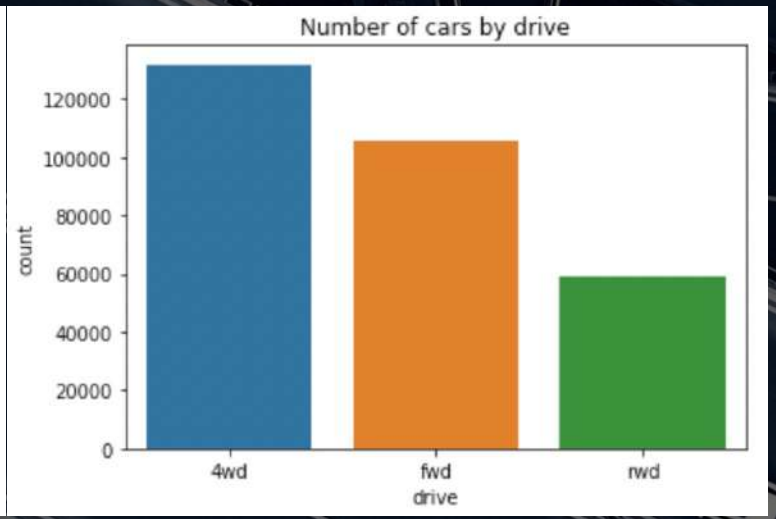

Number of cars by manufacturer
count
manufacturer
ford
chevrolet
toyota
honda
nissan
jeep
ram
gmc
bmw
dodge
mercedes-benz
hyundai
subaru
volkswagen
kia
lexus
audi
cadillac
chrysler
acura
buick
mazda
infiniti
lincoln
volvo
mitsubishi
mini
pontiac
mercury
jaguar
porsche
saturn
alfa-romeo
tesla
fiat
harley-davidson
ferrari
datsun
aston-martin
land rover
morgan

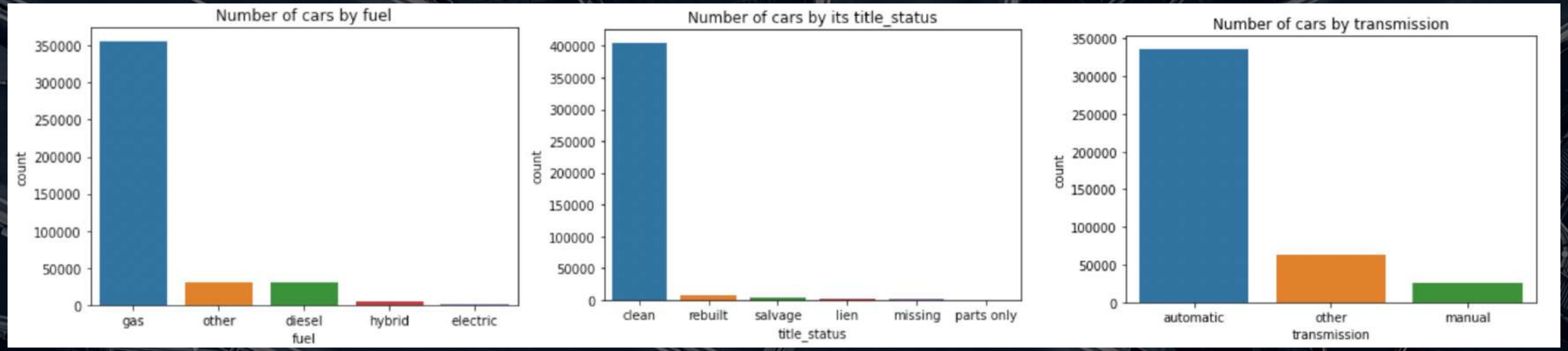

Number of cars by year
count
year

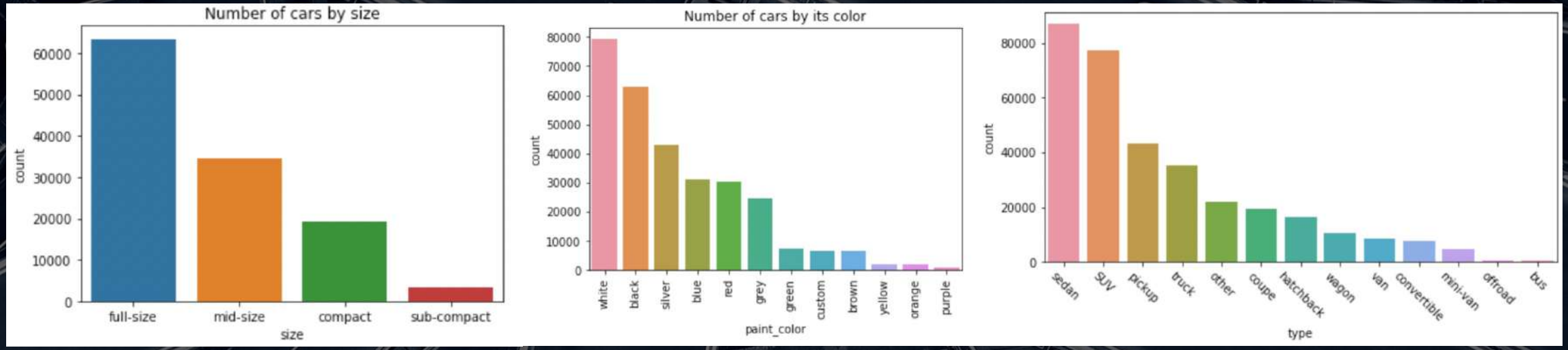

Number of cars by odometer
1e−6
Density
odometer


Number of cars by number of cylinders
count
cylinders
6 cylinders
4 cylinders
8 cylinders
5 cylinders
10 cylinders
other
3 cylinders
12 cylinders


Number of cars by condition
count
condition
good
excellent
like new
fair
new
salvage


Number of cars by drive
count
drive
4wd
fwd
rwd


Number of cars by fuel
count
fuel
gas
other
diesel
hybrid
electric
Number of cars by its title_status
count
title_status
clean
rebuilt
salvage
lien
missing
parts only
Number of cars by transmission
count
transmission
automatic
other
manual


Number of cars by size
count
size
full-size
mid-size
compact
sub-compact
Number of cars by its color
count
paint_color
white
black
silver
blue
red
grey
green
custom
brown
yellow
orange
purple
count
type
sedan
SUV
pickup
truck
other
coupe
hatchback
wagon
van
convertible
mini-van
offroad
bus

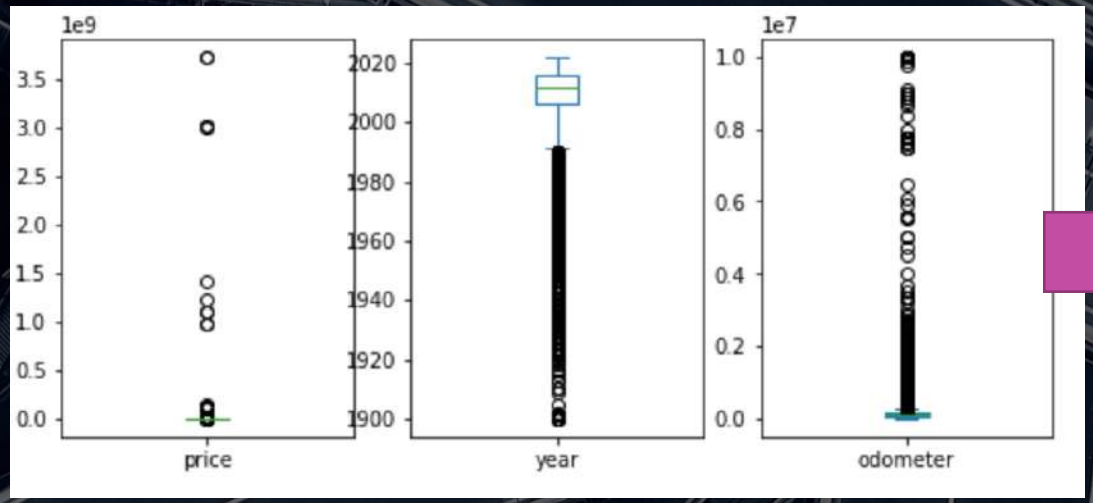
데이터 전처리

데이터 전처리

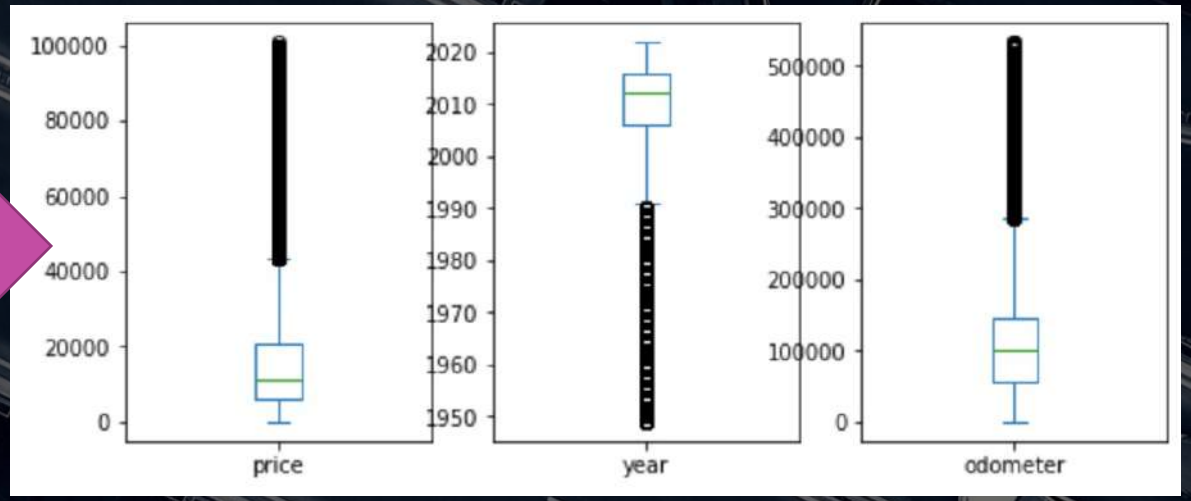
1) 중복값 및 이상치 제거

```
[ ]  #중복행 삭제
     vehicles.drop(vehicles[vehicles.duplicated()].index,inplace=True)

[ ]  #가격이 0 인 값 삭제
     vehicles.drop(vehicles['price'][vehicles['price']==0].index, inplace=True)
```

<이상치 제거 전>

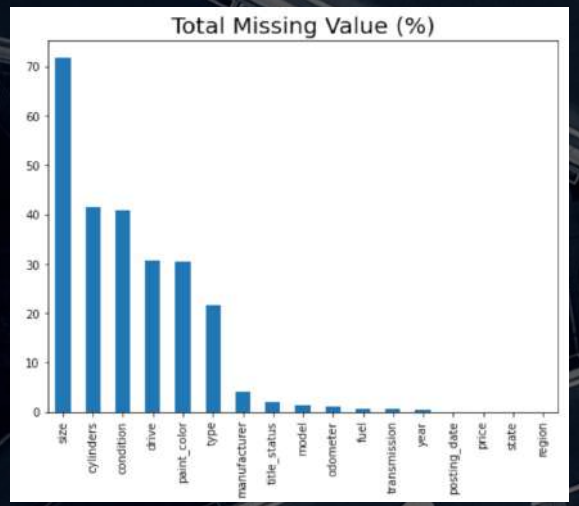


<이상치 제거 후>



데이터 전처리

3) 결측치 처리



size	71.77
cylinders	41.62
condition	40.79
drive	30.59
paint_color	30.50
type	21.75
manufacturer	4.13
title_status	1.93
model	1.24
odometer	1.03
fuel	0.71
transmission	0.60
year	0.28
posting_date	0.02
price	0.00
state	0.00
region	0.00

dtype: float64

1. 결측치가 70%이상 : size → 열 삭제

2. 결측치가 20% ~ 50% : Cylinders 등 5개 → datawig

3. 결측치가 20%미만 : manufactuers 등 8개 → 행 삭제

데이터 전처리

- Datawig ?

범주형 변수의 결측치를 대치해주는 딥러닝 기반 Imputer Library

```
[ ]  import datawig
     imputer = datawig.SimpleImputer(input_columns = ['model'],
                                     output_column = 'cylinders')
     imputer.fit(train_df = vehicles, num_epochs=8, batch_size=500)
```

Cramer's V

Taget column과 상관성인 높은 column

결측치를 대치해주고자 하는
Target column

데이터 전처리

3) 결측치 처리

열 삭제 — 1. 결측치가 70%이상 : size

```
[ ]  vehicles.drop(['size'],axis=1,inplace=True)
```

행 삭제 — 2. 결측치가 20%미만 : manufactuers 등 8개

```
[ ]  vehicles.dropna(subset=['year', 'manufacturer', 'model', 'fuel', 'odometer', 'title_status', 'transmission'],axis=0,inplace=True)
```

datawig — 3. 결측치가 20% ~ 50% : Cylinders 등 5개

Cramer's V 후 datawig 사용

데이터 전처리

3) 결측치 처리

```python
from scipy.stats import chi2_contingency

#범주형 변수간 상관계수의 척도로 사용되는 Cramer's V값을 계산해주는 함수 생성

def cramers_V(var1,var2) :
  crosstab =np.array(pd.crosstab(var1,var2, rownames=None, colnames=None))
  stat = chi2_contingency(crosstab)[0]
  obs = np.sum(crosstab)
  mini = min(crosstab.shape)-1
  return (stat/(obs*mini))
```

	condition	paint_color	cylinders	drive	type
manufacturer	0.00	0.01	0.12	0.19	0.07
model	0.16	0.14	0.56	0.66	0.56
condition	1.00	0.00	0.00	0.00	0.01
cylinders	0.00	0.00	1.00	0.15	0.06
fuel	0.01	0.00	0.05	0.02	0.05
title_status	0.02	0.00	0.00	0.00	0.00
transmission	0.05	0.01	0.01	0.01	0.05
drive	0.00	0.01	0.15	1.00	0.28
type	0.01	0.01	0.06	0.28	1.00
paint_color	0.00	1.00	0.00	0.01	0.01

Cylinders, drive, type : datawig를 통한 결측치 처리

Condition, paint_color : 각 클래스별 비율에 따른 결측치 처리

데이터 전처리

3) 결측치 처리

Datawig를 통한 결측치 처리 : cylinders, drive, type

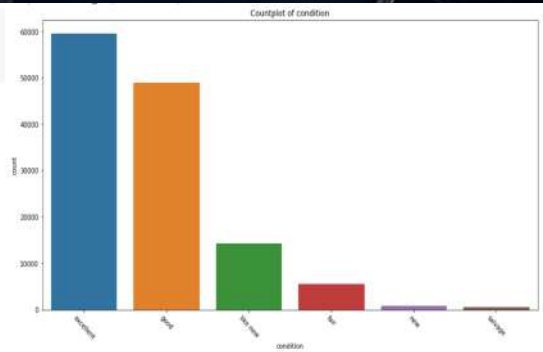
```
[ ] import datawig
    imputer = datawig.SimpleImputer(input_columns = ['model'],
                                    output_column = 'cylinders')
    imputer.fit(train_df = vehicles, num_epochs=8, batch_size=500)
    null_train = vehicles[vehicles['cylinders'].isnull()]
    null_imputed = imputer.predict(null_train)
    imputed_cylinders = pd.DataFrame(null_imputed)
```

```
2022-02-22 03:14:27,780 [INFO]  Epoch[0] Time cost=142.321
2022-02-22 03:14:27,792 [INFO]  Saved checkpoint to "cylinders/model-0000.params"
2022-02-22 03:14:40,126 [INFO]  Epoch[0] Validation-cross-entropy=0.604834
2022-02-22 03:14:40,129 [INFO]  Epoch[0] Validation-cylinders-accuracy=0.815429
2022-02-22 03:15:47,044 [INFO]  Epoch[1] Batch [0-124]  Speed: 934.31 samples/sec
2022-02-22 03:16:53,977 [INFO]  Epoch[1] Train-cross-entropy=0.543874
2022-02-22 03:16:53,979 [INFO]  Epoch[1] Train-cylinders-accuracy=0.826419
2022-02-22 03:16:53,986 [INFO]  Epoch[1] Time cost=133.850
2022-02-22 03:16:53,994 [INFO]  Saved checkpoint to "cylinders/model-0001.params"
2022-02-22 03:17:06,358 [INFO]  Epoch[1] Validation-cross-entropy=0.498854
2022-02-22 03:17:06,363 [INFO]  Epoch[1] Validation-cylinders-accuracy=0.831500
2022-02-22 03:18:13,114 [INFO]  Epoch[2] Batch [0-124]  Speed: 936.21 samples/sec
2022-02-22 03:19:17,711 [INFO]  Epoch[2] Train-cross-entropy=0.473639
2022-02-22 03:19:17,718 [INFO]  Epoch[2] Train-cylinders-accuracy=0.839952
2022-02-22 03:19:17,720 [INFO]  Epoch[2] Time cost=131.354
2022-02-22 03:19:17,731 [INFO]  Saved checkpoint to "cylinders/model-0002.params"
```

각 클래스별 비율을 통한 결측치 처리 : condition, paint_color

```
[ ] round(vehicles_imp['condition'].value_counts(normalize=True),2)
```

```
excellent    0.46
good         0.38
like new     0.11
fair         0.04
new          0.01
salvage      0.00
Name: condition, dtype: float64
```




```
[ ] vehicles_imp['condition'] = vehicles_imp['condition'].fillna(pd.Series(np.random.choice(['good', 'excellent', 'like new', 'fair', 'new', 'salvage'],
                                                                                            p=[0.46, 0.38, 0.11, 0.04, 0.01, 0.0],size=(len(vehicles_imp)))))
```

데이터 전처리

4) LabelEncoding

```
[ ] from sklearn import preprocessing
    vehicles_pre = vehicles_imp.copy()
    vehicles_pre[vehicles_pre.columns[vehicles_pre.dtypes=='O']] = vehicles_pre[vehicles_pre.columns[vehicles_pre.dtypes=='O']].apply(preprocessing.LabelEncoder
```

```
[ ] vehicles_pre.head()
```

	Unnamed: 0	price	year	manufacturer	model	condition	cylinders	fuel	odometer	title_status	transmission	drive0	type	paint_color
0	27	33590	2014.0	14	16168	2	6	2	57923.0	0	2	0	8	11
1	28	22590	2010.0	7	16516	2	6	2	71229.0	0	2	0	8	1
2	29	39590	2020.0	7	16539	2	6	2	19160.0	0	2	0	8	9
3	30	30990	2017.0	38	19044	2	6	2	41124.0	0	2	0	8	9
4	31	15000	2013.0	13	8230	0	5	2	128000.0	0	0	2	10	0

모델 구축 및 구현 결과

모델 구축 및 구현

1) Trainset / Testset split

```
[7]  data = vehicles_pre.drop(['price'],axis=1)
     target = vehicles_pre['price']
```

0초

```
[8]  X_train, X_test, y_train, y_test = train_test_split(data,target)
```

0초

모델 구축 및 구현

1) Random Forest Regressor

```
rf_reg_params = {'n_estimators': [1,5,10,20],
                 'max_features':[1,3,5,7,9]}
print_best_params(rf_reg, rf_reg_params)
```

```
RandomForestRegressor 모델 5 CV시 최적 평균 RMSE: 6587.1585, 최적 alpha: {'max_features': 7, 'n_estimators': 20}
```

```
rf_reg = RandomForestRegressor(max_features=7, n_estimators=20)
rf_reg.fit(X_train,y_train)
```

```
RandomForestRegressor(max_features=7, n_estimators=20)
```

```
from sklearn.metrics import r2_score

#cross validation score 계산하기
cv_rf_reg = cross_val_score(estimator=rf_reg, X=X_train, y=y_train, cv=5)

#train set의 R2 score 계산하기
y_pred_rf_reg_train = rf_reg.predict(X_train)
r2_score_rf_reg_train = r2_score(y_train,y_pred_rf_reg_train)

#test set의 R2 score 계산하기
y_pred_rf_reg_test = rf_reg.predict(X_test)
r2_score_rf_reg_test = r2_score(y_test, y_pred_rf_reg_test)

#test set의 RMSE 계산하기
rmse_rf_reg = (np.sqrt(mean_squared_error(y_test, y_pred_rf_reg_test)))

#점수 종합
print('CV: ', cv_rf_reg.mean())
print('R2_score(train): ',r2_score_rf_reg_train)
print('R2_score(test): ',r2_score_rf_reg_test)
print('RMSE: ', rmse_rf_reg)
```

```
CV:  0.7979295073269188
R2_score(train):  0.9689481221032828
R2_score(test):  0.8012117035293344
RMSE:  6597.737350104824
```

R2_score(test) : 0.801

모델 구축 및 구현

2) Gradient Boosting Regressor

```
[ ] gb_reg_params = {'max_depth': [5,10,20,30],
                     'learning_rate': [0.01,0.05,0.1]}
    print_best_params(gb_reg, gb_reg_params)

    GradientBoostingRegressor 모델 5 CV시 최적 평균 RMSE: 6139.6477, 최적 alpha: {'learning_rate': 0.1, 'max_depth': 10}
```

```
[ ] gb_reg = GradientBoostingRegressor(max_depth=10, learning_rate=0.1)
    gb_reg.fit(X_train,y_train)

    GradientBoostingRegressor(max_depth=10)
```

```
[37] from sklearn.metrics import r2_score

     #cross validation score 계산하기
     cv_gb_reg = cross_val_score(estimator=gb_reg, X=X_train, y=y_train, cv=5)

     #train set의 R2 score 계산하기
     y_pred_gb_reg_train = gb_reg.predict(X_train)
     r2_score_gb_reg_train = r2_score(y_train,y_pred_gb_reg_train)

     #test set의 R2 score 계산하기
     y_pred_gb_reg_test = gb_reg.predict(X_test)
     r2_score_gb_reg_test = r2_score(y_test, y_pred_gb_reg_test)

     #test set의 RMSE 계산하기
     rmse_gb_reg = (np.sqrt(mean_squared_error(y_test, y_pred_gb_reg_test)))

     #점수 종합
     print('CV: ', cv_gb_reg.mean())
     print('R2_score(train): ',r2_score_gb_reg_train)
     print('R2_score(test): ',r2_score_gb_reg_test)
     print('RMSE: ', rmse_gb_reg)

     new_row = {"Model": "GradientBoostingRegressor", "RMSE": rmse_gb_reg,
                'R2 Score(train)' : r2_score_gb_reg_train, 'R2 Score(test)': r2_score_gb_reg_test}
     models = models.append(new_row, ignore_index = True)

     CV:  0.8027333725565109
     R2_score(train):  0.9024687933975384
     R2_score(test):  0.8139194718550168
     RMSE:  5778.772015097007
```

R2_score(test) : 0.814

모델 구축 및 구현

3) XGB Regressor

```
[ ]  xgb_reg = xgb.fit(X_train, y_train)
```

```
[ ]  from xgboost import XGBRegressor

     xgb = XGBRegressor(boosting_type='gbdt',learning_rate=0.1,subsample=0.8,colsample_bytree=0.6,mid_data_in_leaf=800, metric='rmse')
```

```
[ ]  from sklearn.metrics import r2_score

     #cross validation score 계산하기
     #cv_xgb_reg = cross_val_score(estimator=xgb_reg, X=X_train, y=y_train, cv=5)

     #train set의 R2 score 계산하기
     y_pred_xgb_reg_train = xgb_reg.predict(X_train)
     r2_score_xgb_reg_train = r2_score(y_train,y_pred_xgb_reg_train)

     #test set의 R2 score 계산하기
     y_pred_xgb_reg_test = xgb_reg.predict(X_test)
     r2_score_xgb_reg_test = r2_score(y_test, y_pred_xgb_reg_test)

     #test set의 RMSE 계산하기
     rmse_xgb_reg = (np.sqrt(mean_squared_error(y_test, y_pred_xgb_reg_test)))

     #점수 종합
     #print('CV: ', cv_xgb_reg.mean())
     print('R2_score(train): ',r2_score_xgb_reg_train)
     print('R2_score(test): ',r2_score_xgb_reg_test)
     print('RMSE: ', rmse_xgb_reg)
```

```
R2_score(train):  0.7621957380709053
R2_score(test):  0.755352575388196
RMSE:  7319.3066177107075
```

R2_score(test) : 0.755

모델 구축 및 구현

10) Light GBM Regressor

```
[9] import lightgbm as lgb

    train_set = lgb.Dataset(X_train,y_train)
    test_set = lgb.Dataset(X_test,y_test)
```

```
params = {
    'boosting_type':'dart', #gbdt대신 정확도에 민감한 dart
    'objective':'regression',
    'learning_rate':0.1,
    'subsample':0.8,
    'bagging_fraction':1,
    'max_bin':5000,
    'bagging_freq':20,
    'colsample_bytree':0.6,
    'mid_data_in_leaf':800,
    'metric':'rmse'
}

lgb_reg = lgb.train(params, train_set, num_boost_round=10000,
                    early_stopping_rounds=5000, verbose_eval=500,valid_sets=test_set)
```

```
/usr/local/lib/python3.7/dist-packages/lightgbm/callback.py:189: UserWarning: Early s
  warnings.warn('Early stopping is not available in dart mode')
[500]   valid_0's rmse: 6247.52
[1000]  valid_0's rmse: 6090.9
[1500]  valid_0's rmse: 6007.5
```

```
[ ] from sklearn.metrics import r2_score

    #cross validation score 계산하기
    #cv_lgb_reg = cross_val_score(estimator=lgb_reg, X=X_train, y=y_train, cv=5)

    #train set의 R2 score 계산하기
    y_pred_lgb_reg_train = lgb_reg.predict(X_train)
    r2_score_lgb_reg_train = r2_score(y_train,y_pred_lgb_reg_train)

    #test set의 R2 score 계산하기
    y_pred_lgb_reg_test = lgb_reg.predict(X_test)
    r2_score_lgb_reg_test = r2_score(y_test, y_pred_lgb_reg_test)

    #test set의 RMSE 계산하기
    rmse_lgb_reg = (np.sqrt(mean_squared_error(y_test, y_pred_lgb_reg_test)))

    #점수 종합
    #print('CV: ', cv_lgb_reg.mean())
    print('R2_score(train): ',r2_score_lgb_reg_train)
    print('R2_score(test): ',r2_score_lgb_reg_test)
    print('RMSE: ', rmse_lgb_reg)
```

```
R2_score(train):  0.9734838804608775
R2_score(test):  0.8502137583586867
RMSE:  5727.1107114195775
```

R2_score(test) : 0.8502


감사합니다