중고차 가격 예측

학과: 컴퓨터 공학과
학번: 2019108248
이름: 고민석

목차

1. 목적

A. 주어진 중고차의 데이터를 전처리 해본다.

B. 도구를 통해 데이터간 다양한 상관관계를 알아본다.

C. 알고리즘을 사용한 학습을 통해 최대한의 예측을 해본다.

2. 데이터의 특성

- 데이터 Information

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

```
train_data = pd.read_csv('/Users/kominseok/Downloads/archive (1)/train-data.csv')
test_data = pd.read_csv('/Users/kominseok/Downloads/archive (1)/test-data.csv')
```

```
train_data.info()
train_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5872 entries, 0 to 5871
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Location           5872 non-null   object
 1   Year               5872 non-null   int64
 2   Kilometers_Driven  5872 non-null   int64
 3   Fuel_Type          5872 non-null   object
 4   Transmission       5872 non-null   object
 5   Owner_Type         5872 non-null   int64
 6   Seats              5872 non-null   float64
 7   Price              5872 non-null   float64
 8   Mileage(km/kg)     5872 non-null   float64
 9   Engine(CC)         5872 non-null   float64
 10  Power(bhp)         5872 non-null   float64
 11  New_car_Price      823 non-null    float64
dtypes: float64(6), int64(3), object(3)
memory usage: 550.6+ KB
```

```
train_data.shape
```

```
(6019, 13)
```

- 데이터셋에는 13가지의 칼럼과 6019개의 가격 값 데이터가 있다. (지역, 가격, 주행거리, 연료타입 등)

- 각 칼럼의 자료형은 다양하고, 전처리 과정에서 자료형 변형 및 더미화 필요성

2. 데이터의 특성

- 데이터 자세히 보기

```
train_data.tail()
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
6014	6014	Maruti Swift VDI	Delhi	2014	27365	Diesel	Manual	First	28.4 kmpl	1248 CC	74 bhp	5.0	7.88 Lakh	4.75
6015	6015	Hyundai Xcent 1.1 CRDi S	Jaipur	2015	100000	Diesel	Manual	First	24.4 kmpl	1120 CC	71 bhp	5.0	NaN	4.00
6016	6016	Mahindra Xylo D4 BSIV	Jaipur	2012	55000	Diesel	Manual	Second	14.0 kmpl	2498 CC	112 bhp	8.0	NaN	2.90
6017	6017	Maruti Wagon R VXI	Kolkata	2013	46000	Petrol	Manual	First	18.9 kmpl	998 CC	67.1 bhp	5.0	NaN	2.65
6018	6018	Chevrolet Beat Diesel	Hyderabad	2011	47000	Diesel	Manual	First	25.44 kmpl	936 CC	57.6 bhp	5.0	NaN	2.50

```
print(train_data['Location'].unique())
```

```
['Mumbai' 'Pune' 'Chennai' 'Coimbatore' 'Hyderabad' 'Jaipur' 'Kochi'
 'Kolkata' 'Delhi' 'Bangalore' 'Ahmedabad']
```

- 이 데이터는 인도에서 수집한 데이터임을 알 수 있음

2. 데이터의 특성

- 데이터 자세히 보기

```
train_data.tail()
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
6014	6014	Maruti Swift VDI	Delhi	2014	27365	Diesel	Manual	First	28.4 kmpl	1248 CC	74 bhp	5.0	7.88 Lakh	4.75
6015	6015	Hyundai Xcent 1.1 CRDi S	Jaipur	2015	100000	Diesel	Manual	First	24.4 kmpl	1120 CC	71 bhp	5.0	NaN	4.00
6016	6016	Mahindra Xylo D4 BSIV	Jaipur	2012	55000	Diesel	Manual	Second	14.0 kmpl	2498 CC	112 bhp	8.0	NaN	2.90
6017	6017	Maruti Wagon R VXI	Kolkata	2013	46000	Petrol	Manual	First	18.9 kmpl	998 CC	67.1 bhp	5.0	NaN	2.65
6018	6018	Chevrolet Beat Diesel	Hyderabad	2011	47000	Diesel	Manual	First	25.44 kmpl	936 CC	57.6 bhp	5.0	NaN	2.50

- 필요 없는 칼럼 삭제

- 단위를 빼 자료형을 실수형으로 변환 (실수형으로 변환 해야 정확도가 올라감)

2. 데이터의 특성

- 칼럼과 가격과의 상관관계 예측 해보기

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

```
train_data = pd.read_csv('/Users/kominseok/Downloads/archive (1)/train-data.csv')
test_data = pd.read_csv('/Users/kominseok/Downloads/archive (1)/test-data.csv')
```

```
train_data.info()
train_data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5872 entries, 0 to 5871
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Location            5872 non-null   object
 1   Year                5872 non-null   int64
 2   Kilometers_Driven   5872 non-null   int64
 3   Fuel_Type           5872 non-null   object
 4   Transmission        5872 non-null   object
 5   Owner_Type          5872 non-null   int64
 6   Seats               5872 non-null   float64
 7   Price               5872 non-null   float64
 8   Mileage(km/kg)      5872 non-null   float64
 9   Engine(CC)          5872 non-null   float64
 10  Power(bhp)          5872 non-null   float64
 11  New_car_Price       823 non-null    float64
dtypes: float64(6), int64(3), object(3)
memory usage: 550.6+ KB
```

1. Location
2. Year
3. Kilometers_Driven
4. Fuel_Type
5. Transmission
6. Owner_Type
7. Seats
8. Price
9. Mileage(km/kg)
10. Engine(CC)
11. Power(bhp)
12. New_car_Price

- 나머지 요소의 상관관계를 알기 위해 벡터화의 필요성을 느낌

2. 데이터의 특성

- 결측값 확인

```
train_data.isnull().sum()
```

```
Name                    0
Location                0
Year                    0
Kilometers_Driven       0
Fuel_Type               0
Transmission            0
Owner_Type              0
Mileage                 2
Engine                 36
Power                  36
Seats                  42
New_Price            5195
Price                   0
dtype: int64
```

결측값이 Engin, Power, Seats, New_Price에서 주로 발생 되었고, 신차 가격에 압도적인 결측값이 존재함

Engin, Power, Seats을 수정하고, New_Price 칼럼을 삭제해야함

2. 데이터의 특성

- 데이터셋의 개선방안

a. 필요 없는 칼럼 삭제

b. 새로운 상관관계를 알기 위한 새로운 칼럼 추가 (회사명)

c. 결측값이 있는 Engin, Power, Seats을 수정하고, 결측값이 압도적으로 많은 New_Price 칼럼을 삭제

d. 자료형 변환

e. 상관관계를 알기 위해 칼럼들의 벡터 및 더미화

3. 데이터 전처리

a. 필요 없는 칼럼 삭제

```
train_data.tail()
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
6014	6014	Maruti Swift VDI	Delhi	2014	27365	Diesel	Manual	First	28.4 kmpl	1248 CC	74 bhp	5.0	7.88 Lakh	4.75
6015	6015	Hyundai Xcent 1.1 CRDi S	Jaipur	2015	100000	Diesel	Manual	First	24.4 kmpl	1120 CC	71 bhp	5.0	NaN	4.00
6016	6016	Mahindra Xylo D4 BSIV	Jaipur	2012	55000	Diesel	Manual	Second	14.0 kmpl	2498 CC	112 bhp	8.0	NaN	2.90
6017	6017	Maruti Wagon R VXI	Kolkata	2013	46000	Petrol	Manual	First	18.9 kmpl	998 CC	67.1 bhp	5.0	NaN	2.65
6018	6018	Chevrolet Beat Diesel	Hyderabad	2011	47000	Diesel	Manual	First	25.44 kmpl	936 CC	57.6 bhp	5.0	NaN	2.50

```
train_data = train_data.iloc[:,1:]
train_data.head()
```

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Se

iloc()함수를 통해 unnamed 칼럼을 빼서 읽어오고 다시 변수에 대입

3. 데이터 전처리

b. 새로운 상관관계를 알기 위한 새로운 칼럼 추가 (회사명)

```
train_data.tail()
```

	Unnamed: 0	Name	Location
6014	6014	Maruti Swift VDI	Delhi
6015	6015	Hyundai Xcent 1.1 CRDi S	Jaipur
6016	6016	Mahindra Xylo D4 BSIV	Jaipur
6017	6017	Maruti Wagon R VXI	Kolkata
6018	6018	Chevrolet Beat Diesel	Hyderabad

Name 컬럼 안 데이터들은 회사명 + 모델명으로 구성

```
for i in range(train_data.shape[0]):
    train_data.at[i, 'Company'] = train_data['Name'][i].split()[0]
```

split() 함수를 통해 앞 회사명을 분리하고 새로운 Company 컬럼에 데이터 추가

Company
Maruti
Hyundai
Honda
Maruti
Audi

3. 데이터 전처리

c. 결측값이 있는 컬럼 수정 및 삭제

```
print("결측값 수정 전",train_data.shape)
train_data = train_data[train_data['Mileage'].notna()]
print("Mileage 결측값 수정 후 ",train_data.shape)
train_data = train_data[train_data['Engine'].notna()]
print("Engine 결측값 수정 후  ",train_data.shape)
train_data = train_data[train_data['Power'].notna()]
print("Power 결측값 수정 후  ",train_data.shape)
train_data = train_data[train_data['Seats'].notna()]
print("Seats 결측값 수정 후  ",train_data.shape)
```

```
결측값 수정 전 (5975, 14)
Mileage 결측값 수정 후  (5975, 14)
Engine 결측값 수정 후   (5975, 14)
Power 결측값 수정 후   (5975, 14)
Seats 결측값 수정 후   (5975, 14)
```

notna() 함수를 통해 결측값이 있으면 false 반환하고 결측값 삭제

3. 데이터 전처리

d. 자료형 변환

Mileage	Engine	Power
26.6 km/kg	998 CC	58.16 bhp
19.67 kmpl	1582 CC	126.2 bhp
18.2 kmpl	1199 CC	88.7 bhp
20.77 kmpl	1248 CC	88.76 bhp
15.2 kmpl	1968 CC	140.8 bhp

```
for i in range(train_data.shape[0]):
    train_data.at[i, 'Mileage(km/kg)'] = train_data['Mileage'][i].split()[0]
    train_data.at[i, 'Engine(CC)'] = train_data['Engine'][i].split()[0]
    train_data.at[i, 'Power(bhp)'] = train_data['Power'][i].split()[0]
```

```
train_data['Mileage(km/kg)'] = train_data['Mileage(km/kg)'].astype(float)
train_data['Engine(CC)'] = train_data['Engine(CC)'].astype(float)
```

1. split() 함수를 사용해 실수 값과 단위 분리시키고 단위를 포함한 새로운 칼럼에 실수 값 대입
2. astype() 실수형으로 자료형 변환

Mileage(km/kg)	Engine(CC)	Power(bhp)
26.60	998.0	58.16
19.67	1582.0	126.20
18.20	1199.0	88.70
20.77	1248.0	88.76
15.20	1968.0	140.80

3. 데이터 전처리

d. 상관관계를 알기 위해 칼럼들의 벡터 및 더미화

```
var = 'Location'
Location = test_data[[var]]
Location = pd.get_dummies(Location,drop_first=True)
Location.head()

var = 'Fuel_Type'
Fuel_t = test_data[[var]]
Fuel_t = pd.get_dummies(Fuel_t,drop_first=True)
Fuel_t.head()

var = 'Transmission'
Transmission = test_data[[var]]
Transmission = pd.get_dummies(Transmission,drop_first=True)
Transmission.head()
```

```
Location.head()
```

atore	Location_Delhi	Location_Hyderabad	Location_Jaipur
	0	0	0
	0	0	0
	0	0	0
	0	0	0
	0	0	0

```
Fuel_t.head()
```

	Fuel_Type_Diesel	Fuel_Type_LPG	Fuel_Type_Petrol
0	0	0	0
1	1	0	0
2	0	0	1
3	1	0	0
4	1	0	0

```
Transmission.head()
```

	Transmission_Manual
0	1
1	1
2	1
3	1
4	0

4. 데이터 시각화

a. 가격 - 오너 타입

b. 가격 - 연료타입 그래프

c. 가격 - 차량연식

d. 가격 - 구동방식

e. 가격 - 회사

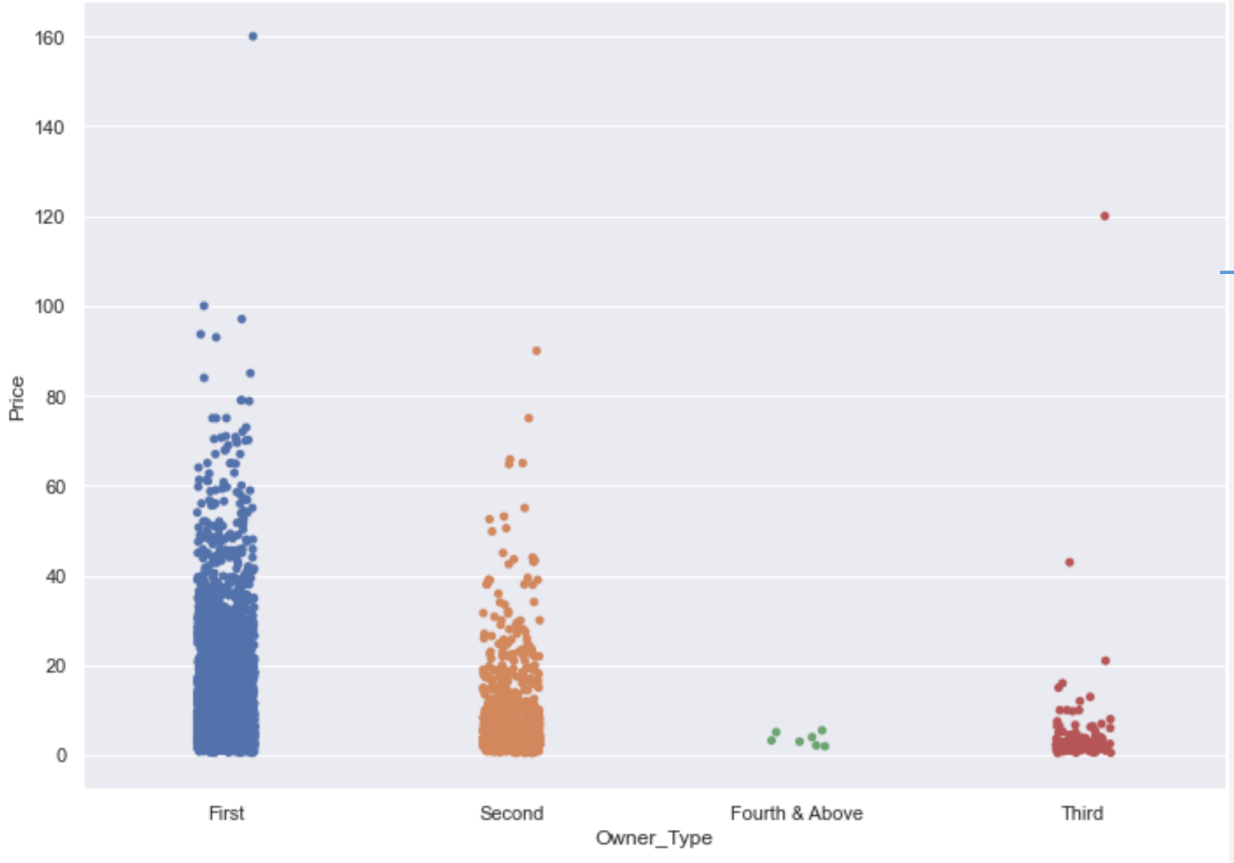
f. 히트맵

4. 데이터 시각화

a. 가격- 오너타입

```
var = 'Owner_Type'
fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.stripplot(x = var, y ='Price', data = train_data)
```

```
<AxesSubplot:xlabel='Owner_Type', ylabel='Price'>
```



First 오너의 자동차가 가장 비싸고 차례대로 가격이 낮아짐

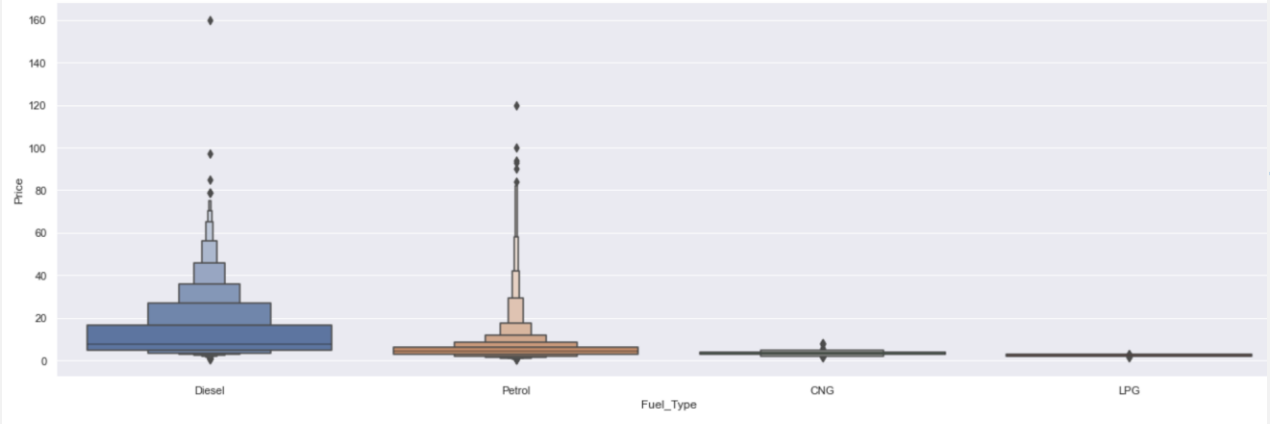
-> 인도의 계급 구조를 나타내는 것으로 추론

4. 데이터 시각화

b. 가격 - 연료타입 그래프

```
sns.catplot(y='Price',x=var,data= train_data.sort_values('Price',ascending=False),kind="boxen",height=6, aspect=3)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

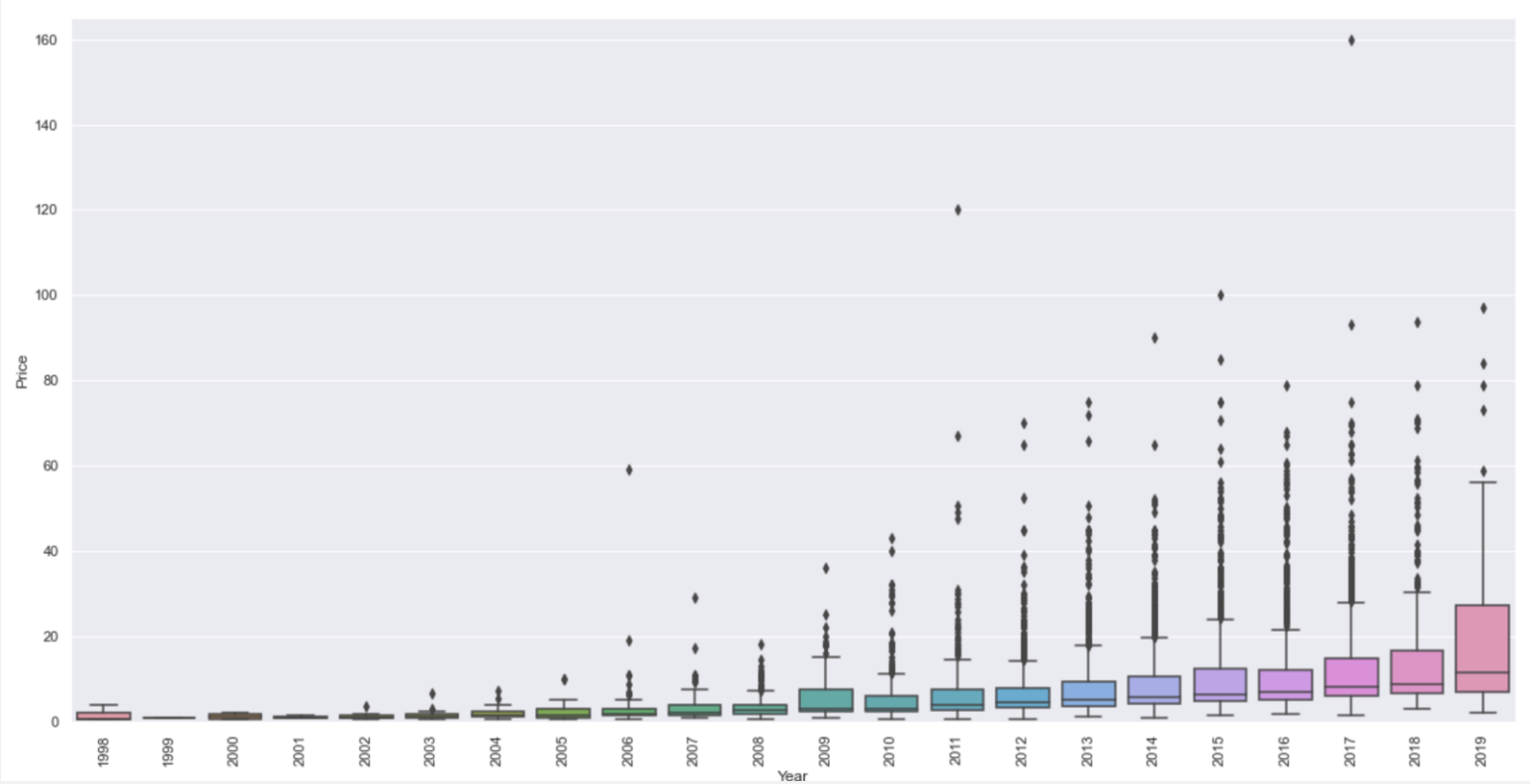


디젤 차량이 수도 많고 상대적으로 가격이 비쌈

4. 데이터 시각화

c. 가격 - 차량연식

```
var = 'Year'
data = pd.concat([train_data['Price'], train_data[var]], axis=1)
f, ax = plt.subplots(figsize=(20, 10))
fig = sns.boxplot(x=var, y="Price", data=data)
fig.axis(ymin=0, ymax=165);
plt.xticks(rotation=90);
```



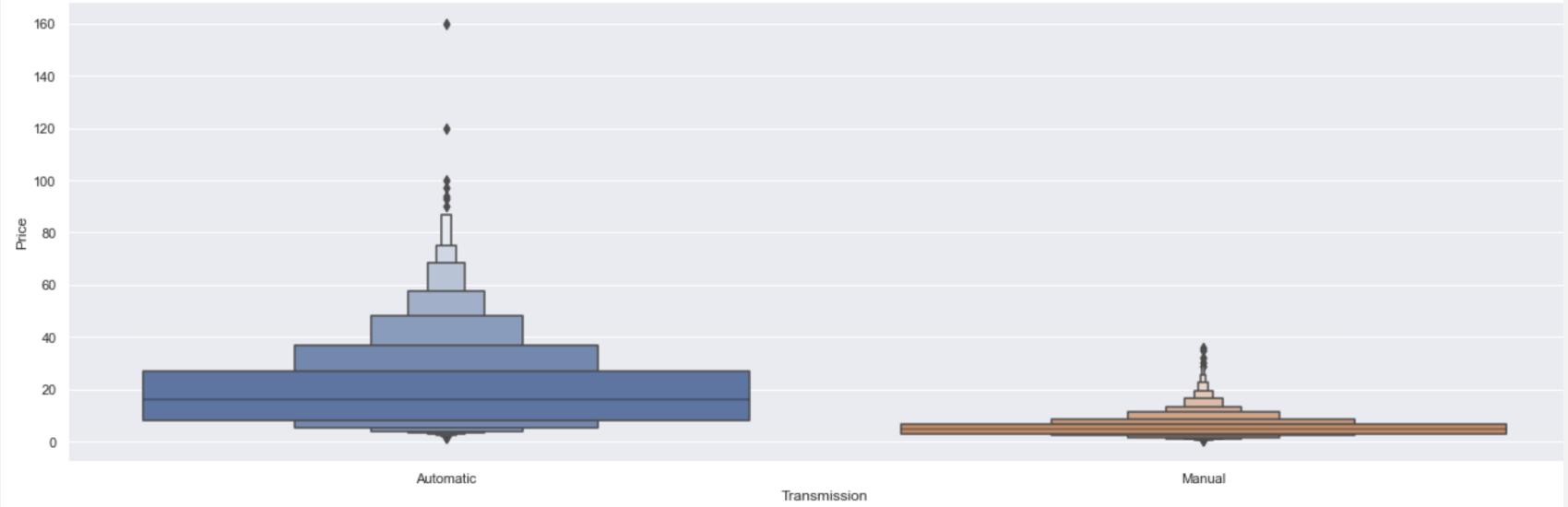
당연하게도 차량의 연식이 오래될수록 가격은 낮아짐

4. 데이터 시각화

d. 가격 - 구동방식

```
sns.catplot(y='Price',x=var,data= train_data.sort_values('Price',ascending=False),kind="boxen",height=6, aspect=3)
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



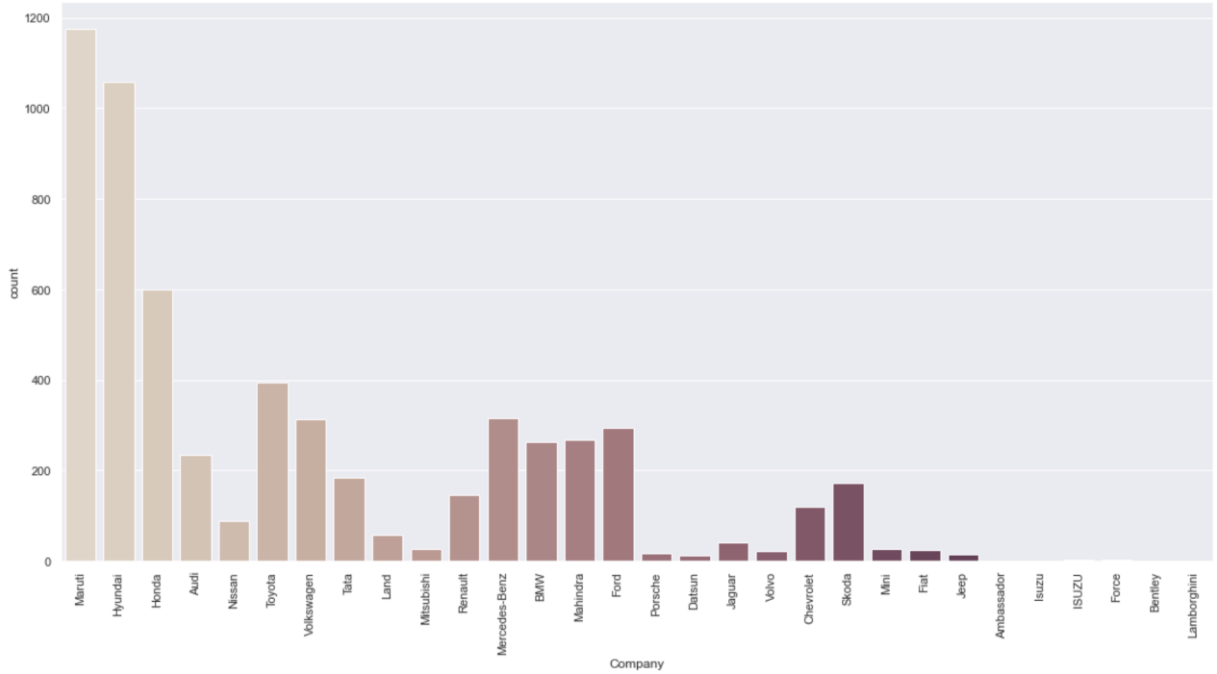
오토 차량이 신차이므로 가격이 비싸다

4. 데이터 시각화

e. 가격 - 회사

```
var = "Company"
plt.figure(figsize=(20, 10))
sns.catplot(x=var, kind="count", palette="ch:.25", height=8, aspect=2, data=train_data);
plt.xticks(rotation=90);
```

```
<Figure size 1440x720 with 0 Axes>
```



Maruti와 현대가 가장 많은 비율을 차지하고 근소하게 Maruti가 앞서고 있다

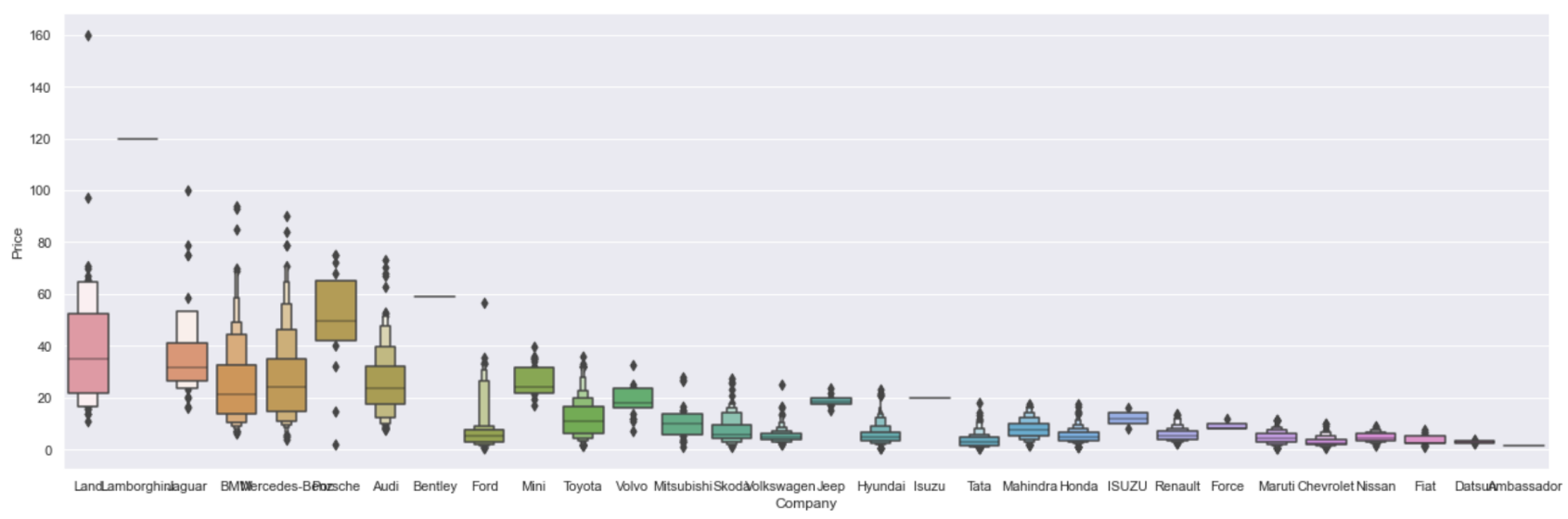
.

-> 인도에서는 현대 차량이 많이 수출되고 있음을 알 수 있다,

4. 데이터 시각화

e. 가격 - 회사

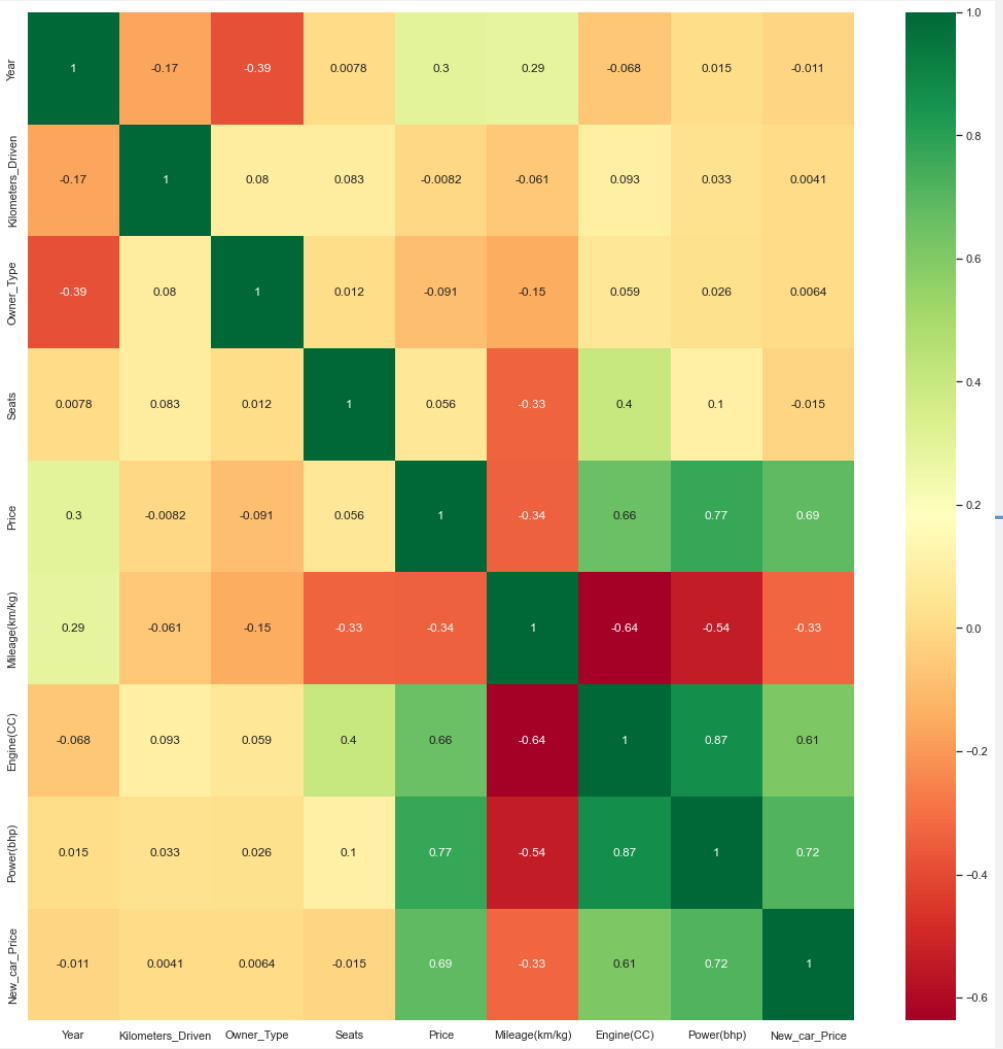
```
sns.catplot(y='Price',x=var,data= train_data.sort_values('Price',ascending=False),kind="boxen",height=6, aspect=3)
plt.show
train_data.drop(["Company"],axis=1,inplace=True)
```



신차가격이 높은 BMW, Audi, Benz 외제 차량이 가격이 높게 형성 되어 있다.

4. 데이터 시각화

f. 히트맵



예상 상관관계

1. Location	
2. Year	적중
3. Kilometers_Driven	적중
4. Fuel_Type	
5. Transmission	
6. Owner_Type	
7. Seats	
8. Price	
9. Mileage(km/kg)	미적중
10. Engine(CC)	적중
11. Power(bhp)	적중
12. New_car_Price	적중

- 중고차 가격에서 가장 높은 상관관계를 보여주는건 엔진, 마력, 신차가격이다.

- 엔진과 연비등 다양한 상관관계도 확인 가능하다.

5. 데이터 예측

a. LinearRegression

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 25
```

```
from sklearn.linear_model import LinearRegression
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
y_pred= linear_reg.predict(X_test)
print("테스트 정확도",linear_reg.score(X_test,y_test))
```

```
테스트 정확도 0.6991016530826976
```

b. RandomForestRegressor모델

```
from sklearn.ensemble import RandomForestRegressor
rf_reg = RandomForestRegressor()
rf_reg.fit(X_train, y_train)
y_pred= rf_reg.predict(X_test)
print("테스트 정확도",rf_reg.score(X_test,y_test))
```

```
/var/folders/02/jnfcdbfd4273wt0bb8p_n6xc0000gn/T/ipyl
tor y was passed when a 1d array was expected. Please
().
  rf_reg.fit(X_train, y_train)
```

```
테스트 정확도 0.9105839954803825
```

6. 느낀점

인공지능은 어렵다고 생각 하고 있었는데 이번 기회에 데이터 셋을 분석해보면서 인공지능은 생각보다 어렵지 않고, 오히려 데이터 전처리 과정이 더 복잡하다 생각했습니다. 다양한 전처리 기술을 알면 인공지능 학습에 박차를 가할 수 있다 생각했습니다.

Q&A