인도 중고차 시장 공략 및 핵심인자 도출과 경쟁력 확보

중고차 가격 예측 보고서

C반 2조 김병우/김준혁/김예원/배빛나리/신지우/심정욱

목차

CONTETNS

01 과제 정의 및 목표 왜 POS_CAR는 인도 중고차 시장을 공략해야 하는가?

중고차 수요 증가로 인한 시장 확대

〈THE TIMES OF INDIA, 아시아타임즈 (2024년)〉

현대차 크레타, 인도 중고차 시장 1위... 신차 출고가보다 더 비싸

The Indian used car market is expanding due to rising demand for luxury vehicles. The sale of used luxury cars increased by 20%.

Due to the high cost of a luxury car, it was difficult to obtain one until recently. However, this trend is changing as consumers can now purchase used luxury vehicles. With easy access to financing options, annual maintenance contracts, and lower entry prices, the market is becoming more organized. Furthermore, the average age of used luxury vehicles entering the market is between 2 and 3 years, versus 5-6 years for a mid-size or small-scale vehicle, making them a better option in some cases.

- 중산층의 증가와 자동차의 보급률 상승, 금융 지원의 확대로 자가용의 수요 증가
- 전 세계 반도체 공급 부족 문제로 신차의 가격 급등
- 2024년 인도 중고차 시장 규모는 31억 달러이나, 2029년에는 63억 달러까지 증가할 것으로 예측

중고차 판매량 증가 전망

〈India Used Cars Growth Opportunities by EMIS (2023년)〉

- 타임즈 오브 인디아에 따르면, 인도 중고차 판매량은 연평균 성장률이 12.5%까지 성장할 것으로 예상
- 유럽과 같은 성숙 시장에서는 신차와 중고차의 비율이 1:3이지만, 인도는 아직 1:1.2이기에 미래 성장의 여지 존재

중고차 시장의 거래 시스템 미흡

〈중고차 판매상 유형별 판매량 비율 (2022년)〉

- 인도의 중고차 시장은 시스템이 자리잡지 않은 상태
- 독립 딜러가 48%를 점유하고, 직거래가 29%를 차지하고 있는 현황
- LVMC의 사례처럼 인도의 중고차 시장에 진입해 시장을 선점할 경우 상당한 이익을 얻을 것으로 전망

01 과제 정의 및 목표

과제 정의

1. 중고차 데이터셋에 대한 기술적 통계 분석 방안 설정
2. 중고차 데이터 내 변수들을 선정하여 통계적 수치를 활용한 정리 및 분석
3. 데이터 분석을 통해 중고차 가격에 주요 영향을 미치는 요소 파악
4. 가격 영향 요소를 바탕으로 경쟁 우위 및 수익 증대 방안 제작
5. 분석 과정 및 단계별 접근 방식 설명

목표

인도 중고차 시장에 진출하여 경쟁력 확보와 수익성 향상을 위해
중고차 가격을 효과적으로 예측할 수 있는 가격 예측 모델 개발과 핵심 영향 인자 도출

1. 도메인 정보 파악

Car.csv

〈데이터 열 목록〉

```
#   Column
---  ------
0   Brand
1   Location
2   Price
3   Year
4   Kilometers_Driven
5   Fuel_Type
6   Transmission
7   Owner_Type
8   Mileage
9   Engine
10  Power
11  Seats
```

[데이터 종류 설명]

Name : 자동차와 모델의 이름 (Audi Sedan S4)

Location : 자동차를 팔거나 구매한 위치 (Mumbai)

Price : 중고차의 가격 (목표변수)

Year : 모델의 년도 혹은 버전

Kilometers_Driven : 이전 소유주의 차량 주행거리(km)

Fuel_Type : 자동차의 사용 연료의 종류

Transmission : 자동차의 변속기 종류 (Automatic / Manual)

Owner_Type : 자동차가 몇 번 중고로 팔렸는지 횟수

Mileage : 표준 주행거리

Engine : 엔진의 배기량(cc)

Power : 엔진의 최대 출력(bhp)

Seats : 차의 좌석 수

New_Price : 신 제품의 가격

2. 데이터 특성 파악 후 전처리

[데이터 종류 및 타입]

```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Brand              6200 non-null   object
 1   Location           6200 non-null   object
 2   Price              6200 non-null   float64
 3   Year               6200 non-null   int64
 4   Kilometers_Driven  6200 non-null   int64
 5   Fuel_Type          6200 non-null   object
 6   Transmission       6200 non-null   object
 7   Owner_Type         6200 non-null   object
 8   Mileage            6200 non-null   float64
 9   Engine             6200 non-null   int64
 10  Power              6200 non-null   float64
 11  Seats              6200 non-null   int64
```

Brand, Location, Fuel_Type, Transmission
Owner_Type은 범주형이므로 변환이 필요

[범주형 데이터 종류]

[Brand]
Land Rover Freelander 2 TD4 SE
Mercedes-Benz C-Class Progressive ...

[Location]
Pune, Coimbatore, Bangalore, Mumbai ...

[Fuel_Type]
Diesel, Petrol, Electric, CNG, LPG

[Transmission]
Automatic, Manual

[Owner_Type]
First, Second, Third, Fourth & Above

그룹화로 파생변수 생성 및
One-Hot 인코딩 적용

3. 범주형 변수의 그룹화

〈Brand의 파생변수 생성〉

브랜드별로 그룹화하여 가격 평균을 확인해본 결과 유의미한 가격의 차이가 존재하는 것을 확인

➔ 가격의 평균을 기준으로 3등분 해서 High / Middle / Low로 범주화 한다.

```
In [29]: df.groupby('Brand')['Price'].mean()
```

```
Out[29]: Brand
Audi              40017.252863
BMW               39660.763590
Chevrolet          4667.039752
Etc               17562.794643
Fiat               5837.610968
Ford              10626.892204
Honda              8523.602283
Hyundai            8394.511436
Jaguar            59961.551628
Jeep              29548.206842
Land              60183.243000
Mahindra          12439.209468
Maruti             7062.446818
Mercedes-Benz     42135.740369
Mini              43103.134828
Mitsubishi        23907.863824
Nissan             7766.459368
Porsche           74116.061111
Renault            8997.053907
Skoda             12937.508571
Tata               5679.734718
Toyota            18202.794610
Volkswagen         8341.832960
Volvo             28824.027619
Name: Price, dtype: float64
```

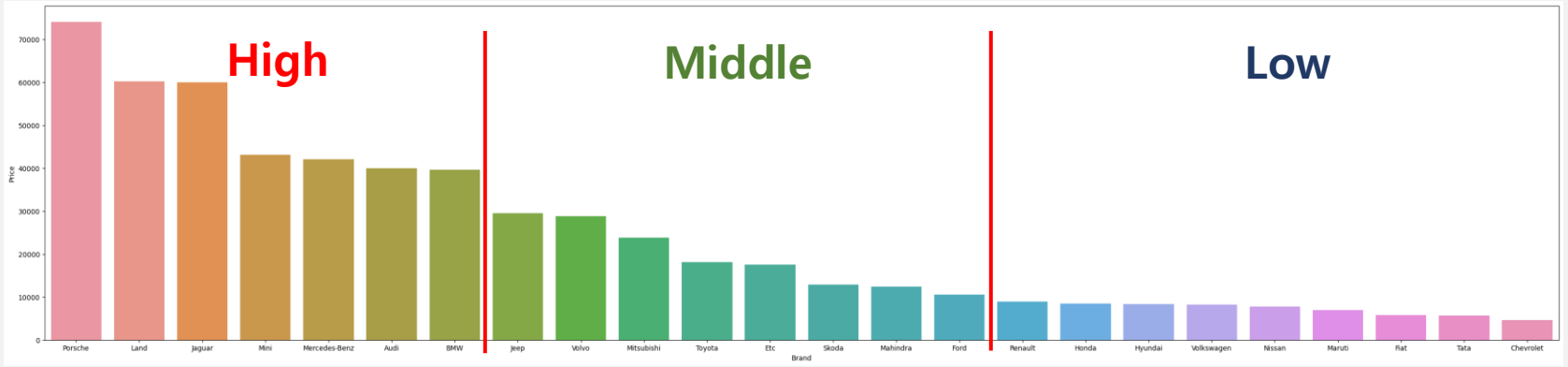
[브랜드 별 가격 평균]

```
upper_33 = np.quantile(list(df.groupby('Brand')['Price'].mean()),0.66)
lower_33 = np.quantile(list(df.groupby('Brand')['Price'].mean()),0.33)

def label_price_category(group, high_price = upper_33, low_price = lower_33):
    group_mean = group.mean()
    top_33 = group.quantile(0.66)
#     print(top_33)
    if group_mean > high_price:
        return 'High'
    elif group_mean > low_price:
        return 'Middle'
    else:
        return 'Low'
```

[Brand]

브랜드를 가격 평균을 기준으로 3등분하여 High / Middle / Low로 범주화



[재분류 한 결과]

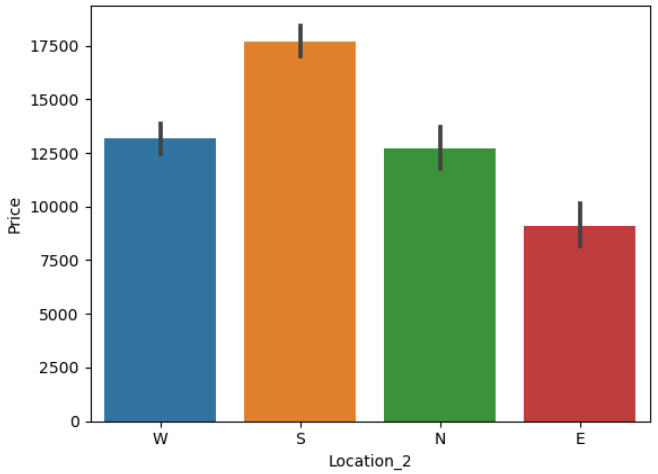
3. 범주형 변수의 그룹화

〈Location의 파생변수 생성〉

[Location]

- Location 열에 있는 11개 데이터를 동(E), 서(W), 남(S), 북(N) 4개의 그룹으로 나누어 범주화

- 범주화된 Location_2 변수는 중고차 시장의 지역적 특성을 분석하고, POS_CAR 사의 매출 극대화를 위한 최적 소재지를 결정하는 데에 중요한 변수임



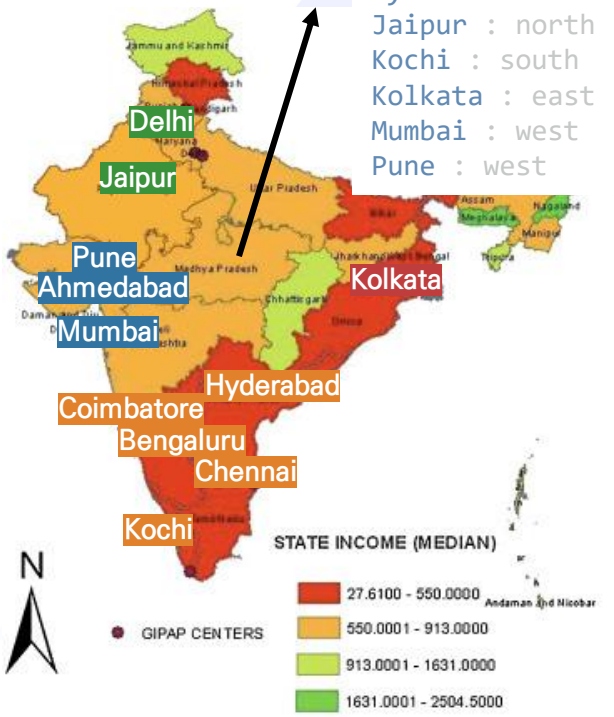
그래프 상으로 지역간 차이가 있음을 확인했고, 카이제곱검정을 통해 차이가 유의한지 검정했다.

H0 : 지역별로 가격 평균에 차이가 없다.
H1 : 지역별로 가격 평균에 차이가 있다.

```
Chi-square Statistic: 18050975.631334573
p-value: 0.0
```

➔ P-value 값이 유의수준 0.05보다 낮게 나타났으므로 대립가설을 채택한다.

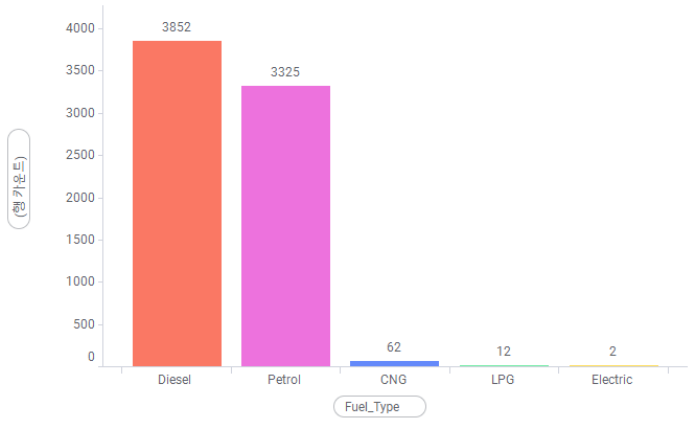
```
ahmedabad : west
bangalore : south
chennai : south
coimbatore : south
Delhi : north
Hyderabad : south
Jaipur : north
Kochi : south
Kolkata : east
Mumbai : west
Pune : west
```

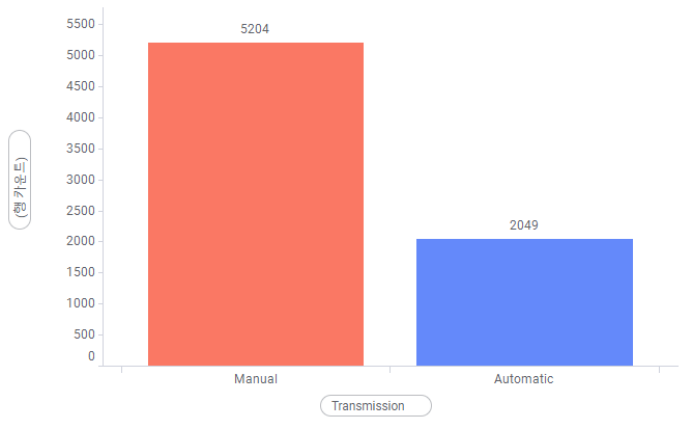


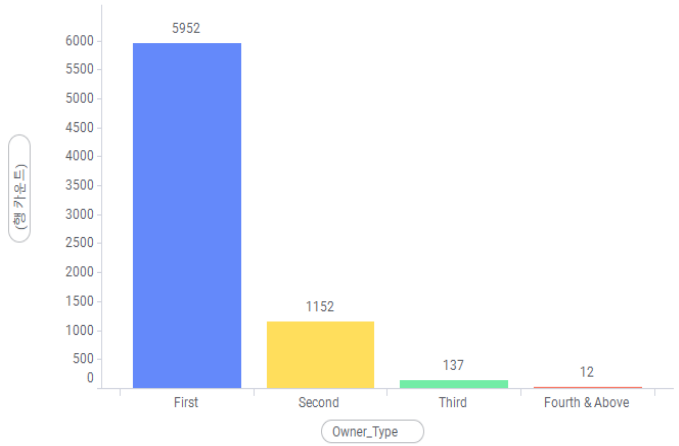
[Location를 활용한 파생변수 생성]

3. 범주형 변수의 그룹화

〈그 외 설명변수들의 파생변수 생성〉







[Fuel_Type]

Diesel, Petrol, Electric, CNG, LPG
↓
Diesel, Petrol, ETC

- Diesel, Petrol, Electric, CNG, LPG 중에서 데이터 통계량 분석 결과 총 데이터 중 Electric, CNG, LPG 의 데이터가 84개 이므로 ETC라는 변수로 통합

[Transmission]

Automatic, Manual

- 변수의 데이터가 2가지 유형으로 존재하기 때문에 그룹화를 할 필요가 없이, 0과 1로 표현
- One-Hot Encoding 적용 (0,1)

[Owner_Type]

First, Second, Third, Fourth & Above
↓
First, Second, Third & Above

- 데이터 통계량 분석 결과 총 7,253개 중 Third, Fourth & Above 의 데이터가 149개이므로 Third와 Four & Above를 Third & Above로 통합

4. 결측값 처리

〈결측값 제거 및 보완〉

원본 데이터 Car.csv에 존재하는 결측값들을 찾아서 제거하거나 보완한다.

Name	0
Location	0
Price	1053
Year	0
Kilometers_Driven	0
Fuel_Type	0
Transmission	0
Owner_Type	0
Mileage	2
Engine	46
Power	46
Seats	53
New_Price	6247

dtype: int64

[변수 별 결측값 통계]

1. Price

목표 변수이므로 임의로 평균 등으로 보완할 경우 학습에 악영향을 끼칠 수 있기에 결측값이 존재하는 행 전체를 제거한다.

2. Mileage, Engine, Power, Seats

차량의 모델 명 Name 변수를 참고하여서 인터넷에서 차량의 정보를 검색하여 직접 보완한다.

Key Specifications of Mahindra Jeep MM 540 DP

Fuel Type	**Diesel**
Engine Displacement	**2112 cc**
No. of Cylinders	4
Seating Capacity	6
Transmission Type	Manual

예시)
Mahindra Jeep MM 540 DP의 차량 정보는 왼쪽 사진과 같으므로 이 정보를 바탕으로 결측값을 전부 채운다,

➔ 차량 87대의 정보를 조사하여 결측값 보완 완료

3. New_Price

전체 데이터 중 90% 이상이 결측값이므로 New_Price 열 전체를 삭제하도록 한다.

1. 이상치 분석 및 제거

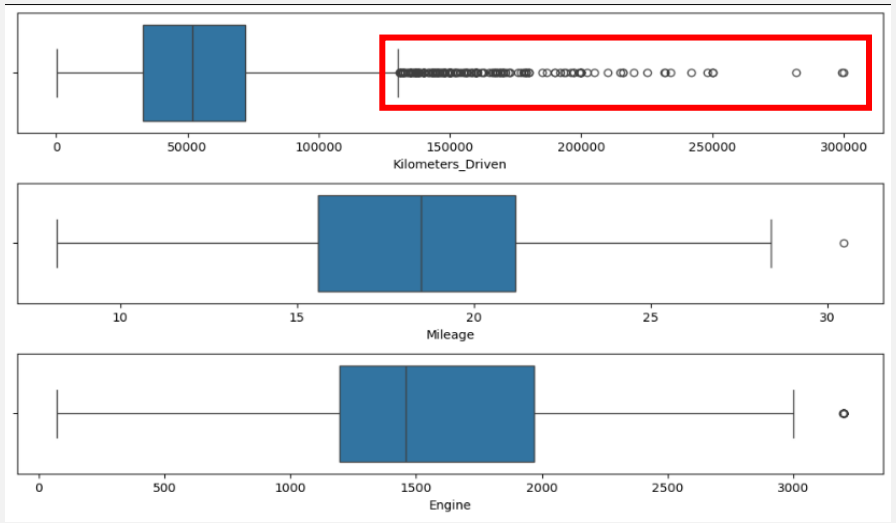
〈연속형 변수들의 이상치 제거〉

Kilometers_Driven Mileage, Engine, Power, Used_Year(현재 년도-Year)들의 이상치를 확인 후 제거한다.

[Box Plots 시각화 코드]

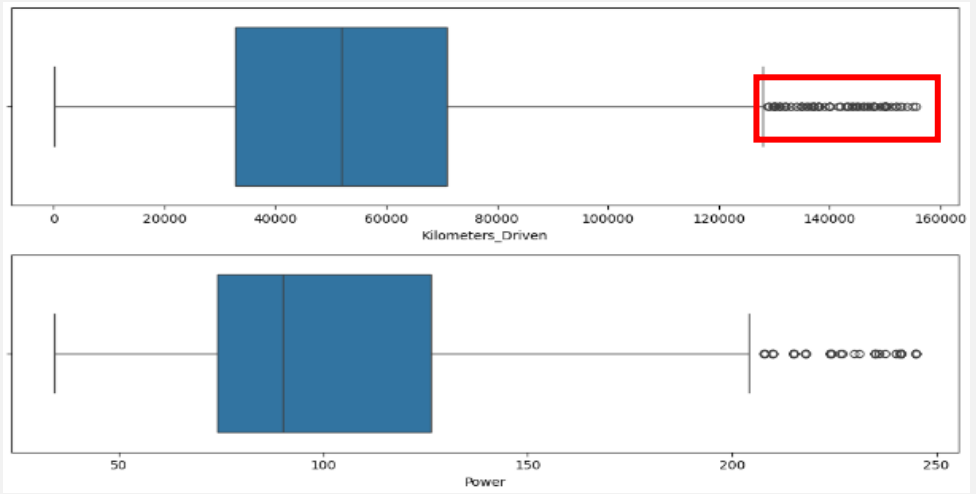
```
fig,axes = plt.subplots(nrows = 5, figsize = (10,10))
cols = ['Kilometers_Driven','Mileage','Engine','Power','Used_Year']


for i,col in enumerate(cols):
    sns.boxplot(x = col, data = df, ax = axes[i])
plt.tight_layout()
```



[Kilometers 변수에서 이상치 발견]

중앙값을 기준으로
± 3σ 밖에 있는 값들을
전부 제거한다.



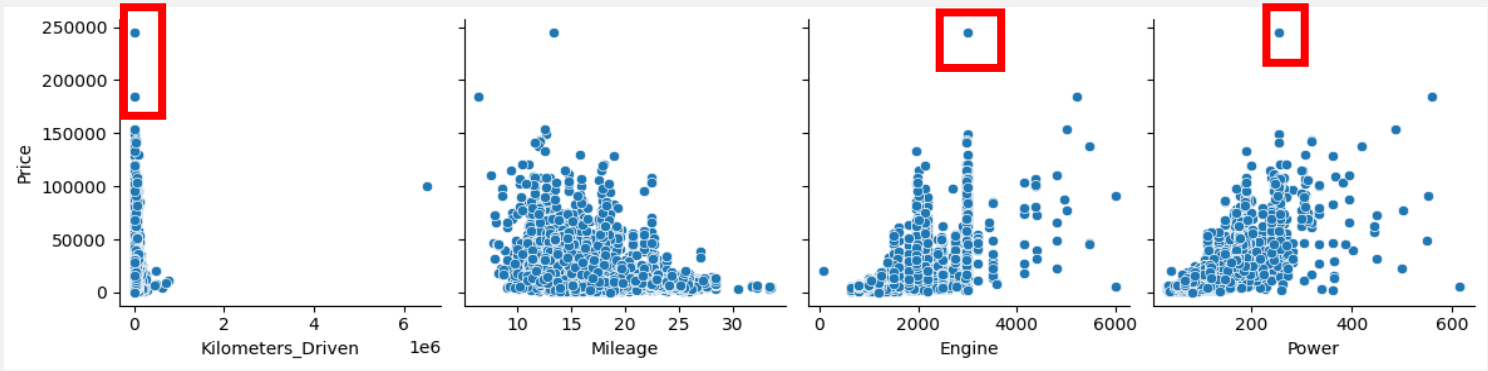
[이상치가 제거된 변수]

-〉 Kilometers_Driven 이외에 이상치가 발견된 다른 변수에서도 이상치를 제거한다.

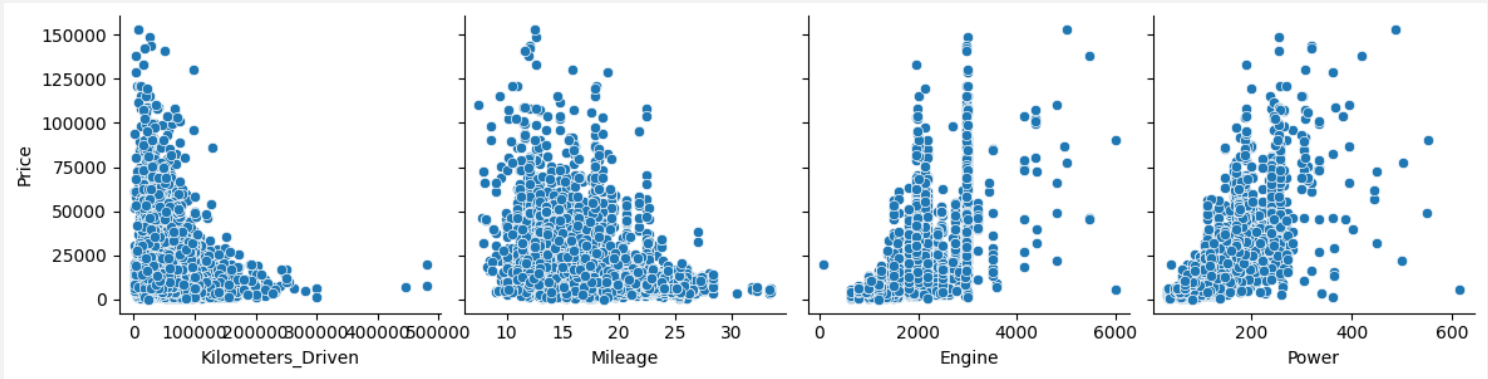
1. 이상치 분석 및 제거

〈연속형 변수들의 이상치 제거〉

[이상치 제거 전]



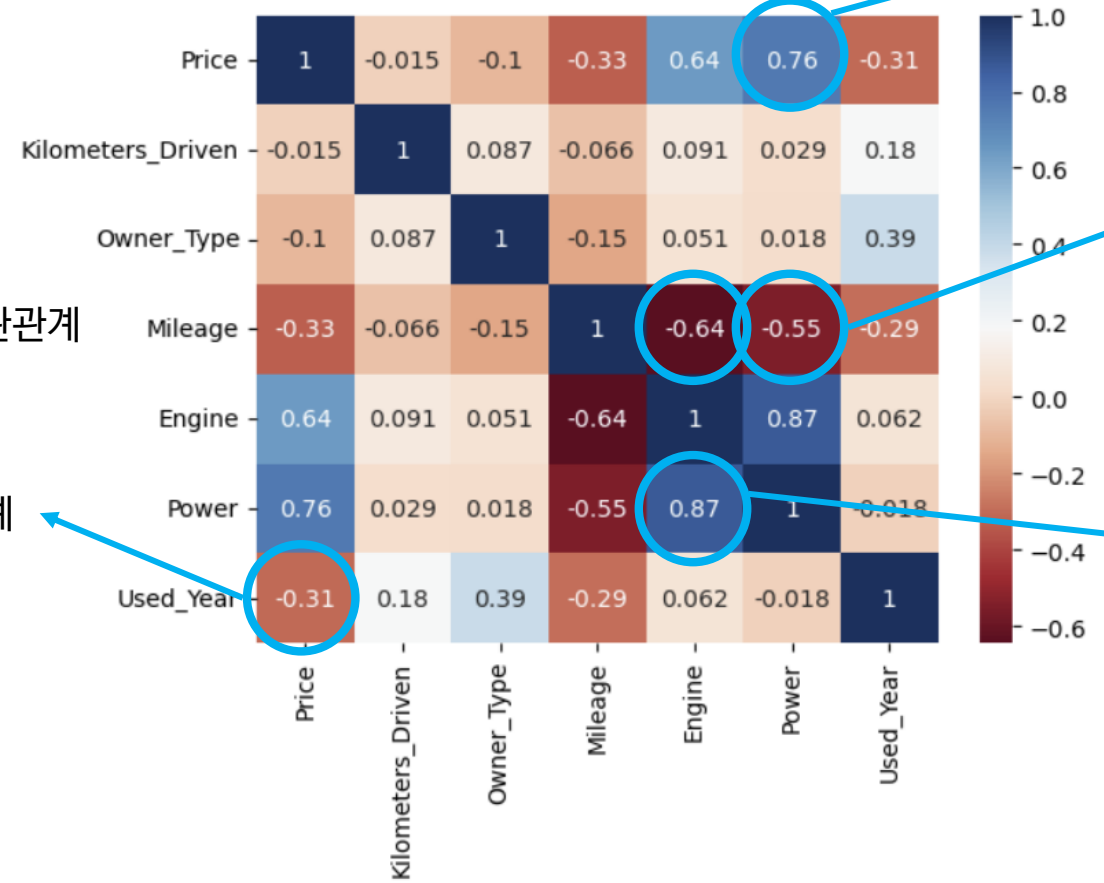
[이상치 제거 후]



2. 상관관계 시각화

〈Heatmap을 통한 각 인자 별 상관관계 시각화 〉

다중 공선성 분석으로 차량의 엔진 최대 출력이 가격에 중요한 영향을 끼칠 수 있음을 추론 가능



Price와 Power간의 양의 상관관계

Power와 Mileage간의 음의 상관관계

Engine과 Mileage간의 음의 상관관계

Price와 Year간의 음의 상관관계

Engine과 Power간의 양의 상관관계

1. 가설 검정

〈설명변수들의 통계적 검정〉

Price와 연관이 있는 변수들을 정규성 검정과 다중공선성 검정을 진행한다.

1. 정규성 검정

Price 변수 정규성 검정

```
Shapiro-Wilk test 결과 - Statistic: 0.6544863090150319, p-value: 2.761766493577184e-77
```

P-value가 0.05 이하이므로 정규성을 만족한다.

Engine 변수 정규성 검정

```
Shapiro-Wilk test 결과 - Statistic: 0.8765522994333391, p-value: 6.726181754403128e-57
```

P-value가 0.05 이하이므로 정규성을 만족한다.

Year 변수 정규성 검정

```
Shapiro-Wilk test 결과 - Statistic: 0.9588158288620903, p-value: 3.840031556599662e-41
```

P-value가 0.05 이하이므로 정규성을 만족한다.

Power 변수 정규성 검정

```
Shapiro-Wilk test 결과 - Statistic: 0.8399658004399069, p-value: 9.722571987849702e-62
```

P-value가 0.05 이하이므로 정규성을 만족한다.

2. 다중공선성 분석

	variable	VIF
2	Kilometers_Driven	1.50
1	Year	1.59
6	Seats	1.81
3	Mileage	2.04
5	Power	6.01
4	Engine	7.40
0	const	696477.24

VIF 수치가 모두 10이하이므로
다중공선성이 존재하지 않는다.

2. 가설 결론

〈통계적 검정 결과〉

상관관계 분석과 통계적 검정 결과를 분석해서 결론을 도출한다.

종합 결론

1. Price 변수는 Power 변수와 양의 상관관계를 갖는다.

-〉 엔진의 최대 출력이 높을수록 중고차 가격이 비싸다.

2. Price 변수는 Mileage 변수와 음의 상관관계를 갖는다.

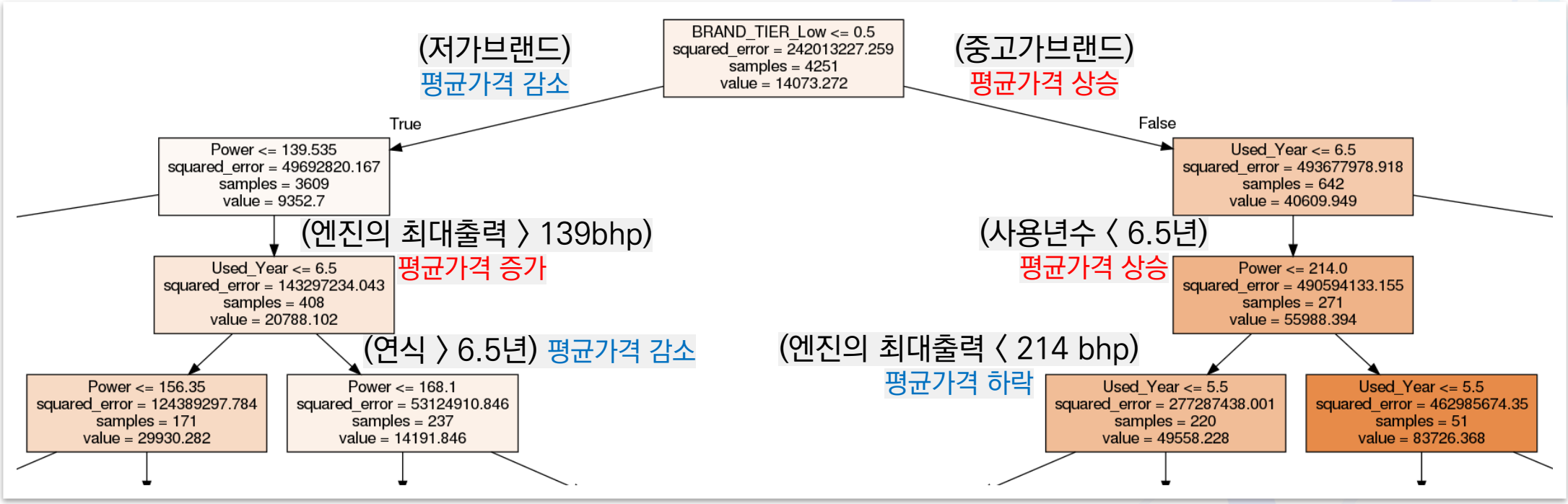
-〉 표준 주행거리가 높을수록 중고차 가격이 싸다.

3. Price 변수는 Year 변수와 음의 상관관계를 갖는다.

-〉 모델이 오래된 모델일수록 중고차 가격이 싸다.

1. Decision Tree

〈트리 모델 시각화 및 해석 : **가격** 에 영향을 미치는 핵심인자〉



저가브랜드의 차인 경우,

Penault, Honda, Hyundai, Volkswagen, Nissan, Manti, Flat, Tata, Chevrolet

성능이 연식에 비해 더 중요하다고 여겨진다.

중고가 브랜드의 차인 경우,

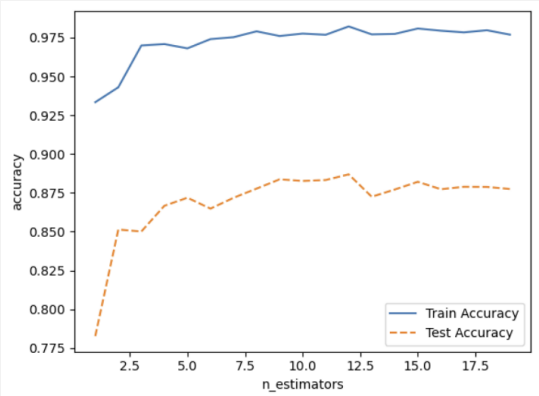
Jeep, Volvo, Mitsubishi, Toyota, Skoda, Mahindra, Ford, Porshe, Land, Jaguar, Mini, Mercedes-Benz, Audi, BMW

연식이 성능에 비해 더 중요하다고 여겨진다.

2. Random_Forest

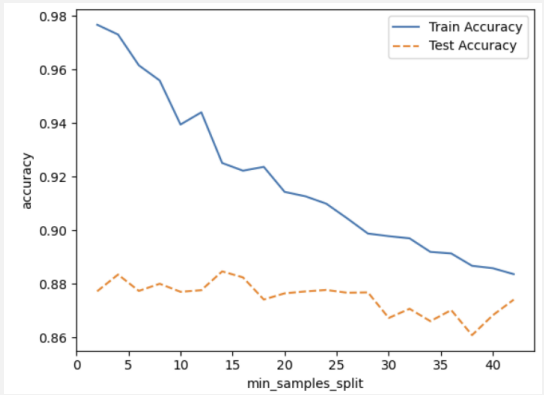
〈Hyper Parameter Tuning〉

모델의 성능을 극대화하기 위해 Hyper Parameter를 변경해가면서 최적의 값을 찾는다.



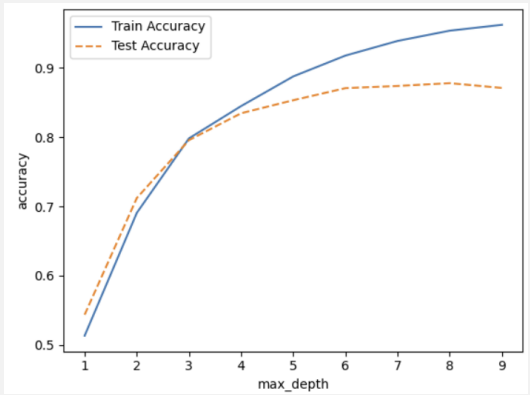
n_estimators = 100

n값이 25 이전에는 과소적합 되지만, 그 이후는
일정한 정확도를 유지하기에 100으로 설정한다.



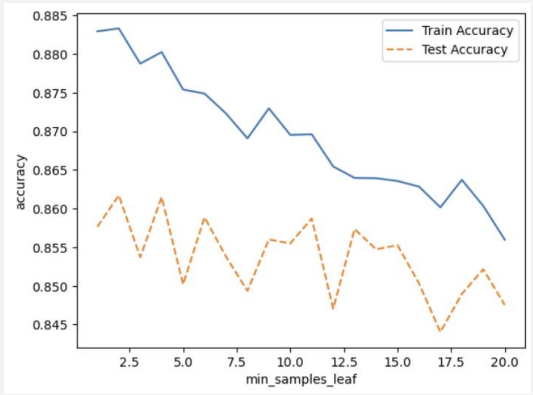
min_samples_split 미지정

Split 값을 늘려도 Test 셋에 대한 정확도는
크게 변화하지 않으므로 지정하지 않는다.



max_depth = 6

Depth가 1일 때에는 과소적합하지만, 6 이후부터는
과적합되므로 6으로 지정한다.



min_samples_leaf 미지정

Leaf의 수를 증가시키더라도 Test 셋의
정확도는 크게 변하지 않으므로 지정하지 않는다.

최종 모델 선정

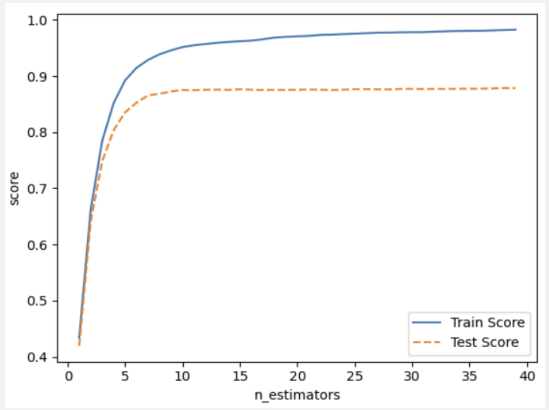
```
# 임의의 모델
rfr = RandomForestRegressor(n_estimators=100, max_depth = 6, random_state=42)
rfr.fit(df_train_x, df_train_y)
```

-〉 성능 검증 결과 : R2_Score : 0.8751, MAE : 2988으로 오차가 2988만큼 발생했다.

3. XGBoost

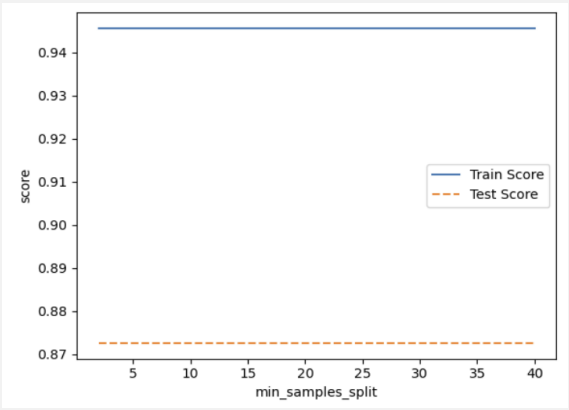
〈Hyper Parameter Tuning〉

모델의 성능을 극대화하기 위해 Hyper Parameter를 변경해가면서 최적의 값을 찾는다.



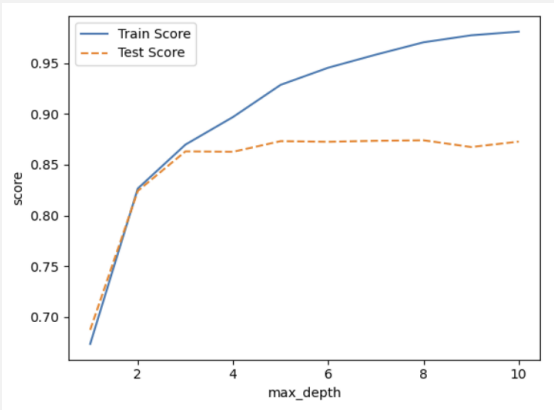
n_estimators = 9

n값이 5 이전에는 과소적합 되지만, 그 이후는 일정한 정확도를 유지하기에 9로 설정한다.



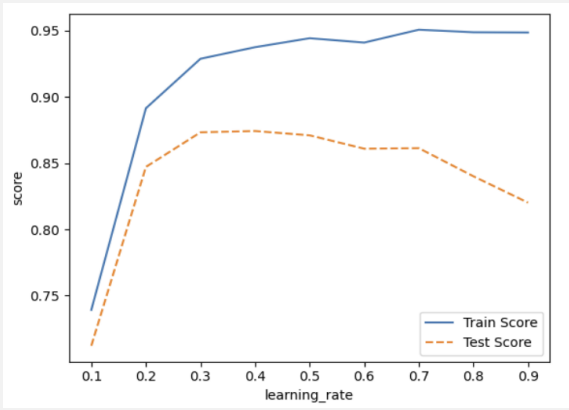
min_samples_split 미지정

Split 값을 늘려도 Test 셋에 대한 정확도는 크게 변화하지 않으므로 지정하지 않는다.



max_depth = 5

Depth가 2일 때에는 과소적합하지만, 6 이후부터는 과적합되므로 5으로 지정한다.



Learning_rate = 0.3

Learning_Rate 0.5 이상을 넘어가면 과적합되서 Learning Rate는 0.3으로 한다.

최종 모델 선정

```
In [100]: xgb_final = XGBRegressor(n_estimators = 9, max_depth = 5,learning_rate = 0.3)
          xgb_final.fit(df_train_x, df_train_y)
```

-〉 성능 검증 결과 : R2_Score : 0.8595, MAE : 2807으로 오차가 2807만큼 발생했다.

4. Linear Regression

〈Training Set 결과 확인〉

OLS Regression Results

Dep. Variable:	Price	R-squared:	0.679
Model:	OLS	Adj. R-squared:	0.679
Method:	Least Squares	F-statistic:	1498.
Date:	Sun, 10 Mar 2024	Prob (F-statistic):	0.00
Time:	03:14:18	Log-Likelihood:	-44595.
No. Observations:	4251	AIC:	8.920e+04
Df Residuals:	4244	BIC:	8.925e+04
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.405e+04	133.626	105.112	0.000	1.38e+04	1.43e+04
Kilometers_Driven	-1002.7222	168.319	-5.957	0.000	-1332.716	-672.728
Owner_Type	29.4898	142.849	0.206	0.836	-250.570	309.549
Mileage	721.2078	189.742	3.801	0.000	349.215	1093.201
Engine	1045.0692	286.556	3.647	0.000	483.270	1606.868
Power	1.112e+04	257.129	43.230	0.000	1.06e+04	1.16e+04
Used_Year	-3987.1902	175.284	-22.747	0.000	-4330.839	-3643.541

Omnibus:	2242.366	Durbin-Watson:	1.959
Prob(Omnibus):	0.000	Jarque-Bera (JB):	27950.942
Skew:	2.233	Prob(JB):	0.00
Kurtosis:	14.741	Cond. No.	4.36

위 결과를 통해 Owner_Type의 p-value값을 통해 유의미하지 않은 변수라는 걸 확인

➔ 이후 Owner_Type을 제외하여 선형회귀 진행

OLS Regression Results

Dep. Variable:	Price	R-squared:	0.679
Model:	OLS	Adj. R-squared:	0.679
Method:	Least Squares	F-statistic:	1798.
Date:	Sun, 10 Mar 2024	Prob (F-statistic):	0.00
Time:	03:14:18	Log-Likelihood:	-44595.
No. Observations:	4251	AIC:	8.920e+04
Df Residuals:	4245	BIC:	8.924e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.405e+04	133.607	105.129	0.000	1.38e+04	1.43e+04
Kilometers_Driven	-1000.4803	167.950	-5.957	0.000	-1329.750	-671.211
Mileage	719.3817	189.514	3.796	0.000	347.835	1090.928
Engine	1043.7591	286.453	3.644	0.000	482.162	1605.357
Power	1.112e+04	257.088	43.239	0.000	1.06e+04	1.16e+04
Used_Year	-3977.0780	168.282	-23.633	0.000	-4306.998	-3647.158

Omnibus:	2242.776	Durbin-Watson:	1.959
Prob(Omnibus):	0.000	Jarque-Bera (JB):	27955.416
Skew:	2.233	Prob(JB):	0.00
Kurtosis:	14.742	Cond. No.	4.32

Owner_Type를 제외하여 선형회귀 진행

➔ p-value를 확인해봤을 때 모든 변수가 유의미하다는 걸 확인

4. Linear Regression

〈Test Set 결과 확인〉

OLS Regression Results

Dep. Variable:	Price	R-squared:	0.622
Model:	OLS	Adj. R-squared:	0.621
Method:	Least Squares	F-statistic:	597.9
Date:	Sun, 10 Mar 2024	Prob (F-statistic):	0.00
Time:	03:14:18	Log-Likelihood:	-19434.
No. Observations:	1823	AIC:	3.888e+04
Df Residuals:	1817	BIC:	3.891e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.413e+04	242.088	58.359	0.000	1.37e+04	1.46e+04
Kilometers_Driven	-1207.5292	297.936	-4.053	0.000	-1791.863	-623.196
Mileage	87.4446	336.643	0.260	0.795	-572.803	747.692
Engine	1579.0223	520.825	3.032	0.002	557.543	2600.502
Power	1.061e+04	463.526	22.886	0.000	9699.101	1.15e+04
Used_Year	-4828.4120	307.065	-15.724	0.000	-5430.649	-4226.175

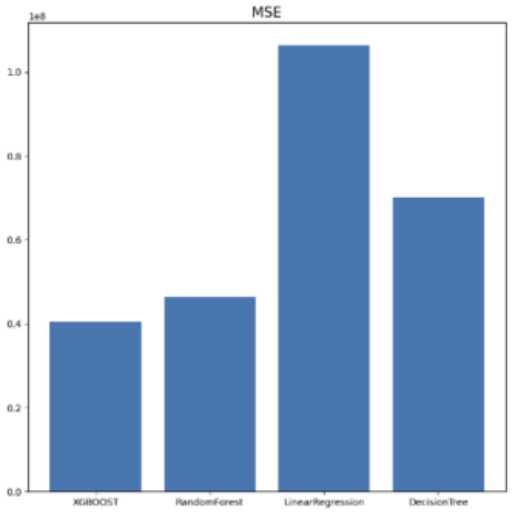
Omnibus:	2020.882	Durbin-Watson:	1.987
Prob(Omnibus):	0.000	Jarque-Bera (JB):	387785.790
Skew:	5.185	Prob(JB):	0.00
Kurtosis:	73.694	Cond. No.	4.42

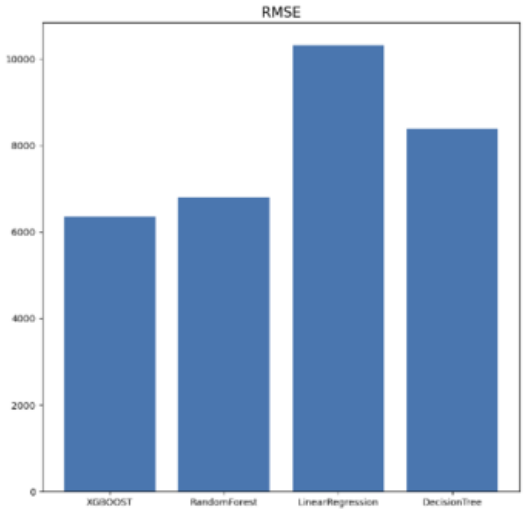
Test 데이터를 검증해본 결과, R2_Score 0.621, MAE는 6187이 나온 것을 확인 가능하다.

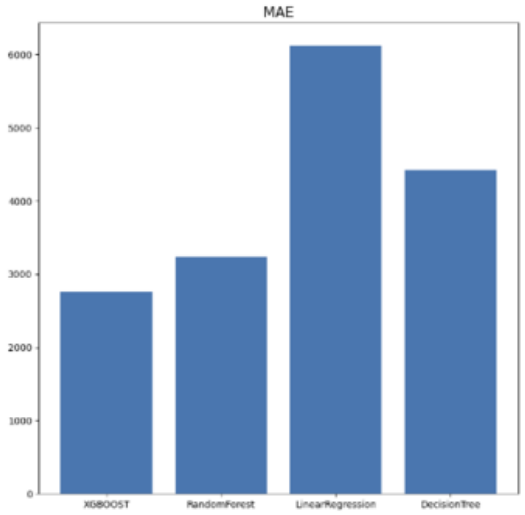
1. 모델 별 오차 분석

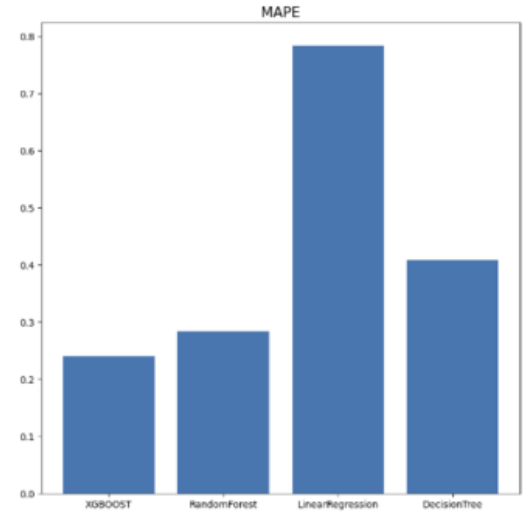
모델	Decision Tree	Random Forest	XGB	Linear Regression
정확도 (기본 모델 기준)	Training : 81.3% Test : 75.0%	Training : 88.7% Test : 82.4%	Training : 92.4% Test : 84.7%	Training : 67.9% Test : 62.2%

➔ 학습 데이터와 테스트 데이터 모두에서 비슷한 정도의 설명력을 보인다. 따라서 위 표의 전반적인 결과를 토대로 XGB를 최종 모델로 선택







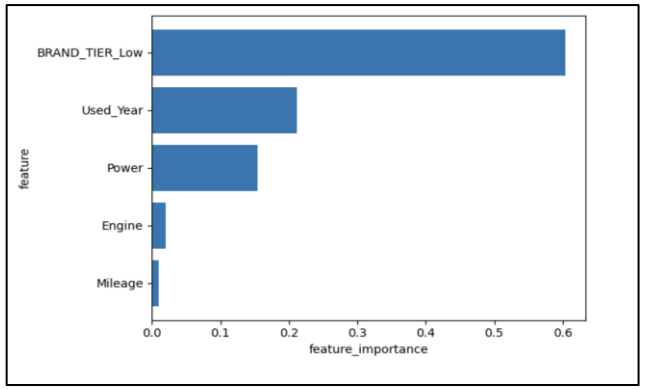


MSE, RMSE, MAE, MAPE에서도
XGB이 가장 성능이 우수하다고 나타남

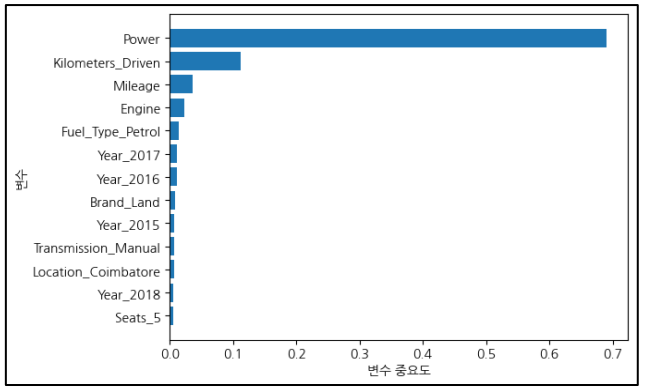
2. 설명 변수의 중요도 결과

〈가격에 영향을 미치는 핵심 인자〉

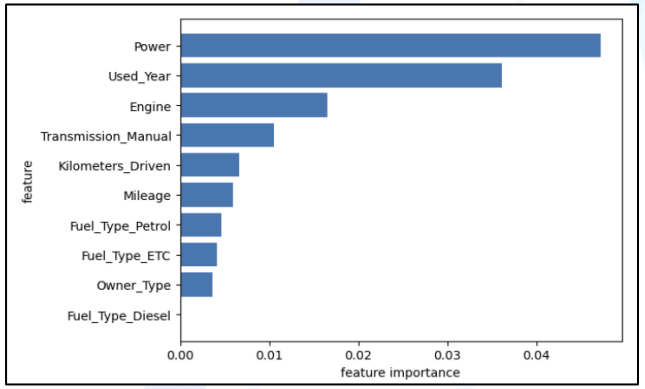
Decision Tree Random Forest, Gradient Boosting 의 중요도 그래프를 비교해보았다.



Decision Tree



Random Forest



XGB

결론

- 모델 별로 순서의 차이가 있지만, Power, Used_Year, Engine이 가격에 영향을 준다는 것을 확인할 수 있다.
- 상관관계를 고려해보았을 때 Power가 클수록, Used_Year이 적을수록, Engine이 클수록 가격이 높다.

1. 프로젝트를 통해 얻은 인사이트

프로젝트를 진행하며 데이터를 적절히 다루는 것의 중요성을 이해하게 되었다. 범주형 변수의 수준이 방대한 경우 과적합을 피하기 위해 복잡한 모델을 단순화할 필요가 있다. 범주형 데이터를 재분류하는 과정에서, 팀 내 다양한 의견이 수렴되었다. 예상보다 낮은 회귀 분석의 설명력으로부터 데이터 처리 방법을 면밀히 검토했으며, 이상치 처리에 있어 백분위수 기반 절삭보다 ESD 방법을 채택함으로써 원 데이터를 보호하고 이상치를 최소한으로 제거하면서도 모델의 성능이 크게 저하되지 않은 결과를 얻을 수 있었다.

모델의 하이퍼 파라미터 튜닝 과정에서는 데이터의 특성에 따라 모델의 성능이 높지 않다보니, 성능이 좋은 파라미터를 선택해야할 지, 아니면 과적합 우려를 해소하기 위해 Train 과 Test 성능 차이가 적은 파라미터를 선택해야할 지에 대한 고민이 있었다. 마찬가지로 토의를 거친 후에 최종적으로는 일반화 능력이 모델의 성능에 있어 더 중요하다는 판단을 내리게 되었다.

마지막으로 협업 과정에서 코딩 컨벤션이 정립되지 않아 애로사항이 있었다. 트리 기반 알고리즘에서는 스케일링이 불필요하지만, 선형 회귀모델에서는 필수적이다. 코드 공유 과정에서 데이터의 표준화 여부가 희미해져서, 회귀 분석의 MSE가 극단적으로 증가된 결과를 확인할 수 있었다. 이를 통해 네이밍 규칙과 주석을 통해 더 효율적인 협업 환경을 만들게 되는 계기가 되었다. 이러한 노력은 앞으로의 프로젝트 진행 과정을 더 순조롭게 만들 것 같다.