중고차 가격 예측하기

김준구

Contents

1. 데이터 소개
2. 분석 계획
3. 분석 결과
4. 결론

Data

- Kaggle에서 주최한 "인도 중고차 예측하기" 데이터셋의 train data를 바탕으로 중고차 가격 예측하기
- https://www.kaggle.com/datasets/avikasliwal/used-cars-price-prediction
- **6019** rows and **14** columns

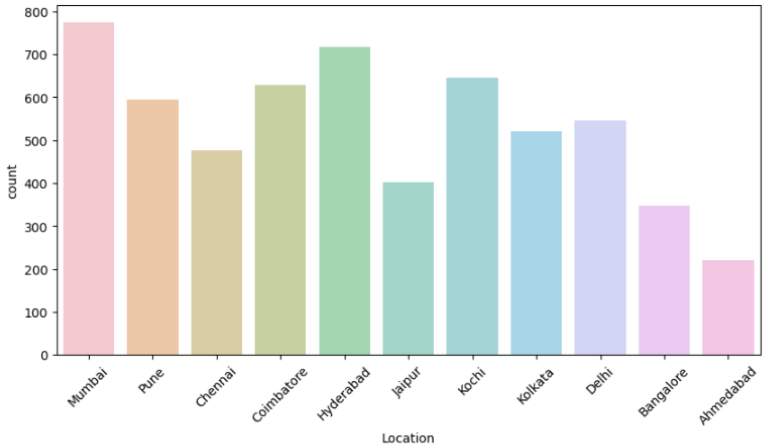
Name	Location	Year	Kilometers Driven	Fuel Type	Transmission	Owner Type	Mileage	Engine	Power	Seats	New Price	Price
Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5		1.75
Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5		12.5

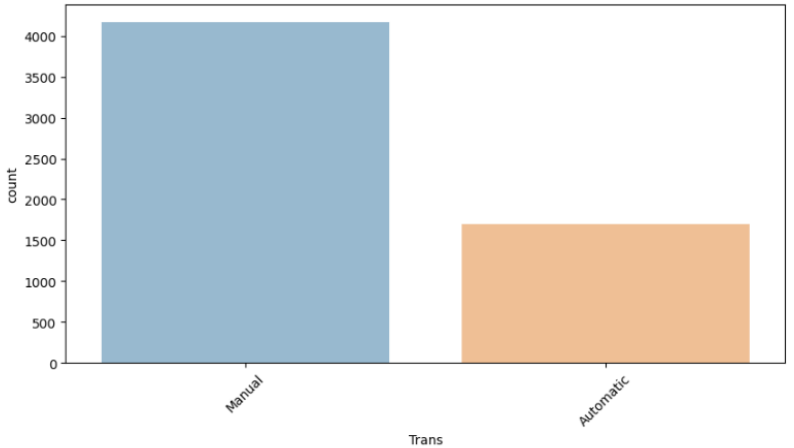
02 분석 계획

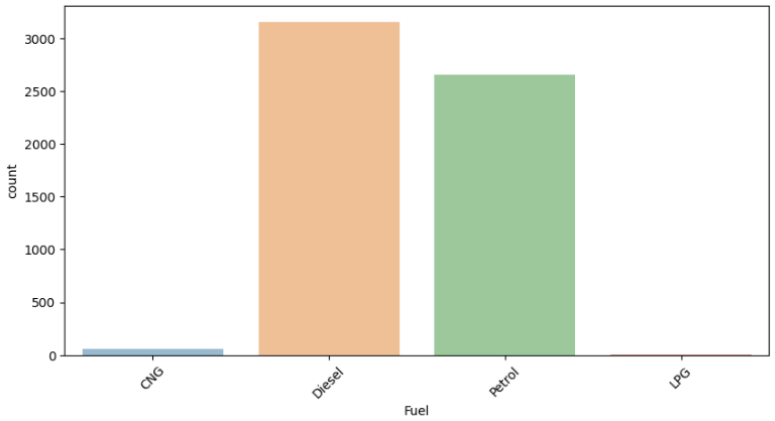
목적	분석 계획	
	분석방법	분석 내용
각 변수의 분포 및 단일 변수 특성 파악	막대그래프	· 지역, 브랜드, 연료 등
	Dist Plot	· 연도, 파워, 가격 변수 등의 분포 확인
중고차 가격 예측 및 지역 예측 모델 개발	선형회귀분석	· 중고차 데이터로 가격 예측 및 지역 예측 모델링
	랜덤포레스트	
	LightGBM	· 평가 지표를 종합적으로 고려하여 가장 높은 모형 선정
	Gradient Boost	
	XGBoost	

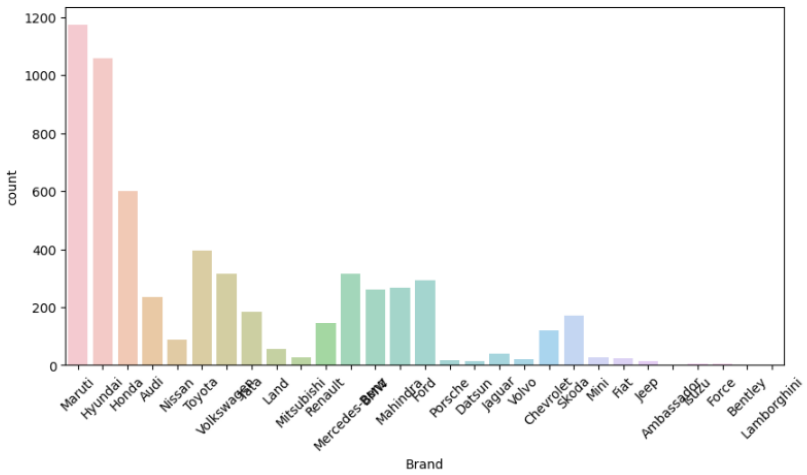
03 분석 결과

- **지역은 비교적 고르게 분포되어 있으나, 연료는 압도적으로 휘발유와 디젤에 분포되어 있음**
  변속기는 수동이 자동보다 두 배 이상 많음.







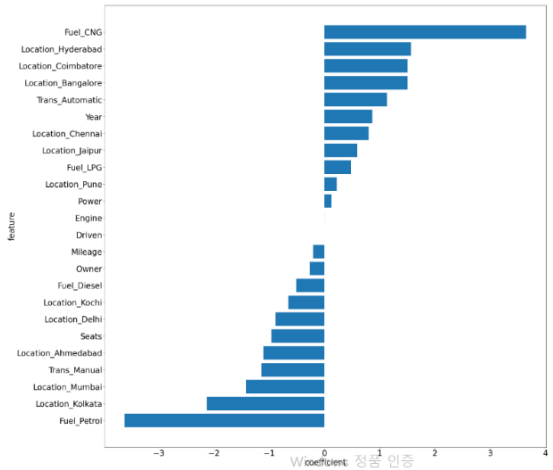


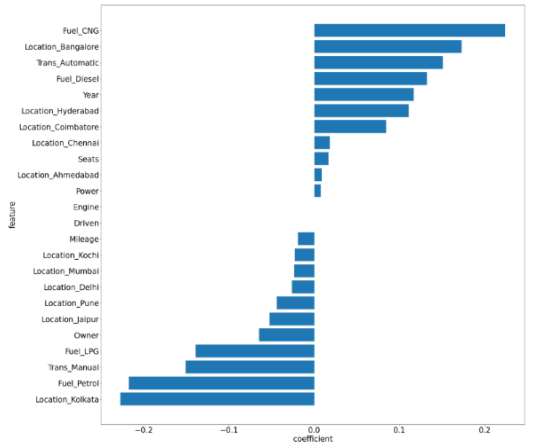
03 분석 결과

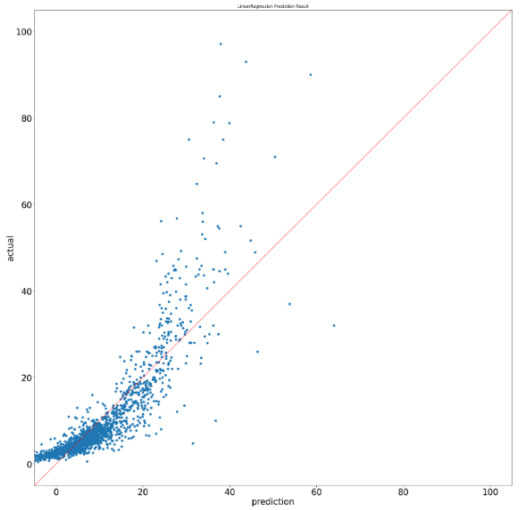
Linear Regression

- 일반 선형회귀모델이 지수함수적으로 증가하고 있으므로 로그를 취해서 해당 패턴을 더 잘 설명 할 수 있음.

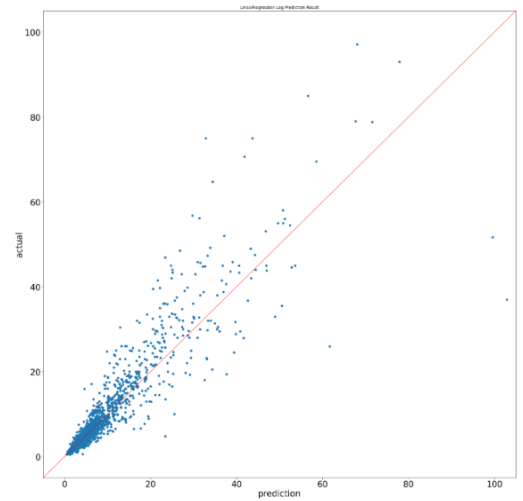
그러나 결정계수와 RMSE를 사용한 결과, 각각 0.71, 5.88 와 0.68, 6.19를 나타내어 두 모델 모두 결과가 좋다고 말할 수 없음.







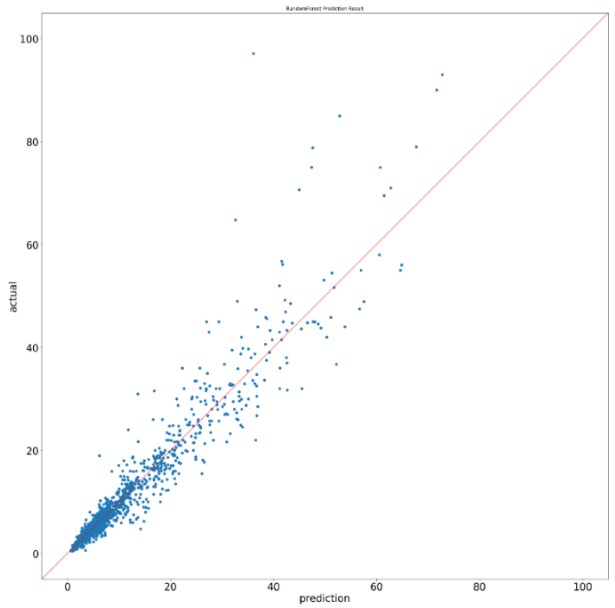
Train data Accuracy : 0.709835367825089
Test data r-square : 0.7171186708836319
Root mean squared error : 5.881483114449467

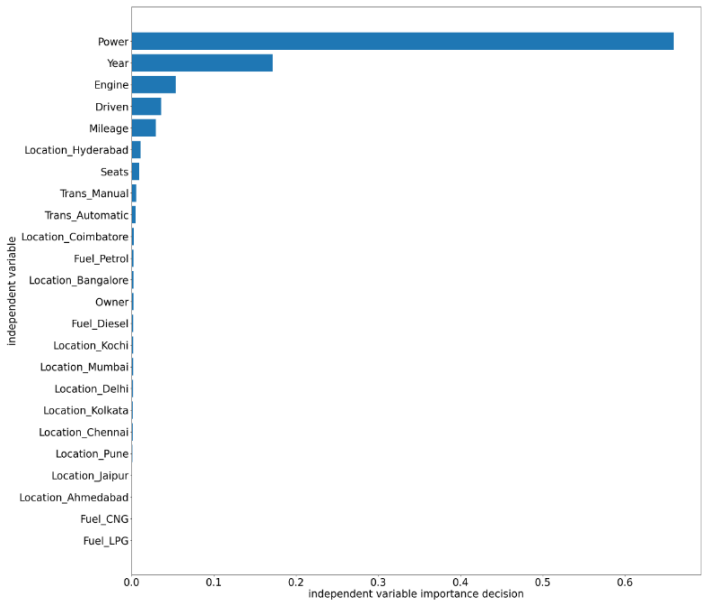


Train data Accuracy : 0.8814551272415928
Test data r-square : 0.6866084389565615
Root mean squared error : 6.190537538207451

Random Forest

- 모델의 설명력을 나타내는 결정계수는 0.90, 모델의 예측 정확도를 나타내는 RMSE는 3.38을 나타내어 Random Forest는 괜찮은 모델이라고 말할 수 있음.



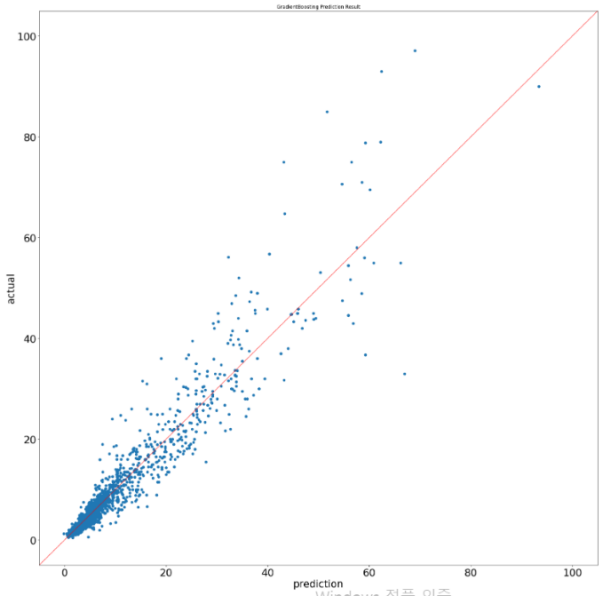


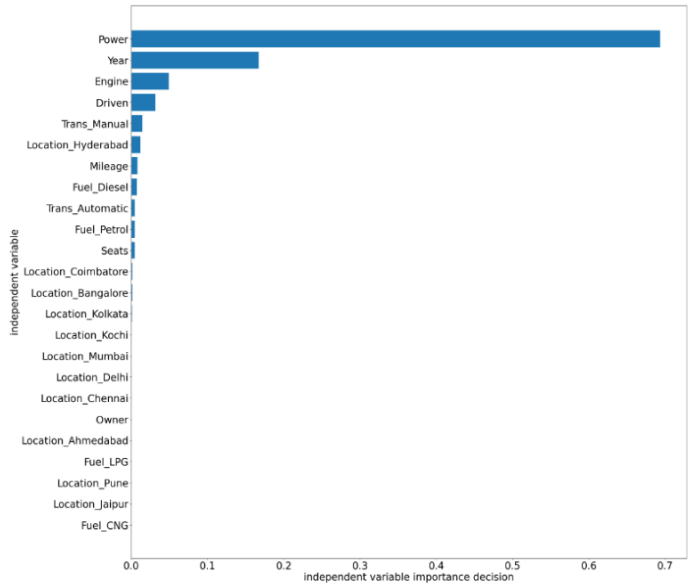

```
Train data Accuracy :  0.9810386031581207
Test data r-square :  0.9061899855239836
Root mean squared error :  3.3869541229428015
```

Random Forest에서 Power의 중요도가 압도적으로 높고, Year가 뒤따름.

Gradient Boost

- 모델의 설명력을 나타내는 결정계수는 0.90, 모델의 예측 정확도를 나타내는 RMSE는 3.48을 나타내어 Gradient Boost 또한 괜찮은 모델이라고 말할 수 있음.



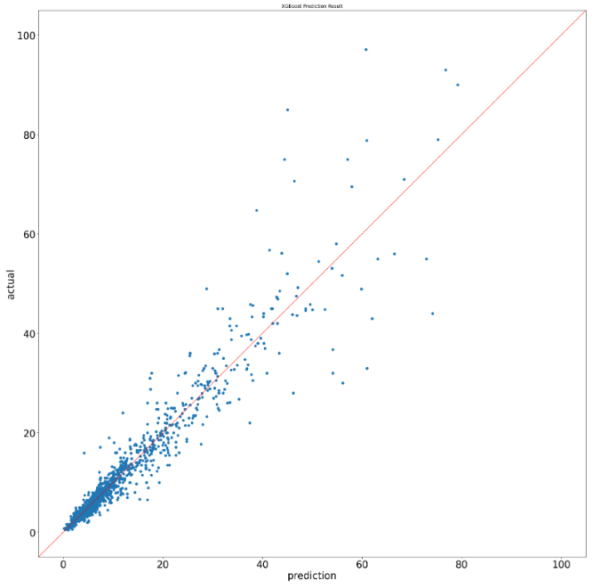


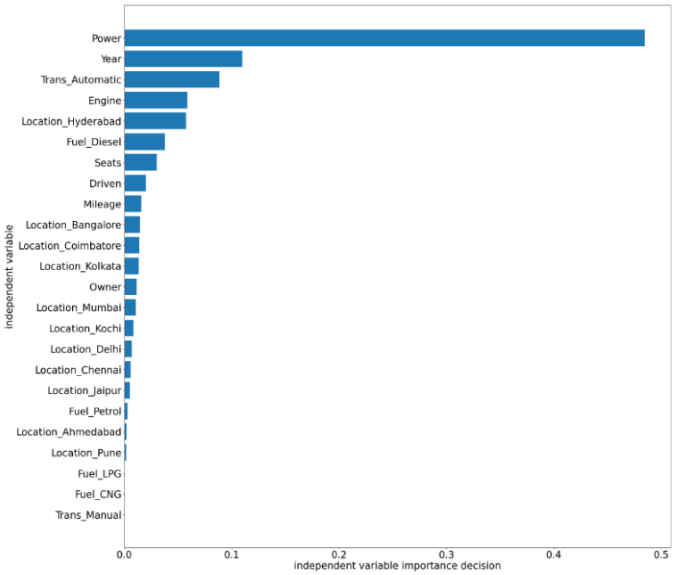

```
Train data Accuracy :  0.9331881249908458
Test data r-square :  0.9005560457736879
Root mean squared error :  3.487176305303609
```

Gradient Boos에서 또한 Power의 중요도가 압도적으로 높고, Year가 뒤따름.

XG Boost

- 모델의 설명력을 나타내는 결정계수는 0.91, 모델의 예측 정확도를 나타내는 RMSE는 3.27을 나타내어 현재까지 XGBoost가 제일 우수한 모델이라고 말할 수 있음.



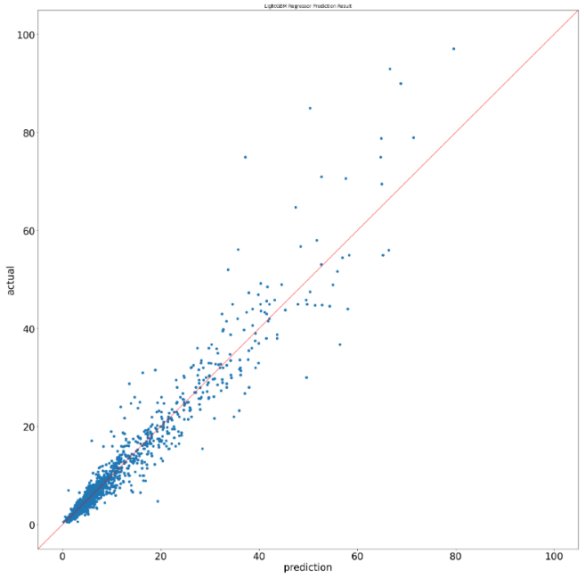


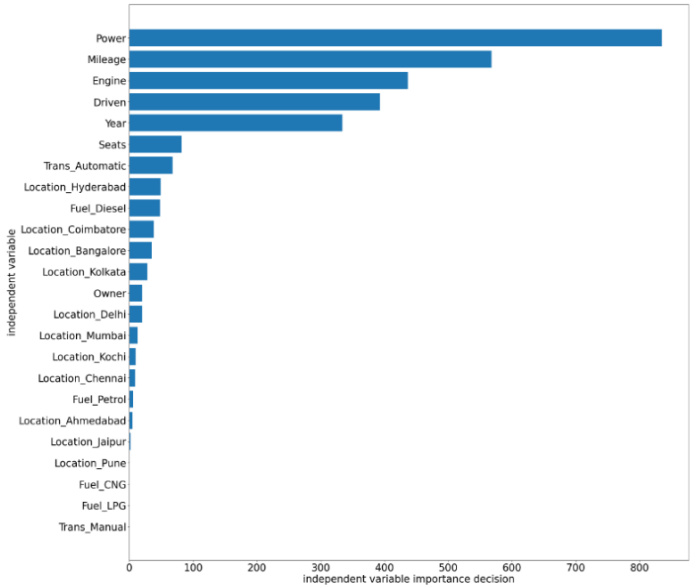

```
Train data Accuracy :  0.9963899855819375
Test data r-square :  0.9121459109765958
Root mean squared error :  3.2776735952646816
```

XGBoost는 Power의 중요도가 압도적으로 높고,
다른 컬럼 값들은 다른 모델들에 비해 비교적 많이 XGBoost에 미치는 영향이 있다.

LightGBM

- 모델의 설명력을 나타내는 결정계수는 0.92, 모델의 예측 정확도를 나타내는 RMSE는 2.98을 나타내어 모든 모델 중에 LightGBM이 제일 우수함.






```
Train data Accuracy :  0,9515517128203499
Test data r-square :  0,9270662432820849
Root mean squared error :  2,98640706683201
```

- LightGBM는 Power의 중요도가 높지만 다른 컬럼 값들은 다른 모델들에 비해 월등히 많이 LightGBM모델에 참여함.

03 분석 결과

- 현대차의 현재 가격은 2.35

- LinearRegression을 적용할 시 1.48
- RandomForest를 적용할 시 2.79
- LightGBM을 적용할 시 3.12

- 앞에서 봤듯이 모델 중 제일 성능이 좋은 LightGBM모델을 사용하면 제일 좋은 결과를 도출할 수 있음

```
# 예측가격

# 일반

# LinearRegression : 1.485
# RandomForest : 2.696
# LightGBM : 3.126

X = get_Brand_df('Hyundai').drop(['Price', 'Brand'], axis = 1)
y = get_Brand_df('Hyundai')['Price']

print(get_car_price_lr(Hcar))
print(get_car_price_rf(Hcar))
print(get_car_price_lgbm(Hcar))
```

```
[1.48521646]
[2.7939]
[3.1261156]
```

```
# 현재 가격 : 2.35

Hcar = np.array([[2012, 75000, 27.43, 814.0, 55.2,
                  5.0, 1, 0, 0,
                  0, 0, 0, 1, 0,
                  0, 0, 0, 0, 0,
                  0, 1, 0 ,0, 1]])
```

- Year : 2012
- Driven : 75000
- Mileage : 27.43
- Engine : 814.0
- Power : 55.2
- Seats : 5.0
- Owner: 1
- Location_Ahmedabad : 0
- Location_Bangalore : 0
- Location_Chennai : 0
- Location_Coimbatore : 0
- Location_Delhi : 0
- Location_Hyderabad : 1
- Location_Jaipur : 0
- Location_Kochi : 0
- Location_Kolkata : 0
- Location_Mumbai : 0
- Location_Pune : 0
- Fuel_CNG : 0
- Fuel_Diesel : 0
- Fuel_LPG : 1
- Fuel_Petrol : 0
- Trans_Automatic : 0
- Trans_Manual : 1

03 분석 결과

- BMW의 현재 가격은 8.47

- LinearRegression을 적용할 시 8.32
- RandomForest를 적용할 시 8.64
- LightGBM을 적용할 시 10.84

- 앞에서 봤듯이 모델 중 제일 성능이 좋은 LightGBM모델을 사용하면 제일 좋은 결과를 도출할 수 있음

```
Bcar = np.array([[2009, 128000, 10.8, 2497.0, 215.0,
                  5.0, 1, 0, 0,
                  0, 0, 0, 0, 0,
                  0, 0, 1, 0, 0,
                  0, 0, 1 ,1, 0]])
```

```
# 예측가격 8.47

# 일반

# LinearRegression : 8.321
# RandomForest : 8.027
# LightGBM : 10.848

X = get_Brand_df('BMW').drop(['Price', 'Brand'], axis = 1)
y = get_Brand_df('BMW')['Price']

print(get_car_price_lr(Bcar))
print(get_car_price_rf(Bcar))
print(get_car_price_lgbm(Bcar))
```

```
[8.32183296]
[8.6438]
[10.84866046]
```

03 분석 결과

- **LightGBM을 사용해서 인도시장에서 중고차 거래 시 가격을 잘 받을 수 있는 인도 지역을 예측해본 결과 Porsche 거래 시 Coimbatore 지역으로 결과가 도출됨**

```
def sell_to_location_lgbm(brand):

    X = get_Brand_df(brand).drop(['Price', 'Brand'], axis=1)
    y = get_Brand_df(brand)['Price']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10)

    lgbm = LGBMRegressor()
    lgbm.fit(X_train, y_train)


    value = lgbm.feature_importances_
    value_s = pd.Series(value, index=X_train.columns)

    print('sell_to_location_lgbm :' , ols_reg.params[7:18].sort_values(ascending=False).index[0])
```

```
sell_to_location_lgbm('Porsche')
```

```
sell_to_location_lgbm : Location_Coimbatore
```

04 결론

- 제일 결과가 좋은 모델: LightGBM

- 이상치 데이터 제거 안하고 선형회귀 분석했을 때: Train data Accuracy < 0.5
- 이상치 제거 후 : Train data Accuracy = 0.70이상

→ 하나의 이상 값으로도 결과에 큰 영향을 미침 (데이터 전처리의 중요성)

- 같은 데이터로 다른 모델을 사용했을 때 결과 다름

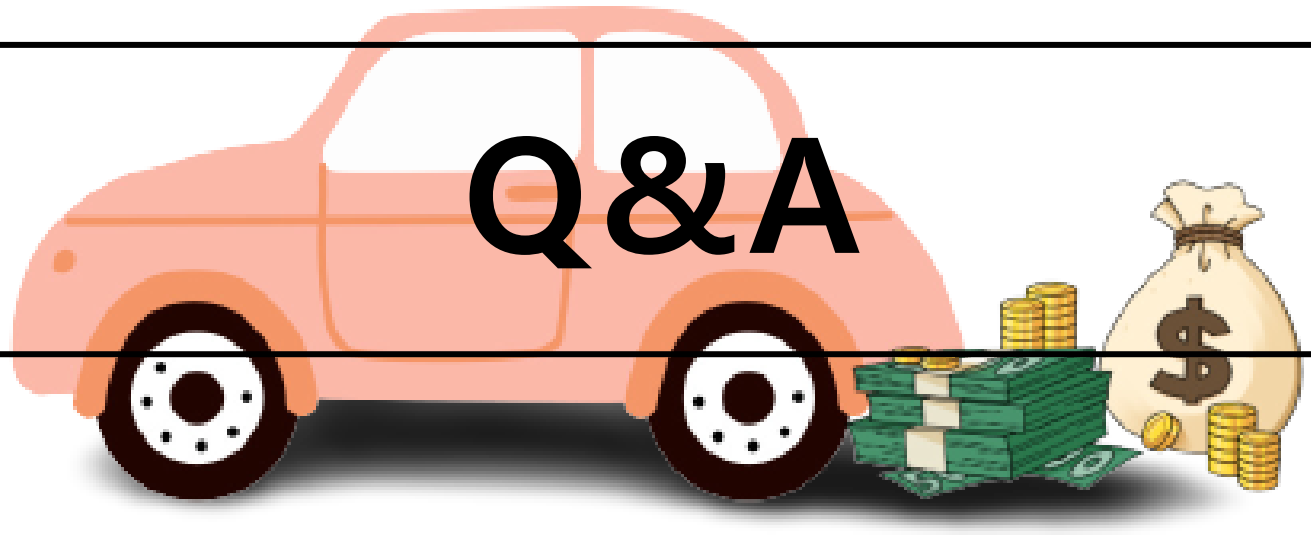
→ 모델 선택의 중요성

증빙 자료

📌 Github 주소
https://github.com/KimJunGu9/portfolio1

📌 Python 코드
https://colab.research.google.com/drive/1ELa_e1JDyCCbJ03bdBFlaLftlxNBeGRE#scrollTo=0KXaWBpXX4tN

Q&A



감사합니다