

# Saudi Arabia Used Cars

Naila Firdusi  
Data Science  
2025



# BUSINESS PROBLEM UNDERSTANDING

# CONTEXT

- 1 Mobil merupakan sarana transportasi manusia untuk kehidupan sehari, namun hingga saat ini harga mobil relatif tinggi sehingga mobil bekas menjadi sebuah 'jalan pintas'.
- 2 Salah satu negara yang cukup tinggi pasar mobil bekas adalah Saudi Arabia.
- 3 Perkembangan digital mempengaruhi terbentuknya transaksi mobil bekas berbasis online.

# PROBLEM STATEMENT

1

Tantangan penjual mobil bekas adalah sulitnya menentukan harga jual

2

Penyebabnya karena kurang kepercayaan konsumen terhadap mobil bekas serta kekurangan mobil bekas sendiri dan juga harga yang kompetitif

# GOALS

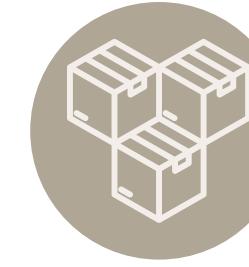
Perlu adanya sebuah “tools” untuk membantu penjual dan pembeli dalam penentuan estimasi harga mobil yang lebih akurat dan mudah berdasarkan kondisi mobil tersebut.

# EVALUATION METRIC



## RMSE (Root Mean Squared Error)

Metrik yang sangat penting karena mempunyai sensitivitas terhadap outliers



## MAPE (Mean Percentage Error)

metrik ini sangat cocok untuk permasalahan bisnis terutama untuk prediksi penjualan atau arus kas serta mudah dipahami karena berbentuk persentase.



## MAE (Mean Absolute Error)

Metriks ini juga penting untuk mengukur regresi dan distribusi frekuensinya.

# Data Understanding

Attribute	Data Type	Description
Type	Object	Type of used car
Region	Object	The region in which the used car was offered for sale.
Make	Object	The company name
Gear_Type	Object	Gear type size of used car
Origin	Object	Origin of used car
Options	Object	Options of used car
Year	Integer	Manufacturing year
Engine_Size	float	The engine size of used car
Mileage	Integer	Mileage of used car
Negotiable	Boolean	True if the price is 0, that means it is negotiable.
Price	Integer	Used car price.

# DATA CLEANING

# DATA INFO

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5624 entries, 0 to 5623
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Type              5624 non-null    object  
 1   Region            5624 non-null    object  
 2   Make              5624 non-null    object  
 3   Gear_Type         5624 non-null    object  
 4   Origin            5624 non-null    object  
 5   Options           5624 non-null    object  
 6   Year              5624 non-null    int64  
 7   Engine_Size       5624 non-null    float64 
 8   Mileage           5624 non-null    int64  
 9   Negotiable        5624 non-null    bool   
 10  Price             5624 non-null    int64  
dtypes: bool(1), float64(1), int64(3), object(6)
memory usage: 445.0+ KB
```

# DATA DUPLICATE

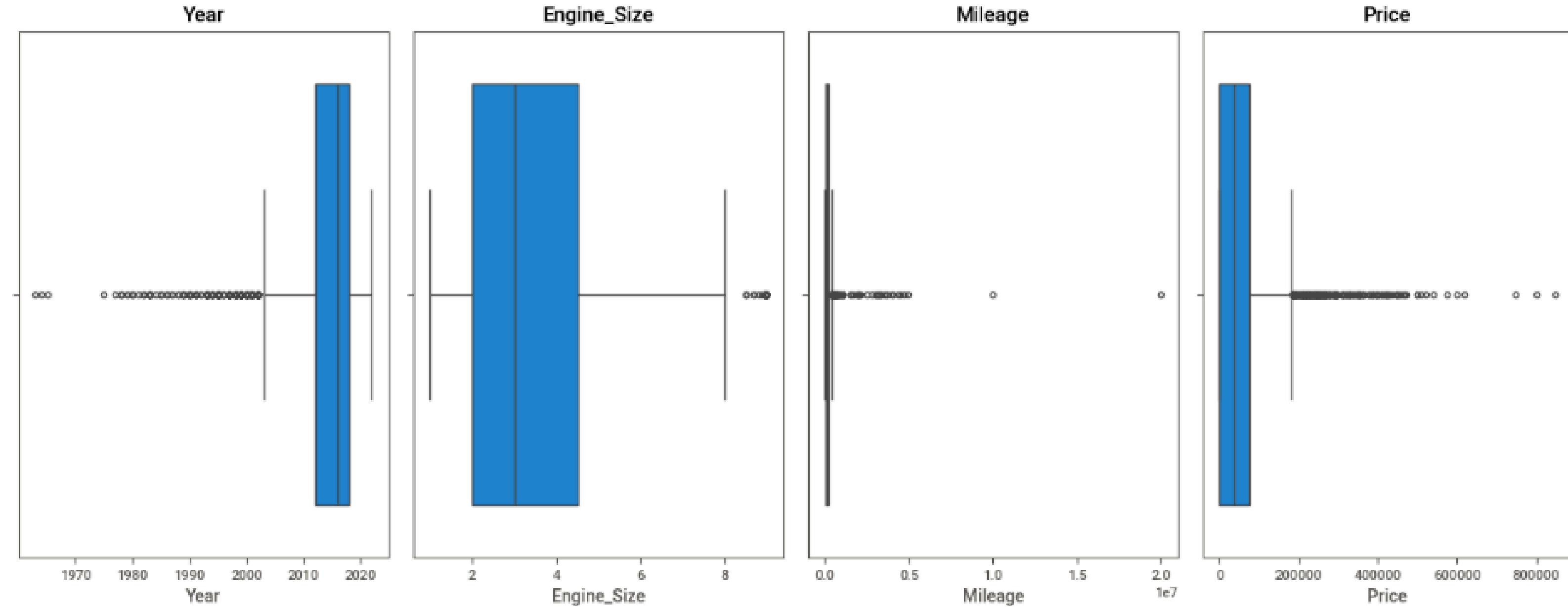
```
df.duplicated().sum()
```

✓ 0.0s

4 data duplikat harus di hapus

# OUTLIERS KOLOM

## "MILEAGE"



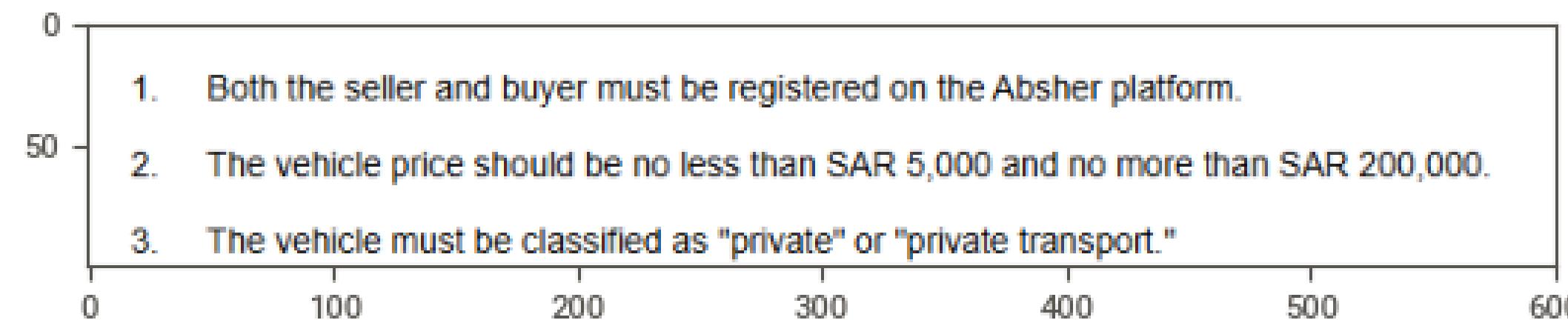
# DATA ANOMALI, PRICE = 0

	Type	Region	Make	Gear_Type	Origin	Options	Year	Engine_Size	Mileage	Negotiable	Price
0	Corolla	Abha	Toyota	Manual	Saudi	Standard	2013	1.4	421000	True	0
1	Yukon	Riyadh	GMC	Automatic	Saudi	Full	2014	8.0	80000	False	120000
2	Range Rover	Riyadh	Land Rover	Automatic	Gulf Arabic	Full	2015	5.0	140000	False	260000
3	Optima	Hafar Al-Batin	Kia	Automatic	Saudi	Semi Full	2015	2.4	220000	False	42000
4	FJ	Riyadh	Toyota	Automatic	Saudi	Full	2020	4.0	49000	True	0

# CUT OFF PRICE

```
df=df.query('Price>=5000')
```

✓ 0.0s



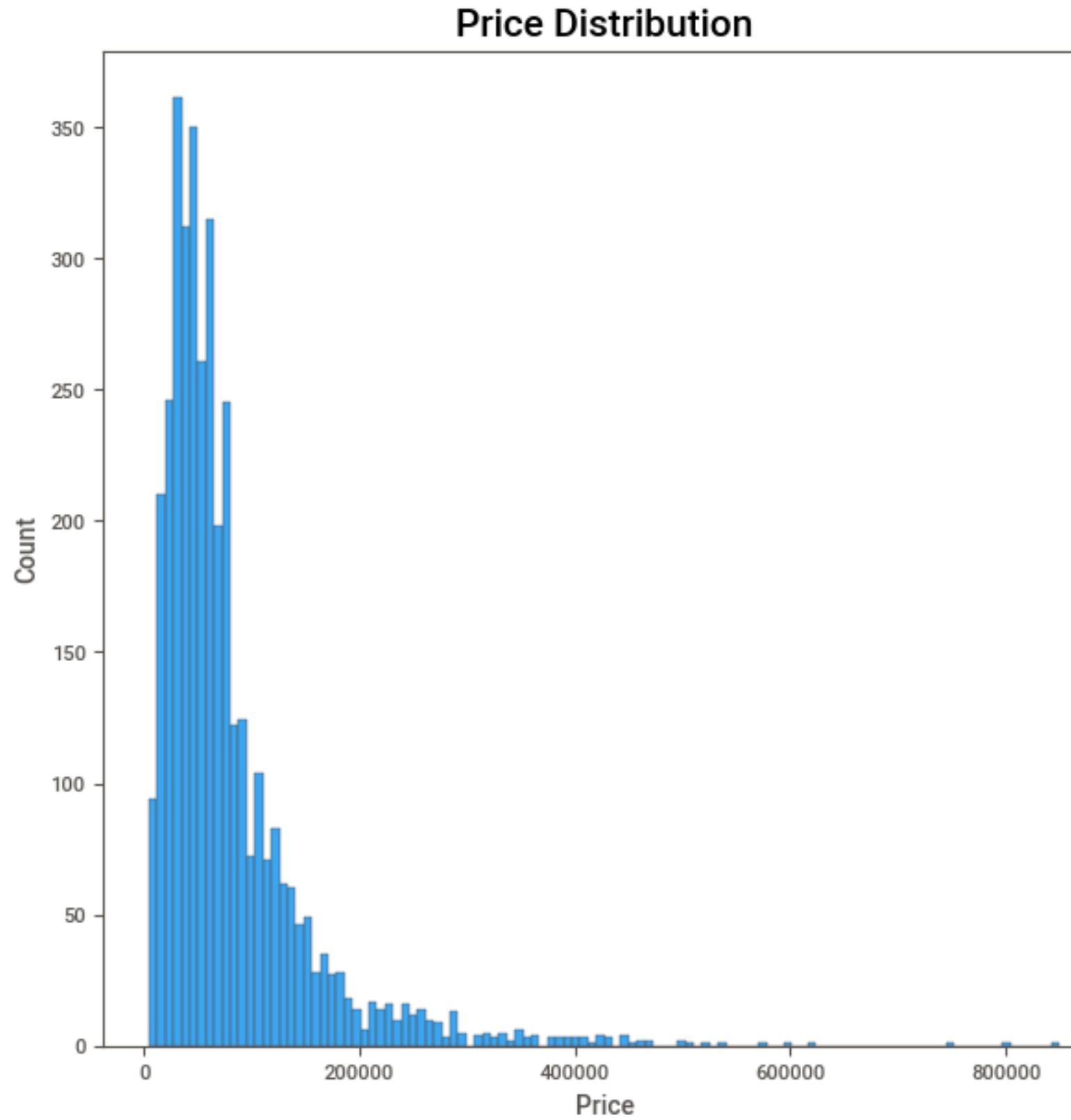
pengambilan minimum price untuk mobil bekas sebesar 5000 riyal, mengambil sebuah "reason" dari platform jual beli bekas mobil bekas di Arab Saudi

# EXPLORATORY DATA ANALYSIS

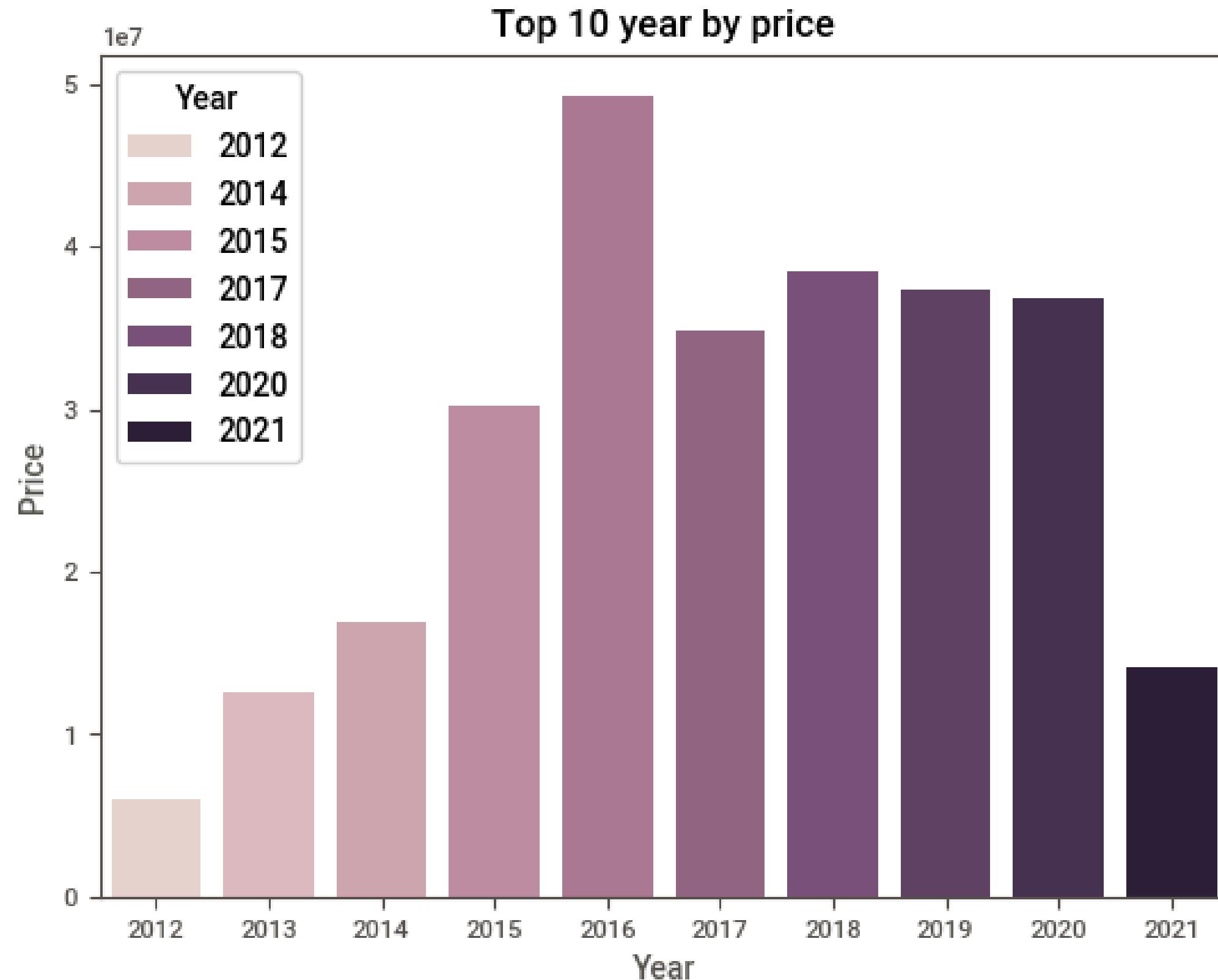
# DESCRIBE NUMERIC

	Year	Engine_Size	Mileage	Price
count	3755.000000	3755.000000	3.755000e+03	3755.000000
mean	2014.803196	3.188043	1.385940e+05	79417.952863
std	5.094043	1.460464	2.013716e+05	72671.122851
min	1963.000000	1.000000	1.000000e+02	5000.000000
25%	2013.000000	2.000000	4.900000e+04	36000.000000
50%	2016.000000	2.700000	1.040000e+05	58500.000000
75%	2018.000000	4.000000	1.840000e+05	95000.000000
max	2021.000000	9.000000	4.500000e+06	850000.000000

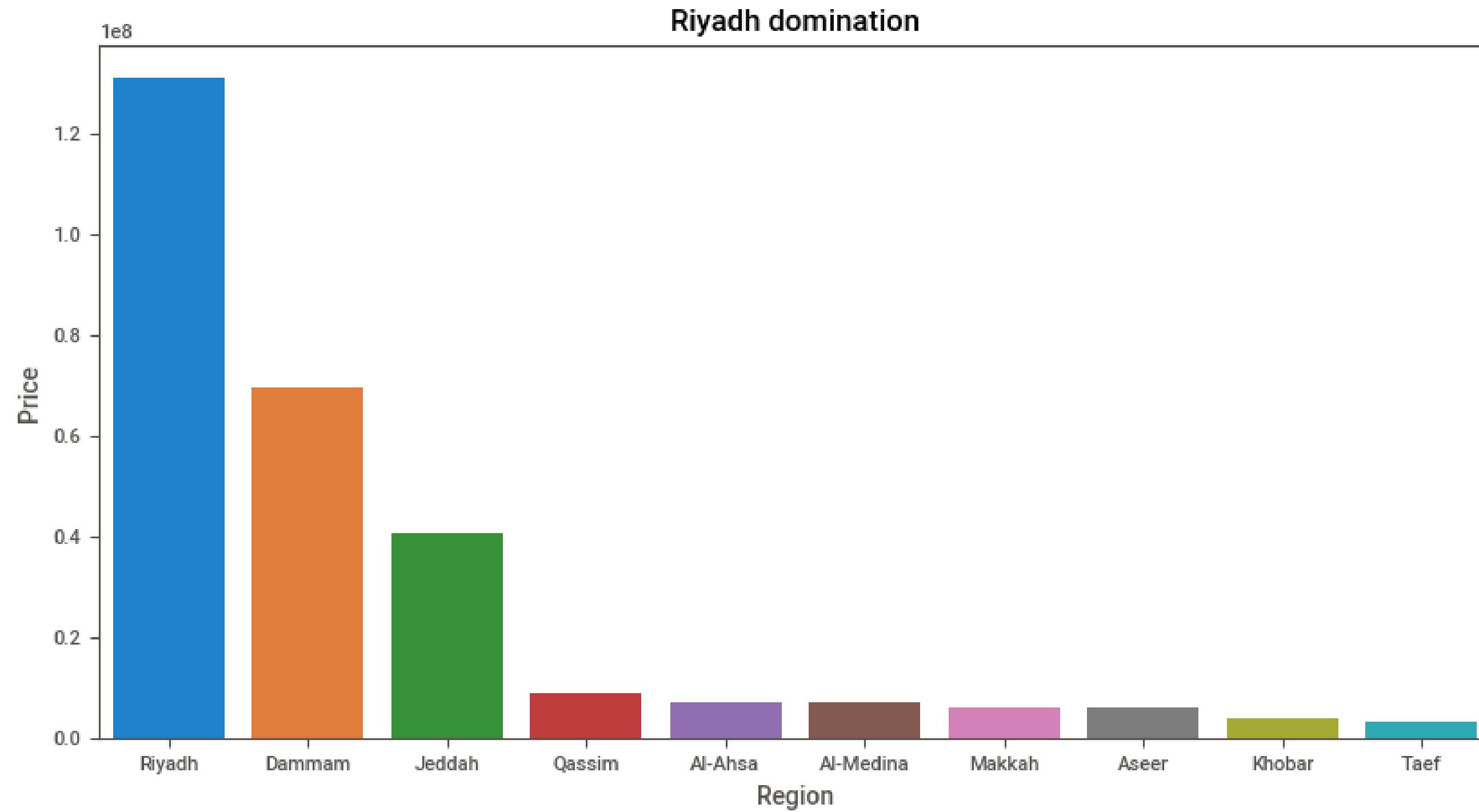
# DISTRIBUTIONS OF PRICE = RIGHT SKEW



# HIGHEST SUM OF PRICE ON 2016



# RIYADH DOMINATION OF MARKET



# DATA PREPROCESSING

# ENCODING, SCALING

Pengubahan data kategorik menjadi numerik menggunakan encoding yaitu **onehot** untuk kategori yang sedikit unique nya dan juga menggunakan **binary** bagi kategori yang memiliki banyak unique

Kemudian untuk data numerik dilakukan scaling menggunakan **robust scaler**

```
transformer = ColumnTransformer([
    ('onehot',OneHotEncoder(),['Gear_Type','Origin','Options']),
    ('binary',BinaryEncoder(),['Type','Region','Make']),
    ('robust',RobustScaler(),['Engine_Size','Mileage','Year'])
],remainder='passthrough')
transformer
```

✓ 0.0s

# DATA SPLITTING

Proses pembagian data antara data training dan data test sebanyak 80:20 untuk memudahkan model untuk belajar mengenai data.

```
x_train,x_test,y_train,y_test=train_test_split(  
    X,  
    y,  
    test_size=0.2,  
    random_state=0  
)  
✓ 0.0s
```

# MODELING

```
#define algoritma yang di gunakan
lr= LinearRegression()
knn= KNeighborsRegressor()
tree=DecisionTreeRegressor(random_state=42)
rf=RandomForestRegressor(random_state=42)
ada=AdaBoostRegressor(random_state=42)
gb=GradientBoostingRegressor(random_state=42)
ridge=Ridge(random_state=42)
lasso=Lasso(random_state=42)
xgb=XGBRegressor(random_state=42)
cat=CatBoostRegressor(random_state=42,verbose=False)
lgb=LGBMRegressor(random_state=42)
```

berikut adalah model model yang digunakan pada percobaan machine learning

```
#define metrik
scorer = ['neg_root_mean_squared_error',
          'neg_mean_absolute_percentage_error',
          'neg_mean_absolute_error']
```

menggunakan 3 evaluasi metrik yaitu **RMSE, MAE, dan MAPE**

```
crossval = KFold(n_splits=5, shuffle=True, random_state=42)
#looping
for algo in models:
    model=algo
    # membuat pipeline supaya tidak terjadi information leakage
    # dan untuk menggabungkan preprocessing dengan modeling

    pipe_model=Pipeline([
        ('preprocessing',transformer),
        ('modeling',model)
    ])

    #membuat cross validation
    cv_score=cross_validate(
        estimator=pipe_model,
        X=X_train,
        y=y_train,
        scoring=scorer,
        cv=crossval,
    )
```

menggunakan kfold dengan split sebanyak 5, dan menggunakan pipeline untuk menjalankan transformer dan model

# CROSSVAL RESULT

```
df_cv.sort_values(by='mean_MAE', ascending=False)
```

✓ 0.0s Python

		name	algo	mean_RMSE	std_RMSE	mean_MAE	std_MAE	mean_MAPE	std_MAPE
9	<catboost.core.CatBoostRegressor object at 0x1...	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-29987.661458	1891.364038	-14155.476174	412.327575	-0.196164	0.008244	
10	LGBMRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-33715.924154	2403.431187	-16206.370027	1082.212344	-0.219005	0.021320	
8	XGBRegressor(base_score=None, booster=None, ca...	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-34591.106168	1965.735369	-16749.435889	919.498656	-0.231105	0.016521	
3	RandomForestRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-37048.963587	2565.474950	-17588.609796	1263.894044	-0.243284	0.025448	
7	GradientBoostingRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-39093.845647	1921.241678	-19682.197930	1190.004980	-0.256386	0.020490	
1	KNeighborsRegressor()	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-41550.033485	1921.846991	-21081.590741	634.894659	-0.333200	0.021599	
2	DecisionTreeRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-52937.235995	5180.852653	-25265.390538	1858.729613	-0.384664	0.043377	
4	Ridge(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-53162.649941	3504.562762	-28138.497710	1637.113495	-0.383892	0.020772	
0	LinearRegression()	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-53055.671559	3416.694011	-28148.500202	1641.877925	-0.384027	0.020427	
6	AdaBoostRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-51600.617360	3184.326684	-29492.023783	969.840376	-0.416311	0.005380	
5	Lasso(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-77452.214361	3792.241795	-44371.515016	1561.446202	-0.727976	0.021761	

berdasarkan mean MAE

# CROSSVAL RESULT

```
df_cv.sort_values(by='mean_MAPE', ascending=False)
```

✓ 0.0s

Python

		name	algo	mean_RMSE	std_RMSE	mean_MAE	std_MAE	mean_MAPE	std_MAPE
9	<catboost.core.CatBoostRegressor object at 0x1...	TransformedTargetRegressor(func=<ufunc 'log'>,...	-29987.661458	1891.364038	-14155.476174	412.327575	-0.196164	0.008244	
10	LGBMRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...	-33715.924154	2403.431187	-16206.370027	1082.212344	-0.219005	0.021320	
8	XGBRegressor(base_score=None, booster=None, ca...	TransformedTargetRegressor(func=<ufunc 'log'>,...	-34591.106168	1965.735369	-16749.435889	919.498656	-0.231105	0.016521	
3	RandomForestRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...	-37048.963587	2565.474950	-17588.609796	1263.894044	-0.243284	0.025448	
7	GradientBoostingRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...	-39093.845647	1921.241678	-19682.197930	1190.004980	-0.256386	0.020490	
1	KNeighborsRegressor()	TransformedTargetRegressor(func=<ufunc 'log'>,...	-41550.033485	1921.846991	-21081.590741	634.894659	-0.333200	0.021599	
4	Ridge(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...	-53162.649941	3504.562762	-28138.497710	1637.113495	-0.383892	0.020772	
0	LinearRegression()	TransformedTargetRegressor(func=<ufunc 'log'>,...	-53055.671559	3416.694011	-28148.500202	1641.877925	-0.384027	0.020427	
2	DecisionTreeRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...	-52937.235995	5180.852653	-25265.390538	1858.729613	-0.384664	0.043377	
6	AdaBoostRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...	-51600.617360	3184.326684	-29492.023783	969.840376	-0.416311	0.005380	
5	Lasso(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...	-77452.214361	3792.241795	-44371.515016	1561.446202	-0.727976	0.021761	

berdasarkan mean MAPE

# CROSSVAL RESULT

```
df_cv.sort_values(by='mean_RMSE', ascending=False)
```

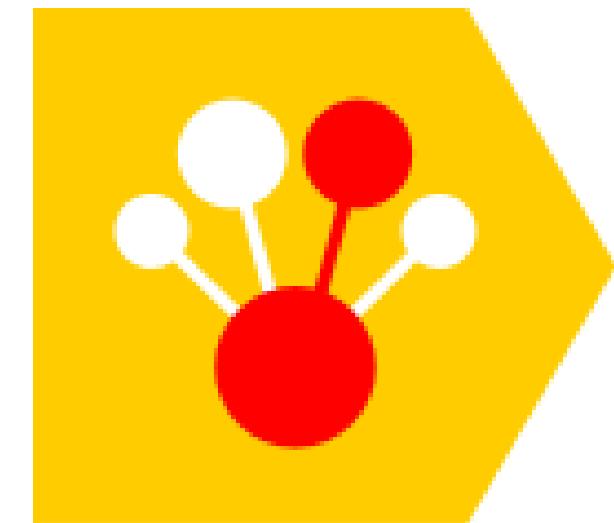
✓ 0.0s

Python

		name	algo	mean_RMSE	std_RMSE	mean_MAE	std_MAE	mean_MAPE	std_MAPE
9	<catboost.core.CatBoostRegressor object at 0x1...	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-29987.661458	1891.364038	-14155.476174	412.327575	-0.196164	0.008244	
10	LGBMRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-33715.924154	2403.431187	-16206.370027	1082.212344	-0.219005	0.021320	
8	XGBRegressor(base_score=None, booster=None, ca...	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-34591.106168	1965.735369	-16749.435889	919.498656	-0.231105	0.016521	
3	RandomForestRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-37048.963587	2565.474950	-17588.609796	1263.894044	-0.243284	0.025448	
7	GradientBoostingRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-39093.845647	1921.241678	-19682.197930	1190.004980	-0.256386	0.020490	
1	KNeighborsRegressor()	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-41550.033485	1921.846991	-21081.590741	634.894659	-0.333200	0.021599	
6	AdaBoostRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-51600.617360	3184.326684	-29492.023783	969.840376	-0.416311	0.005380	
2	DecisionTreeRegressor(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-52937.235995	5180.852653	-25265.390538	1858.729613	-0.384664	0.043377	
0	LinearRegression()	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-53055.671559	3416.694011	-28148.500202	1641.877925	-0.384027	0.020427	
4	Ridge(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-53162.649941	3504.562762	-28138.497710	1637.113495	-0.383892	0.020772	
5	Lasso(random_state=42)	TransformedTargetRegressor(func=<ufunc 'log'>,...)	-77452.214361	3792.241795	-44371.515016	1561.446202	-0.727976	0.021761	

berdasarkan mean RMSE

# BEST MODEL



## CatBoost

# BEST BENCHMARK

```
# Benchmark model terbaik
models = {
    'cat': CatBoostRegressor(random_state=42,verbose=False),
    'log_model' : TransformedTargetRegressor(cat, func=np.log, inverse_func=np.exp)
}

score_rmse = []
score_mae = []
score_mape = []

# Prediksi pada test set
for i in models:

    model = Pipeline([
        ('preprocessing', transformer),
        ('modeling', models[i])
    ])

    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    score_rmse.append(np.sqrt(mean_squared_error(y_test, y_pred)))
    score_mae.append(mean_absolute_error(y_test, y_pred))
    score_mape.append(mean_absolute_percentage_error(y_test, y_pred))

score_before_tuning = pd.DataFrame({'RMSE': score_rmse, 'MAE': score_mae, 'MAPE': score_mape}, index=models.keys())
score_before_tuning
```

# SCORE BEFORE TUNING

		RMSE	MAE	MAPE
	cat	23444.167857	13514.149623	0.225552
	log_model	22519.761849	12158.800863	0.177073

# HYPERPARAMETER TUNING

```
hyperparams= {
    'modeling_regressor_iterations': [500,1000,2000],
    'modeling_regressor_learning_rate': [0.01, 0.05, 0.1, 0.2, 0.3],
    'modeling_regressor_subsample': [0.7,0.8,0.9,1.0],
    'modeling_regressor_l2_leaf_reg': range(1,11),
    'modeling_regressor_border_count':range(32,65),
    'modeling_regressor_early_stopping_rounds':[100],
    'modeling_regressor_max_depth':range(1,11)}

#metrik
scorer = ['neg_root_mean_squared_error',
          'neg_mean_absolute_percentage_error',
          'neg_mean_absolute_error']
```

# HYPERPARAMETER TUNING

```
model = CatBoostRegressor(random_state=42, verbose=False)

# Apply TransformedTargetRegressor
log_model = TransformedTargetRegressor(model, func=np.log, inverse_func=np.exp)

# Preprocessing and pipeline
# Membuat algorithm chains
pipe_model=Pipeline([
    ('preprocessing',transformer),
    ('modeling',log_model)
])
crossval = KFold(n_splits=5, shuffle=True, random_state=42)
#define gridsearch
random=RandomizedSearchCV(
    estimator=pipe_model,
    cv=crossval,
    scoring=scorer,
    n_iter=250,
    n_jobs=-1,
    refit='neg_mean_absolute_percentage_error',
    param_distributions=hyperparams
)
random
```

# TUNING RESULT

CATBOOST

BEST\_SCORE: -0.19301798025217995

BEST\_PARAMS:

'MODELING\_\_REGRESSOR\_\_SUBSAMPLE': 0.8,

'MODELING\_\_REGRESSOR\_\_MAX\_DEPTH': 6,

'MODELING\_\_REGRESSOR\_\_LEARNING\_RATE': 0.1,

'MODELING\_\_REGRESSOR\_\_L2\_LEAF\_REG': 4,

'MODELING\_\_REGRESSOR\_\_ITERATIONS': 1000,

'MODELING\_\_REGRESSOR\_\_EARLY\_STOPPING\_ROUNDS': 100,

'MODELING\_\_REGRESSOR\_\_BORDER\_COUNT': 38

# PREDICT WITH TUNED MODEL

```
# Model XGBoost
model = {'cat': CatBoostRegressor(random_state=42),
          'log_model' : TransformedTargetRegressor(model, func=np.log, inverse_func=np.exp)}

# Define model terhadap estimator terbaik
cat_tuning = random.best_estimator_

# Fitting model
cat_tuning.fit(X_train, y_train)

# Predict test set
y_pred_cat_tuning = cat_tuning.predict(X_test)

# Simpan nilai metrics RMSE, MAE & MAPE setelah tuning
rmse_cat_tuning = np.sqrt(mean_squared_error(y_test, y_pred_cat_tuning))
mae_cat_tuning = mean_absolute_error(y_test, y_pred_cat_tuning)
mape_cat_tuning = mean_absolute_percentage_error(y_test, y_pred_cat_tuning)

score_after_tuning = pd.DataFrame({'RMSE': rmse_cat_tuning, 'MAE': mae_cat_tuning, 'MAPE': mape_cat_tuning}, index=model.keys())
score_after_tuning
```

# SCORE AFTER TUNING

		RMSE	MAE	MAPE
	cat	21916.813351	11839.437242	0.175523
	log_model	21916.813351	11839.437242	0.175523

Dengan menggunakan hyperparameter nilai nya meingkat, dan didapati bahwa persentase error nya di angka 17% model ini cukup bagus untuk digunakan

# CONCLUSION

- Berdasarkan pemodelan yang sudah kita lakukan menggunakan tiga evaluasi metrik **RMSE MAE DAN MAPE** dengan refit metrik nya MAPE didapati setelah hyperparameter tuning bahwa nilai error nya adalah 17%, dapat disimpulkan bahwa ketika nanti model ini digunakan pada data dengan rentang price yang sama, maka perkiraan harganya rata rata meleset kurang lebih 17% dari keakuratan harga aslinya. Namun apabila dilihat dari nilai error nya dibawah 19% dapat disimpulkan model ini cukup bagus untuk dijadikan alat prediksi.
- Kemudian model diperkuat dengan grafik dari redidual yang ternyata model ini bagus dan cenderung tidak bias karena rata rata nilai mendekati garis nol ,
- Selain itu, adanya limitasi pada rentang harga juga dapat membantu dalam mengurangi error prediksi, karena model lebih fokus pada data yang memiliki distribusi serupa, mengurangi kemungkinan prediksi yang jauh meleset.
- Namun tetap model ini masih perlu adanya peningkatan, karena keterbatasan feature dan juga jumlah data membuat model ini belum maksimal

# RECOMMENDATIONS

Beberapa rekomendasi yang dapat membantu meningkatkan performa model:

1. Perlu adanya petimbangan penambahan feature feature yang sesuai dengan kondisi mobil contohnya seperti apakah ada kerusakan pada body mobil, kondisi mesin, apakah ada variasi pada mobilnya, dll. Dengan ini membantu model untuk lebih memahami mengenai data sehingga dapat meningkatkan keakuratan model.
2. Limitasi harga (target) berdasarkan pembelajaran model yang telah kita lakukan dengan menggunakan limitasi harga sangat berpengaruh terhadap penurunan nilai error nya oleh karena itu dapat dilakukan limitasi harga yang lebih tinggi diharapkan dengan itu model bisa menjadi lebih baik lagi.
3. Jika diperlukan ada pembagian antara harga yang tinggi dan rendah dengan limitasi yang akan ditentukan oleh masing masing pihak dengan ini akan lebih tergambar untuk pemilihan model dan juga evaluasi metrik yang akan digunakan.
4. Karena syarah.com merupakan platform berbasis online maka semestinya terdapat data mengenai mobil yang banyak menjadi favorit pengguna dan juga mobil dengan engangement terbanyak hal ini bisa menjadi sebuah tambahan untuk menjadi feature dengan hal tersebut dapat diprediksi estimasi untk harga mobil yang mempunya demand lebih tinggi di bandingkan mobil lainnya.

# Terima Kasih