Used Car Price Prediction

Submitted by:

Yuvarani N

ACKNOWLEDGMENT

Data Sources:

I have scraped the used cars information from the below websites.

Olx - https://www.olx.in/

CarWale - https://www.carwale.com/

CarDheko - https://www.cardekho.com/

Cars24 - https://www.cars24.com/

For Model building and parameter tuning I have referred the https://scikit-learn.org/stable/ .

INTRODUCTION

- Business Problem Framing

With the covid 19 impact in the market, we have seen lot of changes in the car market. Now some cars are in demand hence making them costly and some are not in demand hence cheaper. One of our clients works with small traders, who sell used cars. With the change in market due to covid 19 impact, our client is facing problems with their previous car price valuation machine learning models. So, they are looking for new machine learning models from new data. This model can help the client to understand the current status of the car market and car prices and based on this they can evaluate the car price for sale.

- Conceptual Background of the Domain Problem

In India, due to covid 19 impact the automobile companies were going down in terms of their productivity and sales. Due to lockdown, lot of small- and large-scale industries are stopped their auto mobile parts manufacturing process. It made a big down in automobile industry and they started recovering slowly. This kind of impact affects the car price every year. So, we try to understand the current market price of the cars using the used car information.

- Review of Literature

We have tried collecting different type of cars from different brands and different locations from India. Based on the collected information and the analysis we found that car price is majorly varies based on the transmission type, body type, manufacturing year and fuel type. We observed that automated transmission type cars are highly valuated than manual transmission type cars. In our data set we have more of Maruti branded cars are available in the market and which is reasonably less in price and the cars from the

brand Rolls Royce are the highest car price in the market. This project will help to understand and evaluate the price of the cars based on the key factors. This analysis and model will help the clients to sell or fix the price of the cars based on their features and key factors.

- Motivation for the Problem Undertaken

The major motivation is that the problem is related to the automobile industry. We know that, how the covid 19 impacted the automobile industries. So, I have motivated myself to know and get the better understanding and analysis of the market at current situation.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

In this project we need to predict the car price, car price is the continues value, so it's a linear problem. In our data set we had few records with car price as 0. As its not the realistic data, removed those records from the data set.

There were lot of data cleaning steps taken in this project such as converting all the brand names and variant names data, etc to lower case and removed the special characters between the words.

Cleaned the target variable price as it was in the form of subject and contained some special character.

- Data Sources and their formats

Data sources used are below.

Olx - https://www.olx.in/

CarWale - https://www.carwale.com/

CarDheko - https://www.cardekho.com/

Cars24 - https://www.cars24.com/

Above are the online websites which are used to sell and buy the used cars. Scraped the required data from these websites.

Below are the data field extracted from the above websites.

Brand – The brand of the car(categorical)

Model – The model of the car(categorical)

Variant – The variant of the car(categorical)

Body type – Body type of the car(categorical)

Location – Location where the car is available for sale(categorical)

Km Driven – No.of kms driven(It was collected as object type and converted to continuous)

Transmission type - Transmission type of the car such as automatic or manual(Categorical)

Fuel type – Fuel type used in the car(Categorical)

Manufacture year – Year when the car has been manufactured(Categorical)

No.of.Owners – How many times the car has been transferred and registered to different owners(Categorical as the data type is object)

Price – the target variable(Continuous)

- Data Pre-processing Done

We have cleaned lot of data as we collected the data from different data source, the format of the data is different and we had to clean them to make it sense to the data set.

Below are the few processes followed as part of data cleaning process.

1. There were few missing values in the data set which are tagged as data '-'. Removed those records as it will not make sense to handle them by filling it.
2. Converted all the words to the lower case in the fields 'Brand', 'Model','Variant','Body type', 'location' and

'No.of.Owners'.It helps to remove the duplicate data with different cases.

Eg: Maruti, MARUTI – these two data are now considered as same word 'maruti'.

3. Removed the space and special characters from the above fields, and it also helped to remove the duplicates.

   Eg: Mahindra-Renault, Mahindra Renault – these two are now considered as same word 'mahindrarenault'.

4. Removed the ',' in between the numbers in the field of Km driven.

   Eg: 11,410 – this data is now converted to 11410. Converted from object type to int type data.

5. Cleaned the manufacturing year field by applying round function on the data.

   Eg: 2015.0 – now converted to 2015

6. Cleaned the target variable price. Converted from object type to int type data.

   Eg: 4,70,000 – now converted to 470000
   534999.0 – now converted to 534999

- Data Inputs- Logic- Output Relationships

Based on the data correlation matrix, its clear that transmission type of the car is 43% correlated to the target variable price. It means automatic type cars are higher in price than manual cars.

Body type of the cars are 31% correlated to the target variable price. It means that the cars with body type convertible are higher in price than other cars. Coupe body type cars are second highest in price. Minivan body type cars are lower in price than other body types.

Manufacturing year is 27% correlated to the target variable price. The manufacturing year shows the age of the car. When the age of the car is higher, the price is lower. When the age of the car goes lower the price is higher.

Fuel type of the cars is 20% correlated to the target variable. Hybrid fuel type cars are high in price than other cars. Electric cars

are second highest price cars and followed by diesel cars and petrol cars.

Km driven is 11% correlated to the target variable. It shows that when the number of km driven is less the price of the car is high. When the number of km driven goes high, the car price goes down.

- Hardware and Software Requirements and Tools Used

The is the bare minimum required to run this project

Hardware:

• Process: Intel core i5 and above

• RAM: 4GB and above

• SSD: 250GB and above

Software:

• Anaconda (Jupyter Notebook)

Libraries:

• **import pandas as pd**: pandas is a popular Python-based data analysis toolkit which can be imported using import pandas as pd.

It presents a diverse range of utilities, ranging from parsing multiple file formatsto converting an entire data table into a numpy matrix array. This makes pandas a trusted ally in data science and machine learning.

• **import numpy as np**: NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

• **import seaborn as sns**: Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas' data

structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

• **Import matplotlib.pyplot as plt**: matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure,creates a plotting area in a figure, plotssome linesin a plotting area, decorates the plot with labels, etc.

• from sklearn.preprocessing import StandardScalar

• from sklearn.preprocessing import PowerTransformer

• from sklearn.tree import DecisionTreeRegressor

• from sklearn.ensemble import RandomForestRegressor

• from sklearn.linear_model import LinearRegression

• from xgboost import KNeighborsRegressor

• from sklearn.ensemble import GradientBoostingRegressor

• import xgboost

• from xgboost import XGBRegressor

• from sklearn.metrics import r2_score,mean_squared_error,mean_absolute_error

• from sklearn.model_selection import cross_val_score,train_test_split

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

Our target variable is continuous data. So we need to handle this price prediction problem with regression algorithms.

Applied the PowerTransformer technique to remove data skewness.

Applied zscore technique to remove the outliers from the data set.

Applied LabelEncoder to encode the non-numeric values to numbers.

Applied standard scalar technique to scale the data to the standard scale.

- Testing of Identified Approaches (Algorithms)

Since our target variable is a continuous data I have used below regression algorithms and used ‘r2_score’, ‘mean_absolute_error’ and ‘mean_squared_error’ and ‘root_mean_squared_error’ metrics to calculate the model accuracy and find the best model out of the below models. Also used cross validation technique to make sure the model is not overfitting.

• LinearRegression

• RandomForestRegressor

• DecisionTreeRegressor

• KNeighborsRegressor

• GradientBoostingRegressor

• XGBRegressor

- Run and Evaluate selected models

```
lr=LinearRegression()
rf_reg=RandomForestRegressor()
dt_reg=DecisionTreeRegressor()
kn_reg=KNeighborsRegressor()
gb_reg=GradientBoostingRegressor()
ls=Lasso()
xgb=XGBRegressor()
```

```
model=[lr,rf_reg,dt_reg,kn_reg,xgb,ls,gb_reg]
for m in model:
    m.fit(X_train,y_train)
    y_pred=m.predict(X_val)
    print('Metrics for ',m)
    print('Accuracy score: ',r2_score(y_val,y_pred))
    print('Mean Absolute Error: ',mean_absolute_error(y_val,y_pred))
    print('Mean Squared Error: ',np.sqrt(mean_squared_error(y_val,y_pred)),'\n')
```

```
Metrics for  LinearRegression()
Accuracy score:  0.36203199379489803
Mean Absolute Error:  427240.672063632
Mean Squared Error:  821965.5331783117

Metrics for  RandomForestRegressor()
Accuracy score:  0.9188374652725624
Mean Absolute Error:  63710.88867030447
Mean Squared Error:  293178.391675638

Metrics for  DecisionTreeRegressor()
Accuracy score:  0.8459711352546978
Mean Absolute Error:  71455.00641783755
Mean Squared Error:  403882.6660993724

Metrics for  KNeighborsRegressor()
Accuracy score:  0.7360214797957454
Mean Absolute Error:  134855.51542664986
Mean Squared Error:  528735.4777246282

Metrics for  XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
Accuracy score:  0.9249673473705593
Mean Absolute Error:  96186.52035012053
Mean Squared Error:  281889.76486816054

Metrics for  Lasso()
Accuracy score:  0.3620320651164548
Mean Absolute Error:  427239.85328987165
Mean Squared Error:  821965.4872325412

Metrics for  GradientBoostingRegressor()
Accuracy score:  0.7308774616186435
Mean Absolute Error:  221316.26106138775
Mean Squared Error:  533862.225317374
```

Cross Validation check:

```
for m in model:
    print("Cross val score for ",m,cross_val_score(m,X_sc,y,cv=5).mean())
```

```
Cross val score for  LinearRegression() 0.10169832825605682
Cross val score for  RandomForestRegressor() 0.6612586600757814
Cross val score for  DecisionTreeRegressor() 0.3720882165772373
Cross val score for  KNeighborsRegressor() 0.32695937323487223
Cross val score for  XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None) 0.692748998757812
Cross val score for  Lasso() 0.10170046881300787
Cross val score for  GradientBoostingRegressor() 0.5036874037815621
```

Hyper Parameter tuning

```
from sklearn.model_selection import GridSearchCV
```

```
param={'n_estimators':[150,180,210]}
grd_srch=GridSearchCV(XGBRegressor(),param_grid=param,n_jobs=-1)
grd_srch.fit(X_train,y_train)
print(grd_srch.best_estimator_)
print(grd_srch.best_params_)
print(grd_srch.best_score_)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=210, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
{'n_estimators': 210}
0.9051386735269134
```

Finding best random state for the final model:

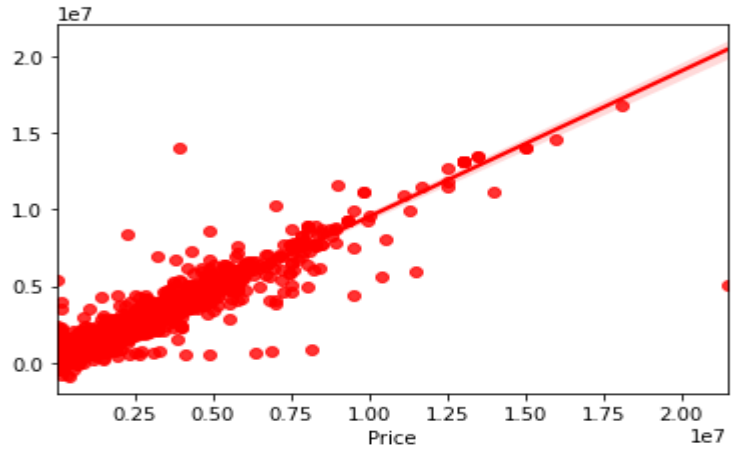
```
r_mean=0
rs=0
for i in range(1,20):
    gb_reg=XGBRegressor(n_estimators=210,random_state=i)
    gb_reg.fit(X_train,y_train)
    y_pred=gb_reg.predict(X_val)
    r_mean_square=np.sqrt(mean_squared_error(y_val,y_pred))
    if(r_mean==0):
        r_mean=r_mean_square
        rs=i
    elif(r_mean_square < r_mean):
        r_mean=r_mean_square
        rs=i
print("Best error: ",r_mean,"RS: ",rs)
gb_reg=XGBRegressor(n_estimators=210,random_state=rs)
gb_reg.fit(X_train,y_train)
y_pred=gb_reg.predict(X_val)
print('Accuracy: ',r2_score(y_val,y_pred))
print('Mean Absolute Error: ',mean_absolute_error(y_val,y_pred))
print('Mean Squared Error: ',mean_squared_error(y_val,y_pred))
print('Root Mean Squared Error: ',np.sqrt(mean_squared_error(y_val,y_pred)))
```

```
Best error:  266817.48610793194 RS:  1
Accuracy:  0.9327766329089393
Mean Absolute Error:  83272.62068522979
Mean Squared Error:  71191570892.95647
Root Mean Squared Error:  266817.48610793194
```

Based on the r2_score and root mean squared error and cross validation score, I have selected XGBRegressor as the best fit model to our data.

Hyper parameter tuning and finding best random state helped to improve the accuracy to 93% and reduce the Root mean squared error score.

```
sns.regplot(y_val,y_pred,color='r')
plt.show()
```

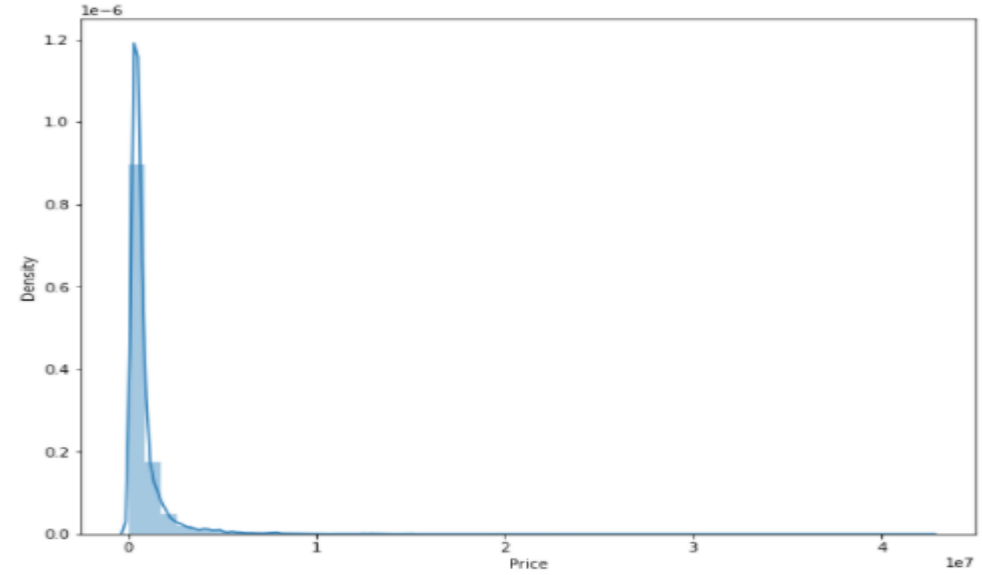


Above is the best fit line generated by our model. It seems not too bad.

- Visualizations

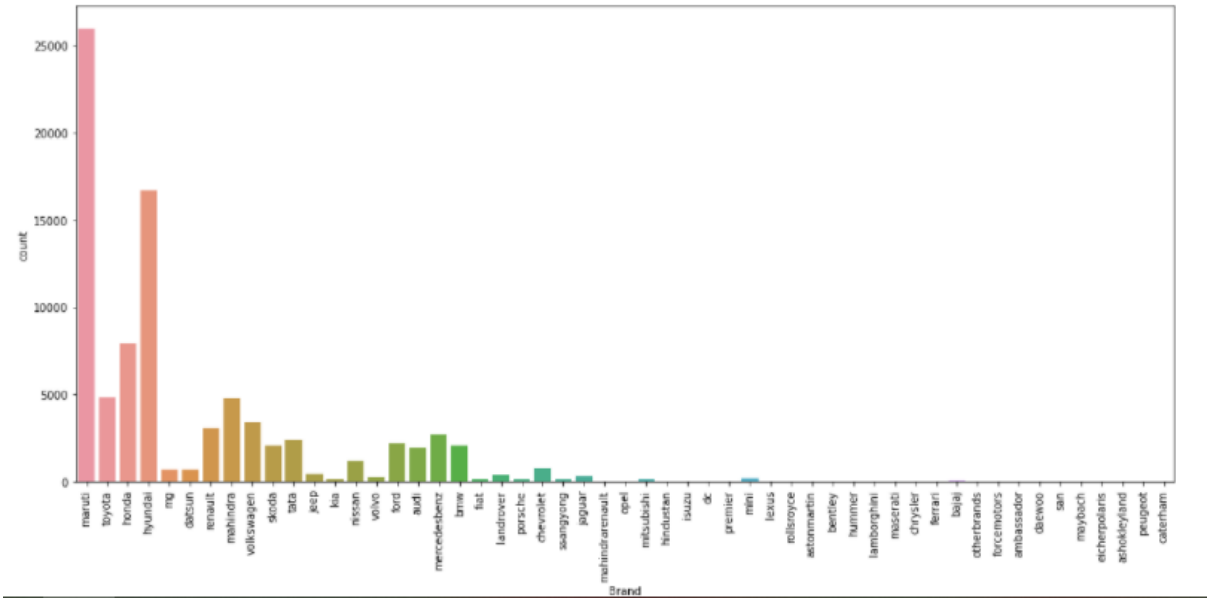
Our target variable ‘price’ is heavily right skewed.

```
plt.figure(figsize=(10,8))
sns.distplot(df['Price'])
plt.show()
```



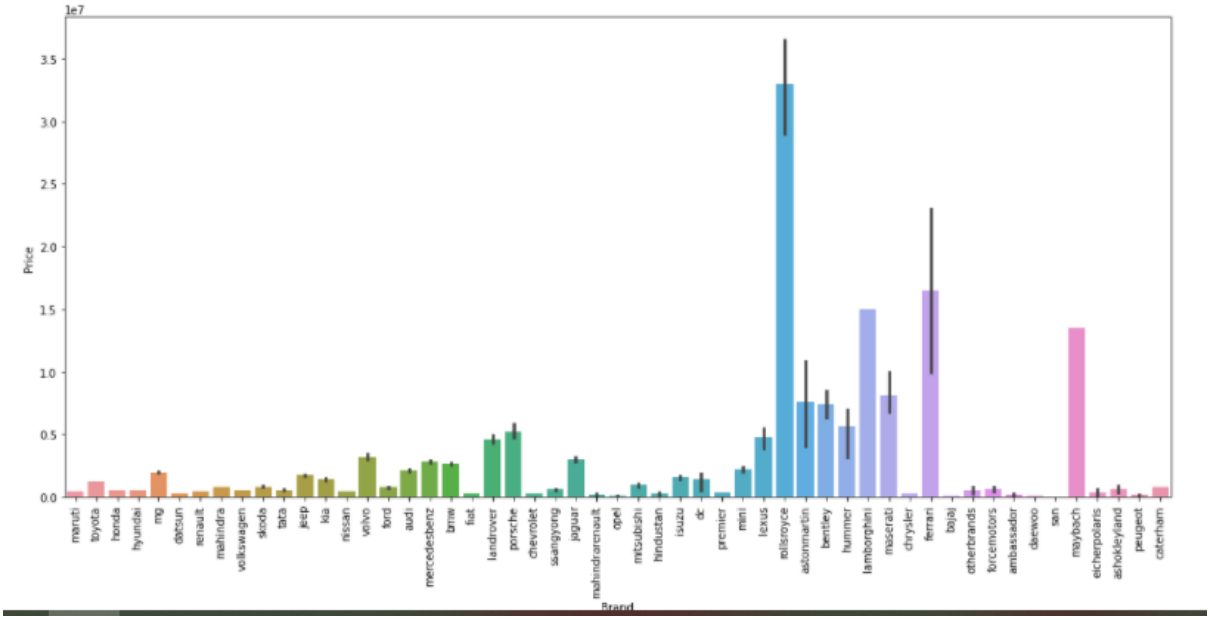
More of Maruti branded cars are available in market for sale.

```
plt.figure(figsize=(18,8))
sns.countplot(df['Brand'])
plt.xticks(rotation=90)
plt.show()
```



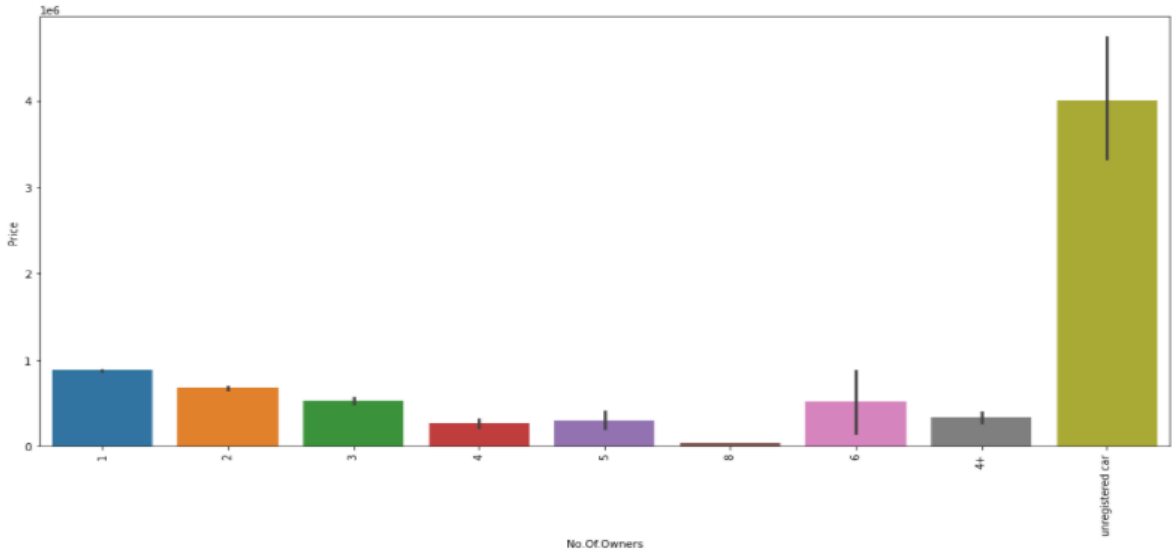
The price of Rolls Royce brand cars higher in price. Ferrari and Lamborghini are the second and third highest price in brands.

```
plt.figure(figsize=(18,8))
sns.barplot(df['Brand'],df['Price'])
plt.xticks(rotation=90)
plt.show()
```



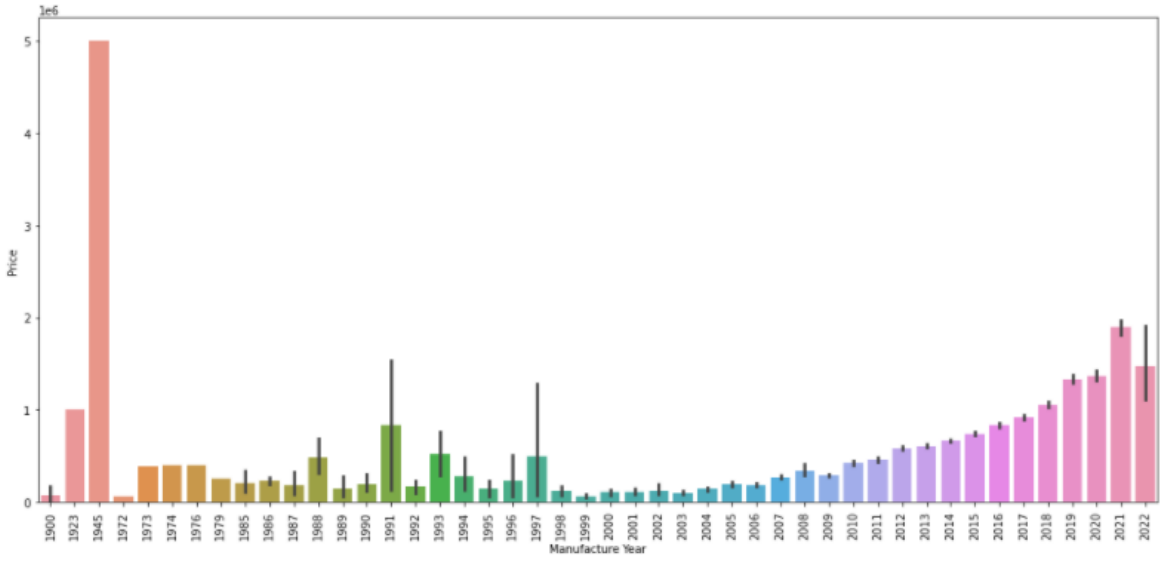
Unregistered cars are higher in price. When the number of owners count goes high, the care price goes down.

```
plt.figure(figsize=(18,8))
sns.barplot(df['No.Of.Owners'],df['Price'])
plt.xticks(rotation=90)
plt.show()
```



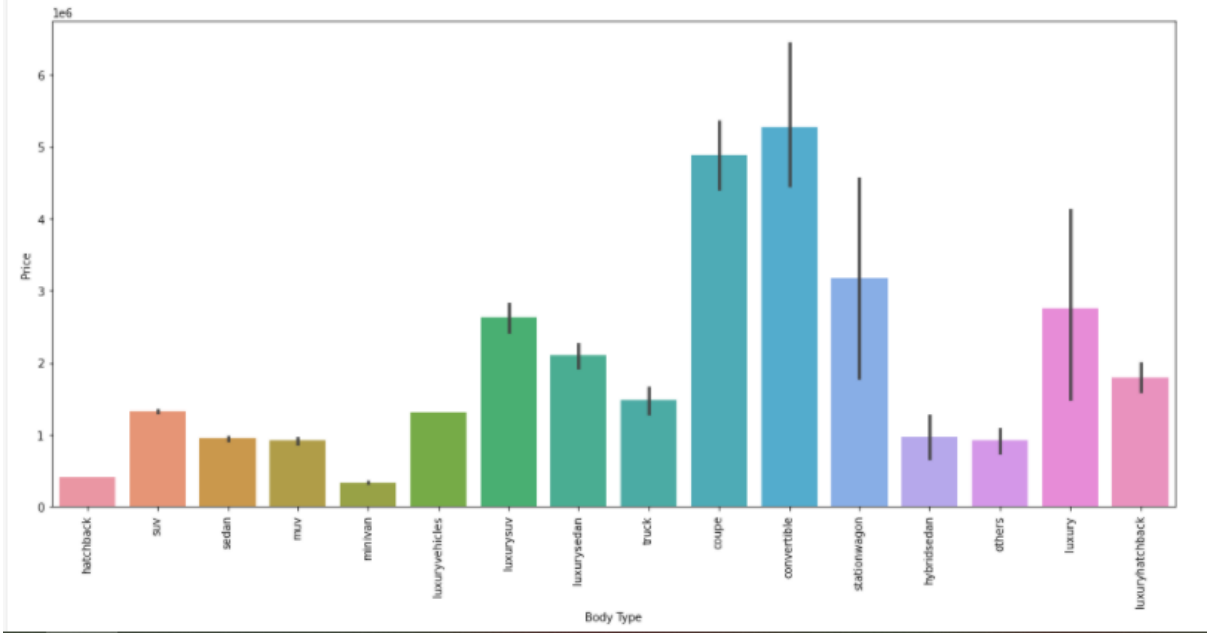
It shows that we have outliers in manufacturing year. The plot shows that when the age of the car is less, the price is high.

```
plt.figure(figsize=(18,8))
sns.barplot(df['Manufacture Year'],df['Price'])
plt.xticks(rotation=90)
plt.show()
```



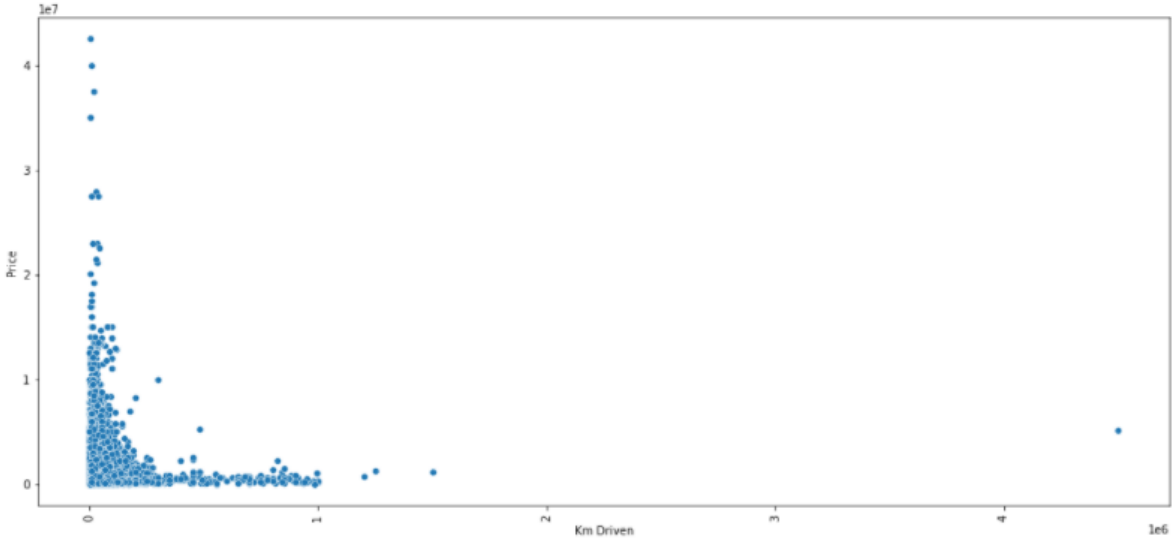
Convertible body type cars are higher in price. Coupe type cars are second highest in price.

```
plt.figure(figsize=(18,8))
sns.barplot(df['Body Type'],df['Price'])
plt.xticks(rotation=90)
plt.show()
```



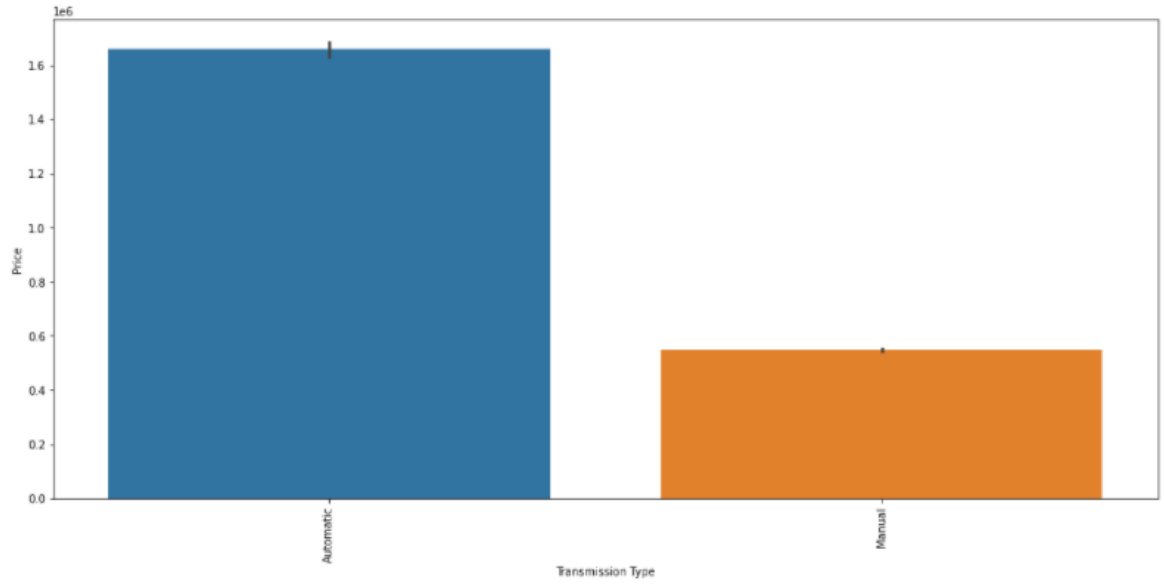
When the Km driven is less, the car price is high.

```
plt.figure(figsize=(18,8))
sns.scatterplot(df['Km Driven'],df['Price'])
plt.xticks(rotation=90)
plt.show()
```



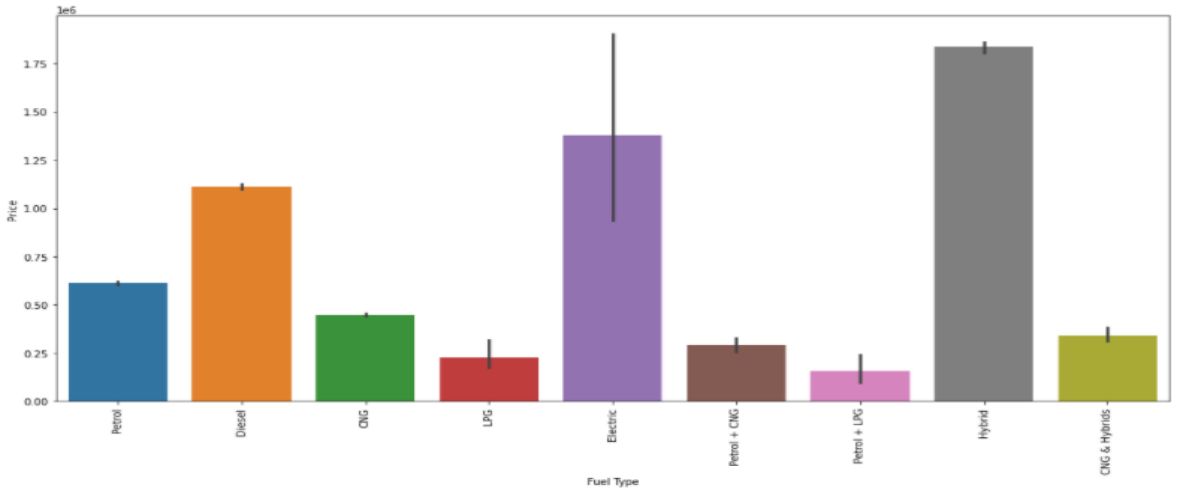
The price of automatic transmission type cars are higher than manual type cars.

```
plt.figure(figsize=(18,8))
sns.barplot(df['Transmission Type'],df['Price'])
plt.xticks(rotation=90)
plt.show()
```



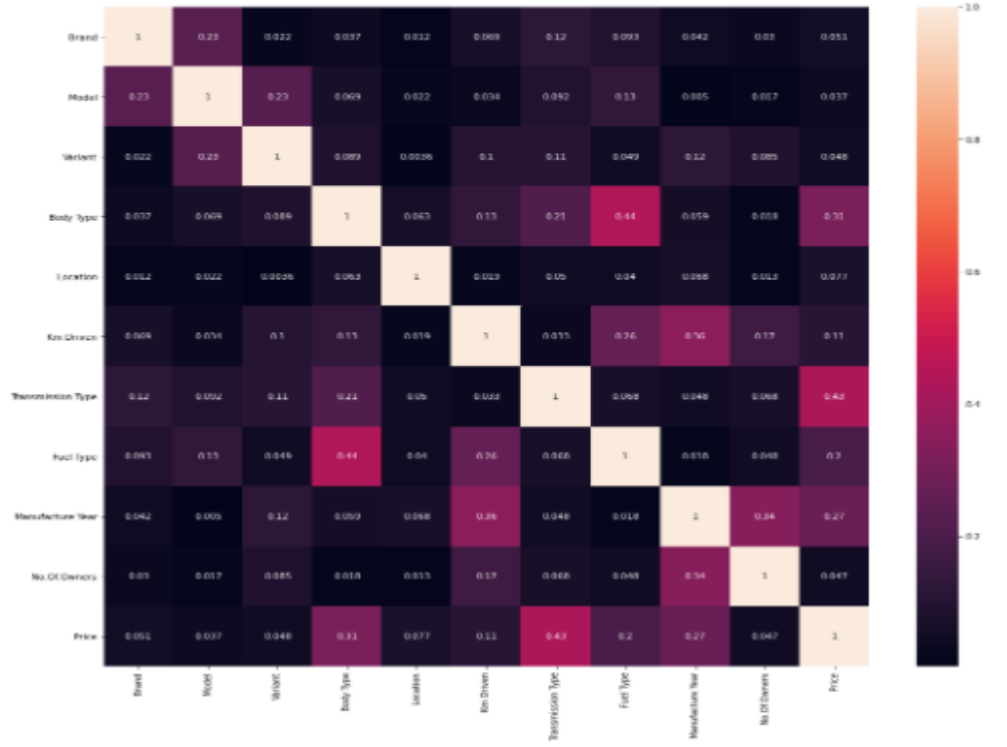
Price of Hybrid fuel type cars are higher than other fuel type cars. Electric cars are the second highest priced cars.

```
plt.figure(figsize=(18,8))
sns.barplot(df['Fuel Type'],df['Price'])
plt.xticks(rotation=90)
plt.show()
```



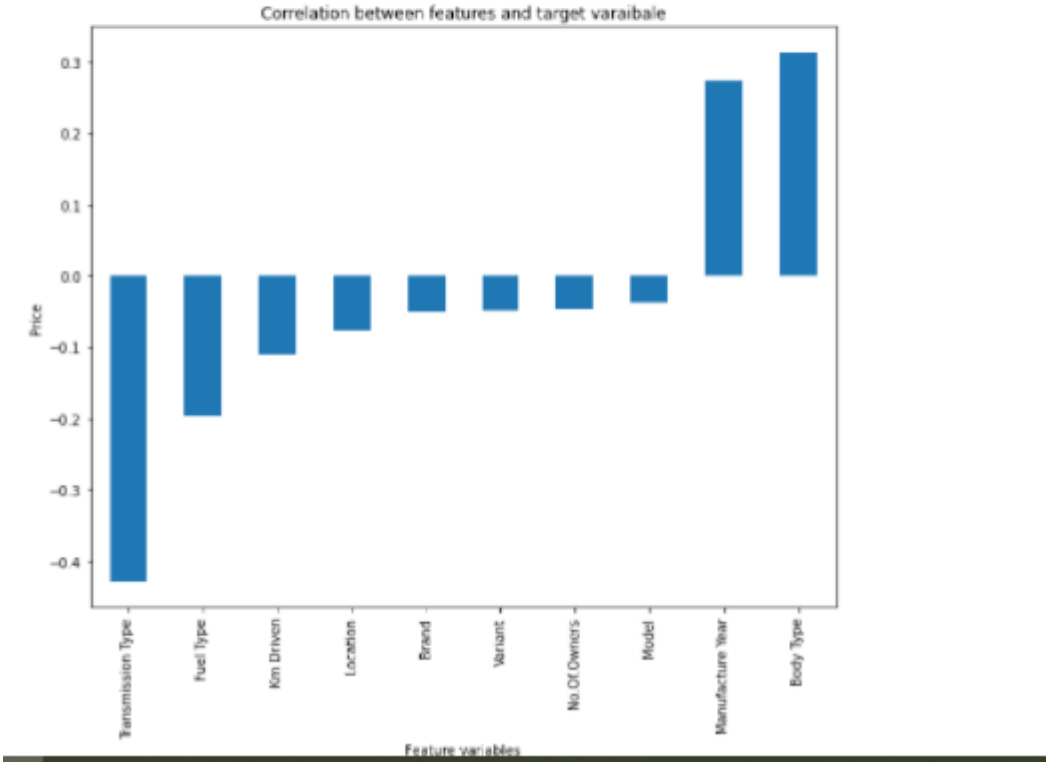
Below plot shows the correlation between features and target variable.

```
plt.figure(figsize=(18,16))
sns.heatmap(df.corr().abs(),annot=True)
plt.show()
```



Below diagram shows the relationship between the feature and target variable.

```
plt.figure(figsize=(10,8))
df.corr()['Price'].sort_values().drop(['Price']).plot(kind='bar')
plt.xlabel('Feature variables')
plt.ylabel('Price')
plt.xticks(rotation=90)
plt.title('Correlation between features and target varaibale')
plt.show()
```



- Interpretation of the Results

We have trained the model with different regression models and done cross validation check.

Based on the r2 score and root mean squared error value, I have selected XGBRegressor as the final model, and we tuned the parameters using GridSearchCV and we able to build a final model with 93% of accuracy.

CONCLUSION

- Key Findings and Conclusions of the Study

From the EDA steps, I could finalize the below points.

1. We have more maruthi brand cars in the market for sale. We can interpret this in such a way that people are do not want to use maruti cars anymore and they are seeking to explore new branded cars.

2. Price of Rolls royce cars are higher than other brand cars.

3. Price of Unregistered cars and 1st owner cars are higher than others.

4. When the number of owners increase, the price of the car is less.

5. The price of convertible body type cars are higher than other body types and followed by coupe.

6. When the Km driven is less, the car price is more.

7. The price of automatic transmission type cars are higher than manual cars.

8. The price of Hybrid fuel type cars are higher than other fuel type cars and followed by Electric cars and diesel cars.

9. RollsRoyce brand cars with automatic transmission type are costlier than other type of cars.

From the model building, we finalised that XGBRegressor suites best to our model.

- Learning Outcomes of the Study in respect of Data Science

It was really an interesting project; we started from scratch and completed the complete life cycle of a data scientist.

I have written web scraping scripts to collect the data and applied lot of cleaning technique and lot of pre-processing pipelines and build the best model using the regression algorithms.

It given a complete picture about the day-to-day work of an data scientist life cycle.

Initially it was bit challenging to scrape the data from the website. Then identified the json extraction approach and able to scrape the data easily using json api found from those websites.

- Limitations of this work and Scope for Future Work

Due to less time and very less experience, I was able to collect less data and very few features for the model. Collecting more data with more features such as colour of the car, may help the model to provide better accuracy.