CAR PRICE PREDICTION

FLIP ROBO

Presented by:

AJEET KUMAR SINGH

Problem Statement

- 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. We have to make car price valuation model. This project contains two phase:
- 1.Data Collection Phase
- 2.Model Building Phase

…Continued…

1.Data Collection Phase:

You have to scrape at least 5000 used cars data. You can scrape more data as well, it's up to you. more the data better the model.
In this section You need to scrape the data of used cars from websites (Olx, cardekho, Cars24 etc.) You need web scraping for this. You have to fetch data for different locations. The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are Brand, model, variant, manufacturing year, driven kilometers, fuel, number of owners, location and at last target variable Price of the car. This data is to give you a hint about important variables in used car model. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.
Try to include all types of cars in your data for example- SUV, Sedans, Coupe, minivan, Hatchback.

Note – The data which you are collecting is important to us. Kindly don't share it on any public platforms.

…Continued…

2.Model Building Phase:

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

Follow the complete life cycle of data science. Include all the steps like.

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model.

EDA(Exploratory Data Analysis)

Data Description

The dataset contains 18865 records (rows) and 9 features (columns). Here, we will provide a brief description of dataset features. Since the number of features is 9, we will attach the data description i.e., 'Model', 'Brand', 'Variant', 'Manufacturing_year', 'Driven_km’, 'Fuel_type', 'Transmission', 'Selling_Price', 'location'.

Target Variable

- Price(Selling Price): It’s continuous type of data, so the model approach is carried out for Regression analysis.

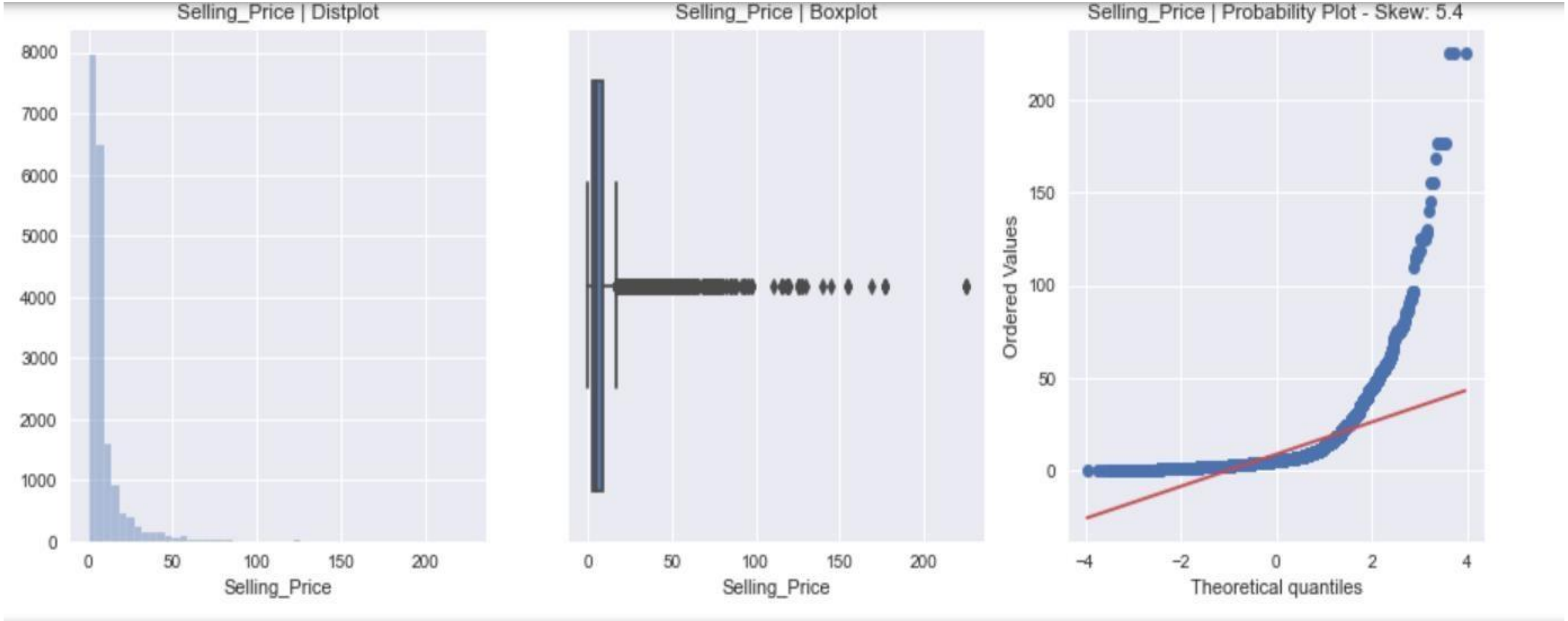
Regression:

It’s an analysis is used when you want to predict a continuous dependent variable from a number of independent variables.

Independent variables with more than two levels can also be used in regression analysis.

Visualization

Target Variable (Selling Price)



Data Pre-processing

Unique function for dataset

```
# Brand feature is replacing using replace function
df["Brand"].replace(["['Maruti']", "['Ford']", "['Mahindra']", "['Audi']", "['Toyota']",
       "['Volkswagen']", "['Honda']", "['Nissan']", "['Hyundai']",
       "['Mercedes-Benz']", "['Kia']", "['Datsun']", "['Tata']",
       "['Renault']", "['Skoda']", "['BMW']", "['Jaguar']",
       "['Chevrolet']", "['MG']", "['Volvo']", "['Jeep']", "['Land']",
       "['Force']", "['Mini']", "['Fiat']", "['Mitsubishi']",
       "['Porsche']", "['Lexus']", "['Ambassador']", "['Isuzu']",
       "['Aston']", "['Bentley']", "['Premier']", "['OpelCorsa']",
       "['Maserati']"],

                    ['Maruti', 'Ford', 'Mahindra', 'Audi', 'Toyota',
       'Volkswagen', 'Honda', 'Nissan', 'Hyundai',
       'Mercedes-Benz', 'Kia', 'Datsun', 'Tata',
       'Renault', 'Skoda', 'BMW', 'Jaguar',
       'Chevrolet', 'MG', 'Volvo', 'Jeep', 'Land',
       'Force', 'Mini', 'Fiat', 'Mitsubishi',
       'Porsche', 'Lexus', 'Ambassador', 'Isuzu',
       'Aston', 'Bentley', 'Premier', 'OpelCorsa',
       'Maserati'],inplace=True)
```

```
# Now, lets see we replaced Brand into unique function for further process.
df["Brand"].unique()
```

```
array(['Maruti', 'Ford', 'Mahindra', 'Audi', 'Toyota', 'Volkswagen',
       'Honda', 'Nissan', 'Hyundai', 'Mercedes-Benz', 'Kia', 'Datsun',
       'Tata', 'Renault', 'Skoda', 'BMW', 'Jaguar', 'Chevrolet', 'MG',
       'Volvo', 'Jeep', 'Land', 'Force', 'Mini', 'Fiat', 'Mitsubishi',
       'Porsche', 'Lexus', 'Ambassador', 'Isuzu', 'Aston', 'Bentley',
       'Premier', 'OpelCorsa', 'Maserati'], dtype=object)
```

No Null values in Dataset

```
#Check the null values in dataset
df.isnull().sum()
```

```
Model                0
Brand                0
Variant              0
Manufacturing_year   0
Driven_km            0
Fuel_type            0
Transmission         0
Selling_Price        0
location             0
dtype: int64
```

Dropped Features

1. Manufacturing Year
2. Current Year
3. Model
4. Variants

Adding Features in Datasets

Adding Current year in data frame

```
df["Current Year"] = 2021
df.head()
```

	Model	Brand	Variant	Manufacturing_year	Driven_km	Fuel_type	Transmission	Selling_Price	location	Current Year
0	['Eeco']	Maruti	5 Seater AC BSIV	2016	45347	Petrol	Manual	3.81	Ahmedabad	2021
1	['Eeco']	Maruti	5 Seater AC	2020	19627	Petrol	Manual	4.70	Ahmedabad	2021
2	['Eeco']	Maruti	5 Seater AC	2012	57341	Petrol	Manual	2.79	Ahmedabad	2021
3	['Eeco']	Maruti	5 Seater AC	2020	17116	Petrol	Manual	4.72	Ahmedabad	2021
4	['Eeco']	Maruti	5 Seater AC BSIV	2019	14161	Petrol	Manual	4.57	Ahmedabad	2021

Created number of year by subtracting current year and manufacturing year

```
df["no_of_year"] = df["Current Year"] - df["Manufacturing_year"]
df.head()
```

	Model	Brand	Variant	Manufacturing_year	Driven_km	Fuel_type	Transmission	Selling_Price	location	Current Year	no_of_year
0	['Eeco']	Maruti	5 Seater AC BSIV	2016	45347	Petrol	Manual	3.81	Ahmedabad	2021	5
1	['Eeco']	Maruti	5 Seater AC	2020	19627	Petrol	Manual	4.70	Ahmedabad	2021	1
2	['Eeco']	Maruti	5 Seater AC	2012	57341	Petrol	Manual	2.79	Ahmedabad	2021	9
3	['Eeco']	Maruti	5 Seater AC	2020	17116	Petrol	Manual	4.72	Ahmedabad	2021	1
4	['Eeco']	Maruti	5 Seater AC BSIV	2019	14161	Petrol	Manual	4.57	Ahmedabad	2021	2

Data Cleaning

Encoding of Data Frame:

The Encoding Technique is used for this problem:

1. One hot encoding technique with multiple variables.
2. One hot encoding technique.

Firstly, proceed with One hot encoding technique with multiple variables for particular features i.e., Brand

```
df = df[['Driven_km', 'Fuel_type', 'Transmission', 'Selling_Price', 'location', 'Brand_Maruti',
         'Brand_Hyundai', 'Brand_Honda', 'Brand_Toyota', 'Brand_Mahindra', 'Brand_Ford',
         'Brand_Volkswagen', 'Brand_Mercedes-Benz', 'Brand_BMW', 'Brand_Renault','no_of_year']]
df.head()
```

	Driven_km	Fuel_type	Transmission	Selling_Price	location	Brand_Maruti	Brand_Hyundai	Brand_Honda	Brand_Toyota	Brand_Mahindra	Brand_Ford
0	45347	Petrol	Manual	3.81	Ahmedabad	1	0	0	0	0	0
1	19627	Petrol	Manual	4.70	Ahmedabad	1	0	0	0	0	0
2	57341	Petrol	Manual	2.79	Ahmedabad	1	0	0	0	0	0
3	17116	Petrol	Manual	4.72	Ahmedabad	1	0	0	0	0	0
4	14161	Petrol	Manual	4.57	Ahmedabad	1	0	0	0	0	0

The new data frame is created using one hot encoding technique with multiple variables.
Secondly, proceed with One hot encoding technique i.e., transmission, location and fuel types.

```
df = pd.get_dummies(df, drop_first = True)
df.head()
```

Fuel_type_Diesel	Fuel_type_Electric	Fuel_type_LPG	Fuel_type_Petrol	Transmission_Manual	location_Bangalore	location_Chennai	location_Delhi NCR
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0

Now, let's we can see all features is converted into numerical one after proceeding with encoding technique.

Statistical Summary

To see statistical information about the non-numerical columns in our dataset:

```python
# Stastistical summary
df_describe=df.describe()
df_describe
```

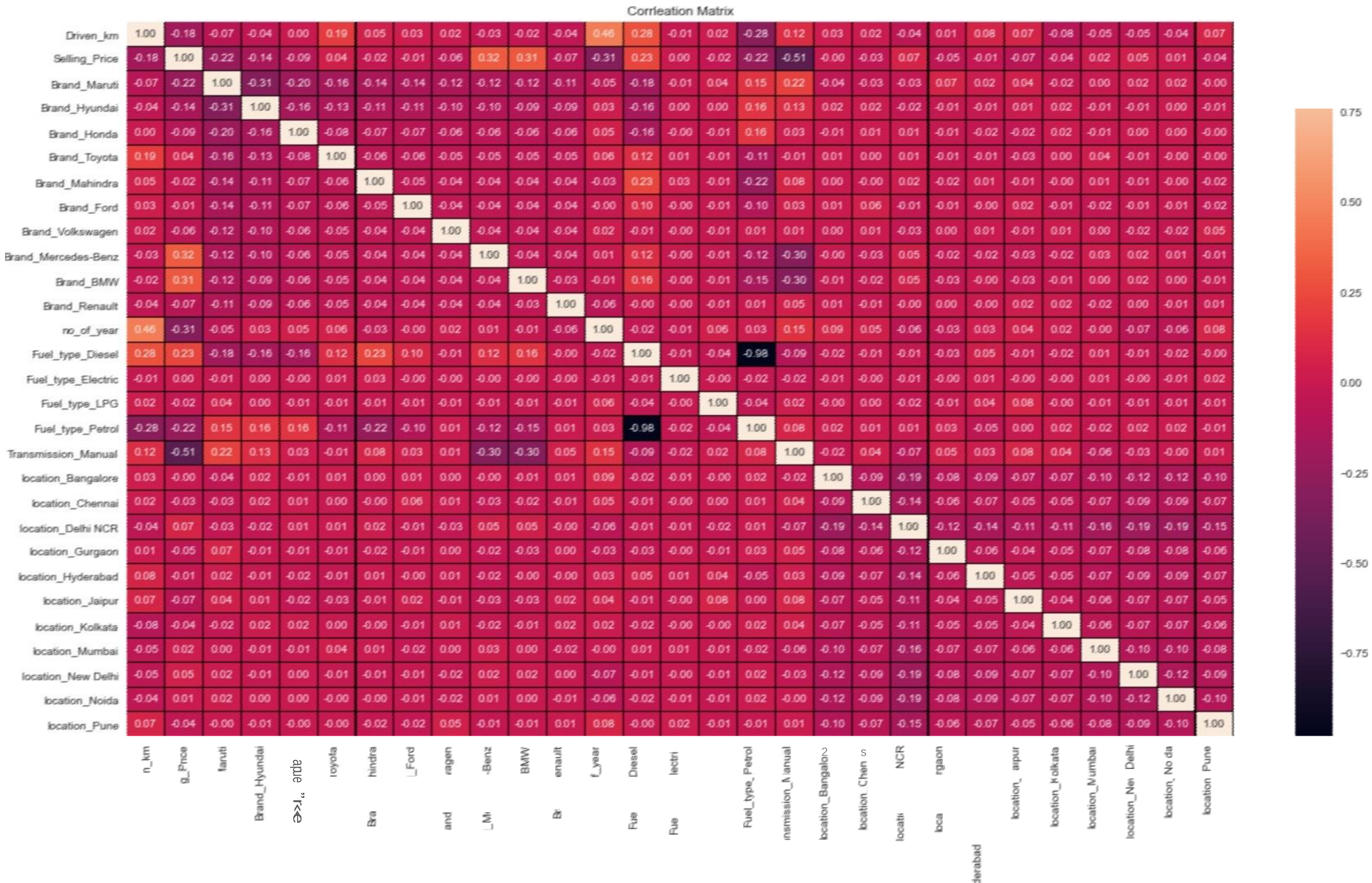
	Driven_km	Selling_Price	Brand_Maruti	Brand_Hyundai	Brand_Honda	Brand_Toyota	Brand_Mahindra	Brand_Ford	Brand_Volkswagen
count	18865.000000	18865.000000	18865.000000	18865.000000	18865.000000	18865.000000	18865.000000	18865.000000	18865.000000
mean	56623.499443	8.956964	0.280891	0.193692	0.092817	0.061914	0.047283	0.045004	0.037000
std	38608.147957	11.926699	0.449446	0.395201	0.290184	0.241005	0.212250	0.207318	0.188766
min	472.000000	0.300000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	32817.000000	3.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	54000.000000	5.470000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	73000.000000	8.900000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	886253.000000	225.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Correlation matrix:

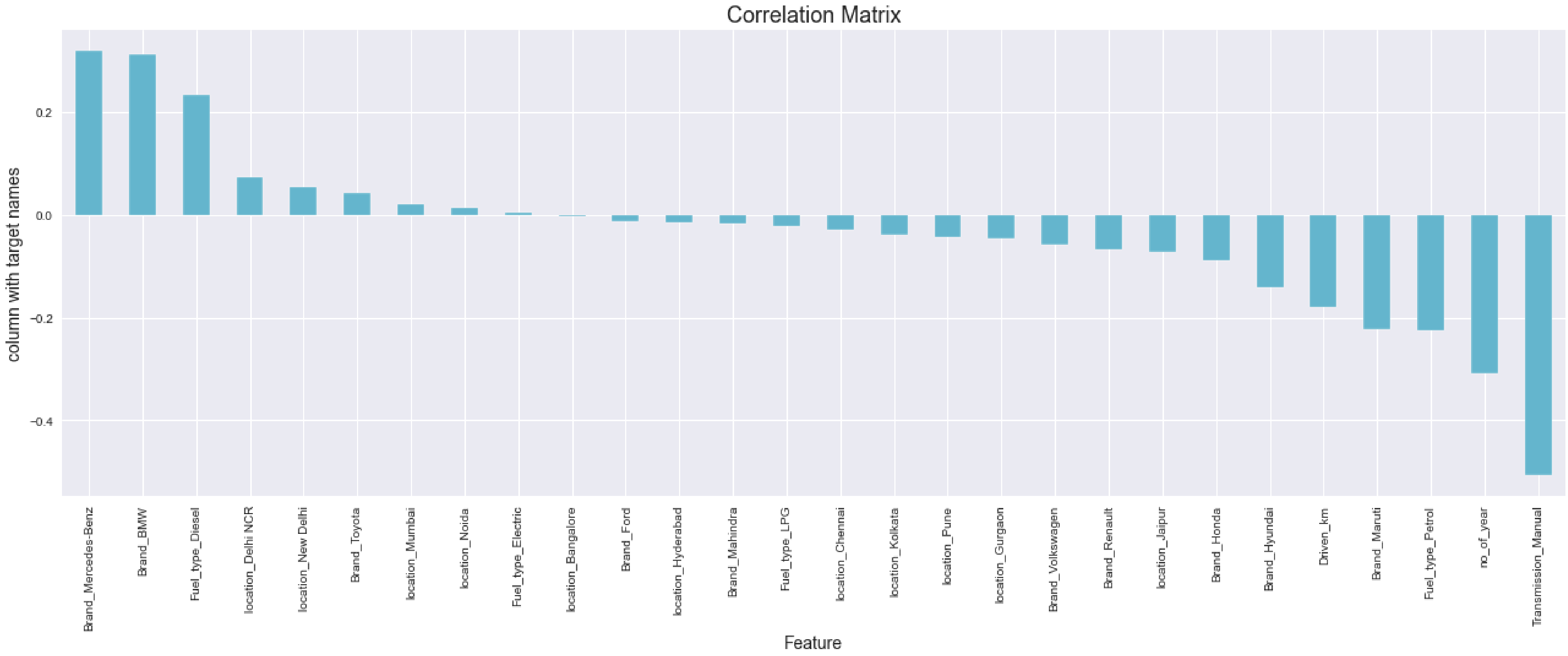
A correlation matrix is simply a table which displays the correlation. The measure is best used in variables that demonstrate a linear relationship between each other. The fit of the data can be visually represented in a heatmap.

```
corr_mat=df.corr()
corr_mat
```

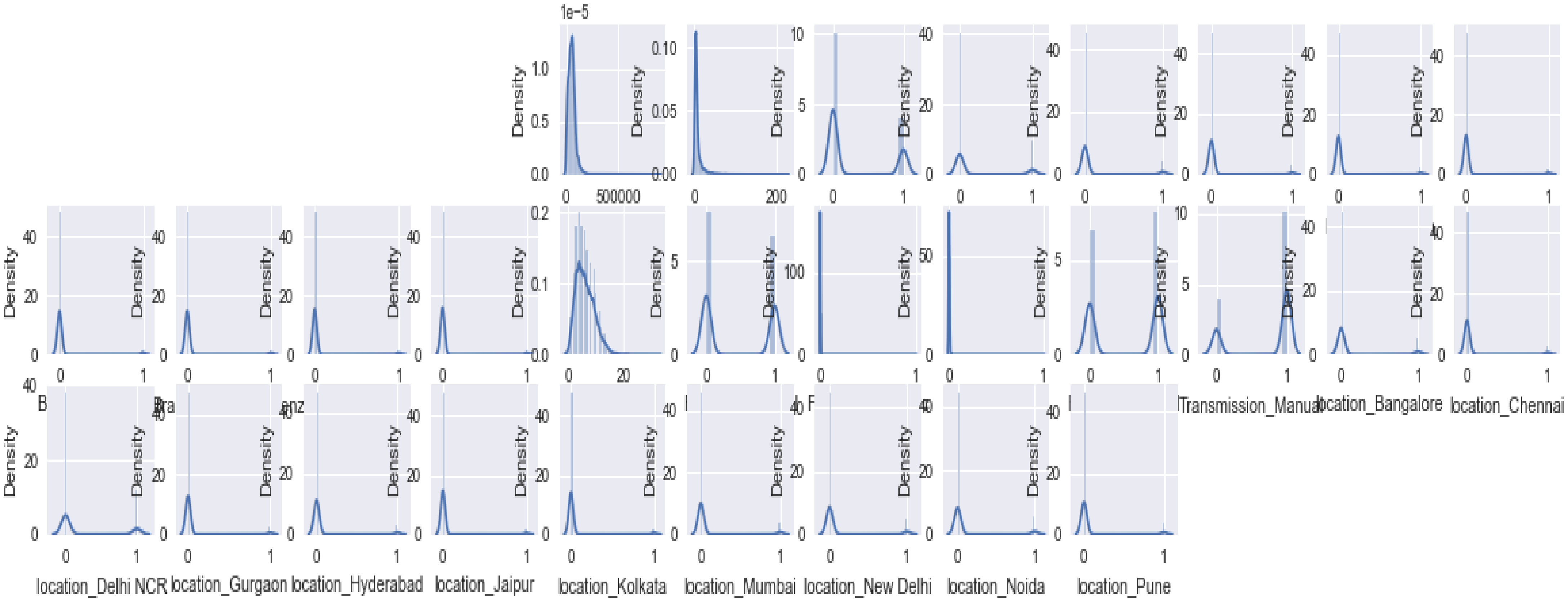
	Driven_km	Selling_Price	Brand_Maruti	Brand_Hyundai	Brand_Honda	Brand_Toyota	Brand_Mahindra	Brand_Ford	Brand_Volkswagen
Driven_km	1.000000	-0.179689	-0.068184	-0.036379	0.002971	0.193974	0.049966	0.028520	0.017549
Selling_Price	-0.179689	1.000000	-0.221589	-0.140579	-0.088945	0.042553	-0.016334	-0.012836	-0.056896
Brand_Maruti	-0.068184	-0.221589	1.000000	-0.306321	-0.199912	-0.160562	-0.139233	-0.135674	-0.122506
Brand_Hyundai	-0.036379	-0.140579	-0.306321	1.000000	-0.156774	-0.125915	-0.109189	-0.106397	-0.096071
Brand_Honda	0.002971	-0.088945	-0.199912	-0.156774	1.000000	-0.082175	-0.071259	-0.069437	-0.062698
Brand_Toyota	0.193974	0.042553	-0.160562	-0.125915	-0.082175	1.000000	-0.057233	-0.055769	-0.050357
Brand_Mahindra	0.049966	-0.016334	-0.139233	-0.109189	-0.071259	-0.057233	1.000000	-0.048361	-0.043667
Brand_Ford	0.028520	-0.012836	-0.135674	-0.106397	-0.069437	-0.055769	-0.048361	1.000000	-0.042551
Brand_Volkswagen	0.017549	-0.056896	-0.122506	-0.096071	-0.062698	-0.050357	-0.043667	-0.042551	1.000000
Brand_Mercedes-Benz	-0.027266	0.318818	-0.122232	-0.095856	-0.062558	-0.050244	-0.043570	-0.042456	-0.038335
Brand_BMW	-0.022306	0.311755	-0.115882	-0.090876	-0.059308	-0.047634	-0.041306	-0.040250	-0.036344
Brand_Renault	-0.039034	-0.067144	-0.112493	-0.088219	-0.057574	-0.046241	-0.040098	-0.039073	-0.035281
no_of_year	0.463704	-0.308634	-0.053099	0.033332	0.048541	0.063387	-0.031814	-0.001253	0.018670
Fuel_type_Diesel	0.275993	0.233153	-0.180579	-0.157495	-0.157331	0.118767	0.227800	0.102011	-0.007800


Correlation Matrix
Driven_km
Selling_Price
Brand_Maruti
Brand_Hyundai
Brand_Honda
Brand_Toyota
Brand_Mahindra
Brand_Ford
Brand_Volkswagen
Brand_Mercedes-Benz
Brand_BMW
Brand_Renault
no_of_year
Fuel_type_Diesel
Fuel_type_Electric
Fuel_type_LPG
Fuel_type_Petrol
Transmission_Manual
location_Bangalore
location_Chennai
location_Delhi NCR
location_Gurgaon
location_Hyderabad
location_Jaipur
location_Kolkata
location_Mumbai
location_New Delhi
location_Noida
location_Pune
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Checking the columns which are positively and negative correlated with the target columns:

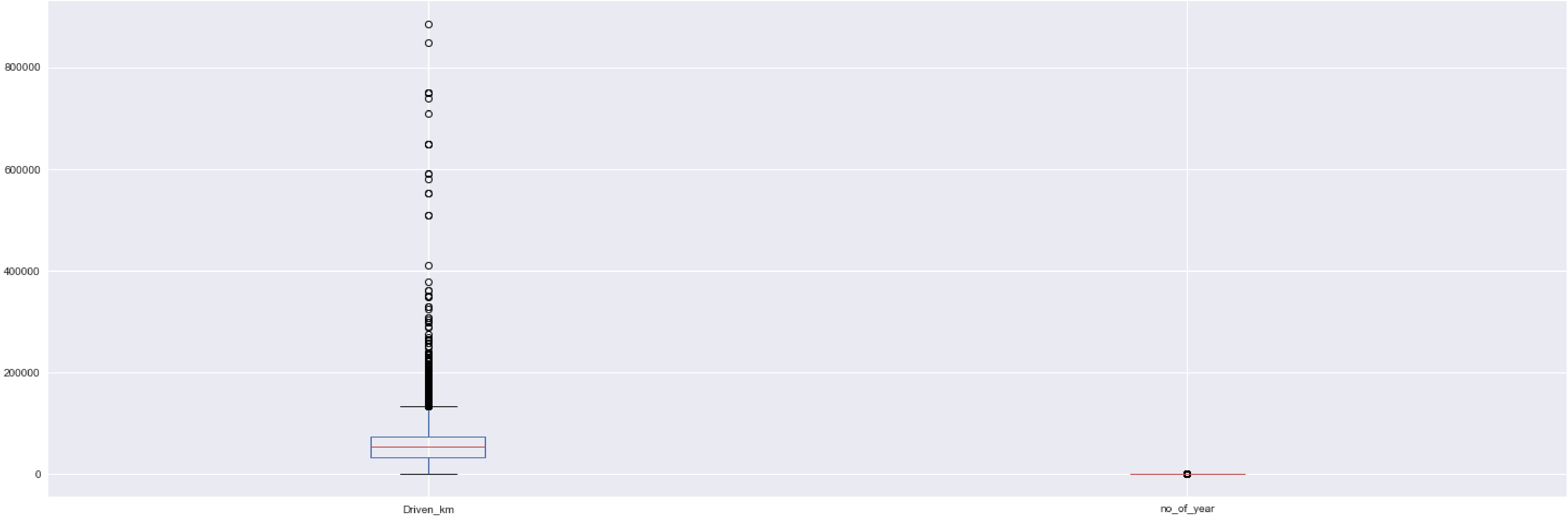


Checking the data distribution among all the columns.



Outliers Check:

In this dataset, we applied one hot encoding method to categorical features. so, we check outlies for nominal features i.e., Diven_Km, no_of_years and Selling Price. Only Driven_km and no_of_years is considered because Selling Price is our target variable.



We can see outliers in Driven km due to various kilometers driven for different cars. so, we proceed with further steps.

Checking Skewness:

Now here, we are going to use Power transform function to handle skewness in dataset

Before handling Skewness

Columns	Skewness	Columns	Skewness
Driven_km	4.840692	Fuel_type_LPG	25.448414
Selling_Price	5.401628	Fuel_type_Petrol	-0.143181
Brand_Maruti	0.975123	Transmission_Manual	-0.994505
Brand_Hyundai	1.550303	location_Bangalore	2.501900
Brand_Honda	2.806672	location_Chennai	3.626443
Brand_Toyota	3.635883	location_Delhi NCR	1.256471
Brand_Mahindra	4.266337	location_Gurgaon	4.217141
Brand_Ford	4.389812	location_Hyderabad	3.643476
Brand_Volkswagen	4.906065	location_Jaipur	4.957997
Brand_Mercedes-Benz	4.917924	location_Kolkata	4.700873
Brand_BMW	5.208302	location_Mumbai	3.164728
Brand_Renault	5.376202	location_New Delhi	2.640141
no_of_year	0.742327	location_Noida	2.526506
Fuel_type_Diesel	0.185166	location_Pune	3.354289
Fuel_type_Electric	68.658574		

After handling Skewness

Columns	Skewness	Columns	Skewness
Driven_km	0.122132	Fuel_type_LPG	25.448414
Brand_Maruti	0.975123	Fuel_type_Petrol	-0.143181
Brand_Hyundai	1.550303	Transmission_Manual	-0.994505
Brand_Honda	2.806672	location_Bangalore	2.501900
Brand_Toyota	3.635883	location_Chennai	3.626443
Brand_Mahindra	4.266337	location_Delhi NCR	1.256471
Brand_Ford	4.389812	location_Gurgaon	4.217141
Brand_Volkswagen	4.906065	location_Hyderabad	3.643476
Brand_Mercedes-Benz	4.917924	location_Jaipur	4.957997
Brand_BMW	5.208302	location_Kolkata	4.700873
Brand_Renault	5.376202	location_Mumbai	3.164728
no_of_year	-0.017734	location_New Delhi	2.640141
Fuel_type_Diesel	0.185166	location_Noida	2.526506
Fuel_type_Electric	68.658574	location_Pune	3.354289

Model Building and Evaluation

These are modelling approach made to build an model :

- Linear
- k-nearest neighbors (KNN)
- Random Forest
- Decision Tree
- XGBoost

Performance Metric

Model Building	R2 score	MAE	MSE	RMSE
Linear	53.63	4.21	56.70	7.53
KNeighbors	65.71	2.67	41.92	6.47
Random	77.45	1.97	27.56	5.25
Decision	61.21	1.98	47.38	6.88
XGBoost	72.02	2.35	34.22	5.85

According to performance metric, the random forest has higher R2 score, So this is our best model.

Comparison:

Performance Metric	Cross -Validation Score
53.63	-4.16
65.71	53.89
77.45	69.19
61.21	54.18
72.02	65.62

Comparing the performance model and cross-validation score the minimum difference is for xgboost. so finally, this is our best model.

Hyper Parameter Tuning

The Hyper parameter tuning is carried out for XGBoost Regressor model.

Because performance metric score is 72.02.

Hyper Parameter Tuning Performance

- **XGBoost Regressor:**

  R2 Score : 78.18

  Cross validation Score : 69.29

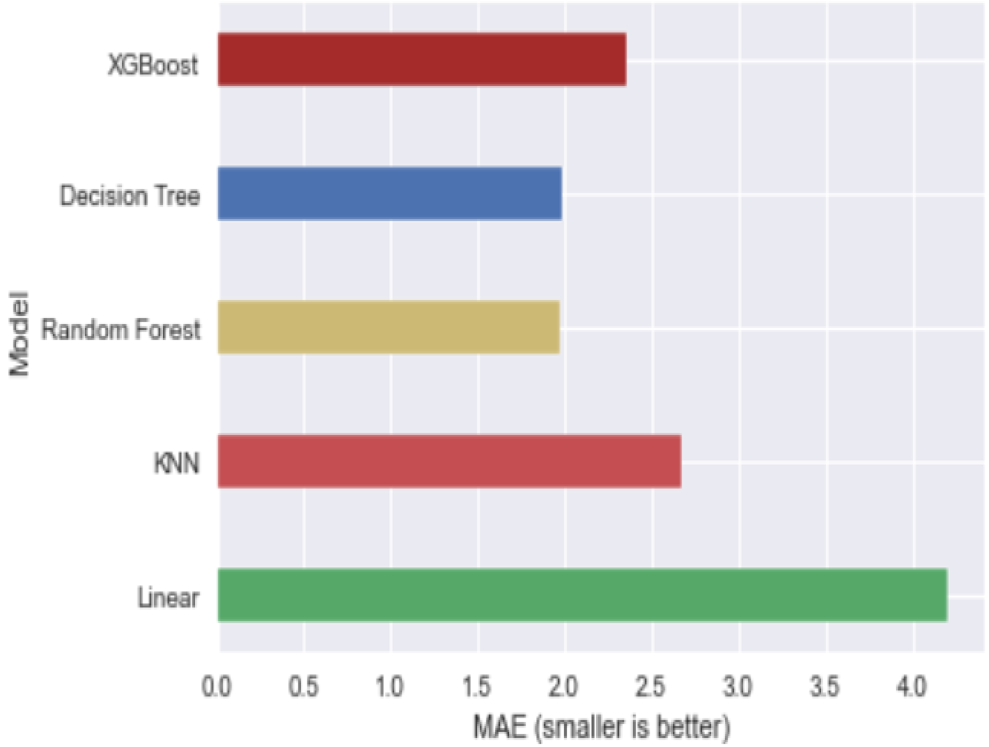
Best Model

Hyper parameter Tuning performance is carried out for XGBoost Regressor:

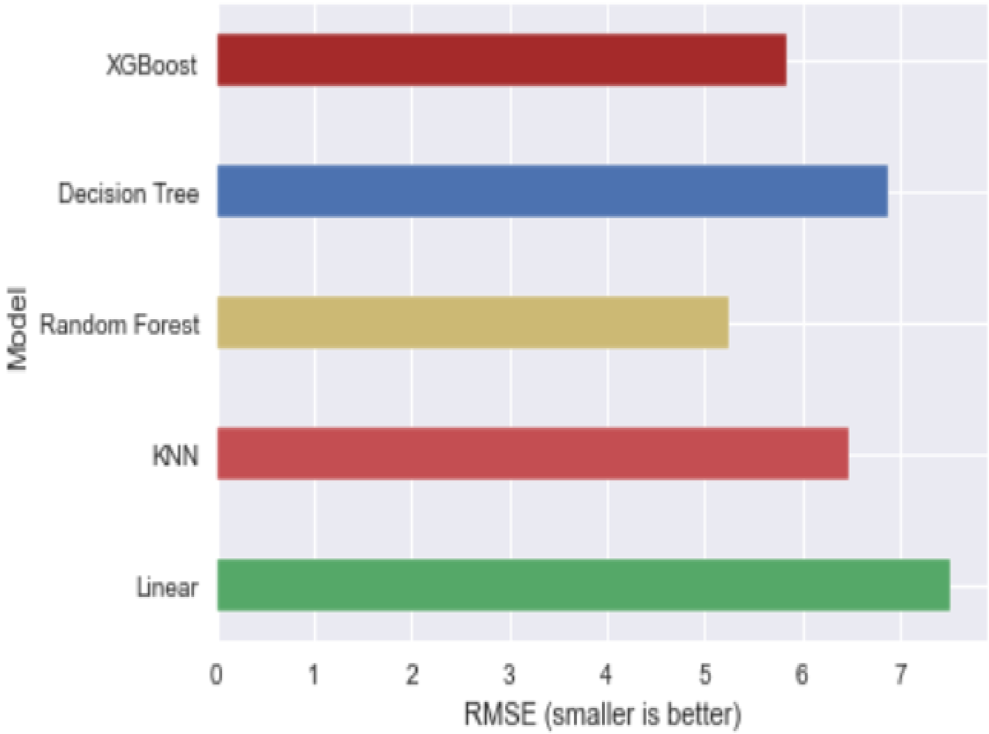
Hyper parameter Tuning i.e.,R2 score and Cross validation score = 78.18% and 69.29% respectively. Finally, XGBoost is best model for these dataset.

Performance Interpretation:

MAE (Mean Absolute Error)



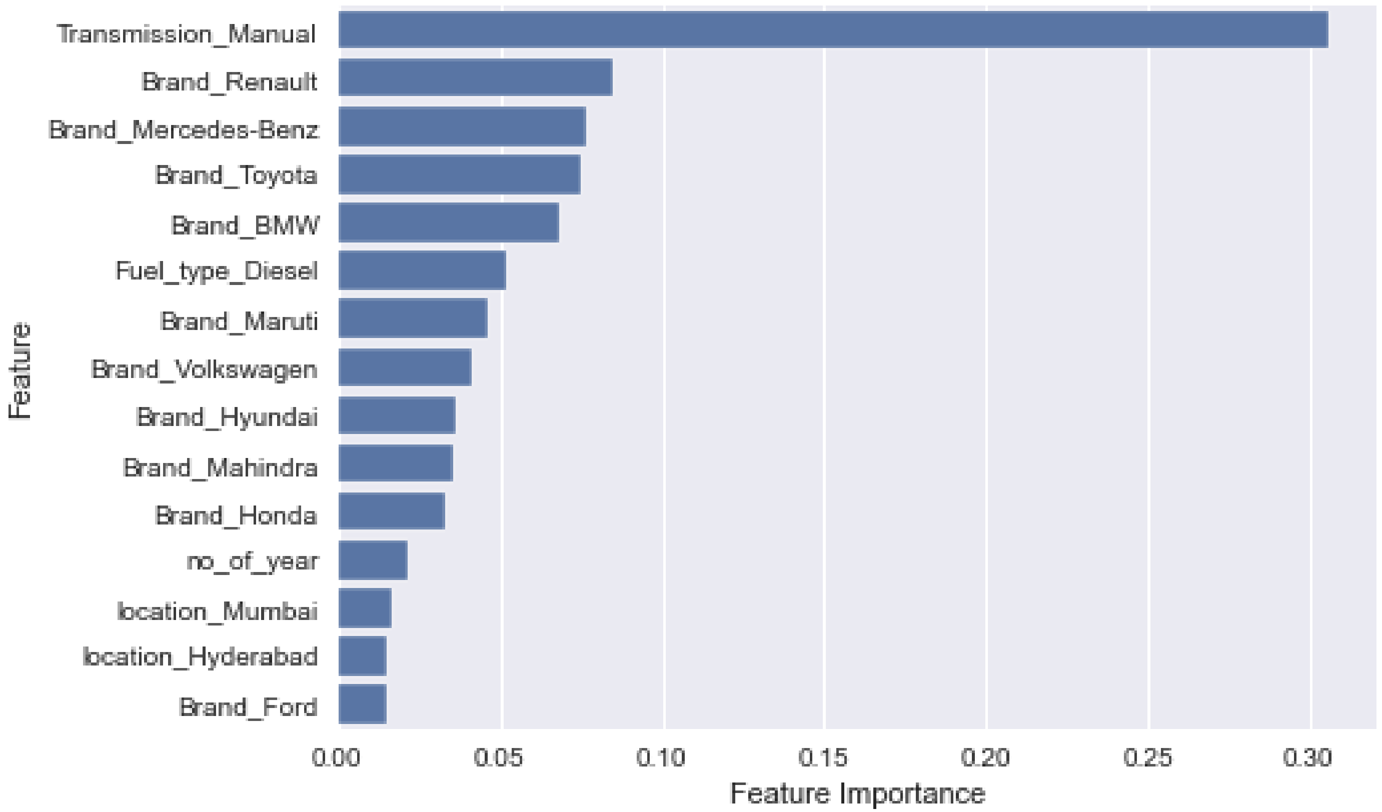
RMSE (Root Mean Squared Error)



Feature Importance’s:

Some of the models we used provide the ability to see the importance of each feature in the dataset after fitting the model. We will look at the feature importance’s provided by XGBoost models. We have 29 features in our data which is a big number, so we will take a look at the 15 most important features.

Feature Importance's:



Notice here in feature importance of XGBoost, the transmission manual feature plays a prominent role for target variable.

Conclusion:

- In this paper, we built several regression models to predict the selling price of cars by given some of the cars features. We evaluated and compared each model to determine the one with highest performance. We also looked at how some models rank the features according to their importance. In this paper, we followed the data science process starting with getting the data, then cleaning and pre-processing the data, followed by exploring the data and building models, then evaluating the results.

- As a recommendation, we advise to use this model (or a version of it trained with more recent data) by car market who want to get an idea about car price. The model can be used also with datasets that covered areas provided that they contain the same features. We also suggest that people take into consideration the features that were deemed as most important as seen in the previous section; this might help them estimate the car price is better.

```
conclusion=pd.DataFrame([loaded_model.predict(x_test)[:],pred_decision[:]],index=['Predicted','Original'])
conclusion
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Predicted	3.69999	5.001836	6.543528	3.405436	5.238691	6.748663	9.596337	5.54765	6.952202	5.292068	3.812086	4.146307	9.322211	2.998825	2.906869
Original	3.69999	5.001836	6.543528	3.405436	5.238691	6.748663	9.596337	5.54765	6.952202	5.292068	3.812086	4.146307	9.322211	2.998825	2.906869

2 rows × 3773 columns

Thank you