FLIP ROBO

CAR PRICE PREDICTION

Submitted by:

JAYASURYA E

Acknowledgment

In this paper, we investigate the application of supervised machine learning techniques to predict the price of used cars in India. The predictions are based on data collected from website of car market. Different techniques like linear regression, k-nearest neighbours, random forest, xgboost and decision trees have been used to make the predictions. The predictions are then evaluated and compared in order to find those which provide the best performances. A seemingly easy problem turned out to be indeed very difficult to resolve with high accuracy. All the five methods provided comparable performance. In the future, we intend to use more sophisticated algorithms.

Introduction

- **Business Problem Framing:**

Thousands of online car purchasing is going on every day. There are some questions every buyer asks himself like: What is the actual price that this car deserves? Am I purchasing a fair product? In this paper, a machine learning model is proposed to predict a car price based on data related to the car market (brand, manufacturing year, driven kilometres etc.). During the development and evaluation of our model, we will show the code used for each step followed by its output. This will facilitate the reproducibility of our work. In this study, Python programming language with a number of Python packages will be used.

- **Conceptual Background of the Domain Problem:**

The main objectives of this study are as follows:

- To scrap a dataset from website of car market.
- To apply data pre-processing and preparation techniques in order to obtain clean data.
- To build machine learning models able to car price prediction based on data is collected from the car market website.
- To analyse and compare model's performance in order to choose the best model.

- ## Literature Review

Machine learning is a form of artificial intelligence which compose available computers with the efficiency to be trained without being veraciously programmed. Machine learning interest on the extensions of computer programs which is capable enough to modify when unprotected to new-fangled data. Machine learning algorithms are broadly classified into three divisions, namely; Supervised learning, Unsupervised learning and Reinforcement learning. Supervised learning is a learning in which we teach or train the machine using data which is well labelled that means some data is already tagged with correct answer. After that, machine is provided with new set of examples so that supervised learning algorithm analyses the training data and produces a correct outcome from labelled data. Unsupervised learning is the training of machine using information that is neither classified nor labelled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data. Unlike, supervised learning, no teacher is provided that means no training will be given to the machine. Therefore, machine is restricted to find the hidden structure in unlabelled data by our-self.

Reinforcement learning is an area of Machine Learning Reinforcement. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behaviour or path it should take in a specific situation. Reinforcement learning differs from the supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of training dataset, it is bound to learn from its experience. Machine learning has many applications out of which one of the applications is prediction of car market. The machine learning models are:

Linear Regression:

To establish baseline performance with a linear classifier, we used Linear Regression to model the price targets, Y, as a linear function of the data, X

$$f(X) = w_0 + w_1 x_1 + \ \dots + w_m x_m + x_m$$

$$= \sum_{j=l:m}^{\infty} \left(w_j x_j \right)$$

Advantage: A linear model can include more than one predictor as long as the predictors are additive. the best fit line is the line with minimum error from all the points, it has high efficiency but sometimes this high efficiency created.

Disadvantage: Linear Regression Is Limited to Linear Relationships. Linear Regression Only Looks at the Mean of the Dependent Variable. Linear Regression Is Sensitive to Outliers. Data Must Be Independent

Random Forest Regression:

The Random Forest Regression (RFR) is an ensemble algorithm that combines multiple Regression Trees (RTs). Each RT is trained using a random subset of the features, and the output is the average of the individual RTs. The sum of squared errors for a tree T is:

Advantages: There is no need for feature normalization. Individual decision trees can be trained in parallel. Random forests are widely used. They reduce overfitting.

Disadvantages: They’re not easily interpretable. They’re not a state-of-the-art

$$S = \sum_{c€leaves(T)i€C} \sum (y_i - m_c)^2$$

$$\text{Where } m_c = \frac{1}{n_c} + \sum\nolimits_{i€C} y_i$$

Related work on car price prediction:

- Surprisingly, work on estimated the price of used cars is very recent but also very sparse. In her MSc thesis, Listiani showed that the regression model build using support vector machines (SVM) can estimate the residual price of leased cars with higher accuracy than simple multiple regression or multivariate regression. SVM is better able to deal with very high dimensional data (number of features used to predict the price) and can avoid both over-fitting and underfitting. In particular, she used a genetic algorithm to find the optimal parameters for SVM in less time. The only drawback of this study is that the improvement of SVM regression over simple regression was not expressed in simple measures like mean deviation or variance.

- In another university thesis, Richardson working on the hypothesis that car manufacturers are more willing to produce vehicles which do not depreciate rapidly. In particular, by using a multiple regression analysis, he showed that hybrid cars (cars which use two different power sources to propel the car, i.e., they have both an internal combustion engine and an electric motor) are more able to keep their value than traditional vehicles. This is likely due to more environmental concerns about the climate and because of its higher fuel efficiency. The importance of other factors like age, mileage, make and MPG (miles per gallon) were also considered in this study. He collected all his data from various websites.

- Wu et al. used neuro-fuzzy knowledge-based system to predict the price of used cars. Only three factors namely: the make of the car, the year in which it was manufactured and the engine style were considered in this study. The proposed system produced similar results as compared to simple regression methods. Car dealers in USA sell hundreds of thousands of cars every year through leasing. Most of these cars are returned at the end of the leasing period and must be resold. Selling these cars at the right price have major economic connotation for their success. In response to this, the ODAV (Optimal Distribution of Auction Vehicles) system was developed by Du et al. This system not only estimates a best price for reselling the cars but also provides advice on where to sell the car. Since the United States is a huge country, the location where the car is sold also has a non-trivial impact on the selling price of used cars. A k-nearest neighbour regression model was used for forecasting the price. Since this system was started in 2003, more than two million vehicles have been distributed via this system.

- ## Motivation of the problem undertaken:

Our client has decided to predict the car price using data analytics and machine learning technique.

Our client is looking at prospective car market is facing a problem 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. so, we required to build a model using Machine Learning in order to predict the actual price of the prospective car and decide the price for the car. For this car market wants to know:

• Which variables are important to predict the price of variable?

• How do these variables describe the selling price of the car?

Analytical Problem Framing

- ### Mathematical/ Analytical Modelling of the Problem Statistical Analysis

Once it comes time to analyse the data, there are an array of statistical model's analysts may choose to utilize. The most common techniques will fall into the following two groups:

- Supervised learning, including regression and classification models.
- Unsupervised learning, including clustering algorithms and association rules

Regression Model:

The regression models are used to examine relationships between variables. Regression models are often used to determine which independent variables hold the most influence over dependent variables information that can be leveraged to make essential decision.
The most traditional regression model is linear regression, decision tree regression, random forest regression, xgboost regression and knn-neighbours.

There are 4 main components of an analytics model, namely: 1) Data Component, 2) Algorithm Component, 3) Real World Component, and 4) Ethical Component.

- ## Data Preparation

In this study, we will scrap data from car market website. It's decided to use data analytics to know car price by their actual selling price values. The data is provided in the full_datasets.csv file.

- ## 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'.

- # Data Pre-processing:
  ## Unique Function for dataset:

There are some column features which is indicated with string values and unwanted brackets. so, we are replacing model feature into unique functions.

```
# 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)
```

Then, we move to see null value 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
```

There is no null value in dataset.

Now we are going to add features in dataset to which is required to check number of years car is used. i.e., current year (2021).

```
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

Then we move to see no of years. The car has been used by seller by adding a feature i.e., no_of_year.
The process carried to create a no of year is 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

Now, Let's see the no of years is created. so, we dropped both manufacturing year and current year from data frame.

```
df.drop(["Manufacturing_year","Current Year"],axis = 1, inplace = True)
df.head()
```

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

The new data frame is created. but still, we can see the categorical columns in data frame. then, proceed with encoding techniques to convert the string data to numerical one. Before going to encoding technique, we are dropping both Model and Variant features.

- ## Data Cleaning:

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.

This newly created data frame is used for machine learning algorithm. so, we create a new excel sheet to proceed with further steps.

```
df.head()
```

	Driven_km	Selling_Price	Brand_Maruti	Brand_Hyundai	Brand_Honda	Brand_Toyota	Brand_Mahindra	Brand_Ford	Brand_Volkswagen	Brand_Mercedes-Benz
0	45347	3.81	1	0	0	0	0	0	0	0
1	19627	4.70	1	0	0	0	0	0	0	0
2	57341	2.79	1	0	0	0	0	0	0	0
3	17116	4.72	1	0	0	0	0	0	0	0
4	14161	4.57	1	0	0	0	0	0	0	0

Then we move to see statistical information about the non-numerical columns in our dataset:

```
# 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

From the table above, we can see, for example, that the mean selling price for the car price prediction in our dataset is 8.96 with a standard deviation of 11.93. We can see also that the minimum is 0.3 and the maximum is 225 with a median of 5.47. Similarly, we can get a lot of information about our dataset variables from the table.

- **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.

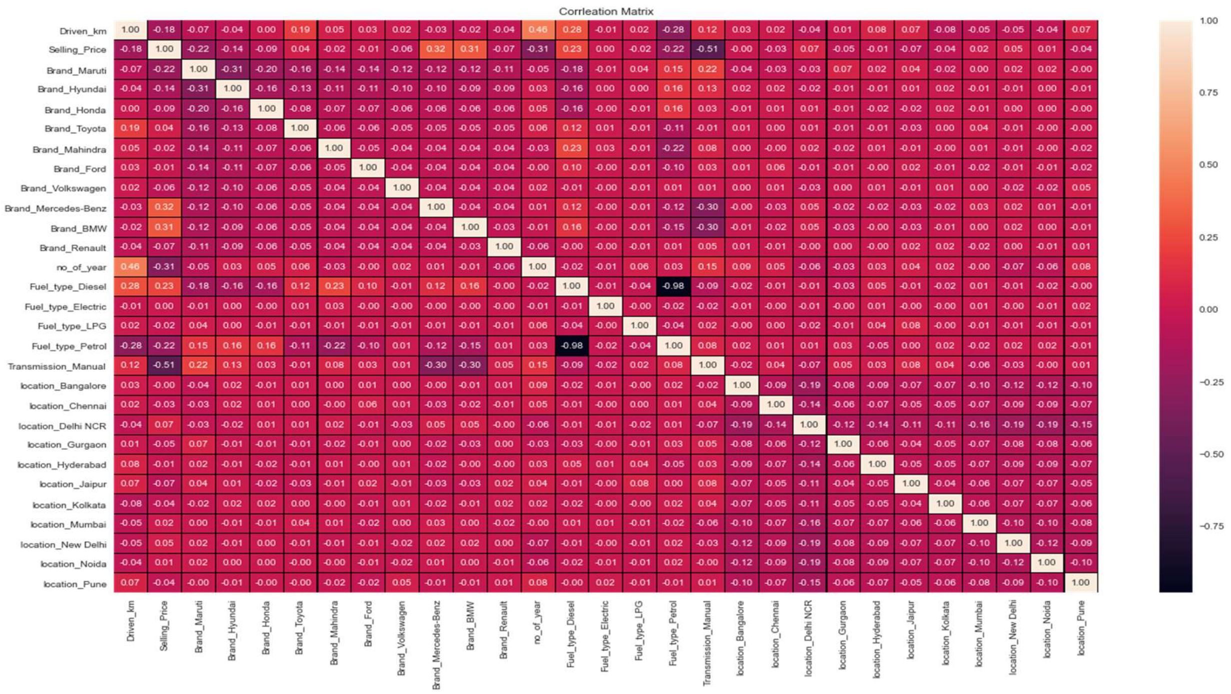
Pandas dataframe. corr() method is used for creating the correlation matrix. It is used to find the pairwise correlation of all columns in the data frame.

To create correlation matrix using pandas, these steps should be taken:

1. Obtain the data.
2. Create the DataFrame using Pandas.
3. Create correaltion matrix using Pandas.

```
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



Observations: We are unable to identify the correlation in above heatmap due to huge number of columns.

How correlation matrix is calculated?

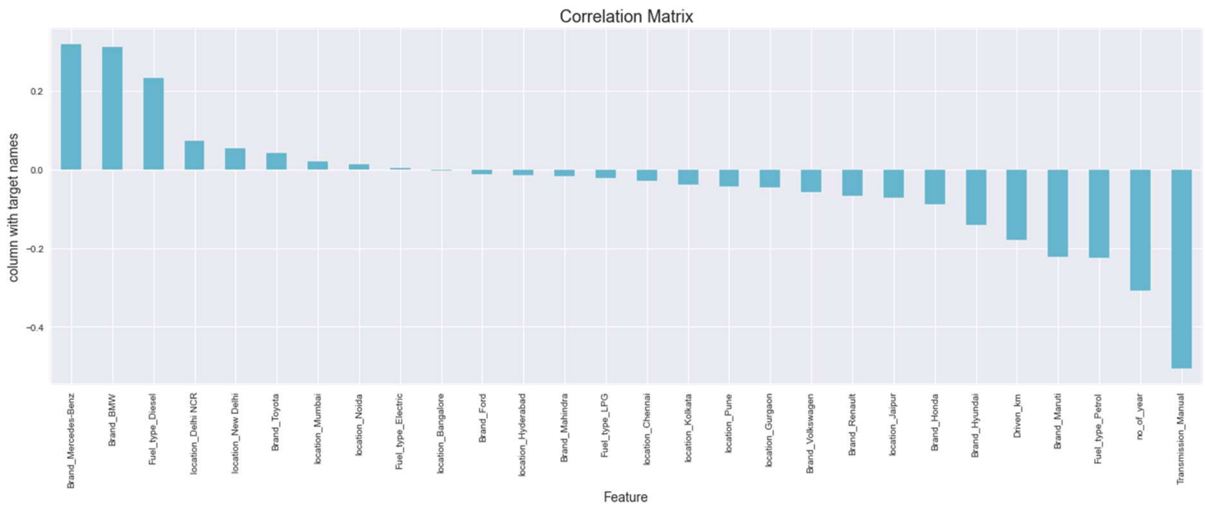
A correlation matrix is a table showing correlation coefficients between sets of variables. Each random variable (x) in the table is correlated with each of the other values in the table x. The diagonal of the table is always a set of ones, because the correlation between a variable and itself is always 1.

Columns	Correlation	Columns	Correlation
Selling_Price	1.000000	location_Chennai	-0.028935
Brand_Mercedes-Benz	0.318818	location_Kolkata	-0.039996
Brand_BMW	0.311755	location_Pune	-0.042884
Fuel_type_Diesel	0.233153	location_Gurgaon	-0.046609
location_Delhi NCR	0.073410	Brand_Volkswagen	-0.056896
location_New Delhi	0.053115	Brand_Renault	-0.067144
Brand_Toyota	0.042553	location_Jaipur	-0.073377

location_Mumbai	0.019571	Brand_Honda	-0.088945
location_Noida	0.014484	Brand_Hyundai	-0.140579
Fuel_type_Electric	0.004589	Driven_km	-0.179689
location_Bangalore	-0.003963	Brand_Maruti	-0.221589
Brand_Ford	-0.012836	Fuel_type_Petrol	-0.223833
location_Hyderabad	-0.014602	no_of_year	-0.308634
Brand_Mahindra	-0.016334	Transmission_Manual	-0.505879
Fuel_type_LPG	-0.022985		

Now we can clearly identify the correlation of independent variables with the target variables "Selling_Price". There are some variables who has less than 0.01 correlation value (very week relationship.)

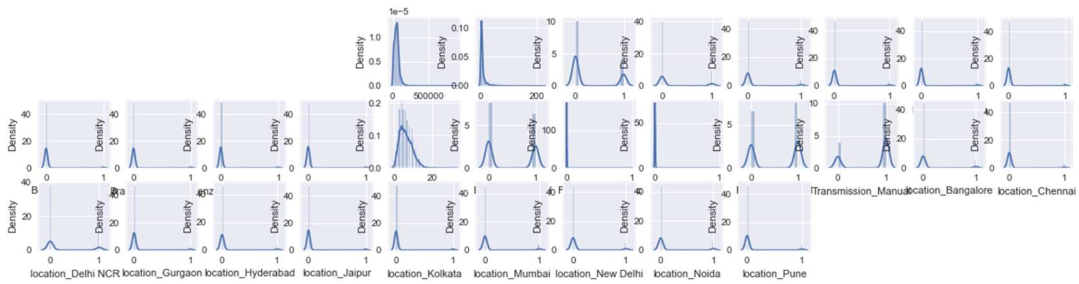
Checking the columns which are positively and negative correlated with the target columns:



Outcome of Correlation:

The Columns of the dataset is Correlated in both Positively and Negatively with target columns.
The Positive and negative correlation values is shown in both numbers and graph.
Max correlation: Brand_Mercedes-Benz
Min correlation: Transmission_Manual

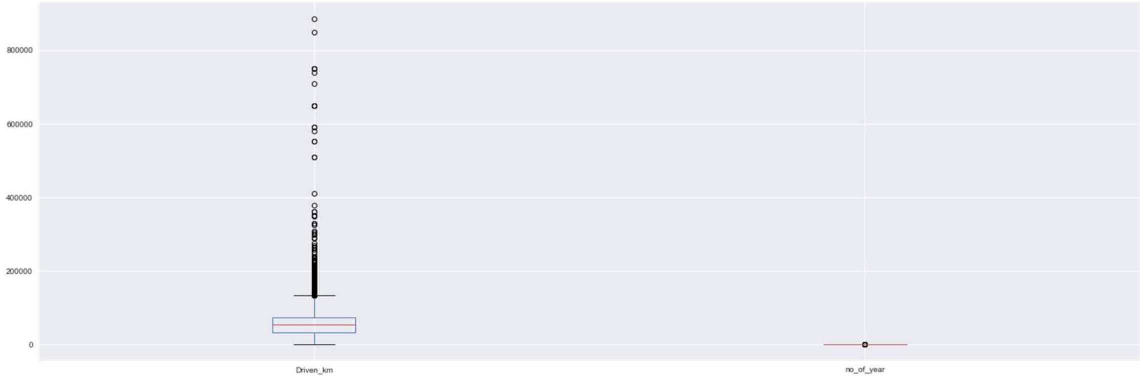
Let's check the data distribution among all the columns.



We can see skewness in data for the multiple columns, will handle the skewness in further steps.

- ## 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 further steps:

- ## Skewness:

Skewness is a measure of symmetry in a distribution. Actually, it's more correct to describe measure of lack of symmetry. A standard normal distribution is perfectly symmetrical and has zero skew. Therefore, we need a way to calculate how much the distribution is skewed.

- ## Checking 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		

To handle skewness of the data using different types of functions:

1. Log Transform

2. Square Root Transform
3. Box-Cox Transform
4. Power transform

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

Then, splitting the independent and target variable in x and y.

In statistics, a power transform is a family of functions applied to create a monotonic transformation of data using power functions. It is a data transformation technique used to stabilize variance, make the data more normal distribution-like, improve the validity of measures of association (such as the Pearson correlation between variables), and for other data stabilization procedures.

```
from sklearn.preprocessing import power_transform
df_new=power_transform(x)
df_new=pd.DataFrame(df_new,columns=x.columns)
```

After performing such statistics, the skewness is removed in dataset as shown below:

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

But still you can see skewness in dataset its all categorical variable not considered for skewness. Skewness is mainly depended on nominal features, so all nominal features skewness has been handled well using power transform function.

- ## Hardware and Software Requirements and Tools Used

PYTHON Jupyter Notebook:

Key Features:

An open-source solution that has simple coding processes and syntax so it's fairly easy to learn Integration with other languages such as C/C++, Java, PHP, C#, etc.
Advanced analysis processes through machine learning and text mining.

Python is extremely accessible to code in comparison to other popular languages such as Java, and its syntax is relatively easy to learn making this tool popular among users that look for an open-source solution and simple coding processes. In data analysis, Python is used for data crawling, cleaning, modelling, and constructing analysis algorithms based on business scenarios. One of the best features is actually its user-friendliness: programmers don't need to remember the architecture of the system nor handle the memory – Python is considered a high-level language that is not subject to the computer's local processor.

Libraries and Packages used:

Matplotlib:

Matplotlib is a Python library that uses Python Script to write 2-dimensional graphs and plots. Often mathematical or scientific applications require more than single axes in a representation. This library helps us to build multiple plots at a time. You can, however, use Matplotlib to manipulate different characteristics of figures as well.

The task carried out is visualization of dataset i.e., heatmap display distribution for correlation matrix and null values, boxplot distribution for checking outliers, scatter plot distribution for modelling approach, subplot distribution for analysis and comparison, feature importance and common importance features.

Numpy:

Numpy is a popular array – processing package of Python. It provides good support for different dimensional array objects as well as for matrices. Numpy is not only confined to providing arrays only, but it also provides a variety of tools to manage these arrays. It is fast, efficient, and really good for managing matrices and arrays.

The Numpy is used to managing matrices i.e., MAE, MSE and RMSE and arrays i.e., described the values of train test dataset.

Pandas:

Pandas is a python software package. It is a must to learn for data-science and dedicatedly written for Python language. It is a fast, demonstrative, and adjustable platform that offers intuitive data-structures. You can easily manipulate any type of data such as – structured or time-series data with this amazing package.

The Pandas is used to execute a Data frame i.e., test set.csv, train set.csv, skewness, co-efficient, predicted values of model approach, conclusion.

Scikit Learn:

Scikit learn is a simple and useful python machine learning library. It is written in python, cython, C, and C++. However, most of it is written in the Python programming language. It is a free machine learning library. It is a flexible python package that can work in complete harmony with other python libraries and packages such as Numpy and Scipy.

Scikit learn library is used to import a pre-processing function i.e., power transform, ordinal encoder, linear, random forest, decision tree, xgboost, k-nearest neighbours, r2 score, mean absolute error, mean squared error, train test split, randomized search cv and ensemble technique.

Models Development and Evaluation

In this section, we choose the type of machine learning prediction that is suitable to our problem. We want to determine if this is a regression problem or a classification problem. In this project, we want to predict the selling price of car with information about it. The selling price we want to predict is a continuous value; it can be any real number. This can be seen by looking at the target variable in our dataset is selling price:

That means that the prediction type that is appropriate to our problem is regression. Now, we move to choose the modelling techniques we want to use. There are a lot of techniques available for regression problems but we are going to use Linear Regression, Decision Trees, Random Forest, XGBoost, k-nearest neighbors (KNN) etc. In this project, we will test many modelling techniques, and then choose the technique(s) that yield the best results. The techniques that we will try are:

1. Linear Regression

This technique models the relationship between the target variable and the independent variables (predictors). It fits a linear model with coefficients to the data in order to minimize the residual sum of squares between the target variable in the dataset, and the predicted values by the linear approximation.

2. Random Forest

Bagging is an ensemble method where many base models are used with a randomized subset of data to reduce the variance of the base model.

3. Decision Trees

For this technique, the goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Each one of these techniques has many algorithmic implementations. We will choose algorithm(s) for each of these techniques in the next section.

4. XGBoost

It’s a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

5. k-nearest neighbors (KNN)

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems.

Model Building and Evaluation

In this part, we will build our prediction model: we will choose algorithms for each of the techniques we mentioned in the previous section. After we build the model, we will evaluate its performance and results.

Feature Scaling:

In order to make all algorithms work properly with our data, A way to normalize the input features/variables is the Min-Max scaler. By doing so, all features will be transformed into the range [0,1] meaning that the minimum and maximum value of a feature/variable is going to be 0 and 1, respectively.

Importing libraries for metrics and model building:

```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from sklearn.neighbors import KNeighborsRegressor
```

Modelling Approach:

For each one of the techniques mentioned in the previous section (Linear Regression, Random Forest Regression, Decision Tree Regression, XGBoost, k-nearest neighbors (KNN) etc etc.), we will follow these steps to build a model:

- Choose an algorithm that implements the corresponding technique
- Search for an effective parameter combination for the chosen algorithm
- Create a model using the found parameters
- Train (fit) the model on the training dataset
- Test the model on the test dataset and get the results

Regression Method:

- Using Scikit-Learn, we can build a model.

```
maxR2=0
BestRS=0
for i in range(1,200):
    x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=.20,random_state=i)
    LR = LinearRegression()
    LR.fit(x_train, y_train)
    predrf = LR.predict(x_test)
    r2=r2_score(y_test, predrf)
    if r2>maxR2:
        maxR2=r2
        BestRS=i
print("Best R2 is " ,maxR2," on Random_state ",BestRS)
```

```
Best R2 is  0.5362843236245256  on Random_state  27
```

Splitting the Dataset:

As usual for supervised machine learning problems, we need a training dataset to train our model and a test dataset to evaluate the model. So, we will split our dataset randomly into two parts, one for training and the other for testing. For that, we will use another function from Scikit-Learn called train_test_split():

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size = 0.20,random_state = BestRS)
```

Performance Metric:

For evaluating the performance of our models, we will use R2 score, mean absolute error (MAE) and mean squared error (MSE). If the predicted value of the element, and the corresponding true value, then for all the elements, RMSE is calculated as:

```
def eval(x):
    mod =x
    mod.fit(x_train, y_train)
    predict_test = mod.predict(x_test)
    print("R2 score is ", r2_score(y_test, predict_test)*100)
    print("Mean Absolute error is", mean_absolute_error(y_test,predict_test))
    print("Mean squared error is", mean_squared_error(y_test,predict_test))
    print("Root mean squared error is", np.sqrt(mean_squared_error(y_test,predict_test)))
```

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

Comparing the Performance metric and Cross validation Score:

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

Here we have handled the problem of the overfitting and the underfitting by checking the R2 score.
Comparing the performance metric and cross-validation score which has minimum difference is xgboost. so finally, this is our best model.

Hyper Parameter Tuning:

Hyperparameters are crucial as they control the overall behaviour of a machine learning model. The ultimate goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results.

Now, we are going to perform hyperparameter tuning for this model to get better result.

Importing RandomizedSearchCV

```
from sklearn.model_selection import RandomizedSearchCV
```

Hyperparameter Tuning for XGBoost Regressor:

Firstly, we will use RandomizedSearchCV() to search for the best model parameters in a parameter space provided by us i.e., n estimator, max depth, verbosity, min child weight and booster.

```
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Verbose is a general programming term for produce lots of logging output.
verbosity = [3]
#minimum sum of instance weight (hessian) needed in a child
min_child_weight  = [1, 2, 5, 10]
# The booster parameter sets the type of learner.
booster = ['gbtree', 'gblinear']
```

```
from xgboost import XGBRegressor
parameters={'n_estimators': n_estimators,
               'max_depth': max_depth,
               'verbosity': verbosity,
               'min_child_weight': min_child_weight,
               'booster': booster}
xgb=XGBRegressor()
clf=RandomizedSearchCV(xgb,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)
```

We defined the parameter space above using reasonable values for chosen parameters.

```
xgb=XGBRegressor(n_estimators = 400, min_child_weight = 2, max_depth = 20, booster = 'gbtre
                         verbosity = 3)
xgb.fit(x_train,y_train)
xgb.score(x_train,y_train)
pred_decision=xgb.predict(x_test)
xgbs=r2_score(y_test,pred_decision)
print('R2 Score:',xgbs*100)
```

```
xgbscore=cross_val_score(xgb,x,y,cv=5)
xgbc=xgbscore.mean()
print('Cross Val Score:',xgbc*100)
```

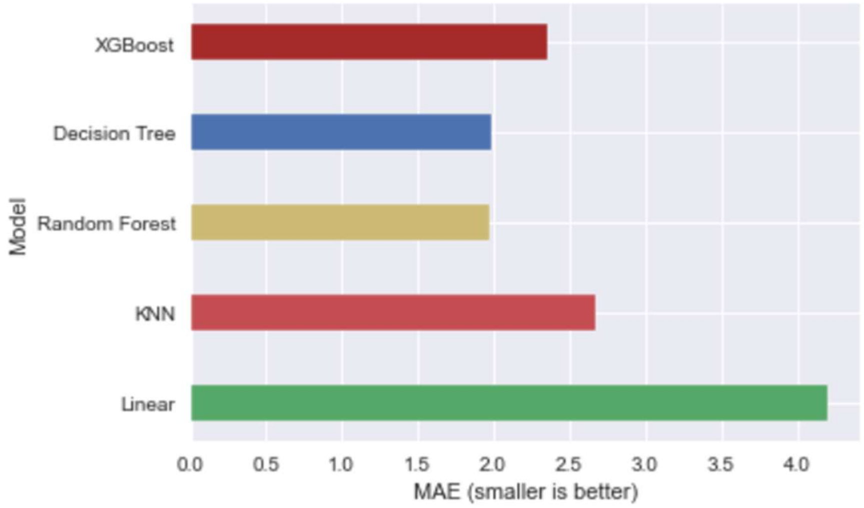
R2 Score: 78.18185367083443
Cross Val Score: 69.28481714694395

We defined the performance model score and cross validation score of hyperparameter tuning for xgboost using chosen parameters. We are getting model accuracy and cross validation has 78.18% & 69.28% respectively. We consider xgboost regressor is our best model for these datasets.

Performance Interpretation:

MAE (Mean Absolute Error):

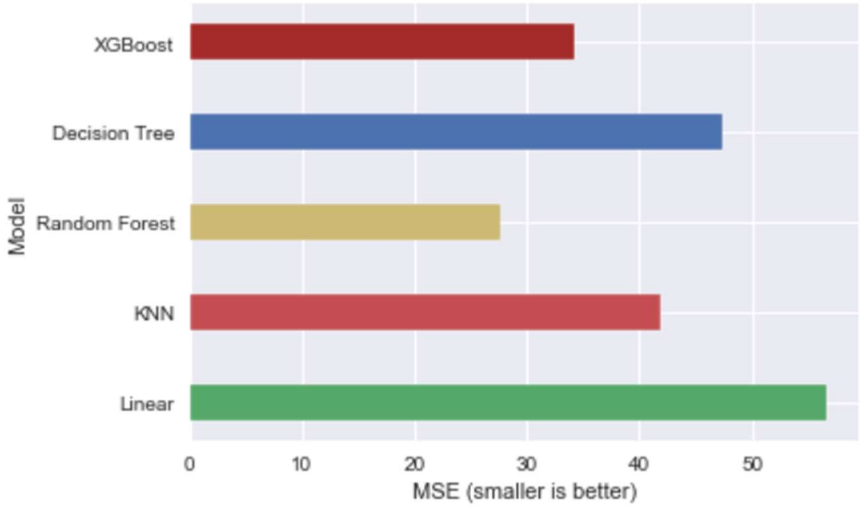
```
x = ['Linear','KNN','Random Forest', 'Decision Tree', 'XGBoost']
y = [4.21, 2.67 , 1.97, 1.99, 2.35]
colors = ["g", "r", "y", "b", "brown"]
fig, js = plt.subplots()
plt.barh(y=range(len(x)), tick_label=x, width=y, height=0.4, color=colors);
js.set(xlabel="MAE (smaller is better)", ylabel="Model");
```



By looking at the table and the graph, we can see that Random Forest has the smallest MAE, 1.97 followed by 1.99 with a little larger error of 0.02. After that, XGBoost come with error of 2.35. At last, the K-Nearest Neighbors and linear comes with a similar error: 2.67 and 4.21 respectively.
So, in our experiment, the best model is Random Forest and the worst model is Linear. We can see that the difference in MAE between the best model and the worst model is significant; the best model has least error of the worst model.

MSE (Mean Squared Error):

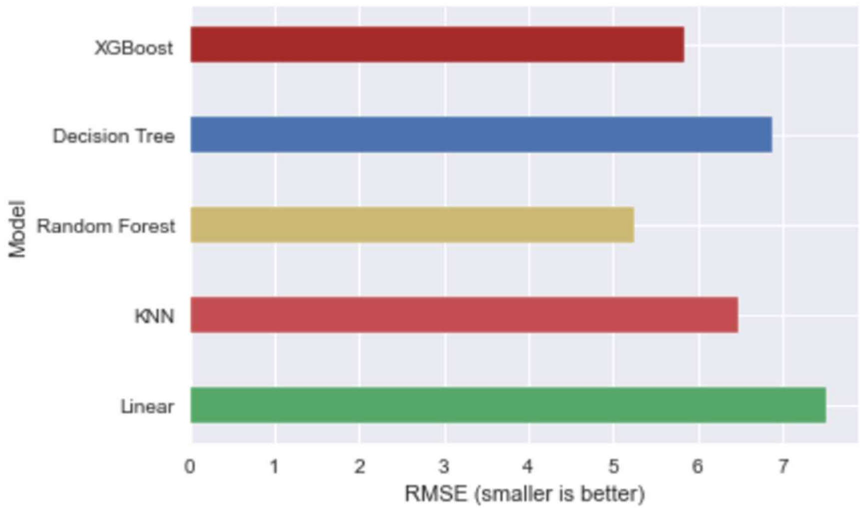
```
x = ['Linear','KNN','Random Forest', 'Decision Tree', 'XGBoost']
y = [56.71, 41.92, 27.57, 47.38, 34.22]
colors = ["g", "r", "y", "b", "brown"]
fig, js = plt.subplots()
plt.barh(y=range(len(x)), tick_label=x, width=y, height=0.4, color=colors);
js.set(xlabel="MSE (smaller is better)", ylabel="Model");
```



By looking at the table and the graph, we can see that Random Forest has the smallest MSE, 27.57. After that, XGBoost and K-Nearest Neighbors comes with similar errors: 34.22 and 41.92 respectively. At last, the Decision Tree and Linear comes with a similar error: 47.38 and 56.71 respectively.
So, in our experiment, the best model is Random Forest and the worst model is Linear. We can see that the difference in MSE between the best model and the worst model is significant; the best model has least error of the worst model.

RMSE (Root Mean Squared Error)

```
x = ['Linear','KNN','Random Forest', 'Decision Tree', 'XGBoost']
y = [7.53, 6.47, 5.25, 6.88, 5.85]
colors = ["g", "r", "y", "b", "brown"]
fig, js = plt.subplots()
plt.barh(y=range(len(x)), tick_label=x, width=y, height=0.4, color=colors);
js.set(xlabel="RMSE (smaller is better)", ylabel="Model");
```



By looking at the table and the graph, we can see that Random Forest has the smallest RMSE of 5.25. After that, XGBoost and K-Nearest Neighbors comes with similar errors: 5.85 and

6.47 respectively. At last, the Decision Tree and linear comes with a similar error: 6.88 and 7.53 respectively.
So, in our experiment, the best model is Random Forest and the worst model is Linear. We can see that the difference in RMSE between the best model and the worst model is significant; the best model has almost least error of the worst model.
We know that our best model is Random Forest but when compared with cross validation score it has overfitting and cross fitting problem. After compared with R2 score, minimum difference is for XGBoost. so finally, I chosen this is our best model for choice then the worst model is Linear.

Feature Importance's:

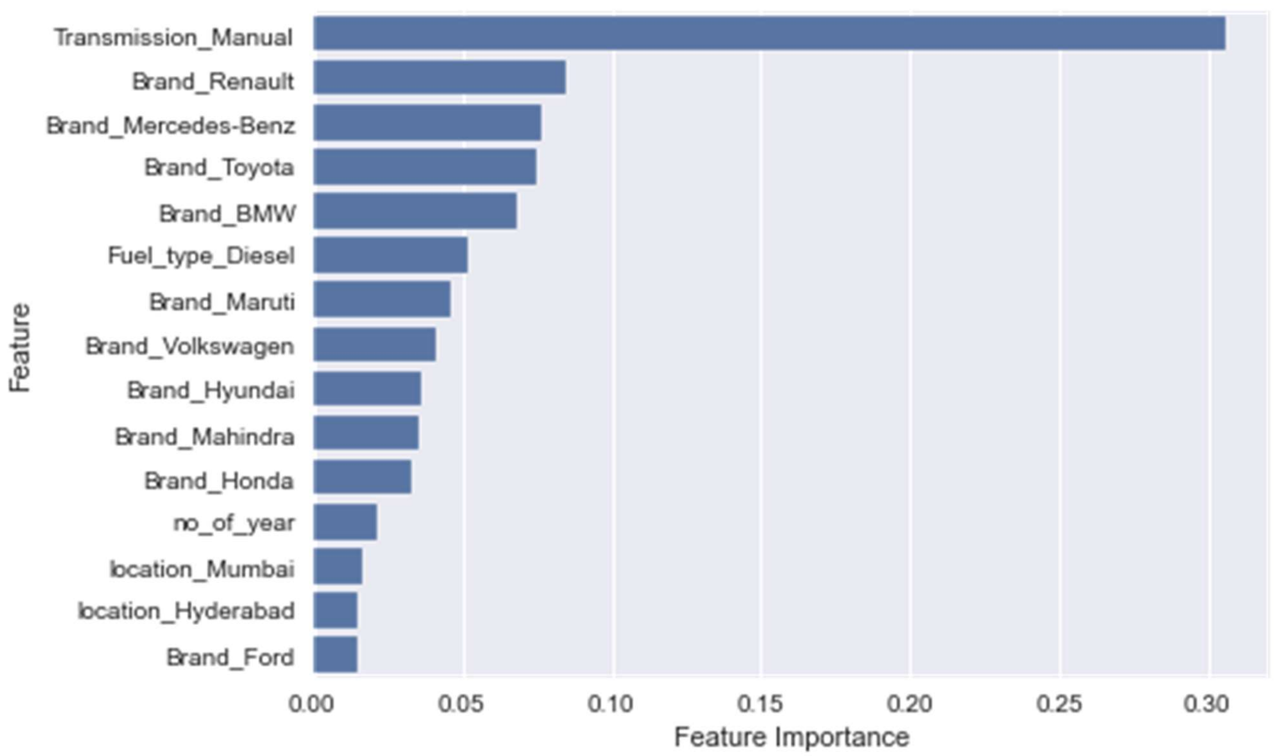
Some of the models we used provide the ability to see the importance of each feature in the dataset after fitting the model. We will look at the feature importance's provided by XGBoost models. We have 29 features in our data which is a larger number, so we will take a look at the top 15 most important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

XGBoost

Now, let's see the most important features as for XGBoost model:

```
xgb_feature_importances = xgb.feature_importances_
xgb_feature_importances = pd.Series(xgb_feature_importances,
                                    index=x_train.columns.values).sort_values(ascending=False).head(15)

fig, js = plt.subplots(figsize=(7,5))
sns.barplot(x=xgb_feature_importances, y=xgb_feature_importances.index, color="b");
plt.xlabel('Feature Importance');
plt.ylabel('Feature');
```



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 from 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.

Learning Outcomes of the Study in respect of Data Science:

- To scrap a dataset from car market websites.
- Obtain, clean/process, and transform data.
- Analyze and interpret data using an ethically responsible approach.
- Use appropriate models of analysis, assess the quality of input, derive insight from results, and investigate potential issues.
- Apply computing theory, languages, and algorithms, as well as mathematical and statistical models, and the principles of optimization to appropriately formulate and use data analyses
- Formulate and use appropriate models of data analysis to solve hidden solutions to business-related challenges

Limitations of this work and Scope for Future Work:

There are many things that can be tried to improve the models' predictions. We can create and add more variables, try different models with different subset of features and/or rows, etc. Some of the ideas are listed below:

- Combine the applicants with 1,2,3 or more dependents and make a new feature as discussed in the EDA part.
- Make independent vs independent variable visualizations to discover some more patterns.
- Arrive at the EMI using a better formula which may include interest rates as well.
- Try neural network using TensorFlow or PyTorch.
- Try developing website by using html code and pycharm for deployment purpose.

****