Car Price Prediction

Submitted by:

HARINATH MALLELA

ACKNOWLEDGMENT

A unique opportunity like this comes very rarely. It is indeed a pleasure for me to have worked on this project. The satisfaction that accompanies the successful completion of this project is incomplete without the mention of the people whose guidance has made it possible for me to complete this project.I am grateful to my internship company **Flip Robo Technologies** with its ideals and inspiration for providing me with facilities that has made this project a success.

I am grateful to **Vaishali Singh** for providing the opportunity to work as a intern in **Flip Robo Technologies**

I like to acknowledge the effort put in by our SME **Sajid Choudhary** in helping me understand the relevant concepts related to Data preprocessing and providing guidance when necessary.

HARINATH MALLELA

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. We have to make car price valuation model.

Conceptual Background of the Domain Problem

The price of a new car in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But, due to the increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features.

Review of Literature

Existing System includes a process where a seller decides a price randomly and buyer has no idea about the car and its value in the present day scenario. In fact, seller also has no idea about the car's existing value or the price he should be selling the car at. To overcome this problem we have developed a model which will be highly effective. Regression Algorithms are used because they provide us with continuous value as an output and not a categorized value. Because of which it will be possible to predict the actual price a car rather than the price range of a car. User Interface has also been

developed which acquires input from any user and displays the Price of a car according to user’s inputs.

Motivation for the Problem Undertaken

As a Data scientist, we are required to apply some data science techniques for the price of cars with the available independent variables. That should help the management to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain price levels.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

Mathematical Summary:

Dimensions of Dataset: There are 9 columns and 5041 rows in this dataset

Null Values: There are no null values in this dataset

Skewness: Skewness is present in only in the Target

column.

Statistical Summary:

Standard deviation is very normal in most of the columns.

There is not much difference between mean and 50^{th} percentile, which means data is skewed

There is not much difference between 75^{th} percentile and max, which means there are outliers

Data Sources and their formats

Brand, Model, Variant, Manufacturing_Year, Driven_Kilometers, Fuel, Number_Of_Owners, Location, Price.

Data Pre-processing Done

Feature Extraction:

At first we had only 9 columns in which Name column consists of Brand, Model and Year information. After extracting the required fields using Regular expression we got 9 columns.

1. Dropping Unnamed: 0 and Variante columns after extraction

```
# droping the Unnamed: 0 column
df.drop('Unnamed: 0',axis=1,inplace=True)
```

```
# droping the Variate column because we have high unique values in this
df.drop('Variant',axis=1,inplace=True)
```

2. Replacing “Lakh” from Price column and converting it to float And Replacing “,” symbol from Driven_Kilometers column and converting it to Int

```
#Lets remove Lakh in price and Kms, "," in Drive_Kilometers
df['Price'] = df['Price'].str.replace('Lakh', '')
df['Driven_Kilometers'] = df['Driven_Kilometers'].str.replace('Kms', '')
df['Driven_Kilometers'] = df['Driven_Kilometers'].str.replace(',', '')
```

```
#Lets change Driven_Kilometers dtype from object to int
df["Driven_Kilometers"] = df.Driven_Kilometers.astype(int)
```

```
#Lets change Price dtype from object to Float
df["Price"] = df.Price.astype(float)
```

1. Final dataframe

```
df
```

	Brand	Model	Manufacturing_Year	Driven_Kilometers	Fuel	Number_Of_Owners	Location	Price
0	Honda	City	2016	46948	Petrol	First	Chennai	745000.0
1	Toyota	Corolla Altis	2014	72497	Petrol	Second	Bangalore	825000.0
2	Honda	Brio	2015	21873	Petrol	First	Bangalore	489000.0
3	Hyundai	Grand i10	2019	17916	Petrol	First	Kolkata	500000.0
4	Hyundai	Creta	2015	27598	Petrol	First	Bangalore	948000.0
...	...	...	...	...	...	...	...	...
5036	Hyundai	Eon	2016	26000	Petrol	First	Chennai	324000.0
5037	Volkswagen	Vento	2015	65000	Diesel	First	Chennai	724000.0
5038	Maruti Suzuki	Swift	2012	90000	Diesel	Second	Chennai	424000.0
5039	Mercedes-Benz	GLC	2017	58500	Diesel	First	Bangalore	4600000.0
5040	Jaguar	F-Pace	2018	16000	Diesel	First	Mumbai	5100000.0

1. Scaling the Input data

Scaling the Data

```
#lets split our x and y column before scaling
#x and y spitting
Y = df_new['Price']
X = df_new.drop('Price',axis = 1)
```

```
#Lets scale the data using standard scaler
scaler = StandardScaler()
scaled_X = pd.DataFrame(scaler.fit_transform(X),columns= X.columns)
scaled_X.head()
```

	Driven_Kilometers	Number_Of_Owners	Location_Agra	Location_Ahmedabad	Location_Akola	Location_Ambala Cantt	Location_Aurangabad	Location_Bangalore
0	-0.190069	-0.494084	-0.067702	-0.242968	-0.042291	-0.042291	-0.066207	-0.378949
1	0.506160	1.800415	-0.067702	-0.242968	-0.042291	-0.042291	-0.066207	2.638876
2	-0.873380	-0.494084	-0.067702	-0.242968	-0.042291	-0.042291	-0.066207	2.638876
3	-0.981211	-0.494084	-0.067702	-0.242968	-0.042291	-0.042291	-0.066207	-0.378949
4	-0.717370	-0.494084	-0.067702	-0.242968	-0.042291	-0.042291	-0.066207	2.638876

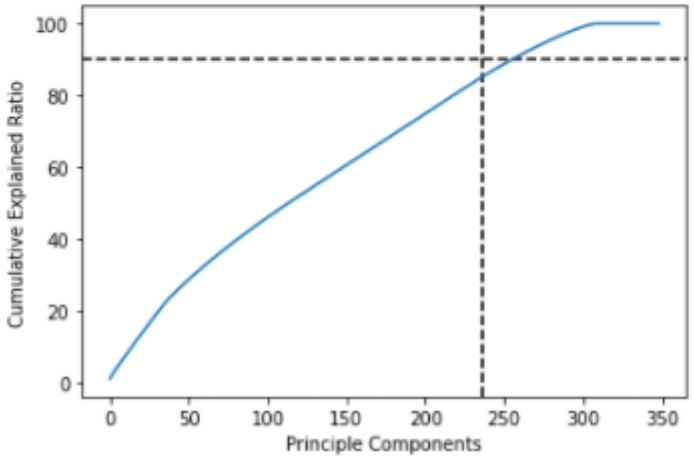
5 rows × 349 columns

2. PCA (principle component Analysis)

```
#lets plot the graph for graphical understanding
plt.ylabel('Cumulative Explained Ratio')
plt.xlabel('Principle Components')

plt.axvline(x = s, color = 'k', linestyle = '--')
plt.axhline(y = 90, color = 'k', linestyle = '--')

plt.plot(cum_score)
plt.show()
```



Data Inputs- Logic- Output Relationships
We don’t see much correlation between kilometre driven and Price.
we can see that diesel variant has more price compare to petrol variant
we can see that 1st Owner has more price compare to 2nd,3rd,4th,5th variant
we can see that if the car is recently purchased than the price is high

Hardware and Software Requirements and Tools Used

Machine: Can use a laptop/desktop.

Operating system: Windows 8 or 10, Mac OS X 10.9 Mavericks or Higher
RAM & Processor: 8 GB+ RAM, i5 5th Generation 2.2 Ghz or equivalent/higher
Tools Used: Jupyter Notebook, Microsoft Excel.

Libraries Used :
Sklearn,Seaborn,Matplotib,Pandas,Numpy,Imbalanced- learn,warnings

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

I started by extracting all the random cars data from CarTrade website from different locations.

After extracting the required information, I had column called Name which consist of three different inputs which includes Manufacturing_Year, Brand and Model.

I used Regular Expression to extract the Year, Brand and Model information from the Name column.

I then dropped the Name column after extracting the required information.

Converted Object data type like price and KM driven to FLOAT AND INT after replacing Symbols.

Testing of Identified Approaches

(Algorithms) Below are the algorithms used for the

training and testing. Lr = LinearRegression()

dtc = DecisionTreeRegressor()

knn =

KNeighborsRegressor(n_neighbors=5)

rf = RandomForestRegressor()

ada = AdaBoostRegressor()

Run and Evaluate selected models

These are the algorithms used along with the snapshot of their code.

1. Linear Regression

```
lrg = LinearRegression()
lrg.fit(x_train,y_train)
pr =lrg.predict(x_test)
print("r2_score of linear refression is :", r2_score(y_test,pr))
print('Error :')
print('mean absolute error :',mean_absolute_error(y_test,pred))
print('mean squared error : ', mean_squared_error(y_test,pred))
print('root mean squared error :',np.sqrt(mean_squared_error(y_test,pred)))
```

```
r2_score of linear refression is : 0.7557804863992295
Error :
mean absolute error : 1401450.7497591735
mean squared error :  34264005951725.082
root mean squared error : 5853546.442262595
```

Main Code for Running Multiple models at a time

```
#lets Choose r2 score of below four Models
dtc = DecisionTreeRegressor()
knn = KNeighborsRegressor(n_neighbors=5)
rf = RandomForestRegressor()
ada = AdaBoostRegressor()
```

```
#checking each model with Cross val score
model_list = [dtc,knn,rf,ada]
least_difference = []
for m in model_list:
    m.fit(x_train,y_train)
    pred = m.predict(x_test)
    cvs = cross_val_score(m,pca_x,Y,cv =5)
    print('\n')
    print(m)
    print('Scores :')
    print('r2 score:',r2_score(y_test,pred))
    print('Cross Val score :',cvs.mean())
    print('Error :')
    print('mean absolute error :',mean_absolute_error(y_test,pred))
    print('mean squared error : ', mean_squared_error(y_test,pred))
    print('root mean squared error :',np.sqrt(mean_squared_error(y_test,pred)))
    print('Difference :')
    difference = np.abs(r2_score(y_test,pred) - cvs.mean())
    print('Diffrence between cross val score and r2 score is : {0:.2f}'.format(difference))
    least_difference.append((m,'Diffrence between cross val score and r2 score error is : {0:.2f}'.format(difference)))
```

2. Decision Tree Regressor

```
DecisionTreeRegressor()
Scores :
r2 score: 0.7876683671643416
Cross Val score : 0.5862701194900034
Error :
mean absolute error : 196080.33033033035
mean squared error :  352815053078.07806
root mean squared error : 593982.36764914
Difference :
Diffrence between cross val score and r2 score is : 0.20
```

3. KNN Regressor

```
KNeighborsRegressor()
Scores :
r2 score: 0.7300517909353378
Cross Val score : 0.6633897072224835
Error :
mean absolute error : 285387.5675675676
mean squared error :  448552061873.8739
root mean squared error : 669740.2943483944
Difference :
Diffrence between cross val score and r2 score is : 0.07
```

4. Random Forest Regressor

```
RandomForestRegressor()
Scores :
r2 score: 0.9063554631455253
Cross Val score : 0.7966674716005798
Error :
mean absolute error : 169207.97804232803
mean squared error :  155601884653.5004
root mean squared error : 394464.0473522275
Difference :
Diffrence between cross val score and r2 score is : 0.11
```

5. AdaBoost Regressor

```
AdaBoostRegressor()
Scores :
r2 score: 0.5313659638628508
Cross Val score : 0.4862422692058897
Error :
mean absolute error : 783148.1920337728
mean squared error :  778692934848.261
root mean squared error : 882435.796445419
Difference :
Diffrence between cross val score and r2 score is : 0.05
```

Key Metrics for success in solving problem under consideration.

R2 Score:

R-squared is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.

Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that

always predicts the expected value of y, disregarding the input features, would get a R2 score of 0.0.

We obtained the r2 score of 87.46 % which is very good.

Root Mean Squared Error (RMSE):

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points. RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. Lower values of RMSE indicate better fit.

Root Mean Squared of this Random forest model is 122,611.3725 which is quite high.

Mean Squared Error:

Mean squared error is the average of the squared error that is used as the loss function for least squares regression. It is the sum, over all the data points, of the square of the difference

between the predicted and actual target variables, divided by the number of data points.

Mean Absolute Error:

In the context of machine learning, absolute error refers to the magnitude of difference between the prediction of an observation and the true value of that observation. MAE takes the average of absolute errors for a group of predictions and observations as a measurement of the magnitude of errors for the entire group.

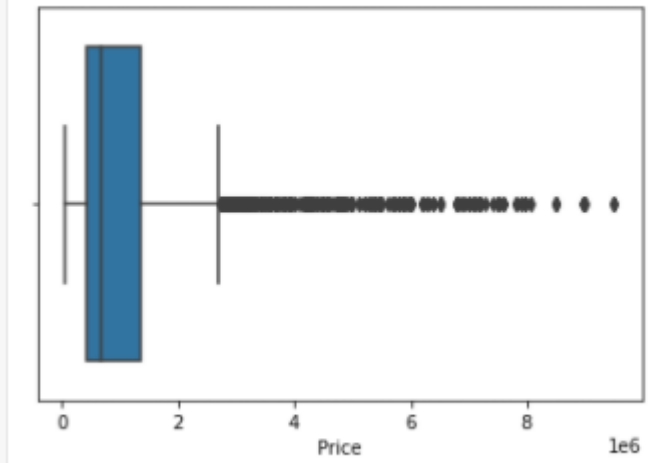
Mean Absolute error of this random forest model is 57,946.

Visualizations

The plots made along with their pictures and the inferences and observations obtained from those.

Outliers in Target column:

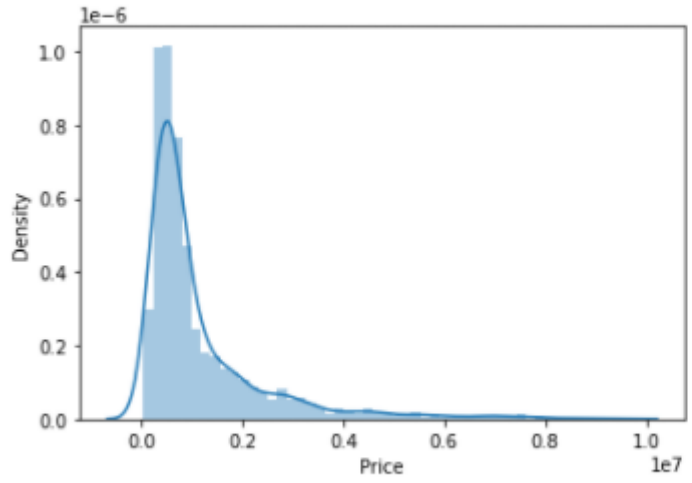
```
#lets check the box plot of our target column, to check if there are outliers
sns.boxplot(df.Price)
plt.show()
```



From the above plot we can see there are many outliers in the target column

Distribution of the target column:

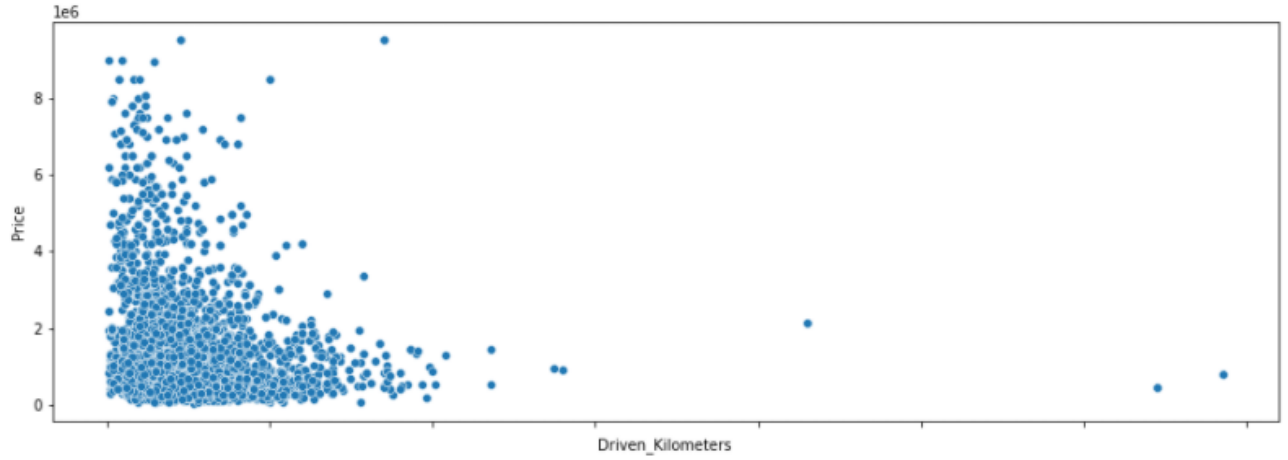
```
#lets check the distribution of the target column
sns.distplot(df.Price)
plt.show()
```



From the above plot we can see that data is right skewed in target column

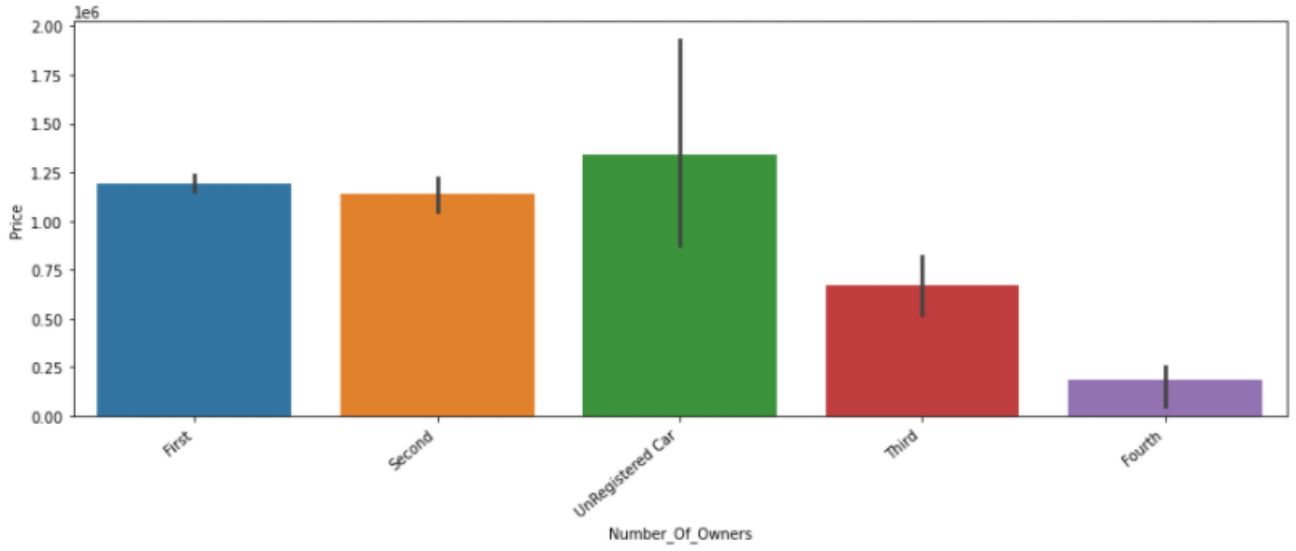
Bivariate Analysis with Target Variable:

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
col1 = sns.scatterplot(x = df['Driven_Kilometers'] , y =df['Price'] )
col1.set_xticklabels(col1.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



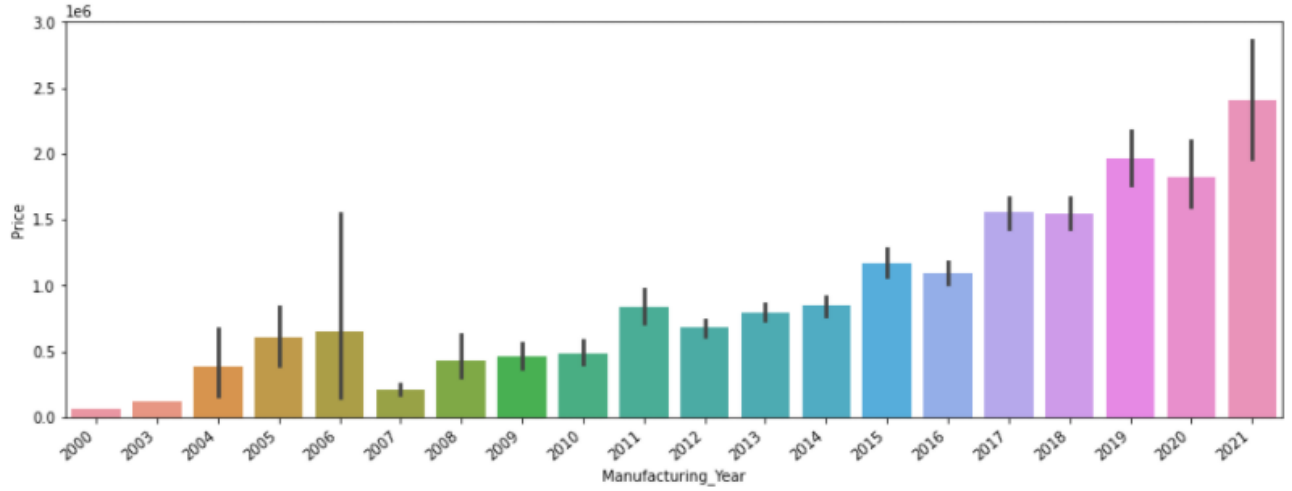
We don't see any linear relationship between Driven_Kilometers and Price.

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
col7 = sns.barplot(x = 'Number_Of_Owners' , y ='Price', data = df )
col7.set_xticklabels(col7.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



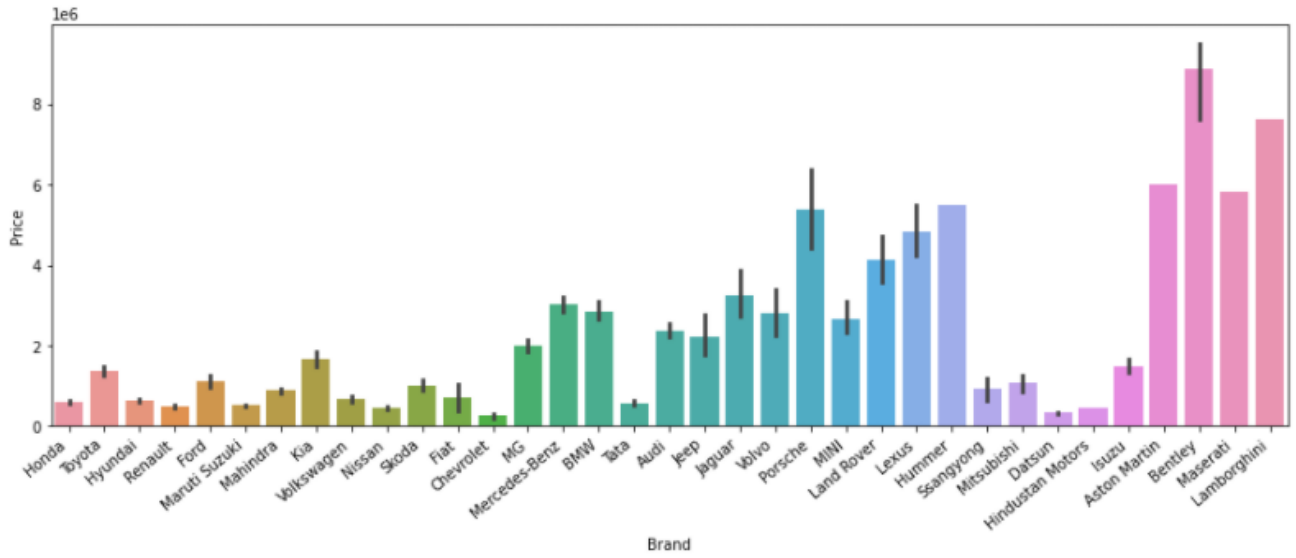
we can see that Unregister car has more prices compare to other

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
col7 = sns.barplot(x = 'Manufacturing_Year' , y ='Price', data = df )
col7.set_xticklabels(col7.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



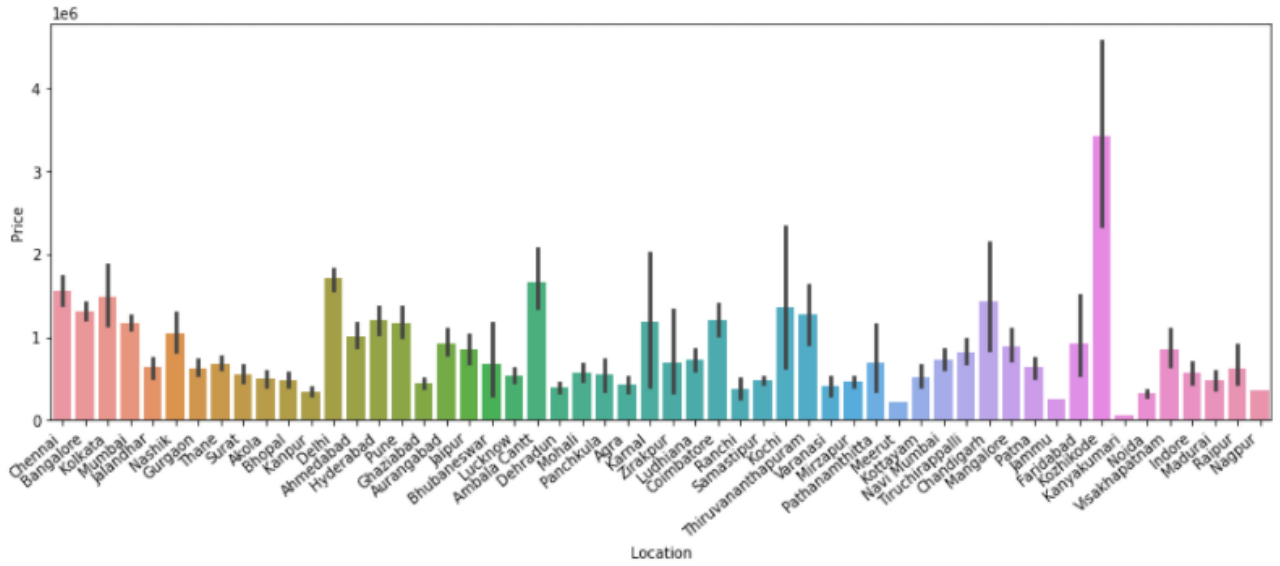
we can see that if the car is recently purchased than the price is high.

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
col7 = sns.barplot(x = 'Brand' , y ='Price', data = df )
col7.set_xticklabels(col7.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



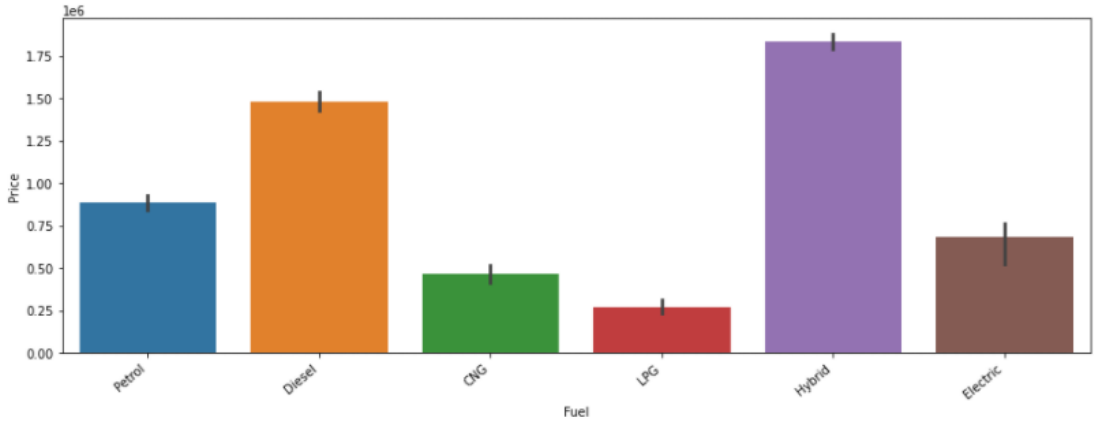
we can see that Bentley, Lamborghini, Aston Martin has more price compare to all.

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
col7 = sns.barplot(x = 'Location' , y ='Price', data = df )
col7.set_xticklabels(col7.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



We can see that Kozhikode location have high price and Kanyakumari location have low prices

```
#Lets check this column againt our target variable
plt.figure(figsize= (15,5))
col7 = sns.barplot(x = 'Fuel' , y ='Price', data = df )
col7.set_xticklabels(col7.get_xticklabels(), rotation=40, ha="right")
plt.show()
```



We can see that Hybrid type of transmission have high price and LPG type of transmission have low price

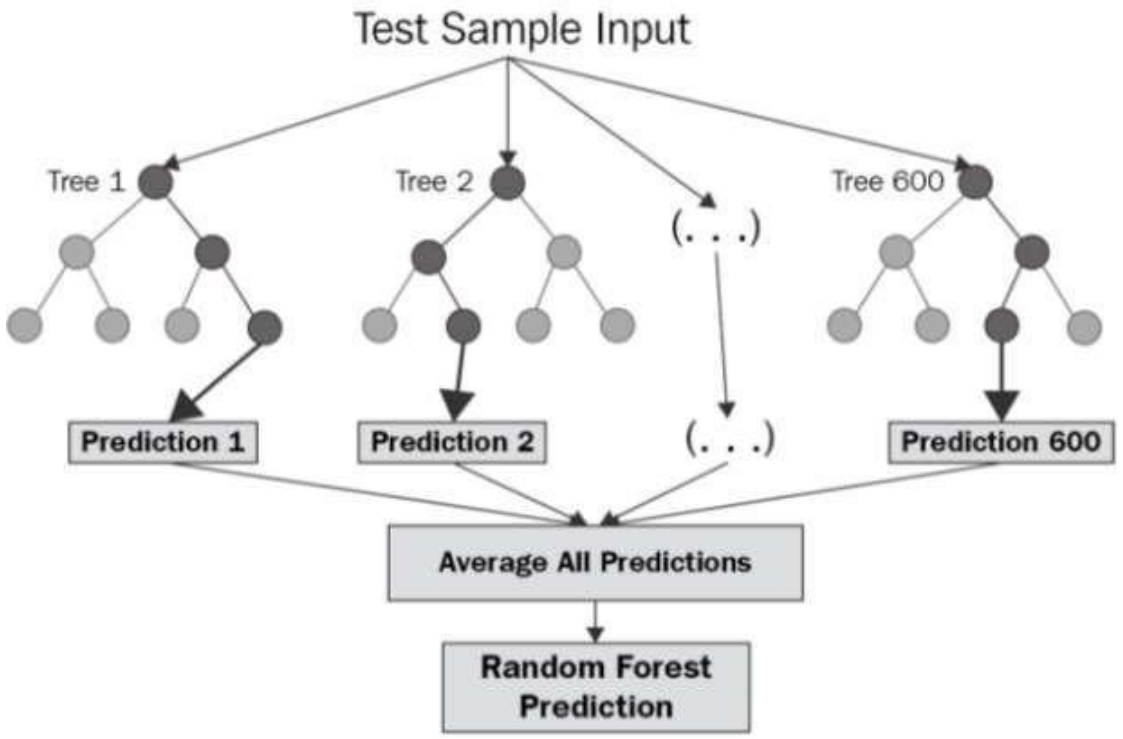
Interpretation of the Results

I choose Random forest as the final model since it has the best r2 score among all the model of 90%. This model is zero difference between cross validation cross and r2 score. So, this model is not over fitted or under fitted.

Creating a Final Model as Random Forest Regressor:

What is Random Forest Regressor?

Random Forest Regression is a supervised learning algorithm that uses **ensemble learning** method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.



The diagram above shows the structure of a Random Forest. We can notice that the trees run in parallel with no interaction amongst them. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees. To get a better understanding of the Random Forest algorithm, let's walk through the steps:

1. Pick at random *k* data points from the training set.

2. Build a decision tree associated to these *k* data points.

3. Choose the number *N* of trees you want to build and repeat steps 1 and 2.

4. For a new data point, make each one of your *N*-tree trees predict the value of *y* for the data point in question and assign the new data point to the average across all of the predicted *y* values.

A Random Forest Regression model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. Disadvantages, however, include the following: there is no interpretability, over fitting may easily occur, we must choose the number of trees to include in the model.

Let's create a random forest model with the best parameters obtained from Hyper Parameter tuning.

1. Random forest regressor - hyperparameter tuning

```
#lets use random forest regressor
#it takes a lot lot of time approx 6 hours, so commented after
parameters = {'n_estimators' : [100,200,300],'criterion':['mse','mae'],'max_features':['auto','sqrt','log2']}
gsvrf = RandomizedSearchCV(rf, parameters, cv=5,scoring="r2")
gsvrf.fit(x_train,y_train)
print(gsvrf.best_score_)
print(gsvrf.best_params_)
```

```
0.8777140176971837
{'n_estimators': 300, 'max_features': 'auto', 'criterion': 'mae'}
```

Creating Random Forest model using these parameters

```
final_model_rf = RandomForestRegressor(n_estimators= 300,criterion= 'mae',max_features = 'auto')
final_model_rf.fit(x_train,y_train)
final_pred = final_model_rf.predict(x_test)
cvs = cross_val_score(final_model_rf,pca_x,Y,cv =5)
print('\n')
print('Scores :')
print('r2 score:',r2_score(y_test,final_pred))
print('Cross Val score :',cvs.mean())
print('Error :')
print('mean absolute error :',mean_absolute_error(y_test,final_pred))
print('mean squared error : ', mean_squared_error(y_test,final_pred))
print('root mean squared error :',np.sqrt(mean_squared_error(y_test,final_pred)))
print('Difference :')
difference = np.abs(r2_score(y_test,final_pred) - cvs.mean())
print('Diffrence between cross val score and r2 score is : {0:.2f}'.format(difference))
least_difference.append((m,'Diffrence between cross val score and r2 score is : {0:.2f}'.format(difference)))
```

Output of Random Forest with best parameters obtained from hyper parameter tuning:

```
Scores :
r2 score: 0.8992801740996946
Cross Val score : 0.8007866478022319
Error :
mean absolute error : 172482.64414414414
mean squared error :  167358345275.54807
root mean squared error : 409094.54319942725
Difference :
Diffrence between cross val score and r2 score is : 0.10
```

```
#now our r2 score got incresed 89.92% after hyperparameter tuning
#diffrence between cross val score and r2 score is also 0.10
#which means no overfitting
```

Saving the Final model:

Serialization using joblib.

Pickled model as a file using joblib: Joblib is the replacement of pickle as it is more efficient on objects that carry large numpy arrays. These functions also accept file-like object instead of filenames.

Joblib.dump to serialize an object hierarchy.

Saving the model:

```
#serialization using joblib
import joblib
joblib.dump(final_model_rf,'Cars_Price.obj')
```

```
['Cars_Price.obj']
```

```
Cars_Price = joblib.load('Cars_Price.obj')
```

```
s_pred = Cars_Price.predict(x_test)
```

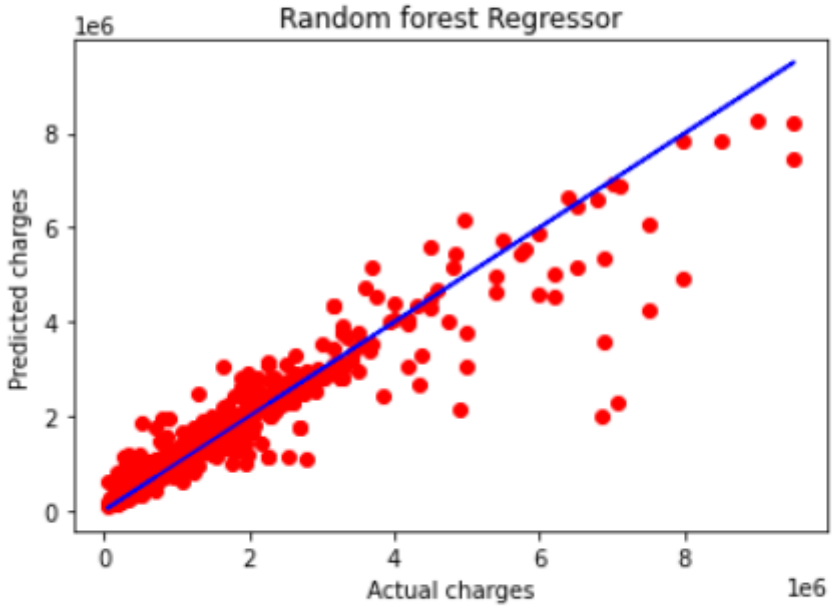
```
r2_score(y_test,s_pred)
```

```
0.8992801740996946
```

Let’s plot y_test vs predicted:

```
#lets plot y_test vs predicted

plt.figure(figsize=(6,4))
plt.scatter(x = y_test,y = s_pred,color = 'r')
plt.plot(y_test,y_test,color = 'b')
plt.xlabel('Actual charges')
plt.ylabel('Predicted charges')
plt.title('Random forest Regressor')
plt.show()
```



we can see that values are very close to the line

We can see that values are very close to the line.

Final Predicted Car Prices:

```
#lets make a dataframe of actual answers vs predicted answers
conclusion = pd.DataFrame((Cars_Price.predict(x_test)[:],y_test[:]),index= ['Predicted','Actual'])
conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	1100	1101
Predicted	679063.333333	363140.0	1.134447e+06	659183.333333	734070.0	2270800.0	1994700.0	828236.666667	4238055.0	617260.0	...	327840.0	416370.0
Actual	695000.000000	365000.0	1.125000e+06	330000.000000	525000.0	2600000.0	1995000.0	825000.000000	7500000.0	650000.0	...	295000.0	340000.0

2 rows × 1110 columns

CONCLUSION

- Key Findings and Conclusions of the Study

  Though this is the simplest model we've built till now, the final predictors still seem to have high correlations. One can go ahead and remove some of these features, though that will affect the adjusted- r2 score significantly (you should try doing that).

  Thus, for now, the final model consists of the 8 variables mentioned above.

- Learning Outcomes of the Study in respect of Data Science

  I choose Random Forest algorithm as my final model since it was having least difference between its cross-validation score and r2 score.

  Random forest is also Robust to outliers, since there were some outliers even after data cleaning, I used the random forest algorithm.

  I learned how to use regular expression to extract the required information from the existing columns

- Limitations of this work and Scope for Future Work

  Yes, there is still room for improvement, like doing a more extensive feature engineering, by comparing and plotting the features against each other and identifying and removing the noisy features. The values of R-squared obtained from the algorithm give the accuracy of the model. In the future, if more data could be scraped such as the Colour, Scratches, Tyre Condition of cars so predicted results will be more accurate.