Car Price Prediction

Submitted By

Shivanchal Asthana

ACKNOWLEDGMENT

It is my sensual gratification to present this report. Working on this project was an incredible experience that will have a tremendous impact on my career. I would like to express my sincere thanks to the company Flip Robo Technologies for a regular follow up and valuable suggestions provided throughout. They always been an origin of spark and direction. I also thank all the respondents who have given their valuable time, views and valid information for this project.

Shivanchal Asthana

INTRODUCTION

- ## Business Problem Framing

  Due the impact of covid19 to the world, we see there is lot of changes in the price in car market, so, it is very helpful for anybody to predict a price of used cars with accurate results.

- ## Conceptual Background of the Domain Problem

  Car price prediction with Machine Learning One of the main areas of research in machine learning is the prediction of the price of cars. It is based on finance and the marketing domain. It is a major research topic in machine learning because the price of a car depends on many factors.

- ## Review of Literature

  We discuss the conceptual background of this problem and the basic idea for this project. We collect our data from cars Dekho website. And we used these ideas and data to make our model with best algorithms and techniques.

- **Motivation for the Problem Undertaken**

  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. So, for him, we need to make car price valuation model.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

  Starting with the dataset, when I looked through the statistical description, we come to see that most of that there are lot of data cleaning needed, some columns contain many things in one column like years, brand name etc. So, we use split command to split the data and slice them, after that join them and make some new columns like purchase year, model name, car name, brand name, etc. Some columns have some outliers, so, to remove the outliers, I used IQR method, but there is no zero values and no null values are in columns.

  Initially, every column is in object form but at the end, we convert this into integer and float. We use 6-7 models to find the best accuracy for our prediction model, use CV

score, K-Fold CV score, plot actual and predicted values and to improve accuracy we use grid search cv also and at last, Linear regression is our best model.

- **Data Sources and their formats**

Data is scrapped by us from Car Dekho website using web scrapping selenium. We use different locations to scrape this data and we were very cautious about the imbalanced data, missing values, zero values and duplicate data. The data is been provided in CSV format with 10 different variables in different columns and 6421 rows.

- **Data Pre-processing Done**

In this dataset, columns contain many values, so, we first split them, slice them and join them to make better cleaning. Then some columns contain outliers, so, we treat them by using IQR method. However, there are no null values found, no zero values found.
But all columns are in string form, so we convert them into integer.

- **Data Inputs- Logic- Output Relationships**

The input data which we have scrapped by using web scrapping - selenium, help us to understand the analysis about used cars like relationship between gear type with number of kilo meter driven, how Number of owners are

related to kilo meter driven and how these things related to our price (output).

- **State the set of assumptions (if any) related to the problem under consideration**

  No as such assumption been done related to the circumstances

- **Hardware and Software Requirements and Tools Used**

  Hardware is used by me that is i5 intel core, 8GB RAM, 64Bit processor, and software is Jupyter Notebook for coding along with MS Word to make useful report, MS – PowerPoint to make presentation. For coding, we need to install NumPy, Pandas, Sklearn libraries.

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

The data set contains more than 6 thousand data with no null values, no zero values, data is not imbalanced. This is clearly a regression problem. First, we see, all columns are in object or in string, which we convert them into integer at the last.

Then we see, there are some columns who contains many things in one column, in simple way, they need some splitting to split their data into two or three columns, so, we did that, we use split them, slice them and join them after that to make data cleaning better. Then after that we, see some column contains outliers, so, we treat them by using IQR method, and we lose some of our useful data also but we successfully improve the skewness of those columns.

Then we label encoder to convert string columns into integer. Then we look forward towards to remove multicollinearity, so, we plot heatmap and use variance inflation factor to check the multicollinearity but we don't get any multicollinearity between independent variables and we got high multicollinearity with one feature (Emi) with our label (price), and this is a good thing for us. We use 6-7 models to find the best accuracy for our prediction model, use CV score, KFold CV score, plot actual and predicted values and to improve accuracy

we use grid search cv also and at last, Linear regression is our best model.

- **Testing of Identified Approaches (Algorithms)**

  - Linear regression’
  - Decision Tree regressor
  - kNeighbors regressor
  - Support Vector Regressor
  - Random forest Regressor
  - Lasso regressor
  - Ridge Regressor
  - AdaBoost Regressor
  - XGBoost Regressor
  - ExtremeGradientBoost Regressor

- **Run and Evaluate selected models**

```
#Model instantiating and training
lr = LinearRegression()
dtr=DecisionTreeRegressor()
rfo = RandomForestRegressor()
svr = SVR()
knn = KNeighborsRegressor()
ada = AdaBoostRegressor()
```

We use here, 6 models, since, this a regression-based problem. So, we use those models also which are best in classification problems like support vector machine etc.

```
print('Linear Regression:----->',lr.fit(x_train, y_train))
time.sleep(2)
print('Decsion Tree:------->',dtr.fit(x_train, y_train))
time.sleep(2)
print('Random Forest:-------->',rfo.fit(x_train, y_train))
time.sleep(2)
print('Support vector machine:-------->',svr.fit(x_train, y_train))
time.sleep(2)
print('KNeighborsRegressor:-------->',knn.fit(x_train, y_train))
time.sleep(2)
print('Adaboost Regressor:-------->',ada.fit(x_train,y_train))
```

```
Linear Regression:-----> LinearRegression()
Decsion Tree:-------> DecisionTreeRegressor()
Random Forest:--------> RandomForestRegressor()
Support vector machine:--------> SVR()
KNeighborsRegressor:--------> KNeighborsRegressor()
Adaboost Regressor:--------> AdaBoostRegressor()
```

Now, we trained the models.

```
#Check How much our model learn
print('Linear regression Score:-------->', lr.score(x_train, y_train))
time.sleep(2)
print('Decision Tree score:------->', dtr.score(x_train,y_train))
time.sleep(2)
print('Random Forest score:------>',rfo.score(x_train, y_train))
time.sleep(2)
print('Support vector machine score:------>',svr.score(x_train, y_train))
time.sleep(2)
print('KNeighborsRegressor score:------>',knn.score(x_train, y_train))
time.sleep(2)
print('Adaboost Regressor:-------->',ada.score(x_train,y_train))
```

```
Linear regression Score:--------> 0.9999999948968942
Decision Tree score:-------> 1.0
Random Forest score:------> 0.9999980246618797
Support vector machine score:------> -0.015397222875344063
KNeighborsRegressor score:------> 0.9632515861028651
Adaboost Regressor:--------> 0.995405242580287
```

We see clearly, here, linear regression, decision Tree, random forest, even ada boost regressor is also giving us 99% accuracy to learn the trained data, KNeighbors is giving us 96% accuracy which is also good. But support vector regressor is giving us worst accuracy to learn the trained data

```
#Let's check how well model fits the test data
print('Linear regression Score:-------->', lr.score(x_test, y_test))
time.sleep(2)
print('Decision Tree score:------->', dtr.score(x_test,y_test))
time.sleep(2)
print('Random Forest score:------>',rfo.score(x_test, y_test))
time.sleep(2)
print('Support vector machine score:------>',svr.score(x_test, y_test))
time.sleep(2)
print('KNeighborsRegressor score:------>',knn.score(x_test, y_test))
time.sleep(2)
print('Adaboost Regressor:-------->',ada.score(x_test,y_test))
```

```
Linear regression Score:--------> 0.9999999952543087
Decision Tree score:-------> 0.9999961567397665
Random Forest score:------> 0.9999980615764602
Support vector machine score:------> -0.015006432620055188
KNeighborsRegressor score:------> 0.9346189259043897
Adaboost Regressor:--------> 0.9959492191103991
```

We see clearly, here, linear regression, decision Tree, random forest, even ada boost regressor is also giving us 99% accuracy, KNeighbors is giving us 96% accuracy which is also good. But support vector regressor is giving us worst accuracy.

- **Visualizations**

```
df.shape #check the shape of the column
```

```
(6421, 10)
```

The shape of the dataset is 6421 rows and 10 columns.

```
df.duplicated().sum() #check the duplicate rows
```

```
0
```

There is no duplicate values found. There is no zero values in the dataset.

```
df.all() #check the z
```

```
Unnamed: 0     False
Brand_Name      True
Car_Name        True
Location        True
Km_Driven       True
Gear_Type       True
Fuel            True
Owner           True
EMI             True
Price           True
dtype: bool
```

```
df.isnull().sum()
```

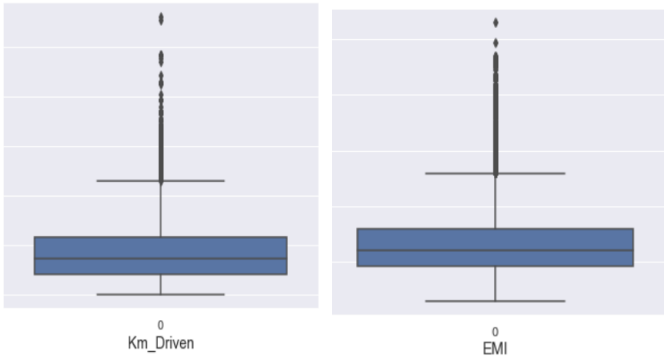
```
Unnamed: 0     0
Brand_Name     0
Car_Name       0
Location       0
Km_Driven      0
Gear_Type      0
Fuel           0
Owner          0
EMI            0
Price          0
dtype: int64
```

No Null values are found

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6421 entries, 0 to 6420
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Location          6421 non-null   object
 1   Km_Driven         6421 non-null   object
 2   Gear_Type         6421 non-null   object
 3   Fuel              6421 non-null   object
 4   Owner             6421 non-null   object
 5   EMI               6421 non-null   object
 6   Price             6421 non-null   object
 7   Year of Purchase  6421 non-null   object
 8   Brand_name        6421 non-null   object
 9   Car_name          6421 non-null   object
 10  Model_name        6421 non-null   object
dtypes: object(11)
memory usage: 551.9+ KB
```

We can see, all columns are in string form, which we need to convert them by using astype method or labelEncoder.



We see, in Km_driven and Emi column we found some outliers, we reomove them by using IQR method.

```
df['Location'].unique() #check the unique values
```

```
array(['New Delhi', 'Bengaluru', 'Mumbai', 'Chennai', 'Hyderabad',
       'Kolkata', 'Ahemdabad', 'Pune'], dtype=object)
```

```
df['Km_Driven'].unique() #check the unique values
```

```
array(['50,739 km', '45,043 km', '70,313 km', ..., '74,780 km',
       '39,630 km', '17,362 km'], dtype=object)
```

```
df['Gear_Type'].unique() #check the unique values
```

```
array(['Manual', 'Automatic'], dtype=object)
```

```
df['Fuel'].unique() #check the unique values
```

```
array(['Petrol', 'Diesel', 'Petrol + CNG', 'Petrol + LPG'], dtype=object)
```

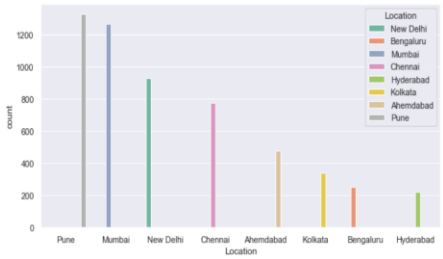
```
df['EMI'].unique() #check the unique values
```

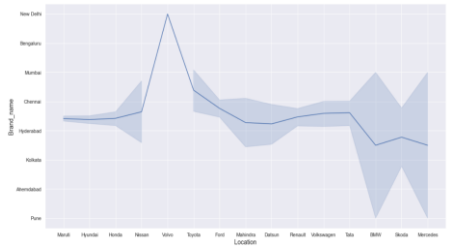
```
array(['₹7,721/month', '₹6,197/month', '₹5,839/month', ...,
       '₹13,017/month', '₹17,880/month', '₹20,156/month'], dtype=object)
```

```
df['Owner'].unique() #check the unique values
```

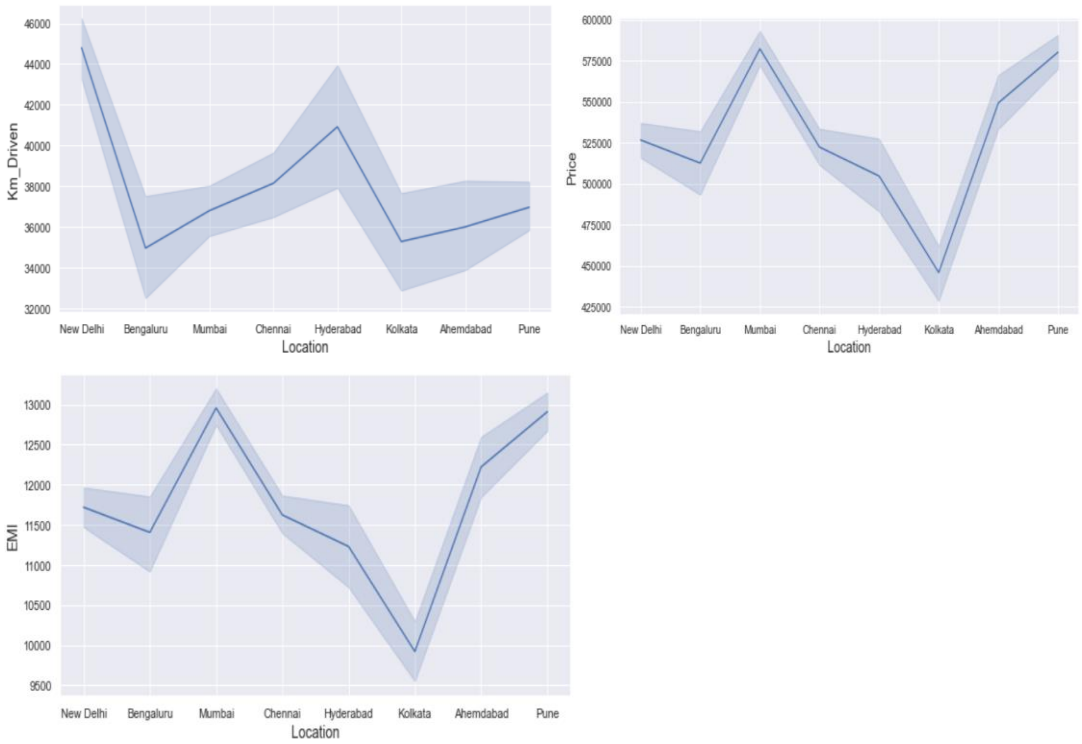
```
array(['1st Owner', '2nd Owner', '3rd Owner', '4th Owner'], dtype=object)
```

We can see, These columns are categorical in nature. Like Location columns contains some of them our countries major cities, gear type contains two types Either it is in manual or Automatic type. Fuel column contains Petrol, Diesel, Petrol+CNG , Petrol +LPG, and last Owener column contains 1st, 2nd, 3rd and 4th type owner and rest km driven and emi acolumn are continous in nature.

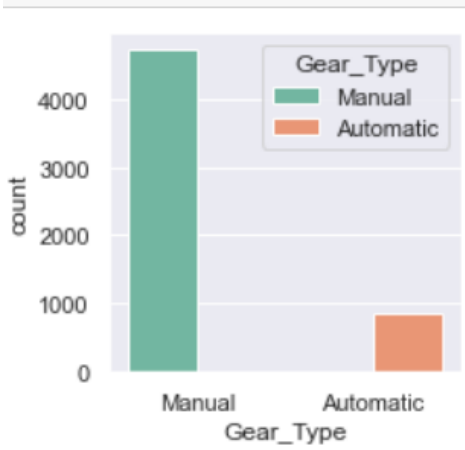




- Pune and Mumbai has more cars in this dataset for prediction.
- Hyderabad has least numbers of cars
- Volvo is from New Delhi only
- By analysing histogram, we can conclude that Maruti is high in numbers and it is mainly from New Delhi, Pune and Mumbai.
- BMW and Mercedes is from Pune and Mumbai.



- New Delhi has cars who travels more kilo meters.
- Bengaluru has cars who travels less kilo meters.
- We can see, In Pune and Mumbai, people have to pay more Emi than other cities.
- In Kolkata, people have to pay low Emi than others
- Used cars price in Kolkata is less while it is higher in Pune and Mumbai.




```
df['Owner'].value_counts()
```

```
1st Owner    4481
2nd Owner     988
3rd Owner      96
4th Owner      14
Name: Owner, dtype: int64
```

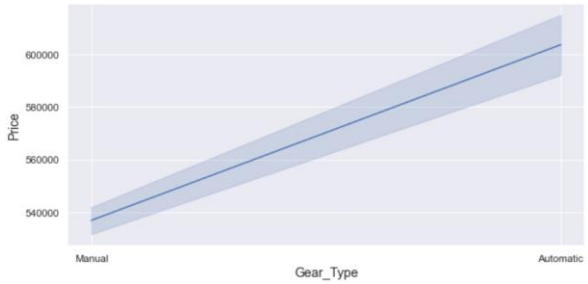
```
df['Fuel'].value_counts()
```

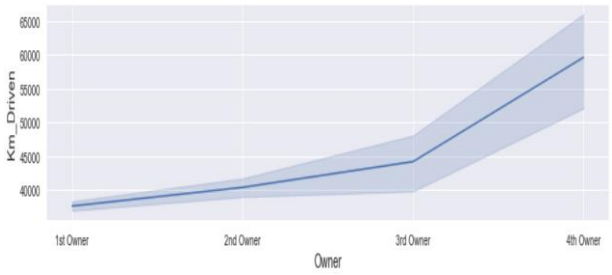
```
Petrol          4401
Diesel          1112
Petrol + CNG      65
Petrol + LPG       1
Name: Fuel, dtype: int64
```

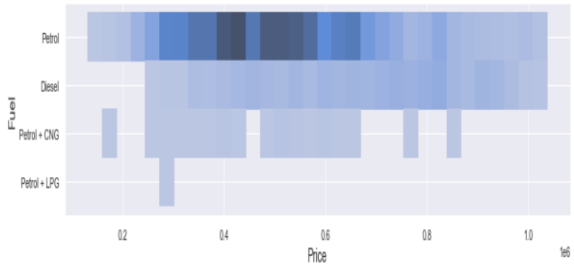
```
df['Price'].value_counts(bins=10)
```

```
(403439.0, 494019.0]      1114
(494019.0, 584599.0]      1098
(312859.0, 403439.0]       830
(584599.0, 675179.0]       774
(675179.0, 765759.0]       502
(222279.0, 312859.0]       431
(765759.0, 856339.0]       413
(856339.0, 946919.0]       251
(946919.0, 1037499.0]      131
(130793.199, 222279.0]      35
Name: Price, dtype: int64
```

- Around 4 lacs to 5 lacs price of used cars are more in count.
- As the price range is increases, Number of used cars are decreasing.
- Petrol cars are more than others.
- Only one Petrol + LPG used car in this dataset.
- Manual used cars are more than Automatic.
- 1st Owner cars are more than others.







- Price of Automatic used cars are more than Manual used cars.
- Price of Petrol used cars are around 4 to 5 lacs
- Price of Petrol + LPG used car is around 3 lacs.
- As the Ownership increases, Kilo meter driven increases.
- Maruti used cars are more, both for Manual and Automatic Gear Type.
- Volvo is Automatic Type.
- 2017 is highest for years of purchase of these cars.
- Mercedes, BMW and Volvo are least and all are automatic.
- Purchased cars are lower in 2007.
- Nissan and Skoda are manual cars.

```
df['Location'].unique()
```

```
array([6, 1, 5, 2, 3, 4, 0, 7])
```

```
df['Owner'].unique()
```

```
array([0, 1, 2, 3])
```

```
df['Gear_Type'].unique()
```

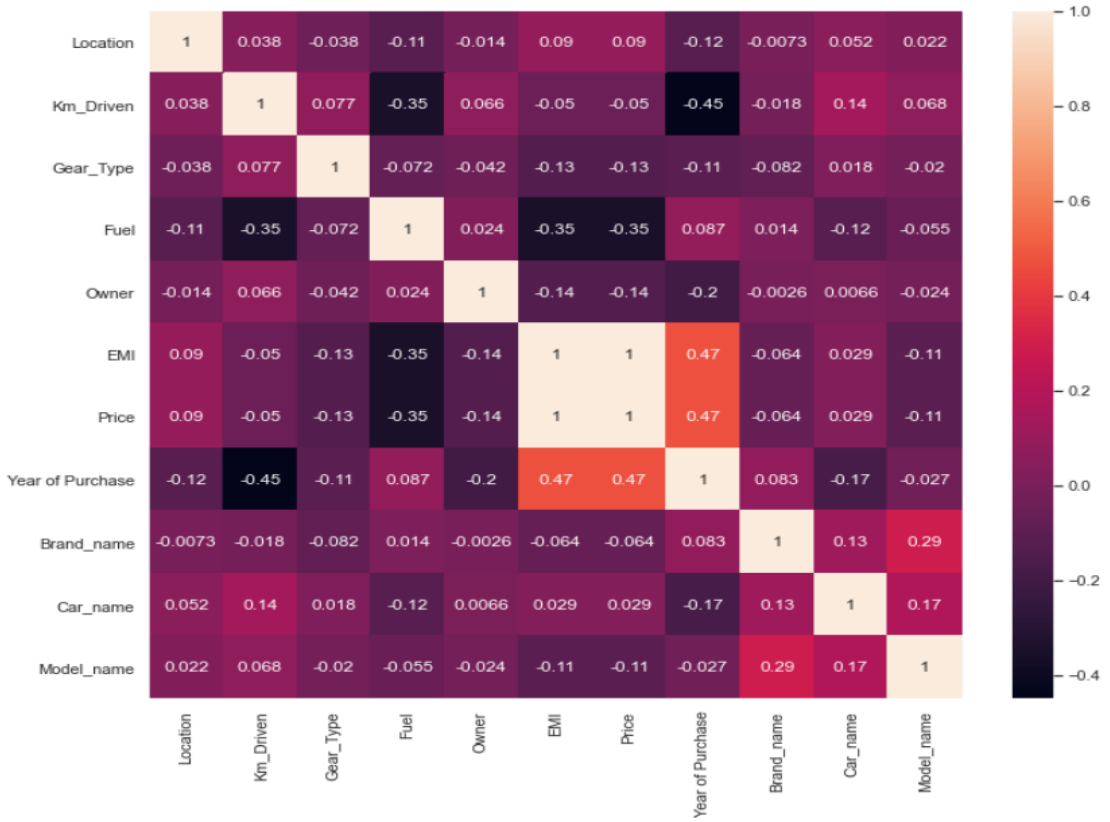
```
array([1, 0])
```

```
df['Fuel'].unique()
```

```
array([1, 0, 2, 3])
```

- We use Label Encoder to convert object columns into integer.
- Gear Type has Manual and Automatic so, It convert 1 for manual and 0 for Automatic.
- Fuel has Petrol, Diesel, Petrol + CNG and Petrol + LPG, so, it converts 1 for petrol, 0 for Diesel, 2 for Petrol + CNG and 3 for Petrol + LPG.
- Owner has 1^{st},2^{nd},3^{rd} and 4^{th} owner. And it converts 0,1,2,3
- Locations has major 8 cities of the country and it converts 6 for New Delhi, 1 for Bengaluru, 5 for Mumbai and so on.

- **Multicollinearity**



	vif	features
0	1.053619	Location
1	1.443722	Km_Driven
2	1.052056	Gear_Type
3	1.408435	Fuel
4	1.052010	Owner
5	1.759220	EMI
6	1.908020	Year of Purchase
7	1.132028	Brand_name
8	1.102613	Car_name
9	1.139932	Model_name

- First we saved our cleaned dataset.
- By observing histogram, there is no multicollinearity between independent variables.

- There is high multicollinearity between Independent variable (Emi) and dependent variable (price), which is good for our model accuracy.
- By using variance inflation factor, there is no high multicollinearity between independent variables.

Feature Scalling

```
#scale the value x
Scaler = StandardScaler()
x_scaled = Scaler.fit_transform(x)
x_scaled
```

```
array([[ 0.66905252,  0.52974933,  0.42689811, ...,  0.21794529,
        -1.28938512,  0.91605117],
       [ 0.66905252,  0.28773006,  0.42689811, ...,  0.21794529,
        -1.28938512,  0.91605117],
       [ 0.66905252,  1.36143563,  0.42689811, ...,  0.21794529,
        -1.28938512,  0.91605117],
       ...,
       [ 1.10911901,  2.17472405, -2.34247931, ...,  1.93898815,
         0.57018317, -0.75994415],
       [ 1.10911901,  0.47264366,  0.42689811, ..., -1.50309757,
        -0.6571319 , -1.38489156],
       [ 1.10911901, -0.30882025,  0.42689811, ..., -1.50309757,
        -0.6571319 , -1.38489156]])
```

We use standard scaler to scale the data.

Model Building

```
#Model instantiating and training
lr = LinearRegression()
dtr=DecisionTreeRegressor()
rfo = RandomForestRegressor()
svr = SVR()
knn = KNeighborsRegressor()
ada = AdaBoostRegressor()
```

```
Linear regression Score:--------> 0.9999999952543087
Decision Tree score:-------> 0.9999961567397665
Random Forest score:------> 0.9999980615764602
Support vector machine score:------> -0.015006432620055188
KNeighborsRegressor score:------> 0.9346189259043897
Adaboost Regressor:--------> 0.9959492191103991
```

- We use, 6 models here to check the accuracy.

- We see here, support vector machine is giving us worst accuracy, but Linear regression is giving us best accuracy here.

Cross Validation Score

```
Linear regression CV Score:
0.9999999948239896

 **********************************************
Decision Tree CV Score:
0.999973856858006

 **********************************************
Random Forest CV Score:
0.9999794106893752

 **********************************************
KNeighbour Classifier CV Score:
0.9268640942704289

 **********************************************
Support Vector Machine CV Score:
-0.04398600594237627

 **********************************************
Ada Boost CV Score:
0.9954871612385767
```

KFold CV

```
from sklearn.model_selection import KFold, StratifiedKFold
kfold_validation = KFold(10)
skfold = StratifiedKFold(n_splits = 5)
```

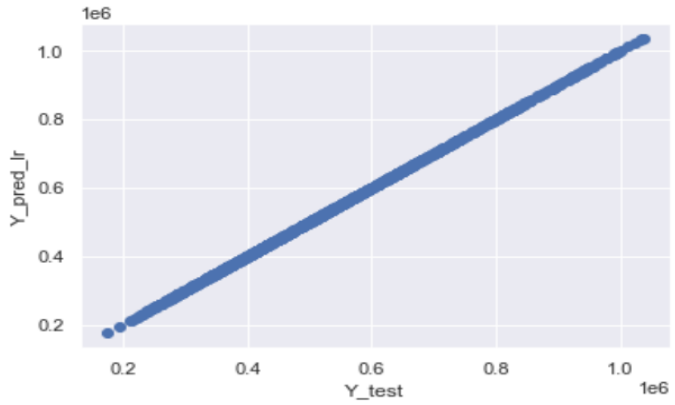
```
Mean value of CV score:-----> 0.9999999946591721
Min value of CV score:-----> 0.9999999928857907
Max value of CV score:-----> 0.999999995250144

 **********************************************
Decision Tree CV Score:
Mean value of CV score:-----> 0.9999819035907175
Min value of CV score:-----> 0.9998515798086546
Max value of CV score:-----> 0.9999994682570283

 **********************************************
Random Forest CV Score:
Mean value of CV score:-----> 0.999981197571163
Min value of CV score:-----> 0.9998285216435538
Max value of CV score:-----> 0.9999991777490377

 **********************************************
KNeighbour Regressor CV Score:
Mean value of CV score:-----> 0.9351333132148347
Min value of CV score:-----> 0.9122169144251997
Max value of CV score:-----> 0.9555137645433837

 **********************************************
Support Vector Machine CV Score:
Mean value of CV score:-----> -0.0714831730210688
Min value of CV score:-----> -0.1494246490820732
Max value of CV score:-----> -0.03181590351962149

 **********************************************
Ada Boost Regressor CV Score:
Mean value of CV score:-----> 0.994482668868341
Min value of CV score:-----> 0.9907893275622675
Max value of CV score:-----> 0.9957831429508748
```

We use **Cross_val_score** and **kfold cv score**.

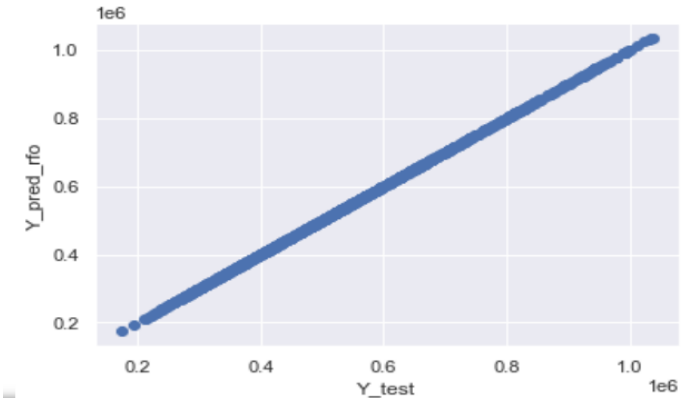
- In kfold cv score, Linear regression is giving good accuracy.
- By using cross val score, linear regression is still giving better accuracy than other models

Plot between Actual and Predicted Values

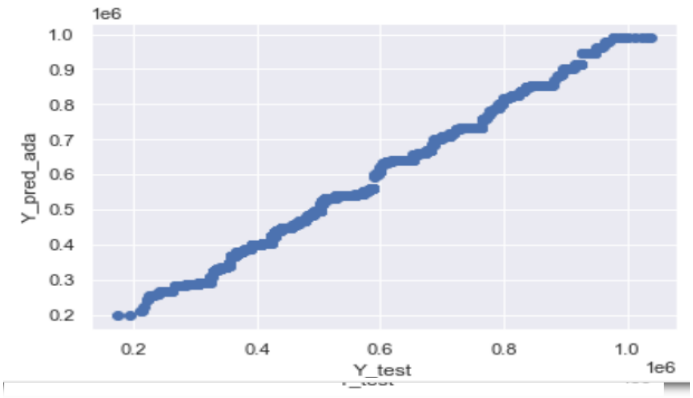
Linear Regression:



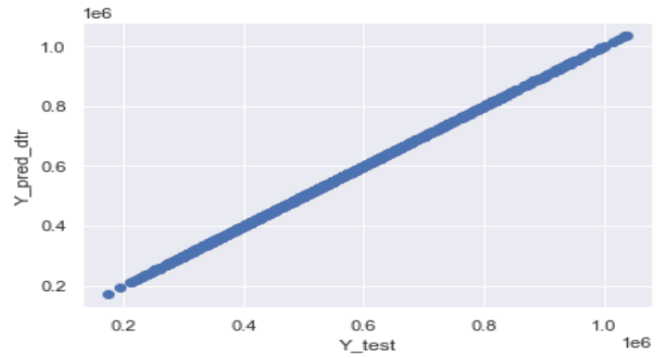
Random Forest:



Ada Booster:



Decision Tree:



- We can see here, Linear Regression, Random forest and Decision Tree is giving us linear relationship between Actual and predicted values.
- Ada boost regressor is also giving us good relationship but not better than other models.

Model Evaluation: MAE, MSE and RSME

```
Linear Regression:
MAE:---------> 10.88715296787411
MSE:-----------> 162.6959171597756
RSME:---------> 12.755230972419731
Decision Tree:
MAE:---------> 144.27903974260968
MSE:-----------> 131757.9917593832
RSME:---------> 362.98483681743954
Random Forest:
MAE:---------> 123.38098709768016
MSE:-----------> 66454.72262372014
RSME:---------> 257.7881351492348
Support vector machine:
MAE:---------> 149475.63564026458
MSE:-----------> 34797333789.91331
RSME:---------> 186540.4347317581
KNeighbors:
MAE:---------> 31938.69911882822
MSE:-----------> 2241450877.2868376
RSME:---------> 47343.96347251503
Ada Booster:
MAE:---------> 9291.32351364222
MSE:-----------> 138872395.47969303
RSME:---------> 11784.413242910867
```

- Linear regression is giving less values in MAE, MSE and RSME than other.
- Support vector machine is giving us worst values as it is mainly used for classification problems.
- Decision Trees and Random Forest are also giving us 99% accuracy but still they are giving us high values in MSE, MAE and RSME

Lasso, Ridge and XGBoost Regressor

```
lasso_reg.score(x_test,y_test)
```

```
0.9999999949366178
```

```
ridge_model.score(x_test,y_test)
```

```
0.999999995254457
```

```
import xgboost as xgb #import libraries
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

```
xgbr = xgb.XGBRegressor()  #to improve accuracy
xgbr.fit(x_train,y_train)
y_pred = xgbr.predict(x_test)
r2_score(y_test, y_pred)
```

```
0.9999836886834499
```

- Lasso Regression is giving us 99% accuracy
- Ridge Regression is also giving us 99% accuracy
- XG Boost Regressor is also giving us 99% accuracy

GradientBoost Regressor

```
# GradientBoostingRegressor
```

```
gradientregressor = GradientBoostingRegressor(max_depth=3, n_estimators=6, learning_rate=.4)
#using GBR
```

```
gbr = gradientregressor.fit(x_train,y_train) #fit the data
```

```
r2_score(y_test,y_pred) #predict it
```

```
0.9999836886834499
```

Grid Search CV by using GradientBoost Regressor as a estimator

```
GridSearchCV(estimator=GradientBoostingRegressor(learning_rate=0.4,
                                                 n_estimators=6),
             param_grid={'learning_rate': array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]),
                         'max_depth': range(4, 12, 2),
                         'min_samples_split': range(4, 8, 2)})

 **************************************************
GridSearchCV best parameters:-----> {'algorithm': 'brute', 'leaf_size': 1, 'n_neighbors': 3}

 **************************************************
Trained Model using best parameters:-----> GradientBoostingRegressor(learning_rate=0.8, max_depth=10, min_samples_split=6,
                          n_estimators=6)
Accuracy score:------> 0.9999976694927434

 **************************************************
Total time taken (in mins):------> 0.38786216974258425
```

- GradientBoostRegressor is also giving us 99% accuracy.
- Using Gridsearch cv, GradientBoostRegressor as a estimator, is giving us 99% accuracy.
- But after gridsearch cv, accuracy is slightly more than GradientBoostRegressor.

Saving the best model

```
# Saving the model

#We see,Linear regression is giving us best accuracy and there MSE, MAE and RSME are better than other regressor,
#so, this is our best model

with open('Car_price_prediction','wb') as f:
    pickle.dump(LinearRegression(),f)
```

Prediction by the model

```
#predict the price of car

print("The price of car is:----->", lr.predict(Scaler.transform([[6,50739.0,1,1,0,7721,2017,6,5,88]])))
```

```
The price of car is:-----> [347099.0840559]
```

- We see, Linear Regression is our best model.
- By predicting some values, we can check our model is predicting right or not.

Conclusion

- **Key Findings and Conclusions of the Study**

Mostly the used cars are in this dataset are from Pune and Mumbai. Majority of cars are from Maruti.

Some cars are Manually operated and some are Automatic. Volvo, Mercedes and BMW are Automatically operated. Older the cars, number of owners are more and kilo meter driven are higher.

Newly purchased cars have high Emi rate. As the price increase, Emi increases.

Mostly, used cars are petrol dependent, some cars are diesel operated.

Mostly, the customers have the intension of repaying. There are certain cases, when the customers have no intension of repayment but the number of such customers are few. With the model built, we can certainly determine customers having intension of repayment or not.

- **Learning Outcomes of the Study in respect of Data Science**

Data contains more than 6 thousand rows and 10 columns. There are no null values, no zero values, no duplicate values found, By removing the outliers by using IQR method, we improve the skewness of some of our features and price column, which is our label. We use 6-7 models to find the best accuracy for our prediction model, use CV score, KFold CV score, plot actual and predicted values and to improve accuracy we use grid search cv also and at last, Linear regression is our best model.

We have used regression analysis and have predicted the selling price of the used car based on various features of the cars, including the Location, Brand name, car name, kilo meter driven. Emi, fuel, owner of the cars.