CAR PRICE PREDICTION

Submitted by: **SHANTY EMERSON**

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped you and guided you in completion of the project.

References:

1. Python Data Analysis: Perform data collection, data processing, wrangling, visualization, and model building using Python, 3rd Edition. by Avinash Navlani (Author), Armando Fandango (Author), Ivan Idris (Author)
2. Notes and classes by Datatrained Academy.

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.

Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across cities in the United States.

Deciding whether a used car is worth the posted price when you see listings online can be difficult. Several factors, including mileage, make, model, year, etc. can influence the actual worth of a car. From the perspective of a seller, it is also a dilemma to price a used car appropriately. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used car price.

As a Data scientist 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

Methodology represents a description about the framework that is undertaken. It consists of various milestones that need to be achieved in order to fulfil the objective. We have various attributes that contribute retaining customers that aids the growth of business.

The following steps represents stepwise tasks that need to be completed:

- Data Collection
- Exploratory Data Analysis
- Data pre-processing
- Data Visualization
- Model Building
- Model Evaluation
- Saving the best model

Data Sources and their formats

We have a Dataset of 20000 entries and it has 8variables.

Data Count :20000

Data Features :8

Data Types : Object/Int/Float

```
#checking the datatype of each column
print(df.dtypes)
```

```
DESCRIPTION      object
LOCATION         object
MANUFACTURER     object
MODEL            object
YEAR              int64
FUEL TYPE        object
KMS DRIVEN      float64
PRICE           float64
dtype: object
```

Data Pre-processing

Data pre-processing is a process of transforming the raw, complex data into systematic understandable knowledge.

- Data Sample:

	DESCRIPTION	LOCATION	MANUFACTURER	MODEL	YEAR	FUEL TYPE	KMS DRIVEN	PRICE
0	Maruti suzuki omni van	Bengaluru	Maruti Suzuki	Omni	2010	Petrol	24000.0	210000.0
1	Skoda Rapid 1.5 Tdi Cr Ambition (make Year 201...	Bengaluru	Skoda	Rapid	2012	Diesel	53000.0	530000.0
2	Hyundai Santro GL Plus	Bengaluru	Hyundai	Santro Xing	2013	Petrol	25400.0	315000.0
3	Chevrolet Beat Lt Petrol (make Year 2013) (pet...	Bengaluru	Chevrolet	Beat	2013	Petrol	26000.0	365000.0
4	Skoda Laura Elegance 2.0 Tdi Cr At (make Year ...	Bengaluru	Skoda	Laura	2010	Diesel	89000.0	790000.0
...	...	...	...	...	...	...	...	...
20000	Skoda Rapid Ambition 1.6 Tdi Cr Mt Plus (make ...	Delhi	Skoda	Rapid	2013	Diesel	58000.0	450000.0
20001	Renault Duster 110 Ps Rxz Diesel Plus (make Ye...	Delhi	Renault	Duster	2014	Diesel	50000.0	875000.0
20002	Mahindra Scorpio S10 (make Year 2015) (diesel)	Delhi	Mahindra	Scorpio	2015	Diesel	12000.0	1365000.0
20003	Ford Figo (make Year 2011) (diesel)	Delhi	Ford	Figo	2011	Diesel	36000.0	275000.0
20004	Maruti Suzuki Swift Lxi 1.2 Bs-iv (make Year 2...	Delhi	Maruti Suzuki	Swift	2011	Petr	NaN	NaN

20005 rows × 8 columns

Target Variable: Target Variable is the Selling price of used car.

Data Preprocessing

In order to get a better understanding of the data, we plotted distribution of data. We noticed that the dataset had some outliers, which was negligible. Typically, models that are the latest year and have low mileage sell for a premium, however, there were many data points that did not conform to this. This is because accident history and condition can have a significant effect on the car's price.

Finding null Values.

```
#finding null values in the database
df.isnull().sum()
```

```
DESCRIPTION     0
LOCATION        0
MANUFACTURER    0
MODEL           1
YEAR            0
FUEL TYPE       0
KMS DRIVEN      1
PRICE           1
dtype: int64
```

can see only 3 null values in the entire dataset, Which is negligible.

```
#Removing nan values.
df.dropna(how='any',axis=0,inplace=True)
```

```
#Finding Correlation of the variables
df.corr()
```

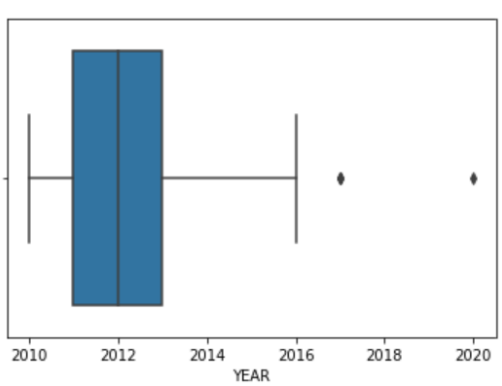
9]:

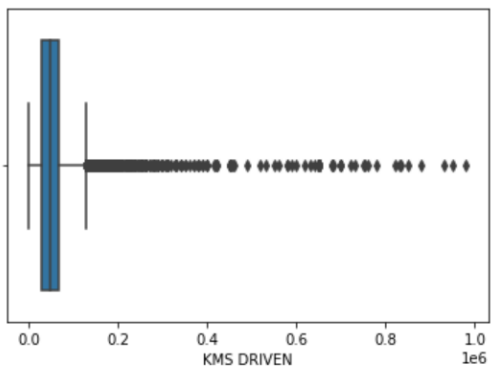
	YEAR	KMS DRIVEN	PRICE
YEAR	1.000000	-0.266563	0.207416
KMS DRIVEN	-0.266563	1.000000	-0.042968
PRICE	0.207416	-0.042968	1.000000

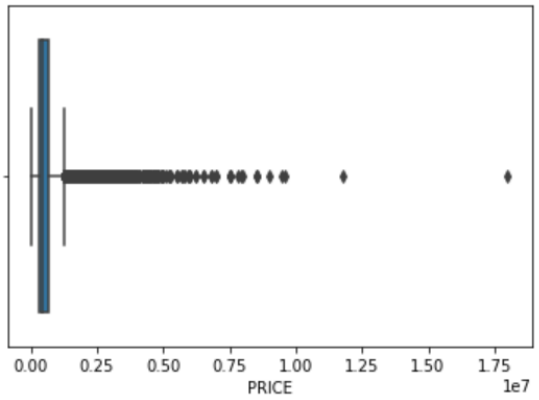
Year has positive correlation with Price.
Kms Driven has negative correlation over Price.

Checking For Outliers:

```
# Cheacking whether the columns has outliers or not
for i in df.describe().columns:
    sns.boxplot(df[i])
    plt.show()
```



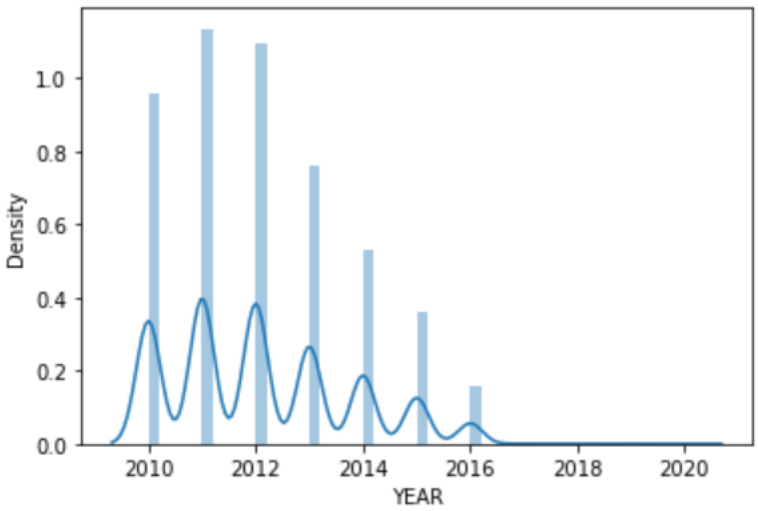


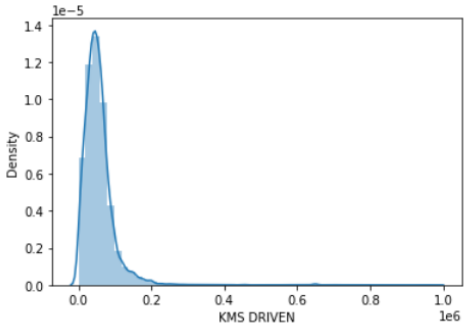


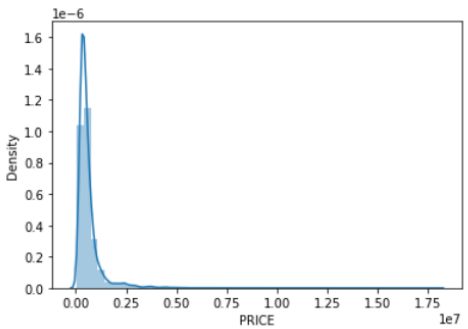
CHECKING FOR SKEWNESS

```
#checking wheather the columns are normally distributed or not

for i in df.describe().columns:
    sns.distplot(df[i])
    plt.show()
```








```
#Skewness Values
df.skew()
```

```
2]: YEAR          0.546710
    KMS DRIVEN    6.541824
    PRICE         5.506024
    dtype: float64
```

KMS DRIVEN and PRICE are highly skewed data (Positively Skewed).

SQUARE ROOT TRANSFORM TO REDUCE SKEWNESS

```
df['KMS DRIVEN']=np.sqrt(df['KMS DRIVEN'])
```

```
df['KMS DRIVEN'].skew()
```

```
54]: 1.2463101889676276
```

```
df['PRICE']=np.sqrt(df['PRICE'])
```

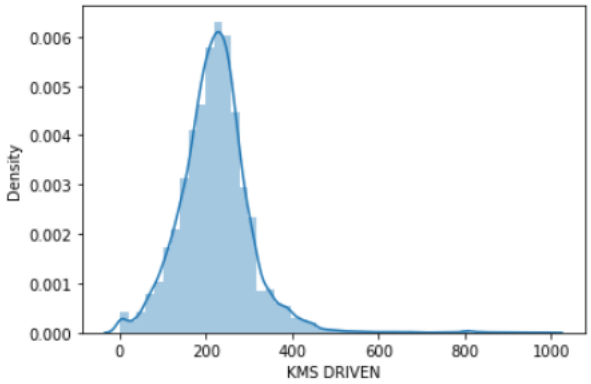
```
df['PRICE'].skew()
```

```
56]: 2.20437436465942
```

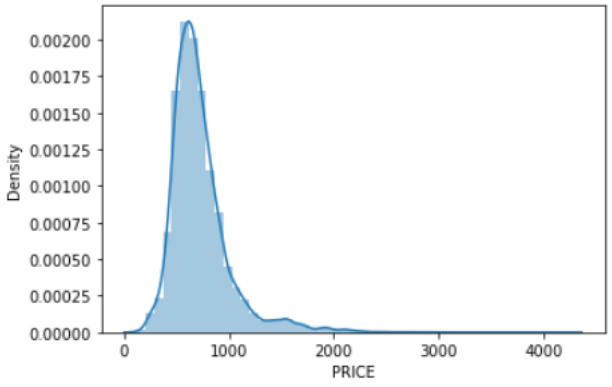
The skew coefficient of KMS DRIVEN went from 6.5 to 1.24 and PRICE went from 5.5 to 2.2, which still is a notable difference. However, the Square root transformation ended with better results.

Data After Normalization:

```
# Distribution plot after Transformation.
sns.distplot(df['KMS DRIVEN']);
```




```
sns.distplot(df['PRICE']);
```



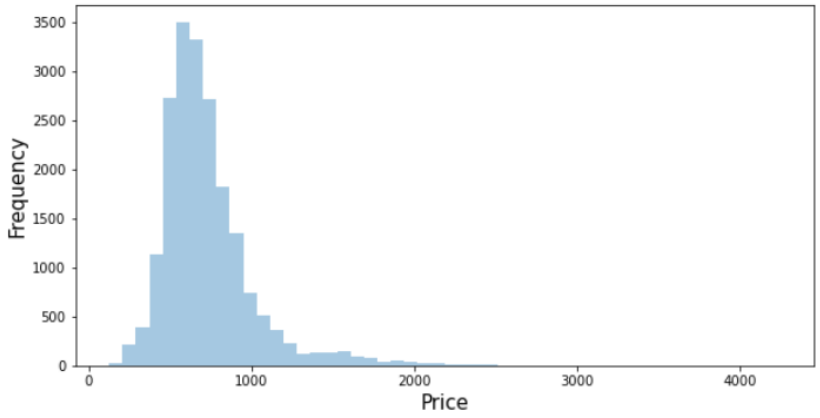
More normalized curve had obtained after transformation

.

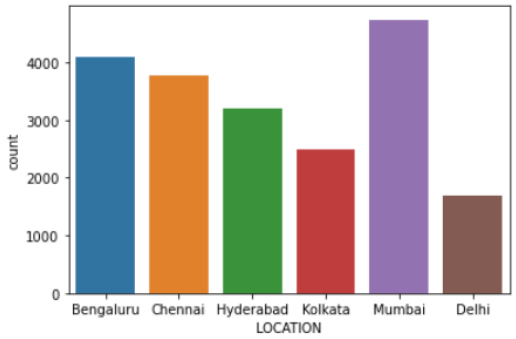
Data Visualization and Analysis

Plotting target variable:

```
# Target Variable
plt.figure(figsize=(10,5))
sns.distplot(df["PRICE"],bins=50,kde=False )
plt.xlabel("Price", Size=15)
plt.ylabel("Frequency", Size=15)
plt.show()
```

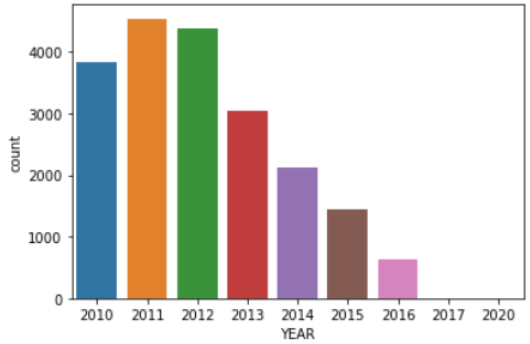



```
# UNIVARIATE ANALYSIS-COUNTPLOT OF LOCATION.
sns.countplot(x='LOCATION',data=df)
plt.show()
```



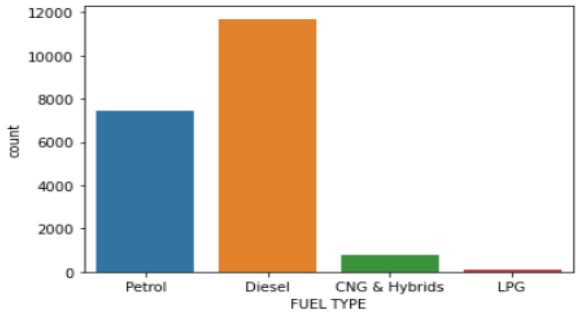
Car availability is higher at Mumbai and lowest at Delhi.
Bengaluru and chennai has comparable values, followed by Hyderabad and kolkata.

```
# UNIVARIATE ANALYSIS-COUNTPLOT OF YEAR.
sns.countplot(x='YEAR',data=df)
plt.show()
```



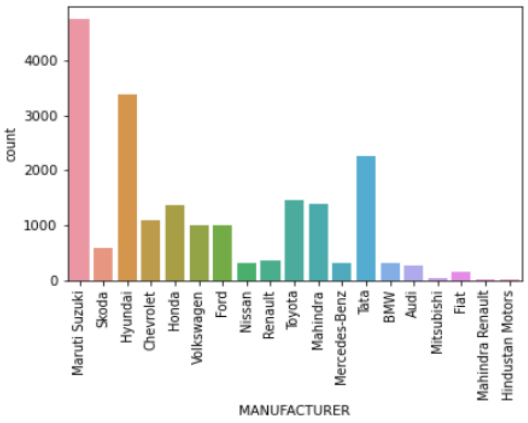
2011 and 2012 are the most sold models, more than 4000 cars.
least availabe are the latest models 2015 and 2016.

```
# UNIVARIATE ANALYSIS-COUNTPLOT OF FUEL TYPE.
sns.countplot(x='FUEL TYPE',data=df)
plt.show()
```



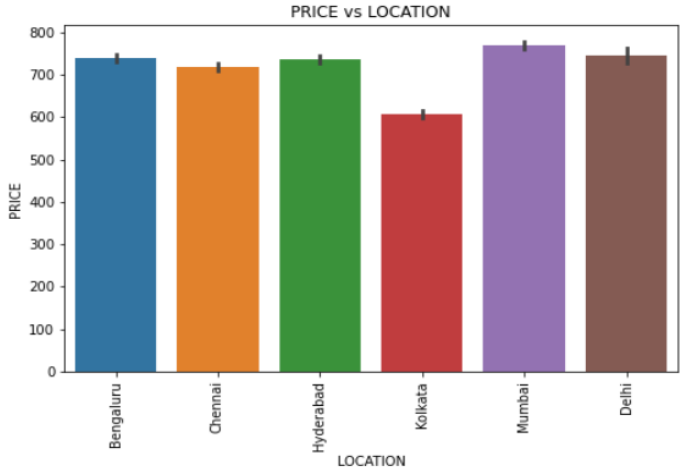
Diesel cars are sold most though fuel price is lower compared to petrol.
CNG and LPG cars are least available in market.

```
# UNIVARIATE ANALYSIS-COUNTPLOT OF FUEL TYPE.
sns.countplot(x='MANUFACTURER',data=df)
plt.xticks(rotation=90)
plt.show()
```



Maruti Suzuki is the highest available model in market more than 4500 cars available.
Hyundai is second in line with more than 3500 cars followed by Tata numbers closed to 3000.
All others are very less available.

```
#Relation with target variables.
plt.figure(figsize=(8,5))
sns.barplot(y="PRICE", x="LOCATION", data=df)
plt.xticks(rotation=90)
plt.xlabel('LOCATION')
plt.ylabel('PRICE')
plt.title('PRICE vs LOCATION');
```

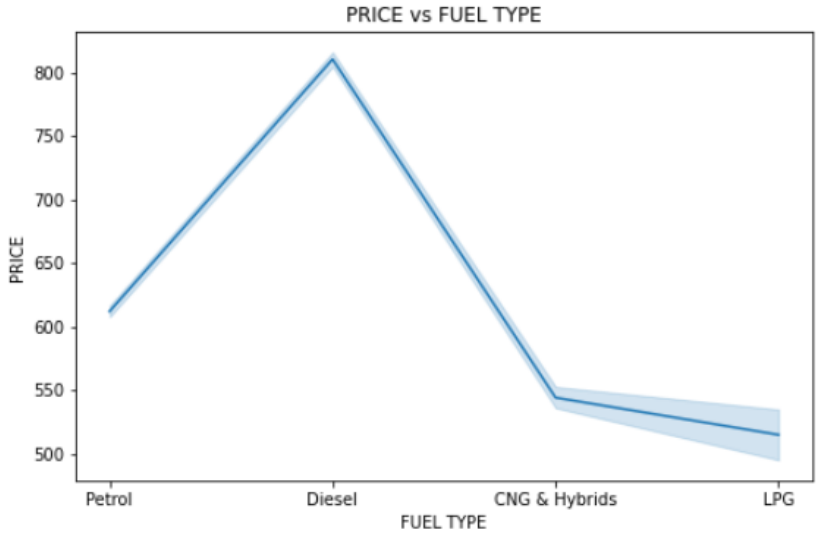


Most of the regions have approximately same price for cars except in kolkata is a little low.

```
#Relation with target variables.
plt.figure(figsize=(8,5))
sns.lineplot(y="PRICE", x="YEAR", data=df)
plt.xlabel('YEAR')
plt.ylabel('PRICE')
plt.title('PRICE vs YEAR');
```

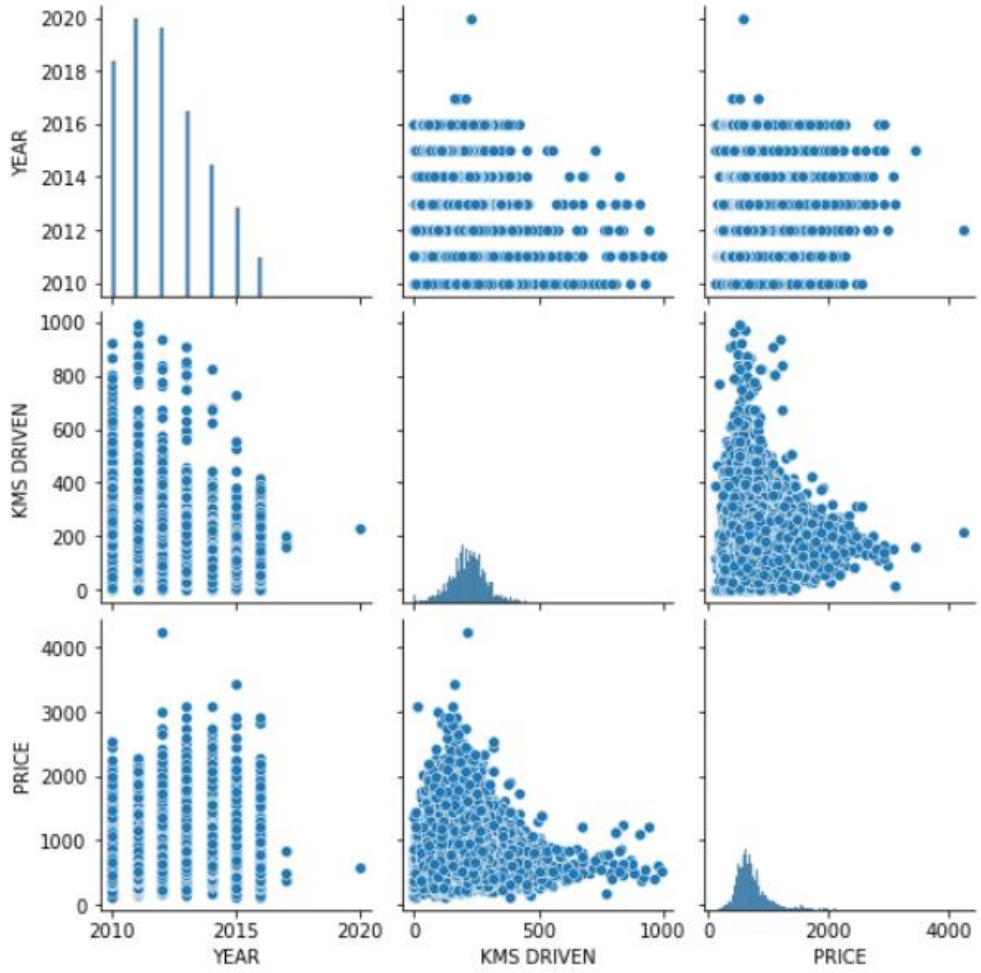
Price is higher for 2014 and 2016 models.

```
#Relation with target variables.
plt.figure(figsize=(8,5))
sns.lineplot(y="PRICE", x="FUEL TYPE", data=df)
plt.xlabel('FUEL TYPE')
plt.ylabel('PRICE')
plt.title('PRICE vs FUEL TYPE');
```



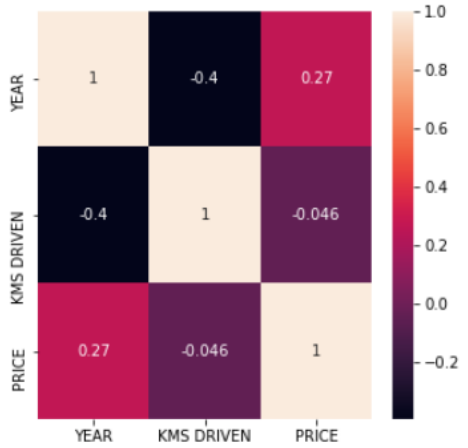
Price is higher for diesel vehicles whereas average for petrol and lowest for CNG and LPG.

```
#Pair plot of numeric values
sns.pairplot(df);
```




```
#Observing corelation between the columns through heatmap
##We observe positive correlation between AveragePrice and year and Kms Driven is negatively corelated

plt.figure(figsize=(5,5))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



DATA PREPARATION

We've also seen that variables are in the correct format, except "symboling", which should rather be a categorical variable (so that dummy variable are created for the categories).

Let's now prepare the data for model building.
Split the data into X and y.

Data Preparation

```
#Removing unwanted column
df= df.drop('DESCRIPTION', axis=1)
```

Train Test Split

```
#SPLITTING AS TEST AND TRAIN DATA.
y= df['PRICE']
x= df.drop('PRICE', axis=1)
```

Creating dummy variables for categorical variables.

```
# subset all categorical variables
df_categorical = x.select_dtypes(include=['object'])
# convert into dummies
df_dummies = pd.get_dummies(df_categorical, drop_first=True)
# drop categorical variables
x = x.drop(list(df_categorical.columns), axis=1)
# concat dummy variables with x
x = pd.concat([x, df_dummies], axis=1)
```

Scaling the features and getting the final list of columns in dataframe for model building.

```
# storing column names in cols, since column names are (annoyingly) lost after
# scaling (the df is converted to a numpy array)
cols = x.columns
x = pd.DataFrame(scale(x))
x.columns = cols
x.columns
```

```
Index(['YEAR', 'KMS DRIVEN', 'LOCATION_Chennai', 'LOCATION_Delhi',
       'LOCATION_Hyderabad', 'LOCATION_Kolkata', 'LOCATION_Mumbai',
       'MANUFACTURER_BMW', 'MANUFACTURER_Chevrolet', 'MANUFACTURER_Fiat',
       ...
       'MODEL_Xuv500', 'MODEL_Xylo', 'MODEL_Yeti', 'MODEL_Z4', 'MODEL_Zen',
       'MODEL_Zen Estilo', 'MODEL_Zest', 'FUEL TYPE_Diesel', 'FUEL TYPE_LPG',
       'FUEL TYPE_Petrol'],
      dtype='object', length=232)
```

Final Train-Test split of data.

```
# split into train and test
x_train, x_test, y_train, y_test = train_test_split(x, y,test_size =0.1, random_state=5)
```

Model Building and Evaluation:

Building the first model with all the features.

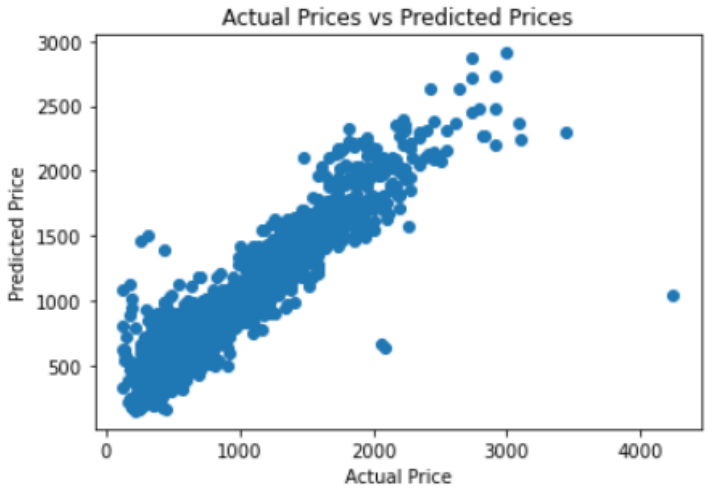
1.Linear Regression

```
lm = LinearRegression()
# fit
lm.fit(x_train, y_train)
# predict
y_pred = lm.predict(x_test)
train_data_pred= lm.predict(x_train)
# metrics
from sklearn.metrics import r2_score
print(r2_score(y_true=y_test, y_pred=y_pred))
```

```
0.9115650876413403
```

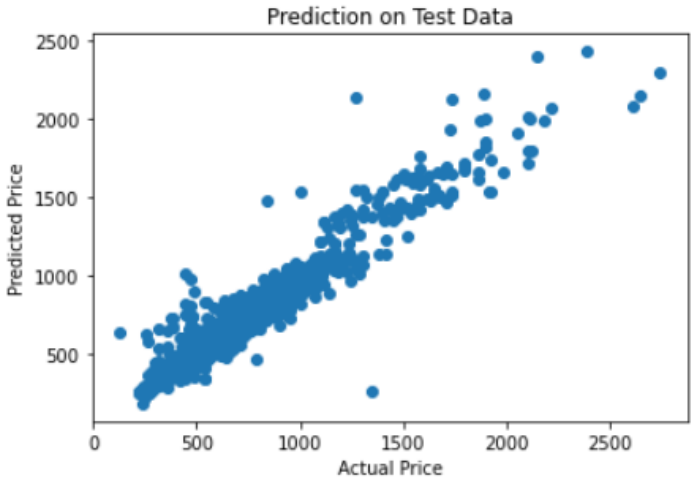
Visualizing Actual Prices and Predicted Prices.

```
plt.scatter(y_train, train_data_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices");
```



Most of the values we can see here are closer to each other.

```
#Evaluating model on basis of test data
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Prediction on Test Data");
```



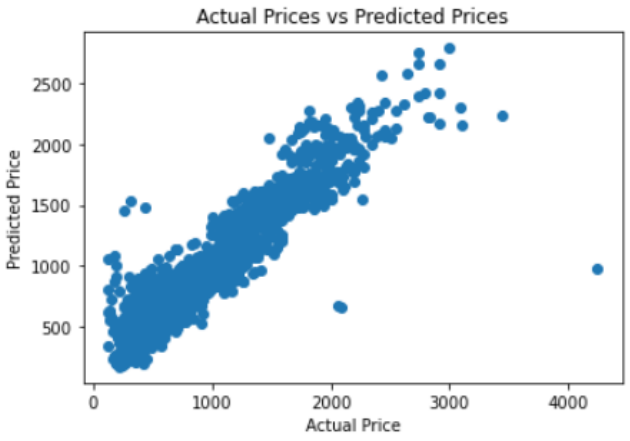
Second Model:

2.Lasso Regression. ¶

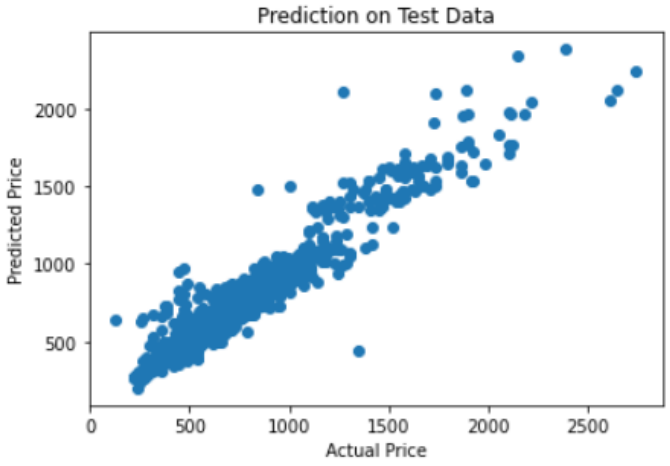
```
ls = Lasso()
# fit
ls.fit(x_train, y_train)
# predict
y_pred1 = ls.predict(x_test)
train_data_pred1= ls.predict(x_train)
print(r2_score(y_true=y_test, y_pred=y_pred1))
```

```
0.9111059255458736
```

```
plt.scatter(y_train, train_data_pred1)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual Prices vs Predicted Prices");
```



```
#Evaluating model on basis of test data
plt.scatter(y_test, y_pred1)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Prediction on Test Data");
```



Values are slightly closer than Linear Regressor.

Saving The Best Model ¶

```
import pickle
# save the model to disk
filename = 'finalized_model_ls.pkl'
pickle.dump(ls,open(filename,'wb'))
#Load the model from disk
loaded_model= pickle.load(open(filename,'rb'))
loaded_model.predict(x_test)
```

```
3]: array([666.45951927, 675.96851202, 659.89825027, ..., 653.67304576,
       508.09058153, 688.08974107])
```

Hardware and Software Requirements and Tools Used

Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

1. Lenovo flex corei5 laptop

2. Jupyter Notebook :The Jupyter Notebook is an interactive environment for running code in the browser. It is a great tool for exploratory data analysis and is widely used by data scientists.

3. MS PowerPoint: For preparing presentation of project.

4. MS word: For preparing report

5. Pandas: is a software library written for the Python programming language for data manipulation and analysis.

6. Numpy: is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

7. Matplotlib: is a plotting library for the Python programming language and its numerical mathematics extension

8. NumPy.Seaborn: is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

9. Sklearn: Scikit-learn (formerly scikits. learn and also known as sklearn) is a free software machine learning library for the Python programming language.

10. LabelEncoder: In label encoding in Python, we replace the categorical value with a numeric value between 0 and the number of classes minus 1. If the categorical variable value contains 5 distinct classes, we use (0, 1, 2, 3, and 4).

CONCLUSION

In this paper we have evaluated the factors influencing Price of used cars. We have evaluated the major features contributing Price of cars. Though this is the simplest model we've built till now, the final predictors still seem to have high correlations. Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across cities.

Linear Regression was chosen as the first model due to its simplicity and comparatively small training time. The features, without any feature mapping, were used directly as the feature vectors. No regularization was used since the results clearly showed low variance. Compared to Linear Regression, Lasso Regressor perform slightly well.