NAME OF THE PROJECT

Car Price Prediction

Submitted by:

Chandan Kumar

ACKNOWLEDGMENT:

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

INTRODUCTION

Problem statement

The main aim of this project is to predict the price of used car. 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. They are looking for new machine learning models from new data. We have to make car price valuation model.

This dataset contains information about used cars. The columns in the given dataset are as follows:

Car_Name

Year

Selling_Price

Present_Price

Kms_Driven

Fuel_Type

Seller_Type

Transmission

Owner

The data consists of records of roughly 301 cars and 9 features.

All the Lifecycle in A Data Science Project is divided into Six parts:

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

Exploratory Data Analysis

Now, let's start with the task of machine learning to predict car prices using Regression. I will start by importing all the necessary libraries that we need for this task and import the dataset.

Importing libraries

Importing the dataset

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.linear_model import LinearRegression
import seaborn as sns
sns.set()
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
```

```
carprice = pd.read_csv('Car Price.csv')
```

The first thing that we can do when tackling a data science problem is getting an understanding of the dataset that you are working with. Key observations and trends in the data were noted down. All correlations within the variables and the output '*selling_price*' were monitored.

```
carprice.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_Type    301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

```
carprice.describe()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

```
carprice.head(10)
```

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0
5	vitara brezza	2018	9.25	9.83	2071	Diesel	Dealer	Manual	0
6	ciaz	2015	6.75	8.12	18796	Petrol	Dealer	Manual	0
7	s cross	2015	6.50	8.61	33429	Diesel	Dealer	Manual	0
8	ciaz	2016	8.75	8.89	20273	Diesel	Dealer	Manual	0
9	ciaz	2015	7.45	8.92	42367	Diesel	Dealer	Manual	0

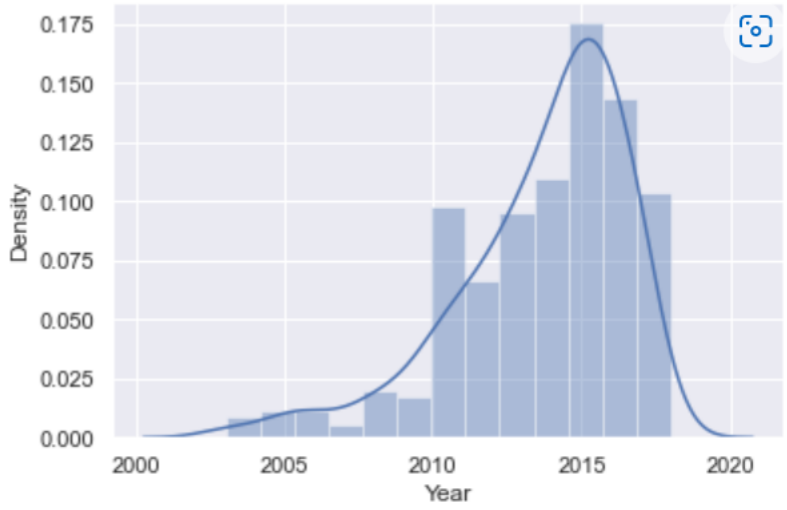
```
carprice.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_Type    301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

You can also use **distplot ()** to fit a parametric distribution to a dataset and visually evaluate how closely it corresponds to the observed data.

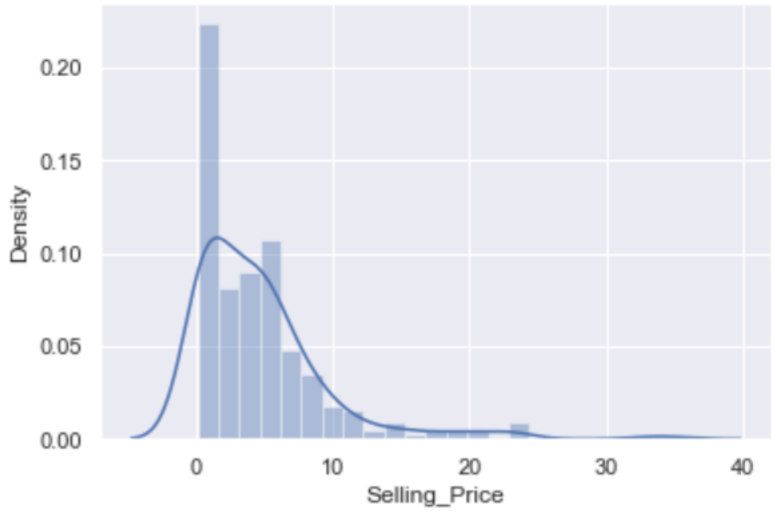
```
sns.distplot(carprice['Year'])
```

```
<AxesSubplot:xlabel='Year', ylabel='Density'>
```



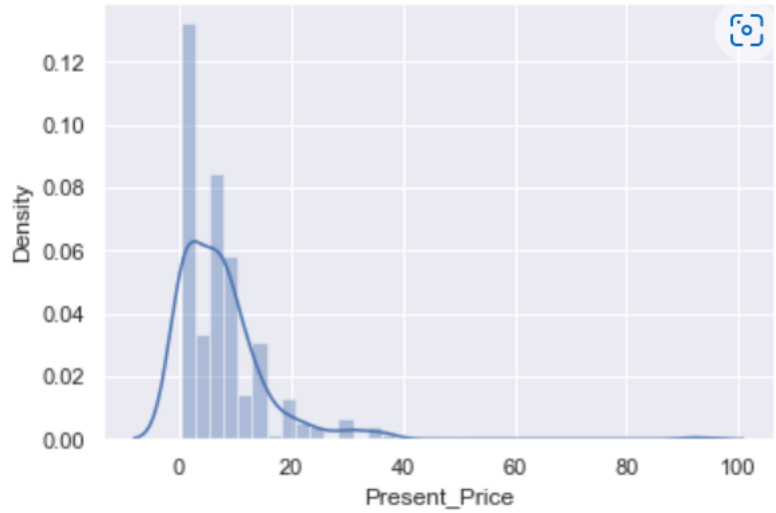
```
sns.distplot(carprice['Selling_Price'])
```

```
<AxesSubplot:xlabel='Selling_Price', ylabel='Density'>
```



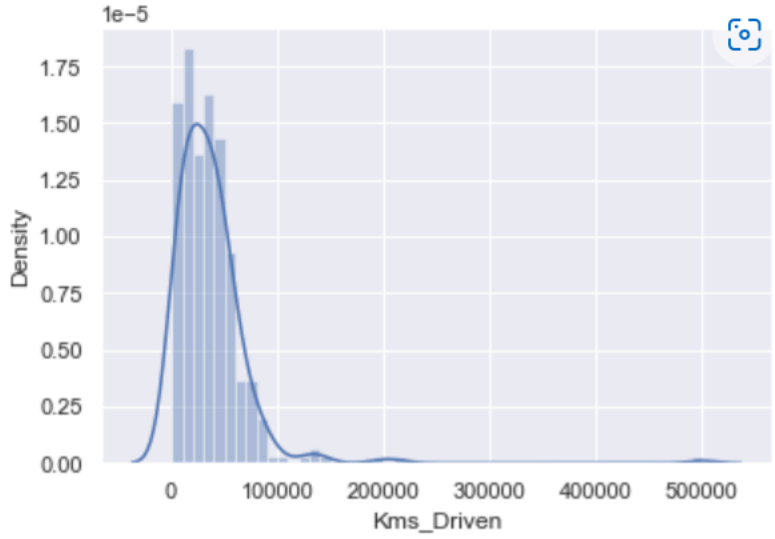

```
sns.distplot(carprice['Present_Price'])
```

```
<AxesSubplot:xlabel='Present_Price', ylabel='Density'>
```



```
sns.distplot(carprice['Kms_Driven'])
```

```
<AxesSubplot:xlabel='Kms_Driven', ylabel='Density'>
```



Scatter plots & Bar plot are used to observe relationships between variables and uses dots to represent the relationship between them. here points are nearly aligned in a line.

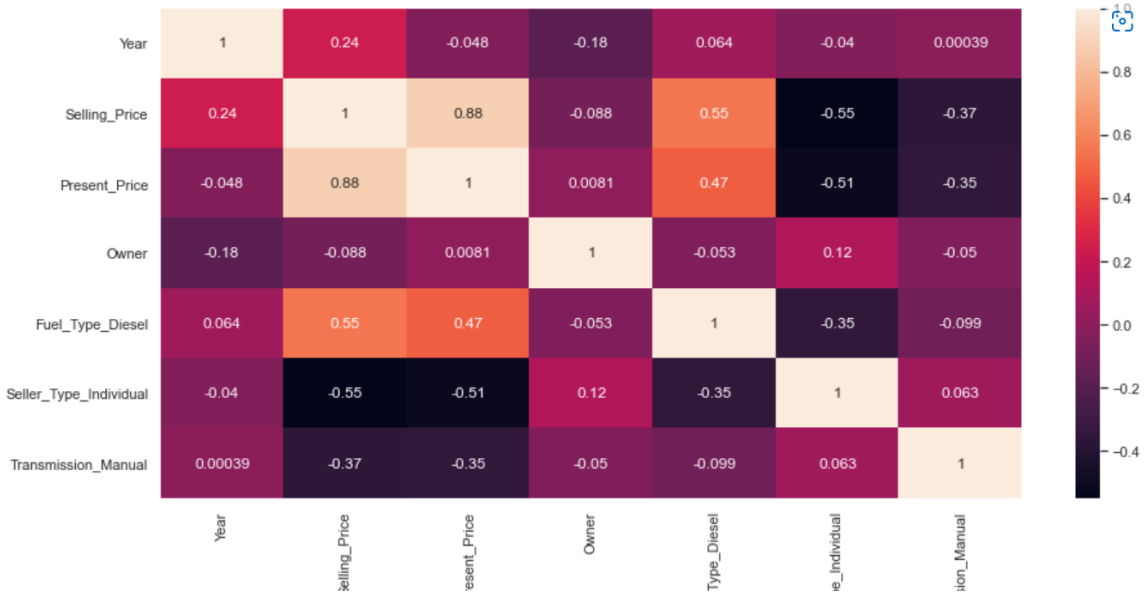
```
plt.figure(figsize=[17,5])
plt.subplot(1,3,1)
sns.barplot(carprice['Seller_Type'], carprice['Selling_Price'])
plt.title('Selling Price Vs Seller Type')

plt.subplot(1,3,2)
sns.barplot(carprice['Transmission'],carprice['Selling_Price'])
plt.title('Selling Price Vs Transmission')

plt.subplot(1,3,3)
sns.barplot(carprice['Fuel_Type'],carprice['Selling_Price'])
plt.title('Selling Price Vs Fuel Type')

plt.show()
```


```
plt.figure(figsize=[17,5])
plt.subplot(1,3,1)
sns.barplot(carprice['Owner'],carprice['Selling_Price'])
plt.subplot(1,3,3)
sns.regplot(carprice['Selling_Price'],carprice['Kms_Driven'])
plt.show()
```


```
plt.figure(figsize=[17,5])
plt.subplot(1,3,1)
sns.regplot(carprice['Selling_Price'],carprice['Present_Price'])

plt.subplot(1,3,2)
sns.distplot(np.log(carprice['Selling_Price']))
plt.title('Distribution of Selling Price')


plt.subplot(1,3,3)
sns.distplot(np.log(carprice['Kms_Driven']))
plt.title('Distribution of KMS Drived')


plt.title('Kilometers Drived')
plt.show()
```

User to heat map.

```
plt.figure(figsize=[15,7])
sns.heatmap(data_no_multicolinearity.corr(), annot=True)
```



Create a multiple Model.

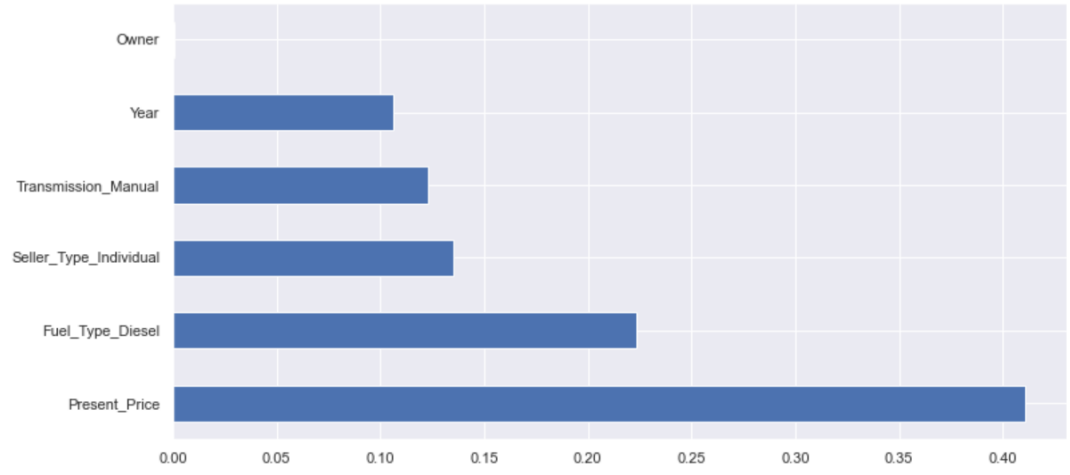
```
from sklearn.feature_selection import f_regression,SelectKBest

X = data_no_multicolinearity.drop('Selling_Price',axis=1)
y = data_no_multicolinearity['Selling_Price']

f_regression(X,y)

p_values = f_regression(X,y)[1]

p_values.round(3)
```

```
array([0.   , 0.   , 0.126, 0.   , 0.   , 0.   ])
```

```
model_summry = pd.DataFrame(data=['Present_Price', 'Owner', 'Year', 'Fuel_Type_Diesel',
                                  'Seller_Type_Individual', 'Transmission_Manual'], columns=['Features'])
model_summry['p-values'] = p_values.round(3)
model_summry.head()
```

	Features	p-values
0	Present_Price	0.000
1	Owner	0.000
2	Year	0.126
3	Fuel_Type_Diesel	0.000
4	Seller_Type_Individual	0.000

```
plt.figure(figsize=[12,6])
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(6).plot(kind='barh')
plt.show()
```



```
print(feat_importances.sort_values(ascending=False))
```

```
Present_Price              0.410476
Fuel_Type_Diesel           0.223630
Seller_Type_Individual     0.135542
Transmission_Manual        0.122875
Year                       0.106579
Owner                      0.000899
dtype: float64
```

```
data_no_multicolinearity.columns
```

```
Index(['Year', 'Selling_Price', 'Present_Price', 'Owner', 'Fuel_Type_Diesel',
       'Seller_Type_Individual', 'Transmission_Manual'],
      dtype='object')
```

```
final_df = data_no_multicolinearity[['Selling_Price', 'Present_Price',
       'Fuel_Type_Diesel', 'Seller_Type_Individual', 'Transmission_Manual']]
```

```
final_df.head()
```

	Selling_Price	Present_Price	Fuel_Type_Diesel	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	0	0	1
1	4.75	9.54	1	0	1
2	7.25	9.85	0	0	1
3	2.85	4.15	0	0	1
4	4.60	6.87	1	0	1

```
X = final_df.drop('Selling_Price', axis=1)
y = final_df['Selling_Price']
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X[['Present_Price']])
```

```
StandardScaler()
```

```
input_scaled = scaler.transform(X[['Present_Price']])
scaled_data = pd.DataFrame(input_scaled, columns=['Present_Price'])
```

```
X_scaled =scaled_data.join(X.drop(['Present_Price'],axis=1))
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.2, random_state=365)
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()


lr.fit(x_train,y_train)


y_pred_lr = lr.predict(x_test)

r_squared = r2_score(y_test,y_pred_lr)
rmse = np.sqrt(mean_squared_error(y_test,y_pred_lr))
print("R_squared :",r_squared)
```

```
R_squared : 0.7703317276495821
```

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()


rf.fit(x_train,y_train)


y_pred_rf = rf.predict(x_test)

r_squared = r2_score(y_test,y_pred_rf)
rmse = np.sqrt(mean_squared_error(y_test,y_pred_rf))
print("R_squared :",r_squared)
```

```
R_squared : 0.778292195767766
```

```
from sklearn.ensemble import GradientBoostingRegres
gbt = GradientBoostingRegressor()


gbt.fit(x_train,y_train)


y_pred_gbt = gbt.predict(x_test)

r_squared = r2_score(y_test,y_pred_gbt)
rmse = np.sqrt(mean_squared_error(y_test,y_pred_gbt
print("R_squared :",r_squared)
```

```
R_squared : 0.7433223454396836
```

```
rf_random.best_params_
```

```
{'n_estimators': 1100,
 'min_samples_split': 10,
 'min_samples_leaf': 2,
 'max_features': 'sqrt',
 'max_depth': 15}
```

```
predictions=rf_random.predict(x_test)
```

```
r_squared = r2_score(y_test,predictions)
rmse = np.sqrt(mean_squared_error(y_test,predictions))
print("R_squared :",r_squared)
```

```
R_squared : 0.7321981215240407
```

Conclusions:

Present price of a car plays an important role in predicting Selling Price, One increases the other gradually increases.

Selling Price of cars with Fuel Type Diesel is higher.

Car of Manual type is of less priced whereas of Automatic type is high.

Cars sold by Individual tend to get less Selling Price when sold by Dealers.