Predicting Car Price using Machine Learning

Problem Description:

There is an automobile company XYZ from Japan which aspires to enter the US market by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts.

They want to understand the factors affecting the pricing of cars in the American market, since those may be very different from the Japanese market. Essentially, the company wants to know:

- Which variables are significant in predicting the price of a car?
- How well those variables describe the price of a car

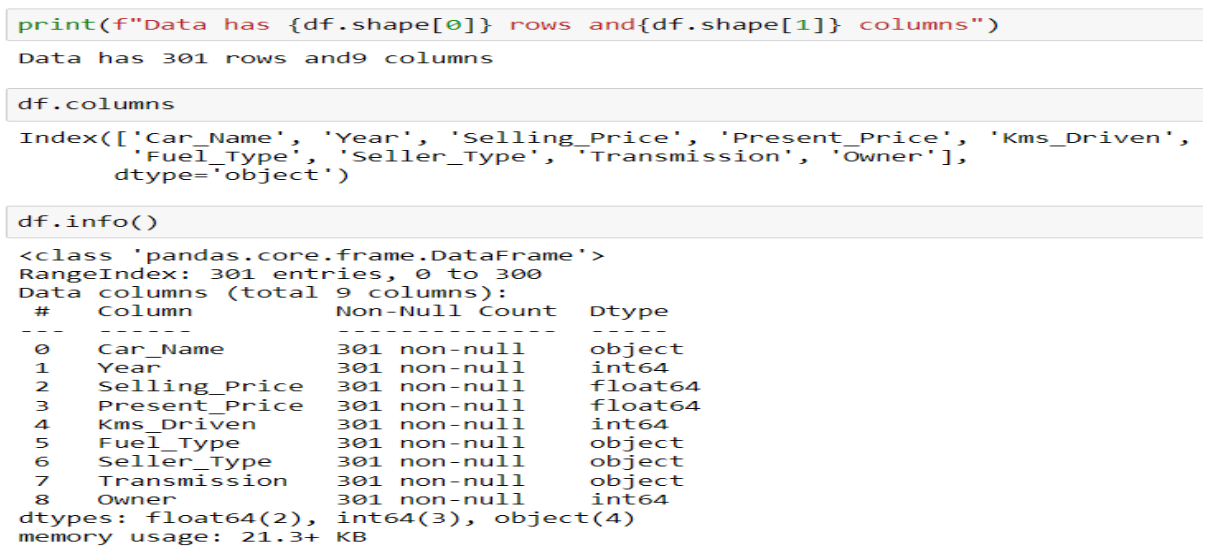
Based on various market surveys, the consulting firm has gathered a large dataset of different types of cars across the American market.

Business Objectives:

You 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.

Data understanding and exploration:

Summary of data: 301 rows, 9 columns, no null values.



```
print(f"Data has {df.shape[0]} rows and{df.shape[1]} columns")
```

```
Data has 301 rows and9 columns
```

```
df.columns
```

```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

```
df.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
```

The column “**Owner**” is the target variable and rest of the columns are independent variables.

The independent variables are again divided into Categorical and Numerical variables.

```
]: numCol=[]
   catCol=[]

   for col in df.columns:
       if df[col].dtype=='O':
           catCol.append(col)
       else:
               numCol.append(col)
```

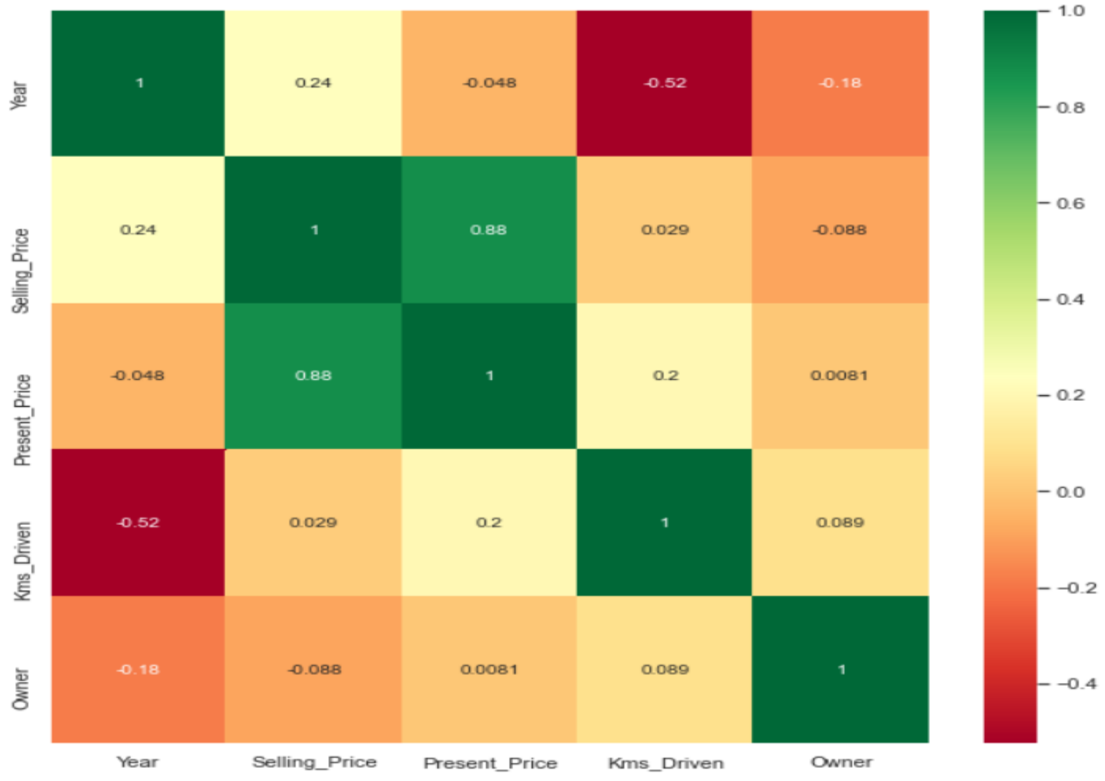
```
]: print("List of categorical columns:",catCol)
   print("List of numerical columns:",numCol)

   List of categorical columns: ['Car_Name', 'Fuel_Type', 'Seller_Type', 'Transmission']
   List of numerical columns: ['Year', 'Selling_Price', 'Present_Price', 'Kms_Driven', 'Owner']
```

Heatmap to show correlation of Numerical and Target variable:

Now let’s plot Heatmap which is pretty useful to visualise multiple correlations among numerical variables. We have also used the Target variable “**Owner**” to understand the correlation of numerical variables with it.

```
<AxesSubplot:>
```



	Year	Selling_Price	Present_Price	Kms_Driven	Owner
Year	1	0.24	-0.048	-0.52	-0.18
Selling_Price	0.24	1	0.88	0.029	-0.088
Present_Price	-0.048	0.88	1	0.2	0.0081
Kms_Driven	-0.52	0.029	0.2	1	0.089
Owner	-0.18	-0.088	0.0081	0.089	1

Data Preparation:

Let’s now prepare the data for model building.

Split the data into X and y.

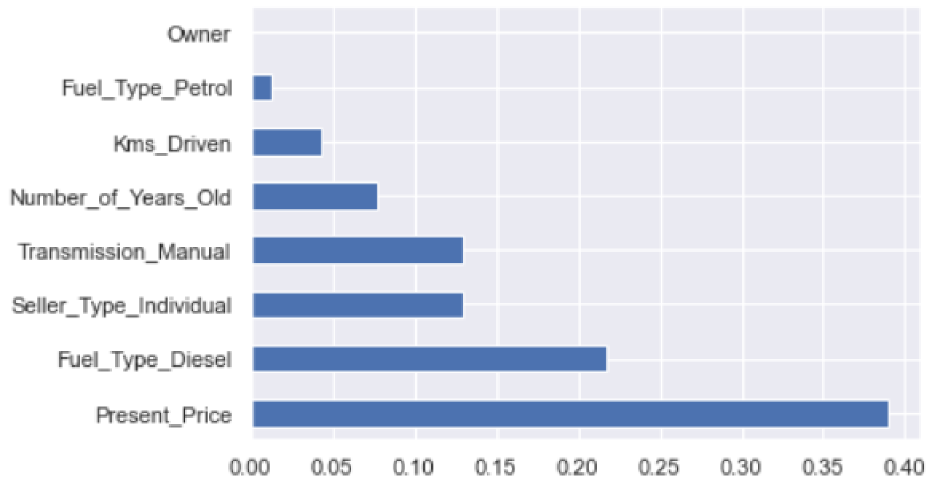
```
X=dataset.iloc[:,:-1]
y=dataset.iloc[:,-1]
```

```
### To determine important features, make use of ExtraTreesRegressor
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
model.fit(X,y)

print(model.feature_importances_)

#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```

```
[0.38913032 0.04358997 0.00085346 0.07716225 0.21793921 0.01200964
 0.13003855 0.12927661]
```



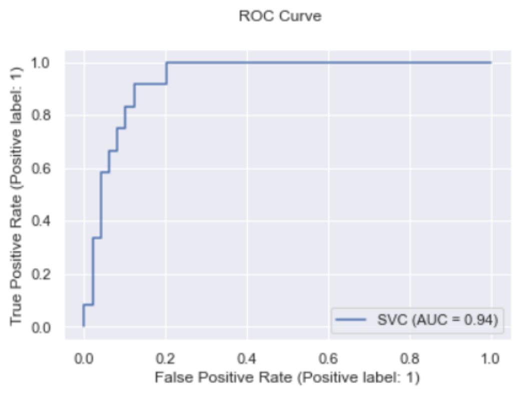
ROC-AUC Curve:

AUC ROC Curve

```
]: Final_Model = SVC(decision_function_shape='ovo', gamma='scale', kernel='rbf', probability=True, random_state=21,
                    shrinking=True, verbose=True)
   Classifier = Final_Model.fit(X_train, Y_train)
   fmod_pred = Final_Model.predict(X_test)
   fmod_acc = (accuracy_score(Y_test, fmod_pred))*100
   print("Accuracy score for the Best Model is:", fmod_acc)
```

```
[LibSVM]Accuracy score for the Best Model is: 88.52459016393442
```

```
]: disp = metrics.plot_roc_curve(Final_Model, X_test, Y_test)
   disp.figure_.suptitle("ROC Curve")
   plt.show()
```

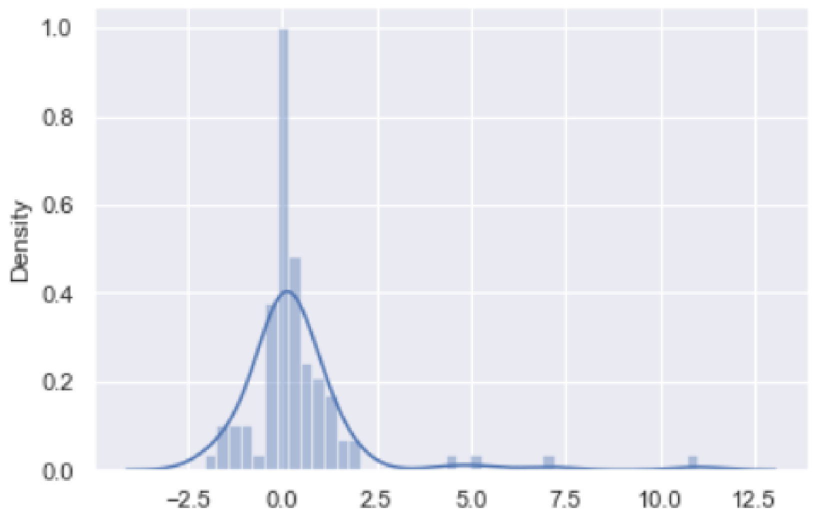


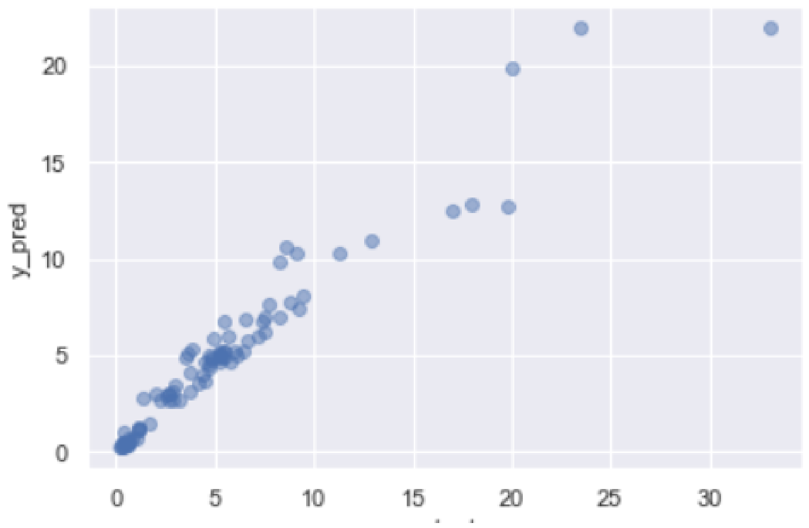
Final Model Evaluation:

Let’s now evaluate the model in terms of its assumptions. We should test that:

- The error terms are normally distributed with mean approximately 0.
- There is little correlation between the predictors.
- Homoscedasticity, i.e. the ‘spread’ or ‘variance’ of the error term (y_true-y_pred) is constant.

```
Voting Regresssor Score on Training set is 0.9774770627437626
Voting Regresssor Score on Test Set is 0.9011856812089695
[0.96999326 0.84164499 0.69207495 0.89624323 0.94134878]
Accuracy: 86.83 %
Standard Deviation: 9.82 %
Mean Absolute Error: 0.8212686079381828
Mean Squared Error: 2.955146621339502
RMSE: 1.7190539902340187
The r2_score is 0.9011856812089695
```





Conclusion:

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).