Machine Learning Project

Table of Contents

1 Task : Predictive Analysis, Supervised Learning

The main objective of task is to predict the cars price with the help of the Linear regression model. To analyze the given cars dataset, follow the below-mentioned steps.

1.1 Data Import and Preparation

Take the important libraries and import them as shown below.

```
#import required libraries
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

Then, import the given cars dataset, as represented in the following screenshot.

```
car_data = pd.read_csv('/kaggle/input/vehicle-dataset-from-cardekho/car data.csv')
```

```
car_data.head()
```

	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

The dataset information is presented as follows.

```
car_data.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
```

From the above result, the given dataset contains 5 rows and 9 columns.

1.2 Data Exploratory Analysis and Transformation

In the data exploratory analysis, the most important features for cars price prediction are selected, as shown below.

```
car_data.columns
```

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

Detect the outliers and null values.

```
car_data.isnull().sum()
```

```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

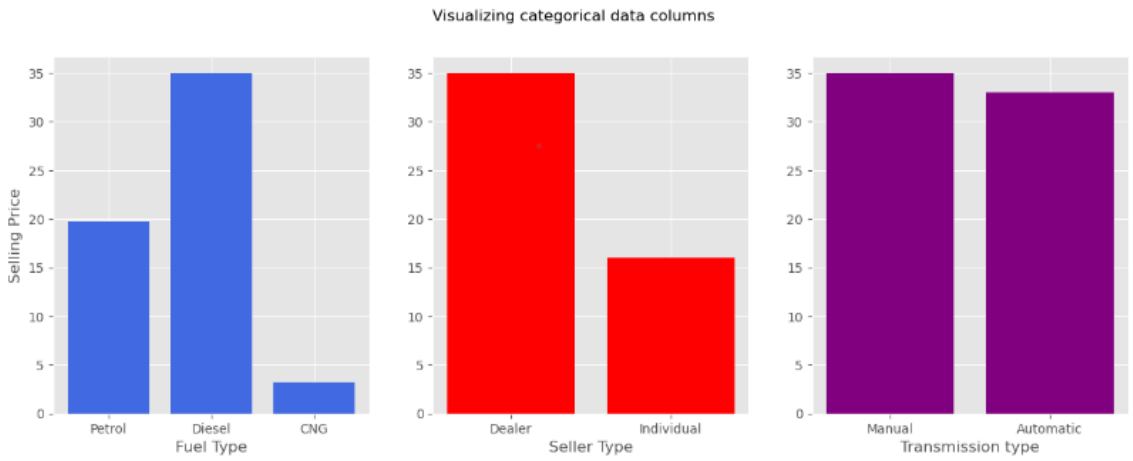
There is no missing Value, all the variables are in numeric form.

Then count the unique values in three specific columns: 'Fuel_Type', 'Seller_Type', and 'Transmission' and print them.

```
print(car_data['Fuel_Type'].value_counts())
print(car_data['Seller_Type'].value_counts())
print(car_data['Transmission'].value_counts())
```

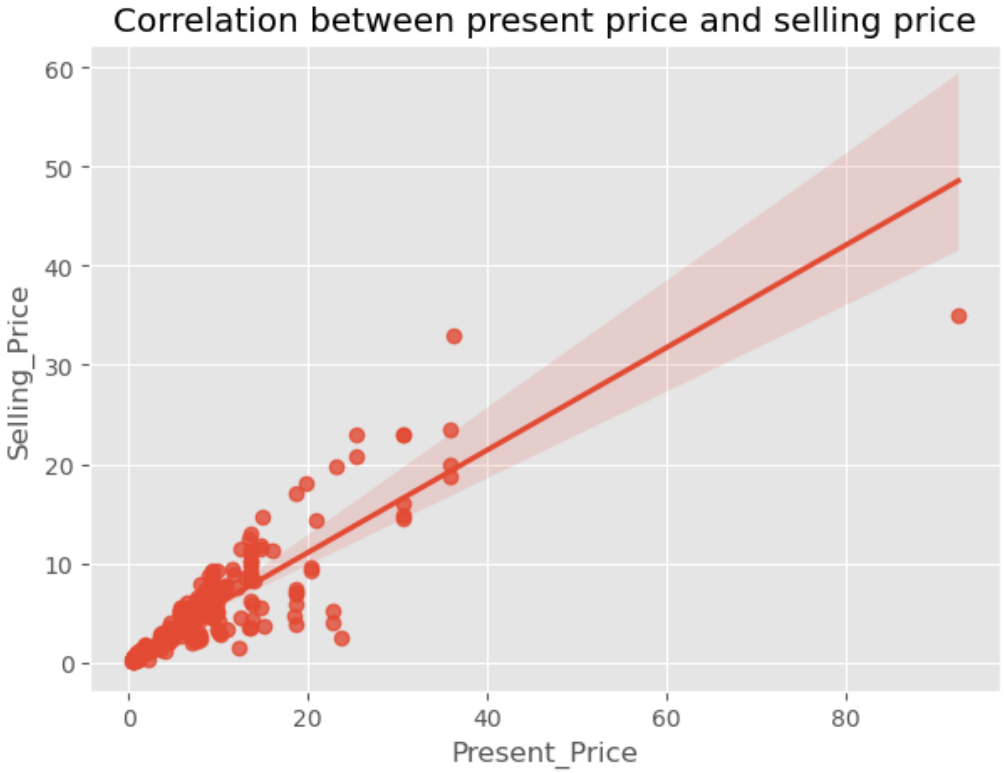
```
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
Dealer        195
Individual    106
Name: Seller_Type, dtype: int64
Manual       261
Automatic     40
Name: Transmission, dtype: int64
```

Visualize of categorical data columns using Matplotlib.



The figure with the three subplots, allowing us to visualize the categorical data columns' relationships with the 'selling_price' variable.

Screenshot of finding the Correlation between present and selling price.



Defining the X features and Y target data.

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

Now, build a model to predict the cars price. Before that, split the dataset into the train (80%) and test (20%) data, as follows:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(X,y, test_size=0.2, random_state=42)
```

1.3 Linear Regression model Analysis

Start linear regression model building:

Import some necessary libraries.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

Then continue building a linear regression model.

```
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
model=LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Model is fit to Run. Then predict the

```
pred = model.predict(X_test)
```

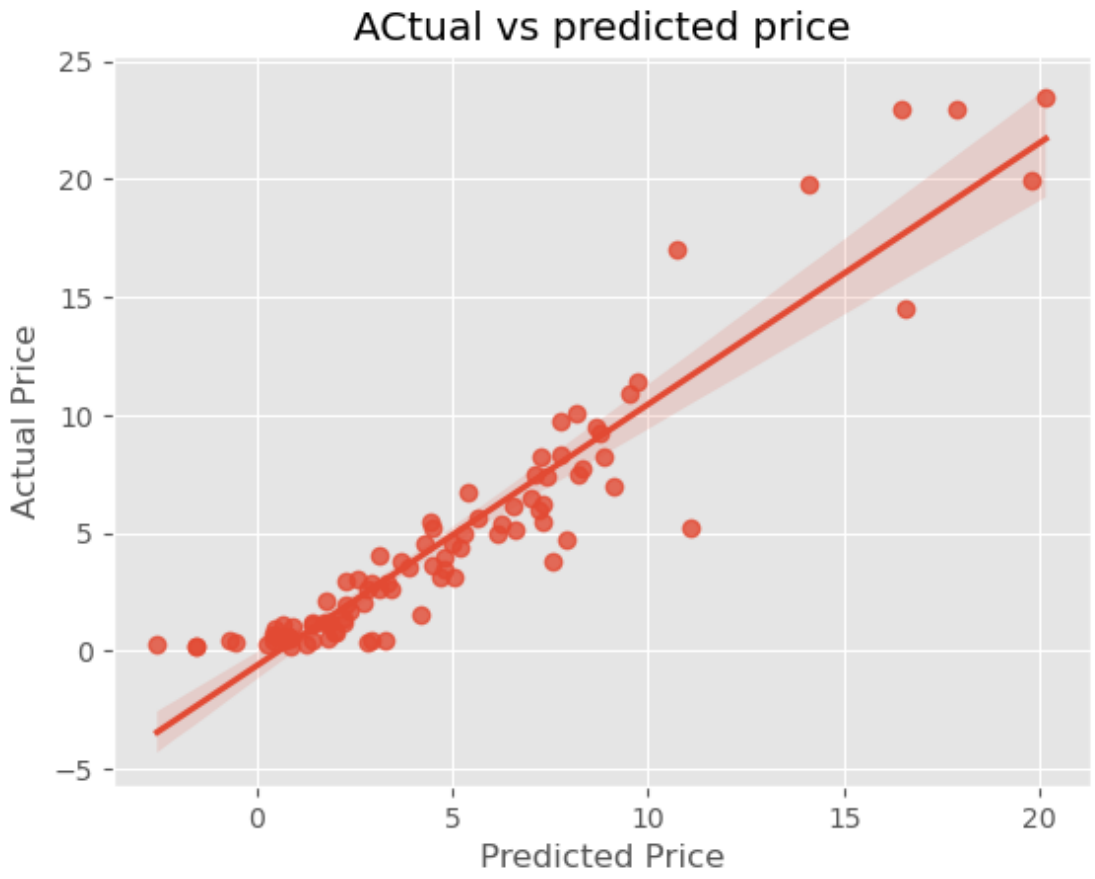
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
print("MAE: ", (metrics.mean_absolute_error(pred, y_test)))
print("MSE: ", (metrics.mean_squared_error(pred, y_test)))
print("R2 score: ", (metrics.r2_score(pred, y_test)))
```

```
MAE:  1.258140470647337
MSE:  3.493286026225147
R2 score:  0.8294933369778817
```

Here, the result shows that a model has been defined for the linear regression model by using the linear model function. Its show model is 80% accurate.

Then plot a graph between predicted price and actual price.



1.4 Result

These results suggest that the model has a relatively low MAE and MSE, indicating that it's making reasonably accurate predictions. The R2 score of approximately 0.8295 also indicates that the model is explaining a substantial portion of the variance in the selling price. However, the interpretation of these metrics can also depend on the specific context and goals of the model.