Project - Used Cars Price Prediciton

So called Second hand's car have a huge market base. Many consider to buy a Used Car intsead of buying of new one, as it's is feasible and a better investment.

The main reason for this huge market is that when you buy a New Car and sale it just another day without any default on it, the price of car reduces by 30%.

There are also many frauds in the market who not only sale wrong but also they could mislead to wrong price.

So, here use this following dataset to Predict the price of used cars.

Multiple Linear Regression

Now you know how to build a model with one X (feature variable) and Y (response variable). But what if you have three feature variables, or may be 10 or 100? Building a separate model for each of them, combining them, and then understanding them will be a very difficult and next to impossible task. By using multiple linear regression, you can build models between a response variable and many feature variables.

Let's see how to do that.

Import necessary libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read car_data.csv and store it in a variable

In [2]:
```
df = pd.read_csv('car_data.csv')
```

View the first five rows

In [3]:
```
df.head()
```

Out[3]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmis
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Ma
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Ma
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Ma
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Ma
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Ma

View the last five rows

In [4]:
```
df.tail()
```

Out[4]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transm
296	city	2016	9.50	11.6	33988	Diesel	Dealer	I
297	brio	2015	4.00	5.9	60000	Petrol	Dealer	I
298	city	2009	3.35	11.0	87934	Petrol	Dealer	I
299	city	2017	11.50	12.5	9000	Diesel	Dealer	I
300	brio	2016	5.30	5.9	5464	Petrol	Dealer	I

Check the database info

In [5]:
```
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
```

Gather the basic statistical information about the dataset

In [6]:
```
df.describe()
```

Out[6]:

	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

Check the shape of the dataframe

In [7]:
```
df.shape
```

Out[7]:
```
(301, 9)
```

Check if there are any null values

In [8]:
```
df.isna().sum()
```

Out[8]:
```
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
```

In [9]:
```
# There seems to be no null values
```

Change the Year to number of years

In [10]:
```
current_year = 2021
df['Year'] = current_year - df['Year']
```

In [11]:
```
df.head()
```

Out[11]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmiss
0	ritz	7	3.35	5.59	27000	Petrol	Dealer	Ma
1	sx4	8	4.75	9.54	43000	Diesel	Dealer	Ma
2	ciaz	4	7.25	9.85	6900	Petrol	Dealer	Ma
3	wagon r	10	2.85	4.15	5200	Petrol	Dealer	Ma
4	swift	7	4.60	6.87	42450	Diesel	Dealer	Ma

Drop Car_Name column

In [12]:
```
df.drop(columns = ['Car_Name'], inplace = True)
```

In [13]:
```
df.head()
```

Out[13]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	7	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	8	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	4	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	10	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	7	4.60	6.87	42450	Diesel	Dealer	Manual	0

Change the categorical variables into numerical by OneHotEncoding (Use pd.get_dummies)

In [14]:
```
newdf = pd.get_dummies(df, drop_first = True)
newdf.head()
```

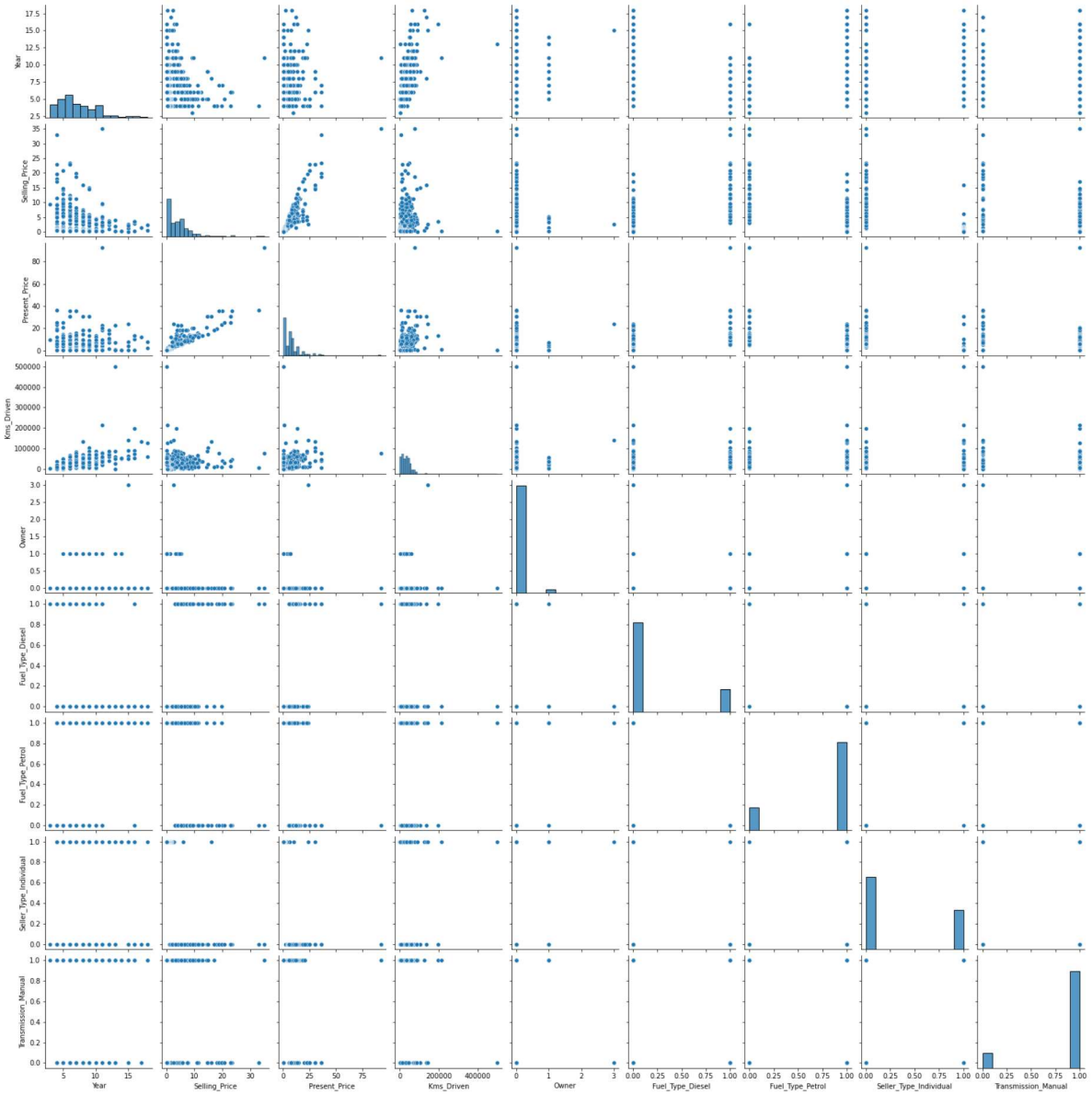
Out[14]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner	Fuel_Type_Diesel	Fuel_Type_Petrol	S
0	7	3.35	5.59	27000	0	0	1	
1	8	4.75	9.54	43000	0	1	0	
2	4	7.25	9.85	6900	0	0	1	
3	10	2.85	4.15	5200	0	0	1	
4	7	4.60	6.87	42450	0	1	0	

Visualization

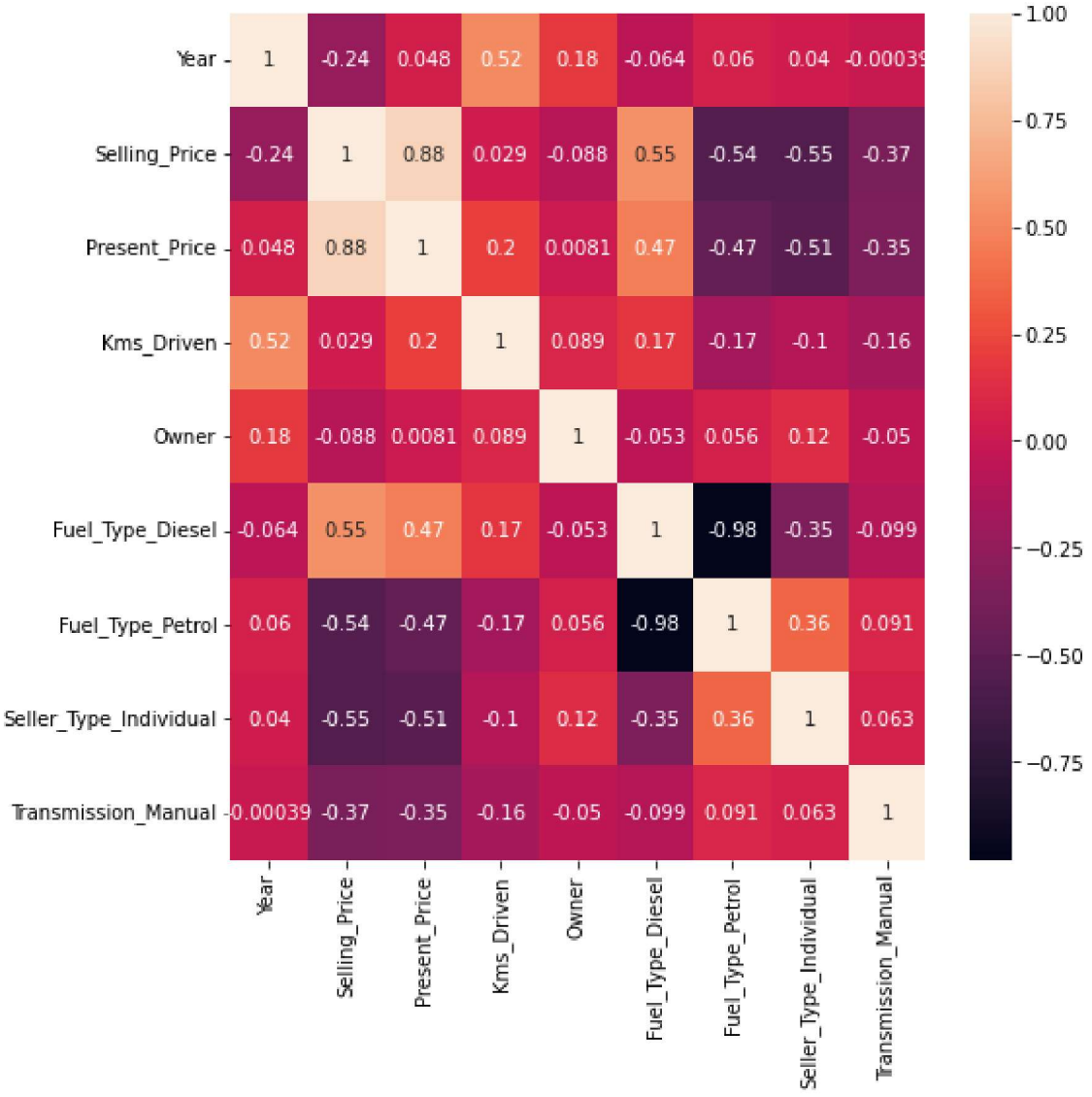
Plot a pair plot

In [15]:
```
sns.pairplot(newdf)
plt.show()
```



Plot a heatmap

In [16]:
```
plt.figure(figsize = (8,8))
sns.heatmap(newdf.corr(), annot = True)
plt.show()
```



Assign X and y variables(independent and dependent variables)

In [17]:
```
X = newdf[['Year', 'Present_Price', 'Kms_Driven', 'Owner',
       'Fuel_Type_Diesel', 'Fuel_Type_Petrol', 'Seller_Type_Individual',
       'Transmission_Manual']]
y = newdf['Selling_Price']
```

Standardise the data using Standard Scaler

In [18]:
```
from sklearn.preprocessing import StandardScaler
```

In [19]:
```
scaler = StandardScaler()
```

In [20]:
```
xcolumns = X.columns.tolist()
```

In [21]:
```
X = scaler.fit_transform(X)
```

In [22]:
```
X = pd.DataFrame(X, columns = xcolumns)
X.head()
```

Out[22]:

	Year	Present_Price	Kms_Driven	Owner	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_T
0	-0.128897	-0.236215	-0.256224	-0.174501	-0.498962	0.509327	
1	0.217514	0.221505	0.155911	-0.174501	2.004162	-1.963374	
2	-1.168129	0.257427	-0.773969	-0.174501	-0.498962	0.509327	
3	0.910335	-0.403079	-0.817758	-0.174501	-0.498962	0.509327	
4	-0.128897	-0.087890	0.141743	-0.174501	2.004162	-1.963374	

Check the shape of X and y

In [23]:
```
X.shape
```

Out[23]:
```
(301, 8)
```

In [24]:
```
y.shape
```

Out[24]:
```
(301,)
```

In [25]:
```
# The X variable must always be a 2 dimensional array and y, a one dimensional
```

Split the data into training and testing set

In [26]:
```
from sklearn.model_selection import train_test_split
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, rand
```

Check the size of X_train and X_test

In [28]:
```
X_train.size
```

Out[28]: 1680

In [29]:
```
X_test.size
```

Out[29]: 728

Create a Linear Regression Model and Train it

In [30]:
```
from sklearn.linear_model import LinearRegression
```

In [31]:
```
model = LinearRegression()
```

In [32]:
```
#Train the model
model.fit(X_train, y_train)
```

Out[32]: LinearRegression()

Check the score of our model

In [33]:
```
model.score(X_train, y_train)
```

Out[33]: 0.895560836468798

Print the values of coefficents

In [34]:
```
pd.DataFrame(model.coef_, index = X_test.columns, columns = ['Coefficeints'])
```

Out[34]:

	Coefficeints
Year	-1.108809
Present_Price	3.484620
Kms_Driven	-0.185312
Owner	0.146933
Fuel_Type_Diesel	0.984041
Fuel_Type_Petrol	0.167000
Seller_Type_Individual	-0.536205
Transmission_Manual	-0.537119

Predict for X_test and store it in a variable

In [35]:
```
y_pred = model.predict(X_test)
```

Evaluate the prediction with mean squared error and r2 score

In [36]:
```
from sklearn import metrics
```

In [37]:
```
print ('Mean Squared Error is: ', metrics.mean_squared_error(y_test, y_pred))
```

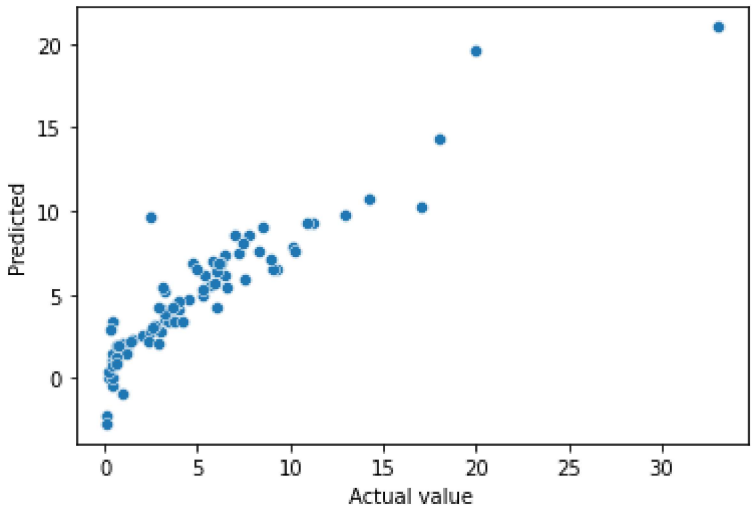
```
Mean Squared Error is:  4.390781949926887
```

In [38]:
```
print('r2 Score is: ', metrics.r2_score(y_test, y_pred) )
```

```
r2 Score is:  0.8316982715837706
```

Plot a scatterplot of actual vs predicted values

In [39]:
```
sns.scatterplot(x = y_test, y = y_pred)
plt.xlabel('Actual value')
plt.ylabel('Predicted')
plt.show()
```



Plot a graph to check the accuracy of our prediction

In [43]:
```
c = [i for i in range(1,92,1)]   #Creating an Index, 61 is used because we hav
fig = plt.figure()
plt.plot(c,y_test, color = 'green') # Plotting y test
plt.plot(c,y_pred, color = 'blue') # Plotting predicted values
fig.suptitle('Actual (Green) Vs Predicted (Blue)') # Set title
plt.show()
```