TASK-3 Car Price Prediction With Machine Learning

**The datasets consist of several independent variables include: 1.Car_Name 2.Year 3.Selling_Price 4.Present_Price 5.Kms_Driven 6.Fuel_Type 7.Seller_Type 8.Transmission 9.Owner

**Modules needed:

pandas: Pandas is an opensource library that allows you to perform data manipulation in Python. Pandas provide an easy way to create, manipulate and wrangle the data.

numpy: Numpy is the fundamental package for scientific computing with Python. numpy can be used as an efficient multi-dimensional container of generic data.

matplotlib: Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of formats.

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

scipy: Scipy is a Python-based ecosystem of open-source software for mathematics, science, and engineering.

In [2]:
```
#importing libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns  #seaborn
import matplotlib.pyplot as plt     #matplotlib
import warnings     # ignore warnings
warnings.filterwarnings("ignore")
import os
```

In [4]:
```
#Loading dataset
df = pd.read_csv('car_price_predicition_data.csv')
```

In [5]:
```
#Displaying dataset
print("Dataset for car price prediction: \n",df)
```

```
Dataset for car price prediction:
     Car_Name  Year  Selling_Price  Present_Price  Kms_Driven Fuel_Type  \
0        ritz  2014           3.35           5.59       27000    Petrol
1         sx4  2013           4.75           9.54       43000    Diesel
2        ciaz  2017           7.25           9.85        6900    Petrol
3    wagon r   2011           2.85           4.15        5200    Petrol
4       swift  2014           4.60           6.87       42450    Diesel
```

```
..       ...   ...           ...           ...         ...       ...
296     city  2016          9.50         11.60       33988    Diesel
297     brio  2015          4.00          5.90       60000    Petrol
298     city  2009          3.35         11.00       87934    Petrol
299     city  2017         11.50         12.50        9000    Diesel
300     brio  2016          5.30          5.90        5464    Petrol

    Seller_Type Transmission  Owner
0        Dealer       Manual      0
1        Dealer       Manual      0
2        Dealer       Manual      0
3        Dealer       Manual      0
4        Dealer       Manual      0
..          ...          ...    ...
296      Dealer       Manual      0
297      Dealer       Manual      0
298      Dealer       Manual      0
299      Dealer       Manual      0
300      Dealer       Manual      0

[301 rows x 9 columns]
```

In [6]:

```
#First 7 rows of dataset
print("Top seven rows of dataset are: ")
df.head(7)
```

```
Top seven rows of dataset are:
```

Out[6]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Tra
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	
5	vitara brezza	2018	9.25	9.83	2071	Diesel	Dealer	
6	ciaz	2015	6.75	8.12	18796	Petrol	Dealer	

In [7]:

```
#Last 5 rows of dataset
#If no number is given then it takes default number as 5
print("Last 5 rows of dataset are: ")
df.tail()
```

```
Last 5 rows of dataset are:
```

Out[7]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	T
296	city	2016	9.50	11.6	33988	Diesel	Dealer	
297	brio	2015	4.00	5.9	60000	Petrol	Dealer	
298	city	2009	3.35	11.0	87934	Petrol	Dealer	

299	city	2017	11.50	12.5	9000	Diesel	Dealer
300	brio	2016	5.30	5.9	5464	Petrol	Dealer

In [8]:
```
#Information of dataset
print("Summarized information of the dataset: \n")
df.info()
```

```
Summarized information of the dataset:

<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
```

In [9]:
```
#Summary statistics of dataset
#describe() function is used which gives all the summary statistics like mean, c
print("Statistics summary of advertisement dataset are: \n")
df.describe()
```

```
Statistics summary of advertisement dataset are:
```

Out[9]:

	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

In [10]:
```
#Dimensions and size information about the dataset
#shape() function gives the dimensions of the dataset
```

```
print("Dimensions of the dataset are: ",df.shape)
print("Size of advertisement dataset are: ",df.size)
```

```
Dimensions of the dataset are:  (301, 9)
Size of advertisement dataset are:  2709
```

In [11]:

```
#List of column names in the dataset
print("Names of columns in sales prediction dataset are: \n",df.columns)
```

```
Names of columns in sales prediction dataset are:
 Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

In [23]:

```
print("Values in the column of type of fuel \n",df.Fuel_Type.value_counts(),"\n"
print("Values in the column of type of sellers \n",df.Seller_Type.value_counts()
print("Values in the column of Transmission values \n",df.Transmission.value_cou
```

```
Values in the column of type of fuel
 Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64

Values in the column of type of sellers
 Dealer        195
Individual    106
Name: Seller_Type, dtype: int64

Values in the column of Transmission values
 Manual       261
Automatic     40
Name: Transmission, dtype: int64
```

In [27]:

```
#Fuel_Type ==> 1 = Petrol , 0 = Diesel , 2 = CNG
#Seller_Type ==> 1 = Manual , 0 = Automatic
#Seller_Type ==> 1 = Dealer , 0 = Individual

print(df.Fuel_Type.replace(regex={"Petrol":"0","Diesel":"1","CNG":"2"},inplace=T
print(df.Seller_Type.replace(regex={"Dealer":"0","Individual":"1"},inplace=True)
print(df.Transmission.replace(regex={"Manual":"0","Automatic":"1"},inplace=True)
#print(df[["Fuel_Type","Seller_Type","Transmission"]]=df[["Fuel_Type","Seller_Ty
```

```
None
None
None
```

In [14]:

```
#Correlation matrix
df.corr()
```

Out[14]:

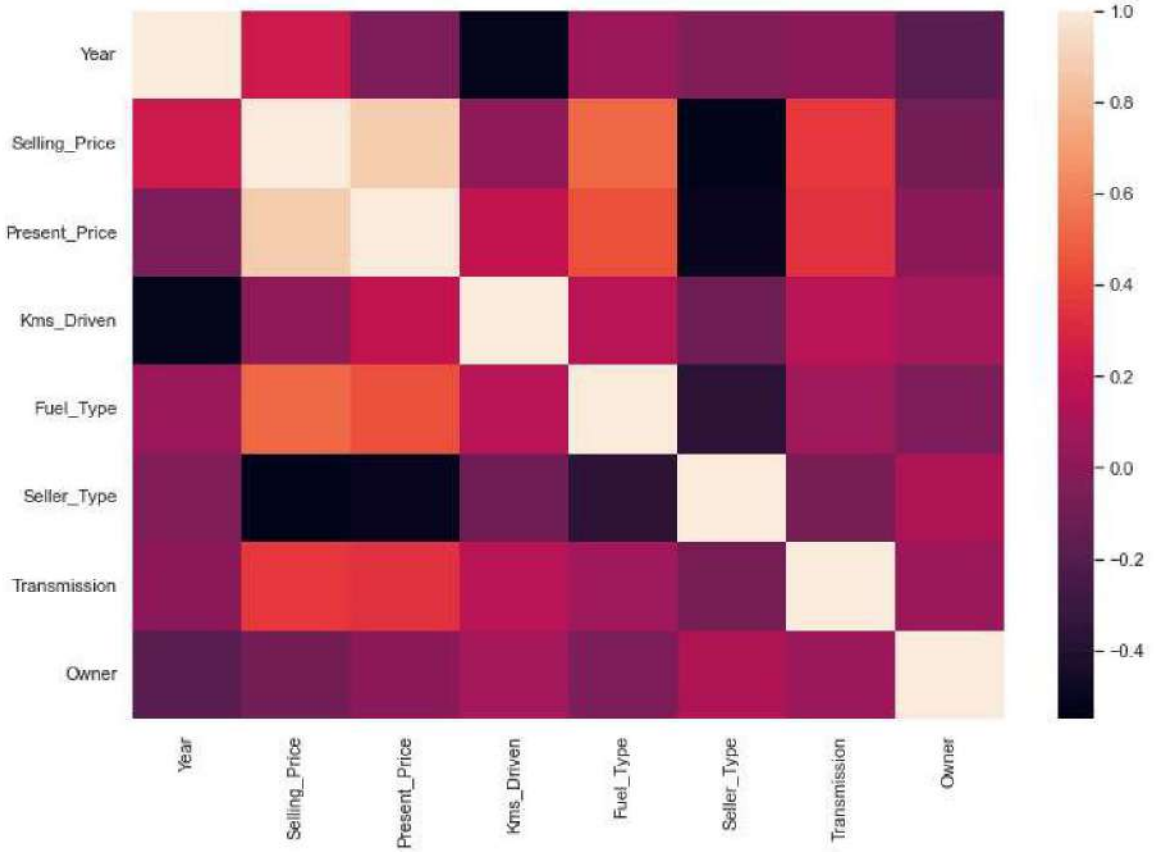
	Year	Selling_Price	Present_Price	Kms_Driven	Owner
Year	1.000000	0.236141	-0.047584	-0.524342	-0.182104
Selling_Price	0.236141	1.000000	0.878983	0.029187	-0.088344

Present_Price	-0.047584	0.878983	1.000000	0.203647	0.008057
Kms_Driven	-0.524342	0.029187	0.203647	1.000000	0.089216
Owner	-0.182104	-0.088344	0.008057	0.089216	1.000000

Heatmap is a data visualization technique, which represents data using different colours in two dimensions. In Python, we can create a heatmap using matplotlib and seaborn library.

In [36]:
```
#Heat map
fig, ax = plt.subplots(figsize=(12,8))
sns.heatmap(df.corr())
```

Out[36]: <AxesSubplot:>

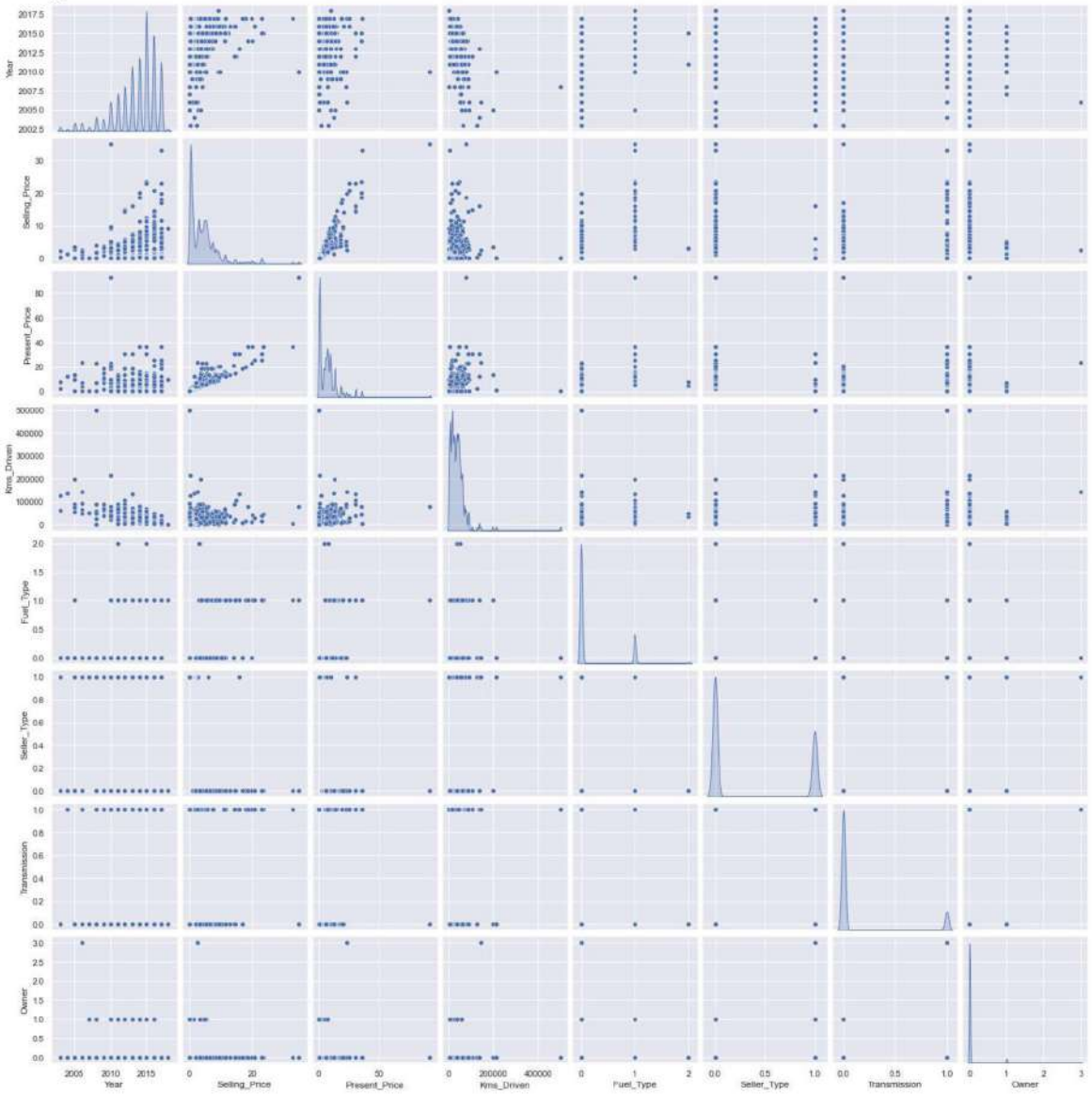


**Pairplot

A pairplot plot a pairwise relationships in a dataset. The pairplot function creates a grid of Axes such that each variable in data will by shared in the y-axis across a single row and in the x-axis across a single column.

In [29]:
```
plt.figure(figsize=(9,9))
sns.pairplot(df,diag_kind="kde", diag_kws=dict(shade=True, bw=.05, vertical=Fals
plt.show()
```

```
<Figure size 648x648 with 0 Axes>
```



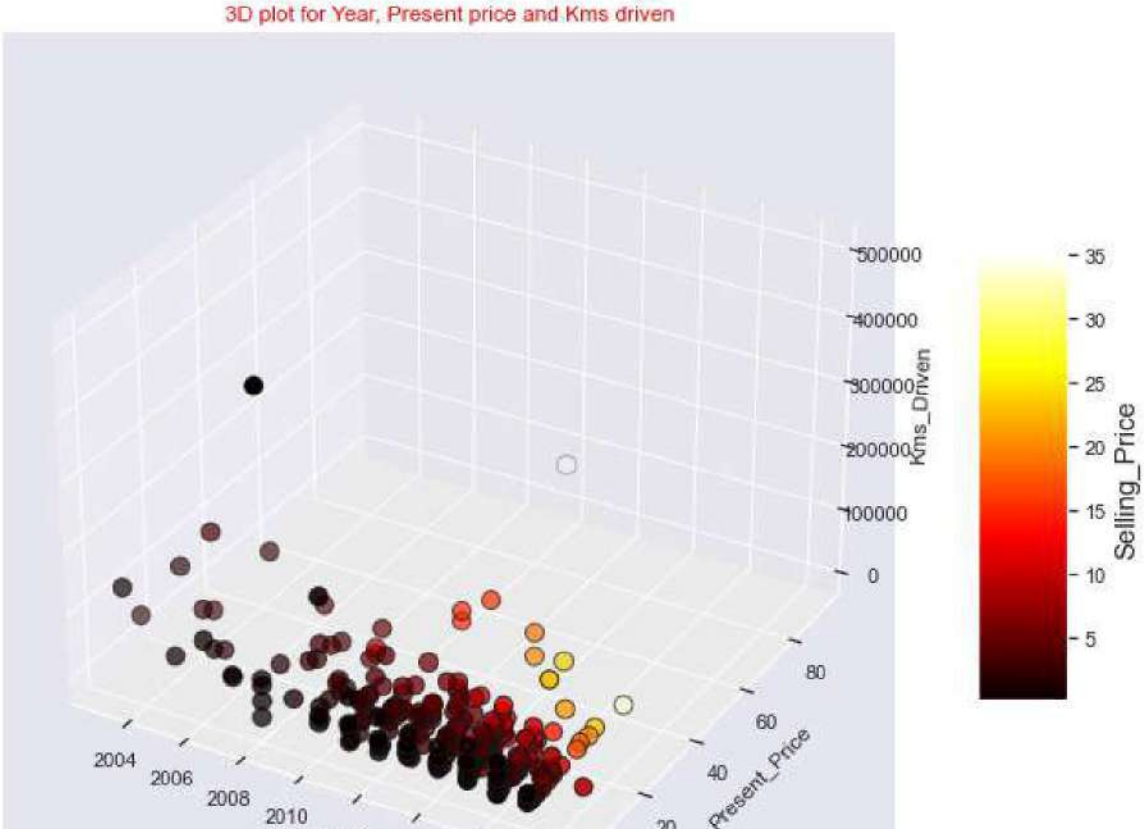
In [30]:
```
y=df.Selling_Price
x=df.drop(["Selling_Price","Car_Name"],axis=1)
print("Dataset after removing columns : \n",x)
```

```
Dataset after removing columns :
      Year  Present_Price  Kms_Driven  Fuel_Type  Seller_Type  Transmission  \
0     2014           5.59       27000          0            0             0
1     2013           9.54       43000          1            0             0
2     2017           9.85        6900          0            0             0
3     2011           4.15        5200          0            0             0
4     2014           6.87       42450          1            0             0
..     ...            ...         ...        ...          ...           ...
296   2016          11.60       33988          1            0             0
297   2015           5.90       60000          0            0             0
298   2009          11.00       87934          0            0             0
299   2017          12.50        9000          1            0             0
300   2016           5.90        5464          0            0             0

     Owner
0        0
1        0
```

```
2        0
3        0
4        0
..     ...
296      0
297      0
298      0
299      0
300      0

[301 rows x 7 columns]
```

In [38]:

```
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(16,9))
ax  = fig.gca(projection = "3d")

plot =  ax.scatter(df["Year"],
           df["Present_Price"],
           df["Kms_Driven"],
           linewidth=1,edgecolor ="k",
           c=df["Selling_Price"],s=100,cmap="hot")

ax.set_xlabel("Year")
ax.set_ylabel("Present_Price")
ax.set_zlabel("Kms_Driven")

lab = fig.colorbar(plot,shrink=.5,aspect=5)
lab.set_label("Selling_Price",fontsize = 15)

plt.title("3D plot for Year, Present price and Kms driven",color="red")
plt.show()
```



Applying Regression Models

In [31]:
```
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=1)
print("Dimensions of x train: ",x_train.shape)
print("Dimensions of x test: ",x_test.shape)
print("Dimensions of y train: ",y_train.shape)
print("Dimensions of y test: ",y_test.shape)
```

```
Dimensions of x train:  (240, 7)
Dimensions of x test:  (61, 7)
Dimensions of y train:  (240,)
Dimensions of y test:  (61,)
```

In [32]:
```
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score
```

In [33]:
```
cv=5 # CV value
r_2 = [] # List for r 2 score
CV = [] # list for CV scores mean

# Main function for models
def model(algorithm,x_train_,y_train_,x_test_,y_test_):
    algorithm.fit(x_train_,y_train_)
    predicts=algorithm.predict(x_test_)
    prediction=pd.DataFrame(predicts)
    R_2=r2_score(y_test_,prediction)
    cross_val=cross_val_score(algorithm,x_train_,y_train_,cv=cv)

    # Appending results to Lists
    r_2.append(R_2)
    CV.append(cross_val.mean())

    # Printing results
    print(algorithm,"\n")
    print("r_2 score :",R_2,"\n")
    print("CV scores:",cross_val,"\n")
    print("CV scores mean:",cross_val.mean())

    # Plot for prediction vs originals
    test_index=y_test_.reset_index()["Selling_Price"]
    ax=test_index.plot(label="originals",figsize=(12,6),linewidth=2,color="r")
    ax=prediction[0].plot(label = "predictions",figsize=(12,6),linewidth=2,color
    plt.legend(loc='upper right')
    plt.title("ORIGINALS VS PREDICTIONS")
    plt.xlabel("index")
    plt.ylabel("values")
    plt.show()
```

In [34]:

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
model(lr,x_train,y_train,x_test,y_test)
```

```
LinearRegression()

r_2 score : 0.8484549412090161

CV scores: [0.89746723 0.88756505 0.83007487 0.81438137 0.75880539]

CV scores mean: 0.837658781192008
```