Name : PUJITHA SORNAPUDI

Email : sornapudipujitha@gmail.com

LinkedIn: PUJITHASORNAPUDI

TITLE OF PROJECT : Car Price Prediction Using Linear Regression Algorithm

Introduction

A car price prediction project uses machine learning to estimate the value of a vehicle based on various features. This project aims to provide accurate price predictions, benefiting both buyers and sellers in the automotive market. By analyzing factors like Fuel Type,Kms Driven, Manufacture Year, Transmission Type (Manual, Automatic, number of Owners etc..., the project helps in setting competitive prices and facilitating informed decisions.

About Dataset

Used Car Price Prediction Dataset is a comprehensive collection of automotive information extracted from the popular automotive marketplace website, https://www.cars.com. This dataset comprises 309 data points, each representing a unique vehicle listing, and includes fuel type, Owner, Kms_Driven, Present_Price, Car_Name, year, Seller_Type, Transmission etc...

Description:

- Car_Name : Name of Car or Model Name
- Year : which Year Car Manufactured
- Selling Price : Price of Car
- Present_Price : Present Car Price
- Kms_Driven : How many Kilometer Driven
- Fuel_Type : Type of Fuel (e.g.Petrol, Diseal, CNG)
- Seller_Type : Type of Seller (e.g. Dealer, Individual)

- Transmission : Car is Manual or Automatic
- Owner : Number of Owners used this car previously

Data Load:

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
Data=pd.read_csv('car data.csv')
Data
```

Out[ ]:

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

301 rows × 9 columns

Exploratory Data Analysis:

In [ ]:
```
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
```

In [ ]:
```
Data.describe(include='number')
```

Out[ ]:

	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 [ ]:
```
Data.describe(include='object')
```

Out[ ]:

	Car_Name	Fuel_Type	Seller_Type	Transmission
count	301	301	301	301
unique	98	3	2	2
top	city	Petrol	Dealer	Manual
freq	26	239	195	261

In [ ]:
```
Data.isnull().sum()
```

Out[ ]:

	0
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 [ ]:

```
Data.shape
```

Out[ ]:

```
(301, 9)
```

In [ ]:

```
Car_Name_Count=Data['Car_Name'].nunique()
Car_Unique_Names=Data['Car_Name'].unique()
print(f'Car_Name_Counts=',Car_Name_Count)
print(f'Car_Unique_Names=',Car_Unique_Names)
```

```
Car_Name_Counts= 98
Car_Unique_Names= ['ritz' 'sx4' 'ciaz' 'wagon r' 'swift' 'vitara brezza' 's cross'
 'alto 800' 'ertiga' 'dzire' 'alto k10' 'ignis' '800' 'baleno' 'omni'
 'fortuner' 'innova' 'corolla altis' 'etios cross' 'etios g' 'etios liva'
 'corolla' 'etios gd' 'camry' 'land cruiser' 'Royal Enfield Thunder 500'
 'UM Renegade Mojave' 'KTM RC200' 'Bajaj Dominar 400'
 'Royal Enfield Classic 350' 'KTM RC390' 'Hyosung GT250R'
 'Royal Enfield Thunder 350' 'KTM 390 Duke ' 'Mahindra Mojo XT300'
 'Bajaj Pulsar RS200' 'Royal Enfield Bullet 350'
 'Royal Enfield Classic 500' 'Bajaj Avenger 220' 'Bajaj Avenger 150'
 'Honda CB Hornet 160R' 'Yamaha FZ S V 2.0' 'Yamaha FZ 16'
 'TVS Apache RTR 160' 'Bajaj Pulsar 150' 'Honda CBR 150' 'Hero Extreme'
 'Bajaj Avenger 220 dtsi' 'Bajaj Avenger 150 street' 'Yamaha FZ  v 2.0'
 'Bajaj Pulsar  NS 200' 'Bajaj Pulsar 220 F' 'TVS Apache RTR 180'
 'Hero Passion X pro' 'Bajaj Pulsar NS 200' 'Yamaha Fazer '
 'Honda Activa 4G' 'TVS Sport ' 'Honda Dream Yuga '
 'Bajaj Avenger Street 220' 'Hero Splender iSmart' 'Activa 3g'
 'Hero Passion Pro' 'Honda CB Trigger' 'Yamaha FZ S '
 'Bajaj Pulsar 135 LS' 'Activa 4g' 'Honda CB Unicorn'
 'Hero Honda CBZ extreme' 'Honda Karizma' 'Honda Activa 125' 'TVS Jupyter'
 'Hero Honda Passion Pro' 'Hero Splender Plus' 'Honda CB Shine'
 'Bajaj Discover 100' 'Suzuki Access 125' 'TVS Wego' 'Honda CB twister'
 'Hero Glamour' 'Hero Super Splendor' 'Bajaj Discover 125' 'Hero Hunk'
 'Hero  Ignitor Disc' 'Hero  CBZ Xtreme' 'Bajaj  ct 100' 'i20' 'grand i10'
 'i10' 'eon' 'xcent' 'elantra' 'creta' 'verna' 'city' 'brio' 'amaze'
 'jazz']
```

In [ ]:
```
Fuel_Types=Data['Fuel_Type'].unique()
print(f'Fuel_Types=',Fuel_Types)
```

```
Fuel_Types= ['Petrol' 'Diesel' 'CNG']
```

In [ ]:
```
print(Data['Fuel_Type'].value_counts())
```

```
Fuel_Type
Petrol    239
Diesel     60
CNG         2
Name: count, dtype: int64
```

In [ ]:
```
Seller_Type=Data['Seller_Type'].unique()
print(f'Seller_Type=',Seller_Type)
```

```
Seller_Type= ['Dealer' 'Individual']
```

In [ ]:
```
print(Data['Seller_Type'].value_counts())
```

```
Seller_Type
Dealer        195
Individual    106
Name: count, dtype: int64
```

In [ ]:
```
Transmission=Data['Transmission'].unique()
print(f'Transmission=',Transmission)
```

```
Transmission= ['Manual' 'Automatic']
```

In [ ]:
```
print(Data['Transmission'].value_counts())
```

```
Transmission
Manual       261
Automatic     40
Name: count, dtype: int64
```

In [ ]:
```
Owner_Type=Data['Owner'].unique()
print(f'Owner_Type=',Owner_Type)
```

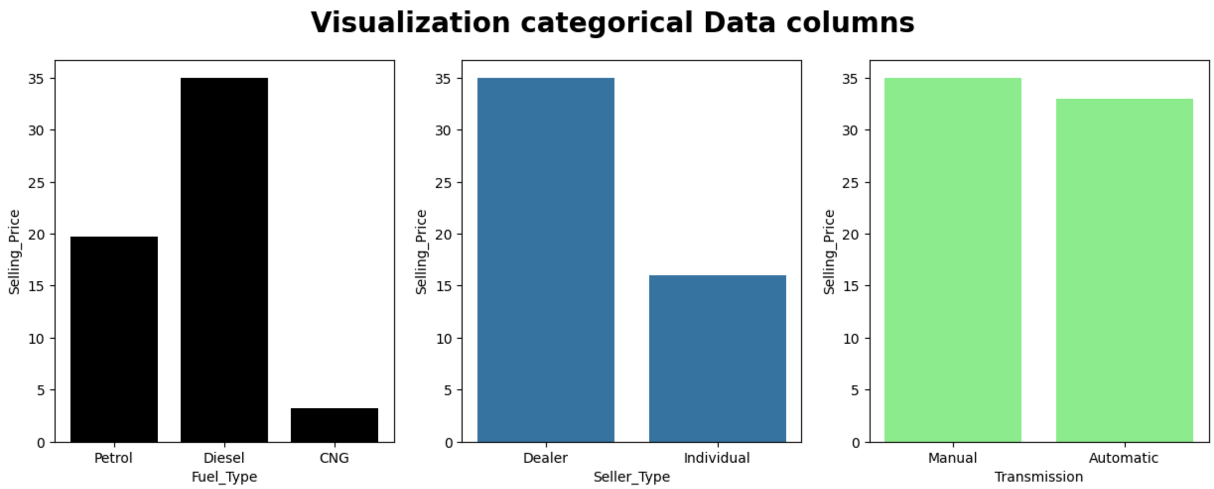
```
Owner_Type= [0 1 3]
```

In [ ]:
```
print(Data['Owner'].value_counts())
```

```
Owner
0    290
1     10
3      1
Name: count, dtype: int64
```

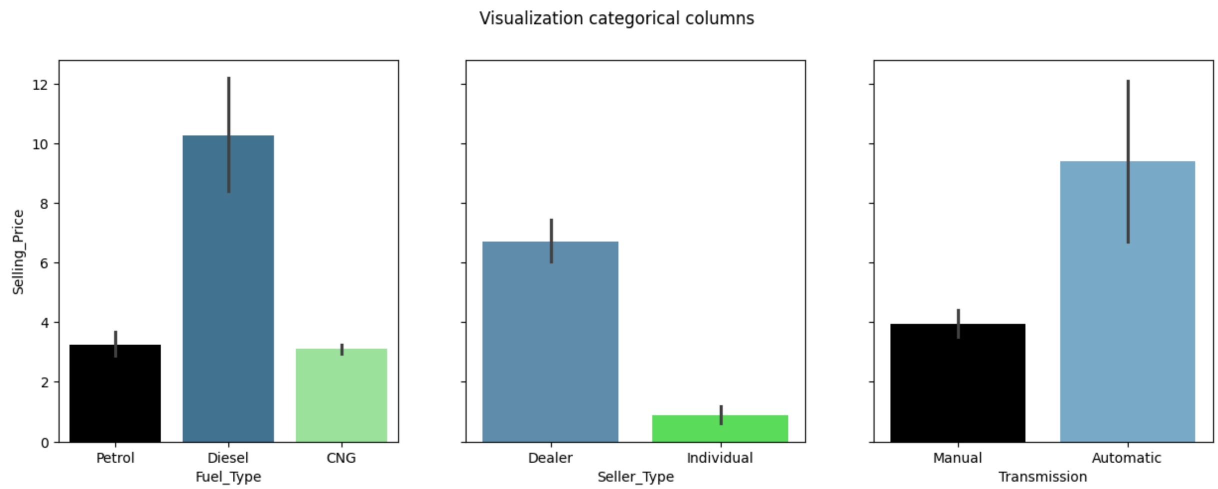
Visualization Chart for analysis

In [ ]:
```
fuel_type=Data['Fuel_Type']
seller_type=Data['Seller_Type']
transmission=Data['Transmission']
Owner=Data['Owner']
selling_price=Data['Selling_Price']
plt.figure(figsize=(15,5))
fig=plt.suptitle("Visualization categorical Data columns",fontsize=20,fontweight='b
plt.subplot(1,3,1)
plt.bar(fuel_type,selling_price,color='black')  #plot 1: Fuel_Type#
plt.xlabel('Fuel_Type')
plt.ylabel('Selling_Price')
plt.subplot(1,3,2)
plt.bar(seller_type,selling_price,color='#3776A1')  #plot 2 : Seller_Type#
plt.xlabel('Seller_Type')
plt.ylabel('Selling_Price')
plt.subplot(1,3,3)
plt.bar(transmission,selling_price,color='lightgreen')  #Plot 3: Transmission#
plt.xlabel('Transmission')
plt.ylabel('Selling_Price')
plt.show()
```



In [ ]:
```
fig, axes=plt.subplots(1,3,figsize=(15,5),sharey=True)
fig.suptitle("Visualization categorical columns")
sns.barplot(x=fuel_type,y=selling_price,palette=["black","#3776A1","lightgreen"], a
sns.barplot(x=seller_type,y=selling_price,palette=["#5293BB","#45f248"],ax=axes[1])
sns.barplot(x=transmission,y=selling_price,palette=["black","#6Eb1D6"],ax=axes[2])
plt.show()
```

```
/tmp/ipython-input-2909329612.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=fuel_type,y=selling_price,palette=["black","#3776A1","lightgreen"],
ax=axes[0])
/tmp/ipython-input-2909329612.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=seller_type,y=selling_price,palette=["#5293BB","#45f248"],ax=axes
[1])
/tmp/ipython-input-2909329612.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.1
4.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=transmission,y=selling_price,palette=["black","#6Eb1D6"],ax=axes[2])
```



The above bar chart are drawn based on their specific mean values.

In [ ]:
```
Petrol_Car_Data=Data.groupby('Fuel_Type').get_group('Petrol')
Petrol_Car_Data.describe()
```

Out[ ]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	239.000000	239.000000	239.000000	239.000000	239.000000
mean	2013.539749	3.264184	5.583556	33528.937238	0.050209
std	3.042674	3.135537	5.290685	40308.984886	0.270368
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.600000	0.940000	13850.000000	0.000000
50%	2014.000000	2.650000	4.600000	25870.000000	0.000000
75%	2016.000000	5.200000	7.980000	44271.000000	0.000000
max	2017.000000	19.750000	23.730000	500000.000000	3.000000

In [ ]:
```
Seller_type=Data.groupby('Seller_Type').get_group('Dealer')
Seller_type.describe()
```

Out[ ]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	195.000000	195.000000	195.000000	195.000000	195.000000
mean	2013.712821	6.721692	10.886308	39850.133333	0.020513
std	2.686275	5.136088	8.806563	24860.401003	0.142111
min	2003.000000	1.050000	2.690000	2071.000000	0.000000
25%	2012.000000	3.750000	6.580000	22148.500000	0.000000
50%	2014.000000	5.250000	8.500000	39485.000000	0.000000
75%	2016.000000	7.625000	13.460000	51785.500000	0.000000
max	2018.000000	35.000000	92.600000	197176.000000	1.000000

In [ ]:

```
Data
```

Out[ ]:

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

301 rows × 9 columns

Column conversion using one-hot encoding technique

In [ ]:

```
Data['Fuel_Type']=Data['Fuel_Type'].map({'Petrol':1,'Diesel':0,'CNG':2})
#one-hot encoding
Data=pd.get_dummies(data=Data,columns=['Seller_Type','Transmission'],drop_first=Tru
```

In [ ]:
```
Data
```

Out[ ]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Owner	Seller_T
0	ritz	2014	3.35	5.59	27000	1	0	
1	sx4	2013	4.75	9.54	43000	0	0	
2	ciaz	2017	7.25	9.85	6900	1	0	
3	wagon r	2011	2.85	4.15	5200	1	0	
4	swift	2014	4.60	6.87	42450	0	0	
...	...	...	...	...	...	...	...	
296	city	2016	9.50	11.60	33988	0	0	
297	brio	2015	4.00	5.90	60000	1	0	
298	city	2009	3.35	11.00	87934	1	0	
299	city	2017	11.50	12.50	9000	0	0	
300	brio	2016	5.30	5.90	5464	1	0	

301 rows × 9 columns

In [ ]:
```
Data=Data.drop(['Car_Name'],axis=1)
Data
```

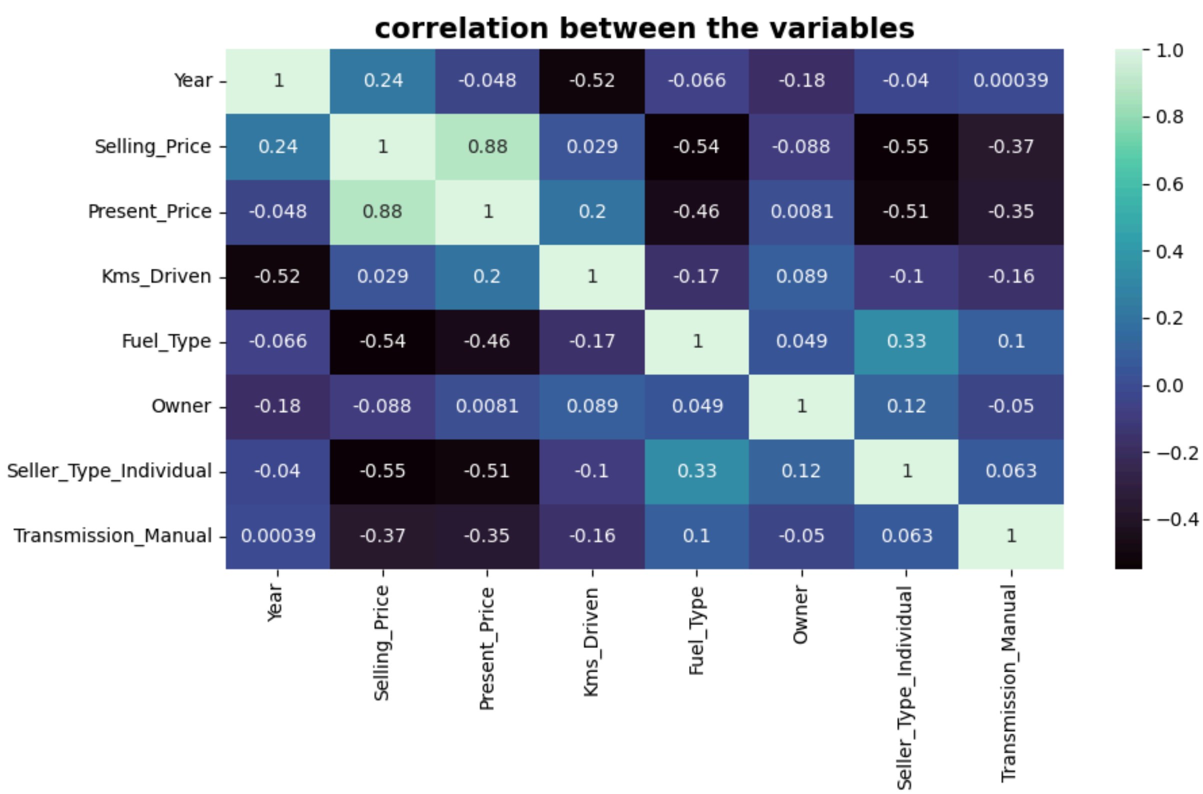
Out[ ]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Owner	Seller_Type_Individu
0	2014	3.35	5.59	27000	1	0	
1	2013	4.75	9.54	43000	0	0	
2	2017	7.25	9.85	6900	1	0	
3	2011	2.85	4.15	5200	1	0	
4	2014	4.60	6.87	42450	0	0	
...	...	...	...	...	...	...	
296	2016	9.50	11.60	33988	0	0	
297	2015	4.00	5.90	60000	1	0	
298	2009	3.35	11.00	87934	1	0	
299	2017	11.50	12.50	9000	0	0	
300	2016	5.30	5.90	5464	1	0	

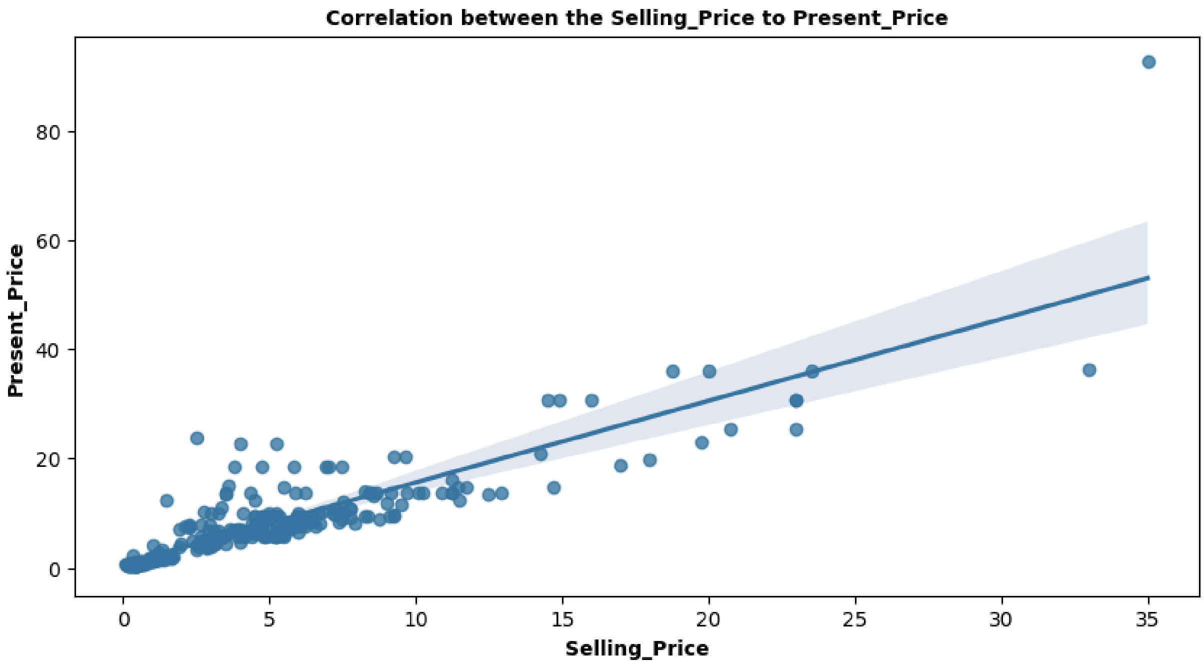
301 rows × 8 columns

In [ ]:

```
plt.figure(figsize=(10,5))
sns.heatmap(Data.corr(),annot=True,cmap='mako')
plt.title('correlation between the variables',fontsize=15,fontweight='bold',color='
plt.show()
```



In [ ]:
```
fig=plt.figure(figsize=(10,5))
sns.regplot(x=Data['Selling_Price'],y=Data['Present_Price'],data=Data,color='#3776A
plt.title('Correlation between the Selling_Price to Present_Price',fontdict={'fonts
plt.xlabel('Selling_Price',fontsize=10,fontweight='bold',color='black')
plt.ylabel('Present_Price',fontsize=10,fontweight='bold',color='black')
plt.show()
```



CarPrice Prediction Using Linear Regression model

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from joblib import dump
DF=pd.read_csv('car data.csv')
DF['Fuel_Type']=DF['Fuel_Type'].map({'Petrol':1,'Diesel':1,'CNG':2})
DF=pd.get_dummies(data=DF,columns=['Seller_Type','Transmission'],drop_first=True,dt
DF=DF.drop(['Car_Name'],axis=1)
X=DF[['Year','Present_Price','Kms_Driven','Fuel_Type','Owner','Seller_Type_Individu
y=DF['Selling_Price']    #Target column#
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)                    #StandardScaler_Formula=>>z= x-
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran
model=LinearRegression()
model.fit(X_train, y_train)
dump(model, 'car_price_prediction_model.joblib')
dump(scaler, 'scaler.joblib')
print("✅ Model and scaler saved successfully!")
```

✅ Model and scaler saved successfully!

In [ ]:
```
import pandas as pd
from joblib import load
model=load('car_price_prediction_model.joblib')
scaler=load('scaler.joblib')

print("🚗 Welcome to the Car Price Prediction App!")
print("Please enter the following car details:")

fuel_type=input("Fuel_Type:").strip().lower()
seller_type=input("Seller_Type:").strip().lower()
transmission=input("Transmission:").strip().lower()
owner=int(input("Number of Previous Owners:"))
year=int(input("Year of Manufacture:"))
kms_driven=float(input("Kilometer Driven:"))
present_price=float(input("Preseent_price(in Lakhs):"))

fuel_type_binary=1 if fuel_type=="petrol" else 0
seller_type_binary= 1 if seller_type=="individual" else 0
transmission_manual= 1 if transmission=="manual" else 0

input_dict={
    'Year':[year],
    'Present_Price':[present_price],
    'Kms_Driven':[kms_driven],
    'Fuel_Type':[fuel_type_binary],
    'Owner':[owner],
    'Seller_Type_Individual':[seller_type_binary],
    'Transmission_Manual':[transmission_manual]
}
input_df=pd.DataFrame(input_dict)
input_scaled=scaler.transform(input_df)
predicted_price=model.predict(input_scaled)
print(f"\n predicted selling price : ₹{predicted_price[0]:,.2f} Lakhs")
```

```
🚗 Welcome to the Car Price Prediction App!
Please enter the following car details:
Fuel_Type:Dieseal
Seller_Type:Individual
Transmission:Automatic
Number of Previous Owners:0
Year of Manufacture:2019
Kilometer Driven:4000
Preseent_price(in Lakhs):7.8

 predicted selling price : ₹8.56 Lakhs
```