In [14]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn import metrics
```

In [15]:
```
car=pd.read_csv("C:\\Users\\Pranav\\Desktop\\DATA SCIENCE DATA\\CVC file\\car data.csv")
car.head()
```

Out[15]:

	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

In [16]:
```
#number of rows and columns
car.shape
```

Out[16]:
```
(301, 9)
```

In [17]:
```
#information about data
car.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 [18]:
```
#to find missing value in data
car.isnull().sum()
```

Out[18]:
```
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 [19]:
```
#describe mathamatical data
car.describe()
```

Out[19]:

	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 [20]:
```
#checking the distribution of categorical data
print(car['Fuel_Type'].value_counts())
print(car['Seller_Type'].value_counts())
print(car['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
```

In [21]:
```
#label encoding
car.replace({'Fuel_Type':{'Petrol':0,'Diesel':1,'CNG':2},'Seller_Type':{'Dealer':0,'Individual':1},'Transmission':{'Manual':0,'Automatic':1}},inplace=True)
car.head()
```

Out[21]:

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

In [24]:
```
# separting Features and target
X=car.drop(['Selling_Price','Car_Name'],axis=1)
y=car['Selling_Price']
```

In [29]:
```
#training and test data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=2)
print("shape of X_train= ",X_train.shape)
print("shape of X_test= ",X_test.shape)
print("shape of y_train= ",y_train.shape)
print("shape of y_test= ",y_test.shape)
```

```
shape of X_train=  (270, 7)
shape of X_test=  (31, 7)
shape of y_train=  (270,)
shape of y_test=  (31,)
```

In [37]:
```
model=LinearRegression()
lin_reg_model=model.fit(X_train,y_train)
```

In [38]:
```
training_data_predction=lin_reg_model.predict(X_train)
```

In [41]:
```
error_score=metrics.r2_score(y_train,training_data_predction)
print("R squared Error:",error_score)
```

```
R squared Error: 0.87994516604937
```

In [45]:
```
#plot scatterplot
plt.scatter(y_train,training_data_predction,color='purple')
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
plt.title('Actual price vs Predicted price')
plt.show()
```

In [55]:
```
lin_reg_model.predict(X_test)
```

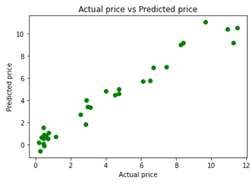
Out[55]:
```
array([11.03495511,  0.0587425 ,  4.98734157,  3.37519499, 10.49461963,
        4.48096185,  3.44194637,  5.79303592, -0.55653713,  5.72317441,
        6.95741558,  4.60885803,  0.65127991,  9.14398511,  2.73019734,
        0.57231936,  1.05331842,  0.90566946,  9.19110393,  4.80724986,
        0.75081153,  8.97436005,  0.52378125, 10.43208421, -0.1215052 ,
        6.98859921,  0.76010365,  0.2       ,  3.99921639,  1.83570043,
        1.57201626])
```

In [56]:
```
testing_data_predction=lin_reg_model.predict(X_test)
```

In [58]:
```
error_score=metrics.r2_score(y_test,testing_data_predction)
print("R squared Error:",error_score)
```

```
R squared Error: 0.961601609160524
```

In [60]:
```
#plot scatterplot
plt.scatter(y_test,testing_data_predction,color='green')
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
plt.title('Actual price vs Predicted price')
plt.show()
```



In [61]:
```
model=Lasso()
Lasso_reg_model=model.fit(X_train,y_train)
```

In [65]:
```
training_data_predction1=Lasso_reg_model.predict(X_train)
```

In [66]:
```
error_score=metrics.r2_score(y_train,training_data_predction1)
print("R squared Error:",error_score)
```

```
R squared Error: 0.8427856123435794
```

In [67]:
```
#plot scatterplot
plt.scatter(y_train,training_data_predction1,color='red')
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
plt.title('Actual price vs Predicted price')
plt.show()
```