car-price-prediction

February 3, 2024

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

[2]:
```
data = pd.read_csv('car data.csv')
```

[3]:
```
data.head()
```

[3]:

	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

[4]:
```
data.shape
```

[4]:
```
(301, 9)
```

[5]:
```
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
```

[6]:
```
data.isnull().sum()
```

[6]:
```
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
```

[7]:
```
data['Fuel_Type'].value_counts()
```

[7]:
```
Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64
```

[8]:
```
data['Seller_Type'].value_counts()
```

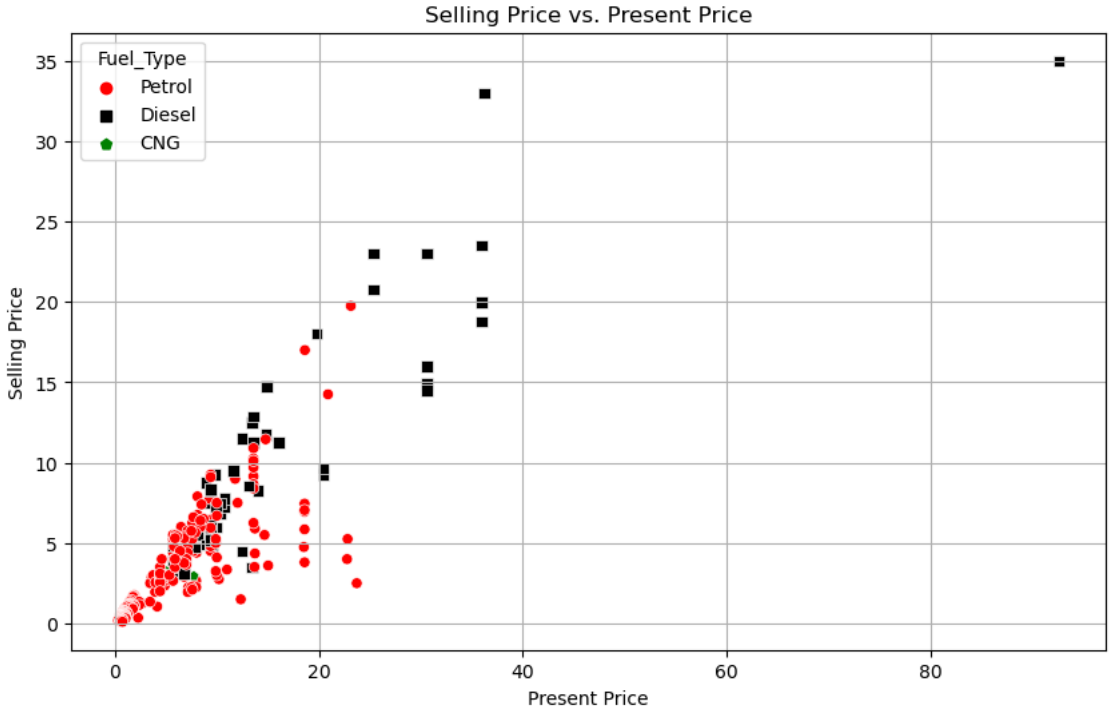
[8]:
```
Dealer        195
Individual    106
Name: Seller_Type, dtype: int64
```

[9]:
```
data['Transmission'].value_counts()
```

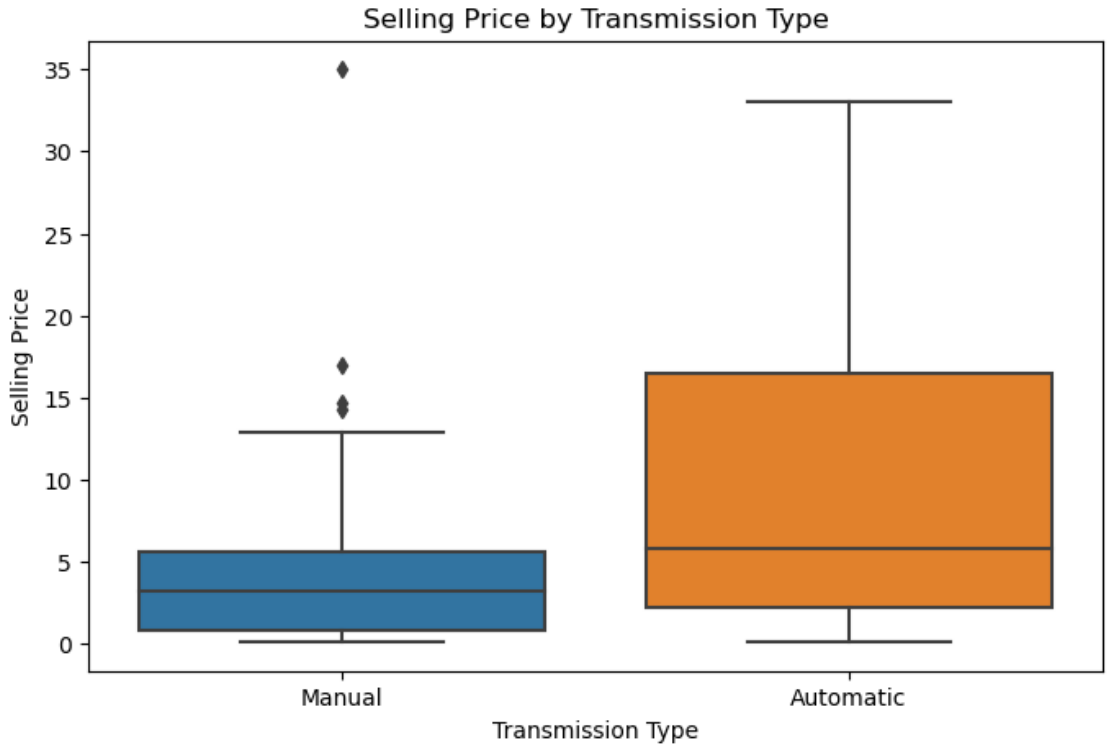
[9]:
```
Manual       261
Automatic     40
Name: Transmission, dtype: int64
```

[10]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='Present_Price', y='Selling_Price',␣
↪hue='Fuel_Type',
                palette={'Petrol': 'red', 'Diesel': 'black','CNG':'green'},␣
↪style='Fuel_Type',
                markers={'Petrol': 'o', 'Diesel': 's','CNG':'p'})
```

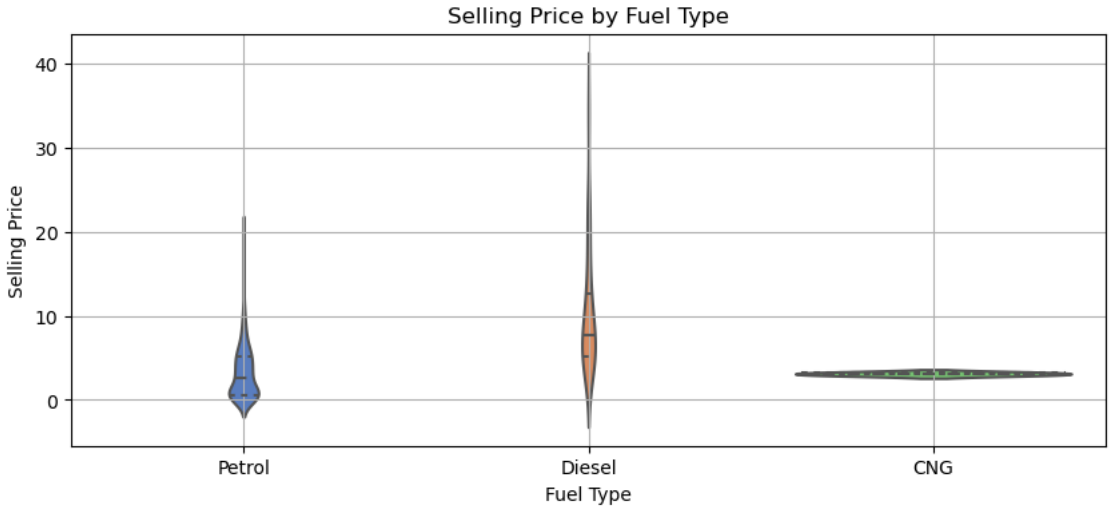
```
plt.title('Selling Price vs. Present Price')
plt.xlabel('Present Price')
plt.ylabel('Selling Price')
plt.grid()
plt.show()
```



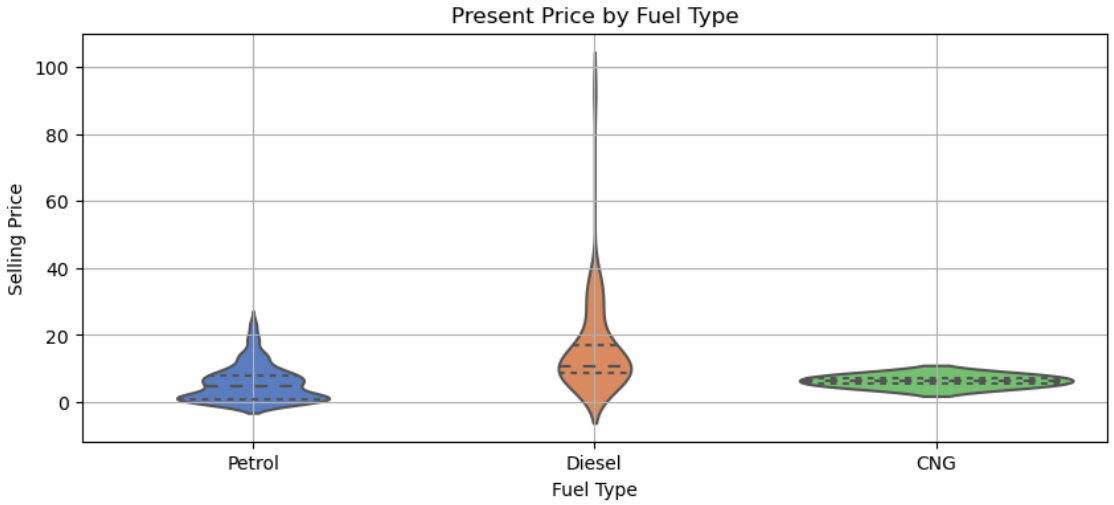
[11]:
```
plt.figure(figsize=(8, 5))
sns.boxplot(data=data, x='Transmission', y='Selling_Price')
plt.title('Selling Price by Transmission Type')
plt.xlabel('Transmission Type')
plt.ylabel('Selling Price')
plt.show()
```



[12]:
```
plt.figure(figsize=(10, 4))
sns.violinplot(data=data, x='Fuel_Type', y='Selling_Price', inner='quartile',␣
↪palette='muted')
plt.title('Selling Price by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Selling Price')
plt.grid()
plt.show()
```



[13]:
```
plt.figure(figsize=(10, 4))
sns.violinplot(data=data, x='Fuel_Type', y='Present_Price', inner='quartile',␣
↪palette='muted')
plt.title('Present Price by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Selling Price')
plt.grid()
plt.show()
```

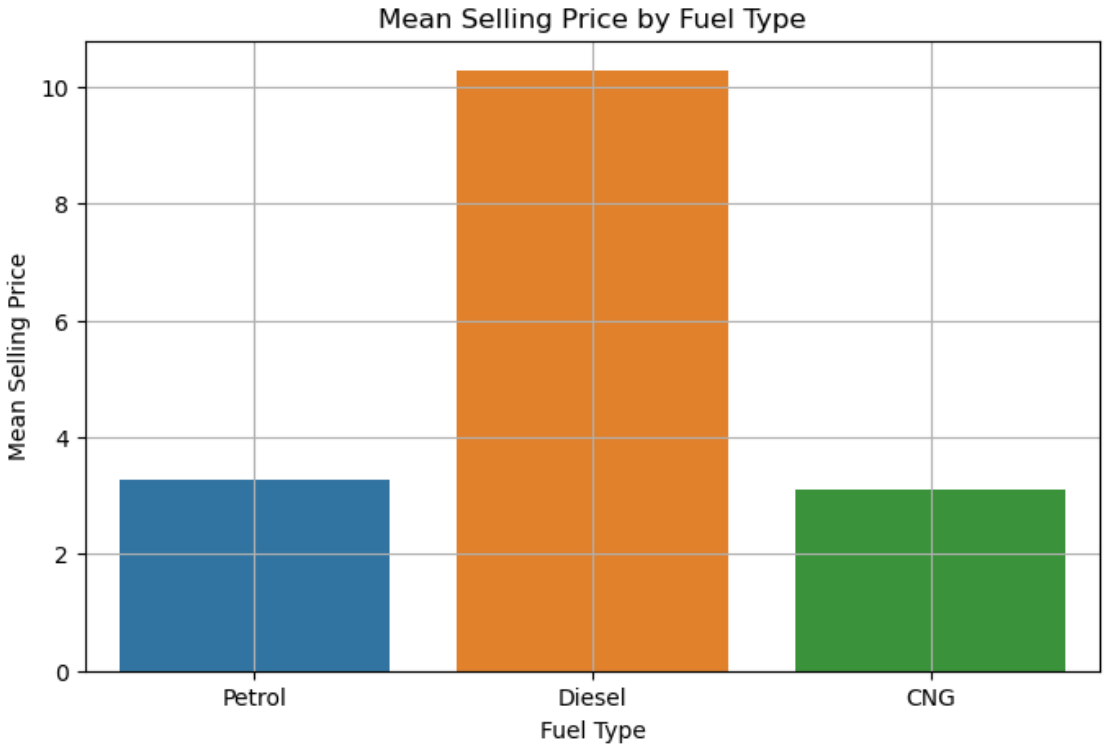


[14]:
```
plt.figure(figsize=(8, 5))
sns.barplot(data=data, x='Fuel_Type', y='Selling_Price', ci=None)
plt.title('Mean Selling Price by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Mean Selling Price')
plt.grid()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_6948\4040734656.py:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=data, x='Fuel_Type', y='Selling_Price', ci=None)
```

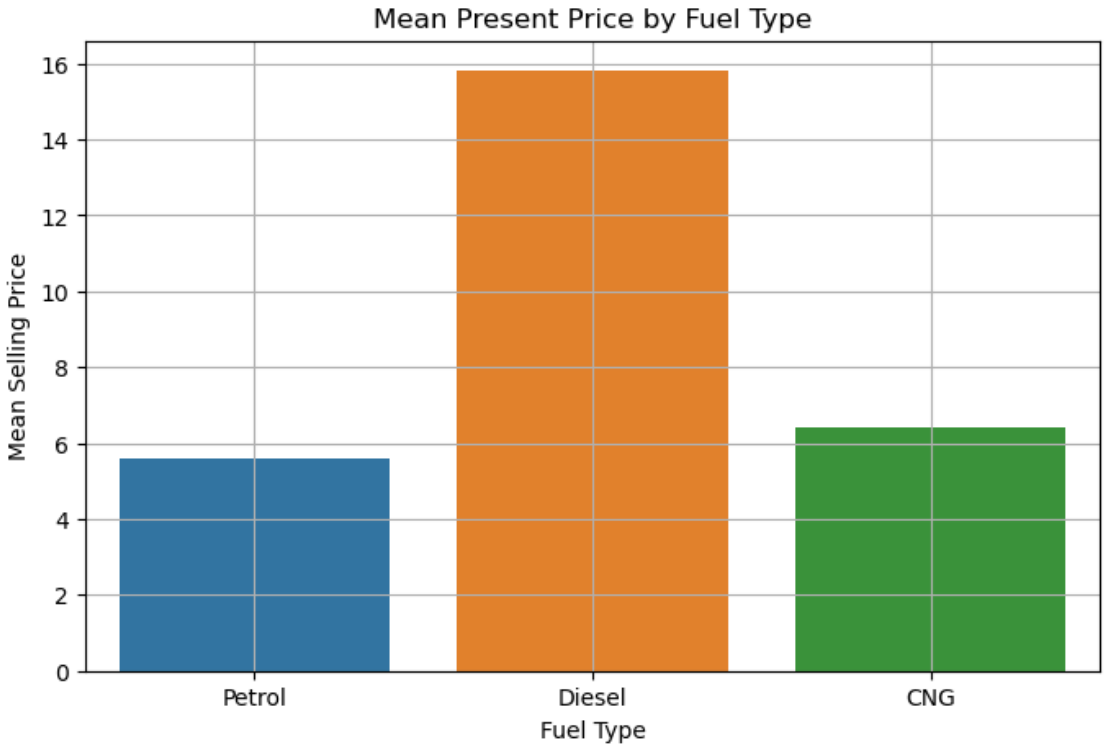


[15]:
```
plt.figure(figsize=(8, 5))
sns.barplot(data=data, x='Fuel_Type', y='Present_Price', ci=None)
plt.title('Mean Present Price by Fuel Type')
plt.xlabel('Fuel Type')
plt.ylabel('Mean Selling Price')
plt.grid()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_6948\796035521.py:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(data=data, x='Fuel_Type', y='Present_Price', ci=None)
```



0.1 encoding the categorical data

[16]:
```
data.replace({'Fuel_Type':{'Petrol':0,'Diesel':1,'CNG':2}}, inplace=True)
data.replace({'Seller_Type':{'Dealer':0,'Individual':1}}, inplace=True)
data.replace({'Transmission':{'Manual':0,'Automatic':1}}, inplace=True)
```

[17]:
```
data.head()
```

[17]:

	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

[18]:
```
x = data.drop(['Car_Name','Selling_Price'], axis=1)
y = data['Selling_Price']
```

[19]:
```
x.head()
```

[19]:

	Year	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	2014	5.59	27000	0	0	0	0
1	2013	9.54	43000	1	0	0	0
2	2017	9.85	6900	0	0	0	0
3	2011	4.15	5200	0	0	0	0
4	2014	6.87	42450	1	0	0	0

[20]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.1,␣
 ↪random_state=42)
```

[21]:
```
print('x_train :',x_train.shape)
print('x_test :',x_test.shape)
print('y_train :',y_train.shape)
print('y_test :',y_test.shape)
```

```
x_train : (270, 7)
x_test : (31, 7)
y_train : (270,)
y_test : (31,)
```

1 LinearRegression

[22]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import  r2_score
lin_reg_model = LinearRegression()
lin_reg_model.fit(x_train,y_train)
```

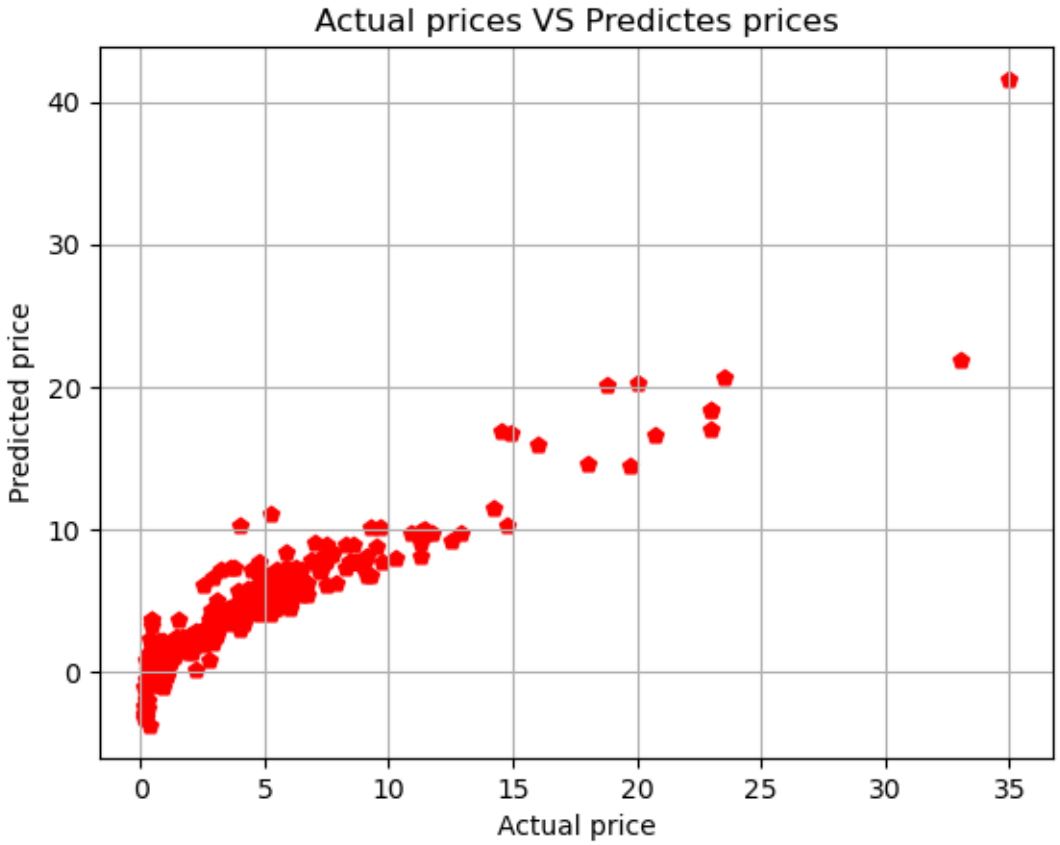
[22]:
```
LinearRegression()
```

1.1 train data

[23]:
```
train_data_pred_1 = lin_reg_model.predict(x_train)
error_score_1 = r2_score(y_train,train_data_pred_1)
print('R squared Error: ',error_score_1)
```

```
R squared Error:  0.8806173715719124
```

[24]:
```
plt.scatter(y_train, train_data_pred_1, color='red',marker='p')
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
plt.title('Actual prices VS Predictes prices')
plt.grid()
plt.show()
```

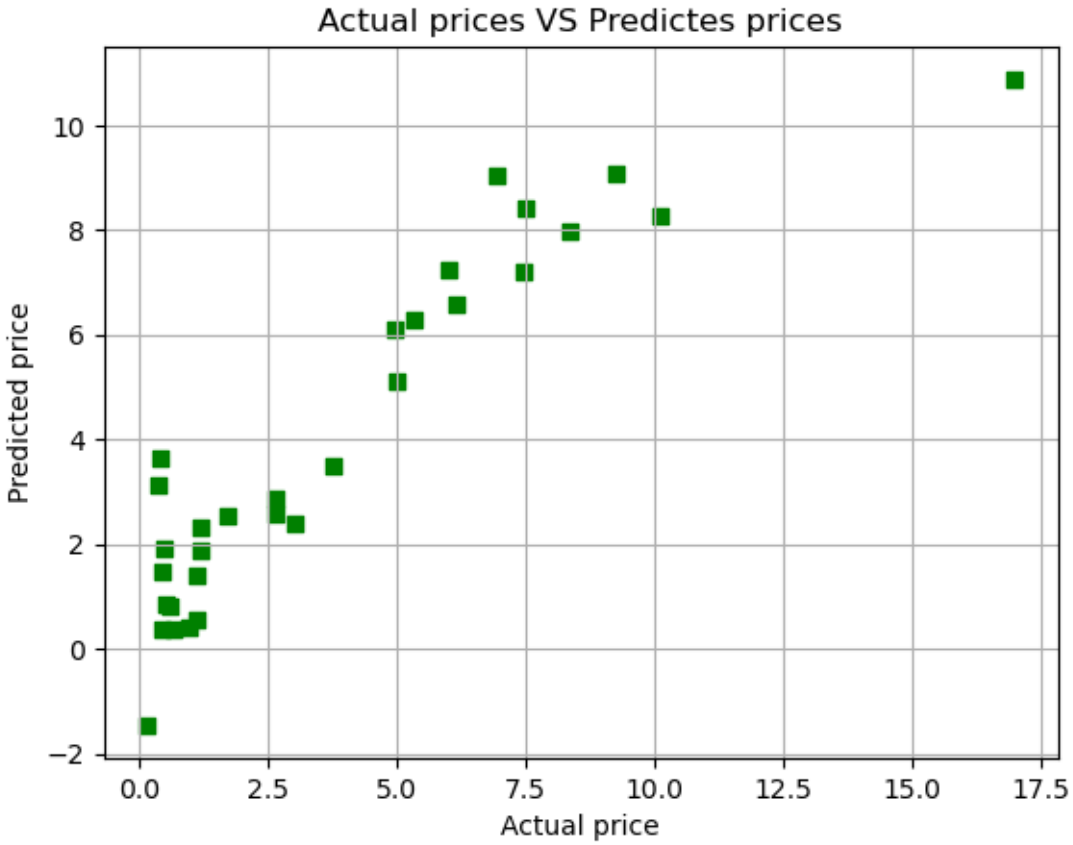


1.2 test data

[25]:
```
test_data_pred_1 = lin_reg_model.predict(x_test)
error_score_1 = r2_score(y_test,test_data_pred_1)
print('R squared Error: ',error_score_1)
```

```
R squared Error:  0.831106947624384
```

[26]:
```
plt.scatter(y_test, test_data_pred_1, color='green', marker='s')
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
plt.title('Actual prices VS Predictes prices')
plt.grid()
plt.show()
```



1.3 Lasso regression

[27]:
```
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score
```

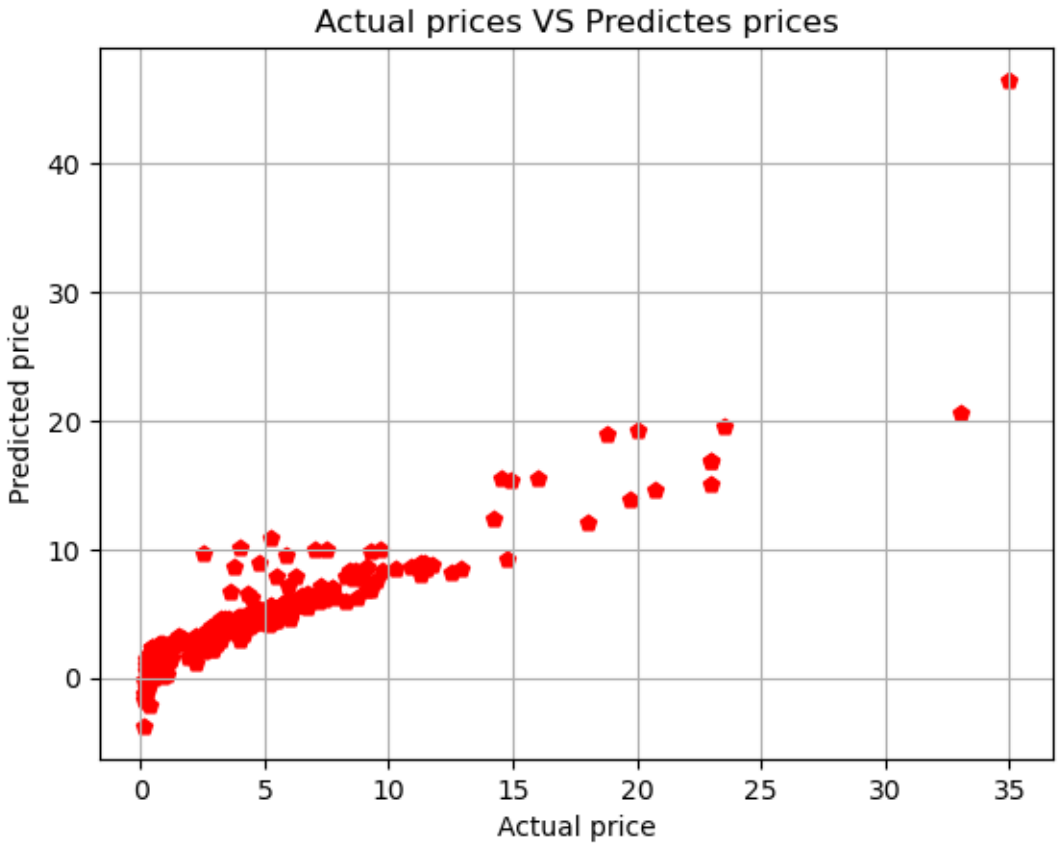
[28]:
```
lasso_reg_model = Lasso()
lasso_reg_model.fit(x_train,y_train)
```

[28]:
```
Lasso()
```

[29]:
```
train_data_pred_2 = lasso_reg_model.predict(x_train)
error_score_2 = r2_score(y_train,train_data_pred_2)
print('R squared Error: ',error_score_2)
```

```
R squared Error:  0.8436909482009372
```

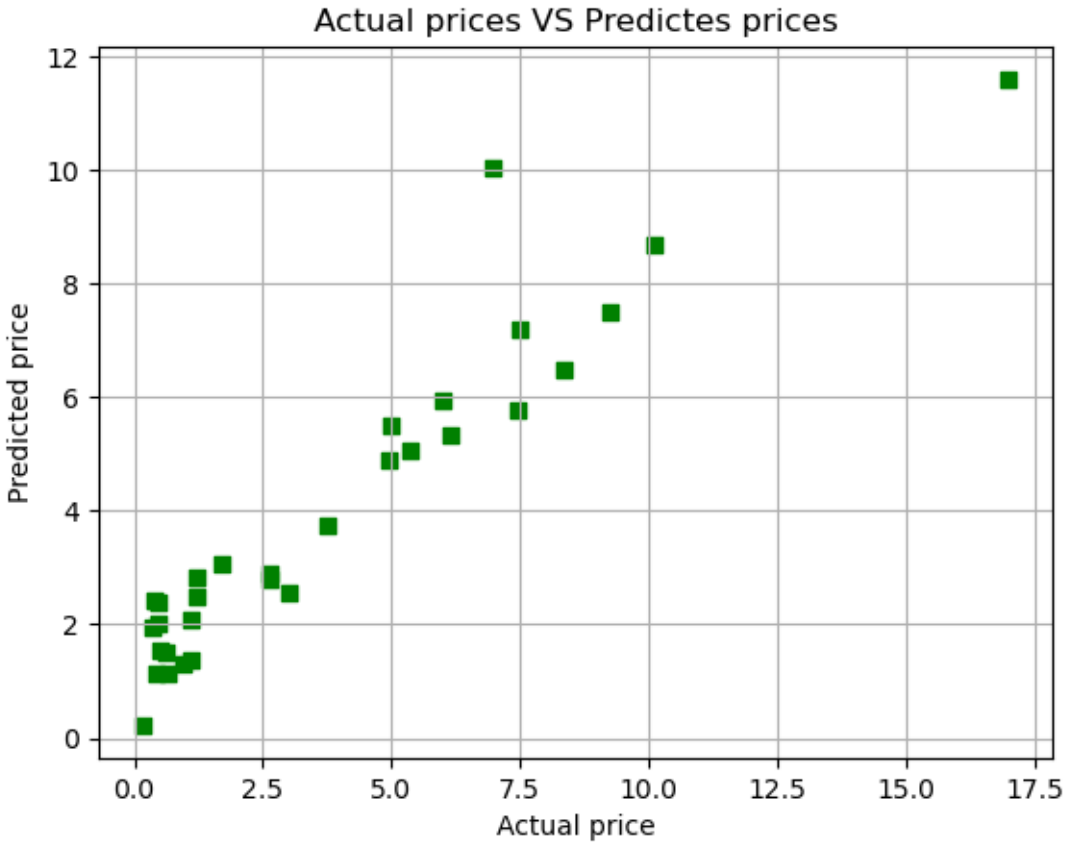
[30]:
```
plt.scatter(y_train, train_data_pred_2, color='red',marker='p')
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
plt.title('Actual prices VS Predictes prices')
plt.grid()
plt.show()
```



[31]:
```
test_data_pred_2 = lasso_reg_model.predict(x_test)
error_score2 = r2_score(y_test,test_data_pred_2)
print('R squared Error: ',error_score2)
```

```
R squared Error:  0.839226320049874
```

[32]:
```
plt.scatter(y_test, test_data_pred_2, color='green', marker='s')
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
plt.title('Actual prices VS Predictes prices')
plt.grid()
plt.show()
```



[ ]: