In [1]:
```python
import pandas as pd
```

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

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

Out[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

In [4]:
```python
df.shape #There are 301 Rows and 9 Columns/Features(Car name,Year,Selling price....)
```

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

In [5]:
```python
#We have Fuel type,Seller_Price,Transmission Owner as catergorical Features.

print(df['Seller_Type'].unique())#Uinque Vales in Seller_type we have Dealer,Individual.

print(df['Transmission'].unique()) #Unique Values Are Manual and Automatic.

print(df['Owner'].unique()) #Unique Values are 0,1,3.

print(df['Fuel_Type'].unique()) #Unique values are Petrol,Diesel,CNG
```

```
['Dealer' 'Individual']
['Manual' 'Automatic']
[0 1 3]
['Petrol' 'Diesel' 'CNG']
```

In [6]:
```python
#To check the Null/Missing Values

df.isnull().sum()  #So there are no Null Values Here
```

Out[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
```

In [7]:
```python
df.describe()  #This shows all the Mean,min,max...Values.
```

Out[7]:

	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 [8]:
```python
df.columns
```

Out[8]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

In [9]:
```python
final_dataset=df[['Year','Selling_Price','Present_Price','Kms_Driven','Fuel_Type','Seller_Type','Transmission','Owner']]
```

In [10]:
```python
final_dataset.head()
```

Out[10]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

In [11]:
```python
final_dataset['Current_Year']=2021  #To get the number of years for the car...we need to cereate new column name Current_Year.
```

In [12]:
```python
final_dataset.head()
```

Out[12]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021

In [13]:
```python
#Subtracting the Current_Year to Year to get No_Years.

final_dataset['No_Years']=final_dataset['Current_Year']-final_dataset['Year']
```

In [14]:
```python
final_dataset.head()
```

Out[14]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year	No_Years
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021	7
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021	8
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021	4
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021	10
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021	7

In [15]:
```python
#Year and Current_Year is not requried because we have No_Years.....

final_dataset.drop(['Year'],axis=1,inplace=True)
```

In [16]:
```python
final_dataset.head()  #So we dont have Year Now
```

Out[16]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year	No_Years
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021	7
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021	8
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021	4
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021	10
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021	7

In [17]:
```python
#Year and Current_Year is not requried because we have No_Years.....

final_dataset.drop(['Current_Year'],axis=1,inplace=True)
```

In [18]:
```python
final_dataset.head()  #So we also removed Current_Year...
```

Out[18]:

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

In [19]:
```python
##Converting all categorical data to Numerical values..

final_dataset=pd.get_dummies(final_dataset,drop_first=True)
```

In [20]:
```python
final_dataset.head() #CNG have been Droped,because we dont have any CNG vehicle here.
```

Out[20]:

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

In [21]:
```python
#To find the co-relation ...(How one features is co-related with another features)

final_dataset.corr()
```

Out[21]:

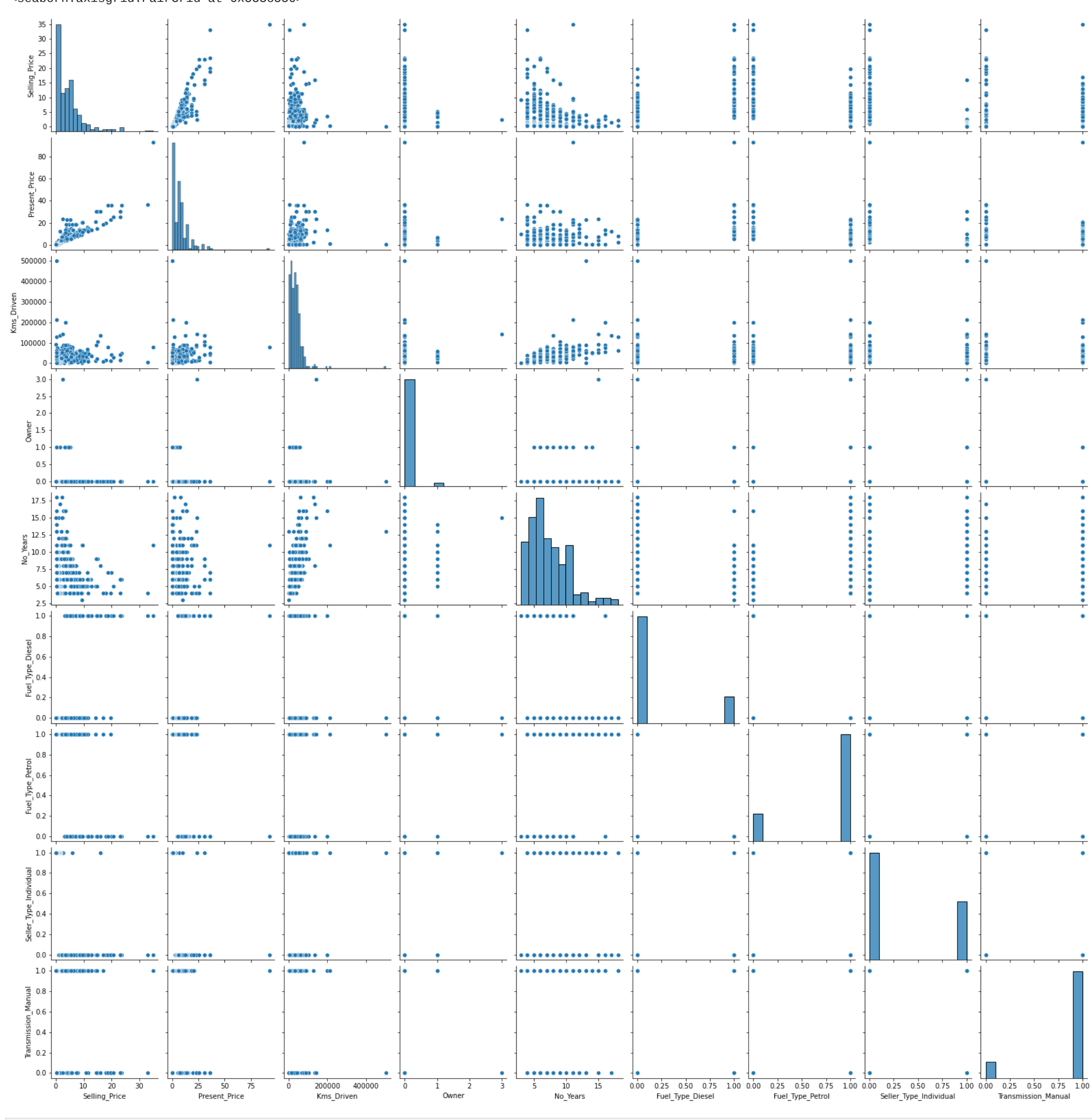
	Selling_Price	Present_Price	Kms_Driven	Owner	No_Years	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.236141	0.552339	-0.540571	-0.550724	-0.367128
Present_Price	0.878983	1.000000	0.203647	0.008057	0.047584	0.473306	-0.465244	-0.512030	-0.348715
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.524342	0.172515	-0.172874	-0.101419	-0.162510
Owner	-0.088344	0.008057	0.089216	1.000000	0.182104	-0.053469	0.055687	0.124269	-0.050316
No_Years	-0.236141	0.047584	0.524342	0.182104	1.000000	-0.064315	0.059959	0.039896	-0.000394
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.064315	1.000000	-0.979648	-0.350467	-0.098643
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.059959	-0.979648	1.000000	0.358321	0.091013
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.124269	0.039896	-0.350467	0.358321	1.000000	0.063240
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.050316	-0.000394	-0.098643	0.091013	0.063240	1.000000

In [22]:
```python
#To visualize the co-relation

import seaborn as sns
```

In [23]:
```python
sns.pairplot(final_dataset)
```

Out[23]:
```
<seaborn.axisgrid.PairGrid at 0x583c550>
```

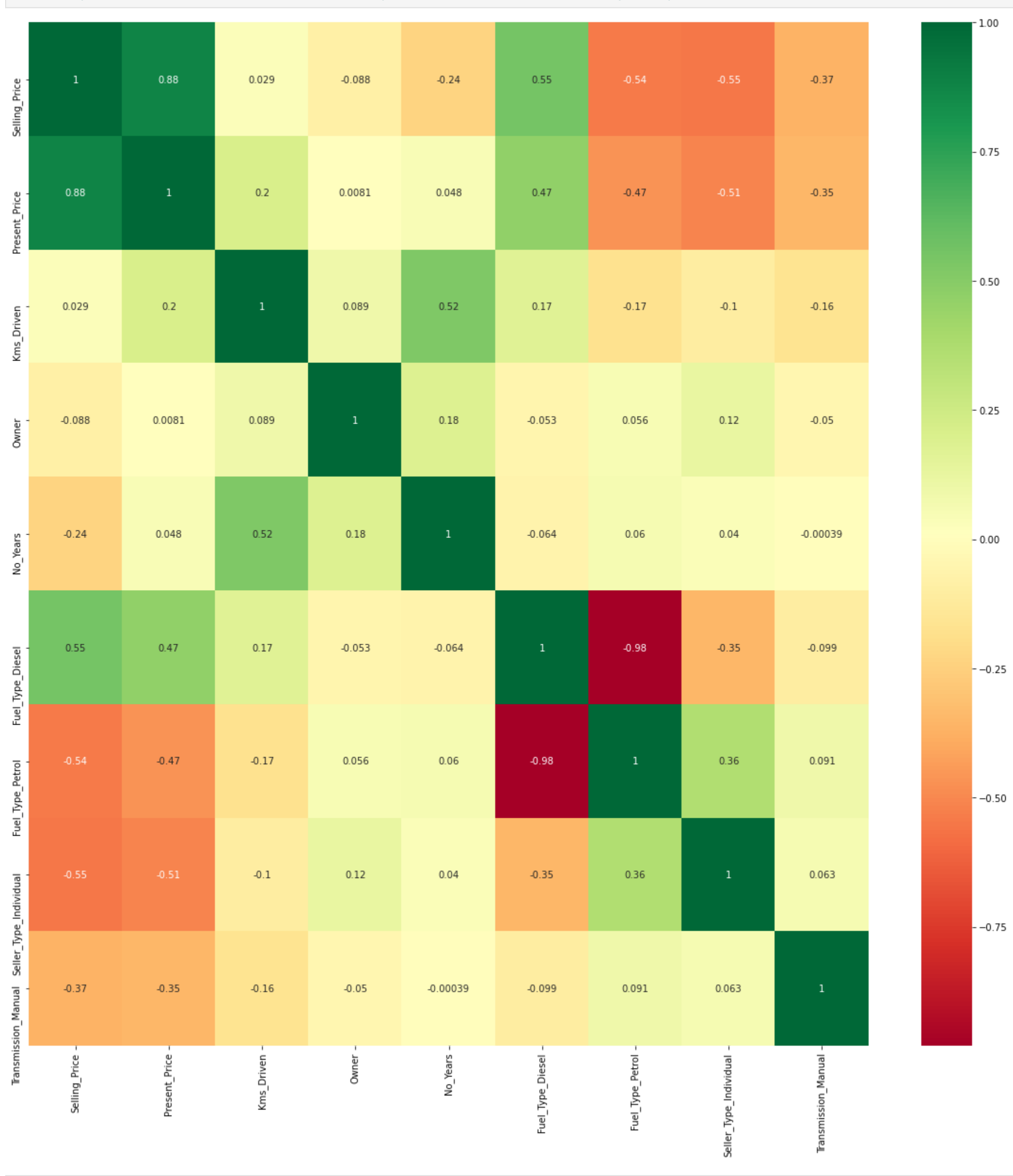


In [24]:
```python
import matplotlib.pyplot as plt
%matplotlib inline
```

In [25]:
```python
#To plot in the form of Heat Map...

corrmat=final_dataset.corr()
top_corr_features=corrmat.index
plt.figure(figsize=(20,20))

#Plotting Heat Map
g=sns.heatmap(final_dataset[top_corr_features].corr(),annot=True,cmap="RdYlGn")
#This shows which features are co-related
#Basically it shows that Green colour is Positively Corelated and red colour is Negatively corelated...
```



In [26]:
```python
final_dataset.head()
#This are the featurs you have, and Selling_Price is Dependent Feature,and all the other are independent feature.
```

Out[26]:

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

In [27]:
```python
#Taking the X Feature
#Independent and dependent feature,

X=final_dataset.iloc[:,1:]
y=final_dataset.iloc[:,0]
```

In [28]:
```python
X.head() #Independent Variable
```

Out[28]:

	Present_Price	Kms_Driven	Owner	No_Years	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	5.59	27000	0	7	0	1	0	1
1	9.54	43000	0	8	1	0	0	1
2	9.85	6900	0	4	0	1	0	1
3	4.15	5200	0	10	0	1	0	1
4	6.87	42450	0	7	1	0	0	1

In [29]:
```python
y.head()  #Dependent Variable
```

Out[29]:
```
0    3.35
1    4.75
2    7.25
3    2.85
4    4.60
Name: Selling_Price, dtype: float64
```

In [30]:
```python
#Feature importance(Understanding Important features..)

from sklearn.ensemble import ExtraTreesRegressor
model=ExtraTreesRegressor()
model.fit(X,y)
```

Out[30]:
```
ExtraTreesRegressor()
```

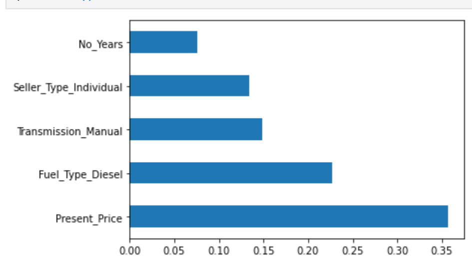
In [31]:
```python
print(model.feature_importances_)

#Shows which feature is importance,like 1st one "Present Features" is having Highest importance,Next importance is "Fuel Type"
```

```
[0.35711444 0.04090748 0.00046391 0.07596428 0.22740164 0.01551388
 0.13380778 0.1488266 ]
```

In [32]:
```python
#Plot Graph of feature Importance for better Visualization...

feat_importances=pd.Series(model.feature_importances_,index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```



In [34]:
```python
##TRain Test Split...

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)
```

In [35]:
```python
X_train.shape
```

Out[35]:
```
(240, 8)
```

In [36]:
```python
#Implementing Random forest Regressor...

from sklearn.ensemble import RandomForestRegressor
rf_random=RandomForestRegressor()
```

In [39]:
```python
##Hypoparameter
import numpy as np
n_estimators=[int(x) for x in np.linspace(start=100, stop=1200, num=12)]
print(n_estimators)
```

```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
```

In [40]:
```python
#Randomize search CV

#Number of trees in random forest
n_estimators=[int(x) for x in np.linspace(start=100, stop=1200, num=12)]

#No of features to consider at every split
max_features=['auto','sqrt']

#Maximum numbers of level of trees
max_depth=[int(x) for x in np.linspace(5, 30,num=6)]

#max_depth.append(None)
#Min number of samples requried to split a node

min_samples_split=[2,5,10,15,100]

#Min numbers of samples req at each leaf node
min_samples_leaf=[1,2,5,10]
```

In [42]:
```python
from sklearn.model_selection import RandomizedSearchCV
```

In [46]:
```python
#Create an random Grid
#This will select the best parameters out of it

random_grid={'n_estimators': n_estimators,
             'max_feature':max_features,
             'max_depth':max_depth,
             'min_samples_split':min_samples_split,
             'min_samples_leaf':min_samples_leaf}
print(random_grid)
```

```
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200], 'max_feature': ['auto', 'sqrt'], 'max_depth': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]}
```

In [47]:
```python
#Use the random grid to search for best hypoparameter
#First create the base model to tune

rf=RandomForestRegressor()
```

In [48]:
```python
rf=random=RandomizedSearchCV(estimator=rf,param_distributions=random_grid,scoring='neg_mean_squared_error',n_iter=10,cv=5,verbose=2,random_state=42
```

In [49]:
```python
rf_random.fit(X_train,y_train)
```

Out[49]:
```
RandomForestRegressor()
```

In [50]:
```python
#Doing Prediction

predictions=rf_random.predict(X_test)
```

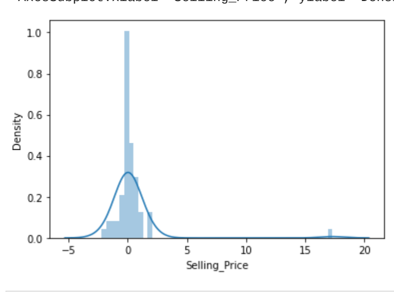
In [52]:
```python
predictions
```

Out[52]:
```
array([ 2.686 ,  0.4509,  6.1573,  1.1883,  3.198 ,  5.6605, 21.515 ,
        0.2053, 22.26  ,  5.0365,  5.0925,  1.8245,  0.2635,  6.5755,
        1.0396,  0.4683,  0.2226,  0.4565,  0.5473,  5.262 , 17.7109,
        3.0245,  4.7515,  7.0904,  6.0235,  4.7912, 21.1223,  4.0295,
        5.1915,  2.7095,  5.6625,  9.008 ,  7.9793,  5.93  ,  0.4531,
       11.1044,  7.4815,  5.168 ,  7.9459,  7.9459,  1.5525, 10.7526,
        0.5057,  0.6509,  5.8835,  8.1615,  1.0613,  5.706 ,  3.0285,
        2.764 ,  6.3418,  0.3984,  3.2695,  0.6718,  5.2275,  4.9905,
        0.7918,  6.5245,  3.991 ,  5.076 ,  4.0585])
```

In [53]:
```python
sns.distplot(y_test-predictions)
```

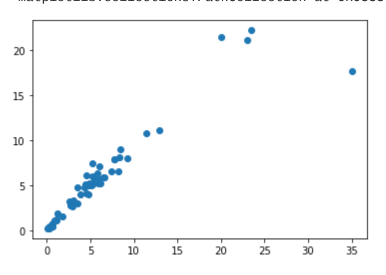
```
C:\Users\win7\anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[53]:
```
<AxesSubplot:xlabel='Selling_Price', ylabel='Density'>
```



In [54]:
```python
plt.scatter(y_test,predictions)
```

Out[54]:
```
<matplotlib.collections.PathCollection at 0x9833e68>
```



In [ ]: