In [90]:
```
import numpy as np
import pandas as pd
from sklearn.compose import make_column_selector
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
import pickle
from sklearn import metrics
```

In [31]:
```
df=pd.read_csv(r"C:/Users/Asus/Downloads/car data.csv")
df.head()
```

Out[31]:

	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 [32]:
```
df.shape
```

Out[32]:
```
(301, 9)
```

In [52]:
```
print(df['Fuel_Type'].unique())
print(df['Seller_Type'].unique())
print(df['Transmission'].unique())
print(df['Owner'].unique())
```

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

In [34]:
```
df.isnull().sum()
```

Out[34]:
```
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 [35]:
```
df.columns
```

Out[35]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

In [37]:
```
final_dataset=df[['Year','Selling_Price','Present_Price','Kms_Driven','Fuel_Type', 'Seller_Type', 'T
```

In [38]:
```
final_dataset.head()
```

Out[38]:

	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 [41]:
```
final_dataset['Current_Year']=2024
```

In [42]:
```
final_dataset
```

Out[42]:

	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	2024
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2024
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2024
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2024
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2024
...	...	...	...	...	...	...	...	...	...
296	2016	9.50	11.60	33988	Diesel	Dealer	Manual	0	2024
297	2015	4.00	5.90	60000	Petrol	Dealer	Manual	0	2024
298	2009	3.35	11.00	87934	Petrol	Dealer	Manual	0	2024
299	2017	11.50	12.50	9000	Diesel	Dealer	Manual	0	2024
300	2016	5.30	5.90	5464	Petrol	Dealer	Manual	0	2024

301 rows × 9 columns

In [43]:
```
final_dataset['no_year']=final_dataset['Current_Year'] - final_dataset['Year']
```

In [44]:
```
final_dataset.head()
```

Out[44]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year	no_year
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2024	10
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2024	11
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2024	7
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2024	13
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2024	10

In [45]:
```
final_dataset.drop(['Year'], axis=1, inplace=True)
```

In [46]:
```
final_dataset.head()
```

Out[46]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year	no_year
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	2024	10
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	2024	11
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	2024	7
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	2024	13
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	2024	10

In [48]:
```
final_dataset.drop(['Current_Year'], axis=1, inplace=True)
```

In [49]:
```
final_dataset.head()
```

Out[49]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	no_year
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	10
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	11
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	7
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	13
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	10

In [50]:
```
final_dataset=pd.get_dummies(final_dataset, drop_first=True)
```

In [51]:
```
final_dataset.head()
```

Out[51]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmis
0	3.35	5.59	27000	0	10	0	1	0	
1	4.75	9.54	43000	0	11	1	0	0	
2	7.25	9.85	6900	0	7	0	1	0	
3	2.85	4.15	5200	0	13	0	1	0	
4	4.60	6.87	42450	0	10	1	0	0	

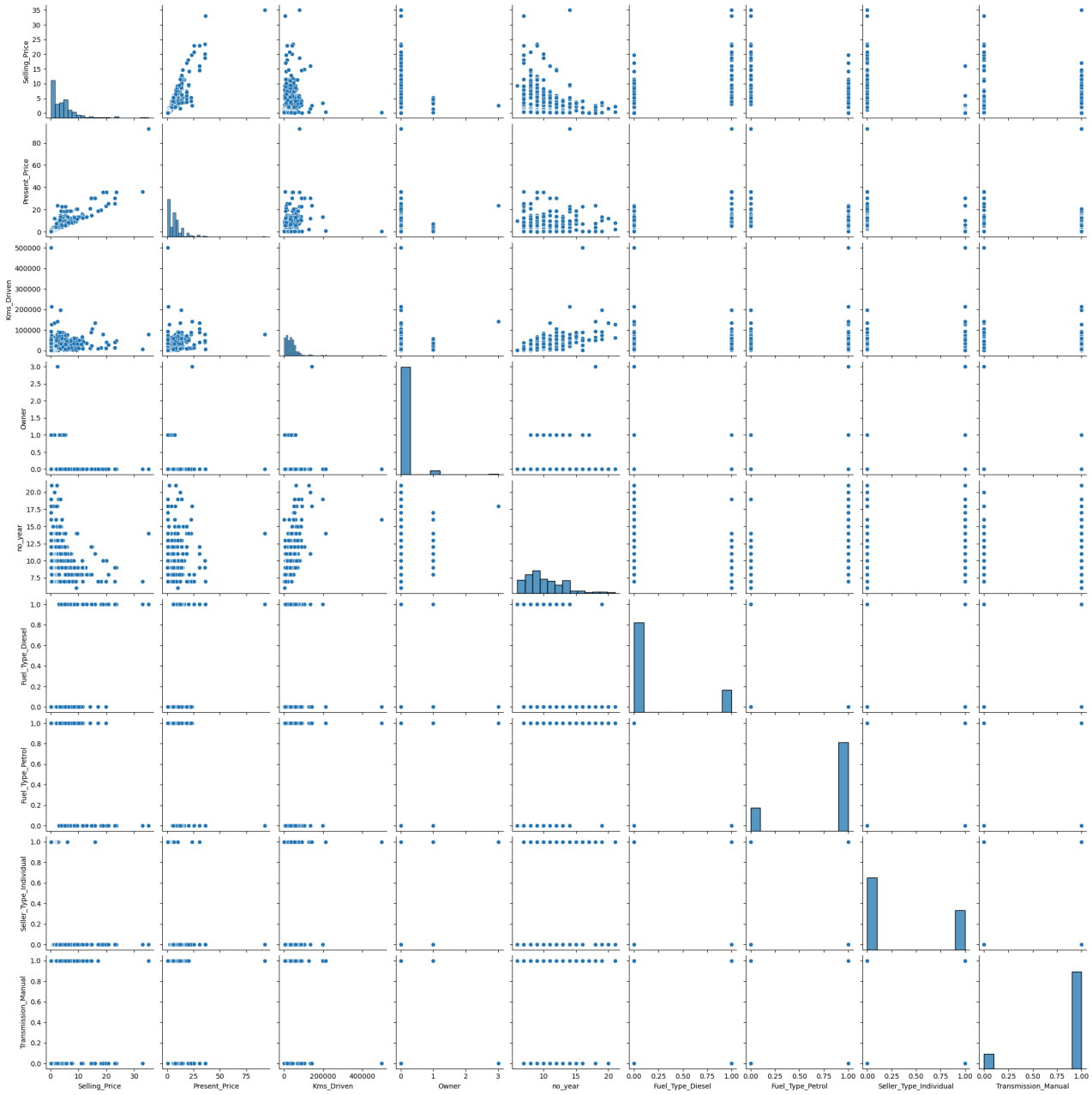
In [53]:
```
final_dataset.corr()
```

Out[53]:

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

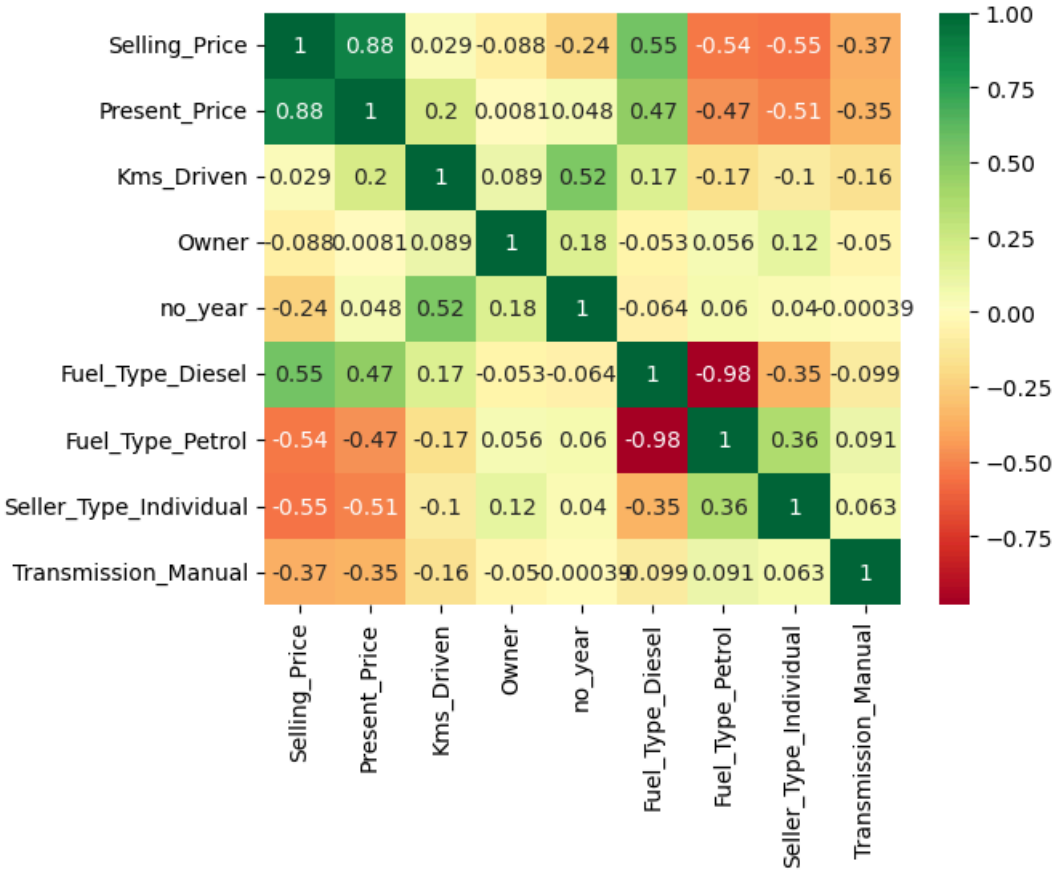
In [57]:
```
sns.pairplot(final_dataset)
```

Out[57]: <seaborn.axisgrid.PairGrid at 0x21945dee5f0>



In [61]:
```
corrmat=final_dataset.corr()
top_corr_features=corrmat.index
plt.figure(figsize=(20,20))
plt.show()
g=sns.heatmap(final_dataset[top_corr_features].corr(), annot=True, cmap='RdYlGn')
```

```
<Figure size 2000x2000 with 0 Axes>
```



In [62]:
```
x=final_dataset.iloc[:,1:]
y=final_dataset.iloc[:,0]
```

In [63]:
```
x.head()
```

Out[63]:

	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	5.59	27000	0	10	0	1	0	1
1	9.54	43000	0	11	1	0	0	1
2	9.85	6900	0	7	0	1	0	1
3	4.15	5200	0	13	0	1	0	1
4	6.87	42450	0	10	1	0	0	1

In [64]:
```
y.head()
```

Out[64]:
```
0    3.35
1    4.75
2    7.25
3    2.85
4    4.60
Name: Selling_Price, dtype: float64
```

Feature Importance

In [66]:
```
model=ExtraTreesRegressor()
model.fit(x,y)
```

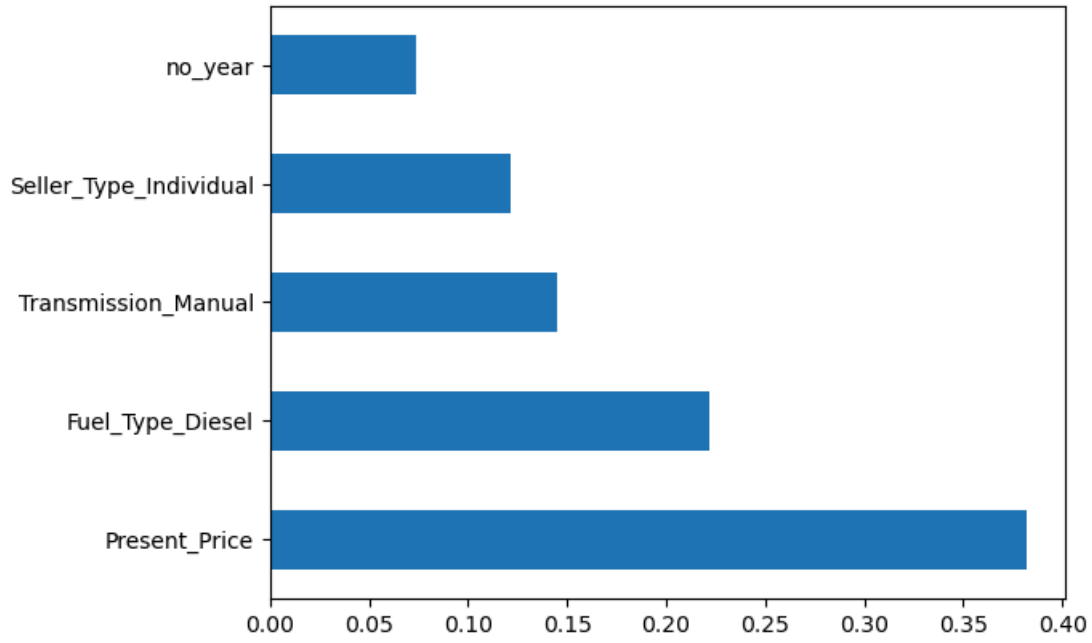
Out[66]:
```
ExtraTreesRegressor
ExtraTreesRegressor()
```

In [67]:
```
print(model.feature_importances_)
```

```
[0.38227273 0.04363357 0.00041062 0.07340178 0.2222526  0.01200687
 0.12133866 0.14468317]
```

Graph plotting for feature importance

In [70]:
```
feat_importances=pd.Series(model.feature_importances_, index=x.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```



In [73]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.25)
```

In [74]:
```
x_train
```

Out[74]:

	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
277	13.60	21780	0	9	0	1	0	1
219	9.40	36000	0	12	0	1	0	1
3	4.15	5200	0	13	0	1	0	1
58	6.80	39485	1	10	0	1	0	1
208	8.10	3435	0	7	0	1	0	1
...	...	...	...	...	...	...	...	...
19	7.98	41442	0	14	0	1	0	1
233	5.70	53000	0	11	1	0	0	1
134	0.81	11800	0	7	0	1	1	1
159	0.51	4000	0	7	0	1	1	0
103	1.60	1200	0	7	0	1	1	1

225 rows × 8 columns

In [ ]:
```
rf_random=RandomForestRegressor()
```

In [77]:
```
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 [78]:
```
n_estimators=[int(x) for x in np.linspace(start=100, stop=1200, num=12)]
max_features=['auto','sqrt']
max_depth=[int(x) for x in np.linspace(5,30, num=6)]
min_samples_split=[2,5,10,15,100]
min_samples_leaf=[1,2,5,10]
```

In [80]:
```
random_grid={'n_estimators':n_estimators,
             'max_features':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_features':
['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 [81]:
```
rf=RandomForestRegressor()
```

In [82]:
```
distributions = random_grid, scoring="neg_mean_squared_error", n_iter = 10,cv = 5, verbose = 2, rando
```

In [83]:
```
rf_random.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   5.4s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   1.3s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   1.0s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   1.0s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=9
00; total time=   1.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=
1100; total time=   1.6s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=
1100; total time=   1.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=
1100; total time=   1.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=
1100; total time=   1.2s
```

In [84]:
```
predictions=rf_random.predict(x_test)
```

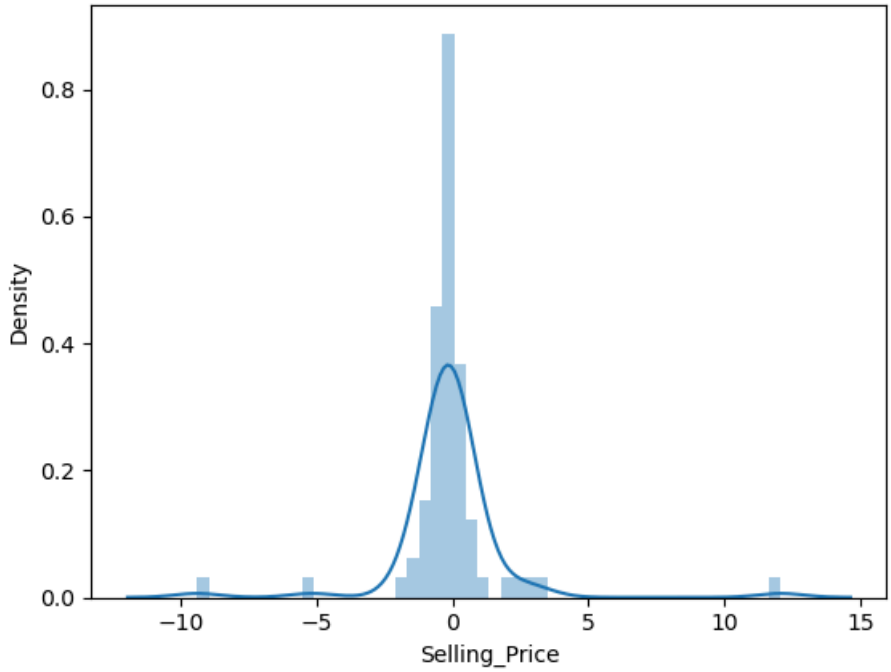
In [85]:
```
predictions
```

Out[85]:
```
array([ 6.60346,  0.70214,  0.57272,  0.70571, 10.05715,  4.97994,
       10.41667, 20.63648,  4.57813,  6.91765, 20.91716,  8.6521 ,
        0.36799,  7.76065,  3.52958,  0.70966,  4.70347,  1.46245,
        0.86414,  4.70028,  5.14682,  4.04439,  0.392  ,  9.03276,
        9.55705,  0.70879,  1.2953 ,  5.65995,  5.53603, 20.30558,
        6.89455,  8.99825,  3.83858,  3.73244,  3.62484,  0.73732,
        5.88186, 10.83142,  4.88446,  0.61599,  4.8917 ,  4.4149 ,
        7.38923, 10.73743,  5.79102,  0.56917,  0.64415,  3.14179,
        9.39761,  0.79671, 17.60473,  0.82279,  6.60796,  7.23036,
        0.8431 , 12.8801 ,  5.44445,  3.03025,  6.28978,  4.61285,
       15.76418,  3.66477,  0.54735,  0.3327 ,  0.50581,  4.53861,
        0.7543 ,  5.04296,  8.27506,  6.99284,  6.63255,  4.50785,
        0.92424,  0.53948,  0.32137,  0.78188])
```

In [86]:
```
sns.distplot(y_test-predictions)
```

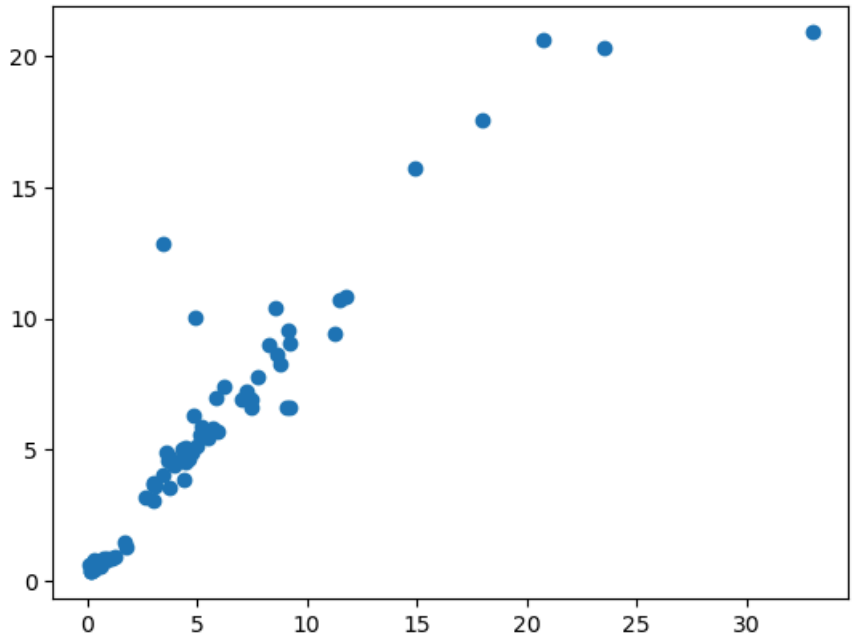
```
C:\Users\Asus\AppData\Local\Temp\ipykernel_12152\2131792714.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(y_test-predictions)
```

Out[86]:
```
<Axes: xlabel='Selling_Price', ylabel='Density'>
```



In [87]:
```
plt.scatter(y_test,predictions)
```

Out[87]:
```
<matplotlib.collections.PathCollection at 0x2194e025210>
```



In [89]:
```
```

In [91]:
```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 0.8513342105263125
MSE: 4.067322150584201
RMSE: 2.016760310642839
```

In [92]:
```
file=open('random_forest_regression_model.pkl', 'wb')
pickle.dump(rf_random, file)
```

In [ ]:
```
```