In [2]:
```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
```

In [3]:
```
1 df=pd.read_csv(r"C:\Users\range...car_prediction_data.csv")
```

In [4]:
```
1 df.head()
```

	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 [5]:
```
1 df.shape
```

```
(301, 9)
```

In [6]:
```
print(df['Seller_Type'].unique())
print(df['Fuel_Type'].unique())
print(df['Transmission'].unique())
print(df['Owner'].unique())
```

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

In [7]:
```
df.isnull().sum()
```

```
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 [8]:
```
1 df.describe()
```

	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 [9]:
```
1 final_dataset=df[['Year','Selling_Price','Present_Price','Kms_Driven','Fuel_Type','Seller_Type','Transm
```

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

	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]:
```
1 final_dataset['Current_Year']=2021
```

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

	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]:
```
1 final_dataset['no_year']=final_dataset['Current_Year']- final_dataset['Year']
```

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

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

In [16]:
```
1 final_dataset.head()
```

	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	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]:
```
1 final_dataset=pd.get_dummies(final_dataset,drop_first=True)
```

In [18]:
```
1 final_dataset.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Current_Year	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_
0	3.35	5.59	27000	0	2021	7	False	True	False
1	4.75	9.54	43000	0	2021	8	True	False	False
2	7.25	9.85	6900	0	2021	4	False	True	False
3	2.85	4.15	5200	0	2021	10	False	True	False
4	4.60	6.87	42450	0	2021	7	True	False	False

In [19]:
```
1 final_dataset=final_dataset.drop(['Current_Year'],axis=1)
```

In [20]:
```
1 final_dataset.head()
```

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

In [21]:
```
1 final_dataset.corr()
```

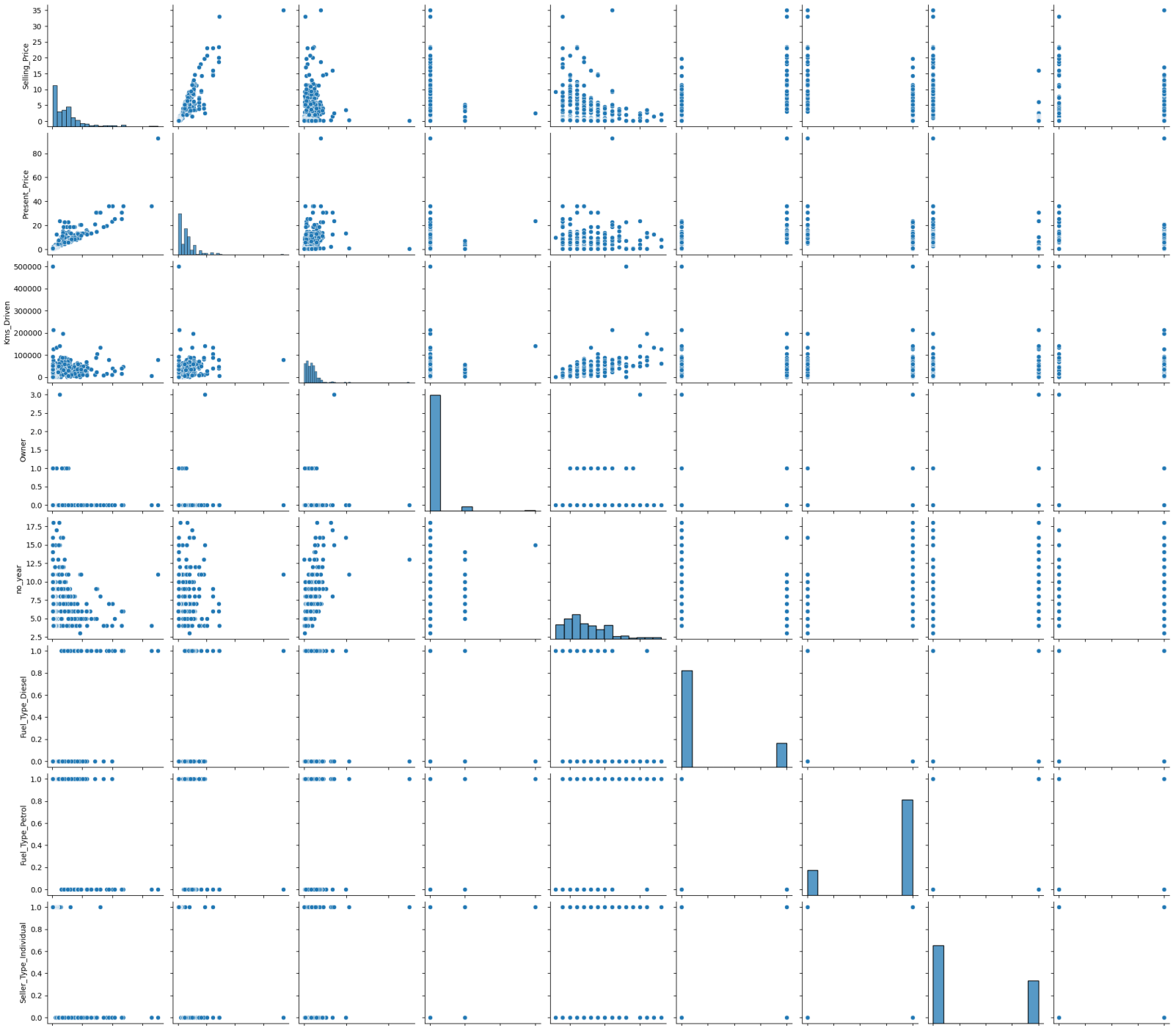
	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.236141	0.552339	-0.540571	-0.5507
Present_Price	0.878983	1.000000	0.203647	0.008057	0.047584	0.473306	-0.465244	-0.5120
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.524342	0.172515	-0.172874	-0.1014
Owner	-0.088344	0.008057	0.089216	1.000000	0.182104	-0.053469	0.055687	0.1242(
no_year	-0.236141	0.047584	0.524342	0.182104	1.000000	-0.064315	0.059959	0.0398{
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.064315	1.000000	-0.979648	-0.3504
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.059959	-0.979648	1.000000	0.3583(
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.124269	0.039896	-0.350467	0.358321	1.0000(
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.050316	-0.000394	-0.098643	0.091013	0.0632(

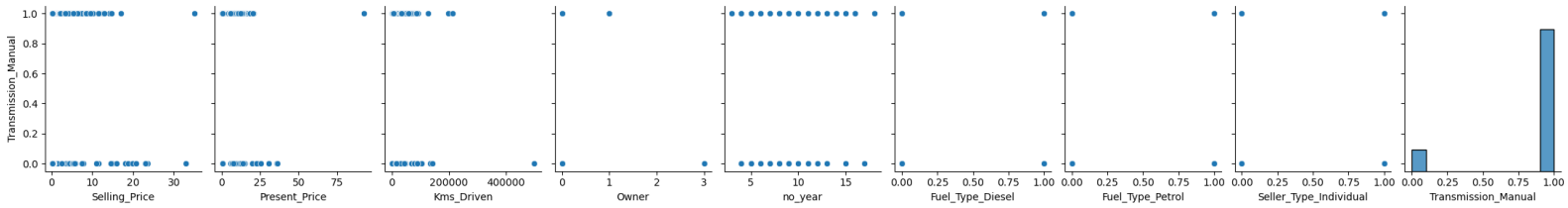
In [22]:
```
1 import seaborn as sns
```

In [23]:

```
1 sns.pairplot(final_dataset)
```

```
<__array_function__ internals>:200: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for compatibility.
<__array_function__ internals>:200: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for compatibility.
<__array_function__ internals>:200: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for compatibility.
<__array_function__ internals>:200: RuntimeWarning: Converting input from bool to <class 'numpy.uint8'> for compatibility.
C:\Users\range\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)


<seaborn.axisgrid.PairGrid at 0x1c9456e1e50>
```


Selling_Price
Present_Price
Kms_Driven
Owner
no_year
Fuel_Type_Diesel
Fuel_Type_Petrol
Seller_Type_Individual



In [25]:
```
X=final_dataset.iloc[:,1:]
y=final_dataset.iloc[:,0]
```

In [26]:
```
X['Owner'].unique()
```

```
array([0, 1, 3], dtype=int64)
```

In [27]:
```
X.head()
```

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

In [28]:
```
1 y.head()
```

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

Feature Importance

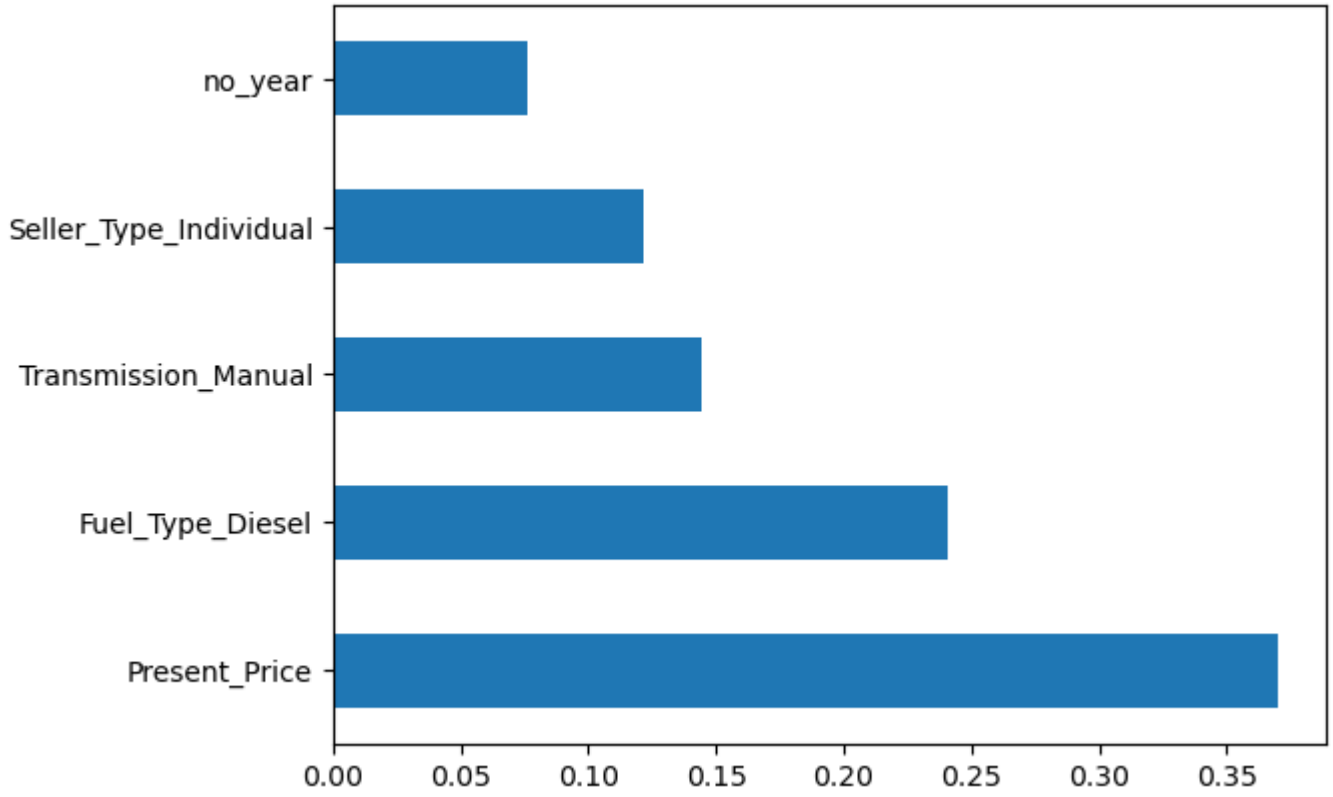
In [29]:
```
1 from sklearn.ensemble import ExtraTreesRegressor
2 model = ExtraTreesRegressor()
3 model.fit(X,y)
```

```
▾ ExtraTreesRegressor
ExtraTreesRegressor()
```

In [30]:
```
1 print(model.feature_importances_)
```

```
[0.37028483 0.03825943 0.00078987 0.07634868 0.24050773 0.00745442
 0.12198921 0.14436582]
```

In [31]:
```
1 feat_importances = pd.Series(model.feature_importances_, index=X.columns)
2 feat_importances.nlargest(5).plot(kind='barh')
3 plt.show()
```



In [32]:
```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Using Random Forest Regressor

In [33]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [34]:
```
regressor=RandomForestRegressor()
```

In [35]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [36]:
```
 #Randomized Search CV

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 [37]:
```
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, 1
5, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]}
```

In [38]:
```
rf = RandomForestRegressor()
```

In [39]:
```
# Use the random grid to search for best hyperparameters
rf=RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error
```

In [40]:
```
rf.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=900; total time=   1.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   1.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   1.4s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   1.3s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; 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=   1.7s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   1.7s
[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=   1.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   1.9s


C:\Users\range\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has been deprecated in 1.
1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also the def
ault value for RandomForestRegressors and ExtraTreesRegressors.
  warn(


[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   0.5s


C:\Users\range\anaconda3\Lib\site-packages\sklearn\ensemble\_forest.py:413: FutureWarning: `max_features='auto'` has been deprecated in 1.
1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or remove this parameter as it is also the def
```

In [41]:
```
rf.best_params_
```

```
{'n_estimators': 1000,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 25}
```

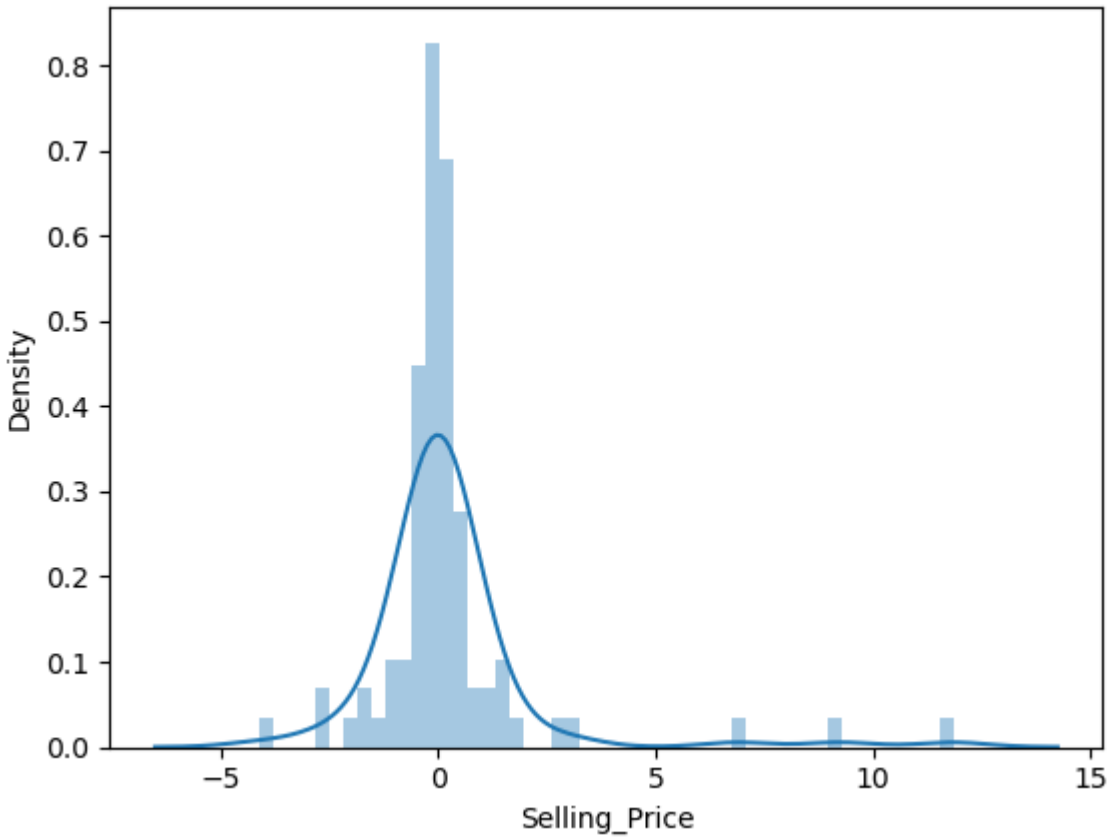
In [42]:
```
1 rf.best_score_
```

```
-3.9930132501533486
```

In [43]:
```
1 predictions=rf.predict(X_test)
```

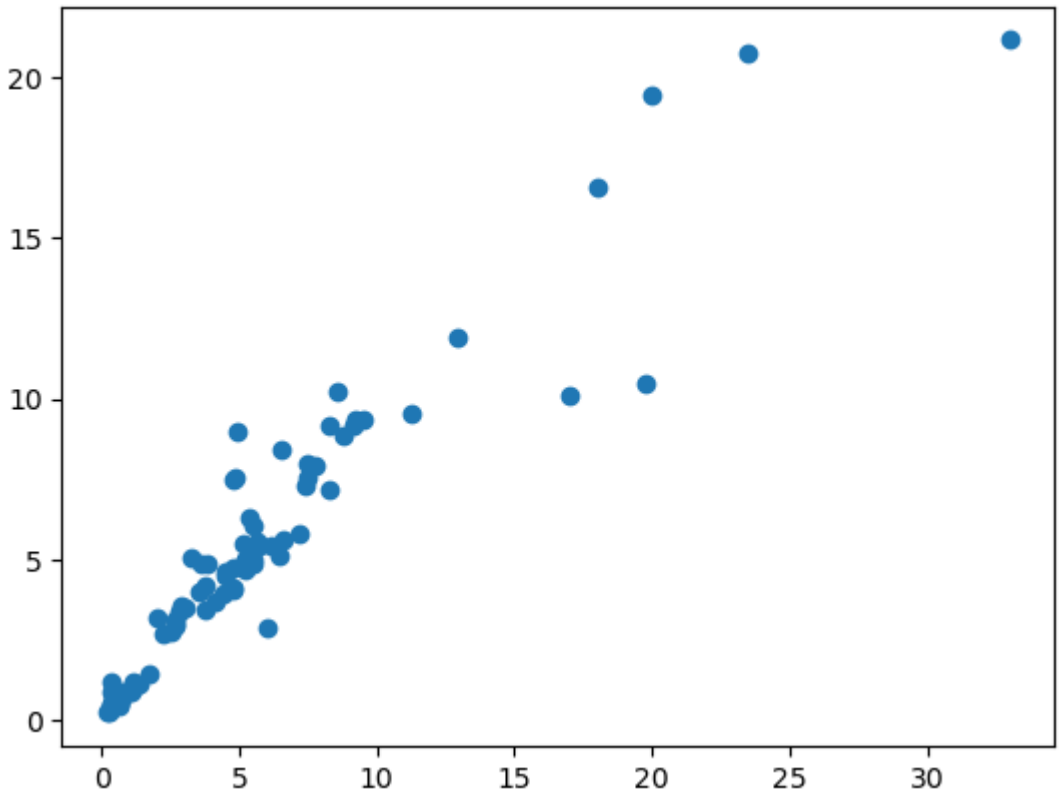
In [44]:
```
1 sns.distplot(y_test-predictions)
```

```
C:\Users\range\AppData\Local\Temp\ipykernel_1276\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)


<Axes: xlabel='Selling_Price', ylabel='Density'>
```


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
−5
0
5
10
15
Selling_Price

In [45]:
```
1 plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x1c94f695390>
```



In [46]:
```
1 from sklearn import metrics
```

In [47]:
```
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: 0.887862417582416
MSE: 3.9217544933780215
RMSE: 1.9803420142435046
```

Using XGBoost Regressor

In [48]:
```
import xgboost as xgb
from scipy.stats import uniform, randint
```

In [49]:
```
xgb_model = xgb.XGBRegressor(objective="reg:linear", random_state=42)
```

In [50]:
```
params = {
    "gamma": uniform(0, 0.5),
    "learning_rate": uniform(0.03, 0.3), # default 0.1
    "max_depth": randint(2, 6), # default 3
    "n_estimators": randint(100, 150), # default 100
    "subsample": uniform(0.6, 0.4)
}
```

In [51]:
```
xgb = RandomizedSearchCV(estimator = xgb_model, param_distributions = params,scoring='neg_mean_squared_
```

In [52]:
```
1 xgb.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END gamma=0.18727005942368125, learning_rate=0.3152142919229748, max_depth=4, n_estimators=107, subsample=0.8394633936788146; total t
ime=   0.0s
[CV] END gamma=0.18727005942368125, learning_rate=0.3152142919229748, max_depth=4, n_estimators=107, subsample=0.8394633936788146; total t
ime=   0.0s


C:\Users\range\anaconda3\Lib\site-packages\xgboost\core.py:160: UserWarning: [12:18:13] WARNING: C:\buildkite-agent\builds\buildkite-windo
ws-cpu-autoscaling-group-i-07f6e447eee219473-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecate
d in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
C:\Users\range\anaconda3\Lib\site-packages\xgboost\core.py:160: UserWarning: [12:18:13] WARNING: C:\buildkite-agent\builds\buildkite-windo
ws-cpu-autoscaling-group-i-07f6e447eee219473-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecate
d in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)
C:\Users\range\anaconda3\Lib\site-packages\xgboost\core.py:160: UserWarning: [12:18:13] WARNING: C:\buildkite-agent\builds\buildkite-windo
ws-cpu-autoscaling-group-i-07f6e447eee219473-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecate
d in favor of reg:squarederror.
  warnings.warn(smsg, UserWarning)


[CV] END gamma=0.18727005942368125, learning_rate=0.3152142919229748, max_depth=4, n_estimators=107, subsample=0.8394633936788146; total t
ime=   0.0s
[CV] END gamma=0.18727005942368125, learning_rate=0.3152142919229748, max_depth=4, n_estimators=107, subsample=0.8394633936788146; total t
```

In [53]:
```
1 xgb.best_score_
```

```
-1.9114686351785255
```

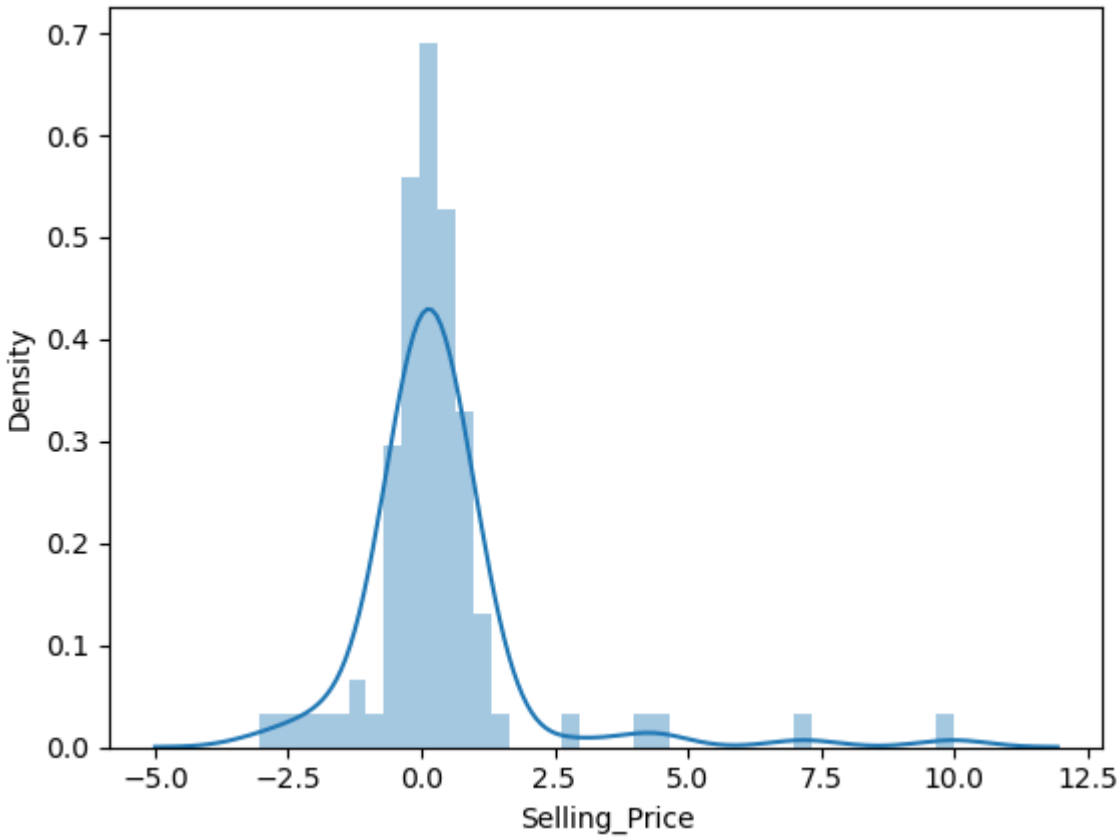
In [54]:
```
1 xgb.best_params_
```

```
{'gamma': 0.3005575058716044,
 'learning_rate': 0.24242177333881365,
 'max_depth': 3,
 'n_estimators': 101,
 'subsample': 0.8887995089067299}
```

In [55]:
```
1 predictions=xgb.predict(X_test)
```

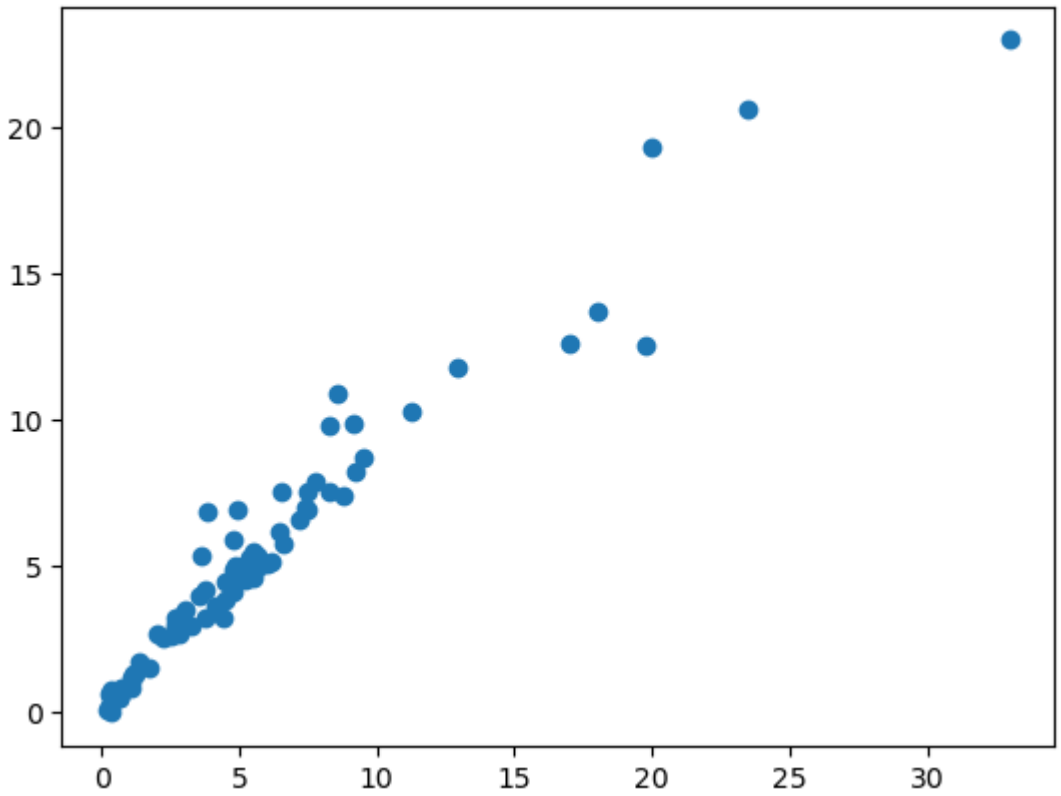
In [56]:
```
1 sns.distplot(y_test-predictions)
```

```
C:\Users\range\AppData\Local\Temp\ipykernel_1276\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)


<Axes: xlabel='Selling_Price', ylabel='Density'>
```


0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
Selling_Price

In [57]:
```
1 plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x1c95631e9d0>
```



In [58]:
```
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: 0.7972087596921802
MSE: 2.6811867849257176
RMSE: 1.637432986392334
```

Using Catboost Regresssor

In [62]:
```
from catboost import CatBoostRegressor
```

In [63]:
```
1 !pip install catboost
```

```
Requirement already satisfied: catboost in c:\users\range\anaconda3\lib\site-packages (1.2.3)
Requirement already satisfied: graphviz in c:\users\range\anaconda3\lib\site-packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in c:\users\range\anaconda3\lib\site-packages (from catboost) (3.7.2)
Requirement already satisfied: numpy>=1.16.0 in c:\users\range\anaconda3\lib\site-packages (from catboost) (1.24.3)
Requirement already satisfied: pandas>=0.24 in c:\users\range\anaconda3\lib\site-packages (from catboost) (2.0.3)
Requirement already satisfied: scipy in c:\users\range\anaconda3\lib\site-packages (from catboost) (1.11.1)
Requirement already satisfied: plotly in c:\users\range\anaconda3\lib\site-packages (from catboost) (5.9.0)
Requirement already satisfied: six in c:\users\range\anaconda3\lib\site-packages (from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\range\anaconda3\lib\site-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\range\anaconda3\lib\site-packages (from pandas>=0.24->catboost) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\range\anaconda3\lib\site-packages (from pandas>=0.24->catboost) (2023.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\range\anaconda3\lib\site-packages (from matplotlib->catboost) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\range\anaconda3\lib\site-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\range\anaconda3\lib\site-packages (from matplotlib->catboost) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\range\anaconda3\lib\site-packages (from matplotlib->catboost) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\range\anaconda3\lib\site-packages (from matplotlib->catboost) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\range\anaconda3\lib\site-packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\range\anaconda3\lib\site-packages (from matplotlib->catboost) (3.0.9)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\range\anaconda3\lib\site-packages (from plotly->catboost) (8.2.2)


[notice] A new release of pip is available: 23.3.2 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [64]:
```
1 cb=CatBoostRegressor()
```

In [65]:
```
1 grid = {'learning_rate': [0.03, 0.1],
2         'depth': [4, 6, 10],
3         'l2_leaf_reg': [1, 3, 5, 7, 9]}
```

In [66]:
```
1 cb = RandomizedSearchCV(estimator = cb, param_distributions = grid,scoring='neg_mean_squared_error', n_
```

In [67]:
```
1 cb.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
0:      learn: 4.9398622        total: 136ms    remaining: 2m 15s
1:      learn: 4.7805986        total: 143ms    remaining: 1m 11s
2:      learn: 4.5730251        total: 146ms    remaining: 48.7s
3:      learn: 4.4161829        total: 149ms    remaining: 37s
4:      learn: 4.2856056        total: 159ms    remaining: 31.7s
5:      learn: 4.1268798        total: 161ms    remaining: 26.6s
6:      learn: 3.9895888        total: 164ms    remaining: 23.3s
7:      learn: 3.8391546        total: 169ms    remaining: 20.9s
8:      learn: 3.7310291        total: 182ms    remaining: 20s
9:      learn: 3.6173828        total: 183ms    remaining: 18.1s
10:     learn: 3.4712492        total: 184ms    remaining: 16.6s
11:     learn: 3.3483253        total: 187ms    remaining: 15.4s
12:     learn: 3.2450271        total: 194ms    remaining: 14.8s
13:     learn: 3.1400164        total: 196ms    remaining: 13.8s
14:     learn: 3.0260343        total: 198ms    remaining: 13s
15:     learn: 2.9346526        total: 199ms    remaining: 12.3s
16:     learn: 2.8376114        total: 211ms    remaining: 12.2s
17:     learn: 2.7572800        total: 213ms    remaining: 11.6s
18:     learn: 2.6724960        total: 224ms    remaining: 11.6s
19:     learn: 2.6150300        total: 234ms    remaining: 11.5s
20:     learn: 2.5479388        total: 237ms    remaining: 11s
21:     learn: 2.4920401        total: 240ms    remaining: 10.7s
22:     learn: 2.4787333        total: 241ms    remaining: 10.2s
```

In [68]:
```
1 cb.best_score_
```

```
-3.2179271237932716
```

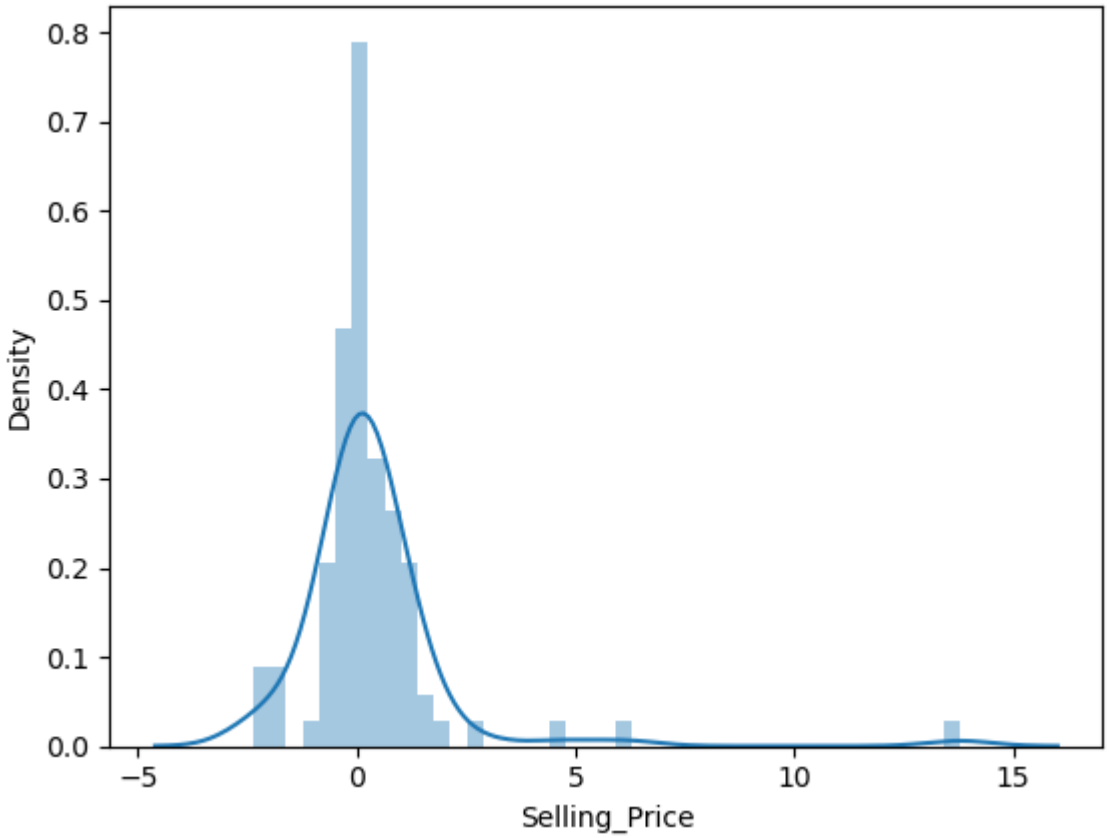
In [69]:
```
1 cb.best_params_
```

```
{'learning_rate': 0.1, 'l2_leaf_reg': 9, 'depth': 4}
```

In [70]:
```
1 predictions=cb.predict(X_test)
```

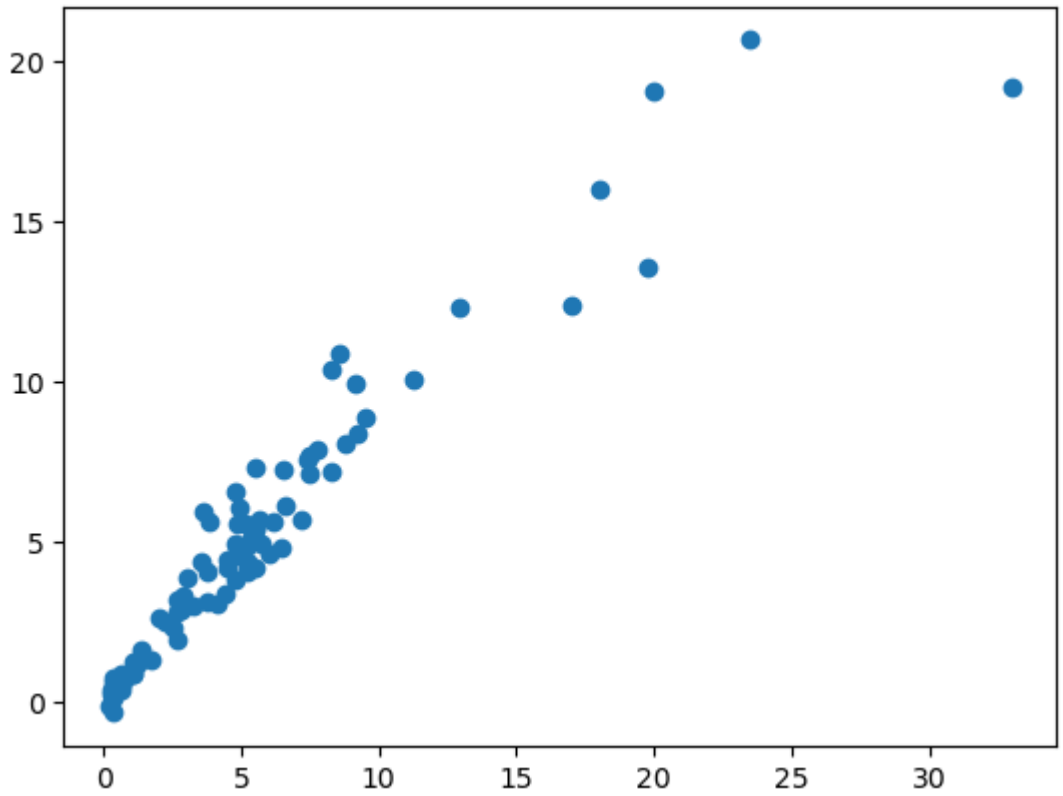
In [71]:
```
1 sns.distplot(y_test-predictions)
```

```
C:\Users\range\AppData\Local\Temp\ipykernel_1276\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)


<Axes: xlabel='Selling_Price', ylabel='Density'>
```


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Density
−5
0
5
10
15
Selling_Price

In [72]:
```
1 plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x1c958859590>
```



In [73]:

```
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: 0.8568669682476366
MSE: 3.4731227057809524
RMSE: 1.8636315906801302
```

In [ ]:

```

```