```
import pandas as pd
```

```
df=pd.read_csv("car data.csv")
```

```
df.head()
```

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

```
df.shape
```

```
(301, 9)
```

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

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

```
##check missing or null values
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
```

```
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

```
df.columns
```

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

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

```
final_dataset.head()
```

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

```
final_dataset['Current year']=2020
```

```
final_dataset.head()
```

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

```
final_dataset['no year']=final_dataset['Current year']-final_dataset['Year']
```

```
final_dataset.head()
```

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

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

```
final_dataset.head()
```

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

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

```
final_dataset.head()
```

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

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

```
final_dataset.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	no year	Fuel_Type_Diesel	Fuel_Typ
0	3.35	5.59	27000	0	6	0	
1	4.75	9.54	43000	0	7	1	
2	7.25	9.85	6900	0	3	0	

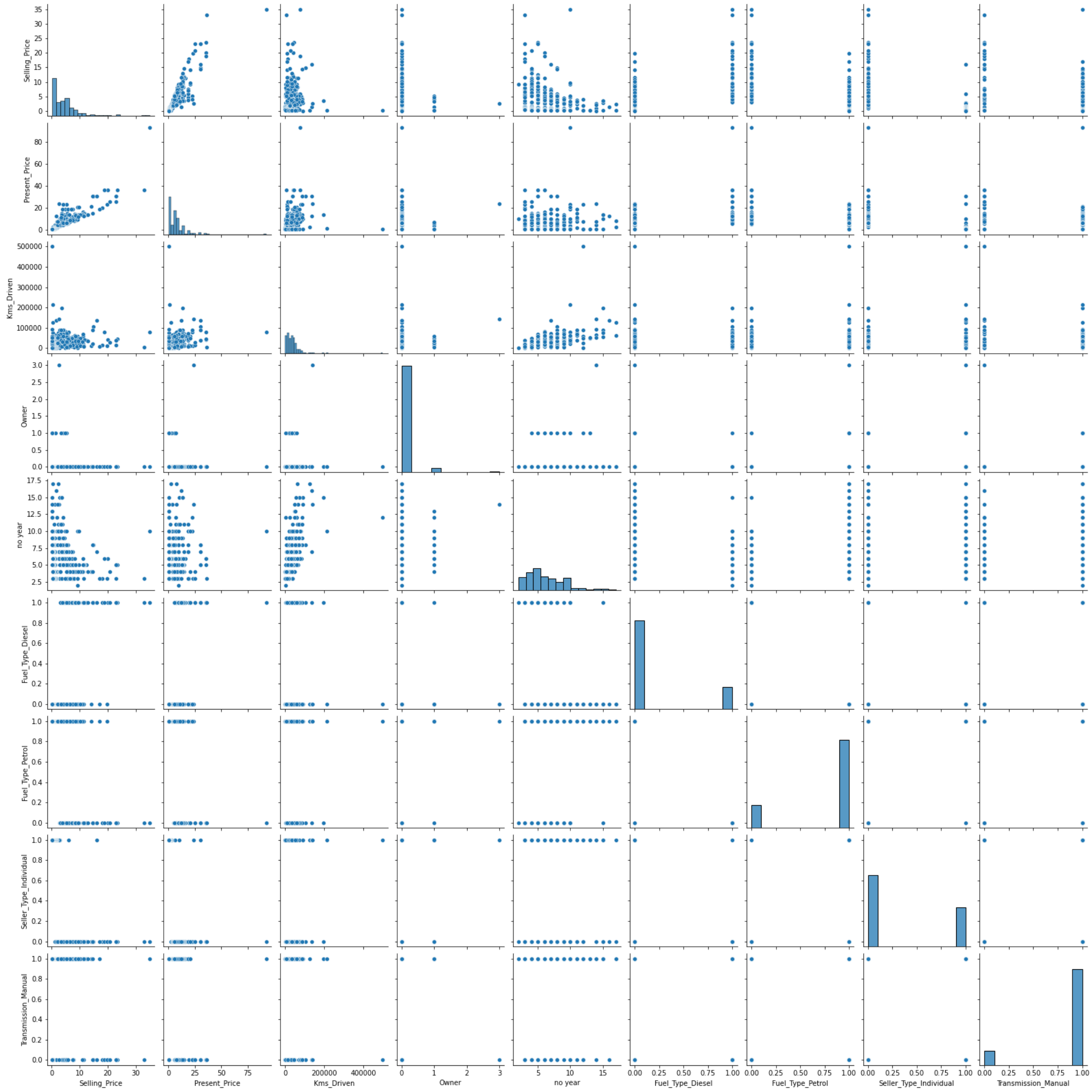
```
final_dataset.corr()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	no year
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.236141
Present_Price	0.878983	1.000000	0.203647	0.008057	0.047584
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.524342
Owner	-0.088344	0.008057	0.089216	1.000000	0.182104
no year	-0.236141	0.047584	0.524342	0.182104	1.000000
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.064315
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.059959
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.124269	0.039896
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.050316	-0.000394

```
import seaborn as sns
```

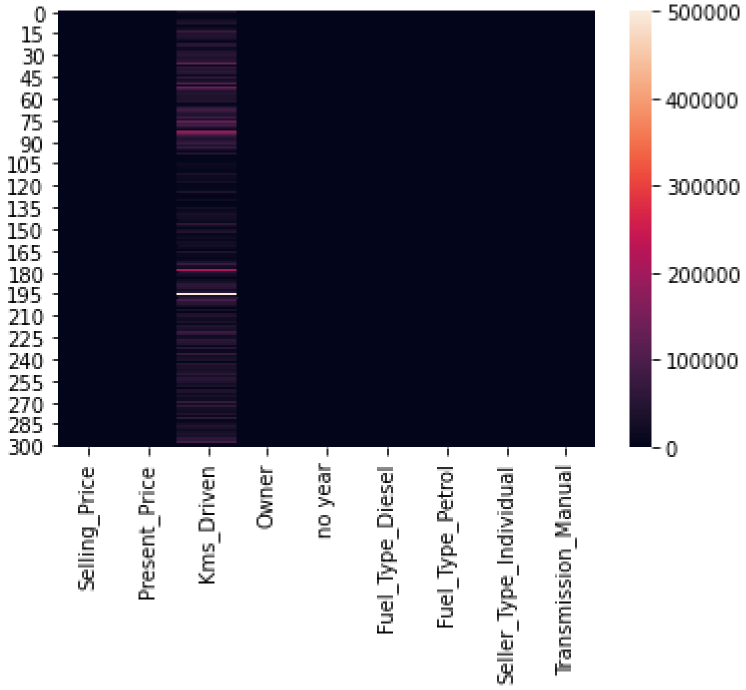
```
sns.pairplot(final_dataset)
```

<seaborn.axisgrid.PairGrid at 0x7f775f888150>



```
sns.heatmap(final_dataset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7744a94c10>
```

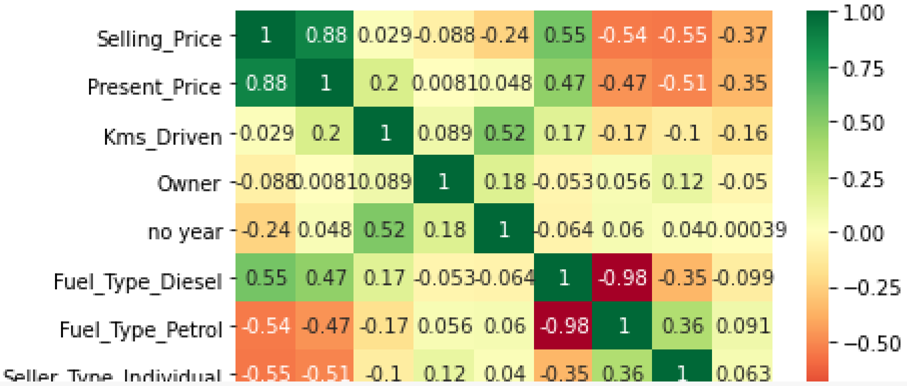



```
import matplotlib.pyplot as plt
%matplotlib inline
```

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

```
<Figure size 1440x1440 with 0 Axes>
<Figure size 1440x1440 with 0 Axes>
```

```
#heatmap
g=sns.heatmap(final_dataset[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```




```
final_dataset.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	no year	Fuel_Type_Diesel	Fuel_Typ
0	3.35	5.59	27000	0	6	0	
1	4.75	9.54	43000	0	7	1	
2	7.25	9.85	6900	0	3	0	
3	2.85	4.15	5200	0	9	0	
4	4.60	6.87	42450	0	6	1	

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

```
X.head()
```

	Present_Price	Kms_Driven	Owner	no year	Fuel_Type_Diesel	Fuel_Type_Petrol	Selle
0	5.59	27000	0	6	0	1	
1	9.54	43000	0	7	1	0	
2	9.85	6900	0	3	0	1	
3	4.15	5200	0	9	0	1	
4	6.87	42450	0	6	1	0	

```
y.head()
```

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

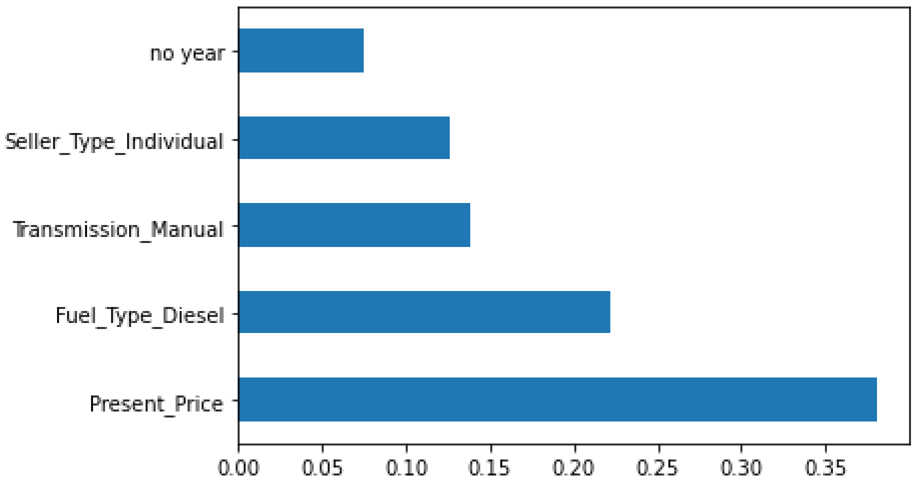
```
## Feature importance
from sklearn.ensemble import ExtraTreesRegressor
model=ExtraTreesRegressor()
model.fit(X,y)
```

```
ExtraTreesRegressor(bootstrap=False, ccp_alpha=0.0, criterion='mse',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    max_samples=None, min_impurity_decrease=0.0,
                    min_impurity_split=None, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=100, n_jobs=None, oob_score=False,
                    random_state=None, verbose=0, warm_start=False)
```

```
print(model.feature_importances_)
```

```
[0.38088183 0.04244871 0.00088483 0.07442966 0.22165906 0.015359
 0.12618957 0.13814733]
```

```
## plot graph of feature imp for better visualization
feat_importances=pd.Series(model.feature_importances_,index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```




```
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)
```

```
X_train.shape
```

```
(240, 8)
```

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

```
### hyperparameters
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]
```

```
# Randomized search CV
# Number of trees in random forest
```

```
# Mumber of features to consider at evry split
max_features=['auto', 'sqrt']

# Maximum numebr of levels in tree
max_depth= [int (x) for x in np.linspace(5,30,num=6)]

# max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2,5,10,15,100]

# Minimum number of samples required at each leaf node
min_samples_leaf = [1,2,5,10]
```

Double-click (or enter) to edit

```
from sklearn.model_selection import RandomizedSearchCV
```

```
# Create the random grid
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], 'm
```

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf= RandomForestRegressor()
```

```
rf_random = RandomizedSearchCV(estimator=rf,param_distributions= random_grid, scoring='neg
```

```
rf_random.fit(X_train,y_train)
```

```
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=s
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=s
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=

[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqr
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sq
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqr
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sq
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqr
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sq
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqr
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sq
```

```
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sq
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqr
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sq
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqr
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sq
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqr
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sq
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqr
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sq
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqr
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sq
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqr
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sq
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=aut
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=au
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=aut
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=au
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=aut
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=au
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=aut
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=au
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=aut
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=au
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:   48.3s finished
RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=RandomForestRegressor(bootstrap=True,
                                                   ccp_alpha=0.0,
                                                   criterion='mse',
                                                   max_depth=None,
                                                   max_features='auto',
                                                   max_leaf_nodes=None,
                                                   max_samples=None,
                                                   min_impurity_decrease=0.0,
                                                   min_impurity_split=None,
                                                   min_samples_leaf=1,
                                                   min_samples_split=2,
                                                   min_weight_fraction_leaf=0.0,
                                                   n_estimators=100,
                                                   n_jobs=None, oob_score=Fals...
                   iid='deprecated', n_iter=10, n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
```

```
predictions = rf_random.predict(X_test)
```

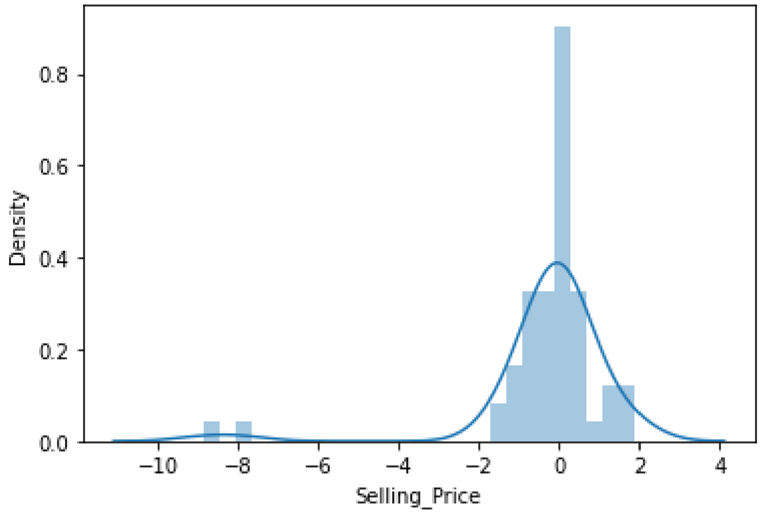
```
predictions
```

```
array([ 5.53079691,  5.49233737,  5.03513946,  0.61248012,  0.42482968,
        0.2818896 ,  1.25651054,  4.34359578,  1.13766351,  0.45514578,
        4.33709643,  0.44247738,  4.00096281,  7.03645182,  3.26980961,
        9.74807657,  9.94241982, 10.77664726,  0.7104641 , 13.13766071,
        2.7785959 ,  4.5792296 ,  4.85229356,  6.20849453,  1.10751991,
        5.90668373,  4.50404459, 12.87585308,  0.66562755,  3.17282896,
        7.2899837 , 10.99970035,  0.74839451, 19.56684694,  3.56501999,
```

```
3.99619842,  0.44147581,  7.62336126,  0.23230915,  1.32839241,
1.04725928,  5.44542631,  5.54283588,  7.62336126,  3.04406344,
2.47768522,  0.1998981 ,  2.83229675,  0.39872321,  7.29111758,
4.84068523,  6.86023713, 21.12200559,  0.45453046, 20.96348533,
3.04406344,  0.38204027,  2.68977242,  5.51051702, 10.60079365,
1.24664316])
```

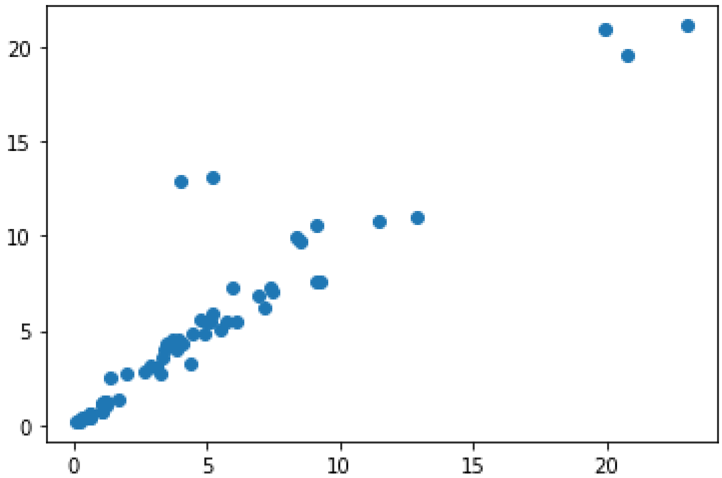
```
sns.distplot(y_test-predictions)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f773d910490>
```




```
plt.scatter(y_test,predictions)
```

```
<matplotlib.collections.PathCollection at 0x7f773d8b11d0>
```




```
import pickle
# open a fle, where you ant to store the data
file = open('random_forest_regression_model.pkl','wb')

# dump information to that file
pickle.dump(rf_random, file)
```

```
print(y_test)
```

```
230    6.15
224    5.11
234    5.50
139    0.60
156    0.48
       ...
178    0.35
42     1.95
251    5.00
260    9.15
112    1.15
Name: Selling_Price, Length: 61, dtype: float64
```

```
print(predictions)
```

```
[ 5.53079691  5.49233737  5.03513946  0.61248012  0.42482968  0.2818896
  1.25651054  4.34359578  1.13766351  0.45514578  4.33709643  0.44247738
  4.00096281  7.03645182  3.26980961  9.74807657  9.94241982 10.77664726
  0.7104641  13.13766071  2.7785959   4.5792296   4.85229356  6.20849453
  1.10751991  5.90668373  4.50404459 12.87585308  0.66562755  3.17282896
  7.2899837  10.99970035  0.74839451 19.56684694  3.56501999  3.99619842
  0.44147581  7.62336126  0.23230915  1.32839241  1.04725928  5.44542631
  5.54283588  7.62336126  3.04406344  2.47768522  0.1998981   2.83229675
  0.39872321  7.29111758  4.84068523  6.86023713 21.12200559  0.45453046
 20.96348533  3.04406344  0.38204027  2.68977242  5.51051702 10.60079365
  1.24664316]
```

```
print(y_test - predictions)
```

```
230    0.619203
224   -0.382337
234    0.464861
139   -0.012480
156    0.055170
         ...
178   -0.032040
42    -0.739772
251   -0.510517
260   -1.450794
112   -0.096643
Name: Selling_Price, Length: 61, dtype: float64
```

```
from sklearn.metrics import r2_score
r2_score(y_test, predictions)
```

```
0.8779809280325832
```

```
from sklearn import metrics
```

```
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)))
```