In [5]:

```
###############This dataset contains information about used cars listed on www.cardekho.com
This data can be used for a lot of purposes such as price prediction to exemplify the use of linear
regression in Machine Learning.
The columns in the given dataset are as follows:

name
year
selling_price
km_driven
fuel
seller_type
transmission
Owner

########## Table of contents ###########
#1. Importing the dataset from kagglset
#2. EDA of the data
#3. Checking missing and null values
#4. visualization of different variable with the dependent variable
#5. correlation matrix
#6. Feature engineering using extratree regressor
#7. Model building by using randomforest regressor and RandomizedSearchCV
#8. checking linearity of the model and R squared value
#9. Final prediction of the price



import pandas as pd
```

In [6]:

```
df= pd.read_csv('cardata.csv')
```

In [8]:

```
df.head()
```

Out[8]:

	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 [10]:

```
df.shape      ##301 rows, 9 features/columns
```

Out[10]:

```
(301, 9)
```

In [11]:

```
##### how many unique values are there in the character variable ######
print(df['Seller_Type'].unique())
print(df['Transmission'].unique())
print(df['Owner'].unique())
```

```
['Dealer' 'Individual']
['Manual' 'Automatic']
```

```
[0 1 3]
```

In [12]:

```
##### checking missing values ######

df.isnull().sum()
```

Out[12]:

```
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 [13]:

```
df.describe()
```

Out[13]:

	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 [14]:

```
df.columns
```

Out[14]:

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

In [17]:

```
#### removing car name from the data set  ######

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

In [18]:

```
final_dataset.head()
```

Out[18]:

	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

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
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 [19]:

```
####### creating new column "current year" to get how old the car is ########

final_dataset['current_year']= 2020
```

In [20]:

```
final_dataset.head()
```

Out[20]:

	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	2020
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2020
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2020
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2020
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2020

In [21]:

```
#### substracting past years from the current year ######

final_dataset['no_year']= final_dataset['current_year']-final_dataset['Year']
```

In [22]:

```
final_dataset.head()
```

Out[22]:

	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	2020	6
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2020	7
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2020	3
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2020	9
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2020	6

In [23]:

```
###### removing 'year' column

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

In [ ]:

```

```

In [24]:

```
final_dataset.head()
```

Out[24]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	current_year	no_year

	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	2020	6
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	2020	7
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	2020	3
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	2020	9
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	2020	6

In [25]:

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

In [26]:

```
final_dataset.head()
```

Out[26]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	no_year
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	6
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	7
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	3
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	9
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	6

In [27]:

```
###### one-hot-encoding to prevent from dummy variable #####

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

In [28]:

```
final_dataset.head()
```

Out[28]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmissio
0	3.35	5.59	27000	0	6	0	1	0	
1	4.75	9.54	43000	0	7	1	0	0	
2	7.25	9.85	6900	0	3	0	1	0	
3	2.85	4.15	5200	0	9	0	1	0	
4	4.60	6.87	42450	0	6	1	0	0	

In [29]:

```
final_dataset.corr()
```

Out[29]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_
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	

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_
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 [30]:

```
import seaborn as sns
```

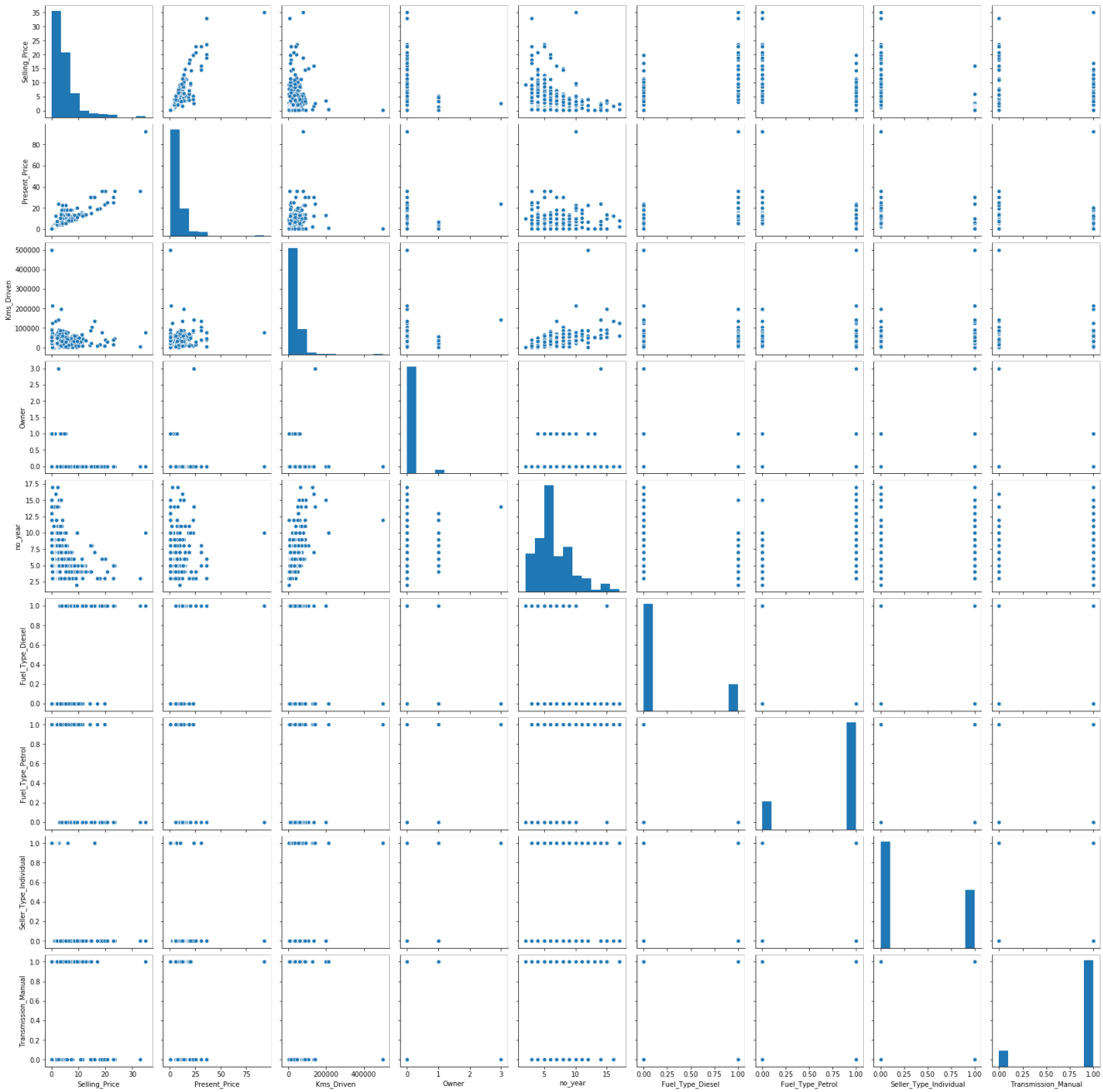
In [31]:

```
############ visualization ############

sns.pairplot(final_dataset)
```

Out[31]:

```
<seaborn.axisgrid.PairGrid at 0x1e5886f9508>
```



In [33]:

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

In [36]:

```
###### heatmap plot to analyse the strong correlated variable which is selling price and present p
rice, and Fuel type, seller type these are negatively correlated ##########

corrmat = final_dataset.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(10,10))
g= sns.heatmap(final_dataset[top_corr_features].corr(),annot=True,cmap='RdYlGn')
```

In [38]:

```
########### Independent and dependent features ################

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

In [39]:

```
x.head()
```

Out[39]:

	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	5.59	27000	0	6	0	1	0	1
1	9.54	43000	0	7	1	0	0	1
2	9.85	6900	0	3	0	1	0	1
3	4.15	5200	0	9	0	1	0	1

	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
4	6.87	42450	0	6	1	0	0	1

In [40]:

```
y.head()
```

Out[40]:

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

In [42]:

```
########## feature importance #########

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

Out[42]:

```
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)
```

In [43]:

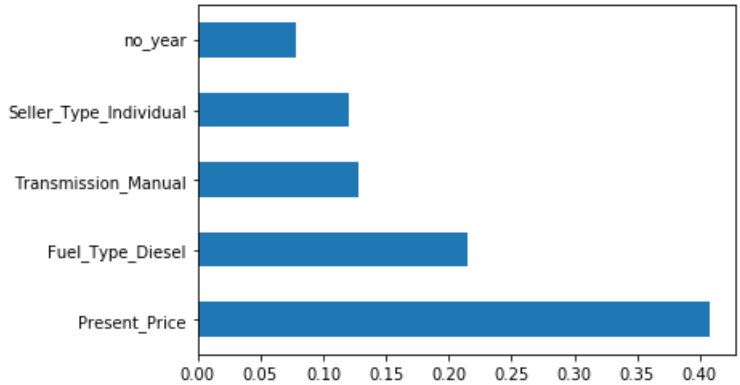
```
print(model.feature_importances_)  #### present price is most important feature followed by fuel t
ype diesel,transmission manual, seller type individual
```

```
[0.40753887 0.03843814 0.00052068 0.07867486 0.2143339  0.01107983
 0.12074051 0.12867322]
```

In [45]:

```
####### plot graph for feature importance to get better visualization ##########

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



In [46]:

```
########## train and test data#############

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 [47]:

```
x_train
```

Out[47]:

	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
189	0.57	55000	0	15	0	1	1	1
198	0.57	35000	1	9	0	1	1	1
192	0.75	49000	1	13	0	1	1	1
207	5.70	34797	0	5	0	1	0	0
223	9.40	61381	0	5	1	0	0	1
...	...	...	...	...	...	...	...	...
206	7.13	12479	0	3	0	1	0	1
59	35.96	41000	0	6	1	0	0	0
26	5.87	55138	0	7	0	1	0	1
210	4.60	35775	0	8	0	1	0	1
74	8.93	83000	0	6	1	0	0	1

240 rows × 8 columns

In [48]:

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

In [49]:

```
rf_random
```

Out[49]:

```
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=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [51]:

```
######### no. of trees in random forest #########

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 [52]:

```
######## no. of features to consider in every split #########

max_features = ['auto','sqrt']

###### max no. of levels in tree############

max_depth = [int(x) for x in np.linspace(5,30, num = 6)]

##### min. no of samples required to split a node ########
```

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

###### min no. of samples required at each leaf node #######

min_samples_leaf = [1,2,5,10]
```

In [55]:

```
from sklearn.model_selection import RandomizedSearchCV

######## create 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], '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 [57]:

```
rf = RandomForestRegressor()
rf_random = RandomizedSearchCV(estimator = rf,param_distributions=
random_grid,scoring='neg_mean_squared_error',n_iter = 10, cv =5,verbose=2,random_state=42,n_jobs=1)
```

In [58]:

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

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10,
total=   1.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.7s remaining:    0.0s

[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10,
total=   1.7s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10,
total=   2.1s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10,
total=   1.7s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10,
total=   1.7s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt,
max_depth=15, total=   2.2s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt,
max_depth=15, total=   2.2s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt,
max_depth=15, total=   2.1s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt,
max_depth=15, total=   2.1s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt,
max depth=15, total=   2.2s
```

```
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto,
max_depth=15, total=   0.6s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto,
max_depth=15, total=   0.9s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto,
max_depth=15, total=   0.9s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto,
max_depth=15, total=   0.8s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto,
max_depth=15, total=   1.2s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=   1.2s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=   1.4s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=   1.3s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=   1.4s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=   1.2s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=   2.1s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=   2.1s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=   2.3s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=   2.7s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=   1.9s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=   3.0s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=   2.9s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=   3.0s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=   3.6s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=   2.9s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
max_depth=5, total=   3.0s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
max_depth=5, total=   2.9s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
max_depth=5, total=   2.9s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
max_depth=5, total=   3.3s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
max_depth=5, total=   3.2s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15,
```

```
total=   0.8s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15,
total=   0.8s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15,
total=   0.8s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15,
total=   0.8s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15,
total=   0.8s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5,
total=   1.8s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5,
total=   1.8s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5,
total=   1.8s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5,
total=   1.8s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5,
total=   2.5s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20,
total=   2.0s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20,
total=   1.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20,
total=   1.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20,
total=   1.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20,
total=   2.0s
```

```
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:  1.6min finished
```

Out[58]:

```
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,
                                                         900, 1000, 1100,
                                                         1200]},
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring='neg_mean_squared_error',
                   verbose=2)
```

In [59]:

```
prediction = rf_random.predict(x_test)
```

In [60]:

```
prediction
```

Out[60]:

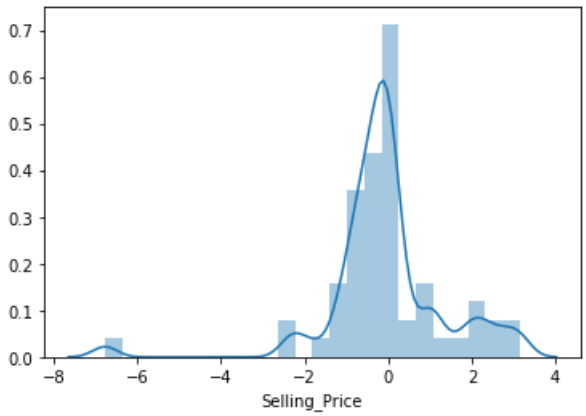
```
array([ 9.34945,  4.34325,  0.48581,  0.72319,  1.09331,  8.17137,
        3.05893,  9.24699,  3.16137,  9.17114,  0.49069,  4.96569,
        6.26666,  6.44772,  0.58256,  4.5273 ,  7.29306,  5.19171,
        5.82555,  0.94339, 10.64395,  6.0283 ,  0.5291 , 10.2691 ,
        5.21871,  2.47569,  4.0275 ,  3.38495,  2.7392 ,  0.44871,
        4.97174,  4.008  ,  0.50244,  0.71016,  0.57762,  5.7402 ,
        1.00214, 21.30089,  1.24255,  6.79779,  6.08851,  6.28257,
        0.61456,  9.23316,  3.64856,  5.35091,  1.03803,  2.4306 ,
        0.58461,  2.8441 ,  0.56369,  0.97113,  4.9638 , 20.43569,
        0.34161,  1.08627, 10.39429, 20.77549,  3.744  ,  6.01297,
        5.35654])
```

In [62]:

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

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e58f90a5c8>
```

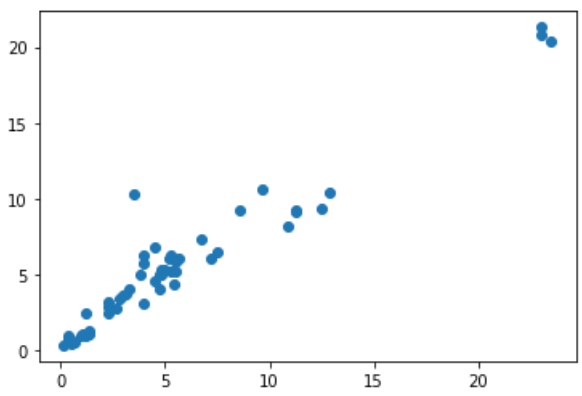


In [63]:

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

Out[63]:

```
<matplotlib.collections.PathCollection at 0x1e58f7e45c8>
```



In [64]:

```
import pickle

file = open('random_forest_regression_model.pkl','wb')

####### dump info to that file #########

pickle.dump(rf_random,file)
```

In [ ]:

```

```