Card Price Prediction

In [1]:

```
import pandas as pd
```

In [2]:

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

In [3]:

```
df.head()
```

Out[3]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Trans
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	

In [4]:

```
df.shape
```

Out[4]:

```
(301, 9)
```

Finding Unique values in dataset

In [5]:

```
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]
```

Checking null Values

In [6]:

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

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]:

```
df.describe()
```

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]:

```
df.columns
```

Out[8]:

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

In [9]:

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

In [10]:

```
final_dataset.head()
```

Out[10]:

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

In [11]:

```
final_dataset['Current_year']=2020
```

In [12]:

```
final_dataset.head()
```

Out[12]:

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

In [13]:

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

In [14]:

```
final_dataset.head()
```

Out[14]:

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

Droping year and current year column

In [15]:

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

In [16]:

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

In [17]:

```
final_dataset.head()
```

Out[17]:

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

Droping First column and convert into 0 & 1

In [18]:

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

In [19]:

```
final_dataset.head()
```

Out[19]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_F
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	

In [20]:

```
final_dataset.corr()
```

Out[20]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_1
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	

In [21]:

```
pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\35389\anaconda3\envs\ca
rprediction\lib\site-packages (0.10.1)
Requirement already satisfied: pandas>=0.22.0 in c:\users\35389\anaconda3
\envs\carprediction\lib\site-packages (from seaborn) (1.1.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\35389\anaconda3\e
nvs\carprediction\lib\site-packages (from seaborn) (1.19.1)
Requirement already satisfied: scipy>=1.0.1 in c:\users\35389\anaconda3\en
vs\carprediction\lib\site-packages (from seaborn) (1.5.2)
Requirement already satisfied: matplotlib>=2.1.2 in c:\users\35389\anacond
a3\envs\carprediction\lib\site-packages (from seaborn) (3.3.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\35389\anaconda3\en
vs\carprediction\lib\site-packages (from pandas>=0.22.0->seaborn) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\35389\an
aconda3\envs\carprediction\lib\site-packages (from pandas>=0.22.0->seabor
n) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\users\35389\anaconda3\en
vs\carprediction\lib\site-packages (from matplotlib>=2.1.2->seaborn) (0.1
0.0)
Requirement already satisfied: certifi>=2020.06.20 in c:\users\35389\anaco
nda3\envs\carprediction\lib\site-packages (from matplotlib>=2.1.2->seabor
n) (2020.6.20)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in
c:\users\35389\anaconda3\envs\carprediction\lib\site-packages (from matplo
tlib>=2.1.2->seaborn) (2.4.7)
Requirement already satisfied: pillow>=6.2.0 in c:\users\35389\anaconda3\e
nvs\carprediction\lib\site-packages (from matplotlib>=2.1.2->seaborn) (7.
2.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\35389\anacond
a3\envs\carprediction\lib\site-packages (from matplotlib>=2.1.2->seaborn)
(1.2.0)
Requirement already satisfied: six>=1.5 in c:\users\35389\anaconda3\envs\c
arprediction\lib\site-packages (from python-dateutil>=2.7.3->pandas>=0.22.
0->seaborn) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [22]:

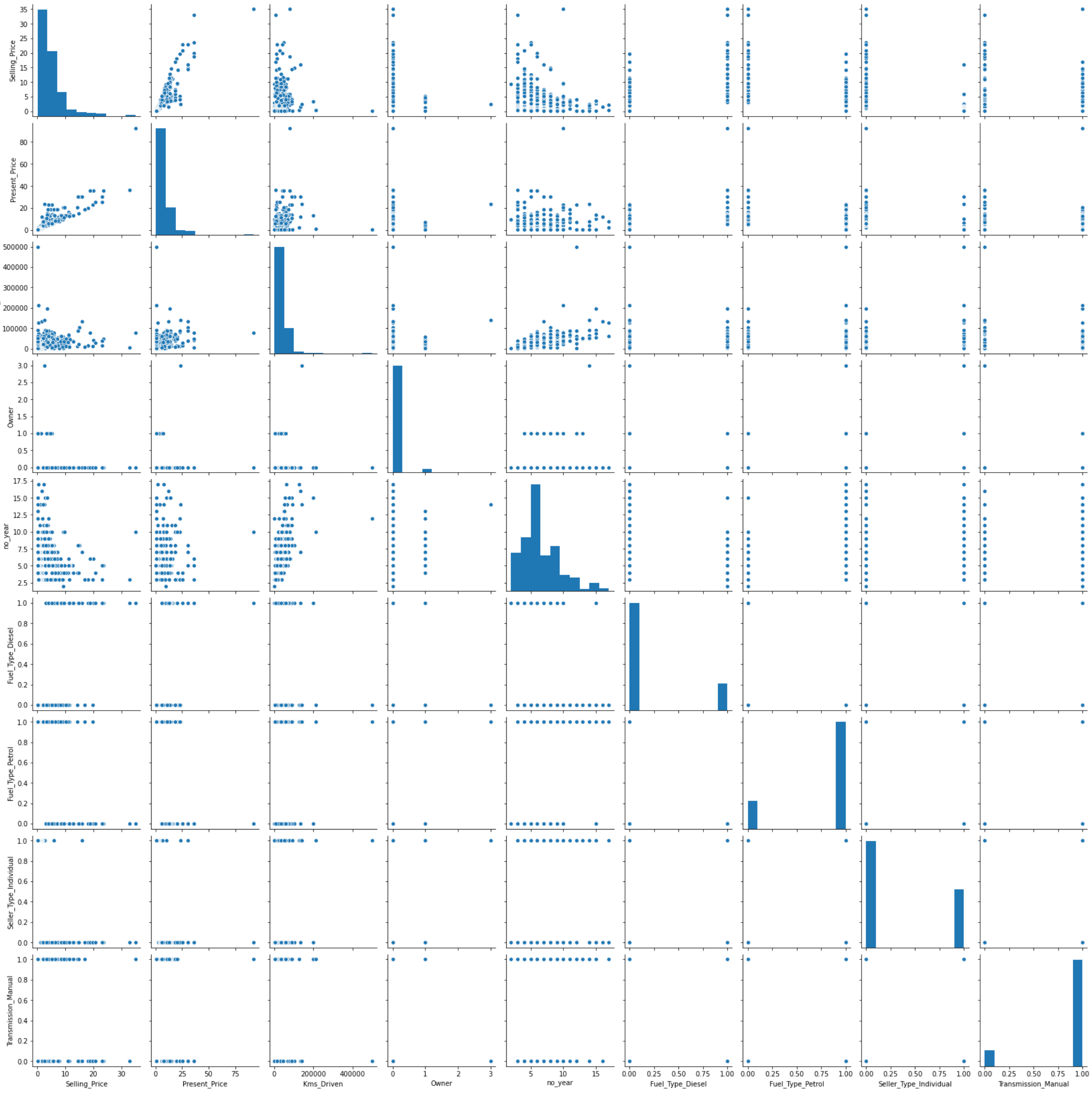
```
import seaborn as sns
```

In [23]:

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

Out[23]:

<seaborn.axisgrid.PairGrid at 0x239dccae448>



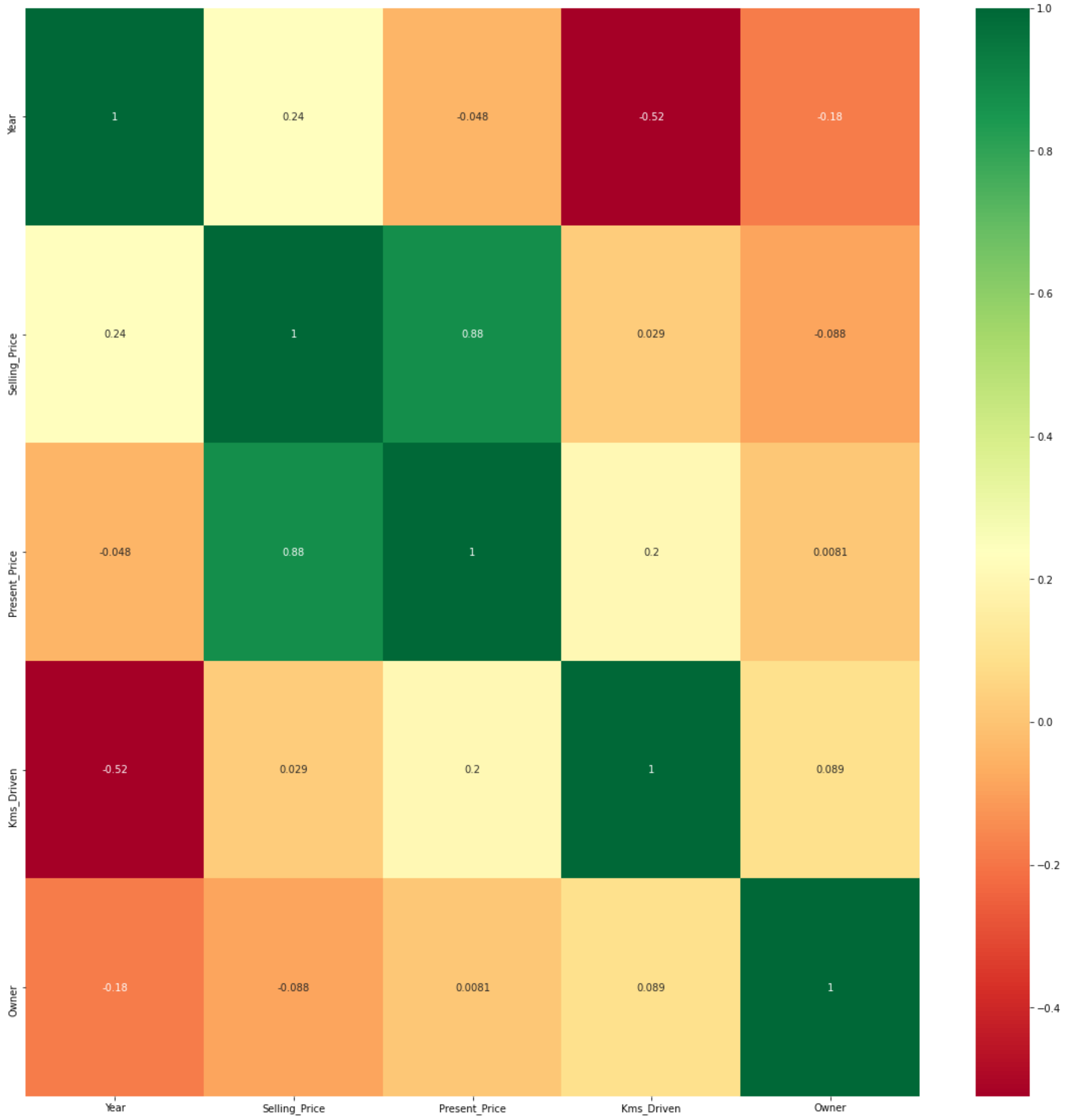
In [24]:

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

Get Correlations of each features in dataset and Plot Heat Map

In [25]:

```
import seaborn as sns
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



In [26]:

```
final_dataset.head()
```

Out[26]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_F
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	

Independent and Dependent Features

In [27]:

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

In [28]:

```
X.head()
```

Out[28]:

	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_T
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	

In [29]:

```
y.head()
```

Out[29]:

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

Features Importance

In [30]:

```
pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\users\35389\anaconda3\envs\ca
rprediction\lib\site-packages (0.0)
Requirement already satisfied: scikit-learn in c:\users\35389\anaconda3\en
vs\carprediction\lib\site-packages (from sklearn) (0.23.2)
Requirement already satisfied: numpy>=1.13.3 in c:\users\35389\anaconda3\e
nvs\carprediction\lib\site-packages (from scikit-learn->sklearn) (1.19.1)
Requirement already satisfied: joblib>=0.11 in c:\users\35389\anaconda3\en
vs\carprediction\lib\site-packages (from scikit-learn->sklearn) (0.16.0)
Requirement already satisfied: scipy>=0.19.1 in c:\users\35389\anaconda3\e
nvs\carprediction\lib\site-packages (from scikit-learn->sklearn) (1.5.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\35389\anac
onda3\envs\carprediction\lib\site-packages (from scikit-learn->sklearn)
(2.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [31]:

```
from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(X,y)
```

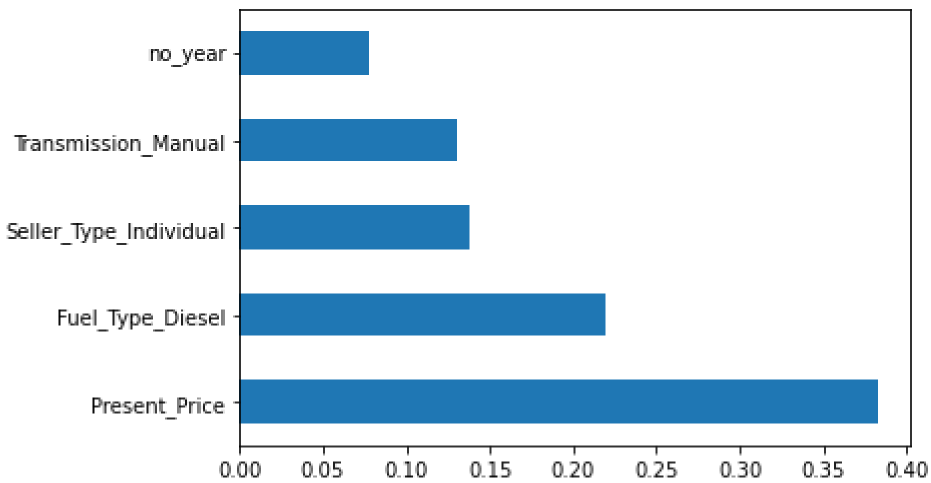
Out[31]:

```
ExtraTreesRegressor()
```

Plot graph for better visualization and understanding

In [32]:

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



Split data into test and train

In [37]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.3)
```

In [38]:

```
X_train.shape
```

Out[38]:

```
(210, 8)
```

In [39]:

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

Hyperparameters

In [41]:

```
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

In [42]:

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

In [43]:

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number 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]
```

Create the Random Grid

In [44]:

```
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]}
```

Initialize random forest

In [45]:

```
rf = RandomForestRegressor()
```

In [52]:

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

In [53]:

```
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_featur
es=sqrt, max_depth=10

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

[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_featu
res=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_featur
es=sqrt, max_depth=10

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.7s remaining:
0.0s
```

```
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_featu
res=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_featur
es=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_featu
res=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_featur
es=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_featu
res=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_featur
es=sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_featu
res=sqrt, max_depth=10, total=   0.8s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_fea
tures=sqrt, max_depth=15, total=   1.0s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_fea
tures=sqrt, max_depth=15, total=   1.0s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_fea
tures=sqrt, max_depth=15, total=   1.1s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_fea
tures=sqrt, max_depth=15, total=   1.0s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_fea
tures=sqrt, max_depth=15, total=   1.0s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feat
ures=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_fea
tures=auto, max_depth=15, total=   0.3s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feat
ures=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_fea
tures=auto, max_depth=15, total=   0.3s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feat
ures=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_fea
tures=auto, max_depth=15, total=   0.3s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feat
ures=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_fea
tures=auto, max_depth=15, total=   0.3s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feat
ures=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_fea
tures=auto, max_depth=15, total=   0.3s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featur
es=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featu
res=auto, max_depth=15, total=   0.4s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featur
es=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featu
```

```
res=auto, max_depth=15, total=   0.4s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featur
es=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featu
res=auto, max_depth=15, total=   0.4s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featur
es=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featu
res=auto, max_depth=15, total=   0.4s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featur
es=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_featu
res=auto, max_depth=15, total=   0.4s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_featu
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feat
ures=auto, max_depth=20, total=   0.6s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_featu
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feat
ures=auto, max_depth=20, total=   0.6s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_featu
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feat
ures=auto, max_depth=20, total=   0.6s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_featu
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feat
ures=auto, max_depth=20, total=   0.6s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_featu
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feat
ures=auto, max_depth=20, total=   0.6s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_featu
res=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=25, total=   1.0s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_featu
res=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=25, total=   1.0s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_featu
res=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=25, total=   1.0s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_featu
res=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=25, total=   1.0s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_featu
res=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=25, total=   1.0s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fea
tures=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fe
atures=sqrt, max_depth=5, total=   1.0s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fea
tures=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fe
atures=sqrt, max_depth=5, total=   1.0s
```

```
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fea
tures=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fe
atures=sqrt, max_depth=5, total=   1.0s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fea
tures=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fe
atures=sqrt, max_depth=5, total=   1.0s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fea
tures=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_fe
atures=sqrt, max_depth=5, total=   1.0s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_featu
res=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=15, total=   0.3s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_featu
res=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=15, total=   0.3s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_featu
res=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=15, total=   0.3s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_featu
res=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=15, total=   0.3s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_featu
res=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feat
ures=sqrt, max_depth=15, total=   0.3s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_featu
res=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=5, total=   0.6s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_featu
res=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=5, total=   0.6s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_featu
res=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=5, total=   0.6s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_featu
res=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=5, total=   0.6s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_featu
res=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feat
ures=sqrt, max_depth=5, total=   0.6s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_featu
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feat
ures=auto, max_depth=20, total=   0.7s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_featu
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feat
ures=auto, max_depth=20, total=   0.7s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_featu
```

```
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feat
ures=auto, max_depth=20, total=   0.7s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_featu
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feat
ures=auto, max_depth=20, total=   0.7s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_featu
res=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feat
ures=auto, max_depth=20, total=   0.7s


[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:   33.3s finished
```

Out[53]:

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 3
0],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 1
5,
                                                              100],
                                        'n_estimators': [100, 200, 300, 40
0,
                                                         500, 600, 700, 80
0,
                                                         900, 1000, 1100,
                                                         1200]},
                   random_state=42, scoring='neg_mean_squared_error',
                   verbose=2)
```

In [54]:

```
rf_random.best_params_
```

Out[54]:

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

In [55]:

```
rf_random.best_score_
```

Out[55]:

```
-2.4860209966038203
```

In [56]:

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

In [57]:

```
predictions
```

Out[57]:

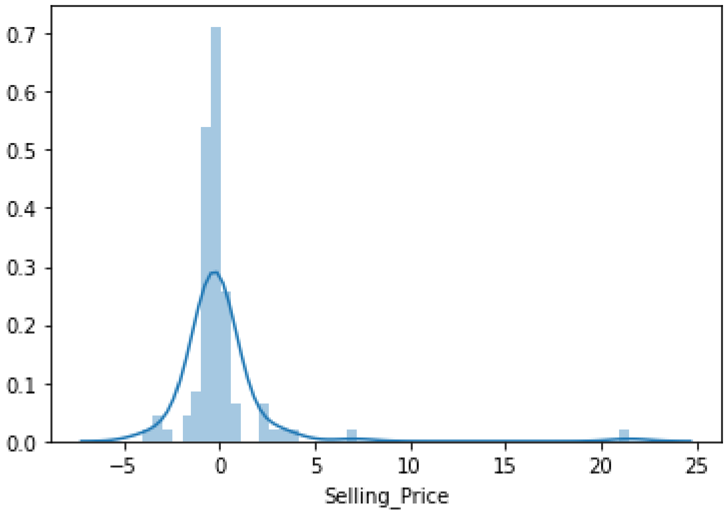
```
array([ 1.22993,  7.00245,  8.93604,  7.46952,  5.2076 ,  9.1255 ,
        4.83272,  5.83854,  0.44541,  4.56738,  3.53709,  7.69006,
        1.19462, 13.52461,  6.27158,  4.57607,  5.6314 ,  0.8523 ,
        5.37358,  4.8391 , 11.02958,  4.15165,  7.91401,  4.64433,
        0.79515,  0.55562, 19.9843 ,  4.6184 ,  0.38957,  5.51086,
        0.9651 ,  5.1019 ,  1.05717,  5.79735,  5.89785,  6.1885 ,
       15.50085,  8.01658,  0.57899,  3.55096,  6.01679,  0.38142,
        2.77745,  1.18624,  0.63726,  6.7049 ,  0.24697, 10.97075,
        3.72258,  2.8675 ,  7.94933,  2.40639,  6.00119,  8.3474 ,
        0.99992,  8.27531,  5.9794 ,  8.56729,  4.5063 ,  5.28435,
        8.1354 ,  0.4816 ,  7.0905 ,  4.5799 ,  0.52601,  5.58627,
        0.93264,  0.61475,  0.36099,  0.63085,  2.81948,  0.74184,
        1.3903 ,  0.76561,  7.24253,  4.02815,  4.79695,  1.03509,
       20.5012 ,  5.65915,  0.68423,  0.44806,  0.452  ,  8.52665,
        0.55644,  4.10591,  1.10929,  3.74854,  0.82503, 20.5012 ,
        9.91197])
```

In [61]:

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

Out[61]:

<AxesSubplot:xlabel='Selling_Price'>

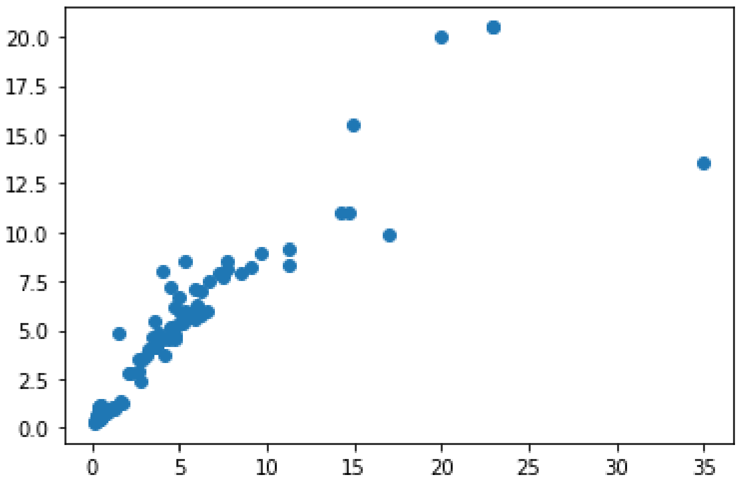


In [62]:

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

Out[62]:

```
<matplotlib.collections.PathCollection at 0x239ec6d84c8>
```



import file through pickle

In [63]:

```
import pickle
file = open('random_forest_regression_model.pkl', 'wb')
pickle.dump(rf_random, file)
```