```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
```

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

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Tra
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("Seller Type:",df['Seller_Type'].unique())
print("Transmission:",df['Transmission'].unique())
print("Owner",df['Owner'].unique())
print("Fuel Type",df['Fuel_Type'].unique())
```

```
Seller Type: ['Dealer' 'Individual']
Transmission: ['Manual' 'Automatic']
Owner [0 1 3]
Fuel Type ['Petrol' 'Diesel' 'CNG']
```

```
df.isnull()
```

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Ow
0	False	False	False	False	False	False	False	False	F
1	False	False	False	False	False	False	False	False	F
2	False	False	False	False	False	False	False	False	F
3	False	False	False	False	False	False	False	False	F
4	False	False	False	False	False	False	False	False	F
...	...	...	...	...	...	...	...	...	
296	False	False	False	False	False	False	False	False	F
297	False	False	False	False	False	False	False	False	F
298	False	False	False	False	False	False	False	False	F
299	False	False	False	False	False	False	False	False	F
300	False	False	False	False	False	False	False	False	F

301 rows × 9 columns

```
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
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

```
df.dtypes
```

```
Car_Name          object
Year               int64
Selling_Price    float64
Present_Price    float64
Kms_Driven         int64
```

```
Fuel_Type         object
Seller_Type       object
Transmission      object
Owner              int64
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_Type    301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

```
df.columns
```

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

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

```
final_df.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission
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

```
final_df['current_year'] = 2020
```

```
final_df.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission
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

```
final_df['Age of car (Years)'] = final_df['current_year']-final_df['Year']
```

```
final_df.head()
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission
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

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

```
final_df.head()
```

	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

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

```
final_df.head()
```

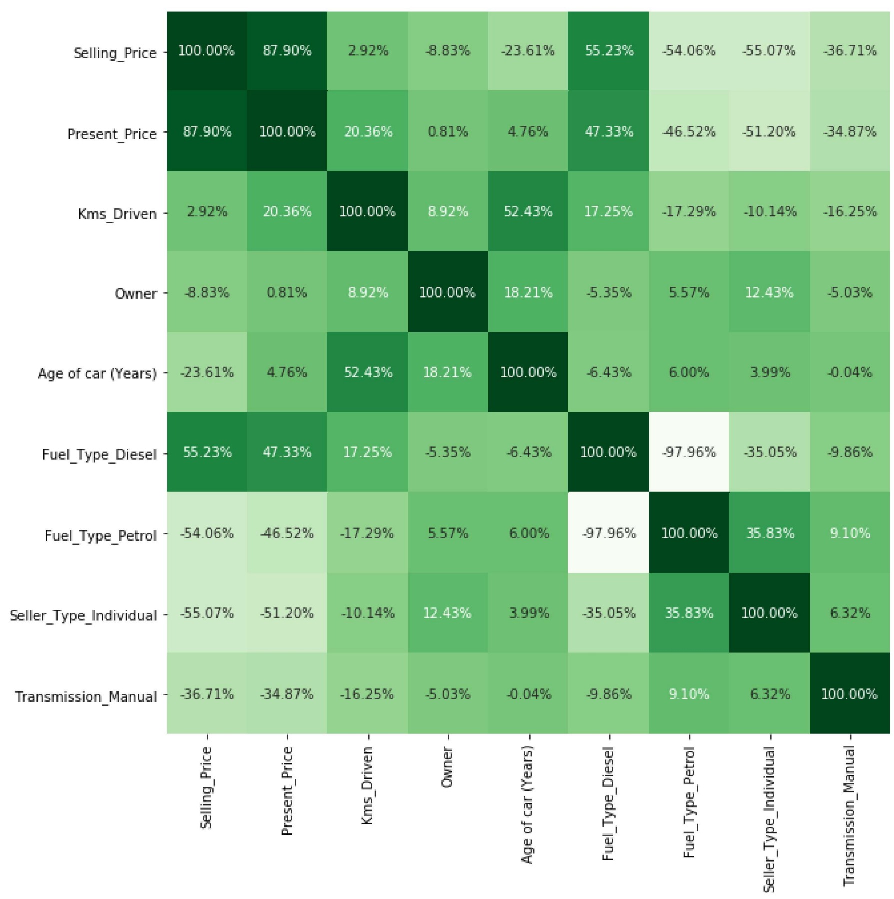
	Selling_Price	Present_Price	Kms_Driven	Owner	Age of car (Years)	Fuel_Type_Diesel	Fuel_Type
0	3.35	5.59	27000	0	6	0	
1	4.75	9.54	43000	0	7	1	
2	7.25	9.85	6900	0	3	0	
3	2.85	4.15	5200	0	9	0	

```
final_df.corr()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age of car (Years)	Fuel_
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	
Age of car (Years)	-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	

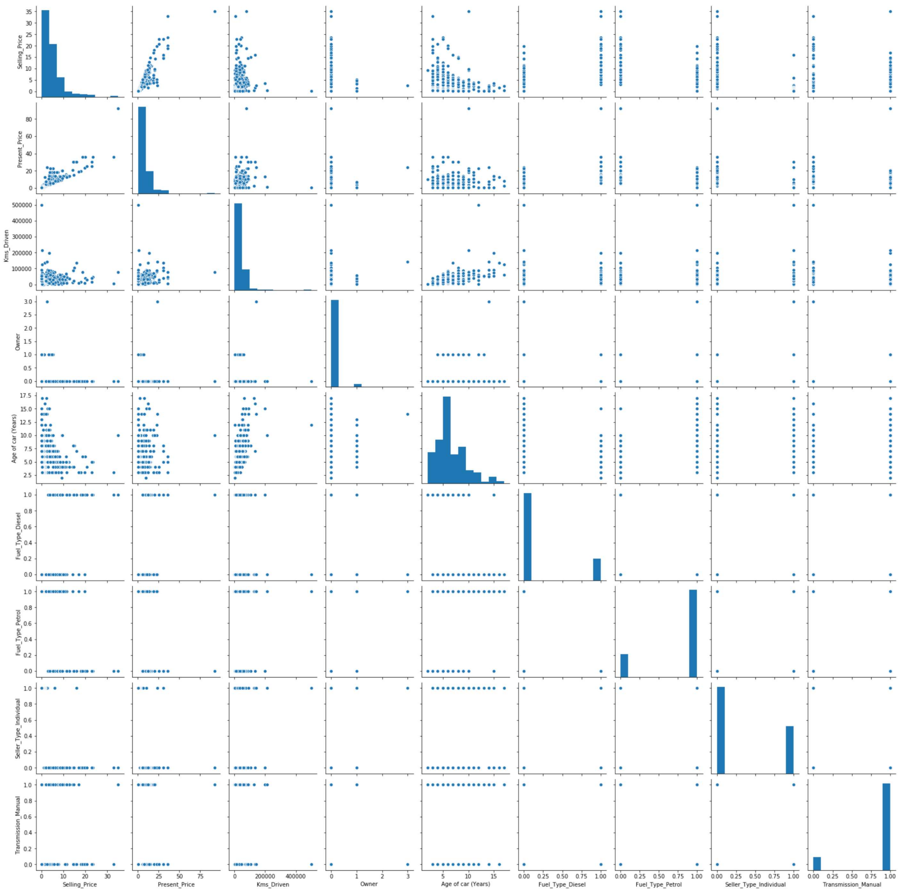
```
# Plotting the heatmap of correlation between features
plt.figure(figsize=(10,10))
sns.heatmap(final_df.corr(), cbar=False, square= True, fmt='.2%', annot=True, cmap='Greens')
```

<matplotlib.axes._subplots.AxesSubplot at 0x2af38845188>




```
sns.pairplot(final_df)
```

<seaborn.axisgrid.PairGrid at 0x2af35399d08>



```
import seaborn as sns
#get correlations of each features in dataset
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(13,13))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

```
final_df.head()
```

	Selling_Price	Present_Price	Kms_Driven	Owner	Age of car (Years)	Fuel_Type_Diesel	Fuel_Type
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	

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

```
X.head()
```

	Present_Price	Kms_Driven	Owner	Age of car (Years)	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller
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	

```
y.head()
```

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

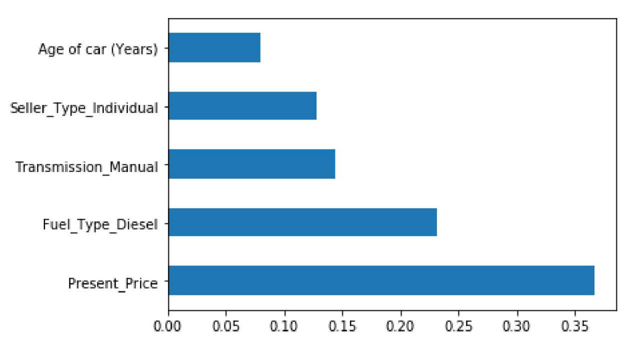
```
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
model.fit(X,y)
```

```
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_)
```

```
[3.66605474e-01 4.00106104e-02 3.23319549e-04 7.91610450e-02
 2.31734695e-01 1.09009115e-02 1.27572369e-01 1.43691576e-01]
```

```
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,random_state=10)
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
regressor=RandomForestRegressor()
```

```
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]
```

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

```
 #Randomized Search CV

# 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
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,
```

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

```
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 10, cv = 5, verbo
```

```
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=   2.2s
[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:    2.1s remaining:    0.0s
[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=   2.2s
[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=   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=   2.7s
[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=   2.7s
[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=   2.7s
[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=   2.7s
[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=   2.7s
[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=   0.7s
[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=   0.7s
[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=   0.7s
[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=   0.7s
[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=   0.7s
[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=   1.0s
[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=   0.9s
[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=   1.1s
[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=   0.9s
[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=   1.0s
[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=   1.6s
[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=   1.7s
[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=   1.5s
[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=   1.5s
[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=   1.8s
[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=   2.8s
[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=   2.7s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
```

```
rf_random.best_params_
```

```
{'n_estimators': 400,
 'min_samples_split': 5,
 'min_samples_leaf': 5,
 'max_features': 'auto',
 'max_depth': 15}
```

```
rf_random.best_score_
```

```
-3.952283233057366
```

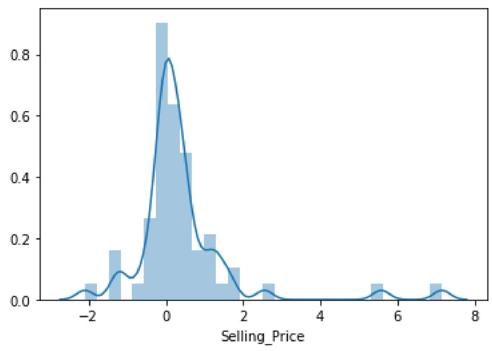
```
predictions=rf_random.predict(X_test)
```

```
predictions
```

```
array([10.41970835,  0.63135756,  2.63228372,  0.68797209,  5.25586668,
        5.46082911, 20.46456956,  0.23189264,  0.58361511,  4.34988626,
        0.57282266,  7.80460086,  4.63689516,  0.45156965,  4.70491165,
        3.83845151,  4.6782604 ,  0.21027498,  2.83074638,  7.40286836,
        0.68797209,  3.77844011,  6.922062  , 20.64240551,  0.28149955,
        0.42968627,  0.35001432,  0.30023241,  0.68109855,  0.21051311,
        2.57862963, 12.57898817,  4.86149551, 10.66757204,  0.55600258,
        5.46906598,  2.8310864 ,  3.43473912,  1.11531245,  4.45061754,
        4.68831089,  0.48184789,  5.49809302,  0.26716738, 11.50701834,
        3.32405635,  0.23711157,  4.58374421,  6.78795037,  2.53645001,
        0.40355573,  5.91114253,  0.52117427, 10.18657   ,  4.53777509,
        2.94457698, 10.95173023,  1.09385671,  8.0382979 ,  9.43725294,
        2.81807447])
```

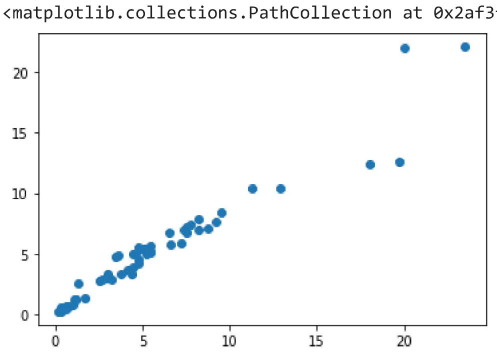
```
sns.distplot(y_test-predictions)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2af3fa7fe88>
```




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

```
<matplotlib.collections.PathCollection at 0x2af3fa6e488>
```




```
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)))
```

```
MAE: 0.7484803289084702
MSE: 3.5856054889060123
RMSE: 1.893569509921939
```

```
import pickle
# open a file, 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)
```

So Here, we concluded our project regarding the car price prediction using machine learning