In [46]:

```
import numpy as np
import pandas as pd
```

In [47]:

```
df=pd.read_csv('G:/anurag1241/datascience/projects/ML/car price prediction/car data.csv')
```

In [48]:

```
df.head()
```

Out[48]:

	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 [49]:

```
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 [50]:

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

Out[50]:

```
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 [51]:

```
df.describe()
```

Out[51]:

	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

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
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 [52]:

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

In [53]:

```
final_dataset.head()
```

Out[53]:

	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 [54]:

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

In [55]:

```
final_dataset.head()
```

Out[55]:

	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 [56]:

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

In [57]:

```
final_dataset.head()
```

Out[57]:

	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

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current Year	no_year
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2020	6

In [58]:

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

In [59]:

```
final_dataset.head()
```

Out[59]:

	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 [60]:

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

In [61]:

```
final_dataset.head()
```

Out[61]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Current Year	no_year	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Tra
0	3.35	5.59	27000	0	2020	6	0	1	0	
1	4.75	9.54	43000	0	2020	7	1	0	0	
2	7.25	9.85	6900	0	2020	3	0	1	0	
3	2.85	4.15	5200	0	2020	9	0	1	0	
4	4.60	6.87	42450	0	2020	6	1	0	0	

In [62]:

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

In [63]:

```
final_dataset.head()
```

Out[63]:

	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 [64]:

```
final_dataset.corr()
```

Out[64]:

	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	
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 [65]:

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

In [66]:

```
X['Owner'].unique()
```

Out[66]:

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

In [67]:

```
X.head()
```

Out[67]:

	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
4	6.87	42450	0	6	1	0	0	1

In [68]:

```
y.head()
```

Out[68]:

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

Feature Importance

In [69]:

```
from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
```

```
model = ExtraTreesRegressor()
model.fit(X,y)
```

Out[69]:

```
ExtraTreesRegressor()
```

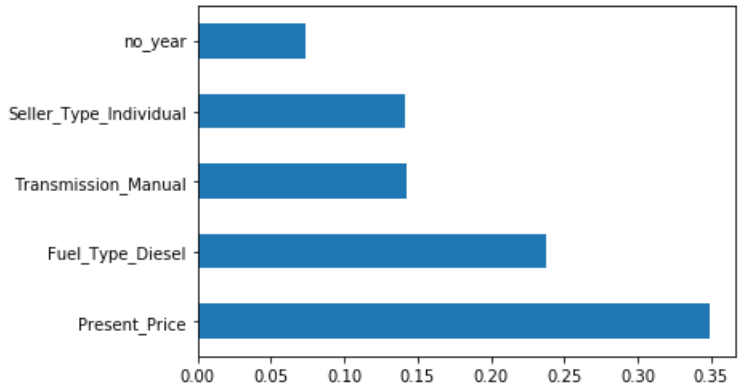
In [70]:

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

```
[0.34898685 0.04338549 0.00138598 0.07328472 0.23777695 0.01131743
 0.14113003 0.14273254]
```

In [71]:

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



In [72]:

```
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, random_state=0)
```

In [73]:

```
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]
```

HyperParameter Tuning

In [74]:

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

In [75]:

```
# 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
```

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

In [76]:

```
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 [77]:

```
rf = RandomForestRegressor()
```

In [78]:

```
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 [79]:

```
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=   2.6s
[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=   2.7s
[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=   2.5s
[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=   2.3s
[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=   2.8s
[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=   2.8s
[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=   2.8s
[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,
```

```
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto,
max_depth=15, total=   0.5s
[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=   0.6s
[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=   0.7s
[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=   0.8s
[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=   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=   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=   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=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=   2.0s
[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=   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=   1.9s
[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=   2.0s
[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=   1.9s
[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=   2.7s
[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=   2.4s
[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=   2.8s
[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=   2.7s
[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=   2.8s
[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=   2.8s
[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=   2.8s
[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=   2.8s
[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
[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=   0.6s
[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=   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=   1.6s
[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=   1.8s
[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=   1.7s
[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[79]:

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), 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]},
                   random_state=42, scoring='neg_mean_squared_error',
                   verbose=2)
```

In [80]:

```
rf_random.best_params_
```

Out[80]:

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

In [81]:

```
rf_random.best_score_
```

```
rf_random.best_score_
```

Out[81]:

```
-4.008836699337154
```

In [82]:

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

In [83]:

```
print(predictions)
```

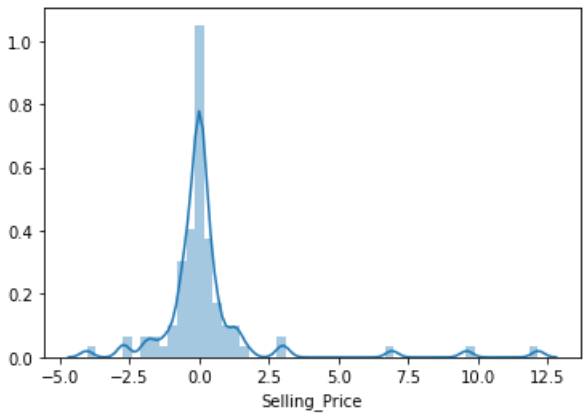
```
[ 7.10615  0.54284  5.08788  9.29444 16.56515  5.07093  3.4199   0.77717
  4.6338   4.46305  3.09463  0.88201  5.01286  8.01351  8.01015 10.15266
  7.36305  4.1137   0.50498  1.50135  4.01227  4.59304  5.59737  9.49179
  0.24245  0.77086  0.49142  0.63518  0.48491  5.04923  5.08877  5.83075
  0.52408  8.89206  3.55175  1.19288  5.62953  7.49     0.25803  9.01542
  9.48284 19.49438  4.89155  4.1059   5.4276  11.76404  0.32847  0.88664
  5.0329   8.47305  7.47123  3.52836  4.7463  20.57319  1.0945   1.0593
  0.50036  2.74637  3.68295  1.27012  3.95951  8.93723  3.20928 20.83336
  4.20095  5.62164 10.25119  4.91103  0.51887  2.96032  3.4072   3.02152
  0.66248  5.47865  0.99331  2.9124   0.51748  9.15098  1.15294  2.73322
  0.49793 10.07384  7.51703  5.37855  5.07795  0.9396   6.3452   6.08222
  0.46035  5.04238  0.60346]
```

In [84]:

```
import seaborn as sns
sns.distplot(y_test-predictions)
```

Out[84]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2c4c97c9e08>
```

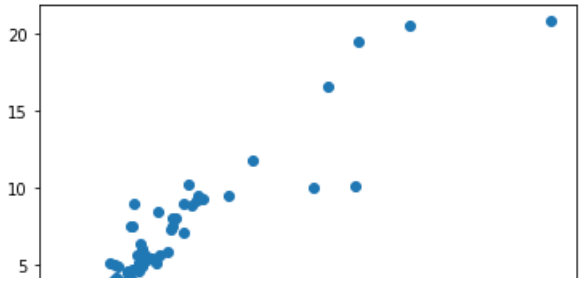


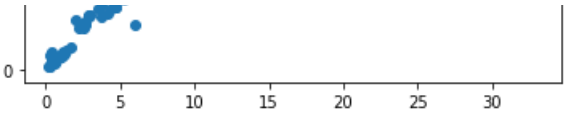
In [85]:

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

Out[85]:

```
<matplotlib.collections.PathCollection at 0x2c4c8113f48>
```





In [86]:

```
from sklearn import metrics
```

In [87]:

```
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: 0.9027013186813186
MSE: 4.110572026780222
RMSE: 2.0274545683640417
```

In [89]:

```
import pickle
file = open('G:/anurag1241/datascience/projects/ML/car price
prediction/random_forest_regression_model.pkl', 'wb')
```

In [90]:

```
pickle.dump(rf_random, file)
```

In [ ]:

```
```