In [1]:
```
#Random Forest Implementation for 4 variables

# packages to be used
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
import seaborn as sns
```

In [2]:
```
new_df = pd.read_csv("FinalNew_house.csv")
```

In [3]:
```
new_df.head()
```

Out[3]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grad
0	221900.0	3.0	1.0	1180.0	5650.0	1.0	0.0	0.0	3.0	7
1	538000.0	3.0	2.0	2570.0	7242.0	2.0	0.0	0.0	3.0	7
2	180000.0	2.0	1.0	770.0	10000.0	1.0	0.0	0.0	3.0	6
3	604000.0	4.0	3.0	1960.0	5000.0	1.0	0.0	0.0	5.0	7
4	510000.0	3.0	2.0	1680.0	8080.0	1.0	0.0	0.0	3.0	8

In [4]:
```
new_df.shape
```

Out[4]:
```
(17498, 19)
```

In [5]:
```
new_df = new_df.iloc[0:1000]
```

In [6]:
```
predictors = ['grade','lat','age','sqft_living15']
outcome = 'price'
```

In [7]:
```
#Partition Data
X = pd.get_dummies(new_df[predictors],drop_first=True)
y = new_df[outcome]

# Split dataset into train and test

train_X, valid_X, train_y, valid_y = train_test_split(X,y, test_size=0.2, rand
```

In [8]:
```
#Importing RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor

#Creating Random Forest Regressor
rfc = RandomForestRegressor(n_estimators = 1)
```

In [9]:
```
#We'll use x_train and y_train to fit our model.
rfc.fit(train_X,train_y)
```

Out[9]:
```
RandomForestRegressor(n_estimators=1)
```

In [10]:
```
#Let's calculate our model's score using valid_x and valid_y.
rfc.score(valid_X,valid_y)
```

Out[10]:
```
0.5733587841422826
```

In [11]:
```
#We have fitted the model and seen its performance. Let us predict the prices
rfc_pred = rfc.predict(valid_X)
print(rfc_pred)
```

```
[ 850000.          648000.          225000.          530000.
  339000.          855000.          280000.          225000.
  850000.          720000.          310000.          500000.
  264000.          631000.          312000.          650000.
  450000.          360000.          610000.          940000.
  374000.          318888.          825000.          650000.
  435000.          310000.          272000.          492000.
  699000.          240000.          780000.          299995.
  686000.          299995.          287500.          220000.
  261000.          235000.          550000.          385000.
  699000.          180250.          305000.          615000.
  250000.          802541.          370000.          309000.
  515000.          510000.          218000.          283000.
  379000.          673000.          385200.          400000.
  310000.          535000.          442000.          370000.
  372500.          399950.          327166.66666667  560000.
  315000.          604000.          275000.          300000.
  485500.          360000.          690000.          430000.
  425000.          245000.          650000.          525000.
  320000.          446500.          385200.          538000.
  261000.          438000.          430000.          920000.
  331000.          576000.          460000.          620000.
  250000.          538000.          245000.          325000.
  347500.          259950.          250000.          625000.
  437500.          180000.          435000.          188500.
  372500.          532170.          370000.          195000.
  554000.          280000.          287653.          653000.
  254000.          669950.          395000.          425000.
  592500.          610000.          245000.          522000.
  240000.          480000.          153000.          254000.
  560000.          220000.          600000.          360000.
  455000.          435000.          325000.          730000.
  310000.          215000.          825000.          550000.
  283000.          530000.          625000.          324000.
  390000.          337000.          625000.          920000.
  232000.          543500.          565000.          352000.
  543500.          425000.          642450.          269950.
  826000.          430000.          565000.          729500.
  208000.          770000.          550000.          550000.
  295000.          324500.          264000.          260000.
  323000.          196500.          287500.          395000.
  333500.          395000.          530000.          256883.
  214000.          327166.66666667  648000.          305000.
  210490.          397500.          949000.          365000.
  299995.          673000.          330000.          520000.
  700000.          442000.          245000.          290900.
  725000.          255000.         1099880.          465750.
  280000.          260000.          250000.          665000.
  543500.          510000.          327166.66666667  245000.
  650000.          488000.          315000.          420000.        ]
```

In [12]:
```
rsquared = metrics.r2_score(valid_y,rfc_pred)
adjusted_r_squared = 1 - (1-rsquared)*(len(valid_y)-1)/(len(valid_y)-valid_X.s
print('Mean absolute error: {}'.format(metrics.mean_absolute_error(valid_y, rf
print('Mean squared error: {}'.format(metrics.mean_squared_error(valid_y, rfc_
print('Root mean squared error: {}'.format(np.sqrt(metrics.mean_squared_error(
print('R Squared value: {}'.format(rsquared))
print('Adjusted R Squared Value: {}'.format(adjusted_r_squared))
```

```
Mean absolute error: 91168.855
Mean squared error: 16338348135.448336
Root mean squared error: 127821.54800912221
R Squared value: 0.5733587841422826
Adjusted R Squared Value: 0.5646071694580217
```

In [13]:
```
from dmba import regressionSummary
```

In [14]:
```
regressionSummary(valid_y, rfc_pred)
```

```
Regression statistics

                    Mean Error (ME) : 1944.3450
         Root Mean Squared Error (RMSE) : 127821.5480
              Mean Absolute Error (MAE) : 91168.8550
            Mean Percentage Error (MPE) : -4.0720
Mean Absolute Percentage Error (MAPE) : 21.1746
```

In [15]:
```
#ran the model 50 times with estimators ranging from 1 to 50. The root mean sq
RMSE_rfc = []
for i in range(1,50):
    rfc = RandomForestRegressor(n_estimators=i)
    rfc.fit(train_X,train_y)
    pred_i = rfc.predict(valid_X)
    RMSE_rfc.append((np.sqrt(metrics.mean_squared_error(valid_y, pred_i))))
print('Minimum Root Mean Squared Error is {} with {} estimators'.format(round(
```

```
Minimum Root Mean Squared Error is 94375.333 with 31 estimators
```

In [16]:
```
rfc = RandomForestRegressor(n_estimators=50)
rfc.fit(train_X,train_y)
pred_p = rfc.predict(valid_X)
rsquared_p = metrics.r2_score(valid_y,pred_p)
adjusted_r_squared_p = 1 - (1-rsquared_p)*(len(valid_y)-1)/(len(valid_y)-valid
print('Mean absolute error: {}'.format(metrics.mean_absolute_error(valid_y, pr
print('Mean squared error: {}'.format(metrics.mean_squared_error(valid_y, pred
print('Root mean squared error: {}'.format(np.sqrt(metrics.mean_squared_error(
print('R squared value: {}'.format(rsquared_p))
print('Adjusted squared value: {}'.format(adjusted_r_squared_p))
```

```
Mean absolute error: 69162.60126666666
Mean squared error: 9319310066.46894
Root mean squared error: 96536.57372451614
R squared value: 0.7566460363831469
Adjusted squared value: 0.7516541602063909
```

In [17]:
```
regressionSummary(valid_y, pred_p)
```

```
Regression statistics

                    Mean Error (ME) : -3433.8801
     Root Mean Squared Error (RMSE) : 96536.5737
          Mean Absolute Error (MAE) : 69162.6013
        Mean Percentage Error (MPE) : -4.9996
Mean Absolute Percentage Error (MAPE) : 16.5753
```

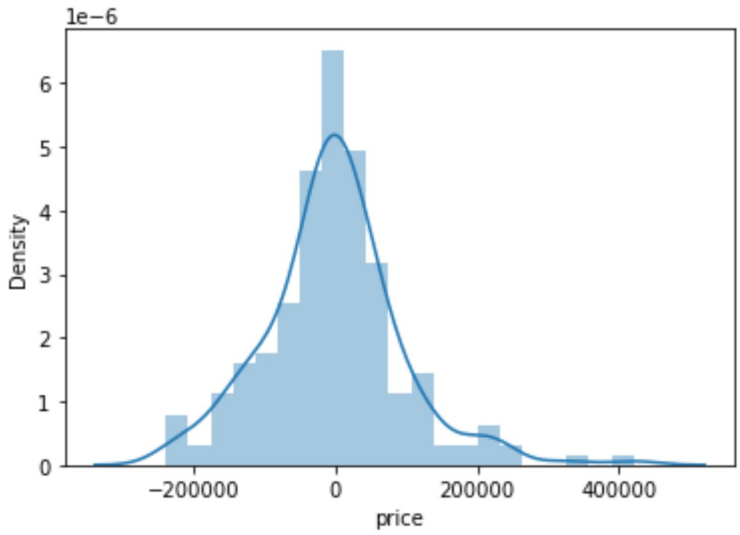
In [18]:
```
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly.offline as ply
import plotly.graph_objs as go
from plotly import tools
init_notebook_mode(connected=True)
from  plotly.offline import plot
import plotly.plotly as py
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode()
```

In [19]:
```
trace0 = go.Scatter(
x = valid_X.iloc[:,2],
y = valid_y,
mode = 'markers',
name = 'Test Set'
)
trace1 = go.Scatter(
x = valid_X.iloc[:,2],
y = pred_p,
opacity = 0.75,
mode = 'markers',
name = 'Predictions',
marker = dict(line = dict(color = 'black', width = 0.5))
)
data = [trace0, trace1]
ply.iplot(data)
```

In [20]:
```
residualr = (valid_y- pred_p)
sns.distplot(residualr);
```

```
C:\Users\kadam\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fut
ureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```



In [ ]: