Regression to predict the price of house using Incremental Extreme Machine Learning

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import keras
import math
from keras.models import Sequential
from keras.datasets import mnist
from keras.layers import Dense
from keras.optimizers import Adam
import random
import keras
import keras.utils
from keras.utils.np_utils import to_categorical
from keras import utils as np_utils
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
import time
from sklearn.metrics import mean_squared_error

import statistics
```

```
Using TensorFlow backend.
```

In [2]:

```
df = pd.read_csv('kc_house_data.csv')
print (df)
```

```
          price  bedrooms  bathrooms  sqft_living  sqft_lot  floors  \
0      221900.0         3       1.00         1180      5650     1.0
1      538000.0         3       2.25         2570      7242     2.0
2      180000.0         2       1.00          770     10000     1.0
3      604000.0         4       3.00         1960      5000     1.0
4      510000.0         3       2.00         1680      8080     1.0
...         ...       ...        ...          ...       ...     ...
21608  360000.0         3       2.50         1530      1131     3.0
21609  400000.0         4       2.50         2310      5813     2.0
21610  402101.0         2       0.75         1020      1350     2.0
21611  400000.0         3       2.50         1600      2388     2.0
21612  325000.0         2       0.75         1020      1076     2.0

       waterfront  view  condition  grade  sqft_above  sqft_basement  \
0               0     0          3      7        1180              0
1               0     0          3      7        2170            400
2               0     0          3      6         770              0
3               0     0          5      7        1050            910
4               0     0          3      8        1680              0
```

In [3]:

```
total_train = df.drop(columns="price")#-----------------------to guess the condition of the
a = df.iloc[:, 0]
total_labels = pd.DataFrame(a)
print(type(total_train))
print(type(total_labels))
print(total_labels)


start = time.time()
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
          price
0      221900.0
1      538000.0
2      180000.0
3      604000.0
4      510000.0
...         ...
21608  360000.0
21609  400000.0
21610  402101.0
21611  400000.0
21612  325000.0

[21613 rows x 1 columns]
```

In [4]:

```
x_train = total_train.loc[0:10000, :]
y_labels = total_labels.loc[0:10000, :]

x_test = total_train.loc[10001:20000, :]
y_test_labels = total_labels.loc[10001:20000, :]
```

In [5]:

```
print(x_test)
```

```
       bedrooms  bathrooms  sqft_living  sqft_lot  floors  waterfront  view
\
10001         5       3.25         3160     10587     1.0           0     0
10002         3       1.50         2020     11358     1.0           0     0
10003         4       3.75         3210      7054     2.0           0     0
10004         3       2.25         2350     51400     1.0           0     0
10005         4       2.50         1910     10221     2.0           0     0
...         ...        ...          ...       ...     ...         ...   ...
19996         3       2.25         1530      1305     2.0           0     0
19997         3       2.50         1600      6315     2.0           0     0
19998         2       1.50         1000      1251     2.0           0     0
19999         4       3.50         2440      3012     3.0           0     1
20000         3       2.50         1260      1102     3.0           0     0

       condition  grade  sqft_above  sqft_basement  yr_built  yr_renovated
\
10001          5      7        2190            970      1960             0
10002          4      6        1190            830      1956             0
10003          4      8        3210              0      1985             0
10004          3      7        1390            960      1977             0
10005          3      8        1910              0      1994             0
...          ...    ...         ...            ...       ...           ...
19996          3      7        1116            414      2007             0
19997          3      8        1600              0      2013             0
19998          3      7         930             70      2006             0
19999          3      8        2440              0      2005             0
20000          3      8        1260              0      2007             0

       zipcode      lat     long  sqft_living15  sqft_lot15
10001    98034  47.7238 -122.165           2200        7761
10002    98033  47.6641 -122.185           2370        9520
10003    98052  47.7268 -122.103           2350        8020
10004    98077  47.7417 -122.053           2350       51400
10005    98038  47.3810 -122.035           2210        8705
...        ...      ...      ...            ...         ...
19996    98177  47.7034 -122.357           1320        1427
19997    98092  47.2611 -122.198           1608        4300
19998    98199  47.6529 -122.384           1420        1187
19999    98117  47.6923 -122.392           1860        4650
20000    98107  47.6750 -122.387           1320        2500

[10000 rows x 18 columns]
```

In [6]:

```
X = (x_train, y_labels)
Y = (x_test, y_test_labels)
```

In [7]:

```
print(x_train)
```

```
       bedrooms  bathrooms  sqft_living  sqft_lot  floors  waterfront  view  \
0             3       1.00         1180      5650     1.0           0     0
1             3       2.25         2570      7242     2.0           0     0
2             2       1.00          770     10000     1.0           0     0
3             4       3.00         1960      5000     1.0           0     0
4             3       2.00         1680      8080     1.0           0     0
...         ...        ...          ...       ...     ...         ...   ...
9996          3       1.50         1700      9579     1.0           0     0
9997          4       1.00         1550      4750     1.5           0     0
9998          3       1.75         1680      8100     1.0           0     2
9999          3       2.25         1680     35127     2.0           0     0
10000         4       2.50         1910     10300     1.0           0     0

       condition  grade  sqft_above  sqft_basement  yr_built  yr_renovated  \
0              3      7        1180              0      1955             0
1              3      7        2170            400      1951          1991
2              3      6         770              0      1933             0
3              5      7        1050            910      1965             0
4              3      8        1680              0      1987             0
...          ...    ...         ...            ...       ...           ...
9996           4      7        1100            600      1962             0
9997           3      7        1550              0      1919             0
9998           3      8        1680              0      1950             0
9999           3      7        1680              0      1987             0
10000          3      8        1910              0      1921          1968

       zipcode      lat     long  sqft_living15  sqft_lot15
0        98178  47.5112 -122.257           1340        5650
1        98125  47.7210 -122.319           1690        7639
2        98028  47.7379 -122.233           2720        8062
3        98136  47.5208 -122.393           1360        5000
4        98074  47.6168 -122.045           1800        7503
...        ...      ...      ...            ...         ...
9996     98023  47.3209 -122.338           1700        9628
9997     98117  47.6824 -122.389           1320        4750
9998     98177  47.7212 -122.364           1880        7750
9999     98092  47.3025 -122.067           1820       35166
10000    98177  47.7581 -122.359           1910        7750

[10001 rows x 18 columns]
```

In [8]:

```
x_train=pd.DataFrame(x_train).to_numpy()

#w = np.random.rand(18,1000)*100#------------------------------creating random weight matri
#weights = pd.DataFrame(w)

#weights =to_numpy.DataFrame(w)
#weights=pd.DataFrame(weights).to_numpy()

#weights_transpose = np.transpose(weights)
#print(weights.shape)
print(x_train.shape)
#print(type(weights))
#print(type(x_train))

y_labels=pd.DataFrame(y_labels).to_numpy()
```

```
(10001, 18)
```

In [ ]:

```

```

In [9]:

```
print(x_train)
```

```
[[ 3.00000e+00  1.00000e+00  1.18000e+03 ... -1.22257e+02  1.34000e+03
   5.65000e+03]
 [ 3.00000e+00  2.25000e+00  2.57000e+03 ... -1.22319e+02  1.69000e+03
   7.63900e+03]
 [ 2.00000e+00  1.00000e+00  7.70000e+02 ... -1.22233e+02  2.72000e+03
   8.06200e+03]
 ...
 [ 3.00000e+00  1.75000e+00  1.68000e+03 ... -1.22364e+02  1.88000e+03
   7.75000e+03]
 [ 3.00000e+00  2.25000e+00  1.68000e+03 ... -1.22067e+02  1.82000e+03
   3.51660e+04]
 [ 4.00000e+00  2.50000e+00  1.91000e+03 ... -1.22359e+02  1.91000e+03
   7.75000e+03]]
```

In [21]:

```
start = time.time()
l=0
acc=0
l_max=50#------------------------------------------------------------setting up number neu
beta=0
error=0
rmse=0
final=0.1
d=0
p=0

for i in range(l_max):





    if(final>0.00005):
        w = np.random.rand(18,i)*100#-----------------------------creating random weight m
        weights = pd.DataFrame(w)

        #weights =to_numpy.DataFrame(w)
        weights=pd.DataFrame(weights).to_numpy()

        weights_transpose = np.transpose(weights)#------------------transposing weight
        h_new = np.dot(x_train,weights)
        h_inv = np.linalg.pinv(h_new)

        beta = np.dot(h_inv, y_labels)
    #print(beta.shape)
    #print(h_new.shape)
        predicted_output = np.dot(h_new,beta)
        rounded_labels=np.round(predicted_output)
        rounded=pd.DataFrame(rounded_labels).to_numpy()
        #rmse = rmse + ((y_labels[i] - predicted_output[i])**2)
        #if(i==0):
            #i=i+1
        #rmse1=math.sqrt((1/i)*rmse)
        #print(rmse1)

        d = mean_squared_error(y_labels, rounded)
        rmse=1/10000*(math.sqrt(d))
        print("RMSE", + i, + rmse)


    #print(predicted_output.shape)



   # print(predicted_output[i][:])
   # print(predicted_output.shape)
        #print(y_train)

   # print("p",+ rounded)
    #print(type(predicted_output))

    #print("y",+rounded)

end = time.time()
```

```
print("Time elapsed",end - start)
    #print(h_inv.shape)
    #print(h_new.shape)
    #print(beta)
    #print(weights.shape)
#print(error[1][9])
```

```
RMSE 0 65.3187934906508
RMSE 1 58.59238919015065
RMSE 2 38.14558851827319
RMSE 3 37.68989053431443
RMSE 4 26.692054763073475
RMSE 5 26.616501627450887
RMSE 6 26.5393831050097
RMSE 7 25.954065728475634
RMSE 8 25.613605014950217
RMSE 9 24.65734693954081
RMSE 10 23.745741270183125
RMSE 11 23.041154691045442
RMSE 12 22.670759022773435
RMSE 13 22.677096144179103
RMSE 14 22.022490048761114
RMSE 15 21.975676470442103
RMSE 16 21.471165237021943
RMSE 17 20.80337475941956
RMSE 18 20.80337479162249
RMSE 19 20.80337478474516
RMSE 20 20.80337478474516
RMSE 21 20.80337478072643
RMSE 22 20.80337478474516
RMSE 23 20.803374781921402
RMSE 24 20.803374790262605
RMSE 25 20.803374790262605
RMSE 26 20.803374790262605
RMSE 27 20.80337478474516
RMSE 28 20.80337478474516
RMSE 29 20.80337478474516
RMSE 30 20.80337479308636
RMSE 31 20.803374790262605
RMSE 32 20.80337478474516
RMSE 33 20.80337479308636
RMSE 34 20.80337479308636
RMSE 35 20.80337478474516
RMSE 36 20.80337479308636
RMSE 37 20.80337478474516
RMSE 38 20.80337478474516
RMSE 39 20.803374790262605
RMSE 40 20.80337478474516
RMSE 41 20.80337478474516
RMSE 42 20.80337478474516
RMSE 43 20.80337478474516
RMSE 44 20.80337479308636
RMSE 45 20.80337478474516
RMSE 46 20.80337479308636
RMSE 47 20.80337478474516
RMSE 48 20.80337478474516
RMSE 49 20.80337479308636
Time elapsed 0.992955207824707
```

In [ ]:

```
#h_inv = np.linalg.pinv(h_new)
#print(h_inv.shape)
```

In [ ]:

```
#beta = np.dot(h_inv, y_labels)
```

In [ ]:

```
#print(beta)
```

In [ ]:

```
#predicted_output= np.dot(h_new, beta) #--------------------------------------------------
#print(predicted_output.shape)
```

In [ ]:

```
#print(np.round(predicted_output))
#rounded_labels=np.round(predicted_output)
#rounded=pd.DataFrame(rounded_labels).to_numpy()
#print(rounded_labels)
#print(type(y_labels))
#print(type(rounded_labels))
```

In [ ]:

```
#acc = accuracy_score(rounded_labels,y_labels)#----------------------------FInding accurac
#print("Accuracy is", + acc*100, "%")
#end = time.time()
#print("Time elapsed",end - start)
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```