Housing Price Prediction Linear Regression

Submitted by Harsh Srivastava

117CS0755

importing libraries

In [1]:

```
import os
import numpy as np
import pandas as pd
%pylab inline
import matplotlib.pyplot as plt
```

```
Populating the interactive namespace from numpy and matplotlib
```

loading the training dataset

In [2]:

```
train_dataset_path = None
for root, dirs, files in os.walk(".", topdown=False) :
    for name in files:
        if name.endswith('train.csv') :
            train_dataset_path = os.path.join(root, name)
            break
    if train_dataset_path != None :
        break

train_dataset = pd.read_csv(train_dataset_path)
train_dataset
```

Out[2]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandConto
0	1	60	RL	65.0	8450	Pave	NaN	Reg	l
1	2	20	RL	80.0	9600	Pave	NaN	Reg	l
2	3	60	RL	68.0	11250	Pave	NaN	IR1	l
3	4	70	RL	60.0	9550	Pave	NaN	IR1	l
4	5	60	RL	84.0	14260	Pave	NaN	IR1	l
...	...	...	...	...	...	...	...	...	
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	l
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	l
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	l
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	l
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	l

1460 rows × 81 columns

independent variable and dependent variables

In [3]:

```
X = train_dataset.iloc[:, 1:-1]

tot_size = X.shape[0]
print(X.shape)
X
```

```
(1460, 79)
```

Out[3]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Ut
0	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	A
1	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	A
2	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	A
3	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	A
4	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	A
...	...	...	...	...	...	...	...	...	
1455	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	A
1456	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	A
1457	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	A
1458	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	A
1459	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	A

1460 rows × 79 columns

In [4]:

```
y = train_dataset['SalePrice'].values
y.shape
```

Out[4]:

```
(1460,)
```

removing columns with more than 50 percent NA values

In [5]:

```
X_drop_NA = X.dropna(axis = 1, thresh = (0.5 * tot_size))
X_drop_NA
```

Out[5]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities
0	60	RL	65.0	8450	Pave	Reg	Lvl	AllPub
1	20	RL	80.0	9600	Pave	Reg	Lvl	AllPub
2	60	RL	68.0	11250	Pave	IR1	Lvl	AllPub
3	70	RL	60.0	9550	Pave	IR1	Lvl	AllPub
4	60	RL	84.0	14260	Pave	IR1	Lvl	AllPub
...	...	...	...	...	...	...	...	...
1455	60	RL	62.0	7917	Pave	Reg	Lvl	AllPub
1456	20	RL	85.0	13175	Pave	Reg	Lvl	AllPub
1457	70	RL	66.0	9042	Pave	Reg	Lvl	AllPub
1458	20	RL	68.0	9717	Pave	Reg	Lvl	AllPub
1459	20	RL	75.0	9937	Pave	Reg	Lvl	AllPub

1460 rows × 75 columns

In [6]:

```
X_drop_NA.mean()
```

Out[6]:

```
MSSubClass          56.897260
LotFrontage         70.049958
LotArea          10516.828082
OverallQual          6.099315
OverallCond          5.575342
YearBuilt         1971.267808
YearRemodAdd      1984.865753
MasVnrArea         103.685262
BsmtFinSF1         443.639726
BsmtFinSF2          46.549315
BsmtUnfSF          567.240411
TotalBsmtSF       1057.429452
1stFlrSF          1162.626712
2ndFlrSF           346.992466
LowQualFinSF         5.844521
GrLivArea         1515.463699
BsmtFullBath         0.425342
BsmtHalfBath         0.057534
FullBath             1.565068
HalfBath             0.382877
BedroomAbvGr         2.866438
KitchenAbvGr         1.046575
TotRmsAbvGrd         6.517808
Fireplaces           0.613014
GarageYrBlt       1978.506164
GarageCars           1.767123
GarageArea         472.980137
WoodDeckSF          94.244521
OpenPorchSF         46.660274
EnclosedPorch       21.954110
3SsnPorch            3.409589
ScreenPorch         15.060959
PoolArea             2.758904
MiscVal             43.489041
MoSold               6.321918
YrSold            2007.815753
dtype: float64
```

Replacing NA values

In [7]:

```
X_fill = X_drop_NA.fillna(X_drop_NA.mean())
X_fill
```

Out[7]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities
0	60	RL	65.0	8450	Pave	Reg	Lvl	AllPub
1	20	RL	80.0	9600	Pave	Reg	Lvl	AllPub
2	60	RL	68.0	11250	Pave	IR1	Lvl	AllPub
3	70	RL	60.0	9550	Pave	IR1	Lvl	AllPub
4	60	RL	84.0	14260	Pave	IR1	Lvl	AllPub
...	...	...	...	...	...	...	...	...
1455	60	RL	62.0	7917	Pave	Reg	Lvl	AllPub
1456	20	RL	85.0	13175	Pave	Reg	Lvl	AllPub
1457	70	RL	66.0	9042	Pave	Reg	Lvl	AllPub
1458	20	RL	68.0	9717	Pave	Reg	Lvl	AllPub
1459	20	RL	75.0	9937	Pave	Reg	Lvl	AllPub

1460 rows × 75 columns

removing columns with only a single value in all rows

In [8]:

```
for col in X_fill.columns:
    if len(X_fill[col].unique()) == 1:
        X_fill.drop(col,inplace=True,axis=1)

X_fill
```

Out[8]:

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities
0	60	RL	65.0	8450	Pave	Reg	Lvl	AllPub
1	20	RL	80.0	9600	Pave	Reg	Lvl	AllPub
2	60	RL	68.0	11250	Pave	IR1	Lvl	AllPub
3	70	RL	60.0	9550	Pave	IR1	Lvl	AllPub
4	60	RL	84.0	14260	Pave	IR1	Lvl	AllPub
...	...	...	...	...	...	...	...	...
1455	60	RL	62.0	7917	Pave	Reg	Lvl	AllPub
1456	20	RL	85.0	13175	Pave	Reg	Lvl	AllPub
1457	70	RL	66.0	9042	Pave	Reg	Lvl	AllPub
1458	20	RL	68.0	9717	Pave	Reg	Lvl	AllPub
1459	20	RL	75.0	9937	Pave	Reg	Lvl	AllPub

1460 rows × 75 columns

adding a column of ones

In [9]:

```
X_fill = pd.concat([pd.Series(1, index=X_fill.index, name='ones'), X_fill], axis=1)
X_fill
```

Out[9]:

	ones	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Ut
0	1	60	RL	65.0	8450	Pave	Reg	Lvl	A
1	1	20	RL	80.0	9600	Pave	Reg	Lvl	A
2	1	60	RL	68.0	11250	Pave	IR1	Lvl	A
3	1	70	RL	60.0	9550	Pave	IR1	Lvl	A
4	1	60	RL	84.0	14260	Pave	IR1	Lvl	A
...	...	...	...	...	...	...	...	...	
1455	1	60	RL	62.0	7917	Pave	Reg	Lvl	A
1456	1	20	RL	85.0	13175	Pave	Reg	Lvl	A
1457	1	70	RL	66.0	9042	Pave	Reg	Lvl	A
1458	1	20	RL	68.0	9717	Pave	Reg	Lvl	A
1459	1	20	RL	75.0	9937	Pave	Reg	Lvl	A

1460 rows × 76 columns

One Hot Encoding the dataframe

In [10]:

```
X_one_hot = pd.get_dummies(X_fill)
X_one_hot
```

Out[10]:

	ones	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemo
0	1	60	65.0	8450	7	5	2003	
1	1	20	80.0	9600	6	8	1976	
2	1	60	68.0	11250	7	5	2001	
3	1	70	60.0	9550	7	5	1915	
4	1	60	84.0	14260	8	5	2000	
...	...	...	...	...	...	...	...	
1455	1	60	62.0	7917	6	5	1999	
1456	1	20	85.0	13175	6	6	1978	
1457	1	70	66.0	9042	7	9	1941	
1458	1	20	68.0	9717	5	6	1950	
1459	1	20	75.0	9937	5	6	1965	

1460 rows × 276 columns

normalizing the columns

In [11]:

```
X_norm = X_one_hot / X_one_hot.max()
X_norm
```

Out[11]:

	ones	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRem
0	1.0	0.315789	0.207668	0.039258	0.7	0.555556	0.996517	0.9
1	1.0	0.105263	0.255591	0.044600	0.6	0.888889	0.983085	0.9
2	1.0	0.315789	0.217252	0.052266	0.7	0.555556	0.995522	0.9
3	1.0	0.368421	0.191693	0.044368	0.7	0.555556	0.952736	0.9
4	1.0	0.315789	0.268371	0.066250	0.8	0.555556	0.995025	0.9
...	...	...	...	...	...	...	...	
1455	1.0	0.315789	0.198083	0.036781	0.6	0.555556	0.994527	0.9
1456	1.0	0.105263	0.271565	0.061209	0.6	0.666667	0.984080	0.9
1457	1.0	0.368421	0.210863	0.042008	0.7	1.000000	0.965672	0.9
1458	1.0	0.105263	0.217252	0.045144	0.5	0.666667	0.970149	0.9
1459	1.0	0.105263	0.239617	0.046166	0.5	0.666667	0.977612	0.9

1460 rows × 276 columns

splitting train and test data again

In [12]:

```
train_size = int(tot_size * 0.8)
test_size = tot_size - train_size

X_train = X_norm.iloc[0: train_size, :]
X_test = X_norm.iloc[train_size: tot_size, :]
y_train = y[0: train_size]
y_test = y[train_size: tot_size]
```

In [13]:

```
X_train
```

Out[13]:

	ones	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRem
0	1.0	0.315789	0.207668	0.039258	0.7	0.555556	0.996517	0.9
1	1.0	0.105263	0.255591	0.044600	0.6	0.888889	0.983085	0.9
2	1.0	0.315789	0.217252	0.052266	0.7	0.555556	0.995522	0.9
3	1.0	0.368421	0.191693	0.044368	0.7	0.555556	0.952736	0.9
4	1.0	0.315789	0.268371	0.066250	0.8	0.555556	0.995025	0.9
...	...	...	...	...	...	...	...	
1163	1.0	0.473684	0.191693	0.059932	0.4	0.444444	0.979602	0.9
1164	1.0	0.421053	0.223802	0.075063	0.5	0.777778	0.984080	0.9
1165	1.0	0.105263	0.252396	0.044326	0.7	0.555556	0.999502	0.9
1166	1.0	0.105263	0.204473	0.048665	0.8	0.555556	0.999005	0.9
1167	1.0	0.315789	0.185304	0.050417	0.6	0.555556	0.995025	0.9

1168 rows × 276 columns

In [14]:

```
X_train_np = X_train.to_numpy()
X_train_np
```

Out[14]:

```
array([[1.        , 0.31578947, 0.20766773, ..., 0.        , 1.        ,
        0.        ],
       [1.        , 0.10526316, 0.25559105, ..., 0.        , 1.        ,
        0.        ],
       [1.        , 0.31578947, 0.2172524 , ..., 0.        , 1.        ,
        0.        ],
       ...,
       [1.        , 0.10526316, 0.25239617, ..., 0.        , 0.        ,
        1.        ],
       [1.        , 0.10526316, 0.20447284, ..., 0.        , 1.        ,
        0.        ],
       [1.        , 0.31578947, 0.18530351, ..., 0.        , 1.        ,
        0.        ]])
```

In [15]:

```
X_test
```

Out[15]:

	ones	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRem
1168	1.0	0.368421	0.383387	0.063778	0.6	0.777778	0.962687	0.9
1169	1.0	0.315789	0.376997	0.166136	1.0	0.555556	0.992537	0.9
1170	1.0	0.421053	0.242812	0.045901	0.6	0.666667	0.983582	0.9
1171	1.0	0.105263	0.242812	0.042370	0.6	0.666667	0.974129	0.9
1172	1.0	0.842105	0.111821	0.018662	0.7	0.555556	0.998010	0.9
...	...	...	...	...	...	...	...	
1455	1.0	0.315789	0.198083	0.036781	0.6	0.555556	0.994527	0.9
1456	1.0	0.105263	0.271565	0.061209	0.6	0.666667	0.984080	0.9
1457	1.0	0.368421	0.210863	0.042008	0.7	1.000000	0.965672	0.9
1458	1.0	0.105263	0.217252	0.045144	0.5	0.666667	0.970149	0.9
1459	1.0	0.105263	0.239617	0.046166	0.5	0.666667	0.977612	0.9

292 rows × 276 columns

In [16]:

```
X_test_np = X_test.to_numpy()
X_test_np
```

Out[16]:

```
array([[1.        , 0.36842105, 0.38338658, ..., 0.        , 1.        ,
        0.        ],
       [1.        , 0.31578947, 0.37699681, ..., 0.        , 1.        ,
        0.        ],
       [1.        , 0.42105263, 0.2428115 , ..., 0.        , 1.        ,
        0.        ],
       ...,
       [1.        , 0.36842105, 0.21086262, ..., 0.        , 1.        ,
        0.        ],
       [1.        , 0.10526316, 0.2172524 , ..., 0.        , 1.        ,
        0.        ],
       [1.        , 0.10526316, 0.23961661, ..., 0.        , 1.        ,
        0.        ]])
```

In [17]:

```
print(y_train)
print(y_test)
```

```
[208500 181500 223500 ... 233170 245350 173000]
[235000 625000 171000 163000 171900 200500 239000 285000 119500 115000
 154900  93000 250000 392500 745000 120000 186700 104900  95000 262000
 195000 189000 168000 174000 125000 165000 158000 176000 219210 144000
 178000 148000 116050 197900 117000 213000 153500 271900 107000 200000
 140000 290000 189000 164000 113000 145000 134500 125000 112000 229456
  80500  91500 115000 134000 143000 137900 184000 145000 214000 147000
 367294 127000 190000 132500 101800 142000 130000 138887 175500 195000
 142500 265900 224900 248328 170000 465000 230000 178000 186500 169900
 129500 119000 244000 171750 130000 294000 165400 127500 301500  99900
 190000 151000 181000 128900 161500 180500 181000 183900 122000 378500
 381000 144000 260000 185750 137000 177000 139000 137000 162000 197900
 237000  68400 227000 180000 150500 139000 169000 132500 143000 190000
 278000 281000 180500 119500 107500 162900 115000 138500 155000 140000
 160000 154000 225000 177500 290000 232000 130000 325000 202500 138000
 147000 179200 335000 203000 302000 333168 119000 206900 295493 208900
 275000 111000 156500  72500 190000  82500 147000  55000  79000 130500
 256000 176500 227000 132500 100000 125500 125000 167900 135000  52500
 200000 128500 123000 155000 228500 177000 155835 108500 262500 283463
 215000 122000 200000 171000 134900 410000 235000 170000 110000 149900
 177500 315000 189000 260000 104900 156932 144152 216000 193000 127000
 144000 232000 105000 165500 274300 466500 250000 239000  91000 117000
  83000 167500  58500 237500 157000 112000 105000 125500 250000 136000
 377500 131000 235000 124000 123000 163000 246578 281213 160000 137500
 138000 137450 120000 193000 193879 282922 105000 275000 133000 112000
 125500 215000 230000 140000  90000 257000 207000 175900 122500 340000
 124000 223000 179900 127500 136500 274970 144000 142000 271000 140000
 119000 182900 192140 143750  64500 186500 160000 174000 120500 394617
 149700 197000 191000 149300 310000 121000 179600 129000 157900 240000
 112000  92000 136000 287090 145000  84500 185000 175000 210000 266500
 142125 147500]
```

Calculating parameters

In [18]:

```
B = np.dot(np.dot(np.linalg.pinv(np.dot(X_train_np.T, X_train_np)), X_train_np.T), y_train)
B
```

Out[18]:

```
array([ 1.14984115e+05, -1.19468905e+04,  2.23632175e+04,  1.43975678e+05,
        6.22517575e+04,  4.93432859e+04,  5.84231316e+05,  2.69478922e+05,
        2.93739933e+04,  9.21150529e+04,  1.55228244e+04, -2.73151475e+03,
        8.77900464e+04,  1.21767955e+05,  6.03182982e+04,  4.90137765e+02,
        1.23391155e+05,  3.72960798e+03,  4.32626808e+02,  1.69333368e+04,
        3.74503895e+03, -3.84116839e+04, -5.20541135e+04,  4.48647349e+04,
        1.93005503e+04, -7.24667411e+04,  3.75823024e+03,  3.51454691e+04,
        1.29333327e+04, -4.48175446e+03,  5.26531655e+03,  9.04729853e+03,
        2.07149800e+04, -1.21913319e+04,  1.06347259e+04, -8.74773575e+03,
       -1.65521873e+06,  4.00454512e+02,  3.01482857e+04,  2.82312707e+04,
        3.04870108e+04,  2.57170901e+04,  4.31770874e+04,  7.18070270e+04,
        2.43860499e+04,  2.78774789e+04,  3.58379152e+04,  2.68826661e+04,
        2.67384006e+04,  3.30896953e+04,  2.18754992e+04,  3.32805159e+04,
        7.45834317e+04,  4.04006828e+04,  2.57682151e+04,  3.18075229e+04,
        1.79769493e+04,  1.44098424e+04,  2.50215816e+04,  4.98063785e+04,
        5.66466331e+04,  8.53110502e+03,  6.81376321e+03,  1.43379407e+04,
        1.18058281e+04,  9.79691771e+02, -2.75322550e+03, -4.86598575e+03,
        1.45800586e+04, -1.84111661e+04, -2.89992147e+03, -7.92725631e+03,
        5.45158275e+03, -1.37091411e+04, -1.24604546e+04,  2.37143559e+04,
       -1.01269664e+04,  3.41394659e+04,  2.52295836e+04, -8.86693210e+03,
       -4.67114000e+03, -5.43183325e+03,  3.18132271e+03,  1.40742374e+04,
        5.08341492e+04, -6.56826823e+03,  8.53442039e+03,  1.25806226e+04,
        1.28436811e+04,  2.31101497e+04,  1.03615337e+04,  2.13311905e+04,
       -9.59061323e+03,  1.72599446e+04,  7.60903259e+03,  1.94785709e+04,
        3.80052078e+04,  4.10057590e+04,  4.12648127e+04,  8.83748918e+04,
       -1.70824792e+05,  3.64720254e-06,  2.73766345e+04,  4.97815984e+04,
        3.16135163e+04,  3.14617042e+04,  2.60058903e+04,  1.14570398e+04,
        1.44459632e+04,  1.83432275e+04,  2.41141196e+04,  1.95288752e+04,
       -4.00004524e+03,  9.51408843e+03,  1.37688348e+04,  1.81083605e+04,
        1.56066495e+04,  1.57207058e+04,  2.25707456e+04,  2.34239540e+04,
        2.17521163e+04,  3.15165907e+04,  2.72349277e-06, -6.28876081e-06,
       -1.47916141e+04,  7.12963614e+04,  4.87742647e+04, -5.00598043e-06,
       -1.16902714e+04, -1.94347772e+04,  4.08301489e+04,  1.50870055e+04,
       -5.87524869e+03,  1.96561217e+04,  2.05084406e+04, -2.34497117e-06,
        2.37384195e+04,  5.35252773e+03,  1.01698840e-06,  1.53969399e+04,
        3.21503141e+03,  6.02412403e+03,  6.43454227e+03,  3.50943769e+03,
       -1.44790952e+03,  3.38467838e+03,  9.76399651e+03,  1.15361754e+04,
        5.61884458e+03,  1.57417762e+04, -6.59665107e-07, -6.01968534e+03,
        8.71197397e+03,  8.28985764e+03,  3.96833586e+03, -1.73158588e+04,
        9.49782324e+03,  2.07327750e+03,  1.59507722e+04,  1.63960183e+04,
        1.83988314e+04,  1.23719706e+04,  9.37514584e+03,  1.53129169e+04,
        1.64520940e+04,  2.07550499e+04,  3.80758785e+04,  3.66255756e+04,
        2.00951908e+04,  2.01874633e+04,  2.75160879e+04,  1.88126353e+04,
        1.85203222e+04,  2.84425022e+04,  2.16925658e+04,  2.04520262e+04,
        2.20886118e+04,  2.32125341e+04,  1.12760042e+04,  4.12822199e+04,
       -3.32728881e+03,  7.31122007e+03, -1.14228093e+04, -1.03020345e+04,
       -8.04298568e+03, -2.11284909e+04, -2.36129738e+04,  4.20389510e+04,
       -1.97540888e+04,  1.13314714e+04,  2.37589983e+04,  9.24549451e+03,
        7.39163129e+03, -4.23736151e+03, -2.74762755e+03,  5.05635443e+02,
       -8.26704229e+03, -5.53581715e+03, -2.17439560e+03,  1.90280702e+04,
        9.45921047e+03,  1.73440159e+04,  8.45907816e+03,  1.13228119e+04,
        1.15961584e+04,  2.52635313e-07,  3.05837494e+04,  2.79880376e+04,
        1.61660843e+04, -2.03915055e-08,  4.02462439e+04,  2.22793913e+04,
        2.45204276e+04,  1.78494441e+04,  3.22079080e+04,  1.81269383e+04,
```

```
 5.78852571e+04,  5.70988572e+04,  2.97554224e+04,  3.03087228e+04,
 1.88402094e+04,  8.05550664e+03,  2.80242568e+04,  4.43430086e+04,
 2.51741599e+04,  2.25738688e+04,  2.28930706e+04,  2.14300146e+04,
 7.03482697e+03,  2.33823625e+04,  2.76021521e+04,  1.50871879e+04,
-1.54476995e+04,  3.58952719e+04,  4.97954452e+03, -5.99981754e+03,
-5.15355724e+03, -8.31491604e+02, -4.93838959e+03, -1.75295930e+04,
-2.50347493e+03,  2.80721513e+03,  1.60852070e+03, -1.28756947e+03,
 1.43627714e+03, -4.71877502e+03, -6.03558976e+03, -4.71426182e+03,
 8.21554324e+04, -2.63304662e+04, -1.39337601e+04, -3.84277066e+04,
-1.89321194e+04, -8.15600648e+04,  1.55078199e+04,  1.86512257e+04,
 1.74134044e+04,  1.45189949e+04,  3.97023523e+04,  3.52367470e+04,
 4.00450103e+04, -5.22979594e+02,  3.37271174e+04,  2.74871606e+04,
 1.52660011e+04,  7.90864250e+03, -3.62434805e+03,  3.17618358e+04,
 2.15000120e+03,  8.30681842e+02,  1.49153323e+04,  2.88303154e+04,
 2.40646614e+04,  1.73043423e+04,  2.25159779e+04,  7.35348205e+03])
```

predicting for Test dataset

In [19]:

```
y_pred = np.dot(X_test, B)
y_pred
```

Out[19]:

```
array([210947.4909753 , 480609.01991827, 128304.61711888, 161166.44024062,
       157588.26485076, 267682.06976307, 228703.03552421, 323621.24219173,
       113633.30033833, 109301.8781949 , 129216.25857178,  84717.01422022,
       224385.30509416, 302912.64931474, 491622.57075511, 149190.74711262,
       192830.54796897, 118588.12610955,  70558.71723295, 268968.94994475,
       181935.16908547, 191340.59838642, 188878.00782165, 172676.66694686,
       137329.18081188, 154853.48312179, 153594.28846108, 167800.95002185,
       230819.18497634, 129641.8117546 , 174261.66664885, 140903.74465343,
        85810.53085437, 197416.70559078, 138298.9288699 , 193009.87750846,
       140887.87261058, 261413.94100972, 129244.89465975, 216273.55409547,
       140576.35071882, 311009.81792994, 204109.56352344, 196660.68118667,
        67122.84173962, 147440.52347524, 120395.33336576, 111360.49676115,
       106073.91587242, 242466.72320594,  84566.72329952, 108112.64053032,
        93418.45053108, 114622.18773442, 167648.33560981, 159855.54107267,
       200460.51252411, 139963.00149435, 238205.20872377, 149475.61530982,
       367752.25018687, 136722.60816692, 160566.97559882, 131628.70297076,
        78274.34254843, 128188.59928787, 129806.16745129, 154420.9566009 ,
       184490.08739924, 198104.47443094, 134048.63485578, 225585.65222569,
       209690.26202557, 264149.33200238, 177245.94217858, 399378.01744337,
       216853.38826188, 205768.28703009, 215750.68735815, 143106.63830625,
       123381.01101964, 118421.45189076, 280168.59019316, 195942.27180831,
       100985.60467299, 277971.0888466 , 178963.81031852, 138324.9831728 ,
       324319.89440129,  79036.48611888, 196170.15261907, 144196.89142394,
       205499.23389275, 119720.95320477, 154417.08998679, 174767.08668441,
       132347.88940163, 168439.61223331, 117553.50212545, 308369.69282335,
       370185.27239466, 133406.96554194, 152194.13415687, 180809.1612152 ,
       127913.73371421, 181668.13832637, 154730.93550981, 167182.10977556,
       149413.03880064, 190423.27667736, 220008.6006239 ,  58793.39735899,
       208533.14406171, 198036.05032798, 153940.91023764, 151238.25999671,
       195069.41670433, 135511.74935961, 150703.04852687, 193783.65575783,
       281275.04922123, 316353.22007484, 144263.94485933, 128870.75917153,
       140381.21082131, 182170.40610047, 113158.92056848, 119520.05410536,
       142709.53169696, 146905.95993828, 778596.2934026 , 169558.59970407,
       238279.42193702, 169739.77130511, 308744.96637144, 238159.05274264,
       135715.04918853, 356042.3118492 , 208648.02423944, 138606.80243001,
       163948.30125286, 184590.98945735, 299611.41058378, 217222.96074411,
       355020.03935599, 323095.0926595 , 102977.8797151 , 197325.28055417,
       288741.28944276, 201379.65267038, 251896.91599548,  90507.2812193 ,
       177092.03385434,  50656.29219301, 249609.89536967, 108687.65636311,
       299029.22066268,  80510.7365162 ,  83015.13417794, 139597.87992143,
       230325.94525819, 182787.69940319, 242455.61645321, 128604.84019022,
        98445.94503118, 135764.22140885, 115572.68554574, 169413.14083462,
       128651.73138323,  56058.33971675, 224624.48892944, 124449.48597255,
       103866.51575381, 153092.49712847, 229175.84375638, 183131.13960272,
       195354.69293161,  85644.42837706, 262416.2461626 , 286951.011544  ,
       242466.91566187, 121903.70087629, 199281.91320425, 165444.08806037,
       149691.3374136 , 385842.64494983, 219912.84556157, 183825.3464329 ,
        90999.44561841, 161153.48897361, 166478.54199496, 347790.02767758,
       216329.09135497, 298358.01186872, 103555.16036017, 184489.36570046,
       138119.27908075, 219990.54247188, 198081.47533973, 131131.55171323,
       129033.86359099, 231580.89827089, 113442.8481895 , 186095.13102047,
       254509.51927373, 460905.89035936, 241649.5168027 , 238972.48765175,
        78847.58566556, 136312.86976053, 101347.67019061, 126499.64469735,
        55976.19712704, 235985.88959584, 150368.29261583, 157046.16602635,
```

```
100262.08452255, 122674.32253505, 270104.70868702, 205561.15534446,
349479.31871034, 130211.41053715, 225790.2897027 , 113349.52043331,
109256.00927618, 149758.1789992 , 223366.17421525, 269079.59634728,
138578.14700504, 144736.01425513, 143555.84045212, 151499.29032061,
116511.45125407, 198798.67514967, 201701.25593182, 263642.09930682,
 95554.01804736, 273429.60847593, 140134.79092214, 113335.45741658,
109978.72977585, 200159.73607467, 211181.62036971, 154509.35940729,
 94359.86193818, 255131.95261931, 182501.22896834, 209785.96409771,
132644.01722299, 323570.1876329 , 124658.32450833, 226220.40367958,
172512.88056425, 137960.52075167, 126361.73385898, 184374.51415384,
137606.2566004 , 148468.15183109, 278039.36457864, 129967.7606209 ,
116123.06744551, 193505.42455572, 195180.18109369, 136269.8003586 ,
 88458.74519736, 198663.24492865, 168781.1628044 , 168924.84300609,
106332.53967391, 383158.83622883, 162942.83680396, 177079.85445222,
226253.07543012, 127732.06499802, 365294.90277948, 104819.80929055,
184596.50695222, 136272.52219332, 146237.95824783, 246678.52719842,
105036.2027695 , 105595.03876311, 119012.24681451, 276865.35095489,
117455.25430846, 138858.54962033, 195464.17841702, 178037.93706227,
219167.94064325, 284676.01989444, 143641.64314526, 145229.82916578])
```

RMSE value

In [20]:

```
RMSE = np.sqrt(np.sum((y_test - y_pred) ** 2) / test_size)
RMSE
```

Out[20]:

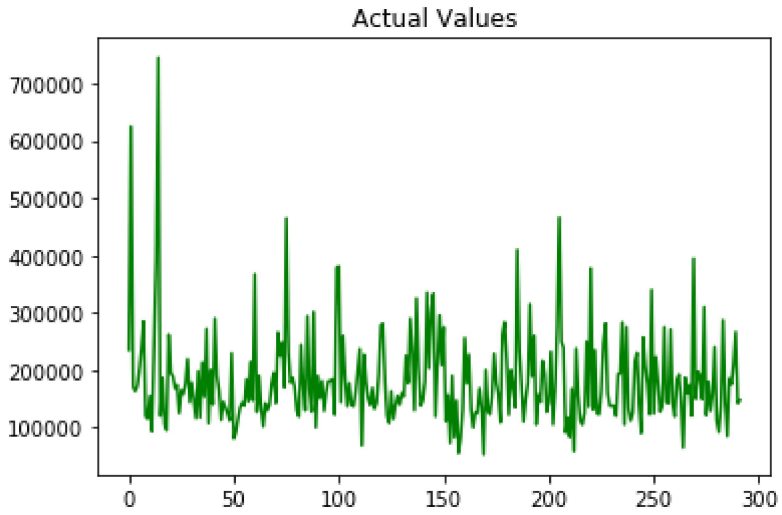
```
46437.27365761077
```

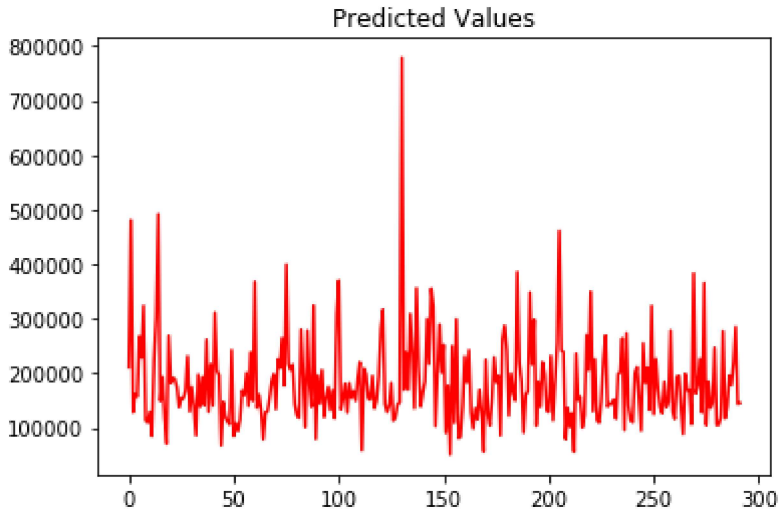
Plotting actual vs predicted values

In [27]:

```
plt.figure()
plt.title('Actual Values')
plt.plot(list(range(0, test_size)), y_test, color='green')

plt.figure()
plt.title('Predicted Values')
plt.plot(list(range(0, test_size)), y_pred, label = '', color='red')
plt.show()
```





In [ ]: