ANN - Multi-layer Perceptron Regressor (MLPRegressor)

1 Loading the libraries and data

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.neural_network import MLPRegressor

from sklearn import metrics

from sklearn.model_selection import GridSearchCV
df = pd.read_csv('house_prices.csv')
df = df.drop(['id', 'date', 'yr_built', 'yr_renovated', 'zipcode', 'lat',
'long'], axis=1)
df
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	sqft_living15	sqft_lot15
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1340	5650
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1690	7639
2	180000.0	2	1.00	770	10000	1.0	0	0	3	6	770	0	2720	8062
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1360	5000
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1800	7503
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21608	360000.0	3	2.50	1530	1131	3.0	0	0	3	8	1530	0	1530	1509
21609	400000.0	4	2.50	2310	5813	2.0	0	0	3	8	2310	0	1830	7200
21610	402101.0	2	0.75	1020	1350	2.0	0	0	3	7	1020	0	1020	2007
21611	400000.0	3	2.50	1600	2388	2.0	0	0	3	8	1600	0	1410	1287
21612	325000.0	2	0.75	1020	1076	2.0	0	0	3	7	1020	0	1020	1357

21613 rows × 14 columns

2 Data pre-processing

```
x = df.drop('price', axis=1)
y = df['price']

trainX, testX, trainY, testY = train_test_split(x, y, test_size = 0.2)
sc=StandardScaler()
```

```
scaler = sc.fit(trainX)
trainX_scaled = scaler.transform(trainX)
testX_scaled = scaler.transform(testX)
```

3 MLPRegressor

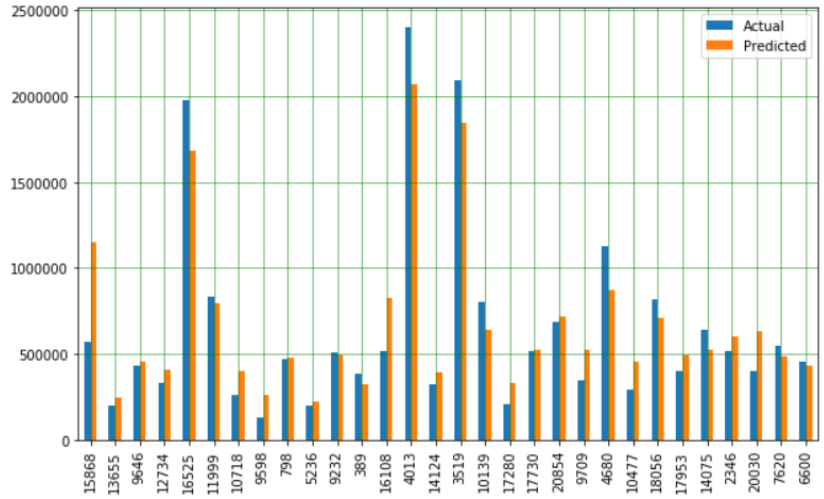
```
mlp_reg = MLPRegressor(hidden_layer_sizes=(150,100,50),
                       max_iter = 300,activation = 'relu',
                       solver = 'adam')

mlp_reg.fit(trainX_scaled, trainY)
```

4 Model Evaluation

```
y_pred = mlp_reg.predict(testX_scaled)
df_temp = pd.DataFrame({'Actual': testY, 'Predicted': y_pred})
df_temp.head()
```

	Actual	Predicted
15868	570000.0	1.153772e+06
13655	201000.0	2.453449e+05
9646	430000.0	4.544849e+05
12734	335000.0	4.060725e+05
16525	1980000.0	1.684505e+06

```
df_temp = df_temp.head(30)
df_temp.plot(kind='bar',figsize=(10,6))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```

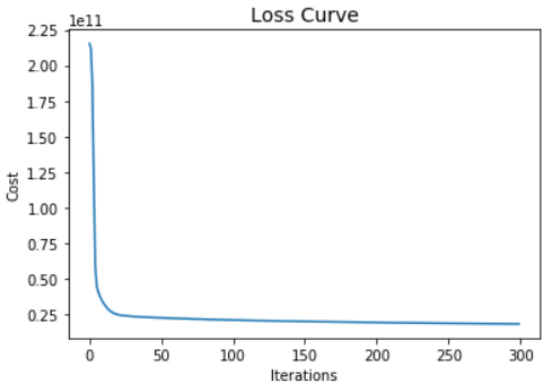


```
print('Mean Absolute Error:', metrics.mean_absolute_error(testY, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(testY, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(testY,
y_pred)))
```

```
Mean Absolute Error: 128896.31195324266
Mean Squared Error: 35889651799.369835
Root Mean Squared Error: 189445.64338978566
```

What these metrics mean and how to interpret them I have described in the following post: [Metrics for Regression Analysis]

```
plt.plot(mlp_reg.loss_curve_)
plt.title("Loss Curve", fontsize=14)
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.show()
```



6 Hyper Parameter Tuning

```
param_grid = {
    'hidden_layer_sizes': [(150,100,50), (120,80,40), (100,50,30)],
    'max_iter': [50, 100],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}
grid = GridSearchCV(mlp_reg, param_grid, n_jobs= -1, cv=5)
grid.fit(trainX_scaled, trainY)

print(grid.best_params_)
```

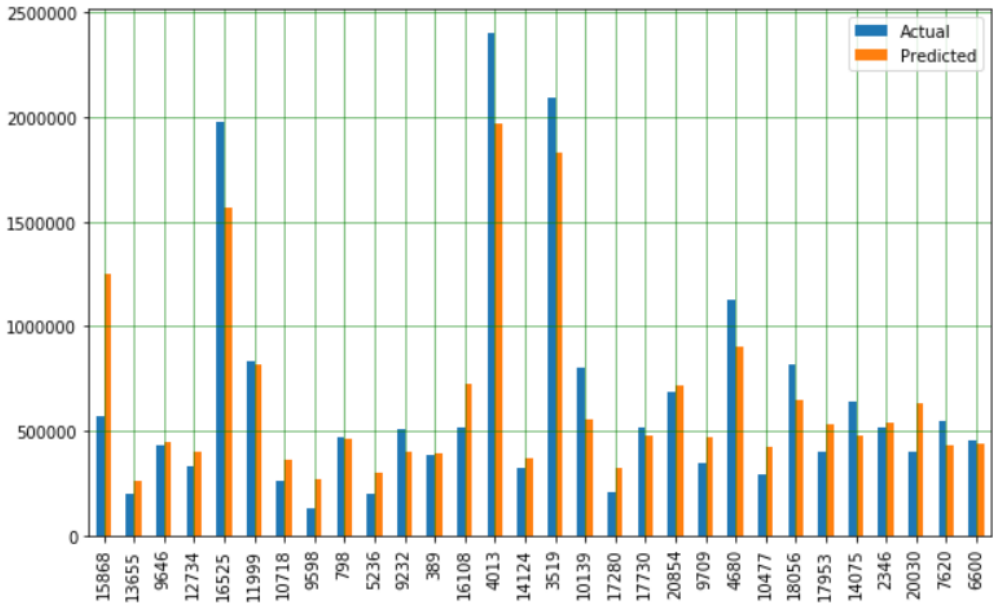
```
{'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (150, 100, 50), 'learning_rate': 'constant', 'max_iter':
100, 'solver': 'adam'}
```

```
grid_predictions = grid.predict(testX_scaled)
df_temp2 = pd.DataFrame({'Actual': testY, 'Predicted': grid_predictions})
```

```
df_temp2.head()
```

	Actual	Predicted
15868	570000.0	1.248729e+06
13655	201000.0	2.644010e+05
9646	430000.0	4.485995e+05
12734	335000.0	4.004703e+05
16525	1980000.0	1.566254e+06

```
df_temp2 = df_temp2.head(30)
df_temp2.plot(kind='bar',figsize=(10,6))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```




```
print('Mean Absolute Error:', metrics.mean_absolute_error(testY,
grid_predictions))
print('Mean Squared Error:', metrics.mean_squared_error(testY,
grid_predictions))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(testY,
grid_predictions)))
```

```
Mean Absolute Error: 137488.7695842236
Mean Squared Error: 39695070411.75198
Root Mean Squared Error: 199236.21762057213
```