Neural Network

December 10, 2017

In [13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [14]:
```
concrete= pd.read_csv('Concrete_Data.csv',names=['cement','slag','ash','water','superpl
```

In [15]:
```
concrete.head()
```

Out[15]:

	cement	slag	ash	water	superplastic	coarseagg	fineagg	age	strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.986111
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.887366
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.269535
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.052780
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.296075

In [16]:
```
concrete.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1030 entries, 0 to 1029
Data columns (total 9 columns):
cement          1030 non-null float64
slag            1030 non-null float64
ash             1030 non-null float64
water           1030 non-null float64
superplastic    1030 non-null float64
coarseagg       1030 non-null float64
fineagg         1030 non-null float64
age             1030 non-null int64
strength        1030 non-null float64
dtypes: float64(8), int64(1)
memory usage: 72.5 KB
```

In [17]:
```
concrete.describe()
```

Out[17]:

	cement	slag	ash	water	superplastic	\
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	

```
mean    281.165631    73.895485    54.187136   181.566359      6.203112
std     104.507142    86.279104    63.996469    21.355567      5.973492
min     102.000000     0.000000     0.000000   121.750000      0.000000
25%     192.375000     0.000000     0.000000   164.900000      0.000000
50%     272.900000    22.000000     0.000000   185.000000      6.350000
75%     350.000000   142.950000   118.270000   192.000000     10.160000
max     540.000000   359.400000   200.100000   247.000000     32.200000

         coarseagg      fineagg          age     strength
count  1030.000000  1030.000000  1030.000000  1030.000000
mean    972.918592   773.578883    45.662136    35.817836
std      77.753818    80.175427    63.169912    16.705679
min     801.000000   594.000000     1.000000     2.331808
25%     932.000000   730.950000     7.000000    23.707115
50%     968.000000   779.510000    28.000000    34.442774
75%    1029.400000   824.000000    56.000000    46.136287
max    1145.000000   992.600000   365.000000    82.599225
```

```
In [18]: from sklearn.preprocessing import MinMaxScaler
```

```
In [19]: min_max_scaler= MinMaxScaler()
```

```
In [23]: concrete = pd.DataFrame(min_max_scaler.fit_transform(concrete), columns=['cement','slag
```

```
In [24]: concrete.describe()
```

```
Out[24]:             cement         slag          ash        water  superplastic  \
count  1030.000000  1030.000000  1030.000000  1030.000000   1030.000000
mean      0.409054     0.205608     0.270800     0.477576      0.192643
std       0.238601     0.240064     0.319822     0.170504      0.185512
min       0.000000     0.000000     0.000000     0.000000      0.000000
25%       0.206336     0.000000     0.000000     0.344511      0.000000
50%       0.390183     0.061213     0.000000     0.504990      0.197205
75%       0.566210     0.397746     0.591054     0.560878      0.315528
max       1.000000     1.000000     1.000000     1.000000      1.000000

         coarseagg      fineagg          age     strength
count  1030.000000  1030.000000  1030.000000  1030.000000
mean      0.499763     0.450524     0.122698     0.417181
std       0.226029     0.201143     0.173544     0.208125
min       0.000000     0.000000     0.000000     0.000000
25%       0.380814     0.343578     0.016484     0.266301
50%       0.485465     0.465404     0.074176     0.400050
75%       0.663953     0.577020     0.151099     0.545732
max       1.000000     1.000000     1.000000     1.000000
```

```
In [25]: from sklearn.model_selection import train_test_split
```

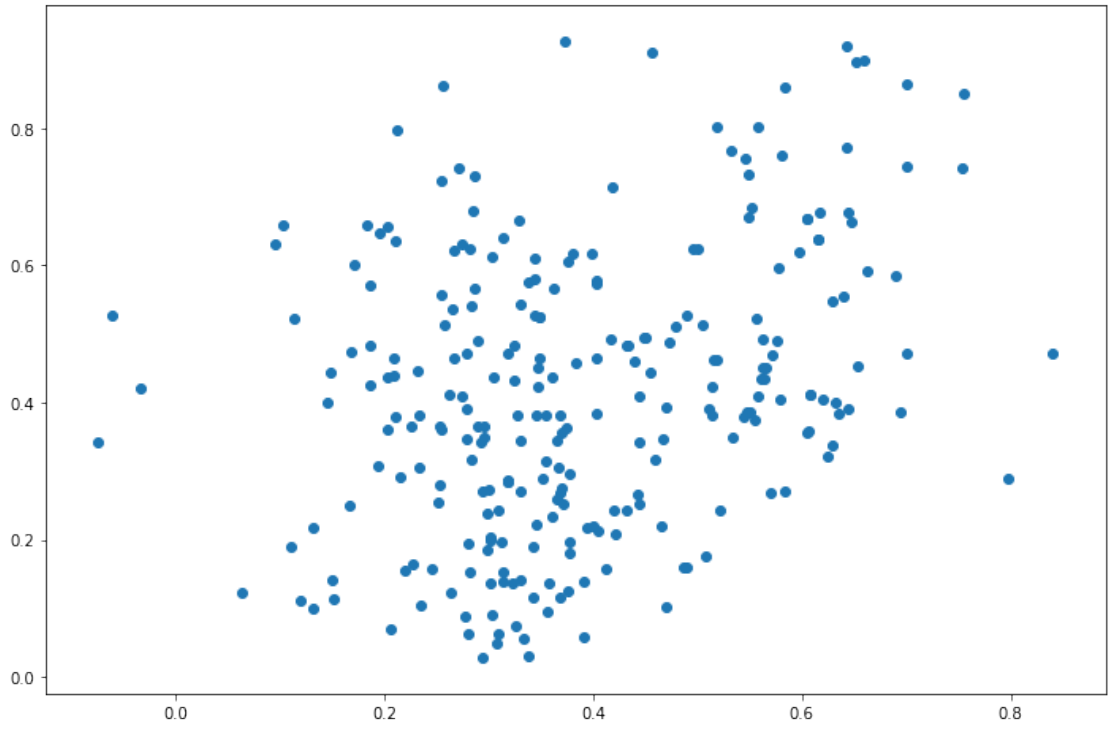
```
In [27]: X= concrete.drop('strength', axis=1)
```

```
In [28]: y= concrete['strength']

In [29]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=

In [30]: from sklearn.neural_network import MLPRegressor

In [33]: model= MLPRegressor(hidden_layer_sizes=(1,))

In [34]: model.fit( X_train, y_train)

Out[34]: MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                      beta_2=0.999, early_stopping=False, epsilon=1e-08,
                      hidden_layer_sizes=(1,), learning_rate='constant',
                      learning_rate_init=0.001, max_iter=200, momentum=0.9,
                      nesterovs_momentum=True, power_t=0.5, random_state=None,
                      shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
                      verbose=False, warm_start=False)

In [35]: model.n_layers_

Out[35]: 3

In [43]: predictions= model.predict(X_test)
         print(predictions)

[ 0.09487643  0.60430434  0.65887549  0.60819067  0.46931616  0.55590281
  0.44336207  0.26579353  0.30472732  0.34358884  0.31894529  0.2355509
  0.84036517  0.66202003  0.35398319  0.52139869  0.57187241  0.40425945
  0.56323938  0.40270848  0.37405703  0.1477784   0.40309658  0.39170168
  0.34717163  0.36909679  0.11935286  0.43170847  0.27471464  0.35844703
  0.26665202  0.40382687  0.39853002  0.21213203  0.41210805  0.51345331
  0.28880647  0.51890479  0.33016052  0.3810789   0.22746383  0.30797724
  0.23244578  0.57757454  0.35687614  0.58022643  0.64719342 -0.03396397
  0.30959563  0.33767215  0.55228501  0.28605334  0.62878821  0.37594267
  0.53234822  0.63172023  0.31832945  0.14979373  0.22604991  0.13147616
  0.51468587  0.39165243  0.26263277  0.63934873  0.3274321   0.31189341
  0.54958887  0.263058    0.35399833  0.37052604  0.3425285   0.29380368
  0.27797129  0.31901088  0.37788018  0.36764676  0.10261754  0.28622256
  0.75287599  0.28158378  0.11350024  0.62037902  0.28937828  0.34933754
  0.18603046  0.60819067  0.48964048  0.51066262  0.29878036  0.34618875
  0.2537455   0.45577637  0.24616264  0.64397202  0.61573361  0.1716944
  0.25533408  0.28292733  0.42073076  0.23322084  0.35177719  0.56293948
  0.34642018  0.32968775  0.75486602  0.13190361  0.28435237  0.46730471
  0.34466848  0.34801834 -0.06138437  0.37591532  0.53385651  0.20297598
  0.56611509  0.54924131  0.37798807  0.64485985  0.27094893  0.61528205
  0.39446024  0.19344373  0.30330091  0.33038059  0.42035953  0.25797396
  0.41809204  0.69049618  0.45081249  0.55523555  0.15217872  0.41672172
  0.0630482   0.7008333   0.32534707  0.27833067  0.2959732   0.2089922
  0.44348046  0.360958    0.27873302  0.60709149  0.36467059  0.20339314
```

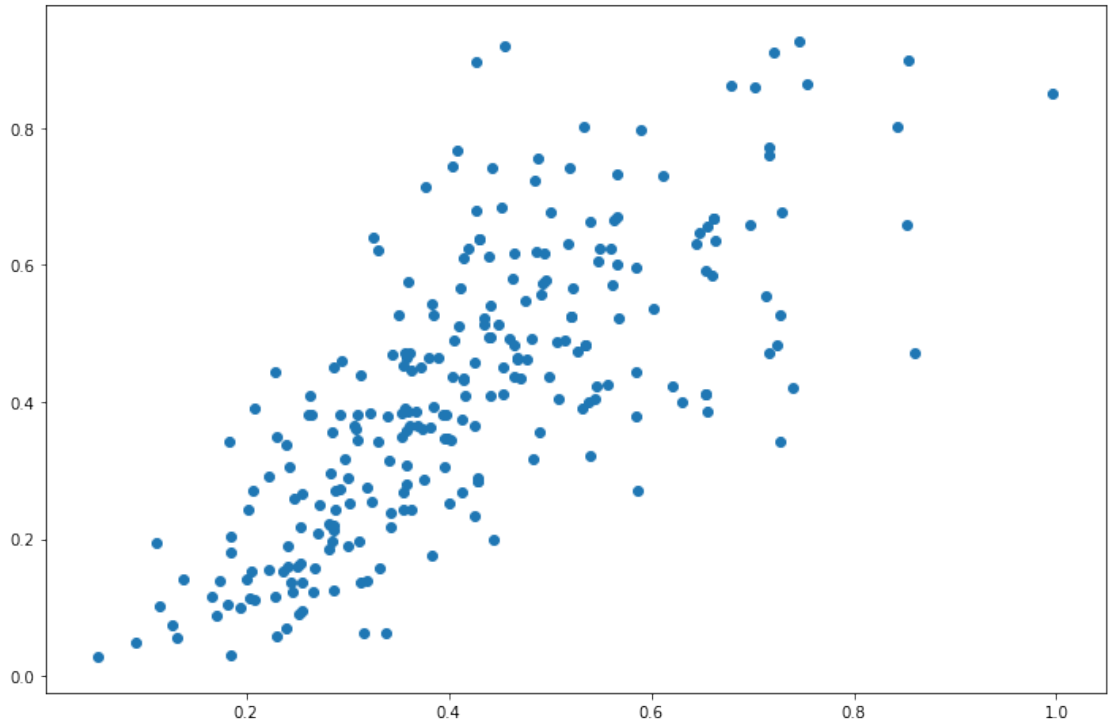
```
0.62853102  0.5049192   0.36673984  0.30162003  0.33404685  0.21929199
0.40311959  0.48726473  0.57547608  0.29233516  0.18595419  0.32505257
0.56995018  0.25680957  0.60430434  0.49624378  0.31308701  0.61741799
0.44390299  0.21039999  0.69438251  0.51850762  0.47256807  0.36554412
0.59684879  0.3600369   0.34364831  0.37743323  0.300287    0.65389835
0.51319487  0.20646137  0.29593876  0.32897213  0.5002632   0.18647717
0.6359189   0.23301238  0.34824912  0.20990453  0.28208587  0.19491775
0.16602866  0.27920504  0.58389989  0.29903188  0.57936241  0.20877918
0.16859349  0.700215   -0.07374108  0.36790835  0.2548816   0.48989074
0.46606519  0.21514769  0.36213722  0.54949041  0.30105754  0.45481377
0.65260752  0.27391947  0.45933348  0.70000835  0.36793784  0.54793239
0.28019756  0.28307119  0.30087672  0.33757636  0.26719407  0.29469249
0.32218599  0.32473771  0.371187    0.58390114  0.79695361  0.64300829
0.54571189  0.11000826  0.43323861  0.43220103  0.25406511  0.50795635
0.25214947  0.44350833  0.40083134  0.30325261  0.38325072  0.44917409
0.43917859  0.14577854  0.62426534  0.5628857   0.31317679  0.30946776
0.54353291  0.47882146  0.18403903  0.55824897  0.55773592  0.60556162
0.37320096  0.28061687  0.20265887  0.64288871  0.46934181  0.33050519
0.56024349  0.25325418  0.31901266  0.313096    0.34629137  0.34330219]
```

In [60]:
```
plt.figure(figsize=(12,8))
plt.scatter(predictions, y_test)
```

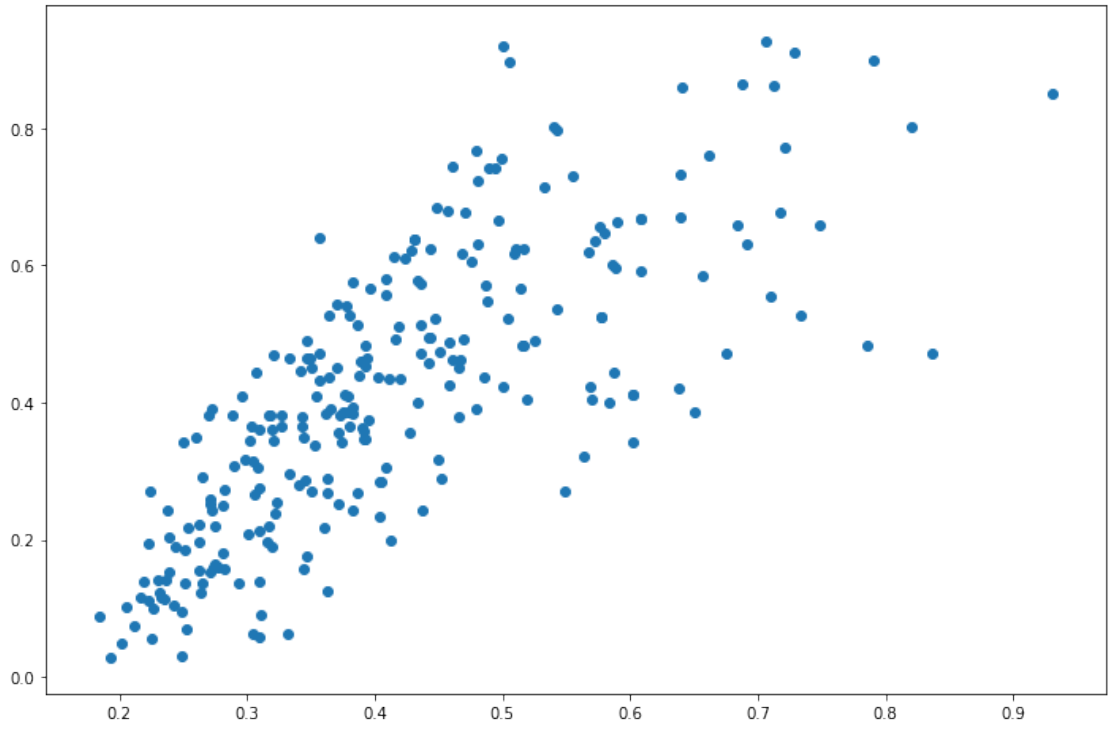
Out[60]: <matplotlib.collections.PathCollection at 0x7efe95d0a450>



```
In [62]: np.corrcoef(predictions, y_test)

Out[62]: array([[ 1.        ,  0.30763283],
                [ 0.30763283,  1.        ]])

In [67]: model= MLPRegressor(hidden_layer_sizes=(10,))

In [68]: model.fit(X_train, y_train)

Out[68]: MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                beta_2=0.999, early_stopping=False, epsilon=1e-08,
                hidden_layer_sizes=(10,), learning_rate='constant',
                learning_rate_init=0.001, max_iter=200, momentum=0.9,
                nesterovs_momentum=True, power_t=0.5, random_state=None,
                shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
                verbose=False, warm_start=False)

In [69]: predictions2 = model.predict(X_test)

In [70]: np.corrcoef(predictions2, y_test)

Out[70]: array([[ 1.        ,  0.7378899],
                [ 0.7378899,  1.       ]])

In [71]: plt.figure(figsize=(12,8))
         plt.scatter(predictions2, y_test)

Out[71]: <matplotlib.collections.PathCollection at 0x7efe95bd4290>
```



```
In [72]: model= MLPRegressor()

In [73]: model.fit(X_train, y_train)

Out[73]: MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
                      beta_2=0.999, early_stopping=False, epsilon=1e-08,
                      hidden_layer_sizes=(100,), learning_rate='constant',
                      learning_rate_init=0.001, max_iter=200, momentum=0.9,
                      nesterovs_momentum=True, power_t=0.5, random_state=None,
                      shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
                      verbose=False, warm_start=False)

In [74]: predictions3 = model.predict(X_test)

In [77]: plt.figure(figsize=(12,8))
         plt.scatter(predictions3, y_test)

Out[77]: <matplotlib.collections.PathCollection at 0x7efe95c4b0d0>
```




```
In [79]: np.corrcoef(predictions3, y_test)

Out[79]: array([[ 1.        ,  0.76247951],
                [ 0.76247951,  1.        ]])
```