Project

From A to D

Compressive strength of concrete samples

The dataset is about the compressive strength of different samples of concrete based on the volumes of the different ingredients that were used to make them.

A. Build a baseline model

In [2]:

```
import pandas as pd
import numpy as np

concrete_data = pd.read_csv('concrete_data.csv') #upload data from .csv file
concrete_data.head()
```

Out[2]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Strength
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30

In [3]:
```
concrete_data.isnull().sum() #check data
```

Out[3]:
```
Cement                0
Blast Furnace Slag    0
Fly Ash               0
Water                 0
Superplasticizer      0
Coarse Aggregate      0
Fine Aggregate        0
Age                   0
Strength              0
dtype: int64
```

Data is pretty good.

In [4]:
```
X = concrete_data[['Cement', 'Blast Furnace Slag', 'Fly Ash', 'Water', 'Superplasticizer', 'Coarse Aggregate', 'F
#concrete_data[concrete_data_columns[concrete_data_columns != 'Strength']] # all columns except Strength
X.head()
```

Out[4]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360

In [5]:
```
y = concrete_data[['Strength']]
y.head()
```

Out[5]:

	Strength
0	79.99
1	61.89
2	40.27
3	41.05
4	44.30

In [6]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4) #split for 30% verificat
print('Train set:', X_train.shape, y_train.shape)
print('Test set:', X_test.shape, y_test.shape)

n_cols = X.shape[1] # number of X - inputs
```

```
Train set: (721, 8) (721, 1)
Test set: (309, 8) (309, 1)
```

Following is to build of our Network, where we have 1 hidden layer of 50 nodes and ReLU activation.

In [7]:

```python
import keras
from keras.models import Sequential
from keras.layers import Dense

# define regression model
def regression_model():
    # create model
    model = Sequential()
    model.add(Dense(50, activation='relu', input_shape=(n_cols,)))
    model.add(Dense(1))

    # compile model
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
```

```
Using TensorFlow backend.
```

To create a loop of 50 times for generation of the mean and standard deviation of the mean squared errors list.

In [65]:

```
from sklearn.metrics import mean_squared_error
import math

errors = list()

for count in range(50):
    # build the model
    model = regression_model()

    # fit the model
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=50, verbose=0)

    pred = model.predict(X_test)

    # evaluate the model
    #scores = model.evaluate(X_test, y_test, verbose=2)

    error = round(mean_squared_error(y_test, pred))

    errors.append(error)
    print("Loop ",count," - error ",error)

print(errors)
```

```
Loop  0  - error  100.0
Loop  1  - error  74.0
Loop  2  - error  87.0
Loop  3  - error  101.0
Loop  4  - error  110.0
Loop  5  - error  79.0
Loop  6  - error  171.0
Loop  7  - error  118.0
Loop  8  - error  116.0
Loop  9  - error  67.0
Loop  10  - error  63.0
Loop  11  - error  53.0
Loop  12  - error  93.0
Loop  13  - error  105.0
Loop  14  - error  149.0
Loop  15  - error  84.0
Loop  16  - error  73.0
Loop  17  - error  71.0
Loop  18  - error  69.0
Loop  19  - error  74.0
Loop  20  - error  77.0
Loop  21  - error  86.0
Loop  22  - error  70.0
Loop  23  - error  78.0
Loop  24  - error  69.0
Loop  25  - error  89.0
Loop  26  - error  112.0
Loop  27  - error  69.0
Loop  28  - error  81.0
Loop  29  - error  61.0
Loop  30  - error  145.0
Loop  31  - error  75.0
Loop  32  - error  81.0
Loop  33  - error  109.0
Loop  34  - error  59.0
Loop  35  - error  74.0
Loop  36  - error  105.0
Loop  37  - error  110.0
Loop  38  - error  74.0
Loop  39  - error  62.0
Loop  40  - error  74.0
Loop  41  - error  89.0
Loop  42  - error  68.0
Loop  43  - error  64.0
Loop  44  - error  78.0
Loop  45  - error  104.0
Loop  46  - error  80.0
Loop  47  - error  138.0
Loop  48  - error  78.0
Loop  49  - error  102.0
[100.0, 74.0, 87.0, 101.0, 110.0, 79.0, 171.0, 118.0, 116.0, 67.0, 63.0, 53.0, 93.0, 105.0, 149.0, 84.0, 73.0,
71.0, 69.0, 74.0, 77.0, 86.0, 70.0, 78.0, 69.0, 89.0, 112.0, 69.0, 81.0, 61.0, 145.0, 75.0, 81.0, 109.0, 59.0,
74.0, 105.0, 110.0, 74.0, 62.0, 74.0, 89.0, 68.0, 64.0, 78.0, 104.0, 80.0, 138.0, 78.0, 102.0]
```

In [66]:

```
errors = np.array(errors)
# defining of mean and std
mean = errors.mean()
std = errors.std()

print("Finale results: mean = ", mean, "; std = ", std)
```

```
Finale results: mean =  88.36 ; std =  24.762681599536023
```

In [ ]:

B. Normalize the data

In [68]:

```
#to normalize data set by subtracting the mean from the individual predictors and dividing by the standard deviat
X_norm = (X - X.mean()) / X.std()
X_norm.head()
```

Out[68]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age
0	2.476712	-0.856472	-0.846733	-0.916319	-0.620147	0.862735	-1.217079	-0.279597
1	2.476712	-0.856472	-0.846733	-0.916319	-0.620147	1.055651	-1.217079	-0.279597
2	0.491187	0.795140	-0.846733	2.174405	-1.038638	-0.526262	-2.239829	3.551340
3	0.491187	0.795140	-0.846733	2.174405	-1.038638	-0.526262	-2.239829	5.055221
4	-0.790075	0.678079	-0.846733	0.488555	-1.038638	0.070492	0.647569	4.976069

In [69]:

```
#and repeat steps
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=4) #split for 30% verificat
print('Train set:', X_train.shape, y_train.shape)
print('Test set:', X_test.shape, y_test.shape)

n_cols = X.shape[1] # number of X - inputs
```

```
Train set: (721, 8) (721, 1)
Test set: (309, 8) (309, 1)
```

In [70]:

```
errors = list()

for count in range(50):
    # build the model
    model = regression_model()

    # fit the model
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=50, verbose=0)

    pred = model.predict(X_test)

    # evaluate the model
    #scores = model.evaluate(X_test, y_test, verbose=2)

    error = round(mean_squared_error(y_test, pred))

    errors.append(error)
    print("Loop ",count," - error ",error)

#print(errors)
```

```
Loop  0  - error  98.0
Loop  1  - error  82.0
Loop  2  - error  107.0
Loop  3  - error  103.0
Loop  4  - error  62.0
Loop  5  - error  158.0
Loop  6  - error  107.0
Loop  7  - error  84.0
Loop  8  - error  82.0
Loop  9  - error  76.0
Loop  10  - error  101.0
Loop  11  - error  71.0
Loop  12  - error  95.0
Loop  13  - error  79.0
Loop  14  - error  78.0
Loop  15  - error  74.0
Loop  16  - error  86.0
Loop  17  - error  70.0
Loop  18  - error  63.0
Loop  19  - error  89.0
Loop  20  - error  70.0
Loop  21  - error  67.0
Loop  22  - error  86.0
Loop  23  - error  65.0
Loop  24  - error  69.0
Loop  25  - error  121.0
Loop  26  - error  116.0
Loop  27  - error  84.0
Loop  28  - error  74.0
Loop  29  - error  84.0
Loop  30  - error  100.0
Loop  31  - error  82.0
Loop  32  - error  138.0
Loop  33  - error  85.0
Loop  34  - error  76.0
Loop  35  - error  111.0
Loop  36  - error  95.0
Loop  37  - error  147.0
Loop  38  - error  113.0
Loop  39  - error  89.0
Loop  40  - error  119.0
Loop  41  - error  115.0
Loop  42  - error  85.0
Loop  43  - error  81.0
Loop  44  - error  124.0
Loop  45  - error  154.0
Loop  46  - error  93.0
Loop  47  - error  152.0
Loop  48  - error  96.0
Loop  49  - error  83.0
```

In [71]:

```
errors_normalize = np.array(errors)
# defining of mean and std
mean_normalize = errors_normalize.mean()
std_normalize = errors_normalize.std()

print("Finale results: mean = ", mean_normalize, "; std = ", std_normalize)
```

```
Finale results: mean =  94.78 ; std =  24.20023966823469
```

C. Increase the number of epochs

New epochs will be increased up to 100.

In [72]:

```
errors_epochs100 = list()

for count in range(50):
    # build the model
    model = regression_model()

    # fit the model
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100, verbose=0)

    pred = model.predict(X_test)

    # evaluate the model
    #scores = model.evaluate(X_test, y_test, verbose=2)

    error = round(mean_squared_error(y_test, pred))

    errors_epochs100.append(error)
    print("Loop ",count," - error ",error)
```

```
Loop  0  - error  95.0
Loop  1  - error  75.0
Loop  2  - error  85.0
Loop  3  - error  79.0
Loop  4  - error  56.0
Loop  5  - error  50.0
Loop  6  - error  60.0
Loop  7  - error  52.0
Loop  8  - error  63.0
Loop  9  - error  66.0
Loop  10  - error  72.0
Loop  11  - error  72.0
Loop  12  - error  68.0
Loop  13  - error  105.0
Loop  14  - error  81.0
Loop  15  - error  74.0
Loop  16  - error  80.0
Loop  17  - error  63.0
Loop  18  - error  96.0
Loop  19  - error  54.0
Loop  20  - error  70.0
Loop  21  - error  100.0
Loop  22  - error  63.0
Loop  23  - error  82.0
Loop  24  - error  72.0
Loop  25  - error  81.0
Loop  26  - error  73.0
Loop  27  - error  64.0
Loop  28  - error  90.0
Loop  29  - error  59.0
Loop  30  - error  95.0
Loop  31  - error  48.0
Loop  32  - error  69.0
Loop  33  - error  51.0
Loop  34  - error  59.0
Loop  35  - error  71.0
Loop  36  - error  58.0
Loop  37  - error  53.0
Loop  38  - error  74.0
Loop  39  - error  98.0
Loop  40  - error  68.0
Loop  41  - error  53.0
Loop  42  - error  72.0
Loop  43  - error  60.0
Loop  44  - error  93.0
Loop  45  - error  56.0
Loop  46  - error  56.0
Loop  47  - error  64.0
Loop  48  - error  57.0
Loop  49  - error  72.0
```

In [73]:

```python
errors_epochs100 = np.array(errors_epochs100)
# defining of mean and std
mean_epochs100 = errors_epochs100.mean()
std_epochs100 = errors_epochs100.std()

print("Finale results: mean = ", mean_epochs100, "; std = ", std_epochs100)
```

```
Finale results: mean =  70.54 ; std =  14.572865195286752
```

In [ ]:

```
```

D. Increase the number of hidden layers

Let's add one more hidden layer with the same amount of nodes.

```
In [75]: # define a new regression model
         def new_regression_model():
             # create model
             model = Sequential()
             model.add(Dense(50, activation='relu', input_shape=(n_cols,)))
             model.add(Dense(50, activation='relu'))
             model.add(Dense(1))
             
             # compile model
             model.compile(optimizer='adam', loss='mean_squared_error')
             return model
```

In [76]:

```
errors_2hidden = list()

for count in range(50):
    # build the model
    model = new_regression_model()

    # fit the model
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100, verbose=0)

    pred = model.predict(X_test)

    # evaluate the model
    #scores = model.evaluate(X_test, y_test, verbose=2)

    error = round(mean_squared_error(y_test, pred))

    errors_2hidden.append(error)
    print("Loop ",count," - error ",error)
```

```
Loop  0  - error  47.0
Loop  1  - error  41.0
Loop  2  - error  50.0
Loop  3  - error  62.0
Loop  4  - error  105.0
Loop  5  - error  79.0
Loop  6  - error  46.0
Loop  7  - error  67.0
Loop  8  - error  63.0
Loop  9  - error  47.0
Loop  10  - error  53.0
Loop  11  - error  56.0
Loop  12  - error  50.0
Loop  13  - error  90.0
Loop  14  - error  40.0
Loop  15  - error  46.0
Loop  16  - error  43.0
Loop  17  - error  42.0
Loop  18  - error  41.0
Loop  19  - error  47.0
Loop  20  - error  46.0
Loop  21  - error  94.0
Loop  22  - error  45.0
Loop  23  - error  51.0
Loop  24  - error  49.0
Loop  25  - error  46.0
Loop  26  - error  48.0
Loop  27  - error  49.0
Loop  28  - error  42.0
Loop  29  - error  67.0
Loop  30  - error  57.0
Loop  31  - error  61.0
Loop  32  - error  66.0
Loop  33  - error  47.0
Loop  34  - error  59.0
Loop  35  - error  96.0
Loop  36  - error  43.0
Loop  37  - error  56.0
Loop  38  - error  42.0
Loop  39  - error  51.0
Loop  40  - error  45.0
Loop  41  - error  49.0
Loop  42  - error  51.0
Loop  43  - error  41.0
Loop  44  - error  44.0
Loop  45  - error  50.0
Loop  46  - error  43.0
Loop  47  - error  38.0
Loop  48  - error  50.0
Loop  49  - error  56.0
```

In [77]:

```
errors_2hidden = np.array(errors_2hidden)
# defining of mean and std
mean_2hidden = errors_2hidden.mean()
std_2hidden = errors_2hidden.std()

print("Finale results: mean = ", mean_2hidden, "; std = ", std_2hidden)
```

```
Finale results: mean =  53.94 ; std =  15.019201043996983
```

In [ ]:

Compare results

```
In [78]: results_raw = {'Parameter': ['mean', 'std'], 'Simple': [mean, std], 'Normalize': [mean_normalize, std_normalize],
              [mean_epochs100, std_epochs100], '2 hidden layers': [mean_2hidden, std_2hidden]}
         results = pd.DataFrame(data=results_raw)
```

```
In [79]: results
```

Out[79]:

	Parameter	Simple	Normalize	100 epochs	2 hidden layers
0	mean	88.360000	94.78000	70.540000	53.940000
1	std	24.762682	24.20024	14.572865	15.019201

Conclusion. With higher complexity of Neural Network we can achieve higher accuracy.

```
In [ ]:
```