In [1]:

```python
# importing libraries

import pandas as pd
import numpy as np
import os
```

In [2]:

```python
# Reading Data
os.chdir(r'C:\Users\HP\Downloads\concrete_data')
concrete_data = pd.read_csv('concrete_data.csv')
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]:

```python
# Check Data Shape
concrete_data.shape
```

Out[3]:

```
(1030, 9)
```

In [4]:

```python
concrete_data.isnull().sum()
```

Out[4]:

```
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
```

The data looks very clean and is ready to be used to build our model.

In [6]:

```python
# Split data into predictors and target

concrete_data_columns = concrete_data.columns
predictors = concrete_data[concrete_data_columns[concrete_data_columns != 'Strength']] #
target = concrete_data['Strength'] # Strength column
```

In [7]:

```python
predictors
```

Out[7]:

	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

...	...	...	...	...	...	...	...	...
1025	276.4	116.0	90.3	179.6	8.9	870.1	768.3	28
1026	322.2	0.0	115.6	196.0	10.4	817.9	813.4	28
1027	148.5	139.4	108.6	192.7	6.1	892.4	780.0	28
1028	159.1	186.7	0.0	175.6	11.3	989.6	788.9	28
1029	260.9	100.5	78.3	200.6	8.6	864.5	761.5	28

1030 rows × 8 columns

In [8]:

```
target
```

Out[8]:

```
0       79.99
1       61.89
2       40.27
3       41.05
4       44.30
        ...
1025    44.28
1026    31.18
1027    23.70
1028    32.77
1029    32.40
Name: Strength, Length: 1030, dtype: float64
```

In [9]:

```
n_cols = predictors.shape[1] # number of predictors
n_cols
```

Out[9]:

```
8
```

In [10]:

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

In [11]:

```
# define regression model
def regression_model():
    # create model
    model = Sequential()
    model.add(Dense(10, activation='relu', input_shape=(n_cols,)))
    model.add(Dense(1))

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

In [12]:

```
from sklearn.model_selection import train_test_split
```

In [13]:

```
X_train, X_test, y_train, y_test = train_test_split(predictors, target, test_size=0.3, r
```

In [14]:

```
# build the model
model = regression_model()
# fit the model
epochs = 50
model.fit(X_train, y_train, epochs=epochs, verbose=1)
```

```
Epoch 1/50
23/23 [==============================] - 1s 4ms/step - loss: 13516.0801
Epoch 2/50
23/23 [==============================] - 0s 4ms/step - loss: 2305.9395
Epoch 3/50
```

```
23/23 [==============================] - 0s 4ms/step - loss: 1091.0249
Epoch 4/50
23/23 [==============================] - 0s 4ms/step - loss: 990.7644
Epoch 5/50
23/23 [==============================] - 0s 4ms/step - loss: 871.6293
Epoch 6/50
23/23 [==============================] - 0s 4ms/step - loss: 768.0619
Epoch 7/50
23/23 [==============================] - 0s 4ms/step - loss: 674.6142
Epoch 8/50
23/23 [==============================] - 0s 4ms/step - loss: 588.1871
Epoch 9/50
23/23 [==============================] - 0s 5ms/step - loss: 512.5783
Epoch 10/50
23/23 [==============================] - 0s 5ms/step - loss: 451.2828
Epoch 11/50
23/23 [==============================] - 0s 5ms/step - loss: 397.7354
Epoch 12/50
23/23 [==============================] - 0s 4ms/step - loss: 356.5443
Epoch 13/50
23/23 [==============================] - 0s 6ms/step - loss: 323.7957
Epoch 14/50
23/23 [==============================] - 0s 5ms/step - loss: 298.7578
Epoch 15/50
23/23 [==============================] - 0s 5ms/step - loss: 276.0735
Epoch 16/50
23/23 [==============================] - 0s 6ms/step - loss: 259.5686
Epoch 17/50
23/23 [==============================] - 0s 6ms/step - loss: 246.0152
Epoch 18/50
23/23 [==============================] - 0s 6ms/step - loss: 234.6922
Epoch 19/50
23/23 [==============================] - 0s 5ms/step - loss: 226.0287
Epoch 20/50
23/23 [==============================] - 0s 5ms/step - loss: 217.0321
Epoch 21/50
23/23 [==============================] - 0s 6ms/step - loss: 210.8709
Epoch 22/50
23/23 [==============================] - 0s 6ms/step - loss: 204.6392
Epoch 23/50
23/23 [==============================] - 0s 6ms/step - loss: 198.3201
Epoch 24/50
23/23 [==============================] - 0s 6ms/step - loss: 194.4356
Epoch 25/50
23/23 [==============================] - 0s 6ms/step - loss: 189.6375
Epoch 26/50
23/23 [==============================] - 0s 6ms/step - loss: 185.0568
Epoch 27/50
23/23 [==============================] - 0s 5ms/step - loss: 180.0086
Epoch 28/50
23/23 [==============================] - 0s 4ms/step - loss: 174.9545
Epoch 29/50
23/23 [==============================] - 0s 5ms/step - loss: 171.2162
Epoch 30/50
23/23 [==============================] - 0s 5ms/step - loss: 168.1709
Epoch 31/50
23/23 [==============================] - 0s 4ms/step - loss: 163.2359
Epoch 32/50
23/23 [==============================] - 0s 4ms/step - loss: 159.5160
Epoch 33/50
23/23 [==============================] - 0s 5ms/step - loss: 153.5508
Epoch 34/50
23/23 [==============================] - 0s 6ms/step - loss: 147.8576
Epoch 35/50
23/23 [==============================] - 0s 5ms/step - loss: 143.7776
Epoch 36/50
```

```
23/23 [==============================] - 0s 4ms/step - loss: 138.6494
Epoch 37/50
23/23 [==============================] - 0s 4ms/step - loss: 136.4717
Epoch 38/50
23/23 [==============================] - 0s 5ms/step - loss: 133.2581
Epoch 39/50
23/23 [==============================] - 0s 5ms/step - loss: 131.5740
Epoch 40/50
23/23 [==============================] - 0s 4ms/step - loss: 130.7674
Epoch 41/50
23/23 [==============================] - 0s 5ms/step - loss: 128.7976
Epoch 42/50
23/23 [==============================] - 0s 5ms/step - loss: 126.9858
Epoch 43/50
23/23 [==============================] - 0s 5ms/step - loss: 125.9078
Epoch 44/50
23/23 [==============================] - 0s 5ms/step - loss: 124.7784
Epoch 45/50
23/23 [==============================] - 0s 4ms/step - loss: 126.3585
Epoch 46/50
23/23 [==============================] - 0s 4ms/step - loss: 123.6073
Epoch 47/50
23/23 [==============================] - 0s 5ms/step - loss: 122.1477
Epoch 48/50
23/23 [==============================] - 0s 5ms/step - loss: 120.8856
Epoch 49/50
23/23 [==============================] - 0s 4ms/step - loss: 119.3726
Epoch 50/50
23/23 [==============================] - 0s 4ms/step - loss: 118.7761
```

Out[14]:

```
<keras.callbacks.History at 0x1b656700ee0>
```

In [16]:

```python
# evaluate the model on the test data
loss_val = model.evaluate(X_test, y_test)
y_pred = model.predict(X_test)
loss_val
```

```
10/10 [==============================] - 0s 2ms/step - loss: 119.2411
10/10 [==============================] - 0s 4ms/step
```

Out[16]:

```
119.24113464355469
```

In [17]:

```python
# Compute the mean squared error between the predicted concrete strength and the actual
from sklearn.metrics import mean_squared_error
mean_square_error = mean_squared_error(y_test, y_pred)
mean = np.mean(mean_square_error)
standard_deviation = np.std(mean_square_error)
print(mean, standard_deviation)
```

```
119.24114890146001 0.0
```

In [18]:

```python
total_mean_squared_errors = 50
epochs = 50
mean_squared_errors = []
for i in range(0, total_mean_squared_errors):
    X_train, X_test, y_train, y_test = train_test_split(predictors, target, test_size=0.
    model.fit(X_train, y_train, epochs=epochs, verbose=0)
    MSE = model.evaluate(X_test, y_test, verbose=0)
    print("MSE "+str(i+1)+": "+str(MSE))
    y_pred = model.predict(X_test)
    mean_square_error = mean_squared_error(y_test, y_pred)
    mean_squared_errors.append(mean_square_error)

mean_squared_errors = np.array(mean_squared_errors)
mean = np.mean(mean_squared_errors)
standard_deviation = np.std(mean_squared_errors)
```

```
print('\n')
print("Below is the mean and standard deviation of " +str(total_mean_squared_errors) + "
print("Mean: "+str(mean))
print("Standard Deviation: "+str(standard_deviation))
```

```
MSE 1: 84.23403930664062
10/10 [==============================] - 0s 2ms/step
MSE 2: 116.73729705810547
10/10 [==============================] - 0s 2ms/step
MSE 3: 84.02308654785156
10/10 [==============================] - 0s 2ms/step
MSE 4: 88.3523941040039
10/10 [==============================] - 0s 1ms/step
MSE 5: 74.02823638916016
10/10 [==============================] - 0s 2ms/step
MSE 6: 65.25355529785156
10/10 [==============================] - 0s 2ms/step
MSE 7: 74.57231140136719
10/10 [==============================] - 0s 2ms/step
MSE 8: 51.271366119384766
10/10 [==============================] - 0s 2ms/step
MSE 9: 56.257591247558594
10/10 [==============================] - 0s 3ms/step
MSE 10: 58.79990768432617
10/10 [==============================] - 0s 2ms/step
MSE 11: 47.94743728637695
10/10 [==============================] - 0s 3ms/step
MSE 12: 42.33832550048828
10/10 [==============================] - 0s 2ms/step
MSE 13: 54.75746536254883
10/10 [==============================] - 0s 2ms/step
MSE 14: 50.08696746826172
10/10 [==============================] - 0s 2ms/step
MSE 15: 46.21696853637695
10/10 [==============================] - 0s 2ms/step
MSE 16: 44.98457717895508
10/10 [==============================] - 0s 2ms/step
MSE 17: 45.966861724853516
10/10 [==============================] - 0s 4ms/step
MSE 18: 45.58372497558594
10/10 [==============================] - 0s 3ms/step
MSE 19: 40.59528350830078
10/10 [==============================] - 0s 2ms/step
MSE 20: 45.994476318359375
10/10 [==============================] - 0s 2ms/step
MSE 21: 44.252159118652344
10/10 [==============================] - 0s 2ms/step
MSE 22: 46.04426193237305
10/10 [==============================] - 0s 1ms/step
MSE 23: 42.76171875
10/10 [==============================] - 0s 2ms/step
MSE 24: 50.80432891845703
10/10 [==============================] - 0s 2ms/step
MSE 25: 46.9375
10/10 [==============================] - 0s 3ms/step
MSE 26: 43.50351333618164
10/10 [==============================] - 0s 2ms/step
MSE 27: 48.35236740112305
10/10 [==============================] - 0s 2ms/step
MSE 28: 42.31876754760742
10/10 [==============================] - 0s 2ms/step
MSE 29: 47.57341766357422
10/10 [==============================] - 0s 2ms/step
MSE 30: 43.902713775634766
10/10 [==============================] - 0s 2ms/step
```

```
MSE 31: 48.07430648803711
10/10 [==============================] - 0s 2ms/step
MSE 32: 40.472206115722656
10/10 [==============================] - 0s 2ms/step
MSE 33: 45.401710510253906
10/10 [==============================] - 0s 2ms/step
MSE 34: 50.55637741088867
10/10 [==============================] - 0s 2ms/step
MSE 35: 43.2892951965332
10/10 [==============================] - 0s 2ms/step
MSE 36: 47.703182220458984
10/10 [==============================] - 0s 2ms/step
MSE 37: 50.787086486816406
10/10 [==============================] - 0s 2ms/step
MSE 38: 46.21717834472656
10/10 [==============================] - 0s 2ms/step
MSE 39: 44.75178146362305
10/10 [==============================] - 0s 2ms/step
MSE 40: 40.03106689453125
10/10 [==============================] - 0s 2ms/step
MSE 41: 47.22334289550781
10/10 [==============================] - 0s 2ms/step
MSE 42: 41.89603042602539
10/10 [==============================] - 0s 2ms/step
MSE 43: 57.00648498535156
10/10 [==============================] - 0s 2ms/step
MSE 44: 50.76432418823242
10/10 [==============================] - 0s 1ms/step
MSE 45: 47.596595764160156
10/10 [==============================] - 0s 2ms/step
MSE 46: 51.230674743652344
10/10 [==============================] - 0s 2ms/step
MSE 47: 46.80618667602539
10/10 [==============================] - 0s 2ms/step
MSE 48: 44.227996826171875
10/10 [==============================] - 0s 2ms/step
MSE 49: 46.70132827758789
10/10 [==============================] - 0s 2ms/step
MSE 50: 48.52796936035156
10/10 [==============================] - 0s 2ms/step


Below is the mean and standard deviation of 50 mean squared errors without normalized da
ta. Total number of epochs for each training is: 50

Mean: 52.274354963573316
Standard Deviation: 14.523158012564373
```

In [ ]: