Regression with Keras

Regression is a type of supervised machine learning algorithm used to predict a continuous label. The goal is to produce a model that represents the ‘Strength' to some observed data, according to an evaluation criterion.

importing libraries

In [4]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
```

importing model from keras sequential

In [5]:
```
model=Sequential()
```

import pandas for loading data

In [6]:
```
import pandas as pd
```

reading the data

In [7]:
```
data=pd.read_csv("concrete_data.csv")
```

In [8]:
```
data.head()
```

Out[8]:

	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 [9]:
```
target_column = ['Strength']
predictors = list(set(list(data.columns))-set(target_column))
data[predictors] = data[predictors]/data[predictors].max()
data.describe()
```

Out[9]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Strength
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	0.520681	0.205609	0.270806	0.735090	0.192691	0.849711	0.779348	0.125102	35.817961
std	0.193530	0.240065	0.319825	0.086454	0.185523	0.067907	0.080774	0.173068	16.705742
min	0.188889	0.000000	0.000000	0.493117	0.000000	0.699563	0.598428	0.002740	2.330000
25%	0.356250	0.000000	0.000000	0.667611	0.000000	0.813974	0.736399	0.019178	23.710000
50%	0.505370	0.061213	0.000000	0.748988	0.198758	0.845415	0.785311	0.076712	34.445000
75%	0.648148	0.397746	0.591204	0.777328	0.316770	0.899039	0.830143	0.153425	46.135000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	82.600000

In [10]:
```
from sklearn.model_selection import train_test_split
X = data[predictors].values
y = data[target_column].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=40)
print(X_train.shape);
print(X_test.shape)
```

```
(721, 8)
(309, 8)
```

In [12]:
```
model.add(Dense(500, input_dim=8, activation= "relu"))
model.add(Dense(100, activation= "relu"))
model.add(Dense(50, activation= "relu"))
model.add(Dense(1))
```

In [13]:
```
model.compile(loss= "mean_squared_error" , optimizer="adam", metrics=["mean_squared_error"])
```

In [14]:
```
model.fit(X_train, y_train, epochs=20)
```

```
Epoch 1/20
23/23 [==============================] - 2s 6ms/step - loss: 1417.5845 - mean_squared_error: 1417.5845
Epoch 2/20
23/23 [==============================] - 0s 5ms/step - loss: 599.2491 - mean_squared_error: 599.2491
Epoch 3/20
23/23 [==============================] - 0s 5ms/step - loss: 265.8033 - mean_squared_error: 265.8033
Epoch 4/20
23/23 [==============================] - 0s 5ms/step - loss: 219.9748 - mean_squared_error: 219.9748
Epoch 5/20
23/23 [==============================] - 0s 5ms/step - loss: 203.1978 - mean_squared_error: 203.1978
Epoch 6/20
23/23 [==============================] - 0s 5ms/step - loss: 184.2809 - mean_squared_error: 184.2809
Epoch 7/20
23/23 [==============================] - 0s 5ms/step - loss: 167.0523 - mean_squared_error: 167.0523
Epoch 8/20
23/23 [==============================] - 0s 5ms/step - loss: 151.7108 - mean_squared_error: 151.7108
Epoch 9/20
23/23 [==============================] - 0s 5ms/step - loss: 139.8390 - mean_squared_error: 139.8390
Epoch 10/20
23/23 [==============================] - 0s 5ms/step - loss: 131.9453 - mean_squared_error: 131.9453
Epoch 11/20
23/23 [==============================] - 0s 5ms/step - loss: 127.3335 - mean_squared_error: 127.3335
Epoch 12/20
23/23 [==============================] - 0s 5ms/step - loss: 125.3999 - mean_squared_error: 125.3999
Epoch 13/20
23/23 [==============================] - 0s 5ms/step - loss: 121.5724 - mean_squared_error: 121.5724
Epoch 14/20
23/23 [==============================] - 0s 4ms/step - loss: 120.9132 - mean_squared_error: 120.9132
Epoch 15/20
23/23 [==============================] - 0s 5ms/step - loss: 116.9368 - mean_squared_error: 116.9368
Epoch 16/20
23/23 [==============================] - 0s 5ms/step - loss: 116.1326 - mean_squared_error: 116.1326
Epoch 17/20
23/23 [==============================] - 0s 5ms/step - loss: 116.0513 - mean_squared_error: 116.0513
Epoch 18/20
23/23 [==============================] - 0s 5ms/step - loss: 113.3102 - mean_squared_error: 113.3102
Epoch 19/20
23/23 [==============================] - 0s 5ms/step - loss: 110.6741 - mean_squared_error: 110.6741
Epoch 20/20
23/23 [==============================] - 0s 5ms/step - loss: 110.1587 - mean_squared_error: 110.1587
```

Out[14]:
```
<keras.src.callbacks.History at 0x1f3d783d010>
```

In [17]:
```
import numpy as np
from sklearn.metrics import mean_squared_error
from math import sqrt
pred_train= model.predict(X_train)
print(np.sqrt(mean_squared_error(y_train,pred_train)))

pred= model.predict(X_test)
print(np.sqrt(mean_squared_error(y_test,pred)))
```

```
23/23 [==============================] - 0s 3ms/step
10.409702231466895
10/10 [==============================] - 0s 4ms/step
11.177184380700046
```

In [ ]:
```

```