In [40]:

```
import keras
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn import preprocessing
```

In [41]:

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

Part A. Build Model ¶

In [42]:

```
def regression_model():
    # create model
    model = Sequential()
    model.add(Dense(10, activation='relu', input_shape=(len(concrete_data.columns)-1,)))
    model.add(Dense(1))

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

In [43]:

```
concrete_data = pd.read_csv('https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-d
concrete_data.head()
```

Out[43]:

	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 [44]:

```
predictors = concrete_data[concrete_data.columns[concrete_data.columns != 'Strength']] # al
target = concrete_data['Strength'] # Strength column
X_train, X_test, y_train, y_test = train_test_split(predictors, target, test_size=0.3, ranc
```

In [45]:

```
model = regression_model()
model.fit(X_train, y_train, validation_split=0.3, epochs=50, verbose=2)
```

```
Train on 504 samples, validate on 217 samples
Epoch 1/50
 - 0s - loss: 10268.5969 - val_loss: 2689.7675
Epoch 2/50
 - 0s - loss: 1395.0632 - val_loss: 1247.4437
Epoch 3/50
 - 0s - loss: 1235.1358 - val_loss: 977.2205
Epoch 4/50
 - 0s - loss: 885.1052 - val_loss: 802.8241
Epoch 5/50
 - 0s - loss: 775.1341 - val_loss: 683.4287
Epoch 6/50
 - 0s - loss: 669.8382 - val_loss: 587.2058
Epoch 7/50
 - 0s - loss: 587.6973 - val_loss: 508.4336
Epoch 8/50
 - 0s - loss: 515.2526 - val_loss: 440.7214
Epoch 9/50
 - 0s - loss: 452.3115 - val_loss: 386.7347
Epoch 10/50
 - 0s - loss: 400.3828 - val_loss: 337.1610
Epoch 11/50
 - 0s - loss: 355.5185 - val_loss: 298.4111
Epoch 12/50
 - 0s - loss: 317.8439 - val_loss: 265.7309
Epoch 13/50
 - 0s - loss: 289.1048 - val_loss: 241.6813
Epoch 14/50
 - 0s - loss: 262.3097 - val_loss: 218.8321
Epoch 15/50
 - 0s - loss: 240.9941 - val_loss: 203.0837
Epoch 16/50
 - 0s - loss: 224.9472 - val_loss: 189.5271
Epoch 17/50
 - 0s - loss: 210.3457 - val_loss: 180.7436
Epoch 18/50
 - 0s - loss: 200.2524 - val_loss: 172.5694
Epoch 19/50
 - 0s - loss: 191.8028 - val_loss: 166.4470
Epoch 20/50
 - 0s - loss: 183.6717 - val_loss: 161.4261
Epoch 21/50
 - 0s - loss: 178.4846 - val_loss: 157.9196
Epoch 22/50
 - 0s - loss: 173.2400 - val_loss: 154.5944
Epoch 23/50
 - 0s - loss: 168.9768 - val_loss: 152.5857
Epoch 24/50
 - 0s - loss: 166.5429 - val_loss: 151.0187
Epoch 25/50
 - 0s - loss: 163.0695 - val_loss: 148.2517
Epoch 26/50
 - 0s - loss: 159.9154 - val_loss: 146.8361
Epoch 27/50
 - 0s - loss: 159.5405 - val_loss: 147.1761
```

```
Epoch 28/50
 - 0s - loss: 154.8907 - val_loss: 143.6492
Epoch 29/50
 - 0s - loss: 153.0973 - val_loss: 142.6078
Epoch 30/50
 - 0s - loss: 151.4816 - val_loss: 141.3300
Epoch 31/50
 - 0s - loss: 149.4848 - val_loss: 139.9784
Epoch 32/50
 - 0s - loss: 147.7185 - val_loss: 139.1919
Epoch 33/50
 - 0s - loss: 146.2990 - val_loss: 137.9541
Epoch 34/50
 - 0s - loss: 144.7135 - val_loss: 136.8433
Epoch 35/50
 - 0s - loss: 143.4827 - val_loss: 135.6464
Epoch 36/50
 - 0s - loss: 142.5174 - val_loss: 134.7813
Epoch 37/50
 - 0s - loss: 141.3471 - val_loss: 134.4854
Epoch 38/50
 - 0s - loss: 140.6160 - val_loss: 132.5681
Epoch 39/50
 - 0s - loss: 138.7125 - val_loss: 132.2208
Epoch 40/50
 - 0s - loss: 138.0131 - val_loss: 131.3336
Epoch 41/50
 - 0s - loss: 136.1823 - val_loss: 130.2135
Epoch 42/50
 - 0s - loss: 135.6456 - val_loss: 129.5249
Epoch 43/50
 - 0s - loss: 134.2210 - val_loss: 128.5946
Epoch 44/50
 - 0s - loss: 133.9389 - val_loss: 127.7005
Epoch 45/50
 - 0s - loss: 132.8082 - val_loss: 126.7347
Epoch 46/50
 - 0s - loss: 131.5243 - val_loss: 126.7427
Epoch 47/50
 - 0s - loss: 130.5799 - val_loss: 125.4540
Epoch 48/50
 - 0s - loss: 130.7525 - val_loss: 125.9542
Epoch 49/50
 - 0s - loss: 129.0626 - val_loss: 124.5556
Epoch 50/50
 - 0s - loss: 131.5369 - val_loss: 127.8681
```

Out[45]:

```
<keras.callbacks.callbacks.History at 0x229f68a4988>
```

In [46]:

```
mean_squared_error(model.predict(X_test), y_test)
```

Out[46]:

```
125.63924117981722
```

In [47]:

```
errors = []
for i in range(50):
    X_train, X_test, y_train, y_test = train_test_split(predictors, target, test_size=0.3)
    model = regression_model()
    model.fit(X_train, y_train, validation_split=0.3, epochs=50, verbose=2)
    errors.append(mean_squared_error(model.predict(X_test), y_test))
errors
```

```
Epoch 38/50
 - 0s - loss: 474.8904 - val_loss: 520.9891
Epoch 39/50
 - 0s - loss: 460.9376 - val_loss: 500.8438
Epoch 40/50
 - 0s - loss: 447.5108 - val_loss: 485.9866
Epoch 41/50
 - 0s - loss: 433.3692 - val_loss: 472.4088
Epoch 42/50
 - 0s - loss: 421.1157 - val_loss: 459.0791
Epoch 43/50
 - 0s - loss: 410.3005 - val_loss: 448.3518
Epoch 44/50
 - 0s - loss: 397.6177 - val_loss: 432.0524
Epoch 45/50
 - 0s - loss: 387.4993 - val_loss: 419.2809
Epoch 46/50
 - 0s - loss: 377.1081 - val_loss: 407.9970
Epoch 47/50
```

In [60]:

```
display(np.mean(errors),np.std(errors))
```

```
589.3226854333582

689.487718010884
```

Part B. Normalize Data

In [57]:

```
x = concrete_data[concrete_data.columns[concrete_data.columns != 'Strength']].values #retur
min_max_scaler = preprocessing.MinMaxScaler()
x_scaled = min_max_scaler.fit_transform(x)
df = pd.DataFrame(x_scaled)
```

In [58]:

```
predictors = df
target = concrete_data['Strength'] # Strength column
errors_norm = []
for i in range(50):
    X_train, X_test, y_train, y_test = train_test_split(predictors, target, test_size=0.3)
    model = regression_model()
    model.fit(X_train, y_train, validation_split=0.3, epochs=50, verbose=2)
    errors_norm.append(mean_squared_error(model.predict(X_test), y_test))
errors_norm
```

```
 - 0s - loss: 1103.3028 - val_loss: 977.1736
Epoch 43/50
 - 0s - loss: 1082.8085 - val_loss: 957.7576
Epoch 44/50
 - 0s - loss: 1062.1736 - val_loss: 938.4862
Epoch 45/50
 - 0s - loss: 1041.5885 - val_loss: 919.2544
Epoch 46/50
 - 0s - loss: 1021.1823 - val_loss: 899.7917
Epoch 47/50
 - 0s - loss: 1000.5790 - val_loss: 880.4953
Epoch 48/50
 - 0s - loss: 980.1605 - val_loss: 861.0785
Epoch 49/50
 - 0s - loss: 959.7425 - val_loss: 841.8042
Epoch 50/50
 - 0s - loss: 939.2112 - val_loss: 822.9333
Train on 504 samples, validate on 217 samples
Epoch 1/50
 - 0s - loss: 1534.3443 - val_loss: 1646.0593
```

In [59]:

```
display(np.mean(errors_norm),np.std(errors_norm))
```

```
402.8560686749054

129.0468074364393
```

Both statistics seem to be smaller.

Part C. Increase Epochs

In [61]:

```
errors_norm_100 = []
for i in range(50):
    X_train, X_test, y_train, y_test = train_test_split(predictors, target, test_size=0.3)
    model = regression_model()
    model.fit(X_train, y_train, validation_split=0.3, epochs=100, verbose=2)
    errors_norm_100.append(mean_squared_error(model.predict(X_test), y_test))
errors_norm_100
```

```
 226.39478172798167,
 222.0478559688034,
 255.33888855180916,
 241.21986461020194,
 213.94146404165113,
 285.764362601592,
 230.7149315944909,
 251.20950554274575,
 238.20372039144607,
 225.72672858046823,
 244.17881486718417,
 227.66485389562285,
 223.9422504283058,
 197.51341983550594,
 210.18096977601104,
 231.2078612253154,
 188.1925847313396,
 198.22245604800588,
 258.7652598813493]
```

In [62]:

```
display(np.mean(errors_norm_100),np.std(errors_norm_100))
```

```
223.17213792818492

21.332118729162755
```

Both statistics seem to be smaller, while standard deviation becomes significantly smaller.

Part D. More Hidden Layers & Nodes

In [63]:

```
def regression_model_3():
    # create model
    model = Sequential()
    model.add(Dense(10, activation='relu', input_shape=(len(concrete_data.columns)-1,)))
    model.add(Dense(10, activation='relu'))
    model.add(Dense(10, activation='relu'))
    model.add(Dense(1))

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

In [64]:

```
errors_norm_3 = []
for i in range(50):
    X_train, X_test, y_train, y_test = train_test_split(predictors, target, test_size=0.3)
    model = regression_model()
    model.fit(X_train, y_train, validation_split=0.3, epochs=50, verbose=2)
    errors_norm_3.append(mean_squared_error(model.predict(X_test), y_test))
errors_norm_3
```

```
 498.5036014251007,
 280.81099846888804,
 573.9167283361937,
 499.3824262203175,
 598.3395492792367,
 294.035260363735,
 270.8079734105078,
 238.9634231070705,
 653.6762979512669,
 323.26802191715325,
 307.2229936447627,
 401.23092057161574,
 803.0662681377381,
 494.489721898701,
 314.97911013688895,
 387.17397875101614,
 306.2550354167196,
 333.91044466525096,
 240.0001818657873]
```

In [65]:

```
display(np.mean(errors_norm_3),np.std(errors_norm_3))
```

```
442.3663639502136

177.7338025645331
```

The statistics seems to be a bit larger but almost ignorable.