Building Regression Model with Keras

In this project, I will build a regression model using the deep learning Keras library, and then I will experiment with increasing the number of training epochs and changing number of hidden layers and you will see how changing these parameters impacts the performance of the model.

Importing Libraries

In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
```

Loading the dataset

In [ ]:
```
df = pd.read_csv('https://cocl.us/concrete_data')
df.head()
```

Out[ ]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fine Aggregate	Age	Streng
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.

In [ ]:
```
df.isnull().sum()
```

Out[ ]:
```
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
```

In [ ]:
```
df.describe()
```

Out[ ]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coars Aggregat
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.00000
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.91893
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.75395
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.00000
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.00000
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.00000
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.40000
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.00000

In [ ]:

```
predictor = df.drop(columns = ['Strength'])
target = df['Strength']
n_cols = predictor.shape[1]
```

In [ ]:

```
predictor.head()
```

Out[ ]:

	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 [ ]:

```
target.head()
```

Out[ ]:

```
0    79.99
1    61.89
2    40.27
3    41.05
4    44.30
Name: Strength, dtype: float64
```

Section: A

Building the baseline Neural Network Model

In [ ]:

```
def regression_model():
  #creating the model
  model = Sequential()
```

```python
    #first hidden layer
    model.add(Dense(10, activation = 'relu', input_shape = (n_cols,)))
    #output layer
    model.add(Dense(1))
    #compiling the model
    model.compile(optimizer = 'adam', loss = 'mean_squared_error')
    return model
```

In [ ]:
```python
model = regression_model()
```

In [ ]:
```python
mean_sq_error = []
for i in range(1,51):
    X_train,X_test,y_train, y_test = train_test_split(predictor, df['Strength'],te
    res = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs =
    mean_squared_error = res.history['val_loss'][-1]
    print('Iteration--->', i, 'mean squared error----->',mean_squared_error )
    mean_sq_error.append(mean_squared_error)
```

```
Iteration---> 1 mean squared error-----> 114.92092895507812
Iteration---> 2 mean squared error-----> 94.17741394042969
Iteration---> 3 mean squared error-----> 87.03341674804688
Iteration---> 4 mean squared error-----> 67.53215026855469
Iteration---> 5 mean squared error-----> 64.21414947509766
Iteration---> 6 mean squared error-----> 80.02247619628906
Iteration---> 7 mean squared error-----> 55.96617126464844
Iteration---> 8 mean squared error-----> 70.28727722167969
Iteration---> 9 mean squared error-----> 61.35123825073242
Iteration---> 10 mean squared error-----> 54.93370819091797
Iteration---> 11 mean squared error-----> 58.0841178894043
Iteration---> 12 mean squared error-----> 66.39708709716797
Iteration---> 13 mean squared error-----> 48.5130729675293
Iteration---> 14 mean squared error-----> 51.276336669921875
Iteration---> 15 mean squared error-----> 49.182151794433594
Iteration---> 16 mean squared error-----> 47.912715911865234
Iteration---> 17 mean squared error-----> 48.07288360595703
Iteration---> 18 mean squared error-----> 54.99568176269531
Iteration---> 19 mean squared error-----> 79.58499908447266
Iteration---> 20 mean squared error-----> 46.385498046875
Iteration---> 21 mean squared error-----> 51.937259674072266
Iteration---> 22 mean squared error-----> 44.85268783569336
Iteration---> 23 mean squared error-----> 55.07088088989258
Iteration---> 24 mean squared error-----> 45.53147506713867
Iteration---> 25 mean squared error-----> 46.26972198486328
Iteration---> 26 mean squared error-----> 51.513187408447266
Iteration---> 27 mean squared error-----> 54.05939483642578
Iteration---> 28 mean squared error-----> 47.752315521240234
Iteration---> 29 mean squared error-----> 43.44196701049805
Iteration---> 30 mean squared error-----> 59.32704544067383
Iteration---> 31 mean squared error-----> 56.60659408569336
Iteration---> 32 mean squared error-----> 55.745296478271484
Iteration---> 33 mean squared error-----> 49.108741760253906
Iteration---> 34 mean squared error-----> 50.989654541015625
Iteration---> 35 mean squared error-----> 49.661556243896484
Iteration---> 36 mean squared error-----> 43.949615478515625
Iteration---> 37 mean squared error-----> 50.89598846435547
Iteration---> 38 mean squared error-----> 58.1651611328125
Iteration---> 39 mean squared error-----> 46.66064453125
Iteration---> 40 mean squared error-----> 46.9597282409668
Iteration---> 41 mean squared error-----> 48.741146087646484
Iteration---> 42 mean squared error-----> 50.25631332397461
Iteration---> 43 mean squared error-----> 54.9888801574707
Iteration---> 44 mean squared error-----> 49.60201644897461
Iteration---> 45 mean squared error-----> 47.1872673034668
Iteration---> 46 mean squared error-----> 55.24275588989258
Iteration---> 47 mean squared error-----> 42.09994888305664
Iteration---> 48 mean squared error-----> 48.621055603027344
Iteration---> 49 mean squared error-----> 45.85637664794922
Iteration---> 50 mean squared error-----> 45.26753616333008
```

In [ ]:

```python
print('mean of mean squared error:', np.mean(mean_sq_error))
print('standard daviation of mean squared error:', np.std(mean_sq_error))
```

```
mean of mean squared error: 55.94411376953125
standard daviation of mean squared error: 13.896112836175579
```

Section: B

Normalizing the predictor columns

In [ ]:
```
predictor = (predictor-predictor.mean())/predictor.std()
predictor.head()
```

Out[ ]:

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

In [ ]:
```
mean_sq_error = []
for i in range(1,51):
  X_train,X_test,y_train, y_test = train_test_split(predictor, df['Strength'],te
  res = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs =
  mean_squared_error = res.history['val_loss'][-1]
  print('Iteration--->', i, 'mean squared error----->',mean_squared_error )
  mean_sq_error.append(mean_squared_error)
```

```
Iteration---> 1 mean squared error-----> 749.7655029296875
Iteration---> 2 mean squared error-----> 292.0740661621094
Iteration---> 3 mean squared error-----> 183.6770782470703
Iteration---> 4 mean squared error-----> 111.27953338623047
Iteration---> 5 mean squared error-----> 112.56428527832031
Iteration---> 6 mean squared error-----> 71.45450592041016
Iteration---> 7 mean squared error-----> 68.31498718261719
Iteration---> 8 mean squared error-----> 60.875877380371094
Iteration---> 9 mean squared error-----> 49.45834732055664
Iteration---> 10 mean squared error-----> 57.024044036865234
Iteration---> 11 mean squared error-----> 44.935997009277344
Iteration---> 12 mean squared error-----> 44.09235763549805
Iteration---> 13 mean squared error-----> 50.500526428222656
Iteration---> 14 mean squared error-----> 49.68511962890625
Iteration---> 15 mean squared error-----> 39.100799560546875
Iteration---> 16 mean squared error-----> 54.63429641723633
Iteration---> 17 mean squared error-----> 39.5147705078125
Iteration---> 18 mean squared error-----> 42.220970153808594
Iteration---> 19 mean squared error-----> 43.17979049682617
Iteration---> 20 mean squared error-----> 42.96100997924805
Iteration---> 21 mean squared error-----> 37.29590606689453
Iteration---> 22 mean squared error-----> 40.378177642822266
Iteration---> 23 mean squared error-----> 38.132198333740234
Iteration---> 24 mean squared error-----> 44.44504165649414
Iteration---> 25 mean squared error-----> 38.319091796875
Iteration---> 26 mean squared error-----> 41.24111557006836
Iteration---> 27 mean squared error-----> 47.928794860839844
Iteration---> 28 mean squared error-----> 44.364505767822266
Iteration---> 29 mean squared error-----> 37.464412689208984
Iteration---> 30 mean squared error-----> 35.12382125854492
Iteration---> 31 mean squared error-----> 31.361221313476562
Iteration---> 32 mean squared error-----> 37.94083786010742
Iteration---> 33 mean squared error-----> 36.42417907714844
Iteration---> 34 mean squared error-----> 36.683738708496094
Iteration---> 35 mean squared error-----> 39.76728057861328
Iteration---> 36 mean squared error-----> 33.729679107666016
Iteration---> 37 mean squared error-----> 41.91026306152344
Iteration---> 38 mean squared error-----> 36.06751251220703
Iteration---> 39 mean squared error-----> 41.17768859863281
Iteration---> 40 mean squared error-----> 38.9312629699707
Iteration---> 41 mean squared error-----> 36.902374267578125
Iteration---> 42 mean squared error-----> 37.854923248291016
Iteration---> 43 mean squared error-----> 41.39990234375
Iteration---> 44 mean squared error-----> 31.65899085998535
Iteration---> 45 mean squared error-----> 33.07505798339844
Iteration---> 46 mean squared error-----> 37.31159591674805
Iteration---> 47 mean squared error-----> 36.02543258666992
Iteration---> 48 mean squared error-----> 38.142181396484375
Iteration---> 49 mean squared error-----> 38.646236419677734
Iteration---> 50 mean squared error-----> 34.08334732055664
```

In [ ]:

```
print('mean of mean squared error:', np.mean(mean_sq_error))
print('standard daviation of mean squared error:', np.std(mean_sq_error))
```

```
mean of mean squared error: 67.02201274871827
standard daviation of mean squared error: 106.3378246473775
```

Section: C

Setting the epoch count to 100 for training

In [ ]:
```
mean_sq_error = []
for i in range(1,51):
  X_train,X_test,y_train, y_test = train_test_split(predictor, df['Strength'],te
  res = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs =
  mean_squared_error = res.history['val_loss'][-1]
  print('Iteration--->', i, 'mean squared error----->',mean_squared_error )
  mean_sq_error.append(mean_squared_error)
```

```
Iteration---> 1 mean squared error-----> 37.38007736206055
Iteration---> 2 mean squared error-----> 32.540470123291016
Iteration---> 3 mean squared error-----> 42.34327697753906
Iteration---> 4 mean squared error-----> 37.019187927246094
Iteration---> 5 mean squared error-----> 38.971763610839844
Iteration---> 6 mean squared error-----> 37.568572998046875
Iteration---> 7 mean squared error-----> 31.379030227661133
Iteration---> 8 mean squared error-----> 39.05885696411133
Iteration---> 9 mean squared error-----> 36.84315490722656
Iteration---> 10 mean squared error-----> 37.591949462890625
Iteration---> 11 mean squared error-----> 36.410335540771484
Iteration---> 12 mean squared error-----> 35.89048767089844
Iteration---> 13 mean squared error-----> 36.10889434814453
Iteration---> 14 mean squared error-----> 38.508914947509766
Iteration---> 15 mean squared error-----> 36.65087127685547
Iteration---> 16 mean squared error-----> 32.48939895629883
Iteration---> 17 mean squared error-----> 36.580848693847656
Iteration---> 18 mean squared error-----> 34.73881912231445
Iteration---> 19 mean squared error-----> 31.89695167541504
Iteration---> 20 mean squared error-----> 34.02641677856445
Iteration---> 21 mean squared error-----> 34.83200454711914
Iteration---> 22 mean squared error-----> 35.41761779785156
Iteration---> 23 mean squared error-----> 38.56675720214844
Iteration---> 24 mean squared error-----> 39.26572036743164
Iteration---> 25 mean squared error-----> 39.07368850708008
Iteration---> 26 mean squared error-----> 37.39773941040039
Iteration---> 27 mean squared error-----> 39.435054779052734
Iteration---> 28 mean squared error-----> 35.35947036743164
Iteration---> 29 mean squared error-----> 36.54876708984375
Iteration---> 30 mean squared error-----> 38.40338134765625
Iteration---> 31 mean squared error-----> 39.78046798706055
Iteration---> 32 mean squared error-----> 40.98807907104492
Iteration---> 33 mean squared error-----> 34.43544387817383
Iteration---> 34 mean squared error-----> 33.994327545166016
Iteration---> 35 mean squared error-----> 41.26124954223633
Iteration---> 36 mean squared error-----> 41.484981536865234
Iteration---> 37 mean squared error-----> 36.09012222290039
Iteration---> 38 mean squared error-----> 40.20849609375
Iteration---> 39 mean squared error-----> 34.78286361694336
Iteration---> 40 mean squared error-----> 37.64076232910156
Iteration---> 41 mean squared error-----> 36.676876068115234
Iteration---> 42 mean squared error-----> 34.878875732421875
Iteration---> 43 mean squared error-----> 36.43764877319336
Iteration---> 44 mean squared error-----> 41.72425079345703
Iteration---> 45 mean squared error-----> 33.902225494384766
Iteration---> 46 mean squared error-----> 43.62229919433594
Iteration---> 47 mean squared error-----> 34.686676025390625
Iteration---> 48 mean squared error-----> 33.79835510253906
Iteration---> 49 mean squared error-----> 35.264225006103516
Iteration---> 50 mean squared error-----> 35.77212905883789
```

In [ ]:

```
print('mean of mean squared error:', np.mean(mean_sq_error))
print('standard daviation of mean squared error:', np.std(mean_sq_error))
```

```
mean of mean squared error: 36.91457672119141
standard daviation of mean squared error: 2.75228126572097
```

Section: D

Neural Network with three Hidden layers, each of 10 nodes and ReLU activation Function

In [ ]:
```
def regression_model():
  #creating the model
  model = Sequential()
  #first hidden layer
  model.add(Dense(10, activation = 'relu', input_shape = (n_cols,)))
  #second hidden layer
  model.add(Dense(10, activation = 'relu'))
  #third hidden layer
  model.add(Dense(10, activation = 'relu'))
  #output layer
  model.add(Dense(1))
  #compiling the model
  model.compile(optimizer = 'adam', loss = 'mean_squared_error')
  return model
```

In [ ]:
```
model = regression_model()
```

In [ ]:
```
mean_sq_error = []
for i in range(1,51):
  X_train,X_test,y_train, y_test = train_test_split(predictor, df['Strength'],te
  res = model.fit(X_train, y_train, validation_data = (X_test, y_test), epochs =
  mean_squared_error = res.history['val_loss'][-1]
  print('Iteration--->', i, 'mean squared error----->',mean_squared_error )
  mean_sq_error.append(mean_squared_error)
```

```
Iteration---> 1 mean squared error-----> 134.85989379882812
Iteration---> 2 mean squared error-----> 83.92335510253906
Iteration---> 3 mean squared error-----> 71.07125854492188
Iteration---> 4 mean squared error-----> 40.500892639160156
Iteration---> 5 mean squared error-----> 45.696449279785156
Iteration---> 6 mean squared error-----> 44.826744079589844
Iteration---> 7 mean squared error-----> 39.3478889465332
Iteration---> 8 mean squared error-----> 33.13433074951172
Iteration---> 9 mean squared error-----> 35.00434112548828
Iteration---> 10 mean squared error-----> 40.123023986816406
Iteration---> 11 mean squared error-----> 36.01560592651367
Iteration---> 12 mean squared error-----> 36.12113952636719
Iteration---> 13 mean squared error-----> 36.834571838378906
Iteration---> 14 mean squared error-----> 31.618337631225586
Iteration---> 15 mean squared error-----> 32.108253479003906
Iteration---> 16 mean squared error-----> 30.435970306396484
Iteration---> 17 mean squared error-----> 31.02277374267578
Iteration---> 18 mean squared error-----> 30.074195861816406
Iteration---> 19 mean squared error-----> 30.900087356567383
Iteration---> 20 mean squared error-----> 29.833538055419922
Iteration---> 21 mean squared error-----> 30.35759162902832
Iteration---> 22 mean squared error-----> 29.497278213500977
Iteration---> 23 mean squared error-----> 31.24148941040039
Iteration---> 24 mean squared error-----> 29.175626754760742
Iteration---> 25 mean squared error-----> 25.999130249023438
Iteration---> 26 mean squared error-----> 27.892986297607422
Iteration---> 27 mean squared error-----> 28.88304328918457
Iteration---> 28 mean squared error-----> 30.72061538696289
Iteration---> 29 mean squared error-----> 28.25157356262207
Iteration---> 30 mean squared error-----> 25.58354949951172
Iteration---> 31 mean squared error-----> 24.707717895507812
Iteration---> 32 mean squared error-----> 28.073862075805664
Iteration---> 33 mean squared error-----> 25.941783905029297
Iteration---> 34 mean squared error-----> 24.863862991333008
Iteration---> 35 mean squared error-----> 23.568439483642578
Iteration---> 36 mean squared error-----> 24.01810073852539
Iteration---> 37 mean squared error-----> 26.433147430419922
Iteration---> 38 mean squared error-----> 27.507322311401367
Iteration---> 39 mean squared error-----> 26.203752517700195
Iteration---> 40 mean squared error-----> 21.713579177856445
Iteration---> 41 mean squared error-----> 28.557125091552734
Iteration---> 42 mean squared error-----> 24.531084060668945
Iteration---> 43 mean squared error-----> 23.74705696105957
Iteration---> 44 mean squared error-----> 24.248607635498047
Iteration---> 45 mean squared error-----> 28.002540588378906
Iteration---> 46 mean squared error-----> 23.531204223632812
Iteration---> 47 mean squared error-----> 23.14558982849121
Iteration---> 48 mean squared error-----> 21.9714298248291
Iteration---> 49 mean squared error-----> 25.653797149658203
Iteration---> 50 mean squared error-----> 29.40066909790039
```

In [ ]:

```python
print('mean of mean squared error:', np.mean(mean_sq_error))
print('standard daviation of mean squared error:', np.std(mean_sq_error))
```

```
mean of mean squared error: 33.73752418518066
standard daviation of mean squared error: 18.1131076923468
```

Comparing Mean Squared Error in all the 4 sections

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.pointplot(x = ['Section A', 'Section B', 'Section C', 'Section D'], y = [55.
plt.xlabel('Section')
plt.ylabel('Mean Squared Error')
```

Out[ ]: Text(0, 0.5, 'Mean Squared Error')