Building a Regression Model in Keras Neural Network

Course Project - Introduction to Deep Learning & Neural Networks with Keras

Ahmed Yahya Khaled \ *July 11, 2020*

Introduction

Here, We'll build a regression model using the deep learning Keras library, and then We'll experiment with increasing the number of training epochs and changing number of hidden layers and we'll see how changing these parameters impacts the performance of the model.

Contents :

Pre-work

1. Import the Dataset
2. Explore the Dataset
3. Data Preprocessing

Activities

A. Build a baseline model \ **B.** Normalize the Data \ **C.** Increase the number of epochs \ **D.** Increase the number of hidden layers

-------------------------------- Pre-work --

Import the Dataset

In [62]:
```
import numpy as np
import pandas as pd
```

In [63]:
```
concrete_data = pd.read_csv('https://cocl.us/concrete_data')
```

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

Explore the Dataset

In [64]:
```
concrete_data.head()
```

Out[64]:

	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 [65]:
```
concrete_data.shape
```

Out[65]: (1030, 9)

In [66]:
```
concrete_data.describe()
```

Out[66]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Agg
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000	1030.(
mean	281.167864	73.895825	54.188350	181.567282	6.204660	972.918932	773.!
std	104.506364	86.279342	63.997004	21.354219	5.973841	77.753954	80.'
min	102.000000	0.000000	0.000000	121.800000	0.000000	801.000000	594.(
25%	192.375000	0.000000	0.000000	164.900000	0.000000	932.000000	730.!
50%	272.900000	22.000000	0.000000	185.000000	6.400000	968.000000	779.!
75%	350.000000	142.950000	118.300000	192.000000	10.200000	1029.400000	824.(
max	540.000000	359.400000	200.100000	247.000000	32.200000	1145.000000	992.(

In [67]:
```
concrete_data.corr()
```

Out[67]:

	Cement	Blast Furnace Slag	Fly Ash	Water	Superplasticizer	Coarse Aggregate	Fi Aggrega
Cement	1.000000	-0.275216	-0.397467	-0.081587	0.092386	-0.109349	-0.2227
Blast Furnace Slag	-0.275216	1.000000	-0.323580	0.107252	0.043270	-0.283999	-0.2816
Fly Ash	-0.397467	-0.323580	1.000000	-0.256984	0.377503	-0.009961	0.0791
Water	-0.081587	0.107252	-0.256984	1.000000	-0.657533	-0.182294	-0.4506
Superplasticizer	0.092386	0.043270	0.377503	-0.657533	1.000000	-0.265999	0.2226
Coarse Aggregate	-0.109349	-0.283999	-0.009961	-0.182294	-0.265999	1.000000	-0.1784
Fine Aggregate	-0.222718	-0.281603	0.079108	-0.450661	0.222691	-0.178481	1.0000
Age	0.081946	-0.044246	-0.154371	0.277618	-0.192700	-0.003016	-0.1560
Strength	0.497832	0.134829	-0.105755	-0.289633	0.366079	-0.164935	-0.1672

In [68]:
```
concrete_data.isnull().sum()
```

Out[68]:
```
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 Preprocessing

Split the Dataset into *predictors* and *target*

In [69]:
```
col_names = concrete_data.columns
X = concrete_data[col_names[col_names != 'Strength']]
y = concrete_data['Strength']
```

In [70]:
```
X.head()
```

Out[70]:

	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 [71]:
```
y.head()
```

Out[71]:
```
0    79.99
1    61.89
2    40.27
3    41.05
4    44.30
Name: Strength, dtype: float64
```

Normalize the data

In [72]:
```
X_norm = (X - X.mean()) / X.std()
X_norm.head()
```

Out[72]:

	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 [73]:
```
n_cols = X.shape[1]
print(n_cols)
```

```
8
```

-------------------------------- Submission --

A. Build a baseline model

Use the Keras library to build a neural network with the following:

- One hidden layer of 10 nodes, and a ReLU activation function
- Use the adam optimizer and the mean squared error as the loss function.

1. Randomly split the data into a training and test sets by holding 30% of the data for testing. You can use the train_test_splithelper function from Scikit-learn.
2. Train the model on the training data using 50 epochs.
3. Evaluate the model on the test data and compute the mean squared error between the predicted concrete strength and the actual concrete strength. You can use the mean_squared_error function from Scikit-learn.
4. Repeat steps 1 - 3, 50 times, i.e., create a list of 50 mean squared errors.
5. Report the mean and the standard deviation of the mean squared errors.

Import keras

In [74]:
```
import keras
from  keras.models import Sequential
from keras.layers import Dense
```

keras Regression Model

In [75]:
```
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
```

Train - Test split

In [76]:
```
from sklearn.model_selection import train_test_split
```

In [77]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, ran
dom_state = 42)
```

Train the model

In [78]:
```
model = regression_model()
```

In [79]:
```
model.fit(X_train, y_train, epochs=50, verbose=1)
```

```
Epoch 1/50
721/721 [==============================] - 0s 150us/step - loss: 272333.7664
Epoch 2/50
721/721 [==============================] - 0s 24us/step - loss: 218301.0654
Epoch 3/50
721/721 [==============================] - 0s 29us/step - loss: 178443.0157
Epoch 4/50
721/721 [==============================] - 0s 35us/step - loss: 145015.4284
Epoch 5/50
721/721 [==============================] - 0s 25us/step - loss: 105364.9945
Epoch 6/50
721/721 [==============================] - 0s 26us/step - loss: 56409.9769
Epoch 7/50
721/721 [==============================] - 0s 30us/step - loss: 24661.5120
Epoch 8/50
721/721 [==============================] - 0s 32us/step - loss: 9762.0628
Epoch 9/50
721/721 [==============================] - 0s 31us/step - loss: 4145.5584
Epoch 10/50
721/721 [==============================] - 0s 31us/step - loss: 2541.8902
Epoch 11/50
721/721 [==============================] - 0s 24us/step - loss: 2166.1307
Epoch 12/50
721/721 [==============================] - 0s 26us/step - loss: 2047.8657
Epoch 13/50
721/721 [==============================] - 0s 26us/step - loss: 1966.7914
Epoch 14/50
721/721 [==============================] - 0s 32us/step - loss: 1887.5234
Epoch 15/50
721/721 [==============================] - 0s 26us/step - loss: 1805.4209
Epoch 16/50
721/721 [==============================] - 0s 26us/step - loss: 1723.6867
Epoch 17/50
721/721 [==============================] - 0s 26us/step - loss: 1648.7437
Epoch 18/50
721/721 [==============================] - 0s 26us/step - loss: 1569.6093
Epoch 19/50
721/721 [==============================] - 0s 25us/step - loss: 1493.4488
Epoch 20/50
721/721 [==============================] - 0s 21us/step - loss: 1419.8601
Epoch 21/50
721/721 [==============================] - 0s 24us/step - loss: 1341.7485
Epoch 22/50
721/721 [==============================] - 0s 22us/step - loss: 1269.7651
Epoch 23/50
721/721 [==============================] - 0s 28us/step - loss: 1196.0326
Epoch 24/50
721/721 [==============================] - 0s 19us/step - loss: 1129.7533
Epoch 25/50
721/721 [==============================] - 0s 21us/step - loss: 1065.2731
Epoch 26/50
721/721 [==============================] - 0s 25us/step - loss: 1007.0188
Epoch 27/50
721/721 [==============================] - 0s 22us/step - loss: 949.8549
Epoch 28/50
721/721 [==============================] - 0s 25us/step - loss: 898.2771
Epoch 29/50
```

```
721/721 [==============================] - 0s 21us/step - loss: 847.9905
Epoch 30/50
721/721 [==============================] - 0s 25us/step - loss: 801.5482
Epoch 31/50
721/721 [==============================] - 0s 32us/step - loss: 760.5527
Epoch 32/50
721/721 [==============================] - 0s 25us/step - loss: 724.4048
Epoch 33/50
721/721 [==============================] - 0s 29us/step - loss: 689.4241
Epoch 34/50
721/721 [==============================] - 0s 29us/step - loss: 655.7091
Epoch 35/50
721/721 [==============================] - 0s 28us/step - loss: 626.8635
Epoch 36/50
721/721 [==============================] - 0s 26us/step - loss: 599.9263
Epoch 37/50
721/721 [==============================] - 0s 29us/step - loss: 575.1667
Epoch 38/50
721/721 [==============================] - 0s 29us/step - loss: 551.8070
Epoch 39/50
721/721 [==============================] - 0s 25us/step - loss: 531.7888
Epoch 40/50
721/721 [==============================] - 0s 25us/step - loss: 513.8963
Epoch 41/50
721/721 [==============================] - 0s 29us/step - loss: 498.4218
Epoch 42/50
721/721 [==============================] - 0s 28us/step - loss: 484.3788
Epoch 43/50
721/721 [==============================] - 0s 22us/step - loss: 472.9572
Epoch 44/50
721/721 [==============================] - 0s 22us/step - loss: 462.0412
Epoch 45/50
721/721 [==============================] - 0s 26us/step - loss: 451.7709
Epoch 46/50
721/721 [==============================] - 0s 28us/step - loss: 443.2711
Epoch 47/50
721/721 [==============================] - 0s 31us/step - loss: 434.6778
Epoch 48/50
721/721 [==============================] - 0s 25us/step - loss: 426.8505
Epoch 49/50
721/721 [==============================] - 0s 26us/step - loss: 420.6635
Epoch 50/50
721/721 [==============================] - 0s 31us/step - loss: 413.9921
```

Out[79]:
```
<keras.callbacks.callbacks.History at 0x22f27afb940>
```

Test **the model**

In [80]:
```
loss = model.evaluate(X_test, y_test)
print(loss)
```

```
309/309 [==============================] - 0s 32us/step
432.7098643478838
```

In [81]:
```
y_pred = model.predict(X_test)
```

Compute *Mean Squared Error*

between the predicted concrete strength and the actual concrete strength

In [82]:
```
from sklearn.metrics import mean_squared_error
```

In [83]:
```
mse = mean_squared_error(y_test, y_pred)
print(mse)
```

```
432.70986816572776
```

***50* mean squared errors**

In [84]:
```
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(X, y, test_size=0.3, r
andom_state=i)
    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_e
rrors) + " mean squared errors without normalized data. Total number of epochs
for each training is: " +str(epochs) + "\n")
print("Mean: "+str(mean))
print("Standard Deviation: "+str(standard_deviation))
```

```
MSE 1: 278.7064796620588
MSE 2: 220.223395782767
MSE 3: 149.24345244867516
MSE 4: 122.80592906667962
MSE 5: 111.60202939919283
MSE 6: 99.53766024691387
MSE 7: 127.19873994993932
MSE 8: 73.05735267713231
MSE 9: 67.49624959704946
MSE 10: 84.33573479019708
MSE 11: 70.44388617358162
MSE 12: 58.035958435157355
MSE 13: 69.99706727555655
MSE 14: 67.32523673097678
MSE 15: 65.1229818893482
MSE 16: 56.46356751772192
MSE 17: 60.794189761757465
MSE 18: 55.01460485859596
MSE 19: 58.20418272049296
MSE 20: 63.644495633813555
MSE 21: 55.48421930109413
MSE 22: 65.00987460929599
MSE 23: 55.00450029342306
MSE 24: 60.63892971890644
MSE 25: 55.558984355247524
MSE 26: 64.60918957361511
MSE 27: 56.6840592171382
MSE 28: 62.7083652459302
MSE 29: 63.45589381591402
MSE 30: 65.45067358171283
MSE 31: 67.13298501320256
MSE 32: 61.52257815611015
MSE 33: 62.33978298644032
MSE 34: 62.60253669220267
MSE 35: 61.08707316870828
MSE 36: 69.12741201209404
MSE 37: 62.369839640497005
MSE 38: 66.60252340866138
MSE 39: 57.41867975278194
MSE 40: 58.01993734859726
MSE 41: 69.7952600374191
MSE 42: 66.24547954670435
MSE 43: 62.7938320443854
MSE 44: 65.823955177875
MSE 45: 64.24974729173778
MSE 46: 65.28268994328273
MSE 47: 63.1895356656664
MSE 48: 66.82876821474736
MSE 49: 70.40150818315524
MSE 50: 64.23157197526358


Below is the mean and standard deviation of 50 mean squared errors without no
rmalized data. Total number of epochs for each training is: 50

Mean: 77.01847104352677
Standard Deviation: 40.40459429889572
```

B. Normalize the data

- Repeat Part A but use a normalized version of the data.
- Recall that one way to normalize the data is by subtracting the mean from the individual predictors and dividing by the standard deviation.
- How does the mean of the mean squared errors compare to that from Step A ?

Train - Test Split

In [85]:
```
X_train_norm, X_test_norm, y_train, y_test = train_test_split(X_norm, y, test_
size = 0.3, random_state = 42)
```

Train the model

In [87]:
```
model.fit(X_train_norm, y_train, epochs=50, verbose=1)
```

```
Epoch 1/50
721/721 [==============================] - 0s 29us/step - loss: 301.1375
Epoch 2/50
721/721 [==============================] - 0s 26us/step - loss: 291.0998
Epoch 3/50
721/721 [==============================] - 0s 22us/step - loss: 281.1601
Epoch 4/50
721/721 [==============================] - 0s 28us/step - loss: 271.6438
Epoch 5/50
721/721 [==============================] - 0s 33us/step - loss: 262.9007
Epoch 6/50
721/721 [==============================] - 0s 31us/step - loss: 254.3113
Epoch 7/50
721/721 [==============================] - 0s 39us/step - loss: 246.3465
Epoch 8/50
721/721 [==============================] - 0s 37us/step - loss: 238.7796
Epoch 9/50
721/721 [==============================] - 0s 33us/step - loss: 231.6693
Epoch 10/50
721/721 [==============================] - 0s 42us/step - loss: 224.8832
Epoch 11/50
721/721 [==============================] - 0s 43us/step - loss: 218.4848
Epoch 12/50
721/721 [==============================] - 0s 37us/step - loss: 212.5280
Epoch 13/50
721/721 [==============================] - 0s 37us/step - loss: 206.7904
Epoch 14/50
721/721 [==============================] - 0s 37us/step - loss: 201.5026
Epoch 15/50
721/721 [==============================] - 0s 36us/step - loss: 196.2742
Epoch 16/50
721/721 [==============================] - 0s 33us/step - loss: 191.4879
Epoch 17/50
721/721 [==============================] - 0s 49us/step - loss: 186.9933
Epoch 18/50
721/721 [==============================] - 0s 35us/step - loss: 182.7930
Epoch 19/50
721/721 [==============================] - 0s 39us/step - loss: 178.6779
Epoch 20/50
721/721 [==============================] - 0s 36us/step - loss: 174.9886
Epoch 21/50
721/721 [==============================] - 0s 37us/step - loss: 171.4659
Epoch 22/50
721/721 [==============================] - 0s 29us/step - loss: 168.2438
Epoch 23/50
721/721 [==============================] - 0s 33us/step - loss: 165.1132
Epoch 24/50
721/721 [==============================] - 0s 35us/step - loss: 162.2993
Epoch 25/50
721/721 [==============================] - 0s 33us/step - loss: 159.6330
Epoch 26/50
721/721 [==============================] - 0s 26us/step - loss: 157.2483
Epoch 27/50
721/721 [==============================] - 0s 31us/step - loss: 154.7967
Epoch 28/50
721/721 [==============================] - 0s 33us/step - loss: 152.6477
Epoch 29/50
```

```
721/721 [==============================] - 0s 29us/step - loss: 150.5757
Epoch 30/50
721/721 [==============================] - 0s 24us/step - loss: 148.6557
Epoch 31/50
721/721 [==============================] - 0s 26us/step - loss: 146.8071
Epoch 32/50
721/721 [==============================] - 0s 39us/step - loss: 145.0550
Epoch 33/50
721/721 [==============================] - 0s 46us/step - loss: 143.5260
Epoch 34/50
721/721 [==============================] - 0s 49us/step - loss: 141.9916
Epoch 35/50
721/721 [==============================] - 0s 36us/step - loss: 140.5413
Epoch 36/50
721/721 [==============================] - 0s 37us/step - loss: 139.2453
Epoch 37/50
721/721 [==============================] - 0s 33us/step - loss: 137.9901
Epoch 38/50
721/721 [==============================] - 0s 29us/step - loss: 136.6825
Epoch 39/50
721/721 [==============================] - 0s 29us/step - loss: 135.5774
Epoch 40/50
721/721 [==============================] - 0s 28us/step - loss: 134.3728
Epoch 41/50
721/721 [==============================] - 0s 26us/step - loss: 133.2744
Epoch 42/50
721/721 [==============================] - 0s 26us/step - loss: 132.2308
Epoch 43/50
721/721 [==============================] - 0s 42us/step - loss: 131.2025
Epoch 44/50
721/721 [==============================] - 0s 28us/step - loss: 130.1829
Epoch 45/50
721/721 [==============================] - 0s 29us/step - loss: 129.2453
Epoch 46/50
721/721 [==============================] - 0s 26us/step - loss: 128.3133
Epoch 47/50
721/721 [==============================] - 0s 26us/step - loss: 127.4131
Epoch 48/50
721/721 [==============================] - 0s 26us/step - loss: 126.5150
Epoch 49/50
721/721 [==============================] - 0s 28us/step - loss: 125.6429
Epoch 50/50
721/721 [==============================] - 0s 35us/step - loss: 124.7850
```

Out[87]:
```
<keras.callbacks.callbacks.History at 0x22f27f9f5f8>
```

Test the model

In [88]:
```
loss = model.evaluate(X_test_norm, y_test)
print(loss)
```

```
309/309 [==============================] - 0s 13us/step
122.12650978063688
```

In [89]:
```
y_pred = model.predict(X_test_norm)
```

Compute *Mean Squared Error*

In [90]:
```
mse = mean_squared_error(y_test, y_pred)
print(mse)
```

```
122.12651083083885
```

50 mean squared errors

In [91]:
```
total_mean_squared_errors = 50
epochs = 50
mean_squared_errors = []
for i in range(0, total_mean_squared_errors):
    X_train_norm, X_test_norm, y_train, y_test = train_test_split(X_norm, y, test_size=0.3, random_state=i)
    model.fit(X_train_norm, y_train, epochs=epochs, verbose=0)
    MSE = model.evaluate(X_test_norm, y_test, verbose=0)
    print("MSE "+str(i+1)+": "+str(MSE))
    y_pred = model.predict(X_test_norm)
    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) + " mean squared errors without normalized data. Total number of epochs for each training is: " +str(epochs) + "\n")
print("Mean: "+str(mean))
print("Standard Deviation: "+str(standard_deviation))
```

```
MSE 1: 97.14688730085552
MSE 2: 95.49397070431014
MSE 3: 74.22389501429684
MSE 4: 58.423432100166394
MSE 5: 49.335282952268535
MSE 6: 49.46778875801556
MSE 7: 46.565087327679386
MSE 8: 34.856987011857015
MSE 9: 39.247259775797524
MSE 10: 39.2620041486129
MSE 11: 39.078239181666696
MSE 12: 34.50168552213502
MSE 13: 43.87015635913244
MSE 14: 43.03107427701981
MSE 15: 34.300878629715314
MSE 16: 31.4853489329514
MSE 17: 35.7773354230961
MSE 18: 35.18904015007143
MSE 19: 33.74489973741056
MSE 20: 34.739442066081516
MSE 21: 31.883312533974262
MSE 22: 32.70110353142698
MSE 23: 30.98669460753407
MSE 24: 33.70213484532625
MSE 25: 33.96202463316686
MSE 26: 35.743776772014535
MSE 27: 31.607845047145215
MSE 28: 30.672679117196587
MSE 29: 37.331322580479494
MSE 30: 33.50381959217652
MSE 31: 31.04189567195559
MSE 32: 31.908509121743606
MSE 33: 31.915014310176318
MSE 34: 33.28810963430065
MSE 35: 34.27111581845577
MSE 36: 39.25434849563154
MSE 37: 28.610294780299117
MSE 38: 34.999076904988215
MSE 39: 31.66178716270669
MSE 40: 27.831065995793512
MSE 41: 33.8682835156092
MSE 42: 28.039523072227304
MSE 43: 32.861701064125235
MSE 44: 35.174819612966
MSE 45: 34.077560967997826
MSE 46: 32.033694011108004
MSE 47: 30.902042413606612
MSE 48: 33.327923512381645
MSE 49: 31.499493836584985
MSE 50: 31.58175261661073


Below is the mean and standard deviation of 50 mean squared errors without no
rmalized data. Total number of epochs for each training is: 50

Mean: 38.59966807664984
Standard Deviation: 14.184869239528343
```

Comparision of the mean of the mean squared errors between step *A* & step *B*

In [92]:
```
print("Step A : mean of the MSEs = 77.02" "\n" "Step B : mean of the MSEs = 3
8.59")
```

```
Step A : mean of the MSEs = 77.02
Step B : mean of the MSEs = 38.59
```

Comment

mean of MSEs decrease after normalizing the predictors. Recommended

C. Increase the number of epochs

- Repeat Part B but using 100 epochs this time for training
- How does the mean of the mean squared errors compare to that from Step B ?

***Train* the model**

In [94]:
```
model.fit(X_train_norm, y_train, epochs= 100, verbose=1)
```

```
Epoch 1/100
721/721 [==============================] - 0s 26us/step - loss: 27.2428
Epoch 2/100
721/721 [==============================] - 0s 25us/step - loss: 27.2223
Epoch 3/100
721/721 [==============================] - 0s 21us/step - loss: 27.2984
Epoch 4/100
721/721 [==============================] - 0s 29us/step - loss: 27.2257
Epoch 5/100
721/721 [==============================] - 0s 36us/step - loss: 27.2511
Epoch 6/100
721/721 [==============================] - 0s 33us/step - loss: 27.2308
Epoch 7/100
721/721 [==============================] - 0s 36us/step - loss: 27.2129
Epoch 8/100
721/721 [==============================] - 0s 32us/step - loss: 27.2340
Epoch 9/100
721/721 [==============================] - 0s 35us/step - loss: 27.2606
Epoch 10/100
721/721 [==============================] - 0s 31us/step - loss: 27.1987
Epoch 11/100
721/721 [==============================] - 0s 33us/step - loss: 27.2377
Epoch 12/100
721/721 [==============================] - 0s 32us/step - loss: 27.2096
Epoch 13/100
721/721 [==============================] - 0s 32us/step - loss: 27.1989
Epoch 14/100
721/721 [==============================] - 0s 32us/step - loss: 27.2676
Epoch 15/100
721/721 [==============================] - 0s 35us/step - loss: 27.2578
Epoch 16/100
721/721 [==============================] - 0s 32us/step - loss: 27.2152
Epoch 17/100
721/721 [==============================] - 0s 29us/step - loss: 27.1899
Epoch 18/100
721/721 [==============================] - 0s 29us/step - loss: 27.2091
Epoch 19/100
721/721 [==============================] - 0s 26us/step - loss: 27.2732
Epoch 20/100
721/721 [==============================] - 0s 22us/step - loss: 27.2031
Epoch 21/100
721/721 [==============================] - 0s 28us/step - loss: 27.1974
Epoch 22/100
721/721 [==============================] - 0s 29us/step - loss: 27.1896
Epoch 23/100
721/721 [==============================] - 0s 26us/step - loss: 27.2041
Epoch 24/100
721/721 [==============================] - 0s 26us/step - loss: 27.2362
Epoch 25/100
721/721 [==============================] - 0s 24us/step - loss: 27.1838
Epoch 26/100
721/721 [==============================] - 0s 25us/step - loss: 27.2576
Epoch 27/100
721/721 [==============================] - 0s 26us/step - loss: 27.1960
Epoch 28/100
721/721 [==============================] - 0s 32us/step - loss: 27.2352
Epoch 29/100
```

```
721/721 [==============================] - 0s 26us/step - loss: 27.1741
Epoch 30/100
721/721 [==============================] - 0s 25us/step - loss: 27.1945
Epoch 31/100
721/721 [==============================] - 0s 25us/step - loss: 27.1933
Epoch 32/100
721/721 [==============================] - 0s 36us/step - loss: 27.1749
Epoch 33/100
721/721 [==============================] - 0s 26us/step - loss: 27.1740
Epoch 34/100
721/721 [==============================] - 0s 29us/step - loss: 27.2309
Epoch 35/100
721/721 [==============================] - 0s 37us/step - loss: 27.1478
Epoch 36/100
721/721 [==============================] - 0s 31us/step - loss: 27.1707
Epoch 37/100
721/721 [==============================] - 0s 33us/step - loss: 27.1900
Epoch 38/100
721/721 [==============================] - 0s 31us/step - loss: 27.1729
Epoch 39/100
721/721 [==============================] - 0s 35us/step - loss: 27.2278
Epoch 40/100
721/721 [==============================] - 0s 28us/step - loss: 27.2039
Epoch 41/100
721/721 [==============================] - 0s 29us/step - loss: 27.1884
Epoch 42/100
721/721 [==============================] - 0s 31us/step - loss: 27.1894
Epoch 43/100
721/721 [==============================] - 0s 32us/step - loss: 27.1664
Epoch 44/100
721/721 [==============================] - 0s 32us/step - loss: 27.1423
Epoch 45/100
721/721 [==============================] - 0s 31us/step - loss: 27.2116
Epoch 46/100
721/721 [==============================] - 0s 24us/step - loss: 27.1341
Epoch 47/100
721/721 [==============================] - 0s 32us/step - loss: 27.1884
Epoch 48/100
721/721 [==============================] - 0s 24us/step - loss: 27.1570
Epoch 49/100
721/721 [==============================] - 0s 28us/step - loss: 27.1581
Epoch 50/100
721/721 [==============================] - 0s 29us/step - loss: 27.1430
Epoch 51/100
721/721 [==============================] - 0s 33us/step - loss: 27.1792
Epoch 52/100
721/721 [==============================] - 0s 26us/step - loss: 27.2140
Epoch 53/100
721/721 [==============================] - 0s 25us/step - loss: 27.1799
Epoch 54/100
721/721 [==============================] - 0s 32us/step - loss: 27.2320
Epoch 55/100
721/721 [==============================] - 0s 22us/step - loss: 27.1879
Epoch 56/100
721/721 [==============================] - 0s 28us/step - loss: 27.2430
Epoch 57/100
721/721 [==============================] - 0s 25us/step - loss: 27.2362
```

```
Epoch 58/100
721/721 [==============================] - 0s 28us/step - loss: 27.1555
Epoch 59/100
721/721 [==============================] - 0s 33us/step - loss: 27.1624
Epoch 60/100
721/721 [==============================] - 0s 31us/step - loss: 27.1572
Epoch 61/100
721/721 [==============================] - 0s 26us/step - loss: 27.1578
Epoch 62/100
721/721 [==============================] - 0s 31us/step - loss: 27.1302
Epoch 63/100
721/721 [==============================] - 0s 24us/step - loss: 27.1561
Epoch 64/100
721/721 [==============================] - 0s 25us/step - loss: 27.1348
Epoch 65/100
721/721 [==============================] - 0s 26us/step - loss: 27.1236
Epoch 66/100
721/721 [==============================] - 0s 25us/step - loss: 27.1172
Epoch 67/100
721/721 [==============================] - 0s 26us/step - loss: 27.1669
Epoch 68/100
721/721 [==============================] - 0s 30us/step - loss: 27.2712
Epoch 69/100
721/721 [==============================] - 0s 25us/step - loss: 27.0843
Epoch 70/100
721/721 [==============================] - 0s 24us/step - loss: 27.1922
Epoch 71/100
721/721 [==============================] - 0s 24us/step - loss: 27.1729
Epoch 72/100
721/721 [==============================] - 0s 24us/step - loss: 27.1192
Epoch 73/100
721/721 [==============================] - 0s 26us/step - loss: 27.1522
Epoch 74/100
721/721 [==============================] - 0s 24us/step - loss: 27.1755
Epoch 75/100
721/721 [==============================] - 0s 28us/step - loss: 27.1403
Epoch 76/100
721/721 [==============================] - 0s 22us/step - loss: 27.1711
Epoch 77/100
721/721 [==============================] - 0s 26us/step - loss: 27.1140
Epoch 78/100
721/721 [==============================] - 0s 21us/step - loss: 27.1276
Epoch 79/100
721/721 [==============================] - 0s 25us/step - loss: 27.1383
Epoch 80/100
721/721 [==============================] - 0s 19us/step - loss: 27.1135
Epoch 81/100
721/721 [==============================] - 0s 22us/step - loss: 27.1183
Epoch 82/100
721/721 [==============================] - 0s 24us/step - loss: 27.1126
Epoch 83/100
721/721 [==============================] - 0s 25us/step - loss: 27.1420
Epoch 84/100
721/721 [==============================] - 0s 25us/step - loss: 27.1065
Epoch 85/100
721/721 [==============================] - 0s 25us/step - loss: 27.1260
Epoch 86/100
```

```
721/721 [==============================] - 0s 31us/step - loss: 27.1371
Epoch 87/100
721/721 [==============================] - 0s 28us/step - loss: 27.1296
Epoch 88/100
721/721 [==============================] - 0s 39us/step - loss: 27.1281
Epoch 89/100
721/721 [==============================] - 0s 29us/step - loss: 27.1134
Epoch 90/100
721/721 [==============================] - 0s 24us/step - loss: 27.1461
Epoch 91/100
721/721 [==============================] - 0s 26us/step - loss: 27.1288
Epoch 92/100
721/721 [==============================] - 0s 28us/step - loss: 27.1014
Epoch 93/100
721/721 [==============================] - 0s 25us/step - loss: 27.1018
Epoch 94/100
721/721 [==============================] - 0s 25us/step - loss: 27.1308
Epoch 95/100
721/721 [==============================] - 0s 28us/step - loss: 27.1299
Epoch 96/100
721/721 [==============================] - 0s 25us/step - loss: 27.0983
Epoch 97/100
721/721 [==============================] - 0s 22us/step - loss: 27.0953
Epoch 98/100
721/721 [==============================] - 0s 29us/step - loss: 27.0919
Epoch 99/100
721/721 [==============================] - 0s 33us/step - loss: 27.2286
Epoch 100/100
721/721 [==============================] - 0s 22us/step - loss: 27.1099
```

Out[94]:
```
<keras.callbacks.callbacks.History at 0x22f27fae4a8>
```

Test **the model**

In [95]:
```python
loss = model.evaluate(X_test_norm, y_test)
print(loss)
```

```
309/309 [==============================] - 0s 16us/step
32.45522461431312
```

In [96]:
```python
y_pred = model.predict(X_test_norm)
```

Compute *Mean Squared Error*

In [97]:
```python
mse = mean_squared_error(y_test, y_pred)
print(mse)
```

```
32.45522401517247
```

50 **mean squared errors**

In [98]:
```
total_mean_squared_errors = 50
epochs = 100
mean_squared_errors = []
for i in range(0, total_mean_squared_errors):
    X_train_norm, X_test_norm, y_train, y_test = train_test_split(X_norm, y, test_size=0.3, random_state=i)
    model.fit(X_train_norm, y_train, epochs=epochs, verbose=0)
    MSE = model.evaluate(X_test_norm, y_test, verbose=0)
    print("MSE "+str(i+1)+": "+str(MSE))
    y_pred = model.predict(X_test_norm)
    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) + " mean squared errors without normalized data. Total number of epochs for each training is: " +str(epochs) + "\n")
print("Mean: "+str(mean))
print("Standard Deviation: "+str(standard_deviation))
```

```
MSE 1: 32.96490092107779
MSE 2: 34.723996881528194
MSE 3: 27.341834947900864
MSE 4: 29.071035798699338
MSE 5: 30.246378667146256
MSE 6: 33.69630527187705
MSE 7: 34.6087549697234
MSE 8: 27.587113223029572
MSE 9: 29.868241739118755
MSE 10: 31.183269476041826
MSE 11: 31.395588624824597
MSE 12: 26.12258835209226
MSE 13: 32.08344797177608
MSE 14: 31.997851004492503
MSE 15: 29.017904608766624
MSE 16: 24.271496923996022
MSE 17: 30.447323530623056
MSE 18: 28.8505007796303
MSE 19: 27.528655326867952
MSE 20: 31.043848216726555
MSE 21: 26.137320521579976
MSE 22: 27.633374229603987
MSE 23: 24.555370843140437
MSE 24: 25.095071416070933
MSE 25: 28.781182952683334
MSE 26: 29.89794867555686
MSE 27: 25.955003954446045
MSE 28: 26.000866109113478
MSE 29: 29.542306338313328
MSE 30: 27.854482496440603
MSE 31: 27.515337626139324
MSE 32: 25.24206582165073
MSE 33: 25.5530434889315
MSE 34: 27.356915168391847
MSE 35: 31.402264962304372
MSE 36: 33.54835247299046
MSE 37: 23.497486274990834
MSE 38: 28.73989382993828
MSE 39: 26.899320065396502
MSE 40: 22.24800235785327
MSE 41: 30.41959178023354
MSE 42: 22.848416640148965
MSE 43: 26.499404678838538
MSE 44: 31.246713446953535
MSE 45: 27.506142711948037
MSE 46: 27.88488846072101
MSE 47: 26.116463275403266
MSE 48: 27.300843988807458
MSE 49: 26.51362936396429
MSE 50: 27.276121417295585


Below is the mean and standard deviation of 50 mean squared errors without no
rmalized data. Total number of epochs for each training is: 100

Mean: 28.422376887687168
Standard Deviation: 2.96299553887173
```

Comparision of the mean of the mean squared errors between step *B* & step *C*

In [99]:
```
print("Step B : mean of the MSEs = 38.59" "\n" "Step C : mean of the MSEs = 2
8.42")
```

```
Step B : mean of the MSEs = 38.59
Step C : mean of the MSEs = 28.42
```

Comment

mean of MSEs decreased after increasing the epochs. there should be a trade-off between the number of epochs and the computational power and time

D. Increase the number of hidden layers

Repeat part B but use a neural network with the following instead:

- Three hidden layers, each of 10 nodes and ReLU activation function
- How does the mean of the mean squared errors compare to that from Step B ?

keras Regression model with *three hidden layers, each of 10 nodes and ReLU activation function*

In [105]:
```
def regression_model_2():
    # create model
    model = Sequential()
    model.add(Dense(10, activation='relu', input_shape=(n_cols,)))
    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
```

Train the model

In [106]:
```
model = regression_model_2()
```

In [107]:

```
model.fit(X_train_norm, y_train, epochs=50, verbose=1)
```

```
Epoch 1/50
721/721 [==============================] - 0s 133us/step - loss: 1588.1386
Epoch 2/50
721/721 [==============================] - 0s 26us/step - loss: 1571.3953
Epoch 3/50
721/721 [==============================] - 0s 36us/step - loss: 1559.3274
Epoch 4/50
721/721 [==============================] - 0s 32us/step - loss: 1547.2591
Epoch 5/50
721/721 [==============================] - 0s 39us/step - loss: 1532.1800
Epoch 6/50
721/721 [==============================] - 0s 36us/step - loss: 1508.0190
Epoch 7/50
721/721 [==============================] - 0s 33us/step - loss: 1462.5932
Epoch 8/50
721/721 [==============================] - 0s 36us/step - loss: 1387.2158
Epoch 9/50
721/721 [==============================] - 0s 33us/step - loss: 1266.3767
Epoch 10/50
721/721 [==============================] - 0s 36us/step - loss: 1091.8466
Epoch 11/50
721/721 [==============================] - 0s 37us/step - loss: 861.8500
Epoch 12/50
721/721 [==============================] - 0s 37us/step - loss: 614.2162
Epoch 13/50
721/721 [==============================] - 0s 37us/step - loss: 400.2724
Epoch 14/50
721/721 [==============================] - 0s 35us/step - loss: 278.6959
Epoch 15/50
721/721 [==============================] - 0s 31us/step - loss: 228.7089
Epoch 16/50
721/721 [==============================] - 0s 32us/step - loss: 207.5212
Epoch 17/50
721/721 [==============================] - 0s 32us/step - loss: 195.3314
Epoch 18/50
721/721 [==============================] - 0s 29us/step - loss: 186.8565
Epoch 19/50
721/721 [==============================] - 0s 32us/step - loss: 180.3127
Epoch 20/50
721/721 [==============================] - 0s 32us/step - loss: 175.1526
Epoch 21/50
721/721 [==============================] - 0s 39us/step - loss: 170.8906
Epoch 22/50
721/721 [==============================] - 0s 42us/step - loss: 167.4735
Epoch 23/50
721/721 [==============================] - 0s 32us/step - loss: 164.2526
Epoch 24/50
721/721 [==============================] - 0s 35us/step - loss: 161.4508
Epoch 25/50
721/721 [==============================] - 0s 33us/step - loss: 158.8502
Epoch 26/50
721/721 [==============================] - 0s 42us/step - loss: 156.5384
Epoch 27/50
721/721 [==============================] - 0s 36us/step - loss: 154.3407
Epoch 28/50
721/721 [==============================] - 0s 26us/step - loss: 152.4201
Epoch 29/50
```

```
721/721 [==============================] - 0s 39us/step - loss: 150.5059
Epoch 30/50
721/721 [==============================] - 0s 33us/step - loss: 148.9757
Epoch 31/50
721/721 [==============================] - ETA: 0s - loss: 106.647 - 0s 31us/
step - loss: 147.5803
Epoch 32/50
721/721 [==============================] - 0s 22us/step - loss: 146.5333
Epoch 33/50
721/721 [==============================] - 0s 28us/step - loss: 144.7944
Epoch 34/50
721/721 [==============================] - 0s 32us/step - loss: 143.8503
Epoch 35/50
721/721 [==============================] - 0s 36us/step - loss: 142.4001
Epoch 36/50
721/721 [==============================] - 0s 35us/step - loss: 141.3185
Epoch 37/50
721/721 [==============================] - 0s 30us/step - loss: 140.5111
Epoch 38/50
721/721 [==============================] - 0s 40us/step - loss: 139.2789
Epoch 39/50
721/721 [==============================] - 0s 32us/step - loss: 138.3027
Epoch 40/50
721/721 [==============================] - 0s 32us/step - loss: 137.5672
Epoch 41/50
721/721 [==============================] - 0s 31us/step - loss: 136.7856
Epoch 42/50
721/721 [==============================] - 0s 36us/step - loss: 135.9902
Epoch 43/50
721/721 [==============================] - 0s 32us/step - loss: 134.9311
Epoch 44/50
721/721 [==============================] - 0s 31us/step - loss: 134.3793
Epoch 45/50
721/721 [==============================] - 0s 32us/step - loss: 133.6648
Epoch 46/50
721/721 [==============================] - 0s 28us/step - loss: 133.0945
Epoch 47/50
721/721 [==============================] - 0s 33us/step - loss: 131.8132
Epoch 48/50
721/721 [==============================] - 0s 31us/step - loss: 131.6402
Epoch 49/50
721/721 [==============================] - 0s 39us/step - loss: 130.6559
Epoch 50/50
721/721 [==============================] - 0s 31us/step - loss: 130.1142
```

Out[107]:
```
<keras.callbacks.callbacks.History at 0x22f293e95f8>
```

Test **the model**

In [108]:
```
loss = model.evaluate(X_test_norm, y_test)
print(loss)
```

```
309/309 [==============================] - 0s 45us/step
140.34336499951803
```

In [109]:
```
y_pred = model.predict(X_test_norm)
```

Compute *Mean Squared Error*

In [110]:
```
mse = mean_squared_error(y_test, y_pred)
print(mse)
```

```
140.34336752643915
```

50 mean squared errors

In [111]:
```
total_mean_squared_errors = 50
epochs = 50
mean_squared_errors = []
for i in range(0, total_mean_squared_errors):
    X_train_norm, X_test_norm, y_train, y_test = train_test_split(X_norm, y, test_size=0.3, random_state=i)
    model.fit(X_train_norm, y_train, epochs=epochs, verbose=0)
    MSE = model.evaluate(X_test_norm, y_test, verbose=0)
    print("MSE "+str(i+1)+": "+str(MSE))
    y_pred = model.predict(X_test_norm)
    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) + " mean squared errors without normalized data. Total number of epochs for each training is: " +str(epochs) + "\n")
print("Mean: "+str(mean))
print("Standard Deviation: "+str(standard_deviation))
```

```
MSE 1: 108.3339368949816
MSE 2: 119.47230445837126
MSE 3: 103.06900654024291
MSE 4: 102.02363300014854
MSE 5: 49.84163405285684
MSE 6: 42.689383954291976
MSE 7: 50.75440423465469
MSE 8: 40.9958013269122
MSE 9: 39.51263657356929
MSE 10: 40.01421743843548
MSE 11: 35.794965558839074
MSE 12: 33.79475678749455
MSE 13: 41.84878355168216
MSE 14: 41.279495449127886
MSE 15: 32.275919300067
MSE 16: 27.396210438996842
MSE 17: 34.456050304918996
MSE 18: 31.689784695029644
MSE 19: 28.529930873981957
MSE 20: 33.36332752017913
MSE 21: 30.458606861941636
MSE 22: 31.06087584017164
MSE 23: 24.629476429960874
MSE 24: 26.309867278657684
MSE 25: 33.23320032632081
MSE 26: 30.282425506986847
MSE 27: 24.25221200591152
MSE 28: 28.8939610669528
MSE 29: 30.13639187118382
MSE 30: 28.231588882150003
MSE 31: 24.664885431431642
MSE 32: 23.4186625187451
MSE 33: 22.664435500851727
MSE 34: 27.089128648578928
MSE 35: 27.45562665130714
MSE 36: 33.30593000492232
MSE 37: 23.507107549500695
MSE 38: 27.597612806894247
MSE 39: 26.652400772934207
MSE 40: 21.66355999079337
MSE 41: 28.743020968143995
MSE 42: 23.488422936991967
MSE 43: 25.393616160914352
MSE 44: 30.68856657207205
MSE 45: 26.39765747465362
MSE 46: 27.169090666045648
MSE 47: 25.21272208698359
MSE 48: 28.5949825854749
MSE 49: 28.341461811250852
MSE 50: 24.487619893836357


Below is the mean and standard deviation of 50 mean squared errors without no
rmalized data. Total number of epochs for each training is: 50

Mean: 37.02322556529744
Standard Deviation: 22.081262027224998
```

Comparision of the mean of the mean squared errors between step *B* & step *D*

In [112]:
```
print("Step B : mean of the MSEs = 38.59" "\n" "Step D : mean of the MSEs = 3
7.02")
```

```
Step B : mean of the MSEs = 38.59
Step D : mean of the MSEs = 37.02
```

Comment

mean of the MSEs slightly decreased after adding two additional hidden layers. for this case increasing hidden layers would not yield to a better model