Coding Tutorial

July 30, 2021

```
In [2]: import tensorflow as tf
        print(tf.__version__)
```

```
2.0.0
```

1 Validation, regularisation and callbacks

Coding tutorials #### Section **??** #### Section **??** #### Section **??** #### Section **??**

Validation sets

Load the data

```
In [3]: # Load the diabetes dataset
        from sklearn.datasets import load_diabetes

        diabetes_dataset = load_diabetes()
        print(diabetes_dataset["DESCR"])
```

```
.. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

  :Number of Instances: 442

  :Number of Attributes: First 10 columns are numeric predictive values
```

```
  :Target: Column 11 is a quantitative measure of disease progression one year after baseline

  :Attribute Information:
      - Age
      - Sex
      - Body mass index
      - Average blood pressure
      - S1
      - S2
      - S3
      - S4
      - S5
      - S6

Note: Each of these 10 feature variables have been mean centered and scaled by the standard de

Source URL:
http://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regress
(http://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
```

In [4]:
```python
# Save the input and target variables
#print(diabetes_dataset.keys())

data = diabetes_dataset["data"]
targets = diabetes_dataset["target"]
```

In [5]:
```python
# Normalise the target data (this will make clearer training curves)
targets = (targets - targets.mean(axis=0)) / targets.std()
targets
```

Out[5]:
```
array([-1.47194752e-02, -1.00165882e+00, -1.44579915e-01,  6.99512942e-01,
       -2.22496178e-01, -7.15965848e-01, -1.83538046e-01, -1.15749134e+00,
       -5.47147277e-01,  2.05006151e+00, -6.64021672e-01, -1.07957508e+00,
        3.48889755e-01,  4.26806019e-01, -4.43258925e-01,  2.45001404e-01,
        1.80071184e-01, -1.05621783e-01, -7.15965848e-01,  2.06043272e-01,
       -1.09256112e+00, -1.33929596e+00, -1.09256112e+00,  1.20596866e+00,
        4.13819975e-01,  6.47568766e-01, -1.96524090e-01, -8.71798376e-01,
       -2.74440354e-01,  1.69943833e+00, -3.00412442e-01, -1.20943552e+00,
        2.45262887e+00, -8.45826288e-01, -1.13151925e+00, -6.51035629e-01,
        1.46568953e+00,  1.60853602e+00,  1.29687096e+00, -8.06868156e-01,
       -6.77007716e-01, -1.26137969e+00, -1.18346343e+00, -7.80896068e-01,
        1.38777327e+00, -1.28735178e+00,  4.91736239e-01, -1.31593871e-01,
       -1.00165882e+00, -1.31593871e-01,  3.72247006e-02,  9.46247777e-01,
```

```
-1.20943552e+00, -6.25063541e-01,  3.87847887e-01, -3.13398486e-01,
-1.30033783e+00, -1.49512849e+00,  2.32015360e-01,  2.32015360e-01,
-1.18346343e+00, -1.05621783e-01, -1.30033783e+00, -3.13398486e-01,
-1.05360299e+00,  1.41113052e-01, -2.77055191e-02, -7.15965848e-01,
 1.02154920e-01,  3.35903711e-01, -1.35228200e+00,  1.53061975e+00,
 6.47568766e-01, -5.34161233e-01, -8.71798376e-01, -1.43019827e+00,
 2.32015360e-01,  6.21596678e-01,  1.29687096e+00, -5.08189145e-01,
-1.18607827e-01, -1.31332387e+00, -1.30033783e+00,  7.51457118e-01,
-1.13151925e+00, -1.44579915e-01, -1.26137969e+00, -2.35482222e-01,
-1.43019827e+00, -5.34161233e-01, -7.02979804e-01,  1.54099096e-01,
-1.35228200e+00, -7.28951892e-01, -8.06868156e-01,  1.28127008e-01,
-2.77055191e-02,  1.64749415e+00, -7.80896068e-01, -8.97770464e-01,
-3.13398486e-01, -6.51035629e-01,  1.94617316e+00,  5.95624590e-01,
-7.41937936e-01, -1.28735178e+00, -2.35482222e-01, -1.05621783e-01,
 1.03715008e+00, -9.23742551e-01, -6.25063541e-01, -1.20943552e+00,
 1.21895470e+00,  1.88124294e+00,  1.37478723e+00,  9.98191953e-01,
 1.59554997e+00,  1.67346624e+00,  3.48889755e-01,  6.21596678e-01,
 6.21596678e-01,  2.70973492e-01,  3.61875799e-01, -8.84784420e-01,
-4.04300794e-01,  1.15140964e-01, -6.89993760e-01, -5.60133321e-01,
-4.82217057e-01,  1.50464767e+00,  1.58256393e+00,  7.61828325e-02,
-5.86105409e-01, -8.97770464e-01, -6.38049585e-01,  1.55659184e+00,
-8.71798376e-01,  1.66048019e+00,  2.38769865e+00,  1.67346624e+00,
-4.43258925e-01,  2.14096382e+00,  1.07610822e+00, -1.19644947e+00,
 2.83959536e-01,  1.38777327e+00,  3.35903711e-01, -3.13398486e-01,
-7.28951892e-01, -3.39370574e-01,  1.76436855e+00, -8.32840244e-01,
 1.81631272e+00, -1.05360299e+00,  5.82638546e-01,  4.39792063e-01,
-1.65096101e+00, -8.84784420e-01, -7.28951892e-01,  5.56666458e-01,
-1.28735178e+00,  8.42359425e-01,  2.57987448e-01, -2.74440354e-01,
 8.03401293e-01, -1.20943552e+00, -1.06658903e+00,  8.81317557e-01,
 1.50464767e+00, -1.73343121e-03, -1.36526805e+00, -1.01464486e+00,
 1.85527085e+00, -6.64021672e-01, -1.47194752e-02, -3.26384530e-01,
 1.10208030e+00,  9.46247777e-01, -9.23742551e-01, -1.47194752e-02,
-5.86105409e-01, -1.14450530e+00, -1.83538046e-01,  4.26806019e-01,
 1.46568953e+00, -6.64021672e-01, -1.96524090e-01, -1.18607827e-01,
-1.44579915e-01, -9.49714639e-01,  1.81631272e+00,  3.35903711e-01,
-7.93882112e-01, -4.69231013e-01, -8.58812332e-01, -3.91314750e-01,
-1.04061695e+00, -3.00412442e-01, -1.31593871e-01, -8.06868156e-01,
 7.61828325e-02, -1.46915640e+00,  5.69652502e-01,  9.07289645e-01,
 1.62152206e+00, -6.89993760e-01,  5.69652502e-01,  6.47568766e-01,
 3.72247006e-02, -9.75686727e-01,  5.04722283e-01, -1.06658903e+00,
-1.02763090e+00, -1.33929596e+00, -1.13151925e+00,  1.43971745e+00,
 1.24492679e+00,  1.86825690e+00,  8.03401293e-01,  4.26806019e-01,
-9.62700683e-01, -7.67910024e-01,  1.29687096e+00, -2.77055191e-02,
-9.75686727e-01,  7.25485030e-01, -9.75686727e-01, -5.73119365e-01,
 1.02154920e-01, -1.28735178e+00,  8.81317557e-01,  2.42386567e-02,
 1.38777327e+00, -8.06868156e-01,  1.21895470e+00, -3.65342662e-01,
-1.10554717e+00, -1.04061695e+00,  1.36180118e+00,  1.42673140e+00,
 1.59554997e+00,  3.22917667e-01, -1.05360299e+00, -1.36526805e+00,
```

```
 4.52778107e-01, -3.52356618e-01, -9.62700683e-01, -1.31332387e+00,
 1.37478723e+00,  8.16387337e-01,  1.95915920e+00,  1.17999657e+00,
-7.93882112e-01, -2.77055191e-02,  2.05006151e+00,  1.12526127e-02,
 2.51755909e+00, -1.15749134e+00, -8.19854200e-01, -1.32630991e+00,
-1.46915640e+00, -6.38049585e-01,  2.02408942e+00, -4.69231013e-01,
-9.26357388e-02, -1.01464486e+00, -1.39124013e+00, -4.82217057e-01,
 1.45270349e+00, -8.45826288e-01,  6.47568766e-01, -3.26384530e-01,
 3.87847887e-01,  1.15402448e+00, -1.11853321e+00, -7.54923980e-01,
 1.69943833e+00, -1.14450530e+00, -6.51035629e-01,  6.21596678e-01,
 1.46568953e+00, -7.54923980e-01,  1.01117800e+00,  3.74861843e-01,
 5.02107446e-02,  1.05013613e+00, -1.19644947e+00,  8.68331513e-01,
-9.36728595e-01, -1.09256112e+00,  2.33575448e+00,  1.24492679e+00,
-8.84784420e-01,  6.21596678e-01, -1.26137969e+00, -8.71798376e-01,
-8.19854200e-01, -1.57304475e+00, -3.00412442e-01, -8.97770464e-01,
 1.59554997e+00, -1.13151925e+00,  5.95624590e-01,  1.08909426e+00,
 1.30985701e+00, -3.65342662e-01, -1.40422618e+00,  2.57987448e-01,
-4.95203101e-01, -1.31593871e-01, -5.60133321e-01,  3.61875799e-01,
-1.05621783e-01,  1.41113052e-01, -6.66636509e-02, -7.15965848e-01,
 8.81317557e-01,  4.91736239e-01, -5.60133321e-01,  5.04722283e-01,
-3.91314750e-01,  1.01117800e+00,  1.16701052e+00,  1.24492679e+00,
 1.25791283e+00,  5.17708327e-01, -2.74440354e-01,  1.10208030e+00,
-9.62700683e-01, -2.22496178e-01,  1.19298261e+00,  6.08610634e-01,
 1.53061975e+00,  1.54099096e-01, -1.04061695e+00, -7.28951892e-01,
 1.99811734e+00, -7.93882112e-01,  8.03401293e-01, -7.41937936e-01,
 8.29373381e-01,  1.43971745e+00,  3.35903711e-01, -5.08189145e-01,
 6.21596678e-01, -1.70552003e-01, -1.70552003e-01, -8.32840244e-01,
-5.36776070e-02, -8.32840244e-01,  1.17999657e+00, -1.05360299e+00,
-9.75686727e-01, -5.60133321e-01,  1.55659184e+00, -1.19644947e+00,
-1.27436574e+00,  8.94303601e-01, -8.06868156e-01,  2.06304756e+00,
 1.67346624e+00,  3.87847887e-01,  2.19290800e+00, -1.22242156e+00,
 1.42673140e+00,  6.99512942e-01,  1.05013613e+00,  1.16701052e+00,
-3.78328706e-01,  1.93057228e-01, -1.15749134e+00,  5.82638546e-01,
-1.05360299e+00,  2.06043272e-01, -1.57565959e-01,  8.42359425e-01,
-4.04300794e-01,  1.07610822e+00,  1.20596866e+00, -1.45617035e+00,
-1.30033783e+00, -6.25063541e-01, -2.61454310e-01, -8.32840244e-01,
-1.07957508e+00,  8.68331513e-01, -1.04061695e+00,  6.34582722e-01,
-5.47147277e-01, -1.31332387e+00,  1.62152206e+00, -1.15749134e+00,
-4.43258925e-01, -1.07957508e+00,  1.56957789e+00,  1.37478723e+00,
-1.41721222e+00,  5.95624590e-01,  1.16701052e+00,  1.03715008e+00,
 2.96945580e-01, -7.67910024e-01,  2.06043272e-01,  1.59554997e+00,
 1.82929877e+00,  1.67346624e+00, -1.04061695e+00, -1.57565959e-01,
 4.78750195e-01,  3.74861843e-01,  7.38471074e-01, -2.09510134e-01,
 1.41374536e+00, -5.08189145e-01, -2.74440354e-01,  2.83959536e-01,
 1.36180118e+00, -1.26137969e+00, -8.84784420e-01, -1.43019827e+00,
-7.96496949e-02,  7.77429206e-01,  1.05013613e+00, -7.93882112e-01,
-5.34161233e-01, -1.73343121e-03, -4.17286837e-01, -1.10554717e+00,
 2.05006151e+00, -7.54923980e-01,  4.00833931e-01, -1.11853321e+00,
 2.70973492e-01, -1.04061695e+00, -1.33929596e+00, -1.14450530e+00,
```

```
-1.35228200e+00,  3.35903711e-01, -6.25063541e-01, -2.61454310e-01,
 8.81317557e-01, -1.23540761e+00])
```

```
In [6]: # Split the data into train and test sets
        from sklearn.model_selection import train_test_split

        train_data, test_data, train_targets, test_targets = train_test_split(data, targets, te

        print(train_data.shape)
        print(test_data.shape)
        print(train_targets.shape)
        print(test_targets.shape)
```

```
(397, 10)
(45, 10)
(397,)
(45,)
```

Train a feedforward neural network model

```
In [7]: # Build the model

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

        def get_model():
            model = Sequential([
                Dense(128, activation='relu', input_shape=(train_data.shape[1],)),
                Dense(128, activation='relu'),
                Dense(128, activation='relu'),
                Dense(128, activation='relu'),
                Dense(128, activation='relu'),
                Dense(128, activation='relu'),
                Dense(1)
            ])
            return(model)
        model = get_model()
```

```
In [8]: # Print the model summary

        model.summary()
```

```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               1408
_________________________________________________________________
```

```
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_2 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_3 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_4 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_5 (Dense)              (None, 128)               16512
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 129
=================================================================
Total params: 84,097
Trainable params: 84,097
Non-trainable params: 0
_________________________________________________________________
```

In [9]:
```
# Compile the model

model.compile(oprimizer='adam', loss='mse', metrics=['mae'])
```

In [10]:
```
# Train the model, with some of the data reserved for validation

history = model.fit(train_data, train_targets, epochs=100,
                    validation_split=0.15, batch_size=64, verbose=2)
```

```
Train on 337 samples, validate on 60 samples
Epoch 1/100
337/337 - 2s - loss: 0.9861 - mae: 0.8521 - val_loss: 0.7935 - val_mae: 0.7539
Epoch 2/100
337/337 - 0s - loss: 0.6922 - mae: 0.7024 - val_loss: 0.5849 - val_mae: 0.6003
Epoch 3/100
337/337 - 0s - loss: 0.5916 - mae: 0.6490 - val_loss: 0.5605 - val_mae: 0.5538
Epoch 4/100
337/337 - 0s - loss: 0.5141 - mae: 0.5982 - val_loss: 0.5749 - val_mae: 0.5621
Epoch 5/100
337/337 - 0s - loss: 0.4764 - mae: 0.5764 - val_loss: 0.5517 - val_mae: 0.5517
Epoch 6/100
337/337 - 0s - loss: 0.4605 - mae: 0.5565 - val_loss: 0.7703 - val_mae: 0.7389
Epoch 7/100
337/337 - 0s - loss: 0.4952 - mae: 0.5912 - val_loss: 0.8240 - val_mae: 0.6668
Epoch 8/100
337/337 - 0s - loss: 0.4789 - mae: 0.5627 - val_loss: 0.5625 - val_mae: 0.5491
Epoch 9/100
337/337 - 0s - loss: 0.4419 - mae: 0.5393 - val_loss: 0.5905 - val_mae: 0.5490
Epoch 10/100
337/337 - 0s - loss: 0.4353 - mae: 0.5327 - val_loss: 0.6753 - val_mae: 0.6037
```

```
Epoch 11/100
337/337 - 0s - loss: 0.5149 - mae: 0.5756 - val_loss: 0.6128 - val_mae: 0.5795
Epoch 12/100
337/337 - 0s - loss: 0.4534 - mae: 0.5497 - val_loss: 0.6170 - val_mae: 0.5613
Epoch 13/100
337/337 - 0s - loss: 0.4295 - mae: 0.5347 - val_loss: 0.6215 - val_mae: 0.6037
Epoch 14/100
337/337 - 0s - loss: 0.4289 - mae: 0.5274 - val_loss: 0.7624 - val_mae: 0.6322
Epoch 15/100
337/337 - 0s - loss: 0.4386 - mae: 0.5300 - val_loss: 0.6021 - val_mae: 0.5967
Epoch 16/100
337/337 - 0s - loss: 0.4228 - mae: 0.5276 - val_loss: 0.6645 - val_mae: 0.6488
Epoch 17/100
337/337 - 0s - loss: 0.4425 - mae: 0.5433 - val_loss: 0.6182 - val_mae: 0.5734
Epoch 18/100
337/337 - 0s - loss: 0.3975 - mae: 0.5007 - val_loss: 0.5992 - val_mae: 0.5789
Epoch 19/100
337/337 - 0s - loss: 0.4599 - mae: 0.5497 - val_loss: 0.5791 - val_mae: 0.5763
Epoch 20/100
337/337 - 0s - loss: 0.3847 - mae: 0.5070 - val_loss: 0.7201 - val_mae: 0.6148
Epoch 21/100
337/337 - 0s - loss: 0.4205 - mae: 0.5216 - val_loss: 0.7097 - val_mae: 0.6618
Epoch 22/100
337/337 - 0s - loss: 0.4566 - mae: 0.5461 - val_loss: 0.6392 - val_mae: 0.6313
Epoch 23/100
337/337 - 0s - loss: 0.4141 - mae: 0.5207 - val_loss: 0.6076 - val_mae: 0.5753
Epoch 24/100
337/337 - 0s - loss: 0.3954 - mae: 0.5060 - val_loss: 0.7731 - val_mae: 0.6551
Epoch 25/100
337/337 - 0s - loss: 0.4132 - mae: 0.5179 - val_loss: 0.5840 - val_mae: 0.5578
Epoch 26/100
337/337 - 0s - loss: 0.3646 - mae: 0.4863 - val_loss: 0.5991 - val_mae: 0.5537
Epoch 27/100
337/337 - 0s - loss: 0.3781 - mae: 0.4984 - val_loss: 0.5946 - val_mae: 0.5689
Epoch 28/100
337/337 - 0s - loss: 0.3556 - mae: 0.4859 - val_loss: 0.8032 - val_mae: 0.6526
Epoch 29/100
337/337 - 0s - loss: 0.4583 - mae: 0.5390 - val_loss: 0.6805 - val_mae: 0.5861
Epoch 30/100
337/337 - 0s - loss: 0.3480 - mae: 0.4719 - val_loss: 0.6763 - val_mae: 0.6168
Epoch 31/100
337/337 - 0s - loss: 0.4203 - mae: 0.5060 - val_loss: 0.6361 - val_mae: 0.5907
Epoch 32/100
337/337 - 0s - loss: 0.3579 - mae: 0.4770 - val_loss: 0.6200 - val_mae: 0.5722
Epoch 33/100
337/337 - 0s - loss: 0.3668 - mae: 0.4921 - val_loss: 0.6291 - val_mae: 0.5625
Epoch 34/100
337/337 - 0s - loss: 0.3502 - mae: 0.4691 - val_loss: 0.6096 - val_mae: 0.5609
```

```
Epoch 35/100
337/337 - 0s - loss: 0.3480 - mae: 0.4698 - val_loss: 0.7659 - val_mae: 0.7064
Epoch 36/100
337/337 - 0s - loss: 0.3670 - mae: 0.4921 - val_loss: 0.6821 - val_mae: 0.5948
Epoch 37/100
337/337 - 0s - loss: 0.4243 - mae: 0.5127 - val_loss: 0.6540 - val_mae: 0.5911
Epoch 38/100
337/337 - 0s - loss: 0.3291 - mae: 0.4552 - val_loss: 0.7614 - val_mae: 0.6791
Epoch 39/100
337/337 - 0s - loss: 0.3979 - mae: 0.5004 - val_loss: 0.6655 - val_mae: 0.5765
Epoch 40/100
337/337 - 0s - loss: 0.3249 - mae: 0.4446 - val_loss: 0.6786 - val_mae: 0.5974
Epoch 41/100
337/337 - 0s - loss: 0.3410 - mae: 0.4666 - val_loss: 0.6321 - val_mae: 0.5780
Epoch 42/100
337/337 - 0s - loss: 0.3041 - mae: 0.4371 - val_loss: 0.6643 - val_mae: 0.5763
Epoch 43/100
337/337 - 0s - loss: 0.3231 - mae: 0.4513 - val_loss: 0.8680 - val_mae: 0.6834
Epoch 44/100
337/337 - 0s - loss: 0.4368 - mae: 0.5335 - val_loss: 0.6510 - val_mae: 0.5701
Epoch 45/100
337/337 - 0s - loss: 0.2909 - mae: 0.4275 - val_loss: 0.7128 - val_mae: 0.6041
Epoch 46/100
337/337 - 0s - loss: 0.3188 - mae: 0.4426 - val_loss: 0.8016 - val_mae: 0.7010
Epoch 47/100
337/337 - 0s - loss: 0.3544 - mae: 0.4753 - val_loss: 0.6331 - val_mae: 0.5824
Epoch 48/100
337/337 - 0s - loss: 0.2846 - mae: 0.4229 - val_loss: 0.6245 - val_mae: 0.5715
Epoch 49/100
337/337 - 0s - loss: 0.3161 - mae: 0.4476 - val_loss: 0.8396 - val_mae: 0.6739
Epoch 50/100
337/337 - 0s - loss: 0.3473 - mae: 0.4575 - val_loss: 0.9099 - val_mae: 0.7240
Epoch 51/100
337/337 - 0s - loss: 0.3794 - mae: 0.4757 - val_loss: 0.6404 - val_mae: 0.5741
Epoch 52/100
337/337 - 0s - loss: 0.2750 - mae: 0.4162 - val_loss: 0.6309 - val_mae: 0.5644
Epoch 53/100
337/337 - 0s - loss: 0.2958 - mae: 0.4327 - val_loss: 0.7191 - val_mae: 0.6067
Epoch 54/100
337/337 - 0s - loss: 0.2896 - mae: 0.4262 - val_loss: 0.7916 - val_mae: 0.6351
Epoch 55/100
337/337 - 0s - loss: 0.3179 - mae: 0.4488 - val_loss: 0.7168 - val_mae: 0.6119
Epoch 56/100
337/337 - 0s - loss: 0.3017 - mae: 0.4302 - val_loss: 0.6559 - val_mae: 0.5941
Epoch 57/100
337/337 - 0s - loss: 0.2678 - mae: 0.4112 - val_loss: 0.7547 - val_mae: 0.6534
Epoch 58/100
337/337 - 0s - loss: 0.2748 - mae: 0.4228 - val_loss: 0.6678 - val_mae: 0.6045
```

```
Epoch 59/100
337/337 - 0s - loss: 0.2615 - mae: 0.4058 - val_loss: 0.6124 - val_mae: 0.5514
Epoch 60/100
337/337 - 0s - loss: 0.2476 - mae: 0.3905 - val_loss: 0.6865 - val_mae: 0.5980
Epoch 61/100
337/337 - 0s - loss: 0.2720 - mae: 0.4095 - val_loss: 0.6317 - val_mae: 0.5794
Epoch 62/100
337/337 - 0s - loss: 0.3207 - mae: 0.4457 - val_loss: 0.6155 - val_mae: 0.5698
Epoch 63/100
337/337 - 0s - loss: 0.2413 - mae: 0.3935 - val_loss: 0.6885 - val_mae: 0.5992
Epoch 64/100
337/337 - 0s - loss: 0.2792 - mae: 0.4263 - val_loss: 0.7008 - val_mae: 0.5844
Epoch 65/100
337/337 - 0s - loss: 0.2255 - mae: 0.3788 - val_loss: 0.7391 - val_mae: 0.6236
Epoch 66/100
337/337 - 0s - loss: 0.3022 - mae: 0.4337 - val_loss: 0.9018 - val_mae: 0.7359
Epoch 67/100
337/337 - 0s - loss: 0.2506 - mae: 0.3917 - val_loss: 0.6836 - val_mae: 0.5812
Epoch 68/100
337/337 - 0s - loss: 0.2812 - mae: 0.4256 - val_loss: 0.6254 - val_mae: 0.5617
Epoch 69/100
337/337 - 0s - loss: 0.2064 - mae: 0.3578 - val_loss: 0.6923 - val_mae: 0.5844
Epoch 70/100
337/337 - 0s - loss: 0.2794 - mae: 0.4144 - val_loss: 0.7031 - val_mae: 0.6120
Epoch 71/100
337/337 - 0s - loss: 0.2077 - mae: 0.3583 - val_loss: 0.7144 - val_mae: 0.6374
Epoch 72/100
337/337 - 0s - loss: 0.2630 - mae: 0.4153 - val_loss: 0.6936 - val_mae: 0.6042
Epoch 73/100
337/337 - 0s - loss: 0.1977 - mae: 0.3479 - val_loss: 0.7663 - val_mae: 0.6352
Epoch 74/100
337/337 - 0s - loss: 0.2786 - mae: 0.4038 - val_loss: 0.6508 - val_mae: 0.6069
Epoch 75/100
337/337 - 0s - loss: 0.2206 - mae: 0.3703 - val_loss: 0.6863 - val_mae: 0.5950
Epoch 76/100
337/337 - 0s - loss: 0.1997 - mae: 0.3539 - val_loss: 0.9364 - val_mae: 0.7491
Epoch 77/100
337/337 - 0s - loss: 0.2762 - mae: 0.4053 - val_loss: 0.7861 - val_mae: 0.6508
Epoch 78/100
337/337 - 0s - loss: 0.2515 - mae: 0.3949 - val_loss: 0.7578 - val_mae: 0.6296
Epoch 79/100
337/337 - 0s - loss: 0.2117 - mae: 0.3585 - val_loss: 0.7137 - val_mae: 0.6126
Epoch 80/100
337/337 - 0s - loss: 0.1665 - mae: 0.3195 - val_loss: 0.8392 - val_mae: 0.6892
Epoch 81/100
337/337 - 0s - loss: 0.2948 - mae: 0.4344 - val_loss: 0.6805 - val_mae: 0.5907
Epoch 82/100
337/337 - 0s - loss: 0.1902 - mae: 0.3401 - val_loss: 0.9025 - val_mae: 0.7457
```

```
Epoch 83/100
337/337 - 0s - loss: 0.2146 - mae: 0.3638 - val_loss: 0.7230 - val_mae: 0.5968
Epoch 84/100
337/337 - 0s - loss: 0.1637 - mae: 0.3189 - val_loss: 0.7582 - val_mae: 0.6236
Epoch 85/100
337/337 - 0s - loss: 0.2716 - mae: 0.4158 - val_loss: 0.7303 - val_mae: 0.6213
Epoch 86/100
337/337 - 0s - loss: 0.1617 - mae: 0.3119 - val_loss: 0.7769 - val_mae: 0.6316
Epoch 87/100
337/337 - 0s - loss: 0.2587 - mae: 0.4008 - val_loss: 0.6916 - val_mae: 0.5878
Epoch 88/100
337/337 - 0s - loss: 0.1790 - mae: 0.3311 - val_loss: 0.7178 - val_mae: 0.6439
Epoch 89/100
337/337 - 0s - loss: 0.2791 - mae: 0.4220 - val_loss: 0.7194 - val_mae: 0.6142
Epoch 90/100
337/337 - 0s - loss: 0.1585 - mae: 0.3117 - val_loss: 0.7513 - val_mae: 0.6078
Epoch 91/100
337/337 - 0s - loss: 0.1734 - mae: 0.3301 - val_loss: 0.8502 - val_mae: 0.6634
Epoch 92/100
337/337 - 0s - loss: 0.2561 - mae: 0.3956 - val_loss: 0.7863 - val_mae: 0.6377
Epoch 93/100
337/337 - 0s - loss: 0.1665 - mae: 0.3145 - val_loss: 0.7715 - val_mae: 0.6176
Epoch 94/100
337/337 - 0s - loss: 0.1323 - mae: 0.2799 - val_loss: 0.8493 - val_mae: 0.6493
Epoch 95/100
337/337 - 0s - loss: 0.2796 - mae: 0.4152 - val_loss: 0.7121 - val_mae: 0.5901
Epoch 96/100
337/337 - 0s - loss: 0.1405 - mae: 0.2881 - val_loss: 1.1013 - val_mae: 0.8222
Epoch 97/100
337/337 - 0s - loss: 0.3155 - mae: 0.4459 - val_loss: 0.7720 - val_mae: 0.6186
Epoch 98/100
337/337 - 0s - loss: 0.1322 - mae: 0.2839 - val_loss: 0.7555 - val_mae: 0.6171
Epoch 99/100
337/337 - 0s - loss: 0.1243 - mae: 0.2705 - val_loss: 0.8348 - val_mae: 0.6669
Epoch 100/100
337/337 - 0s - loss: 0.2736 - mae: 0.4217 - val_loss: 0.7409 - val_mae: 0.6111
```

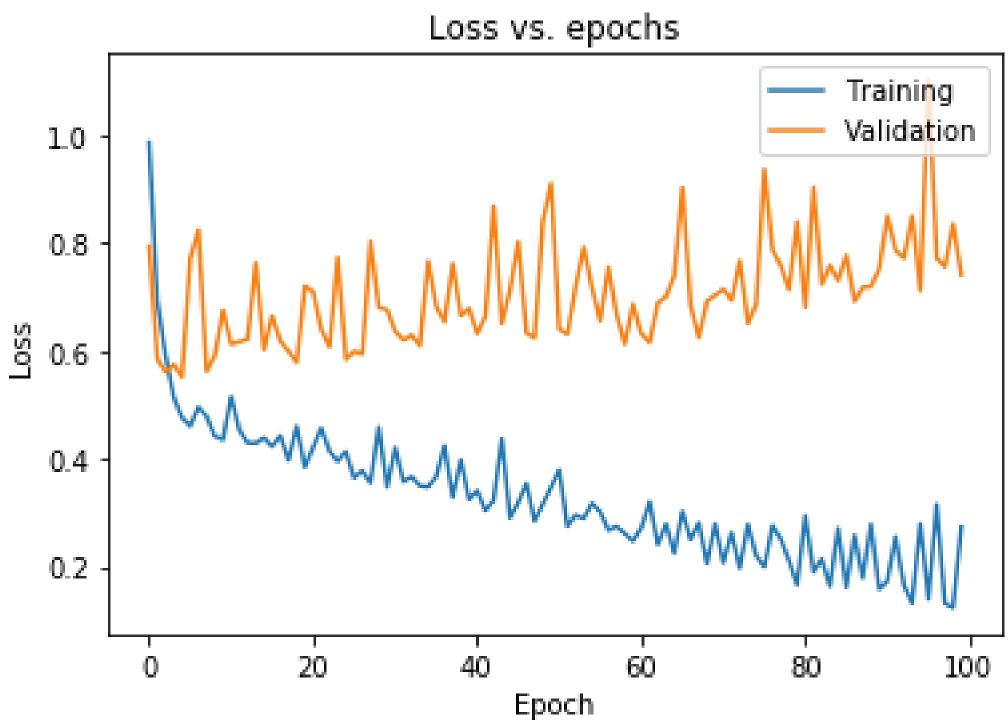
In [11]:
```
# Evaluate the model on the test set

model.evaluate(test_data, test_targets, verbose=2)
```

```
45/1 - 0s - loss: 0.7336 - mae: 0.6691
```

Out[11]:
```
[0.7819557428359986, 0.6691007]
```

Plot the learning curves

```
In [12]: import matplotlib.pyplot as plt
         %matplotlib inline

In [13]: # Plot the training and validation loss

         plt.plot(history.history['loss'])
         plt.plot(history.history['val_loss'])
         plt.title('Loss vs. epochs')
         plt.ylabel('Loss')
         plt.xlabel('Epoch')
         plt.legend(['Training', 'Validation'], loc='upper right')
         plt.show()
```



Model regularisation

Adding regularisation with weight decay and dropout

```
In [14]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense
         from tensorflow.keras.layers import Dropout
         from tensorflow.keras import regularizers
```

In [15]:
```
def get_regularised_model(wd, rate):
    model = Sequential([
        Dense(128, activation="relu", kernel_regularizer=regularizers.l2(wd), input_sl
        Dropout(rate),
        Dense(128, kernel_regularizer=regularizers.l2(wd), activation="relu"),
        Dropout(rate),
        Dense(128, kernel_regularizer=regularizers.l2(wd), activation="relu"),
        Dropout(rate),
        Dense(128, kernel_regularizer=regularizers.l2(wd), activation="relu"),
        Dropout(rate),
        Dense(128, kernel_regularizer=regularizers.l2(wd), activation="relu"),
        Dropout(rate),
        Dense(128, kernel_regularizer=regularizers.l2(wd), activation="relu"),
        Dropout(rate),
        Dense(1)
    ])
    return model
```

In [16]:
```
# Re-build the model with weight decay and dropout layers
model = get_regularised_model(1e-5, 0.3)
```

In [17]:
```
# Compile the model
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

In [18]:
```
# Train the model, with some of the data reserved for validation

history = model.fit(train_data, train_targets, epochs=100,
                    validation_split=0.15, batch_size=64, verbose=2)
```

```
Train on 337 samples, validate on 60 samples
Epoch 1/100
337/337 - 2s - loss: 1.0136 - mae: 0.8569 - val_loss: 0.9659 - val_mae: 0.8394
Epoch 2/100
337/337 - 0s - loss: 1.0065 - mae: 0.8565 - val_loss: 0.9572 - val_mae: 0.8366
Epoch 3/100
337/337 - 0s - loss: 0.9846 - mae: 0.8504 - val_loss: 0.9059 - val_mae: 0.8119
Epoch 4/100
337/337 - 0s - loss: 0.9198 - mae: 0.8224 - val_loss: 0.7885 - val_mae: 0.7474
Epoch 5/100
337/337 - 0s - loss: 0.7733 - mae: 0.7550 - val_loss: 0.6216 - val_mae: 0.6265
Epoch 6/100
337/337 - 0s - loss: 0.6528 - mae: 0.6882 - val_loss: 0.6139 - val_mae: 0.5705
Epoch 7/100
337/337 - 0s - loss: 0.6205 - mae: 0.6355 - val_loss: 0.5953 - val_mae: 0.5637
Epoch 8/100
337/337 - 0s - loss: 0.5944 - mae: 0.6499 - val_loss: 0.6029 - val_mae: 0.5973
Epoch 9/100
337/337 - 0s - loss: 0.6003 - mae: 0.6475 - val_loss: 0.6335 - val_mae: 0.6134
Epoch 10/100
```

```
337/337 - 0s - loss: 0.5550 - mae: 0.6258 - val_loss: 0.5990 - val_mae: 0.5732
Epoch 11/100
337/337 - 0s - loss: 0.5159 - mae: 0.5849 - val_loss: 0.6252 - val_mae: 0.5694
Epoch 12/100
337/337 - 0s - loss: 0.5124 - mae: 0.5723 - val_loss: 0.6357 - val_mae: 0.5775
Epoch 13/100
337/337 - 0s - loss: 0.5304 - mae: 0.5966 - val_loss: 0.6085 - val_mae: 0.5808
Epoch 14/100
337/337 - 0s - loss: 0.4898 - mae: 0.5739 - val_loss: 0.6236 - val_mae: 0.5704
Epoch 15/100
337/337 - 0s - loss: 0.5161 - mae: 0.5891 - val_loss: 0.6306 - val_mae: 0.5838
Epoch 16/100
337/337 - 0s - loss: 0.5463 - mae: 0.6153 - val_loss: 0.5975 - val_mae: 0.5689
Epoch 17/100
337/337 - 0s - loss: 0.5050 - mae: 0.5918 - val_loss: 0.5962 - val_mae: 0.5680
Epoch 18/100
337/337 - 0s - loss: 0.4902 - mae: 0.5754 - val_loss: 0.6027 - val_mae: 0.5604
Epoch 19/100
337/337 - 0s - loss: 0.4828 - mae: 0.5626 - val_loss: 0.6094 - val_mae: 0.5618
Epoch 20/100
337/337 - 0s - loss: 0.5540 - mae: 0.5985 - val_loss: 0.6097 - val_mae: 0.5755
Epoch 21/100
337/337 - 0s - loss: 0.5097 - mae: 0.5924 - val_loss: 0.5755 - val_mae: 0.5700
Epoch 22/100
337/337 - 0s - loss: 0.4889 - mae: 0.5699 - val_loss: 0.5820 - val_mae: 0.5606
Epoch 23/100
337/337 - 0s - loss: 0.5049 - mae: 0.5791 - val_loss: 0.6079 - val_mae: 0.5707
Epoch 24/100
337/337 - 0s - loss: 0.4865 - mae: 0.5656 - val_loss: 0.5728 - val_mae: 0.5617
Epoch 25/100
337/337 - 0s - loss: 0.4735 - mae: 0.5559 - val_loss: 0.6019 - val_mae: 0.5652
Epoch 26/100
337/337 - 0s - loss: 0.4724 - mae: 0.5537 - val_loss: 0.6144 - val_mae: 0.5647
Epoch 27/100
337/337 - 0s - loss: 0.4800 - mae: 0.5711 - val_loss: 0.5875 - val_mae: 0.5671
Epoch 28/100
337/337 - 0s - loss: 0.4851 - mae: 0.5851 - val_loss: 0.5724 - val_mae: 0.5555
Epoch 29/100
337/337 - 0s - loss: 0.4711 - mae: 0.5600 - val_loss: 0.5882 - val_mae: 0.5528
Epoch 30/100
337/337 - 0s - loss: 0.4771 - mae: 0.5603 - val_loss: 0.5641 - val_mae: 0.5472
Epoch 31/100
337/337 - 0s - loss: 0.4566 - mae: 0.5502 - val_loss: 0.5775 - val_mae: 0.5573
Epoch 32/100
337/337 - 0s - loss: 0.4580 - mae: 0.5438 - val_loss: 0.6242 - val_mae: 0.5665
Epoch 33/100
337/337 - 0s - loss: 0.4487 - mae: 0.5438 - val_loss: 0.6069 - val_mae: 0.5680
Epoch 34/100
```

```
337/337 - 0s - loss: 0.4797 - mae: 0.5662 - val_loss: 0.5908 - val_mae: 0.5586
Epoch 35/100
337/337 - 0s - loss: 0.4494 - mae: 0.5479 - val_loss: 0.6198 - val_mae: 0.5644
Epoch 36/100
337/337 - 0s - loss: 0.4612 - mae: 0.5519 - val_loss: 0.5863 - val_mae: 0.5480
Epoch 37/100
337/337 - 0s - loss: 0.4511 - mae: 0.5352 - val_loss: 0.5727 - val_mae: 0.5458
Epoch 38/100
337/337 - 0s - loss: 0.4635 - mae: 0.5465 - val_loss: 0.5914 - val_mae: 0.5552
Epoch 39/100
337/337 - 0s - loss: 0.4301 - mae: 0.5275 - val_loss: 0.5799 - val_mae: 0.5437
Epoch 40/100
337/337 - 0s - loss: 0.4537 - mae: 0.5426 - val_loss: 0.6151 - val_mae: 0.5567
Epoch 41/100
337/337 - 0s - loss: 0.4298 - mae: 0.5198 - val_loss: 0.6287 - val_mae: 0.5694
Epoch 42/100
337/337 - 0s - loss: 0.4675 - mae: 0.5417 - val_loss: 0.6025 - val_mae: 0.5682
Epoch 43/100
337/337 - 0s - loss: 0.4390 - mae: 0.5418 - val_loss: 0.6249 - val_mae: 0.5693
Epoch 44/100
337/337 - 0s - loss: 0.4360 - mae: 0.5310 - val_loss: 0.6124 - val_mae: 0.5586
Epoch 45/100
337/337 - 0s - loss: 0.4330 - mae: 0.5232 - val_loss: 0.6008 - val_mae: 0.5501
Epoch 46/100
337/337 - 0s - loss: 0.4363 - mae: 0.5337 - val_loss: 0.6319 - val_mae: 0.5623
Epoch 47/100
337/337 - 0s - loss: 0.4174 - mae: 0.5144 - val_loss: 0.6102 - val_mae: 0.5552
Epoch 48/100
337/337 - 0s - loss: 0.4303 - mae: 0.5271 - val_loss: 0.6179 - val_mae: 0.5651
Epoch 49/100
337/337 - 0s - loss: 0.4353 - mae: 0.5367 - val_loss: 0.6068 - val_mae: 0.5584
Epoch 50/100
337/337 - 0s - loss: 0.4285 - mae: 0.5124 - val_loss: 0.6407 - val_mae: 0.5734
Epoch 51/100
337/337 - 0s - loss: 0.4170 - mae: 0.5217 - val_loss: 0.6090 - val_mae: 0.5643
Epoch 52/100
337/337 - 0s - loss: 0.4066 - mae: 0.5169 - val_loss: 0.6252 - val_mae: 0.5646
Epoch 53/100
337/337 - 0s - loss: 0.4173 - mae: 0.5196 - val_loss: 0.6442 - val_mae: 0.5706
Epoch 54/100
337/337 - 0s - loss: 0.4275 - mae: 0.5211 - val_loss: 0.6542 - val_mae: 0.5821
Epoch 55/100
337/337 - 0s - loss: 0.4194 - mae: 0.5231 - val_loss: 0.6425 - val_mae: 0.5833
Epoch 56/100
337/337 - 0s - loss: 0.4370 - mae: 0.5328 - val_loss: 0.6232 - val_mae: 0.5678
Epoch 57/100
337/337 - 0s - loss: 0.4136 - mae: 0.5084 - val_loss: 0.6760 - val_mae: 0.5971
Epoch 58/100
```

```
337/337 - 0s - loss: 0.4416 - mae: 0.5337 - val_loss: 0.5975 - val_mae: 0.5564
Epoch 59/100
337/337 - 0s - loss: 0.4325 - mae: 0.5445 - val_loss: 0.6246 - val_mae: 0.5731
Epoch 60/100
337/337 - 0s - loss: 0.4140 - mae: 0.5138 - val_loss: 0.6243 - val_mae: 0.5653
Epoch 61/100
337/337 - 0s - loss: 0.4039 - mae: 0.5040 - val_loss: 0.6432 - val_mae: 0.5719
Epoch 62/100
337/337 - 0s - loss: 0.4069 - mae: 0.5031 - val_loss: 0.6513 - val_mae: 0.5839
Epoch 63/100
337/337 - 0s - loss: 0.3719 - mae: 0.4794 - val_loss: 0.6084 - val_mae: 0.5617
Epoch 64/100
337/337 - 0s - loss: 0.4009 - mae: 0.4955 - val_loss: 0.6288 - val_mae: 0.5768
Epoch 65/100
337/337 - 0s - loss: 0.3762 - mae: 0.4904 - val_loss: 0.6348 - val_mae: 0.5698
Epoch 66/100
337/337 - 0s - loss: 0.4291 - mae: 0.5089 - val_loss: 0.6263 - val_mae: 0.5655
Epoch 67/100
337/337 - 0s - loss: 0.4173 - mae: 0.5308 - val_loss: 0.6001 - val_mae: 0.5581
Epoch 68/100
337/337 - 0s - loss: 0.4319 - mae: 0.5168 - val_loss: 0.6446 - val_mae: 0.5983
Epoch 69/100
337/337 - 0s - loss: 0.4123 - mae: 0.5219 - val_loss: 0.6012 - val_mae: 0.5561
Epoch 70/100
337/337 - 0s - loss: 0.4144 - mae: 0.5069 - val_loss: 0.6495 - val_mae: 0.5833
Epoch 71/100
337/337 - 0s - loss: 0.3892 - mae: 0.4981 - val_loss: 0.6269 - val_mae: 0.5686
Epoch 72/100
337/337 - 0s - loss: 0.3863 - mae: 0.5002 - val_loss: 0.6082 - val_mae: 0.5582
Epoch 73/100
337/337 - 0s - loss: 0.3957 - mae: 0.5033 - val_loss: 0.6406 - val_mae: 0.5791
Epoch 74/100
337/337 - 0s - loss: 0.3718 - mae: 0.4845 - val_loss: 0.6267 - val_mae: 0.5682
Epoch 75/100
337/337 - 0s - loss: 0.4022 - mae: 0.5119 - val_loss: 0.6504 - val_mae: 0.5770
Epoch 76/100
337/337 - 0s - loss: 0.3785 - mae: 0.4885 - val_loss: 0.6275 - val_mae: 0.5691
Epoch 77/100
337/337 - 0s - loss: 0.3742 - mae: 0.4822 - val_loss: 0.6552 - val_mae: 0.5756
Epoch 78/100
337/337 - 0s - loss: 0.3992 - mae: 0.5032 - val_loss: 0.6327 - val_mae: 0.5697
Epoch 79/100
337/337 - 0s - loss: 0.3798 - mae: 0.4978 - val_loss: 0.6301 - val_mae: 0.5749
Epoch 80/100
337/337 - 0s - loss: 0.3680 - mae: 0.4880 - val_loss: 0.6697 - val_mae: 0.5923
Epoch 81/100
337/337 - 0s - loss: 0.3669 - mae: 0.4842 - val_loss: 0.6488 - val_mae: 0.5776
Epoch 82/100
```

```
337/337 - 0s - loss: 0.3793 - mae: 0.4926 - val_loss: 0.6383 - val_mae: 0.5831
Epoch 83/100
337/337 - 0s - loss: 0.4239 - mae: 0.5196 - val_loss: 0.6038 - val_mae: 0.5666
Epoch 84/100
337/337 - 0s - loss: 0.3831 - mae: 0.4914 - val_loss: 0.6643 - val_mae: 0.5864
Epoch 85/100
337/337 - 0s - loss: 0.3885 - mae: 0.4994 - val_loss: 0.6122 - val_mae: 0.5663
Epoch 86/100
337/337 - 0s - loss: 0.3765 - mae: 0.4941 - val_loss: 0.6614 - val_mae: 0.5965
Epoch 87/100
337/337 - 0s - loss: 0.3825 - mae: 0.5041 - val_loss: 0.6526 - val_mae: 0.5816
Epoch 88/100
337/337 - 0s - loss: 0.3758 - mae: 0.4886 - val_loss: 0.6559 - val_mae: 0.5924
Epoch 89/100
337/337 - 0s - loss: 0.3812 - mae: 0.5017 - val_loss: 0.6507 - val_mae: 0.5908
Epoch 90/100
337/337 - 0s - loss: 0.3746 - mae: 0.4869 - val_loss: 0.6932 - val_mae: 0.6089
Epoch 91/100
337/337 - 0s - loss: 0.4073 - mae: 0.4952 - val_loss: 0.6258 - val_mae: 0.5823
Epoch 92/100
337/337 - 0s - loss: 0.3674 - mae: 0.4855 - val_loss: 0.6366 - val_mae: 0.5764
Epoch 93/100
337/337 - 0s - loss: 0.3862 - mae: 0.5044 - val_loss: 0.6545 - val_mae: 0.5852
Epoch 94/100
337/337 - 0s - loss: 0.3633 - mae: 0.4745 - val_loss: 0.6434 - val_mae: 0.5788
Epoch 95/100
337/337 - 0s - loss: 0.3601 - mae: 0.4784 - val_loss: 0.6579 - val_mae: 0.5818
Epoch 96/100
337/337 - 0s - loss: 0.3678 - mae: 0.4887 - val_loss: 0.6359 - val_mae: 0.5758
Epoch 97/100
337/337 - 0s - loss: 0.3804 - mae: 0.4933 - val_loss: 0.6295 - val_mae: 0.5692
Epoch 98/100
337/337 - 0s - loss: 0.3439 - mae: 0.4680 - val_loss: 0.6298 - val_mae: 0.5720
Epoch 99/100
337/337 - 0s - loss: 0.3513 - mae: 0.4791 - val_loss: 0.6641 - val_mae: 0.5884
Epoch 100/100
337/337 - 0s - loss: 0.3579 - mae: 0.4757 - val_loss: 0.6804 - val_mae: 0.5916
```

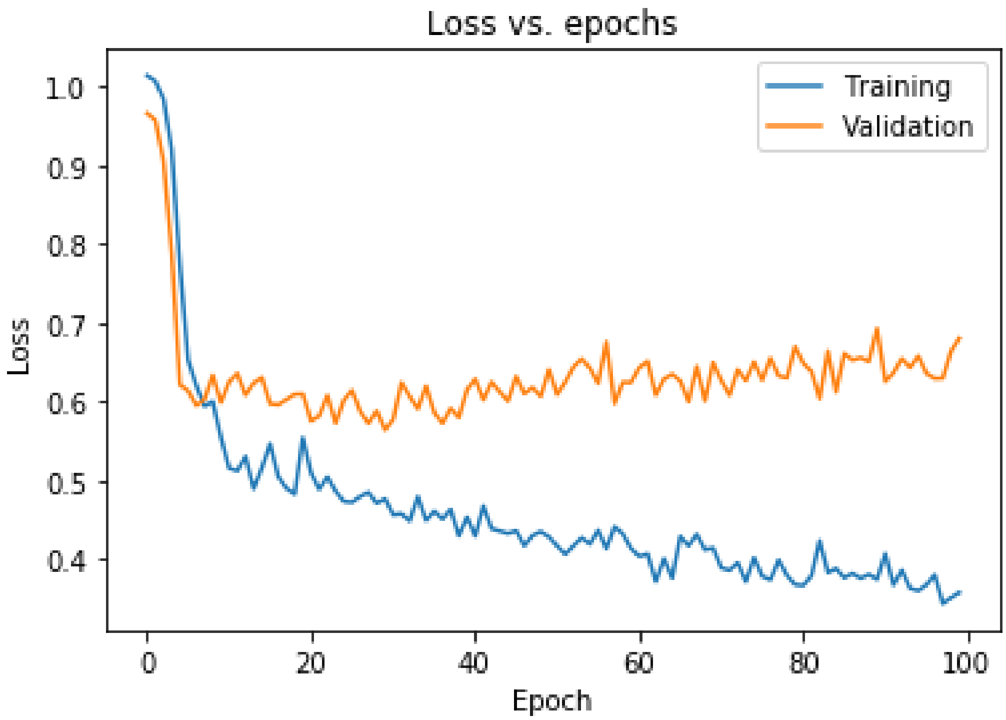
In [19]:
```
# Evaluate the model on the test set

model.evaluate(test_data, test_targets, verbose=2)
```

```
45/1 - 0s - loss: 0.4864 - mae: 0.5813
```

Out[19]:
```
[0.5492449588245816, 0.5813376]
```

Plot the learning curves

```
In [20]: # Plot the training and validation loss

         import matplotlib.pyplot as plt

         plt.plot(history.history['loss'])
         plt.plot(history.history['val_loss'])
         plt.title('Loss vs. epochs')
         plt.ylabel('Loss')
         plt.xlabel('Epoch')
         plt.legend(['Training', 'Validation'], loc='upper right')
         plt.show()
```



Introduction to callbacks

Example training callback

```
In [21]: # Write a custom callback

         from tensorflow.keras.callbacks import Callback

         class TrainingCallback(Callback):
```

```python
    def on_train_begin(self, logs=None):
        print("Starting training.....")

    def on_epoch_begin(self, epoch, logs=None):
        print(f"Starting epoch {epoch}")

    def on_train_batch_begin(Self, batch, logs=None):
        print(f"Training: Starting batch {batch}")

    def on_train_batch_end(self, batch, logs=None):
        print(f"Training: Finished batch {batch}")

    def on_epoch_end(self, epoch, logs=None):
        print(f"Finished epoch {epoch}")

    def on_train_end(self, logs=None):
        print("Finished training")
```

In [22]:
```python
from tensorflow.keras.callbacks import Callback
class TestingCallback(Callback):

    def on_test_begin(self, logs=None):
        print("Starting Testing.....")

    def on_test_batch_begin(Self, batch, logs=None):
        print(f"Testing: Starting batch {batch}")

    def on_test_batch_end(self, batch, logs=None):
        print(f"Testing: Finished batch {batch}")

    def on_test_end(self, logs=None):
        print("Finished Testing")
```

In [23]:
```python
class PredictionCallback(Callback):

    def on_predict_begin(self, logs=None):
        print("Starting Predictions.....")

    def on_predict_batch_begin(Self, batch, logs=None):
        print(f"Predictions: Starting batch {batch}")

    def on_predict_batch_end(self, batch, logs=None):
        print(f"Predictions: Finished batch {batch}")

    def on_predict_end(self, logs=None):
        print("Finished predicting")
```

In [24]:
```python
# Re-build the model
```

```
model = get_regularised_model(1e-5, 0.3)
```

In [25]:
```
# Compile the model

model.compile(optimizer='adam', loss='mse')
```

Train the model with the callback

In [26]:
```
# Train the model, with some of the data reserved for validation
model.fit(train_data, train_targets, epochs=3, batch_size=128, verbose=False, callbac
```

```
Starting training...
Starting epoch 0
Training: Starting batch 0
Training: Finished batch 0
Training: Starting batch 1
Training: Finished batch 1
Training: Starting batch 2
Training: Finished batch 2
Training: Starting batch 3
Training: Finished batch 3
Finished epoch 0
Starting epoch 1
Training: Starting batch 0
Training: Finished batch 0
Training: Starting batch 1
Training: Finished batch 1
Training: Starting batch 2
Training: Finished batch 2
Training: Starting batch 3
Training: Finished batch 3
Finished epoch 1
Starting epoch 2
Training: Starting batch 0
Training: Finished batch 0
Training: Starting batch 1
Training: Finished batch 1
Training: Starting batch 2
Training: Finished batch 2
Training: Starting batch 3
Training: Finished batch 3
Finished epoch 2
Finished training
```

Out[26]:
```
<tensorflow.python.keras.callbacks.History at 0x7f7fb832a4a8>
```

In [27]:
```
# Evaluate the model

model.evaluate(test_data, test_targets, verbose=False, callbacks=[TestingCallback()])
```

```
Starting Testing...
Testing: Starting batch 0
Testing: Finished batch 0
Testing: Starting batch 1
Testing: Finished batch 1
Finished Testing
```

Out[27]: 0.9526961194144354

In [28]:
```
# Make predictions with the model

model.predict(test_data, verbose=False, callbacks=[PredictionCallback()])
```

```
Starting Predictions...
Predictions: Starting batch 0
Predictions: Finished batch 0
Predictions: Starting batch 1
Predictions: Finished batch 1
Finished predicting
```

Out[28]:
```
array([[ 0.03588578],
       [ 0.05638439],
       [ 0.02530061],
       [ 0.03828479],
       [-0.08132552],
       [-0.02473307],
       [-0.10197912],
       [-0.10381686],
       [-0.02753333],
       [ 0.00141064],
       [ 0.01318806],
       [-0.07542612],
       [ 0.00608474],
       [ 0.03526763],
       [-0.01149174],
       [-0.083369  ],
       [-0.07078885],
       [ 0.03809523],
       [-0.00788529],
       [-0.03523815],
       [-0.00690893],
       [-0.05032004],
       [-0.05234443],
```

```
[-0.02720218],
[-0.06887939],
[ 0.0068557 ],
[-0.07656615],
[ 0.02965019],
[-0.01506982],
[-0.06594521],
[-0.07760373],
[-0.07404116],
[-0.0242819 ],
[-0.10583048],
[-0.06592582],
[-0.08002105],
[-0.03147271],
[ 0.01224775],
[-0.00241393],
[-0.09658805],
[-0.10055888],
[ 0.02843363],
[ 0.00745514],
[-0.10653213],
[-0.08140218]], dtype=float32)
```

Early stopping / patience

Re-train the models with early stopping

```
In [42]: # Re-train the unregularised model
         unregularised_model = get_model()
         unregularised_model.compile(optimizer='adam', loss='mse')
         unreg_history = unregularised_model.fit(train_data, train_targets, epochs=100, valida
                                                 batch_size=64,
                                                 verbose=False,
                                                 callbacks=[tf.keras.callbacks.EarlyStopping(pa
```

```
In [43]: # Evaluate the model on the test set

         unregularised_model.evaluate(test_data, test_targets, verbose=2)
```

```
45/1 - 0s - loss: 0.6144
```

```
Out[43]: 0.6202218810717265
```

```
In [41]: # Re-train the regularised model
         regularised_model = get_regularised_model(1e-8, 0.2)
         regularised_model.compile(optimzer='adam', loss='mse')
```

```
reg_history = regularised_model.fit(train_data, train_targets, epochs=100,
                                    validation_split=0.15, batch_size=64, verbose=Fals
                                     callbacks=[tf.keras.callbacks.EarlyStopping(patie
```

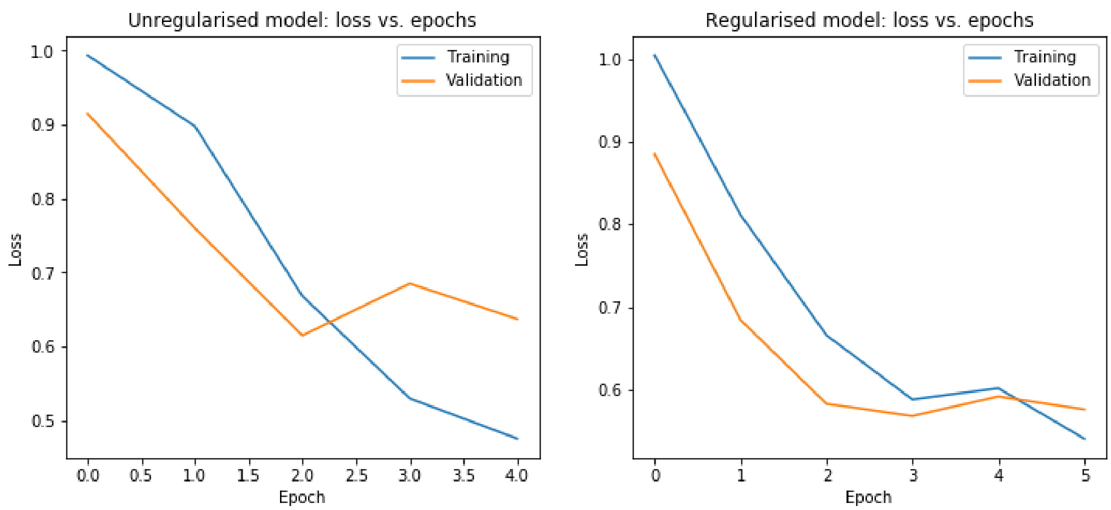
In [44]:
```
# Evaluate the model on the test set

regularised_model.evaluate(test_data, test_targets, verbose=2)
```

```
45/1 - 0s - loss: 0.5913
```

Out[44]: 0.5927419463793436

Plot the learning curves

In [47]:
```
# Plot the training and validation loss

import matplotlib.pyplot as plt

fig = plt.figure(figsize=(12, 5))

fig.add_subplot(121)

plt.plot(unreg_history.history['loss'])
plt.plot(unreg_history.history['val_loss'])
plt.title('Unregularised model: loss vs. epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='upper right')

fig.add_subplot(122)

plt.plot(reg_history.history['loss'])
plt.plot(reg_history.history['val_loss'])
plt.title('Regularised model: loss vs. epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='upper right')

plt.show()
```


Unregularised model: loss vs. epochs
Training
Validation
Loss
Epoch
Regularised model: loss vs. epochs
Training
Validation
Loss
Epoch

In [ ]: