Coding Tutorial

November 14, 2020

```
In [1]: 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 [2]: # 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 dev

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 Regressi
(http://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
```

In [3]:
```
# Save the input and target variables
print(diabetes_dataset.keys())
data = diabetes_dataset['data']
targets = diabetes_dataset['target']
```

```
dict_keys(['data', 'target', 'DESCR', 'feature_names', 'data_filename', 'target_filename'])
```

In [4]:
```
# Normalise the target data (this will make clearer training curves)
targets = (targets - targets.mean())/targets.std()
targets
```

Out[4]:
```
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 [5]:
```
# 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(train_targets.shape)
print(test_data.shape)
print(test_targets.shape)
```

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

Train a feedforward neural network model

In [6]:
```
# Build the model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

def get_model():
    model = Sequential([
        Dense(units = 128, activation = 'relu', input_shape = (train_data.shape[1], ))
        Dense(units = 64, activation = 'relu'),
        Dense(units = 16, activation = 'relu'),
        Dense(units = 1)
    ])
    return model

#define model
model = get_model()
```

In [7]:
```
# Print the model summary
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               1408
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 16)                1040
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
```

```
=================================================================
Total params: 10,721
Trainable params: 10,721
Non-trainable params: 0
_________________________________________________________________
```

```
In [8]: # Compile the model
        model.compile(optimizer = 'adam', loss = 'mse', metrics = ['mae'])

In [9]: # 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 - 1s - loss: 0.9673 - mae: 0.8467 - val_loss: 1.0552 - val_mae: 0.8525
Epoch 2/100
337/337 - 0s - loss: 0.9277 - mae: 0.8286 - val_loss: 1.0190 - val_mae: 0.8364
Epoch 3/100
337/337 - 0s - loss: 0.8774 - mae: 0.8051 - val_loss: 0.9760 - val_mae: 0.8167
Epoch 4/100
337/337 - 0s - loss: 0.8146 - mae: 0.7745 - val_loss: 0.9209 - val_mae: 0.7921
Epoch 5/100
337/337 - 0s - loss: 0.7413 - mae: 0.7365 - val_loss: 0.8640 - val_mae: 0.7630
Epoch 6/100
337/337 - 0s - loss: 0.6613 - mae: 0.6880 - val_loss: 0.8129 - val_mae: 0.7408
Epoch 7/100
337/337 - 0s - loss: 0.5940 - mae: 0.6478 - val_loss: 0.7609 - val_mae: 0.7212
Epoch 8/100
337/337 - 0s - loss: 0.5426 - mae: 0.6159 - val_loss: 0.7386 - val_mae: 0.6963
Epoch 9/100
337/337 - 0s - loss: 0.5092 - mae: 0.5876 - val_loss: 0.7190 - val_mae: 0.6788
Epoch 10/100
337/337 - 0s - loss: 0.4904 - mae: 0.5726 - val_loss: 0.7025 - val_mae: 0.6651
Epoch 11/100
337/337 - 0s - loss: 0.4769 - mae: 0.5634 - val_loss: 0.6798 - val_mae: 0.6494
Epoch 12/100
337/337 - 0s - loss: 0.4659 - mae: 0.5579 - val_loss: 0.6623 - val_mae: 0.6384
Epoch 13/100
337/337 - 0s - loss: 0.4528 - mae: 0.5497 - val_loss: 0.6694 - val_mae: 0.6397
Epoch 14/100
337/337 - 0s - loss: 0.4546 - mae: 0.5468 - val_loss: 0.6540 - val_mae: 0.6300
Epoch 15/100
337/337 - 0s - loss: 0.4569 - mae: 0.5542 - val_loss: 0.6536 - val_mae: 0.6287
Epoch 16/100
337/337 - 0s - loss: 0.4498 - mae: 0.5424 - val_loss: 0.6697 - val_mae: 0.6408
Epoch 17/100
```

```
337/337 - 0s - loss: 0.4458 - mae: 0.5418 - val_loss: 0.6521 - val_mae: 0.6280
Epoch 18/100
337/337 - 0s - loss: 0.4460 - mae: 0.5368 - val_loss: 0.6749 - val_mae: 0.6454
Epoch 19/100
337/337 - 0s - loss: 0.4391 - mae: 0.5318 - val_loss: 0.6514 - val_mae: 0.6300
Epoch 20/100
337/337 - 0s - loss: 0.4416 - mae: 0.5413 - val_loss: 0.6502 - val_mae: 0.6296
Epoch 21/100
337/337 - 0s - loss: 0.4339 - mae: 0.5331 - val_loss: 0.6692 - val_mae: 0.6439
Epoch 22/100
337/337 - 0s - loss: 0.4426 - mae: 0.5347 - val_loss: 0.6507 - val_mae: 0.6307
Epoch 23/100
337/337 - 0s - loss: 0.4298 - mae: 0.5328 - val_loss: 0.6431 - val_mae: 0.6296
Epoch 24/100
337/337 - 0s - loss: 0.4289 - mae: 0.5330 - val_loss: 0.6534 - val_mae: 0.6316
Epoch 25/100
337/337 - 0s - loss: 0.4289 - mae: 0.5275 - val_loss: 0.6515 - val_mae: 0.6306
Epoch 26/100
337/337 - 0s - loss: 0.4250 - mae: 0.5281 - val_loss: 0.6387 - val_mae: 0.6272
Epoch 27/100
337/337 - 0s - loss: 0.4254 - mae: 0.5301 - val_loss: 0.6399 - val_mae: 0.6260
Epoch 28/100
337/337 - 0s - loss: 0.4233 - mae: 0.5261 - val_loss: 0.6456 - val_mae: 0.6291
Epoch 29/100
337/337 - 0s - loss: 0.4210 - mae: 0.5216 - val_loss: 0.6495 - val_mae: 0.6317
Epoch 30/100
337/337 - 0s - loss: 0.4256 - mae: 0.5214 - val_loss: 0.6459 - val_mae: 0.6282
Epoch 31/100
337/337 - 0s - loss: 0.4169 - mae: 0.5205 - val_loss: 0.6442 - val_mae: 0.6286
Epoch 32/100
337/337 - 0s - loss: 0.4159 - mae: 0.5219 - val_loss: 0.6446 - val_mae: 0.6278
Epoch 33/100
337/337 - 0s - loss: 0.4157 - mae: 0.5226 - val_loss: 0.6483 - val_mae: 0.6289
Epoch 34/100
337/337 - 0s - loss: 0.4109 - mae: 0.5181 - val_loss: 0.6484 - val_mae: 0.6274
Epoch 35/100
337/337 - 0s - loss: 0.4096 - mae: 0.5167 - val_loss: 0.6430 - val_mae: 0.6273
Epoch 36/100
337/337 - 0s - loss: 0.4083 - mae: 0.5162 - val_loss: 0.6459 - val_mae: 0.6282
Epoch 37/100
337/337 - 0s - loss: 0.4081 - mae: 0.5139 - val_loss: 0.6530 - val_mae: 0.6308
Epoch 38/100
337/337 - 0s - loss: 0.4088 - mae: 0.5145 - val_loss: 0.6522 - val_mae: 0.6338
Epoch 39/100
337/337 - 0s - loss: 0.4083 - mae: 0.5162 - val_loss: 0.6683 - val_mae: 0.6392
Epoch 40/100
337/337 - 0s - loss: 0.4037 - mae: 0.5105 - val_loss: 0.6519 - val_mae: 0.6329
Epoch 41/100
```

```
337/337 - 0s - loss: 0.4010 - mae: 0.5108 - val_loss: 0.6450 - val_mae: 0.6282
Epoch 42/100
337/337 - 0s - loss: 0.4000 - mae: 0.5106 - val_loss: 0.6456 - val_mae: 0.6278
Epoch 43/100
337/337 - 0s - loss: 0.3970 - mae: 0.5065 - val_loss: 0.6443 - val_mae: 0.6291
Epoch 44/100
337/337 - 0s - loss: 0.3953 - mae: 0.5058 - val_loss: 0.6455 - val_mae: 0.6318
Epoch 45/100
337/337 - 0s - loss: 0.3940 - mae: 0.5038 - val_loss: 0.6472 - val_mae: 0.6325
Epoch 46/100
337/337 - 0s - loss: 0.3929 - mae: 0.5047 - val_loss: 0.6451 - val_mae: 0.6329
Epoch 47/100
337/337 - 0s - loss: 0.3956 - mae: 0.5029 - val_loss: 0.6519 - val_mae: 0.6313
Epoch 48/100
337/337 - 0s - loss: 0.3899 - mae: 0.5017 - val_loss: 0.6505 - val_mae: 0.6340
Epoch 49/100
337/337 - 0s - loss: 0.3871 - mae: 0.4983 - val_loss: 0.6686 - val_mae: 0.6382
Epoch 50/100
337/337 - 0s - loss: 0.3864 - mae: 0.4962 - val_loss: 0.6632 - val_mae: 0.6392
Epoch 51/100
337/337 - 0s - loss: 0.3834 - mae: 0.4941 - val_loss: 0.6780 - val_mae: 0.6419
Epoch 52/100
337/337 - 0s - loss: 0.3835 - mae: 0.4930 - val_loss: 0.6630 - val_mae: 0.6362
Epoch 53/100
337/337 - 0s - loss: 0.3834 - mae: 0.4937 - val_loss: 0.6640 - val_mae: 0.6352
Epoch 54/100
337/337 - 0s - loss: 0.3819 - mae: 0.4931 - val_loss: 0.6624 - val_mae: 0.6380
Epoch 55/100
337/337 - 0s - loss: 0.3821 - mae: 0.4882 - val_loss: 0.6725 - val_mae: 0.6373
Epoch 56/100
337/337 - 0s - loss: 0.3777 - mae: 0.4881 - val_loss: 0.6615 - val_mae: 0.6395
Epoch 57/100
337/337 - 0s - loss: 0.3797 - mae: 0.4877 - val_loss: 0.6778 - val_mae: 0.6390
Epoch 58/100
337/337 - 0s - loss: 0.3728 - mae: 0.4838 - val_loss: 0.6694 - val_mae: 0.6456
Epoch 59/100
337/337 - 0s - loss: 0.3730 - mae: 0.4875 - val_loss: 0.6702 - val_mae: 0.6360
Epoch 60/100
337/337 - 0s - loss: 0.3695 - mae: 0.4811 - val_loss: 0.6651 - val_mae: 0.6415
Epoch 61/100
337/337 - 0s - loss: 0.3838 - mae: 0.4927 - val_loss: 0.6719 - val_mae: 0.6441
Epoch 62/100
337/337 - 0s - loss: 0.3828 - mae: 0.4895 - val_loss: 0.6946 - val_mae: 0.6448
Epoch 63/100
337/337 - 0s - loss: 0.3680 - mae: 0.4796 - val_loss: 0.6934 - val_mae: 0.6675
Epoch 64/100
337/337 - 0s - loss: 0.3847 - mae: 0.4927 - val_loss: 0.6893 - val_mae: 0.6466
Epoch 65/100
```

```
337/337 - 0s - loss: 0.3668 - mae: 0.4799 - val_loss: 0.6724 - val_mae: 0.6471
Epoch 66/100
337/337 - 0s - loss: 0.3740 - mae: 0.4923 - val_loss: 0.6698 - val_mae: 0.6442
Epoch 67/100
337/337 - 0s - loss: 0.3591 - mae: 0.4739 - val_loss: 0.6942 - val_mae: 0.6422
Epoch 68/100
337/337 - 0s - loss: 0.3641 - mae: 0.4766 - val_loss: 0.6779 - val_mae: 0.6515
Epoch 69/100
337/337 - 0s - loss: 0.3671 - mae: 0.4812 - val_loss: 0.6828 - val_mae: 0.6482
Epoch 70/100
337/337 - 0s - loss: 0.3585 - mae: 0.4730 - val_loss: 0.6955 - val_mae: 0.6482
Epoch 71/100
337/337 - 0s - loss: 0.3573 - mae: 0.4703 - val_loss: 0.6925 - val_mae: 0.6455
Epoch 72/100
337/337 - 0s - loss: 0.3566 - mae: 0.4734 - val_loss: 0.6825 - val_mae: 0.6504
Epoch 73/100
337/337 - 0s - loss: 0.3593 - mae: 0.4736 - val_loss: 0.6905 - val_mae: 0.6457
Epoch 74/100
337/337 - 0s - loss: 0.3506 - mae: 0.4688 - val_loss: 0.6861 - val_mae: 0.6541
Epoch 75/100
337/337 - 0s - loss: 0.3524 - mae: 0.4684 - val_loss: 0.6875 - val_mae: 0.6439
Epoch 76/100
337/337 - 0s - loss: 0.3524 - mae: 0.4671 - val_loss: 0.6814 - val_mae: 0.6444
Epoch 77/100
337/337 - 0s - loss: 0.3452 - mae: 0.4635 - val_loss: 0.6891 - val_mae: 0.6538
Epoch 78/100
337/337 - 0s - loss: 0.3484 - mae: 0.4669 - val_loss: 0.6906 - val_mae: 0.6477
Epoch 79/100
337/337 - 0s - loss: 0.3436 - mae: 0.4607 - val_loss: 0.6921 - val_mae: 0.6482
Epoch 80/100
337/337 - 0s - loss: 0.3441 - mae: 0.4579 - val_loss: 0.6876 - val_mae: 0.6496
Epoch 81/100
337/337 - 0s - loss: 0.3469 - mae: 0.4641 - val_loss: 0.6853 - val_mae: 0.6492
Epoch 82/100
337/337 - 0s - loss: 0.3414 - mae: 0.4571 - val_loss: 0.6913 - val_mae: 0.6439
Epoch 83/100
337/337 - 0s - loss: 0.3426 - mae: 0.4595 - val_loss: 0.6945 - val_mae: 0.6522
Epoch 84/100
337/337 - 0s - loss: 0.3396 - mae: 0.4566 - val_loss: 0.7029 - val_mae: 0.6481
Epoch 85/100
337/337 - 0s - loss: 0.3344 - mae: 0.4521 - val_loss: 0.7084 - val_mae: 0.6535
Epoch 86/100
337/337 - 0s - loss: 0.3359 - mae: 0.4553 - val_loss: 0.7063 - val_mae: 0.6546
Epoch 87/100
337/337 - 0s - loss: 0.3334 - mae: 0.4521 - val_loss: 0.7065 - val_mae: 0.6488
Epoch 88/100
337/337 - 0s - loss: 0.3332 - mae: 0.4543 - val_loss: 0.6953 - val_mae: 0.6550
Epoch 89/100
```

```
337/337 - 0s - loss: 0.3315 - mae: 0.4546 - val_loss: 0.7078 - val_mae: 0.6474
Epoch 90/100
337/337 - 0s - loss: 0.3371 - mae: 0.4514 - val_loss: 0.6970 - val_mae: 0.6503
Epoch 91/100
337/337 - 0s - loss: 0.3483 - mae: 0.4693 - val_loss: 0.7042 - val_mae: 0.6520
Epoch 92/100
337/337 - 0s - loss: 0.3403 - mae: 0.4537 - val_loss: 0.7304 - val_mae: 0.6556
Epoch 93/100
337/337 - 0s - loss: 0.3297 - mae: 0.4435 - val_loss: 0.7054 - val_mae: 0.6578
Epoch 94/100
337/337 - 0s - loss: 0.3305 - mae: 0.4484 - val_loss: 0.7103 - val_mae: 0.6490
Epoch 95/100
337/337 - 0s - loss: 0.3371 - mae: 0.4586 - val_loss: 0.7032 - val_mae: 0.6505
Epoch 96/100
337/337 - 0s - loss: 0.3359 - mae: 0.4489 - val_loss: 0.7102 - val_mae: 0.6451
Epoch 97/100
337/337 - 0s - loss: 0.3418 - mae: 0.4619 - val_loss: 0.6979 - val_mae: 0.6562
Epoch 98/100
337/337 - 0s - loss: 0.3280 - mae: 0.4441 - val_loss: 0.7082 - val_mae: 0.6415
Epoch 99/100
337/337 - 0s - loss: 0.3239 - mae: 0.4413 - val_loss: 0.6911 - val_mae: 0.6506
Epoch 100/100
337/337 - 0s - loss: 0.3222 - mae: 0.4456 - val_loss: 0.7077 - val_mae: 0.6464
```

```
In [10]: # Evaluate the model on the test set
         model.evaluate(test_data, test_targets, verbose = 2)
```

```
45/1 - 0s - loss: 0.6149 - mae: 0.6277
```

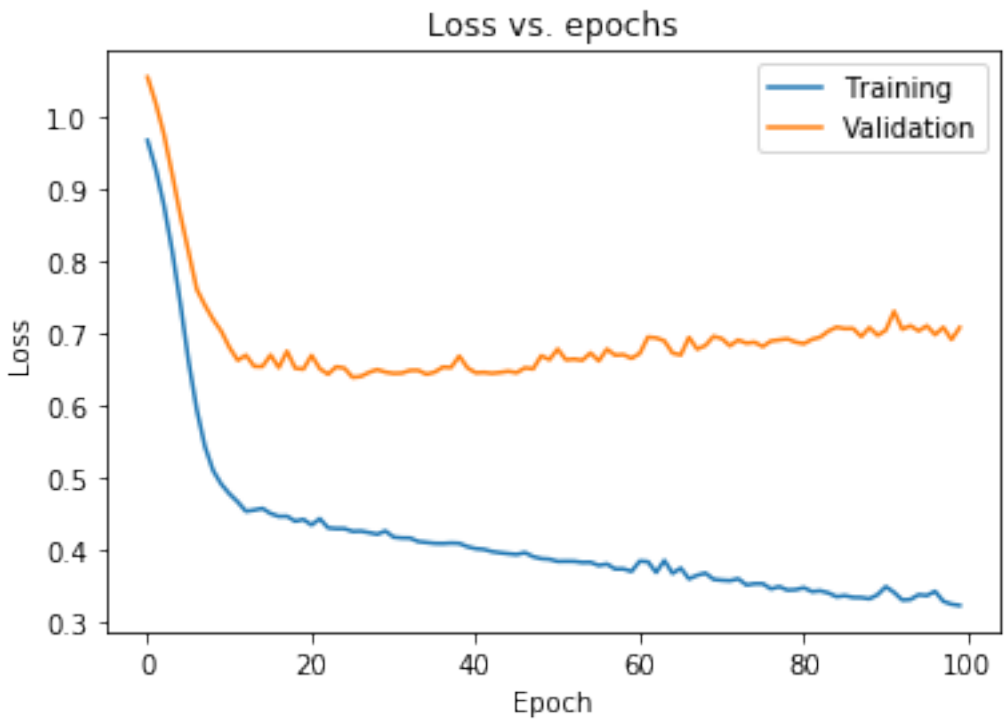
```
Out[10]: [0.6424800250265333, 0.6277188]
```

Plot the learning curves

```
In [11]: import matplotlib.pyplot as plt
         %matplotlib inline
```

```
In [12]: # 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 [8]: from tensorflow.keras.layers import Dropout
        from tensorflow.keras import regularizers

In [9]: def get_regularised_model(wd, rate):
            model = Sequential([
                Dense(128, activation = 'relu', input_shape = (train_data.shape[1], ), kernel_r
                Dropout(rate),
                Dense(64, activation='relu', kernel_regularizer = regularizers.l2(wd)),
                Dropout(rate),
                Dense(16, activation='relu', kernel_regularizer = regularizers.l2(wd)),
                Dropout(rate),
                Dense(1)
            ])
            return model

In [15]: # Re-build the model with weight decay and dropout layers
         model = get_regularised_model(wd = 1e-5, rate = 0.3)
```

```
In [16]: # Compile the model
         model.compile(optimizer = 'adam', loss = 'mse', metrics = ['mae'])

In [17]: # 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 - 1s - loss: 0.9692 - mae: 0.8462 - val_loss: 1.0494 - val_mae: 0.8506
Epoch 2/100
337/337 - 0s - loss: 0.9249 - mae: 0.8251 - val_loss: 1.0103 - val_mae: 0.8305
Epoch 3/100
337/337 - 0s - loss: 0.8766 - mae: 0.8025 - val_loss: 0.9620 - val_mae: 0.8093
Epoch 4/100
337/337 - 0s - loss: 0.8114 - mae: 0.7709 - val_loss: 0.9024 - val_mae: 0.7847
Epoch 5/100
337/337 - 0s - loss: 0.7532 - mae: 0.7475 - val_loss: 0.8460 - val_mae: 0.7629
Epoch 6/100
337/337 - 0s - loss: 0.6763 - mae: 0.6997 - val_loss: 0.7999 - val_mae: 0.7433
Epoch 7/100
337/337 - 0s - loss: 0.6306 - mae: 0.6767 - val_loss: 0.7666 - val_mae: 0.7279
Epoch 8/100
337/337 - 0s - loss: 0.6349 - mae: 0.6699 - val_loss: 0.7558 - val_mae: 0.7104
Epoch 9/100
337/337 - 0s - loss: 0.6022 - mae: 0.6444 - val_loss: 0.7468 - val_mae: 0.6959
Epoch 10/100
337/337 - 0s - loss: 0.5750 - mae: 0.6114 - val_loss: 0.7219 - val_mae: 0.6890
Epoch 11/100
337/337 - 0s - loss: 0.5846 - mae: 0.6324 - val_loss: 0.7104 - val_mae: 0.6835
Epoch 12/100
337/337 - 0s - loss: 0.5318 - mae: 0.6059 - val_loss: 0.6932 - val_mae: 0.6661
Epoch 13/100
337/337 - 0s - loss: 0.5246 - mae: 0.5854 - val_loss: 0.6873 - val_mae: 0.6614
Epoch 14/100
337/337 - 0s - loss: 0.5116 - mae: 0.5896 - val_loss: 0.6812 - val_mae: 0.6584
Epoch 15/100
337/337 - 0s - loss: 0.5384 - mae: 0.5963 - val_loss: 0.6842 - val_mae: 0.6575
Epoch 16/100
337/337 - 0s - loss: 0.5639 - mae: 0.6200 - val_loss: 0.6741 - val_mae: 0.6558
Epoch 17/100
337/337 - 0s - loss: 0.5413 - mae: 0.6153 - val_loss: 0.6716 - val_mae: 0.6549
Epoch 18/100
337/337 - 0s - loss: 0.5360 - mae: 0.6005 - val_loss: 0.6684 - val_mae: 0.6513
Epoch 19/100
337/337 - 0s - loss: 0.5313 - mae: 0.5977 - val_loss: 0.6689 - val_mae: 0.6458
Epoch 20/100
337/337 - 0s - loss: 0.5365 - mae: 0.6041 - val_loss: 0.6768 - val_mae: 0.6471
```

```
Epoch 21/100
337/337 - 0s - loss: 0.4965 - mae: 0.5737 - val_loss: 0.6633 - val_mae: 0.6384
Epoch 22/100
337/337 - 0s - loss: 0.5296 - mae: 0.5885 - val_loss: 0.6596 - val_mae: 0.6384
Epoch 23/100
337/337 - 0s - loss: 0.5132 - mae: 0.5919 - val_loss: 0.6566 - val_mae: 0.6359
Epoch 24/100
337/337 - 0s - loss: 0.4732 - mae: 0.5578 - val_loss: 0.6508 - val_mae: 0.6366
Epoch 25/100
337/337 - 0s - loss: 0.5070 - mae: 0.5858 - val_loss: 0.6437 - val_mae: 0.6327
Epoch 26/100
337/337 - 0s - loss: 0.4894 - mae: 0.5709 - val_loss: 0.6492 - val_mae: 0.6340
Epoch 27/100
337/337 - 0s - loss: 0.4964 - mae: 0.5679 - val_loss: 0.6454 - val_mae: 0.6311
Epoch 28/100
337/337 - 0s - loss: 0.5415 - mae: 0.6014 - val_loss: 0.6477 - val_mae: 0.6352
Epoch 29/100
337/337 - 0s - loss: 0.5064 - mae: 0.5832 - val_loss: 0.6535 - val_mae: 0.6397
Epoch 30/100
337/337 - 0s - loss: 0.5102 - mae: 0.5969 - val_loss: 0.6653 - val_mae: 0.6448
Epoch 31/100
337/337 - 0s - loss: 0.5270 - mae: 0.5930 - val_loss: 0.6403 - val_mae: 0.6345
Epoch 32/100
337/337 - 0s - loss: 0.4972 - mae: 0.5747 - val_loss: 0.6358 - val_mae: 0.6301
Epoch 33/100
337/337 - 0s - loss: 0.4808 - mae: 0.5675 - val_loss: 0.6369 - val_mae: 0.6251
Epoch 34/100
337/337 - 0s - loss: 0.4672 - mae: 0.5648 - val_loss: 0.6362 - val_mae: 0.6237
Epoch 35/100
337/337 - 0s - loss: 0.4971 - mae: 0.5805 - val_loss: 0.6328 - val_mae: 0.6233
Epoch 36/100
337/337 - 0s - loss: 0.4705 - mae: 0.5555 - val_loss: 0.6368 - val_mae: 0.6255
Epoch 37/100
337/337 - 0s - loss: 0.4824 - mae: 0.5781 - val_loss: 0.6440 - val_mae: 0.6313
Epoch 38/100
337/337 - 0s - loss: 0.4706 - mae: 0.5599 - val_loss: 0.6341 - val_mae: 0.6294
Epoch 39/100
337/337 - 0s - loss: 0.4850 - mae: 0.5713 - val_loss: 0.6289 - val_mae: 0.6287
Epoch 40/100
337/337 - 0s - loss: 0.4642 - mae: 0.5637 - val_loss: 0.6346 - val_mae: 0.6263
Epoch 41/100
337/337 - 0s - loss: 0.4873 - mae: 0.5776 - val_loss: 0.6477 - val_mae: 0.6291
Epoch 42/100
337/337 - 0s - loss: 0.4923 - mae: 0.5743 - val_loss: 0.6430 - val_mae: 0.6238
Epoch 43/100
337/337 - 0s - loss: 0.4757 - mae: 0.5629 - val_loss: 0.6372 - val_mae: 0.6237
Epoch 44/100
337/337 - 0s - loss: 0.4906 - mae: 0.5770 - val_loss: 0.6363 - val_mae: 0.6267
```

```
Epoch 45/100
337/337 - 0s - loss: 0.4606 - mae: 0.5510 - val_loss: 0.6248 - val_mae: 0.6217
Epoch 46/100
337/337 - 0s - loss: 0.5167 - mae: 0.5923 - val_loss: 0.6241 - val_mae: 0.6167
Epoch 47/100
337/337 - 0s - loss: 0.4460 - mae: 0.5436 - val_loss: 0.6381 - val_mae: 0.6215
Epoch 48/100
337/337 - 0s - loss: 0.4767 - mae: 0.5627 - val_loss: 0.6334 - val_mae: 0.6177
Epoch 49/100
337/337 - 0s - loss: 0.4565 - mae: 0.5581 - val_loss: 0.6291 - val_mae: 0.6159
Epoch 50/100
337/337 - 0s - loss: 0.4641 - mae: 0.5533 - val_loss: 0.6291 - val_mae: 0.6153
Epoch 51/100
337/337 - 0s - loss: 0.4807 - mae: 0.5766 - val_loss: 0.6207 - val_mae: 0.6118
Epoch 52/100
337/337 - 0s - loss: 0.4616 - mae: 0.5615 - val_loss: 0.6189 - val_mae: 0.6122
Epoch 53/100
337/337 - 0s - loss: 0.4906 - mae: 0.5767 - val_loss: 0.6258 - val_mae: 0.6141
Epoch 54/100
337/337 - 0s - loss: 0.4811 - mae: 0.5703 - val_loss: 0.6306 - val_mae: 0.6169
Epoch 55/100
337/337 - 0s - loss: 0.4485 - mae: 0.5414 - val_loss: 0.6282 - val_mae: 0.6154
Epoch 56/100
337/337 - 0s - loss: 0.4855 - mae: 0.5726 - val_loss: 0.6225 - val_mae: 0.6144
Epoch 57/100
337/337 - 0s - loss: 0.4684 - mae: 0.5575 - val_loss: 0.6238 - val_mae: 0.6156
Epoch 58/100
337/337 - 0s - loss: 0.4455 - mae: 0.5416 - val_loss: 0.6226 - val_mae: 0.6144
Epoch 59/100
337/337 - 0s - loss: 0.4625 - mae: 0.5503 - val_loss: 0.6242 - val_mae: 0.6135
Epoch 60/100
337/337 - 0s - loss: 0.4680 - mae: 0.5489 - val_loss: 0.6278 - val_mae: 0.6138
Epoch 61/100
337/337 - 0s - loss: 0.4650 - mae: 0.5557 - val_loss: 0.6291 - val_mae: 0.6156
Epoch 62/100
337/337 - 0s - loss: 0.4605 - mae: 0.5506 - val_loss: 0.6279 - val_mae: 0.6155
Epoch 63/100
337/337 - 0s - loss: 0.4624 - mae: 0.5488 - val_loss: 0.6243 - val_mae: 0.6153
Epoch 64/100
337/337 - 0s - loss: 0.4666 - mae: 0.5571 - val_loss: 0.6239 - val_mae: 0.6170
Epoch 65/100
337/337 - 0s - loss: 0.4479 - mae: 0.5524 - val_loss: 0.6300 - val_mae: 0.6145
Epoch 66/100
337/337 - 0s - loss: 0.4797 - mae: 0.5628 - val_loss: 0.6287 - val_mae: 0.6139
Epoch 67/100
337/337 - 0s - loss: 0.4483 - mae: 0.5433 - val_loss: 0.6213 - val_mae: 0.6136
Epoch 68/100
337/337 - 0s - loss: 0.4711 - mae: 0.5626 - val_loss: 0.6269 - val_mae: 0.6146
```

```
Epoch 69/100
337/337 - 0s - loss: 0.4385 - mae: 0.5399 - val_loss: 0.6336 - val_mae: 0.6163
Epoch 70/100
337/337 - 0s - loss: 0.4437 - mae: 0.5485 - val_loss: 0.6267 - val_mae: 0.6139
Epoch 71/100
337/337 - 0s - loss: 0.4299 - mae: 0.5234 - val_loss: 0.6226 - val_mae: 0.6135
Epoch 72/100
337/337 - 0s - loss: 0.4376 - mae: 0.5307 - val_loss: 0.6189 - val_mae: 0.6159
Epoch 73/100
337/337 - 0s - loss: 0.4449 - mae: 0.5413 - val_loss: 0.6232 - val_mae: 0.6125
Epoch 74/100
337/337 - 0s - loss: 0.4632 - mae: 0.5477 - val_loss: 0.6243 - val_mae: 0.6129
Epoch 75/100
337/337 - 0s - loss: 0.4344 - mae: 0.5411 - val_loss: 0.6189 - val_mae: 0.6125
Epoch 76/100
337/337 - 0s - loss: 0.4628 - mae: 0.5503 - val_loss: 0.6158 - val_mae: 0.6119
Epoch 77/100
337/337 - 0s - loss: 0.4231 - mae: 0.5279 - val_loss: 0.6232 - val_mae: 0.6117
Epoch 78/100
337/337 - 0s - loss: 0.4506 - mae: 0.5356 - val_loss: 0.6350 - val_mae: 0.6173
Epoch 79/100
337/337 - 0s - loss: 0.4276 - mae: 0.5243 - val_loss: 0.6258 - val_mae: 0.6148
Epoch 80/100
337/337 - 0s - loss: 0.4421 - mae: 0.5388 - val_loss: 0.6203 - val_mae: 0.6131
Epoch 81/100
337/337 - 0s - loss: 0.4402 - mae: 0.5428 - val_loss: 0.6304 - val_mae: 0.6162
Epoch 82/100
337/337 - 0s - loss: 0.4359 - mae: 0.5294 - val_loss: 0.6189 - val_mae: 0.6131
Epoch 83/100
337/337 - 0s - loss: 0.4417 - mae: 0.5389 - val_loss: 0.6133 - val_mae: 0.6126
Epoch 84/100
337/337 - 0s - loss: 0.4544 - mae: 0.5487 - val_loss: 0.6149 - val_mae: 0.6113
Epoch 85/100
337/337 - 0s - loss: 0.4575 - mae: 0.5443 - val_loss: 0.6306 - val_mae: 0.6147
Epoch 86/100
337/337 - 0s - loss: 0.4401 - mae: 0.5372 - val_loss: 0.6306 - val_mae: 0.6152
Epoch 87/100
337/337 - 0s - loss: 0.4432 - mae: 0.5388 - val_loss: 0.6253 - val_mae: 0.6129
Epoch 88/100
337/337 - 0s - loss: 0.4322 - mae: 0.5339 - val_loss: 0.6265 - val_mae: 0.6135
Epoch 89/100
337/337 - 0s - loss: 0.4126 - mae: 0.5237 - val_loss: 0.6275 - val_mae: 0.6156
Epoch 90/100
337/337 - 0s - loss: 0.4455 - mae: 0.5399 - val_loss: 0.6323 - val_mae: 0.6188
Epoch 91/100
337/337 - 0s - loss: 0.4283 - mae: 0.5276 - val_loss: 0.6491 - val_mae: 0.6241
Epoch 92/100
337/337 - 0s - loss: 0.4082 - mae: 0.5030 - val_loss: 0.6307 - val_mae: 0.6173
```

```
Epoch 93/100
337/337 - 0s - loss: 0.4224 - mae: 0.5308 - val_loss: 0.6275 - val_mae: 0.6184
Epoch 94/100
337/337 - 0s - loss: 0.4239 - mae: 0.5329 - val_loss: 0.6378 - val_mae: 0.6230
Epoch 95/100
337/337 - 0s - loss: 0.4528 - mae: 0.5478 - val_loss: 0.6389 - val_mae: 0.6230
Epoch 96/100
337/337 - 0s - loss: 0.4616 - mae: 0.5452 - val_loss: 0.6359 - val_mae: 0.6200
Epoch 97/100
337/337 - 0s - loss: 0.4163 - mae: 0.5321 - val_loss: 0.6341 - val_mae: 0.6196
Epoch 98/100
337/337 - 0s - loss: 0.4100 - mae: 0.5174 - val_loss: 0.6392 - val_mae: 0.6214
Epoch 99/100
337/337 - 0s - loss: 0.4187 - mae: 0.5255 - val_loss: 0.6320 - val_mae: 0.6141
Epoch 100/100
337/337 - 0s - loss: 0.4323 - mae: 0.5324 - val_loss: 0.6319 - val_mae: 0.6120
```

In [18]:
```
# Evaluate the model on the test set
model.evaluate(test_data, test_targets, verbose = 2)
```

```
45/1 - 0s - loss: 0.5950 - mae: 0.6008
```

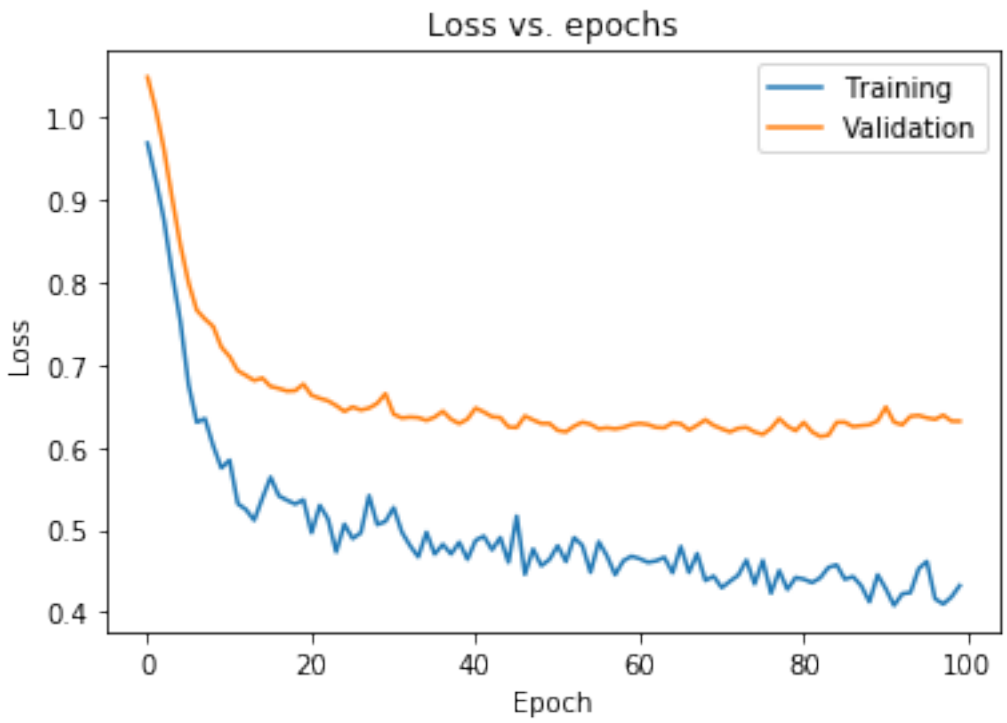
Out[18]:
```
[0.5838679790496826, 0.60075176]
```

Plot the learning curves

In [19]:
```
# 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 [20]: # Write a custom callback
         from tensorflow.keras.callbacks import Callback

         class TrainingCallback(Callback):

             def on_train_begin(self, logs = None):
                 print('..... Start training .......')

             def on_epoch_begin(self, epoch, logs = None):
                 print('Starting epoch ', epoch)

             def on_train_batch_begin(self, batch, logs = None):
                 print('\t Starting batch ', batch)

             def on_train_batch_end(self, batch, logs = None):
                 print('\t Finished batch', batch)

             def on_epoch_end(self, epoch, logs = None):
```

```
                print('Finished epoch')

            def on_train_end(self, logs = None):
                print('....... End training ........')
```

```
In [22]: # Re-build the model
         model = get_regularised_model(wd = 1e-5, rate = 0.3)
```

```
In [23]: # Compile the model
         model.compile(optimizer = 'adam', loss = 'mse', metrics = ['mae'])
```

Train the model with the callback

```
In [25]: # Train the model, with some of the data reserved for validation
         model.fit(train_data, train_targets, epochs = 2, batch_size = 128, callbacks = [Traini

Train on 397 samples
... Start training ...
Starting epoch  0
Epoch 1/2
         Starting batch  0
         Finished batch 0
128/397 [========>...] - ETA: 2s - loss: 0.9201 - mae: 0.8095         Starting batch  1
         Finished batch 1
         Starting batch  2
         Finished batch 2
         Starting batch  3
         Finished batch 3
Finished epoch
397/397 [==============================] - 1s 3ms/sample - loss: 0.9960 - mae: 0.8533
Starting epoch  1
Epoch 2/2
         Starting batch  0
         Finished batch 0
128/397 [========>...] - ETA: 0s - loss: 0.9978 - mae: 0.8567         Starting batch  1
         Finished batch 1
         Starting batch  2
         Finished batch 2
         Starting batch  3
         Finished batch 3
Finished epoch
397/397 [==============================] - 0s 29us/sample - loss: 0.9816 - mae: 0.8472
... End training ...
```

```
Out[25]: <tensorflow.python.keras.callbacks.History at 0x7f37386a8550>
```

```
In [28]: # Evaluate the model
         class TestingCallback(Callback):
```

```
            def on_test_begin(self, logs = None):
                print('........ Start testing ........')

            def on_test_batch_begin(self, batch, logs = None):
                print('\t Starting batch')

            def on_test_batch_end(self, batch, logs = None):
                print('\t Finished batch')

            def on_test_end(self, logs = None):
                print('..... End testing .......')

        model.evaluate(test_data, test_targets, verbose = 2, callbacks = [TestingCallback()])
```

```
... Start testing ...
        Starting batch
        Finished batch
        Starting batch
        Finished batch
45/1 - 0s - loss: 0.8869 - mae: 0.8245
... End testing ...
```

```
Out[28]: [0.9571111732059054, 0.8245286]
```

```
In [30]: # Make predictions with the model
        class PredictionCallback(Callback):

            def on_predict_begin(self, logs = None):
                print('..... Start prediction ......')

            def on_predict_batch_begin(self, batch, logs = None):
                print('\t Starting batch ', batch)

            def on_predict_batch_end(self, batch, logs = None):
                print('\t Finishing batch ', batch)

            def on_predict_end(self, logs = None):
                print('...... End prediction .......')

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

```
... Start prediction ...
        Starting batch  0
        Finishing batch  0
        Starting batch  1
        Finishing batch  1
```

```
... End prediction ...
```

```
Out[30]: array([[ 0.00433877],
                [-0.01777761],
                [ 0.06010732],
                [-0.03390782],
                [-0.02373301],
                [ 0.07767402],
                [-0.00167221],
                [-0.00347514],
                [-0.01407206],
                [ 0.05968182],
                [ 0.0655968 ],
                [-0.03374614],
                [-0.01099645],
                [ 0.04375353],
                [ 0.06103193],
                [-0.02430987],
                [ 0.03179593],
                [ 0.02093973],
                [-0.0256697 ],
                [ 0.00566917],
                [ 0.0284248 ],
                [ 0.07721781],
                [ 0.04418122],
                [ 0.02958088],
                [ 0.01647894],
                [ 0.06688327],
                [-0.03621586],
                [-0.00362249],
                [-0.02287805],
                [ 0.01757196],
                [-0.01498775],
                [-0.02625222],
                [ 0.04791849],
                [-0.02672683],
                [-0.01930223],
                [-0.0087498 ],
                [ 0.01819121],
                [ 0.03932822],
                [-0.01132534],
                [ 0.01338506],
                [ 0.01549018],
                [-0.00234938],
                [-0.02639992],
                [ 0.06876606],
                [-0.0214993 ]], dtype=float32)
```

Early stopping / patience

Re-train the models with early stopping

```
In [18]: # Re-train the unregularised model
         unreg_model = get_model()
         unreg_model.compile(optimizer = 'adam', loss = 'mse')
         unreg_history = unreg_model.fit(train_data, train_targets, epochs = 100,
                                         validation_split = 0.15, batch_size = 64, verbose =
                                         callbacks = [tf.keras.callbacks.EarlyStopping(patie
```

```
Train on 337 samples, validate on 60 samples
Epoch 1/100
337/337 - 1s - loss: 0.9050 - val_loss: 1.1058
Epoch 2/100
337/337 - 0s - loss: 0.8623 - val_loss: 1.0284
Epoch 3/100
337/337 - 0s - loss: 0.8173 - val_loss: 0.9367
Epoch 4/100
337/337 - 0s - loss: 0.7705 - val_loss: 0.8415
Epoch 5/100
337/337 - 0s - loss: 0.7088 - val_loss: 0.7784
Epoch 6/100
337/337 - 0s - loss: 0.6464 - val_loss: 0.6501
Epoch 7/100
337/337 - 0s - loss: 0.5891 - val_loss: 0.5581
Epoch 8/100
337/337 - 0s - loss: 0.5382 - val_loss: 0.5133
Epoch 9/100
337/337 - 0s - loss: 0.5154 - val_loss: 0.4387
Epoch 10/100
337/337 - 0s - loss: 0.4973 - val_loss: 0.4634
Epoch 11/100
337/337 - 0s - loss: 0.4943 - val_loss: 0.4223
Epoch 12/100
337/337 - 0s - loss: 0.4837 - val_loss: 0.3816
Epoch 13/100
337/337 - 0s - loss: 0.4837 - val_loss: 0.3940
Epoch 14/100
337/337 - 0s - loss: 0.4772 - val_loss: 0.4140
Epoch 15/100
337/337 - 0s - loss: 0.4742 - val_loss: 0.3686
Epoch 16/100
337/337 - 0s - loss: 0.4736 - val_loss: 0.4248
Epoch 17/100
337/337 - 0s - loss: 0.4809 - val_loss: 0.4395
```

```
Epoch 18/100
337/337 - 0s - loss: 0.4744 - val_loss: 0.3691
Epoch 19/100
337/337 - 0s - loss: 0.4683 - val_loss: 0.4226
Epoch 20/100
337/337 - 0s - loss: 0.4657 - val_loss: 0.4042
Epoch 21/100
337/337 - 0s - loss: 0.4709 - val_loss: 0.3824
Epoch 22/100
337/337 - 0s - loss: 0.4599 - val_loss: 0.4404
Epoch 23/100
337/337 - 0s - loss: 0.4623 - val_loss: 0.4025
Epoch 24/100
337/337 - 0s - loss: 0.4548 - val_loss: 0.3875
Epoch 25/100
337/337 - 0s - loss: 0.4548 - val_loss: 0.4073
```

```
In [19]: # Evaluate the model on the test set
         unreg_model.evaluate(test_data, test_targets, batch_size = 64, verbose = 2)
```

```
45/1 - 0s - loss: 0.5357
```

```
Out[19]: 0.5357096195220947
```

```
In [20]: # Re-train the regularised model
         reg_model = get_regularised_model(wd = 1e-6, rate = 0.3)
         reg_model.compile(optimizer = 'adam', loss = 'mse')
         reg_history = reg_model.fit(train_data, train_targets, epochs = 100,
                                     validation_split = 0.15, batch_size = 64, verbose = 2,
                                     callbacks = [tf.keras.callbacks.EarlyStopping(patience
```

```
Train on 337 samples, validate on 60 samples
Epoch 1/100
337/337 - 1s - loss: 0.9084 - val_loss: 1.1510
Epoch 2/100
337/337 - 0s - loss: 0.8899 - val_loss: 1.1146
Epoch 3/100
337/337 - 0s - loss: 0.8627 - val_loss: 1.0709
Epoch 4/100
337/337 - 0s - loss: 0.8310 - val_loss: 1.0123
Epoch 5/100
337/337 - 0s - loss: 0.7949 - val_loss: 0.9539
Epoch 6/100
337/337 - 0s - loss: 0.7690 - val_loss: 0.8889
Epoch 7/100
337/337 - 0s - loss: 0.7030 - val_loss: 0.8122
Epoch 8/100
```

```
337/337 - 0s - loss: 0.6849 - val_loss: 0.7283
Epoch 9/100
337/337 - 0s - loss: 0.6354 - val_loss: 0.6506
Epoch 10/100
337/337 - 0s - loss: 0.6218 - val_loss: 0.6063
Epoch 11/100
337/337 - 0s - loss: 0.6177 - val_loss: 0.6051
Epoch 12/100
337/337 - 0s - loss: 0.5852 - val_loss: 0.5656
Epoch 13/100
337/337 - 0s - loss: 0.6061 - val_loss: 0.5045
Epoch 14/100
337/337 - 0s - loss: 0.5816 - val_loss: 0.4931
Epoch 15/100
337/337 - 0s - loss: 0.5887 - val_loss: 0.5231
Epoch 16/100
337/337 - 0s - loss: 0.5616 - val_loss: 0.4874
Epoch 17/100
337/337 - 0s - loss: 0.5474 - val_loss: 0.4729
Epoch 18/100
337/337 - 0s - loss: 0.5623 - val_loss: 0.4922
Epoch 19/100
337/337 - 0s - loss: 0.5464 - val_loss: 0.4884
Epoch 20/100
337/337 - 0s - loss: 0.5766 - val_loss: 0.4946
Epoch 21/100
337/337 - 0s - loss: 0.5353 - val_loss: 0.4985
Epoch 22/100
337/337 - 0s - loss: 0.5442 - val_loss: 0.4594
Epoch 23/100
337/337 - 0s - loss: 0.5565 - val_loss: 0.4467
Epoch 24/100
337/337 - 0s - loss: 0.5678 - val_loss: 0.4703
Epoch 25/100
337/337 - 0s - loss: 0.5230 - val_loss: 0.4581
Epoch 26/100
337/337 - 0s - loss: 0.5446 - val_loss: 0.4402
Epoch 27/100
337/337 - 0s - loss: 0.5456 - val_loss: 0.4624
Epoch 28/100
337/337 - 0s - loss: 0.4994 - val_loss: 0.4550
Epoch 29/100
337/337 - 0s - loss: 0.5183 - val_loss: 0.4459
Epoch 30/100
337/337 - 0s - loss: 0.5103 - val_loss: 0.4484
Epoch 31/100
337/337 - 0s - loss: 0.5253 - val_loss: 0.4316
Epoch 32/100
```

```
337/337 - 0s - loss: 0.5114 - val_loss: 0.4418
Epoch 33/100
337/337 - 0s - loss: 0.5261 - val_loss: 0.4708
Epoch 34/100
337/337 - 0s - loss: 0.4894 - val_loss: 0.4495
Epoch 35/100
337/337 - 0s - loss: 0.5027 - val_loss: 0.4483
Epoch 36/100
337/337 - 0s - loss: 0.5288 - val_loss: 0.4451
Epoch 37/100
337/337 - 0s - loss: 0.5031 - val_loss: 0.4452
Epoch 38/100
337/337 - 0s - loss: 0.5013 - val_loss: 0.4539
Epoch 39/100
337/337 - 0s - loss: 0.5086 - val_loss: 0.4300
Epoch 40/100
337/337 - 0s - loss: 0.5354 - val_loss: 0.4266
Epoch 41/100
337/337 - 0s - loss: 0.4889 - val_loss: 0.4339
Epoch 42/100
337/337 - 0s - loss: 0.4803 - val_loss: 0.4342
Epoch 43/100
337/337 - 0s - loss: 0.4803 - val_loss: 0.4244
Epoch 44/100
337/337 - 0s - loss: 0.5172 - val_loss: 0.4334
Epoch 45/100
337/337 - 0s - loss: 0.5173 - val_loss: 0.4457
Epoch 46/100
337/337 - 0s - loss: 0.5327 - val_loss: 0.4464
Epoch 47/100
337/337 - 0s - loss: 0.5118 - val_loss: 0.4201
Epoch 48/100
337/337 - 0s - loss: 0.5174 - val_loss: 0.4179
Epoch 49/100
337/337 - 0s - loss: 0.4829 - val_loss: 0.4357
Epoch 50/100
337/337 - 0s - loss: 0.5177 - val_loss: 0.4223
Epoch 51/100
337/337 - 0s - loss: 0.4720 - val_loss: 0.4257
Epoch 52/100
337/337 - 0s - loss: 0.4618 - val_loss: 0.4280
Epoch 53/100
337/337 - 0s - loss: 0.4898 - val_loss: 0.4261
Epoch 54/100
337/337 - 0s - loss: 0.4972 - val_loss: 0.4241
Epoch 55/100
337/337 - 0s - loss: 0.4865 - val_loss: 0.4242
Epoch 56/100
```

```
337/337 - 0s - loss: 0.4758 - val_loss: 0.4077
Epoch 57/100
337/337 - 0s - loss: 0.4938 - val_loss: 0.4173
Epoch 58/100
337/337 - 0s - loss: 0.4842 - val_loss: 0.4098
Epoch 59/100
337/337 - 0s - loss: 0.4926 - val_loss: 0.3899
Epoch 60/100
337/337 - 0s - loss: 0.4761 - val_loss: 0.4010
Epoch 61/100
337/337 - 0s - loss: 0.4738 - val_loss: 0.4212
Epoch 62/100
337/337 - 0s - loss: 0.4715 - val_loss: 0.4766
Epoch 63/100
337/337 - 0s - loss: 0.5210 - val_loss: 0.4220
Epoch 64/100
337/337 - 0s - loss: 0.4626 - val_loss: 0.3972
Epoch 65/100
337/337 - 0s - loss: 0.4867 - val_loss: 0.4144
Epoch 66/100
337/337 - 0s - loss: 0.4605 - val_loss: 0.4398
Epoch 67/100
337/337 - 0s - loss: 0.4833 - val_loss: 0.4310
Epoch 68/100
337/337 - 0s - loss: 0.4717 - val_loss: 0.4181
Epoch 69/100
337/337 - 0s - loss: 0.4842 - val_loss: 0.4196
```

```
In [21]: # Evaluate the model on the test set
         reg_model.evaluate(test_data, test_targets, batch_size = 64, verbose = 2)
```

```
45/1 - 0s - loss: 0.5728
```

```
Out[21]: 0.572799026966095
```

Plot the learning curves

```
In [22]: # 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()
```