```
In [126]: inputs.dtype, targets.dtype
Out[126]: (torch.float32, torch.float32)

In [127]: dataset = TensorDataset(inputs, targets)

In [128]: val_percent = 0.15 # between 0.1 and 0.2
     ...: val_size = int(num_rows * val_percent)
     ...: train_size = num_rows - val_size
     ...:
     ...:
     ...: train_ds, val_ds = random_split(dataset, [train_size, val_size]) # Use the
random_split function to split dataset into 2 parts of the desired length

In [129]: batch_size = 128

In [130]: train_loader = DataLoader(train_ds, batch_size, shuffle=True)
     ...: val_loader = DataLoader(val_ds, batch_size)

In [131]: for xb, yb in train_loader:
     ...:     print("inputs:", xb)
     ...:     print("targets:", yb)
     ...:     break
     ...:
inputs: tensor([[41.0000,  0.0000, 31.2340,  1.0000,  0.0000,  3.0000],
        [27.0000,  0.0000, 30.3174,  1.0000,  0.0000,  1.0000],
        [18.0000,  0.0000, 29.2115,  0.0000,  0.0000,  0.0000],
        [19.0000,  1.0000, 32.1070,  0.0000,  0.0000,  3.0000],
        [20.0000,  1.0000, 29.2115,  5.0000,  0.0000,  0.0000],
        [36.0000,  1.0000, 32.8054,  1.0000,  0.0000,  1.0000],
        [51.0000,  0.0000, 33.1740,  1.0000,  0.0000,  3.0000],
        [53.0000,  0.0000, 21.9317,  3.0000,  1.0000,  0.0000],
        [42.0000,  0.0000, 35.1091,  1.0000,  0.0000,  1.0000],
        [39.0000,  1.0000, 34.2410,  2.0000,  1.0000,  3.0000],
        [25.0000,  0.0000, 25.9863,  2.0000,  0.0000,  1.0000],
        [45.0000,  0.0000, 29.9730,  2.0000,  0.0000,  3.0000],
        [38.0000,  0.0000, 29.7693,  1.0000,  0.0000,  2.0000],
        [19.0000,  0.0000, 28.0330,  0.0000,  0.0000,  3.0000],
        [33.0000,  1.0000, 23.8668,  2.0000,  0.0000,  1.0000],
        [41.0000,  1.0000, 35.9385,  2.0000,  0.0000,  1.0000],
        [42.0000,  1.0000, 27.4607,  3.0000,  1.0000,  1.0000],
        [26.0000,  0.0000, 16.6791,  2.0000,  1.0000,  0.0000],
        [62.0000,  1.0000, 29.9487,  3.0000,  1.0000,  1.0000],
        [33.0000,  1.0000, 34.6775,  1.0000,  1.0000,  2.0000],
        [50.0000,  1.0000, 43.4269,  1.0000,  0.0000,  2.0000],
        [45.0000,  1.0000, 38.6109,  0.0000,  0.0000,  0.0000],
        [19.0000,  1.0000, 25.2491,  1.0000,  1.0000,  1.0000],
        [59.0000,  1.0000, 24.6962,  0.0000,  0.0000,  1.0000],
        [24.0000,  1.0000, 31.7190,  0.0000,  1.0000,  3.0000],
        [22.0000,  1.0000, 27.4607,  1.0000,  0.0000,  1.0000],
        [27.0000,  0.0000, 17.4163,  2.0000,  1.0000,  0.0000],
        [44.0000,  0.0000, 31.3698,  1.0000,  0.0000,  2.0000],
        [26.0000,  0.0000, 21.5631,  0.0000,  0.0000,  1.0000],
        [59.0000,  0.0000, 35.4244,  1.0000,  0.0000,  2.0000],
        [42.0000,  0.0000, 22.6689,  0.0000,  1.0000,  0.0000],
        [60.0000,  0.0000, 26.7235,  0.0000,  0.0000,  0.0000],
        [53.0000,  1.0000, 27.7420,  3.0000,  0.0000,  3.0000],
        [51.0000,  0.0000, 33.0770,  0.0000,  0.0000,  2.0000],
        [42.0000,  0.0000, 28.1300,  1.0000,  0.0000,  3.0000],
        [18.0000,  0.0000, 30.9624,  0.0000,  0.0000,  0.0000],
        [44.0000,  0.0000, 36.9182,  0.0000,  1.0000,  2.0000],
        [37.0000,  1.0000, 29.9487,  3.0000,  0.0000,  1.0000],
```

```
[53.0000,  1.0000, 20.2730,  0.0000,  1.0000,  2.0000],
[58.0000,  1.0000, 29.3958,  0.0000,  0.0000,  0.0000],
[30.0000,  1.0000, 37.6651,  1.0000,  0.0000,  2.0000],
[19.0000,  1.0000, 21.9317,  0.0000,  0.0000,  1.0000],
[61.0000,  1.0000, 22.9454,  0.0000,  0.0000,  0.0000],
[28.0000,  1.0000, 35.3080,  1.0000,  1.0000,  3.0000],
[46.0000,  0.0000, 33.5620,  1.0000,  1.0000,  3.0000],
[28.0000,  1.0000, 26.1706,  2.0000,  0.0000,  0.0000],
[36.0000,  0.0000, 25.1230,  1.0000,  0.0000,  3.0000],
[47.0000,  1.0000, 35.1043,  0.0000,  1.0000,  2.0000],
[27.0000,  1.0000, 32.1604,  2.0000,  0.0000,  1.0000],
[55.0000,  1.0000, 31.6899,  1.0000,  0.0000,  2.0000],
[38.0000,  0.0000, 38.9455,  0.0000,  0.0000,  2.0000],
[48.0000,  0.0000, 26.4471,  1.0000,  0.0000,  0.0000],
[56.0000,  1.0000, 31.1467,  1.0000,  0.0000,  0.0000],
[26.0000,  1.0000, 30.1331,  0.0000,  0.0000,  1.0000],
[28.0000,  0.0000, 25.1569,  1.0000,  0.0000,  1.0000],
[52.0000,  1.0000, 35.5990,  0.0000,  0.0000,  3.0000],
[59.0000,  0.0000, 34.1440,  0.0000,  0.0000,  2.0000],
[52.0000,  1.0000, 46.3078,  1.0000,  0.0000,  2.0000],
[24.0000,  1.0000, 22.6980,  0.0000,  0.0000,  3.0000],
[55.0000,  0.0000, 32.5289,  2.0000,  0.0000,  1.0000],
[30.0000,  0.0000, 32.3301,  1.0000,  0.0000,  2.0000],
[22.0000,  1.0000, 31.1467,  0.0000,  0.0000,  1.0000],
[47.0000,  0.0000, 32.3447,  0.0000,  0.0000,  0.0000],
[20.0000,  1.0000, 27.1842,  1.0000,  1.0000,  1.0000],
[28.0000,  0.0000, 25.7147,  2.0000,  0.0000,  2.0000],
[25.0000,  1.0000, 25.9960,  3.0000,  0.0000,  3.0000],
[54.0000,  1.0000, 28.3240,  1.0000,  0.0000,  3.0000],
[64.0000,  0.0000, 22.3003,  0.0000,  1.0000,  2.0000],
[18.0000,  0.0000, 37.0249,  0.0000,  0.0000,  2.0000],
[30.0000,  1.0000, 27.8293,  3.0000,  1.0000,  1.0000],
[30.0000,  0.0000, 22.9454,  3.0000,  1.0000,  1.0000],
[49.0000,  1.0000, 25.0648,  2.0000,  1.0000,  1.0000],
[18.0000,  1.0000, 24.4198,  0.0000,  1.0000,  0.0000],
[56.0000,  0.0000, 34.7260,  1.0000,  0.0000,  3.0000],
[33.0000,  1.0000, 41.1862,  1.0000,  0.0000,  2.0000],
[20.0000,  0.0000, 35.8900,  5.0000,  0.0000,  3.0000],
[56.0000,  0.0000, 27.9214,  0.0000,  0.0000,  0.0000],
[36.0000,  1.0000, 30.5550,  0.0000,  0.0000,  3.0000],
[53.0000,  1.0000, 30.2252,  1.0000,  0.0000,  1.0000],
[55.0000,  1.0000, 32.0100,  0.0000,  0.0000,  2.0000],
[21.0000,  1.0000, 21.6310,  1.0000,  0.0000,  3.0000],
[63.0000,  0.0000, 35.7445,  0.0000,  0.0000,  2.0000],
[43.0000,  0.0000, 34.5708,  1.0000,  0.0000,  2.0000],
[63.0000,  0.0000, 26.9078,  0.0000,  1.0000,  0.0000],
[62.0000,  0.0000, 24.2500,  0.0000,  0.0000,  3.0000],
[49.0000,  0.0000, 29.8566,  1.0000,  0.0000,  0.0000],
[21.0000,  0.0000, 31.6996,  2.0000,  0.0000,  1.0000],
[50.0000,  1.0000, 24.5410,  0.0000,  0.0000,  2.0000],
[42.0000,  1.0000, 34.7260,  2.0000,  0.0000,  3.0000],
[50.0000,  0.0000, 26.5392,  0.0000,  0.0000,  0.0000],
[37.0000,  1.0000, 28.7508,  0.0000,  0.0000,  1.0000],
[22.0000,  0.0000, 23.5710,  0.0000,  0.0000,  3.0000],
[50.0000,  1.0000, 26.6313,  1.0000,  0.0000,  0.0000],
[19.0000,  1.0000, 29.4880,  0.0000,  0.0000,  3.0000],
[59.0000,  0.0000, 26.8884,  3.0000,  0.0000,  2.0000],
[51.0000,  1.0000, 24.0511,  2.0000,  1.0000,  1.0000],
[20.0000,  1.0000, 34.5563,  3.0000,  1.0000,  1.0000],
[30.0000,  0.0000, 31.4280,  1.0000,  0.0000,  3.0000],
[49.0000,  1.0000, 28.9351,  1.0000,  0.0000,  0.0000],
[32.0000,  1.0000, 45.1341,  2.0000,  0.0000,  2.0000],
```

```
        [50.0000,  1.0000, 25.8020,  0.0000,  0.0000,  3.0000],
        [29.0000,  0.0000, 37.6651,  3.0000,  0.0000,  2.0000],
        [39.0000,  0.0000, 24.1433,  3.0000,  1.0000,  0.0000],
        [30.0000,  0.0000, 19.3515,  3.0000,  0.0000,  1.0000],
        [29.0000,  1.0000, 33.3680,  0.0000,  1.0000,  3.0000],
        [23.0000,  1.0000, 33.3680,  0.0000,  0.0000,  3.0000],
        [59.0000,  0.0000, 26.9951,  3.0000,  0.0000,  2.0000],
        [29.0000,  0.0000, 24.8320,  4.0000,  0.0000,  3.0000],
        [44.0000,  1.0000, 35.9870,  2.0000,  0.0000,  3.0000],
        [19.0000,  1.0000, 24.7883,  1.0000,  0.0000,  1.0000],
        [44.0000,  0.0000, 28.9157,  2.0000,  0.0000,  2.0000],
        [27.0000,  0.0000, 29.6723,  1.0000,  0.0000,  0.0000],
        [44.0000,  1.0000, 21.1945,  3.0000,  0.0000,  0.0000],
        [20.0000,  1.0000, 34.2507,  1.0000,  0.0000,  2.0000],
        [39.0000,  0.0000, 23.4982,  5.0000,  0.0000,  1.0000],
        [31.0000,  0.0000, 25.0260,  2.0000,  0.0000,  3.0000],
        [44.0000,  1.0000, 29.2940,  2.0000,  1.0000,  3.0000],
        [62.0000,  1.0000, 36.2780,  0.0000,  0.0000,  3.0000],
        [45.0000,  0.0000, 26.9951,  2.0000,  0.0000,  2.0000],
        [62.0000,  1.0000, 26.7235,  1.0000,  0.0000,  1.0000],
        [22.0000,  0.0000, 20.6416,  3.0000,  0.0000,  1.0000],
        [49.0000,  1.0000, 29.3910,  0.0000,  0.0000,  3.0000],
        [37.0000,  0.0000, 46.1720,  2.0000,  1.0000,  3.0000],
        [27.0000,  1.0000, 32.6502,  0.0000,  0.0000,  2.0000],
        [63.0000,  1.0000, 40.2259,  0.0000,  0.0000,  2.0000],
        [63.0000,  1.0000, 38.6060,  3.0000,  0.0000,  3.0000],
        [58.0000,  0.0000, 22.0869,  0.0000,  0.0000,  2.0000],
        [32.0000,  1.0000, 29.1291,  1.0000,  0.0000,  2.0000]])
targets: tensor([[ 7792.3550],
        [ 4549.4819],
        [ 2533.9927],
        [26545.3984],
        [ 5652.3188],
        [ 6184.0767],
        [11353.6064],
        [28604.3926],
        [ 8560.1895],
        [46119.4727],
        [ 4817.4800],
        [ 9798.0303],
        [ 6873.3560],
        [ 2004.6960],
        [ 6046.1343],
        [ 8355.5576],
        [37705.5781],
        [16623.9902],
        [53725.8867],
        [44025.1602],
        [10417.5400],
        [ 8565.6641],
        [18918.5293],
        [13943.7412],
        [39643.7656],
        [ 3034.8994],
        [17257.5664],
        [ 8778.7783],
        [ 3652.7310],
        [32531.0820],
        [22959.4590],
        [15199.6582],
        [12941.4346],
        [10676.0967],
```

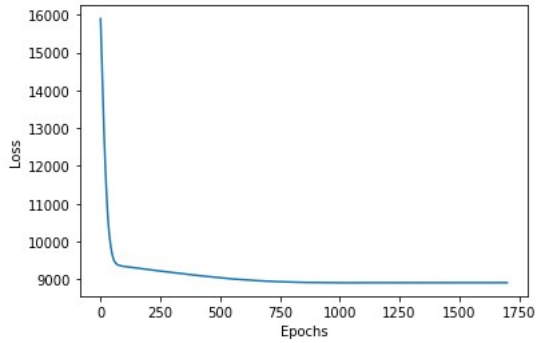
```
[ 8108.2383],
[ 2536.8779],
[56217.9062],
[ 7816.3926],
[24375.1914],
[13728.9941],
[21807.6484],
[ 1872.7416],
[15099.0439],
[58873.7422],
[47910.8438],
[ 5100.3584],
[ 6293.3164],
[47927.4922],
[ 4667.5195],
[12428.6094],
[ 6211.1274],
[10864.3379],
[13527.4512],
[ 3104.5037],
[ 4753.6880],
[10516.2500],
[14081.2109],
[11211.2471],
[ 2265.0562],
[14110.1416],
[ 4773.6831],
[ 2363.6235],
[24010.6016],
[20194.4375],
[ 4991.5068],
[ 4492.0459],
[12001.5107],
[31093.6016],
[ 1876.4186],
[23857.8867],
[21580.7559],
[27378.3262],
[17845.9082],
[13425.2490],
[13025.7217],
[ 5555.2246],
[13407.1357],
[ 5062.5679],
[12031.2764],
[23898.7129],
[ 2418.5420],
[15971.1641],
[ 8447.5859],
[33951.6406],
[15468.7900],
[11245.0996],
[29921.7930],
[ 9709.0674],
[ 8234.1084],
[29505.0625],
[ 5782.3687],
[ 2473.0393],
[11060.3115],
[ 1444.7439],
[16101.3037],
[27562.4902],
```

```
        [43085.1445],
        [ 4772.1963],
        [10681.2305],
        [ 5389.3472],
        [ 9711.1455],
        [ 5908.9951],
        [24908.9199],
        [ 6547.4453],
        [41627.3555],
        [ 2100.8694],
        [16101.4795],
        [ 6565.1973],
        [ 8901.3877],
        [ 2554.7991],
        [ 9452.0850],
        [19315.8730],
        [10224.8105],
        [31882.9316],
        [10310.6650],
        [ 5674.9106],
        [44848.3281],
        [14926.2617],
        [ 9793.1221],
        [16028.3164],
        [ 4940.7119],
        [ 9334.1816],
        [53030.5391],
        [ 2873.1765],
        [15416.1992],
        [17445.5801],
        [13608.8496],
        [ 4685.6216]])

In [132]: input_size = len(input_cols)
     ...: output_size = len(output_cols)
     ...: input_size, output_size
Out[132]: (6, 1)

In [133]: class InsuranceModel(nn.Module):
     ...:     def __init__(self):
     ...:         super().__init__()
     ...:         self.linear = nn.Linear(input_size, output_size)  # fill this
(hint: use input_size & output_size defined above)
     ...:
     ...:     def forward(self, xb):
     ...:         out = self.linear(xb)                          # fill this
     ...:         return out
     ...:
     ...:     def training_step(self, batch):
     ...:         inputs, targets = batch
     ...:         # Generate predictions
     ...:         out = self(inputs)
     ...:         # Calcuate loss
     ...:         loss = F.l1_loss(out, targets)                      # fill this
     ...:         return loss
     ...:
     ...:     def validation_step(self, batch):
     ...:         inputs, targets = batch
     ...:         # Generate predictions
     ...:         out = self(inputs)
     ...:         # Calculate loss
     ...:         loss = F.l1_loss(out, targets)                          # fill
```

```
this
     ...:         return {'val_loss': loss.detach()}
     ...:
     ...:     def validation_epoch_end(self, outputs):
     ...:         batch_losses = [x['val_loss'] for x in outputs]
     ...:         epoch_loss = torch.stack(batch_losses).mean()   # Combine losses
     ...:         return {'val_loss': epoch_loss.item()}
     ...:
     ...:     def epoch_end(self, epoch, result, num_epochs):
     ...:         # Print result every 20th epoch
     ...:         if (epoch+1) % 20 == 0 or epoch == num_epochs-1:
     ...:             print("Epoch [{}], val_loss: {:.4f}".format(epoch+1,
result['val_loss']))
     ...:

In [134]: model = InsuranceModel()

In [135]: list(model.parameters())
Out[135]:
[Parameter containing:
 tensor([[-0.3960,  0.1436,  0.0387,  0.3483,  0.2037,  0.3219]],
        requires_grad=True), Parameter containing:
 tensor([0.1723], requires_grad=True)]

In [136]: def evaluate(model, val_loader):
     ...:     outputs = [model.validation_step(batch) for batch in val_loader]
     ...:     return model.validation_epoch_end(outputs)
     ...:
     ...: def fit(epochs, lr, model, train_loader, val_loader,
opt_func=torch.optim.SGD):
     ...:     history = []
     ...:     optimizer = opt_func(model.parameters(), lr)
     ...:     for epoch in range(epochs):
     ...:         # Training Phase
     ...:         for batch in train_loader:
     ...:             loss = model.training_step(batch)
     ...:             loss.backward()
     ...:             optimizer.step()
     ...:             optimizer.zero_grad()
     ...:         # Validation phase
     ...:         result = evaluate(model, val_loader)
     ...:         model.epoch_end(epoch, result, epochs)
     ...:         history.append(result)
     ...:     return history
     ...:

In [137]: result = evaluate(model, val_loader) # Use the the evaluate function
     ...: print(result)
{'val_loss': 15897.75390625}

In [138]: epochs = 1000
     ...: lr = 1e-2
     ...: history1 = fit(epochs, lr, model, train_loader, val_loader)
Epoch [20], val_loss: 11958.7617
Epoch [40], val_loss: 9993.6934
Epoch [60], val_loss: 9451.5273
Epoch [80], val_loss: 9361.6602
Epoch [100], val_loss: 9336.6543
Epoch [120], val_loss: 9320.7852
Epoch [140], val_loss: 9305.4238
Epoch [160], val_loss: 9291.0059
Epoch [180], val_loss: 9274.4961
```

```
Epoch [200], val_loss: 9257.8623
Epoch [220], val_loss: 9239.9258
Epoch [240], val_loss: 9222.4785
Epoch [260], val_loss: 9207.7256
Epoch [280], val_loss: 9192.8877
Epoch [300], val_loss: 9176.6455
Epoch [320], val_loss: 9162.2871
Epoch [340], val_loss: 9147.8086
Epoch [360], val_loss: 9132.5820
Epoch [380], val_loss: 9118.5908
Epoch [400], val_loss: 9103.8740
Epoch [420], val_loss: 9089.5039
Epoch [440], val_loss: 9075.8965
Epoch [460], val_loss: 9062.7246
Epoch [480], val_loss: 9048.6152
Epoch [500], val_loss: 9036.6133
Epoch [520], val_loss: 9023.9453
Epoch [540], val_loss: 9012.8311
Epoch [560], val_loss: 9002.6914
Epoch [580], val_loss: 8994.0967
Epoch [600], val_loss: 8985.7197
Epoch [620], val_loss: 8976.9980
Epoch [640], val_loss: 8968.4463
Epoch [660], val_loss: 8960.9766
Epoch [680], val_loss: 8953.6719
Epoch [700], val_loss: 8947.3652
Epoch [720], val_loss: 8941.3926
Epoch [740], val_loss: 8937.0703
Epoch [760], val_loss: 8932.5195
Epoch [780], val_loss: 8928.3633
Epoch [800], val_loss: 8924.0664
Epoch [820], val_loss: 8919.5781
Epoch [840], val_loss: 8916.3740
Epoch [860], val_loss: 8913.7227
Epoch [880], val_loss: 8912.6182
Epoch [900], val_loss: 8911.2832
Epoch [920], val_loss: 8909.8799
Epoch [940], val_loss: 8909.0430
Epoch [960], val_loss: 8908.3281
Epoch [980], val_loss: 8907.0879
Epoch [1000], val_loss: 8906.2852

In [139]: epochs = 300
     ...: lr = 1e-3
     ...: history2 = fit(epochs, lr, model, train_loader, val_loader)
Epoch [20], val_loss: 8906.3262
Epoch [40], val_loss: 8906.4902
Epoch [60], val_loss: 8906.6621
Epoch [80], val_loss: 8906.8447
Epoch [100], val_loss: 8906.9141
Epoch [120], val_loss: 8907.0195
Epoch [140], val_loss: 8907.2461
Epoch [160], val_loss: 8907.3516
Epoch [180], val_loss: 8907.3574
Epoch [200], val_loss: 8907.4170
Epoch [220], val_loss: 8907.5508
Epoch [240], val_loss: 8907.5713
Epoch [260], val_loss: 8907.6533
Epoch [280], val_loss: 8907.6787
Epoch [300], val_loss: 8907.8770

In [140]: epochs = 300
```

```
     ...: lr = 1e-4
     ...: history3 = fit(epochs, lr, model, train_loader, val_loader)
Epoch [20], val_loss: 8907.8770
Epoch [40], val_loss: 8907.8887
Epoch [60], val_loss: 8907.9082
Epoch [80], val_loss: 8907.9141
Epoch [100], val_loss: 8907.9229
Epoch [120], val_loss: 8907.9346
Epoch [140], val_loss: 8907.9395
Epoch [160], val_loss: 8907.9531
Epoch [180], val_loss: 8907.9736
Epoch [200], val_loss: 8907.9834
Epoch [220], val_loss: 8907.9941
Epoch [240], val_loss: 8907.9902
Epoch [260], val_loss: 8908.0000
Epoch [280], val_loss: 8908.0186
Epoch [300], val_loss: 8908.0176

In [141]: epochs = 50
     ...: lr = 1e-5
     ...: history4 = fit(epochs, lr, model, train_loader, val_loader)
Epoch [20], val_loss: 8908.0176
Epoch [40], val_loss: 8908.0176
Epoch [50], val_loss: 8908.0176

In [142]: epochs = 50
     ...: lr = 1e-6
     ...: history5 = fit(epochs, lr, model, train_loader, val_loader)
Epoch [20], val_loss: 8908.0186
Epoch [40], val_loss: 8908.0176
Epoch [50], val_loss: 8908.0176

In [143]: val_loss = [result] + history1 + history2 + history3 + history4 + history5
     ...:
     ...: val_loss_array = [v['val_loss'] for v in val_loss]
     ...: plt.plot(val_loss_array)
     ...: plt.xlabel('Epochs')
     ...: plt.ylabel('Loss')
Out[143]: Text(0, 0.5, 'Loss')
```




```
In [144]: def predict_single(input, target, model):
     ...:     inputs = input.unsqueeze(0)
     ...:     predictions = model(inputs)                # fill this
     ...:     prediction = predictions[0].detach()
     ...:     print("Input:", input)
     ...:     print("Target:", target)
     ...:     print("Prediction:", prediction)
     ...:

In [145]: input, target = val_ds[0]
```

```
     ...: predict_single(input, target, model)
Input: tensor([52.0000,  1.0000, 37.4420,  2.0000,  0.0000,  3.0000])
Target: tensor([11873.9873])
Prediction: tensor([11683.9893])

In [146]: input, target = val_ds[10]
     ...: predict_single(input, target, model)
Input: tensor([51.0000,  0.0000, 38.3150,  1.0000,  0.0000,  3.0000])
Target: tensor([11362.0781])
Prediction: tensor([11312.6377])

In [147]: input, target = val_ds[23]
     ...: predict_single(input, target, model)
Input: tensor([26.0000,  0.0000, 38.9795,  0.0000,  0.0000,  1.0000])
Target: tensor([3681.4319])
Prediction: tensor([4130.9175])

In [148]: #Finish

In [149]:
```