Josh Wattay
Machine Learning with Python
Task 2.2
CareerFoundry

Best performance currently achieved with hyperparameters of 20, 32, 128. Softmax is the best performing activation function at this point and the most stations identified was 6.

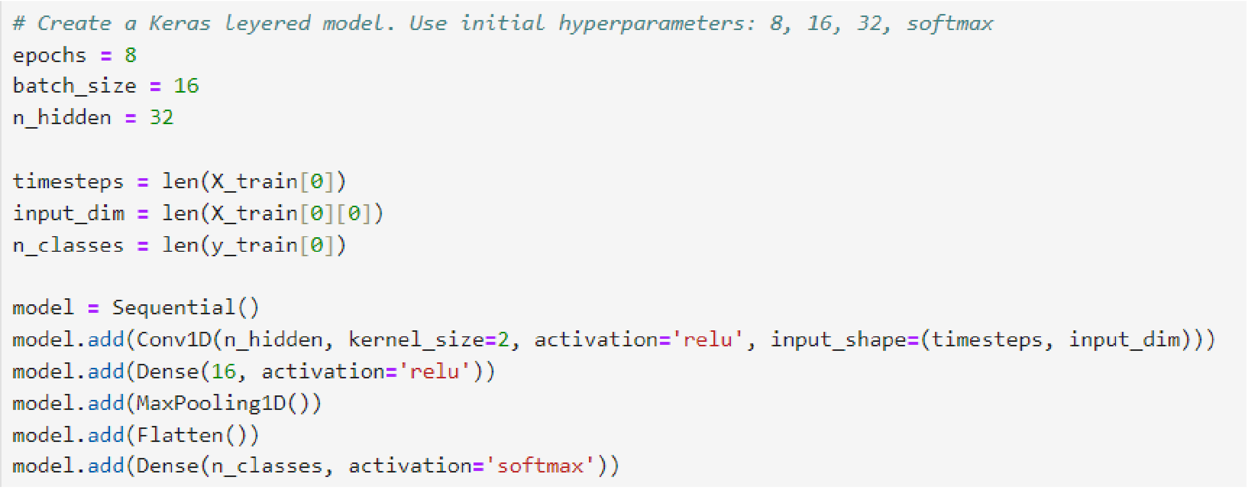
Convolution Neural Network (CNN) Models

Hyperparameters (epochs, batch_size, n_hidden)	**Activation Function**	**Accuracy**	**Stations Identified**
8, 16, 32	softmax	10.34%	4
16, 32, 64	softmax	9.65%	4
32, 64, 128	softmax	12.72%	4
50, 64, 128	Relu	8.48%	4
64, 64, 128	softmax	11.52%	4

Recurrent Neural Network (RNN)

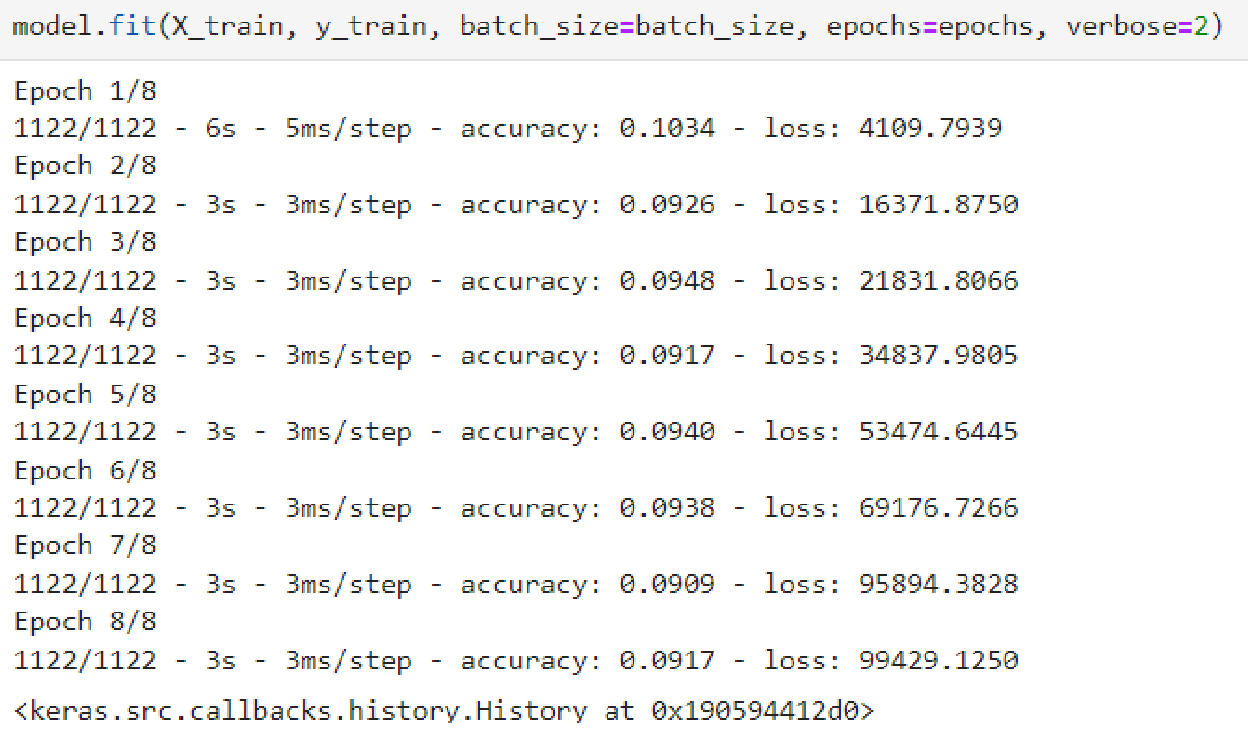
Hyperparameters (epochs, batch_size, n_hidden)	**Activation Function**	**Accuracy**	**Stations Identified**
8, 16, 32	Softmax	13.66%	5
20, 32, 128	Softmax	10.30%	6
32, 32, 128	Sigmoid	10.54%	2

Both models have terrible accuracy at the moment, and I was not able to have either model identify all 15 weather stations.



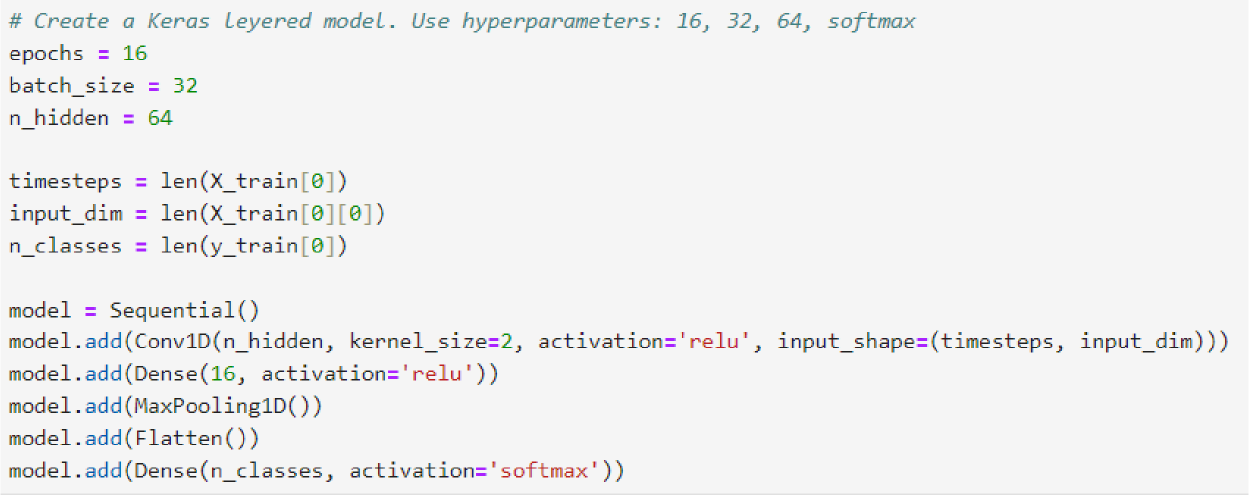
```
# Create a Keras leyered model. Use initial hyperparameters: 8, 16, 32, softmax
epochs = 8
batch_size = 16
n_hidden = 32

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='softmax'))
```

Josh Wattay
Machine Learning with Python
Task 2.2
CareerFoundry



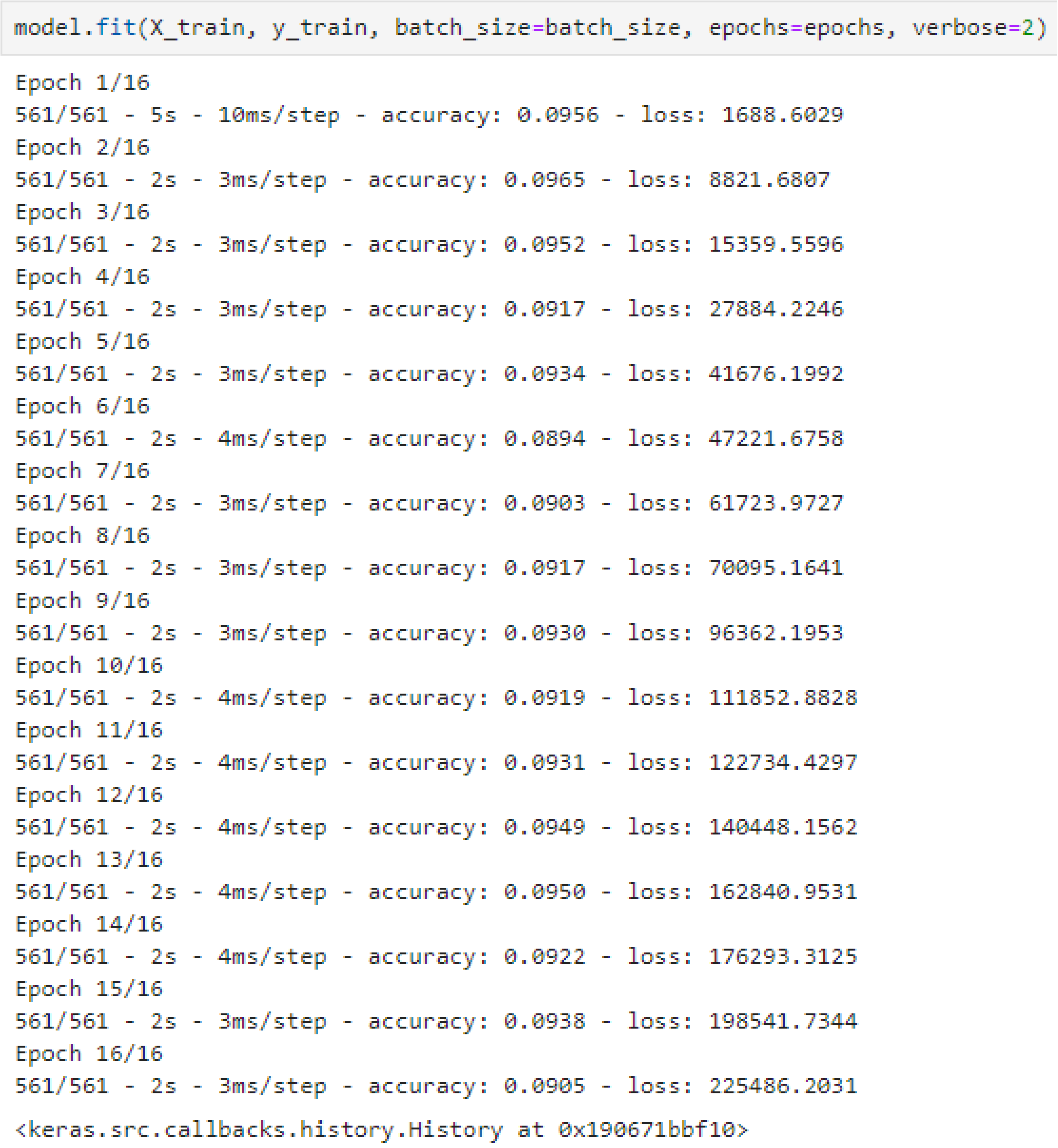
```
model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=2)
```

```
Epoch 1/8
1122/1122 - 6s - 5ms/step - accuracy: 0.1034 - loss: 4109.7939
Epoch 2/8
1122/1122 - 3s - 3ms/step - accuracy: 0.0926 - loss: 16371.8750
Epoch 3/8
1122/1122 - 3s - 3ms/step - accuracy: 0.0948 - loss: 21831.8066
Epoch 4/8
1122/1122 - 3s - 3ms/step - accuracy: 0.0917 - loss: 34837.9805
Epoch 5/8
1122/1122 - 3s - 3ms/step - accuracy: 0.0940 - loss: 53474.6445
Epoch 6/8
1122/1122 - 3s - 3ms/step - accuracy: 0.0938 - loss: 69176.7266
Epoch 7/8
1122/1122 - 3s - 3ms/step - accuracy: 0.0909 - loss: 95894.3828
Epoch 8/8
1122/1122 - 3s - 3ms/step - accuracy: 0.0917 - loss: 99429.1250
<keras.src.callbacks.history.History at 0x190594412d0>
```



```
# Create a Keras leyered model. Use hyperparameters: 16, 32, 64, softmax
epochs = 16
batch_size = 32
n_hidden = 64

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='softmax'))
```

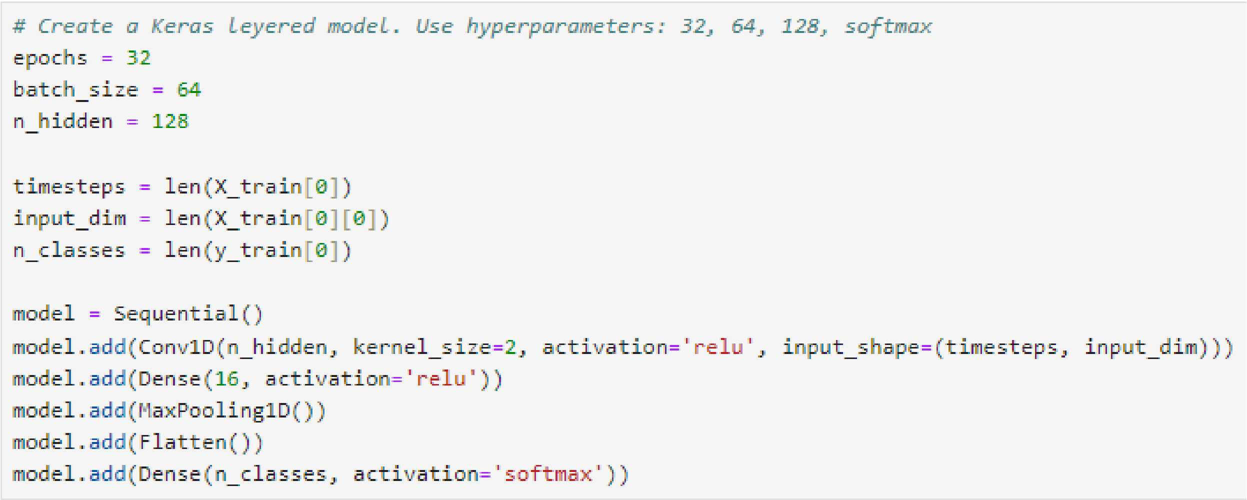
Josh Wattay
Machine Learning with Python
Task 2.2
CareerFoundry



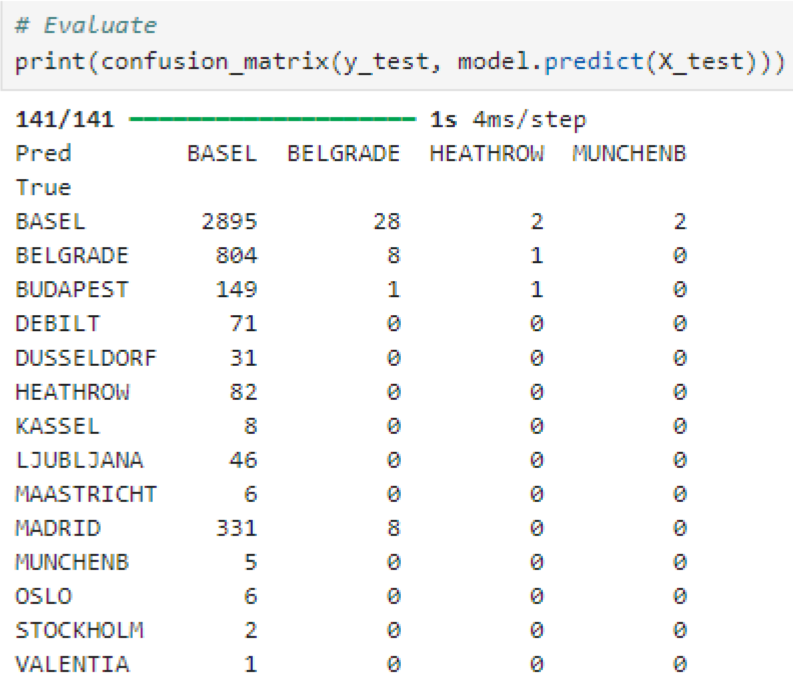
```
model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=2)
```

```
Epoch 1/16
561/561 - 5s - 10ms/step - accuracy: 0.0956 - loss: 1688.6029
Epoch 2/16
561/561 - 2s - 3ms/step - accuracy: 0.0965 - loss: 8821.6807
Epoch 3/16
561/561 - 2s - 3ms/step - accuracy: 0.0952 - loss: 15359.5596
Epoch 4/16
561/561 - 2s - 3ms/step - accuracy: 0.0917 - loss: 27884.2246
Epoch 5/16
561/561 - 2s - 3ms/step - accuracy: 0.0934 - loss: 41676.1992
Epoch 6/16
561/561 - 2s - 4ms/step - accuracy: 0.0894 - loss: 47221.6758
Epoch 7/16
561/561 - 2s - 3ms/step - accuracy: 0.0903 - loss: 61723.9727
Epoch 8/16
561/561 - 2s - 3ms/step - accuracy: 0.0917 - loss: 70095.1641
Epoch 9/16
561/561 - 2s - 3ms/step - accuracy: 0.0930 - loss: 96362.1953
Epoch 10/16
561/561 - 2s - 4ms/step - accuracy: 0.0919 - loss: 111852.8828
Epoch 11/16
561/561 - 2s - 4ms/step - accuracy: 0.0931 - loss: 122734.4297
Epoch 12/16
561/561 - 2s - 4ms/step - accuracy: 0.0949 - loss: 140448.1562
Epoch 13/16
561/561 - 2s - 4ms/step - accuracy: 0.0950 - loss: 162840.9531
Epoch 14/16
561/561 - 2s - 4ms/step - accuracy: 0.0922 - loss: 176293.3125
Epoch 15/16
561/561 - 2s - 3ms/step - accuracy: 0.0938 - loss: 198541.7344
Epoch 16/16
561/561 - 2s - 3ms/step - accuracy: 0.0905 - loss: 225486.2031

<keras.src.callbacks.history.History at 0x190671bbf10>
```

Josh Wattay
Machine Learning with Python
Task 2.2
CareerFoundry



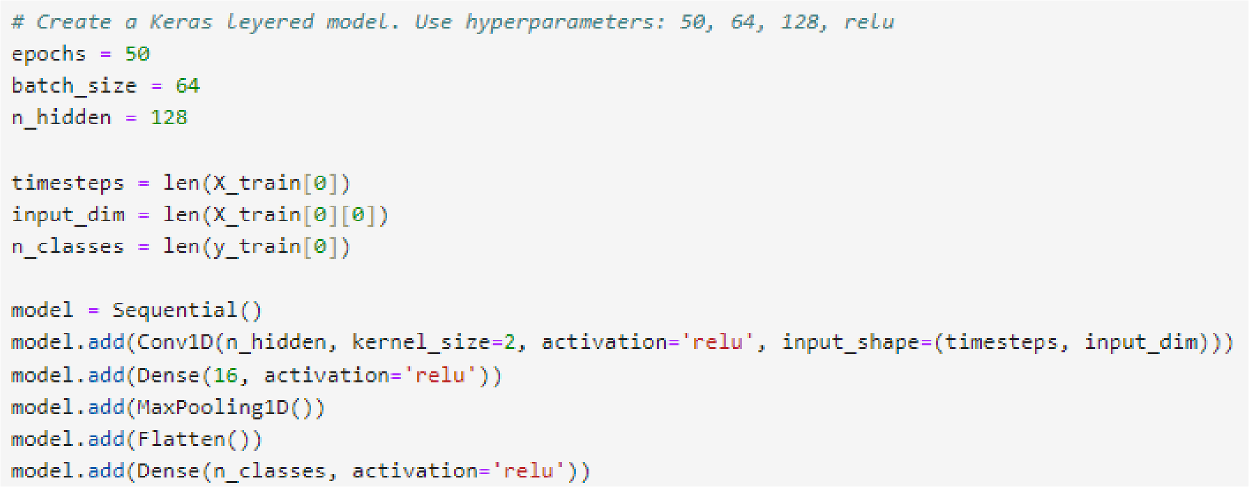
```
# Create a Keras leyered model. Use hyperparameters: 32, 64, 128, softmax
epochs = 32
batch_size = 64
n_hidden = 128

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='softmax'))
```



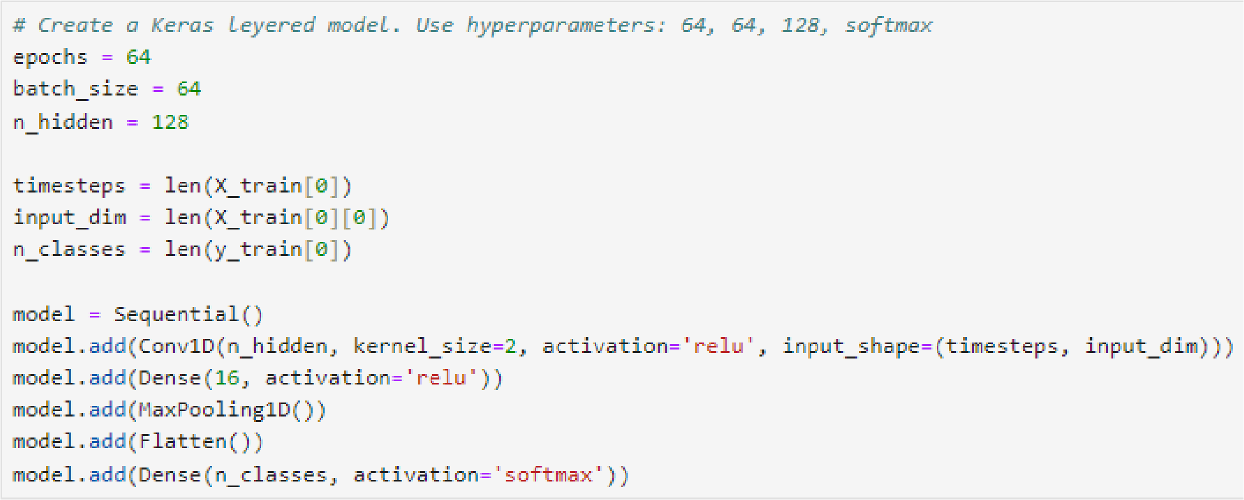
```
# Evaluate
print(confusion_matrix(y_test, model.predict(X_test)))
```

```
141/141 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step
Pred         BASEL  BELGRADE  HEATHROW  MUNCHENB
True
BASEL         2895        28         2         2
BELGRADE       804         8         1         0
BUDAPEST       149         1         1         0
DEBILT          71         0         0         0
DUSSELDORF      31         0         0         0
HEATHROW        82         0         0         0
KASSEL           8         0         0         0
LJUBLJANA       46         0         0         0
MAASTRICHT       6         0         0         0
MADRID         331         8         0         0
MUNCHENB         5         0         0         0
OSLO             6         0         0         0
STOCKHOLM        2         0         0         0
VALENTIA         1         0         0         0
```

Josh Wattay
Machine Learning with Python
Task 2.2
CareerFoundry

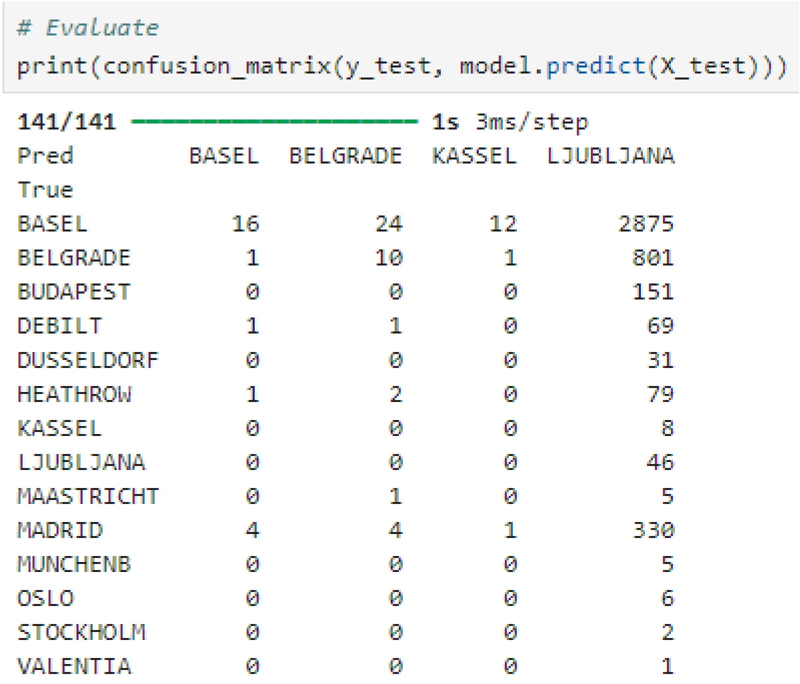


```
# Create a Keras leyered model. Use hyperparameters: 50, 64, 128, relu
epochs = 50
batch_size = 64
n_hidden = 128

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='relu'))
```



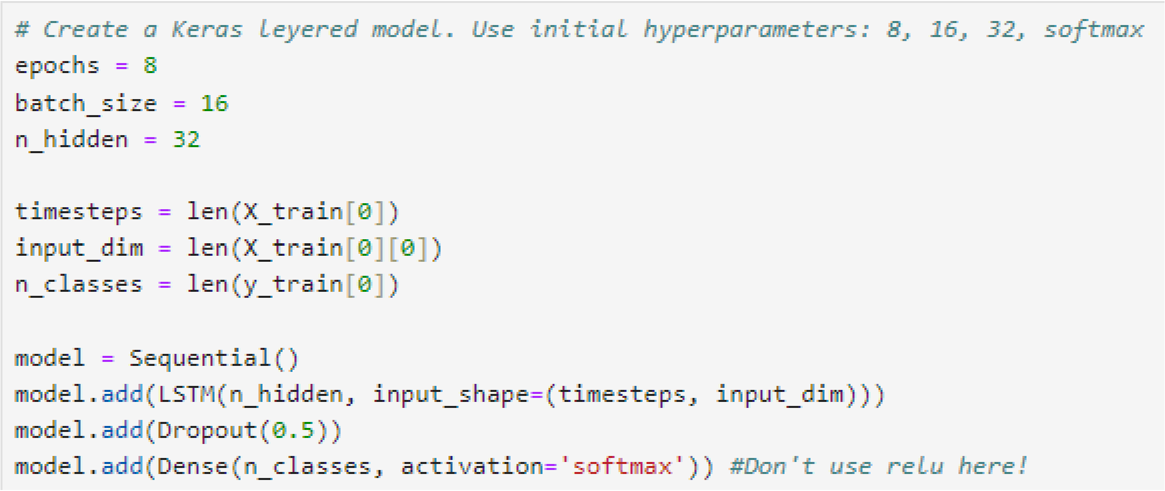
```
# Create a Keras leyered model. Use hyperparameters: 64, 64, 128, softmax
epochs = 64
batch_size = 64
n_hidden = 128

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(Conv1D(n_hidden, kernel_size=2, activation='relu', input_shape=(timesteps, input_dim)))
model.add(Dense(16, activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(n_classes, activation='softmax'))
```

Josh Wattay
Machine Learning with Python
Task 2.2
CareerFoundry



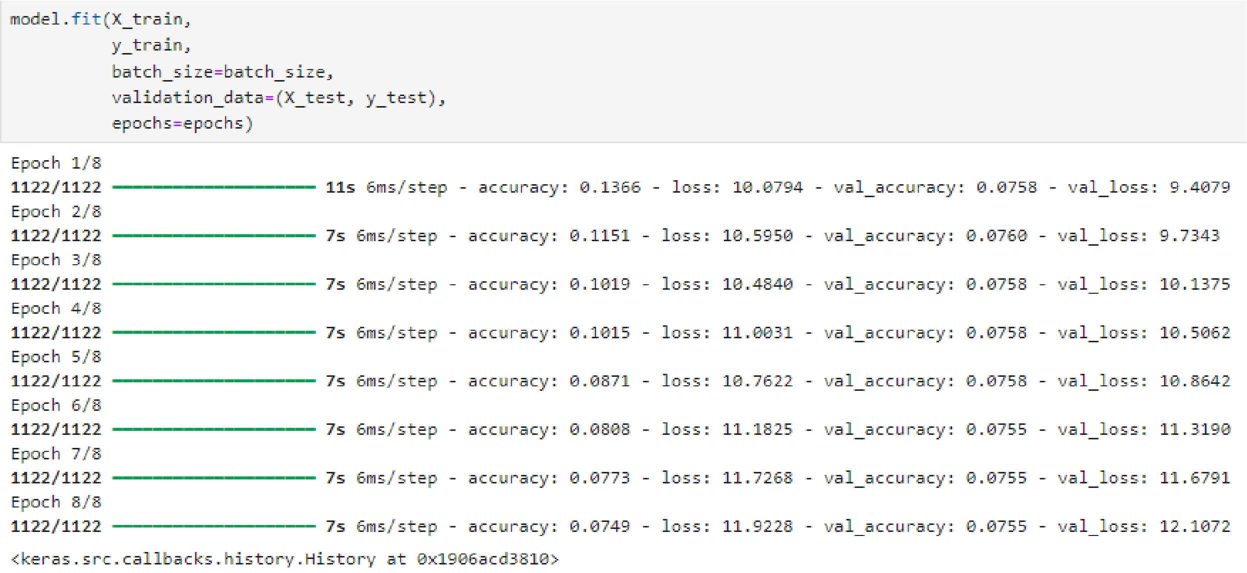
```
# Evaluate
print(confusion_matrix(y_test, model.predict(X_test)))
```

```
141/141 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
Pred        BASEL  BELGRADE  KASSEL  LJUBLJANA
True
BASEL          16        24      12       2875
BELGRADE        1        10       1        801
BUDAPEST        0         0       0        151
DEBILT          1         1       0         69
DUSSELDORF      0         0       0         31
HEATHROW        1         2       0         79
KASSEL          0         0       0          8
LJUBLJANA       0         0       0         46
MAASTRICHT      0         1       0          5
MADRID          4         4       1        330
MUNCHENB        0         0       0          5
OSLO            0         0       0          6
STOCKHOLM       0         0       0          2
VALENTIA        0         0       0          1
```



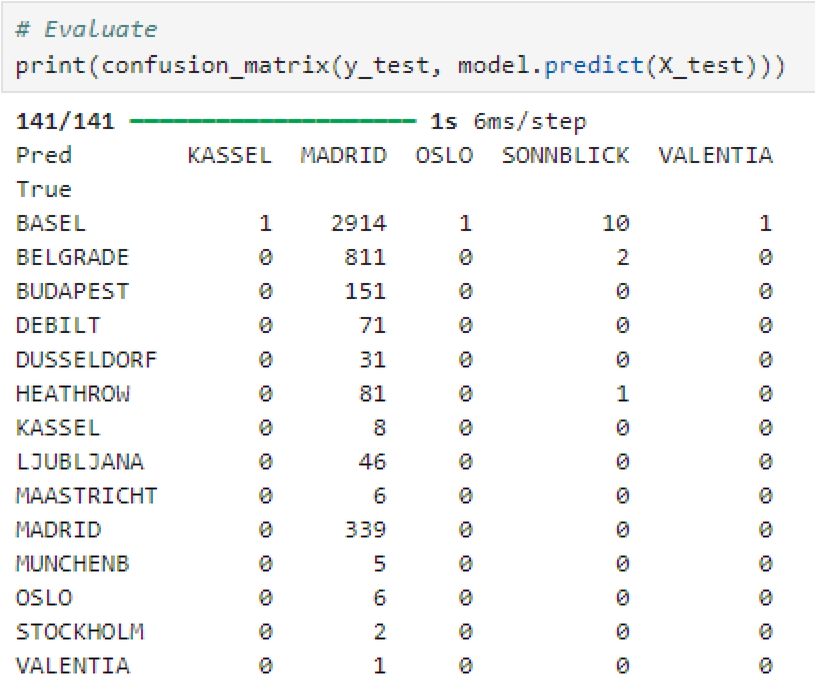
```
# Create a Keras leyered model. Use initial hyperparameters: 8, 16, 32, softmax
epochs = 8
batch_size = 16
n_hidden = 32

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='softmax')) #Don't use relu here!
```

Josh Wattay
Machine Learning with Python
Task 2.2
CareerFoundry



```
model.fit(X_train,
          y_train,
          batch_size=batch_size,
          validation_data=(X_test, y_test),
          epochs=epochs)
```

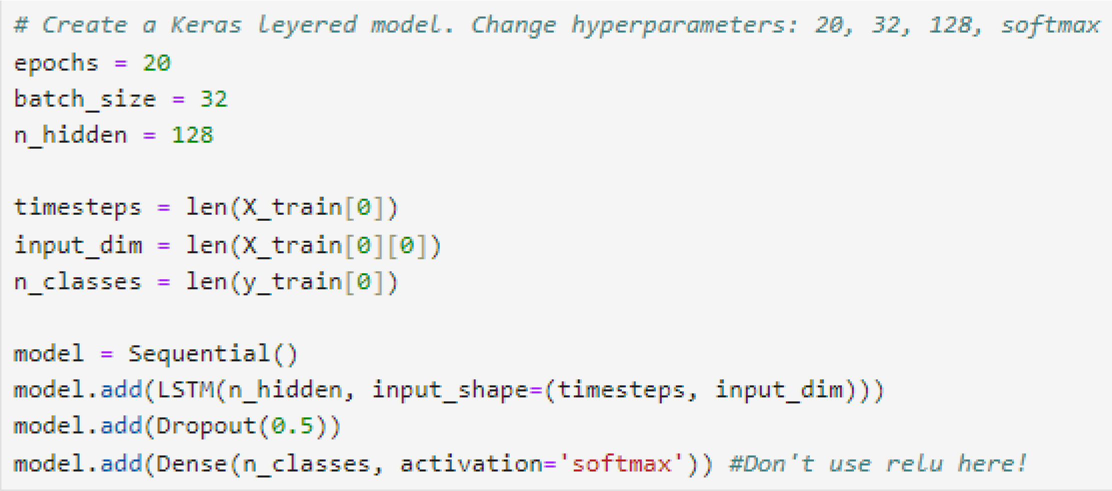
```
Epoch 1/8
1122/1122 ━━━━━━━━━━━━━━━━━━━━ 11s 6ms/step - accuracy: 0.1366 - loss: 10.0794 - val_accuracy: 0.0758 - val_loss: 9.4079
Epoch 2/8
1122/1122 ━━━━━━━━━━━━━━━━━━━━ 7s 6ms/step - accuracy: 0.1151 - loss: 10.5950 - val_accuracy: 0.0760 - val_loss: 9.7343
Epoch 3/8
1122/1122 ━━━━━━━━━━━━━━━━━━━━ 7s 6ms/step - accuracy: 0.1019 - loss: 10.4840 - val_accuracy: 0.0758 - val_loss: 10.1375
Epoch 4/8
1122/1122 ━━━━━━━━━━━━━━━━━━━━ 7s 6ms/step - accuracy: 0.1015 - loss: 11.0031 - val_accuracy: 0.0758 - val_loss: 10.5062
Epoch 5/8
1122/1122 ━━━━━━━━━━━━━━━━━━━━ 7s 6ms/step - accuracy: 0.0871 - loss: 10.7622 - val_accuracy: 0.0758 - val_loss: 10.8642
Epoch 6/8
1122/1122 ━━━━━━━━━━━━━━━━━━━━ 7s 6ms/step - accuracy: 0.0808 - loss: 11.1825 - val_accuracy: 0.0755 - val_loss: 11.3190
Epoch 7/8
1122/1122 ━━━━━━━━━━━━━━━━━━━━ 7s 6ms/step - accuracy: 0.0773 - loss: 11.7268 - val_accuracy: 0.0755 - val_loss: 11.6791
Epoch 8/8
1122/1122 ━━━━━━━━━━━━━━━━━━━━ 7s 6ms/step - accuracy: 0.0749 - loss: 11.9228 - val_accuracy: 0.0755 - val_loss: 12.1072
<keras.src.callbacks.history.History at 0x1906acd3810>
```



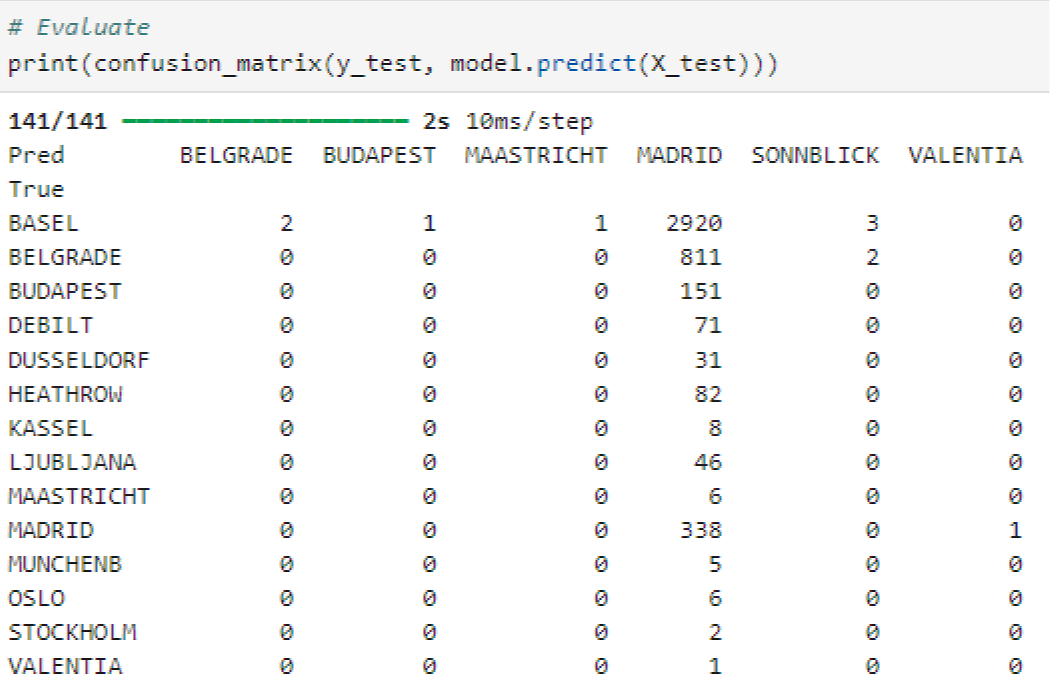
```
# Evaluate
print(confusion_matrix(y_test, model.predict(X_test)))
```

```
141/141 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step
Pred        KASSEL  MADRID  OSLO  SONNBLICK  VALENTIA
True
BASEL            1    2914     1         10         1
BELGRADE         0     811     0          2         0
BUDAPEST         0     151     0          0         0
DEBILT           0      71     0          0         0
DUSSELDORF       0      31     0          0         0
HEATHROW         0      81     0          1         0
KASSEL           0       8     0          0         0
LJUBLJANA        0      46     0          0         0
MAASTRICHT       0       6     0          0         0
MADRID           0     339     0          0         0
MUNCHENB         0       5     0          0         0
OSLO             0       6     0          0         0
STOCKHOLM        0       2     0          0         0
VALENTIA         0       1     0          0         0
```

Josh Wattay
Machine Learning with Python
Task 2.2
CareerFoundry



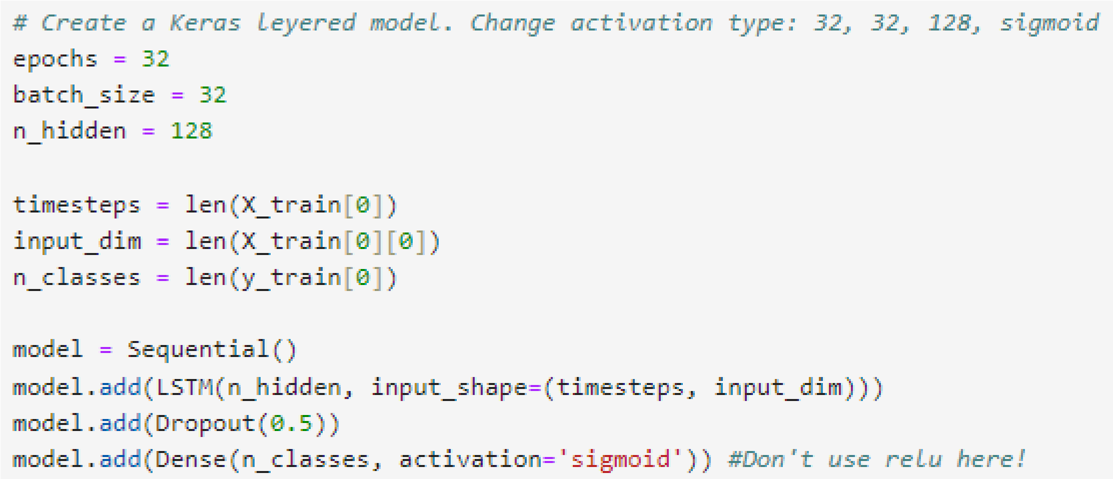
```
# Create a Keras leyered model. Change hyperparameters: 20, 32, 128, softmax
epochs = 20
batch_size = 32
n_hidden = 128

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='softmax')) #Don't use relu here!
```



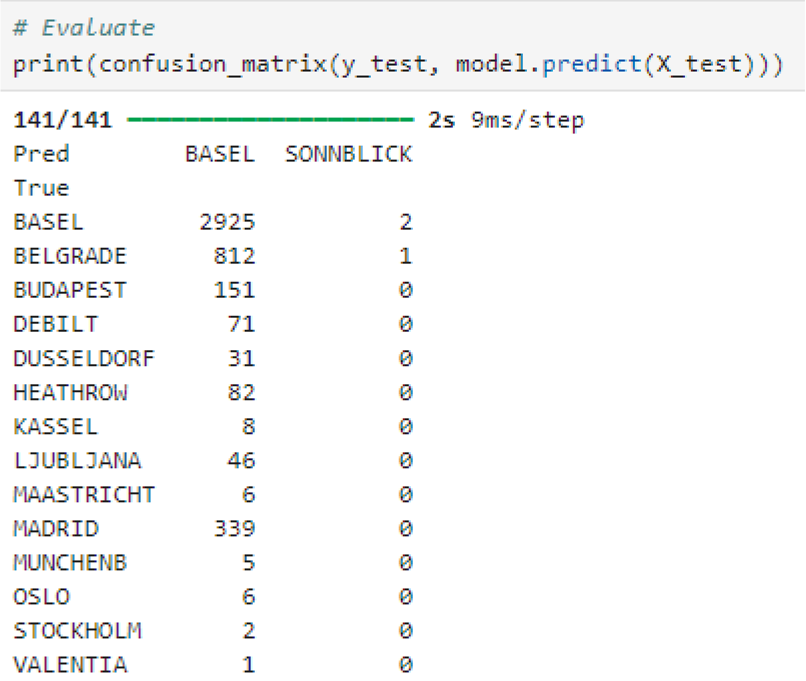
```
# Evaluate
print(confusion_matrix(y_test, model.predict(X_test)))
```

```
141/141 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step
Pred         BELGRADE  BUDAPEST  MAASTRICHT  MADRID  SONNBLICK  VALENTIA
True
BASEL               2         1           1    2920          3         0
BELGRADE            0         0           0     811          2         0
BUDAPEST            0         0           0     151          0         0
DEBILT              0         0           0      71          0         0
DUSSELDORF          0         0           0      31          0         0
HEATHROW            0         0           0      82          0         0
KASSEL              0         0           0       8          0         0
LJUBLJANA           0         0           0      46          0         0
MAASTRICHT          0         0           0       6          0         0
MADRID              0         0           0     338          0         1
MUNCHENB            0         0           0       5          0         0
OSLO                0         0           0       6          0         0
STOCKHOLM           0         0           0       2          0         0
VALENTIA            0         0           0       1          0         0
```

Josh Wattay
Machine Learning with Python
Task 2.2
CareerFoundry



```
# Create a Keras leyered model. Change activation type: 32, 32, 128, sigmoid
epochs = 32
batch_size = 32
n_hidden = 128

timesteps = len(X_train[0])
input_dim = len(X_train[0][0])
n_classes = len(y_train[0])

model = Sequential()
model.add(LSTM(n_hidden, input_shape=(timesteps, input_dim)))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='sigmoid')) #Don't use relu here!
```



```
# Evaluate
print(confusion_matrix(y_test, model.predict(X_test)))
```

```
141/141 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step
Pred        BASEL  SONNBLICK
True
BASEL        2925          2
BELGRADE      812          1
BUDAPEST      151          0
DEBILT         71          0
DUSSELDORF     31          0
HEATHROW       82          0
KASSEL          8          0
LJUBLJANA      46          0
MAASTRICHT      6          0
MADRID        339          0
MUNCHENB        5          0
OSLO            6          0
STOCKHOLM       2          0
VALENTIA        1          0
```