```
# LSTM (Many to One Multiple Numeric Feature)
# ===========================================

# https://stackabuse.com/solving-sequence-problems-with-lstm-in-keras/

%tensorflow_version 2.x

import tensorflow as tf
tf.__version__
```

```
TensorFlow 2.x selected.
'2.0.0'
```

```
# univariate lstm example
import numpy as np
from numpy import array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Bidirectional, Flatten
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.python.keras.callbacks import TensorBoard
# from tensorflow.keras.regularizers import l2

import matplotlib.pyplot as plt
from time import time
```

```
# define dataset
X1 = np.array([x+3 for x in range(0, 135, 3)])
print(X1)

X2 = np.array([x+5 for x in range(0, 225, 5)])
print(X2)


X = np.column_stack((X1, X2))
print(X)

y = np.array([ 24,  48,  72,  96, 120, 144, 168, 192, 216, 240, 264, 288, 312, 336,

```

```
[  3   6   9  12  15  18  21  24  27  30  33  36  39  42  45  48  51  54
  57  60  63  66  69  72  75  78  81  84  87  90  93  96  99 102 105 108
 111 114 117 120 123 126 129 132 135]
[  5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90
  95 100 105 110 115 120 125 130 135 140 145 150 155 160 165 170 175 180
 185 190 195 200 205 210 215 220 225]
[[  3   5]
 [  6  10]
 [  9  15]
 [ 12  20]
 [ 15  25]
 [ 18  30]
 [ 21  35]
 [ 24  40]
 [ 27  45]
 [ 30  50]
 [ 33  55]
 [ 36  60]
 [ 39  65]
 [ 42  70]
 [ 45  75]
 [ 48  80]
 [ 51  85]
 [ 54  90]
 [ 57  95]
 [ 60 100]
 [ 63 105]
 [ 66 110]
 [ 69 115]
 [ 72 120]
 [ 75 125]
 [ 78 130]
 [ 81 135]
 [ 84 140]
 [ 87 145]
 [ 90 150]
 [ 93 155]
 [ 96 160]
 [ 99 165]
 [102 170]
 [105 175]
 [108 180]
 [111 185]
 [114 190]
 [117 195]
 [120 200]
 [123 205]
 [126 210]
 [129 215]
 [132 220]
 [135 225]]
```

```
1 X[:5] , X.dtype
```

```
(array([[ 3,  5],
        [ 6, 10],
        [ 9, 15],
        [12, 20],
        [15, 25]]), dtype('int64'))
```

```
print("X.shape : {}".format(X.shape))

# reshape from [samples, timesteps] into [samples, timesteps, features]
X = X.reshape(15, 3, 2)

print("X.shape2 : {}".format(X.shape))
```

```
X.shape : (45, 2)
X.shape2 : (15, 3, 2)
```

```
X[:3] , X.dtype
```

```
(array([[[ 3,  5],
         [ 6, 10],
         [ 9, 15]],

        [[12, 20],
         [15, 25],
         [18, 30]],

        [[21, 35],
         [24, 40],
         [27, 45]]]), dtype('int64'))
```

```
y[:3], y.dtype
```

```
(array([24, 48, 72]), dtype('int64'))
```

```
X = tf.cast(X,tf.float32)
y = tf.cast(y,tf.float32)
```

```
X[:3] , y[:3]
```

```
(<tf.Tensor: id=571730, shape=(3, 3, 2), dtype=float32, numpy=
 array([[[ 3.,  5.],
         [ 6., 10.],
         [ 9., 15.]],

        [[12., 20.],
         [15., 25.],
         [18., 30.]],

        [[21., 35.],
         [24., 40.],
         [27., 45.]]], dtype=float32)>,
 <tf.Tensor: id=571734, shape=(3,), dtype=float32, numpy=array([24., 48., 72.], d
```

```
# %load_ext tensorboard
# tensorboard = TensorBoard(log_dir="logs/{}".format(time()), histogram_freq=1)
# %tensorboard --logdir logs
```

```
# es = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=5, verbose=1, mode
```

```
# define model

model = Sequential()
model.add(Bidirectional(LSTM(50, activation='relu', input_shape=(3, 2), return_sequ
# model.add(Dense(10, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse', metrics=['mse'])
history = model.fit(X, y, epochs=1000, validation_split=0.2, verbose=0)

model.summary()

```

```
Model: "sequential_34"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
bidirectional_34 (Bidirectio multiple                  21200     
_________________________________________________________________
dense_67 (Dense)             multiple                  101       
=================================================================
Total params: 21,301
Trainable params: 21,301
Non-trainable params: 0
_________________________________________________________________
```

```
# fit model
# model.fit(X, y, epochs=500, validation_split=0.2, verbose=1, callbacks=[tensorboa
# history = model.fit(X, y, epochs=500, validation_split=0.2, verbose=0, callbacks=
```

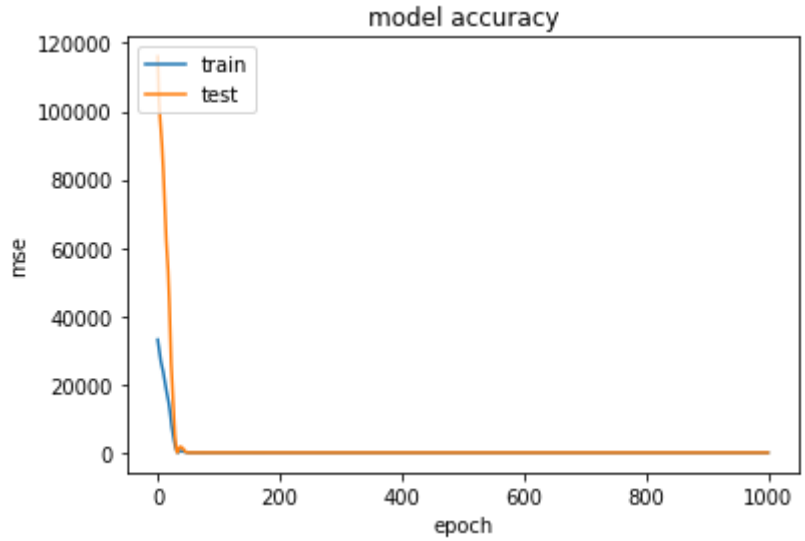
```
# list all data in history
print(history.history.keys())
```

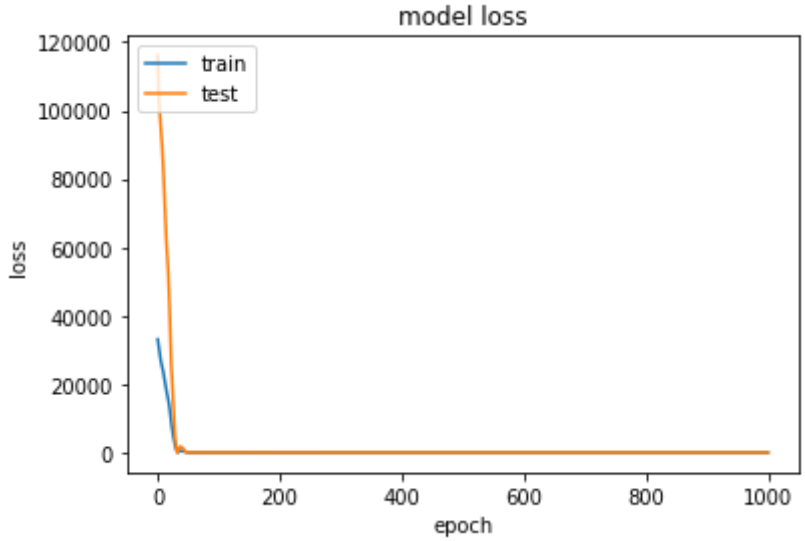
```
plt.show()

# summarize history for accuracy
plt.plot(history.history['mse'])
plt.plot(history.history['val_mse'])
plt.title('model accuracy')
plt.ylabel('mse')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'mse', 'val_loss', 'val_mse'])
```






```
# demonstrate prediction
```

```
x_input = array([[8, 51],
                 [11, 56],
                 [14,61]])
print("x_input.shape {}".format(x_input.shape))

x_input = x_input.reshape((1, 3, 2))
print("x_input.shape2 {}".format(x_input.shape))

x_input = tf.cast(x_input,tf.float32)
print("x_input: {}".format(x_input))

print("expected : ", 75)

yhat = model.predict(x_input, verbose=0)
print("yhat : ", yhat)
```

```
x_input.shape (3, 2)
x_input.shape2 (1, 3, 2)
x_input: [[[ 8. 51.]
  [11. 56.]
  [14. 61.]]]
expected :  75
yhat :  [[77.031746]]
```

```
1
```