```
# LSTM (Many to Many Single Numeric Feature)
# ==========================================

# https://stackabuse.com/solving-sequence-problems-with-lstm-in-keras-part-2/

%tensorflow_version 2.x

import tensorflow as tf
tf.__version__
```

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

```
# univariate lstm example
import tensorflow as tf
import numpy as np
from numpy import array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Bidirectional, Flatten, RepeatVector, Tim
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
X = list()
Y = list()
X = [x for x in range(5, 301, 5)]
Y = [y for y in range(20, 316, 5)]

X = np.array(X).reshape(20, 3, 1)
Y = np.array(Y).reshape(20, 3, 1)
```

```
# X = np.array(X)
# y = np.array(y)

X = X.astype('float32')
y = Y.astype('float32')
```

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

```
(array([[[ 5.],
         [10.],
         [15.]],

        [[20.],
         [25.],
         [30.]],

        [[35.],
         [40.],
         [45.]]], dtype=float32), array([[[20.],
         [25.],
         [30.]],

        [[35.],
         [40.],
         [45.]],

        [[50.],
         [55.],
         [60.]]], dtype=float32))
```

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

```
# %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(100, activation='relu', input_shape=(3, 1), return_seq
model.add(RepeatVector(3))
model.add(Bidirectional(LSTM(100, activation='relu', return_sequences=True)))
model.add(TimeDistributed(Dense(1)))

model.compile(optimizer='adam', loss='mse', metrics=['mse'])
history = model.fit(X, y, epochs=1000, validation_split=0.2, batch_size=3, verbose=

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
bidirectional_4 (Bidirection multiple                  81600     
_________________________________________________________________
repeat_vector_2 (RepeatVecto multiple                  0         
_________________________________________________________________
bidirectional_5 (Bidirection multiple                  240800    
_________________________________________________________________
time_distributed_2 (TimeDist multiple                  201       
=================================================================
Total params: 322,601
Trainable params: 322,601
Non-trainable params: 0
_________________________________________________________________
```

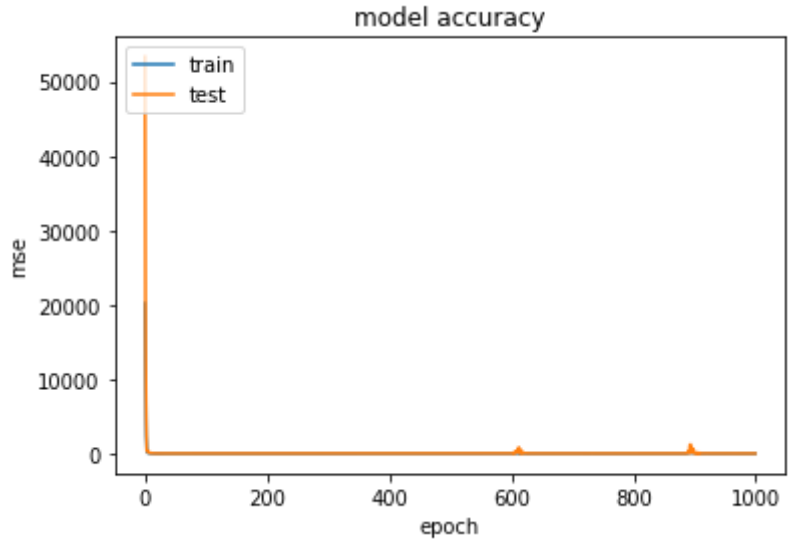
```
# fit model
# model.fit(X, y, epochs=500, validation_split=0.2, verbose=1, callbacks=[tensorboa
# history = model.fit(X, y, epochs=500, validation_split=0.2, verbose=0, callbacks=
```

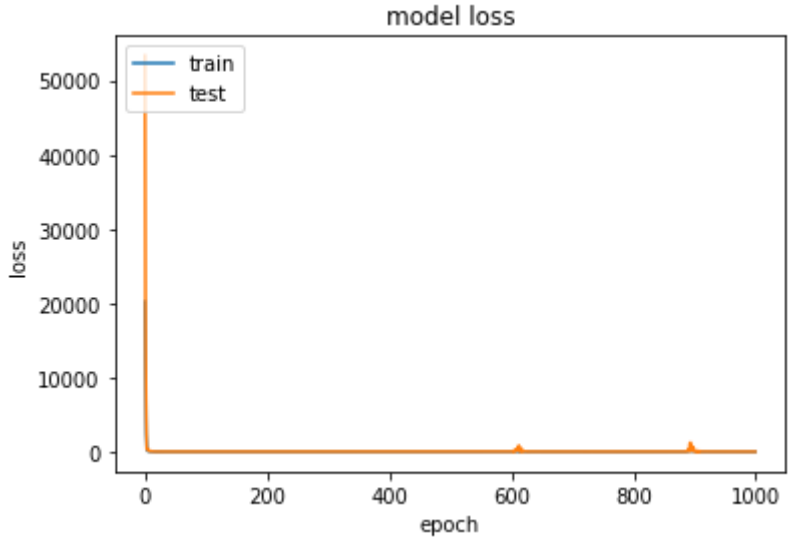
```
# list all data in history
print(history.history.keys())

# 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([300, 305, 310])
print("x_input.shape {}".format(x_input.shape))

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

x_input = tf.cast(x_input,tf.float32)

print("expected : 315, 320, 325")

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

```
x_input.shape (3,)
x_input.shape2 (1, 3, 1)
expected : 315, 320, 325
yhat :  [[[315.8522 ]
  [321.13278]
  [326.33057]]]
```

```
1
```