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

%tensorflow_version 2.x

import tensorflow as tf
tf.__version__
```

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

```
# univariate lstm example
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
X = array([[10, 20, 30], [20, 30, 40], [30, 40, 50], [40, 50, 60],  [50, 60, 70],
y = array([40, 50, 60, 70, 80, 90, 100])
```

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

```
(array([[10, 20, 30],
        [20, 30, 40],
        [30, 40, 50]]), dtype('int64'), array([40, 50, 60]), dtype('int64'))
```

```

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

# reshape from [samples, timesteps] into [samples, timesteps, features]
X = X.reshape((X.shape[0], X.shape[1], 1))

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

```

```
X.shape : (7, 3)
X.shape2 : (7, 3, 1)
```

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

```

```

```
(array([[[10],
         [20],
         [30]],

        [[20],
         [30],
         [40]],

        [[30],
         [40],
         [50]]]), dtype('int64'))
```

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

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

```
(<tf.Tensor: id=1230332, shape=(3, 3, 1), dtype=float32, numpy=
 array([[[10.],
         [20.],
         [30.]],

        [[20.],
         [30.],
         [40.]],

        [[30.],
         [40.],
         [50.]]], dtype=float32)>,
 <tf.Tensor: id=1230336, shape=(3,), dtype=float32, numpy=array([40., 50., 60.],
```

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

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

```
1 # define model
2
3 model = Sequential()
4 model.add(Bidirectional(LSTM(1000, activation='relu', input_shape=(3, 1), return_se
5 model.add(Flatten())
6 model.add(Dense(100, activation='relu'))
7 model.add(Dense(1))
8 model.compile(optimizer='adam', loss='mse', metrics=['mse'])
9 # history = model.fit(X, y, epochs=200, validation_split=0.2, batch_size=8, verbose
10 # history = model.fit(X, y, epochs=200, validation_split=0.2, verbose=0)
11 history = model.fit(X, y, epochs=200, validation_split=0.1, batch_size=3, verbose=(
12
13 model.summary()
14
```

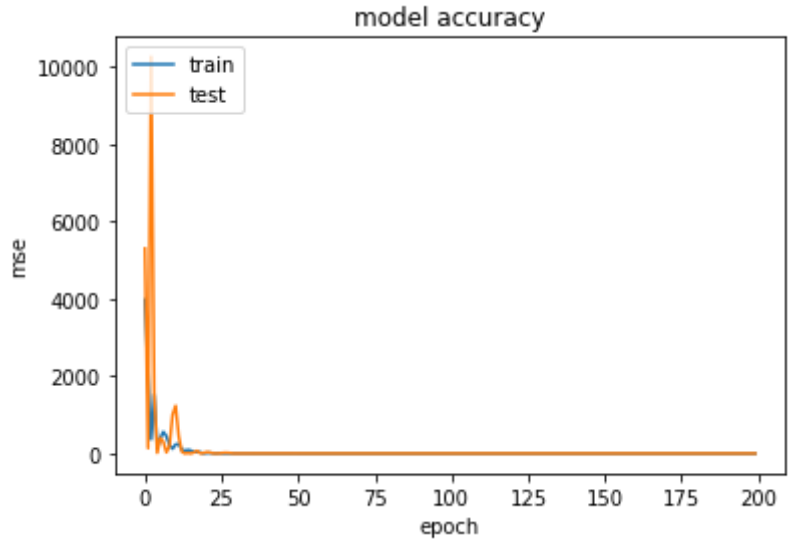
```
14
15
```

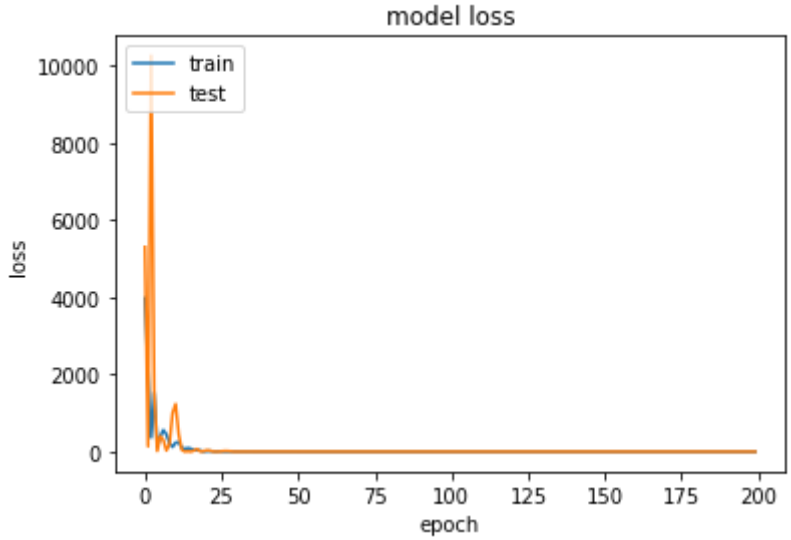
```
Model: "sequential_117"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
bidirectional_119 (Bidirecti multiple                  8016000   
_________________________________________________________________
flatten_59 (Flatten)         multiple                  0         
_________________________________________________________________
dense_246 (Dense)            multiple                  200100    
_________________________________________________________________
dense_247 (Dense)            multiple                  101       
=================================================================
Total params: 8,216,201
Trainable params: 8,216,201
Non-trainable params: 0
_________________________________________________________________
```

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

```
1 # list all data in history
2 print(history.history.keys())
3 
4 # summarize history for accuracy
5 plt.plot(history.history['mse'])
6 plt.plot(history.history['val_mse'])
7 plt.title('model accuracy')
8 plt.ylabel('mse')
9 plt.xlabel('epoch')
10 plt.legend(['train', 'test'], loc='upper left')
11 plt.show()
12 
13 # summarize history for loss
14 plt.plot(history.history['loss'])
15 plt.plot(history.history['val_loss'])
16 plt.title('model loss')
17 plt.ylabel('loss')
18 plt.xlabel('epoch')
19 plt.legend(['train', 'test'], loc='upper left')
20 plt.show()
```

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






```
# demonstrate prediction
x_input = array([80, 90, 100])
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)

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

```
x_input.shape (3,)
x_input.shape2 (1, 3, 1)
yhat :  [[110.528465]]
```

```
1
```