```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyswarms as ps
from pyswarms.utils.plotters import (plot_cost_history)
from sklearn.model_selection import train_test_split


data = pd.read_csv('milling1.csv')
```

```
data
```

	no	Spindle speed (rpm)	Feedrate	Feed per tooth (mm/tooth)	Cutting depth (mm)	Clamping torque of vise (N-m)	Accumulated removed volume per cutter (mm3)	Avg_Ra
0	1	3000	240	0.02	0.5	18	0.0	0.468
1	2	2000	400	0.05	0.5	18	1.2	0.402
2	3	1000	320	0.08	0.5	18	3.2	0.346
3	4	3000	240	0.02	0.7	18	4.8	0.634
4	5	2000	400	0.05	0.7	18	6.5	0.590
...	...	...	...	...	...	...	...	...
148	149	2000	400	0.05	1.0	18	61.4	1.236
149	150	2100	420	0.05	1.0	18	65.4	1.140
150	151	1900	532	0.07	1.0	18	69.6	1.096
151	152	2000	560	0.07	1.0	18	74.9	1.135
152	153	2100	588	0.07	1.0	18	80.5	1.093

```
from keras import Sequential
from keras.layers import LSTM, Dropout, Dense , TimeDistributed, Activation
import tensorflow as tf


col_names = list(data.columns)


                                                inMaxScaler
Creating a copy...
                                                ler()
df_1 = mm_scaler.fit_transform(data)

df = pd.DataFrame(df_1, columns = col_names)


sine_values = []
steps = 1
```

```
steps = 1
input_width = 5

x = []
y = []

for i in range(0, df.shape[0], steps):
  sine_values.append(df.at[i, 'Avg_Ra'])

for i in range(0, len(sine_values) - input_width, steps):
  x.append(sine_values[i: i+input_width])
  y.append(sine_values[i + input_width])

x = np.reshape(x,[-1, input_width,1])
y = np.reshape(y,[-1,1])


trainX , testX, trainY, testY = train_test_split(x,y,test_size = 0.2, shuffle =False)


model  = Sequential()
model.add(LSTM(32, return_sequences = False , input_shape=(None, 1)))
model.add(Dense(1,activation='linear'))

def root_mean_squared_error_loss(y_true, y_pred):
    return tf.sqrt(tf.reduce_mean(tf.math.squared_difference(y_true, y_pred)))

def r_square_loss(y_true,y_pred):
  from keras import backend as K
  SS_res = K.sum(K.square(y_true - y_pred))
  SS_tot = K.sum(K.square(y_true - K.mean(y_true)))
  return 1 - (1-SS_res/(SS_tot + K.epsilon()))


model.compile(loss = root_mean_squared_error_loss , optimizer = 'sgd' , metrics=[root_mean
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 32)                4352
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0
_________________________________________________________________
```

Creating a copy...

```
, batch_size = 50 , epochs = 100 , validation_split =
```

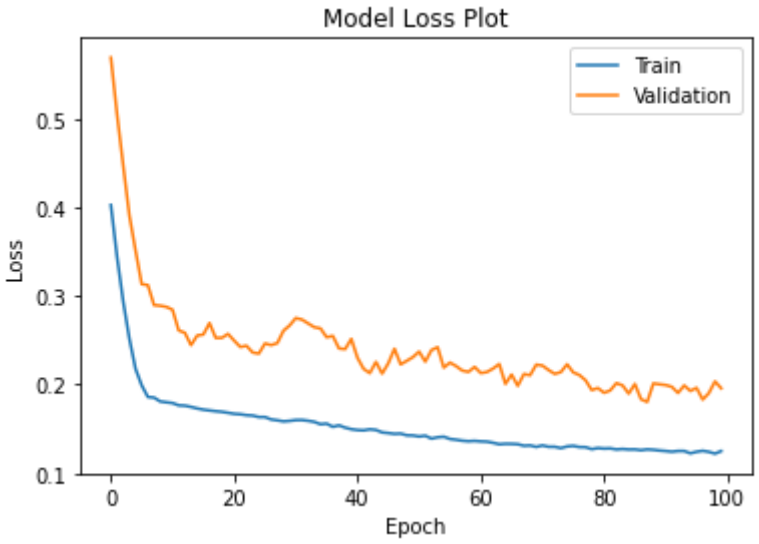
```
Epoch 71/100
3/3 [==============================] - 0s 15ms/step - loss: 0.1313 - root_mean_squ
Epoch 72/100
3/3 [==============================] - 0s 14ms/step - loss: 0.1298 - root_mean_squ
Epoch 73/100
3/3 [==============================] - 0s 20ms/step - loss: 0.1298 - root_mean_squ
```

```
Epoch 74/100
3/3 [==============================] - 0s 17ms/step - loss: 0.1283 - root_mean_squ
Epoch 75/100
3/3 [==============================] - 0s 17ms/step - loss: 0.1302 - root_mean_squ
Epoch 76/100
3/3 [==============================] - 0s 16ms/step - loss: 0.1308 - root_mean_squ
Epoch 77/100
3/3 [==============================] - 0s 19ms/step - loss: 0.1295 - root_mean_squ
Epoch 78/100
3/3 [==============================] - 0s 18ms/step - loss: 0.1293 - root_mean_squ
Epoch 79/100
3/3 [==============================] - 0s 19ms/step - loss: 0.1272 - root_mean_squ
Epoch 80/100
3/3 [==============================] - 0s 15ms/step - loss: 0.1284 - root_mean_squ
Epoch 81/100
3/3 [==============================] - 0s 17ms/step - loss: 0.1278 - root_mean_squ
Epoch 82/100
3/3 [==============================] - 0s 16ms/step - loss: 0.1280 - root_mean_squ
Epoch 83/100
3/3 [==============================] - 0s 15ms/step - loss: 0.1268 - root_mean_squ
Epoch 84/100
3/3 [==============================] - 0s 15ms/step - loss: 0.1273 - root_mean_squ
Epoch 85/100
3/3 [==============================] - 0s 14ms/step - loss: 0.1268 - root_mean_squ
Epoch 86/100
3/3 [==============================] - 0s 15ms/step - loss: 0.1268 - root_mean_squ
Epoch 87/100
3/3 [==============================] - 0s 14ms/step - loss: 0.1260 - root_mean_squ
Epoch 88/100
3/3 [==============================] - 0s 15ms/step - loss: 0.1268 - root_mean_squ
Epoch 89/100
3/3 [==============================] - 0s 14ms/step - loss: 0.1264 - root_mean_squ
Epoch 90/100
3/3 [==============================] - 0s 14ms/step - loss: 0.1256 - root_mean_squ
Epoch 91/100
3/3 [==============================] - 0s 16ms/step - loss: 0.1250 - root_mean_squ
Epoch 92/100
3/3 [==============================] - 0s 21ms/step - loss: 0.1242 - root_mean_squ
Epoch 93/100
3/3 [==============================] - 0s 16ms/step - loss: 0.1251 - root_mean_squ
Epoch 94/100
3/3 [==============================] - 0s 16ms/step - loss: 0.1250 - root_mean_squ
Epoch 95/100
3/3 [==============================] - 0s 16ms/step - loss: 0.1224 - root_mean_squ
Epoch 96/100
3/3 [==============================] - 0s 15ms/step - loss: 0.1242 - root_mean_squ
Epoch 97/100
3/3 [==============================] - 0s 20ms/step - loss: 0.1253 - root_mean_squ
Epoch 98/100
3/3 [==============================] - 0s 15ms/step - loss: 0.1241 - root_mean_squ
Epoch 99/100
3/3 [==============================] - 0s 14ms/step - loss: 0.1221 - root_mean_squ
```

Creating a copy...

```
plt.plot(historyNN.history['loss'])
plt.plot(historyNN.history['val_loss'])
plt.title('Model Loss Plot')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train','Validation'],loc = 'upper right')
```

```
plt.show()
```




```
lossNN = model.evaluate(testX , testY)
```

```
1/1 [==============================] - 0s 17ms/step - loss: 0.1660 - root_mean_squar
```

```
print('loss output for test data is: %f' % lossNN[0])
```

```
loss output for test data is: 0.165995
```

```
predictY = model.predict(testX)
```

```
time = np.arange(predictY.shape[0])

plt.plot(time,predictY,testY)
plt.xlabel('time')
plt.ylabel('amplitude')

plt.show()
```

Creating a copy...


```
pip install pyswarms
```

```
Collecting pyswarms
  Downloading pyswarms-1.3.0-py2.py3-none-any.whl (104 kB)
     |████████████████████████████████| 104 kB 14.7 MB/s
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: attrs in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pyyaml in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: matplotlib>=1.3.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/loca
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from c
Installing collected packages: pyswarms
Successfully installed pyswarms-1.3.0
```

Creating a copy...

✓ 0s completed at 6:02 PM