Time Series Forecasting

Considering first 3 day sales, predicting next day sale, timeseries_data = [110, 125, 133, 146, 158, 172, 187, 196, 210] ex:- 110,125,133 predicting of next day scale for this data will be 146

```
# Lstm example
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
```

```
# Preparing dependent and independent data
def prepare_data(timeseries_data, n_features):
  X, y =[],[]
  for i in range(len(timeseries_data)):
    end_ix = i + n_features
    if end_ix > len(timeseries_data)-1:
      break
    seq_x, seq_y = timeseries_data[i:end_ix], timeseries_data[end_ix]
    X.append(seq_x)
    y.append(seq_y)
  return np.array(X), np.array(y)
```

```
# define input sequence
timeseries_data = [110, 125, 133, 146, 158, 172, 187, 196, 210]
# number of time steps
n_steps = 3
# spliting into samples
X, y = prepare_data(timeseries_data, n_steps)
```

```
print(X)
```

```
[[110 125 133]
 [125 133 146]
 [133 146 158]
 [146 158 172]
 [158 172 187]
 [172 187 196]]
```

```
print(y)
```

```
[146 158 172 187 196 210]
```

```
X.shape
```

```
(6, 3)
```

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

```
# Training model
model = Sequential()
model.add(LSTM(50, activation='relu', return_sequences=True, input_shape=(n_steps, n_features)))
model.add(LSTM(50, activation='relu'))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
# fit model
model.fit(X, y, epochs=300, verbose=1)
```

```
Epoch 200/300
1/1 [==============================] - 0s 3ms/step - loss: 0.4017
Epoch 201/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3734
Epoch 202/300
```

```
1/1 [==============================] - 0s 2ms/step - loss: 0.3834
Epoch 203/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3588
Epoch 204/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3444
Epoch 205/300
1/1 [==============================] - 0s 1ms/step - loss: 0.3542
Epoch 206/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3437
Epoch 207/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3304
Epoch 208/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3233
Epoch 209/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3284
Epoch 210/300
1/1 [==============================] - 0s 996us/step - loss: 0.3363
Epoch 211/300
1/1 [==============================] - 0s 1ms/step - loss: 0.3331
Epoch 212/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3388
Epoch 213/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3360
Epoch 214/300
1/1 [==============================] - 0s 2ms/step - loss: 0.4051
Epoch 215/300
1/1 [==============================] - 0s 7ms/step - loss: 0.4258
Epoch 216/300
1/1 [==============================] - 0s 2ms/step - loss: 0.6551
Epoch 217/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3082
Epoch 218/300
1/1 [==============================] - 0s 2ms/step - loss: 0.3316
Epoch 219/300
1/1 [==============================] - 0s 1ms/step - loss: 0.6958
Epoch 220/300
1/1 [==============================] - 0s 1ms/step - loss: 0.3053
Epoch 221/300
1/1 [==============================] - 0s 1ms/step - loss: 0.3825
Epoch 222/300
1/1 [==============================] - 0s 2ms/step - loss: 0.7552
Epoch 223/300
1/1 [==============================] - 0s 1ms/step - loss: 0.2863
```

```
Epoch 224/300
1/1 [==============================] - 0s 1ms/step - loss: 1.2387
Epoch 225/300
1/1 [==============================] - 0s 2ms/step - loss: 1.5031
Epoch 226/300
1/1 [==============================] - 0s 1ms/step - loss: 2.1327
Epoch 227/300
1/1 [==============================] - 0s 2ms/step - loss: 0.5180
Epoch 228/300
1/1 [==============================] - 0s 6ms/step - loss: 3.0073
Epoch 229/300
```

```
# prediction for next 10 days
x_input = np.array([187, 196, 210])
temp_input=list(x_input)
lst_output=[]
i=0
while(i<10):
    if(len(temp_input)>3):
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input = x_input.reshape((1, n_steps, n_features))
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.append(yhat[0][0])
        temp_input=temp_input[1:]
        lst_output.append(yhat[0][0])  # storing output values
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps, n_features))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.append(yhat[0][0])
        lst_output.append(yhat[0][0])
        i=i+1


print(lst_output)
```

```
[222.3761]
1 day input [196.         210.         222.37609863]
1 day output [[232.61954]]
2 day input [210.         222.37609863 232.61953735]
2 day output [[246.15967]]
3 day input [222.3761  232.61954 246.15967]
3 day output [[258.30603]]
4 day input [232.61954 246.15967 258.30603]
4 day output [[270.1535]]
5 day input [246.15967 258.30603 270.1535 ]
5 day output [[283.90515]]
6 day input [258.30603 270.1535  283.90515]
6 day output [[296.85037]]
7 day input [270.1535  283.90515 296.85037]
7 day output [[310.27136]]
8 day input [283.90515 296.85037 310.27136]
8 day output [[324.76767]]
9 day input [296.85037 310.27136 324.76767]
9 day output [[339.03946]]
[222.3761, 232.61954, 246.15967, 258.30603, 270.1535, 283.90515, 296.85037, 310.27136, 324.76767, 339.03946]
```

```
# predicted output
lst_output
```

```
[222.3761,
 232.61954,
 246.15967,
 258.30603,
 270.1535,
 283.90515,
 296.85037,
 310.27136,
 324.76767,
 339.03946]
```