VIX FORECASTING USING LONG-SHORT TERM MEMORY

- *Prithvi Vadlamani*
CWID: 10476457

VIX Index:

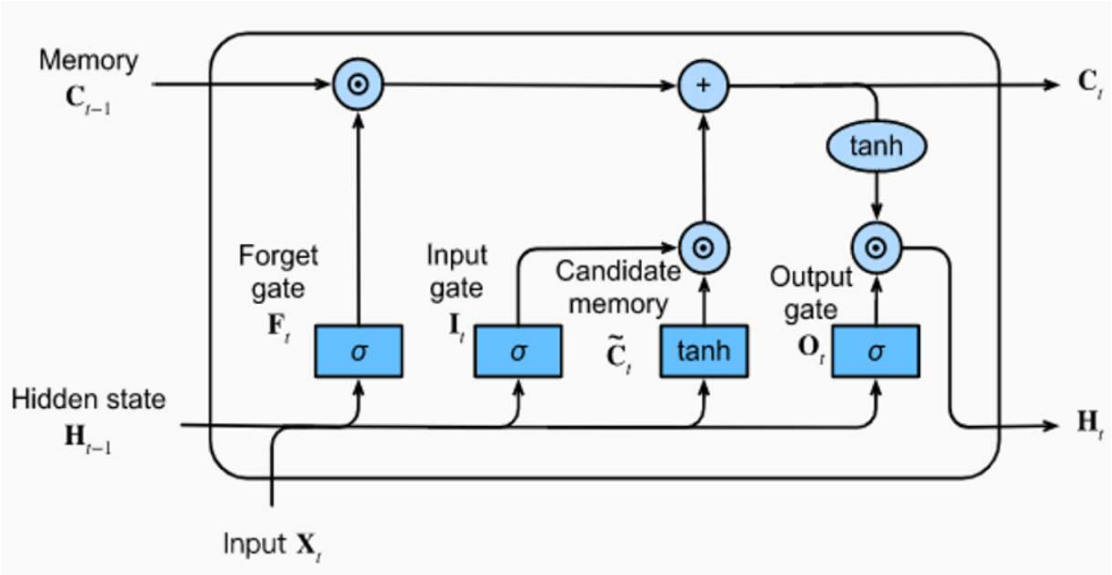
VIX is the ticker symbol and the popular name for the Chicago Board Options Exchange's CBOE Volatility Index, a popular measure of the stock market's expectation of the volatility based on S&P 500 Index Options (*source: Wikipedia*).

Main Idea:

The VIX Index data from the year 2000 to the present is accessed and downloaded using the Bloomberg Terminal. The data is then input into a python IDE where the prediction is performed.

What is LSTM?

LSTM stands for Long Short-Term Memory Networks primarily used in the field of Deep Learning. It is a variety of RNNs (Recurrent Neural Networks). The primary use of LSTM is in sequence prediction problems. The LSTM can do this because of a forget gate present in the model architecture which helps the model capable of learning Long-Term Dependencies. The forget gate is similar to an on/off switch which helps in either forgetting or using the long-term memory in the model.



Packages used:

The main machine learning package used is Keras and the modules imported are LSTM and Sequential along with some supporting modules such as Dense, Dropout, and Activation. To access and read the CSV file in python, the NumPy package is used. The visualization package matplotlib is also used to visualize the data.

```
import matplotlib.pyplot as plt
import numpy as np
import time
from keras.layers.core import Dense, Activation, Dropout
from keras.layers import LSTM
from keras.models import Sequential
```

Process:

First, the NumPy array is converted to a matrix for easier access to the Keras layers used later. Each data point in the array is normalized and the mean of the data is subtracted from each data point.

```
def convertSeriesToMatrix(vectorSeries, sequence_length):
    matrix=[]
    for i in range(len(vectorSeries)-sequence_length+1):
        matrix.append(vectorSeries[i:i+sequence_length])
    return matrix
```

The data is train-test split in a 90/10 ratio. Now the data is input into the hidden layers using the tanh activation function. Another hidden input layer is used so that the model understands the intricacies well. This data is then output using a linear activation function. A dropout ratio of 0.2 is used to avoid the problem of overfitting. An optimizer (rmsprop) is used to optimize/improve the accuracy of the model.

```
train_row = int(round(0.9 * matrix_vix.shape[0]))
train_set = matrix_vix[:train_row, :]
```

```
from keras.backend import tanh
# build the model
model = Sequential()
# layer 1: LSTM
model.add(LSTM(units=1, activation = 'tanh', return_sequences=True))
model.add(Dropout(0.4))
# layer 2: LSTM
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.4))
# layer 3: dense
# linear activation: a(x) = x
model.add(Dense(units=1, activation='linear'))
# compile the model
model.compile(loss="mse", optimizer="rmsprop")
```

The model is run with a batch size of 512, and the number of epochs the model is run is 50.

The test data MSE achieved is 13.76285.

```
test_mse = model.evaluate(X_test, y_test, verbose=1)
print('\nThe mean squared error (MSE) on the test data set is %.5f over %d test samples.' % (test_mse, len(y_test)))
```

```
18/18 [==============================] - 0s 3ms/step - loss: 13.7629

The mean squared error (MSE) on the test data set is 13.76285 over 576 test samples.
```

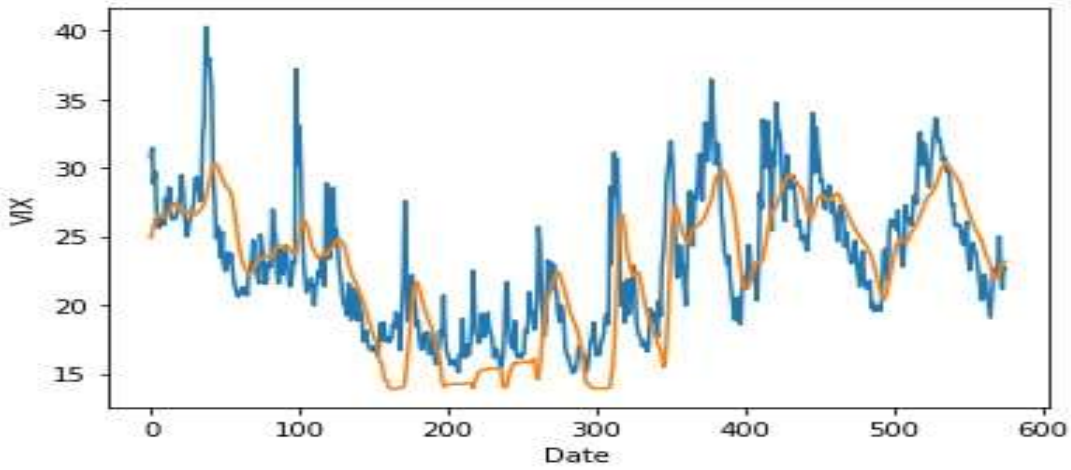


Fig: Graph showing a comparison of test data and predicted data

In the above figure, the blue line indicates the test data, and the orange/yellow line indicates the predicted data.