02 - RNN-LSTM Model Estimation

January 31, 2020

[1]:
```
# If you run this it won't necessarily give you the best RNN model.
# I've saved the best model that I estimated and I use it in the next Jupyter␣
↪Notebook.
# This just an example of the steps I took for estimating the model.
stock = 'Google'
timesteps = 6 # This is the number of lags
```

0.1 Loading Data

[2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import pandas as pd
```

[3]:
```
df = pd.read_csv(f'{stock}.csv')
df.tail()
```

[3]:
```
            Date         Open         High          Low        Close  \
3871  2020-01-06  1350.000000  1396.500000  1350.000000  1394.209961
3872  2020-01-07  1397.939941  1402.989990  1390.380005  1393.339966
3873  2020-01-08  1392.079956  1411.579956  1390.839966  1404.319946
3874  2020-01-09  1420.569946  1427.329956  1410.270020  1419.829956
3875  2020-01-10  1427.560059  1434.928955  1418.349976  1429.729980

        Adj Close   Volume
3871  1394.209961  1732300
3872  1393.339966  1502700
3873  1404.319946  1528000
3874  1419.829956  1500900
3875  1429.729980  1820700
```

[4]:
```
data = pd.DataFrame()
data[f'{stock}'] = df['Adj Close'].copy()
```

[5]:
```
data[f'D{stock}'] = data[f'{stock}'].diff()
data['Variance'] = (data[f'D{stock}']-data[f'D{stock}'].mean())**2
```

[6]:
```
data.dropna(axis=0, inplace=True)
data.tail()
```

[6]:

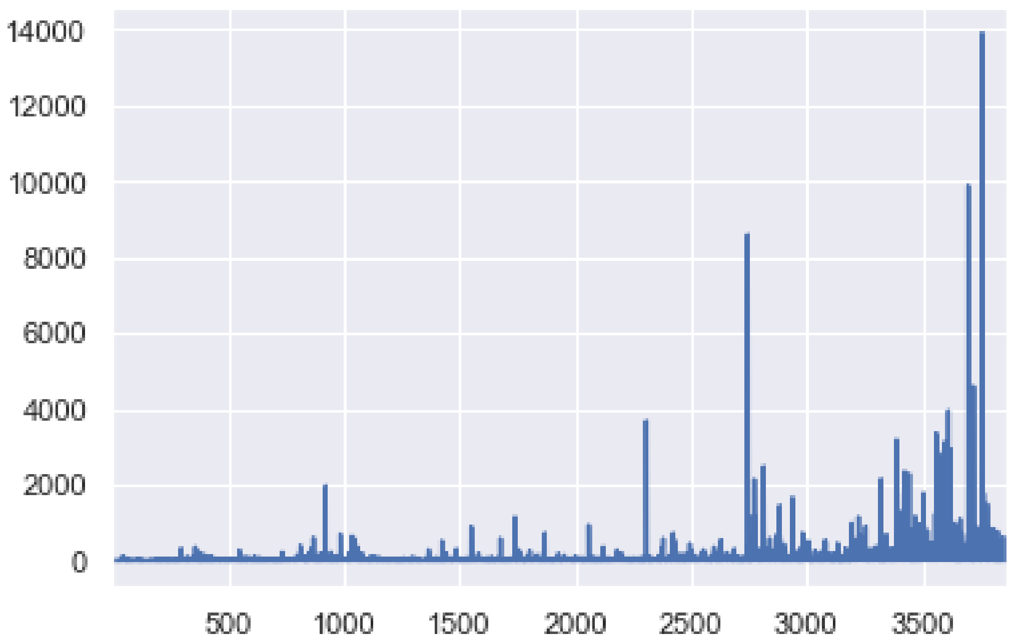
	Google	DGoogle	Variance
3871	1394.209961	33.549927	1101.832552
3872	1393.339966	-0.869995	1.503220
3873	1404.319946	10.979980	112.867595
3874	1419.829956	15.510010	229.642085
3875	1429.729980	9.900024	91.087176

0.2 Splitting the data in training and test sample

[7]:
```
data_training = pd.DataFrame()
data_test = pd.DataFrame()
data_training['Variance'] = data['Variance'][:-timesteps-3].copy()
data_test['Variance'] = data['Variance'][-timesteps-3:].copy()
```

[8]:
```
data_training['Variance'].plot()
```

[8]: <matplotlib.axes._subplots.AxesSubplot at 0x2b9ef3e0908>



0.3 Training Data preprocessing

[9]:
```
import numpy as np

x_train = []
y_train = []

for i in range(timesteps, data_training.shape[0]):
    x_train.append(data_training['Variance'].iloc[i-timesteps:i].values.
 ↪tolist())
    y_train.append((data_training['Variance'].iloc[i]))

x_train, y_train = np.array(x_train), np.array(y_train)

x_train.shape, y_train.shape
```

[9]:
```
((3860, 6), (3860,))
```

0.4 Scaling the data

[10]:
```
# I am using logarithmic scale.
from sklearn.preprocessing import FunctionTransformer
scaler = FunctionTransformer(np.log1p, validate=True)
x_train = scaler.fit_transform(x_train)
x_train = x_train[..., np.newaxis]
x_train.shape
```

[10]:
```
(3860, 6, 1)
```

0.5 Test data preprocessing

[11]:
```
x_test = []
y_test = []

for i in range(timesteps, data_test.shape[0]):
    x_test.append(data_test['Variance'].iloc[i-timesteps:i].values.tolist())
    y_test.append((data_test['Variance'].iloc[i]))

x_test, y_test = np.array(x_test), np.array(y_test)

x_test.shape, y_test.shape
```

[11]:
```
((3, 6), (3,))
```

[12]:
```
x_test = scaler.transform(x_test)
x_test = x_test[..., np.newaxis]
x_test.shape
```

[12]: (3, 6, 1)

0.6 Tensorflow Model

[13]:
```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, GRU, Dropout,␣
↪BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint
```

[14]:
```
model = Sequential()

model.add(LSTM(units = 60, activation = 'relu', return_sequences = True,␣
↪input_shape = (timesteps, 1)))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(LSTM(units = 60, activation = 'relu', return_sequences = True))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(LSTM(units = 60, activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())

model.add(Dense(units = 1, activation = 'relu'))
```

[15]:
```
opt = tf.keras.optimizers.Adam(lr=0.001, decay=1e-6)
```

[16]:
```
model.compile(optimizer=opt, loss = 'mean_absolute_error')
```

[17]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 6, 60)             14880
_________________________________________________________________
dropout (Dropout)            (None, 6, 60)             0
_________________________________________________________________
batch_normalization (BatchNo (None, 6, 60)             240
_________________________________________________________________
lstm_1 (LSTM)                (None, 6, 60)             29040
_________________________________________________________________
dropout_1 (Dropout)          (None, 6, 60)             0
_________________________________________________________________
```

```
batch_normalization_1 (Batch (None, 6, 60)            240
_________________________________________________________________
lstm_2 (LSTM)                (None, 60)               29040
_________________________________________________________________
dropout_2 (Dropout)          (None, 60)               0
_________________________________________________________________
batch_normalization_2 (Batch (None, 60)               240
_________________________________________________________________
dense (Dense)                (None, 1)                61
=================================================================
Total params: 73,741
Trainable params: 73,381
Non-trainable params: 360
_________________________________________________________________
```

[18]:
```
model.fit(x_train, y_train, epochs=50, batch_size=32, validation_split=0.0)
```

```
Train on 3860 samples
Epoch 1/50
3860/3860 [==============================] - 13s 3ms/sample - loss: 79.5320
Epoch 2/50
3860/3860 [==============================] - 6s 1ms/sample - loss: 78.2132
Epoch 3/50
3860/3860 [==============================] - 6s 2ms/sample - loss: 77.4070
Epoch 4/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 77.0473
Epoch 5/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.5964
Epoch 6/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.6291
Epoch 7/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.5122
Epoch 8/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 76.4591
Epoch 9/50
3860/3860 [==============================] - 6s 1ms/sample - loss: 76.3833
Epoch 10/50
3860/3860 [==============================] - 6s 2ms/sample - loss: 76.3837
Epoch 11/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.4064
Epoch 12/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.2600
Epoch 13/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 76.1896
Epoch 14/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.1849
Epoch 15/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 76.2240
```

```
Epoch 16/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.1656
Epoch 17/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.0733
Epoch 18/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.9774
Epoch 19/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.3141
Epoch 20/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.0457
Epoch 21/50
3860/3860 [==============================] - 6s 2ms/sample - loss: 76.0064
Epoch 22/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.0599
Epoch 23/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.0501
Epoch 24/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.9831
Epoch 25/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.3011
Epoch 26/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.0086
Epoch 27/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.0398
Epoch 28/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.7013
Epoch 29/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.9483
Epoch 30/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.8693
Epoch 31/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 76.1916
Epoch 32/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.1003
Epoch 33/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.7137
Epoch 34/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.8636
Epoch 35/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.9689
Epoch 36/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.9316
Epoch 37/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 75.4991
Epoch 38/50
3860/3860 [==============================] - 5s 1ms/sample - loss: 76.0055
Epoch 39/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.7915
```

```
Epoch 40/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.7117
Epoch 41/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.9329
Epoch 42/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.6426
Epoch 43/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.9856
Epoch 44/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.8019
Epoch 45/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.5028
Epoch 46/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.4809
Epoch 47/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.6392
Epoch 48/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.8796
Epoch 49/50
3860/3860 [==============================] - 4s 1ms/sample - loss: 75.6346
Epoch 50/50
3860/3860 [==============================] - 6s 1ms/sample - loss: 75.3645
```

[18]:
```
<tensorflow.python.keras.callbacks.History at 0x2ba6e2bc508>
```

0.7 Evaluating RNN-LSTM Model

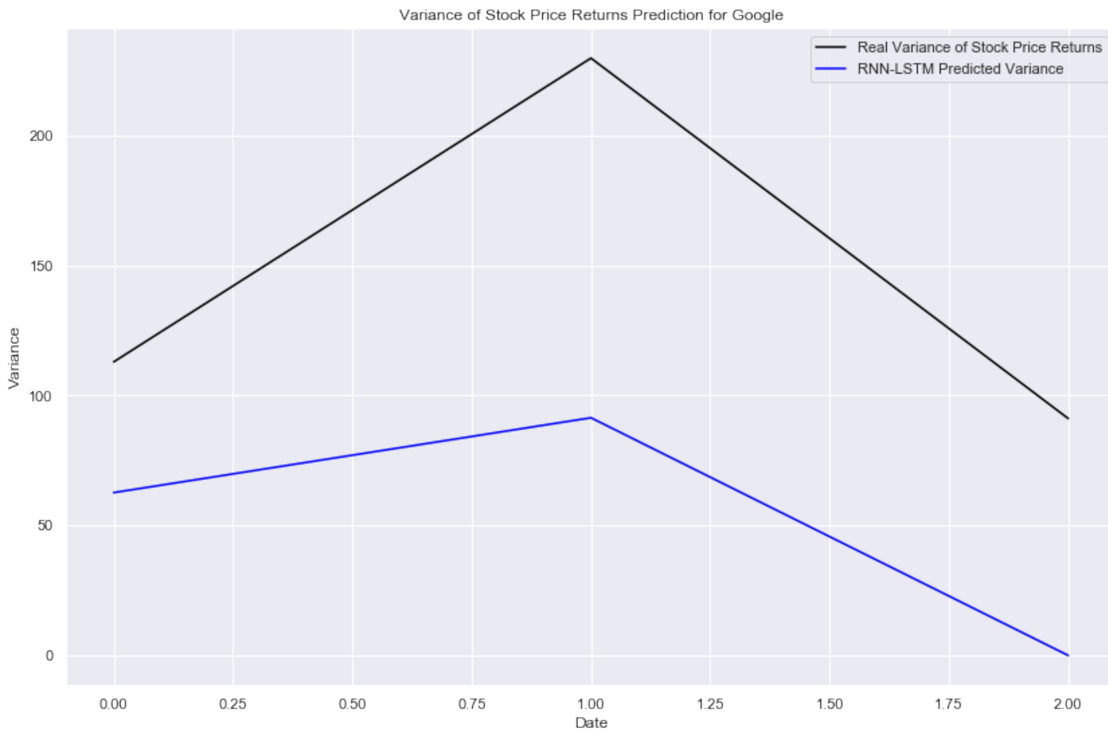
[19]:
```
y_pred = model.predict(x_test)
```

[20]:
```
from sklearn.metrics import mean_absolute_error
mae = round(mean_absolute_error(y_test, y_pred), 3)
print(mae)
```

```
93.201
```

0.8 Visualization

[21]:
```
plt.figure(figsize=(14,9))
plt.plot(y_test, color = 'black', label = 'Real Variance of Stock Price␣
↪Returns')
plt.plot(y_pred, color = 'blue', label = 'RNN-LSTM Predicted Variance')
plt.title(f'Variance of Stock Price Returns Prediction for {stock}')
plt.xlabel('Date')
plt.ylabel('Variance')
plt.legend()
plt.show()
```


Variance of Stock Price Returns Prediction for Google
Real Variance of Stock Price Returns
RNN-LSTM Predicted Variance
Variance
Date
0
50
100
150
200
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00