Predicting Stock Prices using Neural Networks (W/ python)

Imports

In [100…]
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pandas_datareader as web
import datetime as dt

from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM
```

Load Data

In [101…]
```
company = 'FB'

start = dt.datetime(2012,1,1)
end = dt.datetime(2018,12,31)

data = web.DataReader(company,'yahoo', start, end)
data
```

Out[101…]

Date	High	Low	Open	Close	Volume	Adj Close
2012-05-18	45.000000	38.000000	42.049999	38.230000	573576400	38.230000
2012-05-21	36.660000	33.000000	36.529999	34.029999	168192700	34.029999
2012-05-22	33.590000	30.940001	32.610001	31.000000	101786600	31.000000
2012-05-23	32.500000	31.360001	31.370001	32.000000	73600000	32.000000
2012-05-24	33.209999	31.770000	32.950001	33.029999	50237200	33.029999
…	…	…	…	…	…	…
2018-12-24	129.740005	123.019997	123.099998	124.059998	22066000	124.059998
2018-12-26	134.240005	125.889999	126.000000	134.179993	39723400	134.179993
2018-12-27	134.990005	129.669998	132.440002	134.520004	31202500	134.520004
2018-12-28	135.919998	132.199997	135.339996	133.199997	22627600	133.199997
2018-12-31	134.639999	129.949997	134.449997	131.089996	24625300	131.089996

1665 rows × 6 columns

Prepare the data

In [102…]
```
scaler = MinMaxScaler ( feature_range=(0,1))
scaled_data = scaler.fit_transform(data['Close'].values.reshape(-1,1))

prediction_days = 60
```

```
x_train =[]
y_train =[]

for x in range(prediction_days, len(scaled_data)):
    x_train.append(scaled_data[x-prediction_days:x,0])
    y_train.append(scaled_data[x,0])

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

x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

Build Model

In [103…

```
#O modelo que vamos construir é um modelo sequencial, isto significa de grosso m
model = Sequential()

#LSTM ->
model.add(LSTM(units=60, return_sequences=True, input_shape=(x_train.shape[1],1)
model.add(Dropout(0.2))
model.add(LSTM(units=60, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=60))
model.add(Dropout(0.2))
model.add(Dense(units=1))

model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, epochs=25, batch_size=32)
```

```
Epoch 1/25
51/51 [==============================] - 17s 120ms/step - loss: 0.0538
Epoch 2/25
51/51 [==============================] - 6s 114ms/step - loss: 0.0029
Epoch 3/25
51/51 [==============================] - 6s 118ms/step - loss: 0.0026
Epoch 4/25
51/51 [==============================] - 6s 111ms/step - loss: 0.0028
Epoch 5/25
51/51 [==============================] - 6s 111ms/step - loss: 0.0027
Epoch 6/25
51/51 [==============================] - 5s 93ms/step - loss: 0.0019
Epoch 7/25
51/51 [==============================] - 5s 93ms/step - loss: 0.0028
Epoch 8/25
51/51 [==============================] - 7s 145ms/step - loss: 0.0021
Epoch 9/25
51/51 [==============================] - 6s 121ms/step - loss: 0.0023
Epoch 10/25
51/51 [==============================] - 6s 122ms/step - loss: 0.0023
Epoch 11/25
51/51 [==============================] - 5s 103ms/step - loss: 0.0021
Epoch 12/25
51/51 [==============================] - 6s 112ms/step - loss: 0.0022
Epoch 13/25
51/51 [==============================] - 6s 127ms/step - loss: 0.0019
Epoch 14/25
51/51 [==============================] - 6s 120ms/step - loss: 0.0018
Epoch 15/25
51/51 [==============================] - 7s 128ms/step - loss: 0.0024
Epoch 16/25
51/51 [==============================] - 5s 94ms/step - loss: 0.0017
Epoch 17/25
51/51 [==============================] - 4s 81ms/step - loss: 0.0017
```

```
Epoch 18/25
51/51 [==============================] - 4s 84ms/step - loss: 0.0018
Epoch 19/25
51/51 [==============================] - 4s 86ms/step - loss: 0.0026
Epoch 20/25
51/51 [==============================] - 4s 82ms/step - loss: 0.0017
Epoch 21/25
51/51 [==============================] - 5s 91ms/step - loss: 0.0015
Epoch 22/25
51/51 [==============================] - 5s 95ms/step - loss: 0.0016
Epoch 23/25
51/51 [==============================] - 4s 84ms/step - loss: 0.0016
Epoch 24/25
51/51 [==============================] - 4s 82ms/step - loss: 0.0013
Epoch 25/25
51/51 [==============================] - 4s 75ms/step - loss: 0.0015
```

Out[103… `<tensorflow.python.keras.callbacks.History at 0x7f881d4d8970>`

Testing Model Accuracy on Existing data

In [104…

```
test_start = dt.datetime(2019,1,1)
test_end = dt.datetime.now()

test_data = web.DataReader(company, 'yahoo', test_start, test_end)
actual_prices = test_data['Close'].values

total_dataset = pd.concat((data['Close'], test_data['Close']), axis=0)

model_inputs = total_dataset[len(total_dataset)- len(test_data)-prediction_days:
model_inputs = model_inputs.reshape(-1,1)
model_inputs=scaler.transform(model_inputs)
```

Make predictions

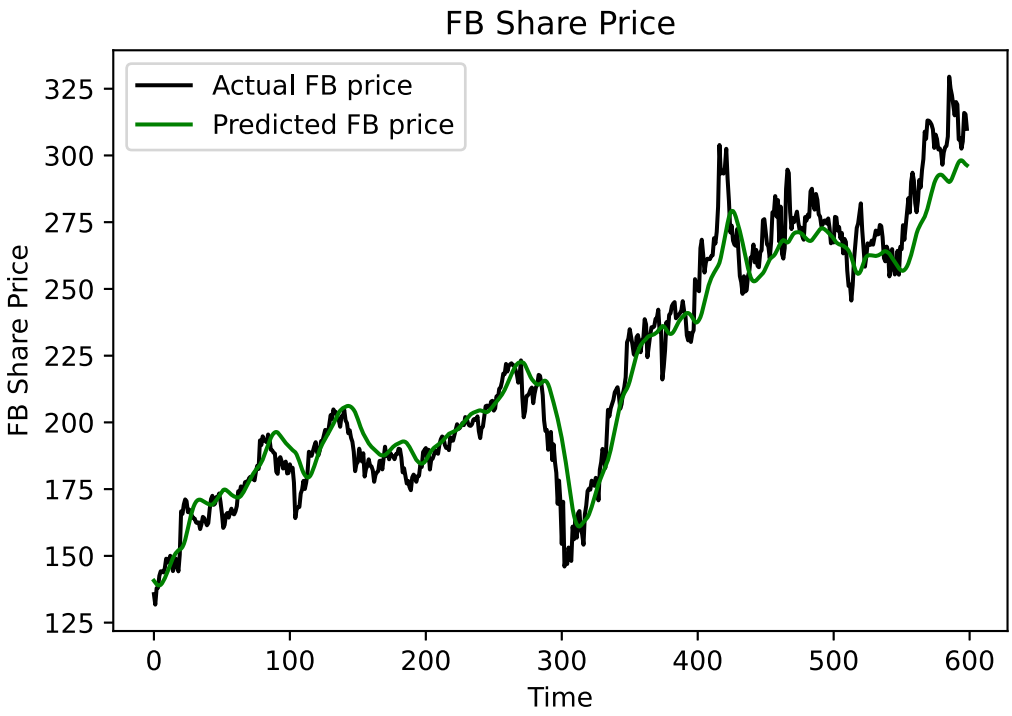
In [105…

```
x_test=[]

for x in range(prediction_days, len(model_inputs)):
    x_test.append(model_inputs[x-prediction_days:x,0])

x_test = np.array(x_test)
# ----------------------------
x_test = np.reshape(x_test,(x_test.shape[0], x_test.shape[1],1))
# ----------------------------

predicted_prices = model.predict(x_test)
predicted_prices = scaler.inverse_transform(predicted_prices)
```

Plot Test Predictions

In [106…

```
plt.plot(actual_prices, color='black', label=f'Actual {company} price')
plt.plot(predicted_prices, color='green', label=f'Predicted {company} price')
plt.title(f'{company} Share Price')
plt.xlabel('Time')
plt.ylabel(f'{company} Share Price')
plt.legend()
plt.show()
```



Predict Next Day

In [107…
```
real_data = [model_inputs[len(model_inputs)+1-prediction_days:len(model_inputs+1

real_data = np.array(real_data)
real_data = np.reshape(real_data, (real_data.shape[0],real_data.shape[1],1))

prediction = model.predict(real_data)
prediction = scaler.inverse_transform(prediction)
print(f'Prediction {prediction}')
```

```
WARNING:tensorflow:Model was constructed with shape (None, 60, 1) for input Kera
sTensor(type_spec=TensorSpec(shape=(None, 60, 1), dtype=tf.float32, name='lstm_3
3_input'), name='lstm_33_input', description="created by layer 'lstm_33_inpu
t'"), but it was called on an input with incompatible shape (None, 59, 1).
Prediction [[295.9717]]
```

In [108…
```
print(dt.datetime.now())
```

```
2021-05-18 21:04:33.478578
```