Predicting Stock Prices using Neural Networks (W/ python)

Imports

In [12]:

```
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 [13]:

```
company = 'FB'

start = dt.datetime(2012,1,1)
end = dt.datetime(2020,1,1)

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

Out[13]:

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
...	...	...	...	...	...	...
2019-12-24	206.789993	205.000000	206.300003	205.119995	6046300	205.119995
2019-12-26	207.820007	205.309998	205.570007	207.789993	9350700	207.789993
2019-12-27	208.929993	206.589996	208.669998	208.100006	10284200	208.100006
2019-12-30	207.899994	203.899994	207.860001	204.410004	10524300	204.410004
2019-12-31	205.559998	203.600006	204.000000	205.250000	8953500	205.250000

1917 rows × 6 columns

Prepare the data

In [14]:

```
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 [15]:

```
model = Sequential()

model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
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
59/59 [==============================] - 11s 70ms/step - loss: 0.0735
Epoch 2/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0047
Epoch 3/25
59/59 [==============================] - 3s 59ms/step - loss: 0.0036
Epoch 4/25
59/59 [==============================] - 3s 57ms/step - loss: 0.0043
Epoch 5/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0034
Epoch 6/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0035
Epoch 7/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0033
Epoch 8/25
59/59 [==============================] - 3s 59ms/step - loss: 0.0028
Epoch 9/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0057
Epoch 10/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0030
Epoch 11/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0027
Epoch 12/25
59/59 [==============================] - 3s 59ms/step - loss: 0.0030
Epoch 13/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0023
Epoch 14/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0026
Epoch 15/25
59/59 [==============================] - 3s 59ms/step - loss: 0.0045
Epoch 16/25
59/59 [==============================] - 4s 59ms/step - loss: 0.0026
Epoch 17/25
59/59 [==============================] - 3s 58ms/step - loss: 0.0025
Epoch 18/25
59/59 [==============================] - 4s 68ms/step - loss: 0.0027
```

```
Epoch 19/25
59/59 [==============================] - 4s 60ms/step - loss: 0.0021
Epoch 20/25
59/59 [==============================] - 4s 70ms/step - loss: 0.0022
Epoch 21/25
59/59 [==============================] - 4s 66ms/step - loss: 0.0021
Epoch 22/25
59/59 [==============================] - 4s 61ms/step - loss: 0.0024
Epoch 23/25
59/59 [==============================] - 4s 60ms/step - loss: 0.0023
Epoch 24/25
59/59 [==============================] - 4s 64ms/step - loss: 0.0021
Epoch 25/25
59/59 [==============================] - 4s 61ms/step - loss: 0.0019
```

Out[15]:

```
<tensorflow.python.keras.callbacks.History at 0x7fa1caebd100>
```

Testing Model Accuracy on Existing data

In [16]:

```python
test_start = dt.datetime(2020,1,1)
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

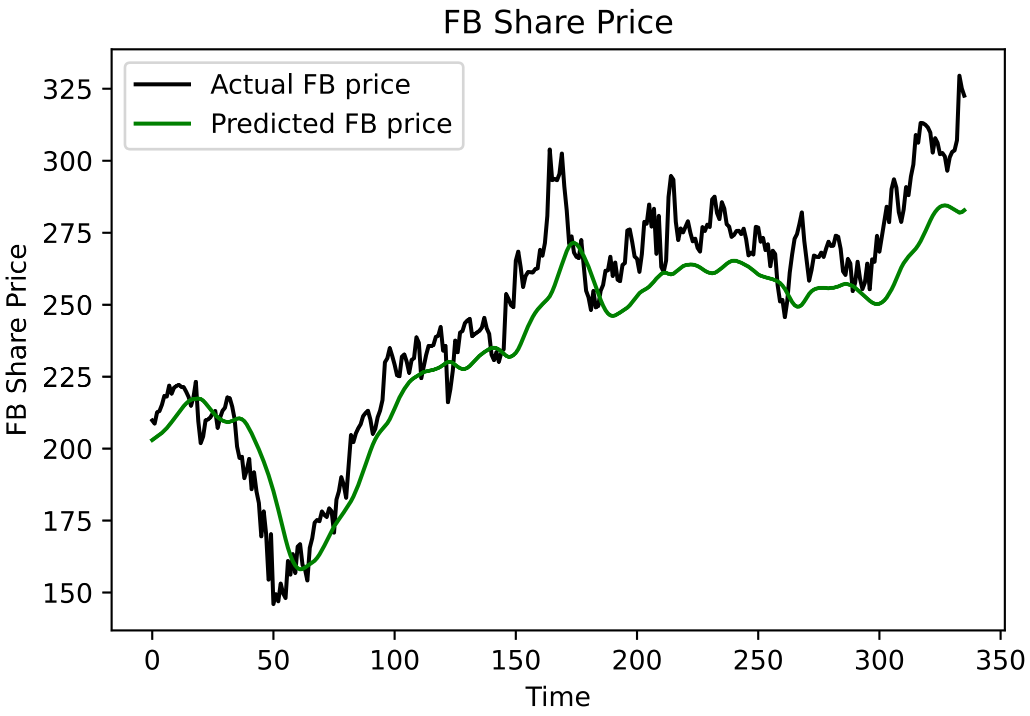
In [17]:

```python
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 [18]:

```python
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 [19]:

```
real_data = [model_inputs[len(model_inputs)+1-prediction_days:len(model_input   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 Ke
sTensor(type_spec=TensorSpec(shape=(None, 60, 1), dtype=tf.float32, name='lstm_3
_input'), name='lstm_3_input', description="created by layer 'lstm_3_input'"), b
ut it was called on an input with incompatible shape (None, 59, 1).
Prediction [[283.80038]]
```

In [20]:

```
print(dt.datetime.now())
```

```
2021-05-03 23:00:36.130131
```