Algorithm Assisted Day Trading

Muhoza U Bizumuremyi
SpringBoard Data Science Career Track

Contents

1. Introduction and Objective

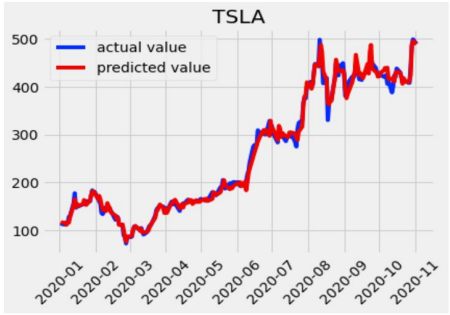
According to the social trading platform Etoro, 80% of all day traders lose money on the stock market.

This has encouraged me to build a function/application that can help day traders to predict a future value of a stock in order to decide whether they should sell or hold that stock.

This function should not be the sole measure of success or failure of a stock but can be used alongside other techniques that traders usually use.

2. How the function works

```
In [3]: forecast_stock('TSLA',584.5)

Out[3]: {'Status': ['HOLD'],
         'Future Price': [633.7675054881902],
         'Number of shares': [1.1938073517440664]}
```

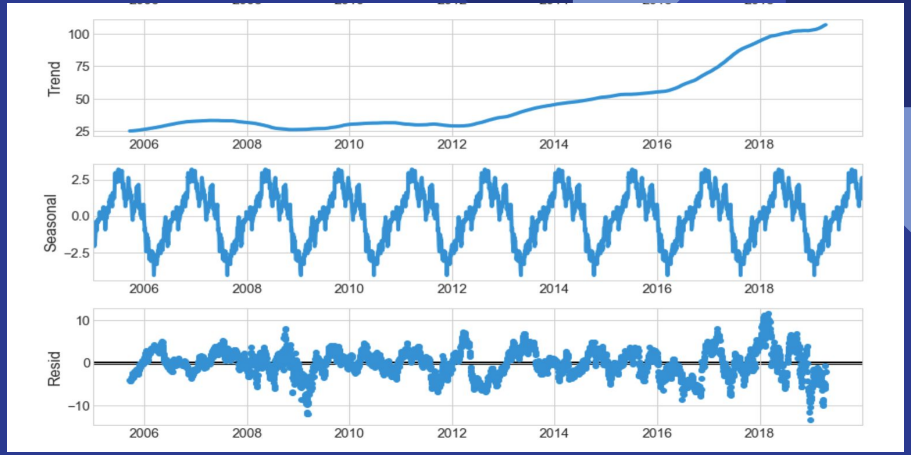


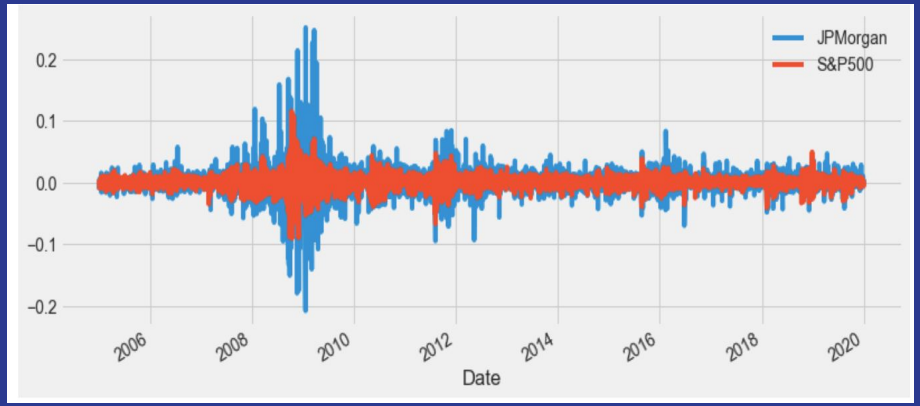
The function requires to input the price of shares a person hold and the ticker of the stock s/he is holding and the function will output the future (the next day) predicted value of those shares and then the function will advise the day trader whether to sell or hold.

If the predicted price is lower than the current price then the function will advise to sell if not the function will advise to hold.

3. Stock Analysis

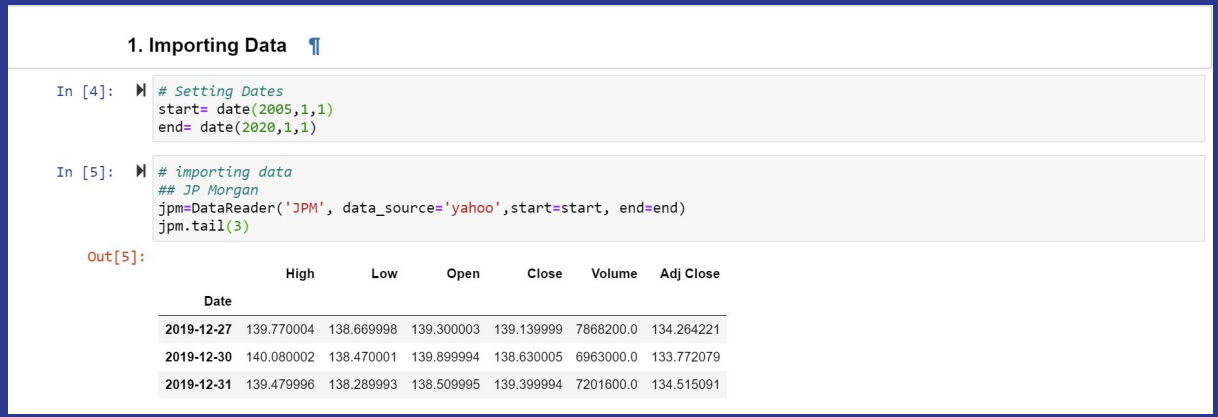
Before jumping into modeling and building the function, we looked at different common ways of analyzing a stock (JP Morgan stock for our case) such as moving averages, trend and seasonality, how a certain stock performs compared with the market and few other methods.





4. Data Source

Stock data that was used in this project can be found on various platforms that offers financial data such Google finance, Bloomberg, Yahoo finance,... but in our case we used Yahoo finance because it does not require an API key to import data.



1. Importing Data ¶

In [4]:
```
# Setting Dates
start= date(2005,1,1)
end= date(2020,1,1)
```

In [5]:
```
# importing data
## JP Morgan
jpm=DataReader('JPM', data_source='yahoo',start=start, end=end)
jpm.tail(3)
```

Out[5]:

Date	High	Low	Open	Close	Volume	Adj Close
2019-12-27	139.770004	138.669998	139.300003	139.139999	7868200.0	134.264221
2019-12-30	140.080002	138.470001	139.899994	138.630005	6963000.0	133.772079
2019-12-31	139.479996	138.289993	138.509995	139.399994	7201600.0	134.515091

5. Pre-Processing, Modeling and Model Performance

For our case we used Long Short Time Memory Network (LSTM), a type of Recurrent Neural Network, in order to model and predict the future shares' price

Using the JPMorgan stock data we can visualize how the model built using LSTM performs.

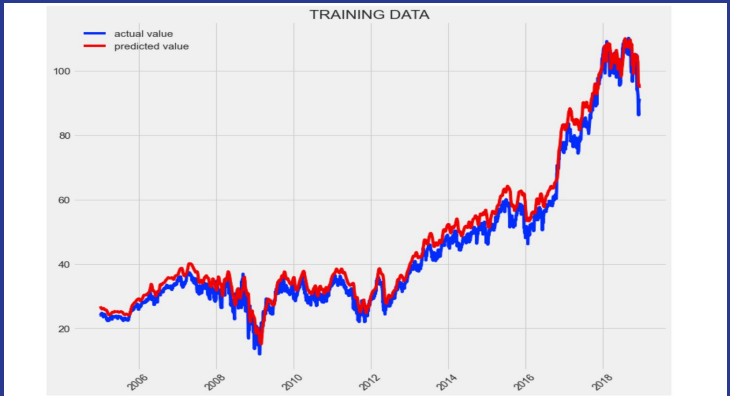
```
In [39]: #this kind of shape
         X_train.shape
Out[39]: (3509, 14, 1)

In [40]: X_test.shape
Out[40]: (238, 14, 1)
```

9. Modeling

```
In [41]: # Calling the model
         model = Sequential()
         model.add(LSTM(50,activation='relu', return_sequences=True,input_shape=(X_train.shape[1], X_train.shape[2])))
         model.add(LSTM(50, activation='relu'))
         model.add(Dense(1))
         model.compile(loss='mean_squared_error', optimizer='adam')
         model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 14, 50)            10400
_________________________________________________________________
lstm_1 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 30,651
Trainable params: 30,651
Non-trainable params: 0
_________________________________________________________________
```

```
In [43]: # fitting/training the data
         model.fit(X_train, Y_train, epochs=200, batch_size=100, validation_data=(X_test, Y_test), verbose=0, shuffle=False)
Out[43]: <tensorflow.python.keras.callbacks.History at 0x1f3527c7910>
```




```
#Lets see how the model has performed
print('Train Mean Absolute Error:', mean_absolute_error(Y_train, train_predict[:,0]))
print('Train Root Mean Squared Error:',np.sqrt(mean_squared_error(Y_train, train_predict[:,0])))
print('Test Mean Absolute Error:', mean_absolute_error(Y_test, test_predict[:,0]))
print('Test Root Mean Squared Error:',np.sqrt(mean_squared_error(Y_test, test_predict[:,0])))
```

```
Train Mean Absolute Error: 1.5739459768595876
Train Root Mean Squared Error: 1.8821028467294718
Test Mean Absolute Error: 2.0317771334608064
Test Root Mean Squared Error: 2.5562835047238l
```

6. Summary

For our function, when a ticker and the price of shares are entered, the function will import data from Yahoo finance using the ticker, then it will use that data to train a model using LSTM networks, then the model will be used to predict one future price (the next day) using the prices of 14 previous days as input.

Once the next day price is predicted, the function will compare it with the current and then advise the day trader to sell or hold the stock.

END