Project definition

Project overview

To better understand stock market behaviour and to improve profitability of investments, hedge funds, investment firms and individual investors have been using financial models to predict future behaviour of stock prices. In recent times, these financial models have become easier to implement due to the abundance of data available about stock prices as well as information available on implementing financial models using python. Financial data relevant to stock prices are readily available through a number of websites using an API's to retrieve the data. Stock price data can be classified as time series data as it is sequential in nature. To make predictions or forecasts using time series data two commonly used approaches exist: classical machine learning models and deep learning models. Classical machine learning models have three main disadvantages namely:

1. They are unable to recognize complex patterns in data
2. Missing data values affect the performance of the model
3. They only work well for short term forecasts and struggle to capture long term trends

Deep learning, however, addresses these disadvantages. A particular deep learning method that is commonly used for time series forecasting and predictions is Long Short-Term Memory (LSTM). This method has become particularly useful in stock price predictions in recent years due to its ability to capture and model the uncertainty of stock prices both in the short and the long term.

Problem statement

This project aims to build a stock prediction model by using stock price data and technical indicators thereof to predict future values of stock prices. For this project an API called *yfinance* is used to extract the stock price data since 2010 to the current date. Each API call extracts the open price, highest price, lowest price, volume of stocks traded and adjusted closing price for a prescribed date period and for a particular stock. The data received from the API is very clean and well structured. In stock trading technical indicators are often used to predict a long term or short term trend. One such technical indicator is the concept of moving averages. A moving average of 5 days, 50 days and 200 days is used in this project to show movement of momentum in stock prices.

More specifically, the opening price, the highest daily trading price, the closing price, the volume of stocks traded as well as the adjusted closing price of a stock is used to predict the future adjusted closing price. The results of this model need to be presented in the form of a functioning web application. The web application should include a page to analyse the stock price data of a selected stock, a training interface page to train a new model with preselected parameters and a page to display the results of an existing pre-trained model. This project is limited to the analysis of three stocks only; Microsoft, S&P500 and Nasdaq. This project will thus explore the implementation of an LSTM model to predict the future prices of stocks using Keras.

Metrics

Since this project aims to predict the numeric value of the adjusted closing price of a stock the metric selected for determining the performance of the model is the mean squares error (MSE). The mean square error is the square of the prediction error, in other words the square of the difference between the predicted values and the actual values.

$$MSE = \frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2$$

Mean square error is used as a metric for measuring a machine learning models' performance for the following reasons (Kampakis):

1. Since the error is squared when calculating MSE, a model using MSE will assign a much larger weight to large error values. (for example an error of 10 is 100 times worse than an error of 1)

With reference to reason 1 mentioned above, MSE is used during cross validation of this LSTM model since it will ensure that the hyperparameters selected for the final model will result in small prediction errors. This is true because the use of MSE penalizes large errors more severely than some other metrics.

Analysis

Data exploration

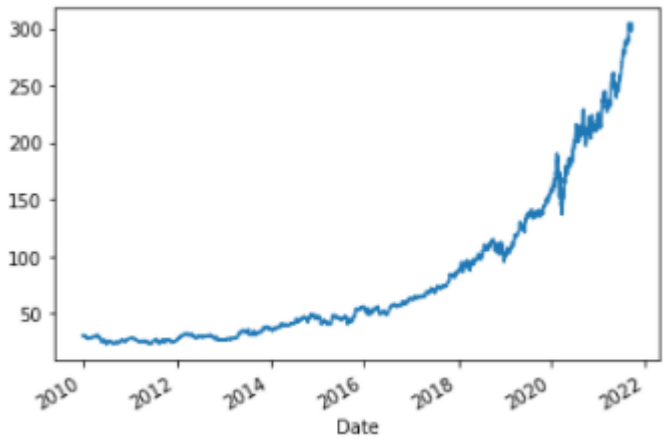
The data is received using an API call to yfinance and returned as shown in the following image:

Date	Open	High	Low	Close	Volume	AdjClose
2009-12-31	30.980000	30.990000	30.480000	30.480000	31929700	23.630186
2010-01-04	30.620001	31.100000	30.590000	30.950001	38409100	23.994564
2010-01-05	30.850000	31.100000	30.639999	30.959999	49749600	24.002319

The data received by the API is clean and already converted into the correct types and thus no further transformations are needed to make the data available to the pre-processing phase.

Data visualization

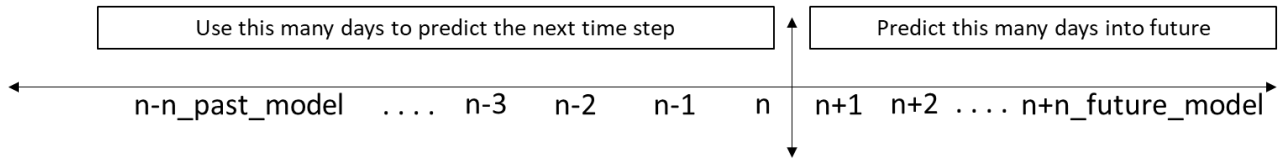
As an example of how the data looks see below the opening price of Microsoft plotted from 2010 to 2021.



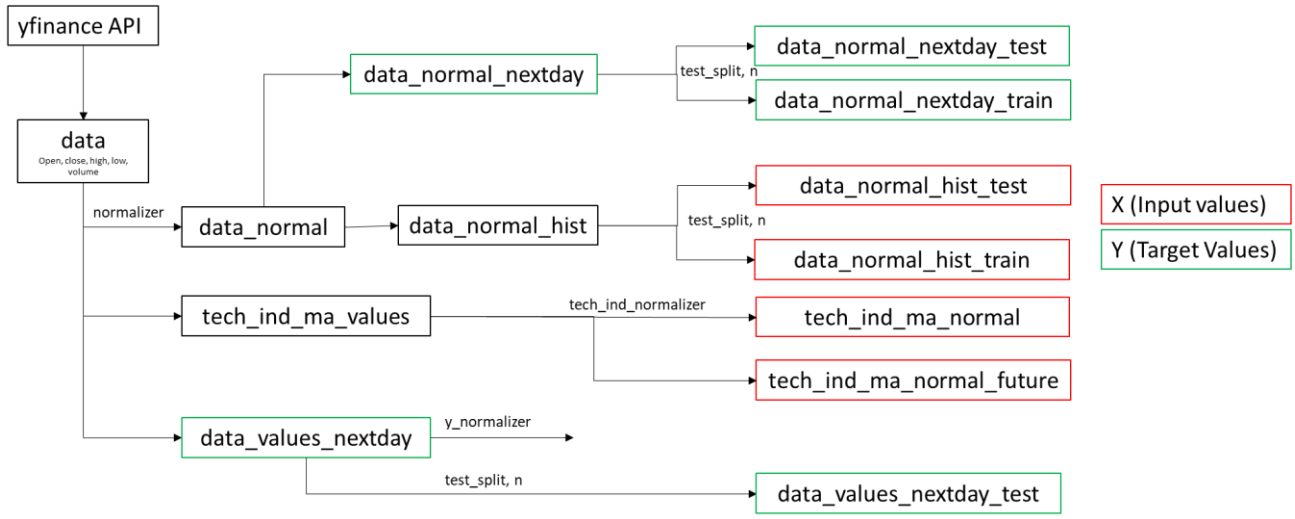
Methodology

Data Pre-processing

The data is received from an API call to *yfinance.* Before this data can be used in a machine learning model it needs to be scaled and since this is time series data it needs to be windowed. The data scaling was done using Sklearn's pre-processing module. The windowing of the data be best described using the below image. As shown in the image, a certain number of days is used to predict the next days stock price. As an example: use the last 50days of stock prices to predict the stock price for tomorrow. This way the prediction for tomorrow is using the behaviour of the stock price from the past 50days to predict a more accurate stock price for 1 day into the future.

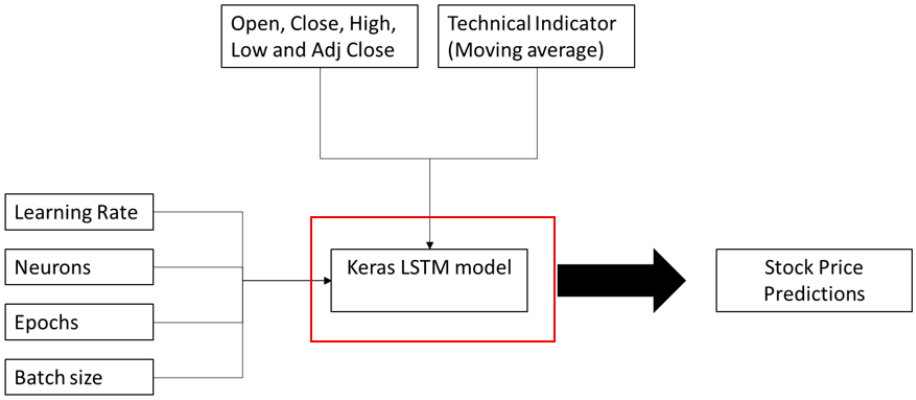


Further pre-processing is done to divide the data into inputs and targets for the machine learning model. In the image below it is shown how the data from the API call gets divided into the different variables to form the features of the machine learning model. All variables indicated in red are values used as inputs and all variables indicated in green are values used as targets for the machine learning model. Additionally, this schematic also indicates how the data is divided into test and train sets and shows how a technical indicator is incorporated into the model (tech_ind_ma_normal and tech_ind_ma_future)



Implementation

The model is created using Keras and generally takes two inputs: the moving average technical indicator and the opening price, closing price, highest price, lowest price, and adjusted closing price of the stock over the period defined. Additionally, the model utilizes 4 different hyper parameters to be considered as these parameters affect the accuracy of predictions.



More details on the Keras LSTM model can be seen in the diagram below. This diagram explains how the model architecture is formed using Keras. Each layer has an indication of an input shape, an output shape, layer type and description of what each layer does. The name in each layer shown in the diagram below can be referenced to the code in the Model_data_prep.py file to see the actual code used to create each of the below mentioned layers using Keras. It is important to note that neurons is a variable that is used during cross validation to find the optimal number of neurons for the LSTM model layer. This is also a hyper parameter as shown in the diagram above. Similarly, n_past is a variable that gets changed within the user interface depending on how many days in the past the user wants to use to predict the future values. Furthermore, time_steps is simply the total number of time steps in each input or output dataset.

To compile the model a mean square error loss function is used to penalize large errors and an *adam* optimizer is used. Complications during the coding process include making sure that the inputs of the one layer match the outputs of the previous layers in array dimensionality. Further complications experienced include the model pre-processing to ensure the correct array dimensionality is achieved during the pre-processing phase.

Refinement

Refinement of the model was done during the implementation of the model. At first a model was created using only the open price, closing price, highest price, lowest price, and adjusted closing price but it was found that prediction accuracy was not sufficient. This then led to the introduction of the moving average technical indicator to be included in the model to improve accuracy.

Results

Model evaluation and validation

To determine the optimal values for the different input parameters to the Keras LSTM model cross validation was used. This process entailed maintaining all other parameters constant while varying one specific parameter. For example: a constant value for the number of neurons and batch size while the number of epochs was varied. For each of these scenarios the mean squared error is calculated to be able to compare the different combinations to one another. The table below summarizes the results of the cross validation and highlights the parameters which resulted in the best mean square error. The following hyper-parameters were found to be result in the best mean square error:

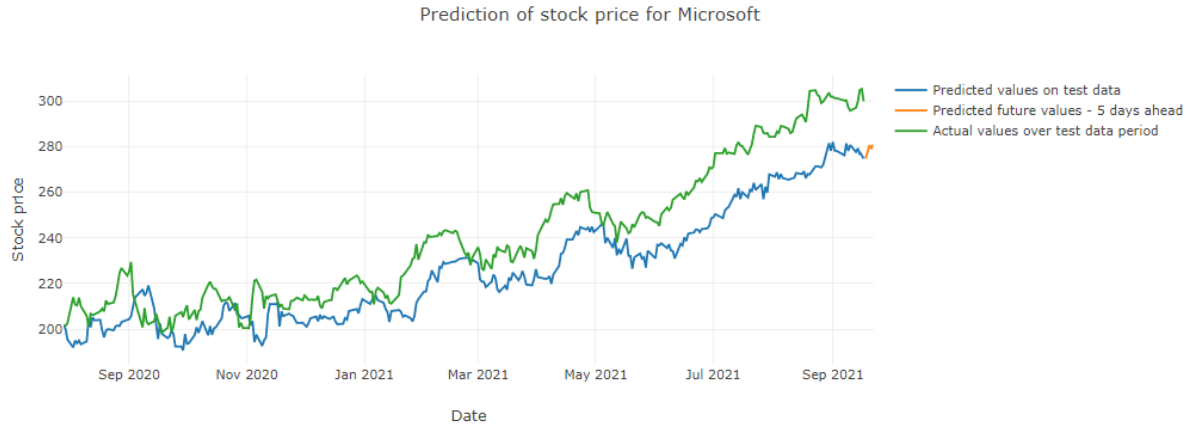
- neurons = 40
- batch_size = 5
- no_epochs = 150
- learning rate = 0.0005 (the learning rate was not varied during cross-validation and should be done in future work)

batch_size	mse_score (average)		no_epochs	mse_score (average)		neuron_num	mse_score (average)
5	214		25	3168		5	8464
10	414		50	2387		10	5352
15	697		75	2271		15	3191
20	1697		100	1037		20	1929
25	1610		125	975		25	1043
30	2855		150	508		30	500
35	2344					35	792
40	2876					40	358
						45	408

The above mentioned hyperparameters are the best parameters to use for the model as proven by cross validation. It should be noted that the batch size is relatively small and this assists during the training phase to reduce the chances of overfitting the data to the training set (additionally drop out layers are also used in the model to prevent over fitting during training). Batch size is known as one of the most important hyper parameters to tune effectively since there is a fine line between achieving a quicker computation time and achieving less overfitting (Shen, 2018). Furthermore, in the data pre-processing phase the data is split into a test and training set. This done to ensure that that model, with its final hyper parameters, can be tested against data it has never seen before. During the testing procedure the model is confirmed to produce results in the same order of magnitude of the true values.

Justification

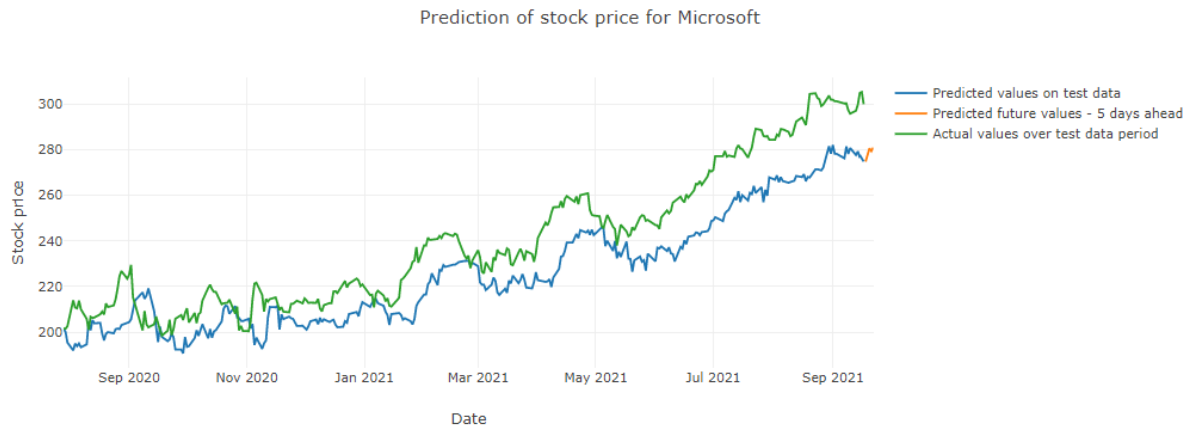
To test the model, the data set was split into a training and testing set using an 80% split. This means that 20% of the data is unseen by the model when testing the model. The testing data set was used to test the model performance on unseen data and the results of the stock Microsoft is shown below. The line indicated in blue are the predicted values and the green line indicated the actual stock price values. The orange line is the stock price predicted 5 days into the future.



Conclusion

Reflection

The web application functions as it should; providing a user with the option to analyse the stock price with various moving averages indicated on the same axis, view the results of an existing model that was pre-trained to provide a visualization of the model performance and then finally a page that allows a user to train their own model with a certain set of their own parameters. The model performance, as measured by the mean square error, is not sufficient to be able to accurately predict the future stock prices with a good level of certainty. On average, the mean square error of the model was +-200%. This means that a true value differs from a predicted value by as much as $14 (square root of 200%). It should however be noted that when the model results are plotted vs. the actual results the predicted values follow the correct trend overall, but it seems like there is a relatively constant offset between actual and predicted values. This is shown in the plot below.



The key steps taken in this project can be outlined in the below table and the corresponding challenges experienced in each step of implementation are shown.

Key step	Interesting / challenges
Retrieve data through API	Not all API's provided online are still active or provide sufficient documentation and thus three different API's were used before finding yfinance to do the job of retrieving the data.
Pre-processing of the data	Because the model makes use of a certain number of values to predict the future value, it proved challenging to make sure that the inputs and the outputs are correctly defined such that for example values 1 -50 are used to predict value number 51.
Creating the LSTM model	Understanding the interaction between the different layers within the model was interesting, specifically the way such a model requires the correct dimensionality in each input and output layer.
Testing the model	Testing the model was a great way to confirm if the model can predict future values with a reasonable efficiency. Here the use of Plotly was interesting as it has a great user interface on the web application to do an analysis of the predicted values.
Refining the model	Running the cross-validation to determine the optimal hyper parameters was challenging as it takes a large amount of time to run through the different parameter values. It was however very interesting to see how each parameter affects the results.
Web application development	Being able to finally understand how a web application sends data from the backend to the front-end using Flask was very interesting. Using html, css and Jinja was very interesting to learn.

The end-to-end solution address the problem statement by providing a web application with the following functionality:

1. An analysis interface for selecting one stock out of three options to analyse. In the analysis interface the user can select to show technical indicators as they wish
2. A training interface provides the user with an opportunity to select the way the model should be trained with the best hyper parameters as determined through cross validation
3. A pre-trained model that shows the model performance with pre-selected hyperparameters and other additional variables that are normally selected using the training interface.

It should however be noted that the model should not be used to invest actual money into the stock market. The accuracy is not sufficient and further work is necessary along with vast amounts of back testing to provide enough confidence for real money to be invested.

Improvement

As mentioned in the reflection section, the model prediction performance is not as good as one would need to invest money in the stock market based on the models' predictions. The following can be done to improve the prediction model:

1. Incorporate more features into the input of the LSTM model. Moving average is one of many technical indicators that investment analysts use to determine trends in the stock market. To improve the performance of the model additional technical indicators can be used as additional features.
2. The web application can be improved by incorporating sentiment analysis on stock news for the three different stocks analysed in this project. Combining sentiment analysis of stocks with stock price predictions will provide a more reliable indicator of when to invest in a particular stock.
3. Incorporate the learning rate into the cross-validation process for future work to determine if another value would yield better results.

Bibliography

Kampakis, S. (. (n.d.). *Performance measures: RMSE and MAE.* Retrieved from The data scientist: https://thedatascientist.com/performance-measures-rmse-mae/

Shen, K. (2018, June 19). *Effect of batch size on training dynamics.* Retrieved from Mini Distill: https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e