Stock market prediction using LSTM

Loris Persico - 12244072

August 19, 2023

1 Motivation

The stock market has always been a topic of great interest to investors, economists and researchers. Accurately predicting stock market movements can provide valuable insights into making informed investment decisions and managing financial risk.

In this project, we aim to predict the stock market movements of the S&P 500 index using a Long Short-Term Memory (LSTM) model, a type of recurrent neural network (RNN) known for its effectiveness in sequence prediction tasks. The forecasting method used in this project is known as single-stage time series forecasting, this approach is in fact similar to technical chart analysis in that it assumes that forecasting the price of an asset is fundamentally a time series problem. The objective is to identify patterns in a time series that indicate the future trend of the series.

The importance of this project lies in its potential to offer a tool that can help investors and financial institutions make more informed decisions. Accurate stock market forecasts can help investors optimize their portfolios, improve risk management and potentially increase their returns.

2 Data

2.1 Data Description

The data used in this project consists of historical daily closing prices of the S&P 500 index, spanning 8 years of market price data.

The S&P 500 is a widely recognized and highly regarded stock market index. It represents the performance of 500 large-cap U.S. companies across various sectors, making it a comprehensive and diverse index.

The dataset also includes additional features such as open price, close price, adjusted closing price, high, low and volume.

- Open and Close represent the starting and final price at which the stock is traded on a particular day.
- High and Low represent the maximum and the minimum price of the share for the day.
- The adjusted closing price considers other factors like dividends, stock splits, and new stock offerings, it can be called a more accurate measure of stocks' value.

- Volume in the stock market means the total number of shares traded in a specified time frame

Date	Open	High	Low	Close	Adj Close	Volume
2015-01-02	2058.899902	2072.360107	2046.040039	2058.199951	2058.199951	2708700000
2015-01-05	2054.439941	2054.439941	2017.339966	2020.579956	2020.579956	3799120000
2015-01-06	2022.150024	2030.250000	1992.439941	2002.609985	2002.609985	4460110000
2015-01-07	2005.550049	2029.609985	2005.550049	2025.900024	2025.900024	3805480000

Figure 1: Example of some values from the dataset

The data is obtained from reliable financial sources such as Yahoo Finance (yahoo.finance.com) via an API which is updated daily. To extract the data, i used the pandas DataReader package, a library that provides a function to extract data from various Internet sources into pandas DataFrames. Historical data for the S&P 500 is readily available and well-documented, covering a long time span. This availability allows for extensive data analysis and model training.

2.2 Data Preprocessing

Data properties:

- Balance: The dataset is balanced consisting of daily observations over several years resulting in a relatively even distribution of data points.
- Scale: The data is continuous and scaled with closing prices typically ranging from hundreds to thousands.
- Preprocessing: Data preprocessing involves handling missing values by filling them with the previous or next available value. Additionally the Min-Max scaling is applied to normalize the price values in our data to a range between 0 and 1.

3 Theoretical part

3.1 Literature Overview

Various deep learning architectures have been developed to solve diverse problems. The reason being, in a basic feedforward neural network architecture, information can only travel forward, and each input is independently processed, thus not retaining information from the previous step.

ARIMA model The ARIMA model gets frequently used in the analysis of time-series data. It combines autoregressive and moving average components known as AR and MA, respectively. This combination represents time-related trends and seasonality patterns. Despite ARIMA being good at demonstrating short-term trends, it may

struggle to capture long-term relationships in stock price data due to its linear nature.

GARCH models GARCH models try to capture how volatility clusters, which frequently happens in financial time periods. Even though GARCH models are outstanding at modelling price fluctuations, they might not directly forecast changes in prices.

Machine learning models, such as random forests and support vector machines use historical patterns to make predictions. While these models can include various aspects and detect nonlinear relationships, they may not be as efficient at handling temporal dependencies as recurrent neural networks (RNNs).

The architecture of RNNs consists of loops that allow relevant information to be kept in memory therefore RNNs are more suitable for sequential data modeling and time series applications, such as stock market forecasting. However recurrent neural networks suffer from short-term memory in fact if a sequence is long enough, they will have difficulty transporting information from earlier to later stages, so for this project i used the LSTM model, an improved version of RNNs specially designed to be able to overcome this problem.

The LSTM model utilises internal mechanisms known as gates to regulate the flow of information. These gates are able to learn which data in a sequence is necessary to keep or discard. This enables relevant information to be passed along the sequence chain and used to make predictions.

3.2 Approach

Learning Pipeline Steps:

- Data Preparation: The raw data goes through preprocessing to handle missing values, and to scale features to keep them in a consistent range. This stage makes certain that the dataset possesses a suitable format for LSTM model's input. I used the MinMaxScaler to normalize the price values in data to a range between 0 and 1.

- Creating the Input Shapes: The information is split into a series of predetermined lengths creating a time frame that is needed for models like LSTM.

- Layout of the LSTM Model: The arrangement of the LSTM model comprises of sequences and a target prediction horizon input. The model has two layers of LSTM stacked on top of each other and two layers of the Dense class from the Keras library, which enables it to identify patterns at various degrees of abstraction and reduce the risks of overfitting.

- Training and Validation: We train the model on past data and reserve a section for validation. This set is for validation, an indicator of how well the model can operate on unseen data.

- Assessment and Comparison: Metrics such as Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are used to assess how well the model performs. These metrics indicate how accurate the predictions are. Moreover, images

are used to demonstrate how the model accurately matches the shape of real stock prices.

Based on other publications, I tried changing the hyperparameters of the model, in particular by changing the batch size I was able to obtain these different results:

Batch size	**RMSE**
4	56.4490
8	57.8267
16	**53.7826**
32	79.7235

As we can see from these results compared with those of a similar online study, we deduce that the best results are obtained using a batch size of 16 as we obtain the lowest RMSE

Table 6. List of the best hyperparameters for multilayer LSTM models.

No. of neurons	Batch size
(10, 5)	4
(20, 10)	16
(50, 20)	16
(100, 50)	16
(150, 100)	16

Figure 2: Results from another paper

4 Implementation

I used a fully connected network with four layers, consisting of two layers of the LSTM class and two layers of the Dense class from the Keras library. I've chosen this architecture because it's one of the most commonly used and it's a relatively simple and a good starting point for addressing this kind of time series problems.

Before training the NN, splitting the data into two separate sets is needed, one for training and one for validation, to make sure they are in the right proportion. The model will then be trained on the last 8 years of market data and we will then use the trained model to predict the next day's closing price based on the performance of the last 50 days. Specifically, we will use 80% of the data as the training set and the remaining 20% as the evaluation set.

I then divided the training data (x_train) in many pieces, so-called mini-batches. The neural network processes the mini-batch one by one during the training process and creates a separate forecast for each mini-batch. In this case my x_train contains

1668 mini-batches. Each contains a series of quotes for 50 dates. In y_train, we have 429 validation values.

When selecting the number of neurons for a neural network layer, there are a few general guidelines to follow:

- A larger number of neurons can allow the model to capture more complex patterns in the data, but it may also increase the risk of overfitting.
- A smaller number of neurons can reduce the risk of overfitting, but it may also limit the model's ability to capture complex patterns.

The input data contains values for 50 dates. Consequently, the input layer must have no less than 50 neurons for each value. The second layer should have 35 neurons, as per the formula ((50 + 1) * 2 / 3). And the last layer must have a solitary neuron since the forecast will comprise just one price point for a single time step.

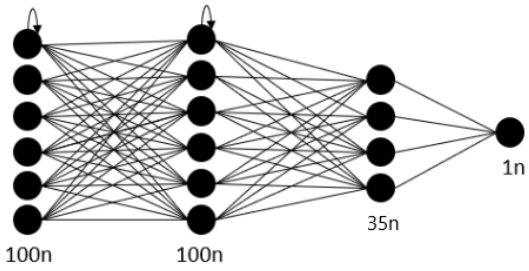

Figure 3: The architecture of the recurrent neural network

Listing 1: Configure the neural network model

```
model = Sequential()
neurons = sequence_length
# Model with sequence_length Neurons
# inputshape = sequence_length Timestamps
model.add(LSTM(neurons, return_sequences=True, input_shape=(x_train.sha
model.add(LSTM(neurons, return_sequences=False))
model.add(Dense(35, activation='relu'))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
# Training the model
model.fit(x_train, y_train, batch_size=16, epochs=25)
```

5 Evaluation

In order to evaluate the model's performance, we provide the test data (x_test) that we have generated in a previous step to the model to get some predictions, while keeping in mind that we initially scaled the input data to 0 and 1, therefore before interpreting the results we must inverse the MinMaxScaling from the predictions.

Listing 2: Inverse the scaling

```
y_pred_scaled = model.predict(x_test)
y_pred = mmscaler.inverse_transform(y_pred_scaled)
y_test_unscaled = mmscaler.inverse_transform(y_test.reshape(-1, 1))
```

By testing the model, with the parameters selected earlier, on the data I obtained these results:

Median Absolute Error (MAE):	41.85
Mean Absolute Percentage Error (MAPE):	1.02%
Median Absolute Percentage Error (MDAPE):	0.83%
Root Mean Square Error (RMSE):	53.78

The median absolute error (MAE) quantifies the median absolute difference between predicted and actual values. In this context, an MAE of 41.85 suggests that, on average the model tends to predict a bit too pessimistically.

The Mean Absolute Percentage Error (MAPE) measures the difference between the predicted and actual values as a percentage. The model's predictions on average deviate from the actual values by approximately 1.02%. A lower value of MAPE indicates relatively small percentage errors, meaning better predictive accuracy of the model.

The Median Absolute Percentage Error (MDAPE) provides a measurement of prediction accuracy, similar to MAPE, though it employs the median absolute percentage difference.

The root mean square error (RMSE) shows the average size of the mistakes in the forecast, and it highlights the biggest miscalculations because of the quadratic factor. An RMSE of 53.78 shows that, on average, the model's predictions differ from the actual values by approximately 53.78 dollars.

The graph where we compare the actual and the predicted values demonstrates the model's ability to capture trends and patterns in the stock market data.

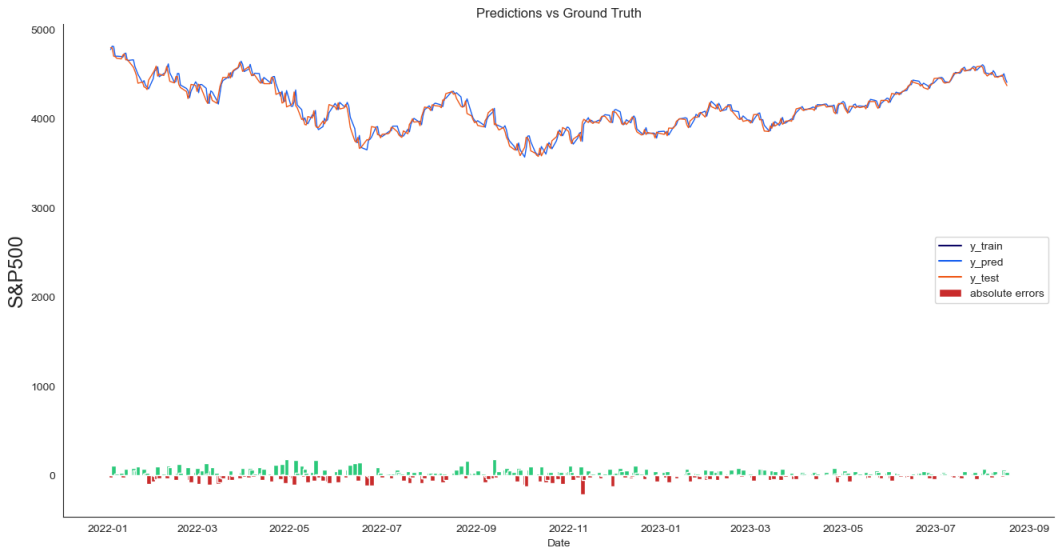

Figure 4: Graph of predicted and actual values

Now using a new dataset as input for our forecast model, we can use it to make a forecast for the next day.

```
1/1 [==============================] - 0s 21ms/step
The close price for S&P500 at 2023-08-19 was 4369.71
The predicted close price for the next day is 4356.9599609375 (-0.29%)
```

Figure 5: Predicting the stock closing price for the next day

6 Conclusions

Predicting share prices is a very interesting challenge indeed, but making an accurate and reliable price prediction becomes increasingly difficult the further into the future you look. This is because the stock market is influenced by a large number of mostly unpredictable factors. In fact, this project has focused on developing an LSTM-based model to predict the general performance of the S&P 500 index, using and combining what little tangible data can be used to make predictions using objective data.

From this point of view, I think I have also achieved a good result in line with other projects and publications found online where the same conclusions and results are reached.

To improve prediction in the near future, textual information such as investor sentiment from social media, company earnings reports, immediate policy news and market analysts' research reports could be incorporated into the model so as to try to influence prediction not only based on objective numerical data but also on information that is difficult to interpret otherwise.

Another potential direction of future work could be the development of hybrid predictive models, combining LSTM with other neural network architectures such as Prophet, a procedure for the prediction of time series data based on an additive model in which non-linear trends are adjusted for annual, weekly and daily seasonality as well as holiday effects.

7 Bibliography

Christopher Olah *Understanding LSTM Networks*
url: **https://colah.github.io/posts/2015-08-Understanding-LSTMs/**

Neptune.ai *Predicting Stock Prices Using Machine Learning*
url: **https://neptune.ai/blog/predicting-stock-prices-using-machine-learning/**

Keras *Keras API referenceDense layer*
url: **https://keras.io/api/layers/core_layers/dense/**

Florian Follonier *Stock Market Prediction using Univariate Recurrent Neural Networks (RNN) with Python*
url: **https://www.relataly.com/univariate-stock-market-forecasting-using-a-recurrent-neural-network/122/h-step-4-creating-the-input-shape**

Roshan Adusumilli *Machine Learning to Predict Stock Prices*
url: **https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233**

Various *Predicting stock market index using LSTM*
url: **https://www.sciencedirect.com/science/article/pii/S2666827022000378sec3**

Pier Paolo *Stock Market Analysis and Time Series Prediction*
url: **https://www.kaggle.com/code/pierpaolo28/stock-market-analysis-and-time-series-prediction/notebook**

FARES SAYAH *Stock Market Analysis + Prediction using LSTM*
url: **https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm**