Machine Learning Engineer Nanodegree

Capstone Proposal

Ana Isabel Casado Gomez
January 5th, 2021

Proposal

Domain Background

In all capitalist economies, financial markets play a very important role. It allocates resources and creates liquidity for businesses and entrepreneurs. They are made by buying and selling different types of financial instruments like equities, bonds, currencies, and derivatives. A lot of people gain and lose money from them, that's why it's so important to try to predict the stock market in order to make decisions safely. As the Machine Learning Field keeps developing and generating better models, it has been applied to financial markets too with the most outstanding technique that involves the use of Neural Networks and Genetic Algorithms. We will base this project on Yahoo! Finance data.

My personal motivation and interest in this project are because I want to learn more about financial markets in order to start investing in them too. For that, I would like to develop a stock predictor algorithm for the companies I believe in, and make sure that I do not waste my savings.

Problem Statement

The main idea of this project is to understand what is the next best action for a given symbol or company. In order to know the next best action, the stocks must be predicted.

The challenge of this proposal is to build a stock price predictor that takes daily trading data over a certain data range as input and outputs the projected estimates for the next k dates. This is a regression problem as it will predict the price value, and not just if the stock price goes up or down, that would be the classification problem. After the predictions are done, one recommendation in order to buy, hold, or sell will be suggested too.

In order to achieve this, we will also compare predictions with time series versus neural network methods, to evaluate what is best for the given problem.

Datasets and Inputs

The dataset used for this project comes from Yahoo! Finance. As Yahoo! finance stopped their historical data API, and it's not accessible anymore, the python *yfinance* library will be used to download historical market data. This library was created to solve the project of getting reliable data from Yahoo after they decommissioned their historical data API.

The library retrieves as pandas data frame the historical dataset of the symbol provided and the following columns:

1. **Date**: day of the specified symbol as the index.
2. **Open**: the open price of the symbol on the given frequency time.
3. **High**: the highest price for the given frequency time.
4. **Low**: the lowest price for the given frequency time.
5. **Close**: the price that the symbol closed for the given frequency-time adjusted for splits.
6. **Adj Close**: the adjusted close price adjusted for both dividends and splits for the given frequency time.
7. **Volume**: the physical number of shares traded of that stock for the given frequency time.

The frequency of the data could be daily, weekly, or monthly. For this project, the data usage will be daily. The companies to analyze will come from the DAX, the blue-chip stock market index holding the 30 major German companies trading on the Frankfurt Stock Exchange.

The date range used to extract the data for training and testing will be from 1st January 2016 until 31st December 2020, both dates inclusive. Some of the companies don't have that full range because they are newer than others. Each equity has 1265 data points except Delivery Hero AG, which only has 884 dates because its IPO started 30th June 2017. It means that it includes 5 full years of data for 29 companies and 2.5 years for one.

The list of the companies following the format "Symbol - Company name - number of data points" are:

- SAP.DE - SAP - 1265 data points.
- DHER.DE - Delivery Hero - 884 data points.
- BAS.DE - BASF - 1265 data points.
- VOW.DE - Volkswagen - 1265 data points.
- SIE.DE - Siemens - 1265 data points.
- BAYN.DE - Bayer - 1265 data points.
- DAI.DE - Daimler - 1265 data points.
- IFX.DE - Infineon Technologies - 1265 data points.
- ALV.DE - Allianz - 1265 data points.
- BMW.DE - BMW - 1265 data points.
- DTE.DE - Deutsche Telekom - 1265 data points.
- EOAN.DE - E.ON - 1265 data points.
- DBK.DE - Deutsche Bank - 1265 data points.
- ADS.DE - Adidas - 1265 data points.
- RWE.DE - RWE - 1265 data points.

- FRE.DE - Fresenius - 1265 data points.
- DB1.DE - Deutsche Boerse - 1265 data points.
- LIN.DE - Linde - 1265 data points.
- HEN3.DE - Henkel - 1265 data points.
- VNA.DE - Vonovia - 1265 data points.
- BEI.DE - Beiersdorf - 1265 data points.
- MURGY - Münchener Rückversicherungs-Gesellschaft - 1265 data points.
- HEI.DE - HeidelbergCement - 1265 data points.
- FME.DE - Fresenius medical care - 1265 data points.
- 1COV.DE - Covestro - 1265 data points.
- DWNI.DE - Deutsche Wohnen - 1265 data points.
- MRK.DE - Merck - 1265 data points.
- CON.DE - Continental - 1265 data points.
- MTX.F - MTU Aero Engines - 1265 data points.
- DTE.DE - Deutsche Telekom - 1265 data points.

As we are dealing with time series data, our observations are not independent from each other and splitting the data randomly is not an option. Otherwise, we will split observations along with the sequences. As we don't have too many data points per equity, we will use a time series cross-validator. It will split our time series into K chunks (probably K = 5 as we have 5 full years of data for most of the equities) and it will use the first segment to train the model and it will test it with the second. In the second round, it will train the model in the first and second segment and use the third for testing. In this way we will do a K-1 times of cross-validation.

At the end of the process we will use a variant of Walk-Forward-Validation and use it to validate the model, the available data points from the current month we are in, January 2021 until it's available (2 weeks data approx.).

Solution Statement

The solution to the exposed problem falls into building a neural network algorithm to predict the stock price. However, firstly we will make an intense exploration of the data to find trends and patterns. The best you know your data, the best you can model it. The price of the stock will be forecasted using Prophet, the Facebook library for time series, and the Neural Network model, in order to evaluate which method has better performance against the benchmark model.

Once the predictions are made, we can make intelligent recommendations to buy or sell stocks.

Benchmark Model

In order to evaluate the success of the Neural Network model, we will benchmark the model first using the well-known statistical method: ARIMA from `statsmodels` library. As it has been one of the first algorithm applied to time series, we will use it in this project to evaluate the machine learning algorithm performance.

Evaluation Metrics

The common metrics to evaluate forecast accuracy, among others, are:

- RMSE (root mean square errors)
- MAE (mean absolute error)
- MAPE (mean absolute percentage error)
- MASE (mean absolute scaled error).

As sometimes MAPE could prevent the neural network from converting in combination with Adam, we will base our success on the RMSE metric, with a close eye on the performance of the other most common metrics.

We will also look into the Sharpe ratio to measure the performance of the model's predictions, as it could help to measure or control the volatility in the model's output.

Project Design

Data Pre-processing

The stock prices data retrieved from *yfinance* library doesn't contain all the data due to the fact that stock markets don't function on weekends or public holidays. This missing data will be imputed via linear interpolation between the last available observation and the next one. In case of just one missing value, the formula will be similar to $(y_1 + y_2) / 2$. For weekends will be divided equally by 3, adding the same separation to each point from the last and the next one available.

For the neural network, we will have to scale the data because if not, the model will learn large weight values that are often unstable, meaning that it could lead to poor performance during the learning stage and sensitivity to input values with a higher generalization error. The proposed method is Normalization under the following formula: $y = (x - min) / (max - min)$. The output predictions will be inverse transformed to show the real predicted price.

Data Exploration and Benchmarking

In the exploration phase, the trends and seasonality will be analyzed. Furthermore, an ARIMA model will be fitted to the data in order to create a benchmarking model that we will use as a base for the success of our neural network model and the prophet model that we will implement too.

Model Learning and Prediction

Two models will be created in order to forecast the stock price of a company. We will use a neural network model and the prophet library. The latest model will be computed as a way to contrast a machine learning model with a more traditional one on top of the benchmarking. It will give us more insights into different types of models and what works better for each one. Hyperparameter tuning will be carried out to obtain the most promising results. The best performing model will be triggered by the web app interface.

The type of Neural network proposed for this project is the Recurrent Neural Network like LSTM that explicitly handles the order between observations when learning a mapping function from inputs to output. They also learn temporal dependence. We will start from the simplest model of LSTM and try to improve the structure by different layers. The simplest is the Vanilla LSTM with just one single hidden layer of LSTM units:

X → LSTM Layer → Dense Layer → Forecast (y_hat)

The next structure to try is the Stacked LSTM:

X → LSTM Layer → LSTM Layer → Dense Layer → Forecast (y_hat)

We could also try the Bidirectional LSTM that could be beneficial as it learns the input sequence from both forward and backwards.

X → Bidirectional LSTM Layer → Dense Layer → Forecast (y_hat)

A more complex model to try if the performance is not good enough is the hybrid model CNN-LSTM, that uses a CNN model to interpret subsequences of input that together are provided as a sequence to an LSTM model to interpret. Example:

X → CONV1D → MaxPooling1D → Flatten → LSTM Layer → Dense Layer → Forecast

The model will be tuned starting by the parameters: training epochs, batch size and neuron numbers. And we can extend, depending on time, to dropout, layers, optimization algorithm or loss function.

Decision Making

The output recommendation from the model to the customer will be developed based on a naive greedy strategy with the assumption that buys and sells all our stocks for a company together. We keep a running average and standard deviation of the actual adjusted stock values of the previous k days. K to be defined after analysis.

- Buy Decision: if the prediction for the next day is n standard deviations less than the mean.
- Sell Decision: if the prediction for the next day is m standard deviations more than the actual adjusted value when it was bought, then we sell.

In any other case, we will wait.

Interface

The interface will be built as a simple HTML web page that lets users choose the favorite company from the DAX market and the date where they bought the stocks if it's applicable.

References

- https://pypi.org/project/yfinance/
- https://towardsdatascience.com/an-end-to-end-project-on-time-series-analysis-and-forecasting-with-python-4835e6bf050b

- Comparison of different Methods for Univariate Time Series Imputation in R, by Steffen Moritz, Alexis Sardá, Thomas Bartz-Beielstein, Martin Zaefferer and Jörg Stork https://arxiv.org/pdf/1510.03924.pdf
- https://finance.yahoo.com/
- Evaluating forecast accuracy https://otexts.com/fpp2/accuracy.html
- Time series models https://www.cs.cmu.edu/afs/cs/academic/class/15782-f06/slides/timeseries.pdf