Implementing a Neural Net for Stock Forecasting

Helen Mehreteab and Anusha Murali

December 8, 2019

1 Introduction

Predicting the price of a company stock is one of the most difficult tasks, even for experienced traders. The future value of a stock is dependent on numerous factors such as the company's financial health, the historical stock prices, analyst repots, and the general market volatility. In this project, we use the historical stock prices of a company to predict its future stock prices. Using three popular machine learning algorithms, namely linear regression, Prophet and Long Short Term Memory (LSTM) algorithm, and elementary data manipulations on widely available CSV data for publicly traded companies, we demonstrate that one could predict stock prices with a high degree of accuracy.

2 Design Overview

Time series forecasting and sequence prediction is one of the most difficult problems in computer science. One of the well-known time-series forecasting problems is stock forecasting. Even though there are numerous factors that affect a stock's value in any given day, in this project, we decided to focus only on the prediction of stock value based on its historical data.

Our project implements stock forecasting using three most popular machine learning algorithms, namely (1) Linear Regression, (2) Facebook Prophet and (3) Long Short Term Memory (LSTM). Our goal was to compare the performance of these three algorithms and to demonstrate that the LSTM outperforms the other two due to its ability to retain relevant information over a long period of time.

2.1 Design Decisions

Following are some of the important design decisions that we took during the course of our project.

2.1.1 Historical Data

The financial data of most of the publicly traded companies are available from numerous internet sources. We had a choice of downloading any required stock data by directly connecting to a website such as `quandl.com`. However, we decided to download the historical data from `finance.yahoo.com` for a handful companies on our local machine.

2.1.2 Training and Validation Data

In order to ensure the accuracy of our machine learning models, we decided to validate them using part of the downloaded historical data. We decided to split the historical data of a company into 80% of training data and 20% of validation data. Any good model that was built using the training data is likely to yield a highly accurate results against the validation data. This cross validation allows us to confidently use our model for stock forecasting in the future.

2.1.3 Python Libraries

One of the most difficult problems that faced during the project was incompatibilities of various libraries. We could not download all the libraries that we needed onto CS50 IDE as we did not have sufficient user space. We could not use the latest Python version (3.8) as some of the libraries did not work on the latest version.

2.1.4 Python Library `fastai`

Based on our research, the Python library `fastai` is quite popular in the Python user community. However, we couldn't install it easily on our laptops. We spent more than six hours trying to resolve all the dependencies before finally getting it to work. We needed `fastai` to decompose `Date` column in to its basic functional parts. Even though we started writing a basic function to decompse `Date` column, we were happy that we were able to successfully install `fastai` on our laptops.

2.1.5 How to Display the Results?

Our initial version of the program was only generating numerical output. Although the initial output was sufficient to demonstrate the efficiency of all the three algorithms, we decided to explore the Python plot library (`matplotlib`) for our purpose. The documentation for `matplotlib` from `python.org` was quite useful in extending our code to generate graphical outputs from our models.

2.1.6 Generating Tabular Output

In addition to generating line plots, we wanted to display the error rates of the three machine learning models. The Python library called `PrettyTable` allows one to organize the results in compact tabular format. However, we spent many hours struggling with this library due

to a bug in the Python IDLE editor. Fortunately, the bug did not appear when we tried our code directly from the terminal.

2.1.7 Accuracy versus Run-time of LSTM

We found that the LSTM model outperformed both the Linear Regression and the Facebook Prophet for `epoch` > 6, demonstrating its power. However, we found that the LSTM model requires a very long time to finish when we increased `epoch` above 10. Therefore, we decided to run the LSTM model with a default `epoch` value of 2.