CS 4850 Fall Semester, 2023

SP-23, Stock Price Prediction Web Application

Anthony Vicente, Gustavo Ramos, Christian Robertson

Sharon Perry, 11/26/23

Website Link: https://avicent5.github.io/StockPredictor-Website/StockPredictorWebsite

GitHub Link: https://github.com/Advacated/SeniorProject/blob/main/DisplayCSV.py

Table of Contents

PROJECT OVERVIEW 3

PROJECT PLAN & DESIGN 3

PROJECT REQUIREMENTS 3

SOFTWARE DESIGN AND ARCHITECTURAL DRAWING 4

VERSION CONTROL DESCRIPTION 5

INITIAL PROJECT STEPS TAKEN FOR IMPLEMENTATION 5

PROJECT RESEARCH 5

BACK END DEVELOPMENT 6

FRONT END DEVELOPMENT USING STREAMLIT 7

CHALLENGES 8

TESTING PLAN & REPORT 8

SUMMARY & FINAL THOUGHTS 9

Project Overview

The aim of this senior project is to build a web application platform which will conduct sophisticated predictions of current stock market prices using Artificial Intelligence. More Specifically, we use Artificial Recurrent Neural Networks such as the Long Short-Term Memory (LSTM) network to conduct these predictions. We use the LSTM to analyze time series data, which we use to build a general forecasting time series frame in which the neural network can highlight long-term patterns from sequential data (Stock prices). Using this data acquired from the LSTM, we use financial indicators, such as Relative Strength Index (RSI)/ Exponential Moving Average (EMA), to make calculations and produce the best accurate prediction possible. All of which will be hosted onto a web application platform for the user to use at any time.

Project Plan & Design

At the initial stage of our project, we outlined a plan that included general tasks to be completed. During our initial team meeting, we as a collective whole discussed what the roles of each member would be, and by the end we all agreed on our roles and tasks associated with it. Since we are a team of three, we decided that two team members would oversee programming development while the other would take on the documentation aspect of the project. We also agreed that members were not limited to only their roles, meaning that team members could help with documentation and programming.

For the actual project plan design itself, we outlined a simple setup to avoid any unnecessary complexities during our development process. We focused on defining the overall project scope and decided that our version control would be done with the help of GitHub. We also listed out what was needed for deliverables such as the SRS and SDD documents. We also listed possible programming languages and frameworks that we could use to implement this project onto a web platform. Ultimately this is just a general summary of our design. There will be a deeper explanation of our design in the discussion section of this report.

Project Requirements

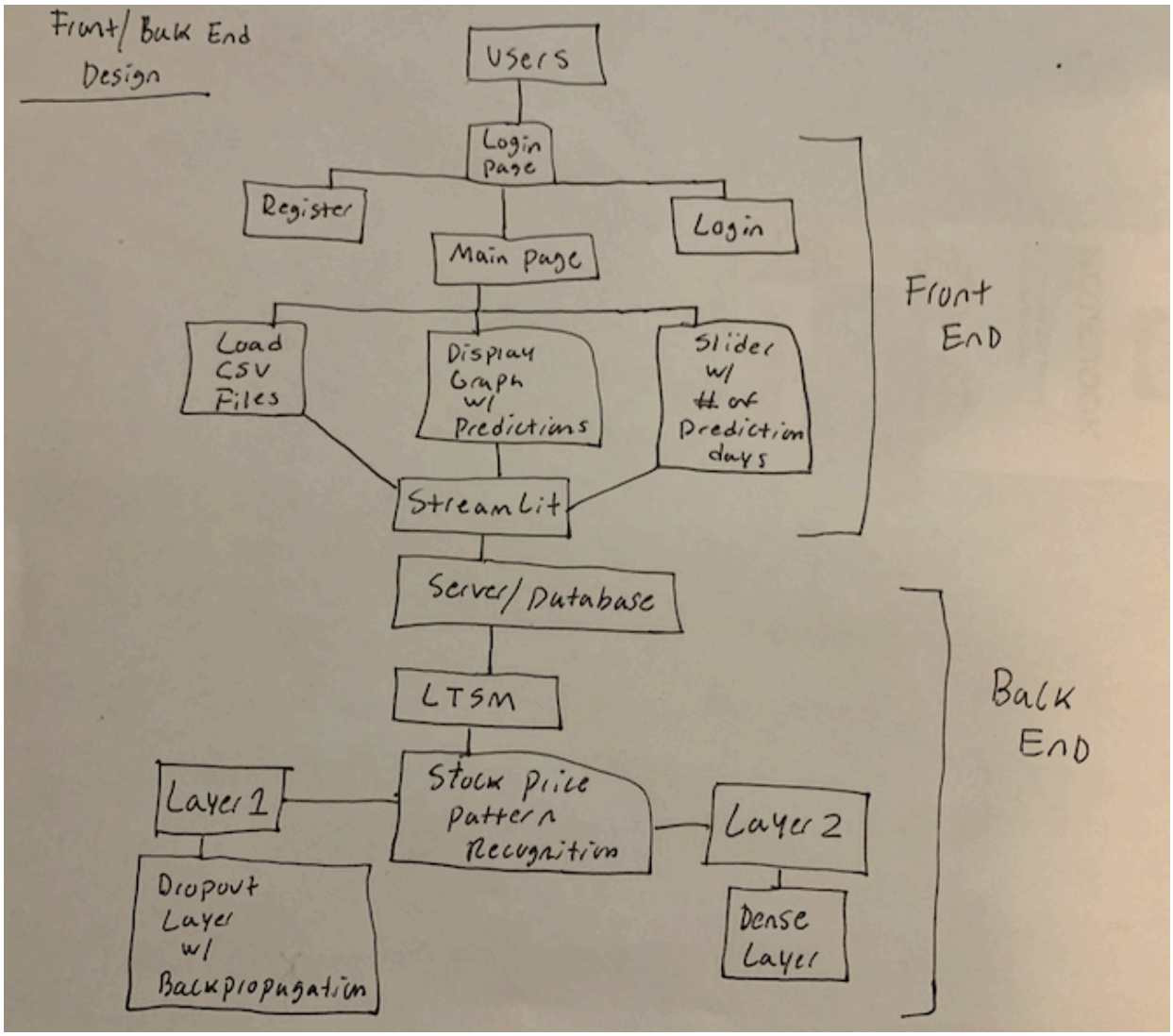
When we were designing our project, one main aspect that we focused on was the project requirements. The requirements that we came up with served as an outline of how we wanted to build our stock price prediction web platform. In the end, our main goal was to meet all functional/nonfunctional requirements listed on our software design checklist.

The list below is a sample of our requirements list. It's also worth noting that these are the most important ones to ensure a reliable and user-friendly stock price prediction web app.

- Does the design ensure user data confidentiality using encryption methods like hashing or passwords?
- Is the user interface intuitive for selecting stocks?
- Are the insights presented in an easy-to understand format (graphs, charts, etc...)

- Is the home page layout responsive and user friendly.
- Are there provisions to adding or removing stocks from the user's watchlist?
- Does the design support interactive charts and graphics?
- Are stock trends updated at an appropriate time interval?
- Does the design clearly differentiate between historical data and predictions?
- Can users customize which financial indicators to display?
- Can the web app manage high volumes of traffic and avoid problems such as crashes or server issues.

Software Design and Architectural Drawing



Version Control Description

We use GitHub as our primary source control, which helped in tracking and managing modifications done to our code when we developed our front/back end for our web app. We also used GitHub for our branching strategy which was divided into two sections, the main branch, and final app. The main branch serves as an overall stable version of our project, meaning that this version is ready for production. The main branch served as our back end and research branch. This is where we developed the necessary python code to be able to predict stock prices and draw graphs for us to reference. We would also research and apply necessary changes to display accurate stock predictions to this branch. The final app branch is where the whole front end was developed all the requirements and front-end code was done here. This branch branched directly off from the main branch allowing us to use the backend code to make the front end around it. Deployment of the web app was also done using this branch hence the chosen name.

Initial Project Steps Taken for Implementation

This next section will outline in detail the steps that were taken for us to create our Stock Price Prediction Web App. As mentioned before, we outlined a simple setup to avoid any unnecessary complexities during our development process. Initially when we began working on this project, we had a big discussion on who we wanted to tailor this project towards. Would it be for trained individuals who have experience in the field of stock market trading, or would it be for someone who had no absolute experience with trading? We spent quite some time dealing with this matter, but we decided that it was best to make this web app for anyone, regardless of their experience level. Thus, our goal in mind was simple, to create an easy-to-use web application that would generate stock price predictions for those who are interested in stock market trading.

Our outlook for this project this semester consisted of separating tasks to achieve one main goal for each month. For instance, August was about defining our project scope and having a general idea of what we wanted to implement. This included team meetings discussing what type of stock market data we would use for our LSTM model. How would we train this neural network to make stock price predictions, and how would we host all this information? August was mainly having a general idea what we would do for our project.

Project Research

For the month of September, we focused on actually delivering on the project scope. Initially we thought it would have been a great idea to add our price predictions as a website extension on existing stock market price websites, but we agreed that a web platform of our own would be more appropriate. We setup our version control which is done with the help of GitHub. We use GitHub as our primary source control, which helped in tracking and managing modifications done to our code when we developed our front/back end for our web app. We also used GitHub for our branching strategy which was divided into multiple sections.

During this month we also decided to begin conducting research on the LSTM neural network and how we could implement it. We began by visualizing stock trends over the course of two weeks while also learning about how the LSTM could detect these patterns efficiently. Once we came up with some ways on how to implement this, we set up our deep learning environment and began training our sample data to build our Neural network model. We trained our LSTM for another two weeks to recognize stock price patterns and we evaluated the results soon after.

As we were analyzing the results from our RNN, we came up with our project requirements and overall design. As mentioned before, our emphasis on this project was to create a web app platform that would be easy to use while having high efficiency. With this in mind, we came up with functional/nonfunctional requirements that would help us achieve this goal. Safety was also a concern as it is a vital component when building a website. We included security aspects like data confidentiality and encryption methods. Methods like hashing for protecting user passwords and account information. We also thought that it would have been a great idea to design a password recovery system which would help users recover their password in case they had forgotten. Two factor authentication was another safety method that was in the talks of adding.

Of course, with any project efficiency was another area of concern. As this was the first time any of us worked with the LSTM, we were curious as to how fast our LSTM model would be making pattern recognition. With this in mind, we designed requirements that would give us an efficient time frame for our project. One of these requirements being home page layout and how responsive it would be for a user. We aimed to have a quick responsive page layout with very little buffer. Another aspect that was focused on was stock trend time intervals. Perhaps the biggest factor in efficiency performance was whether our website would update trends based on real time or defined intervals. We experimented with both these ideas and found that neither time interval has a bigger advantage on performance over the other. So, we decided to update stock trends based off real time to generate the most accurate prediction possible for the user.

Usability was also a key factor that we had in mind for our project design. As mentioned before, we wanted to design a web app platform that would be easy to use. One of the ways we could implement this is by designing our front end to be as simple as possible. Meaning no unnecessary clutter or information. Just the necessary visuals and stock price information. Our visuals were very important as we wanted them to be presented in an easy/visually appealing way. We wanted our graphs to show stock price changes in an effective manner while also avoiding information clutter like other stock market websites. Ultimately our goal in mind was to create a minimalist website showing the predictions of a stock.

Back End Development

Reaching the month of October, we began building our backend which implements the LSTM model and our frontend which was used for our web app frame. We built both our front and back end using the Python programming language. Python was our first choice since it was the most efficient language when dealing with recurrent neural networks. Python also has the ability to build your own web application using their custom frameworks. The main framework that we

used is the streamlit library. All these reasons are why we chose Python as our primary programming language.

The back end of our web app consists of machine learning programming that is used for pattern recognition. Again, we use the pattern recognition functionality of our LSTM to analyze trends in stock market prices. Typically, we use this recurrent neural network to recognize patterns in sequences of data, such as time series, and we implement this with multiple layers. Our first layer in the LSTM model includes 50 neurons and is considered a hyperparameter. This hyperparameter determines the learning capacity of the first layer. So, in essence, the layer can capture more complex patterns with more units. Although having a model with more neurons allows for deeper pattern recognition, it does provide a con with computation which almost always increases it leading it into a higher risk of overfitting.

The first layer also contains returning sequences that let us know if the LSTM layer returns a full sequence for the next layer. This is a useful method within our backend which is beneficial to stacking LSTM layers. Our backend also contains a dropout layer for our LSTM. Dropout is defined as a regularization technique that selects neurons at random. These randomly selected neurons are dropped out during the training phase, meaning that they have no contribution in down streaming neurons during a forward pass. No contribution also means that weights are not updated during backpropagation and is the main reason for avoiding overfitting. This layer includes a 20 percent dropout meaning that 20 percent of the neurons in the next layer are turned off during each training session. This inevitably forces the neural network to learn.

The second layer of our LSTM in our backend is similar to the first layer. The only difference is that this layer does not return sequences like in the first layer. This layer also contains a dense layer which is the final layer in our LSTM model. This layer connects every input from preceding layers to every neuron in its own layer. All these layers in the end create our model training which is what we use to train our data. Within our code, we train data with 100 epochs and a batch size of 32. We also evaluate the model's performance from the test data and print the losses. To make predictions, we start with predicting the next ten days based on a test set from 60 days prior. After each prediction day, a new predicted price is added to our known data and the oldest day is removed. This creates an accurate prediction scale since it uses the most recent data. Finally, to get prices, the values are scaled between 0 and 1 which are then inversely transformed to get actual predicted stock prices.

Front End Development Using Streamlit

The front end of our web application Is the main component for visualization of our back-end code. We use the streamlit library, which is a python library used to build interactive web applications, to build our web skeleton and overall front-end design. We wanted our front end to be as easy to understand as possible. Meaning that we wanted to build a user-friendly interface that includes easy to use leveraging sliders and drop downs for user interaction. Our front-end code contains our stock price prediction feature. It is by the use of data handling which is the processing of stock data using pandas. NumPy and scaling (minMaxScaler). For visualization, we imported the matplotlib module, which allows for graph creations, the ability for plotting, and displaying of the predicted stock prices.

Our front end also contains a preset of csv files which contain company information on their stocks. One thing about our front end is that we wanted it to be as user interactive as possible. We accomplished this by sidebar control. That allows the user to select CSV files within the set of CSV files. Our front end dynamically generates visual outputs for stock price prediction and the actual prices. This allows us to enhance the UI Which provides a smooth application experience for all users. His work creates our final web app platform That is made available across all operating systems and even on mobile devices.

Challenges

Some of the challenges that we faced were during the implementation of streamlit in our project. We can separate them into challenges for the requirements to be able to use this library with the other ones we already had and when we went to deploy the web app. Our backend was built using various libraries, for example TensorFlow. TensorFlow is a popular python library used for machine learning and artificial intelligence. The problem that first came to the surface was when I went to start developing the front end. Our backend used the newest version of TensorFlow, which was 2.12 at the time. Streamlit unfortunately didn't have compatibility with this new version. The next course of action was to see which version of streamlit and TensorFlow were compatible. This ended up being Streamlit version 1.12.0 and TensorFlow 2.11.1. All of which is specified in the requirements text in the zip file for the code.

The final challenge and probably the most stressful one was deploying the app. Streamlit uses a cloud and server to host web apps, and this is where we decided to deploy our web app. To be able to upload your web app to their cloud you need to add a requirement text file. The Streamlit Cloud launches apps directly from any GitHub repo, so your app code and dependencies need to be on GitHub before deploying the web app. Once you are ready to deploy the Streamlit cloud cross references dependences in your project with the ones necessary for your app to be able to be deployed. At first try there were many discrepancies so I had to do some research to see which dependencies I could match to maintain my project working as intended and for the web app to be able to be deployed.

Testing Plan & Report

For any project, testing is a vital component to creating an efficient, well-made product. For our Stock Price Prediction web app, we came up with some test cases that would test for any bugs and overall performance. We made sure to test all functionalities using both white and black testing techniques. Techniques like statement/branch coverage were used to make sure each line of code was tested and working in sync with product functionality. Boundary value analysis was also performed to check the functionality of prediction days. We wanted to make sure the users were unable to access out of bounds predictions days (example –1 days or 12 days). Below is a chart of our test cases and their respective results. In the end, our project passed all test cases which is a strong indication that our project is bug free with high performance.

Test Case	Results
1. Sliders should be fully functional.	PASS
2. Boundary Analysis: Prediction days should only be accessed between 1-days.	PASS
3. All csv files are fully operational and are all accessible	PASS
4. Main Graph should plot stock prices successfully for days 1 – 10 for all CSV files.	PASS
5. Company Logo's should switch when a new CSV file is selected	PASS
6. Website functionality should be efficient with little to no buffer	PASS
7. Login page should successfully track/store both username and passwords of a user.	PASS
8. Login Page should have functional buttons	PASS
9. Login page should successfully allow a user into the main page with correct username/password credentials.	PASS
10. Users using the login page buttons should be able to create a username/password with any type of characters. (including special characters like '@#&*^'). Should also include a prompt displaying a successful register.	PASS
11. Login page should deny users with invalid credentials and should display a message letting them know	PASS

Summary & Final Thoughts

Over the course of this semester, we as a team have learned a lot from building this project. Initially when we started, we all had knowledge on how neural networks worked and how to host something onto a web app, but we never really implemented anything of the sort. Now as the project is complete, we all feel confident that we have obtained the knowledge to do this. As for the project itself, our goal was to create a web app that would generate accurate stock price predictions using the long short-term memory neural network. We implemented this web app by

creating an RNN (LTSM) to recognize patterns in stock prices that we pulled as data in our backend. Our LTSM uses multiple layers to function and in the end performs predictions that we retrieve. We then use streamlit with our front end to create our web app which will display our stock price predictions from our LTSM. We use streamlit as our primary front-end framework since it is what provides the main web app skeleton, along with our other features like displaying visual graphs of these predictions. Along the development process, we encountered challenges when developing with streamlit. As mentioned in our challenges section, one of the libraries that was used was TensorFlow. Our main issues were compatibility issues along with project deployment but in the end all issues were resolved. As we conclude this project, we as a team feel that we have built a fully functional web application that allows users around the world to generate stock price predictions. In the end we feel confident that we have achieved our main goal as well as exceeded our own expectations.