January 15, 2022

C964: Computer Science Capstone

Task 2 parts A, B, C, and D

Armondo Dobbs Jr

ID: 010111115

Part A: Project Proposal for Business Executives

Letter of Transmittal

January 8, 2023

James Kenway, CEO

Connected Enterprises

3200 Tudor Rd.

Sparks, NV 89431

Dear Mr. Kenway,

I am writing to propose the implementation of a stock price prediction program within your organization. Recent studies show how businesses that rely on manual methods of stock price prediction may be at a disadvantage when compared to those that implement a stock price prediction program, which can provide accurate predictions in a more efficient and cost-effective manner. With the use of advanced machine learning algorithms, this program will allow us to analyze historical stock market data and make informed predictions about future stock prices.

This proposed application works by using collected historical data and applying advanced computational algorithms to accurately predict future stock prices. By utilizing this program, you will be able to make more strategic decisions about your investments and potentially increase your return on investments. Additionally, the program can also be used to identify potential market trends and make educated guesses about which industries and companies may perform well in the future. To save on costs in this first iteration of the program, it will be more of a backtesting example with prepopulated training data to show how well the application would work if implemented in the past.

The funding requirements for this proposal are rather simple in this case as we recommend hiring two primary developers at a flat rate of $4,000 per month along with $1,500 to provide the relevant desktop computers and hardware they may need. These developers are highly qualified with certifications in project management and IT leadership. Along with these certifications, they have several previous projects that have proven to do well in the past. When the application eventually goes live, there will be a cost of $100 for maintenance and $100 for licensed APIs to allow for real-time data collection and make real-time predictions. This will leave a total of $9,500 to start the project off along with $200 for maintenance and continued use.

I understand that some members of your team may have concerns about the complexity of such a program, but I assure you that it is user-friendly and can easily be integrated into our existing systems. Furthermore, our team is more than capable of providing training and support to ensure that all members can effectively use the program. I strongly believe that implementing this program will give your company a competitive edge in the market and greatly benefit our organization in the long run. I would be more than happy to schedule a meeting to discuss this proposal in further detail and answer any questions you may have.

Thank you for your time and consideration.

Sincerely,

Armondo Dobbs Jr.

Project Recommendation

Problem Summary

Our stock price prediction project with machine learning involves training a model on historical stock market data to predict future stock prices. This can be done using various techniques such as time series analysis or deep learning. Data preprocessing and feature development are done to clean and format the data, and then the model is trained and evaluated using metrics such as accuracy and mean squared error. Once trained, the model can then be deployed in a program to make predictions on unseen data.

The setting for an application like ours is typically the financial market, where publicly traded companies have their stocks traded on various exchanges. The goal of the program is to predict the future price of a stock so that investors can make informed decisions about buying or selling that stock. There are a variety of factors that can influence stock prices, such as company performance, economic indicators, and market sentiment, and a prediction program can take these into account to make a prediction.

Our application meets a variety of our business needs. For example, if the organization is considering investing in a particular stock, the prediction program can provide an estimate of the future price of that stock, allowing the organization to determine whether the investment is likely to be profitable. Additionally, automating stock price prediction in this manner while using machine learning can save money and resources by reducing the need for human analysts and allowing for faster decision-making.

Deliverables for this proposal will be the completed prediction application with prepopulated company data on the FAANG group (Facebook, Apple, Amazon, Netflix, Google)

for training and visualization on the prediction process. Web deployment and mobile development will be out of scope in this proposal.

Application Benefits

As mentioned above, this proposed application can benefit our business in several ways:

- By providing an estimate of future stock prices, a prediction program can help businesses make more informed investment decisions, which can lead to increased profitability and reduced risk.
- The program can automate and optimize the process of monitoring and analyzing stock prices, which can save time and resources for the business.
- The program can help by providing insight into the factors that are driving stock prices and identifying potential trends in the market.
- As more data becomes available, machine learning algorithms can adapt and continue to improve their predictions, allowing traders to stay up-to-date with the latest market trends.

Application Description

This proposed application will be written using python in an IDE known as PyCharm while also using several of the few default machine learning libraries that are included. The historic stock data of five major companies are included which will be used for training purposes. Depending on which company is selected, the program will then compile all of its data to generate stock charts. Of these stock charts, one will provide a detailed view of the historical training data along with the predicted prices vs the actual prices to provide a valuable visual aid

in representing the data. This solves the time and resource problem of manual analysis as the user will simply enter the company they would like to see and the full report will be generated.

Data Description

The raw quantitative data used in this application will come from the open-sourced website known as Kaggle. The primary information needed from these data sheets is the closing price (independent variable) of each stock as these are what will be used to predict future prices (dependent variable). Data files will be managed and parsed directly through the application data frame. Outliers will be handled in the app however our limitation, in this case, is that our data time frame is between 2015 and 2021 which isn't particularly the ideal amount of data to have. To avoid incurring software costs during this training phase, our predictions will be made on past dates and compared to the actual amount that it was that day. This will allow users to see how the program would work if implemented by the company.

Objectives and Hypothesis

We hypothesize that while using historical stock prices and other relevant data, our machine learning model can accurately predict future stock prices with a high degree of accuracy. This will in turn solve the business need for more man hours as parsing and filtering data will be automatic and help our company make better business decisions.

The most important outcome of the application is that it should produce accurate predictions of future stock prices. This can be measured by comparing the predicted prices to actual prices and calculating the error rate. Getting an exact calculation correct will not be the particular goal, as in this case we are primarily concerned with the trendline in hopes that it matches up well with the actual prices.

Methodology

The methodology used for the development of this application will be agile. It is well suited for this program for several reasons:

- The agile methodology encourages rapid iteration and the ability to make adjustments as needed. In the stock market, new data and market conditions are constantly emerging and a prediction program should be able to adapt and make adjustments quickly.
- It places a strong emphasis on testing and receiving feedback. This is important for our program, as it allows for the program to be tested and refined based on the results of the predictions during testing.
- Agile also promotes continuous improvement and encourages the team to constantly refine and improve the program. In the stock market, this means continuously updating the program with new data and market conditions to improve its predictions over time.

The planning phase of development will cover requirements gathering. During design and development, the team will begin creating models, algorithms, and other components of the program. They will also test the program to ensure it is functioning as intended and make any necessary adjustments. In the testing and feedback phase, the team will test the program and gather feedback from stakeholders and users. This feedback will be used to identify any other issues or areas for improvement in the program. The program will then be deployed and made available to users during deployment. Ongoing maintenance and support will also be provided for the program, including updating it with new data and market conditions as needed.

Funding Requirements

Application development	x2 developers @ $4,000/mo
Hardware	Up to $1,500 for desktop computers
Software/ maintenance	Free IDE and libraries $100/mo for more sophisticated API's down the road. $100/mo maintenance
Total	$9,500 startup cost Additional $200 per month added software and maintenance

Outside of the free software to get the project started, the developers will be paid a flat rate fee of $4,000 for each month of work. The timeline of the project will be only one month so it will be a one-time payment. The only other primary startup costs will be hardware for the developers to use and host the application when finished which will leave a total project cost of $9,500. For continued use and deployment, $200 per month will need to be added.

Data Precautions

There are no data precautions as our data is collected from Kaggle which is a public domain for open use.

Developer's Expertise

The lead developer on this application has had several years of experience in the field of software development. They hold a degree in computer science obtained from western governors university as well as special certifications such as the CompTIA Project+ and ITIL foundations certification which prove that they will have strong leadership skills. Several other projects have been completed by this developer which have received excellent feedback relating to data management and user applications. These qualifications will benefit our application as it will

deal with a lot of data and will need to have a well-designed user interface for employees to interact with. This developer is a great candidate for the job.

Part B: Project Proposal

Problem Statement

Connected Enterprises' current business configuration limits their overall flexibility and ability to make accurate predictions promptly. Currently, excessive hours are used analyzing company records which slows down the process of making informed business decisions. Successful implementation of this application will increase business efficiency overall. These benefits include reduced labor costs, more time for other activities, and automated data management.

Customer Summary

This program is intended for individuals or organizations with an interest in the stock market, such as investors, traders, and financial analysts. They would use the program to make informed decisions about buying and selling stocks by predicting future stock prices based on historical data and other market indicators. The program would provide them with valuable insights and predictions that can be used to make strategic investments and maximize returns. When implemented, this application will improve Connected Enterprises' efficiency by also being automated, saving time and money.

Existing System Analysis

When it comes to stock price calculations, Connected Enterprises currently uses a combination of the following techniques:

- Technical analysis: This involves using charts and other tools to analyze historical stock prices, volume, and other market indicators to identify patterns and make predictions about future prices.

- Economic indicators: This involves analyzing economic indicators such as GDP, inflation rate, interest rate, and other economic data to predict stock prices.
- Financial news and market sentiment analysis: This involves analyzing news articles, social media posts, and other sources of information to gain insight into the general sentiment of the market and how it may impact stock prices.

When looking deeper at these techniques, it can be seen that they take a more labor-intensive approach to data analysis. This is the primary reason why this project will be taken on. Upon successful completion of the project, the application will be deployed on the company's servers. Python 3.9 will be installed with the following libraries: TensorFlow, matplotlib, pandas, and numpy. If the implementation is successful, this will improve outcomes for all involved stakeholders.

Data

The data used by this application will be formatted in the .csv structure of stock data downloaded from Kaggle. This means any data to be added to the application will need to be in .csv format to be processed. Throughout the lifecycle of the project, the data will be a part of several different steps:

- Data collection: This is the first step in the process and involves gathering data from a variety of sources. In this case, the data will come from kaggle.com and be in .csv format
- Data cleaning and preprocessing: After the data has been collected, it needs to be cleaned and pre-processed. This includes removing any missing or irrelevant data and handling outliers.

- Data preparation: During development, preparation will take place to get the data ready for the machine learning model. This includes feature extraction, normalization, and feature scaling.
- Data storage and management: The data will be stored in a secure and easily accessible format. It will also be regularly backed up and versioned to ensure that it can be easily restored in case of any issues.
- Maintenance: The data will be regularly updated to reflect the latest market conditions as well as monitored to ensure that it continues to be accurate and relevant.

Project Methodology

The methodology used for the development of this application will be agile. When using the Agile methodology to develop our stock price prediction algorithm, the process will involve a series of iterations, or sprints, where the development team will work to deliver small, incremental improvements to the algorithm until completed. Below is a detailed report on how each phase will be carried out.

- Discovery and planning: In this phase, the team will gather information about the project, including the requirements and goals of the stock price prediction program. They will also identify any potential risks or challenges that may arise during the development process.
- Design and development: In this phase, the team will begin designing and developing the program. This may include creating models, algorithms, and other components of the program. They will also test the program to ensure it is functioning correctly and make any necessary adjustments.

- Testing and feedback: In this phase, the team will test the program and gather feedback from stakeholders and users. This feedback will be used to identify any issues or areas for improvement in the program.
- Deployment and maintenance: In this final phase, the program will be deployed and made available to users. The team will also provide ongoing maintenance and support for the program, including updating it with new data and market conditions as needed.

Upon completion of the project, the development team will review the project and reflect on what worked well, what didn't work well, and what could be improved for future projects. This feedback will be used to inform future development cycles and improve the overall process.

Project Outcomes

When the project is completed, it will be deployed to the company network. The application will provide a simple command line interface for selecting the desired stocks to view. All training data will be included along with a user guide on how to install and run the application on any windows device. Finally, a milestone documentation report will be provided with all dates to review the entire process.

Implementation Plan

Implementation of the application will be pretty straightforward. The IDE will be installed on all relevant hardware and development will begin. The optimal model will be selected and data will be processed to be used by it. The selected model will then be developed and trained using the collected data and added libraries. In our case, the selected model is Long Short-Term Memory (LTSM). Once the program has been developed, it will be tested and

validated to ensure it is functioning as intended. This includes testing the program on a set of historical data and comparing the predicted prices to the actual prices. After testing and validating, it will be deployed and made available to users. The program will also require ongoing maintenance and support, including updating it with new data and market conditions as needed.

To summarize, the primary phases of the rollout include:

- Data collection and preparation
- Model selection and development
- Testing and validation
- Deployment and maintenance

The completion of the application will rely heavily on the following dependencies:

- Availability of historical stock prices and market conditions. Without data sets to be used, the program will not be able to make accurate predictions.
- Access to relevant news and financial statements to identify market trends to incorporate into the prediction model.
- A team with knowledge in data science and machine learning to efficiently develop the application.

Details for testing and distribution:

The program will be tested on a set of historical data, and the predictions will be compared to actual prices to determine the accuracy of the program. Regular testing will take place over time to ensure the program is still accurate with the latest market conditions.
The completed application will be distributed to users via download through the company's local network and will be continuously updated with the latest market data to improve its predictions over time.

Evaluation Plan

Functional acceptance testing will be one of the primary verification methods used at each stage of development. This involves reviewing how well the program performs when running predictions and how intuitive it is to run by users. During the model selection process, the team will verify that the chosen model is appropriate for the task and that it has been trained and optimized correctly. Multiple testing methods will be conducted to verify that the program is functioning as intended. This may include testing the program on a set of historical data and comparing the predicted prices to the actual prices, as well as testing the program on new data and comparing the predictions to the market prices.

Upon completion of the project, the validation methods will be directed toward the accuracy of the predictions and final functionality. Holdout validation is a common verification method that splits the data into two parts, a training set and a test set. The program is trained on the training set and then the predictions are compared to the actual prices on the test set. This method allows stakeholders to see how well the model generalizes to new unseen data. Our second primary prediction validation method will be backtesting. This method involves testing a predictive model on historical data and comparing the model's predictions to the actual outcomes

that occurred. This method allows for the evaluation of the model's performance over a set period of time, providing an understanding of how well the model would have performed if it had been used in the past. A final verification test has been added directly into the application which will be displayed to the user after each calculation. This is the root mean squared error calculation which indicates how close the final predicted closing prices were to the actual closing prices. The closer the RMS value is to zero, the more accurate the predictions are.

Resources and Costs

For this proposal, there will be two primary developers hired to design the application. These developers will be paid a flat rate per month of $4000 each for the one-month development schedule. The desktop computers needed for the developers will cost up to $1,500 for sufficient hardware. The main software used for development will be free up until the testing phase is completed as an additional $100 per month will be needed when the application goes live to gather stock data for other companies outside of the network. Maintenance will be conducted monthly at $100 per month as well to keep data up to date and continue model optimization. This will set the total project startup cost to $9,500 with an additional $200 per month for continued use and maintenance.

Tabulated view:

Application development	x2 developers @ $4,000/mo
Hardware	Up to $1,500 for desktop computers
Software	Free IDE and libraries $100/mo for more sophisticated API’s down the road.

Maintenance	$100 data updates/model optimization
Total	$9,500 startup cost Additional $100 per month added software later.

Timeline and Milestones

2/1/23 – 2/13/23 - The proposal is accepted and model design begins. Data sets on stocks are collected.

2-14-23 – 2/21/23 - A mockup concept of the program is presented. Data is filtered and configured to be used by the application. Small scale version of the project.

2/22/23 – 3/7/23 - Project is up and running with limited functionality. Debugging process begins. Data can be processed by the application and goes through formal review.

3/8/23 – 3/30/23 - The application is complete and functional. Testing begins. The program will be able to use the given data sets to make accurate stock predictions. If not accurate at first it will be able to learn and self-adjust to make more precise predictions.

4/1/23 – 4/15/23 - Program completed and deployed on company network for business application.

Sprint	Start	End	Tasks
1	2/1/23	2/13/23	Collect relevant data Design prediction model
2	2-14-23	2/21/23	Present mockup of program Configure filtered data

3	2/22/23	3/7/23	Testing/debugging phase Implement data learning
4	3/8/23	3/30/23	Final testing/Adjustments to data
5	4/1/23	4/15/23	Deployment to company network

Part C: Application

Application Files:

- Main.py - Main file for running application. Contains predictive model and user interface
- Amazon.csv - Stock data file
- Apple.csv - Stock data file
- Facebook.csv - Stock data file
- Netflix.csv - Stock data file
- Google.csv - Stock data file

Part D: Post-implementation Report

A Business (or Organization) Vision

The purpose of the stock price prediction program was to provide Connected Enterprises with a useful tool that offered a flexible, scalable, and efficient predictive system. The application was developed to predict stock trends based on historical stock data and visualize the output. The previous predictive configuration implemented by Connected Enterprises was not satisfying the business goals and objectives which called for this software upgrade. Upon implementation and testing, the proposed stock price prediction program was able to solve all previous problems and meet all business requirements. Such requirements include reducing the amount of manual labor to generate the same predictions while also automating the entire prediction process.

Datasets

The raw data used in the prediction application was downloaded from the public domain Kaggle.com. These files were in .CSV format which was readily readable by the PyCharm IDE allowing for minimal processing. The main column used in the application was the “Close” column so the program needed to have column reading functionality. Below is an illustration of

the raw data directly from Kaggle compared to the same data imported into the PyCharm IDE.

Amazon

Date	Open	High	Low	Close
1Jan15 – 28Dec21	286 – 3.74k	291 – 3.77k	285 – 3.7k	287 – 3.73k
2015-01-02	312.579987	314.750000	306.959991	308.519989
2015-01-05	307.010010	308.380005	300.850006	302.190002
2015-01-06	302.239990	303.000000	292.380005	295.290009
2015-01-07	297.500000	301.279999	295.329987	298.420013
2015-01-08	300.320007	303.140015	296.109985	300.459991
2015-01-09	301.480011	302.869995	296.679993	296.929993
2015-01-12	297.559998	298.510010	289.279999	291.410004
2015-01-13	297.480011	301.500000	293.230011	294.739990

```
1  Date,Open,High,Low,Close,Adj Close,Volume
2  2015-01-02,312.579987,314.750000,306.959991,308.519989,308.519989,2783200
3  2015-01-05,307.010010,308.380005,300.850006,302.190002,302.190002,2774200
4  2015-01-06,302.239990,303.000000,292.380005,295.290009,295.290009,3519000
5  2015-01-07,297.500000,301.279999,295.329987,298.420013,298.420013,2640300
6  2015-01-08,300.320007,303.140015,296.109985,300.459991,300.459991,3088400
7  2015-01-09,301.480011,302.869995,296.679993,296.929993,296.929993,2592400
```

Data Product Code

Our prediction application features both descriptive and non-descriptive methods of portraying information. For processing the data, training variables are implemented to parse the datasets and filter based on closing prices. The image shown below depicts the data being filtered and configured into the training set.

```
# GET CLOSING DATA TO USE WITH MODEL
main_data = df.filter(['Close'])
data_set = main_data.values
training_set = math.ceil(len(data_set) * Size)
# print(training_set)
```

Once the data set and training set are formed, the application will then move forward to creating the training model to base the prediction model. The amount of training data used can be selected by the user along with how long they would like the model to train before making the prediction. Once these parameters are selected, the program will build the long short-term memory model and calculate the predictions. Below is a snapshot of how this works in the code.

```
# CREATE TRAINING SET
training_data = data_scaled[0:training_set, :]
x_train = []
y_train = []

for i in range(60, len(training_data)):
    x_train.append(training_data[i - 60:i, 0])
    y_train.append(training_data[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
# print(x_train.shape)

# BUILD LONG SHORT TERM MEMORY (LSTM) MODEL
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

# MODEL COMPILATION
model.compile(optimizer='adam', loss='mean_squared_error')
```

When the predictions are made, the root mean squared error is displayed as the primary descriptive method along with scatter, line, and bar charts to help the user visualize the data. For non-descriptive methods, the program will display a comprehensive line graph showing the user what range of data was used for training, the predicted stock values, and the actual stock prices for the same time frame which is appropriate in this case as it allows the user to make informed investment decisions. These methods were tested many times to ensure that the model was valid and consistent with the primary data. Below is a small snippet of code showing how the plots work.

```
# MAKE PLOTS
train_plot = main_data[0:training_set]
validation_plot = main_data[training_set:]
validation_plot['Prediction'] = prediction

plt.figure(figsize=(16, 8))
plt.title(f"{company[int(number)]} prediction model")
plt.xlabel('Date')
plt.ylabel('Close Price US ($)')
plt.plot(train_plot['Close'])
plt.plot(validation_plot[['Close', 'Prediction']])
plt.legend(['Training', 'Actual', 'Prediction'], loc='lower right')
plt.show()
```

Objective (or Hypothesis) Verification

The objective of the project was to use historical stock prices and other relevant data to develop a machine learning model that can accurately predict future stock prices with a high degree of accuracy. This would in turn solve the business need for more man-hours as parsing and filtering data became automatic and helped the company make better business decisions.

Confirmation of business objectives was verified directly by Connected Enterprises as they were able to perform many predictions efficiently. While also incorporating the root mean squared calculation in the results, the company was also able to verify that the predictions and trends were relatively accurate.

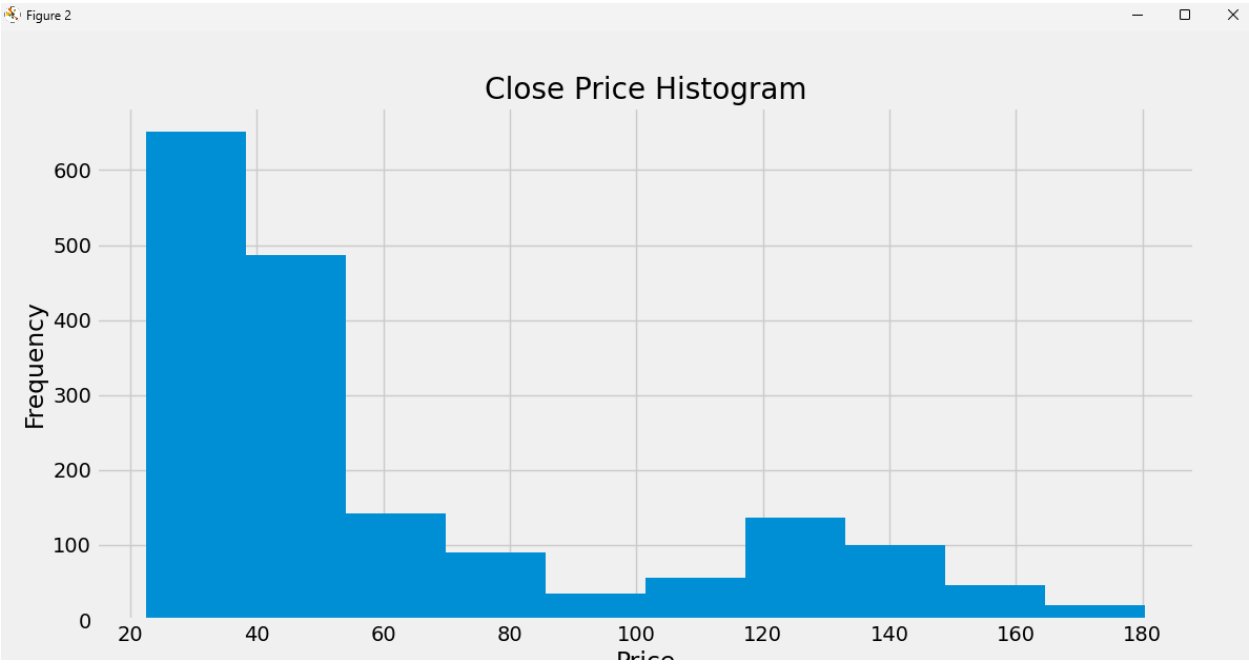
Effective Visualization and Reporting

With multiple descriptive visualizations accessible to the user, they are better able to make sense of the non-descriptive prediction graphs. The PyCharm IDE has several different plot functions available for use using the matplotlib library. Below are some examples of the descriptive method graphs output by the application.

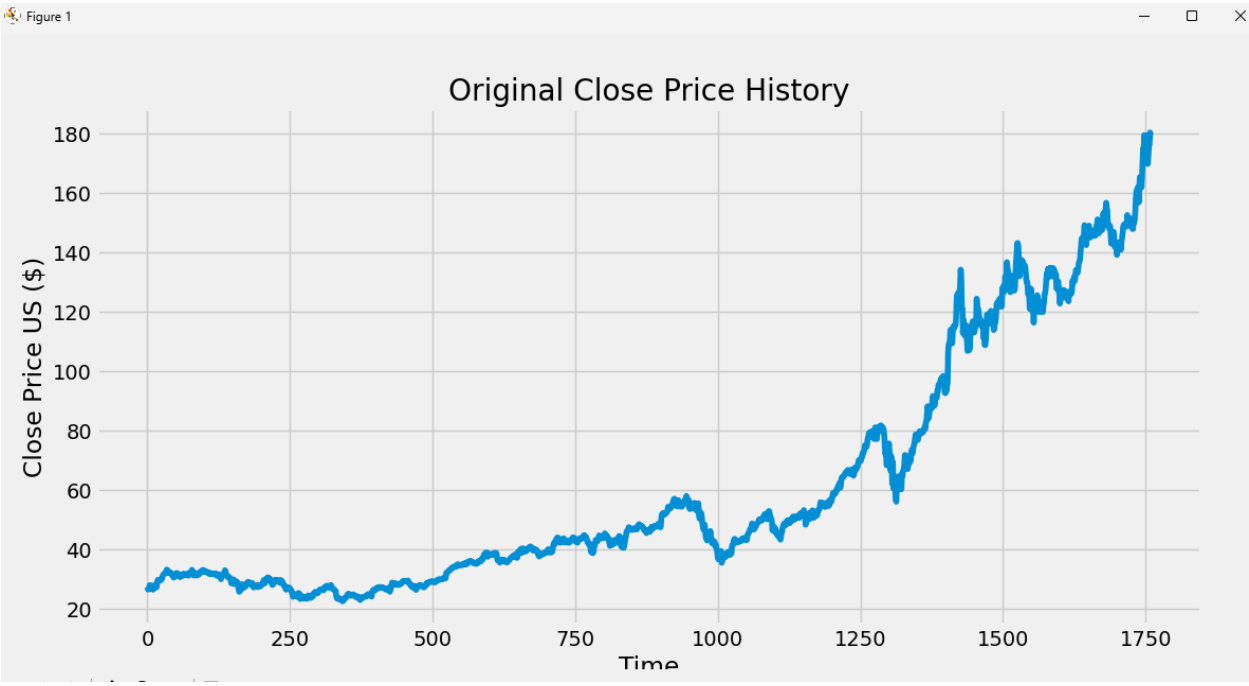
Root Mean Square descriptor to see how accurate a prediction was:

```
Epoch 1/3
391/391 [==============================] - 5s 9ms/step - loss: 2.8901e-04
Epoch 2/3
391/391 [==============================] - 3s 9ms/step - loss: 9.8559e-05
Epoch 3/3
391/391 [==============================] - 3s 9ms/step - loss: 8.0767e-05
17/17 [==============================] - 0s 5ms/step
The Root Mean Squared value is: 2.2676981386834343. The closer the value is to 0, the more accurate the prediction was.
```

Closing price histogram displaying the distribution of closing prices and their frequency:

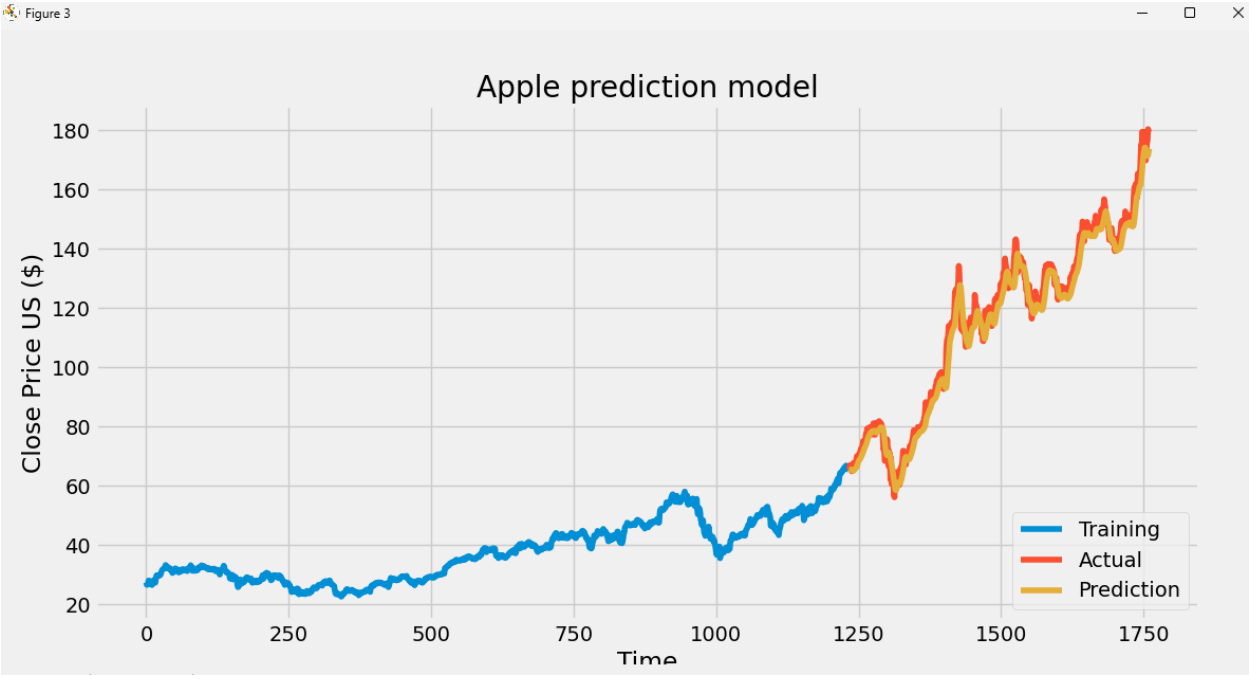


Graph of main closing prices for the user to see how the stock has progressed over time:



The descriptive visualizations support our prediction model results as they help the user verify the prediction and get a better grasp of what they are looking at. They also help the

development team verify the accuracy of each prediction and make modifications to the model to yield better results. Below are two examples of non-descriptive methods used.



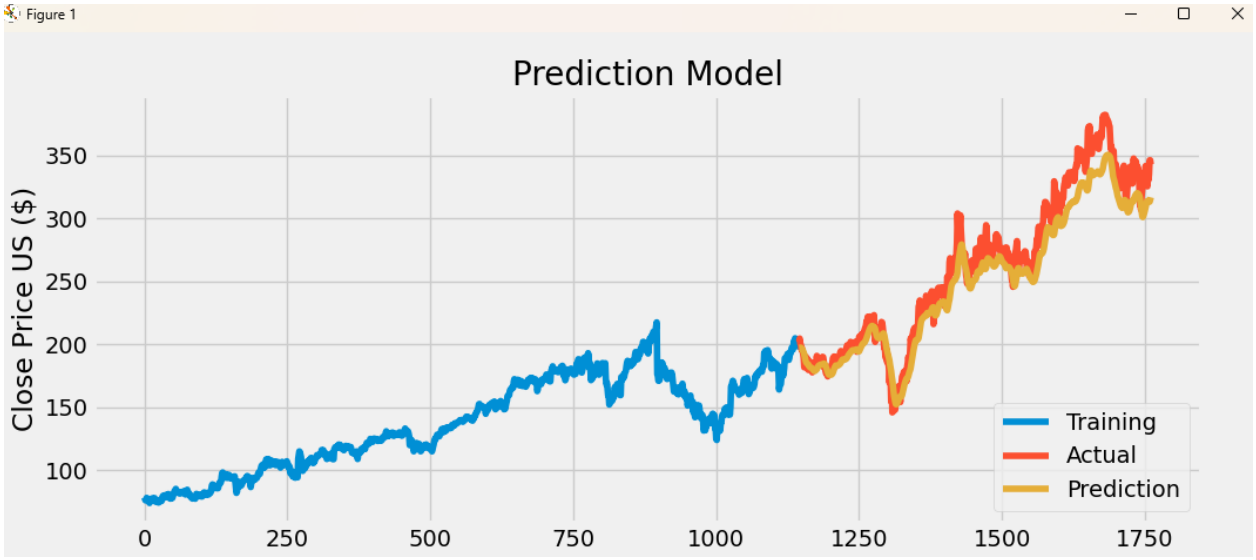

```
Below is a printed version of the prediction graph to see the prediction vs. actual results.

      Prediction       Close
1374  236.004059  235.940002
1375  236.572250  238.789993
1376  237.360748  239.220001
1377  238.238663  242.240005
1378  239.335922  234.020004
...          ...         ...
1756  330.156555  330.450012
1757  330.277771  335.239990
1758  330.630402  346.179993
1759  331.878143  346.220001
1760  333.448120  342.940002

[387 rows x 2 columns]
```

Accuracy Analysis

The primary metric(also descriptive model) used to measure the accuracy of predictions was the Root Mean Squared error metric. Root mean squared is a statistical standard for calculating the difference between the predicted closing values and actual closing values. It should be noted that the longer the application was allowed to train, the better the results were over time. A root mean square value of zero indicates 100 percent accuracy. Below is an image of the output after a test scenario with Facebook:

```
1085/1085 [==============================] - 12s 9ms/step - loss: 0.0010
20/20 [==============================] - 1s 5ms/step
Root Mean Squared: 4.0408842155301965
```

As for the non-descriptive method, a stock chart was generated showing the predictions made by the application along with the training set used and the actual historical prices. Below is the results of the same test case with Facebook:



Through a visual inspection, it was observed that the predicted trendline in yellow was very accurate in the backtesting validation which is exactly what the objective called for. With this implementation, users will be able to see and justify investments in any stock.

Application Testing

Using the agile methodology along with top down development, the application was able to be continuously tested as new modules of code were added. This ensured that the output of each code block being inspected met the expected functionality requirements. Unit and system tests were continuously conducted as development progressed. These tests included:

- Model training tests: This involved testing the training functions of the model to ensure that the model is being trained correctly. This includes testing for the appropriate selection of the model's parameters and the expected accuracy.
- Prediction tests: It is important to test the prediction functions of the program to ensure that the predictions are generated correctly.
- Integration tests: This involved testing the integration of different components of the program, such as the data preparation, model training, and prediction functions, to ensure that they are working together as intended.

The testing results were used to improve the application by identifying areas where the program is performing well or poorly and making adjustments accordingly. All modifications made were followed up with regression testing to be sure no previous code became faulty in the process.

Application Files

The complete application was built and tested with Python 3.9 in the Pycharm community edition IDE.

Below is the comprehensive list of all files used in the application:

main/

Main.py - Main file for running application. Contains predictive model and user interface

Csv_items folder/

Amazon.csv - Stock data file

Apple.csv - Stock data file

Facebook.csv - Stock data file

Netflix.csv - Stock data file

Google.csv - Stock data file

Documentation/proposal - Letter of transmittal and application proposal

In order to successfully run the application, the following libraries will also need to be installed through the PyCharm IDE for the predictive model.

Math

Csv

Keras

Matplotlib

Pandas

Numpy

Sklearn

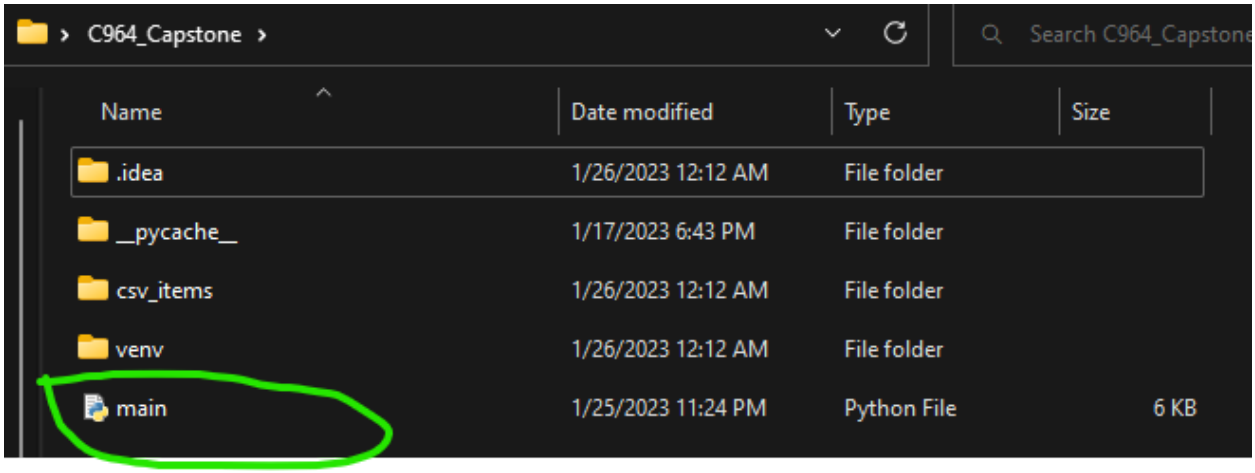
Once all files and libraries are installed, the program can be compiled and run through the IDE directly.

User Guide

Installation and execution steps are as follows:

1. Prerequisites include installing the latest version of the PyCharm IDE and downloading the submitted application files.
2. The downloaded files will then need to be imported/opened in the IDE.
3. Once the main.py file is loaded, The import tags will be underlined in red (lines 3-9). This is normal as that is where the libraries will be installed.
4. Hover your mouse over the red underlined import, a dialog will pop up giving you the option to install the package. Do this for each package. A progress bar will display in the lower right corner of the IDE showing the progress of each package downloaded.
5. After all libraries are installed, verify that the csv_items folder contains the five stock test files for Amazon, Apple, Facebook, Netflix, and Google.

6. Upon completion of steps 1-5, the application will be ready to run. In the top right corner of the IDE, click on the green play button to run the application and make a prediction.
7. OPTIONAL** when all libraries are installed and the application is ready, the PyCharm can then be closed. If you would like to run the program as an application outside of the IDE, simply double click on the main.py file in the source folder where the extracted files are. This will start up the program as an .exe.



8. While running the application, the user will be prompted to select a stock they would like to make a prediction with. After selecting the stock, they will then have the option to choose how many times they would like the model to train on the existing data and how much of the data they would like to use for the prediction.
9. After selecting the desired inputs, the model will make the prediction and provide the user with stock charts and the root mean squared value to verify the accuracy of the prediction.

Summation of Learning Experience

I feel that my previous academic experience gave me a solid foundation for approaching this project. My educational experiences up to this point and my personal interests in the stock market are what inspired me to design the project the way I did. Acquiring certifications in both project management and IT leadership also helped me gain the necessary skills to understand the business side of software development and project life cycles. Also, developing several similar applications from scratch with functional user interfaces equipped me with the ability to approach this more serious program.

Aside from the course supplemental resources and webinars, stack overflow was another learning resource used to design the application. I chose to use Python in this case as it provides much more flexibility when it comes to machine learning when compared to other languages used like Java or C. I kept the overall program particularly simple by keeping it within the PyCharm IDE instead of deploying it to a website or cloud. This was done to save time due to term constraints. Although a simple approach was taken, a majority of the time was spent reading documentation and researching ways to incorporate the various library packages into the application.

This project has had a major contribution to my learning experience. Outside of several conversations with my program mentor and course instructor, this project was completed without much external assistance. Completing this project gives me new confidence in my project development skills and my overall IT qualifications. It has also encouraged me to continue expanding my programming knowledge and take on more projects in the future.

References

No outside sources were quoted, paraphrased, or summarized.