PRODUCT DESIGN SPECIFICATION

A Purposeful Walk Down Wallstreet

Nabeel Asghar | Michael Shields | Shojib Miah | Michael Chen
Wayne State University – March 21st, 2020
Senior Capstone Project

Revision History

Date	Description	Author	Comments
2/25/2020	Version 1	Nabeel Asghar, Michael Shields, Shojib Miah, Michael Chen	Original document that we submitted to canvas
3/18/2020	Version 2	Nabeel Asghar	Deleted the hardware diagram.
4/11/2020	Version 3	Nabeel Asghar	Fixed section 2.1 Assumptions and the Database Design and Use Case Diagram

Document Approval

Printed Name	Title	Date
Michael Shields	Team Lead	3/18/2020
Nabeel Asghar	Documentation Lead	3/18/2020
Michael Chen	Development Lead	3/18/2020
Shojib Miah	Research and Development Lead	3/18/2020

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this product design specification document is to provide developers with information regarding the design of the system and its architecture. This will provide future developers the necessary information required to implement future changes and enhancements to the application. This document is created during the planning phase of the project. The intended audience for this document is primarily aimed at project managers, teams, and developers. The user interface sections can also be utilized to inform the end users about the installation of the application. This document will still primarily focus on relaying information in order to effectively define the system's architecture and design.

2. General Overview and Design Guidelines

2.1 Assumptions:

1. Tableau Visualization Software is installed.
2. Python is installed and set up.
3. All dependencies are installed.

2.2 Technical Constraints:

1. Database Management: MySQL 8.0+
2. Language: Python used for backend
3. Pandas: Structure data for analysis
4. Pandas-DataReader: Reads pandas data structures
5. FuncTools: Wraps callable object to extend it functionality and callability
6. Python statsmodels: Used for ARIMA, SVM, Polynomial Regression, and other data models
7. NumPy: Array processing for data variables
8. Math: Fundamental mathematical operations
9. Statistics: Statistical Analysis Operations
10. Python-DateUtil: Date and time module
11. TabPy: Tableau Python Client/Server module
12. Data Fetching
    a. Quandl: Fetching macro-economic variables
    b. Yahoo Finance: Fetching instrument statistics
13. SQL Engine
    a. SQL Alchemy: Database Abstraction for SQL
    b. pyMySQL: MySQL API Module

14. Testing
    a. PyTest: Testing module
    b. BackTesting: Forecast Accuracy Testing module

2.3 Business Constraints:

1. Connectivity issues with the internet
2. Database connection issues
3. Unavailable or compromised data source
4. User unfamiliar with front end tools

2.4 Design Constraints:

1. Data needs to be loaded into newly structured tables
2. Provide way for data to be displayed comprehensively
3. Must maintain data integrity, closing prices and macro-economic statistics must be validated during design and can be validated by user at any stage
4. Provide an agnostic platform for financial instruments

3. Architecture Design

3.1 Hardware Architecture

There are three main components of the hardware architecture. The first component is the local machine that the user is using for this application. The second component is the database server, which is MySQL. The last component is the external data source (YAHOO) from where the basic statistical data is being obtained from via various python libraries such as back testing and Quandl. The fourth physical component of the system is the user and their interaction with the front-end.

The end-user will interact with the Tableau dashboard that displays all the required information. Developers will interact with the Python classes in PyCharm or any other text editor and run the main file (DataMain.py) which will pull the most up-to-date data. This information is handled by the application before it is stored on the database server. In turn, Tableau will fetch the information from the database and display it for the user. The following are the hardware requirements to install Tableau, MySQL 8.0, and PyCharm Professional Edition respectively:

MySQL 8.0:

- Disk Space: 2 GB
- Memory: 4GB of RAM
- Processor Type: 64-bit x 64-compatible AMD or Intel processor •

Windows Server: Ideal (minimum requirement):

- Storage: 2 GB
- Memory: 4GB of RAM
- Network Connection: Ethernet 10/100/1000baseT (min)
- Active Directory: In use (optional)

Tableau:

- Disk Space: Minimum 5 GB (entire 5GB will not be needed)
- Memory: 4 GB RAM
- Screen Resolution: Minimum screen resolution of 1366 x 768

PyCharm Professional Edition 2019.2:

- 4 GB RAM minimum, 8 GB RAM recommended
- 1.5 GB hard disk space + at least 1 GB for caches
- 1024x768 minimum screen resolution

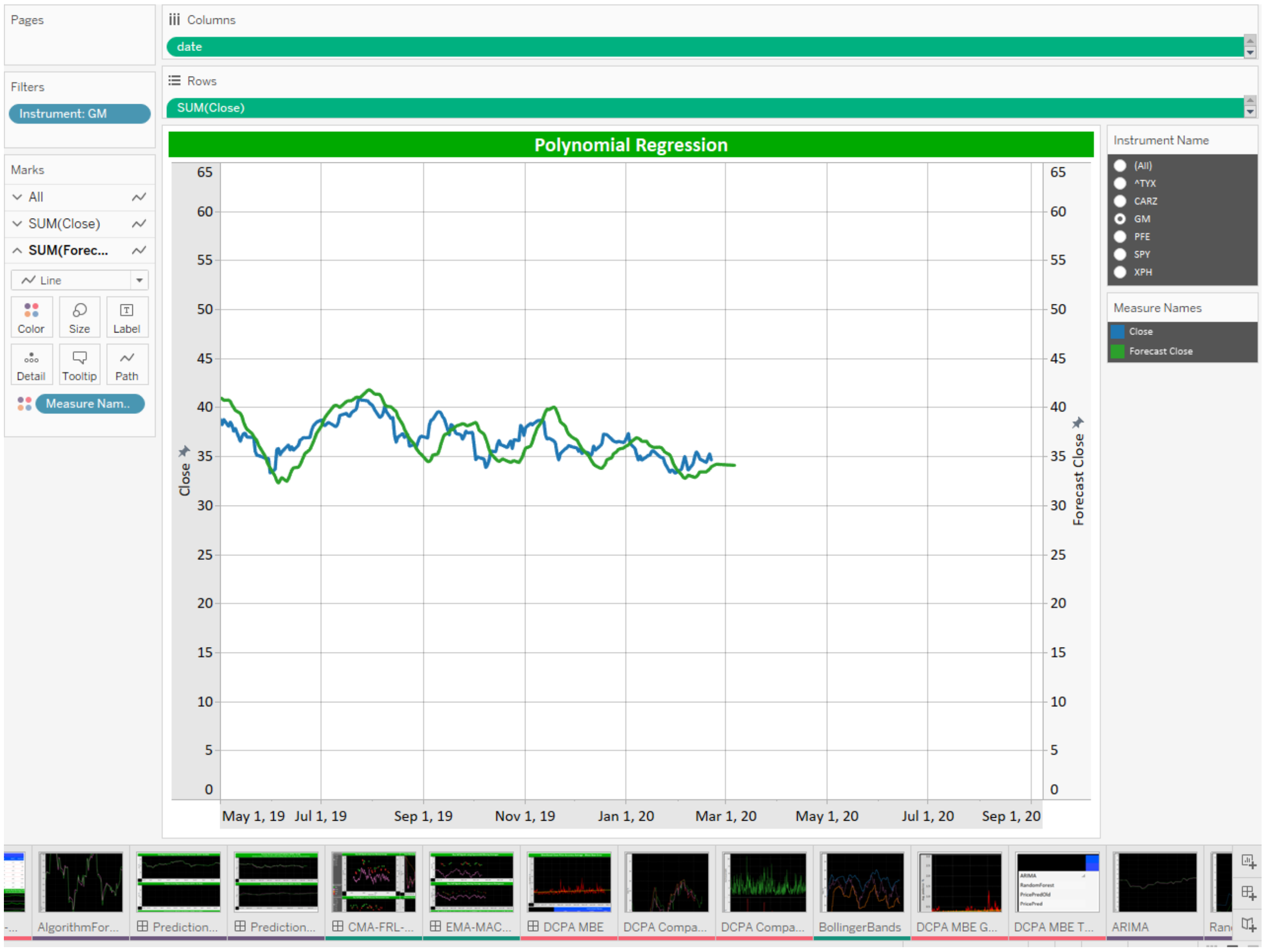
3.2 Software Architecture

The software architecture follows the structure of a front-end, middle-layer, and backend(database). The front end is Tableau, the middle layer is Python scripts, and the backend is MySQL.

Front-End:

The user interface (UI) is displayed in a Tableau Dashboard. This UI provides a detailed analysis of multiple trading algorithms and strategies. These algorithms and strategies will be acting on the GM stock, the CARZ index, the PFE stock, the XPH index, and the S&P 500 index, as well as various macroeconomic variables such as unemployment rate, inflation rate, the misery index, stock market confidence index, GDP, consumer price index, and NPGDP. More stocks and macroeconomic variables via SQL scripts into the database tables.

Tableau can run concurrently with the application, and it will automatically populate by pulling data from the MySQL database. The data retrieval and manipulation will be carried out by Python and SQL Stored Procedures.



Layout of a sheet showing polynomial regression

Middle-Layer:

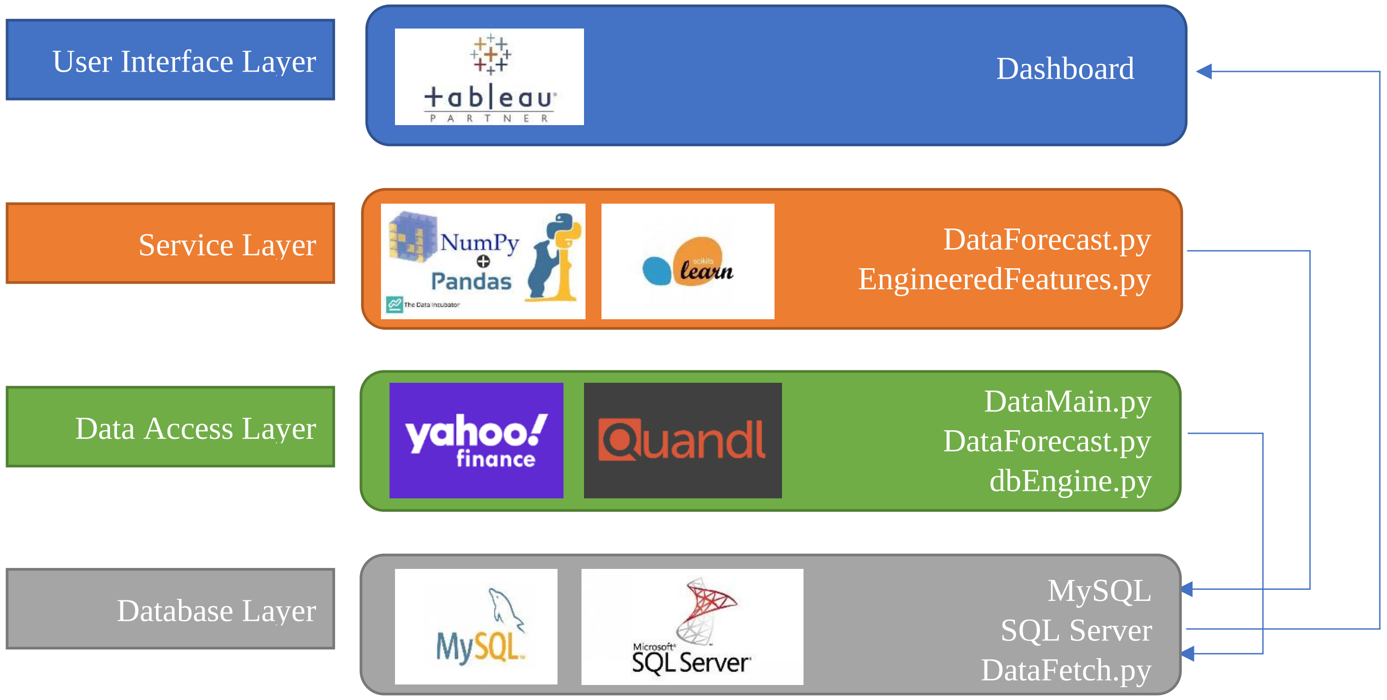
The middle layer of Python provides the logic, which includes buy/sell signaling, algorithm-based predictions, back testing algorithms and any other manipulation to the incoming stock/macroeconomic variable data statistics. It also communicates directly with the MySQL database.

Python will use the Pandas Library (pandas_datareader) to communicate with the external data source (YAHOO finance data exchange server) to pull in the stock data, usually the close price, as well as imported libraries like back testing (BT) and Quandl. Back test allows the user to test the algorithms in an efficient manner, displaying the returns and risk exposure of the algorithm. Quandl is a data scraping tool which also pulls data from YAHOO finance. This requires Python to have Anaconda (an open-source distributor) installed as well to include extra dependencies such as these that are crucial to retrieve and manipulate the data. More precisely Python will use the Pandas, Pandas-DataReader, NumPy, SqlAlchemy, StockStats, SkLearn, StatsModels, Quandl, and BT libraries.

Back-End:

MySQL will further manipulate the data as it is sent from Python, through SQL Stored Procedures. SQLAlchemy, pyODBC and, pyMySQL will orchestrate the communication between the application and the MySQL Database. The implementation of all data manipulation is executed through stored procedures and through various libraries in Python including Quandl and BT.

Tableau has the capability to run Python scripts, a Tableau-Python environment could be setup in Anaconda to facilitate that feature. If there is a need for two-way communication between the application, Tableau, and the database, the TabPy-Client library can be utilized (currently not used).

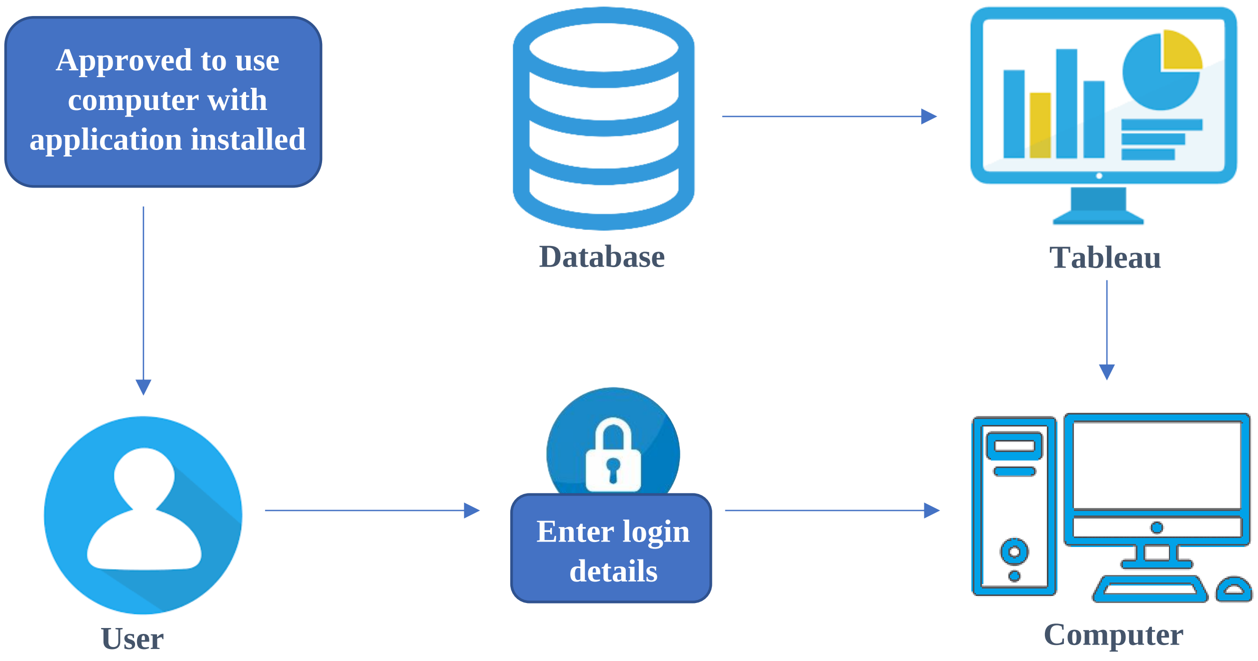


Software Layer

3.3 Security Architecture

Application security is not a direct operational responsibility because its only accessible on the machine that has all the required Software installed, scripts loaded, and proper admin privileges given. Security will not be a responsibility of the developers. The application is designed to be installed on any Windows PC but can be installed on Mac OS as well.

The application is not designed for IoT. It will only be installed and used within the intranet of a corporate department. The organization will decide which PC the application should be installed on and then anyone with access to that PC will have access to the application.



Security Architecture

3.4 Communication Architecture

The communication architecture is based on the internet connection and the database connection. The internet connection is necessary to pull the latest data from YAHOO trader via Pandas API, Quandl and BT python dependencies. When the main file is run, it is essential to have the internet connection to pull the latest data from the various financial instruments. Additionally, the database connection is necessary to store 3 years- worth of data in local database. Within this database there is storing and displaying of the data.

3.5 Performance

Tableau will not have updated information until the user runs the necessary scripts to pull the most up-to-date data. The speed of execution within Python will depend heavily on the connection to the internet and database connection, however, under normal circumstances a complete execution will take less than one minute.

The database will require enough storage to support the application. Since the historical data tables are replaced daily, the storage requirements will remain mostly static after the first execution for that day. Storage needs will be met by meeting the Hardware Requirements as defined above.

Once the main file has been run (without error), Tableau will display the various forecasts and calculations of the outlined financial instruments. The application is designed to be run independent of other instances, so it is possible to install the application on multiple user machines on the same network, or domain, and set them to run for separate financial instruments.

4. System Design

4.1 Use Cases

These are how a user can use this application and the various use-cases association with our application. The application user is a person experienced with reading charts and interpreting economic data through charts and numbers.

UC-1.0: Update Tableau to have the latest data every launch	
Actors:	Application User
Description:	In order to get the most accurate and up-to-date data, results, and predications, we must run the Python application every time we launch the application. This is to ensure that outdated numbers do not populate the database and are displayed through Tableau.
Trigger:	Triggered when user opens application.
Preconditions:	• The database was not updated
Postconditions:	• The rest of the application will have the latest data and latest algorithm results.
Normal Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb."
Alternative Flow:	1. Run DataMain.py located in FinsterTab/F2019. 2. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb."
Exceptions:	There is no need to do this refresh of data if it was already done that day or the closest trading day.
Assumptions:	• All dependencies are installed

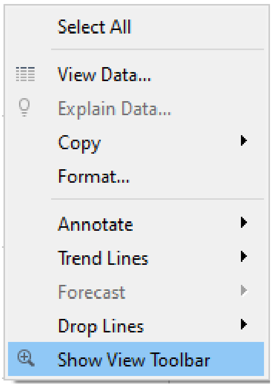
UC-2.0: All Tableau's sheets load	
Actors:	Application User
Description:	For every sheet (page) and graph, Tableau should load the sheet. For every sheet, an SQL query is executed to find all the data points that will be plotted onto the graphs. There should be no blank sheets or sheets that do not display due to any error.
Trigger:	Triggered when user clicks a sheet.
Preconditions:	• UC 1.0
Postconditions:	The application has every sheet displaying data.
Normal Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. Check every sheet to make sure that are not blank canvases.
Alternative Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. Check the sheets to make sure they all display graphs and all work, if any do not work, go to step 3. 3. Return to UC-1 to refresh the database by rerunning DataMain.py.
Exceptions:	There should be no exceptions, all sheets should be working 100% of the time.
Assumptions:	• All dependencies are installed

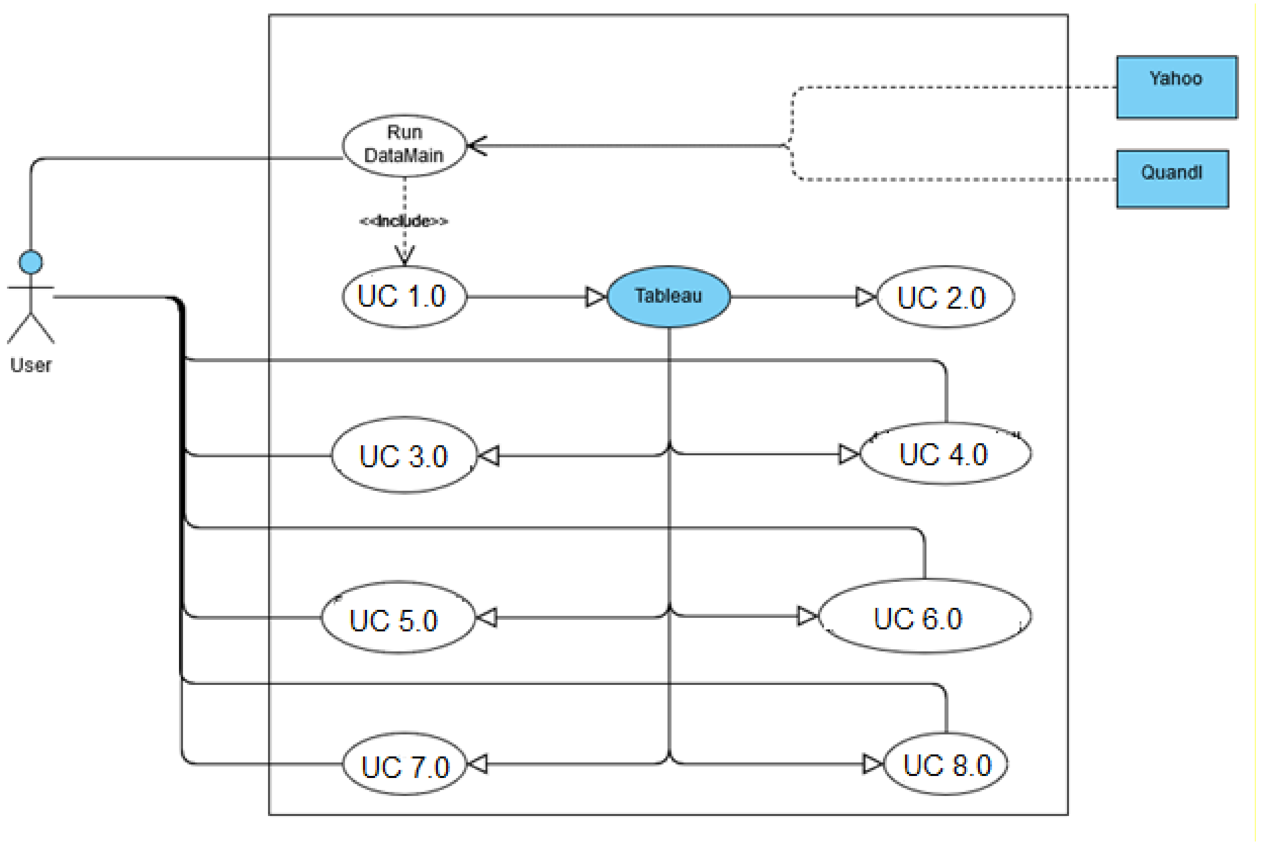
UC-3.0: Tableau displays forecasted values	
Actors:	Application User
Description:	Tableau should show values for its forecast algorithms. There should be no huge outlier in the forecast prices, the stock prices, or the portfolio values.
Trigger:	Triggered when user clicks a forecast sheet.
Preconditions:	• UC 1.0
Postconditions:	The application will show the forecasted values that were calculated using our code.
Normal Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. The forecast algorithm will show the most recent dates.
Alternative Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. Check the forecasted algorithm sheets and see if they are all showing a forecasted price. If not, go to step 3. 3. Return to UC-1 to refresh the database by rerunning DataMain.py.
Exceptions:	There should be no exception, the forecast algorithms should always be working.
Assumptions:	• All dependencies are installed

UC-4.0: Tableau displays signals for buy and sell	
Actors:	Application User
Description:	This application has several algorithms in-place to determine when to buy and when to sell a stock.
Trigger:	Triggered when the user clicks a buy/sell algorithm sheet.
Preconditions:	• UC 1.0
Postconditions:	The application will show the forecasted values that were calculated using our code.
Normal Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. Check if the buy and sell signals are present in the sheets with "Buy and Sell" in the title.
Alternative Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. Check the buy and sell sheets and see if they are all showing buy and sell signals. If not, go to step 3. 3. Return to UC-1 to refresh the database by rerunning DataMain.py.
Exceptions:	There should be no exception, the buy and sell signals should always be working.
Assumptions:	• All dependencies are installed.

UC-5.0: Tableau should display stock values	
Actors:	Application User
Description:	Tableau on any sheet when looking at forecasted values or buy and sell signals, there should always stock close values charted on Tableau. This helps the user to see how accurate our algorithms are.
Trigger:	Triggered when user clicks any forecast algorithm sheet.
Preconditions:	• UC 1.0
Postconditions:	The application will show the stock values charted and visualized in graph format.
Normal Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. Check if the stock values are show for any sheet that predicts stocks prices or buy and sell signals.
Alternative Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. Check the forecast and buy/sell sheets and see if they are all showing the stock prices. If not, go to step 3. 3. Return to UC-1 to refresh the database by rerunning DataMain.py.
Exceptions:	There should be no exception, the stock close values should always be present.
Assumptions:	• All dependencies are installed.

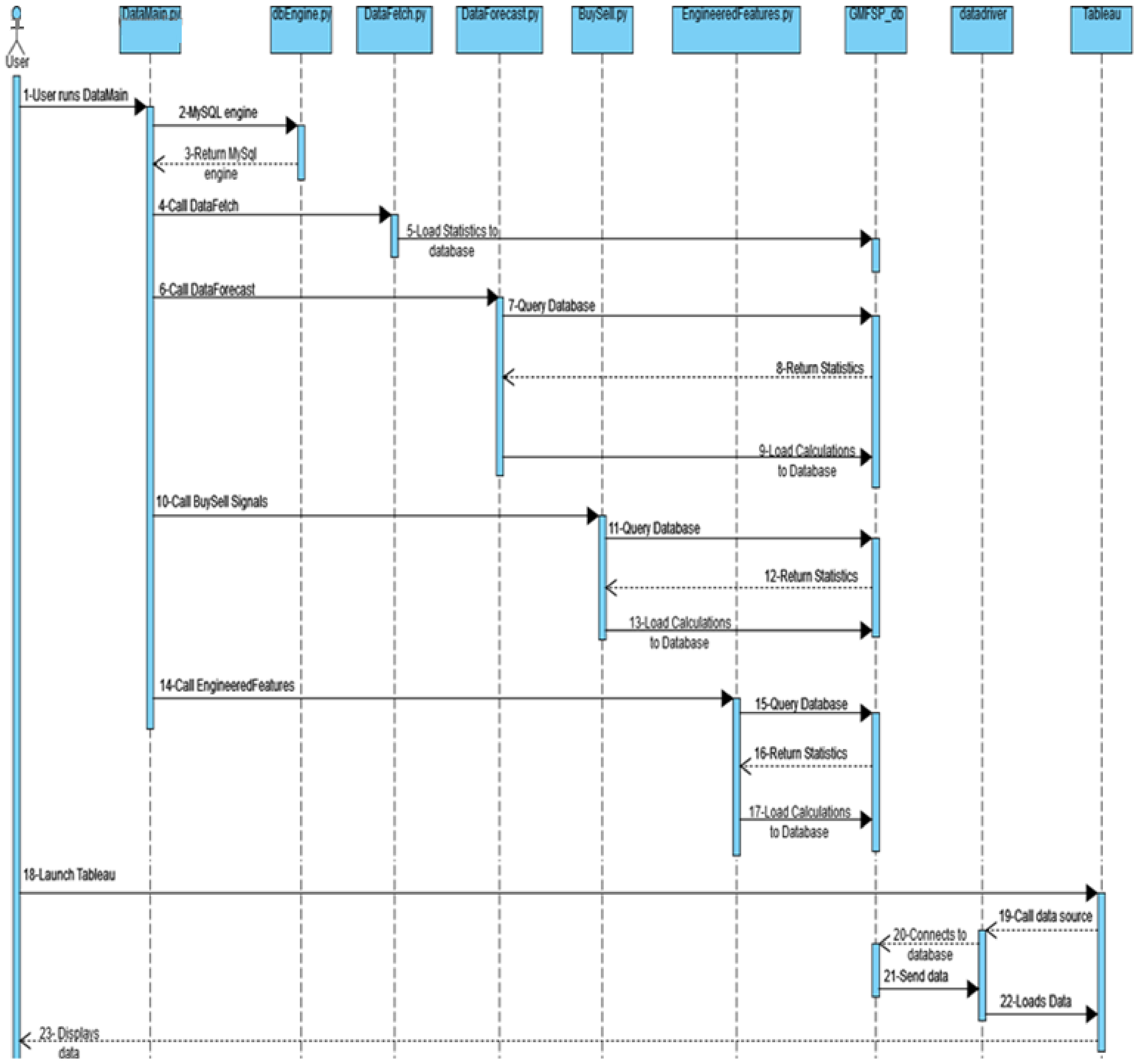
UC-6.0: Tableau should display macroeconomic variables	
Actors:	Application User
Description:	Tableau should show the macroeconomic variables: GDP, Inflation Rate, Unemployment, and Misery. They should be visualized each into graphs and together onto a dashboard.
Trigger:	Triggered when clicking a macroeconomic variables sheet.
Preconditions:	• UC 1.0
Postconditions:	The application will show the macroeconomic variables visualized in graph format.
Normal Flow:	1. Open the Tableau application in the Tableau folder called: “GM_FinTech_Application.twb.” 2. Check if the macroeconomic variables are visualized in their own sheets and combined on a dashboard.
Alternative Flow:	1. Open the Tableau application in the Tableau folder called: “GM_FinTech_Application.twb.” 2. Check the macroeconomic sheets and see if they are all showing the graphs. If not, go to step 3. 3. Return to UC-1 to refresh the database by rerunning DataMain.py.
Exceptions:	There should be no exception, the macroeconomic variables should always be present.
Assumptions:	• All dependencies are installed.

UC-7.0: Tableau zoom tool should be available	
Actors:	Application User
Description:	When viewing a graph, Tableau has a zoom tool. This allows you to zoom in or zoom out a graph to get a close picture or the bigger picture.
Trigger:	Triggered when user clicks zoom in on any graph.
Preconditions:	1. The user must have the application running with all its dependencies and libraries it requires. 2. The user must have Tableau running. 3. UC-1.0. 4. UC-2.0.
Postconditions:	The application will allow the user to zoom in or zoom out of a graph.
Normal Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. Navigate to any sheet and use the zoom toolbar to zoom in or zoom out according to the users' needs.
Alternative Flow:	1. Open the Tableau application in the Tableau folder called: "GM_FinTech_Application.twb." 2. Navigate to any sheet and use the zoom toolbar to zoom in or zoom out according to the users' needs. If not, go to step 3. 3. Right click on the graph and click "Show View Toolbar."  Select All View Data... Explain Data... Copy Format... Annotate Trend Lines Forecast Drop Lines Show View Toolbar
Exceptions:	There should be no exception, the toolbar should be there and if not, the option to display it should be there
Assumptions:	• All dependencies are installed.

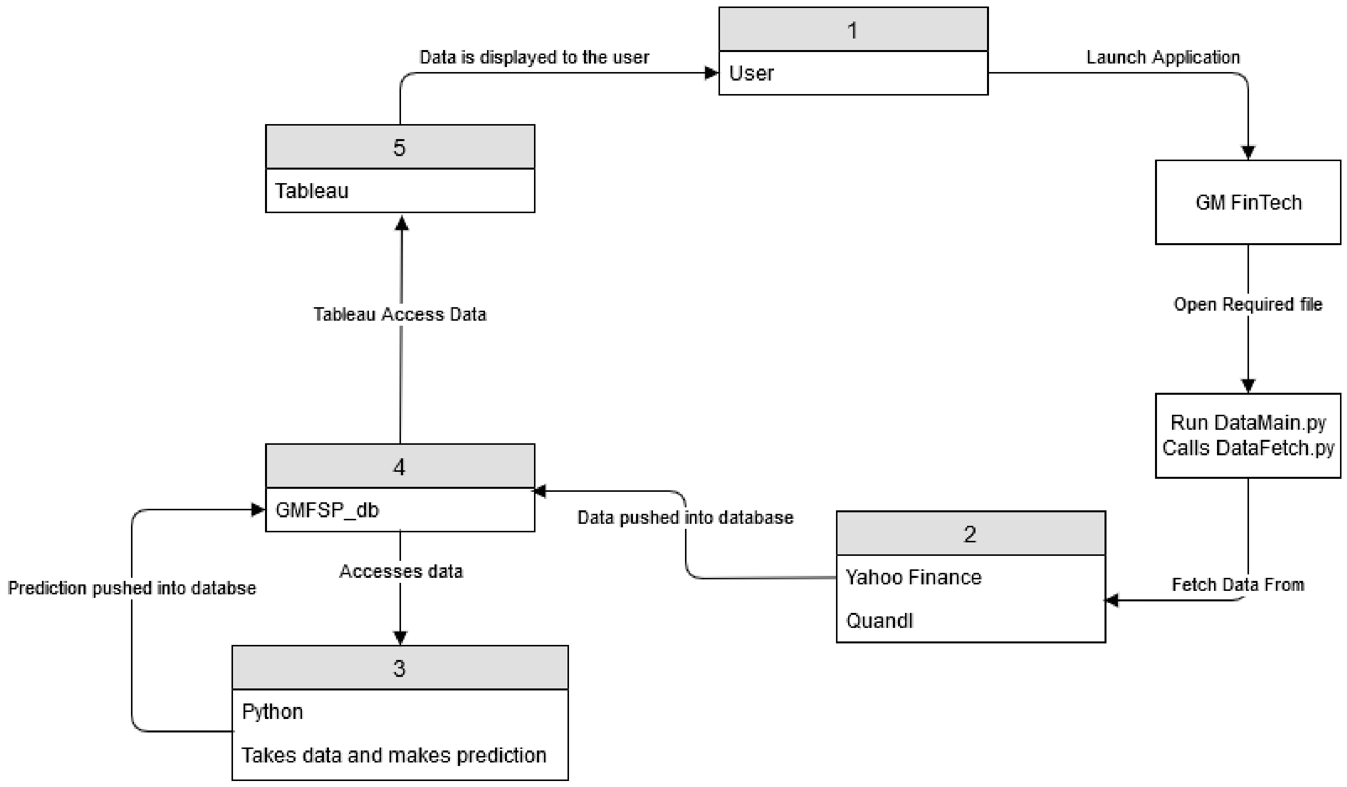


Use Case Diagram

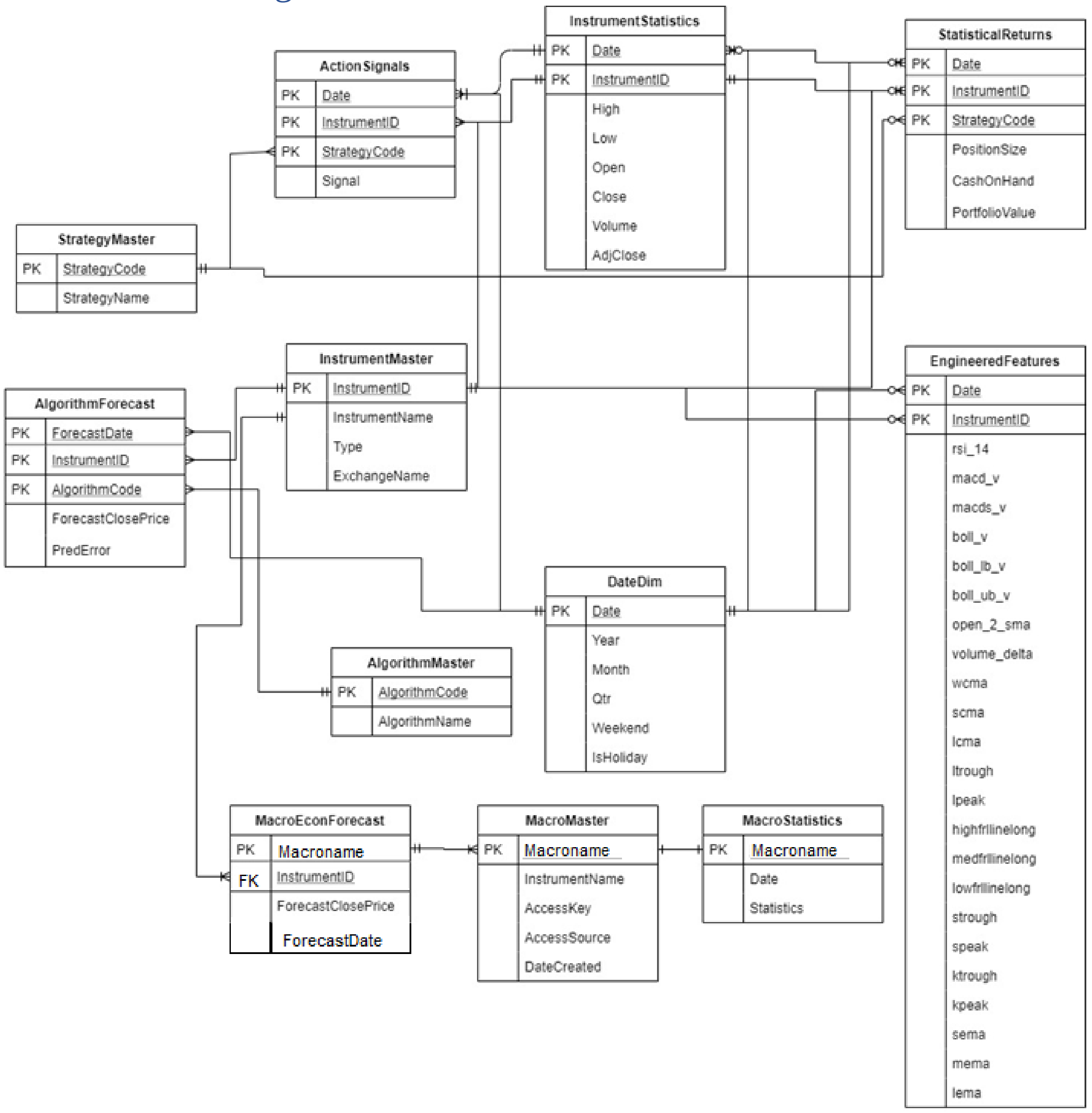
4.2 Sequence Diagram



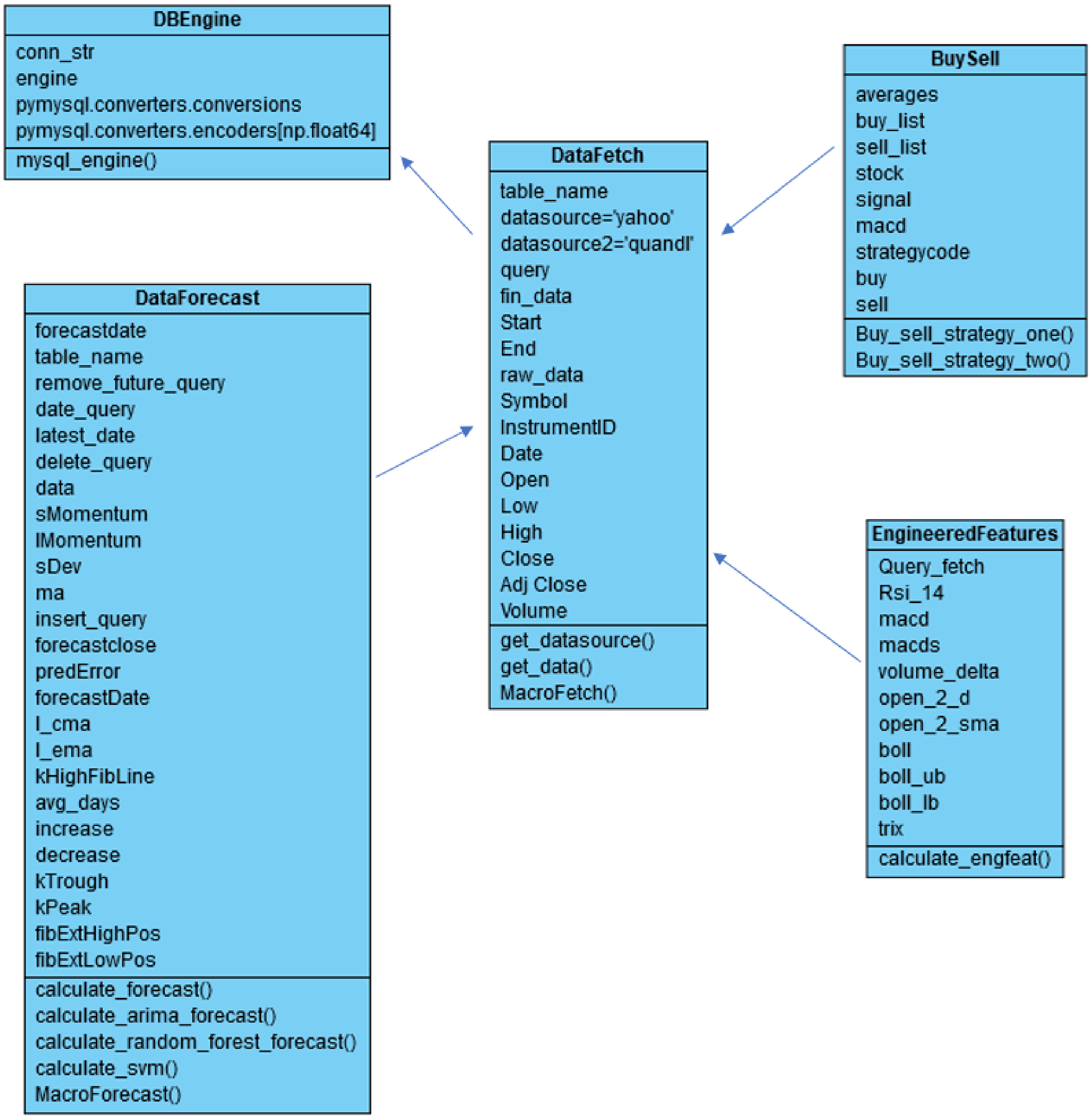
4.3 Data Flow Diagram



4.4 Database Design



4.5 Class Diagram



4.6 Application Program Interfaces

Pandas

We are using Pandas API to establish a connection to the Yahoo Finance data exchange server. Pandas documentation can be found on their website https://pandas.pydata.org/pandas-docs/stable/reference/index.html.

Yahoo Finance Data Exchange

We retrieve from the Yahoo Finance data exchange server the following: Date, High, Low, Open, Close, Volume, and Adjusted Close. This data is stored in the 'InstrumentStatistics' table in the database.

Yahoo Finance provides information only and is not intended to be used for trading or investing purposes. You will be able to find any company's stock information that Yahoo Finance covers. Yahoo Finance provides all information AS-IS and is not responsible for any inaccuracy.

Quandl

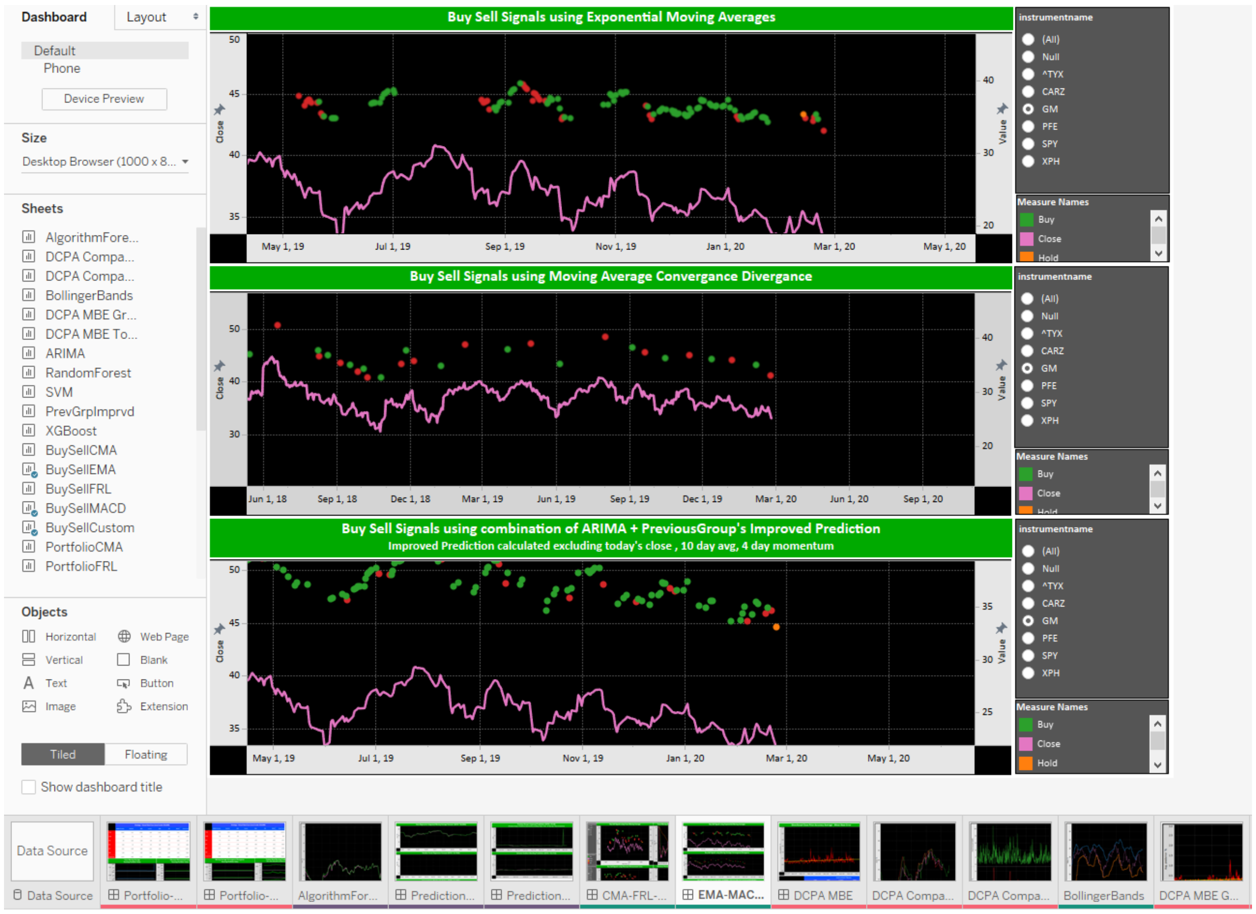
We are using Quandl API to retrieve macroeconomic data such as the GDP, unemployment rate, inflation rate, and misery index. Quandl documentation can be found on their website https://docs.quandl.com/docs/python.

SQLAlchemy

We are using SQLAlchemy to establish a connection and store data to the database. SQLAlchemy documentation can be found on their website https://www.sqlalchemy.org/.

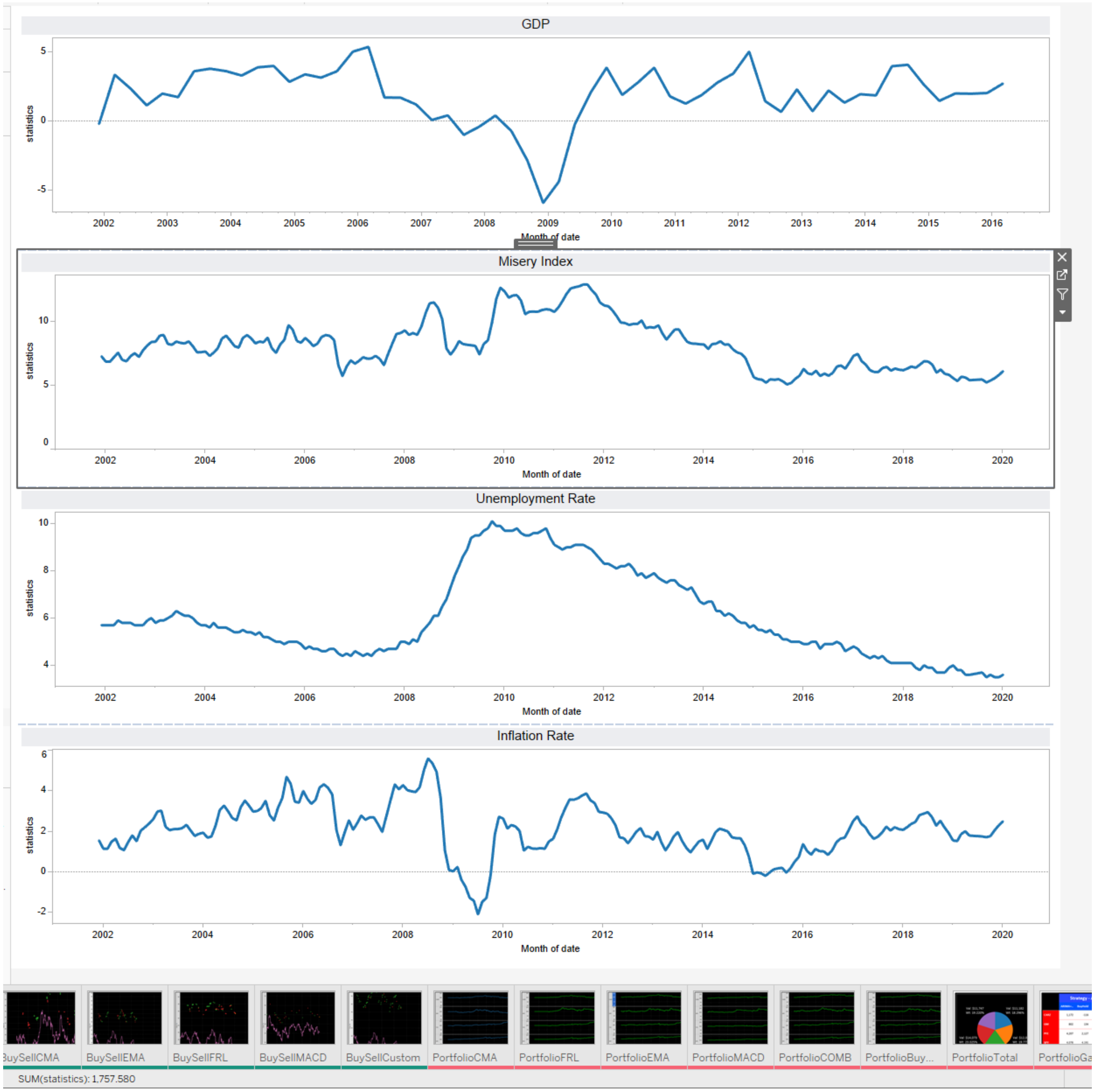
4.7 User Interface Design

The User Interface (UI) is built in Tableau. Tableau is an interactive data visualization software. The UI will be available on a local computer. Tableau will obtain Yahoo Finance information that is stored in a database and will visualize the information for the user. Users will be able to hover over data points on the graph to see specific information. Users will have the ability to zoom in and out to see datapoints in a specific range. Tableau will only be able to read information from the database but cannot make changes to the information.



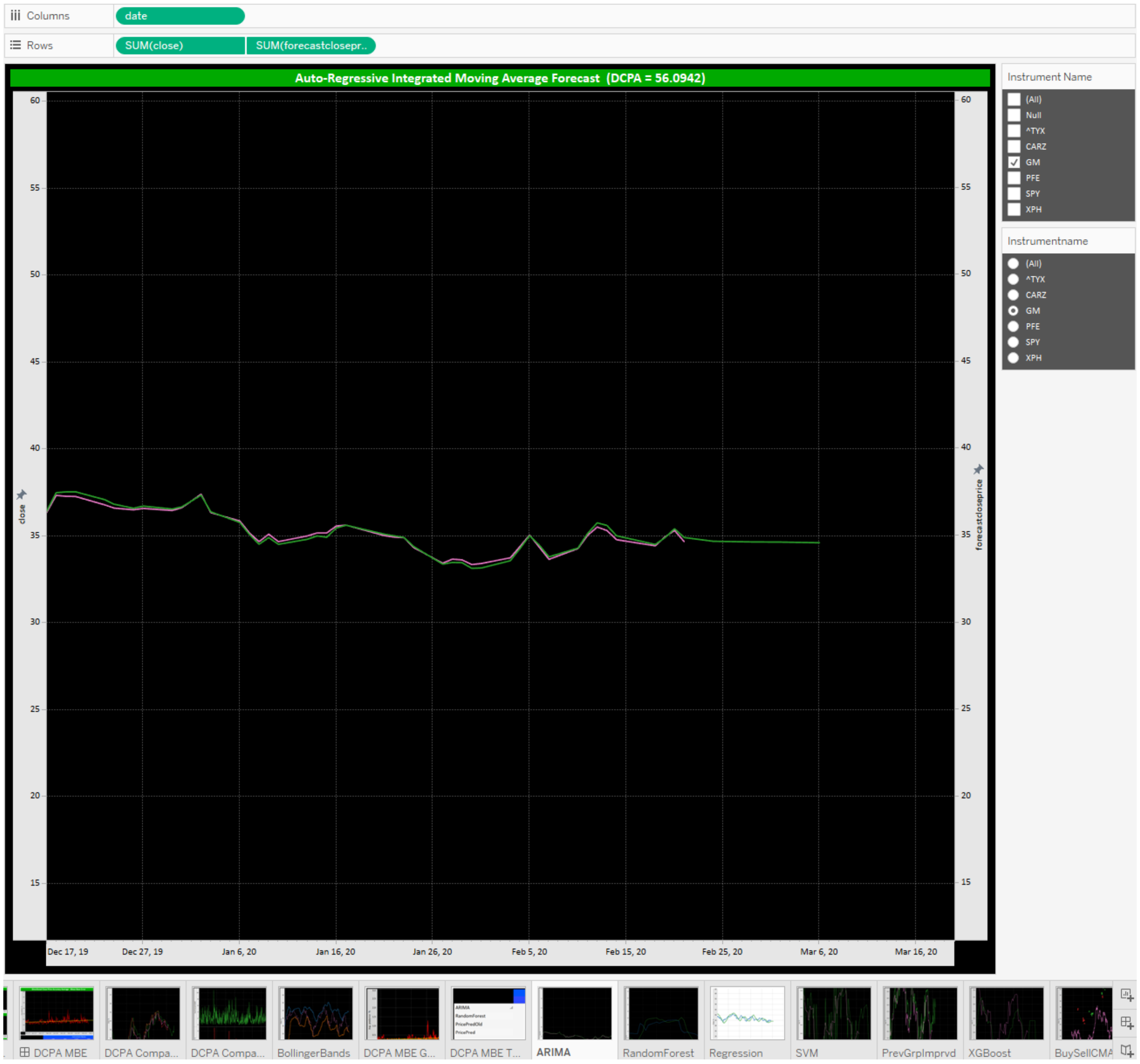
Layout of a dashboard

Users will be able to switch between graphs for different macroeconomic data at the bottom of the dashboard.



Graph of Macroeconomic Variables

Each trading strategy will have its own graph. You will be able to change what stock it will display. It will display the actual close price and the predicted close price.



Layout of trading strategy

5. Product Design Specification Approval

Appendix A: References

- pyODBC
- SQLAlchemy
- Pandas
- Pandas-datareader
- FuncTools
- Statsmodels
- Tabpy
- DateTime
- Statistics
- Math
- NumPy
- pyUse

SOFTWARE LIST
MySQL 8.0 download
https://www.mysql.com/downloads/

MySQL 8.0 documentation
https://dev.mysql.com/doc/

PyCharm free student
https://www.jetbrains.com/student

Tableau free student
https://www.tableau.com/academic/students

Appendix B: Definitions

MySQL: an open-source relational database management system we will use.

YAHOO Finance: a source of financial data from which we grab stock prices.

Quandl: an API (Application Programming Interface) that makes it easier for us to grab financial data specific to macroeconomic variables.

Tableau: a visualization tool that allows use to interact with our data in our database and visual in graphs and tables.

XGBoost: this is an open-source tool for machine learning in Python.

S&P 500: S&P (Standard and Poor) measures the value of the 500 most valuable stocks listed on the stock market in the United States.

NYSE: the NYSE (New York Stock Exchange) is a stock exchange in New York and is the largest stock exchange in the world.

ETF: an ETF (Exchange-Traded Fund) is a basket of securities which means a collection of stocks that you can trade in the NYSE.

CARZ: this is an index that tracks global auto manufacturing companies such as Ford, GM, Tesla, etc.

CMA: CMA (Cross-Moving Average) is an algorithm that detects when a shorter period moving averages intersects a longer period moving average.

EMA: EMA (Exponential-Moving Average) is a moving average that places a heavy emphasis on newer price points rather than older price points.

ARIMA: ARIMA (Auto-Regressive Integrated Moving Averages) is data-science model that allows you to predict future values by basing it off of momentum.

FRL: FRL (Fibonacci Retracement Lines) is a technical analysis tool for determining resistance and support levels.

MACD: MACD (Moving Average Convergence Divergence) is a model that tracks the relationship between two EMA lines, one shorter period and one longer period.

Random Forest Regression: this is a form of regression that is calculated using machine learning and decision trees.

SVM: SVM (Support Vector Machine) is a machine learning classifier.

GDP: Gross Domestic Product (GDP) is measure of all goods and services made by a country.

Misery Index: this is an economic indicator of how the average U.S. citizen is doing.

Adj. Close: this is the adjusted close price of stock factoring in dividends and splits.