**MIS41110**

**Programming for Analytics**

**[ Stock Analysis System ]**

***I declare that all material included in this project is the end result of my own work and that due acknowledgement has been given in the bibliography and references to all sources be they printed, electronic or personal.***

|  |
| --- |
| **Jianming Xu** |
| **20203410** |

**Submission Date  06/12/2020**



# User-manual

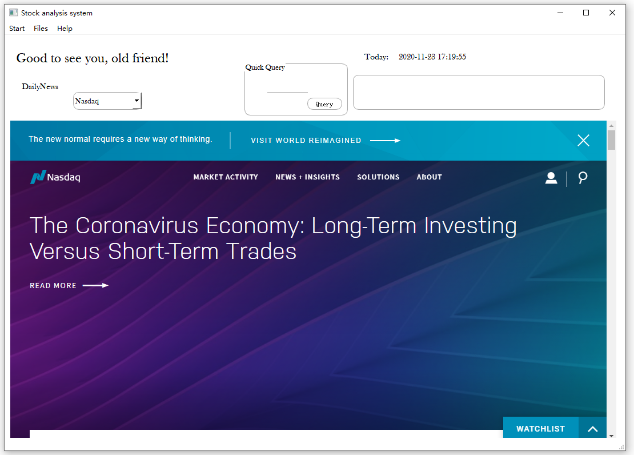
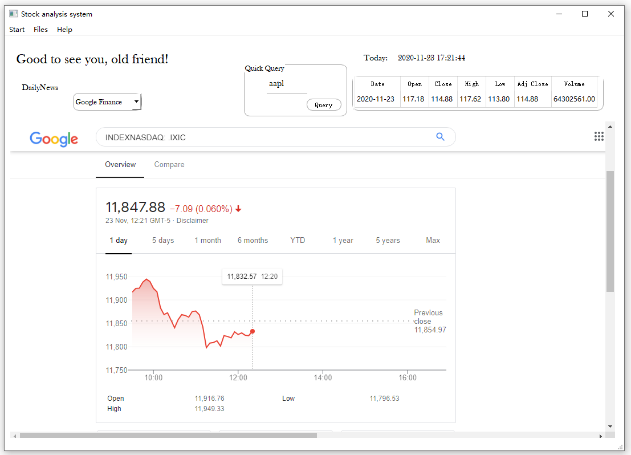
* **Introduction**

To meet the requirements of our Stock analytics in their decision-making process, this Stock Analysis system has been developed. The system provides multiple analytic tools to help analyze the history information of stocks, and perform predictions for future stock trend with Machine Learning Algorithm. It also offers visualization options to aid analysis, and useful output solutions. Additionally, the system also provides comprehensive Graphical User Interface.

* **Features**
  1. **Main menu**

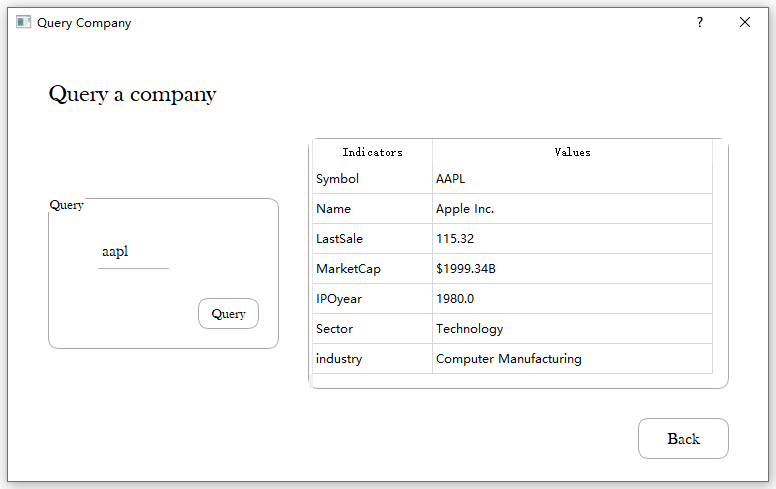
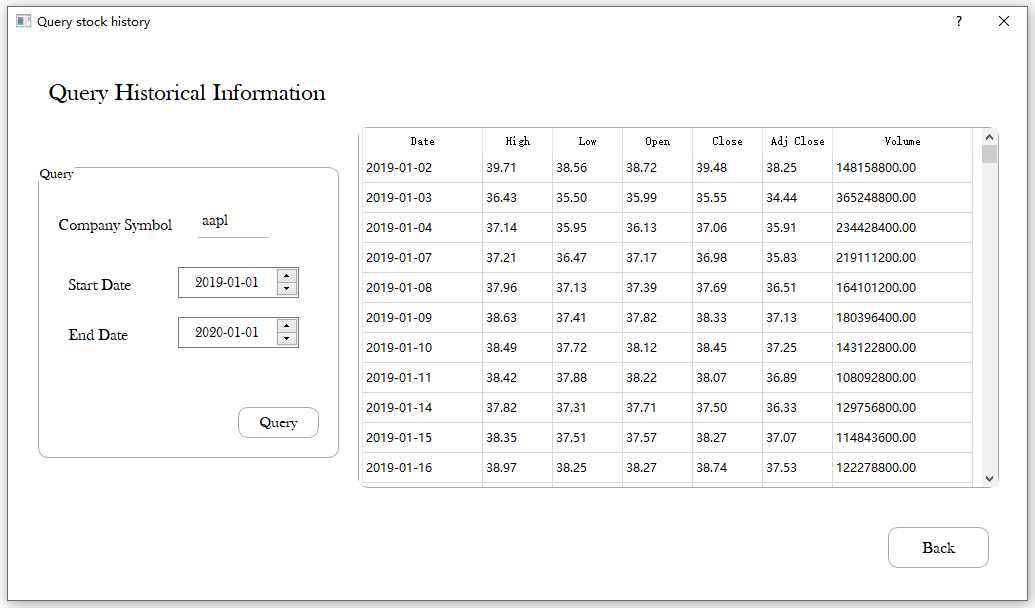
It can be composed of 3 parts: *Quick query*, *Daily News browser* and *Menu bar*. *Quick query* is a quick way to get the daily price of a certain stock, and you can change the website to get newest financial information from combo box, where we provide 5 options: Nasdaq, Google Finance, Yahoo Finance, World Bank and Wall Street Journal.

The *Menu bar* consists of *Start*, *Files* and *Help.* Most of the analytic functions are listed into *Start*, and you can manage the output files with *Files*. With *Help*, users can get instructions of this system.

* 1. **Start**
     1. **Query a company / Query stock history**

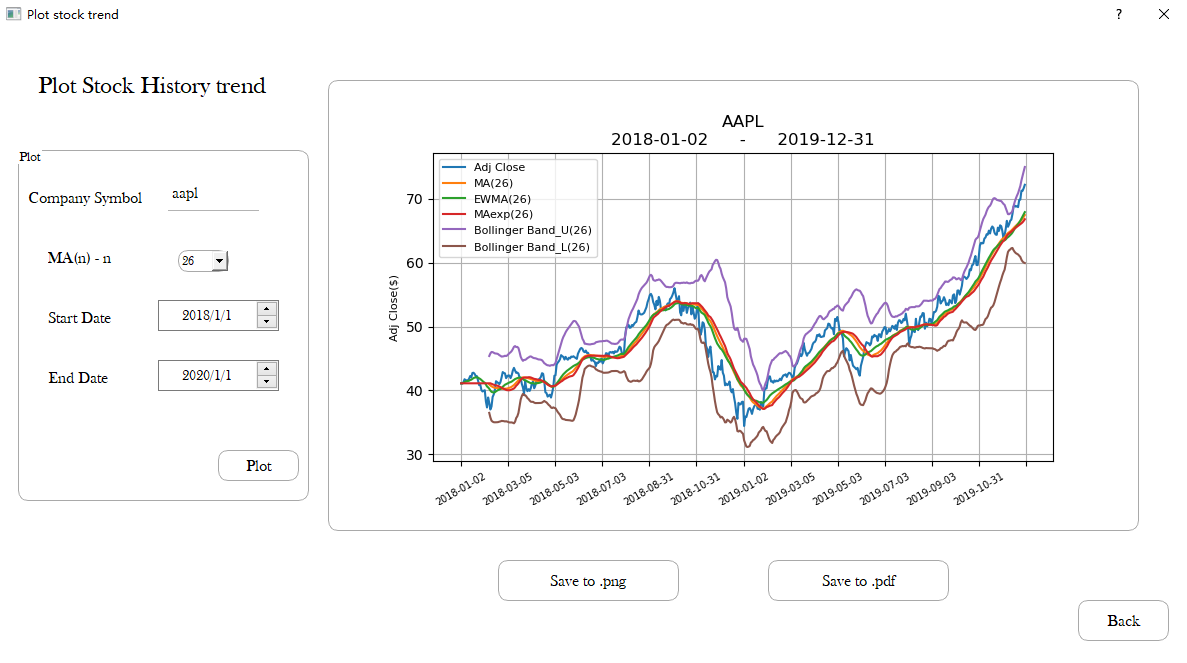
*Query a company* can helpfully present basic information to system users, like Company Full name or IPO date, and a quick way to check whether a company is in our system (companylist.csv) or not. And *Query stock history* aims to help users query raw history data price information.

* + 1. **Plot stock trend**

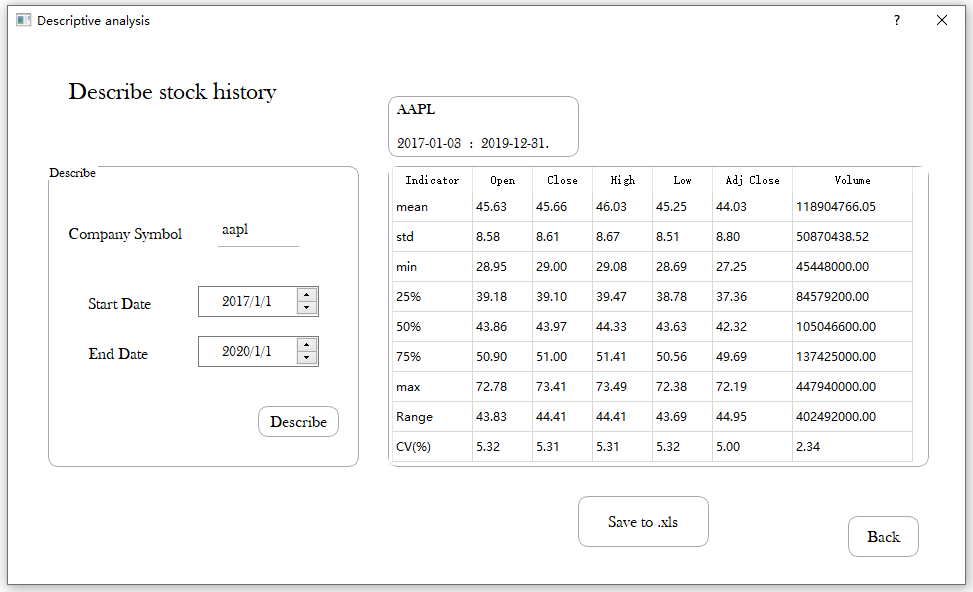
This is our main visualization tool of the system, by simply inputting the company symbol and time period, and selecting ‘n’ for our indicator like MA / EWMA / EXPMA, the picture of this stock in this time period will be presented. What’s more, the Bollinger Bands analysis result was also performed, which would be a great aid in initiating trade strategy.

We also provide users with two options to export the result to the folders we created for you , with PNG and PDF formats. In terms of the folders, you can find these three folders we created for you at the first-time logging into our system: *company\_database, company\_PNG* and *company\_PDF*. *company\_database* is mainly for storing raw data, in the form of .csv, and the descriptive analysis files. The other two folders are to store the visualization outputs, both history trend and prediction plotting tasks.



* + 1. **Descriptive analysis**

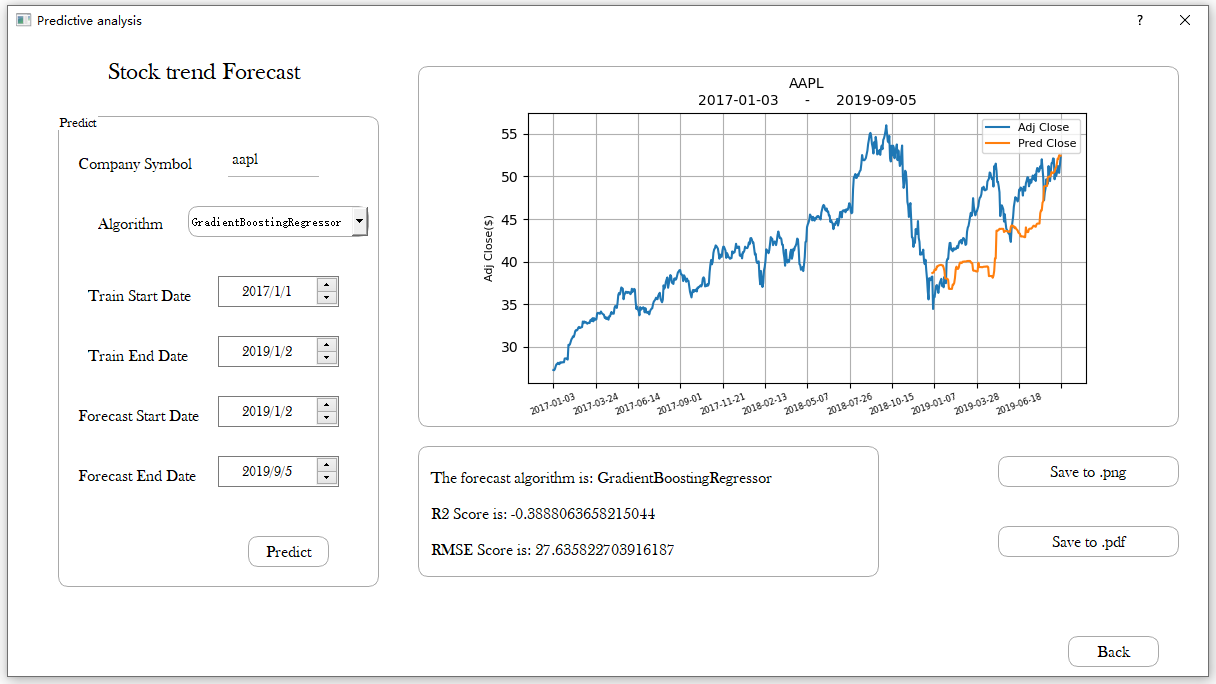
This is for the descriptive analysis of stock market. In addition to some basic Indicators like mean and quarters, we also provide Coefficient Variation for each column.



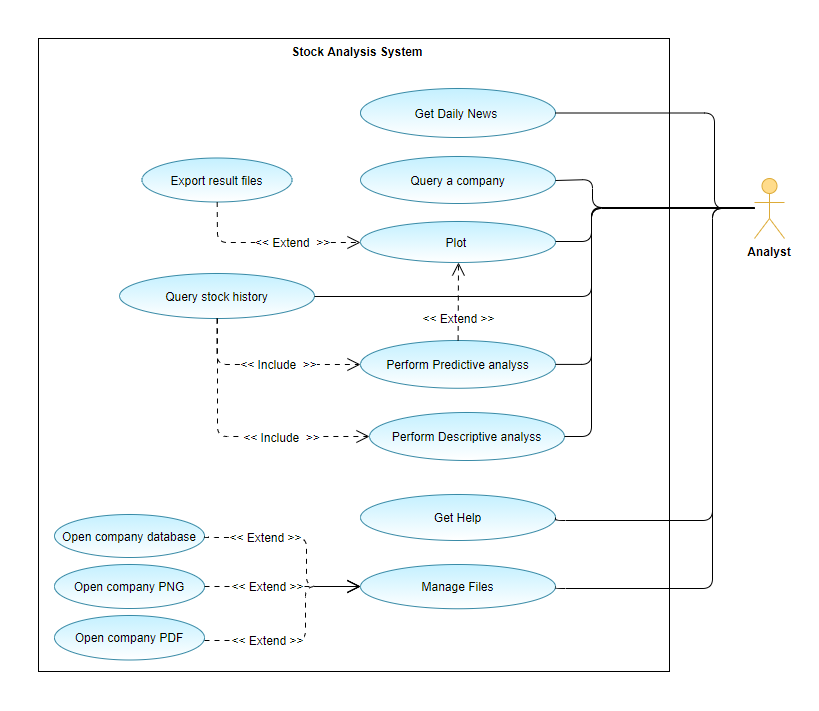
* + 1. **Predictive analysis**

In this part, the system allows users to do Forecast on stock prices, with the application of Machine Learning algorithms. You can choose an algorithm from 4 options: GradientBoosting, ExtraTrees, RandomForest and XGBRegressor. By the way, GradientBoosting is recommended with a relatively better performance.

Users can select a time period as training set, and specify a time period as their target. Then by clicking *Predict,* the result will be presented in the text Browser with R2 and RMSE value be calculated. Just as you expected, the visualization can also be saved to .png or .pdf.



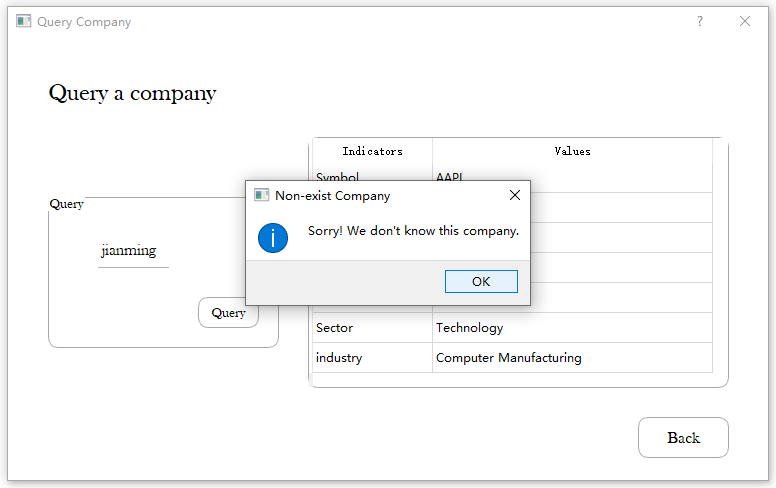
# UML Diagram

****

# Highlights and Innovation

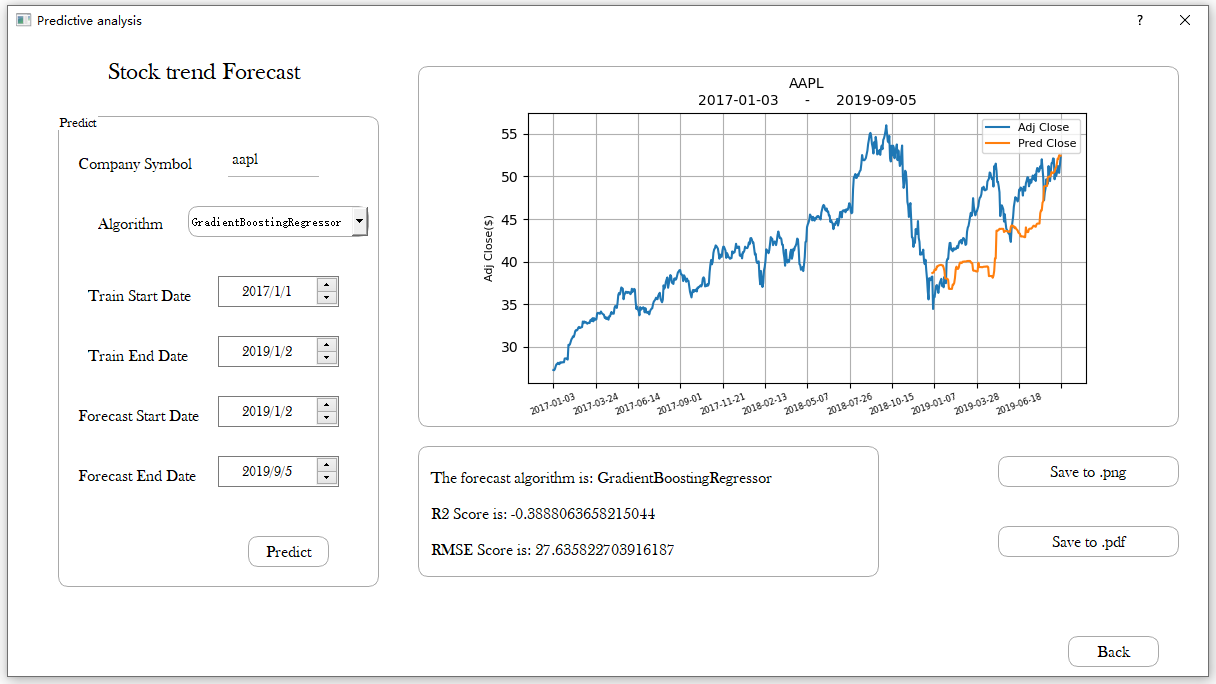
* **Robustness**

In the whole system, there are 6 TextEdit box for user to put in *Company symbol*, and all of them has Illegal input check policy. System will firstly search this company from *companylist.csv*, and only proceed when the company is truly existing. Similarly, it will always check the Date that users input into the program, the start date should always be earlier than the end date, or an error will be raised. What’s more, with the fixed input format of GUI date-editor, the Date format will be checked automatically.

****

* **Forecast Comparison**

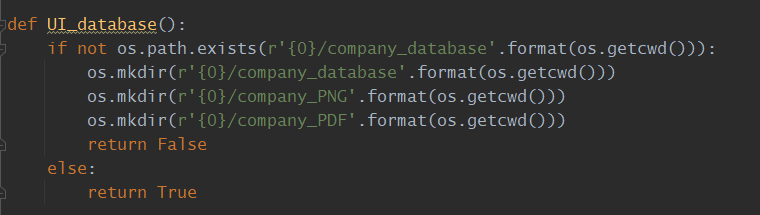
In Predictive analysis, the system can not only evaluate forecast result with R2 and RMSE score, but also plot the result in the same figure with the ground truth data.

****

* **Files Management strategy**

In the first-time of execution with a new environment, the program will automatically create three folders: *company\_database, company\_PNG* and *company\_PDF* in the directory of python program. And they are supposed to store the downloaded data and visualization outputs. In this way, it is very convenient for users to manage the files. The system also provides the entry to access these folders within the system, with *Files* option in menu bar.

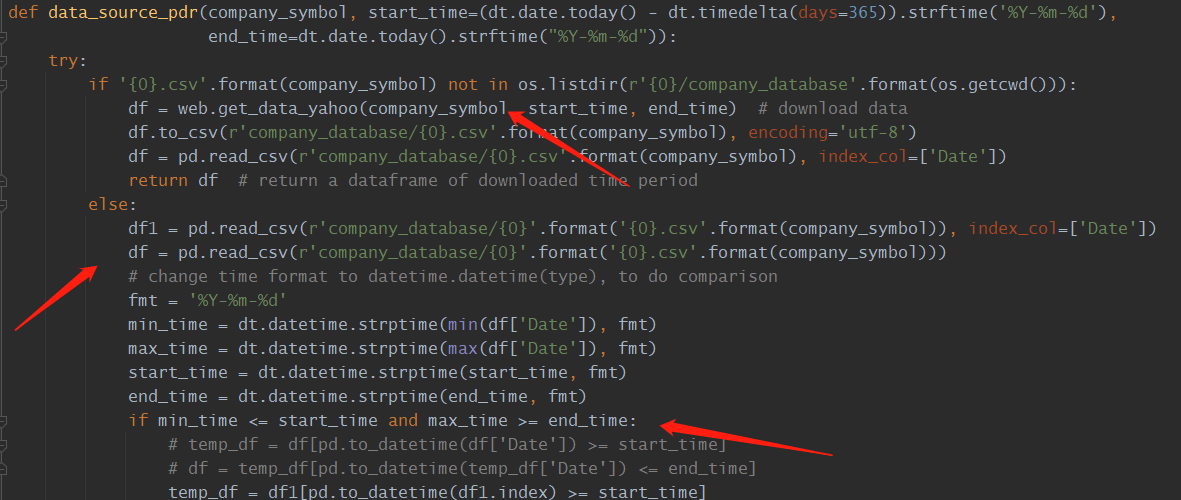
On the other hand, it can also aid data usage management process. In the future update, maybe the system can use real database instead such as MySQL or Hive and so on. The idea of this local files folders is also coming from real database practice.

****

* **Downloaded-data Optimization**

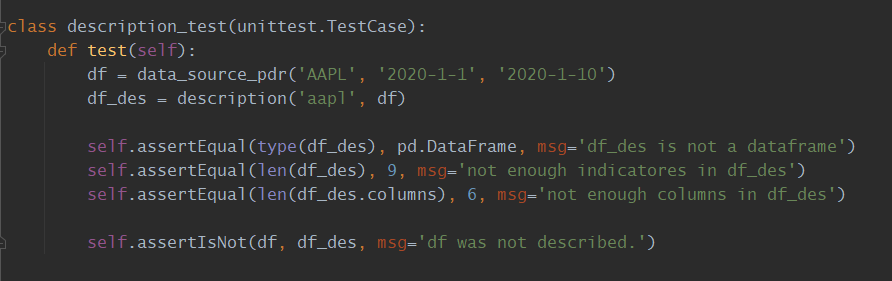
It will be a waste of internet resources if you download data with API from time to time, and may result in exclusive internet traffic. This system can solve the problem with an optimization strategy: Every time user tries to get data, check local files first, and only request new data if we do not have it.

As we mentioned before, if the data file of this company is existing in folder - *company\_database*, then check the time period, if the data user requested is a subset of the whole data file, then just take it from the file, otherwise we request data with API. By doing this way, the program can run faster and some internet waste will be avoided.

****

* **Unittest Module**

The system takes advantage of Unittest framework of python to test a series of module and functions, to better ensure the reliability of the program.

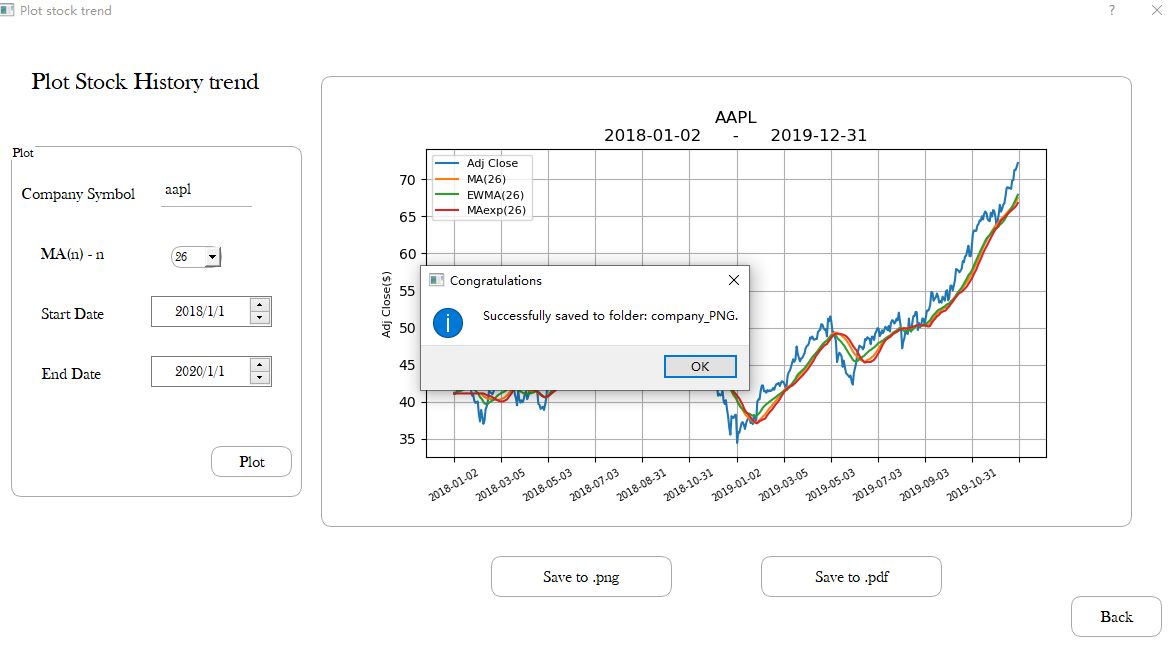


* **GUI design**

The development of GUI in this program is based on PyQt5, there are 8 windows in total, including 1 Main window as main menu, and 7 Dialogs for different modules.

All the buttons were designed with round corner, which may make users feel more comfortable when dealing with complex transactions.

For those buttons with more important functions, some message boxes will be presented to users to let them know whether the orders are executed or not. For instance, there will be a positive response to the users with the *Save to .png* order be executed successfully.



# Reflection and Improvements

* **Forecast strategy**

Although the system provides 4 Machine Learning algorithms to perform Predictive analysis, it still needs more complement.

First, the whole prediction is based on sklearn Python library, and I am still a starter in Machine learning algorithms. Even though I know how to use these algorithms, the Hyperparameter adjustment process was still not performed in this system. Second, some of other algorithms which can be used into forecast were not applied. Prophet, an algorithm developed by Facebook, was used in this program at first, but I left it because I found that it does not give better forecast. ARIMA is another similar time-series forecast strategy, but it is also based on seasonal indicators forecast. However, from what I heard, LSTM can be a good way to predict stock market, but that was far beyond my ability to bring it to this system for now, maybe in the future this will be completed. Besides, the linear trend line approach was performed at very first as the forecast method. However, I do not think it is a proper way to do predictive analysis in stock market, which is the reason why it was removed from the final system.

In general, what I learned from this forecast process is, stock market is not like those stable industry like supermarket, it changes drastically. Any small disturb will result in a great bias from ground truth. On the other hand, the unpredictable features like policy and global events (i.e., COVID-19) cannot be considered into feature engineering process, which is also a great reason that the prediction do not perform well.

* **OOP design**

The Object-oriented programming method was not completely applied in this system, only 40% (mainly in GUI windows design process) of coding works were implemented with the application of Class. For those core functions, most of them were designed separately.

For reflection, it is completely possible that more features are being carried out with OOP method in this system. In prediction analysis, a *ForecastModel* Class will help to normalize the modeling process. What’s more, a *StockDatetime* Class can also aid in processing those troubling date transformation process.

* **Stability and Running speed**

The underlying architecture of this program is not stable, sometimes the program will crash without reasons. It is mainly because the conflicts of different libraries, and problems of requesting data from API. Furthermore, as we discussed, the uncomplete implementation of OOP method can also be a reason. Besides, it takes quite a long time to perform Predictive analysis and visualization, and it seems that there is no fixed time spending when plotting a figure, these can be optimized in the future update.