# Stock Prediction Application - Prompt B

## Problem Statement –

The problem with GIC’s current configuration revolves around a lack of a predictive, flexible, and description solution. This inhibits GIC from optimizing adaptive trading strategies and being able to communicate the results to clients. Adopting a tool that can provide better throughput and potential improved accuracy in trading strategies.

## Customer Summary –

The proposed Stock Prediction Application (SPA) will provide the following opportunities for GIC’s lacking framework:

* Flexibility to evaluate a stock prediction by itself or against other stocks loaded into the dataset.

The SPA will grant users the option to either predict a stock based on itself (labeled, data only), or make a prediction based on other stock datasets. Additional datasets beyond the target stock will be considered independent or descriptive features. By including these capabilities within the SPA, the application sets itself apart by allowing the user to easily see how either positive or negative correlations effect the prediction of the stock. Each feature will also be provided its own scaler within the source code allowing the descriptive features to come from datasets with completely different range (I.E. predicting a penny stock based on Nasdaq).

* Visualization of predictions, correlated data, and hierarchical clustering.

The SPA will provide visualizations of the training data set predictions as well at the actual future predictions. The training and future prediction visualizations will be in the forms of active response linear graphs. These graphs, leveraged from the Plotly package, contain their own tools bars and scroll-over data display to allow for a richer and more dynamic user experience.

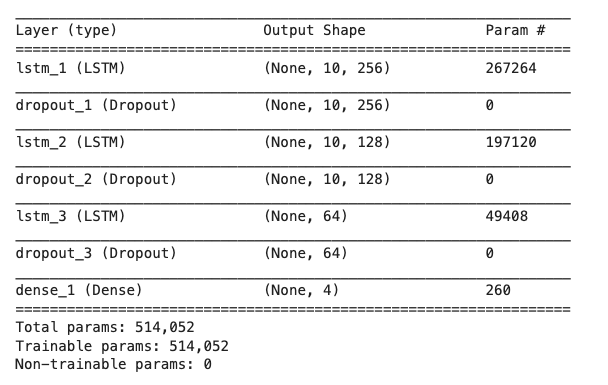
When two or more stock datasets are loaded into the SPA, the application will provide two additional graphs of processed correlated data:

* + Heatmap – this will display a graph in blocks of color, legend provided, that will represent the correlations between the datasets. This graph is also leveraged from the Plotly package and has an interactive toolbar and scroll-over displays of the data.
  + Dendrogram – this graph will chart the hierarchical clustering in a bottom-up approach. This means that the closest correlations will display as connected together and closest to the bottom. From there, every connection up the graph will indicate a weaker correlation. This graph is also a part of the Plotly package.

These visualization tools will set the SPA apart as a user-friendly and viable predictive tool and will allow also GIC to better communicate strategies to clients.

* Ease of model refinement

Leveraging a long short-term memory (LSTM) recurrent neural network (RNN), the source code behind the application will easily be configurable to add or subtract different types of layers to the model. Beyond this, parameters such as: batch size, sample size, number of epochs, step size, and number of days to predict will be able to be extracted to a front panel with little additional work required. For the initial development release of the software the RNN model with solely be located within the source code with the following architecture:



The SPA will be developed as a web application. Initially it will be released on <https://mybinder.org>, pending the approval and interested of GIC, the application would eventually be migrated to their website as a tool for employees and clients.

The application will provide the user a file upload to load the data from the desired datasets. Beyond that, the rest of the interactions with the user will be button clicks to: process data, train the model, and run the model predictions. The application will initially be designed to keep the RNN configuration parameters hidden from users. Keeping the parameters hidden will allow for consistency across datasets. The configuration parameters that will be implemented in the source code are shown in the table below:

|  |  |
| --- | --- |
| Parameter | Value |
| Input (look-back) | 10 |
| Features | Number of dataset columns |
| Epochs | 20 |
| Batch size | 50 |
| Days to predict | 5 |

Existing System Analysis –

GIC's existing system has been implementing a stock tool that is based on the current correlation of stocks based around linear regression. This tool can be used to make trend predictions; however, it is based mainly on correlation only. This measurement ultimately boils down to the equation of a straight line (y = mx+b) as the correlation slope. This kind of stock prediction can be seen as more ineffective because it tries to measure the non-linearity of the stock market within a linear measurement. The current tool is not optimized for creating stock trend prediction based on large datasets.

Implementing the SPA in place of GIC’s current software would provide the functionality of a LSTM RNN. RNNs can be much more effective at learning trends based on datasets because of the ability to remember past values and apply those values to the current measurement. Based on the output of the RNN compared against the validation data, the RNN will then adjust the weights of each neuron within the architecture before processing the next batch of data.

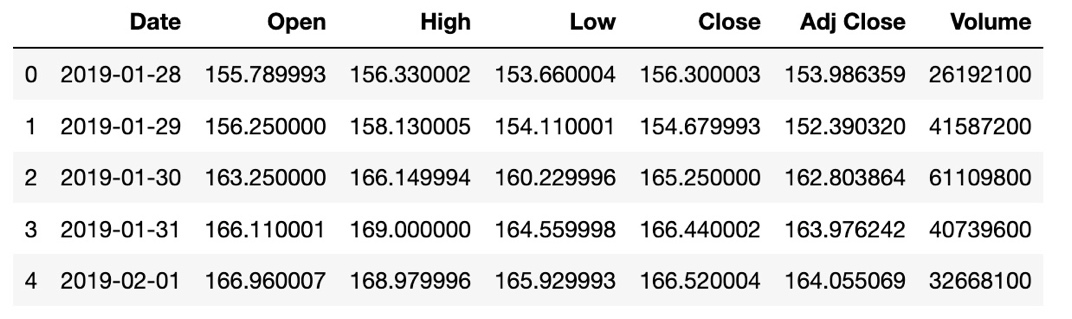
GIC’s present software is a license-based desktop application. This configuration prevents users from accessing the tool on different systems without downloading the application and purchasing another license file. The tool is also not available to clients who would like to self-invest and leverage GIC’s tool framework. Migrating this part of GIC’s toolset to the SPA will allow for employees to access the prediction application from anywhere with internet connectivity. The SPA web-based solution will also make it available for client use.

Data –

The data processed by the application will be based around the .csv structure of historic stock data downloaded from <https://finance.yahoo.com>. Thus, all datasets to be processed by the application must have a file type of .csv. Example datasets will be provided in the “Data\_Samples” folder of the proposal package. Leveraging the Pandas Python package, the application will be able to read the .csv and load it into a Pandas DataFrame. The DataFrame will be the base object of all data manipulation for the application. An example of how this will be performed is shown in the code snippet below:



The previous code will output the following:



The function “.head()” will output the first 5 rows of the DataFrame as displayed above.

*The SPA will assume that the user if the user is providing multiple datasets that each dataset is in the same format and contains the same number of rows*. For consistency it will be recommended to pull stock data from Yahoo! Finance with the same timeline for each stock. This will ensure that the datasets are formatted in the same way. Since the application will make a 5-day prediction the “Date” column of the dataset will need to be in timesteps of days formatted as: YYYY-MM-DD. The feature from each dataset that will be used in the prediction is “Close” column. The collaborative DataFrame built from all the loaded datasets will contain an indexed date for each timestep and the “Close” column from each of the datasets passed in the SPA. Because of this, *it will be necessary that datasets passed into the SPA have “Date” and “Close” columns*.

The processing and cleaning of the datasets will begin by dropping all null rows in the DataFrame, this will aid in ensuring accuracy throughout the measurements. After this, each individual feature (column) of the DataFrame will be pulled out and scaled to a relative number between 0 and 1. The MinMaxScaler functionality from the SciKit-Learn Python package will be leveraged to achieve such scaling. A Python dictionary will contain the scaler objects for each feature, the dictionary will be keyed by the stock name previously uploaded. The process of training the data will happen over two datasets: the training dataset, and the full dataset. The training dataset will consist of the first 80% of the full data, the model will train and be validated against this data. Once the data is scaled and reassembled into the main DataFrame it will partitioned into the training dataset.

After the data has been processed and partitioned it will be passed into the model. LSTM RNNs require that input be in the shape of tensors. Tensors are 3-dimensional datasets in following structure:

A close up of a logo

Description automatically generated

*(*time\_series*, Time series forecasting,* TensorFlow, 2020)

In order for the model to train and validate properly, the data will have to be shaped in a way that allows for progressive validation. This mean the next value after the lookback timesteps will be the value added to the validation dataset for that series of time steps. A rudimentary example of the data series would look like this:

|  |  |
| --- | --- |
| Training Data | Validation (number to be predicted) |
| [[1], [2], [3]] | [4] |
| [[2], [3], [4]] | [5] |
| [[3], [4], [5]] | [6] |

Preparing and shaping the data for the model will be one of the most important steps of the SPA. The DataFrame will be shaped into the tensors will the following algorithm:



The “shape\_data” function will return arrays built using the NumPy Python package. The package will allow for array reshaping within the source code. The algorithm shown above will return the training data (x) in the shape of a 3-dimensional NumPy array which will be the tensor model input. The validation (y) data will be in the shape of a 1-dimensional NumPy array. After the data is shaped into the proper datasets it will be ready to input into the model.

Project Methodology –

The SPA will be developed in an Agile environment. Developing in an Agile environment will allow for rapid revision of the product and constant improvement and debugging. The product development process will begin by research on the best way to implement the RNN. Once the decision on the model to use has been made, focus will shift into locating data resources and data formatting. Different types of independent features will be experimented with to determine the best route for optimal prediction accuracy. The project process will then roll into the development process, the application will go through iterative updates and added features based on the requirements provided by GIC. The application functionality will be segmented out into different runnable portions and will be worked on by different developers. This divide-and-conquer strategy will streamline the development process to ensure that deliverable deadlines are met.

Unit testing will consist of functional testing and accuracy optimization. Refining the configuration of the learning model will be a large portion of the unit testing. This will consist of running many tests with varying numbers of: days to predict, epochs, batch sizes, and timeseries look-backs. The process will attempt to settle on a model configuration that is generic and effective across many different kinds of datasets.

The integration of the application will ensure the accuracy of the graphs against the data being passed in. Varying numbers of properly formatted datasets will also be run over several iterations to verify that the application can perform as necessary under different constraints. The application will then be validated on several browsers and operating systems to ensure its interoperability across platforms.

The SPA system test will run using a stripped-down interface built on <https://mybinder.org> and the Voila package to render the source code dashboard. This will allow for a real-world test of the application and validate the functionality on deployment. Acceptance of the application on the development end will consist of the successful functionality of the application. Since the stock market is largely unpredictable it will be determined by the customer if the software meets the standards put forth.

Project Outcomes –

The project outcome will result with two types of deliverables: development process deliverables, and final outcome deliverables. The development process deliverables will consist milestone achievement and schedule, application piece-parts and runnable code segments. The action items recorded in the scrums will be recorded and charted for the customer as an additional form of accountability to meet delivery deadline. runnable code segments will be available in the project source control repository via GitHub. These deliverables will be scheduled to be delivered every two weeks at the GIC customer requirement sync-ups.

Final outcome deliverables will consist of a functional web application, sample datasets, and complete source code. The source code will be source controlled in GitHub to allow for optimal flexibility and redundancy.

## Implementation Plan --

The implementation process will begin once the application has reached initial acceptable functionality. This will begin with the customer “development release”. The development release will consist of initial user testing and identification of potential software bugs. An opportunity for additional customer requirements will also be derived during this release. After the development release the application will be taken back to development to add additional features and work out identified bugs.

After the development revisions have been made, the “alpha” software release will be delivered to the customer. The alpha release will consist of the following deliverables: a functional application, source code repository, and an operation manual. During this time the application will go through extensive black box user testing by GIC. For one month, GIC will use the application regularly and log additional bugs during this time. Once the alpha release has lapsed the application bug log will be brought back to application development. The bugs will be fixed and the application will go through regression testing to validate the bug fixes did not break any other parts of the application execution.

Application deployment will then roll in the beta release of the software. For a period 3-6 months the application will be regularly used by GIC and GIC clients. All bugs will be recorded by GIC for the beta period. After the beta period the bug fixes will be implemented, regression tested and pushed to the repository. The final release will follow the bug fixes and software implement following the beta period. The deliverables of the final release will include: final SPA, all raw source code files, operations manual, sample datasets, and included software support for the following 6 months.

## Evaluation Plan –

The evaluation plan for the SPA will include functional acceptance and measurement satisfaction. Functional acceptance will come from the application being able to perform the tasks specified in the requirements. These requirements include: the description and processing of datasets loaded into the application, prediction of stock based on selected features, and visualization of the data from the previously listed requirements.

The second part of evaluation will be determined in measurement satisfaction. The nature of the stock market is impossible to completely and accurately predict because of infinite amount of potential independent features. The metric that will be used in the application and provided to GIC is the “root mean squared” (RMS) statistical measurement. The RMS formula is shown as follows:

A picture containing object

Description automatically generated

(RMS, *Root Mean Squared,* Wikipedia, 2020)

RMS will be used to calculate the difference between the prediction of the target feature and the actual value of the target feature. This information will available and printed out after the first iteration of the training data. The RMS will not be calculated for the actual future prediction because there will be no data to validate it against.

## Resources and Costs –

GIC will not be hosting its own server but rather hosting via a third-party; this cost will be reflected in the price breakdown table. GIC already hosts its website so factoring in URL and webpage will not be reflected in the pricing. GIC already has a baseline infrastructure, the SPA will be integrated into the current structure which will save GIC money in cost of hardware and other software tools.

Required costs:

|  |  |
| --- | --- |
| Requirement | Cost |
| Server space | $50 (monthly) |
| Final release build software | $250 (one-time) |

## Environment Costs --

The application leverages open source IDEs, software libraries, and software package; because of this, there will not be any additional cost for the software to develop the application. The only environment cost to GIC will be server space, this pricing is reflected in the table located “[Resources and Costs](#_Resources_and_Costs)” section of this document.

## Human Resource Requirements –

Pricing based on base rate of $35 an hour.

|  |  |  |
| --- | --- | --- |
| Requirement | Time (Hrs.) | Cost |
| RNN Planning | 20 | $700 |
| Data Sourcing and Planning | 20 | $700 |
| Development, bug fixes, and feature additions | 120 | $4,200 |
| Meetings and Milestone Tracking | 15 | $525 |
| Testing (unit, regression, acceptance) | 40 | $1,400 |
| Documentation | 30 | $1,050 |
| **Totals** | **245 hrs.** | **$8,575** |

## Timeline and Milestones–

|  |  |  |
| --- | --- | --- |
| Start Date | End Date | Milestone |
| 2/10/2020 | 2/15/2020 | RNN Planning & Data Sourcing |
| 2/17/2020 | 3/9/2020 | Initial Development |
| 3/9/2020 | 3/16/2020 | Initial Testing |
| 3/16/2020 | 3/30/2020 | Development Release |
| 3/30/2020 | 4/27/2020 | Alpha Release |
| 4/27/2020 | 7/27/2020 | Beta Release |
| 7/27/2020 | 8/31/2020 | Final Release |

# Stock Prediction Application – Prompt C

## Data Methods –

The data methods within the Stock Prediction Application (SPA) include a descriptive method and a non-descriptive method.

Examples of acceptable descriptive methods:

* K-means clustering
* Hierarchical clustering
* Other clustering methods (max distance, min distance, etc.)
* PCA (all variables need to be numeric)
* MCA (all variables need to be factorized)
* FAMD
* Logistic regression with interpretations of estimated coefficients

Multi-linear regression with interpretations of estimated coefficients

Examples of appropriate non-descriptive methods:

* Logistic regression
* Decision trees
* Random forest
* Neural network
* Multi-linear regression (if it includes a strong justification – like ease of

interpretability)

**Datasets** – The use of dataset(s) is a critical element and involves the gathering and measuring of information on targeted variables in a systematic fashion. This could be student collected (Please consider IRB ramifications.) or publicly accessible such as websites (e.g. Kaggle.com), governmental (e.g. Department of Labor), or software related (e.g. GetHub.com). Be sure to consider the methodology used including possible disadvantages and challenges.

|  |
| --- |
| **Analytics**– Using the given data, your application needs to enable decisions to be formulated or support for given trends to be provided. **Data Cleaning** – if applicable, create a function that will make the data usable prior to actually being used by the application. Things such as featuring, parsing, cleaning, and wrangling the datasets.  **Data Visualization** – You need at least three real-time (e.g. using the GUI/dashboard) formats to visualize the data in a graphic format. Look at things like charting, mapping, color theory, plots, diagrams, or other methods (tables must include heat mapping). These, in conjunction with or as a part of your GUI, would enable users to explore or inspect the data characteristics.  **Real-Time Queries** – As part of your GUI enable users to access and manipulate data real-time including data maintenance. This does not deal with data “freshness” but with the query response time being in seconds. |
| **Adaptive Element** – if appropriate for the business need, provide the implementation of machine-learning methods and algorithms to enable the application to improve with |

experience. Examples include learning associations, classification, statistical arbitrage, prediction, extraction, and regression.  
**Outcome Accuracy** – provide functionalities that evaluate the accuracy of the

information/outcomes given by the application. What are the parameters for valid output data and how will those be checked by the application?  
**Security Measures** – include industry-appropriate security features that will control access to the data and/or, how the data is stored or transmitted. The security features should be appropriate for the data product and the sensitivity of the data it interacts with. For example, with a web application this requirement might be satisfied by implementing username/password authentication.

**Product Health Monitoring** – include functionality that will enable the application’s “health” and reliability to be monitored. It should answer the questions, “Is the application performing correctly?” For example, the use of displays or logs to quickly discover, isolate and solve problems that can negatively affect the application’s performance and accuracy.  
**Dashboard** – include a user-friendly, functional dashboard that enables the query and

display of the data, as well as other functionality described in this section. This could be stand-alone, Web-based, or a mobile application interface.