Table of Contents

Stock Prediction Application – Prompt A

Problem Summary –

Growing Investment Company (GIC)'s current configuration insufficiently provides the company the adaptable and predictive functionality necessary for their growing business. These lapses prevent GIC from optimizing adaptive trading strategies and being able to communicate the results to clients. The Stock Prediction Application (SPA) will be the recommended solution to meet GIC and their clients' current and future needs.

Application Benefits –

The SPA will provide the following application benefits:

- Comprehensive and interactive data visualizations
- Correlated calculation of uploaded datasets
- Adaptive and predictive outputs

Application Description –

The SPA will be designed with two main features

- Hierarchical correlation clustering calculate how different datasets are related to each other. These results will be visualized as a data comprehension aid for GIC and their clients.
- GIC's predictive functionality needs will be met by implement a long short-term memory (LSTM) recurrent neural network (RNN). The RNN will be created leveraging the TensorFlow framework. The LSTM RNN is a state-of-the-art and optimal predictive solution that will enable the user to generate a 5-day stock prediction. Results generated by the RNN will also be graphed for the user.

Data Description –

The data that will be used in the proposed SPA will be numeric data in .csv format. The data will be collected from Yahoo! Finance and will base the application input on these files. The dependent feature of the dataset is the “Close” feature of the target dataset (stock to predict). The independent features processed by the application will be the “Close” feature of independent datasets (other stocks). Files will be parsed and managed in Pandas DataFrames within the application.

Objective and Hypothesis –

The objective of the application is to predict the next five days of the target stock. Predictions of exact pricing is the not objective, but rather price trends (up, down, flat). The hypothesis is that if the user inputs valid correlated datasets into the application then the accuracy of the prediction will be greater.

Methodology –

The methodology that will be used for the research and development of the application will be Agile. Agile methodology will allow the research and development phases to parse out functionality of the program as incremental deliverables to customer. The Agile workflow will also allow for greater flexibility to add new customer requirements in the middle of development. Debugging the application segmentations will also occur throughout the development process as a result of structuring development around an Agile methodology.

Funding Requirements –

Service	Cost
Hardware and software requirements	$250 one-time || $50 reoccurring monthly
Human Resource Requirements	$8,575

Stakeholders Impact –

The SPA will generate more potential revenue for GIC by identifying stock trends and aiding investors as a predictive tool. The SPA will also aid stakeholders by reducing the client-interaction workload from GIC employees. Clients will be able to use the SPA on their own if they are choosing to self-invest without a GIC financial consultant. This will result in the SPA saving GIC both time and money.

Data Precautions –

There are no data precautions. Datasets that will be used in the application are public information downloaded from Yahoo! Finance.

Developer's Expertise –

The developer's expertise is rooted in machine learning and data analytics. The developer has primarily been working with health predictions over the past 10 years of their career and has recently shifted into financial predictions and analytics. Specializing in identifying trends, the developer will offer a wealth of information to the project.

Stock Prediction Application - Prompt B

Problem Statement –

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 new tool can provide better throughput and potentially improve 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. 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.

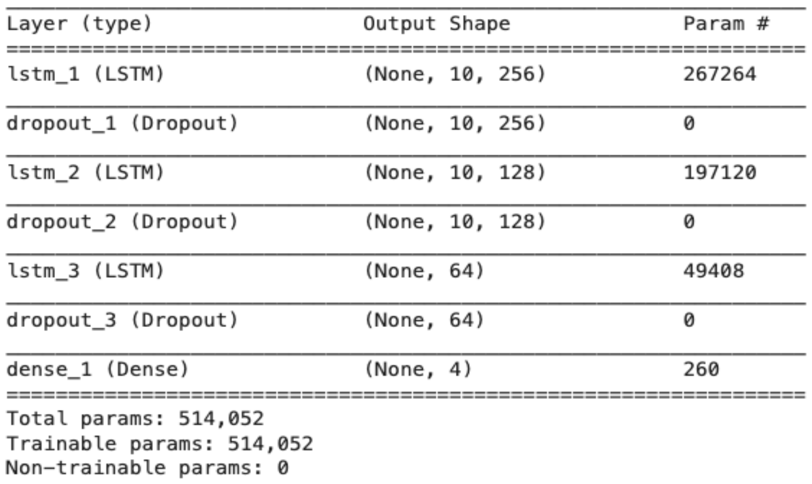
When two or more stock datasets are loaded into the SPA, the application will provide two additional graphs of processed correlated data:

- 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. It 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:



```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 10, 256)           267264
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 256)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 10, 128)           197120
_________________________________________________________________
dropout_2 (Dropout)          (None, 10, 128)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                49408
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 260
=================================================================
Total params: 514,052
Trainable params: 514,052
Non-trainable params: 0
```

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.

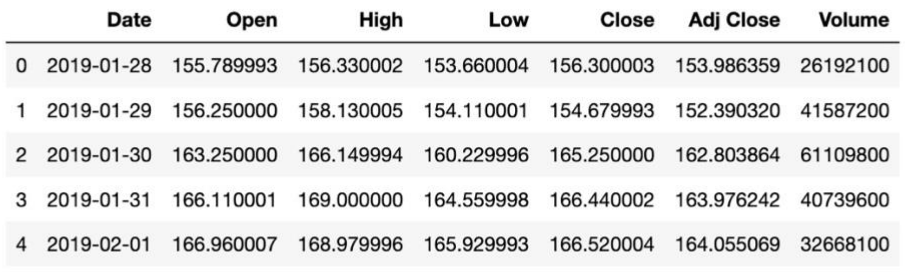
Data –

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:

```python
import pandas as pd

df = pd.read_csv('AAPL.csv')
df.head()
```

The previous code will output the following:

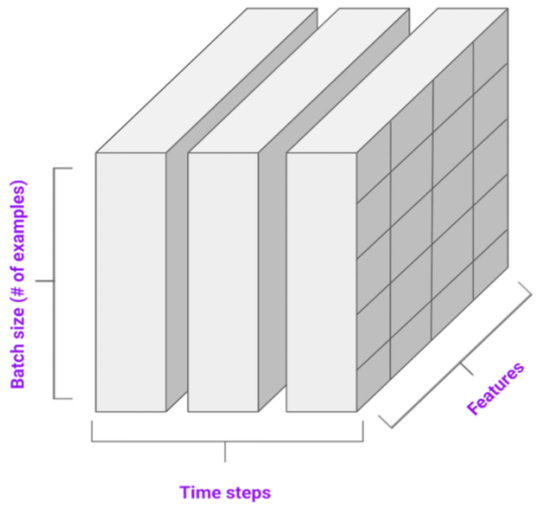


	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-01-28	155.789993	156.330002	153.660004	156.300003	153.986359	26192100
1	2019-01-29	156.250000	158.130005	154.110001	154.679993	152.390320	41587200
2	2019-01-30	163.250000	166.149994	160.229996	165.250000	162.803864	61109800
3	2019-01-31	166.110001	169.000000	164.559998	166.440002	163.976242	40739600
4	2019-02-01	166.960007	168.979996	165.929993	166.520004	164.055069	32668100

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 data will be partitioned into the training dataset and the full 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:

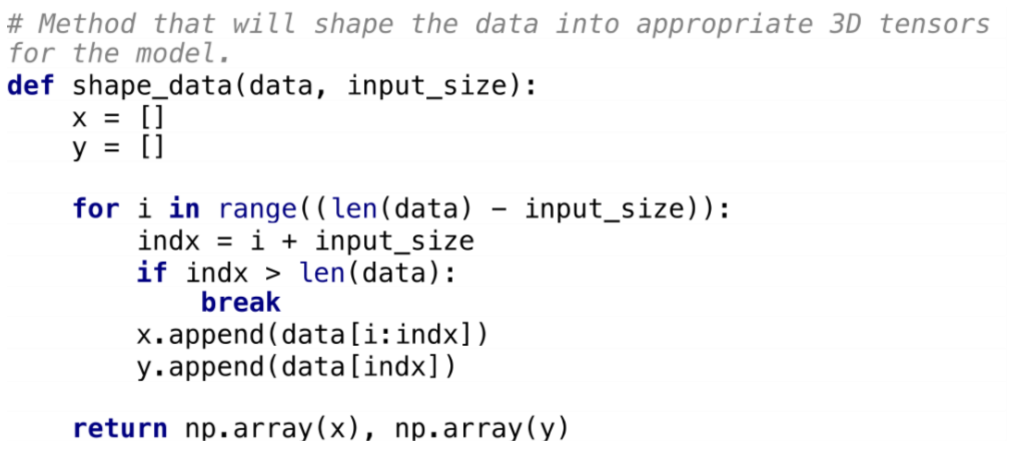


(TensorFlow. 2020. *Time series forecasting.* time_series [digital image]. Retrieved from https://www.tensorflow.org/tutorials/structured_data/time_series)

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 as such:

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:



```python
# Method that will shape the data into appropriate 3D tensors
for the model.
def shape_data(data, input_size):
    x = []
    y = []

    for i in range((len(data) - input_size)):
        indx = i + input_size
        if indx > len(data):
            break
        x.append(data[i:indx])
        y.append(data[indx])

    return np.array(x), np.array(y)
```

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 of 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:

Root Mean Squared Formula

$$x_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}.$$

(Wikipedia. 2020. *Root Mean Squared.* RMS [digital image]. Retrieved from https://en.wikipedia.org/wiki/Root_mean_square)

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" 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 that will be within the Stock Prediction Application (SPA) include a descriptive method and a non-descriptive method.

- Descriptive Method

  The descriptive method that will be used in the SPA in hierarchical clustering. The clustering is based off a correlation table that is constructed using the Pandas package correlation function. The function returns a DataFrame that consists of the correlation values of the DataFrame containing all the descriptive and target features. The correlation is then partitioned into hierarchical clusters and visualized in graphs.

- Non-descriptive Method

  The non-descriptive method that will be used in the SPA is long short-term memory (LSTM) recurrent neural network (RNN). The RNN will be the method used for making all predictions within the application. LSTM RNNs function by have an extra layer of neurons within the model that act a feedback loop to the input neuron. The input neurons contain vector inputs, one input coming from data in the dataset and the other coming from the loopback of the hidden neuron which contains the previous iterations of data.

Datasets –

The example datasets that will be used for the SPA come from Yahoo! Finance. They are public files with a constantly updated and extensible library of information. Yahoo! Finance historical data is the recommended data source for the SPA. All datasets downloaded from “Historical Data” of Yahoo! Finance, provided they contain the same date range, will be compatible with the SPA without any additional user processing.

Analytics –

The SPA will receive its analytics as a result of the RNN model predictions. The RNN will learn throughout 20 epochs and make a prediction based on what it has learned. The RNN model learns by starting at an arbitrary prediction, after the model makes a prediction, the neurons inside the RNN are assigned different weights. These weights help the RNN refine its learning process by evaluating the input data through calibrated neurons and data that has already passed through the RNN. The loopback nature of the LSTM allows the model to hold onto values and make a more accurate prediction. The SPA will output a prediction of the target stock price for the next 5 days.

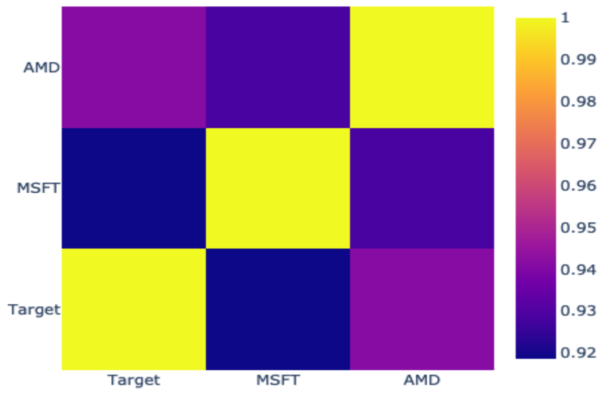
Data Cleaning –

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. The following code snippet will be used to perform the data cleaning:

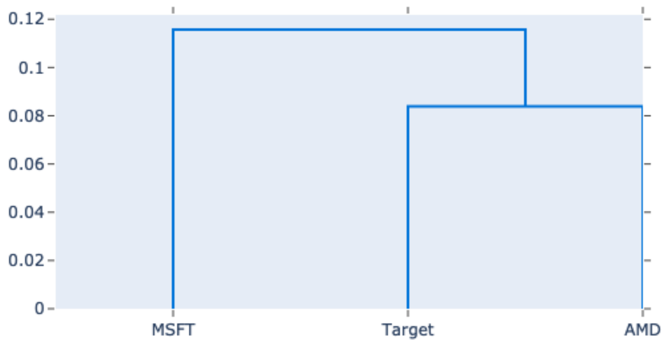
Data Visualization –

The hierarchical clustering descriptive method of the SPA will have two different kind of visualizations for the data: a heatmap and a dendrogram. These visualizations will help the user to better interpret how the datasets are correlated to each other. The caveat with SPA will be that in order to run hierarchical cluster analysis on the data, there must be more than two stocks uploaded to the application. If the user processes data on a single stock, the application will output that it is not capable of running correlations on less than two stocks. Below are examples of the hierarchical clustering visualizations:

Heatmap:



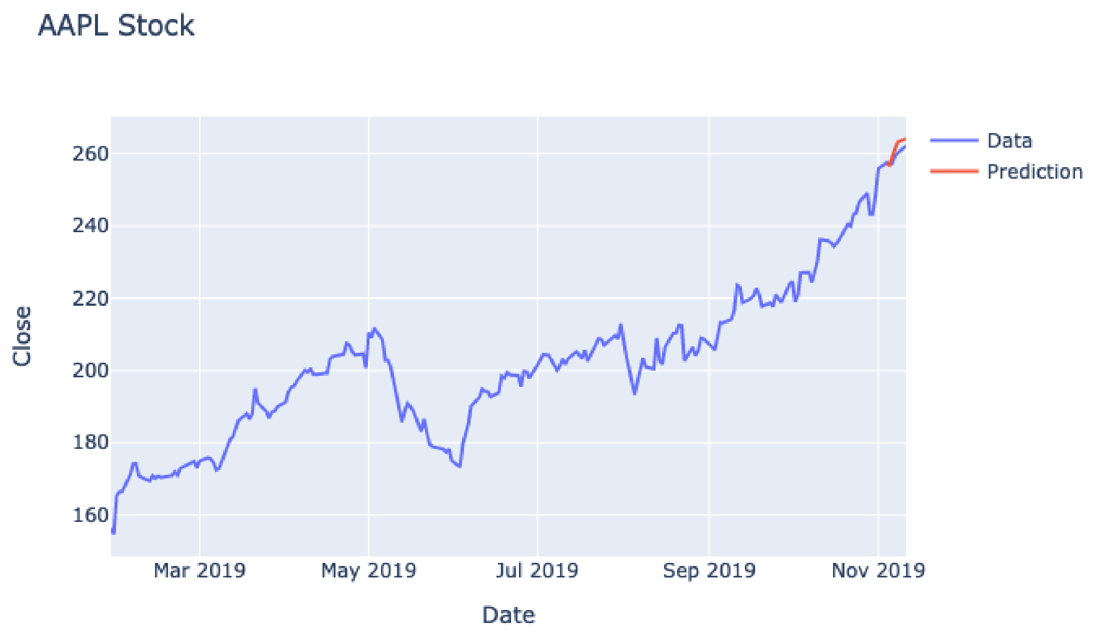
Dendrogram:



Linear graphing will be used to visualization the output from the non-descriptive model. The graph will contain two plots: a real value plot and a predicted value plot. The predicted value is only 5 days and thus will be much shorter than the real value plot. The user will be able to, if desired, select a box on the graph that will some in the selected section. The linear graph will used on the training data results as well as the 5-

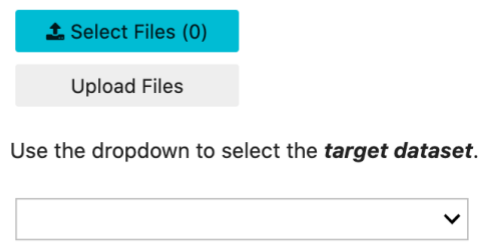
day future forecast results. Below is an example of what these linear graphs will look like:

Linear Prediction Graph:



Real-Time Queries –

The SPA GUI dashboard will allow user to run real time queries by selecting a file after from the file browser that appear when the “Select File” button is pressed. When the application run locally the response time will run very quick; however, when the application will be hosted on Binder the response time can be lagging. Nonetheless, the application will still allow the user to select files, run descriptive correlations, graph the data, and make predictions on the data in real time. The file selection portion of the application is shown below:



Use the dropdown to select the ***target dataset***.

Adaptive Element –

The adaptive element in the SPA will be contained within the LSTM RNN. The more that the RNN model will train on datasets, the more accurate and calibrated it will become. The happens by adjust the weights of each of the neurons based on the output validated again the training set. There are two iterations of training that will occur within running the program, one on the training dataset and one on the full dataset. RNN models can be over-trained on training data which mean that the weights are shift just to cater to predict the training data only. Over-training can lead to more deceptive accuracy during the training and poor results in the actual prediction. It is for this reason the application only allows the model to run through the training dataset once before training on the full dataset.

Outcome Accuracy –

The outcome accuracy of the measurement will come from the RMS measurement that is output as a result of the first iteration of training; a review of description of the RMS can be found in the "Evaluation Plan" portion of this document. This measurement will indicate the accuracy of the prediction. The results can vary based on several different factors such as: the calibrated weights of the model, the correlation of the datasets loaded, and the cumulative timespan of the dataset. GIC as the client will ultimately determine if the accuracy of the measurement is suitable for their needs.

Security Measures –

The tool is intended to be open access, meaning that there will be no login necessary to access the application. Some of the security measures in the application concern performing functionality or attempting to operate the application that is not suitable at the time. These issues are handled by disabling and enabling controls only as the application is ready for the usage. This will prevent out of order sequencing ultimately resulting in the application crashing.

Product Health Monitoring –

The SPA will be a very simple application from the user perspective. The application products health monitoring will come from that user prompt that will occur on button clicks. For instance, if the user tries to upload a file even though a file has not been selected yet, the application will output “Select a file”. This kind of functionality is echoed throughout the application that acts as the product health monitoring.

Dashboard –

The SPA dashboard is a simple user-friendly way to interact with RNN model and make predictions on a stock. The dashboard consists of the following sections:

- Example output of what to expect from the application
- A file upload section
- A train model section
- A run prediction section

These components provide the necessary functionality to operate the SPA.

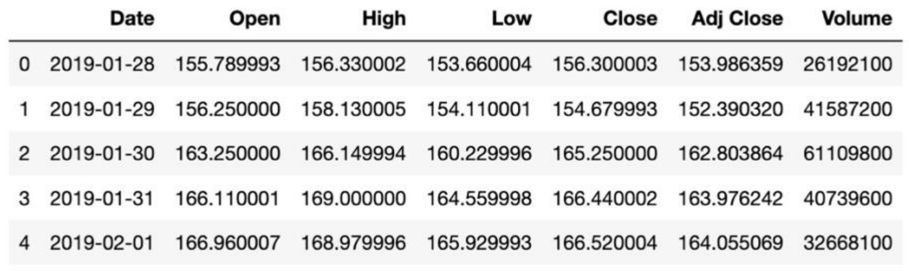
Stock Prediction Application – Prompt D

Project Purpose –

The purpose of the Stock Prediction Application (SPA) project was to provide Growing Investment Company (GIC) with a tool that offered a flexible, scalable, and easily interpretable predictive solution. The application was designed to predict a stock trend based on correlations to other datasets (stocks) and visualize the output. GIC's previous toolset was not meeting the company constraints and needed to be overhauled. The SPA offered a predictive solution that met GIC's needs and constraints.

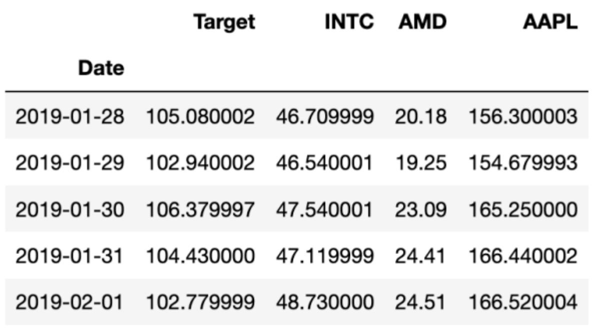
Datasets –

The raw datasets used for the SPA were historic data .csv downloads from Yahoo! Finance. These are the baseline datasets recommend for the application. When a file is uploaded to the application the raw is interpreted as:



	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-01-28	155.789993	156.330002	153.660004	156.300003	153.986359	26192100
1	2019-01-29	156.250000	158.130005	154.110001	154.679993	152.390320	41587200
2	2019-01-30	163.250000	166.149994	160.229996	165.250000	162.803864	61109800
3	2019-01-31	166.110001	169.000000	164.559998	166.440002	163.976242	40739600
4	2019-02-01	166.960007	168.979996	165.929993	166.520004	164.055069	32668100

The DataFrame above is a single dataset. The SPA is optimized when using multiple datasets. The application pulls out the "Close" column from each of the uploaded datasets and construct a collaborative "master" DataFrame with the pertinent features. When construct, the master DataFrame is interpreted as:

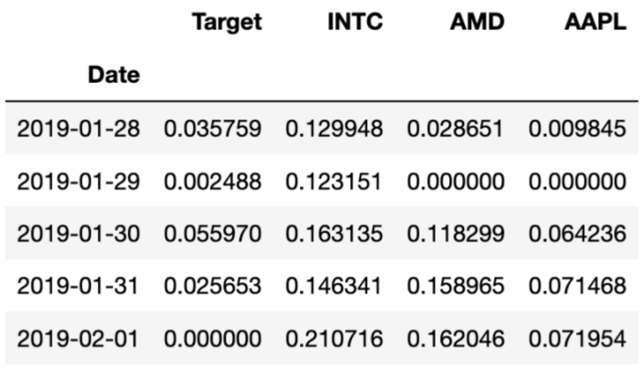


Date	Target	INTC	AMD	AAPL
2019-01-28	105.080002	46.709999	20.18	156.300003
2019-01-29	102.940002	46.540001	19.25	154.679993
2019-01-30	106.379997	47.540001	23.09	165.250000
2019-01-31	104.430000	47.119999	24.41	166.440002
2019-02-01	102.779999	48.730000	24.51	166.520004

The data is then scaled using the following function:

```
#drop all null values in the DataFrame
df.dropna(inplace = True)

'''Normalize and scale the data for each stock to avoid skewing the scale.
Programming scaling in this manner leaves the application
generic to handle other scales of data.'''

#Create a scaler for each of the features
def scale_data(stocks, dataframe):
    scalers = {}
    for i in stocks:
        scalers[i] = MinMaxScaler()
        scalers[i].fit(np.array(dataframe[i]).reshape((-1, 1)))
        dataframe[i] = scalers[i].transform(np.array(dataframe[i]).reshape((-1, 1)))

    return dataframe, scalers
```

The master DataFrame is then represented as:



Date	Target	INTC	AMD	AAPL
2019-01-28	0.035759	0.129948	0.028651	0.009845
2019-01-29	0.002488	0.123151	0.000000	0.000000
2019-01-30	0.055970	0.163135	0.118299	0.064236
2019-01-31	0.025653	0.146341	0.158965	0.071468
2019-02-01	0.000000	0.210716	0.162046	0.071954

Data Product Code –

The SPA uses the descriptive and non-descriptive method to analyze the dataset and provide descriptive outputs.

Implementation of the descriptive hierarchical clustering method is centered around the correlation of the all the of the datasets. Once the correlations are processed, the application with order them and visualize the results. The functionality of this is displayed in the following code:

```
def display_visual(corr):
    hmap = go.Figure(data=go.Heatmap(x=stocks, y=stocks,
                                     z=corr.values))
    hmap.update_layout(width=500, height=500, title="Correlation Heatmap")

    dendro = ff.create_dendrogram(corr.values, labels=stocks)
    dendro.update_layout(width=600, height=400, title="Correlation
Dendrograph")

    hmap.show()
    dendro.show()

build_df(select_target_drop.options)
df_corr = df.corr()
with output:
    display_visual(df_corr)
```

The other method that provided the predictive functionality for the application was wrapped inside the recurrent neural network (RNN) model. The following code snippet shows how the predictive functionality was accomplished within the application.

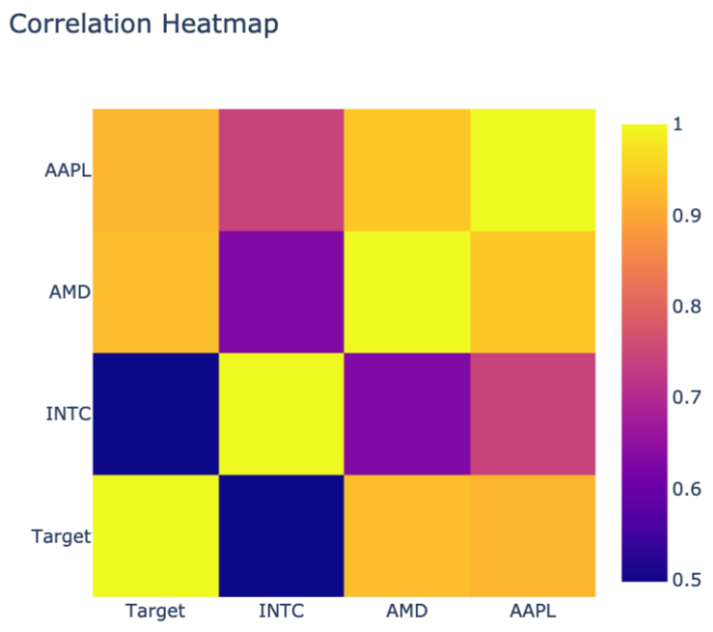
```
#Create the model
model = Sequential()

# Add layers to the model
model.add(LSTM(256, activation='relu', input_shape=x_train.shape[1:],
return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(128, activation='relu', return_sequences=True))
model.add(Dropout(0.2))

model.add(LSTM(64, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(n_feat))
model.compile(Adam(lr=.002), loss='mse')

for i in range(n_predict):
    predictions.append(model.predict(batch)[0])
    batch = np.append(batch[:, 1:, :], [[predictions[i]]], axis=1)
```

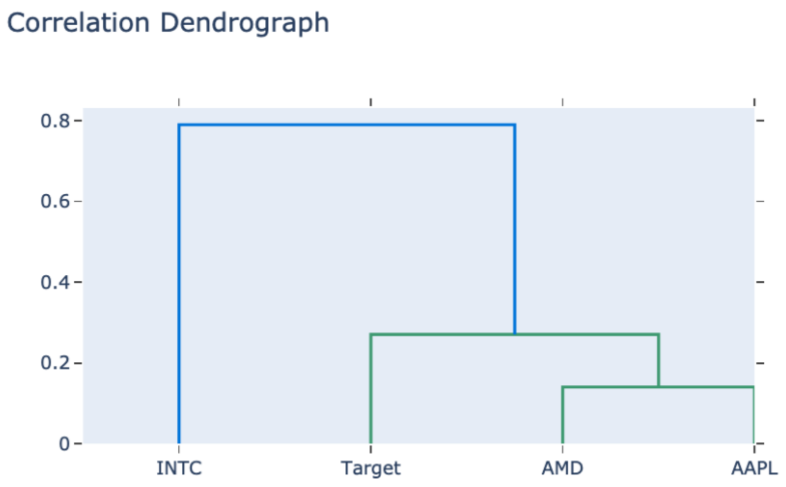
Hypothesis Verification –

Hypothesis verification of the application was approved by GIC. The SPA provides the functionality to the user to perform predictive methods. The results of the SPA will vary greatly based on the types and relations of the datasets input to the application. Because of this, the verification metric, root mean squared, is assessed on an individual measurement basis. Thus, GIC verified the hypothesis by validating the functionality of the application.

Effective Visualizations and Reporting –

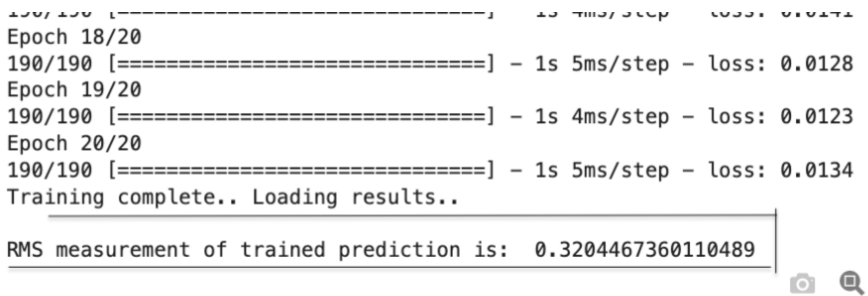
Visualizations used in the application enhance the user experience and provide clearer results. Specific formats of visualizations were chosen to provide active response and interactions with the user, such as display graph or chart values on scroll over. The Plotly visualizations that were used also include tool bars that enable the user to select specific portions the graph, download an image of the graph, as well as many other features. Below is an example of the descriptive method graphs output by the application:



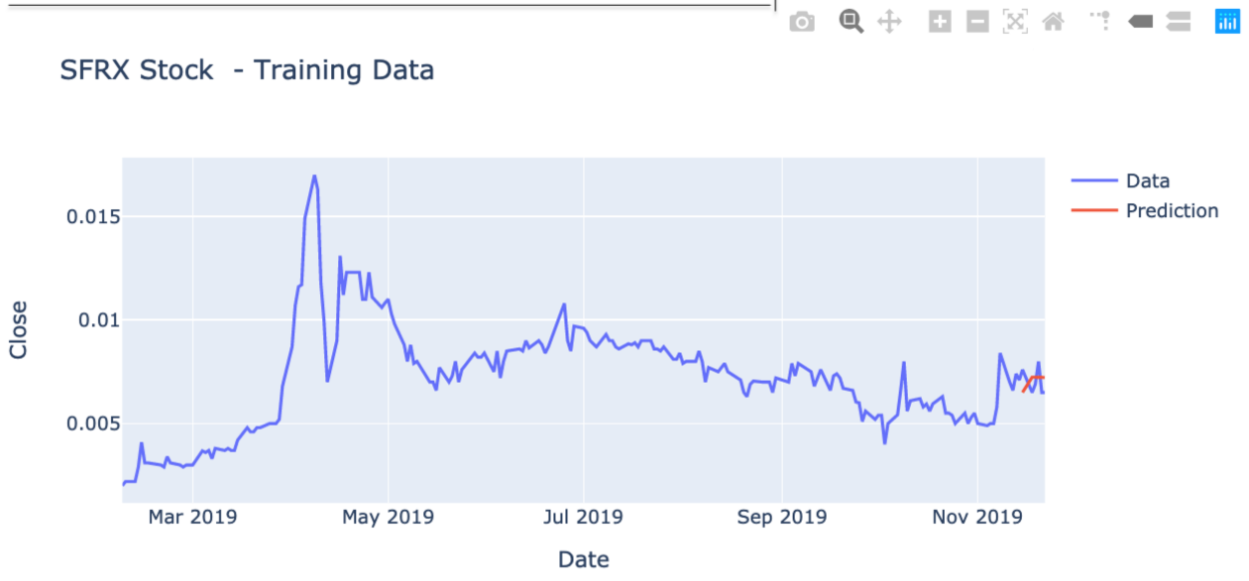


Accuracy analysis –

The root mean squared (RMS) formula is the accuracy metric used for the SPA. RMS is a statistical standard for measuring difference between predicted values and actual value. The RMS value is output at the end of the model training on the training dataset. An example output from the application is shown below:



```
Epoch 18/20
190/190 [==============================] - 1s 5ms/step - loss: 0.0128
Epoch 19/20
190/190 [==============================] - 1s 4ms/step - loss: 0.0123
Epoch 20/20
190/190 [==============================] - 1s 5ms/step - loss: 0.0134
Training complete.. Loading results..

RMS measurement of trained prediction is:  0.3204467360110489
```



Application Testing –

- Unit Testing – occurred continuously throughout the development of the application. Extensive unit testing was conduct before the development release of the SPA. Unit tests consisted of:
    - Processing various numbers of the datasets
    - Processing datasets in collective various lengths
    - Verifying the output against expected output
    - Verifying visualization displayed as expected
    - Verifying the application performed on different system and browsers.
- Regression Testing – occurred after reported bugs and additional features were reported from GIC. The regression tests consisted of validating the unit test still functioned as expected alongside the additional updates.
- System Testing – occurred before each release. The software would be built and validating the fully functional unit tests from a server. System testing was designed to be the final use case of the software before each release.

Application Files –

The application is built and serviced on Binder. The submission will include a file title Stock Prediction Application.txt. Inside this file is a link that will launch Binder and run the application without any user required installations, aside from an internet browser. If attempting to run the raw source code, the following Python packages are required to run the program (this is not recommended, as the installation process can be cumbersome):

- jupyter
- numpy
- ipywidgets
- pandas
- matplotlib
- plotly
- IPython
- scikit-learn
- keras
- voila

These packages cannot be included as a part of the submission because they are local to the machine or server the application is built on. The submission will also include: letter of transmittal, prompt documentation, a zip file with the following files.

- C964_Main_Jupyter.ipynb – this file will be just to ***run*** the raw source code in the proper environment (Jupyter Notebooks)
- C964_Main_Jupyter.py – this will be the file to ***view*** the source code.
- Environment.yml – this file can be used to install the conda environment, provided the Anaconda Python package manager is installed on the machine.
- A Sample_Data folder that will contain example datasets the user can run through the application.

For running the application, it is highly recommended to use the link provided in the Stock Prediction Application.txt file. The application is already built and there are no required downloads to run the software with this method.

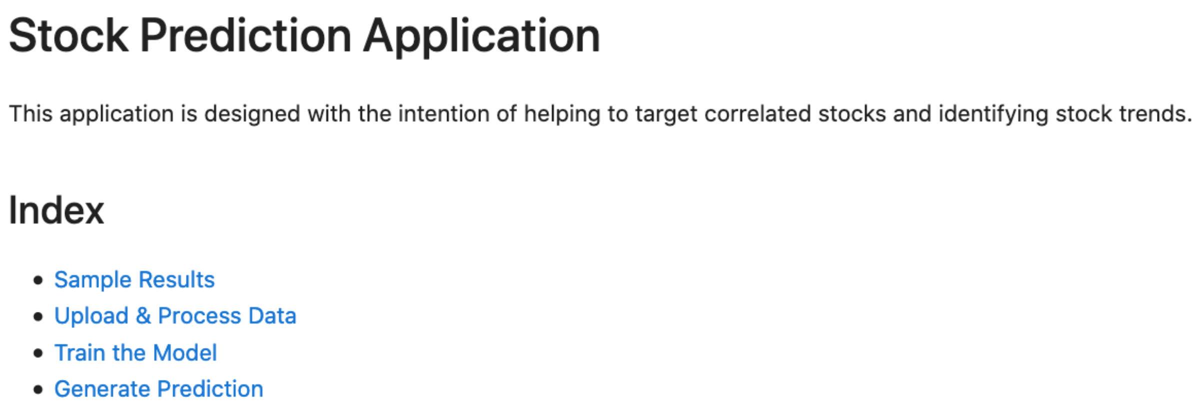
User's Guide –

Note* - because the application is being built directly from the repository via Binder, user interaction objects (buttons, dropdowns, etc.) can take a while to render when application is launched. User interactions such as button click may have a lagging response as well.

The intended order of user experience is:

- Load the application
- View the sample data output and understand what kind visualizations to expect
- Upload the datasets
- Process the data
  - Examine the data visualization
- Train the model
  - Note the RMS output value
  - Examine the data visualization
- Run the actual five-day prediction
  - Examine the data visualization

The application will open displaying the following at the top of the screen:



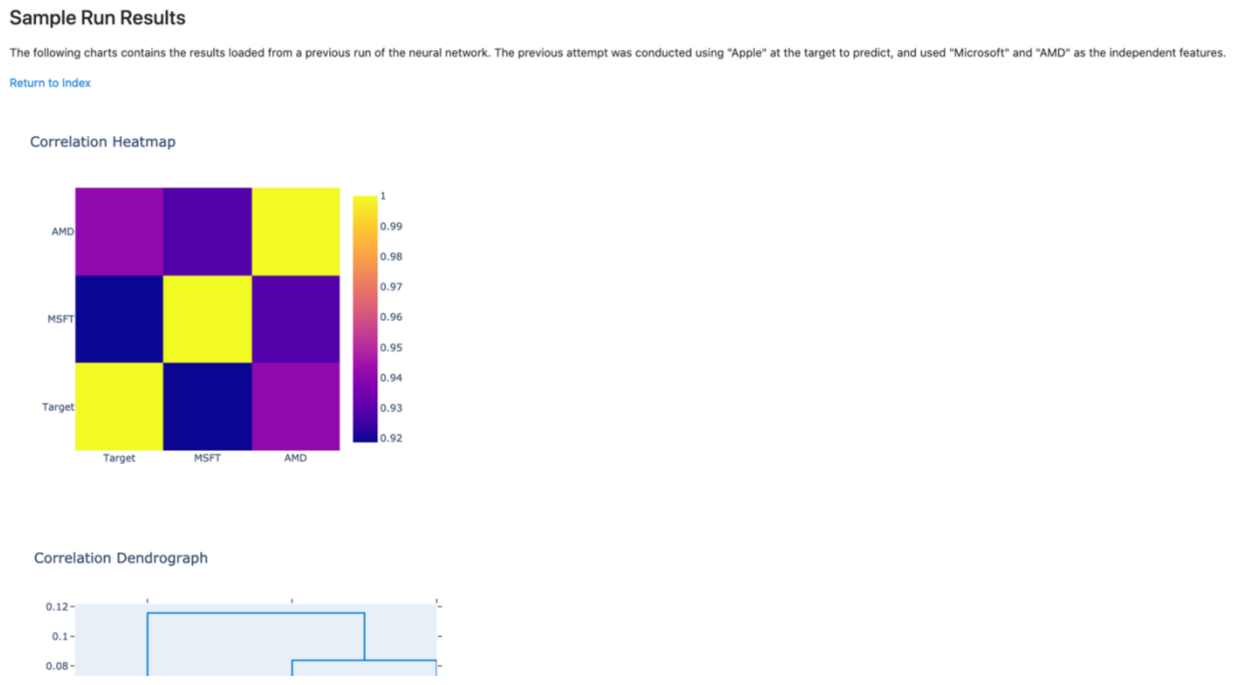
Stock Prediction Application

This application is designed with the intention of helping to target correlated stocks and identifying stock trends.

Index

- The index shown above will help the user navigate the web application.

- When the user scrolls down they would see the “Sample Run Results” section. This part of the application illustrates what kind of output the user should expect during operation. A part of this section is show in the image below:



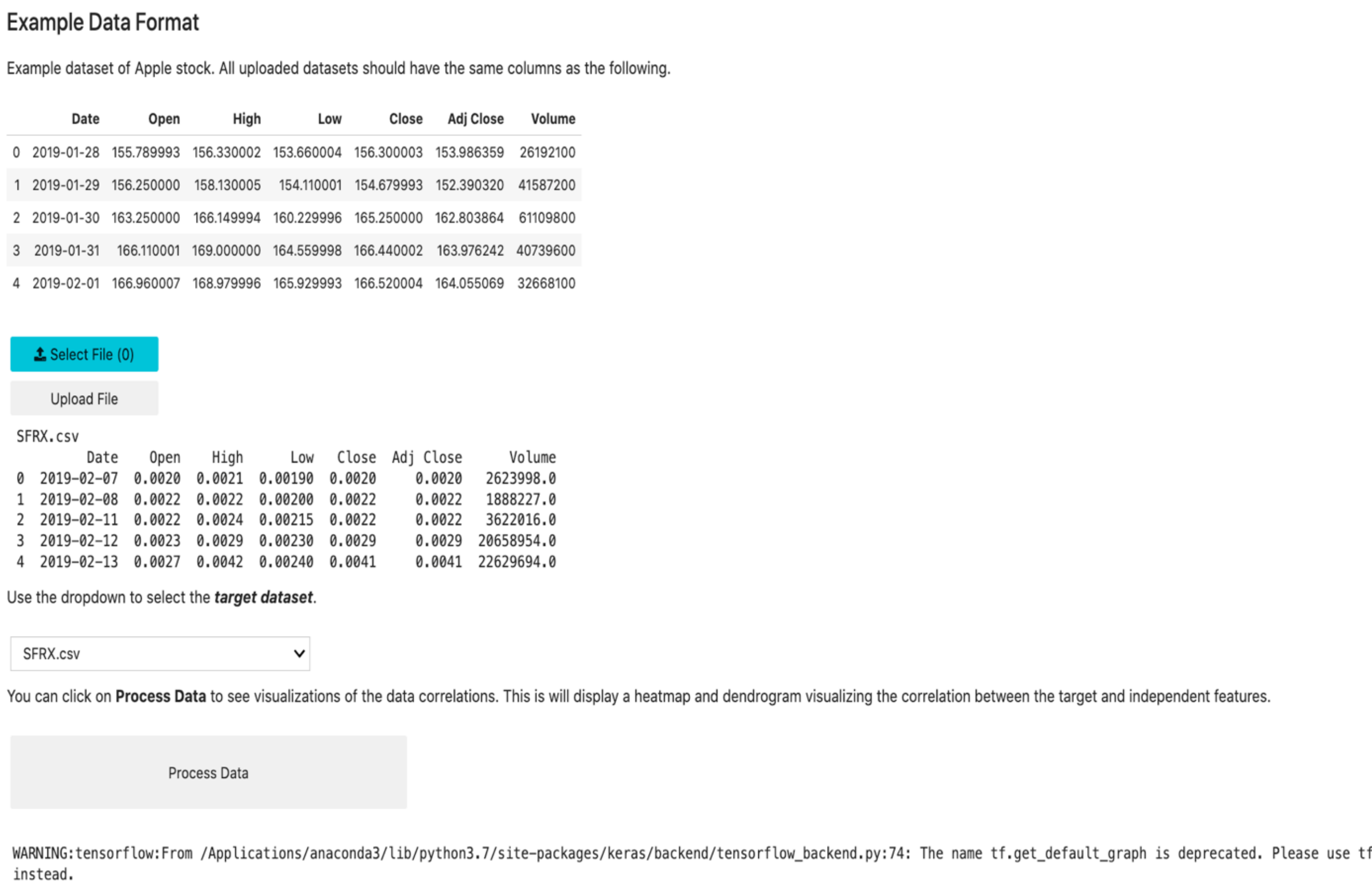
Sample Run Results

The following charts contains the results loaded from a previous run of the neural network. The previous attempt was conducted using "Apple" at the target to predict, and used "Microsoft" and "AMD" as the independent features.

Return to Index

- After the “Same Run Results” section the user will come to the “User Operation” portion of the application.

- The first section within the user operation is “Loading the data set(s)” in the section of the application the user will upload their data, this is done in the following steps:
    - **a. Click “Select File (0)”**
        - **i.** A file browser will appear where the user can navigate to their .csv dataset, select it, and click open in the bottom right of the file browser pane.
    - **b. Click “Upload file”**
        - **i.** The first five rows of data uploaded with display on the screen to validate the data has been loaded by the application.
    - **c.** Repeat steps **a** and **b** for as many datasets required.
    - **d.** Once the datasets are loaded the user will **select from the dropdown** the “target dataset”. The target data set is the stock that the user wants to predict.
    - **e.** After the target data set is selected, the user will click **“Process Data.”**
        - **i.** If the user has uploaded more than two data sets, the application will perform correlations and hierarchical clustering on the datasets.
        - **ii.** If the user has only uploaded one dataset, the application will display a message to the user that “correlations cannot be calculated on less than two datasets”
        - **iii.** If there are no datasets loaded in the application, the application prompt the user saying that “there is no dataset loaded in the application, please upload a dataset”
        - **iv.** There will be a warning displaying from TensorFlow at the bottom of this section, there are no ways to avoid this error from the API.



Example Data Format

Example dataset of Apple stock. All uploaded datasets should have the same columns as the following.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2019-01-28	155.789993	156.330002	153.660004	156.300003	153.986359	26192100
1	2019-01-29	156.250000	158.130005	154.110001	154.679993	152.390320	41587200
2	2019-01-30	163.250000	166.149994	160.229996	165.250000	162.803864	61109800
3	2019-01-31	166.110001	169.000000	164.559998	166.440002	163.976242	40739600
4	2019-02-01	166.960007	168.979996	165.929993	166.520004	164.055069	32668100

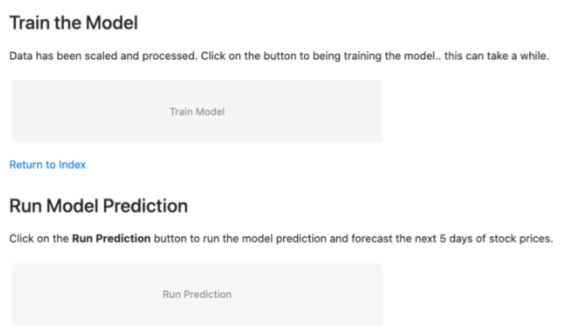
```
SFRX.csv
         Date    Open    High      Low   Close  Adj Close       Volume
0  2019-02-07  0.0020  0.0021  0.00190  0.0020     0.0020    2623998.0
1  2019-02-08  0.0022  0.0022  0.00200  0.0022     0.0022    1888227.0
2  2019-02-11  0.0022  0.0024  0.00215  0.0022     0.0022    3622016.0
3  2019-02-12  0.0023  0.0029  0.00230  0.0029     0.0029   20658954.0
4  2019-02-13  0.0027  0.0042  0.00240  0.0041     0.0041   22629694.0
```

Use the dropdown to select the ***target dataset***.

You can click on **Process Data** to see visualizations of the data correlations. This is will display a heatmap and dendrogram visualizing the correlation between the target and independent features.

```
WARNING:tensorflow:From /Applications/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:74: The name tf.get_default_graph is deprecated. Please use tf
instead.
```

- The next section of the application is the “Train the Model.” In this section the application RNN model will being training and learning on the training dataset.
    a. **Click the “Train Model” button** as listed below.
        - The model will begin training on the data, and will provide an output of “Training… Please wait, this can take a while...” TensorFlow warning messages will also be display as they are a standard part of the API and can’t be removed. After this, the model will display its progress as it iterates over the epochs. The RMS value will be displayed after the training is complete. The output of this section will conclude with a linear graph of the results.

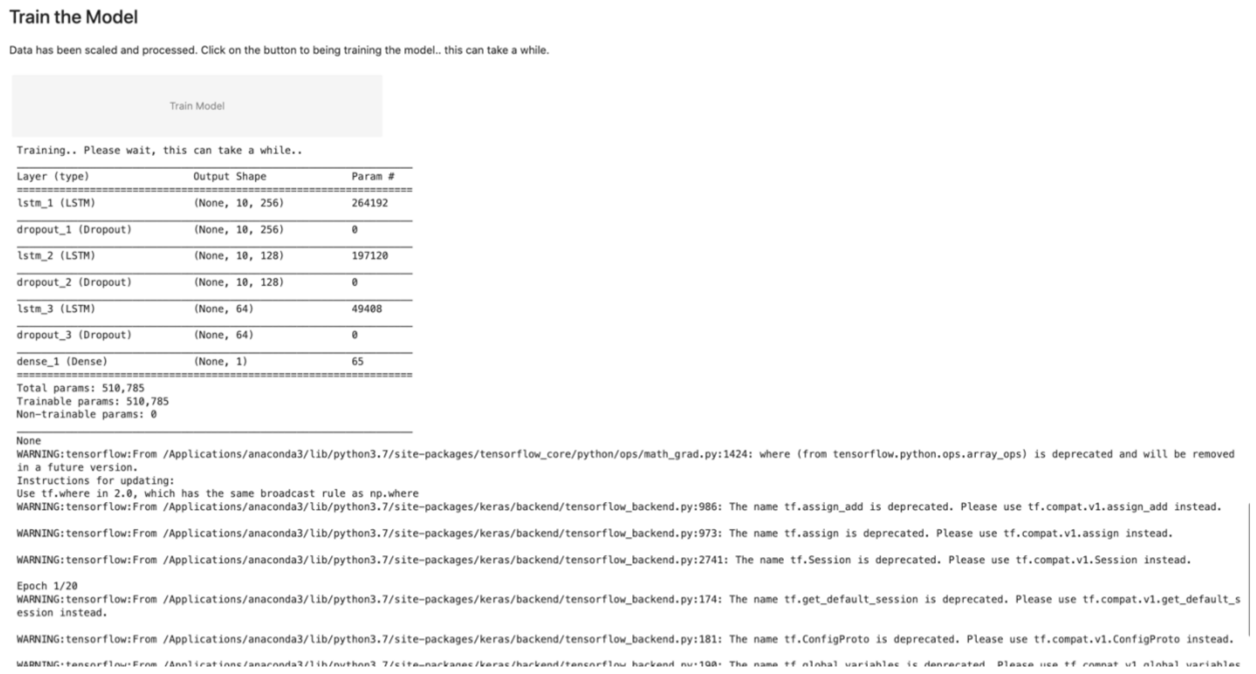


Train the Model

Data has been scaled and processed. Click on the button to being training the model.. this can take a while.

Run Model Prediction

Click on the **Run Prediction** button to run the model prediction and forecast the next 5 days of stock prices.



Train the Model

Data has been scaled and processed. Click on the button to being training the model.. this can take a while.

```
Training.. Please wait, this can take a while..
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 10, 256)           264192
_________________________________________________________________
dropout_1 (Dropout)          (None, 10, 256)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 10, 128)           197120
_________________________________________________________________
dropout_2 (Dropout)          (None, 10, 128)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                49408
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 510,785
Trainable params: 510,785
Non-trainable params: 0
_________________________________________________________________
None
WARNING:tensorflow:From /Applications/anaconda3/lib/python3.7/site-packages/tensorflow_core/python/ops/math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is deprecated and will be removed
in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /Applications/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:986: The name tf.assign_add is deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /Applications/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:973: The name tf.assign is deprecated. Please use tf.compat.v1.assign instead.

WARNING:tensorflow:From /Applications/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:2741: The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

Epoch 1/20
WARNING:tensorflow:From /Applications/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:174: The name tf.get_default_session is deprecated. Please use tf.compat.v1.get_default_s
ession instead.

WARNING:tensorflow:From /Applications/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:181: The name tf.ConfigProto is deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /Applications/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:190: The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables
```

- The final section of the application is the "Run Model Prediction" section. In this section the model will train on the entire dataset and make its 5-day stock prediction.
  a. **Click the "Run Prediction" button.**
     - The application will output the model training and learning from the complete dataset over the 20 epochs. The output will complete with a linear graph of the actual stock and the stock prediction.

Summation of Learning Experience –

The process of the developing this application was extremely difficult. Having very little experience with neural networks or machine learning, it felt that picking a topic in this field was initially too big and undertaking. However, after devoting many hours into reading the API documentation on how these concepts function, I began to understand the concepts and develop confidence in my ability. After seeing breakthrough, after breakthrough I finally had developed some confidence in the application. This learning experience more than any other, has taught me a discipline and diligence that I was unaware I was capable of. I will take these lessons with me throughout my career and life.

Section E

Sources Cited :

TensorFlow. 2020. *Time series forecasting.* time_series [digital image]. Retrieved from https://www.tensorflow.org/tutorials/structured_data/time_series

Wikipedia. 2020. *Root Mean Squared.* RMS [digital image]. Retrieved from https://en.wikipedia.org/wiki/Root_mean_square