# FRE-GY 7043 Capstone Report
# UBS Stock Recommender System
**Fall 2021**

**Chris Abruzzo, Zixi Pan, Jessie Tao, Shan Zhang**
**Supervisor: Norman Niemer**

## Contents

## Exec Summary

Tech companies such as Amazon, Netflix and Facebook have been using recommender systems to greatly increase user engagement and satisfaction. In contrast, most financial services are not very personalized and therefore increasingly difficult to sell to customers who expect such personalization. This is true for end clients and internal clients such as portfolio managers and investment analysts. This paper explores the development of a recommendation system for portfolio managers regarding buying and selling their stock positions. The model takes an input of the investor’s stock portfolio, analyzes the holdings positions, and suggests 10 stocks to buy and 10 stocks to sell. The model is able to learn from the holdings what financial ratios the portfolio manager favors and build the recommendations. Each recommendation is paired with an intuitive explanation for how the preference score was generated based on our model. This increases user confidence and trust in the model. Lastly, all these functions were implemented into a user-friendly web app to provide quick, and accurate recommendations via an interactive, easy-to-follow platform.

## Introduction

Nowadays, recommender systems are almost everywhere in our life. When watching YouTube, the homepage shows different sections of videos based on the previous watch history. Also, Netflix and Amazon have similar systems for recommending content that users may find interesting. Such recommender systems save users time when browsing, and companies can sell their products more efficiently.

Similarly, it is hard for individual investors or portfolio managers to pick a few outstanding stocks from thousands of options in the markets. Screeners exist for financial ratios but fail to be implemented in a personalized, easy-to-use manner. Meanwhile, current existing stock recommender systems mostly focus on historical returns or are based on stock price predictions, which normally recommend a set of similar stocks to all users. This paper tackles a more personalized approach by applying Machine Learning algorithms to the stock recommender system to provide investors personalized recommendations.

However, a downside of using these powerful algorithms is losing the ability to easily explain the recommendations with a certain level of trust and confidence, especially for users not familiar with the technical knowledge of machine learning algorithms. To tackle this challenge, the SHAP library was used to intuitively explain how each recommendation gained its preference score to Buy or Sell.

Lastly, a major driving task for this paper was to implement the model seamlessly to an end-user. Utilizing the platform Streamlit, a web app was developed for ease of use. This app creates a tool that mirrors the intuitive platforms of Netflix and Amazon, while still implementing powerful algorithms. Additionally, the app provides compelling information that builds trust and confidence in what the application is providing for an investor.

## Data Exploration

### *I. Data Description*

The provided dataset has a total of 1240 stocks. 32 stocks are marked as ‘1’ to indicate that they are owned by the portfolio manager. At the same time, the remaining 1208 are labeled as ‘0’ to indicate stocks that are in the watchlist of the portfolio manager (or general markets). Each stock has 30 features representing financial ratios and the ratios compared to the industry benchmarks. For example, the ratios include the Capex to depreciation ratio, dividend payout ratio, free cash flow conversion cycle, return on assets, return on equity, etc. All of these 30 features have also been normalized before use.

First, the dataset was split into training and testing sets. The training set was used to explore the data, train the model, and measure the predictive power of out-of-sample examples.

This dataset was specifically used to explore the impact of the financial ratios had on an investor's preference based on the current holdings within the portfolio.

## *II. Imbalanced Data*

The first problem encountered with the dataset was the imbalanced issue. With most of the stocks being marked as not held, the algorithm would struggle to understand the investor's preferences and determine which stocks should be recommended. There would be an overwhelming emphasis on the stocks that are not owned, resulting in the algorithm to produce poor preference scores for all of the buy recommendations. Specifically, the 1208 stocks not currently held would mask any valuable information provided by the 32 stocks that are held. The model would be overfit to the majority class and would never recommend buying a stock. The bar plot below shows the imbalanced data.

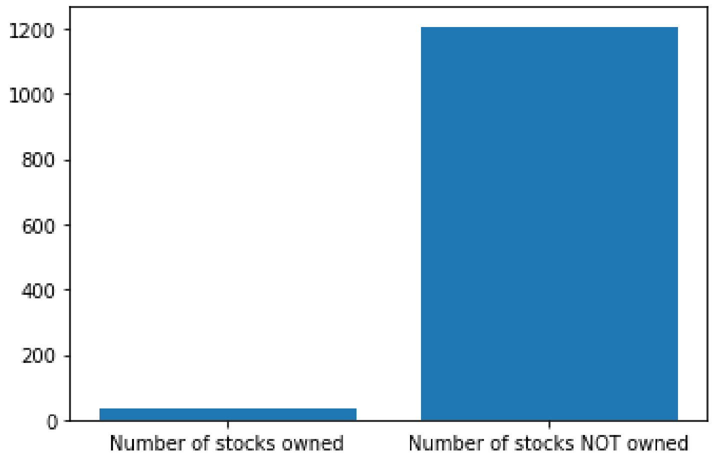


**Figure 1.** Bar plot of classes to emphasis imbalanced data

To solve this problem, we implemented three different methods: oversampling, under-sampling, and class weight balance. Both oversampling and under sampling involve introducing a bias to select more samples from one class than from another, to compensate for an imbalance that is present in the data. Oversampling includes selecting random examples from the minority class with replacement and supplementing the training data with multiple copies of this instance, hence it is possible that a single instance may be selected multiple times. This solves the imbalance issue by adding more emphasis to the minority class. Under sampling is the opposite to oversampling. This method seeks to randomly select and remove samples from the majority class,  reducing the number of examples in the majority class and solving the imbalanced problem. Lastly, the class weight balance considers different weights to both the majority and minority classes. The purpose of doing so is to penalize the misclassification made by the minority class by setting a higher-class weight and, at the same time, reducing the weight for the majority class.

The class weight balance was selected from these three options because of the benefit of using less computational power than over and under-sampling. Additionally, the data set is not transformed, which helps prevent against certain biases by using samples more than once or removing them all together.

## *III. Data Exploration*

Once the imbalance problem was solved, a deeper look was taken into the thirty features for each sample. With plotting the features, no clear trends can be noticed. First, each feature was plotted against its industry benchmark. This can be seen in the following Figure.

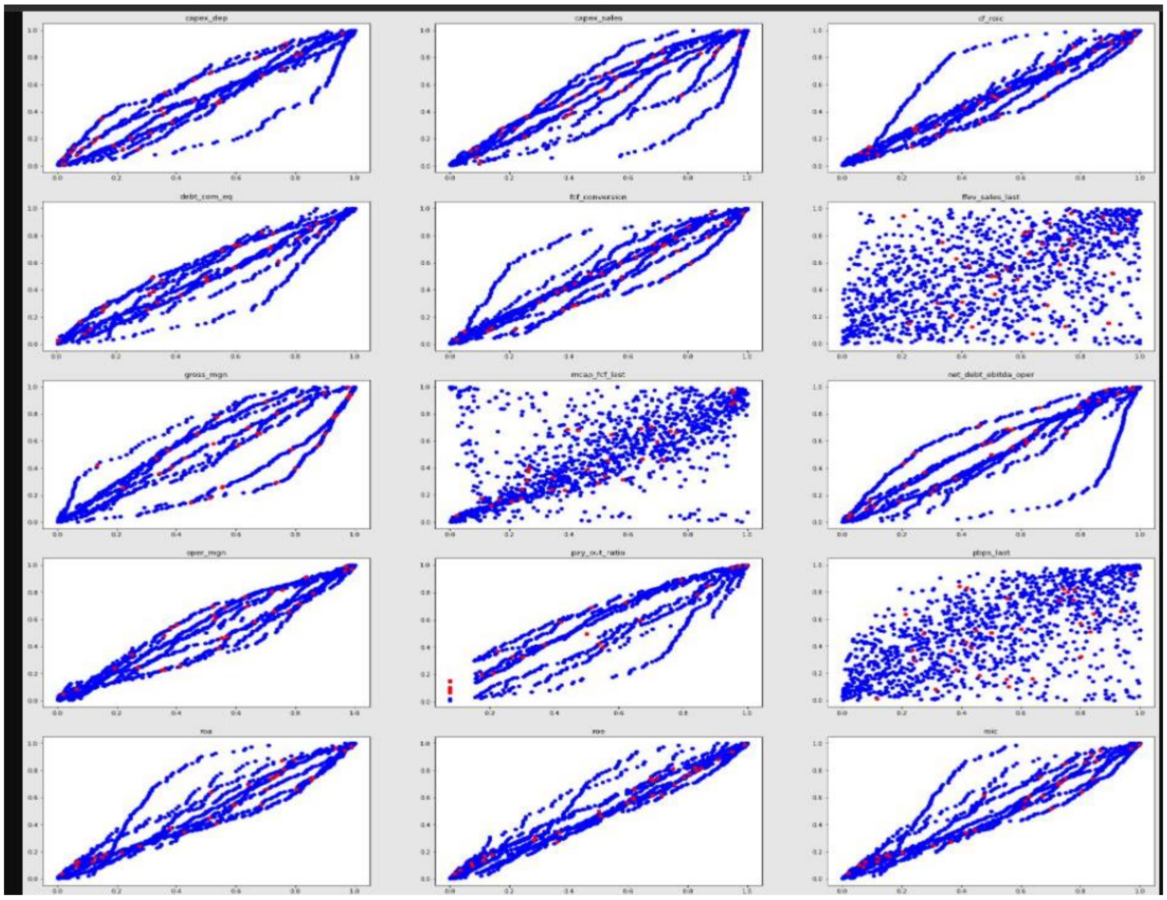


**Figure 2.** Plots of the features vs industry benchmarks of same feature. Blue data points are not held, while red dots are held by the portfolio manager.

From these plots, it can be seen that a non-linear machine learning algorithm will be beneficial in exploring the relationships between the features and what stocks are owned by the portfolio manager. The only clear pattern from this figure is that the portfolio manager likes to diversify their investments according to these financial ratios. Next, each stock was plotted to see its relationship between the portfolio manager's ownership and each of the features. The relationship is shown in Figure 3 below.

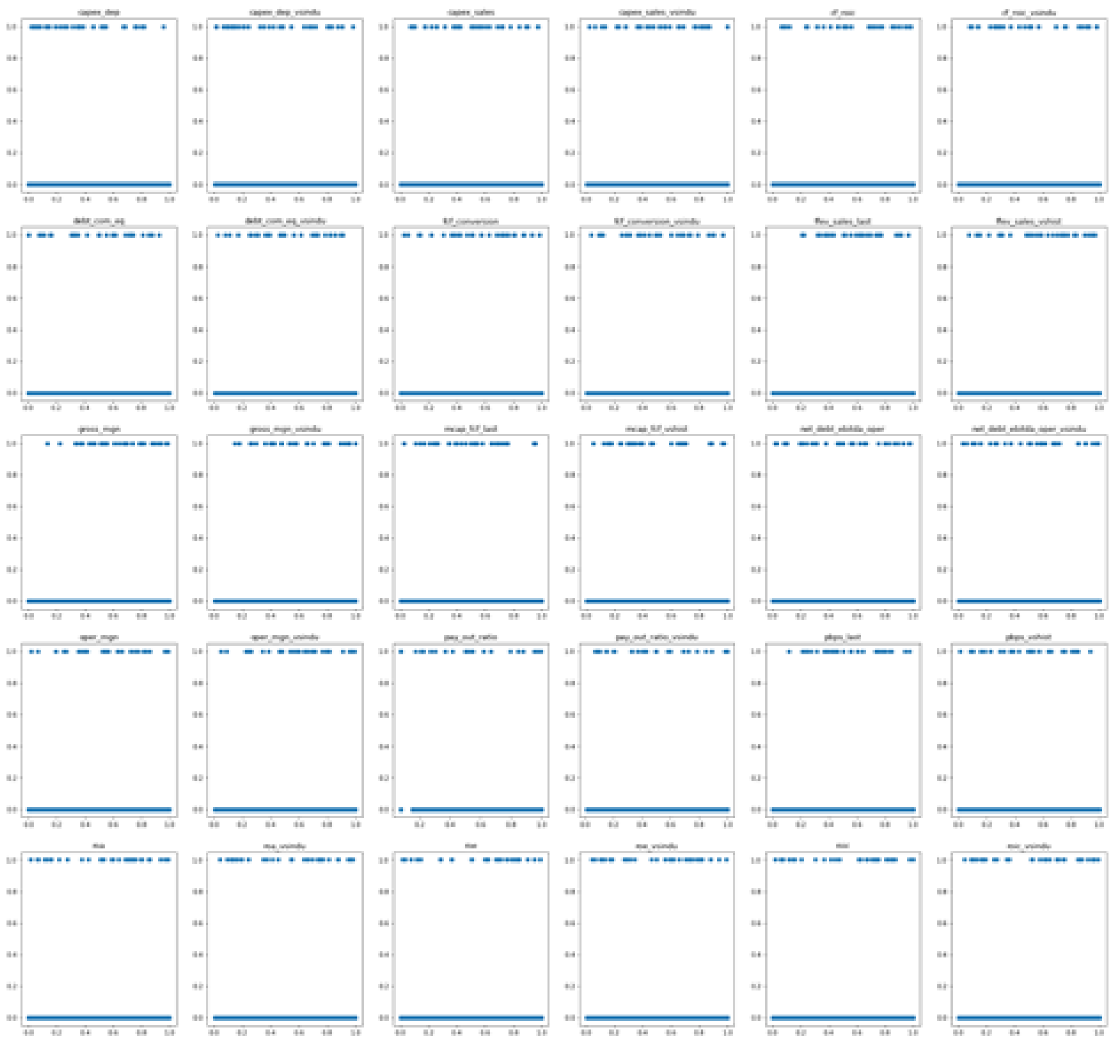


**Figure 3.** Class (owned vs not owned) vs each of the 30 features.

For the graphs above, the x-axis represents the value of each feature, and the y-axis represents whether the stock is in the portfolio. Most of the features have almost no linear relation between its value and whether it's owned by the investor. And we can clearly see that it's highly not applicable to fit logistic regression into the graph. Thus, we want to use weak learners and support vector machines to handle the classification issues.

## Methodology

This paper focuses on two major deliverables: the recommender machine learning model and the web app to implement the model in a user-friendly interface. The recommender model uses the dataset which contains the portfolio manager's holdings and the 1,000-stock universe to suggest 10 stocks to be bought and 10 stocks to be sold. For these recommendations, the investor will only be interested in buying stocks that they don't already own and interested in only selling 10 stocks that they do own. Additionally, the model needs to explain each recommendation and convince the investor that they should act on said recommendation. In order to create personalized recommendations unique to the portfolio manager, the model will utilize machine learning algorithms to identify the key features that drive the investor to own a stock.

First, the recommendation model was set up as a classification problem. The stocks are each assigned with a predictable outcome of "1" to own, or "0" to not own. Each stock will get a probability score assigned to them that is between 0 and 1. The closer the score is to one, the more likely this stock is to be owned by the investor. Throughout the paper, these scores, referred to as Preference Score, were used as the ranking system for the recommendations. The highest scores assigned to stocks that weren't already owned were recommended to buy and the lowest scores assigned to those already owned were recommended to sell.

Next, the machine learning algorithm needed to be selected. In order to choose the algorithm that best captured the preferences of the investor, an accuracy scoring system was established. First, the data was split into train and test sets. The in-and out-of-sample performance of the models were measured and compared. The better-performing models were selected. Second, in order to confirm that the models were picking up the preferences of the investors, an accuracy score was established to narrow down the selection of the models further. This accuracy score measured how many stocks designated with a "1" target were ranked within the top 50 stocks according to their prediction score. The logic behind this was that the more stocks already owned in the top 50 showed that the algorithm was successfully capturing the preferences of the portfolio manager.

In this project, we explored different ML models. The models will include Support Vector Classifiers, Random Forest, Logistic Regression, Gradient Boosting. We will utilize cross-validation, accuracy scores, ROC AUC scores, and confusion matrix to compare all models.

Next, the recommendations needed to be explained and justified to the investor. This was done by exploring the features that drove the prediction scores. Each financial ratio's impact was analyzed by identifying its importance level (magnitude on the preference score) and in which direction the investor preferred (preferred the ratio to be smaller or larger). This was done by utilizing the SHAP library, which associates Shapley values with each feature. These Shapley values each contribute to the preference score and will drive a recommendation based on its value and direction. For example, if the investor preferred stocks with a high gross margin ratio, and the stock being analyzed has a large gross margin ratio, the Shap value would be both positive in value and have a large contribution to the preference score. For each recommendation, the Shap scores for each feature were used to show why the stock was a recommended buy or sell.

Lastly, the web app was developed through a design sprint on a weekly basis. A fully interactive app was developed to allow investors to upload their portfolios, receive a basket of recommendations based on their choice of numbers and visual explanations of each recommendation. This app is user-friendly to any non-technical user with a fully functional interface. The app was prototyped by using Jupyter Notebooks and developed as a final product using the Streamlit platform. Adjustments recommended by peers and project leads were made each week until the final product was completed and delivered.

## Models

After the initial testing and data exploration, focus was stressed on two different classifier algorithms, Support Vector Machine (SVM) and Random Forest (RF). They are commonly used when classifying data with multiple features since they capture the differences between each category in detail and they are good at finding connections between different features.

### *I. Support Vector Machine*

SVM is a powerful algorithm that is used widely in categorizing datasets, and it’s good at dealing with targets with multiple features. This algorithm looks into the dataset and measures the distances between units in the two classification categories. Then, it fits a line between the units in the two categories such that each category’s unit falls on one side of the line. The distances between the units and the line are then maximized. The line can be straight, circular, swirl, spiral etc. The shape can be modified by changing the parameter called ‘kernel’.

When applying this algorithm to the dataset, it was found that the results were extremely accurate. This algorithm is overfitting the desired dataset, which results in poor predictive power. There are 32 stocks having the “target” equals 1, and the model predicts 31 of them correctly, but it turns out that all other stocks are having extremely low and similar prediction scores. It means that it’s hard to pick 10 stocks outside of the portfolio that have outstanding scores. And since all 31 stocks are having extremely high scores, it’s also really hard to pick 10 stocks that’s performing relatively worse. This algorithm isn’t providing any additional information outside of that the portfolio manager prefers the stocks they already own. The resulting scores from SVM can be seen in the Figure below.

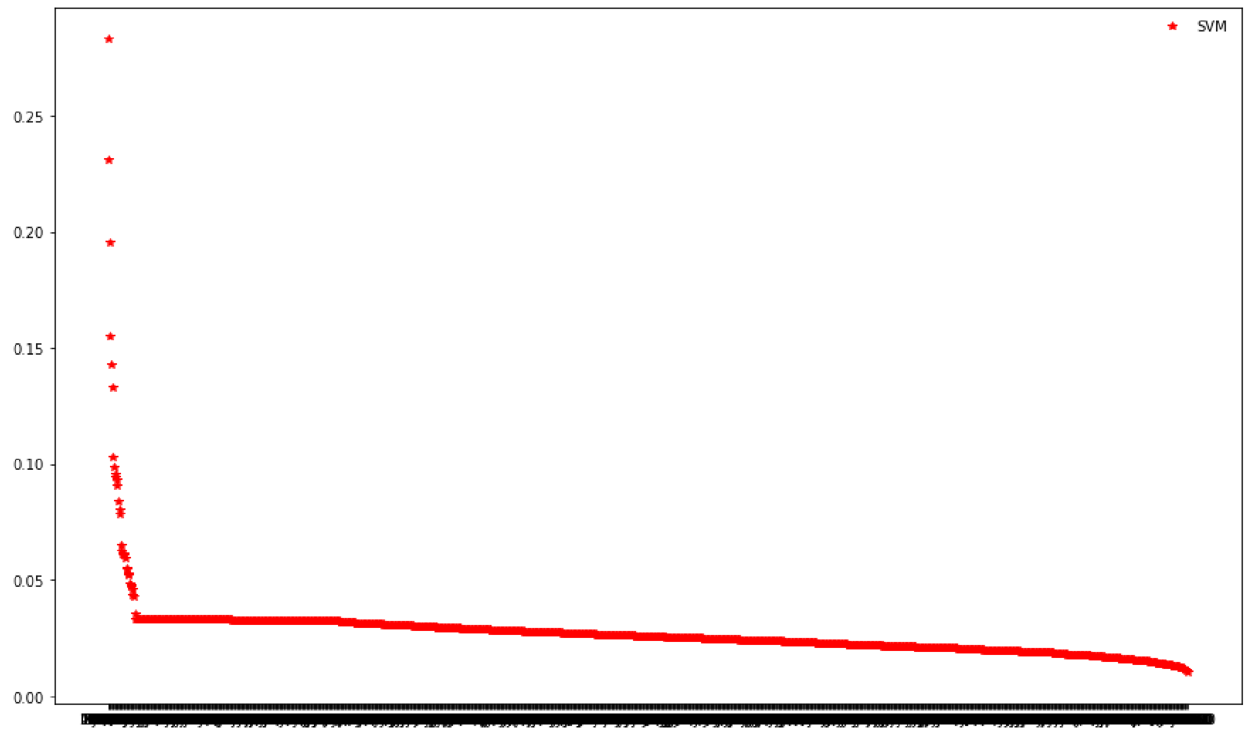


**Figure 4.** Preference Score vs Individual Stock in the dataset

We can observe from the graph, besides a few stocks that have *relatively* high scores, most of the stocks end up having really close results. To deal with this issue, more weight was given to the minority ‘owned’ class, but the results did not improve. Since the goal of the model is to interpret the influence of each of the features on the preference score, the next algorithm was explored.

## *II. Decision Trees*

Next, the gradient boosting and random forest algorithms were applied to the dataset. Both models used decision trees to classify units into different categories by asking multiple different questions. Here, ‘questions’ are used to study the values of each feature of the stocks and divide them into different categories based on a split.

Gradient boosting studies the loss function of decision trees and improves it with additional decision trees. A loss function is a function that’s used to measure the accuracy of the results within the decision tree. So, when gradient boosting is used to study the dataset, it will first use a decision tree to produce a result. Then, it will measure the loss and build another decision tree to optimize the loss. The model repeats this process until the loss function is minimized, resulting in the best fit model. It is similar to stacking each of the decision trees to build one massive tree with great height.

Additionally, the random forest algorithm was used on the dataset. This algorithm, on the other hand, does not study the loss function from the decision tree. Instead, it randomly selects a part of the input data to build the decision tree each time. This process is repeated which creates "forest". In the forest, there are many decision trees, each built with different datasets that's randomly selected from the original dataset. So, the results produced by each tree should be slightly different. Then, the model will let each tree vote to decide the final result.

Both Gradient Boosting and RF performed well in deciding which stocks were owned, and which were not. In the end of the analysis, RF was chosen because it avoided the overfitting problem seen with SVM and is less computationally taxing. In the Figure below, it can be seen how the scores produced by RF are distributed more smoothly, resulting in a more convincing preference score across the various stocks.

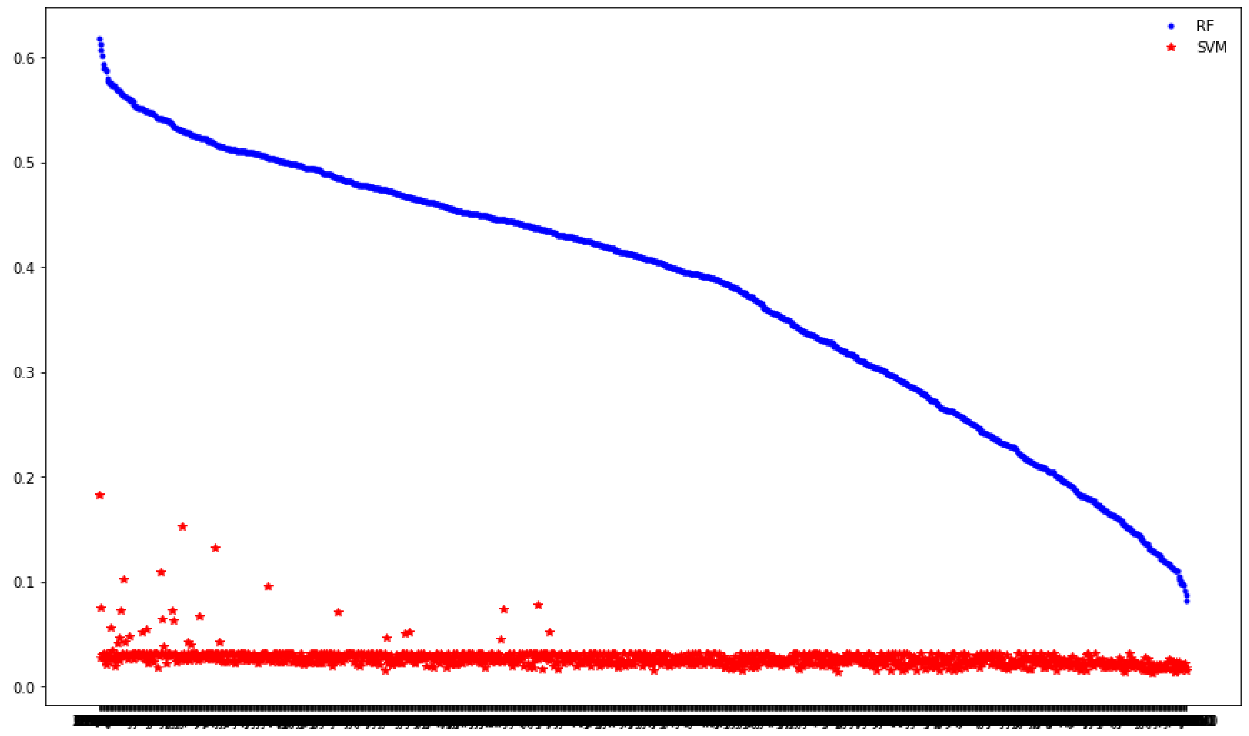


**Figure 5.** Random Forest and SVM preference scores for each stock sample.

## *III. SHAP*

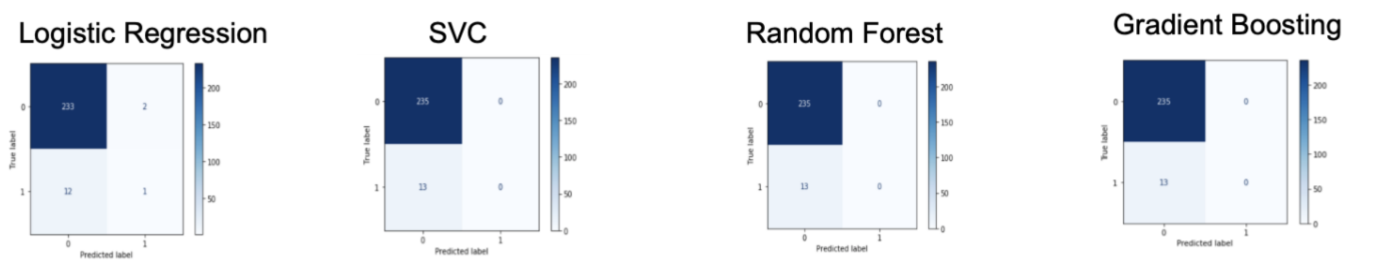
Lastly, SHAP is used to explain how the preference scores are generated for the model as a whole, as well as each individual recommendation. It is a package developed by Lundberg and Lee, which computes Shapley values from coalitional game theory to explain the contribution that each of the features have on the prediction (preference score for the recommender). The feature values of a data sample act as players in a coalition. Shapley values tell us how to fairly distribute the "payout" (the prediction/preference score) among the features. For example, the package will take one stock, analyze how much each of the financial ratios contribute to the preference score, and will rank each of the ratios by importance as well as magnitude and direction of each of the ratios. Using SHAP, we can observe and find the features that had the most influence on each of the results in a way that is intuitive to a non-technical user.

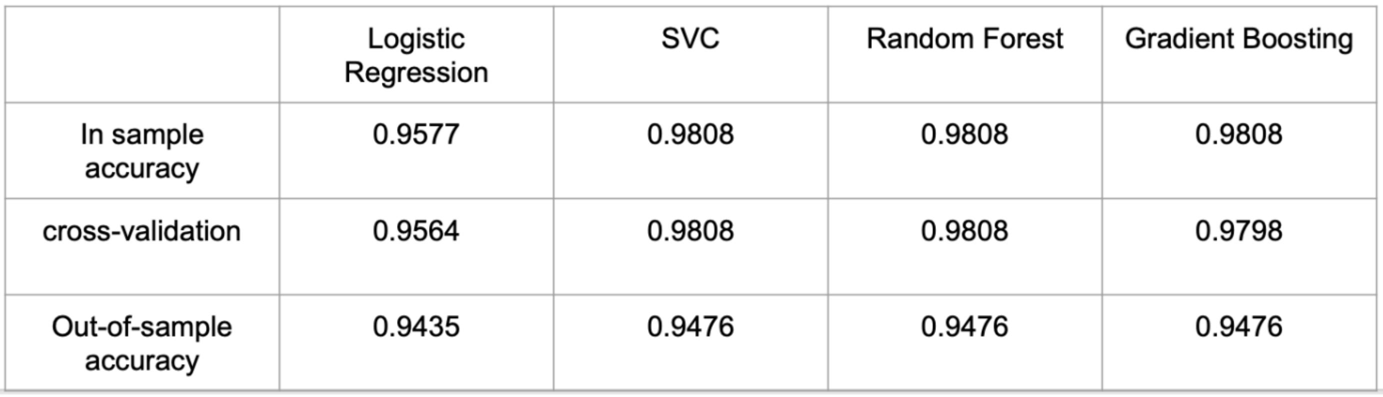
For the RF model, the TreeSHAP function within the package was used. This function was developed to explain predictions from decision tree models such as random forest and gradient boosting. It uses conditional expectation to come up with the result. For a decision tree, TreeSHAP will first look at the outputs of the trees and calculate the average of the output to be the default value. Then, it will add in the features one by one to see how each feature impacts the result. When a feature is added into the prediction of the preference score, the value of each feature is randomly selected. This is repeated for all the possible coalitions in the dataset and the average contribution (contribution = difference between average and new result with added feature value) will become the Shapley value. This value is then used to show the impact of each feature on the preference scores.

## Results and conclusions

### *I. Model selection*

The following figure shows the results from testing the various algorithms on the data set. The final model selected was the Random Forest due to the better accuracy and ability to handle overfitting.





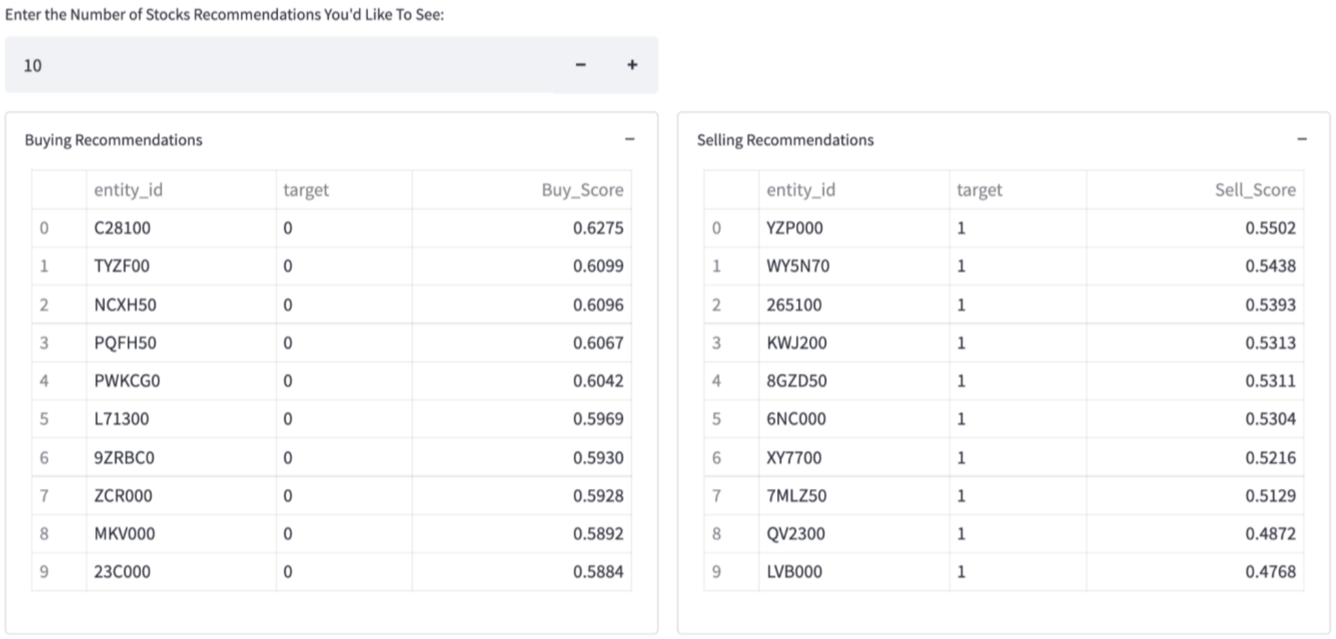
| | Logistic Regression | SVC | Random Forest | Gradient Boosting |
|---|---|---|---|---|
| In sample accuracy | 0.9577 | 0.9808 | 0.9808 | 0.9808 |
| cross-validation | 0.9564 | 0.9808 | 0.9808 | 0.9798 |
| Out-of-sample accuracy | 0.9435 | 0.9476 | 0.9476 | 0.9476 |

**Figure 6.** Results from in- and out-of-sample testing on the various algorithms. Results include confusion matrix, cross-validation, and accuracy scores.

## *II. Buy and Sell Recommendations*

The recommendations are sorted by the scores provided by the model's algorithm. As noted earlier, the Buy Recommendations are for stocks that are NOT currently owned by the portfolio manager, while the Sell Recommendations are for stocks that are currently owned. This system is purely for long investing, no short selling is recommended.

The "Buy_Score" represents the probability that a stock has target equals 1 and the "Sell_Score" represents the probability that a stock has target equals 0. We ranked both scores and give the recommendations for buying and selling. The figure below shows a screenshot from the web app and how the recommendations are presented to the user. The higher the "Buy_Score"  the higher the preference to buy, the higher the "Sell_Score" the higher the preference to sell.


Enter the Number of Stocks Recommendations You'd Like To See:
10
−
+
Buying Recommendations
Selling Recommendations

Buying Recommendations

| | entity_id | target | Buy_Score |
|---|---|---|---|
| 0 | C28100 | 0 | 0.6275 |
| 1 | TYZF00 | 0 | 0.6099 |
| 2 | NCXH50 | 0 | 0.6096 |
| 3 | PQFH50 | 0 | 0.6067 |
| 4 | PWKCG0 | 0 | 0.6042 |
| 5 | L71300 | 0 | 0.5969 |
| 6 | 9ZRBC0 | 0 | 0.5930 |
| 7 | ZCR000 | 0 | 0.5928 |
| 8 | MKV000 | 0 | 0.5892 |
| 9 | 23C000 | 0 | 0.5884 |

Selling Recommendations

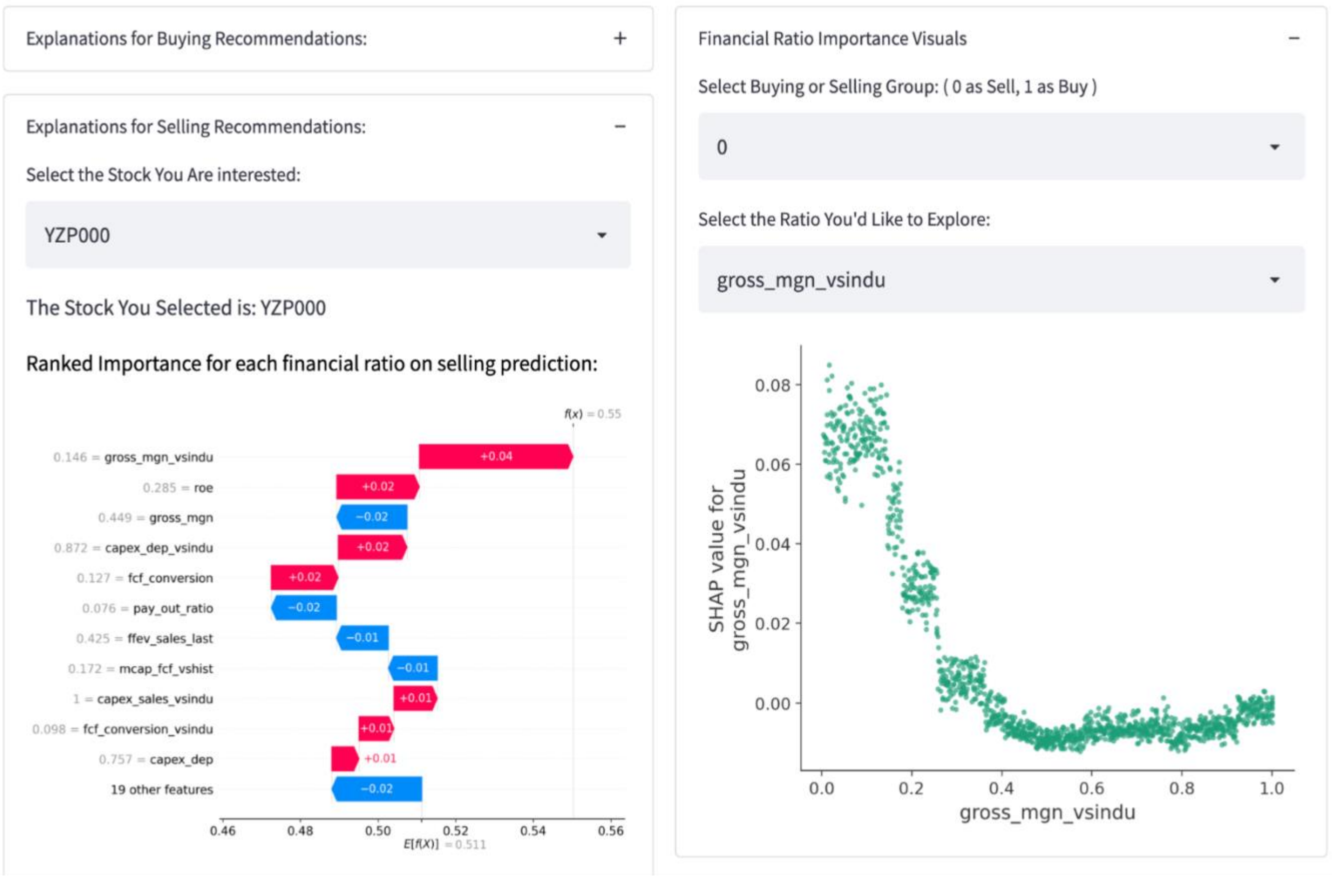
| | entity_id | target | Sell_Score |
|---|---|---|---|
| 0 | YZP000 | 1 | 0.5502 |
| 1 | WY5N70 | 1 | 0.5438 |
| 2 | 265100 | 1 | 0.5393 |
| 3 | KWJ200 | 1 | 0.5313 |
| 4 | 8GZD50 | 1 | 0.5311 |
| 5 | 6NC000 | 1 | 0.5304 |
| 6 | XY7700 | 1 | 0.5216 |
| 7 | 7MLZ50 | 1 | 0.5129 |
| 8 | QV2300 | 1 | 0.4872 |
| 9 | LVB000 | 1 | 0.4768 |

**Figure 7.** Screenshot of the Buy and Sell Recommendations from the web app

In the app, the user also can adjust the number of recommendations seen. That is done using the submenu seen in the screenshot at the top of Figure 7. On the first row, 'Entity_id' was the stock identifier, 'target' represented whether the stock was owned or not, and the score is the preference score of each recommendation.

## *III. Recommendation Explanations*

We used the SHAP value to explain the recommendations and the plots build in the SHAP package. It breaks out each score to show the influence of each feature on the score. The following figure shows the recommendation explanation plots, as well as the plot that shows how each feature influences the preference scores.


Explanations for Buying Recommendations: +

Explanations for Selling Recommendations: −

Select the Stock You Are interested:

YZP000

The Stock You Selected is: YZP000

Ranked Importance for each financial ratio on selling prediction:

Financial Ratio Importance Visuals −

Select Buying or Selling Group: ( 0 as Sell, 1 as Buy )

0

Select the Ratio You'd Like to Explore:

gross_mgn_vsindu

**Figure 8.** Plot on the left shows the waterfall plot of one of the recommendations. Plot on the right shows how the change of value of a selected feature changes the impact on the preference score.

In Figure 8, it can be seen that for the stock labeled as “YZP000”, the prediction score f(x) is 0.55, and the expected score is 0.511. The expected score (E[f(X)] at the bottom of the plot on the left) is the average before the features are added. Each feature then pushes the score lower or higher depending on its SHAP value (which is located within the red or blue arrows in the plot). The axis on the left shows the features and their values ordered from most influential to least. For this example,  it can be seen that gross_mgn_vsindu, which equals 0.146, is giving a positive influence of 0.04 to the preference score, which is the “Sell_Score” here.

The plot to the right of Figure 8 shows the influence for each of the features depending on their value. In the app, the user selects whether they want to see the feature influence for a buy or sell recommendation. Then the user selects the ratio they want to see. For this example, gross_mgn_vsindu was selected again and the plot shows how the value of the ratio changes, the influence it has the preference changes. For this ratio, as gross_mgn_vsindu increases, it has a smaller SHAP value. This means that as this ratio increases, it contributes less to the Sell preference score. Thus, the portfolio manager favors selling smaller gross_mgn_vsindu ratios.

## *IV. Model Feature Importance*

The next section of the app allows the user to see the feature importance across the portfolio as a whole, not just each individual stock. This is done by using the summary plot in the SHAP library. The horizontal axis represents the SHAP value (contribution to Buy or Sell Preference Score). The vertical axis is the list of all the financial ratios, this list is organized by most influential to least influential from top to bottom. Each dot represents a stock of the stock universe. The color of the dots on the plot shows the value of each financial ratio. If the dot is red, the stock has a higher value for the specific ratio. If the dot is blue, stock has a lower value for the specific ratio. This plot in the app is seen in Figure 9 below:

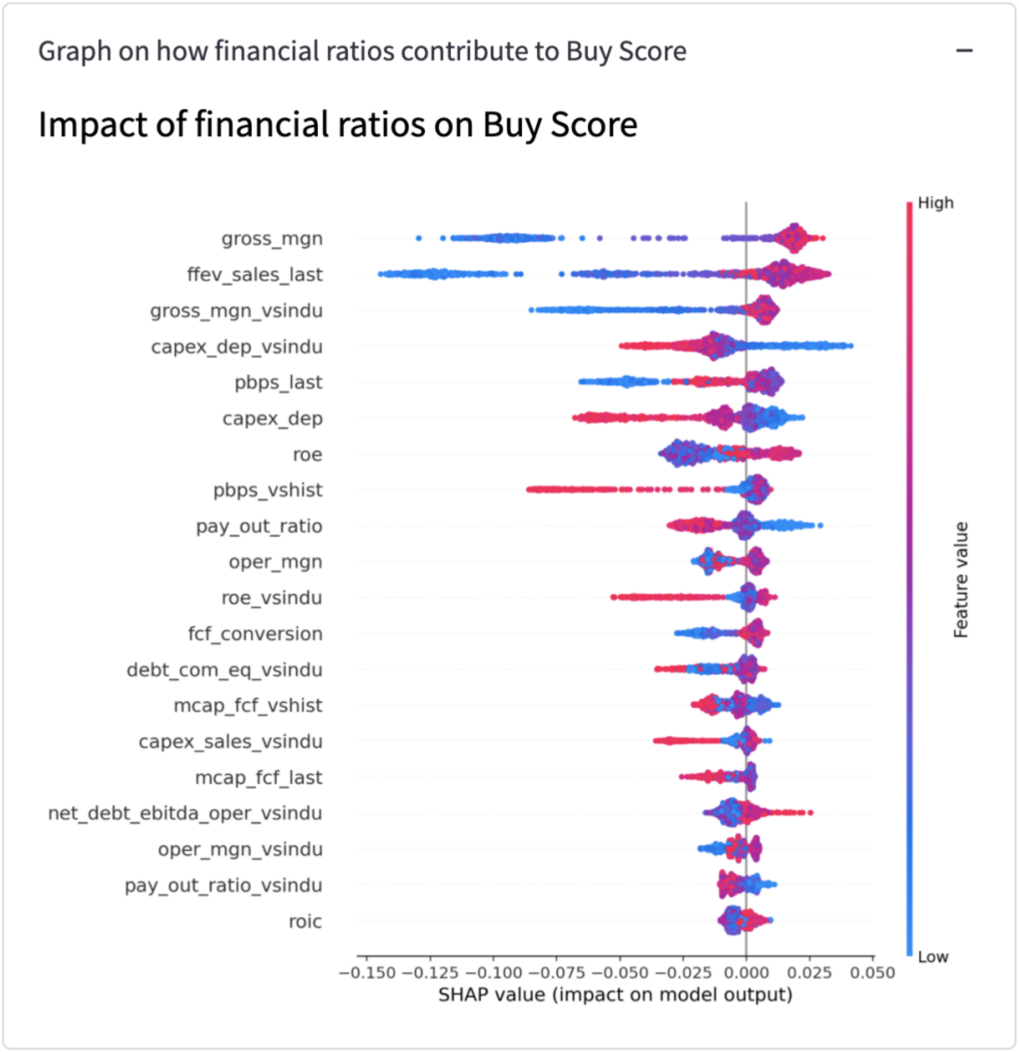


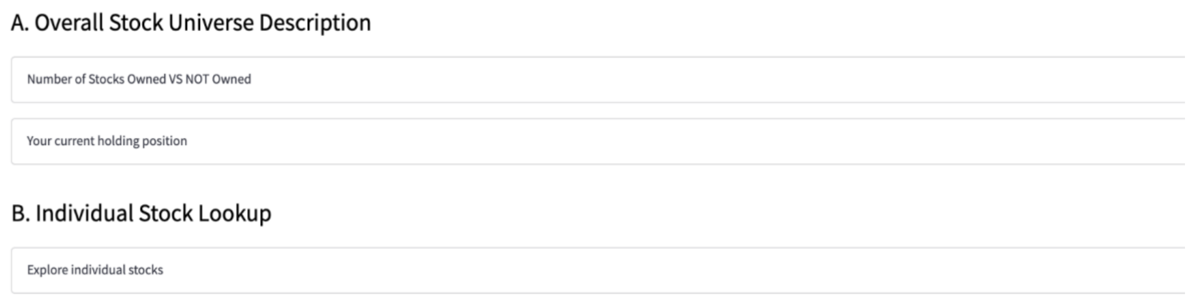
**Figure 9.** Screenshot of summary plot from web app

The following instructions are paired with the graph to better understand the plot. For each financial ratio, if the dots are RED and clustered towards the RIGHT (positive values on the horizontal axis), the ratio is favored to be LARGER and has a positive impact on the score. The further the dots trend to the right, the more significant contribution to the recommendation score. If the dots are BLUE and clustered towards the RIGHT, it means that the ratio is favored to be SMALLER and has a positive impact on the score. The further the dots trend to the right, the greater contribution to the recommendation score.

Vice versa for when the dots trend to the left (negative side of the horizontal axis). If the dots are RED and clustered towards the LEFT, it means that when the ratio is LARGER it has a NEGATIVE impact on the score. The further the dots trend to the left, the lower the recommendation score. If the dots are BLUE and clustered towards the LEFT, it means that when the ratio is SMALLER it has a NEGATIVE impact on the score. The further the dots trend to the left, the lower the recommendation score.

## *V. Extra functions*

In the last section of the web app, some extra functions were included to allow the user to explore the full dataset a little further. Numbers of stock owned and not owned are clearly displayed, and users can review their current stock holding positions. Also, users can explore individual stocks' financial ratios as they like. Finally, they can also sort stocks by adding one or more features and can change the numbers of stocks based on their selection. These functions can be seen in the following figures.


A. Overall Stock Universe Description

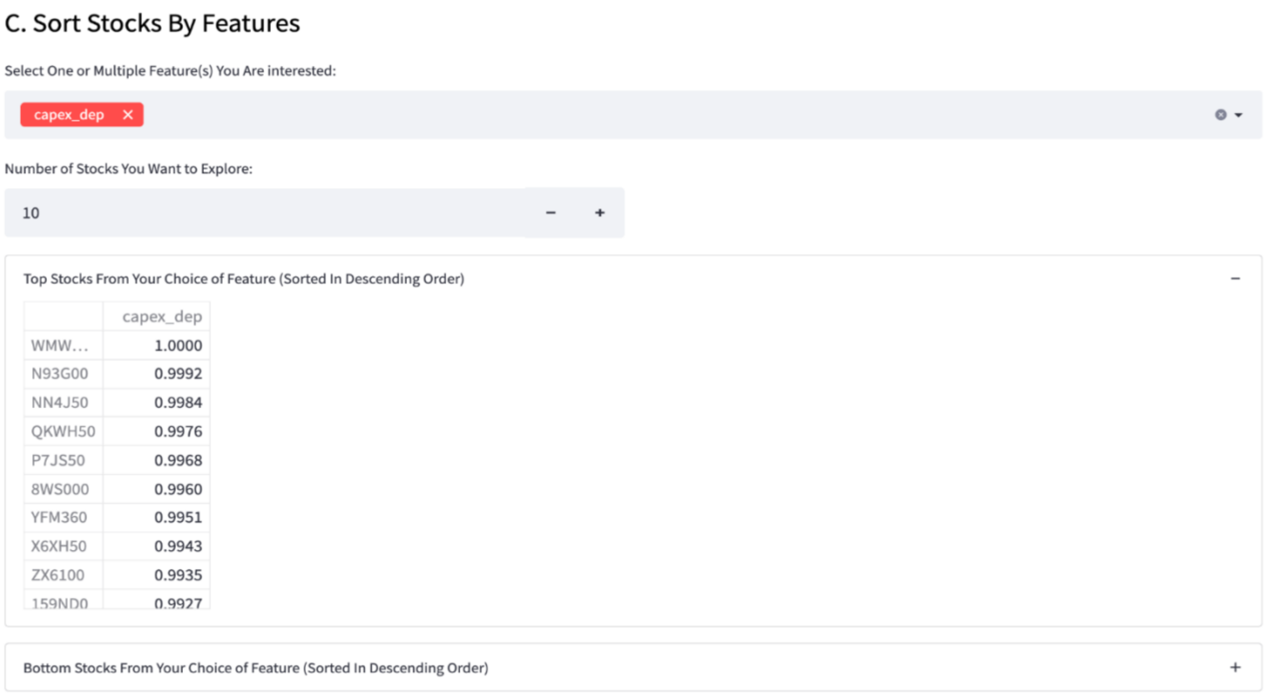
Number of Stocks Owned VS NOT Owned

Your current holding position

B. Individual Stock Lookup

Explore individual stocks

**Figure 10.** Screenshot of individual stock look up and number of owned vs unowned.


C. Sort Stocks By Features

Select One or Multiple Feature(s) You Are interested:

capex_dep

Number of Stocks You Want to Explore:

10

Top Stocks From Your Choice of Feature (Sorted In Descending Order)

| | capex_dep |
|---|---|
| WMW... | 1.0000 |
| N93G00 | 0.9992 |
| NN4J50 | 0.9984 |
| QKWH50 | 0.9976 |
| P7JS50 | 0.9968 |
| 8WS000 | 0.9960 |
| YFM360 | 0.9951 |
| X6XH50 | 0.9943 |
| ZX6100 | 0.9935 |
| 159ND0 | 0.9927 |

Bottom Stocks From Your Choice of Feature (Sorted In Descending Order)

**Figure 11.** Screenshot from web app of the extra function to sort stocks by specific features. Multiple features can be selected, and the number of stocks shown can be changed.

## Conclusion & Further Discussions

Overall, the app deployment was a success. A recommender system capable of explaining specific recommendations was developed. Next steps for this project would be to increase the apps scalability and deploy the finished product to the web. Currently it is only running locally. s

Additionally, the recommender model can be explored further to create better recommendation scores. Possibly deep learning or other current algorithms can be used to develop more accurate predictions.

Lastly, the speed of the app can be updated. Currently, once the user changes their selection, the page automatically refreshes and freezes page for a second. This issue can be improved and solved by finding a better structure to the webpage or developing the app's webpage on its own instead of using closed framework like 'Streamlit.'

## References

**[1]** RecSys ’16 September 15-19, 2016, Boston , MA, USA c 2016 Copyright held by the owner/author(s). ACM ISBN 978-1-4503-4035-9/16/09. DOI: http://dx.doi.org/10.1145/2959100.2959190
**[2]** Gama, J., Zliobait ˇ e, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. 2013. A Survey on Concept Drift Adap- ˙ tation. ACM Comput. Surv. 1, 1, Article 1 (January 2013), 35 pages. DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000
**[3]** Molnar, Christoph. “9.6 SHAP (SHapley Additive ExPlanations) | Interpretable Machine Learning.” Github, 11 Nov. 2021, https://christophm.github.io/interpretable-ml-book/shap.html.
**[4]** Niemer, Norman. “Explaining ‘Blackbox’ ML Models — Practical Application of SHAP.” TowardsDataScience, 26 Apr. 2020, https://towardsdatascience.com/explaining-blackbox-ml-models-practical-application-of-shap-f05f986863cf.
**[5]** Singn, Chandan. “Explainable AI: Why Should Business Leaders Care?” Towards Data Science, 2 Oct. 2021, https://towardsdatascience.com/explainable-ai-why-should-business-leaders-care-5e5078c609b5.