## **Project # 2 Report**

Team 8

Apoorv Thite - *aat5564@psu.edu*

[www.kaggle.com/apoorvthite](http://www.kaggle.com/apoorvthite)

Brian Nelson - brn5123@psu.edu

[www.kaggle.com/bnelly](http://www.kaggle.com/bnelly)

Nathan Kizlinski - *njk5644@psu.edu,*

[www.kaggle.com/nathankizlinski4](http://www.kaggle.com/nathankizlinski4)

**Project Overview:**

The world of stock market trading is buzzing with new strategies on how to maximize profits using Machine Learning and Data Analytics. And that is where you, as a Junior Data Scientist, step in!

The last column of the dataset represent the class of each stock, where:

* If the value of a stock increases during the year (Jan-Dec), then class = 1
* If the value of a stock decreases during the year (Jan-Dec), then class = 0

In other words, stocks that belong to class 1 are stocks that one should buy at the start of year (Jan), and sell at the end of year (Dec).

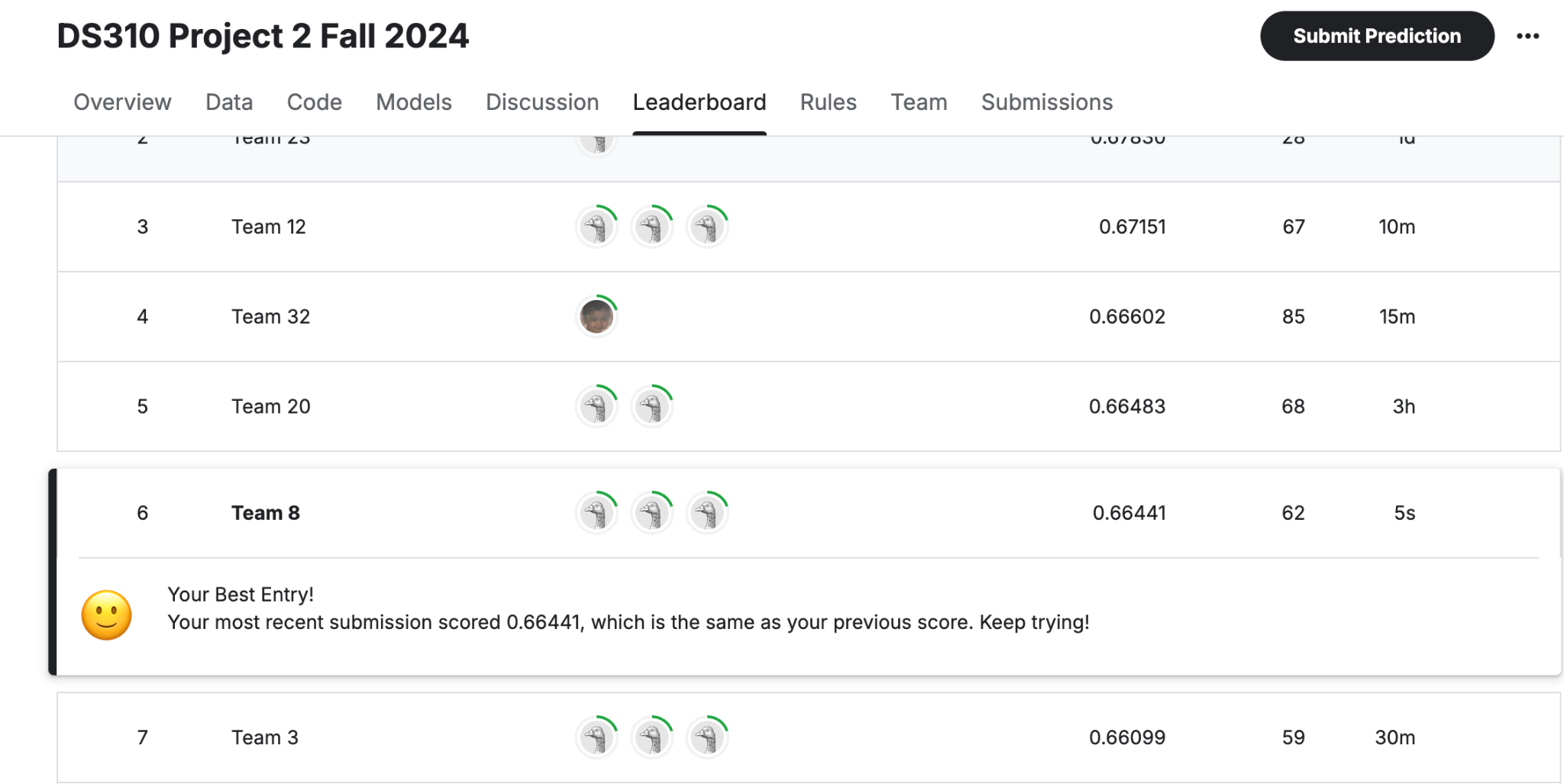
The last column, class, lists a binary classification for each stock, where

* For each stock, the 1 identifies those stocks that an hypothetical trader should BUY at the start of the year and sell at the end of the year for a profit.
* For each stock, the 0 identifies those stocks that an hypothetical trader should NOT BUY, since their value will decrease, meaning a loss of capital.

**Project Description:**

In this project, we aim to leverage machine learning and data analytics to develop a predictive model that helps identify stocks worth buying. The objective is to determine which stocks will appreciate over a given year (from January to December) and, therefore, should be purchased, versus those that will depreciate and should be avoided. The dataset used for this project consists of 224 columns, representing various financial indicators and performance metrics of different companies, with the last column serving as a binary classification label (1 for "BUY" and 0 for "DO NOT BUY"). The training data, which includes these labels, is used to train the model, while the test data without labels is used to evaluate its predictive performance. This approach is designed to support informed decision-making in stock market trading, potentially enhancing profit-maximization strategies.

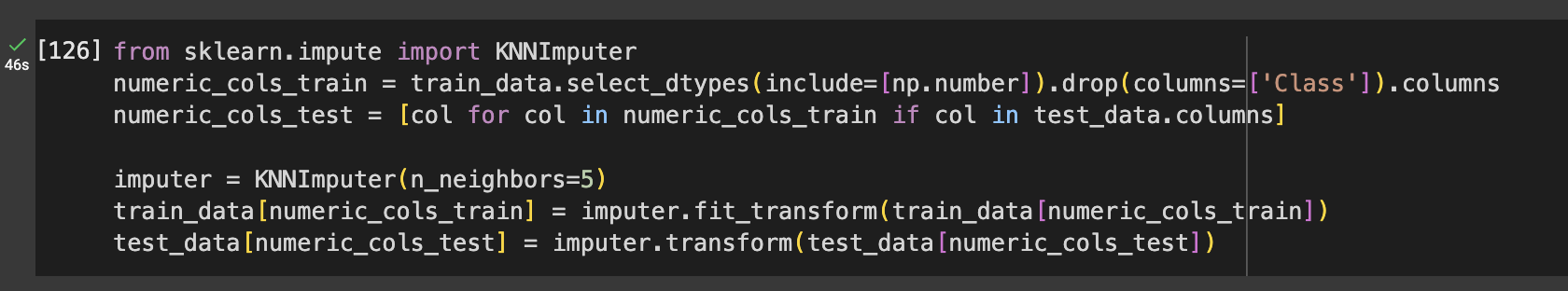
**Final Rank:**

****

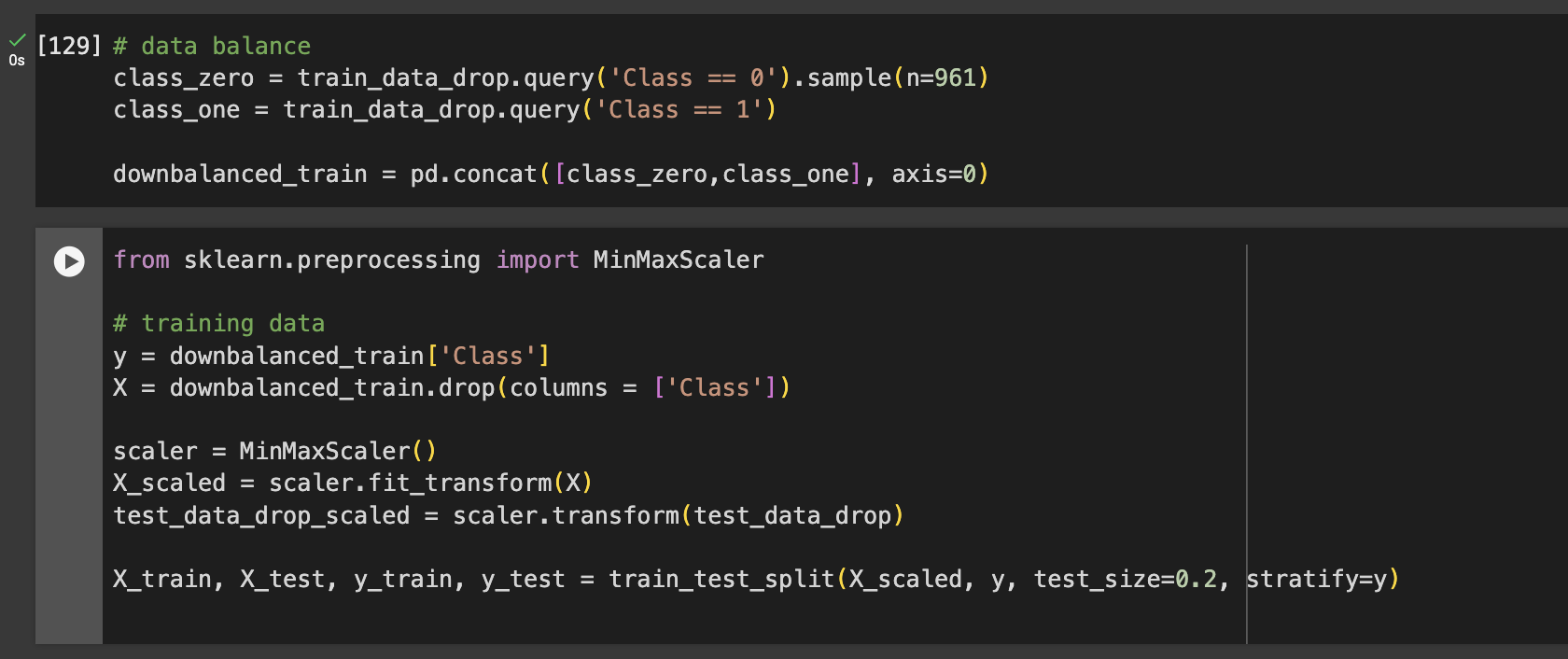
We finished 6th in the group with a score of 0.66441

**Pre-Processing:**

* Unlike in Project 1, the data that we are working with needed to be processed before we could work with it.
* Initially, we loaded the training and testing datasets and handled missing values using the **KNN Imputer**. This approach replaced missing data points by calculating the mean of the nearest neighbors, thus preserving the underlying structure and relationships within the data, unlike simpler techniques such as filling with zero. To enhance the data quality and remove irrelevant information, we dropped non-essential columns such as 'Name' and 'Sector', focusing solely on impactful features.



* Recognizing the class imbalance in the dataset, we implemented a down-sampling strategy where we balanced the majority class to match the size of the minority class, creating a more equitable training set for our model. The data was then normalized using MinMaxScaler to scale the features between 0 and 1, standardizing the dataset and preventing potential bias from dominant features.



* This step was vital to ensure consistent model performance and compatibility with algorithms that are sensitive to feature scales. Finally, we split the pre-processed data into training and testing sets while maintaining the class distribution using stratified sampling to support an unbiased evaluation of our model's performance.

**Feature Engineering:**

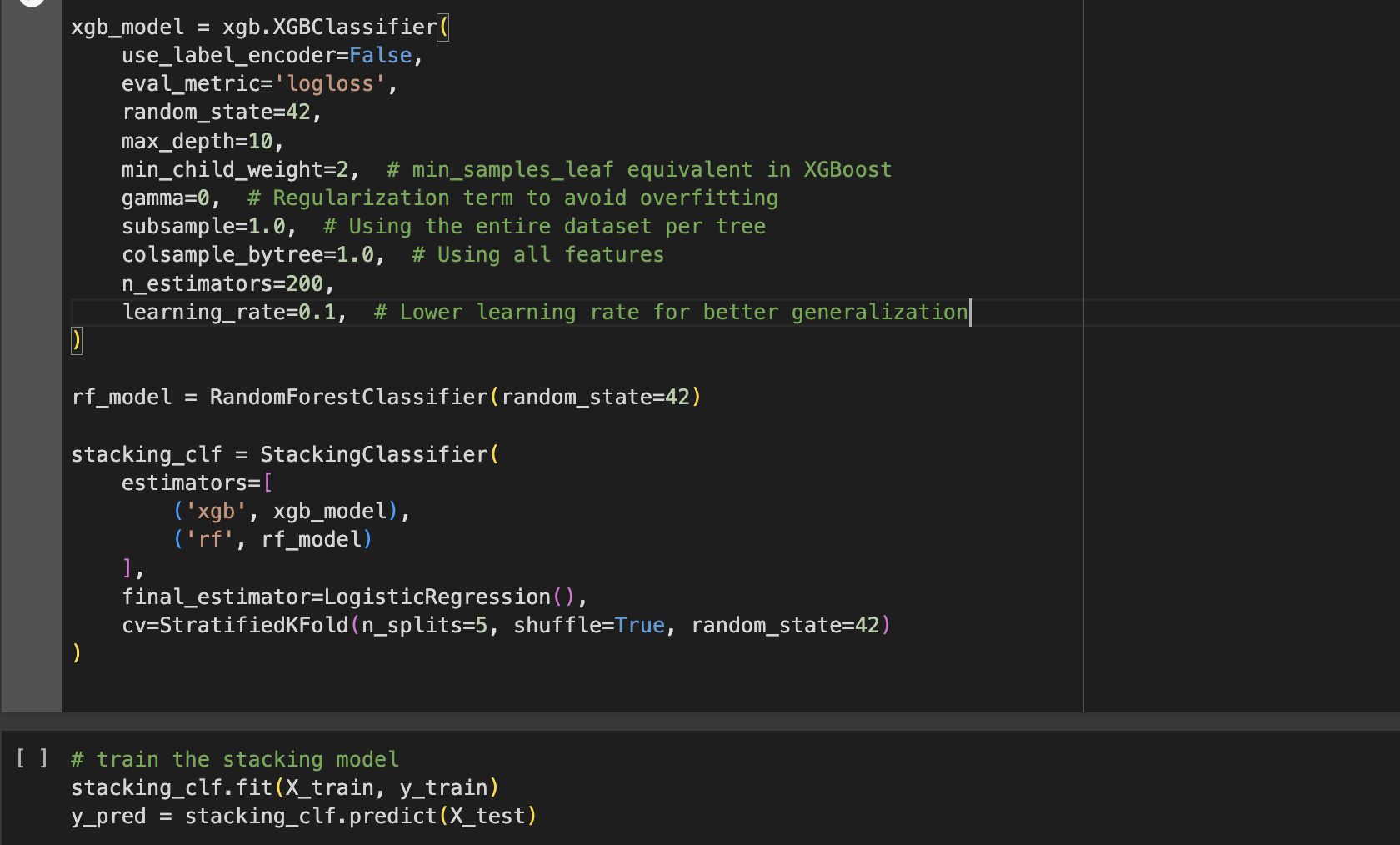
* As a team, we employed feature engineering techniques to optimize the performance of our classification model. As mentioned above, we used multiple techniques to make the processing of the data much easier. We used functions like .query(), .drop() to achieve this. These functions got rid of missing values, manipulated the data to show only what we need, and more, which allowed us to continue on with our project.
* Additionally, we used transformations to standardize data through scaling with **MinMaxScaler**, ensuring all features had a uniform scale between 0 and 1. This normalization step was especially important for algorithms sensitive to feature scaling, such as XGBoost and Random Forest, to perform optimally. We also employed **interaction terms** where needed, to capture relationships between variables that could contribute new information not evident in individual features.

**Model Building/Comparison:**

* Like Project 1, there were many different models to choose from to produce the highest score. Initially, we explored individual models such as **Decision Trees** and **Random Forests** for their straightforward implementation and ability to capture complex feature interactions.
* However, to boost performance, we transitioned to more advanced algorithms, including **XGBoost**, renowned for its gradient boosting framework and efficient handling of large-scale datasets. To maximize model robustness and generalization, we utilized a **stacking ensemble approach**, combining the predictive strengths of XGBoost and Random Forest.
* This ensemble method leveraged the unique strengths of each model by training them as base learners and combining their predictions using a **Logistic Regression** meta-learner for final decision-making.
* Our model-building techniques included hyperparameter tuning with **GridSearchCV** to find the optimal settings for each model and **StratifiedKFold cross-validation** to ensure balanced training across multiple data splits.
* The final model pipeline was designed to leverage the complementary nature of these algorithms, resulting in a robust predictive system capable of accurately classifying stock buy recommendations with optimized F1 scores.

**Hyperparameter Testing and Tuning:**

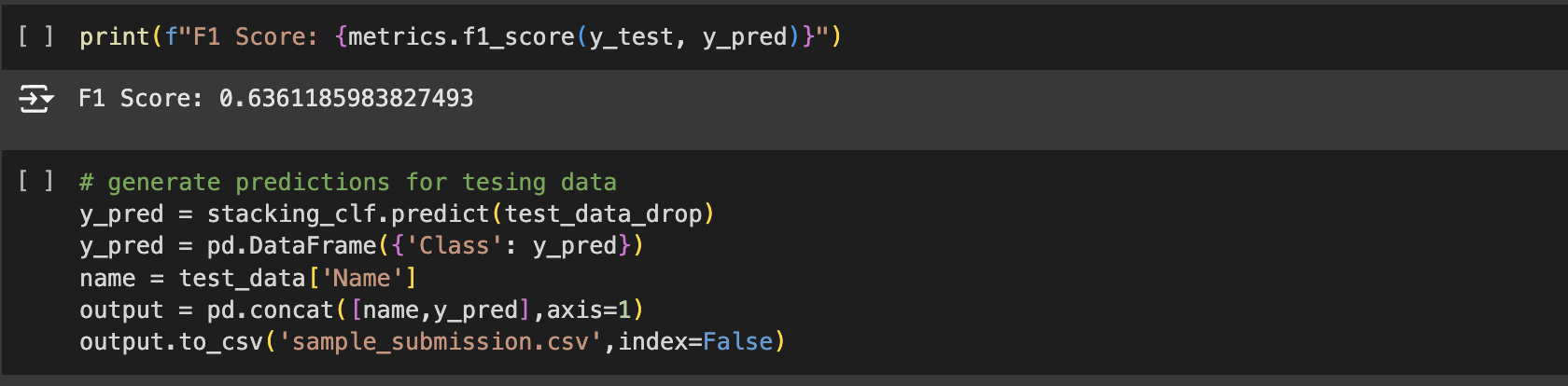
* We utilized advanced model building techniques that included the development of a **stacking ensemble** comprising **XGBoost** and **Random Forest** classifiers. The **XGBoost** model was fine-tuned with the following hyperparameters: max\_depth=10 to control tree complexity, min\_child\_weight=2 (analogous to min\_samples\_leaf) to prevent overfitting, n\_estimators=200 to define the number of boosting rounds, and a learning\_rate=0.1 for balanced training progression.
* Additionally, we set gamma=0 as a regularization parameter to further mitigate overfitting, subsample=1.0 to use the entire dataset per tree, and colsample\_bytree=1.0 to leverage all available features. The **Random Forest** model served as a complementary learner with a standard configuration, enhancing the diversity of predictions within the ensemble.
* The ensemble was wrapped in a **StackingClassifier**, using **Logistic Regression** as the meta-learner to combine base model outputs and provide final predictions. The stacking model was trained and validated using **StratifiedKFold cross-validation** with 5 splits, ensuring balanced class representation and robust performance evaluation. This setup allowed us to leverage the strengths of both gradient boosting and bagging techniques.



**Performance Evaluation:**

For this project we evaluated our model using the F-1 Score. The F-1 Score combines precision and recall, so it is a very strong method at evaluating a classification method. F-1 Score is calculated by the formula : (2TP)/(2TP + FP + FN). In this case a True positive is when we classified a Stock as class 1 and it is truly class 1. A False Positive is when we classify a stock as class 1 and it is truly class 0. A False Negative is when we classify a stock as class 0 and it is truly class 1.

Although the best metric on our machines that we got was around 0.63, which we felt was good enough to land us towards the latter half of the leaderboard. However, publicly our highest score was a .66 which lands us in the top 25%. We performed significant model building and hyperparameter tuning to get the f1 score to 0.66, which included stacking the models and using k-fold cross validation, as specified in the above sections.

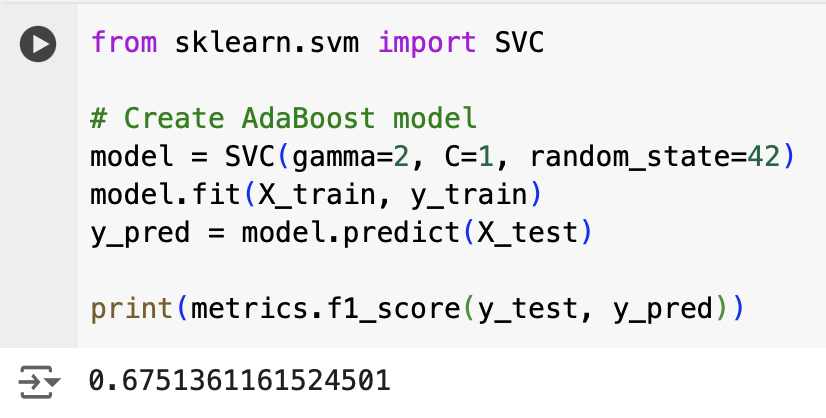


**Group Performance:**

As a group we worked very well and efficiently. On the first day the project was introduced we made a group chat to communicate with each other and set up meeting times. Throughout the project we met multiple times to discuss and plan. Our overall plan was that we would kind of independently write our code (with help when needed) then come together and compare which of our models worked. This worked very well because it allowed us to try as many models as possible while tuning each other's code as we went. Overall we were a very efficient and in-the-loop group where we all gave solid effort and were willing to overcome problems. At the end we all came together to finish the project report as a group together.

**Novel Ideas:**

Similar to Project 1, our results on our training data were different from the results on Kaggle. This was surprising to us because we thought that if we got a very good score on the training data, then we must have a very strong model. For example, a model we tried on the training data yielded a score of 0.63, while on Kaggle it resulted in a score of 0.66, which is vastly different. It is important to know that this was our highest score, although this model didn’t perform the best on Kaggle.

****

**Lessons Learnt:**

This project significantly challenged our skills as data scientists. It highlighted the collaborative nature of data science, making it clear that working alone would have posed considerable difficulties. We discovered that effective communication is essential; we maintained communication through a group chat, which greatly accelerated our progress. Throughout the project, we employed various classification models, each with its unique characteristics. This required regular updates among team members to keep track of what we had attempted so far and what still needed exploration and testing. Ultimately, we learned that tasks are often more complex than they initially appear. We found ourselves needing to plan and test far more extensively than we had anticipated, but this experience was very good for us.