School of Information Studies

Syracuse University

Jane Street Group Responder_6 Prediction

Final Project Report

IST 718 Big Data Analytics

Nikolay Yurashku, Benjamin Tisinger, Kéli Davis

Project Overview

Financial markets are notoriously complex and fiercely competitive. According to economic theory, any profit opportunity created by arbitrage is likely to be discovered quickly and eroded to zero as more market participants rush in to exploit it. In an era of unprecedented access to information, it can be argued that finding true alpha—an investing “edge”—has become nearly impossible without industrial-scale data mining and sophisticated predictive models. Wall Street quant shops such as Jane Street, Millennium, and Citadel dominate the CTA space by employing top PhDs and math Olympiad winners to sift through massive datasets in search of signals that yield elusive market advantages. In this project, despite not having access to the same level of cutting-edge infrastructure, we aim to develop a machine learning model to forecast *responder_6*, predicting its behavior up to six months into the future. Our goal is to demonstrate that even with more modest resources, thoughtful methodology and careful modeling can still provide valuable insights.

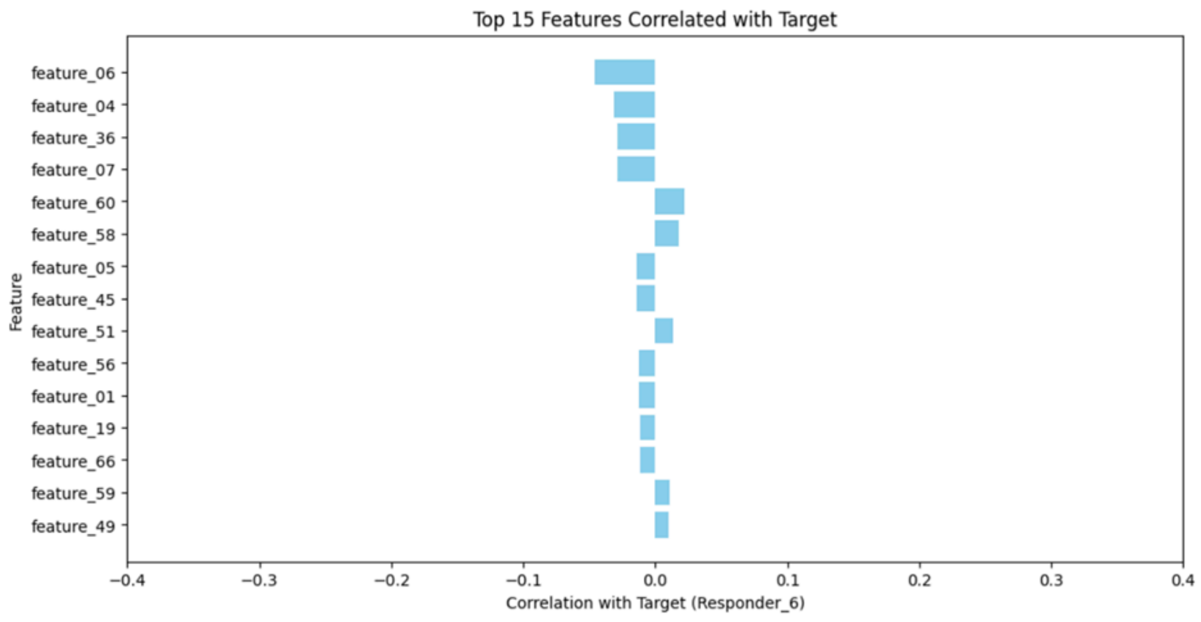
The dataset comes from a Kaggle competition hosted by Jane Street Group, a global proprietary trading firm. It consists of time-series financial data with 92 anonymized columns representing features and responders. The dataset has 47,127,338 rows. One of the biggest challenges of this project is the anonymization of responder_6, which forces our team to rely solely on data-driven insights. The actual Kaggle competition is also ongoing. The highest R^2 score on the leaderboard is only 0.012932. This low of an R^2 score demonstrates how difficult it is to find meaningful predictive patterns in financial data. This also reflects the reality of quantitative trading, where the smallest predictive advantage can lead to an edge over the competition.

Our primary project goal is to identify patterns within the Jane Street dataset, identify the key features influencing responder_6, and forecast responder_6 up to six months into the future.

Data Exploration

For data cleaning and exploration, a 5% randomly sampled dataset was extracted from the full dataset of over 47 million rows. This was done to reduce the computation time required for cleaning, exploration, and modeling. This methodology assumes the randomly sampled data is representative of the full dataset. The original dataset contains 9 responders including the target, responder_6. All responders except for the target were dropped from the sampled dataset. Next, the data was checked for categorical data and missing data. If the missing data was more than 10% of the individual feature, the missing rows were removed. If the missing data was less than 10% of the feature, the missing values were imputed with the median value. The low-variance features were then removed. After cleaning was completed, data exploration began. To better understand the relationship between features and the target variable, a correlation analysis was conducted on the 5% sampled dataset. The graph below highlights the top 15 features with the strongest correlations to the target. Feature_06 had the highest correlation at -0.046, followed by Feature_04 at -0.032—both of which are negatively correlated with responder_6. However, despite being the strongest relationships, these correlation values are generally weak. This indicates that no single feature has a dominant impact on the target variable.

Exhibit 1.1



A feature importance analysis was also conducted to identify the most influential features in predicting the target variable. The graph below shows the top 20 most important features, ranked by the most predictive. Feature_06 is the most significant predictor. Feature_04 ranked second, though its predictive influence is considerably lower than Feature_06.

Exhibit 1.2

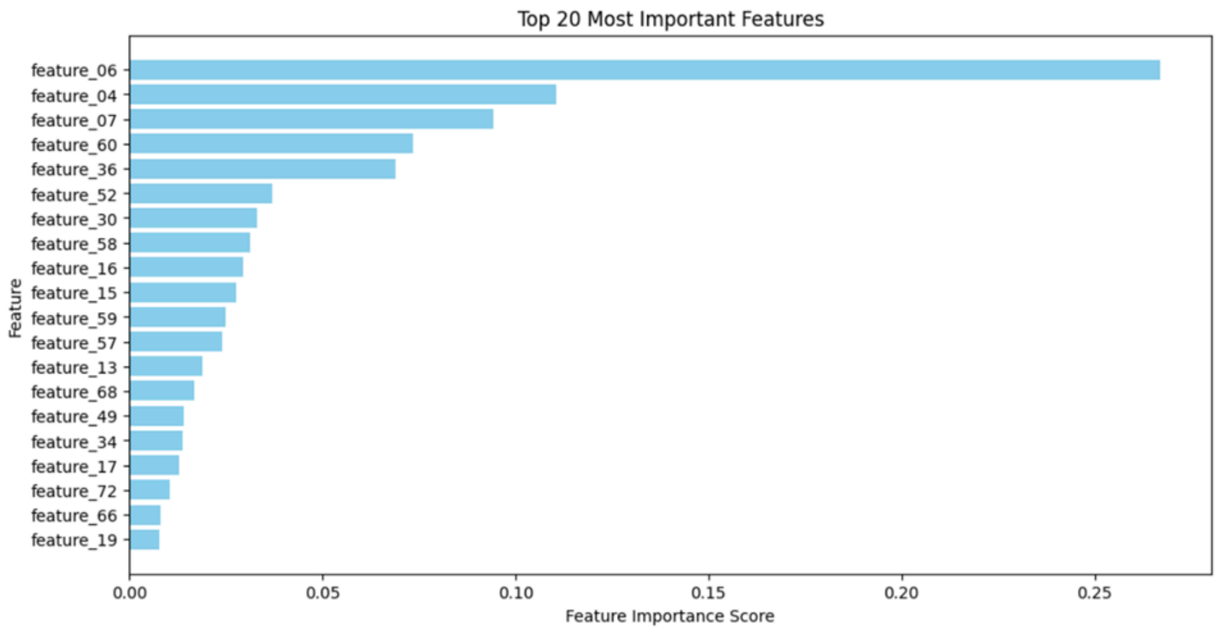
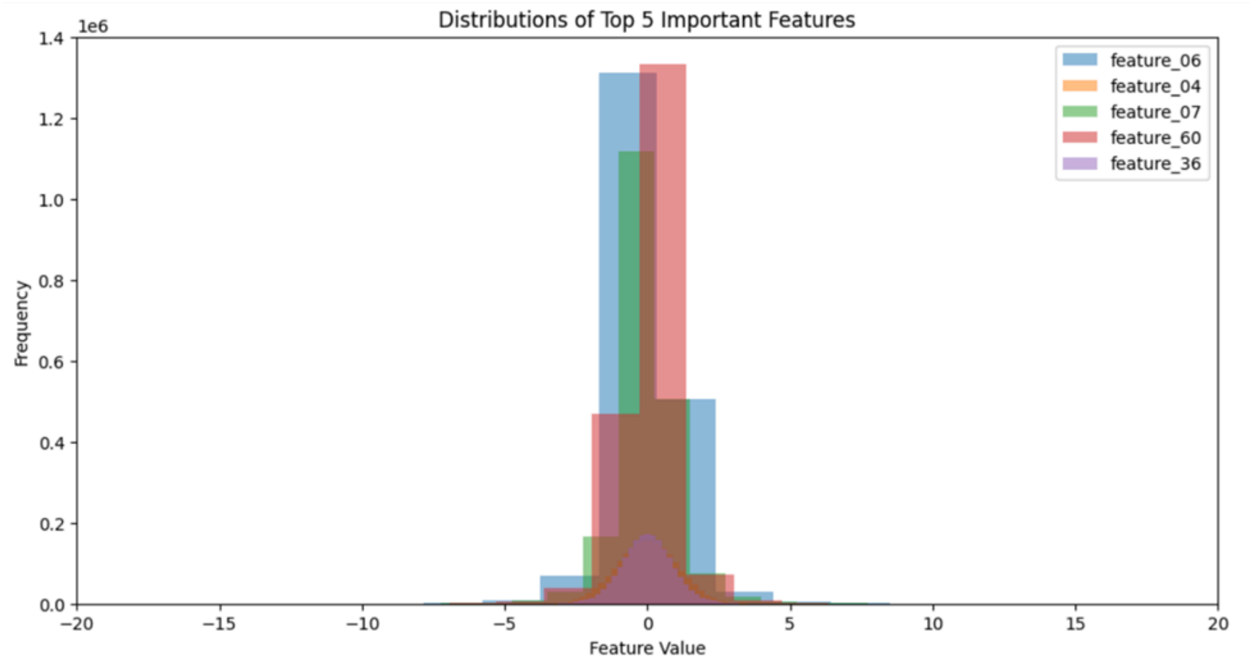


Exhibit 1.3



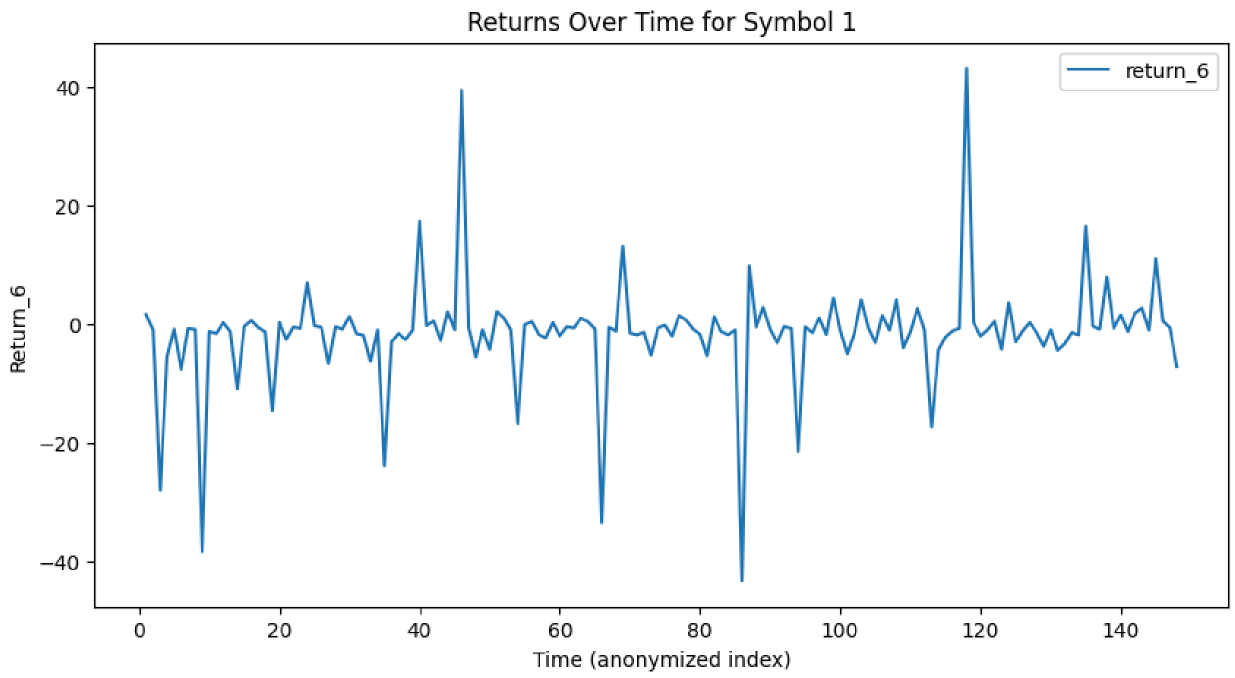
The figure above illustrates the distribution of the top five most important features. It indicates the data for each feature follows a normal distribution pattern.

Further Analysis of Dataset

In an effort to understand the dataset's underlying meaning, the team hypothesized that the feature values might represent a rate of return. Jane Street included columns for date and time; however, both were anonymized and reduced to simple integer values indicating a progression of time.

As shown in Exhibit 1.4, we can plot any symbol in the dataset against this anonymized time index to produce a time series. The resulting plot displays patterns reminiscent of a stock price or a volatility metric, reinforcing the idea that these feature values could indeed be related to returns or price movements.

Exhibit 1.4:



Summary of Methods

Method 1: XGBoost and LightGBoost Benchmark Approach

As a baseline, we applied XGBoost to all available features to see how the model would perform. Because Google Colab was unable to handle the entire 47 million-row dataset due to CPU limitations, we instead trained the model on a more powerful local machine. This approach established a performance benchmark, allowing us to assess both whether additional data genuinely improved the model and how much further optimization might be possible.

We trained the model in chunks using 80% of the available data, ultimately achieving an R^2 score of 0.0079. Since the output could not be viewed in Google Colab, the full results and code are available in the team's GitHub repository: [GitHub link]. LightGBM—a closely related gradient-boosting framework to XGBoost—was also tested on the full dataset (47 million rows), but it produced a worse R^2 score of -0.11 [Github Link]. This negative R^2 suggests that the model performed worse than a simple baseline, indicating potential overfitting or an inability to capture the underlying signal from the high-volume data.

Next, we ran LightGBM on a smaller, more carefully curated subset of the dataset in Google Colab to see if removing noise might enhance performance. This approach indeed yielded better results, improving the R^2 to 0.015. Although still relatively modest, the improvement indicates that a targeted selection of data and features can sometimes outperform brute-force methods on massive datasets, where noise and computational limitations can overshadow meaningful signals.

Method 2: Feature Selection and Sampling

To reduce model computation time and complexity, features found to be not correlated or unimportant during the EDA process, were removed while preparing the data for modeling. This was done by combining correlation analysis and feature importance. Only features with a correlation above an absolute value of 0.01 and also found to be in the top 15 most important features were selected for modeling. This reduced the feature count from 83 to 22. After feature selection, MinMaxScaler was used to normalize the feature values.

The sampled dataset was split into training (70%) and testing (30%) prior to modeling. Two algorithms were used to model the data: XGBoost Regressor and Random Forest Regressor. For the XGBoost model, the appropriate hyperparameters were specified and 3-fold cross-validation was employed to select the optimal hyperparameters. For the Random Forest model, the hyperparameters were manually tuned as cross validation computation time was too high (> 30 minutes).

Table 1.1

Model	XGB	Random Forest
Hyperparameters Selected for Tuning	Max Depth = 3, 5 Learning Rate = 0.05, 0.1	Max Depth = 3, 5 Num of Trees = 20, 30, 50 Subsampling = 0.5, 0.8, none

Results Summary

Method 1 Results:

The team evaluated both XGBoost and LightGBM on the entire dataset of 47 million rows, then applied LightGBM to a more carefully curated subset. XGBoost on the full dataset achieved a modest R^2 of 0.0079, indicating it captured only a small amount of signal. LightGBM, when applied to the full dataset, performed worse with an R^2 of -0.11, suggesting that excess noise may have overwhelmed the model's predictive capacity. However, when LightGBM was used on a curated portion of the dataset, its performance improved to an R^2 of 0.015. Although still modest, this positive shift underscores the value of data curation and feature selection in managing noise and improving model accuracy. These results establish an initial performance benchmark, highlighting that merely increasing data volume does not necessarily lead to better outcomes and illustrating the importance of careful data handling.

Table 1.2

Model	R^2
XGBoost (Entire Dataset)	0.0079
LightGBM (Entire Dataset	-0.11

LightGBM (Curated Dataset)	0.015

Method 2 Results:

The table below shows the resulting R² score and optimal hyperparameters. The XGBoost performed the best at 0.0129. Random Forest Regression had an abysmal R² score of 0.0036. The computation time for the random forest model was very high as compared to the XGBoost model and hyperparameters had to be reduced during each manual tuning iteration. The model's poor performance may suggest issues with model stability or data preprocessing.

Table 1.3

Model	XGB	Random Forest
Best Model	Max Depth = 5 Learning Rate = 0.1	Max Depth = 3 Num of Trees = 20 Subsampling = 0.5
R^2	0.0129	0.0036

Method 3 Results:

The table below shows the resulting R² score and other key statistics done while using a Linear Regression and PCA Linear Regression. The first approach uses standard Linear Regression, where a set of features (from columns 5 to 77 of Kaggle Dataset) is compiled into a feature vector using VectorAssembler and then fitted to the model. The second approach is intended to enhance the model by incorporating (PCA) to reduce the dimensionality of the input features, using the first 10 principal components as the input. Both approaches while successful did consume a large amount of resources and time depending on the sample size of the data.

Table 1.4

Model	Linear Regression	Linear Regression (PCA)
R^2	0.0296	0.00636

Method 4 Results:

The table below shows the resulting R² score and other key statistics done while using a Decision Tree model and applying Clustering techniques to our Decision Tree. The first approach, a DecisionTreeRegressor is applied to a dataset with the selected number of features (from columns 5 to 25 of the Kaggle Dataset). The Decision Tree model is then trained and fitted on the combined data to predict the target of responder_6. The second approach introduces clustering by adding cluster information and date_id to the dataset. This allows the Decision Tree to predict our column responder_6 while considering the impact of clusters and date in the data.

Table 1.5

Model	Decision Trees	Decision Trees (Clustering)
R^2	0.0753	0.09723

Method 5 Results:

The table below shows the resulting R² score and other key statistics done while using a KNN model and KNN Classification Model. In the first approach, KNN Regression is used, where the data is split into training and test sets. The KNeighborsRegressor model is then trained using the training data, with the number of neighbors set to 10 for weighting our predictions. In the second approach, KNN Classification is applied by first transforming the responder_6 variable into a binary classification problem based on whether the values are above or below the average.The output from the KNN highlights that our value of 0.022 and accuracy of 57% is just average at attempting to predict responder_6.

Table 1.6

Model	KNN	KNN Classification Accuracy
R^2	0.02239	0.57%

Method 6 Results:

The table below shows the resulting R² score and other key statistics done while using the Naive Bayes model and the Bayesian Ridge model. We preprocess the data by replacing missing and infinite values with zero and scale the features using MinMaxScaler. The Naive Bayes classifier is then trained on the scaled features to predict our column responder_6. For regression, Bayesian Ridge is applied after splitting the data into training and test sets. The output from the Bayesian Ridge and Naive Bayes models prove that this technique does a subpar job at predicting the output of responder_6.

Table 1.7

Model	Bayesian Ridge	Naive Bayes Accuracy
R^2	0.00449	0.39%

Challenges

This project underscores the intricate challenges inherent in predicting financial market behavior from large, anonymized datasets, particularly in a high-stakes environment where even a minor edge can be significant. The team undertook two main approaches—Method 1 (benchmark modeling on the full dataset using XGBoost and LightGBM) and Method 2 (targeted modeling on a curated 5% sample using XGBoost and Random Forest)—to cope with both computational constraints and the complexity of the data.

Despite using a separate, more powerful machine for Method 1, modeling the entire 47 million-row dataset proved both time-intensive and resource-heavy. The brute-force approach yielded only modest predictive power: XGBoost on the full dataset achieved an R^2 of 0.0079, and LightGBM reached -0.11, suggesting that high data volume alone can overwhelm models if much of that data is noisy. However, applying LightGBM to a smaller, more carefully filtered subset improved the R^2 to 0.015, demonstrating the importance of feature selection and data preprocessing in managing noise.

Method 2, which involved extensive feature removal (based on correlation and importance analyses) and downsampling to 5% of the dataset, produced the project's best in-sample result: an R^2 of 0.0129 using XGBoost with tuned hyperparameters. Yet, this success did not fully translate to the larger Kaggle holdout set, where performance fell to -0.007846, ranking the team 3233rd out of 3412 participants. This drop highlights a persistent challenge in financial modeling: even robust in-sample results can fail to generalize once exposed to the true complexity of market data. Random Forest, tested under the same conditions, struggled further with an R^2 of 0.0036, likely owing to its higher computational demands and potential sensitivity to noisy features.

Several factors help explain these results. First, random sampling may undersample crucial market events or time periods in which important signals occur. Second, the dataset's anonymized nature means that conventional domain knowledge—such as sector-level interactions or known macroeconomic cycles—cannot be leveraged. Third, the project's eight-week timeline limited repeated experimentation, while leading competitors on Kaggle have typically spent months (or more) refining architectures, hyperparameters, and feature engineering strategies. Lastly, the top leaderboard score itself remains under 0.015, illustrating that even the best-performing models capture only a tiny fraction of the inherent signal.

Nonetheless, this project offers critical insights. The data exploration phase clearly showed weak overall correlations and the necessity of data cleaning, feature selection, and normalization. Minor increases in R^2, though not always dramatic, are meaningful in quantitative finance, where any sustainable edge is difficult to find. Future work could explore advanced modeling techniques—such as deep neural networks, ensemble blends of gradient boosting and random forests, or recurrent architectures for time-series signals—while employing more nuanced sampling strategies (e.g., stratified by key time windows or volatility regimes). Above all, these results reinforce that discovering alpha in real financial markets is an ongoing, resource-intensive endeavor, and that careful attention to data quality and targeted experimentation are essential for any measurable predictive success.