Statistical Learning-Classification

Project Title: Jane Street Market Prediction- Kaggle Competition

Project Number: Group 33

Group Members:

Surname, First Name	Student ID	STAT 441	STAT 841	CM 763	Your Dept. e.g. STAT, ECE, CS
Halim, Hansa	20742361	☑	☐	☐	Statistics
Marfua, Samka	20701370	☑	☐	☐	Statistics
Naik, Sanjana	20744518	☑	☐	☐	Statictics
Proshasty, Shawrupa	20704866	☑	☐	☐	Statistics

Your project falls into one of the following categories. Check the boxes which describe your project the best.

1. ☑ **Kaggle project.** Our project is a Kaggle competition.
   - This competition is active ☑ inactive ☐.
   - Our rank in the competition is .226/1114
   - The best Kaggle score in this competition is . 8,846, and our score is 6,704
2. ☐ **New algorithm.** We developed a new algorithm and demonstrated (theoretically and/or empirically) why our technique is better (or worse) than other algorithms.
3. ☐ **Application.** We applied known algorithm(s) to some domain.
   - ☐ We applied the algorithm(s) to our own research problem.
   - ☐ We tried to reproduce results of someone else's paper.
   - ☐ We used an existing implementation of the algorithm(s).
   - ☐ We implemented the algorithm(s) ourself.

Our most significant contributions are (List at most three):

(a) .

(b) .

(c) .

List the name of programming languages, tools, packages, and software that you have used in this project:

Python, Kaggle Kernels
Numpy, Pandas, xgboost, Sklearn, SimpleImputer, Tensorflow, Borupashap, hyperopt

Stat 441 Final Project
Jane Street Market Prediction Kaggle Competition

Hansa Halim, Samka Marfua, Sanjana Naik, Shawrupa Proshasty

1. Summary

Our group took part in the Jane Street Market Prediction Challenge. In this paper we

1. Provide an introduction and description of the challenge and the dataset.
2. Share the results of exploratory analysis performed on the dataset.
3. Describe and reason about the approaches we took to model this problem.
4. Discuss and summarize the results that we obtained.

2. Electronic Trading

Trading floors began with stocks being exchanged physically or over a phone call. However, this practice began to cease in the 1990's when electronic trading became popular. The power of computers enabled thousands of transactions of stocks, and other financial instruments, in fractions of seconds and have also significantly reduced the number of human traders required. Trading firms implement algorithms on this electronic trading platform, and this is known as Algorithmic trading. These algorithms comprise of set of instructions for placing a trade. The benefit of this is that trades are executed at the best possible price and manual errors are avoided.

In this challenge we aim to create a model that predicts whether to trade a financial instrument given some information about that trade. This model can then be implemented in an algorithmic trading platform where it would influence the buy and sell decision.

3. Data description

The data provided by the challenge has a training dataset comprising of 2.39 million rows and 138 columns. Each row represents a trading opportunity for which the goal is to predict the variate $action$. An $action$ of 1 indicates that a trade should be made and 0 indicates to pass it on.

Out of the 138 columns, 130 columns represent anonymous $features$ that influence the trades. To simplify the data, the date column is an integer, which shows which day the trade was placed in. There are a total of 500 days and each day has approx 5000 rows of trading information. The time ordering of the trade is also of high significance which is represented by the ts_id column and the entries range from 1 to 2.39 million.

Each row also has a $weight$ and $resp$ column, and $weight \times resp$ represents the return on a trade. A positive $resp$ implies a positive return rate and a negative $resp$ implies a negative return rate with $resp$ ranging between -1 and 1. There are 4 additional resp columns that show returns over different time horizons.

Rows (trades) with $weight$ = 0 are also included and they don't contribute towards scoring evaluation. The competition is evaluated on a utility score w.r.t to action. For each unique

day i and trade j, we define

$$p_i = \sum_j (weight_{ij} * resp_{ij} * action_{ij}),\ t = \frac{\sum p_i}{\sqrt{\sum p_i^2}} \times \sqrt{\frac{250}{|i|}}$$

and the final utility score, which we aim to maximize, is

$$u = \min(max(t, 0), 6) \sum p_i.$$

A second file provided contains 30 columns (tags) and 130 rows of $features$ with each row showing the tags a feature contains. The competition required a submission of the notebook containing a time series API provided by the organizers. The submitted notebook was required to complete the predictions on the test data and had to compile within 5 hours to count as a successful submission.

4. Exploratory Analysis

Market prediction datasets are usually time series data. However, for this challenge, the training dataset was not a time series data and we needed to predict the $action$ variate based on other variables. Hence due to the nature of this dataset, we applied regression and classification algorithms models. To aid better results, we explored the data and cleaned it up in the following ways:

1. 17% of the trades had $weight$ 0, and since this meant they don't contribute to the scoring, we dropped these rows. We also avoided the rows with $weight$ = 0 when testing the data, which saved up computation time.
2. There were approximately 2% missing values, and they seemed to follow a pattern which can be observed in figure 3. Since it was highly probable that this pattern also occurred in the test data, we didn't ignore these missing values. Instead we imputed them with their mean, since most of the values concentrated around the mean value of the features.
3. We dropped the 4 additional $resp$ columns $resp\_1, resp\_2, resp\_3$ and $resp\_4$ as well since they were highly correlated with $resp$ (all correlation values > 0.3).
4. We made a new action variate/column, with $action_j$ as 1 if $resp_j > 0$ and 0 otherwise for j rows. This produced a balanced dataset with approximately equal number of rows with $action$ = 1 and $action$ = 0.
5. We noticed an interesting pattern from the graph in Figure 2 with $weight$ and $resp$ being highly correlated. Trades with high $resp$ had low weight and vice versa. In real life if a trade is risky or highly volatile (high rate of return), we would invest less or put less weight on that trade. Hence we assumed $weight$ as a risk factor here.
6. Regardless of the $weights$ of a trade, if the $resp$ values were positive then the trade was considered profitable. And since the utility score is calculated by multiplying $weight$ and $action$, we dropped the $weight$ column in the model to predict response variate action.
7. We observe a similar pattern in the heatmap of the second file of $features$ and the heatmap of the correlation matrix. This showed that some of the features were highly correlated and so we investigated this further. The anonymous labelling made the feature selection even more challenging.

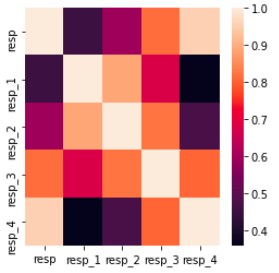

Figure 1. correlation matrix for $resp$ and $resp_{1,2,3,4}$

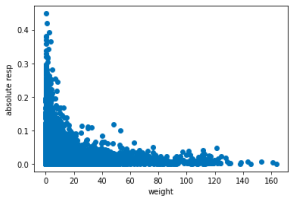

Figure 2. absolute weight vs resp

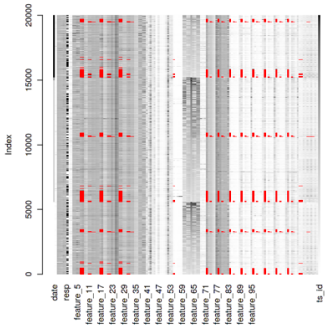

Figure 3. missing data (red) seem to follow a pattern

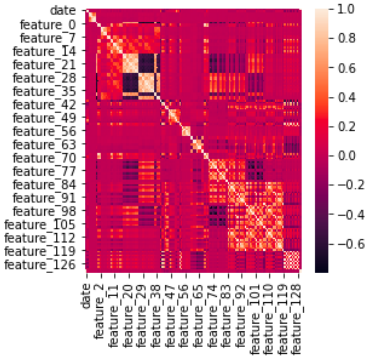

Figure 4. correlation matrix for the features

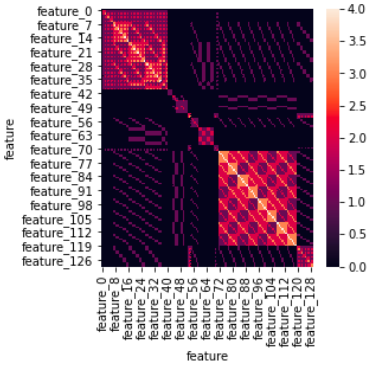

Figure 5. Similarity of the tags contained in the features (between 0 and 4)

5. Feature Selection

Since our exploratory analysis showed significant correlation among some of the features, we investigated this further. There were 130 features and by reducing them we could possibly reduce data over-fitting. Thus, we tried feature selection using 2 methods, SelectKBest (ANOVA) and the Boruta-SHAP Package by Ekeany (github). BorutaShap feature selection algorithm combines both the Boruta feature selection algorithm with SHAP technique. The advantage of using this algorithm was that it found the minimal number of features important to the target variable.
We noticed a significant improvement in the run times as well as our scores after selecting features based on a combination of the two methods.

6. Model fitting

We tried one regression and two classification algorithms on the dataset. We picked the following algorithms as they are all different from each other and have relatively low training duration for this dataset. We avoided other algorithms as they had longer training durations and the organizers had provided us with a time constraint for the submission. Therefore, we were limited and decided to test these three algorithms:

1. **Logistic regression:** Two logistic regression models were tested after data cleanup, one with all the features and the other after feature selection. We found that while

testing the model on the validation data, cross-validation accuracy was 0.52 before feature selection i.e the full model and 0.61 for the model using feature selection. Since both these scores were not high enough, combined with this model taking a long time to compile, we concluded that logistic regression was not the best model for this challenge.

2. **XGBoost:** Over 10 different iterations of XGBoost were created using different kinds of settings and data cleanup. We faced many issues from RAM crashing and evaluation timeouts due to the extremely large training and testing set. From the many iterations that were made, we found that having more estimators and more depth provided a higher utility score. Having a gradient based sampling method also significantly improved the utility score. We also found that having too much depth caused stack overflows and caused the program to crash, so we aimed to avoid this. Our first XGBoost model garnered a utility score of 3,299 which placed us in the top 70%, and our best one sat at 4,356 which is in the top 50%. Thus, we concluded that XGBoost was not the best model to pursue as we wanted to aim for better results.

3. **Neural Networks:** We trained three models, one of which was a pre-trained model which yielded the best score of 6,876 and placed us in the top 10%. While the other two are our own developed models that we trained overnight with our best model yielding a score of 6,704, which placed us in the top 15%. We discovered that having a higher batch size increased the score. However, it was very hard to increase the cross-validation accuracy on the validation set due to unpredicatble features of the dataset.

7. Conclusion

Comparing the results from Logistic Regression, XGBoost and Neural Networks, we conclude that training the model using Neural Networks works best for this classification problem. Data cleansing and imputation, feature selection and higher batch sizes were used to attain the best results. The best model that we trained using Neural Networks provided us with a utility score of 6,704 and placed us in the top 15% on the leaderboard. This competition is very interesting and we've learnt a lot about the stock market, we're going to continue working on our model and aim for a higher score.

8. Discussion

Anonymous features made it hard to perform any kind of fundamental financial analysis on the data. Patterns which we observed in heatmap of the $features$ and the missing data provided some hints on what the real label of these features might have been. These patterns thus need extensive investigation.

Additionally, the time constraint for submission limited the scope of using other kinds of algorithms. Timing is very important in trading as there are other competitors in the market. This also led us to research ways computation time can be minimized which help make algorithms more feasible to implement on the trading platforms.

The objective of this competition was to optimize the utility score, which also corresponds to how profitable the model is. In real life the optimal strategy is to maximize profits by keeping the risk/volatility low. Hence, it was very beneficial for the organizers to consider the risk factor in the models evaluation metrics which was used for calculating the utility score.

9. References

David Guidos (8th December 2020). XGB-Starter.
https://www.kaggle.com/wilddave/xgb-starter

Yirun Zhang (9th December 2020). Jane Street: Neural Network Starter.
https://www.kaggle.com/gogo827jz/jane-street-neural-network-starter

Ekeany. Boruta-Shap
https://github.com/Ekeany/Boruta-Shapreadme