Community Detection Based Feature Selection for Market Prediction

Nathan Simonis,
Ludvig Wärnberg Gerdin

***Abstract*—This study examines the usefulness of community detection using the Louvain method on the Jane Street market prediction challenge on Kaggle. The objective of the challenge is to use machine learning techniques to determine whether one should act on trading opportunities. Using the resulting communities, two methods of feature selection are evaluated: A cluster importance weighted approach and a sub-cluster aggregation approach. The utility scores of the methods were 358.6799 and 256.2080, respectively. Using all features, a utility score of 418.1987 was computed.**

I. Introduction

It has been documented that the NYSE receives 4 to 5 terabytes of data per day (Groenfeldt, 2013). This ever-increasing quantity of data requires powerful methods to analyze it in an automated way. With the advent of the increase in computational power and storage solutions, it is now possible to process very large datasets in machine learning applications to detect meaningful patterns.

However, these algorithms suffer when the dimensionality of the data is large (Ruppert, 2004). It is therefore necessary to select variables that have strong explanatory power. This allows the researcher to understand the data better, reduce the usage of memory, train the algorithms faster and generate models that have greater generalization power. This is performed by removing irrelevant or redundant features.

Mitra et al. (2002) tackled the issue of removing redundant features by performing unsupervised clustering. Their approach to feature selection is done by selecting one feature per cluster. They compare their method with others and obtained comparable information loss while decreasing the dimensionality of the data.

Man and Chan (2020) removed irrelevant features by selecting the top k features that yield the minimal estimate of the generalization error. To compute the feature ranks, they compare 3 methods: Mean Decrease Accuracy (MDA), LIME, and SHAP values. They find that LIME was the most stable method but did not notice any performance difference in the accuracy of the predictions. Using any of the aforementioned methods gave a better out-of-sample performance.

This study examines the selection of variables using unsupervised learning from community detection. The following section describes the data set we used. We then present the different methods and models. The results section compares the two approaches we take to reduce the dimensionality of the data to the benchmark. This is followed by a discussion of the limitations of our work and future research.

II. Data Analysis

A. Contents

The data comes from a competition created by the quantitative trading firm Jane Street on the Kaggle platform (Jane Street Group, 2020). Two files were provided to us. Firstly, a spreadsheet of 2390491 rows and 138 columns. Each line represents a trading opportunity. Among the columns, a date index is given from day 1 to 500, the variable "resp" represents the return on investment of the opportunity, "weight" represents the size of the opportunity. When multiplying "resp" and "weight", we obtain the return of the trade. We are also given other "resp" variables ranging from 1 to 5. They represent the return on investment if the trade had been maintained for a longer period of time. The rest of the columns are explanatory variables. Due to their proprietary nature, we do not know what the features are or the specific date of when they were recorded. A second spreadsheet informs us of the dependencies between each variable. It allows us to know if a variable has been used to calculate another.

B. Feature types

By observing the features, 4 main types can be identified:

- Linear

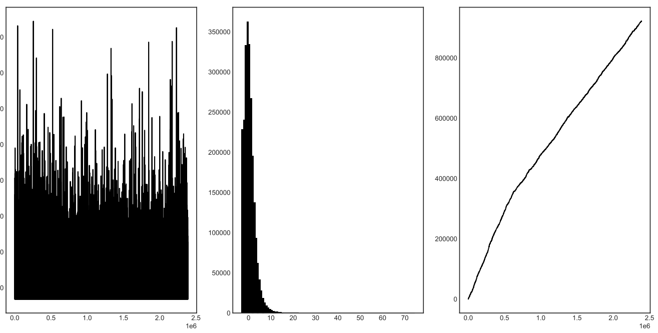

Figure 1. Plot of **feature 1** with respect to time (left), plot of the distribution (center), plot of the cumulative sum with respect to time (right)

- Noisy

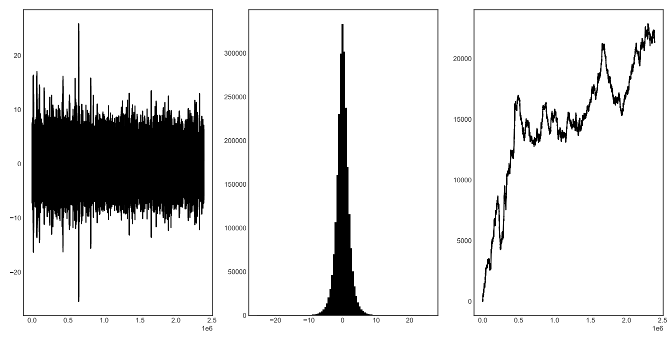
Figure 2. Plot of **feature 3** with respect to time (left), plot of the distribution (center), plot of the cumulative sum with respect to time (right)

- Logarithmic

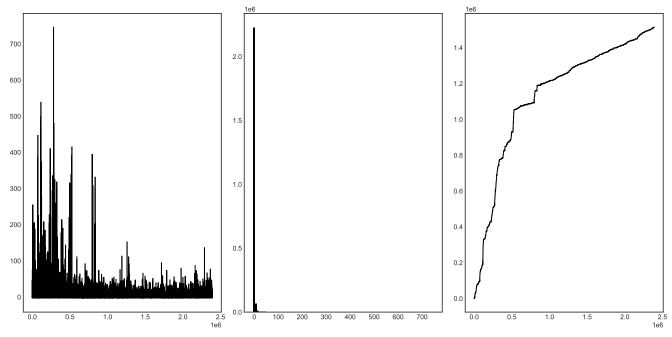
Figure 3. Plot of **feature 55** with respect to time (left), plot of the distribution (center), plot of the cumulative sum with respect to time (right)

- Negative

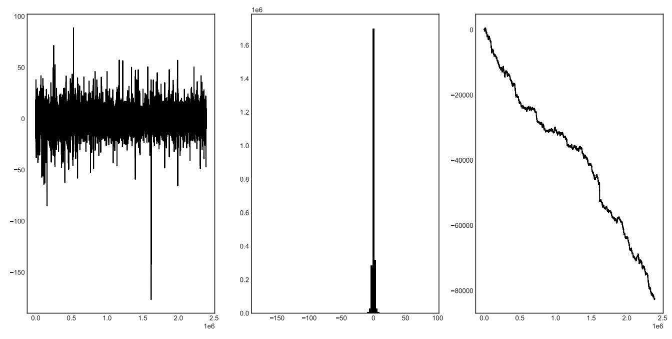
Figure 4. Plot of **feature 82** with respect to time (left), plot of the distribution (center), plot of the cumulative sum with respect to time (right)

This gives us a first intuition that it would be interesting to create groups of these different features.

C. *Missing values*

There is a fair amount of missing data in what has been provided. By visualizing where those are, we see in Figure 5 that they don't appear to occur at random.

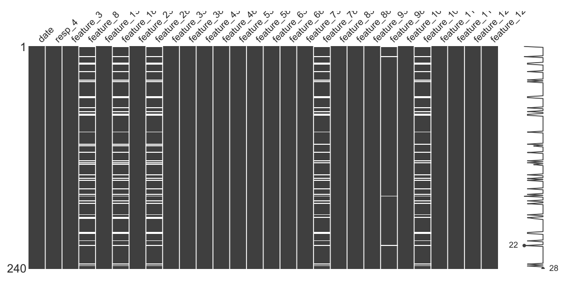
Figure 5. Missing values with respect to time at an interval of 10000 data points.

Thus, instead of dropping the rows that have missing data, we decide to replace them with common summary statistics. In the cross-validation process, the imputation of missing data by the column means or medians are considered as parameters.

D. *Outliers*

In order to analyze potential outliers, we calculate the z-score for our explanatory variables.

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

where μ and σ are the sample mean and standard deviation, respectively. We then observe the number of points that deviate from the mean by a threshold $\tau = 4$. From Figure 6, we can see that this number is very large. However, given that we are dealing with unknown explanatory variables and an unknown trading strategy, we believe that outliers may contain critical information.

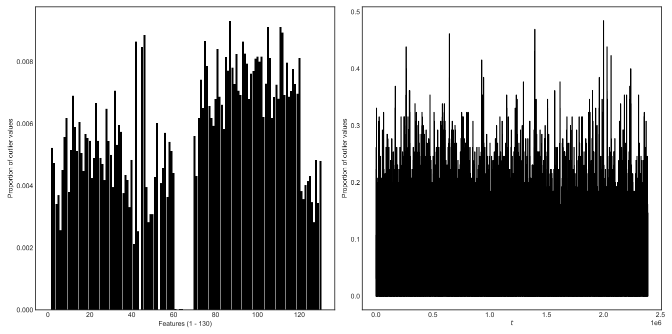
Figure 6. Proportion of the features (left) and rows (right) that have outliers with $\tau > 4$

E. *Correlations*

By calculating the correlations between the explanatory variables, we see that there are indeed redundant variables that correlate almost perfectly with other variables (See

Figure 8 in the Results section). This justifies our choice of feature selection and the procedure to deal with this redundancy will be explained in further sections.

F. *Target variable*

There was no pre-defined target in the sample. Therefore, we defined a binary target where a positive return on a trade corresponds to a 1, whereas zero or negative return corresponds to 0.

III. Models and Methods

IV. Models

A. *Random Matrix Theory*

The empirical correlation matrix is computed based on finite, non-deterministic samples. As a result, it contains a certain amount of noise. This has an unfavorable impact on the calculations made from it (de Prado, 2020).

From the Marcenko-Pastur (MP) theorem, a matrix of random observations has eigenvalues λ that converge to the MP probability distribution function. The maximum and minimum expected eigenvalues are denoted as:

$$\lambda_{\pm} = \left[1 \pm \sqrt{\frac{N}{T}}\right]^2$$

Thus, the eigenvalues of the correlation matrix $\lambda \in [\lambda_-, \lambda_+]$ can be considered to be mostly due to random noise and $\lambda > \lambda_+$ are considered to represent meaningful information.

The matrix can thus be decomposed and written as:

$$C = C^{(r)} + C^{(s)}$$

where $C^{(r)} = \sum_{i:\lambda_i \leq \lambda_+} \lambda_i v_i^T v_i$ corresponds to the part of the matrix containing random noise.

The cleaned correlation matrix $C^{(s)}$ is therefore computed by removing the random component of the original matrix.

$$C^{(s)} = C - C^{(r)}$$

B. *Louvain method*

The Louvain method is a community detection algorithm that aims to maximize modularity, a measure of the density of links inside clusters compared to the links between networks (Blondel et al., 2008). Modularity is defined as

$$\frac{1}{2m} \sum_{i,j} \left[w_{ij} - \frac{k_i k_j}{2m}\right] \delta(c_i, c_j)$$

where w_{ij} is the weight on the edge from node i to node j, $k_i = \sum_j w_{i,j}$ is the sum of the weights connected to node i, c_i is the community of node i, $\delta(x, y) = 1$ if $x = y$ otherwise 0, and $m = \frac{1}{2} \sum_{ij} w_{ij}$ The algorithm iterates over two steps: First, the modularity score is maximized locally, and second it aggregates nodes that are within the same community to bigger communities. The algorithm continues until no further increase in modularity is possible. At first, each node in the network represents a community.

C. *Clustered Mean Decrease Accuracy*

Breiman (2001) proposed the MDA as a measure of feature importance. First, a performance estimate of the predictions is computed by the cross-validated accuracy using the original features. Second, one of the features is randomly shuffled and the cross-validated performance estimate is computed again. The mean decrease in performance is the difference between the first performance estimate and the second performance estimate. We say performance estimate here, as the method is not restricted to using ”accuracy” but can in fact be any model performance estimate.

López de Prado (2020) proposed a way of calculating the importance of clusters or communities by utilizing the MDA. Rather than shuffling only one of the features at a time, the clustered MDA shuffles all features in a particular cluster before calculating the performance estimate. In our setting, we chose to calculate the MDA using the utility score as a performance estimate (see section VI).

D. *Light Gradient Boosting Machine*

Ke et al. (2017) proposed a light gradient boosting machine, henceforth referred to as lightgbm, which is a tree-based model. It differs from other implementations of tree-based models in some respects. For example, it relies on histogram-based algorithms to increase training speed and reduce memory usage, and it will grow the trees leaf-wise rather than level-wise.

Using tree-based methods, we can calculate the importance of features. With lightgbm, we achieve this is by calculating the number of times a feature is used to split the data across all trees (Korobov, 2017).

V. Community detection and feature selection

To use the result from community detection for supervised learning, one has to extract features from the resulting communities. There are two major ways of doing this. The features are extracted as-is from the clusters or are combined to define an aggregated feature space. Both these methods of feature selection were validated.

The first method was conducted in three steps. Firstly, feature importances were computed using the method described in section IV-D. Secondly, community detection was conducted, and clustered MDA was computed to determine the community importance. The Louvain algorithm was carried out 100 iterations, and in each iteration we noted what community the features belonged to. In the end, each feature was attributed to the cluster that it most frequently belonged to. The rationale was to get a more robust estimate of the communities. Thirdly, $n_i, i = 1, \ldots,$ features were picked from each cluster i based on the cluster importance. The variables n_i were defined to be the rounded absolute proportion of MDA of cluster i,

$$n_i = \frac{|MDA_i|}{\sum_i |MDA_i|}$$

The second approach considers an aggregation of explanatory variables. Here the detection of communities is carried out twice. For each community initially found, a new inter-community correlation matrix $C^{(s)*}$ is established and the community detection is carried out once more. This allows us to have sub-communities: the features present in each sub-community are aggregated into a sum, an average, or a product. The selected aggregation method is part of the cross-validation process.

We now have new features $\tilde{X}$ which are an aggregation of a sub-community of the original feature space.

VI. Model evaluation

Model selection and model evaluation were conducted using the utility score as defined by the competition hosts (Jane Street Group, 2020). Let i denote the date and $a_i j$ denote our predicted action at time point j for date i. For date i, define

$$p_i = \sum_j w_{ij} resp_{ij} a_{ij}.$$

Then we have,

$$t = \frac{\sum_i p_i}{\sqrt{\sum_i p_i}} \frac{\sqrt{250}}{|i|}$$

where $|i|$ is the number of unique dates in the fold. The final utility is given by

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

A. Cross-validation

The data was partitioned into two. The first partition is denoted the train-validation partition and the second is denoted the testing partition. The train-validation and testing partition consisted of data points from 400 and 90 days, respectively. A gap of 10 days, which is motivated below, was placed between the partitions. In the train-validation partition, we ran cross-validation using a walk-forward K-fold grouped time series split with a gap. The split is illustrated in Figure 7.

When using walk-forward K-fold time-series cross-validation, the training and test folds are rolling forward in time. This is done by setting an upper bound on the training fold and the test folds. We selected approximately 9 months, i.e. 189 unique dates, for the training fold. The maximum size for the validation folds was set to 3 months, i.e. 63 unique dates. Note here that we assume the trading months to be 21 days. The rationale was that the utility score is defined using 250 days (see the previous section), which correspond to 21 trading days in a month.

We grouped the cross-validation by the date variable to not split samples from one date into two separate folds. A gap of 10 dates were set between the training and validation set in each cross-validation fold. Since there is most likely auto-correlation for the features, the gap reduces the risk that the information contained in previous values in the training folds does not spill over in the validation fold and test partition.

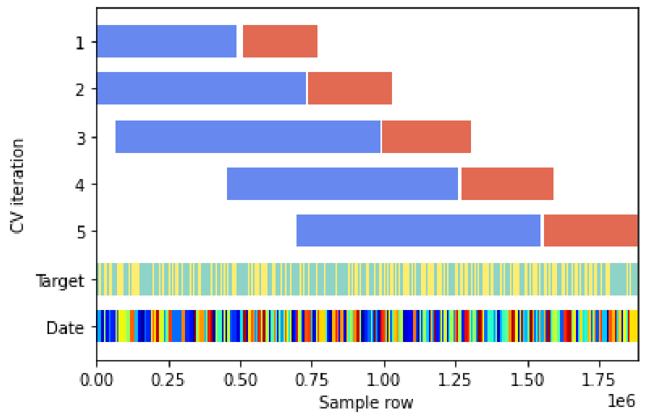

Figure 7. Training and validation folds for each iteration of cross-validation. The blue bars are the training folds whereas the red bars are the validation folds.

The hyper-parameters only changed the data-preparation process, and thus the default hyper-parameters were kept for the lightgbm model during cross-validation. The hyper-parameters that were considered are presented in Table I. The utility scores were averaged over all 5 validation folds for each combination of hyper-parameters. The set of parameters that achieved the best average utility over the folds were retained and re-run on the full training-validation partition. The lightgbm model was finally fitted on the full train-validation partition and evaluated on the test partition. The methods were compared to using all features for prediction.

Table I
Hyper-parameters considered in cross-validation

Cluster-weighted Feature Selection		Sub-cluster Aggregation	
Imputation	Number of features, K	Imputation	Aggregation
Mean Median	Number of clusters 10, 20, 30, 40	Mean Median	Sum Product Division

VII. Results

The best-performing configurations of hyper-parameters are presented in Table II.

Table II
Best performing hyper-parameter configuration for each feature selection method.

Cluster-weighted Feature Selection		Sub-cluster Aggregation	
Imputation	Number of features, K	Imputation	Aggregation
Mean	20	Mean	Sum

The procedure that we describe below was conducted on the training folds in the cross-validation as well as on the full training-validation partition. Firstly, we detect communities among our explanatory variables, which is done using the correlation matrix as the distance measure. Before calculating the correlation matrix, the variables are normalized using Equation 1, and the empirical correlation matrix is estimated and cleaned. The empirical correlation matrix and the cleaned correlation matrix from the training-validation partition are presented in Figure 8.

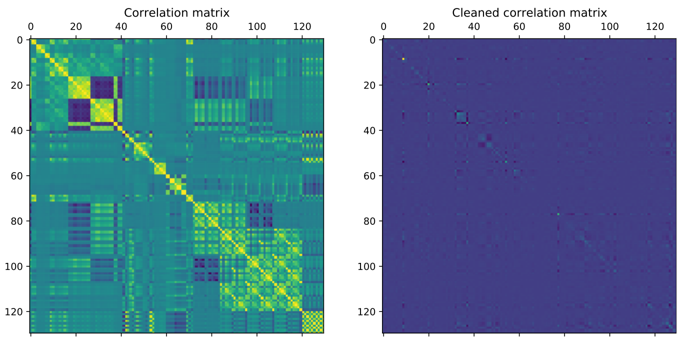

Figure 8. Empirical correlation matrix (left) and cleaned correlation matrix (right).

After running the Louvain clustering algorithm over our cleaned correlation matrix, 4 communities were detected; the structure of which is presented in Figure 9.

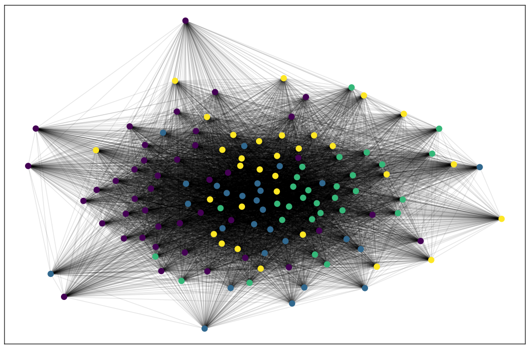

Figure 9. Output of the Louvain Algorithm after 100 iterations. The nodes are positioned using the Fruchterman-Reingold force-directed algorithm.

For these communities, the community importances were computed. The importance values are presented in Figure 10. Community 2 is the most important with an MDA of approximately 125, whereas the least important community is number 1 with an MDA of approximately 110. From these communities, 4, 5, 6, and 5 features were selected from cluster 0, 1, 2, 3, respectively. For community 1, feature number 0, 1, 2, 3 were selected. Feature number 9, 11, 30, 32, 33 were selected for community 2. Feature number 39, 40, 41, 42, 43, and 44 were selected for community 3, and finally feature 70, 72, 74, 77, and 79 were selected for community 4. Note the pattern that clustered features generally seem to be adjacent to each other in terms of their feature number.

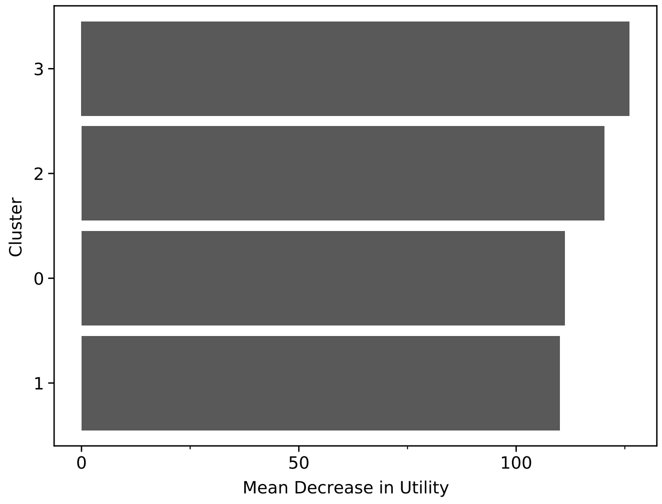

Figure 10. Cluster / community importance according to clustered MDA.

For the sub-clustering approach, the community detection procedure was repeated within each community. The relative importance of the sub-clusters are presented in Figure 11, where the dotted line correspond to the mean importance. When summing the sub-clusters, we end up with 26 features.

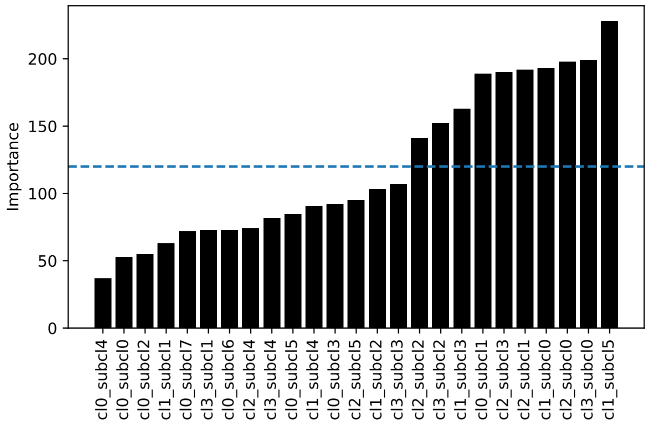

Figure 11. Feature importances of $\tilde{X}$

As a baseline model, we train a lightgbm using all the explanatory variables. The performances of the 3 methods are presented in Table III. It can be seen that the first method described achieves the best out-of-sample performance. However, neither the first nor the second approach yielded better results than the baseline method.

Table III
OUT-OF-SAMPLE UTILITY SCORE FOR BASELINE AND CANDIDATE APPROACHES.

All features	Cluster-weighted Feature Selection	Sub-cluster Aggregation
418.1987	358.6799	256.2080

VIII. DISCUSSION

A. Methodological Considerations and Future Research

Different methods of increasing the speed of the hyper-parameter tuning, training, and validation were tested. Firstly, the training sample was converted into a feather format to increase the speed of loading the data. Secondly, methods from the `dask` library were tested as a part of the cross-validation procedure. However, using delayed objects is not a valid input to the fit method of the lightgbm classifier. We determined the refactoring that was needed to be worse for the readability of our approach, and thus proceeded instead to use the joblib library to run the cross-validation in parallel. Secondly, methods from the `numba` package were applied to increase the speed of calculating the utility score.

The cross-validation procedure was run as an explicit loop. The main advantage of this approach is readability. Another approach would be to define separate classes for each data preparation step using the FunctionTransformer class from the scikit-learn package. That way the data preprocessing can be defined as part of a Pipeline class, and run in a distributed fashion through dask_ml. Besides, by utilizing the dask_ml package, other ways hyper-parameter search methods could be explored. For example, incremental approaches for hyper-parameter search would alleviate the large memory usage that is caused by running a parallel, exhaustive grid search.

There are two general ways of doing time series cross-validation: rolling forward cross-validation and expanding window cross-validation. As presented in the methods, we used the former. The advantage of rolling forward cross-validation compared to expanding window cross-validation is that the procedure adapts to newer information rather than taking old information into account. Since financial time series data is non-stationary, the information contained in newer data should be more informative for prediction purposes.

The anonymized features of the sample forced us to make assumptions about the characteristics of the data. Several issues arise in the model selection and evaluation process. For example, our models may suffer from information leakage. One example of this were given in section VI-A. Consider another example that one feature may be a moving average of another. If the pre-specified gap is not large enough, then the information will spill into the validation folds and the test partition from the training partitions. The competition arrangers did release information on which features were dependent or derived from each other (see the `features.csv` sample), however, there was no information on how the derivations were made.

The use of the dimension reduction techniques and feature selection procedures presented in this report should be explored on samples that have labeled features rather than anonymized features. That way we would be able to prevent data leakage without making assumptions about feature dependencies. This would avoid losing out on information by enforcing too large gaps between the partitions. In addition, in practice, it would be interesting to see what kind of features allow us to make good predictions. This enables us to try to understand which signal is picked up by the algorithm. We could then try to add features that reinforce this signal. One of the methods used in this study picked several features from the clusters whose clustered MDA was high, whereas the other method instead combined similar features to reduce the feature space. The former method may seem counter-intuitive, as the features that are assigned to the same communities are closely connected, and therefore convey similar information to the model. In this sense, the latter method may seem a more appropriate method of feature selection. As presented in Table I, we did an experiment using only the most important cluster from each cluster but experienced that using several features from each cluster yielded better results.

We did not conduct any additional feature engineering or model tuning. We determined it to not be within the scope of the project, as the focus was put on the analysis of the effect of community detection on prediction. That being said, we acknowledge that additional feature engineering may better the performance of our model. For example, as presented in class, the Louvain method could be utilized to detect communities with respect to time, and then utilize the resulting state as a feature in the model. Further, adding missing indicators for the missing data may allow the model to utilize the apparent non-randomness of the missing data (see Figure 5). Additionally, model hyper-parameter tuning may be able to further improve the performance.

The Louvain method has been shown in comparative studies to be one of the most well-performing community detection algorithms in terms of, for example, computation time (Yang et al., 2016). However, there are shortcomings with the model, and novel methods of community detection have been proposed to alleviate these issues. For example, Traag et al. (2019) shows that the Louvain algorithm may in cases generate communities that are internally disconnected. The authors instead propose the Leiden algorithm, which is guaranteed to find γ-connected communities after each iteration, where γ is the resolution parameter in the modularity quality function.

IX. Conclusion

A cluster importance weighted approach to feature selection and feature selection by aggregation in sub-communities were evaluated and compared to the baseline of doing no feature selection. Utility scores of 358.6799 and 256.2080 were computed for the two methods, respectively. Using all features, a utility score of 418.1987 was computed. Thus, neither the first nor the second candidate approach performed better than the baseline. That being said, the approaches that were explored utilized significantly fewer features than the baseline approach. Additionally, the approach to data preparation was rather shallow because we do not know what the features are. As such several aspects of the data preparation, feature selection, and model fitting pipeline could be improved.

References

Blondel, V., Guillaume, J., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*, 10008.

Breiman, L. (2001). Random forests. *Machine Learning*, (45), 5–32. https://doi.org/10.1201/9780429469275-8

de Prado, M. L. (2020). Machine learning for asset managers.

Groenfeldt, T. (2013). At nyse, the data deluge overwhelms traditional databases. *Forbes*.

Jane Street Group. (2020). Evaluation. Retrieved January 3, 2021, from https://www.kaggle.com/c/jane-street-market-prediction/overview/evaluation

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 3146–3154). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Korobov, M. (2017). LightGBM. Retrieved January 7, 2021, from https://eli5.readthedocs.io/en/latest/libraries/lightgbm.html

López de Prado, M. M. (2020). *Machine Learning for Asset Managers*. Cambridge University Press.

Man, X., & Chan, E. (2020). The best way to select features? *ArXiv*, *abs/2005.12483*.

Mitra, P., Murthy, C. A., & Pal, S. (2002). Unsupervised feature selection using feature similarity. *IEEE Trans. Pattern Anal. Mach. Intell.*, *24*, 301–312.

Ruppert, D. (2004). The elements of statistical learning: Data mining, inference, and prediction. *Journal of the American Statistical Association*, *99*, 567–567.

Traag, V. A., Waltman, L., & van Eck, N. J. (2019). From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, *9*(1), 1–12. https://doi.org/10.1038/s41598-019-41695-z

Yang, Z., Algesheimer, R., & Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, *6*(March). https://doi.org/10.1038/srep30750