Predicting Successful Trades In Volatile Markets

Emeka Amadi, eamadi@umich.edu, Stefan Bund, bund@umich.edu

Abstract

We document machine learning that differentiates good and bad trades in volatile financial markets. An applied solution is detailed with 90%+ results. The strategy entails sequence data mining, synthetic minority resampling, and the use of KMeans clustering to achieve classifiable features before classification. Model features imply that markets strongly associate price swings with sell-side order activity, as precursors to upward price swings. We demonstrate the use of several classifiers reaching this accuracy.

Introduction: Applying Data Science in Trading Problems

Typically, 70% of daily market volume is initiated by autonomous, algorithmic agents.[3] These 'bots', Computer Trading Algorithms (CTA), or Artificial Intelligence agents are risking institutional funds and accumulating more importance in the financial industry. At this juncture, this is enabling firms to exit actual cities where trading is done, such as New York and Chicago.

Digital exchanges disclose snapshots of the pending, unfilled orders in a limit order book, or LOB. The limit order book is a multifaceted source of data, detailing at what prices a commodity or digital equity is willingly sold, or purchased. For each order a volume is given. Since demand levels are known before a price rises or falls, a data science problem, given the LOB must consider these features. In our view, the limit order book provides statistical power, when sampled over time.

Prior Work

We are motivated by prior work by Sirignano and Cont, where deep learning methods uncovered signals within markets, leading to price change.[7] Possessing access to adequate stores of data is not in our purview. We borrow corporate data from one year of data collection, and do so under a feature representation where details are aggregated from raw market data. Thus, our use of a neural structure to uncover signaling is not possible.

We are dismayed by ongoing efforts to uncover similar price signals from qualitative, natural language factors such as news stories and other journalism, as used in Riordan. We fear that entities which own publishing outlets can sway their content, to bias these models. Hence, we are seeking a method of identifying time series events (precursors) which serve a similar purpose as Riordan's use of natural language, but in the limit order book domain.[6]

We are keenly following the work of Zheng/Moulines, who approach the same problem of identifying price spikes.[5] Their method emphasizes lasso and linear methods to identify regressions which precede price spikes. We admire this work, and attempt to borrow a lasso method. However, we are more interested in the use of data mining to isolate moments that lead to price change, and are not using linear regression in our work. We aim to capture some of the accuracy of Riordan's model, where conditions lead to price spikes, but do so from a sequence of measurable quantitative features.

Business Constraints We Must Observe

An additional constraint exists within our work. Our focus is on traders who must initiate trades which finish quickly, are not waiting for a long duration to complete a profit, and engage the market daily. This is a risk context for day traders, who exploit short term scenarios. Seeking short term trading horizons are notoriously difficult, given the inhuman pace of order books. However, the temptation exists to try to tap into or harness the daily permutation within prices.

Our project assumes several facts about markets, driven by limit order books.

1. **There are limited features provided by exchanges**. New features must be engineered.
2. **Data exists as a time series**. However, markets present themselves as sequences, where one period of activity is followed by another.

3. We contend that **markets are fundamentally composed of dull periods** of little change, followed by drops.
4. **Sometimes**, **dull periods are followed by surges in price**. These are the most valuable situations within markets.
5. Thus, **a market is a sequence of context, then a result**. As such, the features specific to tradeable, profitable sequences are knowable.

Our study assumes that the profile of a surging market can be observed, statistically, using facets of the limit order book. Hence, our machine learning aims to learn those conditions which result in surges, and identify them, given new and unseen data.

A Useful Lexicon for the Trading Industry

Because the reader may not have professional experience in trading commodities or financial goods, a lexicon is presented here. The authors use these terms in the paper.

1. **Precursors**. Moments before surges. Identified as periods of discontinuous positive momentum below a threshold value, for all samples.
2. **Surges**. Moments in the markets where multiple observations have continuous price momentum. Surges are values in terms of their ability to fulfill price goals for all price points within a precursor. Hence, when a precursor occurs in a market, are the surges able to deliver profit for all price points in the precursor? This is what we define as a desirable trade, for which we will search and classify sequences.
3. **Sequences**. A two part combination of a precursor, followed by a surge. In the sequence there are multiple observations, with prices and types of buy orders.
4. **Bids**. orders to buy a commodity
5. **Asks**. orders to sell a commodity
6. **Capitalization**. The amount of an order, multiplied by its price, expressed as a sum dollar value, of one or more orders. Represents value at risk, during a trade.
7. **Targets**. The goals for a trade, where a commodity is purchased, then sold at a targeted price. Assumes a profit goal, expressed as a percentage.

Design Considerations for High Frequency Trading

Hedge funds and institutional trading is an instance of positioning capital on a short term basis. This differs from all other forms of investing, where a tranche of money is intended to be used for day-long, or minutes-long use. The trader might only do one trade per day, and this may only last for a few minutes, under the right conditions.

In most conditions, the trader is an autonomous agent, intended to monitor markets for a once-per-day opportunity. Hence, forecast models are excellent for long term positions. Also, neural networks are used to model very short term positions, but with marginal efficacy to predict outcomes.

We have found that the use of neural networks is ineffective, due to the novelty of the research. Our data set reached one year in duration, and contained upwards of 284,000 rows, but this in itself was an inadequate data set for a neural solution.

Hence, our constraints are simplified as follows:

1. Identify market conditions that are associated with extremely short term bursts in price values.
2. The surges, chosen for the study values, should represent moments when the purchase price of buy orders was satisfied by the sell price, in the surge. Hence, surges are evaluated based on their ability to deliver profit, based on the purchase price, during the moments before it.
3. Surges must deliver immediate value, and create the least risk for the trader. They must finish in a hasty, no-wait fashion.

Ethical Implications of Algorithmic Trades

For professional money managers, the ethics of executing trades without human supervision is a 20th century conundrum. There is clear risk of harm to the client, who entrusts funds, but there are natural upsides to using highly accurate predictions. Methods which lack explainability lack ethical status, as exemplified by recent decisions by JP Morgan.[8]

However, where investors are informed, and algorithms deliver explanation and evidence, ethical trading can take place. In this work we aim to document both.

The Distribution of Underlying Data

The most desirable kind of trade occurs during moments where the following features are present. Precursors associated with the best surges contain a distribution of values like so. As we will discuss, our data is limited in features. It is also low variance, inside each feature.(fig 2) Highly valuable target classes are easily confused for undesirable trades by classifiers. Significant imbalance exists between low-value trades and desirable ones. Hence, our feature engineering had to overcome this, and resampling was used to synthetically create more balanced, target features. Using a binned label, we identified the relative scarcity of desirable sequences, which occupied class ‘10’ in this histogram.(fig 1)

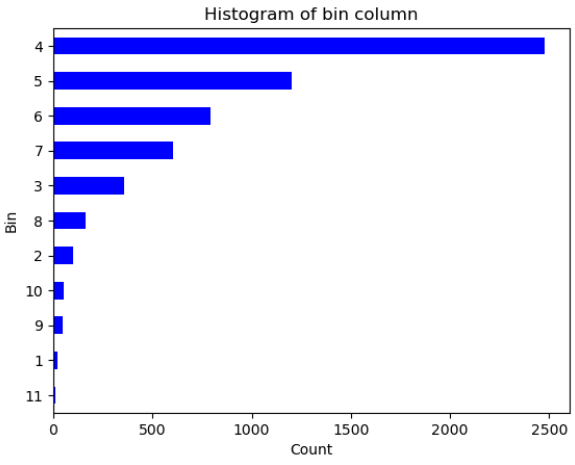


Figure 1. A Minority/Exceptional class, ‘10’

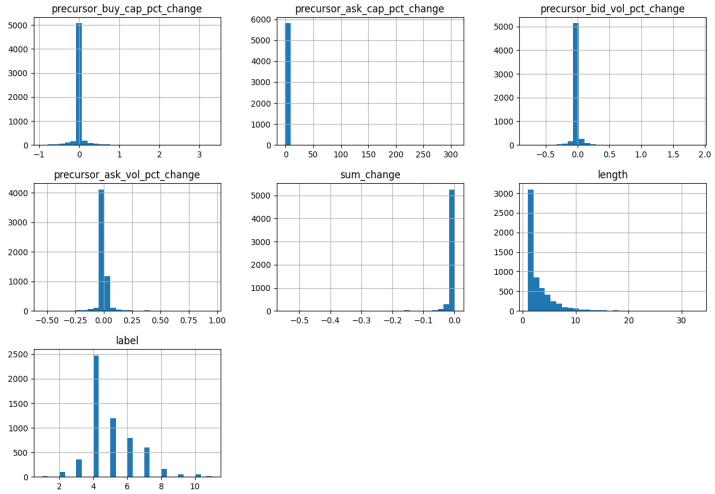


Figure 2. Global distribution, all classes

Design of Experiments

Given the breadth of computational resources, we underwent a two-pronged approach to discover an optimal machine learning method. Fundamentally, we had to deliver decision making power for our trader. This could take a binary yes/no form or a nuanced, graded set of signals. Thus, we organized the study around three key parameters. Machine learning methods would be either supervised or unsupervised, and features would be balanced, imbalanced, and labeling would be binned or clustered, depending on an unsupervised process.

	SUPERVISED		FEATURE ENGINEERING			UNSUPERVISED
test / study	**binary**	**multiclass**	**balanced**	**imbalanced**	**binned**	**cluster feature**
1		x		x	x	
2	x		x		x	
3		x	x			x

Our approach was to integrate supervised and unsupervised learning into two different phases. Initially, we envisioned utilizing clustering as a strategic approach to categorize profitability. Over time, this evolved into adopting clustering as a preprocessing step, which would enhance downstream classification. We found this to be true, in ways we did not foresee. In general we aimed to do clustering before classification, trying different types of labeling, and balancing our data set to better support our desired, minority classes.

We had to design a solution to separate high-performance, highly desirable classes, which were vastly outnumbered. The number of features at our disposal was at a minimum, and the interquartile variance within features was minimal. Additionally, we sought to choose three families of classifiers (binary, tree based, and category based) in order to deliver an optimal result.

Results

Scoring

Our final pipeline reached 90%+ accuracy in test/train conditions where data was resampled using a synthetic minority technique. In this report we are recommending the following pipeline steps for 90%+ accuracy.

- Group time contiguous time series observations into two types: precursor and surge, based on the appearance of above-threshold price variance. Contiguous positive momentum above threshold qualified a series as surge.
- Standardize the Representation of raw exchange data in percentile change format. This represented each precursor condition in general terms, independent of changes in magnitude.
- Bin target values according to a binary distribution, where only the target surges are identified in a one-hot format.
- Discover clustering techniques with high degrees of silhouette, where clear decision boundaries could be identified downstream by a classifier

- In one instance, we used the clustering label as a feature to reach our highest prediction power, while retaining the bin as our y_label. We have this feature engineering technique to recommend, for best results.
- Use a simple classifier (KNNeighbors) after using a simple 2 cluster KMeans cluster
- Use a GridSearch cross validation to identify best performing parameters, for each major choice of machine learning method
- Use synthetic minority class resampling, in order to magnify instances of highly desirable precursor conditions, based on their surge magnitude.
- Binary classifiers (KNeighbors) and tree based classifiers were the most apt to score highest
- Category Boosted classifiers participated augmented voting classifiers, in other high scoring ensemble methods

Based on our comparative study, we affirm that a pipeline where unsupervised learning is used to engineer features is valuable in achieving best-results in binary classification where underlying data is low-feature, low variant, with infrequent target classes.

Error Analysis

In our highest performing classifier, our clustered features led to highest accuracy, and the least false positives.

Learning Curves

In our highest performing multi-class classifier, we found that learning curves where more data was added, increased accuracy.

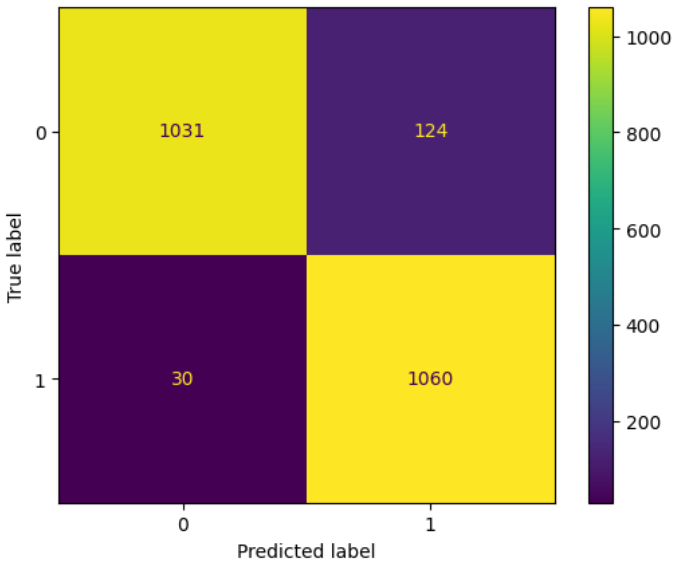


Figure 3. Confusion Matrix

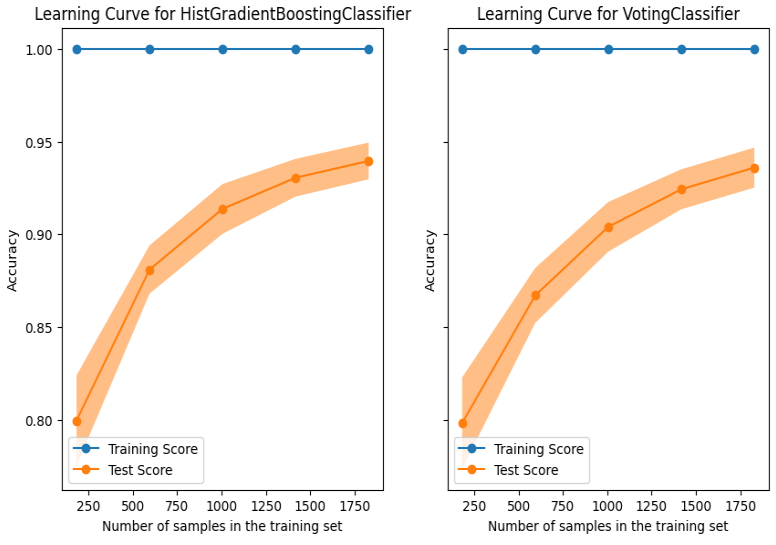


Figure 4. Learning Curve

Discussion

Feature Engineering: Using Data Mining to Expose Sequences in Markets

Once per three to five second interval, market exchange data is downloaded then aggregated into the following statistics. Typically, there are 13,000 orders per order type in the order book, with each order following this schema. We obtain data approximately every 15 seconds via websocket from Coinbase using a node.js API.[4,11]

Data Schema

The data is from the cryptocurrency Avalanche, known as the ticker symbol AVAX-USD. Our javascript application consumes the raw JSON, then delivers a succinct summary of the book snapshot for each observation. We visualize this basic data schema here.

- Bc: bid capitalization
- Ac: ask capitalization
- Tav: total ask capitalization
- Tbv: total bid capitalization
- Mp: midpoint, or price of the commodity, at observation time

In order to standardize these values, a percent change statistic is generated, with the ratio of one value to the next. Hence, the machine learning algorithms have a set of floating point values to use instead of the raw, absolute values, thus increasing the opportunity for an estimator to learn a pattern. The data used in our pipeline is available under the lob_caps folder in our github.[12, or here]

This data was initially analyzed during Milestone 1 by Bund.[13] However, the data is now four times the original size, and is analyzed for the first time in a machine learning context. There is a block of code in our data prep that is used in Milestone 2 to load multiple .csv files, and is the sole form of overlap between the projects.

Given this first principles approach, the market is explained as a precursor and a surge.(fig 5) Since we are seeking the precursors which result in surges, we first organize one year of sampling multiple times per minute. A total of 284,000 samples become available with the above schema.

From this schema, we interpret periods of continuous non-positive momentum as precursors, and periods of continuous momentum as surges.(fig 4) We group these periods together as sequences, and derive bin values for the edges of the `surge_target_met_pct value`. This value describes the percentage of prices, during the precursor, which reached a target value during the surge.

The closer the percentage is to 1.0, the more that surge provided profit for each order in the precursor. This provides us with ample room to label each precursor. Our machine learning strategies were designed to optimally discern profitable sequences from risky, or unprofitable ones. Our narrative describes how we evolved a pipeline to do this, and the evolved big-picture statistics we needed to correctly do so.

Once we identify two -part sequences we quantify precursors by their surge value, which offers us the opportunity to classify them in our machine learning. Simply put, the larger the

surge, the more critical the precursor is to our study. The full breadth of features we engineer during that process is listed here.

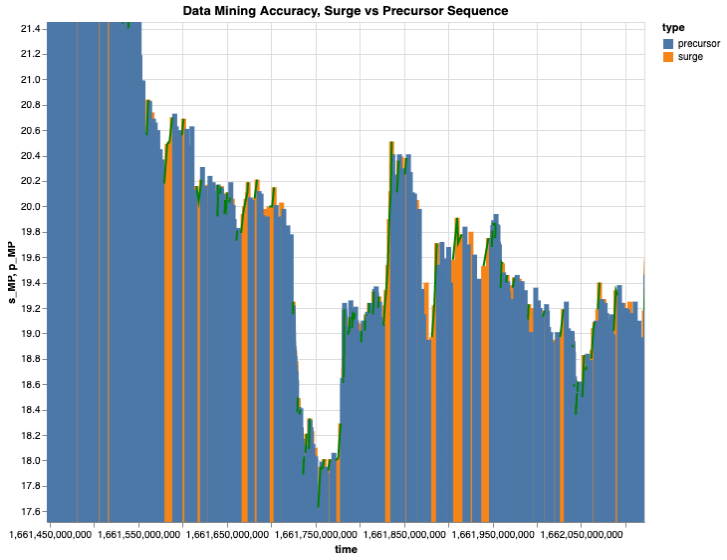


Figure 4. Proof of Surge identification

```
7861        surge
7862        surge
13991    precursor
13992    precursor
13993    precursor
13994    precursor
13995    precursor
13996    precursor
7863        surge
13997    precursor
7864        surge
7865        surge
13998    precursor
13999    precursor
14000    precursor
7866        surge
14001    precursor
14002    precursor
7867        surge
7868        surge
14003    precursor
14004    precursor
14005    precursor
14006    precursor
14007    precursor
```

Figure 5. Python outputs

Using Unsupervised Learning for profitability

After a comprehensive phase of data processing and feature engineering, we delved into clustering our data through an array of methodologies including KMeans, Birch, and Mean Shift. Upon evaluation using the silhouette score, KMeans emerged as the leader, registering a peak score of approximately 0.88. Figure 21 illustrates an optimized KMeans run, starting with 2 clusters, as a result of a grid search to pinpoint the ideal model parameters in respect to silhouette score. This figure reveals that despite a strong silhouette score, the score's inflation is largely attributed to the data set being channeled into two clusters, which is not our desired configuration. Ideally, we'd prefer a configuration close to the number of bins or labels in our supervised approach**,** which is twelve. This configuration can be traced back to the limited variation within our input features which displays the complexities and unpredictabilities of algorithmic approaches in financial markets.

Considering these insights, and with our primary objective of identifying sequences of lucrative trading opportunities in mind, we pivoted our approach. We decided that gauging clustering merely by cluster separation would be insufficient. Our aim shifted to discerning surge profitability via clustering. Hence, created an evaluation of cluster quality through the lens of profitability separation which can be seen in Code Block 1. While not foolproof, this approach allowed us a perspective on the disparity in profitability within a cluster set. By iterating through numerous methodologies and their parameters, we found that Birch, KMeans, and Mean Shift all had iterations with the top profitability scores. However, Mean Shift was dismissed from profitability scoring consideration due to its propensity to form 175 clusters, over 100 of which had a singular data point.

The KMeans vs. Birch Comparative Analysis:

In our quest to discern the best clustering methodology, both K-Means and Birch stood out as formidable contenders. Each method was evaluated based on three pivotal metrics: inertia, silhouette score, and cluster profitability.

Inertia:

K-Means (Figure 26): The "elbow" method suggests an optimal cluster range between 8 to 10. As clusters increase, the inertia reduction showcases diminishing returns, indicating that K-Means effectively segregates data up to a certain point.

Profitability Analysis:

K-Means(Figure 25): Peaks in profitability are observed at 8 clusters, beyond which there's a decline.

Birch(Figure 27): Displays resilience by maintaining higher profitability scores across varied cluster counts. Noteworthy peaks are observed at 6 and 10 clusters, underscoring its proficiency in recognizing patterns conducive to enhanced profitability.

Silhouette Score Analysis:

K-Means(Figure 25): Achieves an optimal silhouette score of around 0.88 with 6 clusters, indicating its prowess in efficiently distinguishing clusters for the given data set.

Birch(Figure 27): Presents a commendable performance with a peak silhouette score of around 0.85 for 6 clusters. This consistency across multiple cluster counts demonstrates its capability to identify cohesive clusters.

Table 1 and Table 2 present detailed metrics on silhouette scores and cluster profitability for KMeans and Birch. Notably, K-Means achieves a peak silhouette score of approximately 0.88, with Birch trailing closely at around 0.86. Both methodologies display their best performance with fewer clusters, further emphasizing a potential challenge in differentiating profitability. In terms of profitability, Birch takes a slight lead with scores cresting at approximately 0.21, while K-Means reaches a peak at about 0.19. Due to how close the results are between the two methodologies we contend that both methods offer distinct advantages and are comparably adept for cluster analysis.

Figures 23 and 24 offer illuminating insights into our analysis. Despite employing our most optimal scoring techniques, a closer look at the individual clusters reveals a limitation: there's an inability to distinctly differentiate profitability across clusters. A consistent observation, particularly in `surge_target_met` values, is their convergence towards a zero average in each cluster. The figures presented show these results in both the K-means and Birch algorithm, optimized using the best model parameters from an exhaustive grid search.

While the test set exhibits greater variability this variance is attributed to the reduced sample size per cluster, compounded by the presence of outliers. From this analysis, a salient revelation emerges: relying solely on clustering as a metric for gauging profitability might not lead to clear-cut conclusions regarding accuracy. However, the silver lining is that these clusters can potentially offer invaluable features which enhance our classification methodologies.

Using Clustering as a Feature Engineering Tool

Lastly we employed multiple resampling techniques to augment our dataset, with the aim of enhancing our supervised score. The code, as seen in code block 2, reflects our iterative exploration across varying cluster counts of resampled data using KMeans. Our findings, stored in the `sampled_df` dataframe, revealed that the KMeans with `n_clusters` set to 2 delivered the most favorable silhouette score. This result is evident from the first image where the `idxmax` function helps pinpoint the best-performing parameters. Moreover, when adopting an alternative cluster profitability metric, this configuration still demonstrated scores close to the top as seen in code block 2. Consequently, the labels derived from these clusters were integrated as input features into our subsequent classification methodology.

Model Iteration One: Using Multiclass Estimators in Binary Decision Problems

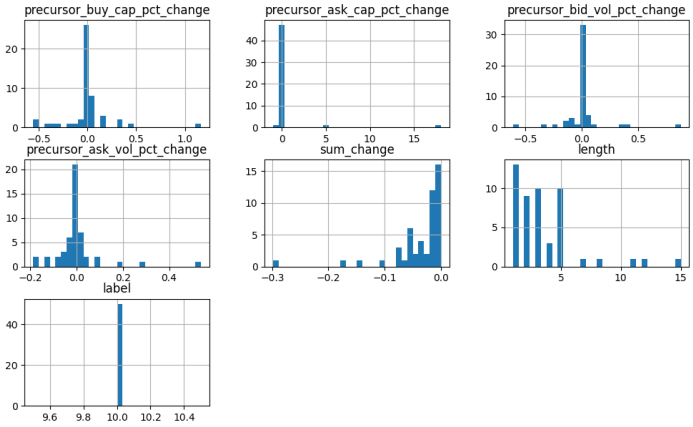
We created reasonably separated classes of data, observable by silhouette scoring.(Fig 7) The adaptation of clustering from a labeling process to a feature engineering step created the most successful type of our classifier, and allowed us to surpass the accuracy enabled via binned labeling.

We adopted tree based classifiers, capable of learning many classes, as well as a Bagging Classifier, then joined them into a voting classifier, to resolve any shortcomings.

Since we needed to discover the precursors needed to deliver complete alpha, we initiated a multiclass approach, where a multitude of surges were visible, within a many-binned labeling approach. We adopted a means to bin precursors by their associated surge.

Results expose profitable clusters of precursors, as an extreme minority class. The need to build more classification instances via synthetic means.

Low accuracy forced us to use feature analysis to discover the classifier which understood the best cases. But also, given the high degree of confusion over these most-desired cases, we needed a way to separate good and bad outcomes before we reached the classifier stage. We saw how one estimator could capture the minority class we sought.(fig. 11) **At this point we realized that we really only needed to discover one condition: whether we should trade the precursor or not.**



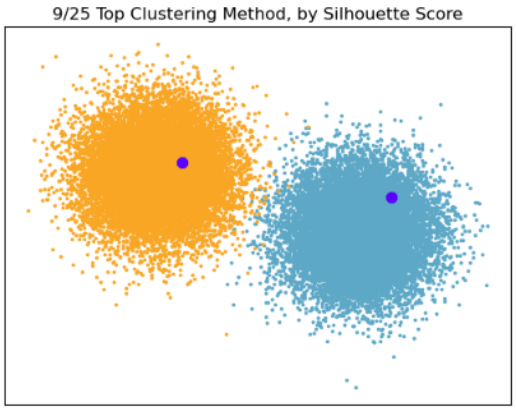


Figure 6. Feature Distributions in the Class ‘10’ we sought

Figure 7. Silhouette for Clustering

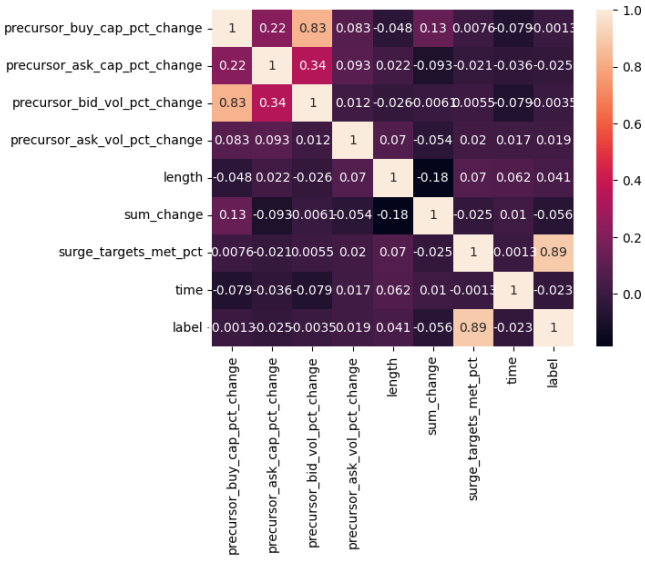


Figure 8. Correlated Features, pre classify

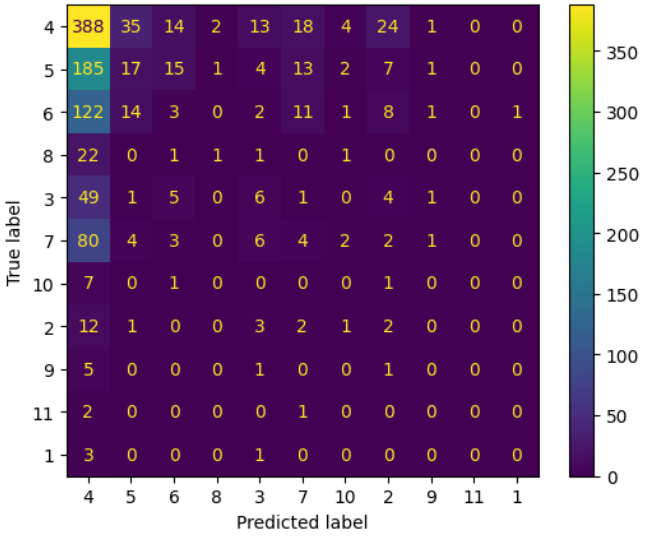


Figure 9. Confusion Matrix for the voting classifier

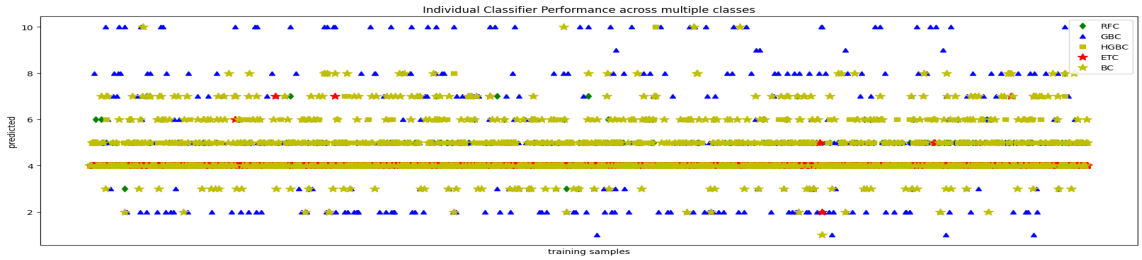


Figure 10. Voting Classifier capability to identify the most-desired class

Figure 11. Individual Classifier strength, identifying most desirable class

Model Iteration Two: Binary Estimators for Binary Problems

Our initial solution underperformed due to imbalance data which failed to recognize desirable minority classes. We sought a binary classifier operating on balanced data, where a

new binned label began use. In this new binning process, only the desired minority class received a ‘1’, while all others received a ‘0’. Our code is available here.

Synthetic cases were generated for the optimal class, 10, where at least all price points in the precursor reach a profitable target. The ADASYN algorithm was chosen for this task, as a means to build replicated renditions of the core ‘Class 10’ which embodied our target class.[11] We used a grid search across several potential classifiers, Logistic Regression, Bernouilli Naive Bayes, and KNeighbors, in hopes they would deliver an appropriate binary decision boundary.[5]

The combined binary bin label, balanced across the dataset, with the binary classifiers doubled our accuracy, as hoped.(fig 12) The KNeighbors classifier moved accuracy to 93%. A voting classifier we used, combining the above classifiers also had a 90% accuracy.

According to our SHAP analysis, the `precursor_ask_vol_pct_change` played the greatest role in classification. In Balanced Random Forest classifier we ran, chosen to harness minority class resampling, we achieved an 83% accuracy, where the volume of change in sales volumes played the largest role in assisting our classifier.(fig 13)

In this transition to binary binning, minority resampling and binary classification, learning curves for the KNN and Voting Classifier increased as more data was cross validated.(fig 15)

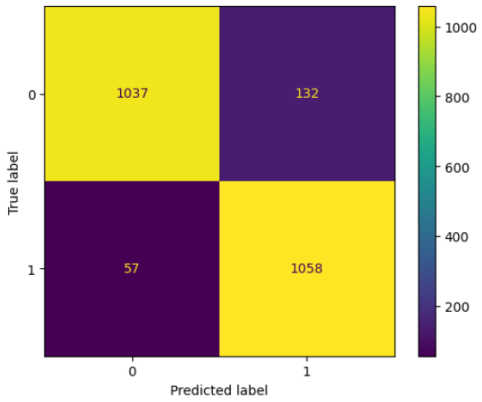


Figure 12. KNeighbors Confusion Matrix

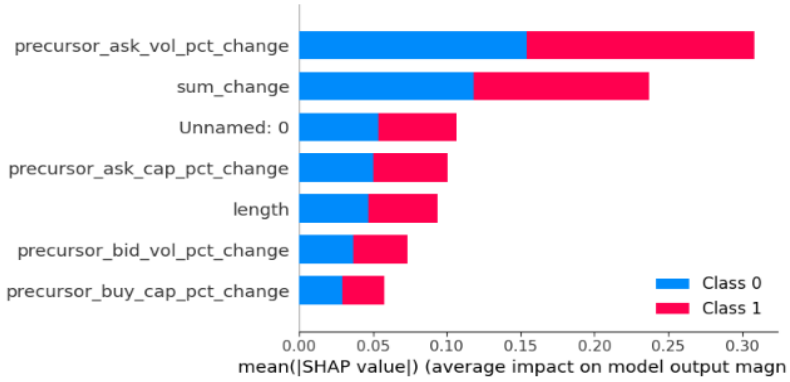


Figure 13. Balanced Random Forest SHAP, Explainability

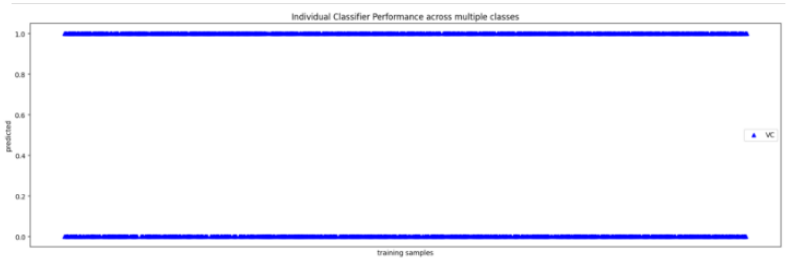


Figure 14. Binary Classifier improvements

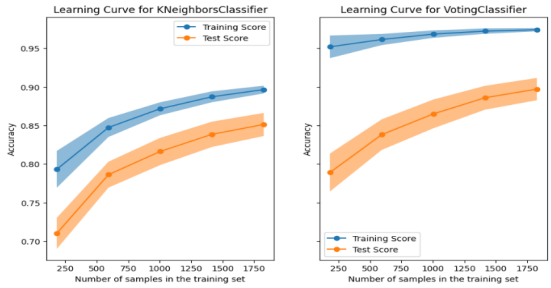


Figure 15. Learning Curves, Model 2

Model Iteration Three: Engineering Features via Unsupervised Learning

Within higher performing models, we observed that a category of data was predicting outcomes in the majority of instances. Two variables associated with selling orders, ask capitalization and ask order volumes were correlated to clustering results, as well as most likely leading to correct prediction. Within our 92% accuracy model where multiclass classifiers were used on balanced data with clustered features, the following correlation matrix was present in the underlying data. (fig 16) Our code is available here. Our most notable incremental improvement occurred when we implemented cluster-driven feature engineering. Our addition of a label derived via unsupervised learning demonstrated the ability to improve on 90%+ accuracy.

Post clustering, cluster labels were associated with the capitalization of sell orders with a .92 p-value. This strongly correlates selling orders with the surges, accompanying the precursor.

Additionally, we found that sales volume was the greatest predictor among all features, and among all classifiers in our feature analysis.

Feature Permutation Importance

One engineered feature contained more potency, in affecting the predictive ability of the model. In each classifier, 'feature 4', or the '`precursor_ask_vol_pct_change`' feature was identified as most associated with decreases in the model's accuracy.(fig 17) During our permutation importance analysis, this feature had the greatest impact on predictor performance. Permutation importance is a measure of how much an estimator declines in accuracy, due to changes in certain features.[1]

Thus we infer that selling activity in 2022-23 has the strongest relationship to the outcome variable, '`surge_target_met_pct`' where the number of price points in the precursor was met, and levels of profit were accomplished.

Parametric Sensitivity

Given the large number of parameters tested during two sets of grid searches (clustering, then classifier), we studied how parameter changes impacted accuracy. We found that in the majority of cases, changes in hyperparameters inside our voting classifier did not create statistically significant impacts on prediction accuracy. (fig 20)

In the case of the most impactful classifier within the multiclass classifier, using clustered features, we found that doubling the `max_iteration` parameter did not create ramifications on accuracy.(fig 18) We emulated a model analysis where multiple scoring models are used, on models with multiple parameters using a cross validated grid search.[2]

In order to study the general impact of hyperparameter changes across the entire voting classifier, we studied the change in accuracy as all parameters changed. We used the `cv_results_` dictionaries from each permutation of test parameters, and scikit-learn's accuracy scores in a global dataframe, to render a global average statistic. Given the large number of parameters and the multiple classifiers acting in concert, we were more interested in how parameter customization improved our model.

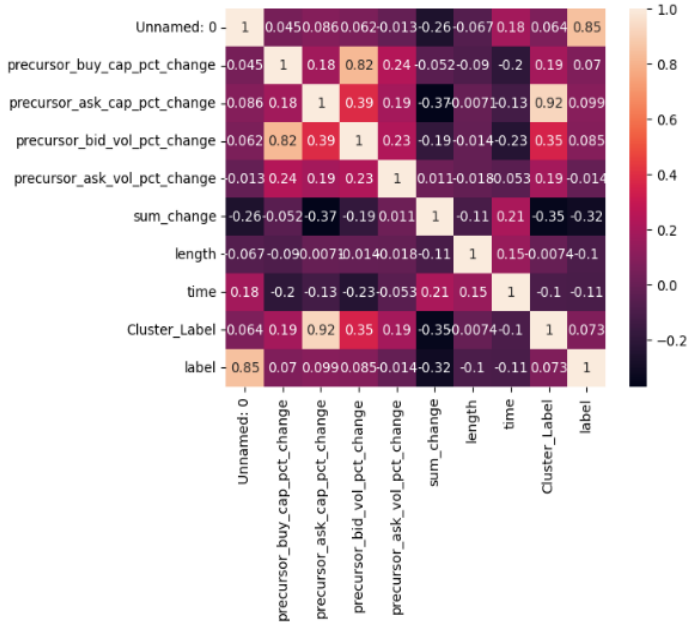


Figure 16. Correlated Features, Clustered

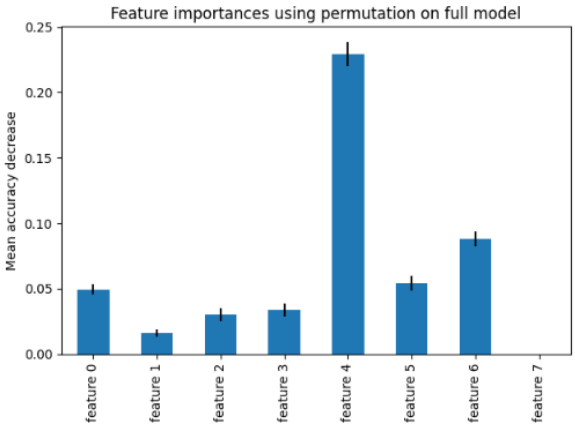


Figure 17. Feature Analysis, sell order feature

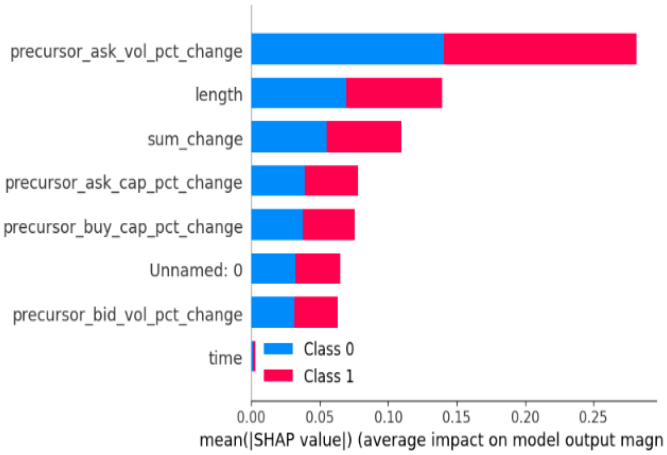


Figure 17-2. SHAP, Sell Volumes, Random Forest

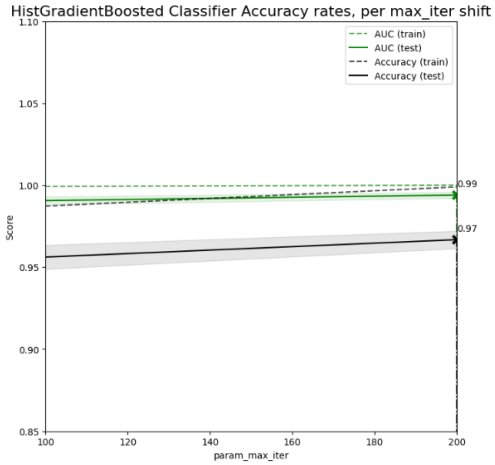


Figure 18. HGB Classifier Sensitivity

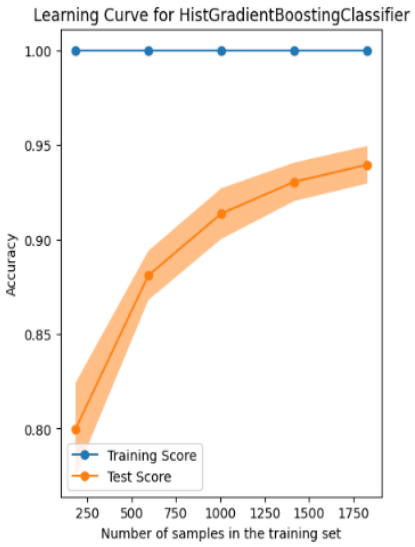


Figure 19. Learning Curve

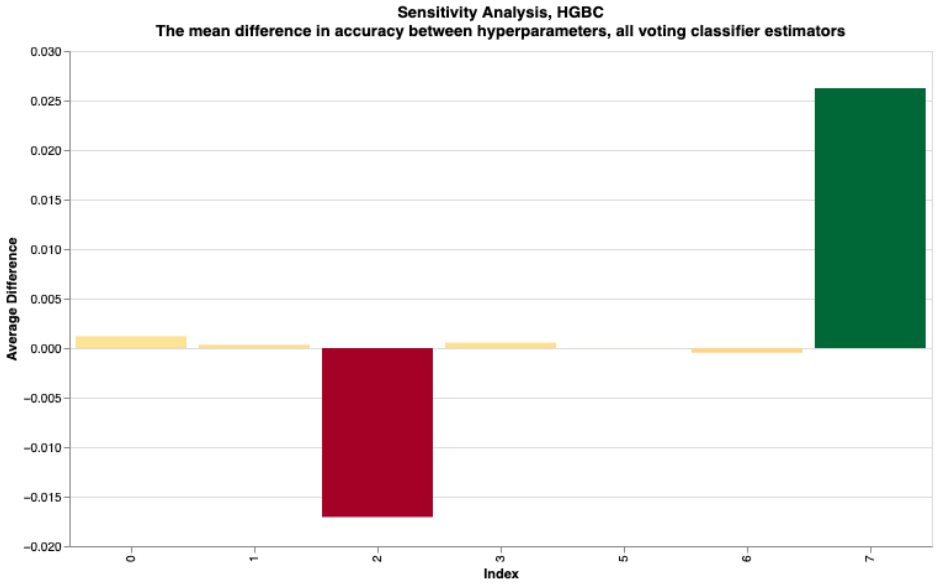


Figure 20. Sensitivity Analysis, all Methods

Statement of Work

Stefan Bund collected the API data from Coinbase, organized the study with Classifiers, and took part in the Data Mining and preprocessing phase.

Emeka Amadi organized the study of clustering and unsupervised learning, and participated in data preprocessing and Data Mining.

Both team members significantly shared responsibility for the content and quality of each other's work. Data Mining, Supervised Learning and Unsupervised Learning components were shared as a team wherever possible.

Bibliography

1. Sci kit permutation importance documentation. https://scikit-learn.org/stable/modules/permutation_importance.html#permutation-feature-importance
2. Demonstration on multi-metric grid search on multiple model parameters. SciKit Learn documentation.https://scikit-learn.org/stable/auto_examples/model_selection/plot_multi_metric_evaluation.html#sphx-glr-auto-examples-model-selection-plot-multi-metric-evaluation-py
3. Carrion, Allen 2013. Very Fast Money: High-Frequency Trading on the NASDAQ. Journal of Financial Markets 16. 680-711.
4. A python version of our Coinbase API data getter. https://github.com/stefanbund/grus-code/blob/main/archived/obaTrader/obaMain.py

5. Nearest Neighbors Classifiers. https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors
6. Ban Zheng, Eric Moulines, Frédéric Abergel. Price Jump Prediction in Limit Order Books. https://doi.org/10.48550/arXiv.1204.1381
7. Ryan Riordan , Andreas Storkenmaier , Martin Wagener , S. Sarah Zhang. Public information arrival: Price discovery and liquidity in electronic limit order markets. https://doi.org/10.1016/j.jbankfin.2012.11.008
8. Justin Sirignano, Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. https://doi.org/10.1080/14697688.2019.1622295
9. Joe Parsons. JP Morgan pulls plug on deep learning model for FX algos. Risk.net. https://www.risk.net/derivatives/7958022/jp-morgan-pulls-plug-on-deep-learning-model-for-fx-algos
10. Coinbase Exchange Websocket API Documentation. https://docs.cloud.coinbase.com/advanced-trade-api/docs/ws-overview
11. Yoga Pristyanto,Anggit Ferdita Nugraha,Akhmad Dahlan,Lucky Adhikrisna Wirasakti,Aditya Ahmad Zein,Irfan Pratama. Multiclass Imbalanced Handling using ADASYN Oversampling and Stacking Algorithm. https://doi.org/10.1109/IMCOM53663.2022.9721632
12. Amadi, Bund Github repository for this project, data only. https://github.com/stefanbund/grus-code/tree/main/lob_caps
13. Stefan Bund. Milestone 1 thesis. https://github.com/stefanbund/m1

Appendix

Cluster Visualization

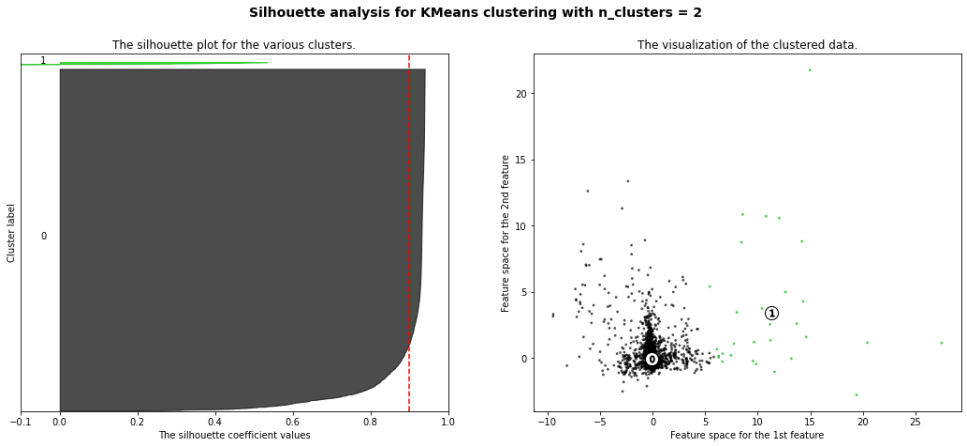


Figure 21

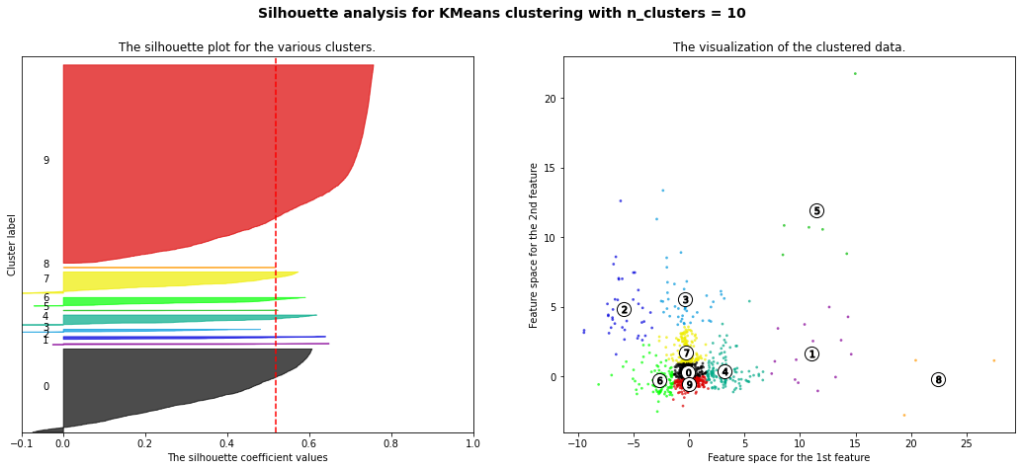


Figure 22

Business Rule Constraint: Profit per Cluster Visualization

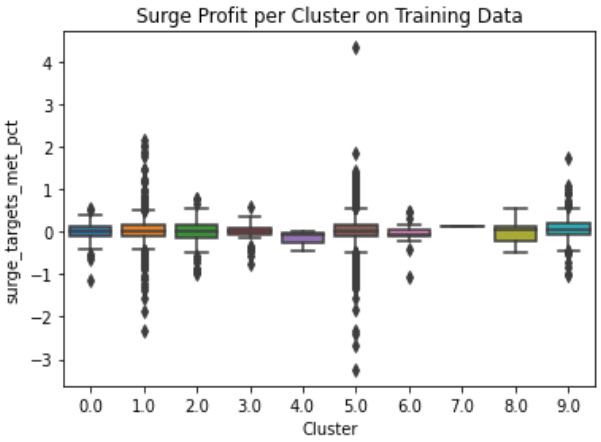


Figure 23.1(K Means)

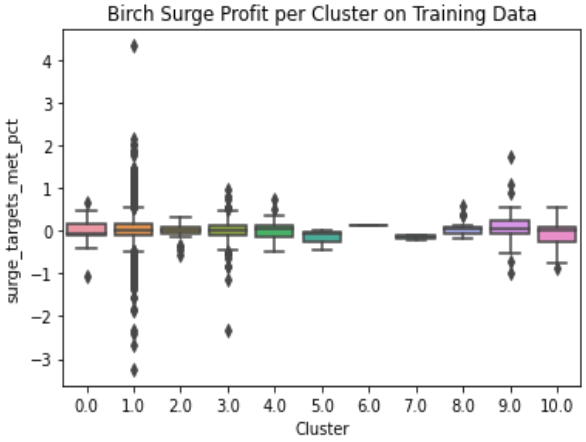


Figure 23.2(Birch)

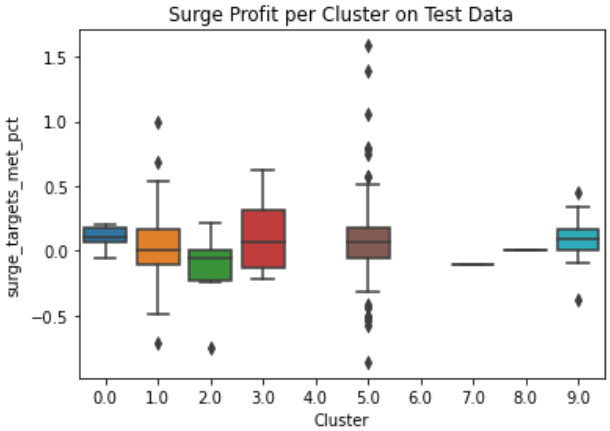


Figure 24.1(K Means)

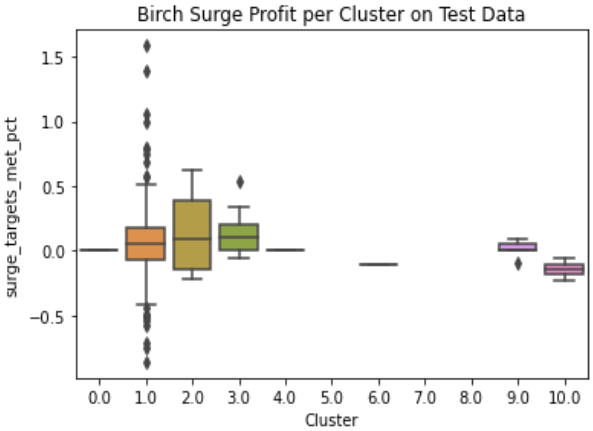


Figure 24.2(Birch)

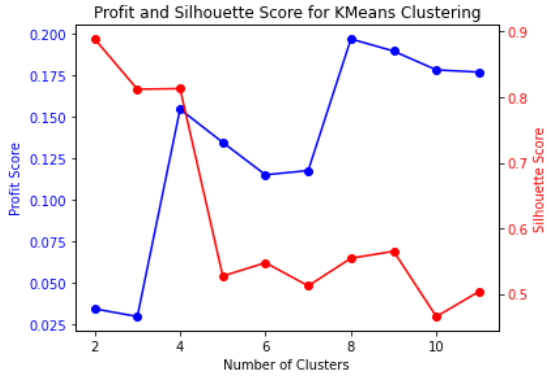


Figure 25

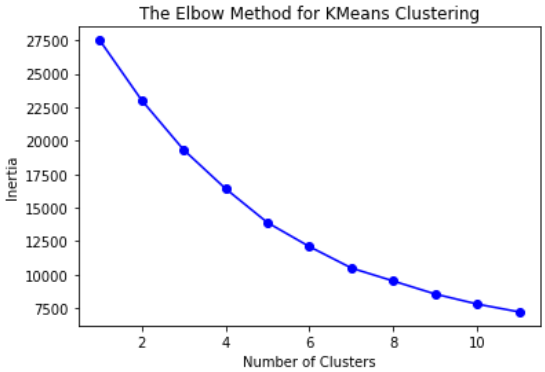


Figure 26

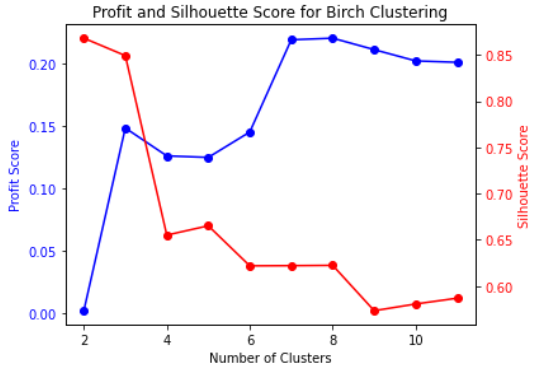


Figure 27

Cluster Profitability Method

```python
def cluster_profitability(model, X, feature_name):
    """
    Compute the profitability score of the clustering model.

    Parameters:
    - model: Fitted clustering model
    - X: DataFrame, the original data with all features
    - feature_name: str, the name of the feature to compute profitability

    Returns:
    - score: float, computed profitability score
    """
    labels = model.labels_
    df = pd.concat([X.reset_index(drop=True), pd.Series(labels, name='Cluster_Label')], axis=1)
    cluster_means = df.groupby('Cluster_Label')[feature_name].mean()

    # Custom scoring logic here.
    score = cluster_means.std()  # Standard deviation between cluster means

    return score
```

Code Block 1

```
sampled_df.loc[sampled_df['silhouette_score'].idxmax()]
```

```
clusters                    2
model              build_smote
silhouette_score      0.793449
profit_score          0.715419
Name: 1, dtype: object
```

```
sampled_df.loc[sampled_df['profit_score'].idxmax()]
```

```
clusters                   12
model              build_adasyn
silhouette_score       0.263896
profit_score           0.762506
Name: 37, dtype: object
```

Code Block 2

	algo	n_clusters	score	profit
0	KMeans	2	0.888867	0.0345768
1	KMeans	3	0.81207	0.0299211
2	KMeans	4	0.813111	0.154526
3	KMeans	5	0.527146	0.134555
4	KMeans	6	0.547412	0.115097
5	KMeans	7	0.512092	0.117663
6	KMeans	8	0.554268	0.196746
7	KMeans	9	0.56489	0.189434
8	KMeans	10	0.465292	0.178133
9	KMeans	11	0.503586	0.176815
10	KMeans	12	0.462176	0.174617

Table 1

	algo	n_clusters	score	profit
11	Birch	2	0.869585	0.00896072
12	Birch	3	0.850691	0.152463
13	Birch	4	0.758282	0.131287
14	Birch	5	0.765198	0.123945
15	Birch	6	0.739571	0.14581
16	Birch	7	0.73976	0.217987
17	Birch	8	0.562683	0.20353
18	Birch	9	0.563139	0.207827
19	Birch	10	0.563539	0.201124
20	Birch	11	0.563999	0.203139
21	Birch	12	0.527189	0.204684

Table 2

Feature Engineering, Data Mining Process

The following columns are generated during the data mining phase.

'group', a faceting of running group by operations
'time', the unix epoch when the precursor begins
's_MP', the price achieved during a surge element
'change', the percent change achieved one row to the next, as a measure of positive momentum
'type', whether an item occurred within a surge or precursor
'p_MP', price at time of a precursor item
'precursor_buy_cap_pct_change', percentage change in price times volume, buy order
'precursor_ask_cap_pct_change', percentage change in the price times volume, sell order
'precursor_bid_vol_pct_change', percentage change in volume, buy order
'precursor_ask_vol_pct_change', percentage change in volume, sell order
'length', the number of orders that fit into a precursor or length, which meet the thresholds for inclusion as precursor or surge
'sum_change', the total price change during a surge, as a sum of 'change' observed between items
'max_surge_mp', largest price witnessed during a surge episode
'min_surge_mp', minimum price witnessed during a surge episode
'max_precursor_mp', largest price witnessed during a precursor episode
'min_precursor_mp', minimum price witnessed during a precursor episode

'area', the arithmetic distance covered by the triangle of the episode, as a base times height divided by two, creates a triangular area to describe the episode
'surge_targets_met_pct', a percentile measure of how many price points inside a precursor were reached inside a subsequent surge, measures the power of the surge, a critical label value

Bin Structure, as Prelude to Clustering and Classification

```
    bins = [
        final_df['surge_targets_met_pct'].min() -1,  # Min value  # -4 to 0 divided into three equal parts
        -4/3,# Second bin edge for negative values
        -4/6,
        -4/12,
        0,
        0.125,
        0.25, 0.5, 0.75, 1,  # Four bins between 0 and 1
        2,  # One bin between 1 and 2
        final_df['surge_targets_met_pct'].max()+ 1]
    bin_labels = list(range(1, len(bins)))

    for i in range(1, len(bins)):
        print(f'Bin {bin_labels[i-1]}: {bins[i-1]:.2f} - {bins[i]:.2f}')
]
Bin 1: -4.24 - -1.33
Bin 2: -1.33 - -0.67
Bin 3: -0.67 - -0.33
Bin 4: -0.33 - 0.00
Bin 5: 0.00 - 0.12
Bin 6: 0.12 - 0.25
Bin 7: 0.25 - 0.50
Bin 8: 0.50 - 0.75
Bin 9: 0.75 - 1.00
Bin 10: 1.00 - 2.00
Bin 11: 2.00 - 5.32
```

Figure 24

Aggregated LOB Schema, pre standardization

bc	ac	tbv	tav	time	mp	minBid
3537895.45	32564599.64	417439.52	328777.66	1680895615420	17.61	17.54

Figure 25