CSE 6740 - Final Report
World of Options: Show Me the Money

Saurabh Doodhwala, Gunjan Gupta, Yuting Guo

Github link: `https://github.gatech.edu/sdoodhwala3/CDA-Options`

May 2021

Abstract

Due to frequent number of financial crisis happening in the last three decades, there has been a strong motivation to study and research financial models to maximize the returns and minimize the risks for the future. Abundant amount of work has been done in this domain and numerous mathematical and Machine Learning models were created to predict the future. Machine learning has long been leveraged to predict stock market changes. Common attempts in use include trial & error techniques, Black-Scholes model and various forms of non-parametric and classification models. As investing becomes more available to the general public through apps such as Robinhood, more investors are participating in stocks and options trading. The focus of this paper is to explore how predictors such as stock price, expiration date of options, premium price, volume and strike prices impact profits.

1 Introduction & Problem Statement

The main motivation of this project is to develop an informed and robust machine learning model which can determine whether a given 'call' option will be profitable or not on the basis of various features of option and stock. This can help investors, traders and speculators to maximize their returns.

There have been abundant research done in the direction of predicting stock prices based on historical data available online [1], [2]. These directions of research vary in many ways, some focus on predicting long term stock price changes for a limited amount of stocks (10 or fewer) while others focus on short term price predictions for the market in general (50 stocks or more) [3]. In the Options direction, researchers have tried to come up with predicting Option price using technical indicators, stock prices by focusing on a few specific stocks like NVIDIA and Intel [4]. However, very few research have considered whether the option bet they are placing right now, will be profitable or not by the time the option expires.

In this work, the attempt is to determine if a given 'call' option will be profitable or not by the time it expires, by looking at various features for that option and stock like the time when option was placed, the time it expires, premium price of the option, news/attention around the stock in the time period, daily volume of stock, etc. The problem is novel in that it takes into consideration the effect of news around the stock for option analysis, which almost no paper has focuses on. Almost every research considers the effect of news on options indirectly through stock prices but the idea is came up with a model that explains this effect on options profitability directly.

The main problem statement is to determine 'call' option profitability through the use of standard and novel(or derived) predictors to develop a robust machine learning model that make informed decisions for our investments.

2 Data & Initial Exploratory Analysis

2.1 Data Sources

The dataset used contains options and stock information from a four year time frame (November 2017 to 2021) from various sources. The data analyzed is sector focused on ETF and Technology. The variables details are shown below: [5] [6]

Predictor	Description	Source	Used?
OptionType	Type of option: calls or puts	Kaggle	Yes: only calls
Time	Time at which the contract was made	Kaggle	No
Sym	Ticker symbol on the contract	Kaggle	Yes
Expiration	Date & Day of week that the contract expires	Kaggle	No
ExpirationDate	Date that the contract expires	Kaggle	Yes
StockPrice	Price of stock at time of contract	Kaggle	Yes
Date	Date of contract	Kaggle	Yes
Sector	Sector that the symbol belongs to	Kaggle	Yes: ETF & Tech
Spent	Dollar amount spent total	Kaggle	Yes
Strike	Set price a contract can be bought or sold	Kaggle	Yes
ChainLocation	Out the money; In the money; At the money	Kaggle	No
Open	Price of symbol when the stock market opened	Yahoo Finance	No
High	Highest price of symbol for a given day	Yahoo Finance	No
Low	Lowest price of symbol for a given day	Yahoo Finance	No
Close	Price of symbol when the stock market closed	Yahoo Finance	Yes
Adj Close	Adjusted closing price of symbol	Yahoo Finance	No
Volume	# of shares that changed hands for a given day	Yahoo Finance	Yes
Spent/contract	Dollar amount spent per contract	Derived	Yes
Spent/share	Dollar amount spent per share	Derived	Yes
break_even	Dollar amount to break even	Derived	Yes
Contract period in hrs	Amount of time before contract expires in hours	Derived	Yes
Contract period in days	Amount of time before contract expires in days	Derived	No
Attention	Search trend via Brownian process	Derived	Yes
Profit	Where or not profit was made from the trade	Derived	Yes

Table 1: Dataset

2.1.1 Attention model

Social trends (i.e. attention) play a significant role in options/stock pricing and can have a big impact on the accuracy of the models [7]. There are many possible ways of describing this attention parameter, such as number of Google searches, Wikipedia requests, or more traditional factors such as volume of transactions. For this dataset, the attention parameter is represented as a function of the number of Google searches for a particular option/stock (available at trends.google.com).

The attention metric is determined based on a specific stock/option symbol or collection of symbols with a date range. The output attention value is a number between 0 and 100. A high value during any specific time period indicates that the symbol was trending or was a hot search topic. This work is based on the following github documentation [8]

2.1.2 Variable Selection

To further analyze the parameters from Table 1, correlation graphs and boxplots were created to spot relationships and anomalies with the variables. The graphs in Figures 6, 7, 8 indicated that many variables have

high correlation. Thus, many highly correlated predictors were removed since they add no new information to the dataset (marked as No Used in Table 1).

2.2 Balanced Dataset Creation

Due to the nature of options trading, the dataset obtained is severely imbalanced with far more losses than profits. Therefore, to ensure the classification models are not biased in favor of the majority class, SMOTE, ADASYN, and NearMiss was used to create a re-balanced datasets. After building the classification models, SMOTE provided the best performance for the data and was used for model fitting and predictions.

2.2.1 Synthetic Minority Over-Sampling Technique (SMOTE)

SMOTE was selected to help generate synthetic samples for the minority class, profit on the options. This technique is effective in low-dimensional settings in reducing class imbalance. Furthermore, it has shown to work particularly well for KNN classifiers when variable selection is performed beforehand [9]. Since the options data is low-dimensional, SMOTE should enable better learning for the models.

SMOTE samples are linear combinations of two similar samples from the minority class with the below definition:

$$s = x + u(x^R - x) \tag{1}$$

where $0 \leq u \leq 1$, x and x^R are samples from the minority class

2.2.2 Adaptive Synthetic Over-Sampling (ADASYN)

In addition to SMOTE, ADASYN is another over-sampling technique used to balance out the dataset. It leverages weighed distribution from different minority class based on the learning difficulty level to determine the number of samples to be generated. As a result, there is a reduction in bias and adjustments to classification decision margins for difficult cases. In research scenarios, ADASYN has shown higher performance than SMOTE [10].

2.2.3 NearMiss Under-Sampling Technique

Under-sampling is the de facto method for balancing imbalanced datasets. The NearMiss technique under samples the majority class by removing option contract losses, thus, balancing out the data. In literature, many scenarios have shown success in using under-sampling for large-scale, imbalanced data [11]. However, due to the high imbalance in the options data, under-sampling removed too much of the majority class which caused the models to learn less effectively.

3 Methodology

3.1 Framework

The following summarized the framework followed for this project:

1. Split the data into 80/20 ratio of train versus test data.
2. Augment the training data with SMOTE, ADASYN and Near miss to deal with imbalances in the data.
3. Preliminary examination shows that SMOTE performs better with options data. Therefore, SMOTE is leveraged to balance the dataset.
4. Accuracy score is leveraged to tune hyper parameters of the ML models.
5. Appropriate metrics (reference section 3.8) are used to assess the prediction performance of the models

3.2 K Nearest Neighbor

KNN was a baseline model used to compare the performance of other ML models. The parameter K = 2 was the most optimal, giving the highest performance results. For the ETF sector, precision and recall metrics are 99% for the non-profit class and 87% and 90% for the profit class. For the Technology sector, non-profit class performance metrics remained high, but the profit class performance metrics dropped sharply to 30% and 35% precision/recall. This indicates, that KNN does not fit the profit class well, even after leveraging SMOTE for a more balanced dataset.

3.3 Logistic Regression

The L2 regularization was used in the logistic regression model. In the ETF sector, the logistic classifier resulted in lowest overall accuracy; 50%. In the Technology sector, the accuracy is 73%. Overall, the model has low precision for the profit class in both the sectors, hence might not be a good fit for the data.

3.4 Support Vector Machine

With SVM, models were created with the Linear and RBF kernels. Similar to Logistic Regression, Linear SVM was unable to correctly classify the majority of the profit class, which indicates that that the complexity of the data does not fit well with linear models overall. The RBF SVM performed slightly better in classifying the minority class. However, as a whole, SVM falls short in comparison to the baseline KNN model, rendering it a poor fit for options data.

3.5 Decision Trees

For Decision Trees the tuned hyper parameter is the maximum depth. The optimal maximum depth is 34 in the ETF sector and 26 in the technology sector. Gini impurity is used as the stopping criterion. The Decision Tree model gives good performance on the minority class for the ETF sector data with precision of 85% but doesn't perform as well on the minority class of technology sector with only 31% precision. The outcome of complex systems such as the stock market depends on not just on a single criterion, but on multiple criteria over multiple features, therefore decision trees performed better than the linear classifiers used above.

3.6 Random Forest

The relative success with Decision Trees led to the use of ensemble methods. In the Random Forest model, two hyper parameters - the maximum depth and number of trees in the forest were tuned using GridSearchCV. Similar to Decision Trees, Gini impurity is used as the stopping criterion. Since Random Forest consist of an ensemble of relatively uncorrelated decision tree models, it gives better performance than the individual an Decision Tree model. In particular, the precision of the minority class in the technology data improved from 31% with Decision Trees to 58% with Random Forest.

3.7 Neural Networks

Neural Network was the toughest model to train with the selected dataset due to the large number of tuning parameters. Keras, Tensorflow and Sklearn packages were leveraged to build the models. In Keras, each layer had to be manually added and checked for performance, making the models difficult to tune. Thus, the MLPClassifier in Sklearn was a better fit to use [12]. With GridSearchCV, the neural network was optimized with respect to hidden layer sizes, regularization parameter, activation function, batch size.

For ETF sector, the accuracy is 97% on test data along with a precision of 72% for the minority profit class. For the Technology sector, an accuracy of 66% was obtained and the performance on minority class was off with a very low precision value. This pertains to the fact that neural networks still require more hyper-parameter tuning which is beyond the scope of this work and can be worked on in future.

3.8 Metrics

When running classification models, it is important to decide on the evaluation metric. The key metrics leveraged are [12]:

1. Accuracy

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+TN+FN}} \tag{2}$$

		Actual	
		Profit {1}	Not profit {0}
Predicted	Profit {1}	TP	FP
	Not profit {0}	FN	TN

Figure 1: Confusion matrix: Accuracy

2. Precision answers the question - What proportion of positive (or negative) identifications was actually correct?

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{3}$$

		Actual	
		Profit {1}	Not profit {0}
Predicted	Profit {1}	TP	FP
	Not profit {0}	FN	TN

Figure 2: Confusion matrix: Precision

3. Recall answers the question - What proportion of actual positives (or negatives) was identified correctly?

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{4}$$

		Actual	
		Profit {1}	Not profit {0}
Predicted	Profit {1}	TP	FP
	Not profit {0}	FN	TN

Figure 3: Confusion matrix: Recall

Precision and recall are helpful metrics in this problem due to severe class imbalance, where poor classification performance in minority class can still be masked with high accuracy scores.

4. F_1 score

$$F_1 = 2\frac{\text{Precision} * \text{recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

F1 score can be useful when seeking balance between precision and recall.

4 Evaluation & Final Results

The analysis was conducted on two main stock sectors: Technology and ETF. The following tables show important metrics for the machine learning models applied on test data for both sectors.

"ETF" Sector								
		Metrics	Class "Non-Profit" (0)			Class "Profit" (1)		
Models		Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
1	k-Nearest Neighbor	0.98	0.99	0.99	0.99	0.85	0.90	0.87
2	Support Vector Machine	0.95	0.99	0.96	0.97	0.56	0.81	0.66
3	Neural Network	0.95	0.99	0.95	0.97	0.55	0.81	0.66
4	Logistic Regression	0.61	0.97	0.60	0.74	0.12	0.77	0.20
5	Decision Tree	0.98	0.99	0.99	0.99	0.85	0.90	0.87
6	Random Forest	0.99	0.99	1.00	1.00	0.97	0.90	0.94

Figure 4: Test Data Results for ETF Sector

Technology Sector								
		Metrics	Class "Non-Profit" (0)			Class "Profit" (1)		
Models		Accuracy	Precision	Recall	F1-score	Precision	Recall	F1-score
1	k-Nearest Neighbor	0.94	0.99	0.95	0.97	0.14	0.39	0.21
2	Support Vector Machine	0.78	0.99	0.79	0.88	0.05	0.52	0.09
3	Neural Network	0.60	0.99	0.59	0.74	0.04	0.78	0.07
4	Logistic Regression	0.72	1.00	0.72	0.83	0.06	0.87	0.11
5	Decision Tree	0.97	0.99	0.98	0.98	0.31	0.48	0.37
6	Random Forest	0.98	0.99	0.99	0.99	0.58	0.48	0.52

Figure 5: Test Data Results for Technology Sector

The cost of predicting profit when there is an actual loss for an option is dangerous for small investors, hence, the ideal model should keep the number of false positives as low as possible. Therefore, we want to choose a model with high precision for Profit class, as low precision implies too many false positives. The F1 score metric can be used to gauge the overall performance of the models considering both precision and recall.

For the ETF sector, all models have high accuracy except for Logistic Classifier. However, accuracy alone cannot be used to determine the performance of the models due to imbalances in the data. Additional considerations include the True Positive Rate (Recall) and Precision for each class. Random Forest, Decision Tree, KNN, and Neural Networks have reasonable amount of precision for Profit Class. Random forest gives the highest precision for profit class in this sector.

Similarly, for Technology sector stocks, the accuracy is in good shape for KNN, Random Forest and Decision Trees. Like before, the key factor will be precision for the Profit Class when deciding on the final model. Precision for Profit Class is the best for Random Forest model similar to that of the ETF sector. One of the main reason for this is that test data contains very few positive (Profit) points (2.1 % of total test set for Technology Sector and 6.4% of total test set for ETF Sector) and tuning models to classify them properly was a challenging task.

In conclusion, linear models such as basic logistic regression and linear SVM are poor fits for the options data. Since the data set is complex, better performance metrics are observed with advanced models like Random Forest and Neural Network. Surprisingly, Decision Trees also fit the data relatively well. This could be due to the fact that tree based methods are more flexible in identifying hidden patterns and relationships. Overall, Random Forest has the best F1 scores; 0.99 for nonprofit classification, 0.52 and 0.94 for Technology and ETF profit classification.

5 Scope of Future Work:

The objective of this study was to predict whether a stock can be profitable or not for a retail options trader. However, this data pool may consists of bets which were made for trading as well as hedging purposes. Further data tuning, by filtering out the bets made for hedging and then using the remaining data for options profit prediction might help to improve the precision of ML models.

The fine-tuning of neural network also presents a scope for improvement, as they have a potential to solve options problem with high precision. We were able to tune neural networks nicely such that we achieved precision of 0.55 for the profit class in 'ETF' sector. However, in Technology sector, more fine tuning will be needed using some advanced techniques in Pytorch, which is beyond the scope of current work.

This work has been successful in developing robust Machine Learning models for ETF and Technology sectors. Future studies could try to focus on other sectors like Finance, Energy by developing models in the similar fashion as shown in current work.

6 Logistics

The code for the project was written using Python 3.7 in Jupyter Notebook.

GitHub details: The GitHub repo containing the source code for the project has been shared with the instructor and TAs for the class. Link: `https://github.gatech.edu/sdoodhwala3/CDA-Options`

The below table, summarizes the logistics and division of work between the members:

Area	Task/Activity	Member
Report	Proposal	All
Report	Final Report	All
Discovery	Literature review & background	All
Data Exploration	Exploratory Analysis	Gunjan, Yuting
Data Exploration	Balance Dataset	Yuting
Data Exploration	Attention Metric	Saurabh
Methodology & Modeling	Logistic Regression, Decision Tree & Random Forest	Gunjan, Saurabh
Methodology & Modeling	KNN & SVM	Yuting
Methodology & Modeling	Neural Network	Saurabh
Evaluation	Performance Metrics	Saurabh, Gunjan

Table 2: Division of Work

References

[1] Shen et al. "Stock market forecasting using machine learning algorithms". In: *Department of Electrical Engineering, Stanford University, Stanford, CA* (2012), pp. 1–5.

[2] Michael David Rechenthin. "Machine-learning classification techniques for the analysis and prediction of high frequency stock direction". In: *The University of Iowa* (2014).

[3] Nikola Milosevic. "Equity forecast: Predicting long term stock price movement using machine learning". In: *arXivpreprint arXiv:1603.00751* (2016).

[4] Abraham Adam. "Stock Option Price Prediction". In: *Stanford University* (2012).

[5] Kaggle Options Market Trades Dataset. *https://www.kaggle.com/bendgame/options-market-trades*. 2019.

[6] *https://finance.yahoo.com/*.

[7] Sheldon Natenberg. *Option Volatility and Pricing: Advanced Trading Strategies and Techniques*. McGraw-Hill, 2014.

[8] Qingzong Tseng. *Reconstruct Google Trends Daily Data for Extended Period*. 2020.

[9] Rok Blagus and Lara Lusa. "SMOTE for high-dimensional class-imbalanced data". In: *BMC Bioinformatics* 14.106 (2013).

[10] Hazel A. Gameng, Bobby D. Gerardo, and Ruji P. Medina. "A Modified Adaptive Synthetic SMOTE Approach in Graduation Success Rate Classification". In: *International Journal of Advanced Trends in Computer Science and Engineering* 8.6 (2019).

[11] Lei Bao et al. "Boosted Near-miss Under-sampling on SVM ensembles for concept detection in large-scale imbalanced datasets". In: *Neurocomputing* 172 (2016).

[12] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

Appendix

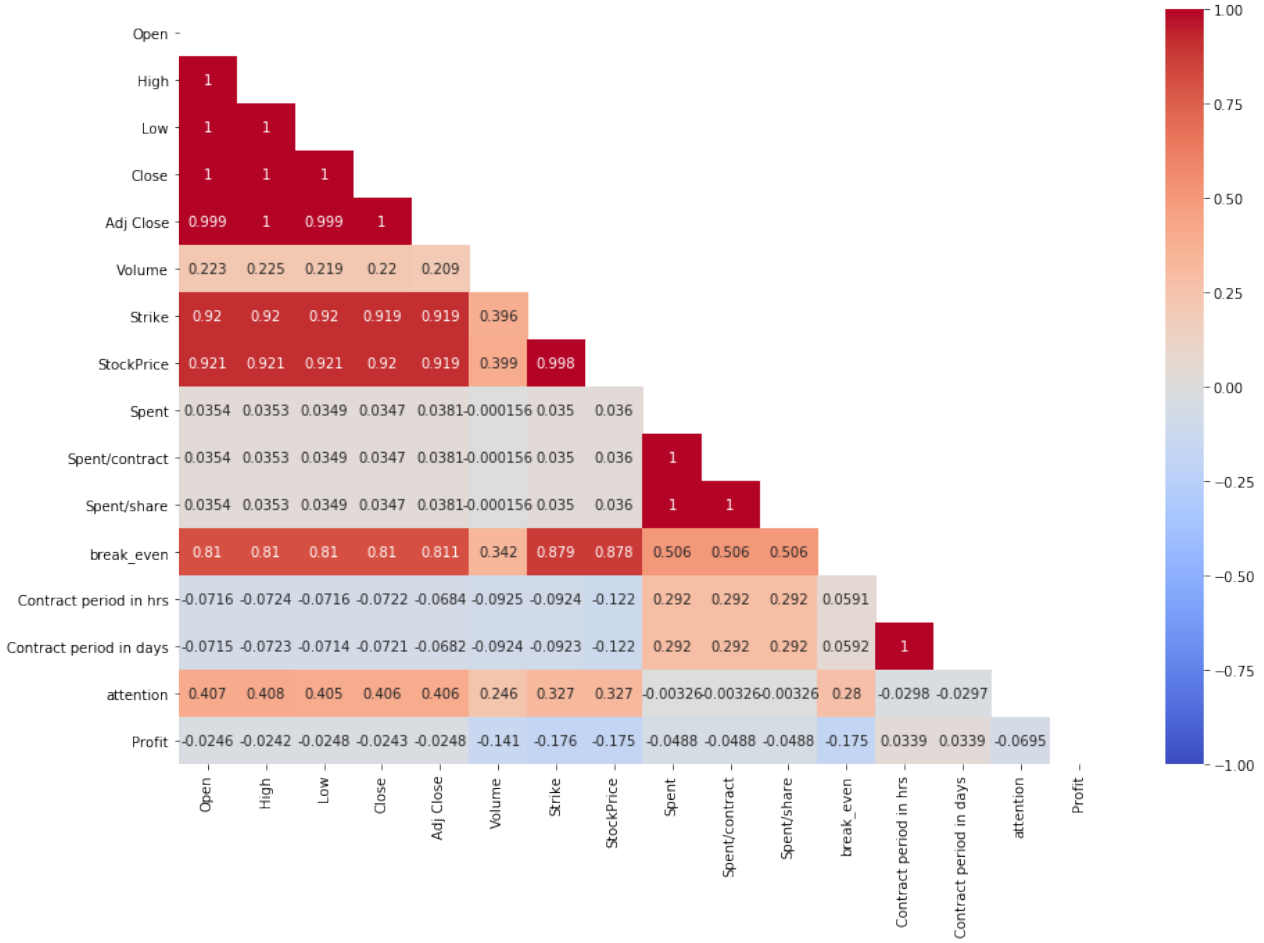

Figure 6: Correlation Graph

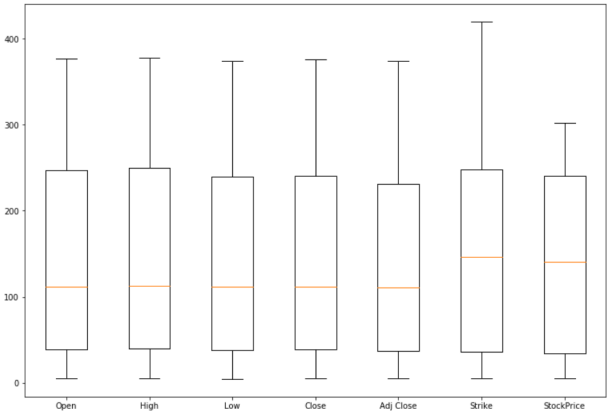

Figure 7: Box Plots

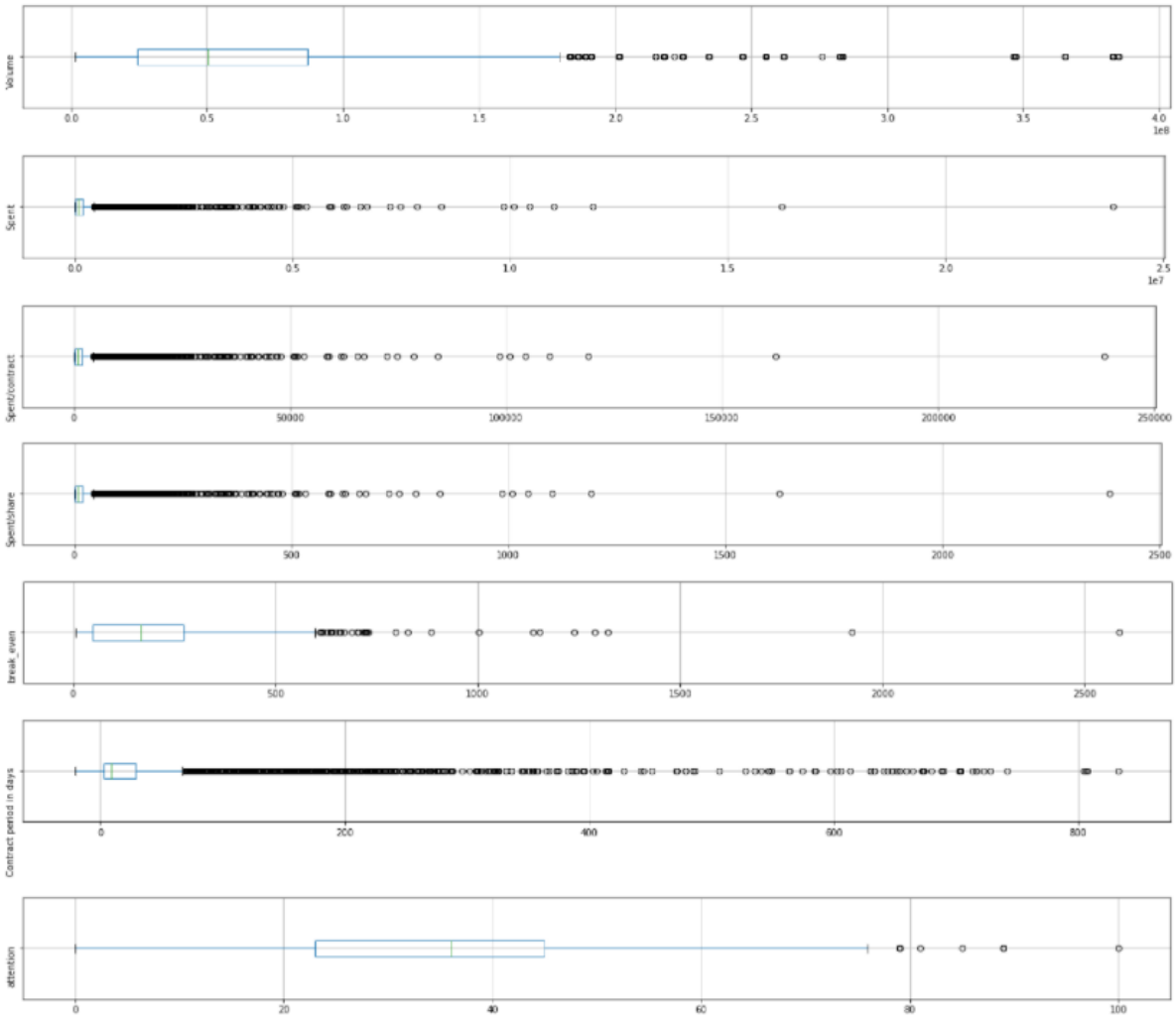

Figure 8: Box Plots