UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Machine Learning models for votes aggregation in crowdsourced market forecasting

Author:
Khrystyna KUBATSKA

Supervisor:
Andriy KUSYY

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2022

Declaration of Authorship

I, Khrystyna KUBATSKA, declare that this thesis titled, "Machine Learning models for votes aggregation in crowdsourced market forecasting" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"A journey of a thousand miles begins with a single step."

Lao Tzu

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Machine Learning models for votes aggregation in crowdsourced market forecasting

by Khrystyna KUBATSKA

Abstract

In the project, we try to solve the votes aggregation problem for the forecasted crowdsourcing market platform. We aggregate initial data in a specific way on the auction level and use machine learning models to derive the asset price from the votes.

Acknowledgements

First of all, thanks to my dear family, who always believes in me and supports me. Many thanks to my supervisor, Andriy Kusyy, for his help, fresh ideas, and valuable feedback. Thanks to the Solex.ai team for the interesting challenge and essential insights. Finally, thanks to the Faculty of Applied Sciences and Ukrainian Catholic University for making the impossible possible.

Contents

List of Figures

List of Tables

List of Abbreviations

ML	**M**achine **L**earning
MIL	**M**ultiple **I**nstance **R**earning
MIR	**M**ultiple **I**nstance **R**egression
MSE	**M**ean **S**quared **E**rror
MAPE	**M**ean **A**bsolute **P**ercentage **E**rror
RF	**R**andom **F**orest
BRT	**B**oosted **R**egression **T**rees
ERT	**E**xtremely **R**andomized **T**rees

To my Family…

Chapter 1

Introduction

Crowdsourcing is an effective method to gather information from a large group of people who submit their data via the Internet. It is widely spread in different areas, from science to politics and journalism.

Its term was coined in 2005 by Jeff Howe and Mark Robinson and was first published in a blog post in June 2006, although crowdsourcing as a phenomenon occurred much earlier (Royal Museums Greenwich, 2021). Howe defined it as "the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call. This can take the form of peer-production (when the job is performed collaboratively) but is also often undertaken by sole individuals." (Jeff Howe, 2006)

Crowdsourcing's main benefits are improved expenses, speed, quality, flexibility, scalability, and diversity. Concerning disadvantages, crowdsourcing challenges include task design, motivation problems and incentive systems, task routing and coordination, quality control of work results, and task accumulation (Buettner, 2015). Nowadays, organizations use crowdsourced forecasting to guide better principal strategic and operational decisions they are facing. Thus, crowdsourcing data can improve management decisions (Horn Christian, 2018).

1.1 Background and Goal

In the project, we try to solve a votes aggregation task with crowdsourced financial data for Solex.ai[1]. It is a voting platform designed to integrate real-time financial market forecasts from user votes.

The design of obtaining the crowdsourced data we use is the following:

- Anyone registered on the platform can vote to predict prices/indexes for financial assets at a given period.
- There are four types of assets on auctions: S&P500, Tesla inc, Bitcoin and WTI OIL.

[1]Solex.ai

- A vote is a predicted price/index of a specific asset for a specific time (a day) by a user.
- All users have a limited number of votes for each asset for a day.
- Users can submit their votes to forecast price/index values 2-31 days in advance, including public holidays and weekends.
- One auction refers to a set of voices submitted to predict an asset for a given time.
- Every auction is opened for 31 days and closes one day before a historical asset value (target) is known.
- When an auction closes, users who voted for it get scores based on several criteria, and one of them is how precise to target their votes is.
- Users with the highest scores for each auction get winning payments.

The purpose of the project is to develop a Machine Learning model to be able to use votes to predict asset values by votes submitted 7-20 days in advance.

As a basic model for initial comparison, we use an algorithm that accumulates last 5 votes submitted for period 7-20 days in advance and considers their mean as a predicted value. MSE and MAPE metrics are used to compare and evaluate results.

1.2 Objectives

In order to achieve the thesis goal, we represent the following objectives:

1. Preparing data and analyzing it to understand the limitations and relationships, selecting features, and splitting data to train and test parts.
2. Training models, choosing parameters for training to predict better than the basic algorithm perform.
3. Selecting a model with the best predictions by analyzing results and extracting insights.

1.3 Paper Structure

Chapter 2 describes related works and analyzes approaches related to our task. Chapter 3 represents the dataset description. Chapter 4 provides the methodology we use to achieve the project goals. Chapter 5 describes the comparison of results for different models. Chapter 6 concludes the obtained results.

Chapter 2

Related works

2.1 General Overview

For the project, we reviewed various papers that cover crowdsourced predictions and aggregation topics. Most of the papers we studied focus on defining/labeling noisy and unreliable data gathered by crowdsourced methods. For instance, in (Edoardo Manino, 2019), the authors developed a new algorithm Streaming Bayesian Inference for Crowdsourcing, to solve the problem of binary classifications from crowdsourced data, or in (Natalie Parde, 2017), scientists proposed a regression-based aggregation approach. However, in the works mentioned above, scientists tried to solve classification, not regression problems.

Some of the papers whose aim was to solve aggregation regression tasks based on crowdsourced data are the follows. In (J. M. Sadler, 2018), the authors applied Poisson and Random Forest regressions to predict flood severity. Their research shows that the Random Forest model performed better than Poisson regression for such tasks. However, the method has a significant limitation - since the RF model predictions are the average of each regression tree's prediction, the RF predictions cannot exceed the range of training values. In (Giuseppe Nebbione, 2018), scientists reformulated the crowd forecasting problem as a neural ranking problem - a deep Neural Network ranks forecasters based on their expected relative accuracy for a given forecasting question. Then the resulting ranking is used to create a weighted aggregation of crowd forecasts for each unique forecasting question about geopolitical events.

2.2 Multiple Instance Regression Problems Approaches

In the section, we will review approaches that are not directly related to crowdsourcing but include technical approaches to solving the task (Multiple Instance Regression problem) we have on the project.

In traditional Single Instance Learning (SIL), an object is defined by one instance, and every training instance is assigned to one categorical or real-valued label. In comparison, in Multiple Instance Learning (MIL), an object is characterized by a collection of instances that is called a bag. Labels are available at the bag level but not in the individual instance. The goal is to train a model that can predict a label of a new bag, having its instances as inputs (Zhuang Wang, 2011). Multiple Instance Regression (MIR) is a subtype of MIL, and its primary specialty is that its data have real-value labels. One of the biggest MIR challenges is that the unknown number of relevant instances can vary from one bag to another.

Existing MIR algorithms can be conditionally divided into three types based on the bag's instance relevance (Trabelsi, 2018). The first one assumes that all instances are relevant to defining bag labels. The second and last ones assume only one primary instance, and a set with primary instances is responsible for the bag's label.

In the Aggregated-MIR (Trabelsi, 2018) (Zhuang Wang, 2011) algorithm, each bag is represented as one meta-instance, calculated as a simple average of all instances. Then a regression model is trained using the meta-instances as input. To predict labels of new bags, meta-instances are used as input to the learned model. Figure 2.1 represents an example of using the approach.

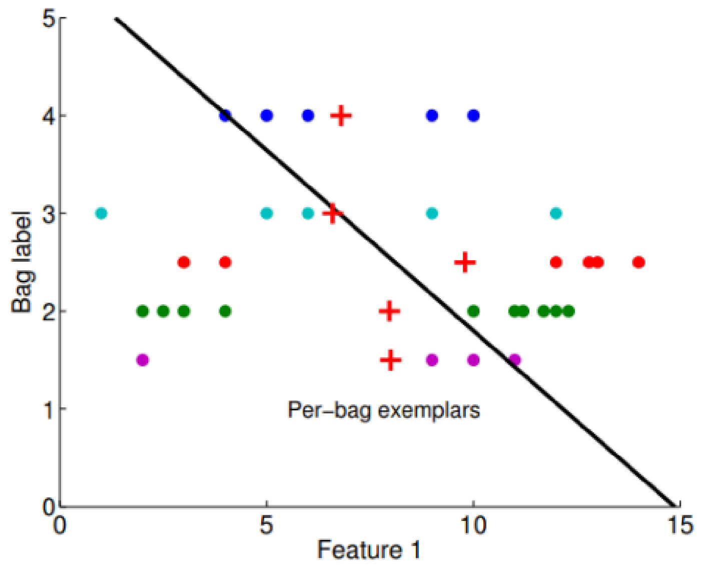

FIGURE 2.1: Example of Aggregated-MIR. The data consists of 5 bags with one-dimensional items. "+" correspond to the average of each bag. Black line is a learned regression model (Kiri L. Wagstaff, 2008)

The Aggregated-MIR approach gives reliable results if all instances in bags are "true instances". Also, if the variance of instances is not small, the approach shows suboptimal results.

The Instance-MIR (Trabelsi, 2018) (Zhuang Wang, 2011) approach treats each instance as a separate label. Firstly, all instances are used as input for regression model training. Secondly, a label for every instance should be predicted using the previously trained model. Thirdly, these labels of all instances should be aggregated (mean or median) to predict the bag's label. The approach assumes that every instance in each bag has the same link to the bag's label and suffers when bags contain many noisy instances. Figure 2.2 illustrates an example of using Instance-MIR.

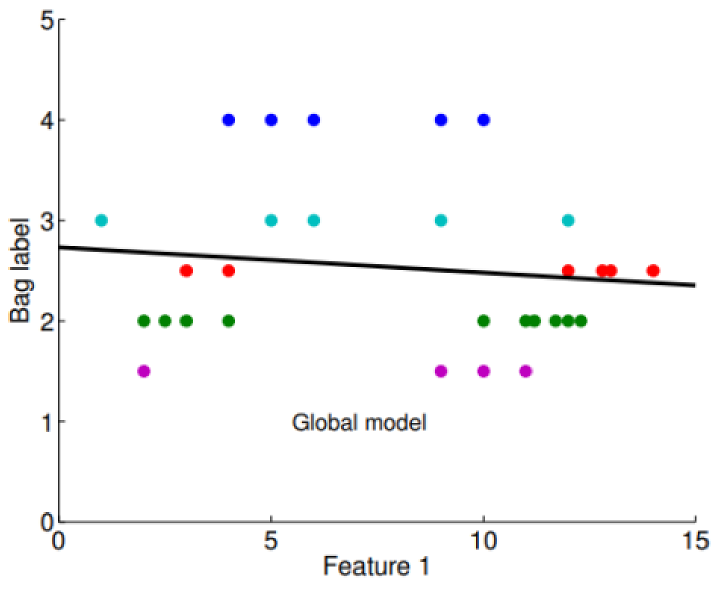

FIGURE 2.2: Example of Instance-MIR. The data includes 5 bags with one-dimensional items. Black line represents a learned regression model (Kiri L. Wagstaff, 2008)

Prime-MIR (Zhuang Wang, 2011) is an iterative algorithm that maintains bags' structures. It is based on the assumption that each bag has a prime instance ("true instance") and other instances are noisy. Its outline is to define the prime instance of each bag and train a linear regression model on it. The algorithm selects the instance from each bag with the lowest predicting error (prime candidates). Then a new predictor is trained using these prime candidates, and the algorithm iterates unless the prediction error over prime candidates decreases. However, it does not predict a label of a new bug. Usually predicted label is counted as the mean or median of its instances output.

The Pruning-MIR algorithm (Zhuang Wang, 2011) is a combination of PIR and Instance-MIR. It uses a small part of all instances for training regression models. It starts with Instance MIR and, on each iteration, removes a small part with most noise exemplars. Then it trains models on the remaining instances. The algorithm iterates as long as the prediction accuracy on the training data keeps improving. Similar to Prime-MIR, to predict a new bag's label, it uses the mean or median of predicted instances output.

MI-ClusterRegress (Du, 2016) (Kiri L. Wagstaff, 2008) algorithm maps instances onto cluster labels. Its primary assumption is that the instances from a bag are with noise from a set of underlying clusters, and one of the clusters is "suitable" to the bag-level labels. A local regression model is set up for each cluster after obtaining k clusters for each bag with Expectation-Maximization based on a clustering method. Then the best-fit model is selected and used to predict labels for test bags. The main drawback of the approach is that clustering is performed unsupervised without consulting the bag labels. This can lead to uninformative clusters and poor prediction accuracy, even when the primary assumption is valid.

Alternating Projections-Salience Algorithm (K. Wagstaff, 2007). It gives each instance in the bag the possibility to impact the bag's label by a weighted amount. The algorithm calculates the best salience values assigned to instances under a fixed regression model. Then, given the fixed salience of instances, the regression model is updated to minimize an objective function. The Alternating Projections-Salience algorithm is based on optimizing the contribution of each instance per bag to the bag's label. However, the approach does not provide a tool for testing where the bag labels and item's relevance are unknown (Trabelsi, 2018).

To sum up, the MIR problem has two main challenges that significantly impact selecting an approach to solve it. The first one is a non-acquaintance of the number of "true instances" within each unlabeled bag, and the different numbers of instances in bags is the second one. In the following chapters, we will make data analysis and explain choosing methodology for models. The idea proposed in this paper aims to overcome those challenges by using aggregations and machine learning models.

Chapter 3

Data

3.1 General DataFrame

In the project, we use the data gathered on the platform described in Chapter 1.2. The data corresponds from the 2020/09/02 to the 2022/05/29 period.

Original data was located in separate CSV files. The first steps were to extract possible features and match them in the one general data frame, which then we used for aggregation and model training and testing.

In the Table 3.1 there is description of the data used for further analysis.

Column name	Metrics	Description
auction._id	id	id of the auction that corresponds to the vote
target	$ or index	historical value of asset
Date	date	auction closing date
type	categorical	asset type
closeValue	$ or index	a predicted asset value
vote.createDate	datetime	date and time when the vote was submitted
vote.createDateLag	hours	how in advance a vote was submitted
user.earnedByAuction	$	user had earned*
user.earnedByAssetByAuction	$	user had earned by that asset type*

TABLE 3.1: Data description

* by the time auction started

Number of rows = 220985
Number of unique auctions = 2043

Each row corresponds to a unique vote and its features. There is information about the vote, an auction to which the vote belongs, and a user who made the vote. In addition, to each vote, there is a target value matched. All features were converted to numeric to make them available for aggregation and using as inputs in models. The dataframe includes information about all four auctions/asset types.

Initial data also have user's characteristics such as age, presence of an image, validation of email address, giving feedback, withdrawal requests and referral information. However, the data are not included in the table since the variables did not show any feature importance during training and testing models, and it was decided to remove them from the training dataset.

Important to mention that for weekends and holidays, target values are calculated using an average between the last known closing value and either the first known open (where available) or close one. If a target is unavailable for more than one day, weights are skewed by closeness. For instance, a Saturday target is calculated as the sum of 2/3 of the Friday target and the 1/3 of Monday.

The dataframe contains only votes submitted within the period of 7-20 days by auction closing (within 168 and 480 hours). In addition, auctions that consist of less than five records were not included in the further analysis. Figure 3.1 illustrates the distribution of all users' votes per auction.

Another noteworthy moment about the data frame is that we also cleaned data from rows with invalid values. For instance, where vote.createDateLag is less than one day. It was discussed and confirmed that these such votes are invalid and should be removed because for technical reasons.

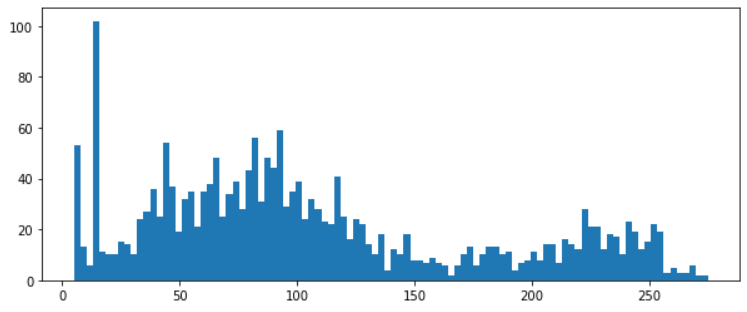

FIGURE 3.1: Distribution of users votes per auction. X-label represents number of votes per auction, Y-label is a frequency

3.2 Aggregated DataFrame

As mentioned in previous chapter, we cannot train our models using the standard dataframe as input since conceptually our task belongs to Multiple Instance Regression problem type.

To prepare data for models training, we created Aggregated Dataframe in the following way:

1. Unique auctions and corresponding types, target and DateOrder values were taken as a basis.

2. Last submitted vote and its user.earnedByAuction and user.earnedByAssetByAuction values for each auction were addedd.

3. Also, we included 1, 2, and 3 top creators' votes as separate features. We defined top creators as users who have the highest user.earnedByAuction values within an auction.

4. In addition to monitor the dynamics of vote changes, firstly, we divided data into the following parts: records only with vote.createDateLag values that are within 20-17, 20-17, 10-14, 20-8, and 20-7 days. Then we calculated mean of all and last 5 votes withing these subsets. The values were selected as separate features to Aggregated Dataframe.

As a result, we got 2043 rows × 61 columns dataframe, where each row refers to a unique auction. However, we did not include all the columns in the final modes since they did not have feature significance as was discovered during experiments.

3.3 Assets Volatility

As a separate point, we describe historical prices/indexes' volatility since it has a significant effect on dividing data into train and test sets and then evaluating results.

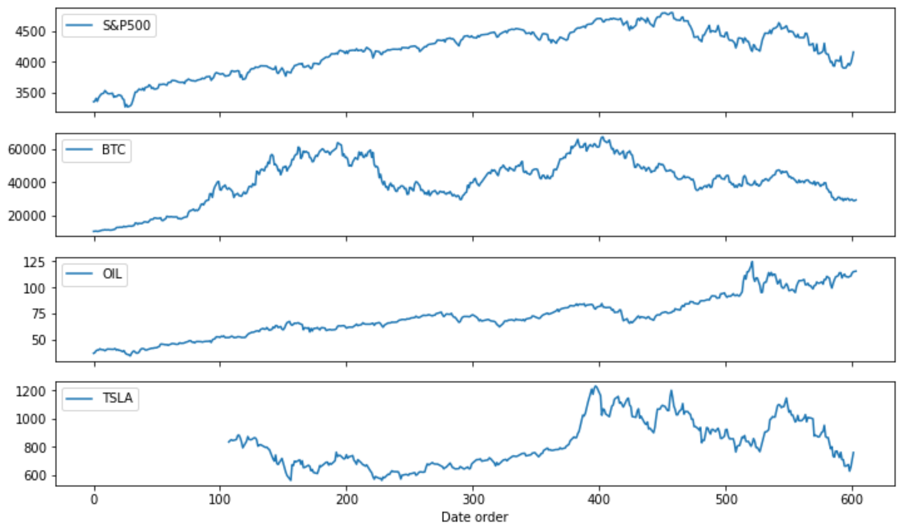

FIGURE 3.2: Historical Assets Prices

Figure 3.2 shows the asset prices during the periods we have our data. The four asset types significantly differ by the value ranges they have. Their volatility changes unevenly. In general, no one visible primary trend would refer to all together asset types.

Results from Table 3.2 are a confirmation of this. Important to mention that the values in the table are calculated only for assets corresponding to auctions from the Aggregated dataframe.

Asset type	Meaning	0-150	151-301	302-453	454-605
S&P500	Mean	3667.08	4168.85	4539.06	4376.7
	Volatility	219.11	152.31	121.75	218.31
	Count	82	151	151	149
BTC	Mean	27528.77	45684.48	51915.89	39623.41
	Volatility	15483.3	10659.2	7325.46	4802.65
	Count	76	151	150	151
OIL	Mean	47.4	66.82	73.96	98.68
	Volatility	10.0	4.91	5.85	11.42
	Count	58	151	151	151
TSLA	Mean	753.0	653.09	893.46	917.9
	Volatility	53.2	44.02	169.27	119.99
	Count	20	151	151	149

TABLE 3.2: Volatility

Chapter 4

Methodology

4.1 Outline

The model we develop should aggregate users' votes using submitted values and additional features corresponding to the votes to get the most accurate results to the target value. However, it should not predict an asset price using historical data.

As the primary approach, we chose Aggregated-MIR modified for our data. We plan to train our models on available features and known bag labels and test them using another part of the data.

As a criterion for comparing results, we selected mean squared error (MSE) and mean absolute percentage error (MAPE), which are calculated by the formulas:

$$MSE = \frac{1}{n} * \sum_{t=1}^{n} (A_t - F_t)^2$$

$$MAPE = \frac{1}{n} \sum_{t=1}^{n} |\frac{A_t - F_t}{F_t}|$$

where n is the number of forecasted values, A_t is the actual value, F_t is the forecast value.

MSE is a scale-dependent measure of prediction accuracy, while MAPE helps compare prediction accuracy across time series.

We do not use just sample means of all instances (votes) for every bag (auction) as aggregated data in final models. Since, while training models, our experiments showed that using only these features does not predict good results. That is why we decided to select also votes of 3 first top auction performers, the last vote at the moment of auction closing, and calculate the mean of all and the last five votes per periods within 20-17, 20-14, 20-11, 20-8 and 20-7 votes.Create lag ranges as described in Chapter 3.

In addition, we tried to use averages of first votes for auction and experimented with selecting different numbers of last votes. Nevertheless, the features did not show relevant results in model training, and we removed the data from the models' inputs. We also counted standard deviation, mean, minimal, and maximal values while aggregating votes on auctions levels but most of the features were highly correlated, and our models were overfitted. Thus, models learn data but not patterns in data.

We split data into two parts in time order: 75%, which refers to 1532 rows with accumulated data for model testing, and the last 25% rows, 511, for testing models. Before running models, the training subset was shuffled to avoid bias.

As a Baseline, we use an algorithm that calculates the predicted values based on the averages of the last five votes for every auction. Predicted results obtained from it we consider as initial, and evaluating on our model will be based on comparison with it. The primary goal is to develop an ML model with more precise vote aggregation than Baseline has.

In Table 4.1, there are the results of the Baseline algorithm. Although Train has a much higher MSE, it shows MAPE better than Test (about 1.5% over).

	Train	Test
MSE:	5,472,055.060	3,904,440.030
MAPE:	0.05063	0.06552

TABLE 4.1: Baseline

We use Random Forests, Boosted Regression Tree, and Extremely Randomized Trees models in the project. Their overview will be described in the following section.

To select the best parameter for models, we use hyperparameter tuning. It is the process of determining the right combination of hyperparameters that maximizes the model performance. For each model type, we make a separate hyperparameter optimization. The following steps are to compare the best results of the three types of models.

4.2 Models

Random Forest Regression is a Supervised Learning algorithm that uses the ensemble learning method for regression. The ensemble method is an ML approach that combines several base models to produce one optimal predictive model. That makes RFs predictions more accurate than a single Decision Tree model does. RF is a bagging technique so that trees in random forests are run in parallel, and there is no interaction among them while building, as it is shown in Figure 4.1 . Every tree is constructed from a different sample of rows, and at each node, a different set of features uses for splitting.

RS is one of the most accurate learning algorithms. It can handle many predictor variables and it runs efficiently on large databases. However on small datasets, it can overfit. It is also can overfit extremely noisy datasets. Also, RF Regression models have an extrapolation problem. The predicted values are never outside the training set values for the target variable. The model type is not used to predict time series data since it does not identifies a growing or decreasing trend. Another disadvantage of RF is that due to basing on the ensemble method, interpretability suffers and fails to define the significance of each variable.

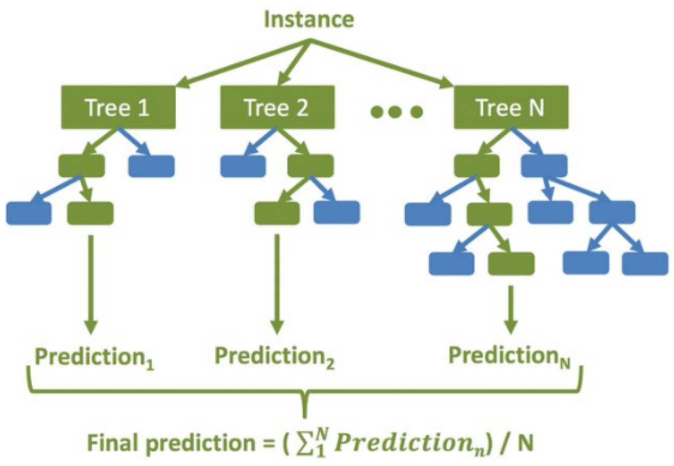

FIGURE 4.1: Random Forest Algorithm (Tyler C. McCandless, 2019)

Boosted Regression Tree (BRT) models are based on decision tree algorithms and boosting methods. They repeatedly fit many decision trees to improve the model's accuracy. BRT takes a random subset of all data for each new tree, and these random subsets have the same number of data . BRTs use boosting, which is an ensemble learning method. It combines a set of weak learners into a strong learner to minimize training errors using weights . So that the model continuously tries to improve tree accuracy. BRT is a robust stochastic algorithm and works very well with large datasets or datasets with many environmental variables. Such models are robust to missing values and outliers, but they require at least two train variables.

Extremely Randomized Trees (ERT, Extra Trees) is another ensemble machine learning algorithm based on decision trees. It creates a large number of unpruned decision trees from the training dataset. Predictions are made by averaging the prediction of the decision trees and taking their average in regression problems.

ERT is quite similar to RF. The first main difference is that ERT does not use bootstrap. The second difference is that Extra Trees split nodes randomly and do not use the best split as RF makes. In addition, the Extra Tree algorithm is faster.

All three model types work well on large datasets. In our task, it will be a big challenge to train models and avoid overfitting because of the relatively small dataset.

Chapter 5

Results

During models training, we faced the problem of overfitting. To handle it, we iteratively removed features that did show any importance. In our final train set, we left only 15 features, and all of them are based on closeValues and asset type variable.

5.1 Hyperparameters Tuning

To select the best parameters for models, we used hyperparameter tuning. Table 5.1 shows the parameters which were defined as the best after tuning, and we use them as inputs to our models.

Parameter	RF	BRT	ERT
number of trees in forest	50	75	40
max number of features	5	5	14
max depth of trees	14	11	14

TABLE 5.1: Tuned Hyperparameters

5.2 Models Analysis and Comparing

After the models ran, we obtained the results shown in Table 5.2. For all of them, MSE and MAPE metrics are significantly higher for the train set than for the test. For instance, the MAPE of the test set is about 7.4 times higher than in the train in the Random Forest model. It confirms that the models are still overfitting. Such results are not acceptable, and we can not consider the models reliable and compare their results with the Baseline.

Figure 5.1, 5.2, 5.3 and 5.4 illustrate in target values, Baseline, and RF model's predicted values for each auction type separately. The assets are significantly diverse. They differ in values that they have during the period data we have, and also, there is one base trend that refers to all the assets, as was mentioned in Chapter 3.3. So, we also experimented with running models for each asset type separately. However, the approach did not give actual better results.

Model	Metric	Train set	Test set
RF	MSE	473,932.990	11,685,704.260
	MAPE	0.01911	0.14005
BRT	MSE	57.510	13,880,764.340
	MAPE	0.01983	0.13368
ERT	MSE	15,594.680	11,995,119.700
	MAPE	0.00326	0.10875

TABLE 5.2: Empirical results

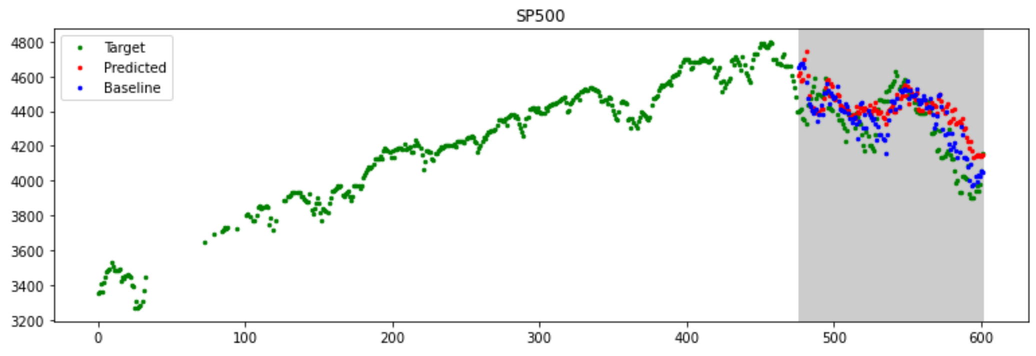

FIGURE 5.1: Random Forest model, comparing results for S&P500

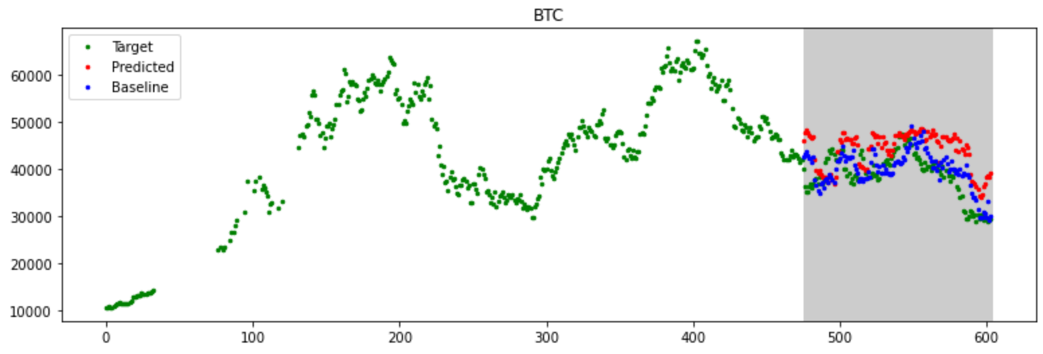

FIGURE 5.2: Random Forest model, comparing results for BTC

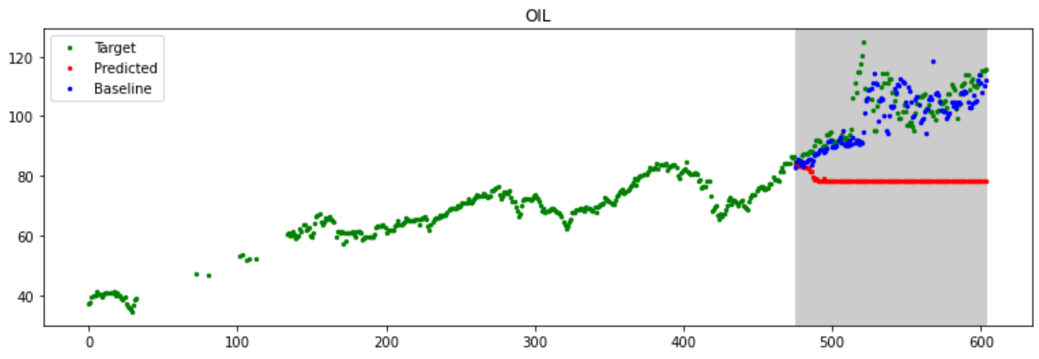

FIGURE 5.3: Random Forest model, comparing results for OIL

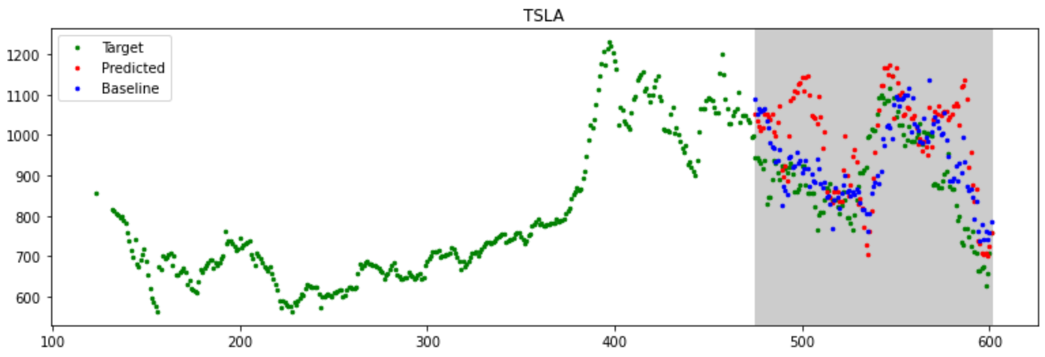

FIGURE 5.4: Random Forest model, comparing results for TSLA

Noteworthy to note that to avoid overfitting, we did many experiments by increasing and decreasing features number, iteratively. However, we failed with training models to predict unseen data.

Furthermore, we experimented with dividing data into a train, validation, and test parts. For instance, as in Figure 5.5, data are split by the following rules. The last 25% off all set are for testing, and auctions from every 80 and 10 days are used for train and validation. We experimented with selecting different ranges for training and validation, and some models showed promising results on the validation set, even better than Baseline. However, they could not work well on unseen data.

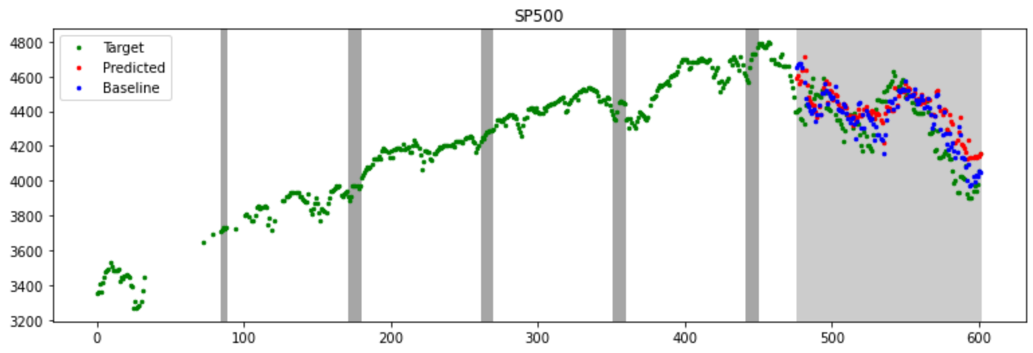

FIGURE 5.5: TrainValidationTest example, Random Forest model, SP500

Chapter 6

Conclusion

During the work, we tried to find the optimal solution to effectively aggregate votes data gathered on a crowdsourced financial platform to have precise predictions. The significant limitation we have is a relatively small set of data. We experimented with different sets of features, traintest split ranges, and parameters for models. Random Forest, Boosted Regression Trees, and Extremely Randomized Trees models were used. However, our current models should be enhanced. The possible improvements of our models can be extracting more features and using other accumulation ways.

Another approach to solving the problem can be using a benchmark for model pretraining. Such dataset should not mandatory include the financial market forecasts. It should have a similar structure, where users can submit votes. In addition, the users should be ranged.

After getting reliable validation results, our following step will be testing models on new unseen data. Also, we plan to validate the hypothesis that models trained on data referred to three auction types can predict the values of the fourth auction type well.

Bibliography

Buettner, Ricardo (2015). “A Systematic Literature Review of Crowdsourcing Research from a Human Resource Management Perspective”. In: DOI: 10.13140/2.1.2061.1845. URL: https://www.researchgate.net/publication/266209653_A_Systematic_Literature_Review_of_Crowdsourcing_Research_from_a_Human_Resource_Management_Perspective/stats.

Du, Xiaoxiao (2016). “Multiple Instance Choquet integral for classifier fusion”. In: DOI: 10.1109/CEC.2016.7743905. URL: https://www.researchgate.net/publication/311252334_Multiple_Instance_Choquet_integral_for_classifier_fusion.

Edoardo Manino Long Tran-Thanh, Nicholas Jennings (2019). “Streaming Bayesian Inference for Crowdsourced Classification”. In: URL: https://proceedings.neurips.cc/paper/2019/hash/54ee290e80589a2a1225c338a71839f5-Abstract.html.

Giuseppe Nebbione Derek Doran, Srikanth Nadella Brandon Minnery (2018). “Deep Neural Ranking for Crowdsourced Geopolitical Event Forecasting”. In: DOI: DOI:10.1007/978-3-030-19945-6_18. URL: https://www.semanticscholar.org/paper/Deep-Neural-Ranking-for-Crowdsourced-Geopolitical-Nebbione-Doran/2472f7bd3d3a3fb2e48fdec3f8d9302ccd9ce428.

Horn Christian Bogers Marcel, Brem Alexander (2018). “Prediction markets for crowdsourcing”. In: *Creating and Capturing Value through Crowdsourcing*. DOI: 10.1093/oso/9780198816225.003.0012. URL: https://oxford.universitypressscholarship.com/view/10.1093/oso/9780198816225.001.0001/oso-9780198816225-chapter-12.

J. M. Sadler J. L. Goodall, M. M. Morsy K. Spencer (2018). “Modeling urban coastal flood severity from crowd-sourced flood reports using Poisson regression and Random Forest”. In: DOI: 10.1016/j.jhydrol.2018.01.044. URL: https://www.researchgate.net/publication/324142279_Modeling_urban_coastal_flood_severity_from_crowd-sourced_flood_reports_using_Poisson_regression_and_Random_Forest.

Jeff Howe, Website (2006). *Crowdsourcing*. URL: https://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html (visited on 05/30/2022).

K. Wagstaff, T. Lane (2007). “Salience Assignment for Multiple-Instance Regression”. In: URL: https://www.semanticscholar.org/paper/Salience-Assignment-for-Multiple-Instance-Wagstaff-Lane/db88aa31d44fa890f57b0ffdb0de2fd125a73fdf.

Kiri L. Wagstaff Terran Lane, Alex Roper (2008). “Multiple-Instance Regression with Structured Data”. In: DOI: 10.1109/ICDMW.2008.31. URL: https://ieeexplore.ieee.org/document/4733948.

Natalie Parde, Rodney D. Nielsen (2017). “Finding Patterns in Noisy Crowds: Regression-based Annotation Aggregation for Crowdsourced Data”. In: DOI: 10.18653/v1/D17-1204. URL: https://www.researchgate.net/publication/322587059_Finding_Patterns_in_Noisy_Crowds_Regression-based_Annotation_Aggregation_for_Crowdsourced_Data.

Royal Museums Greenwich, Website (2021). *Longitude found - the story of Harrison's Clocks | Royal Museums Greenwich*. URL: https://www.rmg.co.uk/stories/topics/harrisons-clocks-longitude-problem (visited on 05/30/2022).

Trabelsi, Mohamed (2018). “Robust fuzzy clustering for multiple instance regression.” In: DOI: 10.1016/j.patcog.2019.01.030. URL: https://ir.library.louisville.edu/cgi/viewcontent.cgi?article=4056&context=etd.

Tyler C. McCandless, Sue Ellen Haupt (2019). “The super-turbine wind power conversion paradox: using machine learning to reduce errors caused by Jensen's inequality”. In: DOI: 10.5194/wes-4-343-2019. URL: https://www.researchgate.net/publication/333612054_The_super-turbine_wind_power_conversion_paradox_using_machine_learning_to_reduce_errors_caused_by_Jensen\%27s_inequality.

Zhuang Wang Liang Lan, Slobodan Vucetic (2011). “Mixture Model for Multiple Instance Regression and Applications in Remote Sensing”. In: DOI: 10.1109/TGRS.2011.2171691. URL: https://www.researchgate.net/publication/238594952_Mixture_Model_for_Multiple_Instance_Regression_and_Applications_in_Remote_Sensing.