





How to balance financial returns with metalearning for trend prediction

Alvaro Valentim Pereira de Menezes Bandeira   [Universidade de São Paulo | avalentim98@usp.br]

Gabriel Monteiro Ferracioli  [Universidade de São Paulo | ferracioligabriel@usp.br]

Moisés Rocha dos Santos  [Universidade de São Paulo | mmrsantos@usp.br]

André Carlos Ponce de Leon Ferreira de Carvalho  [Universidade de São Paulo | andre@icmc.usp.br]

 University of São Paulo, Av. Trab. São Carlense, 400, Centro, São Carlos, SP, 13566-590, Brazil.

Received: 1 May 2023 • Published: 27 February 2024

Abstract

The prediction of market price movement is an essential tool for decision-making in trading scenarios. However, there are several candidate methods for this task. Metalearning can be an important ally for the automatic selection of methods, which can be machine learning algorithms for classification tasks, named here classification algorithms. In this work, we present the use of metalearning for classification in market movement prediction and elaborate new analyses of its statistical implications. Different setups and metrics were evaluated for the meta-target selection. Cumulative return was the metric that achieved the best meta and base-level results. According to the experimental results, metalearning was a competitive selection strategy for predicting market price movement. This work is an extension of Bandeira *et al.* [2022].

Keywords: market movement, metalearning, stock market, machine learning

1 Introduction

The stock market, the gathering of buyers and sellers of stocks, has been an important activity for centuries. It can be defined as an environment where it is possible to buy and sell fractions of a company. Since it works with the supply and demand principle, the price of each stock (fraction of a specific company) varies over time [Teweles and Bradley, 1998]. Predicting this variation has always been a challenging problem that has taken much time from experts in this area. The most common way to represent market fluctuation is through a time series. With the advent of machine learning, predicting behavior in many sectors became possible, and it can also be applied to the stock market [Jordan and Mitchell, 2015].

Predicting market price movement is difficult when it is based on price alone. The most used approach in the literature seeks to characterize the market movement as a binary classification problem: the stock price will either fall or rise. Classical time series prediction methods, such as Autoregressive Integrated Moving Average (ARIMA), have accuracy ranging around 50% [Wen *et al.*, 2019]. New variables were explored in the literature to add relevant information regarding the asset to be traded, such as Natural Language Processing (NLP) and Technical Analysis, to improve the predictive performance [Mehtab and Sen, 2019].

Any binary classification algorithm can be used to classify price movement in the market. Thus, there are many possibilities to consider. The choice of the best algorithm by evaluating the predictive performance of all available approaches has a high computational cost and demands a good

knowledge on business model and machine learning algorithms [Prudêncio and Ludermir, 2004]. Metalearning is a set of methods that have been successfully used as an alternative to the costly work of algorithm selection. It differs from traditional manual selection machine learning algorithms by using the experience gained from past machine learning tasks to obtain better predictive performance faster and more efficiently in future tasks [Brazdil *et al.*, 2008].

This work is an extension of Bandeira *et al.* [2022] published in the 2022 edition of The Symposium on Knowledge Discovery, Mining, and Learning (KDMiLe). In this piece, we focus on improving the understanding of the experiment, going deeper into explanations and the interpretability of the data, adding the industry descriptions layer to the study, and a new section centered on analyzing the metafeatures and their contributions.

The main contribution of this work is to propose an automatic algorithm selection approach through metalearning that can bring benefits to trading market operations, not restricted to the financial market but extendable to other markets, for example, the electricity market. Another relevant analysis consists of considering different metrics for the selection of algorithms, having achieved the best results in terms of financial simulation when considering the financial return accumulated at the meta-level.

This paper is structured as follows. Section 2 goes into a brief explanation about the related works; Section 3 describes the materials and methods for the proposed approach; Section 4 presents the results and evaluation insights; Section 5 presents the metafeatures analysis; Section 6 presents the conclusions and future research directions.

2 Related Work

As cited before, some works used metalearning for algorithm selection and other works used models based on machine learning for trading tasks, but to the best of our knowledge, no work has attempted to solve both problems at the same time. This work's main objective is to apply metalearning to select algorithms for predicting price movement in the financial market. Starting with a general overview, different solutions will be shown in order to present the current state of the art in market movement prediction.

Charkha [2008] provides a comprehensive introduction to trading in the financial market using machine learning. The work presents a straightforward experiment to build a model, explaining how to use a stock dataset as input. Since the work does not use additional features beyond the stock dataset, it is significantly different from this experiment.

2.1 Feature Selection Strategies

Ni *et al.* [2011] used an improved fractal feature selection with an ant colony algorithm to select features for the learning model. The idea behind fractal feature selection is to measure the importance of each technical index (i.e., the features) by testing the impact of removing each one from the dataset. This process demands a lot of computational power to test different features, so the ant colony algorithm becomes useful in order to mitigate this problem.

Feature selection methods can be divided into wrapper methods (which evaluate how good a feature is based on the performance of a model) and filter methods (which select features based on statistical tests on the dataset). Literature has shown works that use only one of the methods, as well as combining them to obtain different strategies [Huang *et al.*, 2008; Lee, 2009].

2.2 Incorporating Relative Return

Zhang *et al.* [2018] is a recent work that goes beyond using feature selection and market movement prediction, it also uses the model relative return to compare models. A similar metric is going to be used in this present experiment. Since this concept of return was already explored in the past, it has potential, which will be explored in section 4.

2.3 Dynamic Model Selection

The most similar work found in the literature, by Dong *et al.* [2021], is able to dynamically evaluate and select prediction models, based on an initial list of models. Despite the existence of a similar idea of dynamic selection, the work uses a clustering algorithm to evaluate the candidates for predictors, which is different from what is presented here.

3 Experiments

In this section, we explain the development process of the recommendation system, from selecting data sources to the

base-level and meta-level evaluation methods. All experiments are publicly available at a GitHub repository ¹.

3.1 Data acquisition

The data used in this experiment is gathered from two main indexes of the stock market and a general list of stocks, which contain an amount of time-series compound by the days of trades, the prices of open, close, high, and low, and the volume of trades.

Although all the data used came from the global stock market, the list of selected companies was obtained with three different strategies:

- **S&P 500 (abbreviation of Standard & Poor's 500 Index) [Tsaih *et al.*, 1998]:** an index that represents a specific list of stocks in the market. In this case, a weighted index of about 500 of the most relevant stock companies in the USA. This index was first introduced in 1957 and is still one of the most famous until now, this shows how useful it can be as a starting point. The selection is not only tied to the top companies because there are other criteria of S&P Dow Jones Indices, the company that maintains the index.
- **Wilshire 5000 [Haugen and Baker, 1991]:** Differently from the S&P 500 index, the Wilshire 5000 tracks the whole stock market in the US. The index company criteria inclusion is that the stock must be publicly traded and the headquarters in the USA. There is no stock limit in the index, despite its name. Just by including smaller companies, the Wilshire 5000 provides a more complete scenario of the US than the S&P 500. It was created in 1974, holding nearly 5000 stocks during launch. In 1998, the number of companies in the index was about 7500, and currently is under 3500. The index companies can be reviewed monthly, and there is a possibility to adjust.
- **Use all stocks available:** With a web crawler algorithm is possible to get a list of all companies that are listed on any stock website. The page we used was Stock Monitor ², and all stocks listed on this site were used in this work, despite being companies with good or bad performance in the market. This can be helpful for metalearning because it can promote diversity of data. This list has about 5100 stock names. Since the first two sources of stocks we used are bound to the US, this was an alternative to get data of other relevant companies, but that are tied to other countries.

After getting the stock list, the methodology is the same for the three variations. We first iterate by reading each stock name and using the Yahoo Finance tool ³ to get the dataset of the company.

Every time series is represented as a Pandas DataFrame⁴. This format arranges the data in a table-like structure. To

¹Project Repository: github.com/ferracioli/Market-Movement-Prediction-Algorithm-Selection-by-Metalearning

²Stock Monitor URL: <https://www.stockmonitor.com>

³Yahoo Finance URL: <https://finance.yahoo.com>

⁴Pandas documentation: <https://pandas.pydata.org/>

maintain the chronological aspect for programming and modeling purposes, each timestamp serves as the index. The columns associated with this index contain data specific to each time observation, as depicted in Figure 1. With a dataframe, it is possible to iterate through each observation in chronological order, preserving the temporal sequence. Alternatively, it is possible to extract all elements from a specific column to gather insights, for instance, to obtain statistical data.

Some of the listed names can not be found while searching the databases. There is only one restriction: The selection of the stocks with at least 800 observations in their time series. That is the minimal data quantity we fixed for a good result. If the stock has more than 1500, we get only the 1500 most recent observations. Both lower and maximum thresholds (800 and 1500 observations) were set as arbitrary values for practicality since focusing on optimizing these parameters can drain time from the main problem. This strategy was set to select only the time series of the last period rows, but there is another reason for standardizing a period of time for the series. Some companies are old enough to accumulate more than 10000 in trading days. While using a small dataset (less than 800 observations) can result in a poor model, with insufficient data to avoid high bias, the large series can store patterns from decades ago that are not compatible with our current stock market. This was the solution we found to fix the unbalanced price history that each company has.

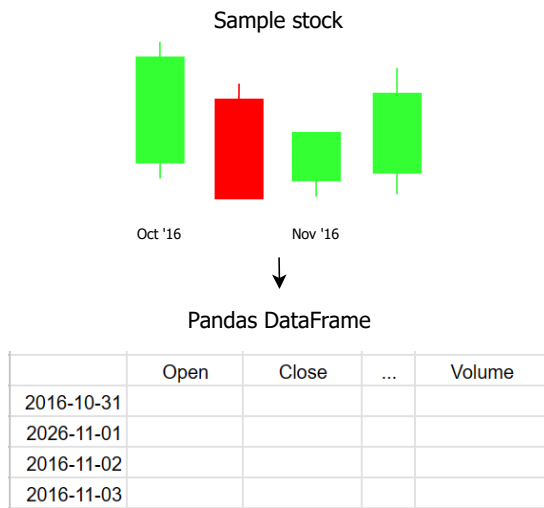


Figure 1. Representation of time series as tabular data

Our final rate of success databases obtained with the whole market list was near 70%, and the sectors with the most stock names discarded were Healthcare and Financial Services since there are a lot of recent companies that were not considered in the minimal row quantity. The reproduction of this step may not result in the same list of companies because some of them may differ according to time, since we rely on Yahoo Finance, a tool that couldn't fetch some companies we asked for. The Stock Monitor list also can change, since some stocks may be added or removed from the stock market. In order to obtain the most faithful reproduction of this mining step, we recommend using the same closing date for the

Yahoo Finance tool, that is April 22, 2022, with the indexes constituents shared in our repository. With the stock lists from different sources, we concatenated them into a unique dataset and removed duplicates as a strategy to improve the diversity of companies during the learning step. In addition to being aware of duplicates, we standardize the names of the sectors, since each source used different names for the same category, e.g. Consumer Cyclical, which in some sources is called Consumer Discretionary. It is important to be aware of different names for the same sector, because we are dealing with indexes that are composed from different sources.

3.2 Price Trend Prediction Problem

The price trend prediction problem involves forecasting the future direction of an asset's price movement. This problem is inherently challenging due to the complex interplay of various market factors, including market sentiment, macroeconomic indicators, trading volumes, and historical price patterns. Researchers and practitioners have developed numerous methodologies, ranging from traditional statistical approaches to more sophisticated machine learning techniques, to tackle this problem [Ghosh et al., 2022].

Some target horizons:

- **Close-to-close:** The close-to-close target horizon encompasses the prediction of price movement from the closure of one trading day to the closure of the subsequent trading day. This horizon serves as an indispensable tool for short-term investors, encapsulating overnight developments that significantly influence market dynamics post-trading hours.
- **Intra-Day:** The intra-day prediction horizon revolves around forecasting price movements within a single trading day. This target is paramount for high-frequency traders seeking to capitalize on short-lived market fluctuations, necessitating real-time data streams and sophisticated modeling techniques.
- **Day ahead:** The day-ahead target horizon involves predicting price movements from the close of the current trading day to the close of the subsequent trading day. This horizon caters to medium-term investors, allowing them to incorporate overnight events and market trends into their strategies.

This work adopts a close-to-close target. The choice of the close-to-close target horizon stems from its unique blend of practicality and stability. While each prediction horizon has its merits, close-to-close predictions strike a balance between short-term volatility and overnight developments, making them particularly valuable to day traders and swing traders.

3.3 Experiment Design

Before explaining the experiment architecture, we are going to present the general pipeline that occurs in metalearning:

- **Base-level learning:** The first step of the pipeline, solves the main problem using various models, and comparing its performances. For generating the meta-dataset, the best algorithm is stored for each time series,

with its metafeatures. The algorithm will be used as the target later on, and the metafeatures are the data used in this classification problem. An in-depth explanation for each of these words will be given in the following sections. This step is expected to be the longer one because the idea is to generate the metadata once, and then use it many times, that is the faster step.

- **Meta-level learning:** step that occurs in the sequence of the previous. Using the generated metadataset, it is possible to improve the solution of new problems. A recommendation system can learn specific features and behaviors that happen in the data distribution, and make suitable predictions of the best model for solving the problem.

3.4 Base-level learning

We are dealing with the stock market and price oscillation, so it is important to define another important section of the experiment, which has more options for implementation, that is how the model will operate trades. There are several possibilities of trade logic to be constructed: compare the difference between the opening and closing price on the same day, the closing of one day and the opening of the next day, or closing prices between two days. For this work, we chose one that seemed more interesting: define the market tendency as the difference between the closing price between this and the previous day. This affects how the experiment will be constructed because up and down trends can be predicted with classification models.

Figure 2 shows a diagram with the pipeline we used during the base-level learning of the experiment, presenting a flow of the steps we are presenting during this section.

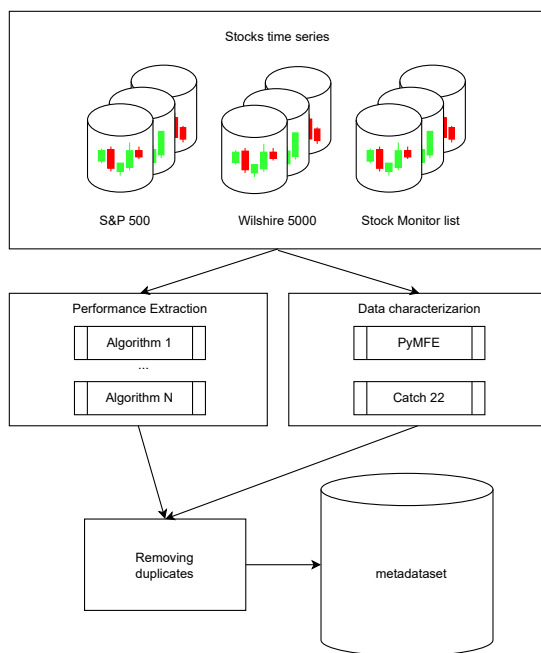


Figure 2. Base-level learning architecture

As shown in Figure 2, two steps are needed to generate the metadata: performance extraction and data characterization.

The data characterization step comprehends extracting relevant information related to the algorithm's performance called metafeatures. We combined two metafeatures sets implemented on Python libraries:

- **PyMFE [Alcobaça et al., 2020]:** is a Python package for extracting various types of metafeatures for the classification task. The PyMFE package aims to enhance the reproducibility of metalearning applications by curating a set of meta-features gathered from various works in the literature. The package encompasses diverse categories of meta-features: General, Statistical, Information-theoretic, Model-based, Landmarking, Clustering, Concept, Itemset, and Complexity. For the present study, we extract 55 General and Statistical metafeatures due to their simplicity and low computational cost of extraction. For this last one, we can cite Pillai's trace [Pillai, 1955], used as a test statistic in multivariate analysis, and Skewness, which is the degree of asymmetry observed in a probability distribution, calculated as the third statistical moment.
- **Catch22 [Lubba et al., 2019]:** a high-performance library that extracts 22 canonical time series metafeatures. The 22 characteristics of time series are selected from a larger set of 7000 features from the `hctsa` package Fulcher et al. [2013]. They are referred to as canonical features as they constitute a concise representation of the 7000 features, empirically prioritizing predictive performance, computational efficiency, and interpretability. As examples, we can cite the time intervals between successive extreme events above (or below) the mean and the proportion of successive differences exceeding 0.04 standard deviation. Further explanation of each of the features can be obtained from its reference.

The data characterization is a step that may have custom functions, so instead of only two libraries, it is possible to add a custom function that extracts information for a specific data domain.

The Performance Extraction occurs in parallel to this step, and again, can have custom models according to the data domain of the problem that is being solved. We apply eight different algorithms during base-level learning. A ranking system selects the best algorithm for that specific stock. The name of the company and its sector are also stored for data analysis reasons. Since the time series of each stock is relatively large, we are using a time series split of size four (this split results in 5 sub-groups of time series and using the sliding-window strategy) and the mean of the result of each split. The algorithms used in the base-level are Random Forest (RF) [Ho, 1995], Decision Tree (DT) [Breiman et al., 2017], Extreme Gradient Boosting (XGB) [Chen and Guestrin, 2016], K-Nearest Neighbors (KNN) [Cover and Hart, 1967], Support-Vector Machine (SVM) [Vapnik, 1998], Naive Bayes (NB) [McCallum et al., 1998], Adaptive Boosting (ADA) [Freund and Schapire, 1997] and Logistic Regression (LogReg) Cox [1958].

For each time series, we extract the performances by running each base-learner and calculating performance metrics. In this work, two metrics were used to define the meta-target,

balanced accuracy, and cumulative return:

- **Balanced Accuracy (BAC):** we have chosen to use a built-in *SKLearn* metric, since is a more efficient solution than rewrite an accuracy function. This metric is a good option for handling imbalanced datasets. It is defined as the average between specificity and recall concerning the Equation:

$$BAC = \frac{recall + specificity}{2} \quad (1)$$

- **Cumulative Return (CRet):** The cumulative return uses a specific function that predicts if the stock is in a trend of High or Down movement. If the market goes up, it buys one unit of the stock and sells the next day. Using the same logic, if the market is going down, it operates in short, selling stock and repurchasing it the next day. The cumulative return is the sum of all the operations done in the whole period. A negative cumulative return means the algorithm lost money. In that way, it is hard to compare cumulative return between different stocks because it is related to the volatility of the stock. So, this is only useful for comparing different algorithms applied in the same stock. It is important to state that we choose to always operate one unit of each stock at a time because it ensures more stability. Another possibility would be to trade a variable volume of stocks, depending on how much money is available for the model to make orders. This option may have exponential growth since any fluctuation or small difference in performance between models can result in a very different ranking. This means the dynamic volume orders are unstable compared with a fixed volume. We are also considering that if the model has a negative balance, we will allocate more money for the following trades during the simulation. The formal notation for CRet in the period i is:

$$CRet_i = (1 + CRet_{i-1}) \times (1 + Ret_i) - 1 \quad (2)$$

where Ret_i is the financial return in the period i .

We can build our metadata by combining the data obtained in data characterization with the data from performance extraction. As stated before, we are dealing with a classification problem, so one must select a target for the metalearning step. We have two target options: Balanced Accuracy and Cumulative Return, data obtained during the performance extraction. We can only test one target type at a time, so two versions of the metadata will be tested.

3.5 Meta-level learning

Next step is applying the metadataset with the meta-level learning. Figure 3 shows the difference between a regular ML model and a model that uses metalearning for recommendation problems.

Metalearner is a machine learning algorithm applied to metadata for meta-knowledge extraction. That is, it will be able to recommend a promising model from past tasks. This

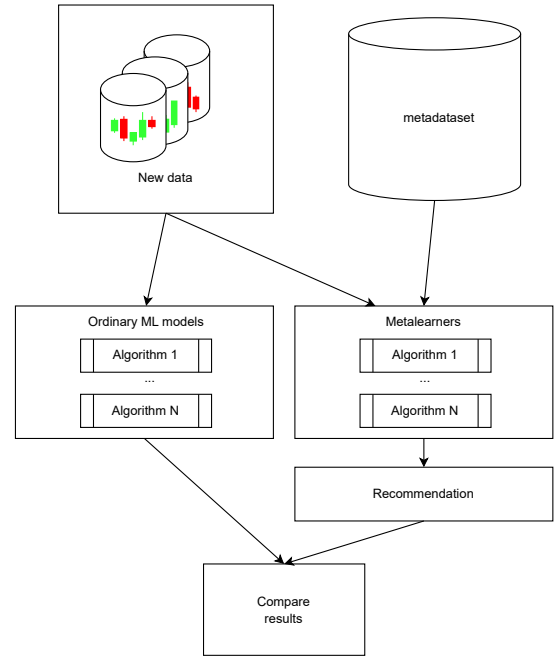


Figure 3. Meta-level learning architecture

work used four metalearners options: KNN, SVM, RF, and DT.

We can use the meta-knowledge obtained by metalearners to verify a new stock, applying the same functions of metafeature extraction and trying to recommend the most promising algorithm. Cross-validation is used in this 80% of datasets for training, and we use the 20% of data for the testing. Using the Balanced Accuracy, we verify if its value is greater than 1/8 (12,5%), the maximal randomness, which means the probability of choosing an algorithm at random given the number of classes.

To add statistical support, we apply a *Mann-Whitney-Wilcoxon test* [Mann and Whitney, 1947] *two-sided with Bonferroni correction* [Dunn, 1961] (which is the nonparametric counterpart of the Student's t-test for independent samples), comparing all the possible pairs of base-level models distributions, considering the following hypothesis:

H_0 : The distributions of both populations are identical.

vs

H_1 : The distributions of both populations are not identical.

4 Results and Discussion

The section shows the performance of each variation of our metadata. Since we are using two variations of the metadata, there will be a comparison between them, with a discussion about on which occasion each one is more relevant than the other.

4.1 Meta-level

We analyze the meta-level data by looking at whether the metadata components can differentiate their models based

on the extracted metafeatures. As the first step in this exploration, we plotted Figure 4 and Figure 5, which show the distribution of the base-level learners using respectively the Balanced Accuracy and Cumulative Return as meta-targets.

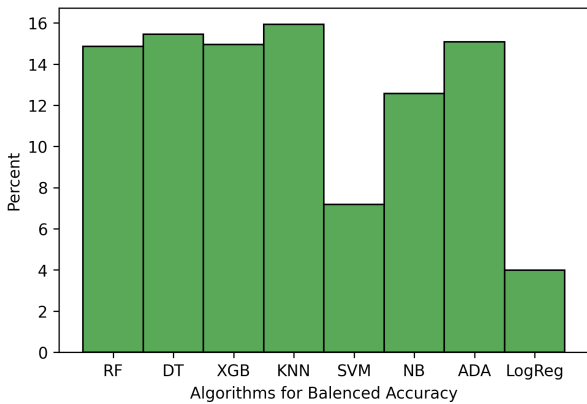


Figure 4. Distribution of algorithms over the metadata with Balanced Accuracy as target

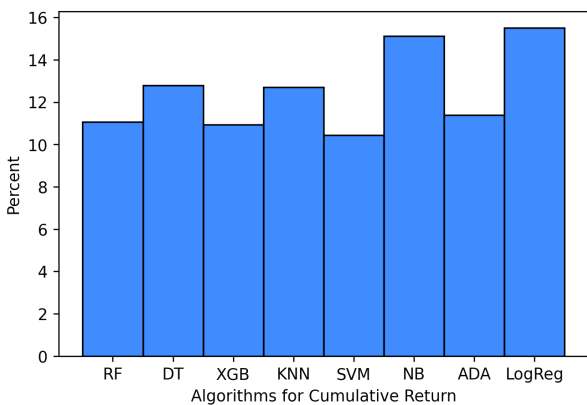


Figure 5. Distribution of algorithms over the metadata with Cumulative Return as target

There is a considerable unbalance of learning algorithms distribution in Figure 4 with a small presence of SVM and LogReg, and the models are more equally distributed in Figure 5. Unbalance in the distribution can be a problem during the recommendation step because it may tend to the most frequent models to classify an unknown stock time series.

As an additional synthesis, we explored the companies' distribution over the metadataset. Understanding the data distribution is useful because if a sector appears with less frequency, its influence in base-level learning will be smaller. Table 1 shows the count of companies grouped by sector, using the same metadataset.

Table 1 shows that the first three sectors, Health Care, Financials and IT represent practically half of the data. Unbalance can lead to biased or inaccurate model predictions, but it is present in the Sectors column of data, so it may have a smaller impact since it is not our primary target. We have to verify in the next steps if this unbalance will affect our experiment.

As a next step in verifying the data unbalance and metadata behavior, we test how accurate recommendations are for both meta-targets. So, doing a 10-fold of the data with

Table 1. Count of companies used in metadataset grouped by sector

Sector	Companies count
Health Care	860
Financials	779
Information Technology	568
Industrials	550
Consumer Discretionary	428
Energy	253
Real Estate	240
Materials	234
Communication Services	224
Consumer Staples	183
Utilities	96
Total	4415

ten repetitions, we trained the four metalearners to see how good are the recommended performances.

Figure 6 and Figure 7 show the performance of the four metalearners in Balanced Accuracy. In the first case, the recommended target was Balanced Accuracy, and for the second, the Cumulative Return.

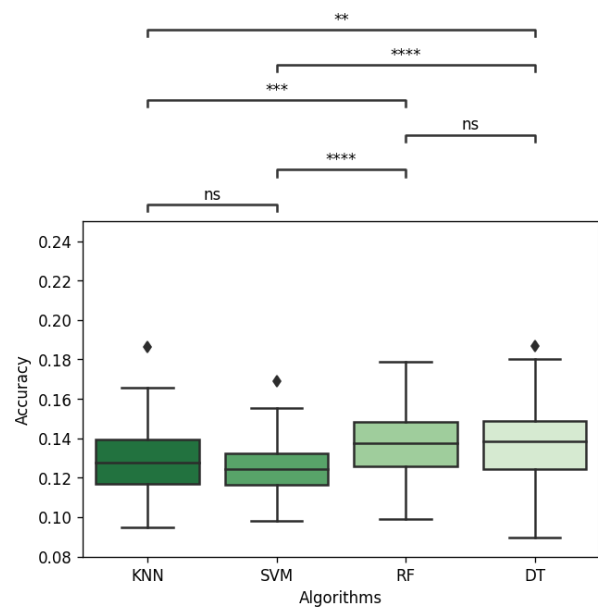


Figure 6. Recommendation based on Balanced Accuracy metric. In the hypothesis test, 'ns' means nonsignificant, and each '*' means a decrease in the p-value for the test.

In Figure 6, with 5% of significance level, we can spot the difference between most of the models for the Balanced Accuracy metric, but they are still at the same level of performance, i.e., although the algorithms have different learning mechanisms, they have essentially the same result. Moreover, all of them have a median of about 0.135, which is a poor result considering the quantity of metadata that was extracted. A possible reason for this result to be near the random guess (0.125) is that the balanced accuracy should not be a good choice of target for this task. Another possible explanation is that the unbalanced model distribution results in the metalearners making wrong recommendations.

Focusing now on Figure 7, according to the test results, with 5% of significance level, there is no significant differ-

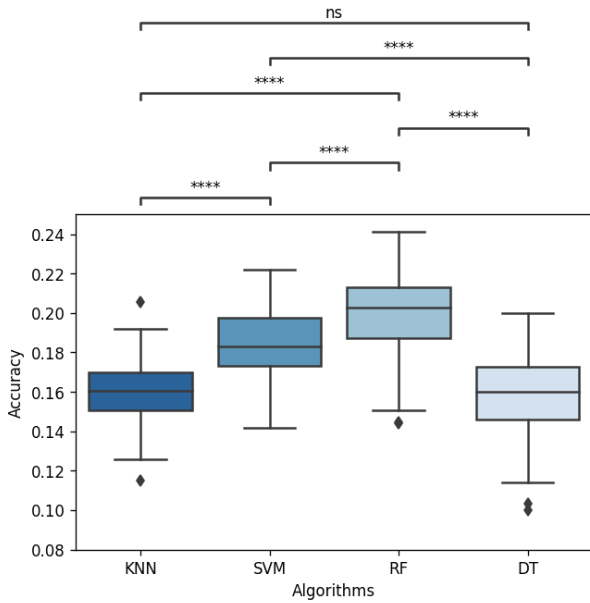


Figure 7. Recommendation based on Cumulative Return metric

In the hypothesis test, 'ns' means nonsignificant, and each '*' means a decrease in the p-value for the test.

ence between the distributions of the pair *KNN* v.s. *Decision Tree*, which is not true in the other cases, where we see that the distributions are different and Random Forest proved to be the best metalearner.

Besides that, the plot shows that even the worst algorithm has the metric median above the probability of randomly choosing the best algorithm for one specific stock. This means that the recommendation using the metadataset for Cumulative Return can outperform a random choice of algorithm. Although the accuracy seems low in this multi-class classification problem, it doesn't necessarily mean a significant loss in the cumulative return, considering that, in most cases, the performance of the best algorithm is not that far from the second and third places, and sometimes a different seed or hyperparameter could close this gap. This shows that it is more promising to apply metalearning with a focus on cumulative return rather than balanced accuracy as a meta-target, as it has a better result in terms of recommendation. To confirm this choice, we analyze metalearning at the base-level.

Considering the Sectors as a new target in the metadataset, instead of Balanced Accuracy or Cumulative Return, we tested how the model could perform classifying the sectors based only on its own metadata. The result is shown in Figure 8, and has the same structure as the previous plots:

There are 11 sectors in total, which means that a random guess model would have a general performance of accuracy of around 0.09. The lowest median of the models has a balanced accuracy above 0.18, considerably higher than the performance of random guess. This means that the characteristics of each sector are unique enough to be used as identifiers. Despite being additional information, this reaffirms the previous discussions about how metalearning can find similar patterns in different stocks.

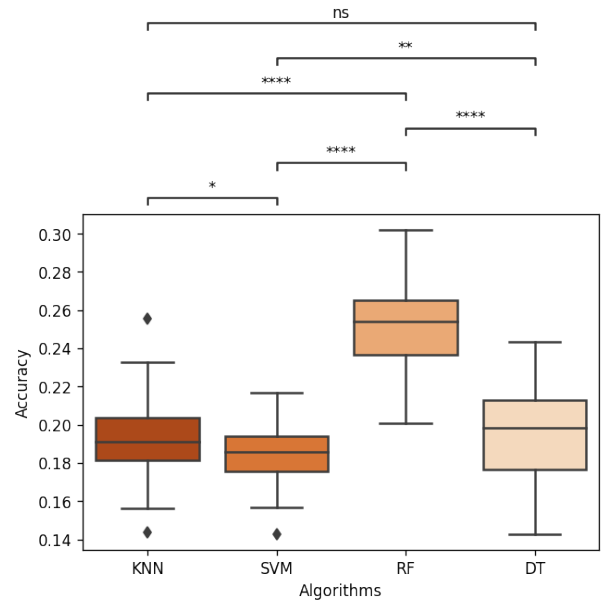


Figure 8. Recommendation based on Sector labels

In the hypothesis test, 'ns' means nonsignificant, and each '*' means a decrease in the p-value for the test.

4.2 Base-level

The base-level evaluation compares the eight models used in the metadataset construction with our recommendation of models that we call MetaMM (Meta Market Movement). There are two variations of this system, BAC (recommendation focused on balanced accuracy meta-target) and CRet (based on the cumulative return meta-target). Our last objective is to see if the MetaMM recommendations can outperform the profit of the best base-level model. During the Meta-level, we saw that the Random Forest model was the most efficient in the task of recommending stocks, so the MetaMM will use it again for comparison in the base level.

As stated in the previous section, each company time series may have a different size, ranging from 800 to 1500 rows, and we are going to use the average cumulative return as the performance metric for each model. MetaMM models will also have an average cumulative return, based on recommendation. We used 20% of data for testing, so we are going to calculate the average gain for 883 companies. Figure 9 shows the performances of cumulative return for the ten different trading models with the average gain, in USD.

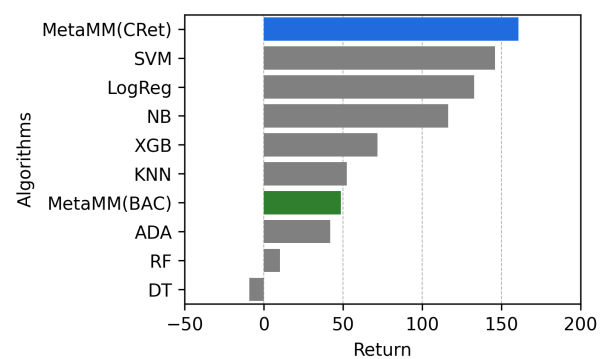


Figure 9. Comparison of Returns between recommendations and other models

Based on the Figure 9 results, the MetaMM (CRet) strat-

egy was the one with the best performance, becoming more efficient than the best base-level model (SVM). Besides that, the MetaMM (BAC) result was an inferior performance. The only model that ended the test with loss was DT, and the others were capable of getting a positive result.

Using the data obtained from the base-level task, we can also show the model performances in a structured way, in order to understand the possible limitations of the models. Table 2 shows additional information about each model, including both MetaMM versions.

Table 2. Model performances during base-level learning

Model	Avg gain	Min	Max
ADA	41.95	-26731.25	61868.40
DT	-9.06	-22236.25	18119.84
KNN	52.34	-29484.25	62378.16
NB	116.47	-9052.80	63034.02
RF	10.08	-26530.25	17290.88
LogReg	132.75	-551.06	64435.34
SVM	146.04	-621.46	63035.34
XGB	71.86	-20682.75	61775.48
MetaMM(CRet)	160.59	-329.4	61868.40
MetaMM(BAC)	48.54	-20682.75	61868.40

This table endorses what was seen before the MetaMM model is capable of avoiding bad models since its minimal gain is the smallest. Even if its maximum gain is not the best, the possibility of avoiding high losses ensures that this model will have a high average gain.

It may not seem so intuitive at first that the minimal gain from MetaMM(CRet) is far different from every other value. Actually, this means that each model can have its minimal value in different time series. Since MetaMM(CRet) uses a recommendation system, it can take advantage of the variation of the performance for each scenario to avoid giant losses, reducing the risk in each operation.

One last piece of knowledge we can extract from the meta-data is that high balanced accuracy does not imply a high cumulative return since the performance of these two metrics is relatively different. We saw with the Figures that the meta-data results with balanced accuracy as the target was inferior, not showing any advantage compared with more straightforward strategies or focusing on cumulative return. So in practice, if we had to choose only one metric, the cumulative return probably would be the best option because protecting assets is more important than having a greater accuracy rate for the financial market.

5 Metafeatures analysis

Given our results, we can analyze which metafeatures are that most contributed to our achievements in this experiment.

For this purpose the Shapley Additive Explanations [Lundberg and Lee, 2017] method was used as resource in this analysis. Originally designed in the game theory subject, this method assigns a unique distribution of payoffs among the players in a coalition game, calculating the average marginal

contribution of each player to the results obtained considering all possible permutations of the players in the game. In the machine learning context, we can consider that the game is the prediction task and the players are the features that cooperate with each other to get the best outcome.

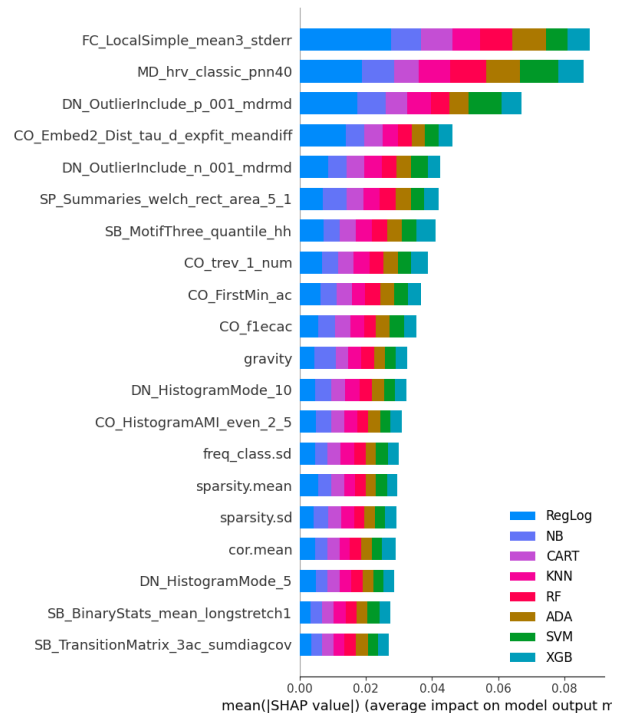


Figure 10. Summary plot for the Shap values

In the Shap values summary plot, represented in Figure 10, all feature importance bars for each class are stacked, giving us an overall score for our metafeatures, which represents how impacting it is for model prediction. Taking the three most important metafeatures we get 'FC_LocalSimple_mean3_stderr', which is a measure of error from using the mean of the previous three values of the time series to predict the next value, 'MD_hrv_classic_pnn40' that represents the proportion of different magnitudes that are greater than 4% of the standard deviation of the time series, and 'DN_OutlierInclude_p_001_mdrmd' that measures the timing intervals of outliers occurrence relative to the start and end of the time series.

As we can notice, not only these three, but all features of the top ten most important are part of the Catch22 library, which indicates that, in this case, specialized time series metafeatures contribute a lot more than the ordinary statistical ones found on PyMFE library.

To proceed with the analysis we compared the relationship between the most important features and their Shap values by a scatter plot hued by another important variable called a Dependency plot.

Observing Figure 11 we can notice that higher Shap values observations are more common where the error of using the mean of the prediction on lag equals 3 has the lowest values. With that said we can state that low values of this feature can be really significant to predict our target in the experiment. In addition to that, higher proportions of different magnitudes

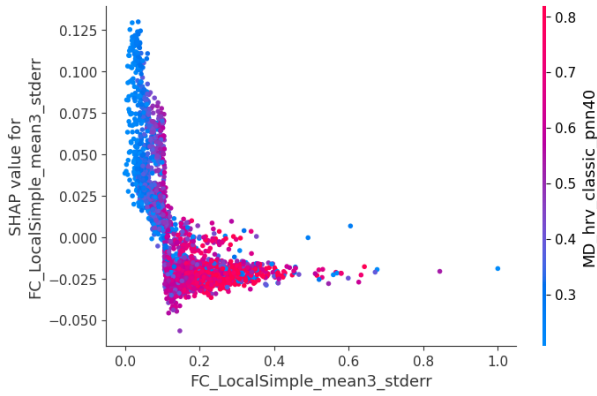


Figure 11. Dependence plot between 'FC_LocalSimple_mean3_stderr' and 'MD_hrv_classic_pnn40'

that are greater than 4% are concentrated around the value zero of Shap values, meaning that it is not representative if compared with its lower values.

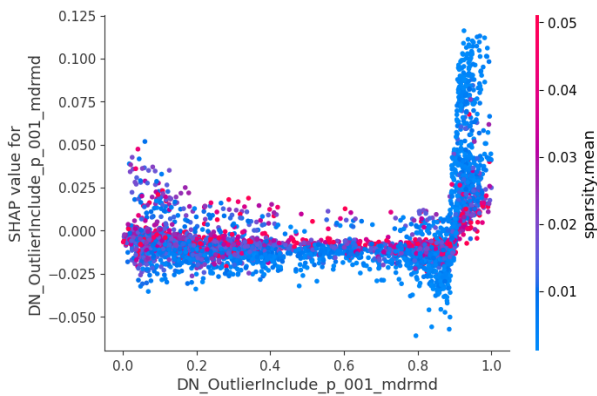


Figure 12. Dependence plot between 'DN_OutlierInclude_p_001_mdrmd' and 'sparsity.mean'

For the Figure 12 we can see the opposite happening to the 'DN_OutlierInclude_p_001_mdrmd' variable, where the highest values achieve the higher Shap values while the other values stay around zero. Comparing the behavior of a feature that is not as important, such as 'sparsity.mean' (native of PyMFE), we can see that its higher values range around zero, implying that in the relation between the two variables only lower values of sparsity means must be considered in model prediction, in this case.

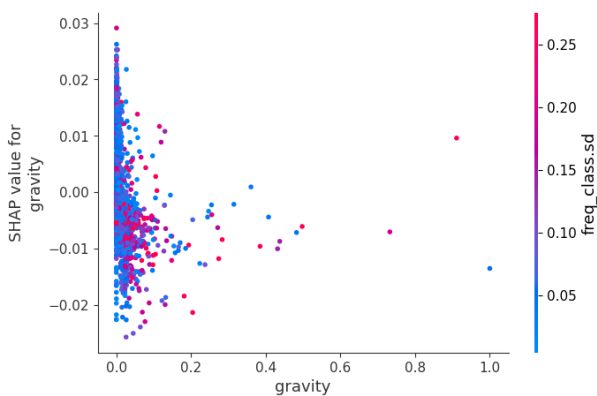


Figure 13. Dependence plot between 'gravity' and 'freq_class.sd'

Now analyzing the interaction of the two better features of PyMFE library ('gravity' and 'freq_class.sd') in

Figure 13 we cannot take a straight conclusion about its importance to the model prediction as we did before with Figures 11 and 12, which corroborates with the previous statement based on Figure 10 that metafeatures extracted from the Catch22 library retains higher importance in our recommendation task if compared with the ones extracted from PyMFE library.

6 Conclusion

The selection of machine learning algorithms in the market movement prediction task is essential in decision-making in stock market trading scenarios. From metalearning, it was possible to recommend promising models for the task, reducing the computational cost and the need for specialist knowledge to select models in the market movement prediction task. An important finding of this analysis was that the meta-target based on cumulative return presented the best results at the meta-level and the base-level, being the best choice for representing the performance in this task. The metafeatures which contribute the most to model recommendation are the ones natives of the Catch22 library. In future research directions, we propose to test other metafeature approaches and explore relationships in which each classification algorithm benefits specific types of time series. Another aspect that can be further improved is to use iteration to discover more efficient thresholds to filter the number of observations in each time series since arbitrary numbers were used.

Acknowledgements

This study was partially funded by grant 2019/10012-2, São Paulo Research Foundation (FAPESP), CEPID-CeMEAI Process 2013/07375-0, USP Research Office (PIPAE project), and CNPq. This R&D project is being developed in partnership with VOLT, CESP (Companhia Energética de São Paulo), and Auren Energia through the Research and Development Program regulated by the National Electric Energy Agency (ANEEL).

References

Alcobaça, E., Siqueira, F., Rivolli, A., Garcia, L. P. F., Oliva, J. T., and de Carvalho, A. C. P. L. F. (2020). Mfe: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, 21(111):1–5.

Bandeira, A., Ferracioli, G., dos Santos, M., and de Carvalho, A. P. L. F. (2022). Market movement prediction algorithm selection by metalearning. In *Anais do X Symposium on Knowledge Discovery, Mining and Learning*, pages 1–8, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/kd-mile.2022.227947.

Brazdil, P., Carrier, C. G., Soares, C., and Vilalta, R. (2008). *Metalearning: Applications to data mining*. Springer Science & Business Media.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (2017). *Classification and regression trees*. Routledge.

Charkha, P. R. (2008). Stock price prediction and trend prediction using neural networks. In *2008 First International*

- Conference on Emerging Trends in Engineering and Technology*, pages 592–594. IEEE.
- Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.
- Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.
- Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232.
- Dong, S., Wang, J., Luo, H., Wang, H., and Wu, F.-X. (2021). A dynamic predictor selection algorithm for predicting stock market movement. *Expert Systems with Applications*, 186:115836.
- Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American statistical association*, 56(293):52–64.
- Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.
- Fulcher, B. D., Little, M. A., and Jones, N. S. (2013). Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of the Royal Society Interface*, 10(83):20130048.
- Ghosh, P., Neufeld, A., and Sahoo, J. K. (2022). Forecasting directional movements of stock prices for intraday trading using lstm and random forests. *Finance Research Letters*, 46:102280.
- Haugen, R. A. and Baker, N. L. (1991). The efficient market inefficiency of capitalization-weighted stock portfolios. *The journal of portfolio management*, 17(3):35–40.
- Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.
- Huang, C.-J., Yang, D.-X., and Chuang, Y.-T. (2008). Application of wrapper approach and composite classifier to the stock trend prediction. *Expert Systems with Applications*, 34(4):2870–2878.
- Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.
- Lee, M.-C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications*, 36(8):10896–10904.
- Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., and Jones, N. S. (2019). catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33(6):1821–1852.
- Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.
- Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.
- McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.
- Mehtab, S. and Sen, J. (2019). A robust predictive model for stock price prediction using deep learning and natural language processing. *arXiv preprint arXiv:1912.07700*.
- Ni, L.-P., Ni, Z.-W., and Gao, Y.-Z. (2011). Stock trend prediction based on fractal feature selection and support vector machine. *Expert Systems with Applications*, 38(5):5569–5576.
- Pillai, K. S. (1955). Some new test criteria in multivariate analysis. *The Annals of Mathematical Statistics*, pages 117–121.
- Prudêncio, R. B. and Ludermir, T. B. (2004). Meta-learning approaches to selecting time series models. *Neurocomputing*, 61:121–137.
- Teweles, R. J. and Bradley, E. S. (1998). *The stock market*, volume 64. John Wiley & Sons.
- Tsaih, R., Hsu, Y., and Lai, C. C. (1998). Forecasting s&p 500 stock index futures with a hybrid ai system. *Decision support systems*, 23(2):161–174.
- Vapnik, V. (1998). *Statistical Learning Theory*. NY: Wiley.
- Wen, M., Li, P., Zhang, L., and Chen, Y. (2019). Stock market trend prediction using high-order information of time series. *Ieee Access*, 7:28299–28308.
- Zhang, J., Cui, S., Xu, Y., Li, Q., and Li, T. (2018). A novel data-driven stock price trend prediction system. *Expert Systems with Applications*, 97:60–69.