Pontifícia Universidade Católica do Rio de Janeiro

Departamento de Economia

Undergraduate Thesis

A Comparison of Boosting Methods for Stock Market Risk Prediction

Rafael Stutz Pereira Martins

2010989

RIO DE JANEIRO
June 2024

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Rafael Stutz Pereira Martins

A Comparison of Boosting Methods for Stock Market Risk Prediction

Advisor: Claudio Cardoso Flores

I declare that the present work is my own and that I did not resort to any form of external help to complete it, except when authorized by the advising professor.

RIO DE JANEIRO
June 2024

The opinions expressed in this work are solely and exclusively the responsibility of the author.

Agradecimentos

Aos meus pais, Edmilson e Solange, por serem meus exemplos fundacionais, por sempre desejarem a minha felicidade e por todos os sacrifícios que fizeram para que eu pudesse chegar até aqui. Agradeço todo o apoio e incentivo para minha formação acadêmica.

À minha companheira de vida, Camila, por trazer uma beleza e profundidade que eu nunca havia conhecido. Agradeço pela ajuda com 'pitacos' sobre o trabalho e me desculpo pelas noites de sono sacrificadas.

Ao meu orientador, Claudio, por todo auxílio na confecção desta monografia. Agradeço sua solicitude e direcionamento, fundamentais para a conclusão do trabalho.

Ao professor João Renato, meu antigo tutor na USP, por toda gentileza e por ajudar a definir meu caminho em um dos momentos mais difíceis da vida.

Aos meus amigos Conrado, Lucca, Gabriel e Victor Hugo, por todo apoio e pela revisão do trabalho. Sem vocês, a minha experiência na graduação teria sido bem mais entediante.

Abstract

The integration of machine learning (ML) algorithms in economic research has grown significantly in recent years. This undergraduate thesis focuses on applying these algorithms to predict stock market risks, which are critical for informing decisions made by policymakers, financial institutions, and investors. This work explores the efficacy of five boosting algorithms in forecasting three risk measures based on the Bovespa index (IBOV) returns: the Conditional Value at Risk, the Standard Deviation of Returns, and the Maximum Drawdown. The study investigates the theory of boosting methods, including AdaBoost, Gradient Boosting, XGBoost, LightGBM, and CatBoost, and compares their performance based on model metrics such as RMSE, MAE, training time, prediction time, and feature importance. The research uses Bovespa index data from August 2006 to May 2023, applying feature selection and hyperparameter tuning with random search, evaluated through cross-validation and key metrics. Key findings indicate that (i) the Spearman correlation was the most effective for feature selection, (ii) market sentiment and technical indicators were the most impactful in model outcomes, and (iii) LightGBM was the best algorithm for general risk prediction, while Adaboost and Gradient Boosting were the best algorithms for specific risk measures.

Keywords: Boosting Algorithms, Supervised Machine Learning, AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost, Bovespa Index, Market Risk, Feature Selection.

Resumo

A integração de algoritmos de aprendizado de máquina (ML) na pesquisa econômica cresceu significativamente nos últimos anos. Esta monografia se concentra na aplicação desses algoritmos para prever os riscos do mercado de ações, que são fundamentais para informar as decisões tomadas por formuladores de políticas, instituições financeiras e investidores. Este trabalho explora a eficácia de cinco algoritmos de boosting na previsão de três medidas de risco com base nos retornos do índice Bovespa (IBOV): o *Conditional Value at Risk*, o desvio padrão dos retornos e o *Maximum Drawdown.* O estudo investiga a teoria dos métodos de boosting, incluindo AdaBoost, Gradient Boosting, XGBoost, LightGBM e CatBoost, e compara seu desempenho com base em métricas como RMSE, MAE, tempo de treinamento, tempo de previsão e importância de variável explicativa. A pesquisa usa dados do índice Bovespa de agosto de 2006 a maio de 2023, aplicando a seleção de variáveis e o otimização de hiperparâmetros com busca aleatória, avaliados por meio de validação cruzada e as métricas-chave. Os principais resultados indicam que (i) a correlação de Spearman foi a mais eficaz para a seleção de variáveis, (ii) os indicadores de sentimento de mercado e técnicos foram os mais impactantes nos resultados do modelo e (iii) o LightGBM foi o melhor algoritmo para a previsão geral de riscos, enquando o Adaboost e o Gradient Boosting foram os melhores algoritmos para a previsão de medidas específicas de risco.

Palavras-chave: Algoritmos de Boosting, Aprendizado de Máquina Supervisionado, AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost, Índice Bovespa, Risco de Mercado, Seleção de Variáveis.

LIST OF SYMBOLS AND NOTATION

$\mathbf{x}$	A vector of predictors
x_i	The i-th vector or scalar predictor example from a dataset
y_i	The i-th response example associated with x_i
(x_i, y_i)	A supervised dataset ordered pair
$\hat{y}_i$	The i-th predicted value associated with x_i
$\tilde{y}$	A binary $\{-1, +1\}$ target
$\hat{y}_{R_T}$	the mean response for the training examples within the t-th high-dimensional region
θ	A parameter vector
θ_t	A set of parameters at the t-th iteration of an algorithm
$\hat{\theta}_t$	An estimated set of parameters at the t-th iteration of an algorithm
Θ	A parameter space
ω_{it}	Normalized weights at the t-th iteration for the i-th data point
W_t^+	A weighted sum of weights at the t-th iteration, where each weight ω_{it} is included in the sum when the predicted value $F(x_i, \hat{\theta})$ equals the true response value $\tilde{y}$
W_t^-	A weighted sum of weights at the t-th iteration, where each weight ω_{it} is included in the sum when the predicted value $F(x_i, \hat{\theta})$ does not match the true response value $\tilde{y}$
$\mathcal{D}$	A dictionary of weak learners
g_{it}	The empirical gradient of the loss function at the t-th iteration for the i-th data point
h_{it}	The empirical hessian of the loss function at the t-th iteration for the i-th data point
$R(\cdot, \cdot)$	A risk function
$L(\cdot, \cdot)$	A loss function
$b(\mathbf{x}, \theta_t)$	A basis function, where $\mathbf{x}$ are input features and θ_t is a vector of parameters

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

The relevance of Machine Learning in economic research is on the rise, as evidenced by recent studies, such as those by Gogas and Papadimitriou (2021) and Athey (2018). The former examines the early applications of machine learning in economics, citing studies from as early as 1974, and observes a contemporary trend of merging Machine Learning models with traditional econometric methodologies. The latter predicts changes in the empirical work done by economists, pointing to the increase in the adoption of off-the-shelf Machine Learning methods for intended tasks such as prediction, classification, and clustering. In essence, Athey's work predicts the development of more robust statistical models and advancements that will lead to new research areas, methods, and questions with the help of machine learning.

The economy suffers detrimental effects from financial crises, resulting in rising unemployment and depressed economic growth. Particularly, the poor could be disproportionally affected, as shown in Rewilak's paper (2018). Chatzis et al. (2018) analyze the transmission mechanisms of crash events in stock markets by applying a range of statistical machine learning techniques. Their findings suggest interdependence among stock market crashes and cross-effects within stock, bond, and currency markets. Moreover, developing effective forecasting methods for stock market risk is not a simple task, especially when considering the context of the Brazilian stock market (COSTA; BAIDYA, 2003). Therefore, accurate predictions for stock market risks are crucial for policymakers, financial institutions, economists, and a wide range of other groups.

Among all supervised machine learning models (2.1), our focus lies in exploring boosting algorithms in the context of stock market risk prediction. Boosting is a versatile machine learning technique that emphasizes the incremental improvement of weak learners to create a high-performing model. It has gained prominence in various applications, winning multiple data science competitions. The paper (GRINSZTAJN; OYALLON; VAROQUAUX, 2022) found that tree-based models, such as the gradient-boosted decision tree models, still outperform deep learning methods in tabular data, yielding better predictions with less computational costs. Given its effectiveness, it is possible and valuable to apply boosting algorithms for risk forecasting.

1.2 OBJECTIVE

The objective of this work is to study boosting algorithms and compare their performance on stock market risk prediction. First, this monograph develops a description of machine learning concepts and different implementations of boosting algorithms. Subsequently, it employs an empirical analysis of boosting methods applied to stock market risk prediction, using the returns of the Bovespa index (IBOV) to calculate market risk. This work objective is to explore the following:

1. Investigate the theory behind boosting methods;
2. Compare different boosting implementations in stock market risk prediction;
3. Identify the most effective dataset and feature selection method for each risk measure based on model performance metrics;
4. Investigate the interpretability of boosting models in the context of stock market risk prediction.

In this research, risk forecasting is carried out using five boosting methods of varying sophistication. The ultimate goal is to compare them and determine which boosting method offers the most robust and reliable performance in financial risk prediction, as well as to identify the best dataset and feature selection method for each risk measure.

It is important to emphasize that this work is not limited to the empirical analysis alone. Consequently, the underlying theory of boosting methods is thoroughly explored to establish a deeper understanding of how these techniques work and why they can be effective.

This study does not aim to evaluate causal effects or compare traditional forecasting tools and other machine learning models for volatility and other risk measures with boosting models. Instead, it focuses on comparing boosting models within their group, exploring their theoretical background, and analyzing their empirical results.

1.3 STRUCTURE

This undergraduate thesis is organized into six chapters. They are arranged in a logical progression: Introduction, Foundational Concepts, Literature Review, Methodology, Results, and Conclusion. The specific order is defined to facilitate the reader's understanding of this research. Notably, suppose the reader lacks familiarity with boosting and its context. In that case, the Foundational Concepts (2) and Literature Review (3) chapters aim to provide the necessary background information for understanding the subsequent chapters.

The Foundational Concepts chapter (2) first elaborates on prerequisite concepts necessary for understanding boosting and subsequently explains boosting algorithms. Chapter (3), the literature review, examines the evolution of boosting algorithms and part of the existing research on boosting applications in finance. The methodology chapter (4) aims to provide the empirical research design, exploratory data analysis, and evaluation procedures. Results are presented in (5), with the empirical findings along with discussions. Conclusions drawn from the overall study are presented in (6) with an analysis of the research limitations and possible future works. Moreover, there is a supplementary material section exploring some of Adaboost's properties.

2 FOUNDATIONAL CONCEPTS

2.1 SUPERVISED LEARNING

Machine learning techniques enable systems to discover patterns in data, making predictions and decisions without explicit programming. In this study, we are focused on a particular type of machine learning paradigm: supervised learning.

In supervised learning, we provide the learning algorithm with labeled data, where each example consists of input features and corresponding output labels (LEARNED-MILLER, 2014). Through the process of learning from these labeled examples, supervised learning algorithms can generalize and make predictions on unseen data, and our goal is to achieve optimal performance in this process.

Data is commonly separated into two subsets: the training set and the testing set. The training data consists of $\{(x_i, y_i)\}_{i=1}^{n}$ ordered pairs, where $\mathbf{x}$ is the set of predictors and y is the response variable. The testing data consists of $\{(x_i, y_i)\}_{i=n+1}^{m}$ ordered pairs, analogously to the training set but unseen by the algorithm.

Given the input-output pairs, we aim to discover a good approximation of the true functional relationship between $\mathbf{x}$ and y, $y = f(\mathbf{x})$. This function that aims to approximate f is a hypothesis. Moreover, f frequently takes the form of a joint probability distribution model $P(x, y)$. Essentially, learning is the process of discovering a hypothesis that generalizes well to new instances beyond the training set, such that it correctly predicts y in new examples within the test set.

The setting of the learning problem can be described as an optimization task. Ultimately, the learning algorithm, which is typically a computer program, seeks to adjust its parameters or model structure to minimize a defined objective function, commonly referred to as a loss function. This function, denoted as $L : \mathbb{R} \times \mathbb{R} \to [0, \infty)$, measures how different the prediction $f(x, \hat{\theta}), \hat{\theta} \in \Theta$, provided by the learning algorithm, is from the true value y. The goal is to minimize the risk functional, defined as

$$R(\theta) = \mathbb{E}[L(f(x, \theta), y)] = \int L(f(x, \theta), y) dP(x, y)$$

over the class of functions $f(x, \theta), \theta \in \Theta$, where Θ is a parameter space. However, the joint probability distribution $P(x, y)$ is not known to the learning algorithm, and $R(\theta)$ can't be evaluated. To solve this problem, the risk functional is replaced by an estimate, the empirical risk functional

$$R_{emp}(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i, \theta), y_i)$$

which is the average of the loss function built upon the training set. From the Empirical Risk Minimization (ERM) principle,

$$\hat{\theta} \in \text{argmin}_{\hat{\theta}\in\Theta} R_{emp}(\theta),$$

the learning algorithm should select a hypothesis - or prediction rule - that minimizes the empirical risk (VAPNIK, 1999). With some theoretical guarantees, discussed in-depth by Vapnik in [1991], the empirical risk minimization performs comparably well to the true risk minimization.

2.2 WEAK AND STRONG LEARNERS

The notion of weak learnability, rooted in PAC (Probably Approximately Correct) learning theory, was established by Kearns and Valiant (KEARNS; VALIANT, 1989). A weak learner, by definition, is a model that exhibits only a marginal improvement over random guessing. In regression problems, it gives results slightly better than the mean.

In the Standard PAC Model, the learner must be able to produce a hypothesis with error at most ε, for arbitrarily small positive values of ε, and must find a good approximation to the target with probability at least $1-\delta$, for arbitrarily small positive values of δ. Therefore, this model - known as a strong learning model - demands that the learner finds good enough hypotheses with small error rates. In contrast, these conditions are relaxed in the Weak Learning Model: the learner is required to produce hypotheses with error rates slightly less than 1/2.

To qualify a weak learning algorithm more formally in the context of classification, we define a training sample $\{(x_i, \tilde{y}_i)\}_{i=1}^{n}$, where $x \in \mathbb{R}^n$ and $\tilde{y} \in \{-1, +1\}$, along with a probability distribution $p = \{p_i\}_{i=1}^{n}$ over the training sample, where $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = 1$. A learning algorithm $F(\cdot, \theta)$ is γ-weak with margin $\gamma > 0$ if, for each p, there exists $\hat{\theta}$ such that:

$$\sum_{i=1}^{n} p_i \mathbb{1}_{\{F(x_i;\hat{\theta}) \neq \tilde{y}_i\}} \leq \frac{1}{2} - \gamma$$

Therefore, for all the distribution we assume there exists a parameter $\hat{\theta}$ such that $F(\cdot, \hat{\theta})$ is at least a little better than a random guess.

Furthermore, the decision tree (2.4) stands out as the most frequently employed weak learning model, primarily due to the ease with which its weakness can be regulated by adjusting the depth of the tree during its construction.

2.3 ENSEMBLE LEARNING

Ensemble learning involves training and combining multiple learners to address a learning problem. Essentially, it deals with two steps: building a dictionary $\mathcal{D} = \{F_t(\mathbf{x})\}_{t=1}^{T}$ of weak learners $F_t(\mathbf{x})$ and then fitting a model $f_T(\mathbf{x}) = \sum_{t \in \mathcal{D}} \beta_t F_t(\mathbf{x})$. This approach can be further categorized into two traditional methods: parallel and sequential ensembles.

In the parallel approach, exemplified by techniques like Bagging and Random Forests, each model is built independently, and the main idea is to combine them to reduce variance. On the other hand, in the sequential approach, represented by methods like Boosting and Bayesian Additive Regression Trees, the models are built sequentially, trying to add new learners that do well where the previous ones underperform. Moreover, an ensemble typically exhibits a significantly enhanced generalization ability compared to an individual learner, especially when dealing with weak learners (ZHOU, 2021).

2.4 DECISION TREES

Decision trees are a non-parametric supervised learning technique - they make no strong assumptions about the underlying distribution of the data being studied. In the context of boosting, they are the most used weak learners. Due to their flexibility, it is straightforward to configure them as base learners, limiting their maximum depth and, consequently, model complexity. Thus, shallow trees commonly have high bias, low variance, and limited predictive ability, being good candidates for weak learners. In this context, a decision stump is a single-level decision tree (shallow tree) that can be represented as a step function

$$f(x) = \begin{cases} k_1 & x \leq s \\ k_2 & x > s \end{cases}$$

with two leafs denoted by k_1 and k_2, and a split point defined by s, which sets the boundary between the leaves. Figure 2.1 represents a stump, in which we can visualize a hierarchical structure that begins with the root node, in white, and extends to the leaf or terminal nodes, in orange color. Figure 2.2 depicts the step function f, with its single input feature and the threshold value s.

Figure 2.1 – *Decision stump*

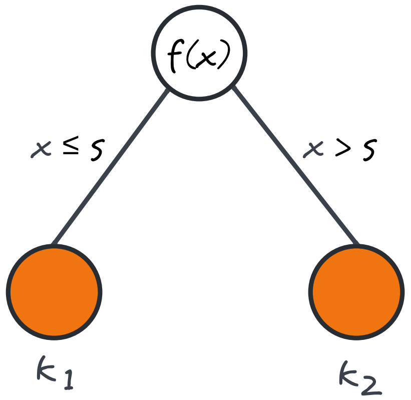
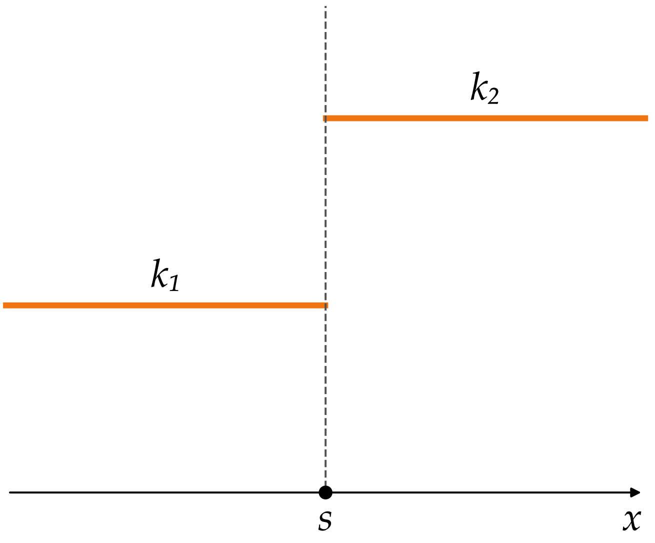


Figure 2.2 – *A basic stump as a step function*



In essence, trees aim to segment the predictor space - the space of the input variables - into simple rectangle areas by a specific rule system. With greater formality, trees partition the set of $\mathbf{x}$ values into T distinct regions $\{R_t\}_{t=1}^{T}$ that do not overlap. To determine these regions, we need to find the rectangles that minimize the Residual Sum of Squares (RSS)

$$\sum_{t=1}^{T} \sum_{i \in R_T} (y_i - \hat{y}_{R_T})^2.$$

As clearly explained in (JAMES et al., 2014), this process is done by recursive binary splitting since it is not computationally feasible to consider every partition of the predictor space into T high-dimensional rectangles. This recursive process is fulfilled by selecting a particular feature x_t and a threshold s that, when used to divide the feature space into regions $\{\mathbf{x} \mid x_t \geq s\}$ and $\{\mathbf{x} \mid x_t < s\}$, results in the greatest decrease in RSS. This approach, however, normally builds trees that are too complex, leading to overfitting (HOTHORN; ZEILEIS, 2006). Pruning is a common way to mitigate this problem, as it leads to a reduction in tree size. This technique works by removing parts of the tree that do not substantially decrease its predictive performance, improving its generalization capacity.

Finally, it is important to observe that trees, as discussed in (HASTIE et al., 2009), have several advantages since they can naturally handle missing data, are fast to fit, scale well to large datasets, are robust to outliers, can capture nonlinear relationships in the data, and have numerous other benefits. However, they also come with drawbacks, including instability, high variance, and a lack of smoothness. Another important factor to consider is interpretability, which varies substantially and depends on the complexity of the decision trees: shallow trees can be easily understood, while deep ones might be very difficult to interpret. Therefore, their explainability is heavily influenced by the depths of their leaves, as shown empirically by Piltaver et al. in (2016).

2.5 REGULARIZATION TECHNIQUES

The primary focus when training a machine learning model is its generalization capacity. Boosting models exhibit susceptibility to overfitting - the model fails to capture the functional relationship between the input data and the response variable but focuses on memorizing the training data. However, regularization techniques enhance the model's ability to perform well on unseen data by helping to control the training process. This discussion will solely address important regularization methods to the development of this study. Specifically, we will explore shrinkage, which is an explicit technique, as well as early stopping and subsampling, which are implicit techniques.

Shrinkage is a widely used explicit regularization method. Essentially, in the context of boosting, it is employed to diminish the influence of an added weak learner, penalizing the relevance of each successful iteration of the algorithm. The direct proportional shrinkage is a simple implementation (NATEKIN; KNOLL, 2013) :

$$\hat{f}_t(x) = \hat{f}_{t-1}(x) + \lambda\hat{\beta}_t F_t(x_i, \theta)$$

, where $0 < \lambda \leq 1$. In this method, each update is scaled by the value of λ, which significantly impacts the training error, and smaller values lead to better generalization but slower convergence.

It is also important to understand L_1 and L_2 shrinkage strategies, or Lasso (TIBSHIRANI, 1996), and Ridge (HOERL; KENNARD, 1970) regression since they are used in modern boosting models, such as XGBoost. To illustrate the optimization problem when applying both L_1 and L_2 regularization techniques:

$$\text{argmin}_w \left\{ \sum_{i=1}^{n} L(y_i, F(x_i, \theta)) + \lambda_1 ||\mathbf{w}||_1 + \lambda_2 ||\mathbf{w}||_2^2 \right\}$$

where, using trees as base learners, $\mathbf{w}$ is a vector of leaf weights, $||\mathbf{w}||_p$ is the ℓ^p-norm of the vector $\mathbf{w} = (w_1, ..., w_n)$, and $\lambda_i \in \mathbb{R}^+$. The penalties are applied to leaf weights in this context rather than feature weights, as seen in linear or logistic regression. In L_1 regularization, the ℓ^1 norm of $\mathbf{w}$, $||\mathbf{w}||_1$, is added to the loss function and regulated by the parameter λ_1, controlling the sum of absolute values of the leaf weights, causing them to shrink towards zero. This process results in a more sparse model and performs variable selection. Conversely, in L_2 regularization, the squared ℓ^2 norm of $\mathbf{w}$, $||\mathbf{w}||_2^2$, is added to the objective function and is controlled by λ_2, composing a penalty term to regulate the sum of squares of the leaf weights, encouraging them to decrease similar to L_1 regularization, but ensuring that they do not reach zero. Consequently, the L_2 penalties tend to produce smoother weight distributions compared to L_1 penalties.

Another important regularization technique to consider is subsampling. This method introduces randomness at each iteration of the boosting algorithm by employing only a random part of the training data to fit a successive weak learner. It requires a parameter, $0 < \omega \leq 1$, to determine the proportion of the training data used at each iteration. Suppose $\omega = 1$, the original training procedure is applied without subsampling. It is important to note that introducing randomness leads to a decrease in accuracy, enhancing the variance of individual base learners. However, the fitted learners will be more dissimilar, and this increased diversity has the positive effect of decreasing covariance between the predictors. Thus, this method implies a trade-off between diversity and accuracy (CHANDRA; CHEN; YAO, 2006), increasing the variance of individual weak learners but typically reducing the overall variance of the ensemble model.

Finally, we discuss Early Stopping: a regularization method that essentially relies on having a validation set, a performance metric, and defined stopping criteria. This technique enables us to determine the optimal number of iterations necessary for building a model that generalizes well to unseen data. In this sense, the iterative algorithm runs for a predetermined number of iterations, and it reduces overfitting by monitoring the training and validation performance gap throughout the training phase and stopping it at an optimal point. Thus, we choose a performance metric, such as MSE or Accuracy, and can set a rule to stop the algorithm if the validation error increases for more than k, where $k \in \mathbb{R}^+$, monitoring and computing the performance metric on the validation set after each iteration of the boosting algorithm.

2.6 GRADIENT DESCENT

Gradient descent is a method, or iterative algorithm, for the optimization of a multivariable differentiable function without imposing constraints on the function variables. This method is useful for minimizing a loss function $L(F(\mathbf{x}, \theta))$, parametrized by a set of parameters θ, iteratively moving in the opposite direction of the gradient of the function toward a local minimum. Moving in the direction of the gradient results in approaching a local maximum of a function, so we take the opposite direction to find the steepest descent path.

Assuming that $L(\cdot)$ is defined and differentiable, we do the following:

$$\theta_{t+1} = \theta_t - \rho \nabla L(\theta_t)$$

where $\nabla L(\theta_t)$ is the gradient of the loss function at the step t, $\rho \in \mathbb{R}^+$ is an adjustable learning rate that determines the step size to reach the local minimum, and θ_t is the parameter vector at step t. The term $\rho\nabla L(\theta_t)$ is subtracted from θ_t, moving in the direction in which L decreases locally the fastest, finally updating the value of θ in θ_{t+1}, as illustrated in the algorithm:

```
Algorithm 1: Gradient Descent
  Input: Function to minimize $L(\cdot)$; learning rate $\rho$; initial value $\theta_0$; number of
         iterations $T$
  Output: Optimal parameter $\theta_T$
  $\theta_1 \leftarrow \theta_0$;
  for $t = 1$ to $T$ do
      $\theta_{t+1} \leftarrow \theta_t - \rho\nabla L(\theta_t)$;  // Direction of locally greatest function decrease
  return $\theta_T$;
```

2.7 BOOSTING ALGORITHMS

2.7.1 Forward Stagewise Additive Modeling

Boosting can be understood in the context of a statistical approach known as additive modeling (FRIEDMAN; HASTIE; TIBSHIRANI, 2000). This framework can be generally defined as

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \beta_t b(\mathbf{x}, \theta_t)$$

where $\{b(\mathbf{x}, \theta_t)\}_{t=1}^{T}$ are called "basis functions" given that they span a function subspace, θ_t are a set of parameters, $\mathbf{x}$ are input features and $\beta_t \in \mathbb{R}$ is a multiplier. These basis functions are predictors, such as a neural network or a tree, and we use them as base learners - weak learners in the context of boosting - to fit an additive model F_T. One can use a backfitting procedure or a *greedy* forward stagewise approach to solve for an optimal set of parameters. We'll focus on the latter to explain the development of boosting algorithms. Effectively, the Forward Stagewise Additive Modeling algorithm starts with a simple function $f_0(\mathbf{x}) = 0$ and iteratively adds base learners to minimize the risk of $\hat{f}_{t-1}(\mathbf{x}) + \hat{\beta}_t b(\mathbf{x}, \hat{\theta}_t)$.

```
Algorithm 2: Forward Stagewise Additive Modeling
  Input   : Training data $D = \{(x_i, y_i)\}_{i=1}^{n}$; Number of iterations $T$
  Output: Additive model $\hat{f}(x)$
  Initialize $f_0(x) \leftarrow 0$;
  for $t = 1$ to $T$ do
      $\hat{\theta}_t, \hat{\beta}_t \leftarrow \arg\min_{\theta,\beta} \sum_{i=1}^{n} L(f_{t-1}(x_i) + \beta b(x_i, \theta), y_i)$;
      $\hat{f}_t(x) \leftarrow \hat{f}_{t-1}(x) + \hat{\beta}_t b_t(x_i, \hat{\theta}_t)$;
  end
  return $\hat{f}(x) = \hat{f}_T(x)$;
```

At each iteration t, the output value $\hat{f}_t(x)$ is updated based on the previous output value $\hat{f}_{t-1}(x)$ and the solution of the base learner $\hat{\beta}_t b(\mathbf{x}, \hat{\theta}_t)$. It is important to note that once a particular base learner is fitted, it is not changed. Furthermore, considering a useful particular case where the base learner is represented by a tree structure, $F_t(x_i; \hat{\theta}_t)$ would be a tree with regions determined by $\hat{\theta}_t$.

2.7.2 Boosting for Regression

In the context of a regression task, we can use a quadratic loss function $L(\cdot, \cdot)$ and apply the forward stagewise additive modeling algorithm to the optimization problem. At the iteration t we want to minimize:

$$L(y_i, \hat{f}_{t-1}(x_i) + \beta F(x_i; \theta)) = (y_i - \hat{f}_{t-1}(x_i) - \beta F(x_i; \theta))^2 = (r_{it} - \beta F(x_i; \theta))^2$$

where r, the residual $r_{it} = y_i - \hat{f}_{t-1}(x_i)$, is a function of $\mathbf{x}$ and $\mathbf{y}$. Moreover, since the basis function F is scalable, it is sufficient to take $\beta = 1$ and choose $\hat{\theta}_t$ to train the tree $F(x; \theta_t)$ on the residuals $r_{it} = y_i - \hat{f}_{t-1}(x_i)$, $i = 1, \ldots, n$.

Algorithm 3: Boosting for Regression

```
Input: Training data $D = \{(x_i, y_i)\}_{i=1}^n$; Number of iterations $T$; Shrinkage
       parameter $\lambda$
Output: Boosted regression model $\hat{f}(x)$
Initialize $f_0(x) = 0$ and $r_{i0} = y_i$ for all $i$ in the training set;
for $t = 1$ to $T$ do
    Fit a weak learner $\hat{\theta}_t = \text{argmin}_\theta \sum_{i=1}^n (r_{i,t-1} - F(x_i; \theta))^2$;
    Update predictions $\hat{f}_t(x) = \hat{f}_{t-1}(x) + \lambda F(x_i; \hat{\theta}_t)$;
    Update residuals $r_{it} = r_{i,t-1} - \hat{f}_t(x_i)$ for all $i$;
return $\hat{f}(x) = \hat{f}_T(x)$;
```

We note that at each iteration t, the output value $\hat{f}_t(x)$ is updated based on the previous output value $\hat{f}_{t-1}(x)$ and the estimated weak learner $\lambda F(x_i; \hat{\theta}_t)$, where $\lambda \in \mathbb{R}^+$ is a shrinkage parameter that controls the learning rate. The residual r_{it} is updated based on $r_{i,t-1}$ and the output value $\hat{f}_t(x_i)$. Another important parameter to observe is the number of iterations T, which is usually chosen via cross-validation and should also be understood as the number of fitted weak learners. Tuning these parameters could be challenging: if T is too big, the boosting model may overfit, and while a small value of λ might lead it to learn more about the patterns in the data, it would demand a larger value of T to have good performance. An additional and important parameter to consider is the depth of the trees used as weak learners, specified by θ. This factor determines model complexity, and excessive tree depth can lead to overfitting.

2.7.3 AdaBoost

To this point, we've been focusing on the regression problem, where $\mathbf{x}$ and $\mathbf{y}$ have some joint distribution, the response variable $\mathbf{y}$ is quantitative, and we are interested in modeling the conditional mean $\mathbb{E}(\mathbf{y}|\mathbf{x})$. However, it is equally important to consider boosting in the context of a classification task. In this scenario, the squared error loss is generally not a good choice. We'll substitute it with the exponential loss

$$L(\tilde{y}, F(x)) = e^{-\tilde{y}F(x)}$$

where $\tilde{y} \in \{-1, 1\}$ and show that AdaBoost can be understood as a stagewise algorithm for fitting an additive model. This new algorithm fits weak learners to weighted versions of the data instead of fitting them to the residuals.

Denoting the basis function as $F(x;\theta)$ and normalized weights as $\omega_{it} = e^{-\tilde{y_i}\hat{f}_{t-1}(x_i)}$, at the t-th iteration we choose β and θ to minimize

$$\sum_{i=1}^{n} e^{-\tilde{y_i}(\hat{f}_{t-1}(x_i)+\beta F(x_i;\theta))} = \sum_{i=1}^{n} \omega_{it} e^{-\beta\tilde{y_i}F(x_i;\theta)} = e^{-\beta} \sum_{i:\tilde{y_i}=F(x_i,\theta)} \omega_{it} + e^{\beta} \sum_{i:\tilde{y_i}\neq F(x_i,\theta)} \omega_{it}$$

$$= (e^{\beta} - e^{-\beta}) \sum_{i=1}^{n} \omega_{it} \mathbb{1}_{\{\tilde{y_i}\neq F(x_i;\hat{\theta})\}} + e^{-\beta} \sum_{i=1}^{n} \omega_{it}$$

Therefore, we solve for

$$\hat{\theta}_t = \text{argmin}_{\theta} \sum_{i=1}^{n} \omega_{it} \mathbb{1}_{\{\tilde{y_i}\neq F(x_i;\hat{\theta})\}}$$

$$\hat{\beta}_t = \text{argmin}_{\beta} (e^{\beta} - e^{-\beta}) \sum_{i=1}^{n} \omega_{it} \mathbb{1}_{\{\tilde{y_i}\neq F(x_i;\hat{\theta})\}} + e^{-\beta} \sum_{i=1}^{n} \omega_{it}$$

Firstly, we obtain the value $\hat{\theta}_t$ fitting a tree $F_t(x, \hat{\theta})$ to the training data with weights $\{\omega_i\}_{i=1}^{n}$. Then, we calculate $\hat{\beta}_t$ with the expressions

$$err_t = \frac{\sum_{i=1}^{n} \omega_i \mathbb{1}_{[\tilde{y_i}\neq F(x_i;\hat{\theta}_t)]}}{\sum_{i=1}^{n} \omega_i}$$

$$\hat{\beta}_t = \frac{1}{2} \log \left(\frac{1 - err_t}{err_t} \right)$$

Where err_t is a weighted error rate, we defer the proof of the $\hat{\beta}_t$ expression to Appendix D.1. After finding $\hat{\theta}_t$ and $\hat{\beta}_t$, we update $\hat{f}_t(x) = \hat{f}_{t-1}(x) + \hat{\beta}_t F(x_i; \hat{\theta}_t)$ and update the weights so that it is possible to calculate the successive iterations. Finally, this process results in a classifier where the class will be the sign of the trained additive model. This algorithm, which is based on the exponential loss function, is called AdaBoost. An analysis of the algorithm convergence in training is done in Appendix D.2.

Algorithm 4: AdaBoost

```
Input: Training data $D = \{(x_i, y_i)\}_{i=1}^n$; Number of iterations $T$
Output: Boosted model $\hat{f}(x)$
Initialize weights $\omega_i = \frac{1}{n}$ for $i = 1, 2, \dots, n$;
for $t = 1$ to $T$ do
    Fit tree $F_t(x, \hat{\theta})$ to training data with weights $\{\omega_i\}_{i=1}^n$;
    Compute $err_t = \frac{\sum_{i=1}^n \omega_i \mathbb{1}_{[\tilde{y}_i \neq F(x_i;\hat{\theta}_t)]}}{\sum_{i=1}^n \omega_i}$;
    Compute $\beta_t = \frac{1}{2} \log\left(\frac{1-err_t}{err_t}\right)$;
    Update the model: $\hat{f}_t(x) = \hat{f}_{t-1}(x) + \hat{\beta}_t F(x_i; \hat{\theta}_t)$;
    Update weights: $\omega_{i,t+1} = \omega_{it} e^{-\hat{\beta}_t \tilde{y}_i F(x_i;\hat{\theta}_t)}$;
    Renormalize $\omega_{i,t+1}$ such that $\sum_i \omega_{i,t+1} = 1$;
return $\hat{f}(x) = sign\left(\sum_{t=1}^T \beta_t F(x; \hat{\theta}_t)\right)$;
```

2.7.4 Gradient Boosting

Up to this point, we had to derive a new boosting algorithm for each distinct loss function. However, there are ways in which the boosting paradigm can be generalized. Gradient Boosting, for instance, is a generalization of boosting to arbitrary differentiable loss functions by leveraging gradient descent. The main idea of this algorithm is to create new base learners that have a high correlation with the negative gradient of the loss function.

We define a function $\mathbf{f}$ and minimize the loss $\hat{\mathbf{f}} = \text{argmin}_{\mathbf{f}} L(y, \mathbf{f})$ using gradient descent in a function space. Since the function space is infinite-dimensional, for the step t, the gradient g_{it}:

$$g_{it} = \left.\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right|_{f(x)=\hat{f}_{t-1}(x)}$$

is only defined at the data points $\{(x_i, y_i)\}_{i=1}^n$. Therefore, to extend this to other $\mathbf{x}$ values, we limit the search space by a parametric family of functions $F(x, \theta), \theta \in \mathbb{R}^n$, that are highly correlated with $-g_{it}$ and solve for θ

$$\theta_t = \text{argmin}_\theta \sum_{i=1}^n (-g_{it} - F(x_i; \theta))^2$$

to then update

$$\hat{f}_t(x) = \hat{f}_{t-1}(x) + \gamma_t \lambda F(x; \theta_t)$$

where the value γ_t is determined using line search

$$\gamma_t = \text{argmin}_\gamma \sum_{i=1}^n L(y_i, \hat{f}_{t-1}(x) + \gamma F(x; \theta_t))$$

and λ is a shrinkage term.

```
Algorithm 5: Gradient Boosting
Input: Training data $D = \{(x_i, y_i)\}_{i=1}^n$; Number of iterations $T$; Shrinkage
       parameter $\lambda \in [0, 1]$; choice of the loss function $L(y_i, F(x_i))$; choice of the
       basis function $F(x_i, \theta)$;
Output: Boosted model $\hat{f}(x)$
Initialize $f_0(x) = \text{argmin}_F \sum_{i=1}^n L(y_i, F(x_i))$;
for $t = 1$ to $T$ do
    Compute the gradient residual using $g_{it} = \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\Big|_{f(x)=\hat{f}_{t-1}(x)}$;
    Use the weak learner to compute $\hat{\theta}_t = \text{argmin}_\theta \sum_{i=1}^n (-g_{it} - F(x_i; \theta))^2$;
    Determine using line search $\hat{\gamma}_t = \text{argmin}_\gamma \sum_{i=1}^n L(y_i, \hat{f}_{t-1}(x) + \gamma F(x; \theta_t))$;
    Update $\hat{f}_t(x) = \hat{f}_{t-1}(x) + \hat{\gamma}_t \lambda F(x; \hat{\theta}_t)$;
return $\hat{f}(x) = \hat{f}_T(x)$;
```

Hence, we have a least-squares problem, minimizing the squared difference between the negative gradient and the predictions $F(x_i; \theta)$, and a one-dimensional optimization problem with γ_t. The shrinkage term is added to improve the model's generalization ability, controlling the learning rate. Furthermore, as standard practice, decision trees are commonly used as weak learners in this method.

2.7.5 XGBoost

Modern approaches are built upon the gradient boosting algorithm. Extreme Gradient Boosting, known as XGBoost, can be understood as an efficient implementation of Gradient Boosted Decision Trees (GBDT) that includes several improvements. Particularly, it uses Newton Boosting, taking into consideration second-order information and being likely to learn better tree structures. Thus, this method uses a second-order approximation for the loss function that includes the Hessian alongside the gradient. Furthermore, it regularizes the number of leaves and predicted values in each region and samples predictors during the selection of the best split.

To use the Newton Method, we need to calculate the Hessian,

$$h_{it} = \frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\Bigg|_{f(x)=\hat{f}_{t-1}(x)}$$

which is only defined at the data points $\{(x_i, y_i)\}_{i=1}^n$, similar to the gradient boosting algorithm. Therefore, we also need to limit the basis function to a restricted set of functions to extend the empirical Hessian to other $\mathbf{x}$ values, solving for θ (unregularized)

$$\hat{\theta}_t = \text{argmin}_\theta \sum_{i=1}^n \frac{1}{2}\hat{h}_{it}[(-\frac{\hat{g}_{it}}{\hat{h}_{it}}) - F(x_i; \theta)]^2$$

and then updating

$$\hat{f}_t(x) = \hat{f}_{t-1}(x) + \rho F(x_i; \hat{\theta}_t)$$

where ρ is a learning rate.

Algorithm 6: A Generic XGBoost Algorithm

```
Input: Training data $D = \{(x_i, y_i)\}_{i=1}^n$; Number of iterations $T$; choice of the
       differentiable loss function $L(y_i, F(x_i))$; learning rate $\rho$;
Output: Boosted model $\hat{f}(x)$
Initialize $f_0(x) = \text{argmin}_F \sum_{i=1}^n L(y_i, F(x_i))$;
for $t = 1$ to $T$ do
    Compute the gradient $g_{it} = \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\Big|_{f(x)=\hat{f}_{t-1}(x)}$;
    Compute the hessian $h_{it} = \frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2}\Big|_{f(x)=\hat{f}_{t-1}(x)}$;
    Use the weak learner to compute
      $\hat{\theta}_t = \text{argmin}_\theta \sum_{i=1}^n \frac{1}{2}\hat{h}_{it}[(-\frac{\hat{g}_{it}}{\hat{h}_{it}}) - F(x_i; \theta)]^2 + \Omega(F_t) + constant$;
    Update $\hat{f}_t(x) = \hat{f}_{t-1}(x) + \rho F(x_i; \hat{\theta}_t)$;
return $\hat{f} makeshat f_T(x)$;
```

The penalization term can be expressed as

$$\Omega(F_t) = \sum_{t=1}^{T} [\, \gamma T_t + \lambda_1 ||\omega_t||_1 + \frac{1}{2}\lambda_2 ||\omega_t||_2^2 \,]$$

and it is one of the key enhancements of the XGBoost algorithm, applying L_1 and L_2 regularization on leaf weights to reduce model complexity. There is also another regularization parameter γ to penalize the number of terminal nodes in each individual fitted tree.

Another important factor of this model is the use of randomization in the training process. Applying row and column subsampling to make individual trees more dissimilar, reducing their covariance and possibly the ensemble's overall variance. Consequently, it improves the model's generalization performance, making it more robust to overfitting.

2.7.6 LightGBM

Light Gradient Boosting Machine (LightGBM) is another modern boosting approach built upon GBDTs. This algorithm has many similarities with XGBoost, such as using second-order information of the loss function and regularizing to fit trees. However, it has important computational improvements, leading to faster training speed and reduced memory usage.

First, it applies Exclusive Feature Bundling (EFB) to reduce feature dimension. This technique works by noting that, in sparse feature spaces, merging mutually exclusive features into a single feature is possible. First, it searches for mutually exclusive features—features that never have nonzero values at the same time—and then bundles them into a new feature, effectively decreasing the feature dimension.

The implementation of Gradient-based One-Side Sampling (GOSS) is used to choose a subset of the data to train the model and increase training efficiency. This algorithm ranks the training data instances according to the absolute value of their gradient, keeping the samples with larger gradients and applying random sampling on the ones with small gradient values. The key idea is that a data instance with a large gradient value reflects a higher prediction error at iteration t during training. As a result, it is beneficial to prioritize such instances in the training process for iteration $t+1$. Therefore, this technique makes the training procedure focus on the data instances with larger gradients, observing that they have more influence on the information gain. Moreover, it is important to note that GOSS also decreases computational costs by using a smaller subset of the data to estimate the variance gain and determine the split point of the trees.

Another important factor to consider is the use of a histogram-based algorithm, as applied in XGBoost's algorithm, to define the optimal segmentation point when building the trees. This technique groups continuous feature values into discrete bins that represent the range of their values. For each feature, a histogram is built, and when constructing the trees, the histogram's statistics are used to decide the segmentation points. Thus, to make the splitting procedure more efficient without sacrificing much accuracy, the histogram algorithm uses an approximate solution based on quantiles to define the optimal splits. Consequently, this method reduces memory consumption and training time, diminishing computational costs.

Finally, LightGBM uses a leaf-wise tree growth method instead of the traditional level-wise algorithm. The main difference between these strategies is the order in which the tree is built: the leaf-wise algorithm uses a best-first approach, while the level-wise uses a depth-first approach. Using the leaf-wise growth, the split with the greatest information gain is chosen independent of the depth level, while the level-wise growth splits all the nodes in the same depth before moving to the next level. Therefore, by choosing the best-first approach, LightGBM can reduce training time and often improve the prediction performance.

2.7.7 CatBoost

CatBoost is a gradient-boosted decision tree method designed to deal effectively with categorical features and mitigate prediction shifts. To address the target leakage issue, the Yandex researchers introduced ordered boosting. This technique generates a new training dataset at each boosting step, preventing the previous model from seeing the labels in the new dataset. This helps reduce bias in the model.

CatBoost employs one-hot encoding for categorical features with low cardinality. For handling high cardinality categorical variables, researchers introduced the "Ordered Target Statistics" technique. Additionally, CatBoost utilizes an oblivious tree growth approach, as described in (KOHAVI, 1994), where a uniform splitting criterion is consistently applied across the entire level of the tree.

CatBoost's algorithm is formally defined in Prokhorenkova et al. (2018) supplementary materials, specifically in section B, as follows:

```
Algorithm 7: CatBoost
  Input: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $I$, $\alpha$, $L$, $s$, Mode
  $\sigma_r$ = random permutation of $[1, n]$ for $r = 0..s$;
  $M_0(i) \leftarrow 0$ for $i = 1..n$;
  if $Mode = Plain$ then
      $M_r(i) \leftarrow 0$ for $r = 1..s$, $i : \sigma_r(i) \leq 2^{j+1}$;
  if $Mode = Ordered$ then
      for $j \leftarrow 1$ to $[\log_2 n]$ do
          $M_{r,j}(i) \leftarrow 0$ for $r = 1..s$, $i = 1..2^{j+1}$;
  for $t = 1$ to $I$ do
      $T_t, \{M_r\}_{r=1}^s \leftarrow$ BuildTree($\{M_r\}_{r=1}^s$, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\alpha$, $L$, $\{\sigma_i\}_{i=1}^s$, Mode);
      $leaf_0(i) \leftarrow$ GetLeaf($\mathbf{x}_i$, $T_t$, $\sigma_0$) for $i = 1..n$;
      $grad_0 \leftarrow$ CalcGradient($L$, $M_0$, $y$);
      foreach $leaf\ j$ in $T_t$ do
          $b_j^t \leftarrow -\text{avg}(grad_0(i)$ for $i : leaf_0(i) = j)$;
      $M_0(i) \leftarrow M_0(i) + \alpha b_{\text{leaf}_0(i)}^t$ for $i = 1..n$;
  return $F(x) = \sum_{t=1}^I \sum_j \alpha b_j^t \mathbb{1}_{\{GetLeaf(\mathbf{x}, T_t, ApplyMode)=j\}}$;
```

In this algorithm, there are many notational differences from the previous ones explained in this thesis. Thus, to clarify the notation in 7: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is the training data, I is the number of iterations, T_t is a decision tree at the t-th iteration, b_j are leaf values, α is the learning rate, L is the loss function, s represents the number of permutations generated (it determines the number of different orderings of the training dataset used during training), σ_r represents a random permutation of the indices of the training dataset, M represents the supporting models (specifically, $M_{r,j}(i)$ is the current prediction for the i-th example based on the first j examples in the permutation σ_r).

The algorithm first generates $s + 1$ independent random permutations of the training data and initializes supporting models for each permutation. Then, if using the “Plain” or “Ordered” mode, set up supporting models based on certain conditions. Afterward, for each boosting iteration, the algorithm builds a decision tree based on the current supporting models and training data, assigns training examples to tree leaves, calculates gradients based on the assigned leaves, and updates the supporting models using the calculated gradients. Ultimately, it returns the final boosted model.

2.8 BOOSTING METHOD COMPARISON

In this section, we aim to evaluate the algorithmic differences between distinct boosting models: Gradient Boosting, AdaBoost (Adaptive Boosting), XGBoost (Extreme Gradient Boosting Model), LightGBM (Light Gradient Boosting Model), and CatBoost (Categorical Boosting). At this stage, we intend to compare the methods strictly from a theoretical perspective, noting that the experiment results will be discussed in detail later in this study. Hence, it is important to emphasize the differences and similarities between these algorithms to enrich the comparative analysis of the thesis findings.

The essence of **AdaBoost** (FREUND; SCHAPIRE, 1995), along with other boosting methods, involves training predictors sequentially, with each attempting to correct the errors of its predecessor. This method pioneered adaptive boosting algorithms since it adapts its parameters according to the actual method performance at the current iteration t. AdaBoost's generalization problem, as thoroughly discussed in (MAYR et al., 2014), is widely recognized. In this context, there is a consensus that, despite the potential for the algorithm to overfit, it demonstrates resistance in practice. A parameter is used to stop the algorithm at predefined iteration t_{stop}, controlling model complexity. However, stopping the algorithm early, yielding a small number of iterations, may lead to underfitting. Another important factor, as indicated by (CHENGSHENG; HUACHENG; BING, 2017) and (WANG; SUN, 2021), is that AdaBoost is a fast and easily programmed boosting method but performs poorly with imbalanced data. In this context, modern applications of this algorithm involve modifications, such as combining it with other statistical techniques or adjusting the weights computation, increasing its robustness.

Gradient Boosting (FRIEDMAN, 2001) is a generalization of the boosting paradigm to arbitrarily differentiable loss functions, employing a training process through numerical optimization, where the objective is to minimize the model's loss using a gradient descent procedure. Different from AdaBoost, this approach does not involve updating the weights of the samples but fitting weak learners that are highly correlated with the negative gradient of the loss function evaluated at the $t-1$ iteration. Moreover, the model has greater flexibility than AdaBoost and more regularization options, such as the adjustable learning rate. Therefore, the Gradient Boosting Machine (GBM), introduced by Friedman, inspired many modern and high-performing boosting methods. These methods, such as XGBoost, LightGBM, and CatBoost, stem from the fundamental concept of gradient boosting, yet they diverge in their specific modeling details.

XGBoost (CHEN; GUESTRIN, 2016) follows the same principle as Gradient Boosting but with a more advanced implementation, focusing specifically on decision trees as base learners. One of the most impactful additions of this method is the use of second-order information, computing the Hessian of the loss function with respect to the predictions, often improving the predictive performance. Furthermore, XGBoost employs various regularization strategies, such as L_1 and L_2 shrinkage and individual terminal node penalization by the parameter γ, enhancing its generalization capability. As the study (BENTÉJAC; CSÖRGŐ; MARTÍNEZ-MUÑOZ, 2021) discusses, this algorithm focuses on reducing the computational complexity of finding the best tree split using a compressed column-based structure where data is pre-sorted, allowing for parallel processing and reducing the need for repeated sorting. Moreover, XGBoost includes column subsampling to make training more efficient and reduce overfitting. Consequently, XGBoost includes many optimization techniques and often exhibits superior performance compared to older implementations of boosting, both in explaining the output variable and in computational efficiency.

Light Gradient Boosting (KE et al., 2017) is an optimized implementation of Gradient Boosting developed by Microsoft, known for its computational efficiency. The algorithm also incorporates the regularization features and the use of second-order information of the loss function from XGBoost. However, it is modeled to have higher computational performance and scalability, both explained by Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), techniques used to reduce feature dimension and the amount of data used to estimate the tree split. Furthermore, LightGBM uses a leaf-wise growth approach when building trees, while XGBoost uses a level-wise method (nowadays, XGBoost can also be implemented with the leaf-wise strategy for the histogram-based method). The first approach might lead to trees with better performance in larger datasets since it is more flexible but might be prone to overfitting. The second approach keeps the trees balanced but splits nodes with small information gain, scarcely affecting the results and wasting resources (ALSHARI; SALEH; ODABAS, 2021). Figures 2.3 and 2.4 illustrate the process of level-wise and leaf-wise growth.

Figure 2.3 – *Level-wise growth*

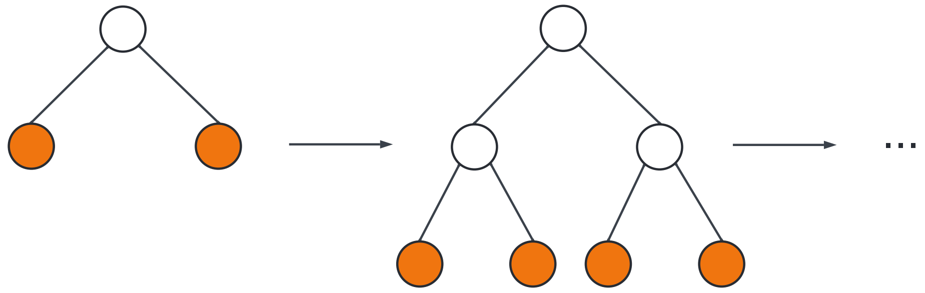
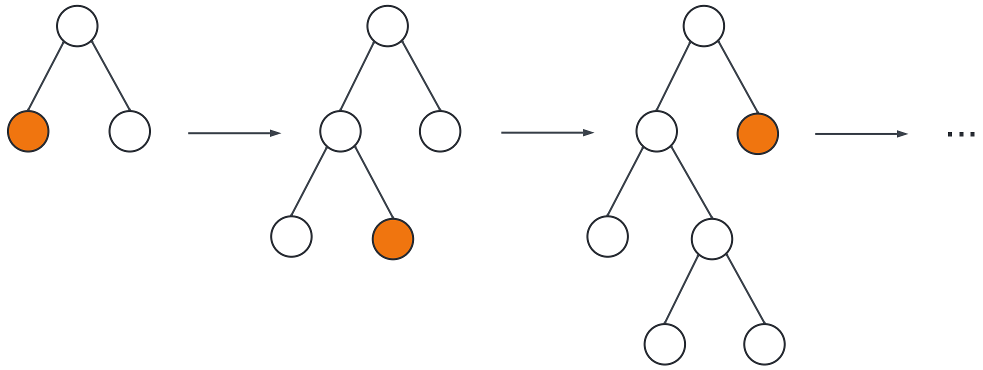
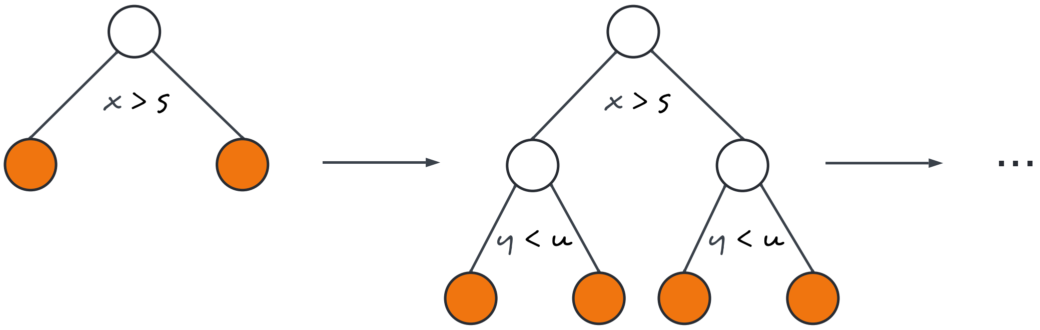


Figure 2.4 – *Leaf-wise growth*



CatBoost grows a balanced tree that is less prone to overfitting and speeds up execution at testing time. It uses consistent splitting criteria across an entire level of the tree, employing the same features to split learning examples into right and left branches. Figure 2.5 demonstrates the process of oblivious tree growth, where "x" and "y" represent input features, and "s" and "u" threshold values. For instance, in the first level, it evaluates "$x > s$" to split the learning examples, and in the second level, it evaluates "$y < u$".

Figure 2.5 – *Oblivious tree growth*



2.9 RISK MEASURES

2.9.1 Downside Risk Measure: Conditional Value at Risk

Value at Risk (VaR) is a widely used risk measure in the financial industry. VaR quantifies the maximum potential loss in the value of an asset or portfolio over a specified time period, given a certain confidence level $\alpha \in]0,1[$.

Formally, given a random variable R representing the distribution of returns over the desired time horizon, VaR is defined as the α-quantile (q_α) of R:

$$\text{VaR}_\alpha(R) = \inf\{r : Prob(R \leq r) > \alpha\} = q_\alpha(R)$$

VaR only considers the specific quantile of interest, disregarding any information beyond that point. Conditional Value at Risk (CVaR) addresses this limitation. While the VaR informs the loss threshold, the CVaR presents the severity of tail losses.

CVaR is an absolute measure of downside risk and has superior mathematical properties compared to VaR (SARYKALIN; SERRAINO; URYASEV, 2008). It calculates the average losses that exceed VaR and is defined as:

$$\mathrm{CVaR}_\alpha(R) = \mathbb{E}[R \mid R < \mathrm{VaR}_\alpha = q_\alpha(R)]$$

In the empirical portion of this thesis, CVaR will be the relevant downside risk measure since it is a more sensitive and informative measure of tail risk.

2.9.2 Dispersion Risk Measure: Volatility

Volatility is a commonly used dispersion risk measure that quantifies the degree of variation in the returns of an asset or portfolio over time. It is typically measured by the standard deviation of returns, which captures the extent to which returns deviate from their mean.

Formally, given a series of returns R_t for $t = 1, 2, \ldots, n$, the volatility (standard deviation) $s(R)$ over a rolling window of size w is defined as:

$$s_t(R) = \sqrt{\frac{1}{w-1} \sum_{i=t-w+1}^{t} (R_i - \mathrm{SMA}_t(R))^2}$$

where $\mathrm{SMA}_t(R)$ is the average of the returns over the rolling window w, given by:

$$\mathrm{SMA}_t(R) = \frac{1}{w} \sum_{i=t-w+1}^{t} R_i$$

Higher volatility indicates greater dispersion of returns and, hence, higher risk. In financial markets, volatility is often used to gauge the stability and predictability of asset prices.

Although there are several types of volatility, including implied volatility, realized volatility, and expected volatility, only historical volatility was considered in this study. The term "volatility" will henceforth be used interchangeably with "standard deviation of returns".

2.9.3 Drawdown Risk Measure: Maximum Drawdown

Maximum Drawdown (MDD) measures the maximum observed loss from a peak (highest price P_k) to a trough (lowest price P_j) before a new peak is attained. Therefore, it is represented as a percentage decrease from the highest value and is path-dependent. This risk measure provides a picture of potential maximum losses.

For a more formal definition, consider P_t as the price of an asset at time t and consider $0 < t \leq T$. The maximum drawdown over the period T is defined as:

$$MDD = \min_{0 \leq k \leq j \leq T} \left(\frac{P_j - P_k}{P_k} \right)$$

Max Drawdown is an important measure because it highlights the worst-case scenario for an investor by showing the most significant percentage drop from a peak during a specified period. For that reason, it will be considered the drawdown risk measure in the empirical part of the study.

3 LITERATURE REVIEW

This chapter uses a dual narrative approach. First, it centers on an examination of seminal papers that have contributed to the evolution of boosting methods. In this context, the review explores foundational works that set the theoretical groundwork for boosting algorithms, highlighting advancements and methodologies proposed by researchers. Second, the chapter transitions to an investigation into applying boosting techniques in the context of financial risk prediction. This segment explores relevant studies and research findings that showcase the adaptability and effectiveness of boosting algorithms in forecasting financial risks. Therefore, exploring these two perspectives, the literature review aims to offer a holistic understanding of the historical development of boosting methods and their contemporary applications in financial risk prediction.

3.1 HISTORICAL OVERVIEW

The Probably Approximately Correct (PAC) machine learning framework underlies the development of boosting theory. The term “probably” denotes the probabilistic nature of the learning process, and “approximately correct” indicates a low error rate. The paper by Valiant (1984) focuses on exploring knowledge acquisition in the absence of explicit programming, providing insights into the design of learning machines. From a computational viewpoint, the author addresses the question of solvability, introducing a framework that quantitatively defines the learnability of a problem. This framework incorporates key parameters, such as the concept class - a collection of concepts that the algorithm aims to learn; the hypothesis space - a set of hypotheses that the algorithm can consider solutions; and the learning algorithm - a procedure that selects a hypothesis from the hypothesis space based on the training examples. A problem is deemed PAC learnable when a learning algorithm can discover a solution with an error rate below a specified threshold, doing it with high probability. Finally, an algorithm meeting these criteria is classified as a strong learner.

The study by Kearns and Valiant (1989) focuses on proving the intractability of learning several classes of Boolean functions - functions that take a set of binary inputs and produce a single binary output - in the PAC model. This paper’s profound impact in boosting is that it proposes a definition for weak learners: algorithms that exhibit only marginal improvement over random guessing.

Subsequently, in 1990, Schapire’s paper (1990) demonstrated the potential of these weak learners. The key insight was that if a strong learner is capable of solving a problem, then a collective assembly of weak learners can achieve the same. This was underscored by introducing a technique known as the ‘hypothesis boosting mechanism.’ In essence, the mechanism employs filtering to modify the distribution of examples, directing the weak learning algorithm's attention to the more challenging regions of the distribution.

More specifically, Schapire employs an algorithm A that generates hypotheses closely approximating the target concept c - the underlying pattern the learning algorithm tries to discover from the data - with an error rate of α. He proposes an enhanced algorithm A' that simulates A under three distinct distributions, yielding hypotheses that significantly better represent c. Thus, an ensemble hypothesis from three weak sub-hypotheses is created, each trained on a distinct distribution. It was mathematically demonstrated that if the three weak sub-hypotheses have an error rate of $\alpha < 1/2$ concerning the distribution they were trained on, then the resulting ensemble hypothesis at each subsequent iteration will have an error rate of $3\alpha^2 - 2\alpha^3$, which is remarkably lower than α since $0 < \alpha < 1/2$. This investigation, therefore, resulted in a technique that converts any learning algorithm that performs just slightly better than random guessing into one that performs with arbitrarily small error rates. That proved the equivalence of the weak and strong notions of learnability.

The first experiments with the early boosting algorithms were done on the OCR (Optical Character Recognition) task in the paper by Drucker, Schapire, and Simard (1993). They used neural networks as base learners and four different handwritten databases consisting of 12,000 digits extracted from segmented ZIP codes provided by the United States Postal Service (USPS) and 220,000 digits, 45,000 uppercase alphas, and 45,000 lowercase alphas obtained from the National Institute of Standards and Testing (NIST). The effects of boosting on the four databases using different architectures showed that the ensemble of networks improved the performance significantly over the single network.

The introduction of the adaptive boosting algorithm, known as AdaBoost, as presented in the paper by Freund and Schapire (1995), is one of the most influential contributions within the scope of boosting algorithms. This new approach to boosting solved various difficulties of its predecessors (FREUND; SCHAPIRE, 1999). AdaBoost assigns a weight to each weak hypothesis during the creation of the final decision. Similarly, during the formulation of a weak hypothesis, each sample is assigned a weight. Because the algorithm focuses its weights on the harder-to-learn samples, the outliers frequently are the examples with the highest weight. Hence, AdaBoost can identify outliers and has a big advantage in terms of the generalization capacity.

Additionally, the algorithm does not need prior knowledge about the weak learner, making it flexible to pair with any method for discovering weak hypotheses. AdaBoost brings along some theoretical guarantees as long as there is enough data and a weak learner that can reliably provide moderately accurate weak hypotheses. Consequently, there is a shift in the process of the learning system design, from making a machine learning algorithm that is accurate over the entire domain to finding weak learning algorithms that only need to be better than random. Over the years, AdaBoost has undergone substantial evolution, mainly noted in the development of several variants, such as the Logit Boost, Emphasis Boost, Reweight Boost, and many others. A more in-depth analysis is done in the paper by Ferreira and Figueiredo (2012).

Gradient Boosting is introduced in the paper by Friedman (2001), where the boosting paradigm is extended to any defined and differentiable loss function. Thus, this work generalizes boosting algorithms, presenting the Gradient Boosting Machine (GBM): a forward stagewise additive modeling algorithm that minimizes differentiable loss functions using steepest-descent steps constrained by the negative gradient direction. In this sense, gradient boosting fits weak learners to pseudo-residuals, which are derived from the negative gradient iteratively.

Effectively, the paper begins by discussing an approach to function estimation focusing on numerical optimization within a function space. This approach links stagewise additive expansions and steepest-descent minimization, culminating in the generalization of the boosting paradigm. Additionally, the paper presents and develops a gradient-boosting strategy for different popular loss functions, such as the Least-Squares, Least Absolute Deviation, Huber-M, and Multiclass Logistic Likelihood Loss Functions. Furthermore, Friedman develops enhancements when the individual additive components are regression trees, named “TreeBoost” models, along with tools for interpreting such models. Finally, the study compares the GBM to existing models, such as AdaBoost and LogitBoost, and connects this new approach to existing ones, emphasizing its importance in the statistical learning field.

Chen and Guestrin (2016) developed XGBoost, a tree-boosting technique renowned for its effectiveness. This method is a highly scalable gradient boosting approach, implementing several techniques to reduce computational complexity. In this sense, XGBoost improves the split-finding process, which is the most time-consuming portion when building decision tree algorithms. This enhancement is accomplished by storing data in a compressed column-based structure, eliminating the need for repeated sorting and enabling parallel processing. Moreover, this innovative approach incorporates both a method based on percentiles to test only a subset of candidate splits and a sparsity-aware algorithm to remove missing values. Thus, XGBoost implements various techniques to reduce training time, focusing on improving computational efficiency.

In addition, XGBoost has other innovative modeling strategies focused on improving the method’s predictive performance. In particular, it uses supplementary regularization in the learning objective, specifically L_1 and L_2 shrinkage, along with a parameter to control the number of terminal nodes. These strategies collectively assist in regulating model complexity and mitigate overfitting. Another important implementation is the use of the hessian of the loss function to calculate the optimal split and leaf weights, often resulting in better tree structures. Furthermore, the model uses randomization techniques, such as column subsampling at tree and node levels, to improve generalization and training speed. A detailed review of XGBoost's algorithm is performed by Nielsen in (2016), covering fundamental concepts, three boosting models, their historical development, and algorithm comparisons.

Ke et al. (2017) developed LightGBM, a gradient-boosted decision tree method designed to deal with large datasets efficiently. This method incorporates two key techniques: Gradient-based One-Side Sampling and Exclusive Feature Bundling. GOSS excludes data instances with small gradients from the tree-building process, and EFB bundles mutually exclusive features, reducing feature dimension. Another relevant implemented technique is the Leaf-wise approach when building decision trees, which might lead to better learning performance. In this sense, we observe that LightGBM incorporates multiple techniques to handle computational complexity, thereby increasing efficiency.

Prokhorenkova et al. (2018) created CatBoost, representing "Categorical Boosting", which is a gradient boosted decision tree approach that effectively processes categorical features and implements ordered boosting - a modification of the standard gradient boosting algorithm. These techniques aim to mitigate prediction shifts, a special type of target leakage that introduces bias and hinders the generalization ability of the trained model. This problem arises from discrepancies between the distribution of the estimated model and the testing samples. To successfully overcome this issue, the Yandex researchers developed ordered boosting, a technique where a new training dataset is generated at each boosting step, ensuring the previous model hasn't been exposed to the labels in the new dataset, consequently decreasing bias.

CatBoost's paper addresses a statistical issue that was not previously formally defined in other methods, introduces a new solution to mitigate it, and develops an empirical study comparing CatBoost with popular new methods, such as XGBoost and LightGBM. In the comparative analysis, CatBoost outperforms the other models on all the considered datasets. However, it is important to observe that the comparison was mainly conducted in heterogeneous and categorical datasets, which introduces limitations to the analysis and may lead to potential biases.

3.2 IMPLEMENTATIONS IN FINANCE

Lao et al. (2021) intended to predict the monthly financing risk of grid companies using the XGBoost model. The study builds a financing risk indicator system considering inputs like cash flow, capital operation ability, profitability, development ability, solvency, financing scale, and economic environment. The paper made a comparative analysis between the XGBoost model, the Support Vector Regressor model, and back-propagation neural network models. The results show the XGBoost model has better prediction accuracy and stability, and MAE and RMSE values for the boosting algorithm were significantly smaller. The stability of XGBoost is asserted by analyzing the decrease in the number of samples in the training set and the increase in the test set, resulting in a slight decrease in MAE and RMSE.

So (2023) compares multiple boosting models and has fundamental importance for this present thesis. The comparative study utilizes modern boosting approaches such as XGBoost, LightGBM, and CatBoost gradient boosting libraries for constructing predictive models. It is important to note that the relative analysis is not limited to evaluating the predictive results. Still, the paper elaborates deeply on the key theoretical differences between the methods, expanding on the methods of tree splitting and handling of categorical data.

These boosting algorithms are used for building claim frequency models from zero-inflated insurance claim data. The examinations are based on two datasets: the French Motor Third-Party Liability (MTPL) dataset and a synthetic telematics dataset. The paper then introduces two scenarios for training zero-inflated Poisson boosted tree models: one where the inflation probability p is a function of the distribution mean μ, and one where p and μ are unrelated. Models are finally applied and evaluated on these two auto insurance claim datasets. To better understand the contribution of each feature, the author uses SHAP values on the prediction of the claim frequency. CatBoost is found to have the best performance, and zero-inflated Poisson models outperform others depending on data characteristics.

Nabipour et al. (NABIPOUR et al., 2020) focuses on reducing the risk of stock market trend prediction using machine learning and deep learning algorithms. In this context, the study analyses four stock market groups from the Tehran stock exchange for experimental evaluations: diversified financials, petroleum, non-metallic minerals, and basic metals. It finally compares the performance of nine machine learning models and two deep learning methods: Decision Tree, Random Forest, Adaptive Boosting, eXtreme Gradient Boosting, Support Vector Classifier (SVC), Naïve Bayes, K-Nearest Neighbors (KNN), Logistic Regression, Artificial Neural Network (ANN), Recurrent Neural Network (RNN) and Long short-term memory (LSTM).

The study used ten technical indicators from ten years of historical data as input values. The data is used and analyzed in two manners: continuous and binary. The former is based on the actual time series, normalized in the range of $(0, +1)$, and the latter goes through a preprocessing step for the conversion to binary, defined by $+1$ as the sign of upward trend and -1 as the sign of downward trend. Results show that RNN and LSTM outperform other prediction models in continuous and binary data evaluation. The XGboost and AdaBoost methods are well-performant but not as much as the LSTM and RNN.

The paper by Chen, Chen, and Cai (2023) builds a stock market risk early warning system for China under the background of the Stock Connect programs. It uses Value at Risk to classify stock market risk into multiple categories - a widely used risk measure that refers to the maximum potential loss that an asset or portfolio may experience over a specified period under market volatility and a certain confidence level. The study divides stock market risk into four levels: high risk, medium risk, low risk, and lowest risk, based on the VaR values at different quantiles (0.1, 0.3, 0.5).

A set of technical, macroeconomic, and other market indicators are used to build prediction models. These models were Long short-term memory, gate recurrent unit, multilayer perception, and XGBoost. The results show that macroeconomic and other basic indicators have an important influence on predicting China's stock market risk, and the performance of the early warning system improves when the conventional and other indicators are considered. Analyzing the performance metrics for the models across three periods, we can validate that the XGBoost model consistently outperformed the LSTM, GRU, and MLP models concerning accuracy. Additionally, the boosting method performs best for precision, recall, and f1-score in multiple periods.

Qin (2022) develops a corporate financial management risk assessment model based on the XGBoost algorithm. The paper uses data from profit and loss, bank loans, employee performance, e-commerce profit and loss, and cross-border business data. The results show that the XGBoost model can accurately classify financial data with errors within 3%, with a maximum error of only 2.48%, and predict risk trends over time well. Qin concludes that the XGboost approach has obvious advantages over traditional financial methods. Moreover, the visual comparison between predicted and actual values over time can be seen in a boxplot and a pie chart. Analyzing the boxplot, it is possible to assert that the predicted value of the risk level over time conforms with the actual risk level value.

Branco, Rubesam, and Zevallos (2024) work offers an evaluation of various forecasting models for volatility, including linear, nonlinear, and machine learning approaches. The study encompasses the realized volatility (RV) of ten global stock market indices from January 2000 to December 2021. The primary objective is to ascertain whether nonlinear machine learning models can outperform traditional linear models in forecasting RV. Boosting models generally performed better than some other nonlinear models like Bagging and Random Forest but were outperformed by simpler models like HARX and linear models with regularization (Ridge, Lasso, ENET) in several cases.

4 METHODOLOGY

This chapter presents the methodological framework adopted in this study, detailing the processes involved in data collection, preprocessing, feature selection, and model evaluation. The structure of this chapter is as follows: the empirical design (4.1), tools and implementation (4.2), descriptive statistics (4.3.1), data preprocessing (4.3.2), feature selection and datasets (4.3.3), model training and validation (4.4.1), hyperparameter tuning (4.4.2), and evaluation metrics (4.5).

4.1 EXPERIMENTAL DESIGN

The primary objective of this study is to evaluate the performance of five different boosting models—Adaboost, Gradient Boosting, XGBoost, LightGBM, and CatBoost—in predicting risk measures.

Seven distinct datasets, each containing 3220 data points, were used to evaluate the predictive performance for three target variables: the 10-day Conditional Value at Risk (CVaR), the 10-day Standard Deviation of Returns (volatility), and the 10-day Maximum Drawdown. The first dataset has 80 feature variables, with six additional datasets created using specific feature selection methods.

For the analysis of the risk measures, the return series of the Bovespa index is used, calculated using the following expression:

$$R_t = \ln\left(\frac{IBV_t}{IBV_{t-1}}\right)$$

where R_t is the return series of the Bovespa index and IBV denotes the closing price series of the index.

Using this return series, the following risk measures are derived:

- **10-day Standard Deviation of Returns (Volatility)**: Volatility is calculated by taking the standard deviation of the log returns over a rolling window of 10 days.

- **10-day Conditional Value at Risk (CVaR)**: CVaR is calculated based on the Value at Risk (VaR). For a given window of 10 days, the VaR is first determined at a **1%** significance level. CVaR is then calculated as the average of the log returns below this VaR threshold.

- **10-day Maximum Drawdown (MDD)**: Maximum Drawdown measures the largest peak-to-trough decline in the index value over a rolling window of 10 days.

For the experiment pipeline, this sequence was followed:

1. Build the seven datasets used in the experiment;
2. Calculate the descriptive statistics for the variables used to derive the risk measures;
3. Split the datasets into train and test sets;
4. Define the search space for hyperparameter tuning;
5. Train the boosting models on the training datasets;
6. Validate the models using cross-validation;
7. Evaluate the models using the test datasets;
8. Analyze the performance metrics to assess the models' performance.

4.2 TOOLS

Python 3.11.4 was employed for the implementation. The code is available on Github. In addition to commonly used data science tools such as *numpy*, *scipy*, *pandas*, *matplotlib*, and *seaborn*, the most important packages utilized are:

- **yfinance** was used to download historical market data from Yahoo Finance. It facilitated the retrieval of the Bovespa index, other international stock market indices, exchange rates, and commodity prices.
- **ipeadatapy** was used to download the EMBI+ series.
- **bcb** was used to download Brazilian macroeconomic variables.
- **statsmodels** was mainly used for data exploratory analysis.
- **sklearn** provided a wide range of tools for modeling. Tools for feature selection and model evaluation were used. The library was also utilized to implement the Adaboost and Gradient Boosting algorithms. Furthermore, **sklearn** facilitated the use of time series cross-validation with *TimeSeriesSplit* and hyperparameter tuning with *RandomizedSearchCV*.
- **xgboost** was used for implementing the XGBoost algorithm.
- **lightgbm** was used for implementing the LightGBM algorithm.
- **catboost** was used for implementing the CatBoost algorithm.

Table 22 provides a complete list of utilized Python packages and their versions.

The experiments were executed with the following settings: Notebook GalaxyBook2; Processor: 12th generation Intel Core i3; 8GB RAM; Storage Type: SSD.

4.3 DATA

4.3.1 Descriptive Statistics

Descriptive statistics provide a foundational understanding of the IBOV and risk data, summarizing central tendency, dispersion, and distribution characteristics. These statistics are essential for identifying patterns, checking data quality, and providing context for further analysis.

Table 1 summarizes key statistical measures for the Bovespa index closing price, including the number of days, mean, standard deviation, minimum, and maximum values.

Table 1 – *Descriptive statistics of the IBOV close price*

Statistic	**Value**
Number of Days	3220
Mean	70967.32
Standard Deviation	23907.92
Minimum	29435.00
Maximum	130776.00

The wide range between the minimum and maximum values suggests considerable fluctuations in the index's closing prices over the observed period (3220 days). The high standard deviation further confirms the substantial variability in the Brazilian stock market, as observed in Araújo et al. (2021).

The descriptive statistics of the Bovespa index logarithmic return series are summarized in Table 2. This table provides an overview of the central tendency, dispersion, and distributional characteristics, along with results from several statistical tests. These tests assess the normality of the return series, the presence of serial autocorrelation, volatility clustering, and the presence of a unit root to determine if the return series is stationary.

Table 2 – *Descriptive statistics of the IBOV return series*

Statistic	**Value**
Mean	0.00006
Standard Deviation	0.01740
Minimum	-0.15993
Maximum	0.13022
Kurtosis	9.97351
Skewness	-0.65099
Shapiro-Wilk	0.91774***
Ljung-Box	59.64552***
ARCH	878.33866***
ADF	-20.82591***

*The Shapiro-Wilk test assesses the normality of the data. The Ljung-Box Q test, applied with 20 lags, evaluates the presence of serial autocorrelation. Engle's ARCH test checks for conditional heteroskedasticity. The ADF test examines the presence of a unit root. The symbol *** denotes statistical significance at the 1% level.*

The Shapiro-Wilk test results in a statistic of 0.91774, rejecting the null hypothesis of normality at the 1% significance level. Therefore, this indicates that the IBOV returns do not follow a normal distribution. The skewness (-0.65099) and high kurtosis (9.97351) also indicate the result of non-normality, suggesting a distribution with heavy tails and more frequent extreme returns.

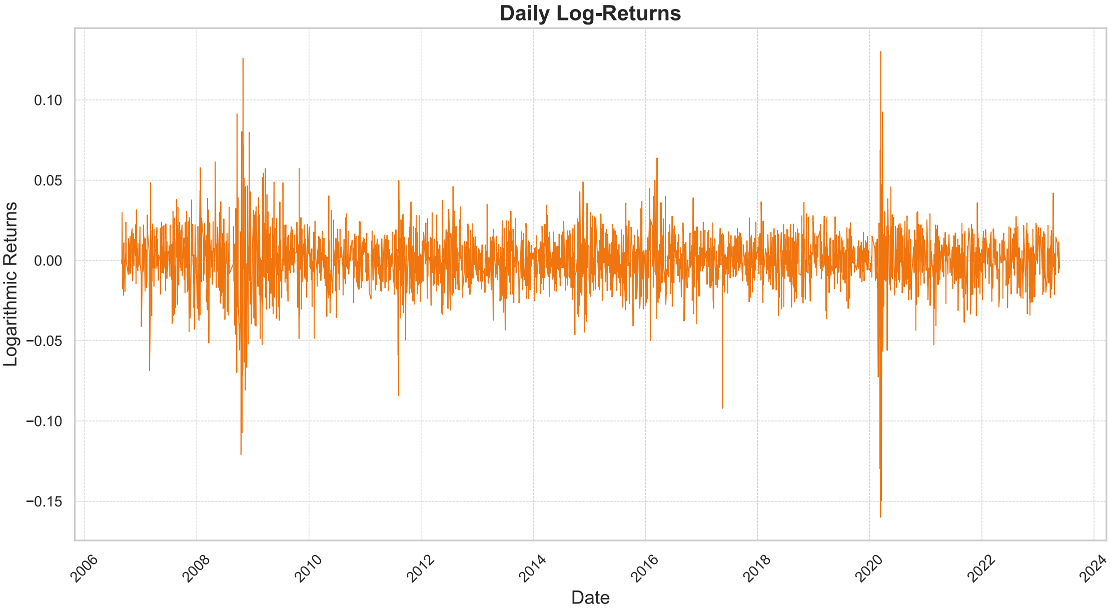
The test results in a statistic of 59.64552 for the Ljung-Box Q test, which is significant at the 1% level. This indicates that there are statistically significant serial autocorrelations in the IBOV returns.

The Lagrange Multiplier (ARCH) test yields a statistic of 878.33866, significant at the 1% level. This confirms the presence of conditional heteroskedasticity, indicating that periods of high volatility tend to be followed by periods of high volatility and vice versa.

Lastly, the ADF test results in a statistic of -20.82591, which is highly significant at the 1% level. This result rejects the null hypothesis of a unit root, indicating that the index return series is stationary.

Figure 4.1 illustrates the daily logarithmic returns of the IBOV from August 2006 to May 2023. This graph provides insights into the behavior and characteristics of the returns, which are consistent with the results of the descriptive statistics.

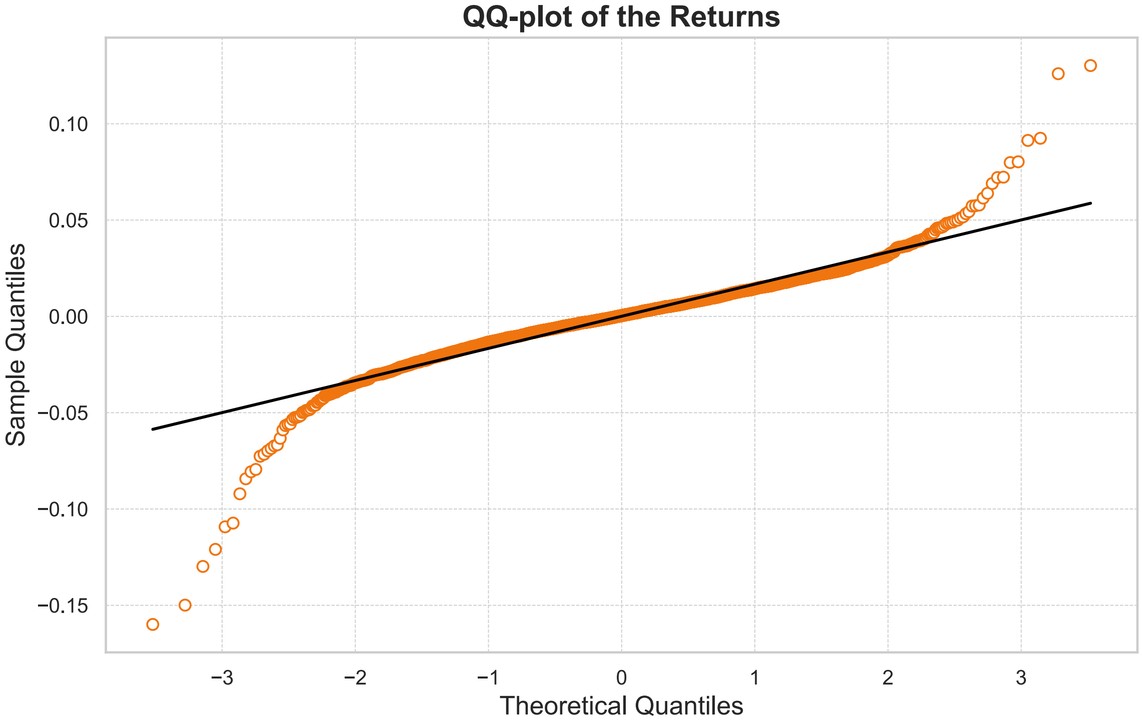
Figure 4.1 – *Daily returns of the IBOV*



Volatility clustering can be observed, and this is consistent with the significant result from the ARCH test, which indicated the presence of conditional heteroskedasticity. There has been a noticeable spike in volatility between 2008 and 2020, corresponding to the global financial crisis and the COVID-19 pandemic. During this period, the returns show extreme fluctuations, with uncommon negative and positive daily returns.

The QQ-plot in Figure 4.2 visually assesses how well the returns of the IBOV follow a normal distribution.

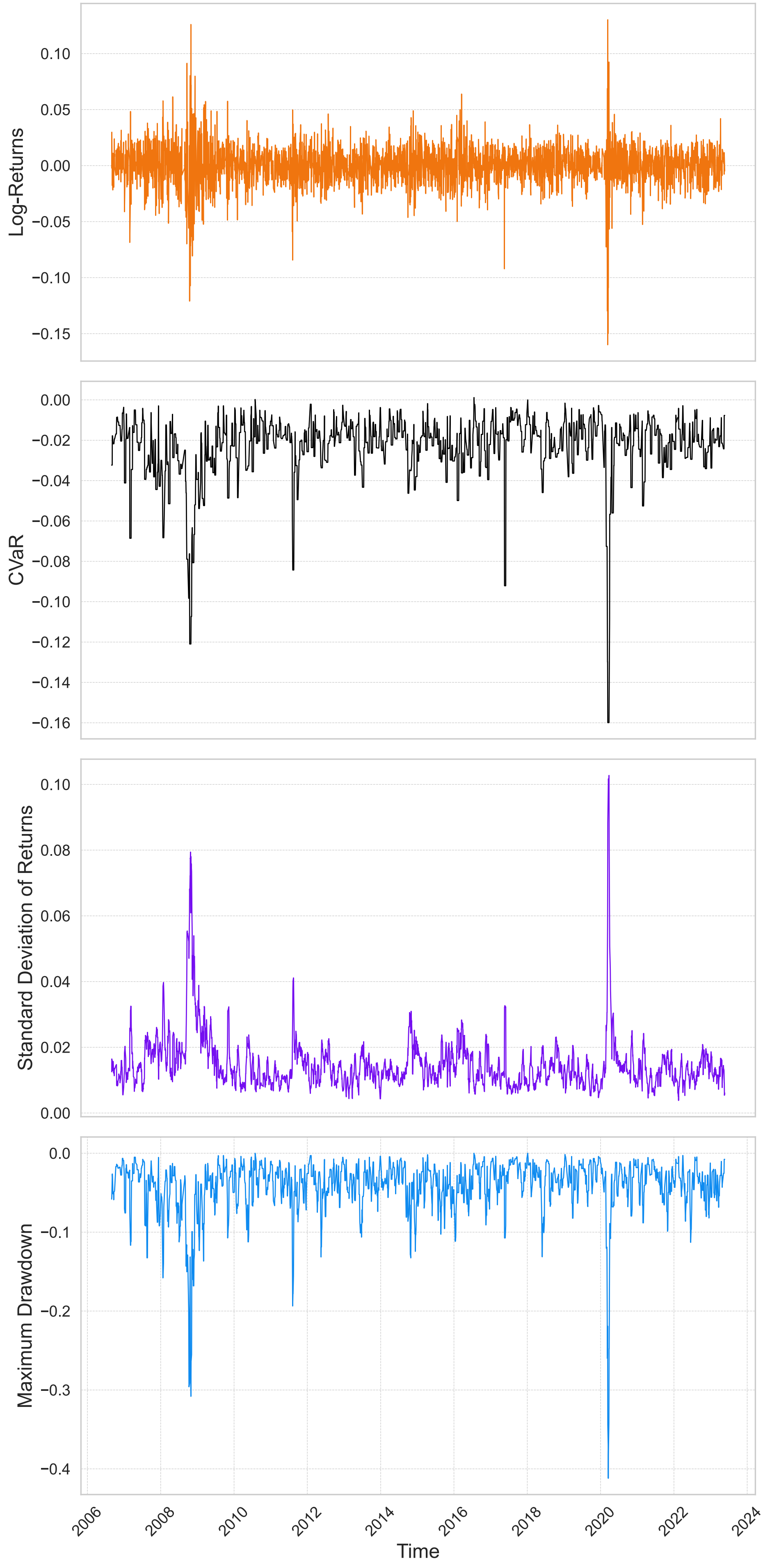
Figure 4.2 – *QQ-plot of the Logarithmic Returns*



In the central region of the plot, the points lie close to the reference line, indicating that the central part of the return distribution aligns well with the normal distribution. At the lower end (left tail) and the upper end (right tail) of the distribution, the points deviate below and above the reference line, respectively. Thus, this deviation in both tails suggests that the return distribution exhibits heavy tails, meaning there are more extreme values than would be expected under a normal distribution, corroborating with the results of the Shapiro-Wilk test.

Figure 4.3 provides a comprehensive visualization of the index's daily log returns and the associated 10-day risk measures.

Figure 4.3 – *Return (first panel) and the 10-day risk measures on the second, third, and fourth panels*



4.3.2 Data Preprocessing

The datasets were built by the author. There were no missing values. Thus, no imputation or deletion of data points was necessary. Given the nature of the data and the robustness of the tree-based models employed, neither normalization nor standardization were necessary. The dataset did not contain any categorical variables that required encoding. Lastly, the train-test split is detailed in 4.4.1.

4.3.3 Datasets

Seven distinct datasets were used. They were designed to evaluate the predictive performance of the selected features across three target variables: the 10-day Conditional Value at Risk, the 10-day Standard Deviation of Returns (volatility), and the 10-day Maximum Drawdown.

The **Original** dataset consists of the initial set of 80 features collected for analysis. Subsequent datasets were derived using specific feature selection methods:

The **F-Test** dataset includes features selected based on an F-test criterion. The **Pearson** dataset consists of features chosen through Pearson correlation coefficients. Features in the **Spearman** dataset were selected using Spearman correlation. The **Intersection** dataset contains features common to both the Pearson and Spearman-selected sets. The **Combined** dataset incorporates all features selected by either Pearson or Spearman methods. Finally, the **Vix** dataset includes only the feature representing the VIX index.

4.3.3.1 Original

The 'Original' dataset contains eighty features organized into categories: basic indicators, technical indicators, overseas return rate indicators, market sentiment indicators, and macroeconomic indicators. Basic indicators are variables directly related to the IBOV, such as prices and their lagged variations. Technical indicators include variables related to momentum, trends, and volatility, for example. Overseas return rate indicators refer to daily return rates of major international stock market indices. Market sentiment indicators include Vix, a volatility index calculated based on the prices of S&P500 index options, and EMBI+, an index that tracks the performance of sovereign bonds issued by emerging market countries. Finally, macroeconomic indicators are variables related to exchange rates, commodity prices, interest rates, and monetary metrics.

The primary dataset description is provided in [21]. Table 3 summarizes this dataset.

Table 3 – *Summary of the Original Features*

Type	Variable	Indicator	Definition
Basic indicators	x_1	Open	The opening price
	x_2	High	The highest price
	x_3	Low	The lowest price
	x_4	Volume	The trading volume
		...	
Technical indicators	x_{26}	Price_Change	The price change
	x_{27}	RSI	The Relative Strength Index
	x_{28}	SMA_10	The 10-day Simple Moving Average
	x_{29}	SMA_30	The 30-day Simple Moving Average
		...	
Overseas return rate indicators	x_{43}	SP500	Daily return of the S&P500 index
	x_{44}	DJIA	Daily return of the Dow Jones Industrial Average
		...	
Market Sentiment Indicator	x_{57}	Vix	Volatility Index (VIX)
	x_{80}	EMBI+	EMBI+ Index
		...	
Macroeconomic Indicators	x_{78}	C_i_C	Currency in Circulation
	x_{79}	Bank_Reserves	Bank Reserves
	x_{80}	R_monet_base	Restricted Monetary Base

4.3.3.2 F-test

The ‘F-test’ dataset is built using the F-test for feature selection. We use the ‘f_regression’ function from the ‘sklearn.feature_selection’ module. This method performs univariate linear regression tests, returning F-statistics and p-values to assess the significance of each feature with respect to the target variable y. The F-test evaluates whether there are significant differences between the means of different groups, identifying the features most likely to statistically influence the target variable. The ten best features are selected using the F-test criterion to build the ‘F-test’ dataset.

The ‘f_regression’ function works in two steps:

1. The Pearson correlation between each regressor and the target is computed; the formula is detailed in 4.3.3.3.
2. This correlation is converted to an F score and then to a p-value.

This method focuses on the strength and significance of the relationship rather than its direction since it is irrespective of the sign of the association. Table 4 lists the selected variables by the method.

Table 4 – *Chosen F-test Features*

Target	**Selected Features**
CVaR	RSI, MACD, MACD_Signal, CP_Std_Dev, MFI, Williams_%R, FTSE100, KOSPI, Vix, EMBI+
Volatility	MACD, MACD_Signal, CP_Std_Dev, FTSE100, HSI, ASX200, KOSPI, Vix, Copper, EMBI+
Max Drawdown	RSI, MACD, MACD_Signal, CP_Std_Dev, MFI, Williams_%R, FTSE100, HSI, Vix, EMBI+

4.3.3.3 Pearson

The ‘Pearson’ dataset is built based on the Pearson correlation coefficient. It measures the statistical dependence between the set of features $\mathbf{x}$, as detailed in Table 21, and the target variable y, evaluating the extent to which their relationship can be described using a linear function. The dataset is populated with the top ten features selected based on this correlation criterion.

The formula for Pearson correlation coefficient is given by:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where r_{xy} is the Pearson correlation coefficient between x and y, x_i and y_i are the individual values of samples of x and y, $\bar{x}$ and $\bar{y}$ are the means of samples of x and y, n is the total number of samples. Table 5 lists the selected variables by this method.

Table 5 – *Chosen Pearson Features*

Target	**Selected Features**
CVaR	Vix, MACD, CP_Std_Dev, MACD_Signal, RSI, EMBI+, FTSE100, Williams_%R, MFI, KOSPI
Volatility	Vix, CP_Std_Dev, MACD, MACD_Signal, FTSE100, EMBI+, KOSPI, HSI, Copper, ASX200
Max Drawdown	MACD, Vix, MACD_Signal, CP_Std_Dev, RSI, Williams_%R, MFI, EMBI+, FTSE100, HSI

4.3.3.4 Spearman

The 'Spearman' dataset is built based on the Spearman correlation coefficient. We measure the statistical dependence of the set $\mathbf{x}$ of features, detailed in 21, and the target variable y, assessing how well the relationship between the variables can be described using a monotonic function. The ten best features are selected using this correlation criterion to build the 'Spearman' dataset.

The formula for Spearman correlation coefficient is given by:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

Where ρ is the Spearman correlation coefficient, d_i is the difference between the ranks of corresponding variables x_i and y_i, and n is the number of observations. Table 6 lists the selected variables by this method.

Table 6 – *Chosen Spearman Features*

Target	**Selected Features**
CVaR	RSI, MACD, Williams_%R, Vix, MFI, MACD_Signal, FTSE100, KOSPI, EMBI+, HSI
Volatility	Vix, KOSPI, FTSE100, Low, Lower_Band, Low_Lag_1, Low_Lag_2, Low_Lag_3, Low_Lag_4, MACD
Max Drawdown	RSI, Williams_%R, MACD, MFI, MACD_Signal, Vix, FTSE100, Low, HSI, Low_Lag_1

4.3.3.5 Intersection

The 'Intersection' dataset is built using only the features selected by both the Pearson and the Spearman correlation. Table 7 lists the selected features.

Table 7 – *Intersection of Features between Pearson and Spearman*

Target	**Selected Features**
CVaR	Williams_%R, Vix, FTSE100, MFI, RSI, MACD_Signal, EMBI+, MACD, KOSPI
Volatility	MACD, Vix, KOSPI, FTSE100
Max Drawdown	FTSE100, Williams_%R, MACD, MFI, RSI, HSI, Vix, MACD_Signal

4.3.3.6 Combined

The 'Combined' dataset is built using all features the Pearson and Spearman correlation selected. Table 9 lists the selected features.

Table 8 – *All Features selected by Pearson and Spearman*

Target	Selected Features
CVaR	MFI, RSI, MACD, KOSPI, Williams_%R, Vix, HSI, FTSE100, MACD_Signal, CP_Std_Dev, EMBI+
Volatility	MACD, Low, Low_Lag_1, Copper, Low_Lag_2, Low_Lag_3, ASX200, Vix, Low_Lag_4, HSI, FTSE100, CP_Std_Dev, EMBI+, KOSPI, Lower_Band, MACD_Signal
Max Drawdown	FTSE100, Williams_%R, MACD, HSI, Low, Vix, MACD_Signal, EMBI+, Low_Lag_1, CP_Std_Dev, MFI, RSI

4.3.3.7 Vix

The 'Vix' dataset is composed solely of the 'Vix' feature. As observed in previous analyses, this feature is highly significant and informative, demonstrating a strong relationship with all the target variables. Table 9 illustrates this dataset.

Table 9 – *All Features selected by Pearson and Spearman*

Target	Selected Features
CVaR	Vix
Volatility	Vix
Max Drawdown	Vix

4.4 MODELING

4.4.1 Training and Validation

The datasets were divided using a temporal train-test split, where 80% of the data points were allocated for training and 20% for testing across all seven datasets. For all the data splits, the training data spans from August 30, 2006, to January 15, 2020, while the testing data covers the period from January 16, 2020, to May 22, 2023. Figures 4.4, 4.5, and 4.6 illustrate the training and testing sets for the three target variables, where the vertical dashed line represents the split.

Figure 4.4 – *CVaR - Training and Testing Sets*

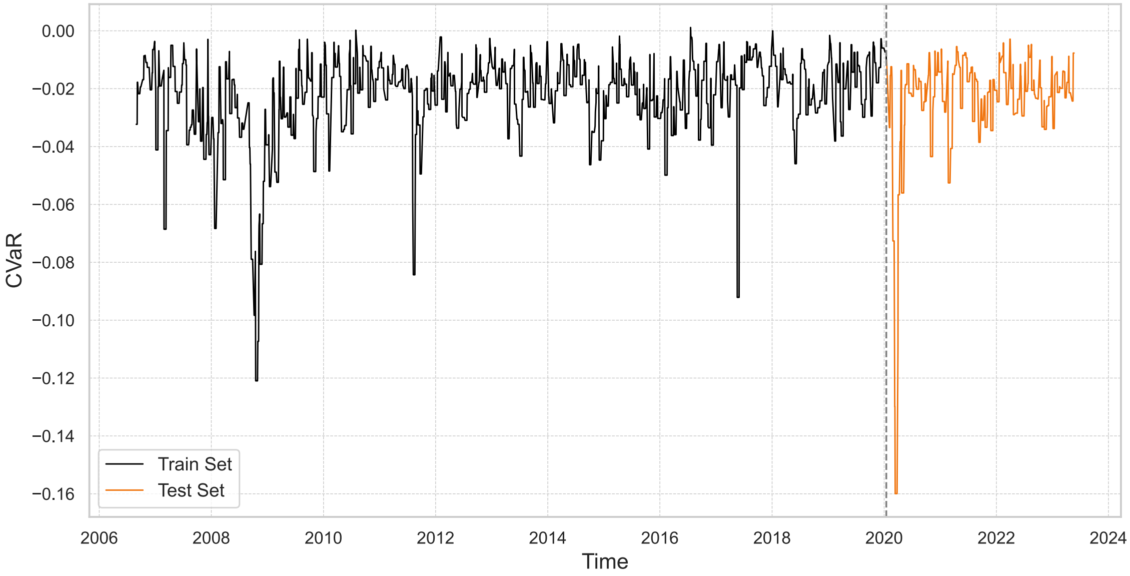


Figure 4.5 – *Volatility - Training and Testing Sets*

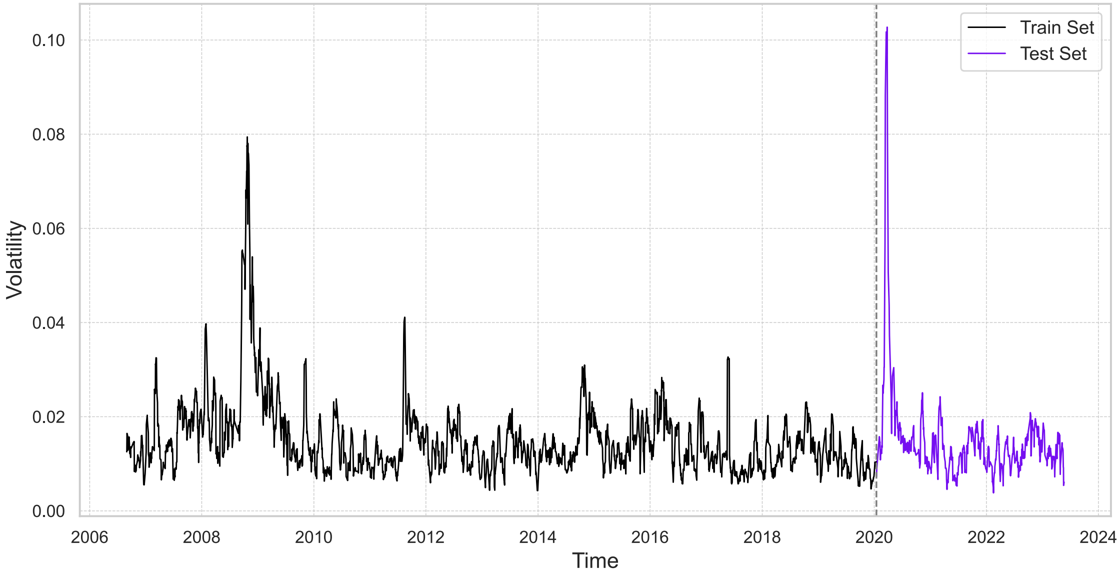
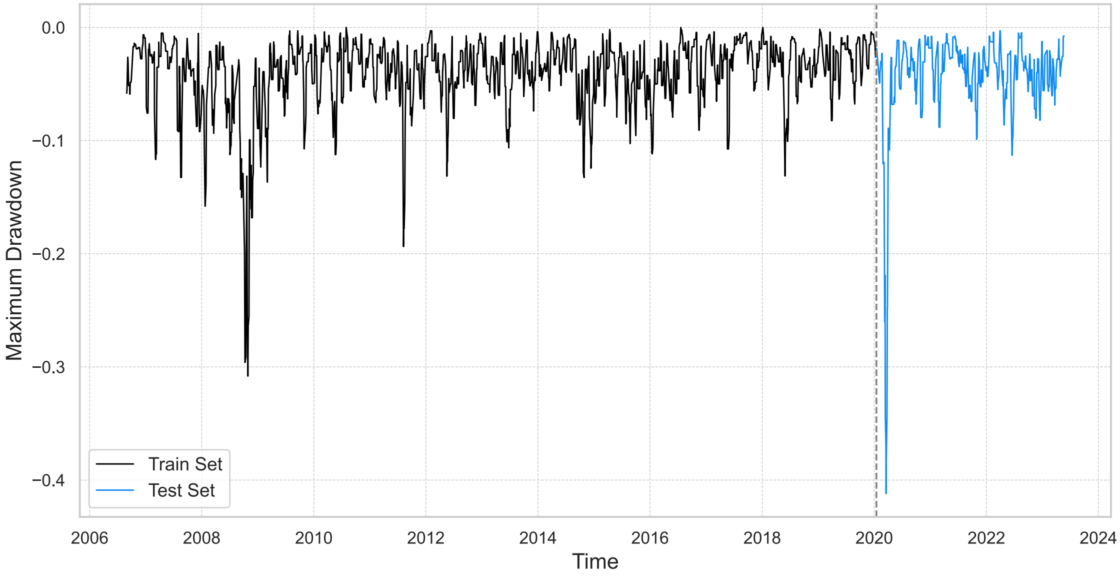


Figure 4.6 – *Maximum Drawdown - Training and Testing Sets*



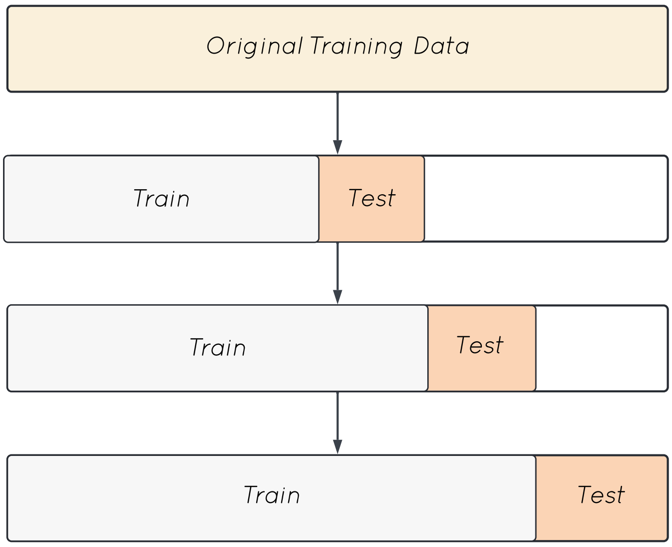
4.4.2 Hyperparameter Tuning

Hyperparameters are parameters that the models don't learn directly from the data. However, they greatly influence how an algorithm effectively learns, playing a critical role in optimizing performance. Therefore, hyperparameter tuning is used in this study to identify optimal configurations; the best results are detailed in 15.

Random search was employed for model tuning. This search strategy samples a subset of the search space instead of trying all parameter values, reducing its computational expense compared to grid search. Despite its lower computational cost, random search can achieve accuracy improvements comparable to those of grid search (BERGSTRA; BENGIO, 2012).

To ensure robustness in evaluation, temporal cross-validation was utilized, partitioning the data into ten sequential subsets. This approach maintains the temporal order, ensuring that models cannot access future data during training, which could otherwise bias performance evaluation. Figure 4.7 illustrates the temporal cross-validation procedure across three splits.

Figure 4.7 – *Time Series Cross-Validation Example*



Five models were configured, each with its respective parameter distributions. For each model, 'RandomizedSearchCV' sampled 100 different parameter configurations, evaluating them using the RMSE within the temporal cross-validation framework. This process resulted in 1000 model fits per algorithm, encompassing all folds and parameter combinations.

In summary, the following hyperparameters were optimized using these parameter distributions:

– **AdaBoost**:

  · The learning rate (*learning_ rate*): Randomly selected from a uniform distribution between 0.01 and 0.5.

  · The number of boosting iterations (*n_ estimators*): Randomly selected between 5 and 200.

– **Gradient Boosting**:

  · The learning rate (*learning_ rate*): Randomly selected from a uniform distribution between 0.01 and 0.5.

  · The number of boosting iterations (*n_ estimators*): Randomly selected between 5 and 200.

  · The maximum depth of the tree (*max_ depth*): Randomly selected between 1 and 6.

– **XGBoost**:

  · The learning rate (*learning_ rate*): Randomly selected from a uniform distribution between 0.01 and 0.5.

    - The number of boosting iterations (*n_ estimators*): Randomly selected between 5 and 200.
    - The maximum depth of the tree (*max_ depth*): Randomly selected between 1 and 6.
    - The minimum loss reduction (*gamma*): Randomly selected from a uniform distribution between 0 and 1.
    - The $L2$ regularization term (*reg_ lambda*): Randomly selected between 0 and 5.

  - **LightGBM**:
    - The learning rate (*learning_ rate*): Randomly selected from a uniform distribution between 0.01 and 0.5.
    - The number of boosting iterations (*n_ estimators*): Randomly selected between 5 and 200.
    - The maximum depth of the tree (*max_ depth*): Randomly selected between 1 and 6.
    - The maximum number of leaves in one tree (*num_ leaves*): Randomly selected between 10 and 40.
    - The $L2$ regularization term (*reg_ lambda*): Randomly selected between 0 and 5.

  - **CatBoost**:
    - The learning rate (*learning_ rate*): Randomly selected from a uniform distribution between 0.01 and 0.5.
    - The maximum number of trees that can be built (*iterations*): Randomly selected between 5 and 200.
    - The maximum depth of the tree (*max_ depth*): Randomly selected between 1 and 6.
    - The $L2$ regularization term (*l2_ leaf_ reg*): Randomly selected between 0 and 5.
    - The degree of randomness applied to score splits during the selection of the tree structure (*random_ strength*): Randomly selected between 0 and 5.

4.5 EVALUATION METRICS

The Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) are used as quantitative metrics for evaluating the predictive performance.

The Mean Absolute Error is computed as the average of the sum of absolute differences between the actual values and the predicted values:

$$MAE = \frac{1}{N}\sum_{t=1}^{N} |y_t - \hat{y}_t|$$

This metric is simple to understand and interpret. Moreover, it is less sensitive to outliers compared to RMSE.

The RMSE is a measure of the dispersion of errors between actual values and predictions:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}$$

The Root Mean Squared Error penalizes larger errors more than MAE, being useful when larger errors are undesirable. This is the metric used for hyperparameter optimization.

4.6 STATISTICAL TESTS USED IN THE RESULTS

4.6.1 Levene's Test for Homogeneity of Variances

Levene's test was used to assess the homogeneity of variances across different groups. This test evaluates the null hypothesis that the variances are equal across groups. A significant result (p-value less than 0.05 at the 5% significance level, for example) indicates that the assumption of equal variances is violated, suggesting that the variances between groups are significantly different. Therefore, if the test showed a violation of the homogeneity of variances, the Kruskal-Wallis test wasn't performed since it demands homoscedasticity.

4.6.2 Kruskal-Wallis Test

The Kruskal–Wallis test is a nonparametric alternative to traditional analysis of variance (ANOVA). It was used to avoid relying on the assumption of normality for the results. Although it does not require the data to be normally distributed, it does require homoscedasticity.

The test applies the analysis of variance to the ranks of the observations. By ranking the observations and replacing each value with its rank, this method is less likely to be distorted by nonnormality and outliers. The null hypothesis is that the medians of all groups are equal, while the alternative hypothesis is that at least the median of the population of one group is different from the median of the population of at least one other group. Therefore, applying the test to the ranks of the observations provides a robust comparison of group medians.

4.6.3 Post Hoc Analysis: Conover-Iman Test

When significant differences were detected by the Kruskal-Wallis test, post hoc analysis was performed using the Conover-Iman test. This test tests for stochastic dominance among multiple pairwise comparisons between groups while controlling for the *family-wise error rate.* P-values from the Conover-Iman test were adjusted using the Bonferroni correction to account for multiple comparisons.

5 RESULTS

This chapter presents the results of this undergraduate thesis. The analysis focuses on the performance of Adaboost, Gradient Boosting, XGBoost, LightGBM, and CatBoost in predicting three risk measures: the 10-day Conditional Value at Risk (CVaR), the 10-day Standard Deviation of Returns (Volatility), and the 10-day Maximum Drawdown.

First, in 5.1, we analyze the average performance metrics of the algorithms across the different risk measures, focusing on predictive performance and then their computational efficiency.

Following this, in 5.2, we evaluate the performance metrics of the models across seven different datasets; each created using specific feature selection methods. This section aims to understand how different datasets affect model performance.

Next, in 5.3, we identify and discuss the best individual models for each risk measure, providing details about their hyperparameters and feature importance.

Finally, in 5.4, statistical tests are conducted to validate the significance of the observed performance differences for RMSE.

5.1 AVERAGE ALGORITHM RESULTS

Table 10 presents a summary of the algorithms' performance metrics, including the Average RMSE, Standard Deviation of RMSE, Average MAE, and Standard Deviation of MAE for each risk measure. The lowest values are highlighted in bold font.

Table 10 – *Summary of Performance Metrics Grouped by Algorithm and Target*

Algorithm	**Target**	**Average RMSE**	**Standard Deviation (RMSE)**	**Average MAE**	**Standard Deviation (MAE)**
Adaboost	CVaR	**0.013686**	0.002039	0.009814	0.000996
	Max Drawdown	**0.029071**	0.002600	0.020614	**0.001470**
	Volatility	0.007329	0.001036	0.004700	0.000516
Gradient Boosting	CVaR	0.014629	0.001201	0.010229	0.001006
	Max Drawdown	0.030029	0.002462	0.018943	0.003118
	Volatility	**0.006800**	0.000666	**0.004386**	0.000372

Continued on next page

Algorithm	**Target**	**Average RMSE**	**Standard Deviation (RMSE)**	**Average MAE**	**Standard Deviation (MAE)**
LightGBM	CVaR	0.015486	**0.000524**	0.010357	**0.000586**
	Max Drawdown	0.029214	0.003742	**0.018657**	0.003105
	Volatility	0.007414	**0.000393**	0.004600	**0.000294**
XGBoost	CVaR	0.015543	0.001249	0.010457	0.000739
	Max Drawdown	0.031129	0.002414	0.019786	0.002363
	Volatility	0.008643	0.001688	0.004743	0.000443
CatBoost	CVaR	0.015100	0.001060	**0.009771**	0.000652
	Max Drawdown	0.033200	**0.001159**	0.020171	0.002869
	Volatility	0.007529	0.000479	0.004529	0.000309

For the CVaR risk measure, Adaboost shows the lowest average RMSE (0.013686), making it the most effective model for minimizing larger errors. Similarly, Adaboost is, on average, the better model for predicting Max Drawdown, showing the lowest average RMSE (0.029071). Gradient Boosting outperforms the other models with the lowest average RMSE (0.006800) and MAE (0.004386) for the Volatility risk measure. Additionally, LightGBM stands out for its consistency, exhibiting the lowest standard deviations for both RMSE and MAE in predicting CVaR and Volatility.

Since hyperparameter tuning was performed using RMSE, it is reasonable to prioritize RMSE to evaluate model performance. However, it is also important to analyze the results for the MAE metric, considering all individual differences are weighted equally in the average. Within this context, the results show that Catboost is the most effective model for the CVaR risk measure prediction, with the lowest average MAE (0.009771). Furthermore, LightGBM is the better model for the Max Drawdown prediction, with the lowest average MAE (0.018657).

Gradient Boosting performs best for volatility prediction when considering both RMSE and MAE. Adaboost has the lowest average RMSE for CVaR and Max Drawdown, CatBoost excels for CVaR when MAE is prioritized, and LightGBM is superior for Max Drawdown based on MAE. These results suggest that Adaboost and Gradient Boosting are preferable for minimizing larger errors, whereas CatBoost and LightGBM are better for reducing the overall average error. Moreover, LightGBM can be considered the best overall model due to its consistency.

Figures 5.1 and 5.2 depict the RMSE and MAE across the algorithms, visually confirming the analysis of the metrics. The downward arrows in the figures highlight the models with the best performance for each risk measure based on the respective error metric:

Figure 5.1 – *RMSE Across Algorithms*

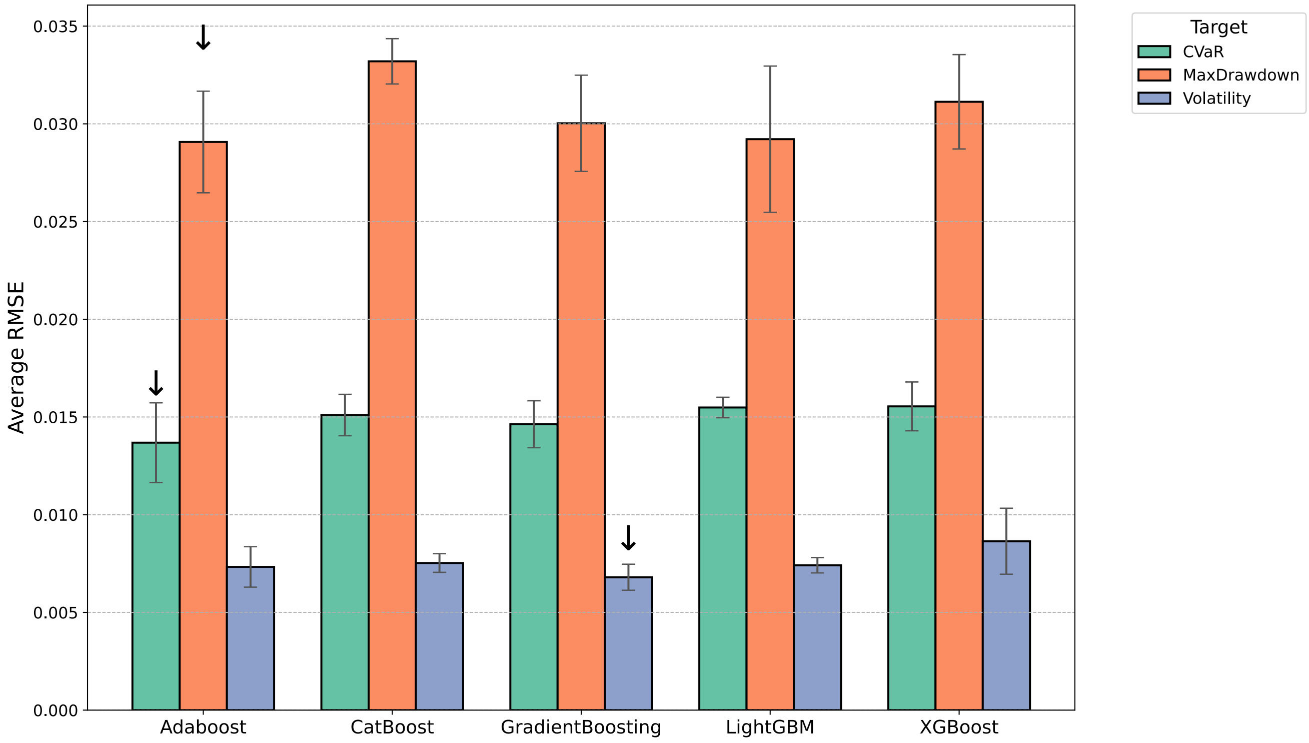


Figure 5.2 – *MAE Across Algorithms*

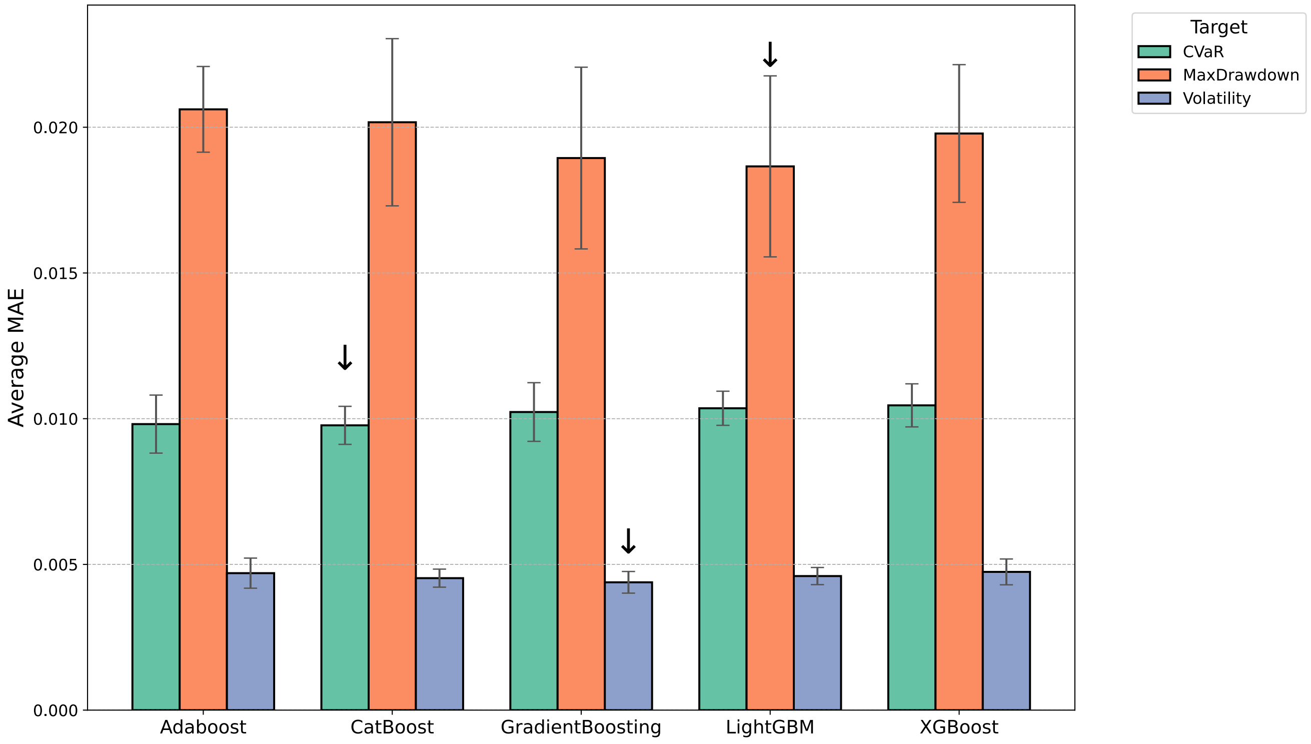


Table 11 summarizes the training and prediction times for each algorithm across the three risk measures. It summarizes the algorithms' computational efficiency, including the Average Training Time, Standard Deviation of the Average Training Time, Average Prediction Time, and Standard Deviation of the Average Prediction Time. The lowest values are highlighted in bold font.

Table 11 – *Summary of Training and Prediction Metrics Grouped by Algorithm and Target*

Algorithm	**Target**	**Average Training Time (s)**	**Training Time Std Dev (s)**	**Average Prediction Time (s)**	**Prediction Time Std Dev (s)**
Adaboost	CVaR	3.862043	7.059236	0.051629	0.057815
	Max Drawdown	0.829071	1.031130	0.020414	0.028619
	Volatility	0.501814	0.604052	0.008914	0.008339
Gradient Boosting	CVaR	2.375614	3.762860	0.007800	**0.007310**
	Max Drawdown	3.423157	2.111680	0.007871	0.007997
	Volatility	1.355014	1.982535	**0.003786**	0.007490

Continued on next page

Algorithm	**Target**	**Average Training Time (s)**	**Training Time Std Dev (s)**	**Average Prediction Time (s)**	**Prediction Time Std Dev (s)**
XGBoost	CVaR	0.127171	**0.151499**	0.011300	0.015311
	Max Drawdown	**0.077157**	**0.038229**	0.007671	0.011802
	Volatility	**0.055300**	**0.020023**	0.005714	**0.004266**
LightGBM	CVaR	**0.120629**	0.262523	0.010343	0.014334
	Max Drawdown	0.085800	0.062924	0.009143	**0.006453**
	Volatility	0.134571	0.271994	0.006400	0.005040
CatBoost	CVaR	0.411371	0.482158	**0.006771**	0.009876
	Max Drawdown	0.143529	0.098571	**0.005543**	0.007868
	Volatility	0.163700	0.176333	0.003986	0.005854

XGBoost and LightGBM demonstrate the shortest training times across all risk measures, whereas Adaboost and Gradient Boosting have longer training times, especially for CVaR and Max Drawdown. Moreover, XGBoost and LightGBM have low standard deviations in both training and prediction times, indicating consistent performance. For prediction times, CatBoost shows the shortest values.

When considering both computational efficiency and predictive performance, LightGBM emerges as the best algorithm due to its low training and prediction times and consistent performance.

5.2 AVERAGE DATASET RESULTS

In this section, we evaluate the performance of the models across different datasets to understand how they generalize to various data conditions.

Table 12 summarizes the datasets' performance metrics, including the Average RMSE, Standard Deviation of RMSE, Average MAE, and Standard Deviation of MAE for each risk measure. The lowest values are highlighted in bold font.

Table 12 – *Summary of Dataset Performance Metrics Grouped by Dataset and Target*

Dataset	Target	Average RMSE	Standard Deviation (RMSE)	Average MAE	Standard Deviation (MAE)
Combined	CVaR	0.01506	0.001612	0.01072	0.000955
	Max Drawdown	0.02972	0.002995	0.01760	0.002691
	Volatility	0.00722	0.001087	0.00436	0.000404
FTest	CVaR	0.01456	0.000820	0.01004	0.000152
	Max Drawdown	0.03082	0.002007	0.02104	0.001396
	Volatility	0.00708	0.000370	0.00462	0.000286
Intersection	CVaR	0.01410	0.001334	0.00936	0.000483
	Max Drawdown	0.02982	0.002213	0.01964	0.000568
	Volatility	**0.00668**	**0.000327**	**0.00416**	**0.000055**
Original	CVaR	0.01634	0.001544	0.01078	0.000965
	Max Drawdown	0.03016	0.002194	0.01878	0.000736
	Volatility	0.00858	0.001965	0.00484	0.000397
Pearson	CVaR	0.01480	0.001377	0.01000	0.000689
	Max Drawdown	0.03152	0.001492	0.02188	0.000773
	Volatility	0.00774	0.001210	0.00498	0.000409
Spearman	CVaR	**0.01358**	0.000923	**0.00930**	0.000339
	Max Drawdown	**0.02720**	0.003116	**0.01594**	0.002072
	Volatility	0.00776	0.000709	0.00436	0.000195
Vix	CVaR	0.01578	**0.000179**	0.01068	**0.000084**

Continued on next page

Dataset	**Target**	**Average RMSE**	**Standard Deviation (RMSE)**	**Average MAE**	**Standard Deviation (MAE)**
	Max Drawdown	0.03446	**0.000723**	0.02256	**0.000416**
	Volatility	0.00774	0.000483	0.00482	0.000084

The Spearman dataset shows the best overall performance in terms of minimizing both RMSE and MAE for CVaR and Max Drawdown, indicating it is more effective for these risk measures. Although the Vix dataset does not have the lowest average RMSE or MAE, it is the most consistent for both CVaR and Max Drawdown, suggesting it provides stable predictions with minimal variability.

Unlike Pearson correlation, which measures linear relationships, Spearman correlation assesses the strength and direction of monotonic relationships, making this feature selection method more robust in capturing non-linear dependencies. Another important factor is that the Spearman correlation coefficient is less sensitive to outliers. These factors might explain why the Spearman dataset performed better.

Additionally, for Volatility, the Intersection dataset stands out in both average RMSE and MAE. It shows high consistency with the lowest standard deviations, making it the optimal dataset for volatility prediction by combining accuracy and stability. This dataset likely balances the strengths of the Pearson and Spearman datasets, reducing noise and enhancing the signal related to volatility. This balance contributes to both accuracy and stability in predictions.

The Original dataset shows the highest error metrics for CVaR and Volatility. This dataset may include irrelevant or redundant features that do not contribute positively to predicting these risk measures. Although tree-based methods were used in this study and are known to be robust to irrelevant features, the dataset may not be sufficiently large to handle this particular regression task effectively. That could explain why overfitting was a significant challenge in this study.

On the other hand, the Vix dataset shows the highest error metrics for the Max Drawdown target. It may lack sufficient information to capture the complexity required for accurate predictions of the Max Drawdown.

Figures 5.3 and 5.4 illustrate the RMSE and MAE across the datasets, visually confirming the analysis of the metrics. The downward arrows in the figures highlight the datasets with the best performance for each risk measure based on the respective error metric.

Figure 5.3 – *RMSE Across Datasets*

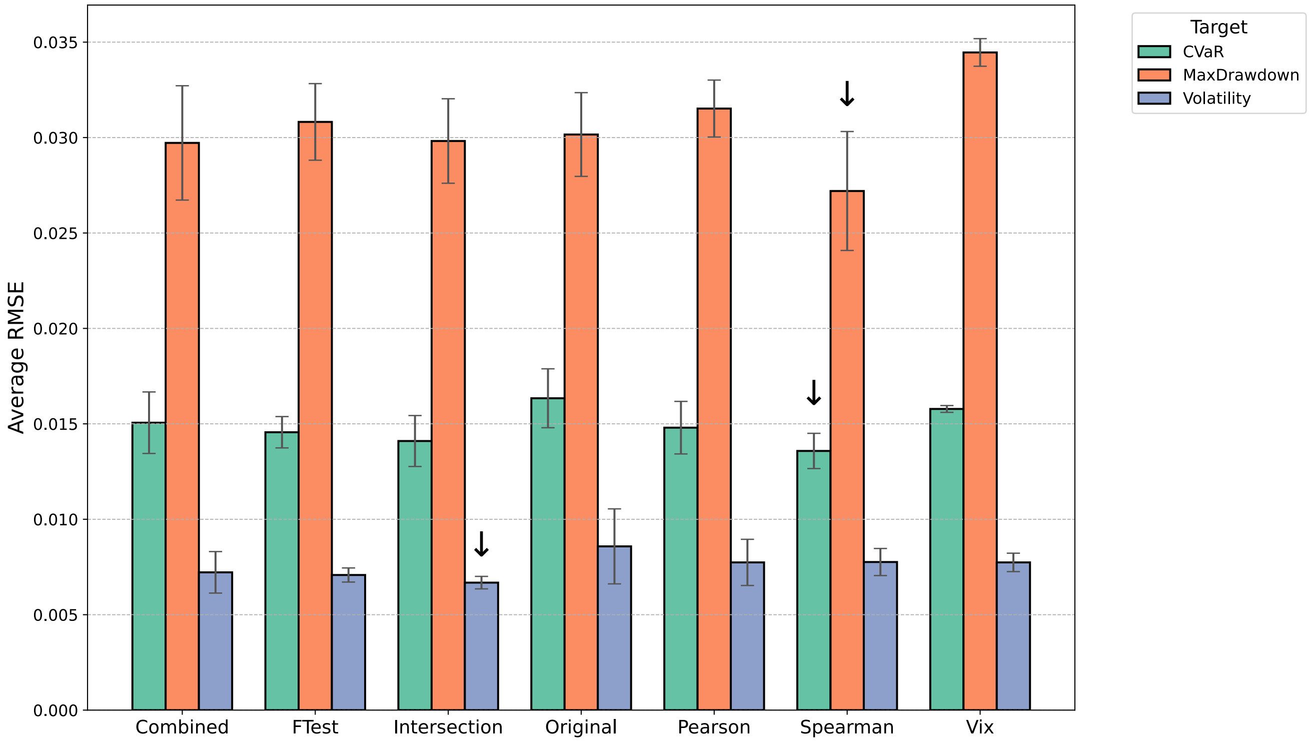


Figure 5.4 – *MAE Across Datasets*

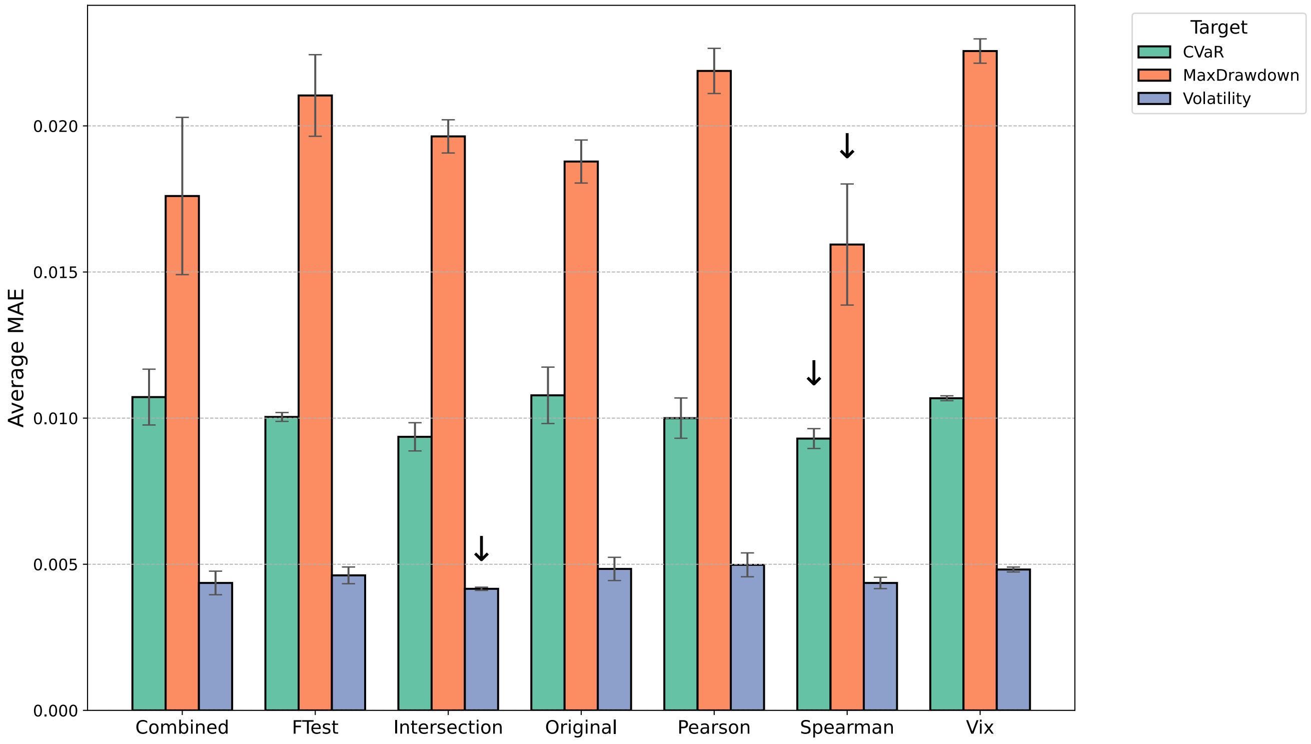


Table 13 summarizes the datasets' computational efficiency, including the Average Training Time, Standard Deviation of the Average Training Time, Average Prediction

Time, and Standard Deviation of the Average Prediction Time. The lowest values are highlighted in bold font.

Table 13 – *Summary of Training and Prediction Metrics Grouped by Dataset and Target*

Dataset	**Target**	**Average Training Time (s)**	**Training Time Std Dev (s)**	**Average Prediction Time (s)**	**Prediction Time Std Dev (s)**
Combined	CVaR	2.20056	3.814153	0.01094	0.012073
	Max Drawdown	1.43958	1.781385	0.02386	0.032395
	Volatility	0.60608	0.847022	0.00676	0.006718
FTest	CVaR	0.20596	0.186672	**0.00166**	**0.003386**
	Max Drawdown	0.92290	1.790412	0.00144	0.003220
	Volatility	0.24622	0.369589	**0.00166**	0.003712
Intersection	CVaR	0.20672	0.197284	0.01040	0.007847
	Max Drawdown	0.29934	0.536755	**0.00132**	**0.002385**
	Volatility	0.15458	0.227862	0.00248	0.005545
Original	CVaR	5.74700	8.084585	0.05484	0.055153
	Max Drawdown	1.67404	2.797593	0.01520	0.011883
	Volatility	1.73952	2.231398	0.01360	0.004116
Pearson	CVaR	0.18932	0.187217	0.00606	0.006386
	Max Drawdown	0.92542	1.715134	0.00860	0.006669
	Volatility	0.09316	**0.032912**	0.00276	**0.003330**
Spearman	CVaR	0.99478	2.008702	0.02718	0.050066
	Max Drawdown	1.05174	1.705097	0.01430	0.011589
	Volatility	0.18294	0.207246	0.00682	0.007698
Vix	CVaR	**0.11122**	**0.093123**	0.01190	0.012658
	Max Drawdown	**0.06918**	**0.055904**	0.00618	0.006559
	Volatility	**0.07206**	0.058240	0.00624	0.005770

The Vix dataset consistently shows the lowest average training times across all targets, while the Original dataset exhibits the highest average training times. This is expected since the Vix dataset has only one feature variable, while the Original dataset has the most features. For prediction times, the Vix and the F-test datasets perform well. Additionally, the Original dataset exhibits the highest standard deviations for training and prediction times, exhibiting greater variability.

5.3 BEST INDIVIDUAL MODELS

The last sections focused on the average performance of the algorithms and datasets, where we obtained an overview of general performance. In this section, we'll analyze the best individual models.

The interpretability of the models will be considered, with discussions focusing on why certain features might be more relevant to a particular model outcome. However, it is important to note that the discussions do not indicate causal effects. The models used are not causal models, and any observed associations should not be interpreted as evidence of causation.

Table 14 presents a summary of the best models considering RMSE and MAE. It includes the Algorithm, Target, Dataset, RMSE, MAE, Training time, and Prediction time. The lowest values are highlighted in bold font.

Table 14 – *Best Individual Models and Datasets per Target*

Algorithm	**Target**	**Dataset**	**RMSE**	**MAE**	**Training time (s)**	**Prediction time (s)**
Adaboost	CVaR	Intersection	0.0120	0.0087	0.4380	0.0157
LightGBM	Max Drawdown	Spearman	0.0239	0.0141	0.0774	0.0156
Gradient Boosting	Volatility	Combined	0.0060	0.0038	2.0172	0.0000

Adaboost performs best for the CVaR target using the Intersection dataset, with an RMSE of 0.0120 and MAE of 0.0087, highlighting it as the best performer for this target since it also has the best average performance metrics. LightGBM is optimal for the Max Drawdown target using the Spearman dataset, with an RMSE of 0.0239 and MAE of 0.0141, and in average performance analysis, LightGBM is noted for its consistency and lowest MAE. Gradient Boosting is the best algorithm for the Volatility target with the Combined dataset, achieving an RMSE of 0.0060 and MAE of 0.0038, and is also the best model for Volatility in average performance analysis.

Table 15 shows the hyperparameters of the best specific models.

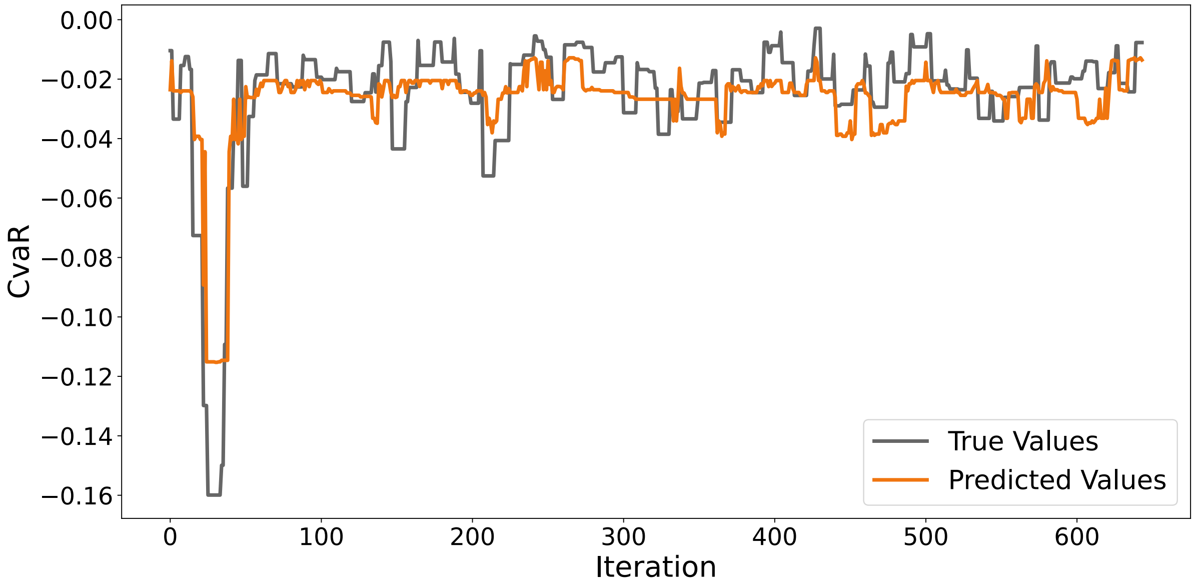
Table 15 – *Hyperparameters of the Best Performing Models*

Algorithm	**Hyperparameter**	**Value**
Adaboost	*learning_ rate*	0.06009116724413254
	n_ estimators	15
LightGBM	*learning_ rate*	0.3317260257815028
	n_ estimators	182
	max_ depth	3
	num_ leaves	32
	reg_ lambda	0
Gradient Boosting	*learning_ rate*	0.05284105248216985
	n_ estimators	37
	max_ depth	4

5.3.1 CVaR

In this subsection, we analyze the performance of the Adaboost model in predicting the Conditional Value at Risk using the Intersection dataset. Figure 5.5 illustrates the comparison between the true CVaR values and the predicted values generated by the Adaboost model.

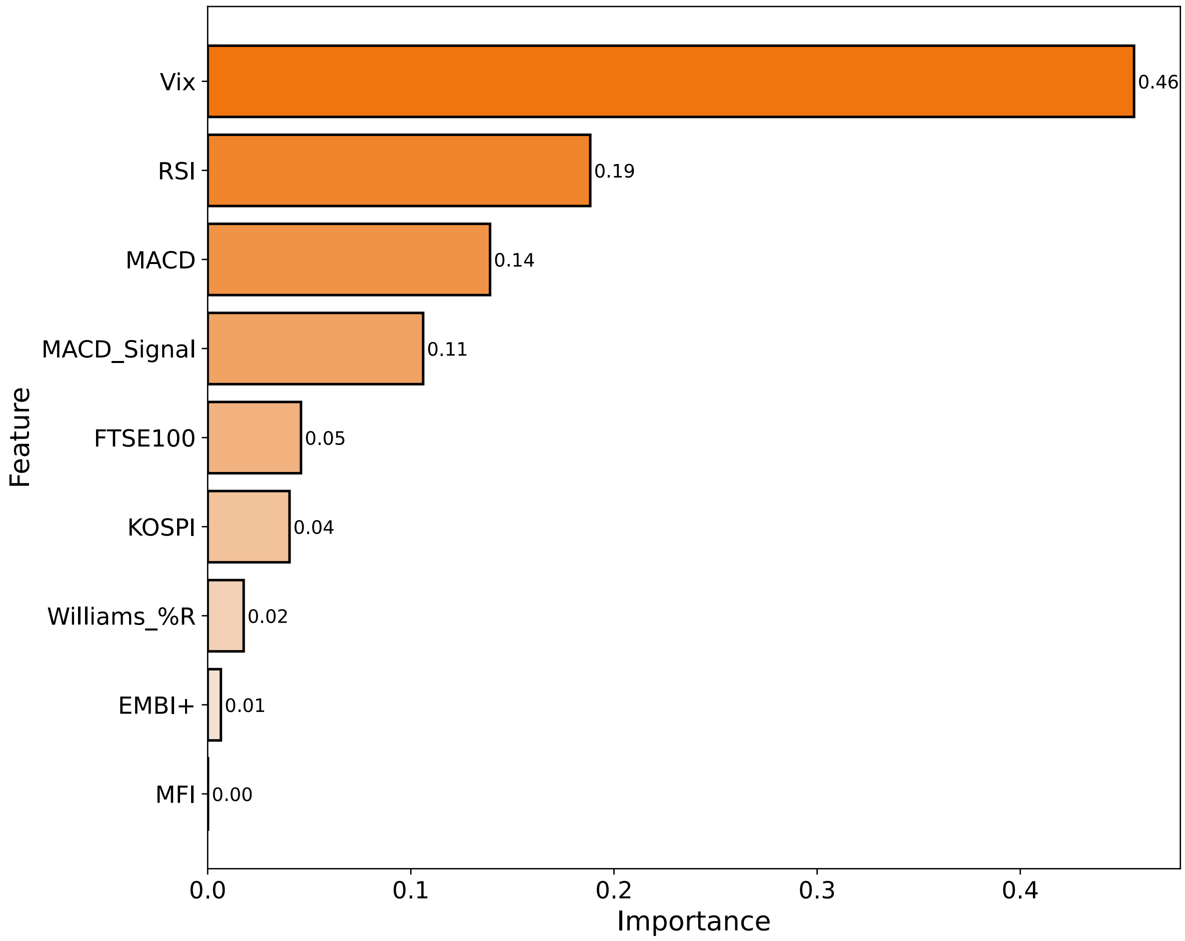
Figure 5.5 – *Comparison of Predicted vs. Actual CVaR Values Using Adaboost on the Intersection Dataset*



The predicted values (orange line) generally follow the trend of the actual values (gray line), indicating that the Adaboost model captures the overall pattern of the CVaR values reasonably well. At the start (first 50 iterations), there is a significant discrepancy between the actual and predicted values. This substantial drop in the true values, which the model fails to capture accurately, could be explained by an unusual event not present in the training phase (COVID-19 pandemic). Even though the model may capture major CVaR trends, it might be further improved.

Figure 5.6 presents the feature importance for the best specific CVaR model.

Figure 5.6 – *Relative Importance of Features in Predicting CVaR Using Adaboost on the Intersection Dataset*



Vix is the most important factor in predicting CVaR, followed by momentum indicators such as RSI and MACD.

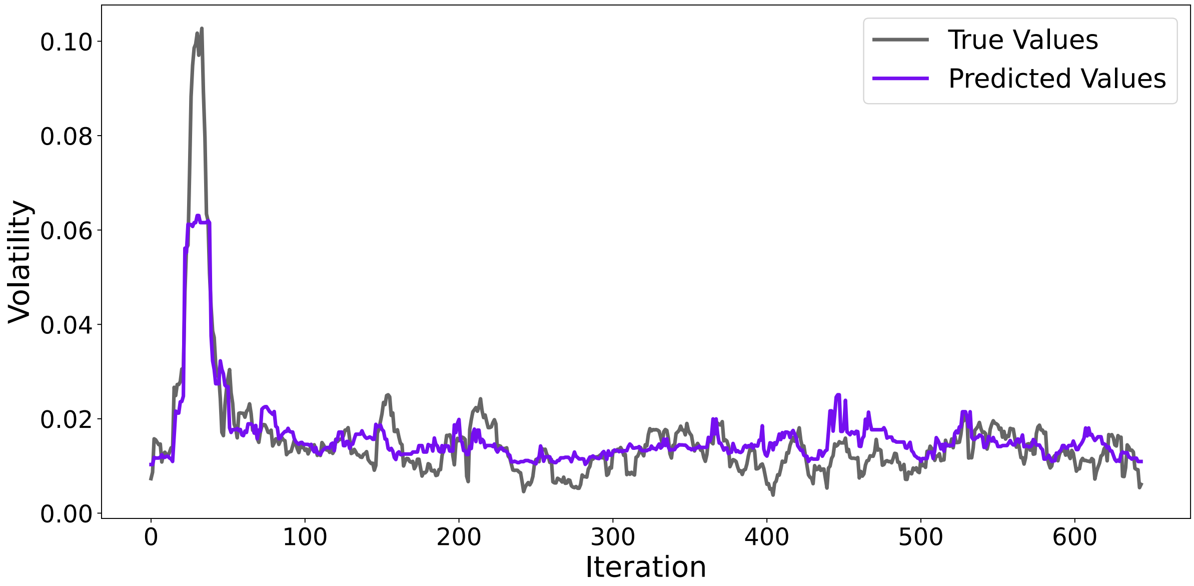
Vix, the Chicago Board Options Exchange Volatility Index, is a key market sentiment indicator. Since high volatility usually correlates with increased risk of significant losses, it is an important factor in risk assessments like CVaR, which focuses on extreme downside risk. This could potentially explain the effect of this feature. Even though the feature uses SP500 data, it plays a critical role in predicting CVaR for the IBOV. This could be explained due to the interconnectedness of global financial markets and the transmission of risk sentiment.

The Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD) are momentum indicators that help identify the strength and direction of price trends. These features can provide early warnings of shifts in market conditions, helping predict CVaR.

5.3.2 Volatility

In this subsection, we analyze the performance of the Gradient Boosting model in predicting volatility (standard deviation of returns) using the Combined dataset. Figure 5.7 illustrates the comparison between the true volatility values and the predicted values generated by the Gradient Boosting model.

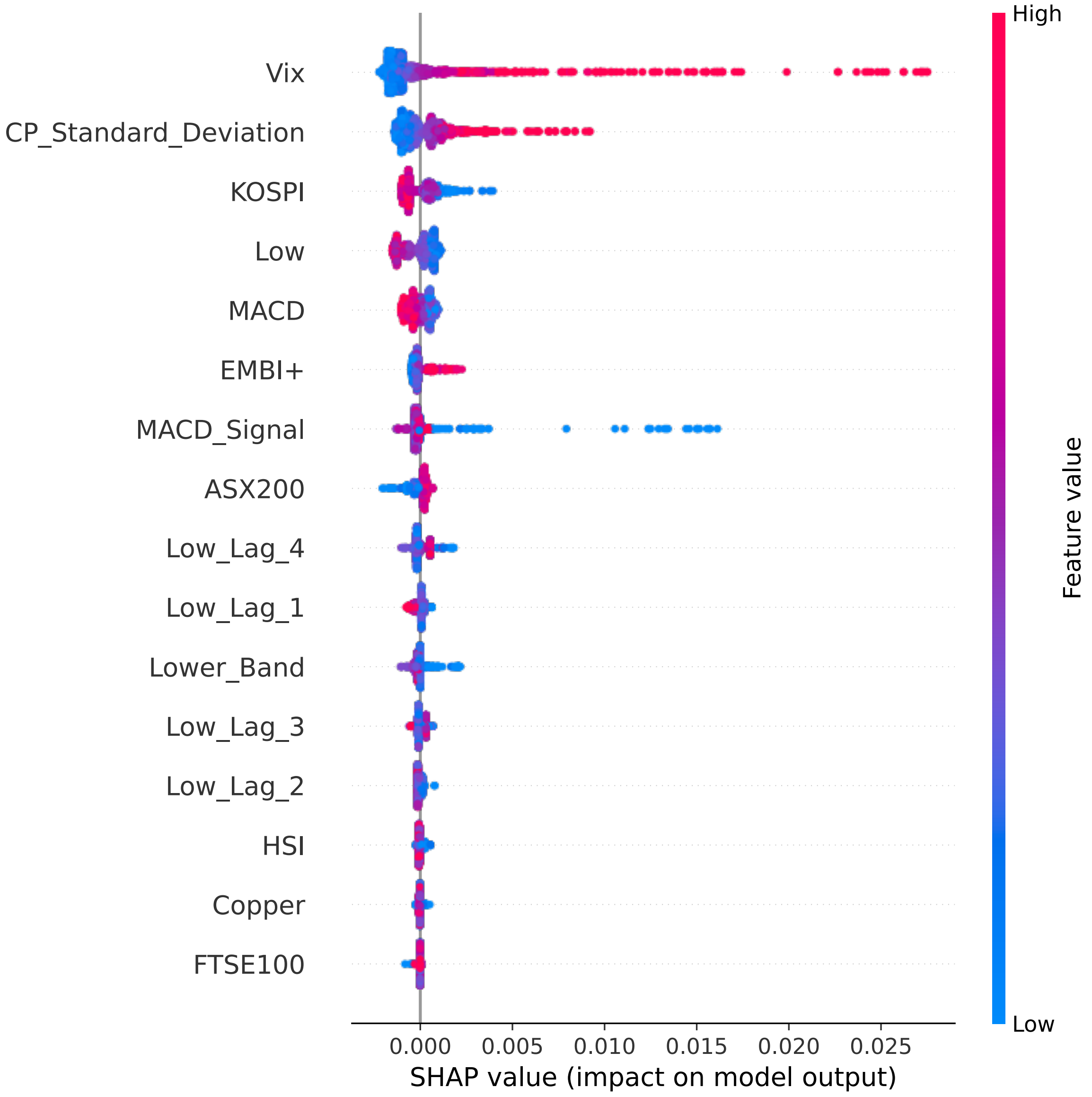
Figure 5.7 – *Comparison of Predicted vs. Actual Volatility Values Using Gradient Boosting on the Combined Dataset*



The predicted values (purple line) generally follow the major trends of the actual values (gray line). The Gradient Boosting model may be capturing the overall patterns in volatility. However, visually, both the actual and predicted volatility values show more noise and fluctuations than would be expected. The model captures the initial peak, but there is a noticeable deviation between the actual and predicted values. As in the CVaR prediction, this could be explained by an unusual event not present in the training phase, which means the algorithm couldn't learn to extrapolate to such extreme events. The model shows increased noise and inconsistency in following the actual values in the mid to late iterations. Therefore, even though the model seems to follow the major trends in volatility, it might be further improved.

Figure 5.8 presents a SHAP values summary plot for the Gradient Boosting model on the Combined dataset. The most impactful features are at the top.

Figure 5.8 – *Shap Values - Summary Plot for Gradient Boosting on the Combined Dataset*

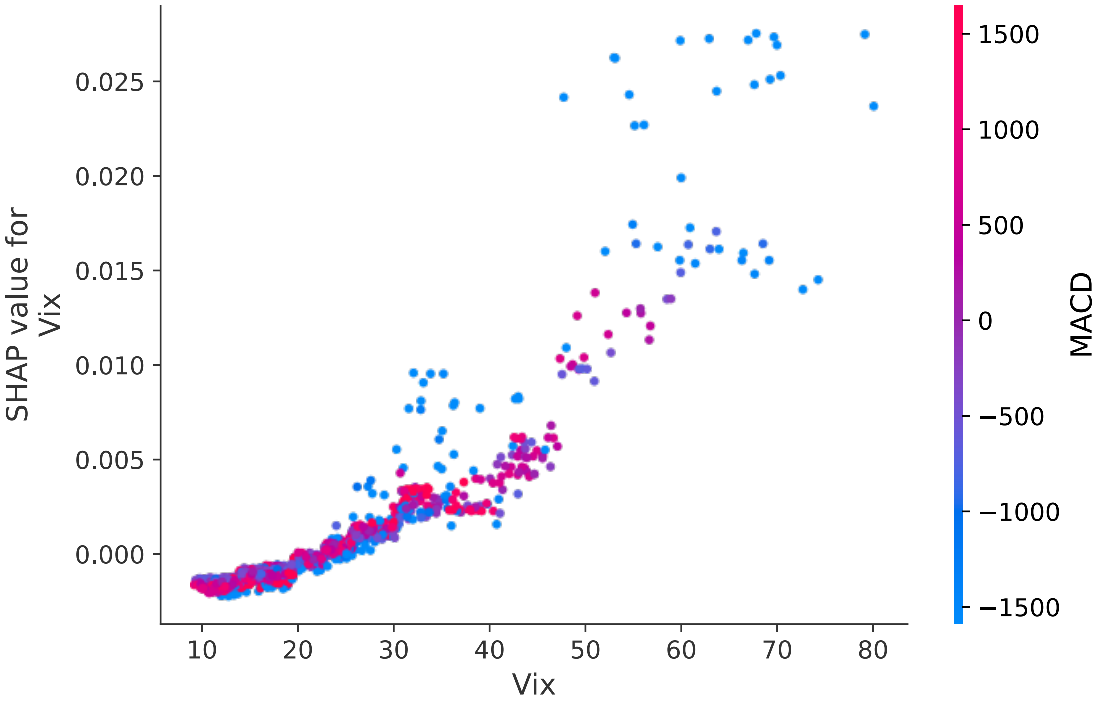


Vix is the most impactful variable for volatility prediction with Gradient Boosting. This result is intuitive since this variable measures volatility using SP500 data, and global financial markets are interconnected. Both high and low values of Vix (represented by red and blue dots, respectively) have a considerable impact on the model's output, evidenced by a wide range of SHAP values. The distribution of Vix values shows relevant influence across the entire range. However, when Vix is low, it probably represents periods of lower market volatility, which might not significantly change the model's output, and this explains why the SHAP values for low Vix are tightly clustered. In contrast, during periods of high volatility, the model's predictions are likely more sensitive and variable, resulting in a broader spread of SHAP values.

Another important feature is the standard deviation of the Close Price. It shows a relevant spread of SHAP values, indicating its strong influence. The distribution of the standard deviation of the close price values affects the model's output consistently, with both high and low values contributing to the predictions. Furthermore, features such as KOSPI, Low, MACD and EMBI+ also influence the model's predictions substantially.

Figure 5.9 illustrates the dependence of SHAP Values on Vix with MACD Interaction.

Figure 5.9 – *Shap Values - Dependence Plot Vix*



A clear positive trend exists between the Vix values and their corresponding SHAP values. The upward slope suggests that the model is highly sensitive to changes in the Vix value, with higher Vix levels contributing more significantly to the predictions.

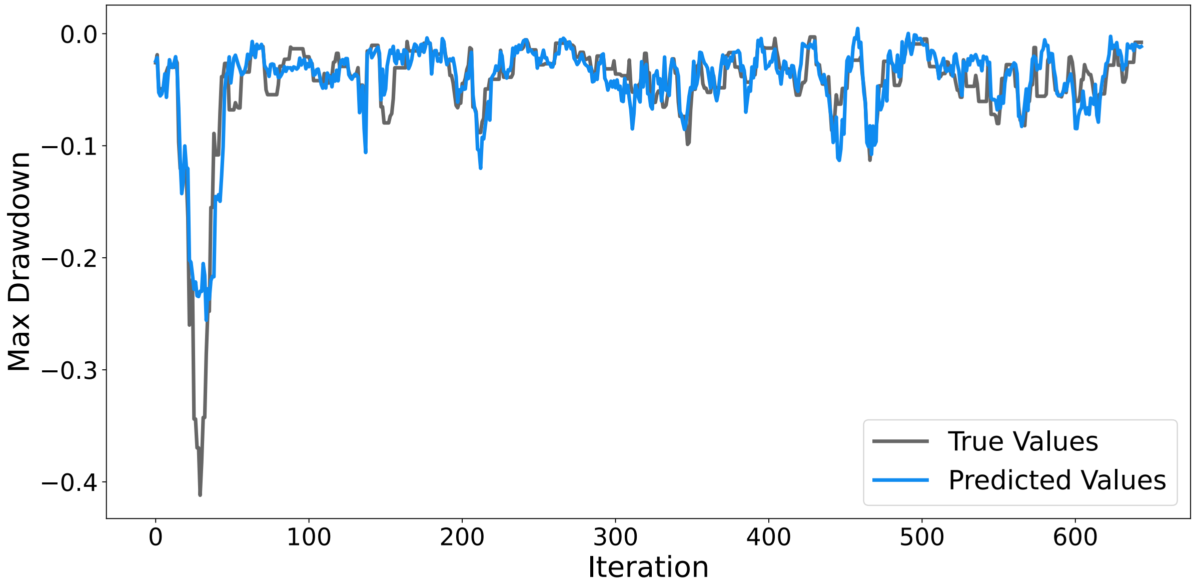
The color of the points represents the value of the MACD feature, with blue indicating low values and red indicating high values. There is a visible interaction between Vix and MACD. For lower Vix values, the color distribution is more mixed, and for higher Vix values, the points tend to be more blue, indicating lower MACD values.

In the lower range of Vix, there is a dense concentration of points around the lower SHAP values, showing that low Vix values have a smaller but consistent impact on the model's output. Still, as Vix values increase, the points spread out more, showing that higher Vix values have a more varied impact on the model's output.

5.3.3 Maximum Drawdown

In this subsection, we analyze the performance of the LightGBM model in predicting maximum drawdown using the Spearman dataset. Figure 5.10 illustrates the comparison between the actual max drawdown values and the predicted values generated by the LightGBM model.

Figure 5.10 – *Comparison of Predicted vs. Actual Maximum Drawdown Values Using LightGBM on the Spearman Dataset*

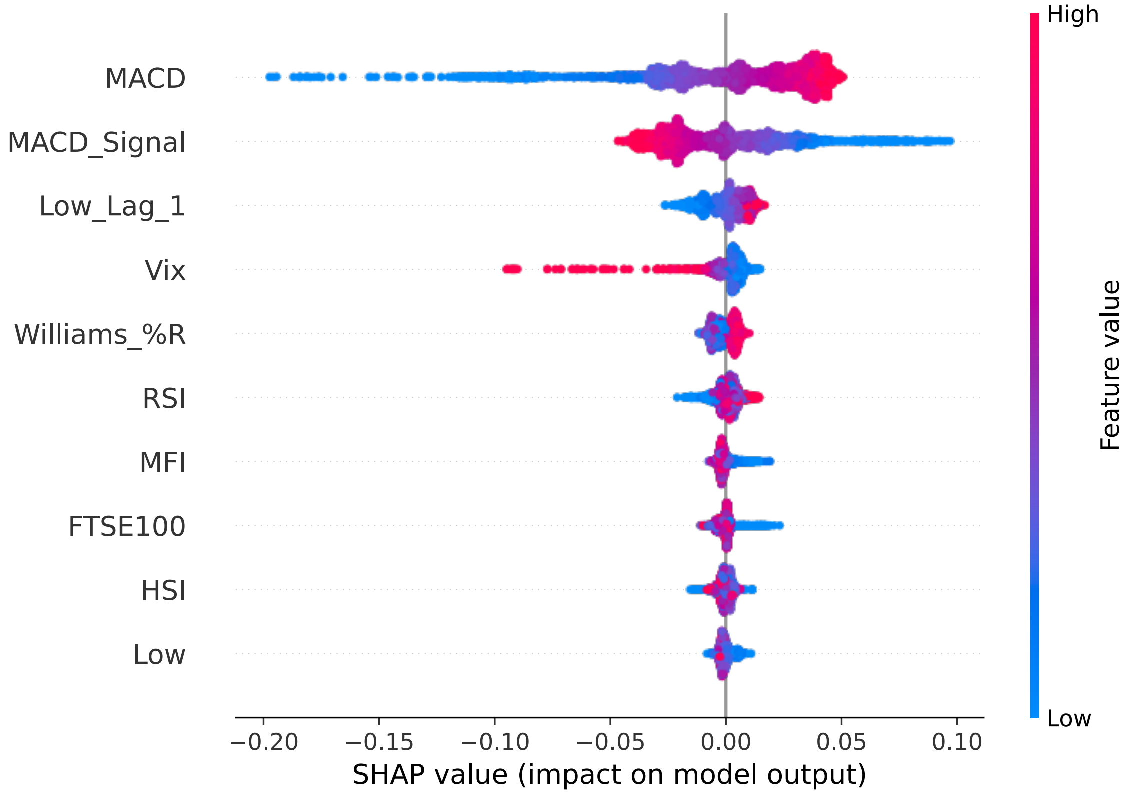


The predicted values (blue line) generally follow the actual values (gray line) trend. This indicates that the LightGBM model effectively captures the Max Drawdown patterns. At the start (first 50 iterations), there is a significant discrepancy between the true and predicted values, similar to the other models. This can be explained similarly to previous instances: although the 2008 crisis was present in the training phase, it was not as extreme as the Covid-19 pandemic. Therefore, the model may not have adequately learned to handle such unprecedented events.

Overall, visually, the model performs well in predicting maximum drawdown, suggesting that the LightGBM model with the Spearman dataset is effective.

Figure 5.11 presents a SHAP values summary plot for the LightGBM model on the Spearman dataset. The most impactful features are at the top.

Figure 5.11 – *Shap Values - Summary Plot for LightGBM on the Spearman Dataset*



The Moving Average Convergence Divergence value (MACD) is the most impactful feature, with high MACD values positively affecting the model's output and low MACD values generally having a negative impact ('negative impact' refers to the direction of the impact, not worsening the model). It makes reasonable sense that low MACD values suggest strong negative momentum, increasing the risk of drawdowns.

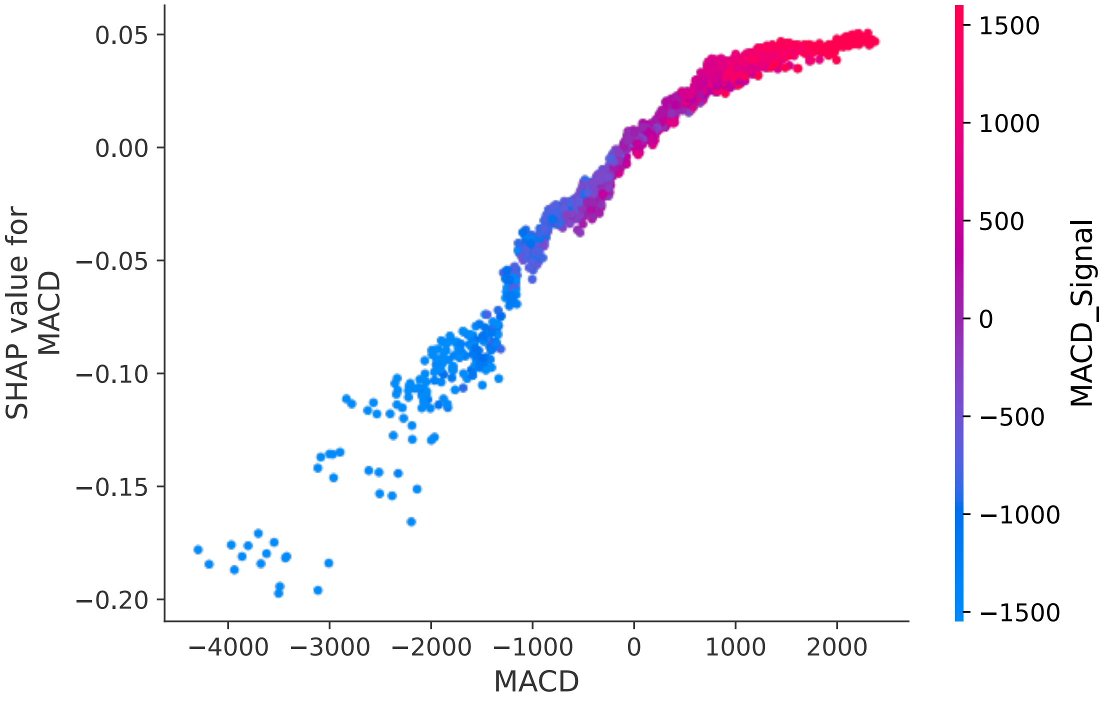
The MACD signal line shows a different trend: high values negatively influence the model's output. High values of the MACD signal line might lead the model to predict a higher risk of drawdown (negative impact) due to potential market corrections.

The lagged value of the 'Low' price demonstrates a balanced impact, with both high and low values influencing the model's predictions in various directions.

Even though Vix is less impactful than MACD's signal line, it has a similar interpretation, where high values negatively influence the model's output. This suggests that higher SP500 volatility leads to lower predictions in the Bovespa drawdown (the target risk measure is negative), which is intuitive.

Figure 5.12 depicts the dependence of SHAP Values on MACD with MACD signal line Interaction.

Figure 5.12 – *Shap Values - Dependence Plot MACD*



A clear positive correlation exists between MACD values and their corresponding SHAP values. As the MACD value increases, the SHAP value also increases, confirming what the summary plot indicated: higher MACD values increase the predicted output for Max Drawdown, while low MACD values decrease it.

In the higher range of MACD, there is a dense concentration of points around the higher SHAP values, showing that high MACD values have a more consistent impact on the model's output. In the lower range of MACD, the data points are more dispersed, but they have a bigger impact on the model's output.

5.4 STATISTICAL ANALYSIS

We validate the significance of the performance differences among algorithms and datasets based on RMSE. Statistical tests were conducted with a significance level of 5%. This section details the results of these tests. Statistical significance is highlighted in red.

5.4.1 Algorithms

5.4.1.1 Levene and Kruskal-Wallis Tests

The results of Levene's and Kruskal-Wallis tests for each target are summarized in Table 16.

Table 16 – *Levene's and Kruskal-Wallis Test Results for Algorithms: P-values*

Target	Levene	Kruskal-Wallis	Significant Differences
CVaR	0.5072	0.1191	No
Volatility	0.0488	-	No
Max Drawdown	0.5839	0.0204	Yes

For CVaR, both Levene's and Kruskal-Wallis's tests indicate no significant differences among algorithms. The Volatility target violated the homoscedasticity assumption (Levene's p-value = 0.0488). Hence, it is not adequate to use the Kruskal-Wallis test for this particular risk measure. For Max Drawdown, significant differences were identified (Kruskal-Wallis p-value = 0.0204), indicating that at least one algorithm stochastically dominates another. However, the Kruskal-Wallis test does not pinpoint where these differences occur. Further post hoc analyses are performed in the following subsection to identify specific pairs of algorithms with significant differences.

5.4.1.2 Post Hoc Analysis

Significant pairwise differences identified by the Conover-Iman test are summarized in Table 17.

Table 17 – *Significant Pairwise Differences from Conover-Iman Post Hoc Tests for Algorithms*

Target	Pairwise Comparison	p-value
Max Drawdown	Adaboost vs. CatBoost	0.0158
	CatBoost vs. LightGBM	0.0473

The Kruskal-Wallis test with the post hoc analysis indicated significant differences between Adaboost, CatBoost, and LightGBM for the Max Drawdown target. This is consistent with the performance metrics presented earlier, where LightGBM and Adaboost outperformed CatBoost.

5.4.2 Datasets

5.4.2.1 Levene and Kruskal-Wallis Tests

The results of Levene's and Kruskal-Wallis tests for each target are summarized in Table 18.

Table 18 – *Levene's and Kruskal-Wallis Test Results for Datasets: P-values*

Target	Levene	Kruskal-Wallis	Significant Differences
CVaR	0.6059	0.0190	Yes
Volatility	0.1530	0.0855	No
Max Drawdown	0.7264	0.0128	Yes

For CVaR and Max Drawdown, significant differences were identified between datasets, indicating that at least one dataset stochastically dominates another for these targets. The following subsection performs further post hoc analysis to identify specific pairs of datasets with significant differences. No significant differences were found for Volatility.

5.4.2.2 Post Hoc Analysis

Significant pairwise differences identified by the Conover-Iman test are summarized in Table 19.

Table 19 – *Significant Pairwise Differences from Conover-Iman Post Hoc Tests*

Target	Pairwise Comparison	p-value
CVaR	Original vs. Spearman	0.0156
Max Drawdown	Combined vs. Vix	0.0362
	Intersection vs. Vix	0.0465
	Spearman vs. Vix	0.0012

Post hoc tests revealed specific pairwise differences, such as between Original and Spearman datasets for CVaR. For Max Drawdown, differences were identified between Combined and Vix, Intersection and Vix, and Spearman and Vix. These findings support the previous results, showing that Spearman and Intersection datasets perform better for these targets, indicating that these datasets' features significantly impact predictive performance.

6 CONCLUSION

The study comprehensively evaluates the performance of different algorithms and datasets for predicting three market risk measures. The analysis revealed that specific algorithms and datasets performed better than others, providing valuable insights for optimizing market risk prediction.

Adaboost and Gradient Boosting were the most effective for specific measures. Adaboost is recommended for predicting CVaR and Max Drawdown, while Gradient Boosting was the better performer in predicting Volatility. LightGBM had consistent performance and computational efficiency, particularly for Max Drawdown, making it a well-rounded solution with reliable results across various risk measures.

The Spearman and Intersection datasets demonstrated superior performance. The former had the best results for predicting CVaR and Max Drawdown. The latter is preferred for Volatility prediction. Conversely, the Original dataset showed higher error metrics and variability, indicating the importance of feature selection. The Spearman correlation method, along with its combination with the Pearson correlation, likely includes more predictive and less noisy features, making it a better choice for model training. The statistical tests confirmed significant differences between the Original and Spearman datasets for CVaR and among the Combined, Intersection, and Spearman datasets compared to the Vix dataset for Max Drawdown.

The market sentiment indicators, particularly Vix, are the most important CVaR and volatility prediction variables. For Max Drawdown, MACD was the most impactful feature, and Vix also strongly influenced model output. Contrary to previous studies, market sentiment and technical indicators were the most important variables for risk prediction in the Brazilian stock market, while macroeconomic variables were found to be less impactful in this analysis.

6.1 LIMITATIONS AND FUTURE WORK

Hyperparameter tuning was performed using Random Search, and this technique might miss the optimal combination of hyperparameters due to its stochastic nature and the predefined search space, potentially leading to suboptimal model configurations.

Overfitting remained a significant challenge in this study despite the implementation of cross-validation, hyperparameter tuning, and regularization techniques. Some models couldn't generalize well to unseen data. Future work should explore additional regularization methods, such as early stopping, and focus on acquiring more data points to improve model generalization. Although CatBoost offers an overfitting detector, it was not utilized in this study. Exploring this feature in future works could be beneficial.

Another relevant direction for future investigation is to conduct studies across different stock market indexes beyond the Brazilian market. Furthermore, comparing boosting algorithms with alternative machine learning models for market risk prediction in the Brazilian context would be valuable.

BIBLIOGRAPHY

ALSHARI, H.; SALEH, A.; ODABAS, A. Comparison of gradient boosting decision tree algorithms for cpu performance. **Erciyes Tip Dergisi**, p. 157–168, 04 2021.

ARAUJO, E.; BRITO, R. D.; SANVICENTE, A. Z. Long-term stock returns in brazil: Volatile equity returns for u.s.-like investors. **International Journal of Finance & Economics**, v. 26, n. 4, p. 6249–6263, 2021. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/ijfe.2118.

ATHEY, S. The impact of machine learning on economics. In: **The economics of artificial intelligence: An agenda**. [S.l.]: University of Chicago Press, 2018. p. 507–547.

BENTÉJAC, C.; CSÖRGŐ, A.; MARTÍNEZ-MUÑOZ, G. A comparative analysis of gradient boosting algorithms. **Artificial Intelligence Review**, Springer, v. 54, p. 1937–1967, 2021.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. **J. Mach. Learn. Res.**, JMLR.org, v. 13, n. null, p. 281–305, feb 2012. ISSN 1532-4435.

BRANCO, R. R.; RUBESAM, A.; ZEVALLOS, M. Forecasting realized volatility: Does anything beat linear models? **Journal of Empirical Finance**, v. 78, p. 101524, 2024. ISSN 0927-5398. Disponível em: https://www.sciencedirect.com/science/article/pii/S0927539824000598.

CHANDRA, A.; CHEN, H.; YAO, X. Trade-off between diversity and accuracy in ensemble generation. In: ______. **Multi-Objective Machine Learning**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 429–464. ISBN 978-3-540-33019-6. Disponível em: https://doi.org/10.1007/3-540-33019-4_19.

CHATZIS, S. P. et al. Forecasting stock market crisis events using deep and statistical machine learning techniques. **Expert Systems with Applications**, v. 112, p. 353–371, 2018. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417418303798.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. **CoRR**, abs/1603.02754, 2016. Disponível em: http://arxiv.org/abs/1603.02754.

CHEN, W.; CHEN, B.; CAI, X. Forecasting china's stock market risk under the background of the stock connect programs. **Soft Computing**, 2023. Disponível em: https://doi.org/10.1007/s00500-023-08496-z.

CHENGSHENG, T.; HUACHENG, L.; BING, X. Adaboost typical algorithm and its application research. In: EDP SCIENCES. **MATEC Web of Conferences**. [S.l.], 2017. v. 139, p. 00222.

COSTA, P. H. S.; BAIDYA, T. K. N. Métodos de medição de risco de mercado: um estudo comparativo. **Production**, SciELO Brasil, v. 13, p. 18–33, 2003.

DRUCKER, H.; SCHAPIRE, R.; SIMARD, P. Boosting performance in neural networks. **International Journal of Pattern Recognition and Artificial Intelligence**, v. 07, n. 04, p. 705–719, 1993. Disponível em: https://doi.org/10.1142/S0218001493000352.

FERREIRA, A. J.; FIGUEIREDO, M. A. T. Boosting algorithms: A review of methods, theory, and applications. In: . [s.n.], 2012. Disponível em: https://api.semanticscholar.org/CorpusID:61106421.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. In: **Proceedings of the Second European Conference on Computational Learning Theory**. Berlin, Heidelberg: Springer-Verlag, 1995. (EuroCOLT '95), p. 23–37. ISBN 3540591192.

FREUND, Y.; SCHAPIRE, R. E. A short introduction to boosting. In: . [s.n.], 1999. Disponível em: https://api.semanticscholar.org/CorpusID:9621074.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). **The Annals of Statistics**, Institute of Mathematical Statistics, v. 28, n. 2, p. 337 – 407, 2000. Disponível em: https://doi.org/10.1214/aos/1016218223.

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. **The Annals of Statistics**, Institute of Mathematical Statistics, v. 29, n. 5, p. 1189–1232, 2001. ISSN 00905364. Disponível em: http://www.jstor.org/stable/2699986.

GOGAS, P.; PAPADIMITRIOU, T. Machine learning in economics and finance. **Computational Economics**, Springer, v. 57, p. 1–4, 2021.

GRINSZTAJN, L.; OYALLON, E.; VAROQUAUX, G. Why do tree-based models still outperform deep learning on typical tabular data? **Advances in neural information processing systems**, v. 35, p. 507–520, 2022.

HASTIE, T. et al. **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer, 2009. v. 2.

HOERL, A. E.; KENNARD, R. W. Ridge regression: Biased estimation for nonorthogonal problems. **Technometrics**, [Taylor Francis, Ltd., American Statistical Association, American Society for Quality], v. 12, n. 1, p. 55–67, 1970. ISSN 00401706. Disponível em: http://www.jstor.org/stable/1267351.

HOTHORN, K. H. T.; ZEILEIS, A. Unbiased recursive partitioning: A conditional inference framework. **Journal of Computational and Graphical Statistics**, Taylor and Francis, v. 15, n. 3, p. 651–674, 2006. Disponível em: https://doi.org/10.1198/106186006X133933.

JAMES, G. et al. **An Introduction to Statistical Learning: with Applications in R**. [S.l.]: Springer Publishing Company, Incorporated, 2014. ISBN 1461471370.

KE, G. et al. Lightgbm: A highly efficient gradient boosting decision tree. In: **Proceedings of the 31st International Conference on Neural Information Processing Systems**. Red Hook, NY, USA: Curran Associates Inc., 2017. (NIPS'17), p. 3149–3157. ISBN 9781510860964.

KEARNS, M.; VALIANT, L. G. Crytographic limitations on learning boolean formulae and finite automata. In: **Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing**. New York, NY, USA: Association for Computing Machinery, 1989. (STOC '89), p. 433–444. ISBN 0897913078. Disponível em: https://doi.org/10.1145/73007.73049.

KOHAVI, R. Bottom-up induction of oblivious read-once decision graphs. In: BERGADANO, F.; RAEDT, L. D. (Ed.). **Machine Learning: ECML-94**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994. p. 154–169. ISBN 978-3-540-48365-6.

LAO, Y. et al. A prediction method based on extreme gradient boosting tree model and its application. **Journal of Physics: Conference Series**, v. 1995, p. 012017, 08 2021.

LEARNED-MILLER, E. G. Introduction to supervised learning. **I: Department of Computer Science, University of Massachusetts**, v. 3, 2014.

MAYR, A. et al. The evolution of boosting algorithms. **Methods of information in medicine**, Schattauer GmbH, v. 53, n. 06, p. 419–427, 2014.

NABIPOUR, M. et al. Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis. **IEEE Access**, v. 8, p. 150199–150212, 2020.

NATEKIN, A.; KNOLL, A. Gradient boosting machines, a tutorial. **Frontiers in Neurorobotics**, v. 7, 2013. ISSN 1662-5218. Disponível em: https://www.frontiersin.org/articles/10.3389/fnbot.2013.00021.

NIELSEN, D. Tree boosting with xgboost - why does xgboost win "every" machine learning competition? In: . [s.n.], 2016. Disponível em: https://api.semanticscholar.org/CorpusID:114191144.

PILTAVER, R. et al. What makes classification trees comprehensible? **Expert Systems with Applications**, v. 62, p. 333–346, 2016. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417416302901.

PROKHORENKOVA, L. et al. Catboost: unbiased boosting with categorical features. In: BENGIO, S. et al. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2018. v. 31. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf.

QIN, R. The construction of corporate financial management risk model based on xgboost algorithm. **Journal of Mathematics**, 2022. Disponível em: https://doi.org/10.1155/2022/2043369.

REWILAK, J. The impact of financial crises on the poor. **Journal of International Development**, Wiley Online Library, v. 30, n. 1, p. 3–19, 2018.

SARYKALIN, S.; SERRAINO, G.; URYASEV, S. Value-at-risk vs. conditional value-at-risk in risk management and optimization. In: **Tutorials in Operations Research**. Hanover: INFORMS, 2008. p. 270–294.

SCHAPIRE, R. E. The strength of weak learnability. **Machine Learning**, v. 5, p. 197–227, 1990. Disponível em: http://dx.doi.org/10.1007/BF00116037.

SO, B. Catboost versus xgboost and lightgbm: Developing enhanced predictive models for zero-inflated insurance claim data. 07 2023.

TIBSHIRANI, R. Regression shrinkage and selection via the lasso. **Journal of the Royal Statistical Society. Series B (Methodological)**, [Royal Statistical Society, Wiley], v. 58, n. 1, p. 267–288, 1996. ISSN 00359246. Disponível em: http://www.jstor.org/stable/2346178.

VALIANT, L. G. A theory of the learnable. **Commun. ACM**, v. 27, n. 11, p. 1134–1142, 1984. Disponível em: http://dblp.uni-trier.de/db/journals/cacm/cacm27.html#Valiant84.

VAPNIK, V. Principles of risk minimization for learning theory. In: **Proceedings of the 4th International Conference on Neural Information Processing Systems**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991. (NIPS'91), p. 831–838. ISBN 1558602224.

VAPNIK, V. N. An overview of statistical learning theory. **IEEE transactions on neural networks**, IEEE, v. 10, n. 5, p. 988–999, 1999.

WANG, W.; SUN, D. The improved adaboost algorithms for imbalanced data classification. **Information Sciences**, v. 563, p. 358–374, 2021. ISSN 0020-0255. Disponível em: https://www.sciencedirect.com/science/article/pii/S0020025521002875.

ZHOU, Z.-H. Ensemble learning. In: ______. **Machine Learning**. Singapore: Springer Singapore, 2021. p. 181–210. ISBN 978-981-15-1967-3. Disponível em: https://doi.org/10.1007/978-981-15-1967-3_8.

A EXPERIMENTAL RESULTS

CONSOLIDATED RESULTS TABLE

Table 20 – *Boosting Model Results for Different Targets and Datasets*

Algorithm	Target	Dataset	RMSE	MAE	Training Time (s)	Prediction Time (s)
Adaboost	CVaR	Original	0.0174	0.0114	19.4908	0.1509
		F Test	0.0134	0.0099	0.4316	0.0000
		Pearson	0.0126	0.0094	0.4802	0.0156
		Spearman	0.0122	0.0088	4.5853	0.1160
		Intersection	0.0120	0.0087	0.4380	0.0157
		Combined	0.0126	0.0097	1.3455	0.0313
		Vix	0.0156	0.0108	0.2629	0.0319
Adaboost	Volatility	Original	0.0092	0.0047	1.7312	0.0156
		F Test	0.0066	0.0050	0.1107	0.0000
		Pearson	0.0081	0.0056	0.0977	0.0000
		Spearman	0.0070	0.0043	0.5292	0.0156
		Intersection	0.0067	0.0042	0.0893	0.0000
		Combined	0.0062	0.0042	0.7851	0.0156
		Vix	0.0075	0.0049	0.1695	0.0156
Adaboost	Max Drawdown	Original	0.0266	0.0194	1.2880	0.0078
		F Test	0.0293	0.0209	0.1100	0.0000
		Pearson	0.0298	0.0212	0.4403	0.0144
		Spearman	0.0265	0.0189	0.9210	0.0312
		Intersection	0.0281	0.0197	0.0970	0.0000
		Combined	0.0290	0.0209	2.9098	0.0807

Continued on next page

Algorithm	Target	Dataset	RMSE	MAE	Training Time (s)	Prediction Time (s)
		Vix	0.0342	0.0233	0.0374	0.0088
Gradient Boosting	CVaR	Original	0.0137	0.0098	6.6231	0.0129
		F Test	0.0145	0.0100	0.3588	0.0000
		Pearson	0.0146	0.0101	0.2371	0.0046
		Spearman	0.0136	0.0095	0.2257	0.0042
		Intersection	0.0136	0.0093	0.1135	0.0167
		Combined	0.0168	0.0123	8.9629	0.0000
		Vix	0.0156	0.0106	0.1082	0.0162
Gradient Boosting	Volatility	Original	0.0064	0.0043	5.5829	0.0198
		F Test	0.0069	0.0046	0.9024	0.0000
		Pearson	0.0068	0.0048	0.1346	0.0000
		Spearman	0.0078	0.0043	0.2185	0.0000
		Intersection	0.0062	0.0041	0.5586	0.0000
		Combined	0.0060	0.0038	2.0172	0.0000
		Vix	0.0075	0.0048	0.0709	0.0067
Gradient Boosting	Max Drawdown	Original	0.0299	0.0194	6.6013	0.0000
		F Test	0.0297	0.0205	4.1252	0.0072
		Pearson	0.0318	0.0220	3.9799	0.0156
		Spearman	0.0273	0.0142	4.0345	0.0167
		Intersection	0.0278	0.0188	1.2585	0.0000
		Combined	0.0292	0.0154	3.8027	0.0000

Continued on next page

Algorithm	Target	Dataset	RMSE	MAE	Training Time (s)	Prediction Time (s)
		Vix	0.0345	0.0223	0.1600	0.0156
XGBoost	CVaR	Original	0.0174	0.0118	0.4540	0.0438
		F Test	0.0150	0.0100	0.0341	0.0077
		Pearson	0.0163	0.0108	0.0300	0.0000
		Spearman	0.0136	0.0095	0.0732	0.0157
		Intersection	0.0146	0.0100	0.0690	0.0000
		Combined	0.0161	0.0105	0.1706	0.0076
		Vix	0.0158	0.0106	0.0593	0.0043
XGBoost	Volatility	Original	0.0116	0.0054	0.0747	0.0120
		F Test	0.0070	0.0042	0.0684	0.0083
		Pearson	0.0097	0.0051	0.0628	0.0020
		Spearman	0.0089	0.0047	0.0682	0.0040
		Intersection	0.0066	0.0042	0.0407	0.0000
		Combined	0.0081	0.0049	0.0547	0.0092
		Vix	0.0086	0.0047	0.0176	0.0045
XGBoost	Max Drawdown	Original	0.0318	0.0188	0.1393	0.0312
		F Test	0.0306	0.0197	0.1148	0.0000
		Pearson	0.0318	0.0221	0.0450	0.0000
		Spearman	0.0260	0.0153	0.0723	0.0080
		Intersection	0.0325	0.0201	0.0296	0.0000
		Combined	0.0318	0.0201	0.0748	0.0145
		Vix	0.0334	0.0224	0.0643	0.0000

Continued on next page

Algorithm	Target	Dataset	RMSE	MAE	Training Time (s)	Prediction Time (s)
LightGBM	CVaR	Original	0.0163	0.0112	0.7154	0.0414
		F Test	0.0156	0.0103	0.0183	0.0000
		Pearson	0.0154	0.0105	0.0227	0.0090
		Spearman	0.0148	0.0096	0.0119	0.0000
		Intersection	0.0151	0.0096	0.0135	0.0039
		Combined	0.0152	0.0106	0.0461	0.0110
		Vix	0.0160	0.0107	0.0165	0.0071
LightGBM	Volatility	Original	0.0081	0.0049	0.7501	0.0110
		F Test	0.0074	0.0047	0.0154	0.0000
		Pearson	0.0072	0.0049	0.0575	0.0080
		Spearman	0.0074	0.0043	0.0193	0.0000
		Intersection	0.0068	0.0042	0.0110	0.0124
		Combined	0.0074	0.0044	0.0483	0.0090
		Vix	0.0076	0.0048	0.0404	0.0044
LightGBM	Max Drawdown	Original	0.0304	0.0176	0.2172	0.0193
		F Test	0.0302	0.0207	0.0905	0.0000
		Pearson	0.0305	0.0211	0.0721	0.0090
		Spearman	0.0239	0.0141	0.0774	0.0156
		Intersection	0.0288	0.0194	0.0633	0.0055
		Combined	0.0254	0.0153	0.0676	0.0081
		Vix	0.0353	0.0224	0.0125	0.0065

Continued on next page

Algorithm	Target	Dataset	RMSE	MAE	Training Time (s)	Prediction Time (s)
CatBoost	CVaR	Original	0.0169	0.0097	1.4517	0.0252
		F Test	0.0143	0.0100	0.1870	0.0006
		Pearson	0.0151	0.0092	0.1766	0.0011
		Spearman	0.0137	0.0091	0.0778	0.0000
		Intersection	0.0152	0.0092	0.3996	0.0157
		Combined	0.0146	0.0105	0.4777	0.0048
		Vix	0.0159	0.0107	0.1092	0.0000
CatBoost	Volatility	Original	0.0076	0.0049	0.5587	0.0096
		F Test	0.0075	0.0046	0.1342	0.0000
		Pearson	0.0069	0.0045	0.1132	0.0038
		Spearman	0.0077	0.0042	0.0795	0.0145
		Intersection	0.0071	0.0041	0.0733	0.0000
		Combined	0.0084	0.0045	0.1251	0.0000
		Vix	0.0075	0.0049	0.0619	0.0000
CatBoost	Max Drawdown	Original	0.0321	0.0187	0.1244	0.0177
		F Test	0.0343	0.0234	0.1740	0.0000
		Pearson	0.0337	0.0230	0.0898	0.0040
		Spearman	0.0323	0.0172	0.1535	0.0000
		Intersection	0.0319	0.0202	0.0483	0.0011
		Combined	0.0332	0.0163	0.3430	0.0160
		Vix	0.0349	0.0224	0.0717	0.0000

B DATA DICTIONARY

VARIABLES TABLE

Table 21 – *Data Dictionary*

Type	**Variable**	**Indicator**	**Definition**
Basic indicators	x_1	Open	The IBOV opening price
	x_2	High	The IBOV highest price
	x_3	Low	The IBOV lowest price
	x_4	Volume	The IBOV trading volume
	x_5	Volume_Var	The percentage change in the IBOV trading volume compared to the previous period
	x_6	Close_Lag_1	The lagged closing price (lag 1)
	x_7	Open_Lag_1	The lagged opening price (lag 1)
	x_8	High_Lag_1	The lagged highest price (lag 1)
	x_9	Low_Lag_1	The lagged lowest price (lag 1)
	x_{10}	Close_Lag_2	The lagged closing price (lag 2)
	x_{11}	Open_Lag_2	The lagged opening price (lag 2)
	x_{12}	High_Lag_2	The lagged highest price (lag 2)
	x_{13}	Low_Lag_2	The lagged lowest price (lag 2)
	x_{14}	Close_Lag_3	The lagged closing price (lag 3)
	x_{15}	Open_Lag_3	The lagged opening price (lag 3)
	x_{16}	High_Lag_3	The lagged highest price (lag 3)
	x_{17}	Low_Lag_3	The lagged lowest price (lag 3)
	x_{18}	Close_Lag_4	The lagged closing price (lag 4)
	x_{19}	Open_Lag_4	The lagged opening price (lag 4)
	x_{20}	High_Lag_4	The lagged highest price (lag 4)
	x_{21}	Low_Lag_4	The lagged lowest price (lag 4)
	x_{22}	Close_Lag_5	The lagged closing price (lag 5)
	x_{23}	Open_Lag_5	The lagged opening price (lag 5)
	x_{24}	High_Lag_5	The lagged highest price (lag 5)
	x_{25}	Low_Lag_5	The lagged lowest price (lag 5)
Technical indicators	x_{26}	Price_Change	The daily difference in closing price
	x_{27}	RSI	The Relative Strength Index

Continued on next page

Type	Variable	Indicator	Definition
	x_{28}	SMA_10	The 10-day Simple Moving Average of the Close Price
	x_{29}	SMA_30	The 30-day Simple Moving Average of the Close Price
	x_{30}	SMA_100	The 100-day Simple Moving Average of the Close Price
	x_{31}	OBV	On Balance Volume
	x_{32}	EMA_10	The 10-day Exponential Moving Average of the Close Price
	x_{33}	EMA_30	The 30-day Exponential Moving Average of the Close Price
	x_{34}	EMA_100	The 100-day Exponential Moving Average of the Close Price
	x_{35}	MACD	Moving Average Convergence Divergence value (EMA_12 - EMA_26)
	x_{36}	MACD_Signal	MACD Signal Line (EMA_9 of MACD)
	x_{37}	CP_Std_Dev	Close Price Standard Deviation
	x_{38}	Middle_Band	Middle Bollinger Band
	x_{39}	Upper_Band	Upper Bollinger Band
	x_{40}	Lower_Band	Lower Bollinger Band
	x_{41}	MFI	Money Flow Index
	x_{42}	Williams_%R	Williams %R
Overseas return rate indicators	x_{43}	SP500	Daily return of the S&P500 index
	x_{44}	DJIA	Daily return of the Dow Jones Industrial Average
	x_{45}	NASDAQ	Daily return of the NASDAQ index
	x_{46}	NYSE	Daily return of the NYSE index
	x_{47}	FTSE100	Daily return of the FTSE100 index
	x_{48}	DAX	Daily return of the DAX index
	x_{49}	CAC40	Daily return of the CAC40 index
	x_{50}	NIKKEI	Daily return of the NIKKEI index
	x_{51}	HSI	Daily return of the HSI index
	x_{52}	ASX200	Daily return of the ASX200 index
	x_{53}	KOSPI	Daily return of the KOSPI index
	x_{54}	TSEC	Daily return of the TSEC index

Continued on next page

Type	Variable	Indicator	Definition
	x_{55}	SSECI	Daily return of the SSE Composite Index
	x_{56}	Shenzhen	Daily return of the Shenzhen Component Index
Market Sentiment Indicators	x_{57}	Vix	Volatility Index (VIX)
	x_{58}	EMBI+	EMBI+ Index
Macroeconomic Indicators	x_{59}	USD/BRL	USD/BRL Exchange Rate
	x_{60}	GBP/BRL	GBP/BRL Exchange Rate
	x_{61}	EUR/BRL	EUR/BRL Exchange Rate
	x_{62}	Gold	Price of Gold
	x_{63}	Silver	Price of Silver
	x_{64}	Crude_Oil	Price of Crude Oil
	x_{65}	Copper	Price of Copper
	x_{66}	Natural_Gas	Price of Natural Gas
	x_{67}	Corn	Price of Corn
	x_{68}	Soy	Price of Soy
	x_{69}	Wheat	Price of Wheat
	x_{70}	Live_Cattle	Price of Live Cattle
	x_{71}	Coffee	Price of Coffee
	x_{72}	Selic_Target	SELIC Target Rate
	x_{73}	Selic_Daily	Daily SELIC Rate
	x_{74}	CDI	CDI Rate
	x_{75}	IMA-S	IMA-S Index
	x_{76}	IRF-M	IRF-M Index
	x_{77}	IMA-B	IMA-B Index
	x_{78}	C_i_C	Currency in Circulation
	x_{79}	Bank_Reserves	Bank Reserves
	x_{80}	R_monet_base	Restricted Monetary Base
Target Variables	–	CVaR_10day	Conditional Value at Risk (CVaR) over 10 days
	–	Std_Dev_10day	Standard Deviation of Returns over 10 days
	–	MDD_10day	Maximum Drawdown over 10 days

C TABLE OF TOOLS

Table 22 – *Tools*

Python library	**Version**
arch	7.0.0
bcb	0.3.0
catboost	1.2.5
ipeadatapy	0.1.9
lightgbm	1.2.5
matplotlib	3.9.0
numpy	1.24.3
pandas	2.2.2
requests	2.31.0
seaborn	0.12.2
scipy	1.10.1
shap	0.45.1
sklearn	1.3.0
statsmodels	0.14.0
xgboost	2.0.3
yfinance	0.2.38

D ADABOOST

D.1 ADABOOST: MULTIPLIER'S EXPRESSION

We want to minimize the exponential loss function $L(\tilde{y}, F(x)) = e^{-\tilde{y}F(x)}$ to seek the additive model $\sum_{t=1}^{T} \beta_t F(x; \theta_t)$. Therefore, with the AdaBoost algorithm, we minimize a convex loss function over a convex set of functions. When we were discussing AdaBoost's optimization problem, we saw that the loss function can be expressed as

$$L = e^{-\beta} \sum_{i:\tilde{y}_i = F(x_i,\theta)} \omega_{it} + e^{\beta} \sum_{i:\tilde{y}_i \neq F(x_i,\theta)} \omega_{it}$$

Hence, to determine β that minimizes the loss, it is sufficient to differentiate L with respect to β, set it to zero and solve for β:

$$\frac{\mathrm{d}L}{\mathrm{d}\beta} = \frac{\mathrm{d}(e^{-\beta} \sum_{i:\tilde{y}_i = F(x_i,\theta)} \omega_{it} + e^{\beta} \sum_{i:\tilde{y}_i \neq F(x_i,\theta)} \omega_{it})}{\mathrm{d}\beta} = 0$$

$$-e^{-\beta} \sum_{i:\tilde{y}_i = F(x_i,\theta)} \omega_{it} + e^{\beta} \sum_{i:\tilde{y}_i \neq F(x_i,\theta)} \omega_{it} = 0$$

$$e^{-\beta} \sum_{i:\tilde{y}_i = F(x_i,\theta)} \omega_{it} = e^{\beta} \sum_{i:\tilde{y}_i \neq F(x_i,\theta)} \omega_{it}$$

$$-\beta + \ln(\sum_{i:\tilde{y}_i = F(x_i,\theta)} \omega_{it}) = \beta + \ln(\sum_{i:\tilde{y}_i \neq F(x_i,\theta)} \omega_{it})$$

$$-2\beta = \ln(\frac{\sum_{i:\tilde{y}_i \neq F(x_i,\theta)} \omega_{it}}{\sum_{i:\tilde{y}_i = F(x_i,\theta)} \omega_{it}})$$

$$\beta = \frac{1}{2} \ln(\frac{\sum_{i:\tilde{y}_i = F(x_i,\theta)} \omega_{it}}{\sum_{i:\tilde{y}_i \neq F(x_i,\theta)} \omega_{it}})$$

The weighted error rate is $err_t = \frac{\sum_{i=1}^{n} \omega_i \mathbb{1}_{[\tilde{y}_i \neq F(x_i;\hat{\theta}_t)]}}{\sum_{i=1}^{n} \omega_i}$, then

$$\beta = \frac{1}{2} \ln(\frac{1 - err_t}{err_t})$$

$\square$

D.2 ADABOOST: CONVERGENCE IN TRAINING

This section focuses on analyzing the AdaBoost algorithm's convergence rate, noting that it achieves zero training error. To do so, we assume that the basis function F is a weak learner and demonstrate that the training accuracy of AdaBoost increases exponentially fast. Essentially, we find explicit expressions for the loss function at the periods t and $t-1$; examine the weak learner F; compare both loss expressions; and ultimately confirm that the training error decays exponentially fast.

Firstly, we define

$$W_t^+ = \sum_{i:\tilde{y_i}=F(x_i,\hat{\theta})} \omega_{it} \ , \ W_t^- = \sum_{i:\tilde{y_i}\neq F(x_i,\hat{\theta})} \omega_{it}$$

Then, substitute the result from the last section $\beta = \frac{1}{2}\ln\left(\frac{\sum_{i:\tilde{y_i}=F(x_i,\hat{\theta})}\omega_{it}}{\sum_{i:\tilde{y_i}\neq F(x_i,\hat{\theta})}\omega_{it}}\right) = \frac{1}{2}\ln\left(\frac{W_t^+}{W_t^-}\right)$ in the loss expression

$$\begin{aligned}
L_t(\hat{\beta}_t,\hat{\theta}_t) &= e^{-\frac{1}{2}\ln\left(\frac{\sum_{i:\tilde{y_i}=F(x_i,\hat{\theta})}\omega_{it}}{\sum_{i:\tilde{y_i}\neq F(x_i,\hat{\theta})}\omega_{it}}\right)} \sum_{i:\tilde{y_i}=F(x_i,\hat{\theta})} \omega_{it} + e^{\frac{1}{2}\ln\left(\frac{\sum_{i:\tilde{y_i}=F(x_i,\hat{\theta})}\omega_{it}}{\sum_{i:\tilde{y_i}\neq F(x_i,\hat{\theta})}\omega_{it}}\right)} \sum_{i:\tilde{y_i}\neq F(x_i,\hat{\theta})} \omega_{it} \\
&= e^{-\frac{1}{2}\ln\left(\frac{W_t^+}{W_t^-}\right)} W_t^+ + e^{\frac{1}{2}\ln\left(\frac{W_t^+}{W_t^-}\right)} W_t^- \\
&= W_t^+\sqrt{\frac{W_t^-}{W_t^+}} + W_t^-\sqrt{\frac{W_t^+}{W_t^-}} \\
&= 2\sqrt{W_t^+W_t^-}
\end{aligned}$$

And it will be useful to note that, given $\hat{f}_{t-1}(x_i) = \sum_{k=1}^{t-1}\hat{\beta}_k F(x_i,\hat{\theta}_k)$ and the value $\omega_{it} = e^{-\tilde{y_i}\hat{f}_{t-1}(x_i)}$, the loss function at the iteration $t-1$ will be

$$\begin{aligned}
L_{t-1}(\hat{\beta}_{t-1},\hat{\theta}_{t-1}) &= \sum_{i=1}^{n} e^{-\tilde{y_i}\hat{f}_{t-1}(x_i)} \\
&= \sum_{i:\tilde{y_i}=F(x_i,\hat{\theta})} \omega_{it} + \sum_{i:\tilde{y_i}\neq F(x_i,\hat{\theta})} \omega_{it} \\
&= W_t^+ + W_t^-
\end{aligned}$$

Recalling the weak learner definition and observing the probability distribution $p_i = \frac{\omega_{it}}{W_t^+ + W_t^-}$, if $F(\cdot,\theta)$ is a γ-weak algorithm there exists a θ that

$$\begin{aligned}\sum_{i=1}^{n} p_i \mathbb{1}_{\{F(x_i;\hat{\theta})\neq \tilde{y}_i\}} &\leq \frac{1}{2} - \gamma \\ \sum_{i=1}^{n} \frac{\omega_{it}\mathbb{1}_{\{F(x_i;\hat{\theta})\neq \tilde{y}_i\}}}{W_t^+ + W_t^-} &\leq \frac{1}{2} - \gamma \\ \frac{1}{W_t^+ + W_t^-} \sum_{i:F(x_i;\hat{\theta})\neq \tilde{y}_i} \omega_{it} &\leq \frac{1}{2} - \gamma\end{aligned}$$

Moreover, we assume that $\hat{\theta}_t$ can be $\hat{\theta}$ - the estimated parameter at time t may be the best parameter throughout the iterative process. Therefore,

$$\begin{aligned}\frac{W_t^-}{W_t^+ + W_t^-} &\leq \frac{1}{2} - \gamma \\ W_t^- &\leq \frac{1}{2}(W_t^+ + W_t^-) - \gamma(W_t^+ + W_t^-) \\ (\frac{1+2\gamma}{1-2\gamma})W_t^- &\leq W_t^+\end{aligned}$$

When we substitute $\hat{\beta} = \frac{1}{2}\ln(\frac{1+2\gamma}{1-2\gamma})$ into the minimum definition of L_t we get

$$\begin{aligned}L_t(\hat{\beta}_t, \hat{\theta}_t) &\leq W_t^+\sqrt{\frac{1-2\gamma}{1+2\gamma}} + W_t^-\sqrt{\frac{1+2\gamma}{1-2\gamma}} \\ &\leq W_t^+\sqrt{\frac{1-2\gamma}{1+2\gamma}} + W_t^-(1-2\gamma+2\gamma)\sqrt{\frac{1+2\gamma}{1-2\gamma}} \\ &\leq W_t^+\sqrt{\frac{1-2\gamma}{1+2\gamma}} + W_t^-(1-2\gamma)\sqrt{\frac{1+2\gamma}{1-2\gamma}} + W_t^-(2\gamma)\sqrt{\frac{1+2\gamma}{1-2\gamma}} \\ &\leq W_t^+\sqrt{\frac{1-2\gamma}{1+2\gamma}} + W_t^-(1-2\gamma)\sqrt{\frac{1+2\gamma}{1-2\gamma}} + W_t^+(2\gamma)\frac{1-2\gamma}{1+2\gamma}\sqrt{\frac{1+2\gamma}{1-2\gamma}} \\ &\leq W_t^+\sqrt{\frac{1-2\gamma}{1+2\gamma}} + W_t^-(1-2\gamma)\sqrt{\frac{1+2\gamma}{1-2\gamma}} + W_t^+(2\gamma)\sqrt{\frac{1-2\gamma}{1+2\gamma}} \\ &\leq W_t^+(1+2\gamma)\sqrt{\frac{1-2\gamma}{1+2\gamma}} + W_t^-(1-2\gamma)\sqrt{\frac{1+2\gamma}{1-2\gamma}} \\ &\leq W_t^+\sqrt{1-4\gamma^2} + W_t^-\sqrt{1-4\gamma^2} \\ &\leq (W_t^+ + W_t^-)\sqrt{1-4\gamma^2} \\ &\leq \sqrt{1-4\gamma^2}L_{t-1}(\hat{\beta}_{t-1}, \hat{\theta}_{t-1})\end{aligned}$$

Thus, $L_t(\hat{\beta}_t, \hat{\theta}_t) \leq \sqrt{1-4\gamma^2}L_{t-1}(\hat{\beta}_{t-1}, \hat{\theta}_{t-1})$. Using this result and a initial value $\hat{\beta}_0 = 0$, such that $L_0(\hat{\beta}_0, \hat{\theta}_0) = 1$, we observe that the training error is decreasing exponentially

$$
\begin{aligned}
L_1(\hat{\beta}_1, \hat{\theta}_1) &\leq \sqrt{1-4\gamma^2} L_0(\hat{\beta}_0, \hat{\theta}_0) = \sqrt{1-4\gamma^2} \\
L_2(\hat{\beta}_2, \hat{\theta}_2) &\leq \sqrt{1-4\gamma^2} L_1(\hat{\beta}_1, \hat{\theta}_1) = (\sqrt{1-4\gamma^2})^2 \\
&\vdots \\
L_t(\hat{\beta}_t, \hat{\theta}_t) &\leq \sqrt{1-4\gamma^2} L_{t-1}(\hat{\beta}_{t-1}, \hat{\theta}_{t-1}) = (\sqrt{1-4\gamma^2})^t \\
L_t(\hat{\beta}_t, \hat{\theta}_t) &\leq (1-4\gamma^2)^{\frac{t}{2}}
\end{aligned}
$$

Noting that $1 + x \leq e^x \;\; \forall x \in \mathbb{R}$, we conclude

$$
\begin{aligned}
1 + x &\leq e^x \\
1 + (-4\gamma^2) &\leq e^{-4\gamma^2} \\
(1-4\gamma^2)^{\frac{t}{2}} &\leq e^{-2t\gamma^2} \\
L_t(\hat{\beta}_t, \hat{\theta}_t) &\leq (1-4\gamma^2)^{\frac{t}{2}} \leq e^{-2t\gamma^2} \\
L_t(\hat{\beta}_t, \hat{\theta}_t) &\leq e^{-2t\gamma^2}
\end{aligned}
$$

□

Completing the analysis of the algorithm's convergence, asserting that the training error with AdaBoost's exponential loss function decays exponentially fast.