Facultad de Ciencias Economicas y Empresariales, ICADE

PREDICTING BOND PRICE CHANGES USING MACHINE LEARNING AND FACTOR ANALYSIS

Author: Álvaro Olivié Molina

Director: María Coronado Vaca

MADRID | June 2024

PREDICTING BOND PRICE CHANGES USING MACHINE LEARNING AND FACTOR ANALYSIS

Author: Álvaro Olivié Molina
Director: María Coronado Vaca
Collaborating Entity: ICADE– Universidad Pontificia Comillas

ABSTRACT

In this final thesis, a thorough investigation was first undertaken to understand the current use of machine learning in finance, with a special interest in how machine learning models are being used to identify trends and factors in the bond market. The focus of this study was to predict changes in the price of bonds at different horizons (1, 3, 6, 12 months). Multiple models have been developed to then compare and analyse results. These models were then used to generate portfolios to see a real-world application.

The data used was collected from Bloomberg and shows the High Yield and Investment Grade US bond markets from 2014 to 2021. The data was then stored in a CSV and separated into train and test datasets so that a portfolio could be tested on data that no model has seen. An initial exploration of the data was performed that allows for a deeper understanding of the data and visualizing relationships between features.

Multiple models were then trained using the train dataset. These models included Linear Regression, Random Forest, Support Vector Regression, Gradient Boosting, and Neural Networks. All models were implemented in Python using specialized libraries to facilitate development. A final ensemble model was developed that uses all previous model predictions.

Hyperparameter tuning was performed, along with cross-validation, to optimize performance and help determine their real-world performance. All models were measured with the same metrics to make comparisons. These metrics are Mean Squared Error (MSE), R squared (R2), and Hit Ratio. The results are then compared to identify which models performed better.

All models were able to find trends in the data with the Neural Network and ensemble model standing out as the most consistent. In addition to the model evaluation, a proof of concept was implemented in the form of a portfolio using the models at the 1-month horizon and then tested on the test dataset.

In conclusion, this work contributes to the field by providing a detailed comparison of multiple models applied to the bond market. The study also demonstrates a real-work application by simulating a portfolio.

Key Words: Machine Learning, Bond Market, Factor Analysis, Portfolio Simulation

PREDICCIÓN DE LA EVOLUCIÓN DE LOS PRECIOS DE LOS BONOS MEDIANTE MACHINE LEARNING Y ANÁLISIS FACTORIAL

Autor: Álvaro Olivié Molina
Director: María Coronado Vaca
Entidad Colaboradora: ICADE– Universidad Pontificia Comillas

ABSTRACT

En este trabajo de fin de carrera, primero se llevó a cabo una investigación exhaustiva para comprender el uso actual del aprendizaje automático en las finanzas, con especial interés en cómo se están utilizando los modelos de *machine learning* para identificar tendencias y factores en el mercado de bonos. El objetivo de este estudio era predecir los cambios en el precio de los bonos en diferentes horizontes (1, 3, 6, 12 meses). Se han desarrollado múltiples modelos para luego comparar y analizar los resultados. Estos modelos se utilizaron después para generar carteras y ver una aplicación en el mundo real.

Los datos utilizados proceden de Bloomberg y muestran los mercados de bonos estadounidenses de *High Yield* e *Investment Grade* desde 2014 hasta 2021. Los datos se almacenaron en un CSV y se separaron en conjuntos de datos de entrenamiento y de prueba para poder probar una cartera con datos que ningún modelo ha visto. Se realizó una exploración inicial de los datos que permite una comprensión más profunda de los datos y la visualización de las relaciones entre las variables.

A continuación, se entrenaron varios modelos utilizando el conjunto de datos de entrenamiento. Estos modelos incluían la Regresión Lineal, el Bosque Aleatorio, la Regresión de Vectores de Soporte, el Refuerzo de Gradiente y las Redes Neuronales. Todos los modelos se implementaron en Python utilizando librerias especializadas para facilitar el desarrollo. Se desarrolló un modelo de aprendizaje por conjuntos que utiliza todas las predicciones de los modelos anteriores.

Se realizó un ajuste de hiperparámetros, junto con una validación cruzada, para optimizar el rendimiento y ayudar a determinar su rendimiento en el mundo real. Todos los modelos se midieron con las mismas métricas para realizar comparaciones. Estas métricas son el error cuadrático medio (ECM), R-cuadrado (R2) y el *Hit Ratio*. A continuación se comparan los resultados para determinar qué modelos funcionan mejor.

Todos los modelos fueron capaces de encontrar tendencias en los datos, destacando la red neuronal y el modelo ensemble como los más consistentes. Además de la evaluación de los modelos, se implementó una prueba de concepto en forma de cartera utilizando los modelos implementados en el horizonte de 1 mes y se probó en el conjunto de datos de prueba.

En conclusión, este trabajo proporciona una comparación detallada de múltiples modelos aplicados al mercado de bonos. El estudio también demuestra una aplicación real mediante la simulación de una cartera.

Palabras clave: Machine Learning, Mercado de Bonos, Análisis Factorial, Simulación de Carteras

Index

Figures Index

Table Index

Chapter 1. INTRODUCTION

In recent years, the rise of machine learning has allowed researchers to analyse complex, nonlinear relationships in financial data. Despite significant advances in applying machine learning techniques to equity markets, the bond market has received less attention. This study seeks to bridge that gap by developing and validating models that can predict bond price changes over various horizons using machine learning methods.

By focusing on key bond characteristics such as current price, yield to maturity, and index rating, we aim to demonstrate the use of advanced predictive models in enhancing decision-making for bond investments. This study contributes to the existing literature by applying machine learning to bond markets and offers a practical application of these models.

1.1 OBJECTIVES

The primary objective of this research is to develop and validate models that can predict the change in prices of bonds at various horizons—1, 3, 6, and 12 months—based on key bond characteristics. Specifically, we focus on factors such as current price, yield to maturity, maturity, weight, coupon, bond class, and index rating, and we ignore time data. By employing a range of machine learning methods, including regression, Support Vector Machines (SVM), Random Forests, Gradient Boosting, and neural networks, we aim to determine how effective these techniques are at capturing the complex, nonlinear relationships in bond markets.

A secondary objective is to explain how these models could be used to create a fixed-income portfolio that trades based on the predictions made at each horizon. This involves using the model's forecasts to make informed decisions about buying, selling, or holding various bonds based solely on the possible change in price. The goal is to demonstrate how advanced predictive models can help the decision-making process when investing in bonds.

1.2 MOTIVATION

The motivation to undertake this research comes from two different angles. As we will see when commenting on the prior work done in the field, little has been done in terms of applying machine learning to the bond market. Equities, as expected, has received a lot of attention and so a lot of research has been done. With this study we plan to reduce that lack of knowledge.

The other angle is the data itself. We have come across a very rich dataset that allows us to test multiple models and approaches. This dataset in and of itself is a large motivation as rarely do researchers find an opportunity similar to this.

1.3 METHODOLOGY

The following methodology has been developed to reach the desired objective. The first step has been to study the current literature on the subject regarding asset pricing, with a specific focus on how machine learning has been applied to this field in the past. The focus of this study was to identify if machine learning has been applied to the bond market and in if so, how. The literature also identifies how portfolios have been built and how this is done in the bond market.

Then, the data used in this paper has been gathered and cleaned, and a wide range of machine learning methods have been proposed. Three different metrics have been chosen to compare the models. The models where then trained and tested on the dataset.

The results for all the models and their explanation where then presented. The models have shown how machine learning, applied to bond data, can provide meaningful results. A final model has been developed to join all previous models in what is known as an ensemble model.

Finaly, a portfolio is developed as a proof of concept. This portfolio is tested using a test dataset over a one-year period with revisions every month. Conclusions are then drawn from the results.

1.4 STRUCTURE

This study has been divided into six key components. The first part provides an analysis of what has been done in the field to get a better understanding of the next steps. The second component focuses the tools and technologies used during the study. The third part discusses the dataset used and how it has been manipulated to fit the needs of the project. The fourth portion explains the different models to be used and how each model was trained. This part also explains the key metrics used by all models. The fifth section comments the results of each of the models and the sixth component of this study focuses on how a portfolio could be built based on the findings. Finaly, the study ends with some conclusions and what the next steps could be.

Chapter 2. PRIOR ART

2.1 INTRODUCTION

The objective of this chapter is to review the existing literature on bond price prediction, factor models and the use of different machine learning models in financial markets. The aim is, by looking at previous literature, identify gaps, similarities and differences that help push the development of the rest of this study.

This review looks at studies like Fama and French (1992) and Carhart (1997) that identify factors that affect asset returns. This chapter also includes sections that focus on more recent advancements in machine learning and big data in finance. Some examples of studies that share this focus are Bali er al. (2020) and Kaufmann et al. (2020).

This review of the literature wishes to highlight how current models and techniques can be applied to the bond market to predict bond price changes at various horizons. It also comments on the importance of back testing and the benefits of a practical application through the development of a portfolio.

The chapter is divided into 3 main sections; Fundamental Factor Models in Finance, Advanced Machine Learning Techniques, Validation & Ensuring Integrity. The first section analyses the work done in multifactor models and how they have been applied to asset pricing. The second segment focuses on the use of machine learning and advanced data techniques to increase the accuracy and effectiveness of financial models. The final section centres on validation and how to avoid overfitting.

2.2 FUNDAMENTAL FACTOR MODELS IN FINANCE

Exploring fundamental factor models in finance provides a strong foundation for understanding asset pricing. These models have been pivotal in identifying the key factors

that influence returns, such as size, value, and momentum. This first section reviews papers that have contributed significantly to this field.

2.2.1 FOUNDATIONAL MULTIFACTOR MODELS

The work by Fama and French (1992) identifies size and book-to-market equity as key factors in explaining cross-sectional variations in stock returns. The model proposed by Fama and French (1992) shows a linear regression using their defined factors. Their model demonstrates that smaller firms and firms with higher book-to-market ratios tend to have higher average returns. This challenged the traditional CAPM by incorporating multidimensional risk factors.

This study shows the how multifactor models help capture the complexities of asset pricing. This research, even though it focuses on equities, is instrumental in creating a strong foundation for this project as factors like size and value are analogous to features like market value and credit ratings which will be looked at in this study.

2.2.2 MOMENTUM AND PERFORMANCE PERSISTENCE

Carhart (1997) then extends this framework by showing that mutual fund performance persistence can be attributed to momentum, alongside market, size, and value factors. Much like the previous study, there is a symmetry in the factors used in this study and those proposed later for the bond market. This study also highlights the significance of considering transaction costs and other expenses.

Another important piece of literature is Jegadeesh and Titman (1993). In this study the authors validate the effectiveness of momentum strategies in stocks and demonstrate that stocks with strong past performance continue to perform well in the short term. Jegadeesh and Titman (2001) confirm the persistence of momentum profits in the short term but also warns of long-term reversals and suggest focusing on short-term horizons. This focus on short-term horizons is reflected in this study but applied to bonds. The study also mentions validating with out-of-sample data which will is also applicable to this study.

Fama and French (2012) further reinforce the global applicability of size, value, and momentum factors across different regions and so validating the robustness of their model. The fact that the model can generalize to many regions helps validates their theory and motivates the search for a similar results in the bond market.

2.2.3 ADAPTING TO BOND MARKETS

Size, value, and momentum are three equity variables that have been successfully applied to the corporate bond market, as shown by Houweling and van Zundert (2017). They illustrate the potential for cross-asset factor investing by demonstrating how these factors can be modified to predict bond returns. The foundation for including equity factors in bond market models is established by this study. This is furthered by Zaremba (2017), which confirms the existence of significant momentum effects in government bonds and proposes that momentum portfolios, which are widely used in equities markets, can be modified for use in bond markets.

2.2.4 REAL-WORLD IMPLEMENTATION

BlackRock's approach to factor investing in bonds uses the same principles that have proven successful in equities like identifying persistent drivers of return that can be exploited. By focusing on factors such as value, momentum, quality, and size, their fixed-income portfolios are constructed to harness these return drivers across various bond sectors. This strategy enhances returns while managing risk more effectively as shown by Parker et al. (2023). As seen in Figure 1: Systematic fixed income ETFs relative to broad market benchmarks and peers (Source: Parker et al., 2023)

Figure 2: Seasonal Decomposition for the Average Price of Bonds (Source: Own elaboration)Figure 1, their ETF iShares High Yield Systematic Bond ETF (HYDB) consistently outperformed the market and other ETF's.

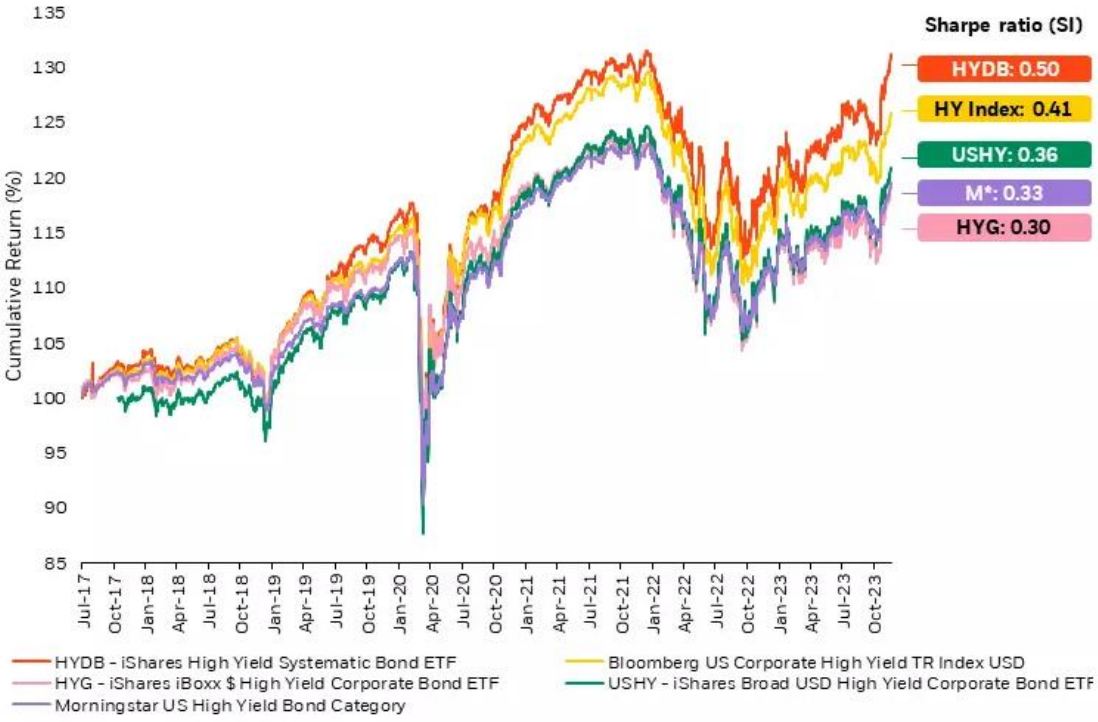

Figure 1: Systematic fixed income ETFs relative to broad market benchmarks and peers (Source: Parker et al., 2023)

Figure 2: Seasonal Decomposition for the Average Price of Bonds (Source: Own elaboration)Figure 3: Systematic fixed income ETFs relative to broad market benchmarks and peers (Source: Parker et al., 2023)

Their portfolios have outperformed traditional benchmarks by consistently applying these factor strategies, which involve rebalancing to maintain exposure to desired factors and dynamically adjusting allocations based on market conditions. This approach captures the long-term benefits of factor investing and provides resilience through diversified exposure. This demonstrates the importance of using robust factors and being adaptable to optimize bond portfolio performance.

2.2.5 CONCLUSION

All in all, these studies collectively establish the importance of multifactor models in asset pricing, emphasizing the roles of size, value, and momentum. They address key questions

about the drivers of asset returns, the persistence of performance factors, and the impact of transaction costs. By adapting these factors to bond markets, this project aims to enhance bond price prediction models.

In conclusion, these articles show that credit ratings may be used to represent the value component and that the size factor can be modified to match market value. Using 1, 3, 6, or 12-month horizons, momentum strategies can be used for short-term bond price predictions. The last BlackRock example demonstrates the practicality of these approaches. The fact that Carhart (1997) included transaction fees emphasizes how crucial it is to include real-world restrictions in predictive models. When working with short-term transactions, these expenses must be considered because they might rise quickly. Furthermore, Fama and French (2012)'s illustration of the universal application of these elements shows that they can be adjusted to global bond markets.

2.3 ADVANCED MACHINE LEARNING TECHNIQUES

In the realm of machine learning applied to finance, several studies illustrate the potential of sophisticated models in predicting bond prices and enhancing investment strategies. Grouped by their focus, these studies provide insights into using ensemble methods, specialized algorithms, and deep learning.

2.3.1 ENSEMBLE METHODS

Kaufmann, Messow, and Vogt (2020) and Bali et al. (2020) demonstrate the power of ensemble methods. Kaufmann et al. (2020) enhance momentum strategies in corporate bonds using Boosted Regression Trees. Their approach significantly improves the performance of traditional momentum strategies by capturing nonlinear relationships and interactions between different factors.

In addition, Bali et al. (2020) highlights the effectiveness of using extensive datasets and advanced techniques like Random Forests and Gradient Boosting to predict bond returns.

Both studies show the significant improvements in predictive accuracy when using machine learning and big data, laying a strong foundation for their application in bond markets.

2.3.2 SPECIALIZED ALGORITHMS

Poh et al. (2021) introduce Learning-to-Rank (LTR) algorithms to improve asset ranking and selection in cross-sectional strategies. By learning pairwise and listwise relationships, LTR algorithms offer more accurate asset rankings, leading to better-performing portfolios. Adapting this approach to rank bonds based on predicted price changes can optimize portfolio construction and performance.

In the 2021 study, Bali et al. (2021), the authors explore return predictability across stocks and bonds, emphasizing the need for tailored models for different asset classes. They highlight distinct predictability drivers for stocks and bonds, supporting the development of specialized machine learning models focused on bond-specific return drivers. These comprehensive approaches show the necessity of integrating diverse datasets and sophisticated techniques to enhance bond price prediction models. Both studies highlight the importance of using advanced algorithms to enhance predictive accuracy and investment decision-making.

2.3.3 NEURAL NETWORKS

Lim, Zohren, and Roberts (2021) apply deep neural networks to time-series momentum strategies. Their Deep Momentum Networks (DMNs) automatically learn trend estimation and position sizing, optimizing for risk-adjusted performance metrics like the Sharpe ratio. This study illustrates the potential of deep learning techniques to capture complex patterns in the data, providing a more accurate prediction of bond prices. The use of neural networks in bond price prediction can significantly enhance the effectiveness of trading strategies.

2.3.4 CONCLUSION

These studies collectively illustrate the potential of advanced machine learning techniques in finance. By utilizing ensemble methods, specialized algorithms, and deep learning, we

can address key challenges in predicting bond prices, such as capturing nonlinear relationships and working with large amounts of data. In their book, "Machine Learning for Factor Investing: Python version" (Coqueret and Guida, 2023) the authors provide a framework that helps researchers apply these multiple models as they have done for equities.

Integrating these advanced methodologies into a model that predicts the changes in bond prices allows the models to capture the intricate and nonlinear relationships within bond markets, leading to more accurate and reliable predictions.

2.4 VALIDATION & ENSURING INTEGRITY

The importance of robust validation and avoiding overfitting in predictive models cannot be overstated. To ensure that the models developed for bond price prediction are reliable and applicable in real-world scenarios, it is crucial to adopt rigorous validation techniques.

2.4.1 TESTING & VALIDATION

Arnott, Harvey, and Markowitz (2020) emphasize the significance of having robust back testing protocols to avoid overfitting in machine learning models. The study comments on the need of using realistic and comprehensive validation techniques to ensure the reliability and robustness of predictive models. It is necessary to apply these techniques into any study that works with machine learning models as it will validate the reliability of the results. The study outlines the risks associated with overfitting, where models perform well on historical data but fail to generalize to new, unseen data. To tackle this, the authors propose the use of out-of-sample testing, cross-validation and window analysis to test each model.

Coqueret and Guida (2023) also provides practical guidance on implementing machine learning techniques properly. Their methodology emphasizes the importance of rigorous testing and validation, consistent with the principles outlined by Arnott, Harvey, and Markowitz (2020).

Bali et al. (2020) leverages big data and machine learning to enhance the predictive accuracy of bond returns. The study has a strong emphasis on high-quality datasets to produce accurate results. The study also pushes for large datasets so that overfitting becomes less of an issue.

2.4.2 CONCLUSION

By taking insights from these articles, we can develop various machine learning models, that through robust back testing, high-quality datasets, and practical portfolio applications, are not only theoretically sound but also practically viable.

Chapter 3. TOOLS AND TECHNOLOGIES

In this project, several advanced tools and technologies are utilized to facilitate the analysis and modelling of bond market data. The most important are mentioned bellow.

3.1.1 TENSORFLOW

TensorFlow (Abadi, M., et al., 2016) is an open-source machine learning library developed by Google that is employed for building and training neural networks. TensorFlow's flexibility and scalability make it suitable for creating deep learning models that can capture intricate patterns in large datasets. In this project, TensorFlow is used to develop neural networks that predict bond returns based on various factors, as well as creating a final ensemble model. How each model is created will be explained later.

TensorFlow makes use of Keras (Chollet, F. & others, 2015) which is a high-level api that provides tools to simplify the network building process. Keras simplifies the process by allowing you to build your model using predefined layers that allow for editing. Layers can also be reused multiple times. In this study, the sequential model building tool was used which allows users to define the network in a sequential manner, where dimensions are inferred from pervious layers.

3.1.2 SCIKIT-LEARN

Scikit-Learn (Pedregosa, F., et al., 2011), commonly known as Sklearn, is a machine learning library in Python that provides simple and efficient tools for data mining and data analysis. Sklearn includes a wide range of algorithms for classification, regression, clustering, and dimensionality reduction. In this project, Sklearn is used for implementing traditional machine learning models such as regression, support vector machines, and ensemble methods. The library's features for model evaluation and validation are crucial for assessing the performance of these models and ensuring their reliability. The library also

allows for data splitting into train and test datasets used for validation. Specific uses of this technology will be explained as needed.

3.1.3 XGBOOST

XGBoost (Chen, T. & Guestrin, C., 2016) is an optimized gradient boosting library. It is known for its speed and performance and is particularly effective in handling large datasets and complex models. It provides advanced functionalities for regression and classification tasks, making it ideal for predicting bond returns. XGBoost can handle missing values and prevent overfitting through regularization techniques, this enhances the robustness of the models developed. Once again, more specific use cases will be discussed further on in the study.

Chapter 4. DATA ANALYSIS

4.1 DATA PREPROCESSING

For this research, two key indices from Bloomberg were utilized: the Bloomberg Barclays US Aggregate Total Return Unhedged USD Index., 2021, known as the LUACTRUU index and The Bloomberg Barclays US Corporate High Yield Total Return Index Unhedged USD., 2021, also referred to as LF98TRUU index.

LUACTRUU Index: This index represents the performance of the US Investment-Grade bond market. It includes corporate bonds that have a high credit score and are considered optimal for investment due to their low risk.

LF98TRUU Index: This index measures the performance of US dollar-denominated, high-yield, fixed-rate corporate bonds. This index focuses on the lower credit quality segment of the corporate bond market.

These indices were chosen for their broad representation of the bond market, encompassing both investment-grade and high-yield segments, thus providing a comprehensive view of bond price movements across different credit qualities. These indices were consolidated into a single CSV file, where each row represents a unique bond at a specific date. This structure allowed for the calculation of changes in the price of bonds at various future time intervals, specifically at 1 month (1m), 3 months (3m), 6 months (6m), and 12 months (12m). These served as the target variables for the machine learning models. The data itself has the following features available in Table 1.

Feature Name	Definition
ISIN	International Securities Identification Number, a unique identifier for the bond.

Price	The current market price of the bond.
YTW	Yield to Worst, the lowest potential yield that can be received on a bond without the issuer actually defaulting.
OAD	Option-Adjusted Duration, a measure of the bond's sensitivity to interest rate changes, accounting for embedded options
Par Val	The face value of the bond to be paid at maturity.
MV	Market Value, the current value of the debt in the market.
Weight	The proportion of the bond's market value relative to the total market value of the portfolio or index.
YTM	Yield to Maturity, the total return anticipated on a bond if held until it matures.
Maturity	The date on which the bond will mature, and the issuer will pay the bondholder the face value.
Index Rating	The credit rating of the bond as assigned by a credit rating agency.
BCLASS 2	Bond Class 2, a categorical classification of the bond.
Cpn	Coupon, the interest rate that the bond issuer will pay to the bondholder, usually expressed as a percentage of the par value.
Date	The specific date associated with the bond's data point.
Payment Rank	The seniority of the bond in the event of the issuer's liquidation, indicating the order in which bondholders will be paid.

R1M	The change in the price of a bond 1 month in the future
R3M	The change in the price of a bond 3 months in the future
R6M	The change in the price of a bond 6 months in the future
R12M	The change in the price of a bond for 12 months in the future

Table 1: Feature Definitions (Source: Own elaboration)

The ISIN and was retained temporarily for exploratory data analysis (EDA) but will be dropped during model training to ensure that each row is treated independently without identifying bonds by their unique IDs. To ensure that the models only depend on the features used and not on past trends, all date information was removed after the EDA. Each bond at each time interval is treated as a unique instance with no temporal knowledge of previous performance. This approach emphasizes the relationship between bond characteristics and the change in price, rather than their historical performance.

4.2 EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) involves investigating the dataset to uncover patterns, identify anomalies, and understand the relationships between variables. In this project, EDA aids in understanding the key characteristics of bond-related factors like price, yield to maturity, and credit rating, providing insights into their distributions and correlations.

At first glance, after a seasonal decomposition, we can see from Figure 2: Seasonal Decomposition for the Average Price of Bonds (Source: Own elaboration)

Figure 3: Payment Rank Distribution (Source: Own elaboration)Figure 2 that the price of bonds remains relatively stable during the years. There is a clear a seasonal component that explains about a 3% change in the price of bonds and a smooth long-term trend but most of

the movement can be attributed to randomness. This clearly suggests that there is more behind the data than a simple time series and other factors are at play.

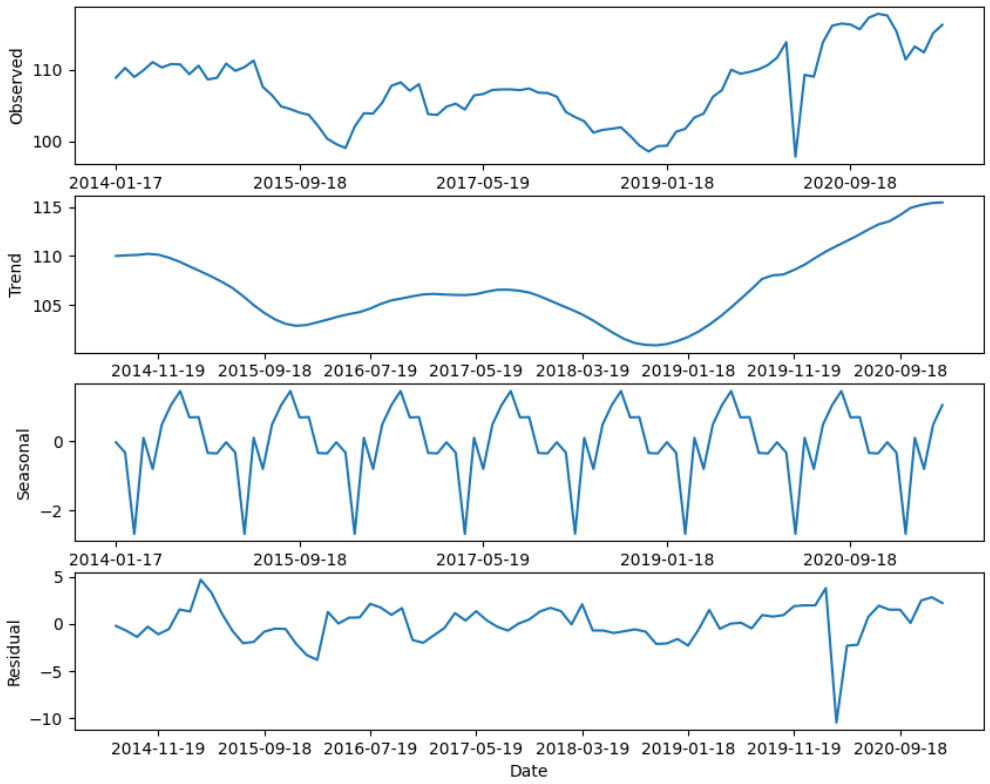

Figure 4: Seasonal Decomposition for the Average Price of Bonds (Source: Own elaboration)

Figure 5: Payment Rank Distribution (Source: Own elaboration)Figure 6: Seasonal Decomposition for the Average Price of Bonds (Source: Own elaboration)

The next step was to encode categorical variables to facilitate their use in machine learning models. Encoding categorical variables converts categorical data into a numerical format for machine learning algorithms. Common methods include label encoding, where each category is assigned a unique integer; one-hot encoding, which creates binary columns for

each category; ordinal encoding, used for categories with a specific order; and frequency encoding, based on the occurrence frequency of categories.

The variable Index Rating was made up of multiple ratings with a varied distribution as shown in Figure 4: Index Rating Distribution (Source: Own elaboration)

Figure 5: Correlation Matrix for Factors and Predictors (Source: Own elaboration)Figure 4. We applied label encoding to this class so that each rating had its corresponding number which allowed it to be treated as a numeric feature.

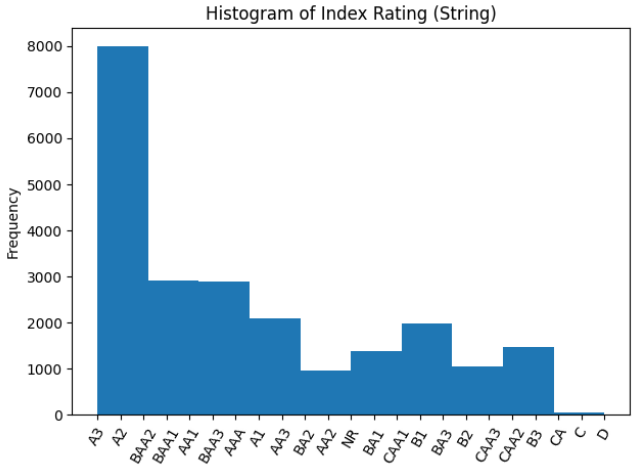

Figure 10: Payment Rank Distribution (Source: Own elaboration)

Figure 11: Index Rating Distribution (Source: Own elaboration)Figure 12: Payment Rank Distribution (Source: Own elaboration)

BCLASS 2 was encoded as it had 3 distinct categories FINANCIAL_INSTITUTIONS,

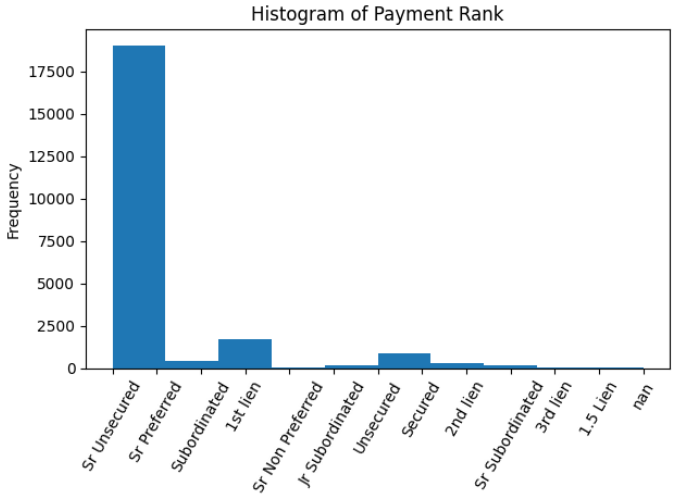

Figure 7: Index Rating Distribution (Source: Own elaboration)

Figure 8: Correlation Matrix for Factors and Predictors (Source: Own elaboration)Figure 9: Index Rating Distribution (Source: Own elaboration)

INDUSTRIAL, and UTILITY with no clear bias. The variable was encoded using label encoding where the categories were encoded as 1, 2, and 3. Payment Rank was also used to filter as the large part of the dataset belonged to a single class as seen in Figure 3: Payment Rank Distribution (Source: Own elaboration)

Figure 4: Index Rating Distribution (Source: Own elaboration)Figure 3. Due to this bias we decided to only keep bonds whose rank was *Sr Unsecured*.

Three variables were dropped due to their high correlation with others as they were redundant: YTW, Par Val, and OAD, this can be seen in Figure 5: Correlation Matrix for Factors and Predictors (Source: Own elaboration)

Figure 6: Distributions for Target Variables (Source: Own elaboration)Figure 5. This step helps streamline the model, reducing multicollinearity and improving its overall accuracy and interpretability. By eliminating these redundant variables, we ensure that the model is

more efficient and less prone to overfitting. In this case *YTM*, *MV* and *Maturity* where kept as they have all the relevant information and are more common factors.

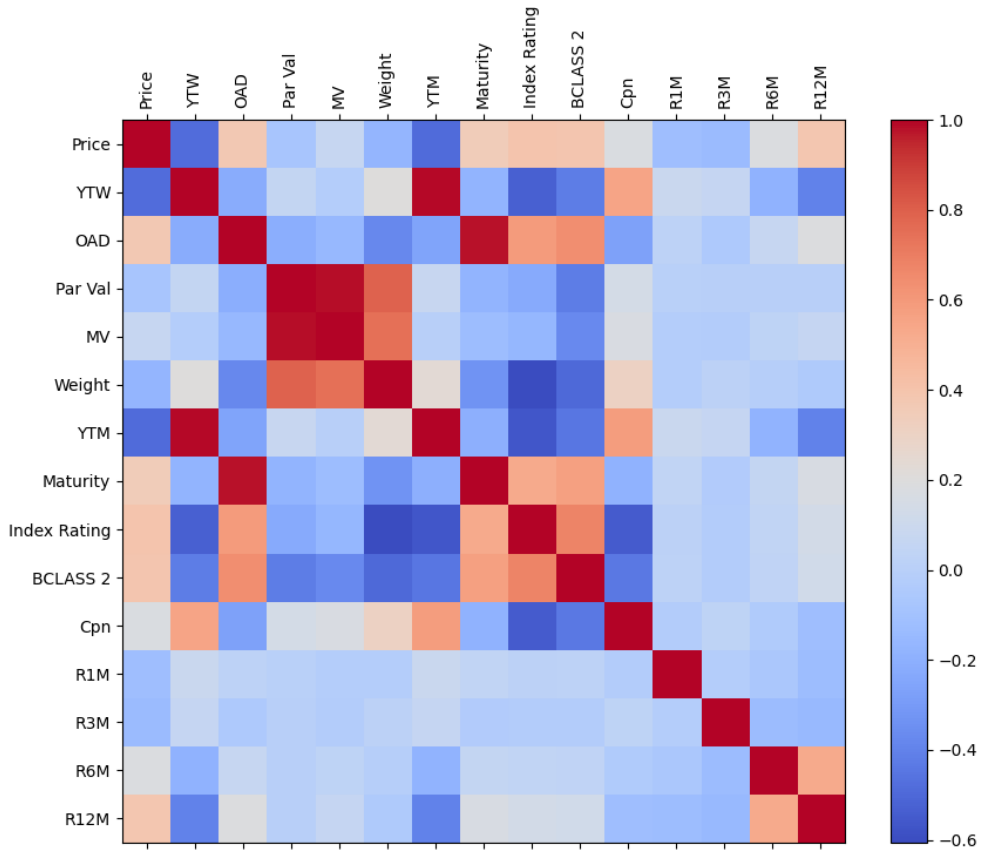

Figure 13: Correlation Matrix for Factors and Predictors (Source: Own elaboration)

Figure 14: Distributions for Target Variables (Source: Own elaboration)Figure 15: Correlation Matrix for Factors and Predictors (Source: Own elaboration)

After cleaning and processing the data, the final dataset includes eight factors: *Price*, *MV*, *Weight*, *YTM*, *Maturity*, *Cpn*, *BCLASS 2* (encoded), and *Index Rating* (encoded). The dataset comprises 917 unique bonds and 30,939 rows, with each row representing a bond at a unique point in time and the calculated returns for the specified future intervals.

The final step is to look at the target variables. From the following distributions in Figure 6: Distributions for Target Variables (Source: Own elaboration)

Figure 7: SVR compared to SVM (Source: Achsan, B. M., 2024)Figure 6 we can see that they are centred at 0. It makes sense for *R1M* to have the lowest change and for them to increases up to *R12M* as the latter covers more time which allows for more changes in the price of the bond. Having the data centred around 0 means it may be harder to correctly predict changes in price close to 0 as we may be mathematically close but predict a small negative change when it was a small, but positive, change.

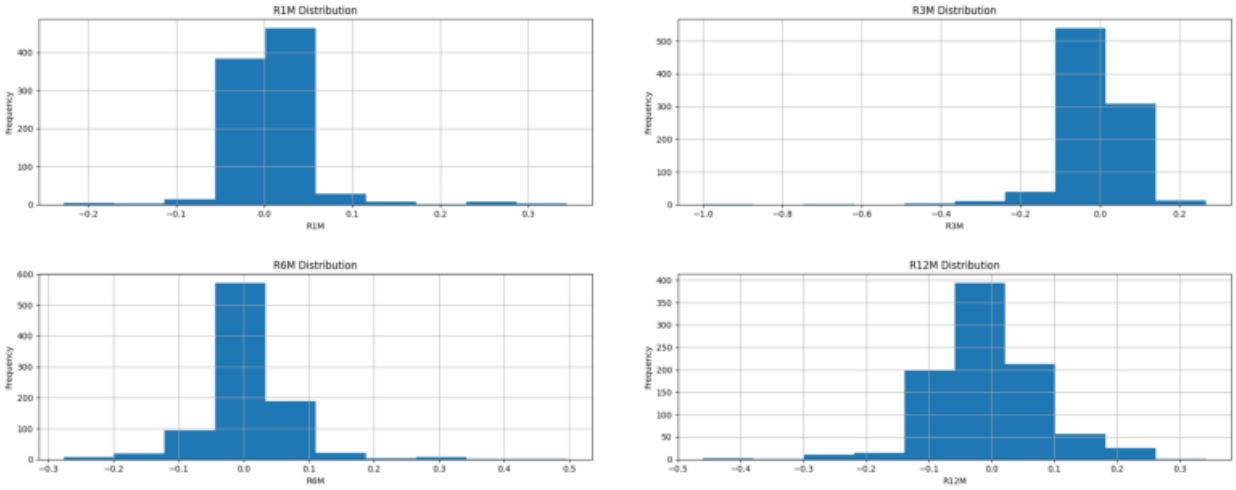

Figure 16: Distributions for Target Variables (Source: Own elaboration)

Figure 17: SVR compared to SVM (Source: Achsan, B. M., 2024)Figure 18: Distributions for Target Variables (Source: Own elaboration)

Before any model has been trained, the data was split into a train and test dataset. The test dataset is made up of all the data of the last available year. This data will only be used for building the portfolio to ensure that none of the models have been trained on this data. The train dataset will be used to train the models and is made up of the rest of the data.

Chapter 5. METRICS & MODELS

This chapter focuses on explaining the models proposed: Linear Regression, Support Vector Machines, Random Forest, Gradient Boosting, and Neural Network. An explanation will be provided for each model along with any relevant example. Before discussing the five models we will first explain the evaluation metrics used to train and test them: Mean Squared Error (MSE), R-squared (R2), and Hit Ratio.

5.1 METRICS

Metrics are used to evaluate the performance of a model by looking at how well the model's predictions match the actual target. One metric is used as the loss function that models will try to minimize to achieve the best results while other metrics will be calculated after the model is run to better compare different aspects of each model between them.

5.1.1 MEAN SQUARED ERROR

For training regression models, Mean Squared Error (MSE) is often the primary metric used, as it provides a clear indication of the model's accuracy. It measures the average of the squared differences between the predicted values and the actual values. The formula for MSE is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where:

- n is the number of observations
- y_i represents the actual value for the *i-th* observation
- $\hat{y}_i$ represents the prediction for the *i-th* observation

MSE, in the context of this study, tells us how close the model´s prediction got to the actual value. This metric will be the one all models will use as the loss function and will try to minimize to achieve the best results.

5.1.2 R-SQUARED

R-Squared (R2) is a statistical measure used to assess how well the regression model represents the real data. It represents the proportion of the variance in the dependent variable that is predictable from the independent variables. The formula for R-squared is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where:

- n is the number of observations
- y_i represents the actual value for the *i-th* observation
- $\hat{y}_i$ represents the prediction for the *i-th* observation
- $\bar{y}$ represents the mean of the actual values

R-squared ranges from 0 to 1, where 0 indicates that the model does not explain any of the variance in the dependent variable, and 1 indicates that the model explains all the variance. An R-squared value closer to 1 signifies a better fit, meaning the model's predictions closely match the actual data. It is important to use R-squared along with other metrics as a high value for this metric alone is not representative of an accurate model.

5.1.3 HIT RATIO

The final, but most important metric, is Hit Ratio. Hit Ratio is used in this study to measure how accurate a model is at finding the trend in which prices of bonds will move. In this case, the formula used for Hit Ratio is the following:

$$Hit\ Ratio = \frac{\sum_{i=1}^{n} 1(sign(y_i) = sign(\hat{y}_i))}{n}$$

Where:

- n is the number of observations
- y_i represents the actual value for the *i-th* observation
- $\hat{y}_i$ represents the prediction for the *i-th* observation
- $sign(x)$ is a function that returns +1 if the sign of x is positive, and -1 if the sign is negative
- $1(x)$ returns +1 if x is *true*, if x is *false* then 0 is returned

This formula calculates the proportion of predictions where the sign of the predicted value matches the sign of the actual value. This metric will be used to analyse which models better capture the trend in changes in bond price.

5.2 MODELS

5.2.1 LINEAR REGRESSION

Linear Regression (LR) is a relatively simple model. In this case an Ordinary Least Squares (OLS) Regression was used. It assumes that the target variable is a linear combination of the features used. It fits a linear equation to the data and finds the coefficients for each variable that minimizes the MSE of the model. The formula for the model is the following:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon$$

Where:

- y is the target variable
- x represents each independent feature
- n is the number of features
- β_0 is the intercept
- β represents the coefficient for each independent feature
- ϵ is the error

One of the main advantages of this model is how simple it is to explain the results using the coefficients. Before this can be done it is recommended to scale the input data as different scales can affect the coefficients when it comes to explaining what features had a large effect on the results. If all the data is scaled, then the size and sign of the coefficient can tell us how each variable is affecting the change in price compared to the other variables. For example, if after training the model we find that *MV* has a large negative β we can conclude that *MV* has a strong inverse correlation with the target.

Another important advantage is the computational efficiency which is important when training on such a large dataset. The model is substantially faster to train and test than other models which makes it ideal as a baseline to stack the other models up to.

The model also has its drawbacks. The model assumes a linear relationship between the target and the features when there may not be one, it is very sensitive to outliers as it tries to fit the regression to all the points, and the model is prone to overfitting as it creates a model that mimics the training data but has not learned to generalize.

5.2.2 SUPPORT VECTOR MACHINES

The exact model used during the project is a version of Support Vector Machines (SVM) called Epsilon-Support Vector Regression (SVR). SVM is suited for classification problems while SVR is tailored for regression. SVM works by finding the best hyperplane by using multiple vectors to separate classes. When applied to regression, the hyperplane is tuned so that instead of using the vectors to create subspaces, the vectors are used to approximate a function that describes the target along with a margin of error so as to not overfit. In Figure 7: SVR compared to SVM (Source: Achsan, B. M., 2024)

Figure 8: Decision Tree diagram (Source: Javapoint, 2024)Figure 7 we can see the difference between these two models.

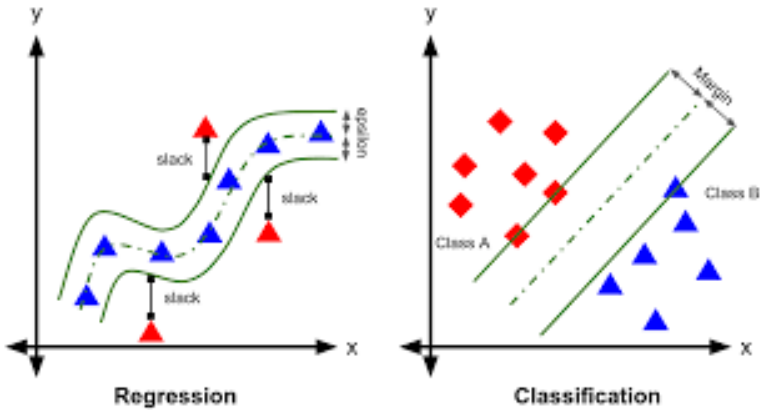

Figure 19: SVR compared to SVM (Source: Achsan, B. M., 2024)

Figure 20: Decision Tree diagram (Source: Javapoint, 2024)Figure 21: SVR compared to SVM (Source: Achsan, B. M., 2024)

SVR can capture nonlinear relationships as multiple vectors can be used. The model is very versatile as the kernel and multiple hyperparameters, can be optimized. The kernel handles non-linear relationships by implicitly mapping inputs into high-dimensional feature spaces. At these higher dimensions linear relationships can be made. There are multiple kernels such as linear, polynomial, and sigmoid but the one used in this paper was radial basis function (RBF) for two reasons; it can adapt to the structure of the underlying data, and it has fewer hyperparameters to tune.

Epsilon, which decides the margin of error, is set before training the model. What this margin does is treat real values inside this margin of error as correct and only considers values outside the margin as wrongly predicted. These values found outside the margins are the only ones that contribute to the loss function. The other hyperparameters we can set before training is the Regularization Parameter (*C*), *C* is how much the model will adapt to the data. A high value will cause the error to reduce but we may encounter overfitting. If a low value is set, then we will not have overfitting, but we may be missing out performance as the regression function would be too flat.

To optimize the hyperparameters used for the final model GridSearchCV was used. GridSearchCV is a tool provided by Sklearn that allows for dictionaries with multiple parameters as the input along with the model. The tool then trains the model using all the possible combinations between hyperparameters to find the best model. The tool also applies cross validation to the training process. This means that the tool divides the data into multiple subsets, training the model on some subsets, and validating it on the remaining ones, typically rotating the training and validation sets across iterations to ensure a robust evaluation.

SVR has a problem when it comes to large datasets as is the case. It can be very complicated for the model to assimilate all the data as it is increasing the dimension which can be memory intensive. The model is also very sensitive to the choice of epsilon and *C* which can lead to errors or loss of performance if chosen incorrectly. Finaly, as the data is converted to a higher dimension, explicability is lost. We could identify if the model worked as expected but we cannot determine which variables had the most impact on the target.

5.2.3 RANDOM FOREST

Random Forest (RF) is an ensemble model created from decision trees. An ensemble model is any model that is created by combining multiple models into one. In this case multiple decision trees are built to reduce the variance and increase the ability of the model to generalize. Each tree is built on a random subset of the data. The predictions from the trees are aggregated to produce the final prediction.

A decision tree, the model behind RF, is a decision-making tool used for prediction and classification. The model is made up of treelike structure, at each node a feature is chosen, and a decision is made based on that feature to split up the data. This feature and separation are chosen so that the next step has the best possible classification up to that point by minimizing the entropy (randomness) in the data. This makes the results easy to interpret as we can follow the decision process by hand and clearly see which features are more important to the model.

A tree consists of a root node where all the data is found. This root node then divides the data and subtrees are created. Each new node created can end up in a leaf node if no more decisions are made or it can continue to another decision node. An example of this is shown in Figure 8: Decision Tree diagram (Source: Javapoint, 2024)

Figure 9: 1-month Decision Tree (Source: Own elaboration)Figure 8. The depth of the tree is how many nodes we find from the root node to the lowest leaf node. The depth, along with a few other hyperparameters, can be tuned to limit the tree.

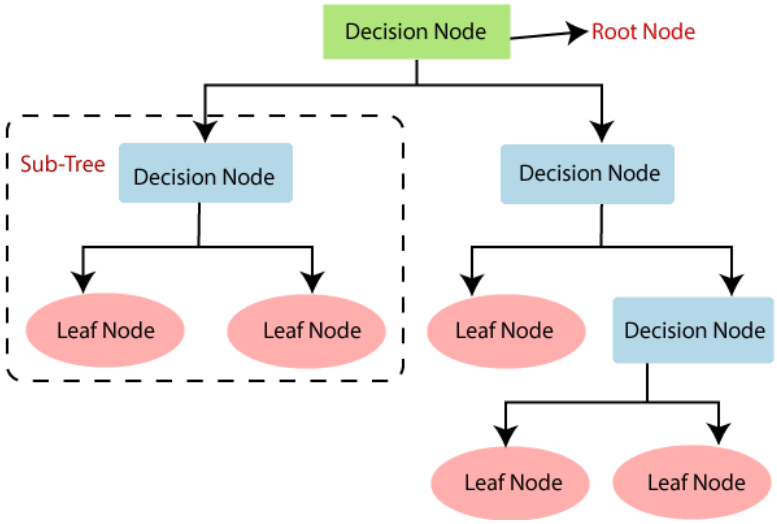

Figure 22: Decision Tree diagram (Source: Javapoint, 2024)

Figure 23: 1-month Decision Tree (Source: Own elaboration)Figure 24: Decision Tree diagram (Source: Javapoint, 2024)

Decision Trees are used in this study to identify key features as they allow for clear interpretation of the data but do not perform well when trying to create a regression model. Random Forest works well as a regression model, as by considering the predictions created by each tree, we can get more specific results. On the other hand, this aggregation of trees means the model lacks the explicability a decision tree offers. For example, a decision tree for the 1-month horizon is shown below in Figure 9: 1-month Decision Tree (Source: Own elaboration)

Figure 10: Different Learning Rates (Source: Donges, N., 2023)Figure 9. From the image we can see that one tree does not capture the trend correctly as it is dividing the samples too drastically.

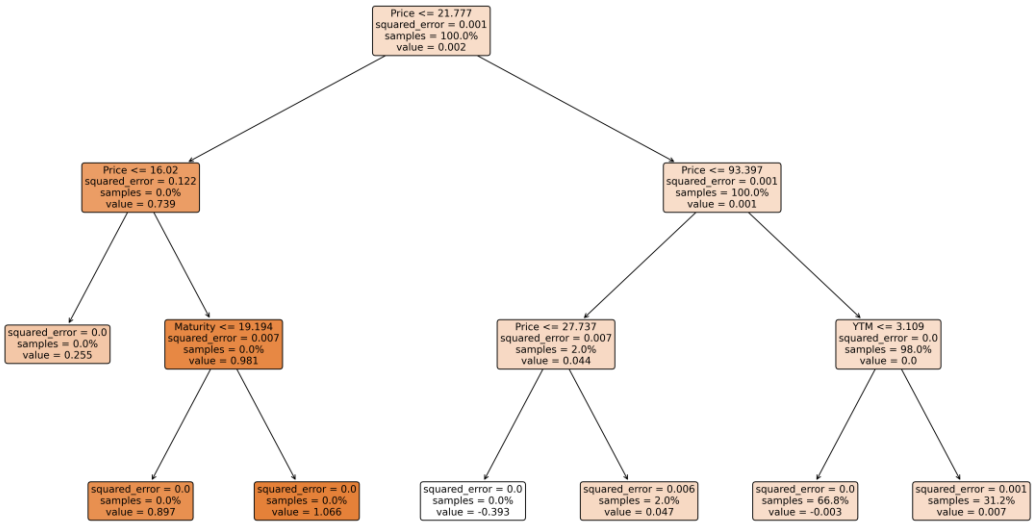

Figure 25: 1-month Decision Tree (Source: Own elaboration)

Figure 26: Different Learning Rates (Source: Donges, N., 2023)Figure 27: 1-month Decision Tree (Source: Own elaboration)

When training the Random Forest model, 4 hyperparameters were tuned using GridSearchCV. The first was hyperparameter tuned was the number of features to consider when looking for the best split. This can be done via two methods: the square root of the total number of features or the logarithm base two of the total number of features. By reducing the number of features, we can help reduce overfitting. The maximum depth of the tree was also tuned. A lower maximum depth means that each individual tree is simpler, but we are less likely to overfit the data and the computation time is much less. We can also tune how many samples a node needs to consider a split. A low number of samples could mean

that the model learns very specific patterns that do not translate well to other data. The final hyperparameter is very similar to the previous, this hyperparameter considers the minimum number of samples per leaf node. Again, a low number of samples means that the model could be learning patterns specific to this dataset that we could not use with other data.

Due to using so many hyperparameters and applying cross validation, the computation time was considerably high. This is expected when working with Random Forest, so a subsample of the data was chosen to train this model.

5.2.4 GRADIENT BOOSTING

Gradient Boosting (GB) is also an ensemble model that starts with a simple model, in this case it also uses decision trees, and sequentially builds a new model that attempts to correct the mistakes made by the previous iteration. In this study the XGBoost library was used as it provides a very efficient version of Gradient Boosting that allows for large datasets such as the one used here.

The model works by first initializing a relatively simple model with a high error. New models are then added in an iterative manner so that the new model fixes the residual errors of the combination of previous models. The residual errors are the difference between the actual and predicted values. The ensemble model uses gradient descent to minimize the loss function so that when the ensemble model adds a new model, it can help decrease the error. The learning rate used is typically low as a high learning could make the ensemble model to simple by reaching what the model thinks is the lowest point of the loss function to fast (Figure 10: Different Learning Rates (Source: Donges, N., 2023)

Figure 11: Neural Network (Source: Robinson, S., 2023)Figure 10).

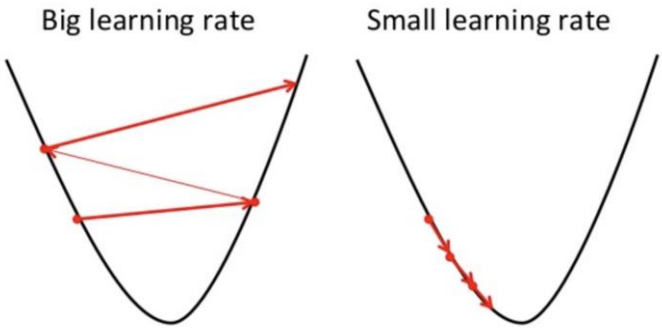

Figure 28: Different Learning Rates (Source: Donges, N., 2023)

Figure 29: Neural Network (Source: Robinson, S., 2023)Figure 30: Different Learning Rates (Source: Donges, N., 2023)

XGBoost is very efficient with large datasets as it can operate in parallel, when possible, even though the model is sequential. It also takes advantage of a very sophisticated tree pruning algorithm which stops large trees. It is necessary to stop these trees as the ensemble model should improve by way of a new model being added and not by complicating the current model.

Three different hyperparameters where optimized during the training process. The max depth for each tree was tested as we searched for the best performance without overfitting the data. The learning rate was also optimized so that we used the best possible rate so that we find the minimum MSE as fast as possible. The last hyperparameter we tuned is gamma, gamma defines the minimum loss reduction requires to consider another partition on a leaf node. A large gamma makes the model more conservative; this makes it less prone to overfitting but takes longer to reach the minimum in the loss function.

As with Random Forest, the model was trained on a subsample as using the whole dataset made training impossible on the available hardware.

5.2.5 NEURAL NETWORKS

A Neural Network (NN) is an imitation of how the human brain works. A Feed Forward Network like the ones used in this study are made up of neurons set up in layers with information highways set up between them. The result of the network is the prediction.

The simplest component of a neural network is a neuron. A neuron works by receiving an input and does some sort of calculation to generate an output. This calculation works by taking the input and applying a weighted some to it with some sort of bias. The result of this sum is then passed through an activation function. This function is what allows a network to introduce non-linearity which can help it learn complex patterns. Some of the main activation functions used are sigmoid, ReLU (Rectified Linear Unit), and tanh. Each activation function has its uses, some of which will be explained further on.

The neurons are organized in layers with information flowing in one direction between them. The first layer is known as the input data and receives the raw data. The data then passes through to the hidden layers. There can be multiple hidden layers, each with a specific number of neurons that transform the data until it reaches the output layer. The last layer creates a prediction based on the information it has received. An example of this can be seen in Figure 11: Neural Network (Source: Robinson, S., 2023)

Figure 12: Feature Importance for 1-month LR (Source: Own elaboration)Figure 11.

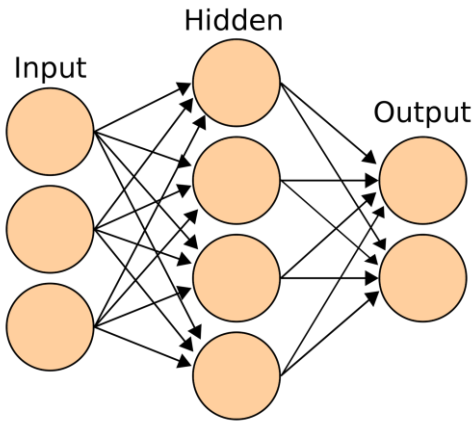

Figure 31: Neural Network (Source: Robinson, S., 2023)

Figure 32: Feature Importance for 1-month LR (Source: Own elaboration)Figure 33: Neural Network (Source: Robinson, S., 2023)

Neural networks learn by updating the weights between neurons. To do this an input is fed into the network and the network tries to predict the output. The error in the prediction is calculated and this error is then sent back through the network to adjust the weights so that the error is decreased. This process is known as backpropagation. With each iteration of inputs, the error gest smaller and smaller until a plateau is reached, and the network has reached its limit. The network is trained in batches and does not update the weights until each batch has been processed as it would take too long otherwise.

Neural networks excel at finding complex and obscure relationships in the data that other linear models cannot find, they also adapt to all sorts of data and are very efficient with large datasets, being computationally more effective than other models like Random Forest. On the other hand, they can be very sensitive to tunning and require careful manipulation. They also lack explicability as it is nearly impossible to follow data through the network.

In this study two different networks have been proposed. Both networks have been developed using TensorFlow and using the sequential architecture provided by Keras. The activation function used in both models is ReLU (the formula can be seen below). ReLU has

become the default activation function due to its simplicity and its ability to add sparsity to the data by introducing zeros.

$$f(x) = \max(x, 0)$$

Where:

- x is the input of the activation function
- $\max(x, y)$ is a function that returns the maximum between x and y

The first network has been designed to predict the change in price of bonds form the data. It is made up of an input layer that then feeds into the first hidden layer which is called a dense layer. In this case, the layer was made up of 64 neurons, with each neuron taking all the features as input along with the weights. Each neuron has the ReLU set as its activation function. The output of each neuron is passed to what is known a dropout layer. This layer does not have any neurons but instead drops connections in between layers. In this case the network was set to drop 20% of connections randomly. This is done to ensure that all neurons are activated so that the network can generalize. The next step sends the data to another dense layer with 32 neurons. Finaly, the output layer is reached which condenses the model into a single prediction.

The second network is very similar to the first but has a different purpose. This network is a type of ensemble model that takes the predictions from the other models and uses those as input. The model still predicts the change in the price of bonds but does not use bond data. The structure of this network is an input layer that feeds into a dense layer made up of 64 neurons. This layer feeds into another dense layer with 32 neurons and then a final layer that makes the prediction.

Chapter 6. RESULTS

To simplify the comparison between models this chapter provides sections for each implemented model where key findings are commented along with any necessary annotation. The final section of this chapter includes a table summarizing results for all the models along with a conclusion.

6.1 LINEAR REGRESSION

Linear Regression is commonly used as a baseline model to compare more complex models. This should make it clear that this model is not expected to perform particularly well. The model is very useful as it the coefficients provide an intuitive way to find which features have a strong effect on the change in price.

For the 1-month time horizon we can clearly see how the current price along with the yield to maturity clearly push the change in price. This can be seen in Figure 12: Feature Importance for 1-month LR (Source: Own elaboration)

Figure 13: Feature Importance for 12-month LR (Source: Own elaboration)Figure 12. It must be mentioned that the size of the coefficients is not as important as the relative size of them. This is because the data has been scaled so all the coefficients work on the same scale, it just so happens that for the 1-month time horizon this scale is relatively small. This trend is even more exaggerated when looking at the 3-month predictions. Weight, which had a significant effect on the 1-month model, decreases in importance compared to coupon rate.

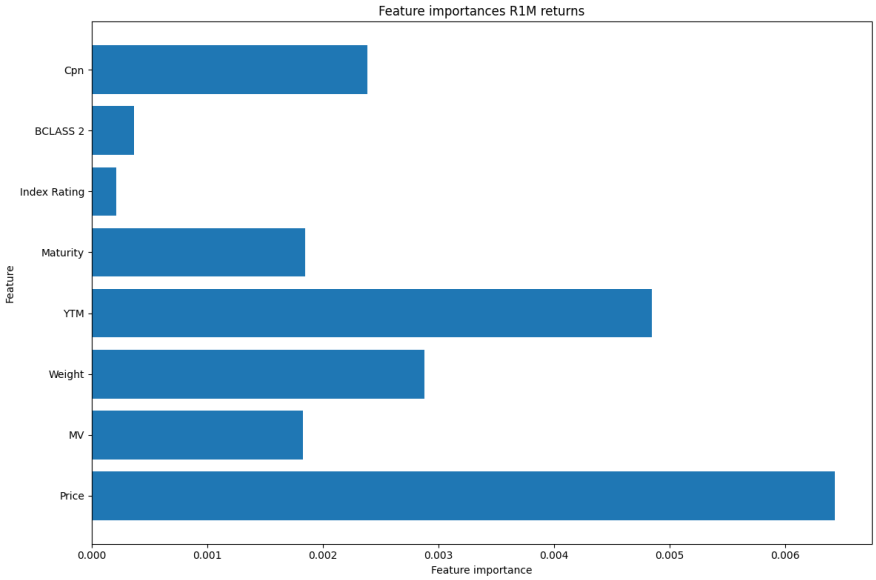

Figure 34: Feature Importance for 1-month LR (Source: Own elaboration)

Figure 35: Feature Importance for 12-month LR (Source: Own elaboration)Figure 36: Feature Importance for 1-month LR (Source: Own elaboration)

This makes sense as LR uses is only plotting a linear function and the current price, at a 1-month difference, will be heavily correlated with the future price. The yield to maturity, along with the coupon rate, provide key insights into how the market will price the bond in the future and the LR model recognizes that relationship.

When we look at the 6 and 12-month horizon we find that the model favours the coupon rate as this horizon allows for coupons to be paid so the price will be heavily influenced by this interest. The 6-month horizon depends mostly on the coupon rate, but the 12-month model allows for more features, especially yield to maturity, to affect the prediction. For example, in Figure 13: Feature Importance for 12-month LR (Source: Own elaboration)

Figure 14: Change in Feature Importance RF (Source: Own elaboration)Figure 13 we can see how the coupon rate has the largest coefficient. Yield to maturity is also very high but other factors like maturity have started to affect the change in price.

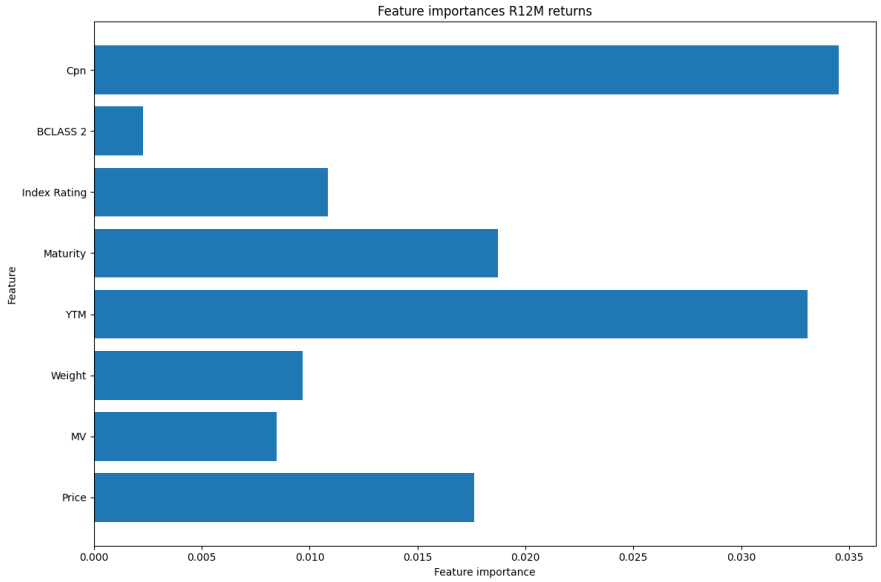

Figure 37: Feature Importance for 12-month LR (Source: Own elaboration)

Figure 38: Change in Feature Importance RF (Source: Own elaboration)Figure 39: Feature Importance for 12-month LR (Source: Own elaboration)

The model has a surprisingly good hit ratio across all models, except for the 6-month LR, meaning that it clearly identifies which bonds have a positive return and which do not. The problem, as expected with LR, is that the model has a low R2 score which indicates that there is a lot of randomness in the prediction. When looking at MSE we find that only the 1-month LR has a low score, with all the other models having very high scores. This means that all the other models, except the 1-month LR, have a very high error when it comes to the predictions.

6.2 EPSILON-SUPPORT VECTOR REGRESSION

Before a final model was tested, various iterations were trained with varying hyperparameters to choose the best model. As mentioned before, both C and Epsilon were varied during testing to choose the best model for each horizon. All models where first optimized using a subset of the data and the best model was then retrained on the whole training dataset to get a more accurate result. The results for each of the models are shown in the table below.

	1-month	**3-month**	**6-month**	**12-month**
C	0.1	0.1	1	1
Epsilon	0.01	0.01	0.05	0.05

Table 2: Hyperparameters for SVR (Source: Own elaboration)

The table shows how the value for each hyperparameter increases with the horizon. This falls into what was expected as C represents the inverse strength of the regularization that will take place. In smaller horizons we wish to be harsher as changes are smaller. This is also the reason why Epsilon is smaller for the smaller horizons as it represents the margin given. In smaller horizons the changes in price are smaller so a smaller margin is given.

The SVR model performed significantly better when compared to the LR model. Even though it produced only a slightly better hit ratio, both R2 and MSE provided much better results. R2 was higher across the board while MSE was smaller at all horizons. This shows that a more complicated model provided more accurate results.

6.3 RANDOM FOREST

Random Forest is the first of our two ensemble models and, as such, is very computationally intensive. For this reason, the model was trained on a subset that encompassed 20% of the original dataset. For hyperparameter tunning, a further sample of 10% was generated to try

all the possible combinations. These shortcuts were taken to reduce computation time. Again, the hyperparameters chosen for each model are shown below.

	1-month	**3-month**	**6-month**	**12-month**
Max_features	Sqrt	Log2	Log2	Log2
Max_depth	9	9	7	5
Min_samples_split	2	4	2	2
Min_samples_leaf	1	1	4	4

Table 3: Hyperparameters for Random Forest (Source: Own elaboration)

For the 1-month horizon, using *max_features* set to *sqrt* captures detailed short-term patterns, while a *max_depth* of 9 allows for complex model structures. The *min_samples_split* of 2 and *min_samples_leaf* of 1 further enable the model to capture fine details. For the 3-month horizon, *max_features* is set to *log2*, reducing complexity, and a *max_depth* of 9 maintains model depth, with a *min_samples_split* of 4 introducing some regularization. The 6-month and 12-month horizons also use *max_features* as *log2* but reduce *max_depth* to 7 and 5, respectively, to prevent overfitting over longer periods. Both horizons use a *min_samples_split* of 2 and increase *min_samples_leaf* to 4, ensuring sufficient data per leaf for better generalization.

The model can interpret the different trees it is made from to identify feature importances. In this case, all horizons have more in common than in LR. Across all horizons we find that YTM is the defining feature with Price trailing close behind and even surpassing it at the 1-month horizon. Price decreases in importance as we increase the window used. In Figure 14: Change in Feature Importance RF (Source: Own elaboration)

Figure 15: Feature importance for GB (Source: Own elaboration)Figure 14 we can see how Price decreases in importance from the 1-month to the 12-month horizon. The rest of the features maintain a similar weight in the final prediction.

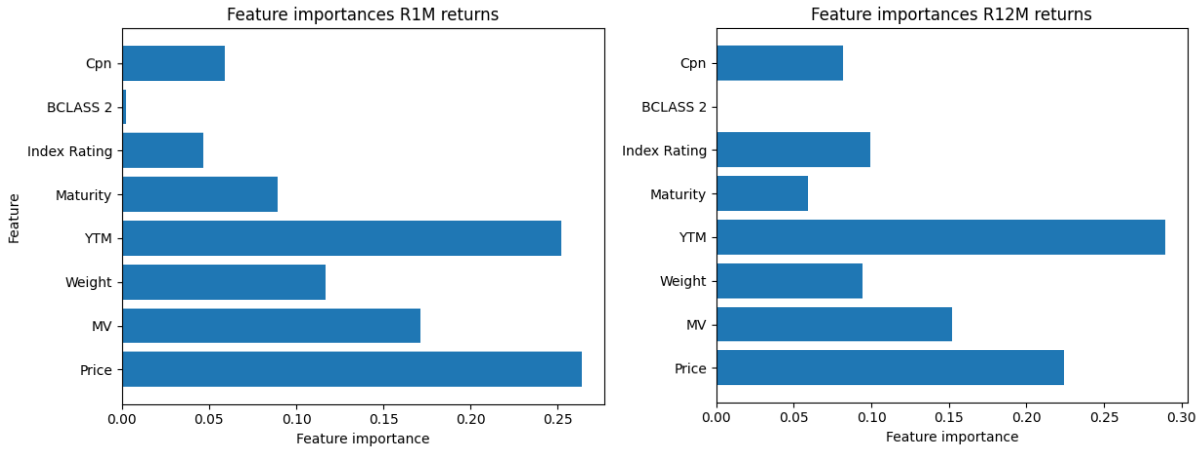

Figure 40: Change in Feature Importance RF (Source: Own elaboration)

Figure 41: Feature importance for GB (Source: Own elaboration)Figure 42: Change in Feature Importance RF (Source: Own elaboration)

All four RF models outperformed the LR models at their respective horizons. The hit ratios achieved where not significantly larger but both R2 and MSE provided much better results. When comparing MSE, all four models had lower results which indicate a more accurate prediction. R2 received the biggest outperformance, specially at the 3-, 6- and 12-month horizons. This tells us that the model is more reliable with its predictions and can capture more information from the data to feed its predictions. The model, when compared to SVR, obtained similar, but worse, results with small changes per horizon.

6.4 *GRADIENT BOOSTING*

Gradient Boosting (GB) is the second ensemble model. This model is not as intensive as RF but still requires the use of subsampling for hyperparameter tunning to reduce the training time. The results from this tuning are shown in Table 4.

	1-month	3-month	6-month	12-month
Max_depth	3	7	3	5
Learning_rate	0.1	0.01	0.01	0.01
Gamma	0	0	0	0

Table 4: Hyperparameters for GB (Source: Own elaboration)

A *learning_rate* of 0.1 for the 1-month horizon allows the model to quickly adapt to short-term patterns while the rest of the models choose a lower *learning*_rate to generalize. The *max_depth* changes at random from horizon to horizon which could mean that the model is not accurately capturing the variance and trend in the data. The *gamma* remains 0 across all horizons, keeping the model flexible without additional constraints on partitioning.

The same idea that the model is not accurately capturing the trend in the data can be seen when looking at feature importance in Figure 15: Feature importance for GB (Source: Own elaboration)

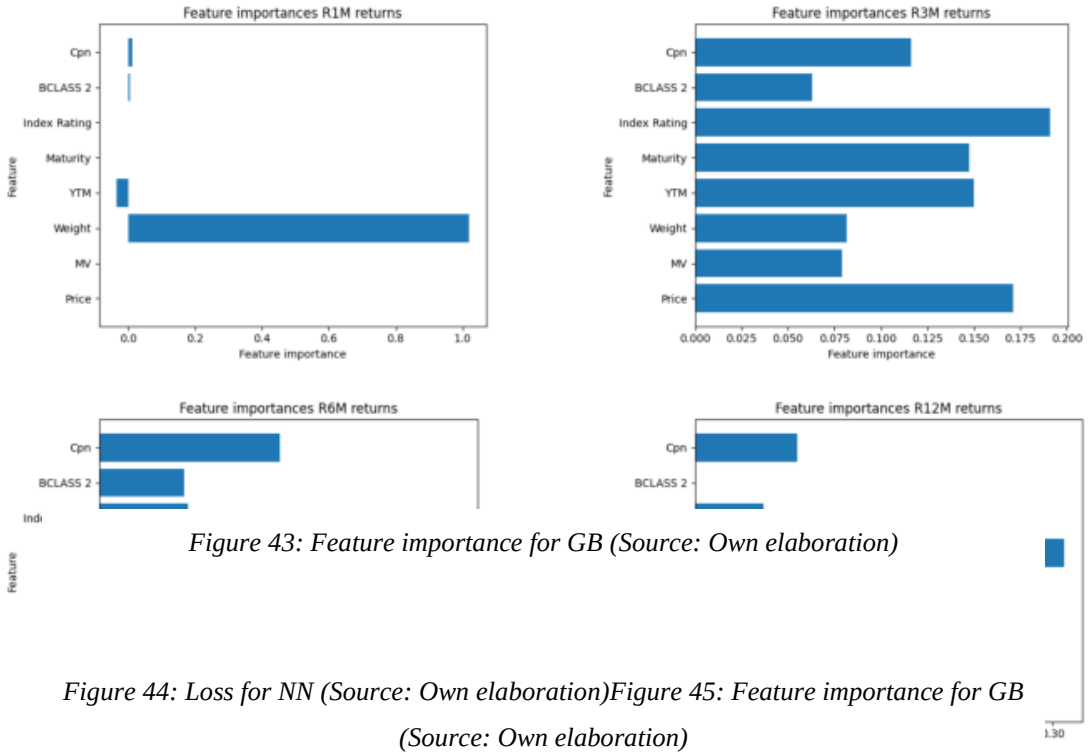

Figure 43: Feature importance for GB (Source: Own elaboration)

Figure 44: Loss for NN (Source: Own elaboration)Figure 45: Feature importance for GB (Source: Own elaboration)

Figure 16: Loss for NN (Source: Own elaboration)Figure 15. No model has the same features pushing the changes in price. There is no clear visual trend in importance which shows that the model is likely not finding a real trend.

The results also validate that this model is likely not capturing the trends in the data. For the 1-month horizon and the 6-month horizon, the model has a very low hit ratio. In addition, both the 1-month and the 12-month horizon have very low R2 which indicate that they are not capturing the variance in the data. Finaly, the MSE is like that of other models. The model is still better than LR, but the model would not be capable of generalizing to other datasets as well as the other models.

6.5 NEURAL NETWORK

The neural network trains itself and needs no optimization. The issue with neural networks is that they lack explicability, this means that it is impossible to visualize which features push the changes in price. The models used are trained over multiple iterations using

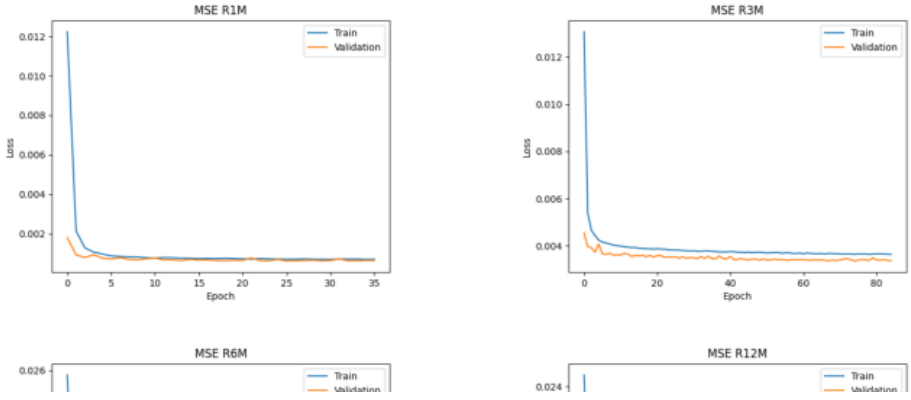

Figure 46: Loss for NN (Source: Own elaboration)

Figure 47: Loss for Ensemble NN (Source: Own elaboration)Figure 48: Loss for NN (Source: Own elaboration)

subsamples of the data in what are known as epochs. The network is given data until the loss function reaches its minimum using a technique called early stopping. This allows the model to stop training when a constant loss value is reached to not overfit the model.

In Figure 16: Loss for NN (Source: Own elaboration)

Figure 17: Loss for Ensemble NN (Source: Own elaboration)Figure 16 we can see how each network has stopped training at different intervals. In all models, the validation set produces similar or better results than the train set. This means that no model is overfitting the data.

All four models performed as expected with high Hit Ratios and low MSE. What was not expected was the high R2 score which indicated that all models could capture the variance more accurately. While the other two metrics are similar to other models like RF, R2 stands out as being exceptionally high.

6.6 NEURAL NETWORK ENSEMBLE

This final model, as explained before, is a neural network trained on the predictions from the rest of the proposed models. As commented before, neural networks can be hard to explain as internal connections and weights are hard to interpret. In Figure 17: Loss for Ensemble NN (Source: Own elaboration)

Figure 18: Change in price per month using 1-month holding periods (Source: Own

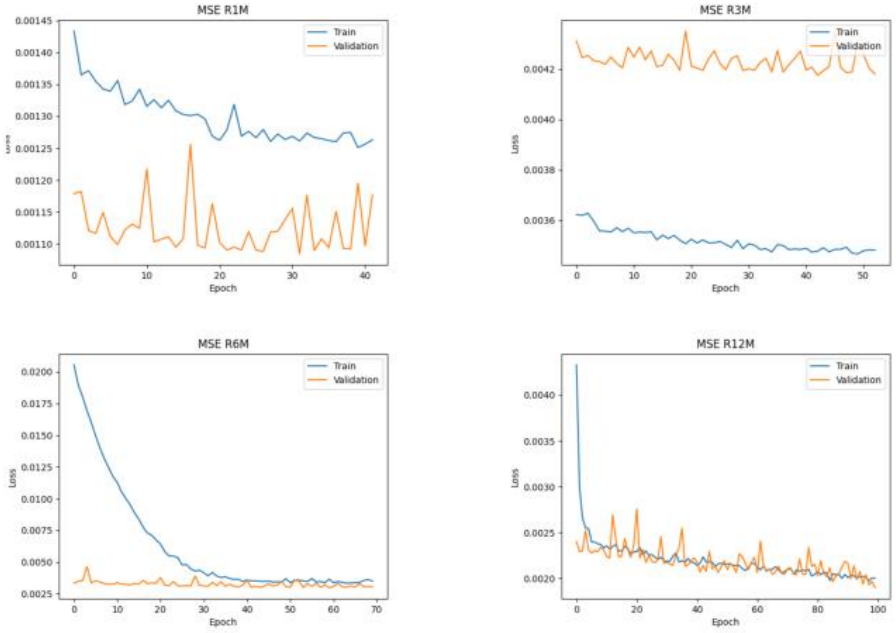

Figure 49: Loss for Ensemble NN (Source: Own elaboration)

Figure 50: Change in price per month using 1-month holding periods (Source: Own elaboration)Figure 51: Loss for Ensemble NN (Source: Own elaboration)

elaboration)Figure 17 we can see how the model, at different horizons, struggles to reach a consensus. This may mean that each model used as input predicts different bonds correctly and the network is having a difficult time extracting this relationship.

The model still produces accurate results for the 6-month and 12-month horizon with very high hit ratios but struggles to accurately predict the other two horizons as shown by the loss plot above. In this case, as R2 depends on the input data, we cannot compare it to the other models. This network does not take the original dataset as input so we cannot compare it to the previous models. It does provide some insight into how well the model is predicting as R2 is very high for the last two horizons but very low for the first two. This suggests that the model has found a trend in the 6 and 12-month models but has not been able to find it in the previous horizons.

6.7 SUMMARY & CONCLUSION

The following tables shows each metric over every model and horizon. This way we can visualize where each model performs best. The best result for every metric and horizon has been highlighted to make it clear to the reader.

In Table 5 we can clearly see that the use of an ensemble model has clearly brought down the error. For the first two horizons, as discussed before, we find that other models perform in a better or similar way to the Ensemble model but when we reach the last two horizons, we find that the Ensemble model lowers the error significantly. It is normal for the error to increase as the window increases as the target variable oscillates between bigger values.

MSE	**1-month**	**3-month**	**6-month**	**12-month**
LR	0.00072	0.00375	0.01232	0.01721
SVR	0.00072	0.00333	0.00963	0.01500
RF	0.00051	0.00349	0.00925	0.01183
GB	0.00054	0.00360	0.00958	0.01163

NN	0.00069	0.00324	0.00984	0.01429
Ensemble	0.00131	0.00301	0.00355	0.00369

Table 5: MSE for all models (Source: Own elaboration)

When looking at Table 6 we find that the Neural Network provides the highest R2 scores. The ensemble NN has been added to the table but not used when comparing values due to previously mentioned reasons regarding the validity of the results. It expected for NN to have the highest R2 scores as it is the best implementation of non-linear relationships meaning it can understand more of the data than the other models. That said, these R2 scores are not as high as we would like and represent an area of improvement for the future.

R2	**1-month**	**3-month**	**6-month**	**12-month**
LR	0.133	0.043	0.075	0.038
SVR	0.127	0.149	0.277	0.162
RF	0.134	0.156	0.196	0.069
GB	0.089	0.130	0.168	0.085
NN	0.161	0.173	0.261	0.201
Ensemble	0.112	0.175	0.328	0.629

Table 6: R2 for all models (Source: Own elaboration)

Finaly, the Hit Ratios shown in Table 7 show what we have already seen from the models. For the horizons at 6 and 12 months we find that the Ensemble model produces better results but when looking at the earliest horizons we find other models that outperform it.

Hit Ratio (%)	1-month	3-month	6-month	12-month
LR	62.1	69.1	54.3	76.8
SVR	64.8	73.0	64.6	78.9
RF	63.2	71.8	60.2	76.7
GB	59.6	70.4	57.5	76.8
NN	63.2	73.6	66.5	78.4
Ensemble	55.2	63.2	73.7	79.8

Table 7: Hit Ratio for all models (Source: Own elaboration)

From these results we can conclude that for horizons 1 and 3, a model like NN or SVR would prove more valuable. Once we reach the 6- and 12-month mark, we find that the Ensemble model better captures what the market is doing and can adapt to the data.

Chapter 7. PORTFOLIO ELABORATION

7.1 PORTFOLIO ELABORATION

In this final chapter we look at how a portfolio can be implemented using the models and the viability of this strategy. As this is just a proof of concept the simplest implementation has been proposed. The portfolio will use the models at the 1-month horizon to predict the following month's change in price for each bond and will either sell, buy, or hold the bond for the following month. This will be done for over a year using the test data separated at the beginning of the study to not use data that the models have already seen. The models will then buy the top 15% of bonds so as to reduce the risk of just buying the best bond which may be incorrectly predicted.

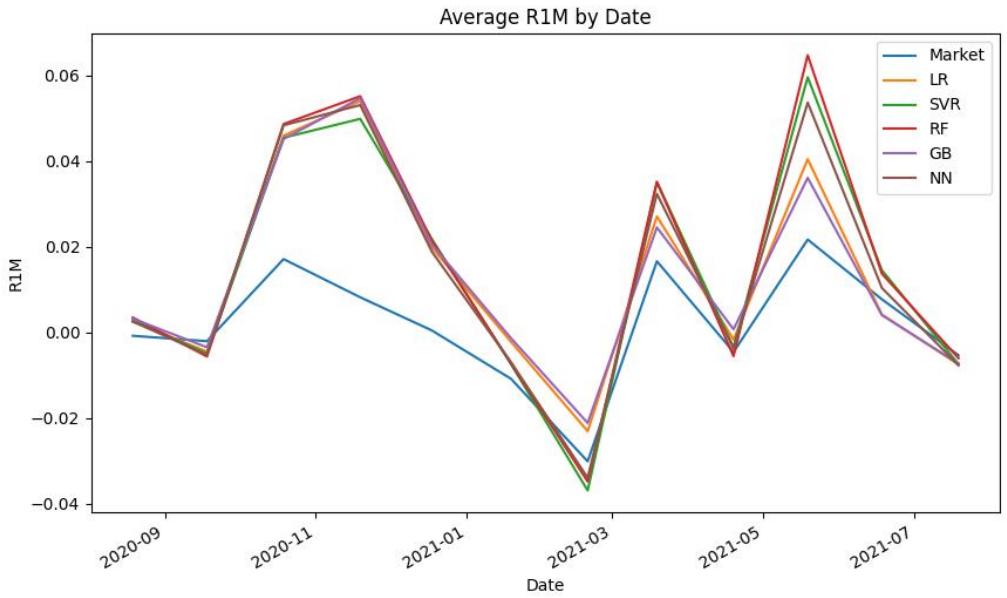

Figure 52: Change in price per month using 1-month holding periods (Source: Own elaboration)

Figure 53: Cumulative return over 1-year period (Source: Own elaboration)Figure 54: Change in price per month using 1-month holding periods (Source: Own elaboration)

In Figure 18: Change in price per month using 1-month holding periods (Source: Own elaboration)

Figure 19: Cumulative return over 1-year period (Source: Own elaboration)Figure 18 we can see how most portfolios follow a similar trend. We see how RF, SVR and NN are clearly choosing the bonds with the highest change. When we look at the cumulative return on investment of each individual portfolio in Figure 19, we see that this trend holds. In figure we see that RF has the best return on investment with 19.3%. NN and SVR follow closely behind with 18.4% and 17.8% respectively. It must be said, as mentioned at the beginning of this study, that this portfolio does not represent real world returns as no costs are

considered. This proof of concept has been made to validate the theory that these models can correctly identify trends and patterns to predict changes in price.

7.2 CONCLUSIONS

From these findings, we can say that for this dataset the best performance came from RF but when we trained the models RF did not have the best results. The most likely solution is that

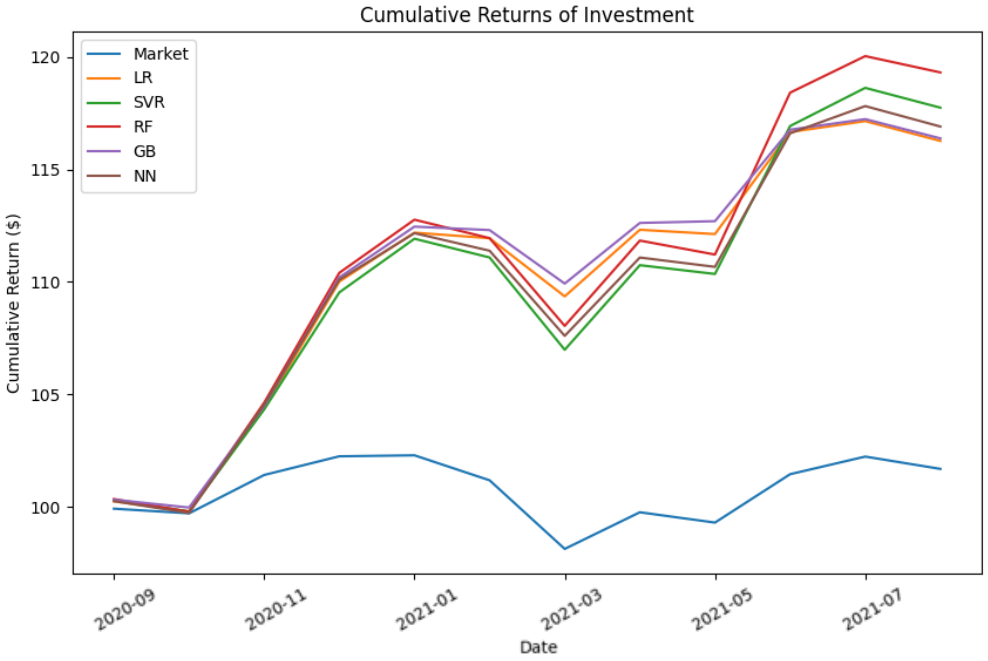

Figure 55: Cumulative return over 1-year period (Source: Own elaboration)

RF was better suited for this new dataset and so yielded better results. If we were to apply the same model to another dataset, we may find that another model is provides the best performance.

In this proof of concept, the Ensemble NN was not used as it needed to use the predictions from all the models and, as we saw in training, did very poorly at the 1-month horizon. For other horizons and for generalizing to other datasets we do believe that the Ensemble model is still the best choice.

Chapter 8. CONCLUSIONS & FUTURE WORK

8.1 CONCLUSIONS

The primary objective of this research was to develop and validate models capable of predicting bond price changes over various horizons using key bond characteristics. The models employed, including Linear Regression, Support Vector Machines, Random Forests, Gradient Boosting, and Neural Networks, were successful in capturing the complex, nonlinear relationships in bond markets. The results indicated meaningful predictive power, although the R2 values suggest there is still room for improvement in model accuracy.

One of the key achievements of this study was the demonstration of the real-world applicability of the developed models through a proof-of-concept portfolio. This portfolio, constructed based on the model predictions, showcased the practical benefits of these models. The portfolio's performance underscored the potential for enhanced decision-making in bond trading. It is important to remember that no other factors like costs or macroeconomic trends have been considered and these can heavily influence returns.

8.2 FUTURE WORK

For future research, several avenues could be explored to further improve the predictive capabilities and practical relevance of the models. Firstly, shifting the focus from predicting bond price changes to predicting bond returns could provide a broader understanding of bond performance but require more computation which make it difficult. This could also help increase the R2 scores as the models may find more relationships by having the full picture.

Additionally, incorporating transaction costs into the portfolio construction process is critical. While this study demonstrated the feasibility of using model predictions to guide trading decisions, real-world applications must account for transaction costs, which can significantly impact net returns. By integrating these costs into the predictive models and

portfolio simulation, future work can provide more realistic and applicable investment strategies.

In conclusion, while this study made showed the feasibility of applying machine learning to bond price prediction and portfolio management, ongoing research and expanded methodologies will continue to enhance the accuracy and practical utility of these models.

IAG USE DECLARATION

Declaración de Uso de Herramientas de Inteligencia Artificial Generativa en Trabajos Fin de Grado

ADVERTENCIA: Desde la Universidad consideramos que ChatGPT u otras herramientas similares son herramientas muy útiles en la vida académica, aunque su uso queda siempre bajo la responsabilidad del alumno, puesto que las respuestas que proporciona pueden no ser veraces. En este sentido, NO está permitido su uso en la elaboración del Trabajo fin de Grado para generar código porque estas herramientas no son fiables en esa tarea. Aunque el código funcione, no hay garantías de que metodológicamente sea correcto, y es altamente probable que no lo sea.

Por la presente, yo, Álvaro Olivié Molina, estudiante de Grado en Business Analytics de la Universidad Pontificia Comillas al presentar mi Trabajo Fin de Grado titulado "Predicting Bond Price Changes Using Machine Learning and Factor Analysis", declaro que he utilizado la herramienta de Inteligencia Artificial Generativa ChatGPT u otras similares de IAG de código sólo en el contexto de las actividades descritas a continuación:

1. **Referencias:** Usado conjuntamente con otras herramientas, como Science, para identificar referencias preliminares que luego he contrastado y validado.
2. **Interpretador de código:** Para realizar análisis de datos preliminares.
3. **Corrector de estilo literario y de lenguaje:** Para mejorar la calidad lingüística y estilística del texto.
4. **Sintetizador y divulgador de libros complicados:** Para resumir y comprender literatura compleja.

Afirmo que toda la información y contenido presentados en este trabajo son producto de mi investigación y esfuerzo individual, excepto donde se ha indicado lo contrario y se han dado los créditos correspondientes (he incluido las referencias adecuadas en el TFG y he explicitado para que se ha usado ChatGPT u otras herramientas similares). Soy consciente de las implicaciones académicas y éticas de presentar un trabajo no original y acepto las consecuencias de cualquier violación a esta declaración.

Fecha: 21/06/2024 Firma: _____

BIBLIOGRAPHY

Abadi, M., et al., 2016. Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation*. pp. 265–283.

Arnott, R., Harvey, C.R. and Markowitz, H., 2019. A backtesting protocol in the era of machine learning. *The Journal of Financial Data Science*, 1(1), pp.32-43.

Bali, T.G., Goyal, A., Huang, D., Jiang, F. and Wen, Q., 2020. The cross-sectional pricing of corporate bonds using big data and machine learning. *The Review of Financial Studies*, 33(8), pp.3773-3813.

Bali, T.G., Goyal, A., Huang, D. and Jiang, F., 2021. Different strokes: Return predictability across stocks and bonds with machine learning and big data. *The Journal of Financial Data Science*, 3(1), pp.1-16.

Bloomberg, 2021. Bloomberg US Corporate High Yield Total Return Index Value Unhedged USD. *Bloomberg.com*. Available at: https://www.bloomberg.com/quote/LF98TRUU:IND

Bloomberg, 2021. Bloomberg US Corporate Total Return Value Unhedged USD. *Bloomberg.com*. Available at: https://www.bloomberg.com/quote/LUACTRUU:IND

Carhart, M.M., 1997. On persistence in mutual fund performance. *The Journal of Finance*, 52(1), pp.57-82.

Chen, T. & Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 785–794. Available at: http://doi.acm.org/10.1145/2939672.2939785.

Chollet, F. & others, 2015. Keras. Available at: https://github.com/fchollet/keras.

Coqueret, G. and Guida, T., 2023. Machine Learning for Factor Investing: Python version. *Boca Raton: CRC Press*.

Donges, N., 2023. Gradient Descent in Machine Learning: A Basic Introduction. *Built In*. Available at: https://builtin.com/data-science/gradient-descent.

Fama, E.F. and French, K.R., 1992. The cross-section of expected stock returns. *The Journal of Finance*, 47(2), pp.427-465.

Fama, E.F. and French, K.R., 2012. Size, value, and momentum in international stock returns. *Journal of Financial Economics*, 105(3), pp.457-472.

Houweling, P. and van Zundert, J., 2017. Factor investing in the corporate bond market. *Financial Analysts Journal*, 73(2), pp.100-115.

Achsan, B. M., 2024. Support Vector Machine: Regression. *Medium.com* Available at: https://medium.com/it-paragon/support-vector-machine-regression-cf65348b6345

Javatpoint, 2024. Decision Tree Algorithm in Machine Learning. *Javapoint.com*, Available at: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm.

Jegadeesh, N. and Titman, S., 1993. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1), pp.65-91.

Jegadeesh, N. and Titman, S., 2001. Profitability of momentum strategies: An evaluation of alternative explanations. *The Journal of Finance*, 56(2), pp.699-720.

Kaufmann, H., Messow, P. and Vogt, J., 2020. Boosting momentum in corporate bond markets. *The Journal of Fixed Income*, 29(4), pp.27-41.

Lim, B., Zohren, S. and Roberts, S., 2021. Enhancing time series momentum strategies using deep neural networks. *The Journal of Financial Data Science*, 3(1), pp.1-20.

Parker, T., et al., 2023. Factors in fixed income? It's (not so) academic. *Black Rock* Available at: https://www.blackrock.com/us/individual/insights/fixed-income-factor-investing.

Pedregosa, F., et al., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, pp. 2825-2830.

Poh, L.H., Xie, L., Liu, B. and Liu, Y., 2021. Building cross-sectional systematic strategies by learning to rank. *The Journal of Financial Data Science*, 3(2), pp.1-18.

Robinson, S., 2023. Introduction to Neural Networks with Scikit-Learn. *Stack Abuse*. Available at: https://stackabuse.com/introduction-to-neural-networks-with-scikit-learn/.

Zaremba, A., 2017. Momentum effect in government bonds. *International Review of Financial Analysis*, 52, pp.257-272.

CODE INFORMATION

A Github repository has been created to house all the code for this project. The dataset has not been included but all models, graphs and metrics have been uploaded. All necessary information about the structure of the project can be found in the README.md file. The Github repository can be found at:

https://github.com/Alvaro-Olivie/TFG