Analysis of Financial Metrics of Major US Companies Using Machine Learning Techniques for Stock Recommendation

Sergio Beamonte González

November 3, 2025

UNIVERSIDAD POLITÉCNICA DE MADRID

Departamento Inteligencia Artificial

Contents

Abstract

This study develops and evaluates a set of machine learning models to predict analyst stock recommendations (classified into 5 categories, from ”Strong Buy” to ”Sell”) for a subset of companies from the Russell 3000 index. Using a dataset of 49 financial and market metrics obtained via the Yahoo Finance API, the analysis rigorously compares the impact of various preprocessing strategies, Feature Subset Selection (FSS) methods, and resampling techniques (SMOTE).

A wide range of algorithms was evaluated, including non-probabilistic models (Decision Trees, SVM, Neural Networks), probabilistic models (Logistic Regression, Bayesian Classifiers), and meta-classifiers (Stacking, Bagging, Random Forest).

The results indicate that data preprocessing, specifically the treatment of outliers and the transformation of skewed distributions (TProc), was the most influential factor on model performance, surpassing feature selection. The SMOTE oversampling technique proved ineffective, reducing accuracy in most cases. The best-performing models were complex classifiers such as Logistic Regression, RBFClassifier, and SVM, along with ensembles like LMT and Stacking. The optimal model was a Stacking meta-classifier (SMO, A1DE, meta=Logistic), which achieved the highest correct percentage (37.80%) and a weighted AUC of 0.69. The main conclusion is that a combination of robust preprocessing and advanced ensemble models is the most effective strategy for tackling this complex financial classification problem.

1 Dataset Explanation and Problem Description

1.1 Data Source, Collection and Objectives

The dataset used in this analysis comprises financial and market metrics for a subset of companies from the Russell 3000 index. The data was programmatically collected using the Yahoo Finance (yfinance) API.

Thus, the objective of this analysis is to predict the rating that analysts will give to a specific company, knowing all available public financial metrics. With these metrics, and as a secondary objective, an attempt will be made to discover which metrics are most used by experts, what patterns exist, or at least, to get a general idea of how each metric affects a company's prosperity.

Before proceeding, it is important to note potential challenges when analyzing the following data. On one hand, this is not objective data; the reliability or intent of the analysts is unknown. Perhaps, even if only by luck, a more accurate prediction than that of a particular analyst could be achieved here. Despite this, even if time proved us right, for the model evaluation, it would be considered an incorrect result. On the other hand, the accuracy of the data has not been cross-verified with other sources. Occasionally, during the preliminary analysis before training, data points have proven incorrect or inconsistent.

One further caveat: the metrics were downloaded on 10/19/2025, which coincided with a Sunday when the market was not open. However, although most of these metrics are updated daily or even more frequently, analyst recommendations are not. It is possible that the landscape in which they issued their predictions was different from the moment the data was downloaded. Nevertheless, fundamental company metrics were specifically chosen, as these do not usually vary with daily or weekly market fluctuations.

1.2 Feature Description

The final dataset contains 49 features (columns) used for the analysis. The target variable is `recommendationClass`, a categorical label indicating the consensus analyst rating for the stock based on the final 0-4 scale described above.

The following table provides a comprehensive description of each feature, its data type, and its financial relevance.

Variable	**Data Type**	***Description***
Company Information		
`numberOfAnalystOpinions`	INTEGER	*Number of analysts who provided a rating for the stock.*
`sector`	CATEGORICAL	*Industry sector of the company (e.g., 'Healthcare', 'Technology').*
`In_SP500`	BINARY	*1 if the company is part of the S&P 500 index, 0 otherwise.*
`In_NASDAQ`	BINARY	*1 if the company is listed on the NASDAQ exchange, 0 otherwise.*
`fullTimeEmployees`	INTEGER	*Total number of full-time employees.*
`has_benefits`	BINARY	*1 if the company is known to offer employee benefits, 0 otherwise.*
Market Metrics		
`marketCap`	INTEGER	*Total market value of the company's outstanding shares (Shares * Price).*
`enterpriseValue`	INTEGER	*A measure of a company's total value (Market Cap + Total Debt - Total Cash).*
`averageVolume`	INTEGER	*Average number of shares traded daily over a recent period.*

(Continued on next page)

Table 1 – (Continued from previous page)

Variable	**Data Type**	***Description***
sharesOutstanding	INTEGER	*Total number of shares held by all shareholders.*
floatShares	INTEGER	*Number of shares available for public trading (excludes insider/restricted shares).*
sharesShort	INTEGER	*Total number of shares that have been sold short by investors.*
sharesPercentSharesOut	FLOAT	*Percentage of total outstanding shares that are currently held short.*
shortRatio	FLOAT	*Days to Cover: Ratio of shorted shares to average daily volume.*
shortPercentOfFloat	FLOAT	*Percentage of float shares (publicly available) that are shorted.*
heldPercentInsiders	FLOAT	*Percentage of shares held by company insiders (e.g., executives, directors).*
heldPercentInstitutions	FLOAT	*Percentage of shares held by institutional investors (e.g., hedge funds).*
Stock Performance		
currentPrice	FLOAT	*The most recent available closing price of the stock.*
allTimeHigh	FLOAT	*The highest price the stock has ever reached.*
allTimeLow	FLOAT	*The lowest price the stock has ever reached.*
beta	FLOAT	*A measure of the stock's volatility (systematic risk) relative to the overall market.*
trailingAnnualDividendYield	FLOAT	*The annual dividend per share as a percentage of the stock's current price.*
payoutRatio	FLOAT	*The percentage of a company's earnings that are paid out as dividends.*
Financial Health		
totalRevenue	INTEGER	*Total income generated from goods or services (aka "top line" or sales).*
grossProfits	INTEGER	*Total revenue minus the cost of goods sold (COGS).*
ebitda	INTEGER	*Earnings Before Interest, Taxes, Depreciation, and Amortization.*
netIncomeToCommon	INTEGER	*Net income available to common shareholders after preferred dividends.*
totalCash	INTEGER	*Total amount of cash and cash equivalents on the balance sheet.*
totalCashPerShare	FLOAT	*Total cash divided by the number of shares outstanding.*
totalDebt	INTEGER	*Total outstanding debt, including both short-term and long-term obligations.*
operatingCashflow	INTEGER	*Cash generated from normal day-to-day business operations.*
freeCashflow	INTEGER	*Operating cash flow minus capital expenditures (cash available to investors).*
quickRatio	FLOAT	*Liquidity ratio: (Current Assets - Inventory) / Current Liabilities.*
currentRatio	FLOAT	*Liquidity ratio: Current Assets / Current Liabilities.*
bookValue	FLOAT	*The net asset value of a company (Total Assets - Intangible Assets - Liabilities).*
is_Insolvent	BINARY	*A binary flag indicating potential insolvency (likely derived).*
_debtToEquity	FLOAT	*Total debt divided by shareholder equity (D/E). Measures financial leverage.*
Valuation Ratios		
priceToBook	FLOAT	*Ratio of market cap to book value (P/B).*
enterpriseToRevenue	FLOAT	*Ratio of enterprise value to total revenue (EV/Sales).*
profitMargins	FLOAT	*Net income as a percentage of revenue.*
grossMargins	FLOAT	*Gross profit as a percentage of revenue.*
ebitdaMargins	FLOAT	*EBITDA as a percentage of revenue.*
operatingMargins	FLOAT	*Operating income as a percentage of revenue.*

(Continued on next page)

Table 1 – (Continued from previous page)

Variable	**Data Type**	***Description***
returnOnAssets	FLOAT	*Net income relative to total assets (ROA). Measures asset efficiency.*
returnOnEquity	FLOAT	*Net income relative to shareholder equity (ROE). Measures profitability.*
revenuePerShare	FLOAT	*Total revenue divided by shares outstanding.*
trailingEps	FLOAT	*Trailing Twelve Months (TTM) earnings per share.*
_PER	FLOAT	*Price-to-Earnings ratio (Current Price / EPS).*
Target Variable		
recommendationClass	CATEGORICAL	*The target variable for classification, discretized into 5 groups (0=Strong Buy, 1=Buy, 2=Hold, 3=Underperform, 4=Sell).*

Table 1: Full description of all 49 dataset features.

The features have been grouped into the categories above for the following reasons, highlighting their importance in predicting a company's potential:

- **Company Information:** Provides basic context about the company (sector, size, index membership). This is important because factors like the industry sector or company size can significantly influence the rest of company metrics such us margins, growth potential or inherent risks.
- **Market Metrics:** Describes the overall market valuation (market capitalization, enterprise value) and the liquidity/interest in its stock (volume, float shares, short interest, ownership structure by insiders and institutions). These are crucial for understanding market sentiment, investor confidence, and potential stock price volatility.
- **Stock Performance:** Reflects the historical performance and volatility of the stock price (current price, historical highs/lows, beta), as well as direct returns to shareholders (dividends and payout ratio). This helps assess the risk associated with the stock and provides context on past returns, which can sometimes be indicative of future trends.
- **Financial Health:** Measures the company's core ability to generate revenue, profits, and cash flow (revenue, various margins, EBITDA, operating and free cash flows), and its solvency (debt levels, liquidity ratios like quick and current ratios, book value). This category is fundamental for evaluating the company's sustainability, operational efficiency, and intrinsic capacity for future growth.
- **Valuation Ratios:** Compares the company's market value (stock price, market cap, enterprise value) to its underlying financial performance metrics (earnings, book value, revenue, assets, equity). Ratios like P/B, EV/Revenue, P/E, ROA, and ROE are essential tools for determining if the stock appears overvalued, undervalued, or fairly valued relative to its fundamentals and comparable companies in the market.

2 Dataset Preprocessing and Model Evaluation

2.1 Preprocessing Overview

Before training the models, the data will undergo preprocessing to improve model effectiveness and achieve more generalizable and accurate predictions. Throughout the data preparation process, a Python notebook was used with the libraries pandas, scikit-learn, and imbalanced-learn.

The process begins with an exploratory data analysis. Then, based on the characteristics of each variable, a specific preprocessing step will be chosen. Likewise, to verify if the preprocessing is effective, a test will be conducted on the different training models. If effectiveness improves, that preprocessing step will be maintained throughout the subsequent analysis, discarding the others.

2.2 Feature Scales

A simple glance at the data reveals that the scales of the values vary significantly between variables. For example, `_PER` (*range: 0 to 5927, mean: 40*) or `beta` (*range: -6.4 to 8.7, mean: 1.2*) are vastly different from `ebitda` (*range: -3.07e9 to 1.57e11, mean: 2.35e9*). This discrepancy causes certain models to give more importance to variables with higher values, overlooking those with more modest values even if they predict the target variable more accurately. Some of the models most sensitive to this are, for example: K-Nearest Neighbors (KNN) or Support Vector Machine (SVM) (which use distances), Neural Networks (MLP, RBF) (where the gradient is affected by magnitudes), and Logistic Regression (where coefficients depend on predictor scales and can become unstable). Others, however, are immune, such as decision trees or rule-based methods.

Thus, one of the techniques that will be applied is the rescaling of data. Here, we will use the **standardization** (set mean to 0 and variance to 1) over the **min-max normalization** as is less sensitive to outlayers, preserves variance information and it generally works better for the major part of the models.

2.3 Outliers and Transformations

At first glance, plotting histograms of the variables reveals that almost all are highly asymmetrical, with very long tails. Therefore, to use parametric models that assume Gaussian distributions, such as Naive Bayes or Linear Discriminant Analysis (LDA), among others, the variables will be transforme so they more closely resemble normal distributions. Three different techniques will be used here, applying the most appropriate one to each variable: **removing outliers**, **logarithmic scale transformation**, or **no processing**. To choose the best of the three, the distributions will be compared, and kurtosis and skewness metrics will be used to make the process more robust (See graph).

Some outliers are extremely exaggerated (e.g., very high market capitalization or revenue values) and may represent data entry errors or exceptional cases that could unduly influence the model. The plan is to identify and remove rows corresponding to these most extreme outliers. Less pronounced outliers will be kept initially, but their impact will be monitored, and techniques like robust scaling or transformations (e.g., log transformation) might be applied selectively for models sensitive to them.

2.4 Target Variable Balance

We examined the distribution of the target variable, `recommendationClass`, as shown in Figure 1. The analysis reveals a significant imbalance: 'Hold' recommendations are considerably more frequent than 'Buy' or 'Strong Buy', while 'Sell' and 'Underperform' categories are even less represented. This imbalance needs to be considered during model evaluation, potentially requiring techniques like **stratified sampling**, **over/undersampling** (e.g., **SMOTE**).

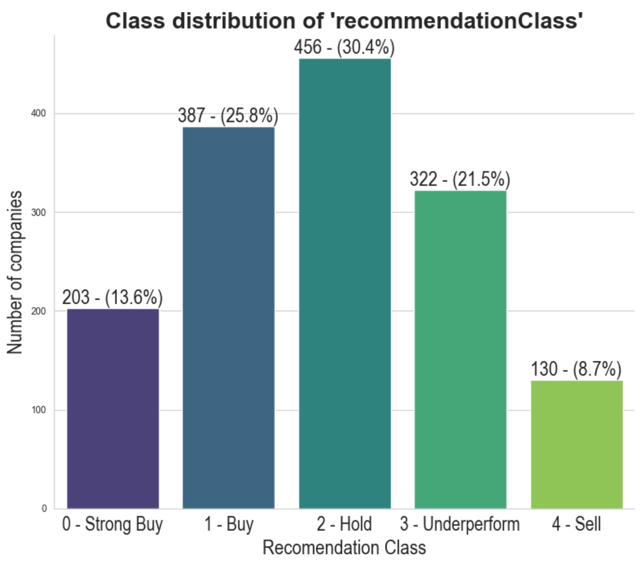

Figure 1: Distribution of Target Variable Classes (`recommendationClass`)

In this case, both scenarios will be tested: with and without rebalancing. We will attempt to rebalance the classes using the SMOTE (Synthetic Minority Over-sampling Technique) approach to balance the classes by generating synthetic samples for the minority classes. The results after applying this technique will be checked to see if model performance improves, particularly for the under-represented classes. SMOTE will only be applied to the training data within each cross-validation fold to prevent data leakage.

2.5 Test statistics and scores

To assess the accuracy of our models, we will perform testing on previously unseen data. We will use a **10-fold cross-validation** procedure, where the dataset is randomly divided into 10 parts—9 folds for training and 1 for testing. This process will be repeated five times with shuffled data to obtain averaged results, allowing for a more general evaluation of the model's performance.

Model accuracy will be measured using three statistics: **Percentage of Correctly Classified Instances (PC)**, which indicates the proportion of correct versus incorrect predictions; **Weighted Average Area Under the ROC Curve (AUC)**, which assesses how well the models discriminate between classes and their robustness in classification; and finally, **Root Mean Squared Error (RMSE)**, since our target variable is ordinal—meaning that misclassifying a class 0 instance as class 4 is a much more severe error than predicting it as class 2.

3 Feature Subset Selection

3.1 Introduction to Feature Subset Selection

To improve model performance, a Feature Subset Selection (FSS) process will be conducted before applying supervised learning models. Specifically, we aim to evaluate the behavior of these models both without FSS and after applying three complementary approaches: (1) a univariate filter, selecting the most relevant variables based on their individual relationship with the target variable; (2) a multivariate filter, considering interdependencies between attributes; and (3) a model-specific Wrapper method, where selection is directly guided by predictive performance. This experimental approach will allow us to identify which models benefit most from dimensionality reduction, if at all. It is expected that models more sensitive to noise or attribute redundancy (such as KNN, SVM, or Naive Bayes) will experience significant improvements, while others more robust to irrelevant variables (such as Random Forest) may not show such drastic changes in their effectiveness.

We will begin by selecting one univariate and one multivariate filter, considering the characteristics of our data.

3.2 Univariate Filter

Information Gain was selected as the main univariate filter, given that the problem involves a nominal classification target and a set of mixed attributes (both numerical and categorical). Unlike other correlation-based filters, it does not assume linear relationships. On the other hand, although there are feature selection methods designed for purely numerical predictors (such as ANOVA or t-tests), these would exclude categorical variables and are therefore disregarded. Finally, it was observed that other tests, such as Chi-Squared, produced worse results during the initial classification model evaluations.

As can be seen in the variable selection in Table 3.1, Mutual Information selects a few variables that it calculates maximize the mutual information between themselves and the target variable. The top 15 have been chosen (3 for each variable theme) and because there was a significant jump in the MI score. In this way, variables that already seemed to indicate a good or bad company, such as `PER` (Price Earning Ratio), which indicates how expensive the stock is, `totalRevenue`, the company's profit, or the sector it belongs to (if given the choice right now, one would invest in a technology company much sooner than a traditional automotive or fossil fuel mining company). However, it is important to note that many variables from the Financial Health section were selected. This is because the variables in this section actually carry more weight in analysts' decisions, but they are highly correlated with each other (e.g., `operatingCashflow`, `freeCashflow`).

3.3 Multivariate Filter

The multivariate feature subset selection (FSS) method chosen was Correlation-Based Feature Subset Selection (CFSS), due to its empirical superiority with the main models. Furthermore, this choice is justified by the high inter-variable correlations present in our dataset, as this evaluator is specifically designed to address the issue of high redundancy among attributes. Unlike ranking-based methods (such as ReliefF), which may select multiple correlated attributes (e.g., ebitdaMargins and profitMargins) and thereby bias distance calculations, CFSS aims to find a diverse "team" of attributes—prioritizing those with high correlation with the target class but low correlation among themselves.

Thus, this corrects what occurred in the univariate filter. Now the variables are distributed across the 5 sections (between 2 and 4 in each) and, in most cases, they align with those from Mutual

Information.

3.4 Wrapper Methods

Finally, Wrapper methods were employed, as they are particularly beneficial for powerful models that are sensitive to noise and redundancy, such as IBk, KStar, NaiveBayes, or Logistic Regression. These algorithms lack internal attribute selection mechanisms, and their performance can degrade when the dataset contains irrelevant or highly correlated variables. By directly evaluating model performance across different attribute subsets, the Wrapper approach identifies the optimal combination that maximizes accuracy. As a result, it substantially enhances distance-based and simple probabilistic models, while its effect is less pronounced on more robust algorithms such as RandomForest, which inherently perform internal feature selection.

3.5 FSS Comparison

The tables 2 and 3 summarize the features selected by each of the main filter and wrapper methods implemented.

Initial filter methods: **Mutual Information** and **Correlation-Based Feature Subset Selection (CFSS)** have been already analized. They offer a robust foundation for identifying the most relevant features for the analysis, unlike wrapper methods, which are biased toward optimizing a specific learning algorithm. They evaluate features based purely on the intrinsic statistical properties of the data itself, independent of any model.

Now that both feature selection methods have been computed, the attention turns to the variables *consistently* selected by both. Their repeated appearance signifies fundamental importance, not a model-specific preference. We can observe a clear consensus on features like `sector`, `totalRevenue`, `grossProfits`, `_PER`, `priceToBook`, and `numberOfAnalystOpinions`. Because these selections are not skewed or biased by any single classifier, they provide a far more reliable and unbiased insight into which metrics *truly* drive the classification, marking them as the most important variables for a general interpretation. Therefore, special caution must be taken to maintain the consistency and accuracy of the data for these specific variables.

Table 2: Comparison of features selected by FSS methods (Part 1 of 2)

FSS Method	numberOfAnalystOpinions	sector	In_SP500	In_NASDAQ	currentPrice	allTimeHigh	beta	averageVolume	marketCap	enterpriseValue	sharesOutstanding	floatShares	sharesShort	sharesPercentSharesOut	shortRatio	shortPercentOfFloat	heldPercentInsiders	heldPercentInstitutions	fullTimeEmployees	has_benefits	trailingAnnualDividendYield	payoutRatio	totalRevenue	revenuePerShare
Mutual Information	X	X							X	X											X	X	X	
Correlation Based	X	X							X			X		X			X				X	X	X	X
Wrapper JRip	X								X												X	X		
Wrapper C4.5 (J48)			X	X					X														X	
Wrapper Dec.Table														X									X	X
Wrapper SMO				X							X					X	X			X	X			X
Wrapper Logistic	X		X	X										X	X	X					X		X	
Wrapper NaiveBayes				X				X						X										
Wrapper OneR																				X				
Wrapper IBK				X																X				
Wrapper A1DE	X	X		X	X	X	X					X		X		X	X				X		X	X
Wrapper KStar									X															
Wrapper MLP		X			X		X		X													X		
Wrapper RBF	X						X			X														
Wrapper TAN		X	X						X					X			X						X	
Wrapper SimpleLogistic	X			X										X								X		

Table 3: Comparison of features selected by FSS methods (Part 2 of 2)

FSS Method	grossProfits	ebitda	netIncomeToCommon	totalCash	totalCashPerShare	totalDebt	operatingCashflow	freeCashflow	quickRatio	currentRatio	bookValue	is_Insolvent	_debtToEquity	priceToBook	enterpriseToRevenue	profitMargins	grossMargins	ebitdaMargins	operatingMargins	returnOnAssets	returnOnEquity	trailingEps	_PER
Mutual Information	X	X	X			X	X	X							X								X
Correlation Based	X					X			X					X	X								X
Wrapper JRip	X														X								X
Wrapper C4.5 (J48)																							
Wrapper Dec.Table							X								X								X
Wrapper SMO			X		X				X														
Wrapper Logistic	X						X							X							X		X
Wrapper NaiveBayes				X					X				X	X	X						X		
Wrapper OneR																							
Wrapper IBK																							
Wrapper A1DE									X		X		X	X		X		X	X	X		X	X
Wrapper KStar																							
Wrapper MLP		X				X			X											X		X	
Wrapper RBF														X		X				X		X	X
Wrapper TAN			X				X						X	X				X					X
Wrapper SimpleLogistic	X						X																X

4 Models

4.1 Non-probabilistic Models

4.1.1 Decision Trees

The **ID3 (Random Tree)** and **C4.5 (J48)** models (an evolution of *ID3*) have been evaluated. According to the results in the table (the results table at the end of chapter 4), the *C4.5* algorithm outperforms *ID3*, as expected. On the other hand, decision trees do not improve when balancing techniques are applied; on the contrary, if one looks at the SMOTE column, it is seen that the accuracy percentage and even the Root Mean Squared Error (RMSE) is reduced (used to penalize errors between distant classes more heavily). The same happens when processing the data by standardizing the variables, although a slight improvement is observed when using the transformation, compared to simple standardization. This is because trees already handle data consistently even with extreme values and are precisely based on boundaries that do not depend on the variable scales.

Different parameter configurations have been tested, with the best ones appearing in the table. A difference was also observed when applying Error Pruning to the *C4.5* algorithm, even though it already performed pruning in its own routine.

To understand which were the most relevant variables, the trees generated with the most precise model (*C4.5* with multivariate FSS) are examined. As the first nodes, the `totalRevenue` variable is found, which splits the tree the most; does the company have high or low profits? This is the first question one would ask, and apparently, it is present in all analysts' models. This is followed by the number of analysts and the sector: the hypothesis here is that the greater the number of analysts, the more the average opinion is centered on a central position (see graph), the same for the sector where the rest of the variables vary greatly depending on the sector (see graph). Finally, for the final decision, it has been observed that the `_PER` variable is often repeated in the last branches, as this could be the final decision when classifying a company. Once all financial health metrics, historical values, sector, etc., are known, what is the cost-benefit of buying the stock? Is it very expensive? If so, the PER will tell me this: if it's a mediocre company but very cheap, it is classified as `buy`, there is a great potential profit if it prospers; if it is very expensive, as `sell` because the probabilities of prospering do not exceed the expected return.

4.1.2 Artificial Neural Networks (ANN)

The **MLP** and **RBFClassifier** ANN models have been used, with diverse results. While **MultilayerPerceptron** achieves a result similar to the best models from other groups, albeit with a much higher computational cost, *RBFClassifier* manages to surpass it by a wide margin with a similar execution time. For both, it was expected that class rebalancing would improve the result; and although this has significantly improved the Root Mean Squared Error (RMSE) (used to penalize errors between distant classes more heavily); the rest of the results are worse (in general, poorer classification, with a higher number of errors but less severe errors). A undoubtedly more conservative option would be to use the prediction from the model trained with SMOTE.

This may be because ANNs tend to be sensitive to outliers and noise. SMOTE introduces synthetic data into classes 0 and 4, which are often the ones with the most extreme and, therefore, worst values. This is also confirmed by the fact that in all cases, the statistic improves. Both FSS methods help, as they reduce noise from other variables that can confuse the network. However, standardization does not help; in *RBF*, this may be because the algorithm internally normalizes the variable with min-max, distorting it.

It has been proven that an increase in training time and a small decrease in the learning rate produce significantly better results (up to 15% more in accuracy percentage). The effect of increasing hidden layers is not as clear.

4.1.3 Induction Rules

Models have been used in order from major to minor complexity: **OneR** (only one rule per attribute), **Decision Table**, and the **RIPPER** algorithm. Observing the table (ref to results table), the following can be seen: the results for *Decision Table* and *RIPPER* are better in terms of accuracy percentage than those of *OneR*. On the other hand, it is observed that SMOTE does help *RIPPER* achieve a higher accuracy percentage and a lower RMSE (used to penalize errors between distant classes more heavily), unlike with the *Decision Table* where no difference is appreciated. With feature selection, barely any significant change is seen in the *Decision Table*; this is because the table itself generates a subset of variables. In *RIPPER*, the change is small and only with fssm (multivariate FSS), which could be explained by the reduction in correlation between variables. In any case, *RIPPER* and *DecisionTable* are very similar; the former [*RIPPER*] commits less serious errors (lower RMSE) but creates less distinction between classes (lower AUC) and fails more (lower Accuracy %).

In table 4 it is shown the first 5 rules from *RIPPER* algorithm.

Rule Description	**Class (S / E)**
R1: Low `grossProfits`, **moderate** `priceToBook`, and **high** `sharesPercentSharesOut`	0 (24 / 6)
R2: Low `grossProfits`, **high** `priceToBook`, and **low** `totalDebt`	0 (37 / 15)
R3: Low `trailingAnnualDividendYield`, **solid** `marketCap`, and **weak** `grossProfits`	1 (114 / 53)
R4: Low `trailingAnnualDividendYield`, **moderate** `totalDebt`, and **moderate** `_PER`	1 (49 / 18)
R5: Low `trailingAnnualDividendYield`, **high** `enterpriseToRevenue`, **low** `heldPercentInsiders`	1 (27 / 5)

Table 4: Simplified decision rules by predicted class (Samples / Errors).

The rules show that low (`grossProfits`) combined with valuation ratios such as `priceToBook` or `totalDebt` strongly characterize the *Strong Buy (0)* class. For the *Buy (1)* class, rules involve low (`trailingAnnualDividendYield`) and moderate leverage, typical of companies reinvesting profits rather than distributing them. Instances that do not fit these patterns default to the *Hold (2)* class, suggesting more neutral or undefined financial behavior.

4.1.4 K-Nearest Neighbors (KNN)

The simple **K-NN (IBk)** and **K*-NN (KStar)** models have been used. They yield the worst results, resembling the simplest models. The conclusion that can be drawn from them is that they do need to be standardized because the distance to near neighbors depends heavily on the scale. No attempt was made to investigate further given the nature of our dataset, neither by varying the number of neighbors nor by varying the type of distance. It is a dataset in which the classes are quite mixed among themselves. Furthermore, the imbalance does not help, as *k-NN* models tend to overestimate the most frequent class (SMOTE fixes the result a bit), and a very high RMSE is observed compared to the rest.

4.1.5 Support Vector Machines (SVM)

A non-linear **Support Vector Machine (SVM)** model was used, which yielded the best classification among the simple non-probabilistic classifiers. Above all, it achieves a high accuracy rate and a lower RMSE, while also improving class classification with a higher AUC — almost as precise as the *RBF* network. It has been observed that the *SVM* model performs significantly better on processed datasets due to the reduction of skewness in each variable (it is recalled that *SVMs* have a geometrical approach).

Variable	**0vs1**	**0vs2**	**0vs3**	**0vs4**	**1vs2**	**1vs3**	**1vs4**	**2vs3**	**2vs4**	**3vs4**
In_SP500	0.0599	0.0938	0.9160	1.6843	0.0014	0.7121	0.3209	0.3384	0.0281	0.0005
ebitda	-1.2407	-1.3304	-1.6123	-1.7028	-0.0002	-0.8182	-0.2493	-1.1943	-0.1551	-0.0013
totalCashPerShare	0.4127	0.5058	1.0200	0.6885	-0.0002	0.5542	0.4936	0.7401	0.0649	0.0010
totalDebt	0.6031	1.0445	2.5846	0.3484	0.0010	0.7037	-0.1863	0.0419	-0.0521	-0.0012
quickRatio	-0.1980	-0.6339	-2.4132	-1.5648	-0.0026	-2.0933	-0.9446	-2.0463	-0.1671	-0.0027
trailingAnnualDividendYield	0.3738	0.7835	2.1689	2.5395	0.0053	3.9377	2.5780	2.1366	0.1506	0.0015
payoutRatio	-0.2939	-0.1389	-0.6363	-0.0774	0.0028	-0.8820	-0.9044	-1.0339	-0.1446	-0.0023
sharesOutstanding	0.5096	0.5806	1.2612	0.2114	0.0234	-0.1758	-0.1112	0.2934	0.1631	0.0117
floatShares	0.8945	0.7429	1.5324	0.1143	-0.0287	-0.1250	-0.6611	0.4496	-0.1560	-0.0134
shortPercentOfFloat	-0.4662	-0.6597	1.8693	1.3889	0.0002	3.4157	1.6547	2.2082	0.1840	0.0013
heldPercentInsiders	0.8256	0.0881	0.2750	-0.0439	-0.0035	-0.1280	0.0175	1.0093	0.0642	-0.0027
heldPercentInstitutions	1.3881	1.0738	0.4151	-0.2445	-0.0012	-0.8157	-0.8532	0.0718	-0.0538	-0.0030
has_benefits	0.0818	-0.1767	-0.1295	0.0937	-1.9977	-0.0758	0.0618	0.3054	0.0069	-0.0004
Intercept	-0.7145	-0.2819	-2.5461	-1.1164	1.0034	-1.2260	-0.7462	-1.5371	-0.9864	-0.9954

Table 5: Colored weights of the linear SVM classifiers (BinarySMO) for each combination of classes. Positive values are shown in green, negative values in red, with softer intensity. The target classes 0: Strong Buy, 1: Buy, 2: Hold, 3: Underperformance, 4: Sell.

Multiclass *SVM* interpretation is not as simple as it is for binary classification. In this case, the model uses a one-vs-one binary classification for all combinations of target classes. Attempts were made to configure the *SVM* in different ways, but the one with the polynomial kernel and support vectors with not-too-rigid penalties for misclassifications has been proved to work the best. In the table 5 it is shown the weights of the linear classifiers for each combination of classes, using the data set with SMV wrapper although no feature selection data set has return the best result among all (with fewer variables the results can be interpreted better).

From the results The most influential variables for distinguishing between classes are primarily financial indicators: `ebitda`, `quickRatio`, `totalDebt`, and `trailingAnnualDividendYield`. These variables carry the largest weights, particularly in comparisons between extreme classes, "strong buy" (0) and "sell" (4), indicating that companies with higher or lower values in these metrics are more clearly associated with these extreme recommendations. Ownership-related variables, such as `shortPercentOfFloat` and `heldPercentInstitutions`, also play a significant role, especially in separating intermediate classes or refining distinctions between extremes. Positive weights in the table indicate that higher values of a variable push the model toward predicting the first class in the pair (e.g., 0 in 0vs4), whereas negative weights favor the second class (e.g., 4 in 0vs4). Some class combinations, particularly those involving adjacent recommendation levels (e.g., 2vs3 or 3vs4), show very small weights across most variables, suggesting that these classes are inherently harder to separate and the model relies less on individual features. As a curiosity, it is surprising to see that the separation between the "Buy" (1) and "Hold" (2) classes is based almost exclusively on whether the company is profitable or not.

4.2 Probabilistic Models

Models in this category output a probability distribution over the classes.

4.2.1 Logistic Regression

A **Logistic Regression** model has been used to try to predict the class to which each company corresponds. Its results are better than most of the probabilistic and non-probabilistic methods.

This model does require data preprocessing, as can be observed: both the PC (Percentage Correct) and AUC improve substantially when outliers are removed or a logarithm and standardization are applied so that the distributions are not extreme. In addition to improving convergence and thereby the result, adjusting the scales of all variables to be comparable with each other helps improve the interpretability of the coefficients. Moreover, in this case, where an error estimation with a penalty term using ridge was applied to avoid overfitting in case of many correlated variables (as can be seen by the fact that there is no difference in the result between the dataset treated with CBFSS and the dataset with all variables), a very large difference in scale causes instability in the calculation.

This model is very useful because, in addition to offering una buena precisión, unlike other models with similar performance such as *ANNs* or *SVMs*, *logistic regression* allows for the interpretation of its associated coefficients.

Specifically, for a training run with the subset of variables provided by the *logistic* wrapper, a series of *log-odds* coefficients were obtained. Table 6 shows the estimated *odds ratios* for each rating class compared to the reference class, 4 (“sell”). These values indicate how the odds ratio of belonging to a class X versus the “sell” class varies when an explanatory variable increases by one standardized unit.

Classifier Model: *Logistic Regression* with ridge parameter of 1.0E-8

Table 6: Logistic Regression Odds Ratios vs. Class 4 (”sell”)

Variable	**Class 0**	**Class 1**	**Class 2**	**Class 3**
In_SP500=1	0.0233	0.109	0.2205	0.4797
currentPrice	5.6121	3.3324	2.2643	1.5375
enterpriseValue	0.1367	0.5237	0.7496	1.0301
priceToBook	1.9018	1.5033	1.1645	1.1135
ebitda	7.71	7.0165	5.0655	4.7264
operatingCashflow	0.4701	0.2573	0.3952	0.3488
freeCashflow	0.5406	0.568	0.6413	0.7455
payoutRatio	0.9054	0.7962	0.9233	0.9674
sharesOutstanding	3.1593	2.9468	1.8801	1.5318
sharesPercentSharesOut	3.5036	2.4423	1.4536	1.3219
shortPercentOfFloat	0.1653	0.2081	0.3444	0.6111

A specific example can offer highly relevant information. For instance, companies that are part of the S&P 500 appear to be overvalued; if a company is part of the S&P 500,[1] its probability of being classified as ”strong buy” is ~42 times lower than being classified as ”sell,” and approximately double the probability of being classified as ”underperform” (however, one must be cautious, as this could be related to other variables in the subset).

It is also surprising to find that according to the model, the higher the `freeCashFlow`, the higher the probability of belonging to the ”sell” class compared to the rest. It must also be taken into account that this could be caused by the correlation of this variable with others included in the subset (see graph). It is reminded that CFSS was not used, but rather a Wrapper. Other information that is more intuitive would be the increase in `ebitda`, `shortPercentOfFloat`, or `sharesOutstanding`.

If the same model is trained with the variables from the subset given by CFSS (very similar accuracy), it is discovered that according to the model, analysts foresee that the best sectors

[1]The S&P 500 is a stock market index that tracks the performance of 500 large companies listed on stock exchanges in the United States. Because its constituent companies are used in a multitude of indices and it is a very widespread investment vehicle, the price of these companies can rise significantly.

to invest in are: *Basic Materials* and, far above, *Energy*, with the worst being *Consumer Cyclical* followed by *Real Estate*.

4.2.2 Discriminant Analysis

This is the only model that has not been tested for two reasons. The first is that it requires all variables to be numerical, and in all the subsets obtained by the FSS filter and multitude of Wrappers, categorical variables are chosen; that is, they are almost certainly important for classification. And although this is a problem that could be solved with various treatments to convert them into continuous numerical variables, the second and main reason is that they do not meet the main hypotheses required by the model, and therefore, a great result is not expected from this model.

It is required that all variables considered have a conditional probability density with the classes as a multivariate Gaussian. With a previous inspection of the marginal distributions of the variables for each of the classes (see graph), it can already be intuited, even if some transformation is applied to the variables. Even so, Mardia's test will be used to perform a hypothesis test on the probability distribution of our variables. The null hypothesis, therefore, will be that they follow a multivariate normal distribution for at least one class, and the alternative hypothesis that they do not. As the p-value is infinitesimal ($<$0.02) in all of them, the null hypothesis is rejected. With this, it is concluded that an adequate result will not be obtained for this model.

4.2.3 Bayesian Classifiers

Various Bayesian classifier models were used. In order of increasing accuracy, these were: standard **Naive Bayes**, a **Bayesian Network with a Greedy Stepwise search algorithm**, **Naive Bayes with a Kernel Density Estimator (KDE)**, the **Tree-Augmented Naive Bayes (TAN) network**, the **Averaged One-Dependence Estimator (AODE)**, and the **Averaged 2-Dependence Estimator (A2DE)**.

All these models rely on estimating the probability of a specific instance belonging to each class based on the probability distributions of the variables for each class. While results vary between models, they share common characteristics.

First, simpler models like the standard *Naive Bayes* require processed variables (e.g., on a logarithmic scale or with outliers removed) to better approximate Gaussian density functions for numerical features. This is confirmed in the results table (Ref to results table), which shows improved performance with processed data.

However, a more robust method is to use a Kernel Density Estimator (KDE). The KDE does not assume normality and instead applies non-parametric transformations to estimate probability distributions. By doing so, the *Naive Bayes with KDE* is able to outperform the initial *Bayesian network* that used transformed variables.

More sophisticated models (*TAN*, *AODE*, *A2DE*) appear to incorporate similar data handling internally, as their precision shows no significant change regardless of whether the input data is transformed or not.

Regarding feature selection, the datasets with the best performance were those processed with *Wrapper* filters (though this approach was computationally prohibitive to apply to all models) or those processed with *CFSS* for the simpler models. These simpler models are sensitive to variable correlation due to their assumption of *Class-Conditional Independence*. In contrast, more advanced models such as *TAN*, *AODE*, and *A2DE* perform well without external feature selection (FSS), as their algorithms inherently handle feature dependencies by allowing each feature to depend on a "parent" feature, thus mitigating the strict independence assumption. Additionally, a **Markov Blanket** filter was applied to the best-performing models (*TAN* and *AODE*), but it did not result in any significant improvement in their performance.

The Figure 2 shows the resulting network structure learned by the *TAN* classifier on the original, untreated dataset.

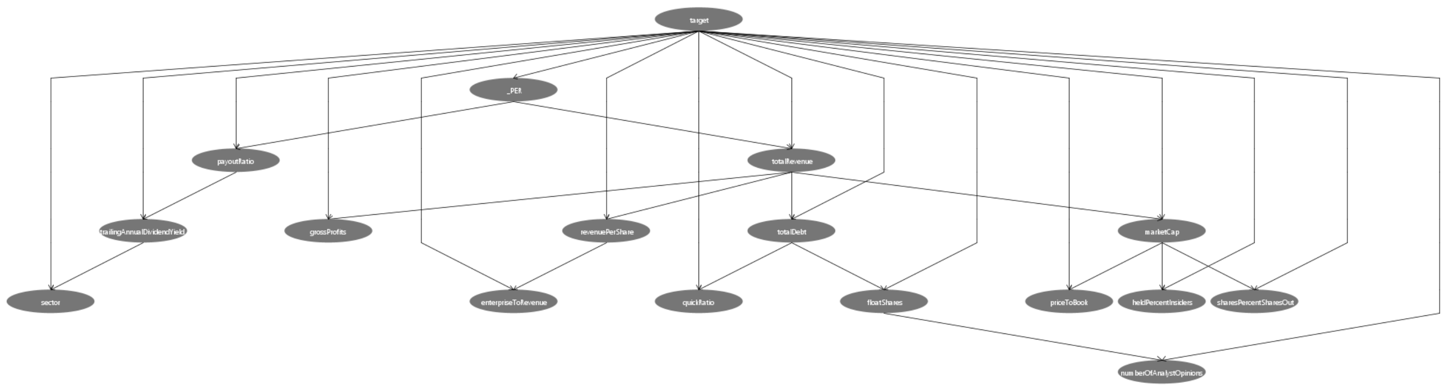

Figure 2: Tree-Augmented Naive Bayes (TAN) network structure.

Observe the dependencies captured by the *TAN* model. The `target` class variable is the central parent. Key financial metrics like `_PER`, `totalRevenue`, and `marketCap` act as parent nodes, influencing other variables. For example, `totalRevenue` directly influences `revenuePerShare` and `totalDebt`. This structure visualizes the inter-feature relationships that *TAN* models, which are ignored by the standard *Naive Bayes* classifier.

4.3 Metaclasifiers

The goal is to combine multiple models so that they complement each other, increasing predictive performance and improving generalization (i.e., making the ensemble less sensitive to noise). In order to improve the predictions the base models should be diverse and focus on different predictors so that they combine their strenght.

In addition to the base models several meta-classifiers were applied to enhance the performance of the best algorithms: *SVM (SMO)* the Bayesian models, *RIPPER* or the *Logistic Regression*. Besides, other hybrid algorithms were tried with very good results like **Logistic Model Trees (LMT)** and **RandomForest**.

The **Logistic Model Tree (LMT)** approach was evaluated. This method combines *decision trees* with *logistic regression* models applied at their leaves, which are optimized through a *boosting* process. Whereas **Random Forest** increases model diversity by randomly selecting the subset of features considered during each split.

On the other hand, **Bagging** and **AdaBoostM1** were combined with *SVM* to increase robustness against noise and improve boundary generalization. For the Bayesian classifiers (e.g., *Naive-Bayes*, *AODE*, *TAN*), *LogitBoost* were used to refine class probability estimates and mitigate the effects of conditional independence violations.

4.3.1 Conclusions on Meta-Classifier Performance

Meta-classifiers, produced the best overall results in the study. The **Stacking (SVM, A1DE, meta = Logistic)** configuration (*Model 29*) emerged as the absolute top performer, achieving the highest **Correct Percentage (37.80%)** and **AUC (0.69)** in the *TProc 1* dataset. These results highlight that **Stacking** is the most effective ensemble strategy for this problem, as it successfully combines the predictive strengths of diverse models—such as SVM (`SMO`) and Bayesian classifiers (`A1DE`)—while allowing a meta-classifier (`Logistic`) to learn higher-level decision patterns from their outputs.

Both stacking configurations (*Models 26 and 29*) outperform all other approaches, including their individual base models, confirming that it yields superior generalization when combining heterogeneous learners. Similarly, **Bagging(JRip)** demonstrates a substantial improvement over its base algorithm (`RIPPER`), increasing accuracy from 32.8% to 36.8%, confirming the benefit of variance reduction for rule-based methods.

In contrast, models such as **Bagging(Logistic)** and **Vote** fail to get meaningful gains. Bagging(Logistic) slightly underperforms the simple Logistic Regression (36.0% vs. 37.0%), while the Vote ensemble yields weaker results (35.5%) and exhibits an undesirably high RMSE (0.50), indicating inconsistent predictive confidence. Meanwhile, **LMT** and **AdaBoostM1(SVM)** perform very well on their own (around 37.0%), but their improvements relative to base models are very low.

Non-Probabilistic Models

Dataset Key: TMain: Train Dataset no processing; TStd: Train Dataset with standardization; TProc: Train Dataset with logistic transformation or outliers clipped. (1: No FSS, 2: Univariant, 3: Multivariant, 4: Wrapper)

Table 7: Correct Percentage (Non-Probabilistic)

Modelo	TMain 1	TMain 2	TMain 3	TMain 4	TStd 1	TStd 2	TStd 3	TStd 4	TProc 1	TProc 2	TProc 3	TProc 4
(1) RandomTree	29.97	30.24	29.36	-	28.40	28.66	28.40	-	29.87	29.16	29.21	-
(2) RandomTree.SMOTE	28.29	28.91	28.75	-	27.15	27.39	27.44	-	28.24	28.81	26.28	-
(3) J48	32.61	32.06	33.31	33.17	32.35	30.98	33.38	33.17	31.54	32.78	33.17	32.97
(4) J48.SMOTE	28.70	28.48	27.65	27.50	30.34	28.46	30.11	29.93	28.93	28.71	29.19	28.92
(7) MultilayerPerceptron	30.32	32.12	32.21	32.46	31.29	32.12	32.19	32.41	32.83	32.50	32.21	32.51
(8) MultilayerPerceptron.SMOTE	30.48	28.43	29.65	29.83	28.97	27.76	29.80	29.96	30.62	30.37	30.51	30.64
(11) RIPPER	32.21	32.32	32.91	32.91	32.39	32.03	32.87	32.87	32.83	31.86	32.23	32.23
(12) RIPPER.SMOTE	31.91	33.31	33.07	33.07	32.65	33.17	33.45	33.45	31.53	32.85	33.17	33.17
(13) OneRule	27.98	28.95	28.61	28.61	28.61	29.08	29.10	29.10	30.22	30.57	28.59	28.59
(14) OneRule.SMOTE	25.08	24.89	25.65	25.65	24.77	25.92	26.02	26.02	25.58	26.23	24.38	24.38
(15) DecisionTable	33.23	33.40	33.97	33.97	33.18	33.54	33.89	33.89	32.08	31.41	32.72	32.72
(16) DecisionTable.SMOTE	27.80	28.97	28.18	28.18	27.91	28.14	28.29	28.29	28.26	28.49	27.13	27.13
(17) K-NN	28.70	30.84	29.91	29.78	28.70	30.84	29.91	29.73	30.07	30.84	29.58	29.43
(18) K-NN.SMOTE	28.68	29.39	28.92	28.72	28.25	30.03	28.40	28.21	29.90	30.57	28.29	28.11
(19) K*-NN	13.92	13.92	14.01	14.48	28.40	32.61	30.45	30.79	30.89	31.61	31.11	31.35
(20) K*-NN.SMOTE	13.92	13.92	14.01	14.37	27.62	31.42	30.22	30.33	30.07	30.72	29.46	29.63
(21) SVM (Polynomial)	33.38	32.01	32.17	32.11	33.47	32.01	32.06	31.95	36.50	32.95	34.19	34.08
(22) SVM (Polynomial).SMOTE	28.70	25.17	27.58	27.40	28.75	25.71	27.15	27.00	34.36	29.88	31.28	31.14
(23) RBFClassifier	35.29	32.96	34.04	34.29	34.78	33.00	33.97	34.19	36.65	34.27	35.65	35.82

Table 8: Weighted Avg. AUC (Non-Probabilistic)

Modelo	TMain 1	TMain 2	TMain 3	TMain 4	TStd 1	TStd 2	TStd 3	TStd 4	TProc 1	TProc 2	TProc 3	TProc 4
(1) RandomTree	0.55	0.55	0.55	-	0.54	0.54	0.54	-	0.55	0.54	0.54	-
(2) RandomTree.SMOTE	0.54	0.55	0.55	-	0.54	0.54	0.54	-	0.54	0.54	0.53	-
(3) J48	0.61	0.61	0.62	0.61	0.60	0.60	0.62	0.61	0.59	0.59	0.60	0.59
(4) J48.SMOTE	0.56	0.55	0.54	0.53	0.56	0.56	0.57	0.56	0.55	0.56	0.56	0.55
(7) MultilayerPerceptron	0.60	0.61	0.61	0.62	0.60	0.61	0.61	0.62	0.61	0.61	0.61	0.62
(8) MultilayerPerceptron.SMOTE	0.61	0.61	0.62	0.63	0.60	0.60	0.61	0.62	0.61	0.61	0.60	0.61
(11) RIPPER	0.54	0.54	0.55	0.55	0.54	0.53	0.55	0.55	0.55	0.53	0.54	0.54
(12) RIPPER.SMOTE	0.58	0.57	0.57	0.57	0.58	0.57	0.58	0.58	0.56	0.57	0.57	0.57
(13) OneRule	0.52	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.54	0.52	0.52
(14) OneRule.SMOTE	0.54	0.53	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.53	0.53
(15) DecisionTable	0.62	0.63	0.63	0.63	0.62	0.63	0.63	0.63	0.59	0.61	0.61	0.61
(16) DecisionTable.SMOTE	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.60	0.59	0.60	0.58	0.58
(17) K-NN	0.54	0.55	0.55	0.54	0.54	0.55	0.55	0.54	0.55	0.55	0.54	0.53
(18) K-NN.SMOTE	0.55	0.56	0.55	0.54	0.54	0.56	0.55	0.54	0.55	0.56	0.54	0.53
(19) K*-NN	0.51	0.51	0.51	0.52	0.57	0.61	0.59	0.60	0.59	0.60	0.59	0.60
(20) K*-NN.SMOTE	0.51	0.51	0.51	0.52	0.58	0.62	0.59	0.60	0.59	0.61	0.60	0.61
(21) SVM (Polynomial)	0.61	0.56	0.58	0.57	0.61	0.56	0.58	0.57	0.65	0.59	0.61	0.60
(22) SVM (Polynomial).SMOTE	0.62	0.59	0.60	0.59	0.61	0.59	0.60	0.59	0.65	0.61	0.62	0.61
(23) RBFClassifier	0.65	0.63	0.64	0.65	0.65	0.62	0.65	0.66	0.67	0.65	0.66	0.67

Table 9: Root Mean Squared Error (RMSE) (Non-Probabilistic)

Modelo	TMain 1	TMain 2	TMain 3	TMain 4	TStd 1	TStd 2	TStd 3	TStd 4	TProc 1	TProc 2	TProc 3	TProc 4
(1) RandomTree	0.52	0.52	0.52	-	0.53	0.52	0.53	-	0.52	0.52	0.52	-
(2) RandomTree.SMOTE	0.53	0.52	0.53	-	0.53	0.53	0.53	-	0.53	0.52	0.53	-
(3) J48	0.40	0.39	0.39	0.40	0.40	0.39	0.39	0.40	0.40	0.39	0.39	0.40
(4) J48.SMOTE	0.50	0.49	0.49	0.50	0.49	0.49	0.49	0.50	0.49	0.49	0.49	0.50
(7) MultilayerPerceptron	0.42	0.38	0.39	0.38	0.42	0.38	0.39	0.38	0.47	0.40	0.42	0.41
(8) MultilayerPerceptron.SMOTE	0.43	0.39	0.40	0.39	0.43	0.40	0.40	0.39	0.48	0.41	0.42	0.41
(11) RIPPER	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38
(12) RIPPER.SMOTE	0.40	0.40	0.40	0.40	0.40	0.39	0.39	0.39	0.40	0.39	0.39	0.39
(13) OneRule	0.53	0.53	0.53	0.53	0.53	0.52	0.52	0.52	0.52	0.52	0.53	0.53
(14) OneRule.SMOTE	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54	0.54
(15) DecisionTable	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38
(16) DecisionTable.SMOTE	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39
(17) K-NN	0.53	0.52	0.52	0.53	0.53	0.52	0.52	0.53	0.52	0.52	0.52	0.53
(18) K-NN.SMOTE	0.53	0.52	0.52	0.53	0.53	0.52	0.53	0.54	0.52	0.52	0.53	0.54
(19) K*-NN	0.39	0.39	0.39	0.38	0.50	0.42	0.44	0.43	0.49	0.43	0.44	0.43
(20) K*-NN.SMOTE	0.39	0.39	0.39	0.38	0.50	0.43	0.44	0.43	0.49	0.43	0.44	0.43
(21) SVM (Polynomial)	0.38	0.38	0.38	0.39	0.38	0.38	0.38	0.39	0.37	0.38	0.38	0.39
(22) SVM (Polynomial).SMOTE	0.39	0.39	0.39	0.40	0.39	0.39	0.39	0.40	0.38	0.38	0.38	0.39
(23) RBFClassifier	0.37	0.38	0.38	0.37	0.37	0.38	0.38	0.37	0.37	0.37	0.37	0.36

Probabilistic Models (Experiment 2)

Dataset Key: TMain: Train Dataset no processing; TStd: Train Dataset with standardization; TProc: Train Dataset with logistic transformation or outliers clipped. (1: No FS, 2: Univariant, 3: Multivariant, 4: Wrapper)

Table 10: Correct Percentage (Experiment 2 - Probabilistic)

Modelo	**TMain 1**	**TMain 2**	**TMain 3**	**TMain 4**	**TStd 1**	**TStd 2**	**TStd 3**	**TStd 4**	**TProc 1**	**TProc 2**	**TProc 3**	**TProc 4**
(1) NaiveBayes	22.77	20.66	24.29	24.55	22.77	20.57	24.33	24.55	24.16	23.70	30.96	31.14
(2) NaiveBayes.SMOTE	21.03	17.06	23.63	23.75	20.61	16.30	23.27	23.35	23.62	21.80	27.25	27.40
(3) NaiveBayes (con kernel)	24.28	21.55	26.72	26.89	24.29	21.64	26.60	26.79	29.70	31.13	33.56	33.78
(4) NaiveBayes SMOTE (con kernel)	19.78	18.37	22.75	22.94	19.50	17.98	22.81	23.00	25.45	25.83	30.15	30.33
(5) A1DE	34.27	34.17	34.84	34.69	34.36	34.20	34.81	34.69	34.02	34.39	34.75	34.59
(6) A1DE.SMOTE	31.54	30.55	31.49	31.35	31.98	31.22	31.85	31.75	31.13	31.37	31.46	31.35
(7) A2DE	35.02	34.15	34.08	-	35.02	34.22	34.15	-	34.01	35.02	34.94	-
(8) A2DE.SMOTE	31.47	30.55	31.45	-	31.38	31.13	31.71	-	31.61	30.97	32.11	-
(9) BayesNet (TAN)	33.29	33.84	33.68	33.47	33.36	34.01	33.71	33.57	33.49	34.49	33.87	33.68
(10) BayesNet (TAN).SMOTE	31.98	31.31	31.59	31.45	31.76	30.61	31.12	30.94	32.21	31.00	30.46	30.33
(11) BayesNet (Greedy)	31.53	32.10	33.84	-	31.53	32.06	33.88	-	31.49	31.46	33.27	-
(12) BayesNet (Greedy).SMOTE	30.66	30.42	30.84	-	30.00	29.56	30.83	-	30.27	30.14	30.79	-
(13) NBTree	31.01	32.12	33.84	-	31.13	31.98	33.36	-	32.06	31.58	32.78	-
(14) NBTree.SMOTE	29.58	29.06	29.29	-	28.05	28.27	28.72	-	28.81	28.36	28.79	-
(15) Logistic	34.51	32.13	33.89	34.19	34.51	32.13	33.88	34.08	37.00	34.37	36.97	37.15
(16) Logistic.SMOTE	31.83	27.80	30.88	31.14	32.10	28.00	30.68	30.84	35.48	31.01	33.03	33.17
(17) SimpleLogistic	33.54	31.51	33.07	33.07	33.54	31.51	33.07	33.07	36.97	34.16	36.77	36.77
(18) SimpleLogistic.SMOTE	31.21	28.49	30.89	30.89	31.67	28.16	30.51	30.51	35.65	31.54	32.72	32.72

Table 11: Weighted Avg. AUC (Experiment 2 - Probabilistic)

Modelo	**TMain 1**	**TMain 2**	**TMain 3**	**TMain 4**	**TStd 1**	**TStd 2**	**TStd 3**	**TStd 4**	**TProc 1**	**TProc 2**	**TProc 3**	**TProc 4**
(1) NaiveBayes	0.58	0.55	0.59	0.60	0.58	0.55	0.59	0.60	0.62	0.61	0.63	0.64
(2) NaiveBayes.SMOTE	0.58	0.55	0.58	0.59	0.58	0.55	0.58	0.59	0.61	0.61	0.62	0.63
(3) NaiveBayes (con kernel)	0.57	0.55	0.59	0.60	0.58	0.55	0.59	0.60	0.63	0.63	0.64	0.65
(4) NaiveBayes SMOTE (con kernel)	0.57	0.55	0.59	0.60	0.57	0.55	0.59	0.60	0.62	0.63	0.64	0.65
(5) A1DE	0.66	0.65	0.66	0.65	0.66	0.65	0.66	0.65	0.65	0.65	0.65	0.64
(6) A1DE.SMOTE	0.65	0.65	0.65	0.64	0.65	0.64	0.65	0.64	0.64	0.64	0.64	0.63
(7) A2DE	0.66	0.65	0.66	-	0.66	0.65	0.66	-	0.65	0.65	0.65	-
(8) A2DE.SMOTE	0.65	0.64	0.65	-	0.65	0.64	0.65	-	0.64	0.64	0.64	-
(9) BayesNet (TAN)	0.65	0.64	0.64	0.63	0.65	0.64	0.64	0.63	0.64	0.64	0.64	0.63
(10) BayesNet (TAN).SMOTE	0.64	0.64	0.64	0.63	0.64	0.63	0.64	0.63	0.64	0.63	0.63	0.62
(11) BayesNet (Greedy)	0.63	0.63	0.65	-	0.63	0.63	0.65	-	0.63	0.63	0.64	-
(12) BayesNet (Greedy).SMOTE	0.63	0.63	0.64	-	0.63	0.63	0.64	-	0.63	0.63	0.64	-
(13) NBTree	0.60	0.62	0.64	-	0.60	0.62	0.64	-	0.59	0.61	0.63	-
(14) NBTree.SMOTE	0.58	0.58	0.58	-	0.56	0.58	0.58	-	0.57	0.58	0.57	-
(15) Logistic	0.65	0.62	0.64	0.65	0.65	0.62	0.64	0.65	0.67	0.64	0.66	0.67
(16) Logistic.SMOTE	0.64	0.62	0.64	0.65	0.64	0.62	0.63	0.64	0.67	0.64	0.65	0.66
(17) SimpleLogistic	0.65	0.61	0.64	0.64	0.65	0.61	0.64	0.64	0.67	0.64	0.66	0.66
(18) SimpleLogistic.SMOTE	0.65	0.62	0.64	0.64	0.65	0.62	0.63	0.62	0.67	0.64	0.65	0.65

Table 12: Root Mean Squared Error (RMSE) (Experiment 2 - Probabilistic)

Modelo	**TMain 1**	**TMain 2**	**TMain 3**	**TMain 4**	**TStd 1**	**TStd 2**	**TStd 3**	**TStd 4**	**TProc 1**	**TProc 2**	**TProc 3**	**TProc 4**
(1) NaiveBayes	0.53	0.52	0.50	0.49	0.53	0.52	0.50	0.49	0.50	0.47	0.42	0.41
(2) NaiveBayes.SMOTE	0.54	0.53	0.50	0.49	0.54	0.53	0.50	0.49	0.52	0.50	0.44	0.43
(3) NaiveBayes (con kernel)	0.53	0.53	0.49	0.48	0.53	0.53	0.49	0.48	0.47	0.44	0.41	0.40
(4) NaiveBayes SMOTE (con kernel)	0.54	0.52	0.49	0.48	0.54	0.52	0.49	0.48	0.50	0.47	0.43	0.42
(5) A1DE	0.40	0.39	0.39	0.40	0.40	0.39	0.39	0.40	0.40	0.39	0.38	0.39
(6) A1DE.SMOTE	0.42	0.40	0.40	0.41	0.43	0.40	0.40	0.41	0.43	0.40	0.41	0.42
(7) A2DE	0.39	0.38	0.38	-	0.39	0.38	0.38	-	0.39	0.38	0.38	-
(8) A2DE.SMOTE	0.42	0.40	0.40	-	0.42	0.40	0.40	-	0.42	0.40	0.40	-
(9) BayesNet (TAN)	0.40	0.38	0.39	0.40	0.40	0.38	0.39	0.40	0.39	0.38	0.38	0.39
(10) BayesNet (TAN).SMOTE	0.41	0.40	0.40	0.41	0.42	0.40	0.40	0.41	0.41	0.40	0.40	0.41
(11) BayesNet (Greedy)	0.44	0.42	0.41	-	0.44	0.42	0.41	-	0.44	0.42	0.40	-
(12) BayesNet (Greedy).SMOTE	0.46	0.44	0.43	-	0.46	0.44	0.43	-	0.46	0.44	0.43	-
(13) NBTree	0.43	0.41	0.41	-	0.44	0.41	0.40	-	0.43	0.41	0.40	-
(14) NBTree.SMOTE	0.47	0.44	0.44	-	0.47	0.44	0.44	-	0.47	0.44	0.44	-
(15) Logistic	0.38	0.38	0.38	0.37	0.38	0.38	0.38	0.37	0.38	0.38	0.37	0.36
(16) Logistic.SMOTE	0.39	0.39	0.39	0.38	0.39	0.39	0.39	0.38	0.38	0.38	0.38	0.37
(17) SimpleLogistic	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.38	0.37	0.37	0.37	0.37
(18) SimpleLogistic.SMOTE	0.38	0.39	0.39	0.39	0.38	0.38	0.38	0.39	0.38	0.38	0.38	0.38

Metaclassifier Models (Selection)

Dataset Key: TMain: Train Dataset no processing; TStd: Train Dataset with standardization; TProc: Train Dataset with logistic transformation or outliers clipped. (1: No FS, 2: Univariant, 3: Multivariant, 4: Wrapper)

Table 13: Correct Percentage (Metaclassifiers)

Modelo	TMain 1	TMain 2	TMain 3	TMain 4	TStd 1	TStd 2	TStd 3	TStd 4	TProc 1	TProc 2	TProc 3	TProc 4
(5) trees.RandomForest	35.59	32.26	33.79	-	35.79	32.61	33.71	-	35.57	33.14	33.51	-
(6) RandomForest.SMOTE	32.78	30.72	30.89	-	32.10	30.95	31.31	-	32.67	31.63	31.44	-
(9) trees.LMT	33.64	31.36	33.15	-	33.49	31.71	33.58	-	37.01	34.27	36.77	-
(10) LMT.SMOTE	31.12	29.21	30.88	-	30.77	29.48	32.15	-	33.21	29.90	31.36	-
(21) meta.Bagging(NaiveBayes)	20.55	-	-	-	20.52	-	-	-	24.33	-	-	-
(22) Bagging(NaiveBayes).SMOTE	22.92	-	-	-	21.28	-	-	-	23.96	-	-	-
(25) AdaBoostM1(SMV)	35.80	-	-	-	35.90	-	-	-	36.60	-	-	-
(26) Stacking(A1DE, meta=SMV)	35.90	-	-	-	36.00	-	-	-	37.50	-	-	-
(27) meta.Bagging(JRip)	35.25	-	-	-	35.30	-	-	-	36.80	-	-	-
(28) meta.Bagging(SMV)	35.51	-	-	-	35.60	-	-	-	36.40	-	-	-
(29) Stacking(SMV,A1DE,meta=Log)	37.32	-	-	-	37.40	-	-	-	37.80	-	-	-
(30) Vote(SMV,A1DE,Log)	34.98	-	-	-	35.10	-	-	-	35.50	-	-	-

Table 14: Weighted Avg. AUC (Metaclassifiers)

Modelo	TMain 1	TMain 2	TMain 3	TMain 4	TStd 1	TStd 2	TStd 3	TStd 4	TProc 1	TProc 2	TProc 3	TProc 4
(5) trees.RandomForest	0.66	0.64	0.64	-	0.65	0.63	0.64	-	0.64	0.63	0.62	-
(6) RandomForest.SMOTE	0.65	0.63	0.63	-	0.65	0.63	0.63	-	0.64	0.62	0.62	-
(9) trees.LMT	0.65	0.61	0.64	-	0.65	0.62	0.64	-	0.67	0.64	0.66	-
(10) LMT.SMOTE	0.62	0.59	0.61	-	0.62	0.59	0.61	-	0.64	0.59	0.61	-
(21) meta.Bagging(NaiveBayes)	0.59	-	-	-	0.59	-	-	-	0.62	-	-	-
(22) Bagging(NaiveBayes).SMOTE	0.58	-	-	-	0.58	-	-	-	0.62	-	-	-
(25) AdaBoostM1(SMV)	0.66	-	-	-	0.66	-	-	-	0.67	-	-	-
(26) Stacking(A1DE, meta=SMV)	0.67	-	-	-	0.67	-	-	-	0.68	-	-	-
(27) meta.Bagging(JRip)	0.64	-	-	-	0.64	-	-	-	0.67	-	-	-
(28) meta.Bagging(SMV)	0.64	-	-	-	0.64	-	-	-	0.66	-	-	-
(29) Stacking(SMV,A1DE,meta=Log)	0.68	-	-	-	0.68	-	-	-	0.69	-	-	-
(30) Vote(SMV,A1DE,Log)	0.65	-	-	-	0.65	-	-	-	0.66	-	-	-

Table 15: Root Mean Squared Error (RMSE) (Metaclassifiers)

Modelo	TMain 1	TMain 2	TMain 3	TMain 4	TStd 1	TStd 2	TStd 3	TStd 4	TProc 1	TProc 2	TProc 3	TProc 4
(5) trees.RandomForest	0.37	0.38	0.38	-	0.37	0.38	0.38	-	0.37	0.38	0.38	-
(6) RandomForest.SMOTE	0.38	0.39	0.38	-	0.38	0.39	0.38	-	0.38	0.39	0.39	-
(9) trees.LMT	0.38	0.38	0.38	-	0.38	0.38	0.38	-	0.37	0.38	0.37	-
(10) LMT.SMOTE	0.43	0.45	0.43	-	0.44	0.47	0.45	-	0.42	0.45	0.44	-
(21) meta.Bagging(NaiveBayes)	0.47	-	-	-	0.47	-	-	-	0.47	-	-	-
(22) Bagging(NaiveBayes).SMOTE	0.48	-	-	-	0.49	-	-	-	0.49	-	-	-
(25) AdaBoostM1(SMV)	0.40	-	-	-	0.40	-	-	-	0.39	-	-	-
(26) Stacking(A1DE, meta=SMV)	0.37	-	-	-	0.37	-	-	-	0.36	-	-	-
(27) meta.Bagging(JRip)	0.38	-	-	-	0.38	-	-	-	0.37	-	-	-
(28) meta.Bagging(SMV)	0.38	-	-	-	0.38	-	-	-	0.37	-	-	-
(29) Stacking(SMV,A1DE,meta=Log)	0.38	-	-	-	0.38	-	-	-	0.37	-	-	-
(30) Vote(SMV,A1DE,Log)	0.51	-	-	-	0.51	-	-	-	0.50	-	-	-

5 Conclusions

The comparative analysis of the models reveals that the quality of the data and a proper preprocessing has more influence on model performance than either feature selection or resampling strategies. The **TProc 1 dataset (processed, without feature reduction)** consistently results to be the most effective configuration in the most simple models, and achieving top results across nearly all. This, combined with applying a great Feature Selection (using Filters) provide meaningful improvements. Univariate filters (Mutual Information) although it improves a little the performance, tend to oversimplify the feature space, not removing very correlated predictors, while multivariate evaluators (CFS) effectively reduce redundancy but may discard attributes crucial to specific models. The **Wrapper** method is a partial exception, as it maintains performance for noise-sensitive algorithms (e.g., NaiveBayes), yet still does not outperform the use of the full feature set and it can be very computationally expensive in some cases.

The **SMOTE oversampling technique** proves to be ineffective in this context. In nearly all non probabilistic cases, and all the probabilistic its application leads to a decline in accuracy and an increase in RMSE, suggesting that synthetic data generation fails to enhance generalization, likely due to the intrinsic class overlap that exists. How ever it is true that its application makes a significant amount of cases a reduction of the fatal errors (classifying 0 into 4 or viceversa).

Regarding model performance, the *Logistic Regression*, *Simple Logistic*, *RBFClassifier*, and *SVM* models along with the meta-classifiers *LMT (Logistic Model Trees)*, *RandomForest* and **Stacking** two or more of the previous (specially if uses SVM), outperforme the rest clearly. These models exhibit the highest accuracy scores (up to 37.80%), the lowest RMSE ($\approx$ 0.36), and strong discriminative ability (AUC $\approx$ 0.69). Their success lies in their ability to capture nonlinear relationships and complex interactions between variables without requiring aggressive feature selection. Conversely, *NaiveBayes*, *OneRule*, and *K** consistently perform poorly, demonstrating their sensitivity to high-dimensional data and correlated predictors. Although this simplistic models give much information about the nature of variables and their intrinsic relation with the problem. Are very useful to extract simple rules that may work as *rules of thumb*.

However, the numbers (specially the percentage of correct classifications) might sound low but it has to be taken into account that there are 5 classes so it is really hard to correctly set every result in each one. Besides, it is not clear that there is a objective rule for determine whether a company should be in one group or another; even the experts disagree in most cases. If the companies were grouped into fewer classes, 3 for instance, a much higher number would be obtain.

Regarding the dificulty of separating the clases: here it is the ROC curve for the Random forest classifier across 5 classes:

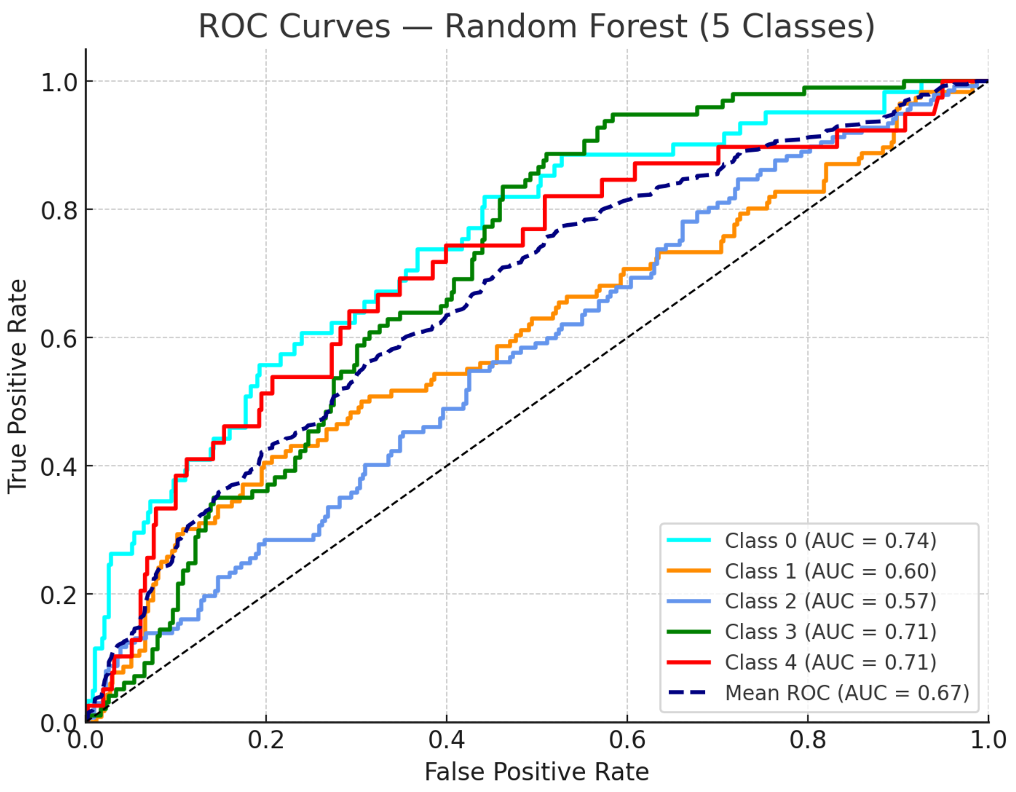

Figure 3: Multiclass ROC curve.

The multi-class (ROC) curve shows the trade-off between True Positive Rate and False Positive Rate for each class against all others (one-vs-rest approach). The Area Under the Curve (AUC) values provide a robust metric for assessing the model's ability to distinguish between classes.

The area under the ROC curve shows strong performance for certain classes, particularly Class 0 (AUC = 0.78), Class 3 (AUC = 0.71) and Class 4 (AUC = 0.71), the extreme ones. Conversely, Class 1 (AUC = 0.60) and Class 2 (AUC = 0.57) exhibit weaker performance, suggesting more overlap with other classes and a greater challenge for the model to differentiate them accurately. The mean ROC AUC of 0.67 highlights the overall discriminative capability of the LMT model across all classes.

This indicates that the LMT model has varying levels of success in its different categories, highlighting challenges in distinguishing between classes "buy" and "hold" as it has been shown before, due to inherent class overlap and complexity.

Reviewing the literature, this results were also found in previous analysis, such as in the study of corporate bond classification (AAA, AA+, B, C, CCC...) based on companies' financial metrics (Assign a reference here). It is explained that the best predictions are those obtained through complex *ANNs* and through *SVMs*; in them, up to 23% accuracy was obtained if they were separated into the different classes and up to 87% when grouping them into 3[1].

References

[1] Daniel Caridad, Jana Hančlová, Hosn el Woujoud Bousselmi, and Lorena Caridad y López del Río (2019). *Corporate rating forecasting using Artificial Intelligence statistical techniques*. *Investment Management and Financial Innovations*, 16(2), 295–312. doi:10.21511/imfi.16(2).2019.25.

[2] Pedro Larrañaga and Concha Bielza (s.f.). *Machine Learning Lectures*. Computational Intelligence Group, Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid.