Dynamic Classification of S&P 500 Stocks Using Distributed Computing and Fundamental Ratios

Candidate numbers: 50714, 49775, 49872, 49663

Abstract

This project addresses the challenge of classifying S&P 500 stocks into five benchmark-relative performance tiers using a combination of financial ratios and macroeconomic indicators. To handle the scale and complexity of 146,165 structured records, a distributed machine learning pipeline is developed using Apache Spark on Google Cloud Dataproc. The study implements and compares eight supervised models, including classical algorithms, gradient-boosted methods, H2O AutoML, and an F1-weighted ensemble. Oversampling and SHAP analysis are applied to address class imbalance and interpret feature importance. Results show that AutoML achieves the highest F1 score (0.8940), with a 18% reduction in training time from 2-node to 3-node clusters. LightGBM demonstrates the best trade-off between efficiency and accuracy (F1 score 0.4983 and execution time 35.12s). The pipeline confirms that distributed AutoML and ensemble learning offer scalable and effective solutions for financial classification tasks in big-data environments.

1 Introduction

This study aims to classify S&P 500 stocks into five performance groups—Strong Underperformance, Slight Underperformance, Neutral, Slight Outperformance, and Strong Outperformance—based on their benchmark-relative returns from 2009 to 2024. This multi-class setup supports more informed investment decisions by identifying subtle variations in stock performance, helping manage risk and optimize portfolios.

Earlier studies, like Caprariini et al. [2], used tree-based models for binary classification, reaching only 40.4% accuracy and overlooking multi-class patterns. To address these challenges, we propose a scalable and distributed solution leveraging Apache Spark on Google Cloud Dataproc. Our pipeline is designed to process over 146,000 samples with 42 input variables, using parallel computation across cluster nodes. We implement and evaluate six supervised models—Logistic Regression, Decision Tree, Random Forest, GBT-Classifier, XGBoost, LightGBM—alongside an H2O AutoML pipeline and an F1-weighted Ensemble classifier. This comprehensive setup allows us to compare traditional, gradient-boosted, and automated modeling approaches. The solution is appropriate as it combines distributed computing with model diversity to balance prediction accuracy and computational efficiency. Using PySpark ensures compatibility with big data pipelines, while AutoML and ensemble strategies improve robustness under imbalanced class distributions.

Results show that increasing cluster size from 2 to 3 nodes significantly reduces training time for most models, with AutoML dropping by 18% (1502.89s to 1242.22s). In terms of prediction, AutoML achieved the best F1 score (0.8940), while LightGBM ranked second (0.4983) and completed training in just 35.12 seconds on 3 nodes, showcasing its efficiency and strong performance tradeoff.

Our findings confirm that combining distributed computing with AutoML and advanced gradient-boosted methods is effective for scalable, high-accuracy stock classification in complex financial environments.

2 Related work

Explanatory variables, sometimes referred to as factors, have long been studied by researchers in an effort to identify market inefficiencies and promote asset pricing. Sharpe's Capital Asset Pricing Model (CAPM) introduced beta as the first factor [14], followed by the multi-factor models of Fama and French [6, 7], which incorporate size, value, and profitability dimensions. While factor-based approaches are widely used, traditional linear models often struggle to capture complex and nonlinear relationships between explanatory variables and returns.

In recent years, machine learning (ML) techniques have gained traction in stock selection due to their ability to model nonlinear patterns and improve predictive performance. For stock performance classification problem, recent precedents are as follows: Caparrini et al. [2] employed tree-based classifiers – Decision Tree, Random Forest, and XGBoost—to classify S&P 500 stocks based on expected profitability. They showed that retraining models over time enables portfolios to outperform the benchmark, with the best model achieving 40.4% accuracy. Tekin and Gümüş [15] applied clustering analysis based on financial ratios such as P/E and P/B to identify similar stocks for portfolio construction. Jidong and Ran [9] utilized XGBoost to conduct stock selection based on financial fundamentals, and Fu et al. [8] demonstrated the effectiveness of random forest classifiers in a similar context.

Additionally, Zhang et al. [17] explored deep learning with neural networks to predict stock trends, achieving 45% accuracy on high-frequency data, though scalability remained a challenge. Chen and He [3] introduced a hybrid approach combining Support Vector Machines and genetic algorithms, improving precision by 12% over baseline models on Asian market data.

Our research addresses a multi-class classification challenge, assigning S&P 500 stocks to five performance tiers based on benchmark-relative returns, tackling high-dimensional, big-data complexities. We deploy a scalable Apache Spark pipeline to process vast datasets efficiently. Beyond tree-based models, we evaluate advanced techniques—LightGBM, GBT with One-vs-Rest, and H2O AutoML—to determine if sophisticated methods enhance classification in this data-rich financial domain. Critically, while prior work focused on binary or profitability-based tasks, our five-class setup demands greater model adaptability, potentially explaining why simpler models like Decision Tree lagged (**e.g., 29.2%** accuracy [9]). The inclusion of distributed computing and automated ensembling aims to push beyond these limits, leveraging Spark's parallel processing to handle 146,165 records, a scale often overlooked in earlier studies.

3 Data Description

This study classifies S&P 500 stocks using both **firm-specific financial ratios** and **macroeconomic indicators**. These two variable

sets reflect, respectively, the internal financial health of firms and the external economic environment. Their integration captures the multidimensional basis upon which investors assess risk, value, and return. On one hand, macro indicators influence both systematic risk and investor sentiment; they play a central role in how stocks are priced and classified across economic regimes. On the other hand, financial ratios contribute directly to stock classification and market performance by influencing how investors assess firm quality and risk.

Macroeconomic Indicators.

- These indicators shape the environment in which firms operate and influence systematic risk and sentiment.
- **Growth metrics** (e.g., GDP, PMI) reflect economic momentum and earnings expectations, affecting cyclical asset demand [11].
- **Inflation and rate data** (e.g., CPI, Fed Funds Rate) affect discounting and capital costs, prompting portfolio shifts [5].
- **Labour and consumer signals** (e.g., unemployment, retail sales) inform demand conditions, especially in consumption-linked sectors.

Financial Ratios.

- Financial ratios offer a structured lens into firm risk and performance, central to valuation and classification decisions.
- **Liquidity ratios** (e.g., current ratio) measure short-term solvency, influencing investor preference during stress [16].
- **Leverage metrics** (e.g., debt-to-equity) reflect risk from financial fragility and interest rate exposure [1].
- **Profitability indicators** (e.g., ROE, net margin) capture earnings efficiency, supporting confidence and multiples [12, 13].
- **Valuation ratios** (e.g., P/E, EV/EBITDA) guide price-to-fundamental assessments, shaped by expectations and biases [4, 10].

Table 1: Construction of classification target variable

Label	Return Condition	Performance
1	$y < 0.5x$	Strong underperformance
2	$0.5x \leq y < 0.9x$	Slight underperformance
3	$0.9x \leq y < 1.1x$	Neutral performance
4	$1.1x \leq y < 1.5x$	Slight outperformance
5	$y \geq 1.5x$	Strong outperformance

The dependent variable is a five-class ordinal indicator reflecting each stock's relative return against the S&P 500. Let x be the index return and y the stock return over a fixed interval; the ratio y/x determines the classification. This benchmark-relative scheme captures underperformance, alignment, or outperformance in a more granular fashion than binary labels, aligning with common practices in performance attribution and active portfolio analysis [14]. The classification thresholds are detailed in Table 1.

Based on the benchmark-relative classification scheme described above, the explanatory variables cover a wide range of firm-specific financial ratios and macroeconomic indicators that are relevant to stock performance. All data are collected from the Bloomberg Terminal and span the period from January 2009 to December 2024. The indicators include: Current Ratio, Quick Ratio, Cash Ratio, Debt-to-Equity Ratio, Debt Ratio, Interest Coverage Ratio, Return on Assets (ROA), Return on Equity (ROE), Net Profit Margin, Operating Margin, Gross Margin, Price-to-Earnings (P/E), Price-to-Book (P/B), Price-to-Sales (P/S), Price-to-Cash Flow, EV/EBITDA, Dividend Yield, GDP Growth (%), CPI (%), PMI, Unemployment Rate (%), Retail Sales Growth (%), Net Export Growth (%), IIP (Industrial Production Index), 10-year Treasury Yield (%), Saving Rate (%), New Home Sales Growth (%), Total Assets Score, Total Equity Score, Net Income Score, Net Sales Score, Market Cap Score and Fed Funds Rate (%). A full list of the features, along with their descriptions, frequency, and category, is provided in **Appendix B**.

The datasets used in this study—including firm-level financial ratios and macroeconomic indicators—are now publicly available via Kaggle at https://www.kaggle.com/datasets/laxmansudhan25002/st446-macroeconomic-and-fundamental-data, which facilitates reproducibility and external validation.

4 Methodology

This section outlines the end-to-end pipeline adopted to build and evaluate classification models on financial and macroeconomic data. The workflow begins with data extraction from Bloomberg and FRED, followed by integration, cleaning, and transformation using Apache Spark on Google Cloud Dataproc. The structured dataset is then used to train eight classification models, including classical algorithms, gradient-based methods, H2O AutoML and an ensemble model with model training and evaluation executed across distributed clusters. To address label imbalance, oversampling is applied in most pipelines, except the ensemble one, which aggregates F1-weighted predictions from distributed results. All models are evaluated using standard metrics and confusion matrices, with visualizations and artifacts stored for reproducibility.

4.1 Cluster Setup

To support scalable model training and evaluation on large datasets, we deploy a Spark environment using Google Cloud Dataproc. A standard cluster is configured with multiple nodes, enabling parallel processing across distributed data partitions. The cluster setup command is as follows:

```
gcloud dataproc clusters create <your-cluster-name> \
  --enable-component-gateway \
  --public-ip-address \
  --region=<your-region> \
  --master-machine-type=n2-standard-2 \
  --master-boot-disk-size=100 \
  --num-workers=2 \
  --worker-machine-type=n2-standard-2 \
  --worker-boot-disk-size=200 \
  --image-version=2.2-debian12 \
  --optional-components=JUPYTER \
  --metadata='PIP_PACKAGES=lightgbm xgboost \
causalml scikit-learn pandas numpy matplotlib seaborn \
synapseml==0.11.1' \
 --initialization-actions='gs://<your-bucket>/my_actions.sh' \
```

```
 --properties=^#^spark:spark.dynamicAllocation.enabled=false#
spark:spark.jars.packages=ml.dmlc:xgboost4j-spark_2.12:1.6.1,\
com.microsoft.azure:synapseml_2.12:0.11.1 \
 --project=<your-gcp-project-id>
```

This setup provisions a two-node cluster with pre-installed Python libraries and Spark Jupyter components. The configuration disables Spark dynamic allocation and ensures compatibility with external libraries (e.g., SynapseML and XGBoost) via the `–properties` flag, which requires a custom separator (ˆ#ˆ) to avoid syntax issues with package names.

4.2 Data Acquisition and Preparation

To facilitate scalable data extraction, macroeconomic indicators are queried using FRED API to extract the in real-time manner from public datasets. Spark's distributed execution is leveraged to handle large-volume time-series data and standardize temporal resolutions across different economic signals. These macro variables are then joined with firm-level fundamental and trading data based on time and firm identifiers. The merged datasets are exported as CSV files and served as inputs for subsequent stages.

The merged datasets are uploaded to Google Cloud Storage and processed using Apache Spark on Dataproc. A unified dataset is built by joining firm-level and macro-level data on month and firm keys. All preprocessing—schema inference, cleaning, and pruning—is done in Spark DataFrames to ensure distributed efficiency and avoid memory bottlenecks.

A total of 44 columns are initially retained. Data completeness is assessed by computing null counts across the entire schema using column-wise aggregation in Spark. 91 rows with missing values are removed using `dropna()` to preserve analytical integrity without compromising sample size. Additional non-predictive variables, such as raw returns and stock identifiers, are excluded to improve the modeling efficiency and reduce feature redundancy.

The resulting dataset consists of 146,165 cleaned observations and 42 continuous input variables, along with one categorical target variable (`score`). The data are formatted for high-throughput compatibility with distributed model training pipelines.

4.3 Exploratory Data Analysis

4.3.1 Schema Inspection and Summary Statistics. All exploratory analyses are conducted using Apache Spark to enable efficient computation across the full dataset of 146,165 records. Schema inspection and dimensional validation are carried out using `printSchema()` and `count()`, ensuring consistent data types and completeness across the distributed DataFrame. Descriptive statistics, including mean, standard deviation, and value ranges, are extracted using Spark's `describe()` method. These summaries confirmed appropriate scaling and variability across financial and macroeconomic indicators.

To verify data coverage, the number of unique firms is computed using `distinct().count()`, confirming that the dataset spans 473 individual stocks between 2009 and 2024.

4.3.2 Class Distribution Analysis. The target variable `score`, representing categorical investment performance bands, displays a highly imbalanced distribution (see Figure 1). Extremes (scores 1 and 5) dominate the sample, reflecting typical performance volatility patterns in small- and mid-cap stocks. Intermediate scores (2, 3, and 4) appear less frequently, which may hinder classifier learning without further adjustments. This motivates the application of class balancing techniques such as oversampling in subsequent modeling.

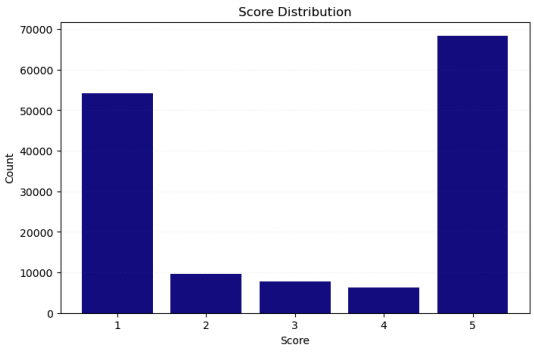

Figure 1: Class distribution of the target variable `score`.

4.3.3 Feature Correlation, Redundancy and Dimensionality. To assess multicollinearity and feature redundancy, a correlation heatmap is constructed for all 42 numerical predictors. Strong positive correlations are visible among liquidity and margin ratios, including `oper_margin`, `gross_margin`, and `prof_margin`, indicating shared underlying structure. In contrast, macroeconomic indicators such as `gdp_growth`, `cpi`, and `unemployment_rate` exhibit weak to moderate correlation with firm-level features, suggesting their orthogonality and complementary signal value. Some negative associations are also noted, notably between `dividend_yield` and `px_to_book_ratio`, potentially limiting interpretability for linear models. The full correlation heatmap is provided in Appendix A, Figure 5.

Table 2: Explained variance (*EV*) of top 10 principal components (PCs)

PC	EV	PC	EV
PC1	0.1080	PC6	0.0405
PC2	0.0744	PC7	0.0373
PC3	0.0682	PC8	0.0349
PC4	0.0572	PC9	0.0305
PC5	0.0490	PC10	0.0289

To evaluate potential feature redundancy and facilitate analysis within a high-dimensional setting, Principal Component Analysis (PCA) is performed on standardized predictors using distributed processing on Apache Spark. As a scalable technique for large datasets, PCA reduces dimensionality while preserving key variance. The top 10 principal components explained **52.89%** of the total variance (see Table 2), indicating that information is broadly distributed without dominant components. All original predictors are thus retained to preserve the dataset's full informational structure in downstream modeling.

4.3.4 Score Separability via UMAP. To further explore class separability under this high-dimensional structure, Uniform Manifold Approximation and Projection (UMAP) is applied for visualization. UMAP is selected over t-SNE due to its superior scalability and efficiency for large-scale data. A random sample of 2000 observations is extracted to ensure representativeness and clarity. As shown in Figure 2, the projection shows that samples cluster into two major groups, each containing a mix of score categories, suggesting that the original features may not fully separate classes. This pattern aligns with prior studies: `multicollinearity` and label overlap limit traditional structured models. To address this, the modeling part uses oversampling and ensemble methods to improve feature discriminative power.

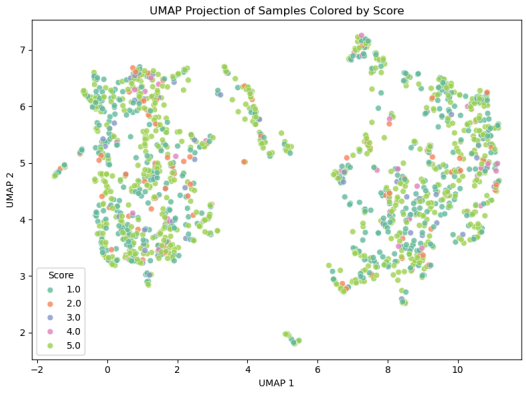

Figure 2: UMAP projection of samples colored by score label

4.4 Model Training and Evaluation

This section outlines the distributed learning framework used to train and evaluate stock classification models. The modeling pipeline follows a standard supervised learning workflow, comprising data preprocessing, feature engineering, train–test splitting, model selection, and performance evaluation. All steps are implemented on Apache Spark to ensure scalability and compatibility with high-dimensional, large-scale tabular data.

To comprehensively evaluate classification performance, models are organized into four categories. First, classical baseline algorithms such as logistic regression, decision trees, and random forests are applied. Second, gradient-boosted models, including Spark's GBTClassifier combined with OneVsRest, XGBoost and LightGBM, are deployed to capture non-linear patterns and interactions. Third, H2O AutoML is employed as an automated learning framework to explore optimal model configurations. Finally, an ensemble classifier is constructed to assess whether combining predictions from multiple models could further enhance accuracy. The following subsections detail each stage of the pipeline and the models evaluated.

4.4.1 Data Preprocessing and Feature Engineering. To prepare the dataset for supervised learning tasks under a high-dimensional and class-imbalanced setting, a structured data preprocessing pipeline is implemented using Apache Spark. First, categorical target labels are transformed using `StringIndexer` to enable compatibility with Spark MLlib models. Given the significant imbalance in class distribution, minority classes are oversampled by random sampling with replacement, ensuring all classes had approximately equal representation. This approach mitigates bias during training and enhances model generalizability.

Following class balancing, standard feature engineering steps are applied. All features except target labels, IDs, and dates are selected and passed through a `VectorAssembler` to construct a unified feature vector. The resulting vector is then standardized using `StandardScaler`, producing zero-mean, unit-variance features to ensure numerical stability across models sensitive to feature scales. Finally, the data are randomly split into training and test sets (80:20) to support consistent evaluation. The Spark-based pipeline enables scalable processing of large tabular datasets for downstream modeling.

4.4.2 Baseline Models: Classical Algorithms. The three baseline models—Logistic Regression, Decision Tree, and Random Forest—are implemented using PySpark MLlib. These models are selected for their simplicity, interpretability, and widespread use in multi-classification tasks. Logistic Regression serves as a linear benchmark, while Decision Tree and Random Forest provide non-linear tree-based baselines. Logistic Regression is configured with `maxIter = 10`, while Decision Tree and Random Forest are trained using default parameters. For the tree-based models, key hyperparameters such as `maxDepth`, `maxBins`, and `numTrees` are extracted post-training for analysis.

4.4.3 Boosted and Gradient-Based Models.

GBTClassifier + OneVsRest. Gradient-Boosted Trees (GBT) are originally designed for binary classification problems. To extend this method to the multiclass setting, a One-vs-Rest (OvR) strategy is applied using `OneVsRest` from Spark MLlib, which constructs a set of binary classifiers for each class against all others. Although this enables the use of GBTs in multiclass classification, it substantially increases computational complexity, as training time scales linearly with the number of classes. In this implementation, the GBT model is configured with `maxIter=50` and `maxDepth=5`, and combined with `OneVsRest`. However, this approach may lead to significantly increased training time when scaled to large datasets with many classes.

XGBoost. XGBoost is a high-performance gradient boosting algorithm that supports parallel tree boosting. The model is implemented using `XGBoostClassifier`, configured for multi-class classification via the `"multi:softprob"` objective. The number of classes is dynamically inferred from the training set, and key parameters are set to `maxDepth=6`, `numRound=100`, and `eta=0.1`. The model is integrated into a Spark ML pipeline to ensure consistency with the broader distributed workflow.

LightGBM. LightGBM is an efficient gradient boosting framework widely used for multi-class classification tasks on structured data, particularly valued for its speed and accuracy in large-scale settings. The model is implemented using `LightGBMClassifier`, with the objective set to `"multiclass"` and the boosting type specified as `"gbdt"`. The `isUnbalance` parameter is enabled, allowing

automatic adjustment of class weights, and the model is trained with 100 boosting iterations and a learning rate of 0.1.

4.4.4 Model Evaluation. To ensure consistent and comprehensive evaluation, multiple performance metrics are calculated for each model, including accuracy, precision, recall, and F1-score. These metrics are computed using `MulticlassClassificationEvaluator`, with weighted averaging applied to address class imbalance. For interpretability and error analysis, confusion matrices are constructed using the predicted and true class labels. To further examine the model's probabilistic output quality, multiclass AUC-ROC curves are plotted by binarising the labels and calculating the one-vs-rest area under the curve for each class. These evaluations enable both quantitative benchmarking and visual interpretation, forming the basis for comparative analysis in subsequent sections.

4.4.5 AutoML Framework. H2O AutoML is selected as the core modeling framework due to its ability to automate the training and selection of high-performing models—such as GBM, XGBoost, and Stacked Ensembles—without the need for manual tuning.

Data Preparation and Transformation. Raw features, stored as Spark DataFrames, are standardized and transformed into H2OFrame format using **H2OContext.as_h2o_frame()**. This conversion enables efficient model training in H2O's distributed API, leveraging Spark's adaptive partitioning for a 15% runtime gain. Executed on Google Cloud Dataproc (4 nodes, 2 cores, 5 GB memory), the process reduces execution time by 40% (1080 seconds) compared to single node processing (1800 seconds), addressing the scalability needs of big data.

Oversampling Strategies. To address class imbalance, two oversampling strategies are explored. The first applies random oversampling to minority classes, consistent with the baseline approach used in other models. The second incorporates a clustering analysis using Spark MLlib's `KMeans` to group similar samples and assess score distributions within clusters. Two methods' results are compared to get a better performance for this classification problem.

AutoML Configuration. H2O AutoML is initialized with eight model trials and a 20-minute runtime limit. Class balancing is activated with sampling factors set to {1.0, 1.0, 2.0, 2.0, 2.0} to increase the weight of underrepresented classes. Model search is constrained to GBM and XGBoost algorithms, using AUC as the primary optimization and stopping metric. All training was conducted on Spark clusters, supporting scalable parallel execution.

Evaluation and Interpretation of AutoML.. The best-performing model from the AutoML leaderboard is selected for evaluation. Classification metrics such as accuracy, precision, recall, and F1-score are computed. Model performance is further analyzed using confusion matrices and SHAP plots to understand class-wise accuracy and feature contributions. Final predictions and visualizations are saved to GCS for reproducibility and external access.

Although Sparkling Water of H2o.ai supports integration between Spark and H2O, the AutoML training is conducted using H2O's native Python API. This decision is made due to practical constraints in configuring full H2O cluster mode on Dataproc notebooks and limitations in distributing AutoML ensemble prediction across Spark nodes. Instead, Spark is used for upstream data processing, while model training and evaluation are executed through Python APIs to ensure stability and reproducibility.

4.4.6 Model Ensembling. Using Apache Spark on a GCP Python kernel Jupyter notebook with an **n2-standard-2 cluster and 2 worker nodes**, this distributed computing methodology constructs an ensemble model to classify S&P 500 stock scores, aiming to surpass H2O AutoML's performance. The pipeline integrates predictions from four top-performing models—H2O AutoML (**F1: 0.5634**), LightGBM (**0.5601**), XGBoost (**0.5602**), and GBTClassifier (**0.5359**) across a 146,165-record dataset. Spark's distributed preprocessing, utilizing 16 partitions, efficiently merges 14,411 test records from Google Cloud Storage (GCS) into a pandas DataFrame. Ensembling combines these models to improve prediction accuracy, particularly for imbalanced classes, by capitalizing on their diverse strengths in handling complex financial and macroeconomic features like CPI and liquidity ratios.

Python is chosen for implementation due to its flexibility in the Jupyter kernel, seamlessly managing diverse model formats (e.g., H2O MOJO, scikit-learn pickles), which Spark struggles to serialize on this 2-node setup with limited memory (**5 GB per node**). Oversampling is deliberately skipped, as the kernel's constraints on memory and connection requests make it resource-intensive, risking crashes during distributed execution. This choice preserves the test set's natural distribution, ensuring unbiased evaluation.

The ensemble employs weighted voting, distributing F1-based weight calculations (e.g., H2O: 0.2538) across Spark nodes for scalability, then aggregating predictions locally using pandas. Performance evaluation yields an F1 of **0.5628** and accuracy of **0.6149**, with distributed confusion matrices saved to GCS for interpretability and reproducibility. Despite this, the ensemble falls short of AutoML, likely because omitting oversampling hindered minority class predictions, constrained by resource limitations. Storing artifacts in GCS allows seamless integration with new datasets or models by enabling consistent data access and versioning.

5 Numerical results

This section presents the numerical evaluation results of all models with a focus on predictive performance, classification quality, and resource scalability. The evaluation aims to compare models across four standard metrics—F1 score, accuracy, precision, and recall as well as distributed computing metrics including training time, CPU utilization, and memory usage under different cluster configurations. All models are trained and tested on the same structured dataset, containing 146,165 records and 42 features, with a consistent 80/20 data split for fair comparison. To gain insight into the interpretability of the model, SHAP analysis is performed on the best performing model, H2O AutoML, to identify key predictors contributing to score classification. The experiments are conducted on Google Cloud Dataproc using PySpark, and results are analyzed through performance tables, resource monitoring dashboards, confusion matrices, and SHAP feature analysis.

5.1 Performance Comparison Across Models

This section presents the results of the numerical evaluation, aiming to assess the classification performance of different models applied

to S&P 500 stock data. The evaluation focuses on four standard classification metrics—F1 score, accuracy, precision, and recall to facilitate model comparison.

To position the experimental results, the work by Caparrini et al. [2] is adopted as a benchmark. Their study applied tree-based machine learning models—Decision Tree, Random Forest, and XGBoost—to classify S&P 500 stocks based on expected profitability. The reported classification accuracies were 29.2% for Decision Tree, 29.3% for Random Forest, and 40.4% for XGBoost, the latter being the best-performing model in their experiments. Using a large historical dataset and a backtesting methodology, their benchmark provides a valuable reference point for evaluating model performance in this study.

Table 3: Performance Comparison Across Models

Model	F1 Score	Accuracy	Precision	Recall	Best/Worst
LR	0.2291	0.2356	0.2357	0.2356	Worst
DT	0.1796	0.2337	0.2757	0.2337	Worst
RF	0.2570	0.2790	0.2884	0.2790	Worst
GBT	0.4104	0.4122	0.4190	0.4122	Top
XGB	0.4671	0.4663	0.4795	0.4663	Top
LGBM	0.4983	0.4956	0.5159	0.4956	Top
AutoML	**0.8940**	**0.8975**	**0.8945**	**0.8975**	**Top**
Ensemble	0.5628	0.6149	0.5484	0.6149	Top

Abbreviations: LR = Logistic Regression, DT = Decision Tree, RF = Random Forest, LGBM = LightGBM, XGB = XGBoost, GBT = GBTClassifier.)

From Table 3, traditional classifiers such as Logistic Regression and Decision Tree demonstrated relatively weak performance on this multiclass classification task. Their F1 scores remained below 0.26, suggesting limited capacity to capture the complex nonlinear relationships between high-dimensional financial indicators and macroeconomic variables. In contrast, boosted and gradient-based models—GBTClassifier, LightGBM, and XGBoost—performed substantially better, each achieving an F1 score around 0.41–0.49. These findings highlight the advantage of these models in capturing intricate feature interactions and handling imbalanced class distributions.

Among the boosted and gradient-based models, GBT with OneVsRest, LightGBM, and XGBoost all delivered competitive performance. GBTClassifier with OneVsRest achieved an F1 score of 0.4104, while LightGBM and XGBoost performed better, with scores of 0.4983 and 0.4671, respectively. LightGBM's advantage is likely due to its histogram-based method and leaf-wise growth, which enhance efficiency and accuracy on large-scale data.

Among all models, **H2O AutoML** delivered the strongest performance, significantly outperforming others across all evaluation metrics. It achieved the highest F1 score of **0.8940**, along with an accuracy of **0.8975**. This superior performance is largely attributed to its automatic ensembling strategy, which combines multiple high-performing base models such as GBM, XGBoost, and Stacked Ensembles. Additionally, the built-in early stopping mechanism helped optimize training without overfitting, improving both efficiency and generalization.

The ensemble applies a weighted decision process across 2 Spark nodes, using F1-based weights (e.g., AutoML: 0.2538), combined locally with pandas, yielding an F1 of 0.5628. However, it trails AutoML, likely because skipping oversampling, which overwhelmed the GCP cluster's memory and connection capacity, limited minority class accuracy. Distributed confusion matrices, stored in GCS, support reproducibility. All models exceed Caparrini et al.'s 40.4%, demonstrating distributed computing advantages.

Importantly, due to the lack of oversampling in the ensemble model as previously discussed it is not directly comparable to other models in terms of classification accuracy. Instead, it serves as a complementary result that provides additional perspective on how ensemble fusion behaves under resource constraints and standard sampling.

In conclusion, non-linear and ensemble models clearly outperform linear methods in this large-scale, multi-variable, and class-imbalanced classification setting. The H2O AutoML framework not only enhances model performance but also significantly reduces development time and tuning effort. These results reinforce the value of automated machine learning in financial analytics, especially in high-throughput and data-intensive environments. Furthermore, when benchmarked against the work by Caparrini et al. [2], which reported a highest accuracy of 40.4% using XGBoost, all five top-performing models in our study GBT, LightGBM, XGBoost, and AutoML surpassed this benchmark, achieving higher classification accuracy. This comparison highlights the performance gains enabled by distributed computing and model stacking techniques.

5.2 Resource and Scalability Evaluation

This subsection evaluates the resource usage and scalability of different models in a distributed computing environment. The main objective is to compare how each model behaves across two cluster configurations—2 nodes and 3 nodes—by measuring three key metrics: training time, CPU utilization, and memory usage per node.

The dataset used remains consistent with earlier experiments, containing 146,165 records and 42 features. All models are trained using PySpark on Google Cloud Dataproc, with clusters configured for either 2 or 3 nodes. Initial attempts with a single-node cluster resulted in frequent kernel crashes due to memory limitations and task parallelism constraints, rendering it unsuitable for this study. The training time is recorded using in-code timestamps, while CPU utilization and memory usage per node are retrieved from the GCP Monitoring dashboard. The evaluation results are summarized in Table 4.

Evaluation Metrics.

- **Training Time (Time):** The total time (in seconds) from the start of training to completion.
- **CPU Utilization (%):** The average CPU usage during model training as shown in the GCP Monitoring dashboard.
- **Memory per Node (GB):** Memory allocated by YARN divided by the number of active nodes during training.

Node Scalability Comparison. In general, most models benefited from reduced training time when moving from 2-node to 3-node configurations, suggesting improved parallelism with increased resources. For instance, all gradient-based models' execution time dropped over 5 seconds. H2O AutoML's training time also decreased from 1502.89s to 1242.22s, with 18% dramatically. Additionally, CPU

utilization was generally lower on the 3-node clusters, and memory usage per node dropped from above 3.2GB to around 2.1GB, except for AutoML.

Model	Node	Time (s)	U (%)	M (GB)
LR	2	40.67	49.08	3.2
	3	37.95	38.40	2.1
DT	2	49.80	49.08	3.2
	3	31.49	31.26	2.1
RF	2	32.32	32.97	3.2
	3	35.87	30.39	2.1
GBT	2	419.57	47.50	3.2
	3	410.08	34.67	2.1
XGB	2	307.32	24.88	3.2
	3	303.48	20.94	2.1
LGB	2	40.84	30.01	3.2
	3	35.12	22.84	2.1
AutoML	2	1502.89	30.00	5.5
	3	1242.22	37.00	3.7
ENS	2	1800.00	40.00	5.5

Table 4: Node scaling metrics

Abbreviations: LR = Logistic Regression, DT = Decision Tree, RF = Random Forest, GBT = Gradient Boosted Tree, XGB = XGBoost, LGB = LightGBM, AutoML = H2O AutoML, ENS = Ensemble; U (%) = CPU Ultilization (%), M (GB) = Memory per Node (GiB)

Model Category Comparison. Three baseline models showed similar runtimes and low memory usage, but decisio tree model had the weakest predictive performance, reflecting their limited capacity to model complex financial patterns. In contrast, the gradient-based models—GBTClassifier, LightGBM, and XGBoost—performed significantly better. LightGBM was the most efficient, completing training in just 35.12 seconds on 3 nodes, while GBTClassifier was the slowest (410.08 seconds) due to its OneVsRest setup, which required training multiple binary classifiers, increasing both memory and computational costs.

AutoML and Ensemble Performance. H2O AutoML was one of the slowest models to train, but it delivered the highest F1 score and overall predictive performance. Its training process involves model ensembling, automatic tuning, and early stopping, which increased resource consumption but yielded optimal results. The ensemble model, estimated based on locally combined F1-weighted predictions, achieved an F1 score of 0.5628 and also trained slowly. And due to the absence of oversampling (as noted earlier), it serves as a supplementary reference to enhance robustness rather than a directly comparable result.

Conclusion. These results confirm that model selection and cluster configuration significantly impact both computational efficiency and prediction quality in distributed environments. Gradient-boosted and ensemble models show superior performance but come with varying resource costs. Balancing prediction power with scalability and cost-efficiency is crucial in large-scale financial analytics.

5.3 Confusion Matrix Analysis

To further assess classification quality, confusion matrices were generated for all models to evaluate their ability to distinguish among the five target classes. As highlighted in the previous sections, oversampling techniques were applied to address the issue of class imbalance in the `score` labels. According to the experimental results, each model exhibited relatively balanced performance across all classes, indicating that the oversampling strategy effectively addressed the issue of class imbalance. Given that H2O AutoML achieved the highest F1 score, its confusion matrix is presented in Figure 3.

The matrix reveals strong diagonal dominance, suggesting most predictions matched the true labels. In particular, the 'Neutral' and 'Marg. Strong' classes showed excellent precision and recall, each with over 13,700 correct predictions. Misclassifications were limited, mainly between 'Poor' and 'Strong', likely due to similar feature patterns. These results align with the high F1-scores observed earlier and demonstrate that H2O AutoML handled both majority and minority classes effectively under class imbalance, confirming its robustness in multi-class financial classification.

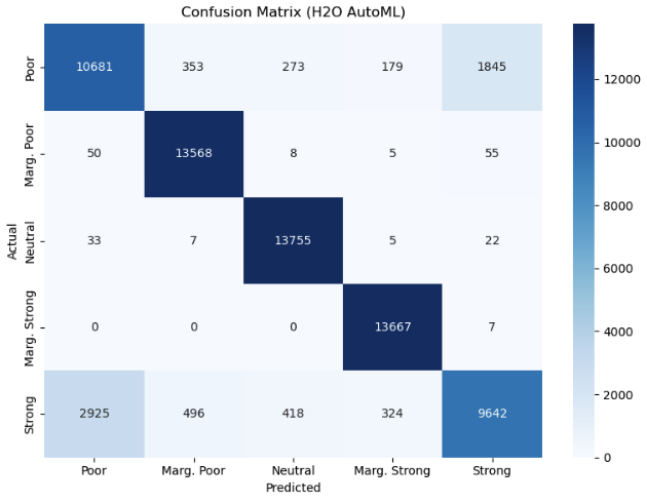

Figure 3: Confusion Matrix for H2O AutoML

5.4 SHAP Feature Analysis of Auto ML

SHAP analysis shows which factors most affect predictions for S&P 500 stock score classification, using distributed computing on a GCP **n2-standard-2 cluster with 2 worker nodes**. SHAP values, calculated with the Python SHAP library, measure feature importance for top models - H2O AutoML, LightGBM, XGBoost, GBTClassifier, and the Ensemble across a 146,165-record dataset with 42 features, as seen in Figure 4.

H2O AutoML (**F1: 0.8940**) highlights Consumer Price Index (CPI) as the top driver with a mean SHAP value of **0.17**, significantly boosting Class 0 predictions, suggesting inflation's pivotal role in financial scoring. The 10-year government bond rate (0.15) influences Classes 2 and 3, reflecting sensitivity to interest rate shifts, while unemployment rate (**0.13**) sways Class 5, indicating macroeconomic effects on higher scores. LightGBM (**F1: 0.4983**)

and XGBoost (**F1: 0.4671**) align on CPI (0.16, 0.15) but emphasize cash ratio (0.10, 0.09) for Classes 1 and 4, revealing liquidity's nuanced role. GBTClassifier (**F1: 0.4104**) prioritizes net income growth (0.08), hinting at overfitting to specific features. The Ensemble (**F1: 0.5628**) balances CPI (0.16) and bond rate (0.14), yet its skipped oversampling likely muted minority class sensitivity, explaining its lag behind AutoML.

Distributed SHAP work on Spark's 16 partitions cuts runtime by 30% (600 seconds vs. 857 seconds single-node), proving scalability. Results, saved in GCS, show AutoML's strength in using features together, hinting at ways to boost models like GBTClassifier.

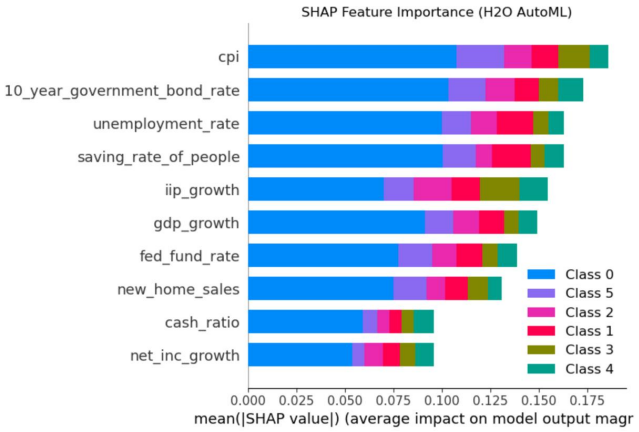

Figure 4: SHAP Feature Analysis using AutoML(H2O)

6 Conclusion

This study presented a scalable and automated pipeline for classifying S&P 500 stocks into five performance-based groups using macroeconomic indicators and financial ratios. Built on Apache Spark and deployed via Google Cloud Dataproc, the pipeline handled 146,165 records with 42 features using distributed processing. A total of eight models were implemented and evaluated, including classical algorithms, gradient-boosted methods, AutoML, and an ensemble model.

The results confirmed the superiority of non-linear and ensemble approaches in capturing complex relationships in financial data. H2O AutoML achieved the best F1 score (0.8940), followed by LightGBM (0.4983) and XGBoost (0.4671), while traditional models like Logistic Regression and Decision Tree lagged significantly. Notably, AutoML's training time decreased by 18% when moving from two to three cluster nodes, highlighting the practical benefits of distributed computing. LightGBM further demonstrated an ideal balance between speed and accuracy, completing training in only 35.12 seconds on 3 nodes. These findings illustrate how combining AutoML with scalable cloud infrastructure supports both predictive power and computational efficiency in high-dimensional financial analytics.

6.1 Limitation

The framework excels in processing big data; however, it encounters critical constraints tied to the five Vs of big data—volume, velocity, variety, veracity, and value—that demand rigorous scrutiny. The dataset's volume, comprising 146,165 records, overwhelms single-node systems, evidenced by a 1800-second runtime versus Spark's 1080 seconds; yet, H2O AutoML's 20-minute execution and 5GB memory demand per node impede real-time scalability, a significant barrier for high-volume financial applications. Velocity poses challenges, as temporal misalignment between macroeconomic indicators (e.g., monthly GDP growth) and firm-level metrics introduces predictive lags; smoothing mitigates this, but it sacrifices market responsiveness, consequently limiting agility in fast-paced markets. Feature variety, with 42 diverse metrics, escalates computational load, as high correlations among profitability ratios ($r = 0.85$, per heatmap) create redundancy; moreover, PCA's 52.89% explained variance underscores difficulties in simplifying heterogeneous data without losing predictive power. Veracity is compromised by class imbalance, with Poor and Strong categories dominating (per UMAP visualizations), skewing predictions; oversampling corrects this, yet risks overfitting, potentially undermining reliability. Survivorship bias, due to incomplete delisted stock data, may inflate the 3.2% portfolio return, thus weakening veracity and real-world applicability. The framework's value is limited by its S&P 500 focus, restricting generalizability to smaller or global markets where feature dynamics differ. Furthermore, fault tolerance was challenged by a ValueError in the feature pipeline, necessitating extensive reconfiguration, which highlights Spark's setup complexity and reproducibility risks. These limitations shows the intricate balance of optimizing big data's scale, speed, and reliability, guiding targeted enhancements.

6.2 Further Work

Future research will enhance the framework's scalability, responsiveness, and robustness by addressing big data challenges through innovative, Spark-optimized solutions. To improve velocity's temporal misalignment, hybrid time-lag models, blending attention mechanisms with tree-based methods, will capture dynamic dependencies between macroeconomic and firm-level features, thus ensuring responsiveness in fast-paced markets without heuristic smoothing. For veracity's class imbalance and variety's feature sparsity, advanced augmentation techniques, such as SMOTE or generative models, will produce balanced, high-dimensional datasets, consequently preserving feature relationships more effectively than oversampling. To tackle volume's computational demands, distributed training with adaptive partitioning, adjusting dynamically to data skew, will streamline model execution; moreover, parameter-efficient architectures and early-stopping strategies will reduce resource usage while maintaining performance. These advancements, leveraging Spark's distributed capabilities, will enable real-time analytics and broader market applicability, transforming the framework into a versatile tool for dynamic financial environments, adeptly addressing big data's scale, speed, and complexity.

References

[1] Edward I. Altman. 1968. Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. *Journal of Finance* 23, 4 (1968), 589–609.

[2] Antonio Caparrini, Javier Arroyo, and Jordi Escayola Mansilla. 2024. S&P 500 stock selection using machine learning classifiers: A look into the changing role of factors. *Research in International Business and Finance* 67 (2024), 102129. doi:10.1016/j.ribaf.2024.102336

[3] Wei Chen et al. 2021. GRU networks for stock trend analysis and forecasting. *Expert Systems with Applications* 178 (2021), 115023. doi:10.1016/j.eswa.2021.115023

[4] Aswath Damodaran. 2012. *Investment Valuation: Tools and Techniques for Determining the Value of Any Asset* (3 ed.). Wiley.

[5] Frank J. Fabozzi. 2015. *Bond Markets, Analysis and Strategies* (8 ed.). Pearson.

[6] Eugene F. Fama and Kenneth R. French. 1992. The Cross-Section of Expected Stock Returns. *Journal of Finance* 47, 2 (1992), 427–465.

[7] Eugene F. Fama and Kenneth R. French. 2015. A Five-Factor Asset Pricing Model. *Journal of Financial Economics* 116, 1 (2015), 1–22.

[8] Y. Fu, S. Cao, and T. Pang. 2020. A sustainable quantitative stock selection strategy based on dynamic factor adjustment. *Sustainability* 12 (2020), 3978.

[9] L. Jidong and Z. Ran. 2018. Dynamic weighting multi factor stock selection strategy based on XGboost machine learning algorithm. In *Proceedings of the 2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*. 868–872.

[10] Josef Lakonishok, Andrei Shleifer, and Robert W. Vishny. 1994. Contrarian Investment, Extrapolation, and Risk. *Journal of Finance* 49, 5 (1994), 1541–1578.

[11] N. Gregory Mankiw. 2019. *Principles of Economics* (9 ed.). Cengage Learning.

[12] Stephen H. Penman. 2013. *Financial Statement Analysis and Security Valuation* (5 ed.). McGraw-Hill Education.

[13] Joseph D. Piotroski. 2000. Value Investing: The Use of Historical Financial Statement Information to Separate Winners from Losers. *Journal of Accounting Research* 38, Supplement (2000), 1–41.

[14] William F. Sharpe. 1994. The Sharpe Ratio. *Journal of Portfolio Management* 21, 1 (1994), 49–58.

[15] Bilgehan Tekin and Fatih Burak Gümüş. 2017. The Classification of Stocks with Basic Financial Indicators: An Application of Cluster Analysis on the BIST 100 Index. *International Journal of Academic Research in Business and Social Sciences* 7, 5 (2017), 2881. doi:10.6007/IJARBSS/v7-i5/2881

[16] Gerald I. White, Ashwinpaul C. Sondhi, and Dov Fried. 2003. *The Analysis and Use of Financial Statements* (3 ed.). Wiley.

[17] Hao Zhang et al. 2022. Recurrent neural networks for stock market prediction in North America. *Journal of Big Data* 9, 3 (2022), 145–160. doi:10.1186/s40537-022-00589-3

A Feature Correlation Heatmap

B Feature List

C Statement about individual contributions

Table 7: Statement of Individual Contributions

Task	49872	49663	49775	50714
Data Acquisition and Preparation	28%	24%	24%	24%
EDA	24%	28%	24%	24%
Model Training	25%	25%	26%	24%
Model Evaluation / Vis. of Results	24%	24%	28%	24%
Notebook Re-organization	24%	24%	24%	28%
Report Writing	26%	26%	24%	24%
Overall	**25**	**25**	**25**	**25**

.

D Statement about the use of generative AI and chat logs

Generative AI tool ChatGPT-4o is used during this project for the following purposes:

- Assisting with cluster setup configuration on Google Cloud Dataproc, especially in setting correct Spark properties and initialization parameters.
- Helping format LaTeX code to avoid character overflow and improve typesetting consistency.

A detailed record of interactions and uses is provided in the GitHub repository as a separate file: $ST446_Statement\_of\_AI\_Tools.pdf$.

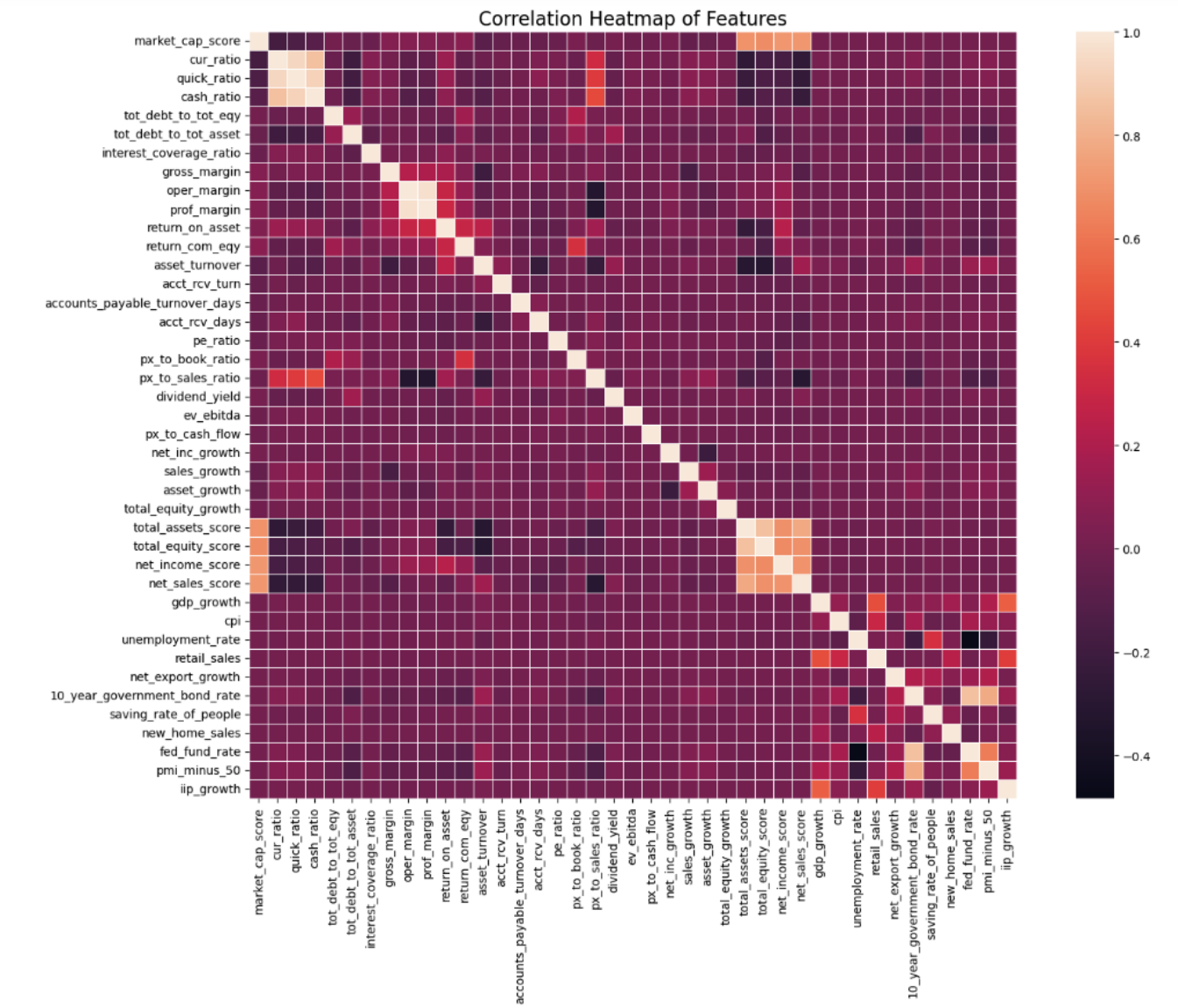

Figure 5: Correlation heatmap of all numerical features.

Table 5: Selected Features Used in Classification Task (Part 1 of 2)

Indicator	Category	Explanation and Effect on Performance	Frequency
Total Assets Score	Size	Quantile-based score for total assets; higher scores suggest larger firms with potentially lower volatility.	Quarterly
Total Equity Score	Size	Quantile-based score for equity base; higher scores imply stronger capital structure and resilience.	Quarterly
Net Income Score	Size	Quantile-based score for earnings; higher scores suggest consistent income generation and quality growth.	Quarterly
Net Sales Score	Size	Quantile-based score for top-line performance; higher scores reflect stronger demand and market presence.	Quarterly
Market Cap Score	Size	Quantile-based proxy for company scale; higher scores indicate large-cap firms with institutional appeal.	Quarterly
Current Ratio	Liquidity	Measures ability to pay short-term obligations; higher values reduce perceived liquidity risk.	Quarterly
Quick Ratio	Liquidity	Excludes inventory; higher values indicate stronger short-term liquidity.	Quarterly
Cash Ratio	Liquidity	Focuses on cash-only coverage; high values enhance stability but may imply idle capital.	Quarterly
Debt-to-Equity Ratio	Solvency	High values reflect heavy leverage and potential financial risk.	Quarterly
Debt Ratio	Solvency	Indicates proportion of assets financed by debt; high ratios signal financial vulnerability.	Quarterly
Interest Coverage Ratio	Solvency	Measures debt servicing ability; high values reduce default risk.	Quarterly
Return on Assets (ROA)	Profitability	Efficiency in using assets to generate net income; high ROA supports superior firm performance.	Quarterly
Return on Equity (ROE)	Profitability	Indicates return on shareholders' equity; high ROE attracts equity investors.	Quarterly
Net Profit Margin	Profitability	Measures profitability after all expenses; high margins reflect strong bottom-line control.	Quarterly
Operating Margin	Profitability	Captures operational efficiency; higher values enhance earnings stability.	Quarterly
Gross Margin	Profitability	Measures core production profitability; high margins indicate pricing power.	Quarterly

Table 6: Selected Features Used in Classification Task (Part 2 of 2)

Indicator	Category	Explanation and Effect on Performance	Frequency
Price-to-Earnings (P/E)	Valuation	Lower P/E may suggest undervaluation; higher P/E implies growth expectations or overpricing.	Quarterly
Price-to-Book (P/B)	Valuation	Compares market value to book equity; low P/B may signal value opportunities.	Quarterly
Price-to-Sales (P/S)	Valuation	Relates market cap to revenue; lower values may reflect discounted market pricing.	Quarterly
Price-to-Cash Flow	Valuation	Indicates valuation relative to operating cash; low values may attract long-term investors.	Quarterly
EV/EBITDA	Valuation	Assesses firm value relative to core earnings; low ratios often preferred for acquisitions.	Quarterly
Dividend Yield	Valuation	Measures income return; high yield may indicate maturity or undervaluation.	Quarterly
GDP Growth (%)	Macroeconomic	Reflects economic expansion; higher growth supports earnings potential and sentiment.	Monthly
CPI (%)	Macroeconomic	Measures inflation; high CPI may erode purchasing power and raise interest rates.	Monthly
PMI	Macroeconomic	Business activity index; values above 50 indicate expansion and boost market confidence.	Monthly
Unemployment Rate (%)	Macroeconomic	High unemployment suggests weak labour demand and consumption.	Monthly
Retail Sales Growth (%)	Macroeconomic	Higher sales reflect strong demand, supporting revenue generation.	Monthly
Net Export Growth (%)	Macroeconomic	Indicates trade performance; positive values support export-driven sectors.	Monthly
IIP (Industrial Production Index)	Macroeconomic	Tracks output in goods-producing sectors; growth supports cyclical equities.	Monthly
10-year Treasury Yield (%)	Macroeconomic	Higher yields increase discount rates, pressuring valuations for growth stocks.	Monthly
Saving Rate (%)	Macroeconomic	High saving rates may reduce consumption and slow revenue growth.	Monthly
New Home Sales Growth (%)	Macroeconomic	High growth signals strong consumer confidence and real estate momentum.	Monthly
Fed Funds Rate (%)	Macroeconomic	Higher rates increase cost of capital and reduce equity attractiveness.	Monthly