Financial News and Stock Price Trend Analysis using Latent Semantic Analysis, Logistic Regression, Naïve Bayes, Random Forrest, and Support Vector Machine

Jinsheng Li
Khoury College of Computer Science
Northeastern University
Boston, MA
li.jinsh@northeastern.edu

June 30, 2024

1 Objectives and significance

Finance plays a crucial role in fostering global innovation and serves as the backbone of our economy. The goal of financial management is to effectively allocate limited resources to achieve the best outcomes given the inherent risks. The overall performance of financial management and various other market factors should be reflected in stock price movements. One of the emerging areas in finance is the analysis of financial news to predict stock price trends, combining traditional financial theories with advanced machine learning techniques.

In this project, driven by my intrigue and background in finance, I will delve into the financial market by leveraging Latent Semantic Analysis (LSA) and machine learning models such as Logistic Regression, Naïve Bayes, Random Forest, and Support Vector Machine (SVM) to analyze financial news and predict stock price movements. The financial news dataset will be sourced from Kaggle's "Daily Financial News for 6000+ Stocks," which includes about 4 million articles for 6000+ stocks from between 2009 and 2020. I plan to conduct my research on approximately 5 selected stocks and retrieve their financial data from Yahoo Finance using Python. The objective is to extract meaningful insights from news articles, correlate them with subsequent stock price trends as input labels, and then feed them into different machine learning algorithms to compare the outputs, thereby enhancing the decision-making process for investors.

By employing LSA, I aim to uncover the latent semantic structures in the text data, which are then used as features for our machine learning models. The combination of these sophisticated models will allow me to build a robust framework for predicting stock price trends based on the sentiment and content of financial news. Through rigorous preprocessing, feature extraction, and model training, I strive to achieve high accuracy and reliability in our predictions by comparing different outputs from different models.

Ultimately, this research highlights the importance of integrating textual data analysis with quantitative stock performance metrics. By comparing the performance of different machine learning models, I seek to identify the most effective approach for predicting stock price trends. This study not only contributes to the field of financial analysis but also demonstrates the potential of advanced data-driven methodologies in improving investment strategies.

2 Dataset

The dataset used for this project is extracted from Kaggle's "Daily Financial News for 6000+ Stocks." Specifically, I utilized the "raw_analyst_ratings.xlsx" file, which contains directly-scraped raw analyst ratings. The columns include index, headline, URL, article author (with the publisher always being Benzinga), publication timestamp, and stock ticker symbol. Due to the large scale of the dataset, I selected only five stocks: 'AAPL', 'JNJ', 'PG', 'AMZN', and 'PXD,' focusing on news dates from January 1, 2018, to the most recent date. Using the Yahoo Finance Python library, I extracted the stock prices for the date of the news and the previous day to determine if there was a price movement on the news date, categorizing it as either up or down. During the stock price extraction process, I encountered missing data, and I excluded those rows. I used the valid rows' closing prices to calculate the RSI and SMA values, which served as input features for the algorithms. Additionally, I employed web scraping techniques to extract the news content from each website and combined it with the news titles to create a new column used as the "Text" field, along with the RSI and SMA values, for algorithm input.

3 Algorithms

Latent Semantic Analysis (LSA) is a technique for uncovering the underlying semantic structure of textual data. It starts by constructing a Term-Document Matrix (TDM) with rows representing terms and columns representing documents, where each cell indicates term frequency. Term Frequency-Inverse Document Frequency (TF-IDF) weighting is applied to highlight important terms. LSA employs Singular Value Decomposition (SVD) to decompose the TF-IDF matrix into three matrices: U, Σ, and V^T. By selecting the top k singular values, dimensionality reduction is achieved, capturing significant semantic relationships and making the high-dimensional text data more manageable.

Logistic Regression is a widely used algorithm for binary classification, predicting whether an input belongs to a specific class, such as an upward or downward stock trend. It employs the logistic (sigmoid) function to model probabilities between 0 and 1, transforming a linear combination of input features into a probability. A threshold, typically 0.5, is applied to classify the input into one of the two classes. Logistic Regression is appreciated for its interpretability, with model coefficients indicating each feature's impact on the positive class probability. It is computationally efficient and effective for linearly separable data. Regularization techniques like L1 (Lasso) and L2 (Ridge) are used to prevent overfitting by adding penalties to the loss function based on the coefficients' values.

Naïve Bayes is a probabilistic algorithm suitable for text classification tasks. It leverages Bayes' theorem to calculate the posterior probability of a class given input features, assuming feature independence given the class. This assumption simplifies computations. The algorithm estimates the posterior probability by multiplying the conditional probabilities of each feature given the class and the prior probability of the class. Naïve Bayes is easy to implement and computationally efficient, performing well with large feature sets. Smoothing techniques like Laplace smoothing handle zero probabilities, ensuring robustness even with sparse data.

Random Forest is an ensemble learning method that enhances classification and regression tasks by combining multiple decision trees. It uses bagging, where multiple decision trees are trained on different subsets of the training data generated through bootstrapping. Each tree is also trained on a random subset of features, adding randomness and reducing correlation among trees. For classification tasks, the final prediction is determined by a majority vote among the trees. Random Forest is robust, reducing overfitting and improving generalization by averaging the predictions of multiple trees. It also provides feature importance estimates, aiding in understanding the contributions of different features. Performance optimization can be achieved through hyperparameter tuning, such as adjusting the number of trees, maximum depth, and minimum samples per leaf.

Support Vector Machine (SVM) is a classification algorithm that aims to find the optimal hyperplane separating different classes with the maximum margin. It maximizes the margin between the hyperplane and the nearest data points from each class, known as support vectors. SVM is effective in high-dimensional spaces and for data with a clear margin of separation. For non-linearly separable data, SVM uses the kernel trick to transform input features into a higher-dimensional space, enabling complex decision boundaries. Common kernel functions include linear, polynomial, and radial basis function (RBF). Regularization techniques for SVM include kernel parameter tuning, early stopping, and the soft margin (C parameter). Kernel parameter tuning adjusts the parameters of the kernel functions to control decision boundary complexity. Early stopping monitors validation performance and halts training to prevent overfitting. The C parameter balances achieving low training error and minimizing the norm of the weights, with a smaller C value encouraging a larger margin and reducing overfitting.

Referencing the paper "Predicting the Stock Trend Using News Sentiment Analysis and Technical Indicators in Spark" by T. Kabbani and F. Usta, incorporating certain financial model metrics can be crucial for capturing momentum trends in stock prices, which is beneficial for predicting future price movements.

The Relative Strength Index (RSI) is a key momentum indicator used by traders to determine whether a stock is overbought or oversold over a specified period. In this project, I utilized a 14-day period to calculate the RSI, which ranges between 0 and 100. The RSI is calculated using the formula:

$$RSI = 100 - [\frac{100}{1 + \frac{Average\ of\ 14\ days'\ close\ Up}{Average\ of\ 14\ days'\ close\ Down}}]$$

As a feature, RSI is beneficial in capturing momentum trends in stock prices, crucial for predicting future price movements. By identifying overbought and oversold conditions, RSI enhances the decision-making process in our predictive models.

The Simple Moving Average (SMA) is a widely used indicator that calculates the average of a selected range of prices, typically closing prices, over a specified number of periods. In this project, I am planning to use a 14-day period for calculating the SMA, computed using the following equation:

$$SMA = \frac{P_1 + P_2 + P_3 + .. + P_n}{n}$$

where n is the number of trading days, in this case, 14. SMA smooths out price data and identifies trends by averaging out fluctuations. This feature is valuable for our models as it provides a clear indication of the stock's price trend over time, aiding in the prediction of future movements.

4 Related Work

In the Kabbani and Usta research paper, the authors tackle the challenge of stock market trend prediction by framing it as a machine learning classification problem. They create a "tomorrow_trend" feature to be predicted, using technical indicators derived from stock price history and sentiment scores from financial news as input features. Three machine learning models—Logistic Regression, Random Forest, and Gradient Boosting Machine—are tested in Spark, with Random Forest achieving the best performance with a 63.58% test accuracy.

In the study "Stock Trend Prediction Using News Sentiment Analysis" by J. Kalyani, H. Bharathi, and R. Jyothi, the authors investigate the impact of financial news articles on stock market trends. Recognizing the limitations of the Efficient Market Hypothesis, they explore the relationship between news sentiment and stock trends. The project involves predicting future stock trends by classifying news articles as positive or negative using three different classification models: Random Forest (RF), Support Vector Machine (SVM), and Naïve Bayes. Their experiments demonstrate that RF and SVM outperform Naïve Bayes, achieving an accuracy of over 80%, a significant improvement over random labeling.

These studies are similar to my idea of combining financial news analysis with technical indicators to predict stock trends. The first paper aligns more closely with my initial ideas, as it incorporates both sentiment analysis and technical indicators, providing a comprehensive approach to stock trend prediction. Therefore, I plan to use some of the techniques and methods, including the financial metrics discussed, in my own study. The second paper also offers valuable perspectives and methodologies that I can incorporate to enhance the robustness and accuracy of my predictive models.

5 Training Assessment and Evaluation Methodology

To train the models, I first utilized LSA in conjunction with TF-IDF to vectorize the text data and reduce its dimensionality to 1500 components. This reduction process helps capture essential semantic relationships while mitigating the curse of dimensionality. To ensure that there are no negative values in the data, I applied a transformation that set the minimum value to zero. The resulting vectorized text data was then merged with the RSI and SMA values, forming the complete feature set (X input). The dataset was split into 80% for training and 20% for testing. Subsequently, the four algorithms—Logistic Regression, Naïve Bayes, Random Forest, and SVM—were trained and evaluated on this dataset. This approach allowed for a comprehensive comparison of model performances, aiding in the selection of the most effective predictive model.

To assess the effectiveness and reliability of the predictive models, I employed a comprehensive set of performance evaluation metrics. These metrics provided a detailed understanding of the models' strengths and weaknesses. Accuracy was used to indicate the proportion of correctly predicted stock price movements out of the total predictions. Precision measured the ratio of true positive predictions to the total positive predictions made by the model, reflecting its ability to avoid false positives. Recall, also known as sensitivity, assessed the ratio of true positive predictions to the total actual positives, indicating the model's ability to identify all relevant instances of the positive class. The F1-Score, the harmonic mean of precision and recall, offered a balanced metric, particularly useful for evaluating models on datasets with imbalanced classes.

Additionally, the ROC-AUC metric was employed to assess the model's capability to distinguish between positive and negative classes. A higher AUC value signified better performance in differentiating between upward and downward stock trends, providing a comprehensive measure of the classifier's effectiveness. This metric was valuable for evaluating the model's ability to make accurate classifications across various decision thresholds, offering a nuanced understanding of its performance.

The 'scikit-learn' library in Python was utilized to calculate these metrics, as it provides built-in functions for evaluating model performance, simplifying the implementation and analysis of these evaluation measures. This approach ensured a thorough and standardized evaluation of the predictive models.

To obtain robust and unbiased estimates of the models' performance, I employed K-Fold Cross Validation. In this strategy, the dataset was divided into k subsets (folds), typically set to 5 or 10. Each fold served as a validation set while the remaining k−1k - 1k−1 folds were used for training. The model was trained k times, each time with a different fold as the validation set, ensuring that every data point was used for both training and validation. This method provided a comprehensive evaluation of the model, calculating performance metrics such as accuracy, precision, recall, F1-Score, and ROC-AUC for each iteration, and averaging them across all k iterations. This averaging process yielded a more reliable estimate of the model's performance, mitigating the impact of data variability and preventing bias from any specific data subset.

To further understand the impact of various factors on model performance, I conducted ablation studies by systematically modifying different aspects of the data and models. This included adjusting the number of features extracted from LSA, experimenting with different text preprocessing techniques (e.g., stop words removal, lemmatization, stemming), and varying the complexity of the machine learning models (e.g., the number of trees in Random Forest, regularization strength in SVM). By systematically altering these factors and recording the performance outcomes, I was able to rank each change based on its impact on the final performance. This process helped identify the most influential elements in the predictive framework and determine the optimal configuration for each algorithm.

References

1. J. Kalyani, H. Bharathi, and R. Jyothi. “Stock Trend Prediction Using News Sentiment Analysis.” https://doi.org/10.48550/arXiv.1607.01958

2. T. Kabbani and F. Usta. “Predicting the Stock Trend Using News Sentiment Analysis and Technical Indicators in Spark.” https://doi.org/10.48550/arXiv.2201.12283

3. M. Aenlle. “Daily Financial News for 6000+ Stocks.” Kaggle. https://www.kaggle.com/datasets/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests

4. Yahoo Finance. https://finance.yahoo.com.