Analyst Rating Stock Prediction Analysis
Group Assignment Report
(Machine Learning)

Group:
Colin Halliday (colinhalliday@cmail.carleton.ca)
Dherya Jain (jaindherya16@gmail.com)
Marko Siou (sioumarko@gmail.com)
Zarrin Tasneem (z2tasnee@uwaterloo.ca)

Summer 2025

Executive Summary

This report explains a machine learning project designed to determine if analyst price target changes have a short-term, next-day, predictable impact on stock prices. The hypothesis is that ratings from influential institutions, such as banks, combined with the magnitude of the price target change and an analyst's past performance, could forecast a stock's next-day movement.

A model was developed using two datasets, one of news headlines and the other of historical stock prices from 2009 to 2020. A Random Forest Classifier machine learning model achieved slightly under 52% accuracy in correctly predicting whether a stock would move up or down the day after a rating change. The model accurately predicted positive movements more than 52% of the time, but accurately predicted negative ones 51% of the time. Key features that influenced the model were the percentage difference of the price target compared to the current stock price and the analyst's historical accuracy.

The findings suggest that a predictive model for this problem is difficult to build with the current data. However, it was able to find a slight correlation between analyst ratings and next-day stock movement. Further improvements could be performed with the features used in the dataset, including ones related to sentiment analysis, which may improve the accuracy of the model.

1. Objective

The primary objective of this machine learning project was to determine if analyst price target forecasts, as announced in news headlines, have a predictive next-day impact on stock prices.

There is an opportunity for short-term investors to make trading decisions based on the machine learning model. It may be difficult for an individual to monitor analyst ratings of thousands of stocks and be able to decide which ones are likely to be more predictive of the next day's stock movements. This project sought to determine if a machine learning model could find patterns in this data to provide investors with a competitive advantage.

1.1 Hypothesis

The project's central hypothesis was twofold:

An analyst's price target, particularly from a well-known financial institution, can influence a stock's next-day price movement. Since these institutions have access to top talent and financial data, these announcements will influence investors en masse to buy or sell stocks.

The magnitude of the price target change and the analyst's historical accuracy are key predictive features of this movement. The theory here is that a more significant change in a price target, such as a large predicted percentage increase, should correspond to a more significant market reaction. Similarly, an analyst with a proven track record of successful predictions should have a greater impact on a stock's price than a less accurate one.

An understanding of this relationship could provide an advantage to short-term investors seeking to capitalize on this information. A model capable of a slight improvement over the market average could be the basis for an extremely profitable trading strategy, as even a 1-2% edge can yield significant returns if leveraged over many trades.

2. Data Preparation

Two main data sources were utilized for this project: a news headlines dataset (analyst_ratings_processed.csv) from 2009 to 2020 accessed through Kaggle and the yfinance Python library for fetching historical stock prices. The data preparation pipeline was a multi-stage process designed to clean, enrich, and format the raw data for machine learning.

2.1 Data Sourcing and Challenges

The news headlines dataset provided headlines, dates, and associated stock tickers. A key challenge was the poor data quality and inconsistency of the tickers; many were invalid or referred to stocks which were not accessible via the yfinance library. This required a validation process, where a script was attempted to fetch a small window of price data for each unique ticker. This step was essential but resulted in a significant reduction of the dataset. The initial dataset contained headlines for approximately 23,000 stocks. After validation, this was reduced to roughly 16,000, which were deemed to be valid and active in the market during the period. The remaining data was still substantial enough to build a robust model.

Another challenge involved isolating and classifying the issuing institutions. Most headlines began with the institution's name, but not all of them. A feature was engineered by extracting the first word of the headline, and a minimum count filter was applied to retain only recognizable financial institutions, such as major banks. This was an iterative process that required manual review of the filtered list to make sure it contained only genuine institutions and not general market terms. This process was important for creating meaningful institution features that could be used for our analyst_accuracy_score. Without this careful cleaning, the feature would have been noisy and unreliable.

2.2 Progression of the Project

Our initial approach to modeling was based on simple features extracted from the raw data. We experimented with a basic flag for price target direction (pt_dir) and a text-based model using a TF-IDF vectorizer on the news headline titles. However, early results were poor, with accuracy barely above random chance. The models were not capturing the nuances of market behavior, resulting in the refinement of our features. We realized that a model needed more context than just the raw headlines. To address this, we developed more sophisticated features to capture the nuances of analyst behavior and market reaction.

Price Target Difference (pt_diff_perc): This quantified the magnitude of the analyst's forecast by measuring the percentage difference between the new price target and the stock's current closing price. An increase of 5% price target is not significant compared to a 50% increase, and this feature enables the model to learn that distinction.

Analyst Accuracy Score: This generated a score of how accurately the specific institution predicted the next-day movements of the stock.

These features slightly improved the model from an accuracy of just above 50% to an accuracy of 52%.

3. Model Design

Two machine learning models were developed and evaluated to predict a stock's next-day movement. The dataset was split chronologically, with the first 80% used for training and the remaining 20% for testing. This prevented potential data leakage from occurring. The final models were trained only on "Up" and "Down" labels, as "No Change" events (events where the price changed between -0.1% and 0.1%) hindered model accuracy. We chose two models, specifically Random Forest and Gradient Boosting.

Model 1: Tuned Random Forest Classifier

This model was designed to use the structured, engineered features. A Random Forest is an ensemble learning method that builds multiple decision trees during training and outputs the mode of the classes for classification. Its key advantage is its ability to handle a mix of numerical and categorical data, as well as its robustness against overfitting.

Features: The model's input features were the pt_diff_perc and its binned categorical version (pt_diff_bin), analyst_accuracy_score, a flag for a price target change, the number of days since the last rating, and a binary flag for whether the stock was in the S&P 500. The inclusion of the binned price target feature alongside its numerical counterpart was an intentional design choice to allow the model to learn both the continuous magnitude and the categorical significance of different price target ranges (e.g., a "Small Positive" increase might be interpreted differently than a "High Positive" one).

Preprocessing: A scikit-learn pipeline with a ColumnTransformer was used to manage preprocessing. A SimpleImputer filled in missing numerical values with the median, and a OneHotEncoder handled categorical features. This pipeline was designed to be modular and ensure that all preprocessing steps are applied consistently during training and prediction, which is important for reproducibility and preventing data leakage.

Architecture: Hyperparameter tuning was performed using GridSearchCV to optimize the model's performance on the training data. This process involved systematically testing a range of hyperparameters, such as the number of estimators, max depth, and minimum samples per leaf, to find the combination that yielded the best macro F1-score. This step was essential for preventing the model from becoming overfitted to the training data and ensuring its performance generalized to unseen data.

Model 2: Gradient Boosting Classifier

This model was designed as an alternative approach, incorporating the raw text of the headlines. Gradient Boosting is another ensemble method that builds models sequentially, with each new tree attempting to correct the errors of the previous ones. It is often more powerful than Random Forest but can be more prone to overfitting.

Features: The model's primary feature was the headline title text, which was processed with a TfidfVectorizer. This vectorizer converts text into a numerical matrix, where each cell represents the importance of a word to a document. The model also included a few key engineered features like the pt_diff_perc and the analyst_accuracy_score.

Preprocessing: A Pipeline with a ColumnTransformer was used to apply the text vectorizer and other preprocessing steps.

Architecture: A GradientBoostingClassifier was selected as a strong alternative to the Random Forest, providing a valuable point of comparison. The model was configured with hyperparameters that balanced predictive power with computational efficiency.

4. Model Evaluation

The two models were evaluated on the test dataset. The primary goal was to measure their performance on the "Up" and "Down" labels. Accuracy was chosen as a metric, along with macro F1-score, as it gives equal weight to both classes.

Model 1 (Tuned Random Forest) Performance

The Tuned Random Forest model, which was dependent on engineered features, delivered the best performance. Results from the random forest model are shown in Table 1.

Table 1: Results from random forest model

Class	Precision	Recall	F1-Score
Down	0.5173	0.3900	0.4447
Up	0.5212	0.6460	0.5769
Accuracy	0.5197		

Macro Average

0.5192

0.5180

0.5108

The model achieved a final accuracy of 52%, representing a slight improvement over the initial models. The Macro-F1 score of 51% shows the model's performance is balanced, but not highly effective, across both classes. A major weakness is its poor recall on the "Down" class (0.3900), meaning it failed to predict over 60% of actual downward movements.

Model 2 (Gradient Boosting) Performance

The Gradient Boosting model showed a similar pattern but slightly lower overall performance. It was found that the engineered features were more predictive than the text-based features, reaffirming the importance of contextual and analyst-specific data. The text-based model's weakness likely stemmed from the brevity of the news headlines, which often lack the nuanced language that TfidfVectorizer is designed to capture.

Feature Importance

The feature importance analysis for the best-performing model revealed which signals were most valuable for prediction. The two most important features, pt_diff_perc and analyst_accuracy_score, accounted for over two-thirds of the model's predictive power. This confirms the hypothesis that the magnitude of an analyst's forecast and their historical credibility are the most significant signals. It is also interesting to note the importance of days_since_last_rating, which suggests that a rating after a long period of silence carries more weight.

5. Conclusions

5.1 Hypothesis Validation

The project successfully demonstrated that analyst price target forecasts do contain small but measurable signals for predicting short-term stock movements. The model's 52% accuracy, while not robust enough for high-stakes trading, is an improvement over random chance. The initial hypothesis was partially validated, as key features like the price target difference and analyst accuracy were found to be predictive.

5.2 What We Learned

Feature engineering is key: The custom-engineered features, especially analyst_accuracy_score and pt_diff_perc, proved to be far more valuable than standard features or a text-based model. The success of these features highlights the importance of domain knowledge in machine learning-understanding the underlying business problem and creating features that directly represent the drivers of that problem.

Data quality matters: A significant portion of the project was dedicated to cleaning the dataset. The initial data was rife with invalid tickers, requiring a robust filtering and validation process that reduced the size of the dataset. This reinforced the principle that a clean, well-prepared dataset is the foundation of any successful machine learning project.

5.3 Recommendations for Future Improvements

To enhance the model's accuracy and utility, the following steps are recommended:

Experiment with Different Targets: Instead of predicting a simple "Up" or "Down" label, the model could predict the magnitude of the movement (e.g., a multi-class problem with labels like "Small Up," "Big Up," "Small Down"). This would provide a more nuanced and potentially more valuable prediction for a trader.

Expand Data Sources: Explore new data sources beyond news headlines, such as social media sentiment, company financial statements, or even analyst conference call transcripts.

Test on recent market data to validate if the relationship still holds today.