Name: Vishal Sehgal
Student Number: 2109374

Classifying Nvidia Stock Price Movements Based on News Articles

Introduction

The goal of this task is to predict Nvidia’s stock price movement (up or down) based on related news articles. Using an NLP pipeline, I transformed the unstructured news into features for classifying stock price behavior. The aim is to determine if Nvidia's stock price will rise or fall after news events by analyzing the sentiment and content of the articles.

Dataset Creation

The task was to predict Nvidia’s stock price movement using relevant news articles. I began with a dataset of articles tagged with metadata like company ticker symbol, title, content, and release date. I filtered the dataset for Nvidia-specific articles by selecting those with the NVDA ticker symbol and by searching for "Nvidia" and "NVDA" in the title and content. This resulted in 3,464 articles, later reduced to 3,450 after removing duplicates based on content and title. This filtering removed unrelated articles to avoid noise and ensured no article was overrepresented.

Next, I integrated Nvidia’s historical stock data, including Date, Open, Close, High, Low, and Volume. I merged this data with the filtered articles based on the release date, aligning each article with stock performance on that day. This allowed for accurate ground truth and generation of target labels (stock price increase or decrease) for model training.

Preprocessing

To prepare the text data for modeling, I applied several key preprocessing steps:

1. Lowercasing: I converted all text to lowercase to ensure consistency, treating "Nvidia" and "nvidia" the same. This reduces vocabulary complexity and avoids duplicates due to capitalization.
2. Punctuation Removal (except $ and %): I removed punctuation except for $ and %, as these are important in financial contexts (e.g. $10 million or 5% growth). This reduces noise while retaining relevant financial symbols.
3. Tokenization and Stopword Removal: I tokenized the text and removed stopwords like "the" and "is" in both English and Spanish. This improves feature relevance by focusing on meaningful words.
4. Lemmatization: I reduced words to their base form (e.g., "running" to "run"), simplifying vocabulary while keeping the meaning intact, so that the model can generalize better.
5. Removing Unwanted Numbers: I removed irrelevant numbers but kept those related to stock performance, such as prices and percentages, to ensure only meaningful data is included.

I avoided stemming, as it can be too aggressive and distort word meanings. Lemmatization was a better choice to simplify the text without losing important context.

Analysis

1. **Descriptive Statistics:** The dataset contains 3,103 articles about Nvidia, providing sufficient content for predicting stock price movements. Each article averages 389.71 words, offering enough detail for the model to interpret market sentiment. With 29,908 unique words, the dataset captures a broad range of financial and contextual terms, helping identify nuanced relationships between news content and stock price changes. The dataset's lexical richness is 0.02, meaning key financial terms are frequently repeated, which is typical in financial news. Common phrases include "year," "revenue," and "stock."
The label distribution is fairly balanced, with 1,657 articles (53.4%) linked to stock price increases (label 1) and 1,446 articles (46.6%) linked to price decreases (label 0). This balance helps the model generalize to new data by exposing it to both upward and downward price movements.
2. **Most Common Words:** The 50 most frequent words include 'year,' 'company,' 'stock,' and others like 'nasdaq,' 'revenue,' 'zacks,' and 'earning,' reflecting the dataset's focus on Nvidia and stock performance.
3. **Most Indicative Words:** Using a Logistic Regression model, the most indicative words for stock price movements were identified. For price decreases (class 0), words like "80m," "board," "nice," "expanding," and "fell" often appear in negative contexts, signaling challenges or downturns. Words like "stiff" and "bid" also indicate financial difficulties.
For price increases (class 1), words like "21," "rose," "advancing," "strategist," and "seems" signal positive market activity. Terms like "remained," "discretionary," and "flat" suggest market stability.

Numerical Representation

Using TF-IDF for document representation provided a more meaningful approach than simple word counts. TF-IDF emphasizes words that are frequent in a document but rare across the dataset, highlighting important terms while downplaying common, less informative words like "the" or "and."
By calculating cosine similarity between TF-IDF vectors, I identified the most and least similar document pairs. The most similar articles had significant content overlap, especially regarding Nvidia's stock performance on specific days, while the least similar pairs involved articles on unrelated topics, such as Nvidia stock versus articles on other companies.

Pros and Cons of Training Your Own Word2Vec Model vs. Using a Pretrained One

Training your own Word2Vec model:

Pros	Cons
It better captures the nuances of vocabulary related to stock movements and company performance.	Requires significant computational power and a large enough corpus for effective training.
Custom training on the Nvidia-specific corpus allows the model to learn domain-specific context and relationships.	May struggle with general word relationships if the dataset is small, potentially affecting performance on common terms.

Using a Pretrained Word2Vec model:

Pros	Cons
Pretrained models are trained on large, general datasets, capturing broad and well-established word relationships.	Pretrained models may fail to capture domain-specific relationships relevant to Nvidia and financial news.
Ideal for smaller datasets, as pretrained embeddings already include valuable linguistic information.	Financial terms and subtle language specific to this context may not be well-represented in general pretrained models.

I chose the Skip-Gram algorithm for Word2Vec because it works better with smaller datasets and captures the meanings of rare words, which are common in financial news. Skip-Gram predicts context words from a target word, creating richer representations for uncommon terms. CBOW trains faster and works better with larger datasets but is less effective at capturing rare words. I trained the Word2Vec model on the entire corpus, not just Nvidia-related content, to capture both general financial terms and Nvidia-specific relationships. Focusing only on Nvidia articles would limit the richness of embeddings, especially for important but less frequent terms in the financial context. To create document embeddings, I averaged the word vectors in each document, producing a single vector representing the document's overall meaning. I applied TF-IDF weighting, giving more important words higher significance. This resulted in TF-IDF weighted averages of word vectors, improving each document's representation by emphasizing key terms.

Training Procedure and Evaluation

To train the model and optimize parameters, I followed a structured approach

involving data preprocessing, dataset splitting, and applying various machine learning algorithms. The key steps were as follows: I first converted the text data into document embeddings using Word2Vec and TF-IDF weighted averages of the word vectors. These embeddings served as input features for the models.The dataset was then split into training and testing sets with an 80-20 split, preserving the label distribution (price increases and decreases) through stratified sampling. This ensured that both sets were representative of the overall dataset. The features (X) were the document embeddings, and the target (y) was the label indicating stock price movement (1 for increase, 0 for decrease).

Model Training and Evaluation

I tested several machine learning algorithms to assess their effectiveness in predicting stock price movements based on news articles. Here's a summary of the models and their performance:

1. Naive Bayes (GaussianNB): I trained a Naive Bayes model, which assumes feature independence, making it efficient for high-dimensional data like text. The model achieved 50.08% accuracy on the test set. It was better at predicting stock price increases (precision: 0.53, recall: 0.60) than decreases (precision: 0.46, recall: 0.38). While simple and efficient, Naive Bayes struggled to capture complex relationships in the data.

2. Gradient Boosting Classifier: Training: Gradient Boosting builds decision trees sequentially to minimize loss. I trained it to iteratively improve predictions. The model achieved 53% accuracy. It performed better in predicting stock price increases (precision: 0.55, recall: 0.68) than decreases (precision: 0.50, recall: 0.37). This improved performance indicates that Gradient Boosting captured more complex patterns compared to Naive Bayes.

I also experimented with other machine learning models like Logistics Regression, RFM, SVM and neural network to compare the performance .

Hyperparameter Tuning

I used cross-validation and grid search techniques for tuning hyperparameters to optimize model performance. This process involved:

- Random Forest and Gradient Boosting: Tuning the number of trees, learning rates, and other hyperparameters.
- SVM: Optimizing the kernel and regularization parameters.
- Neural Network: Adjusting learning rates, network architecture, and the number of layers to find the best balance between learning speed and model accuracy.

Experimental Design for Evaluation

To ensure robust model evaluation, I used the following approach:

- Train-Test Split: I split the data while maintaining proportional representation of both classes (price increases and decreases) in the training and test sets.
- Evaluation Metrics: I evaluated the models using accuracy, precision, recall, and F1-score. Given the slight label imbalance, I prioritized precision and recall for both classes. The F1-score was particularly useful as it balances precision

and recall, providing a comprehensive measure of the models' ability to predict both stock price increases and decreases effectively.
- Cross-Validation: I applied cross-validation during hyperparameter tuning to avoid overfitting and ensure consistent performance across different data splits.

Discussion & Limitations

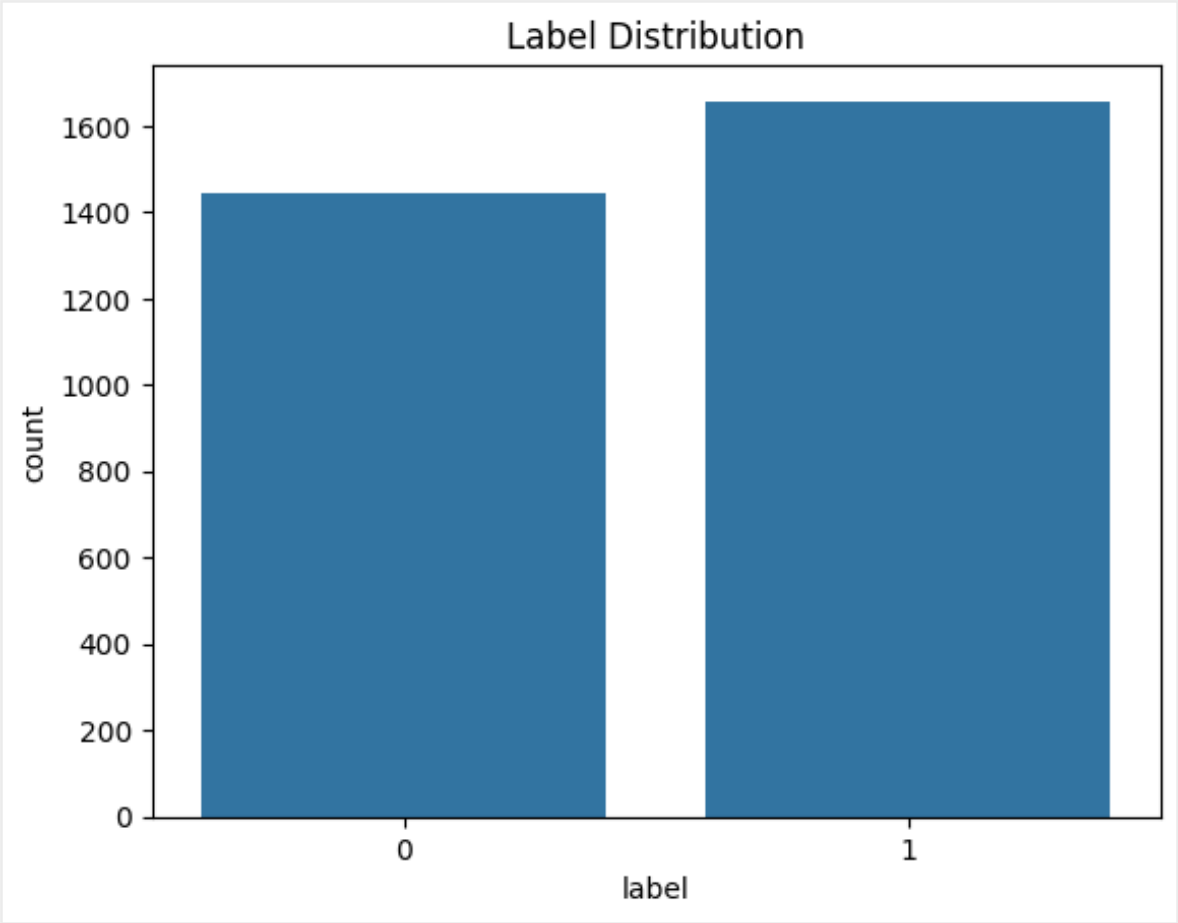
Practical Implementation
1. Automated Data Pipeline: Set up a system to scrape, clean, and preprocess financial news articles continuously, transforming them into document embeddings for timely predictions.
2. Real-Time Financial Integration: Integrate real-time financial APIs to ensure predictions are based on the latest news and stock data.[5]
3. Periodic Model Re-Training: Re-train models periodically or after major market events to capture evolving patterns.
4. User Interface: Develop a dashboard where users can input new articles and get predictions, with explanations for transparency.[4]

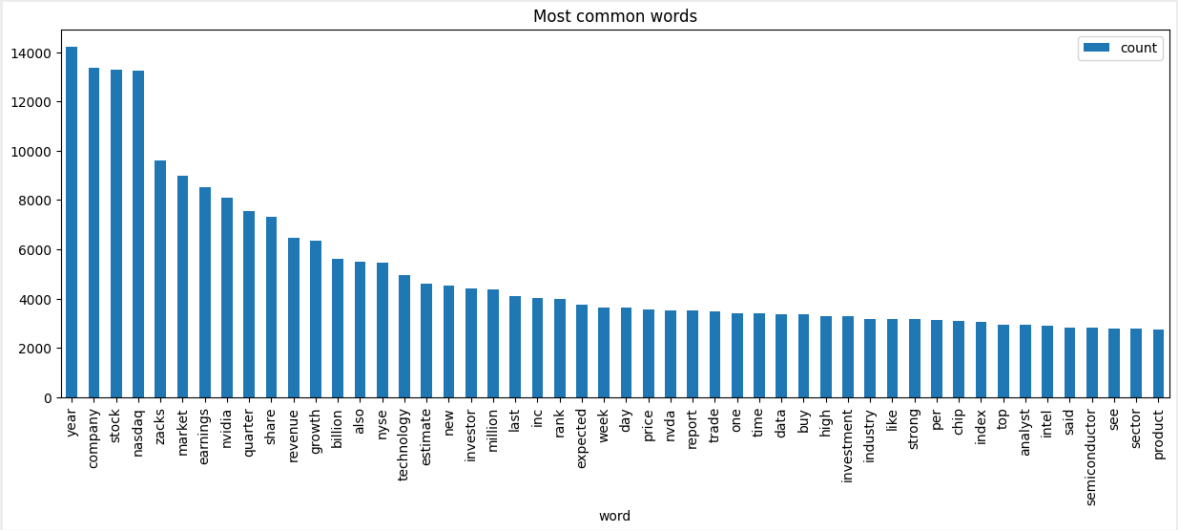
Limitations
1. Loss of Context: Embeddings like Word2Vec or TF-IDF may miss nuanced financial language.
2. News Data Reliance: Predictions depend on the availability and timeliness of news articles.
3. Limited Generalization: The model is trained on Nvidia-specific news, limiting its broader application.
4. Binary Stock Movement: Simplifies predictions to increase/decrease; regression models could offer more insights.
5. Computational Costs: Complex models like Gradient Boosting can be costly for large datasets and real-time predictions.[6]

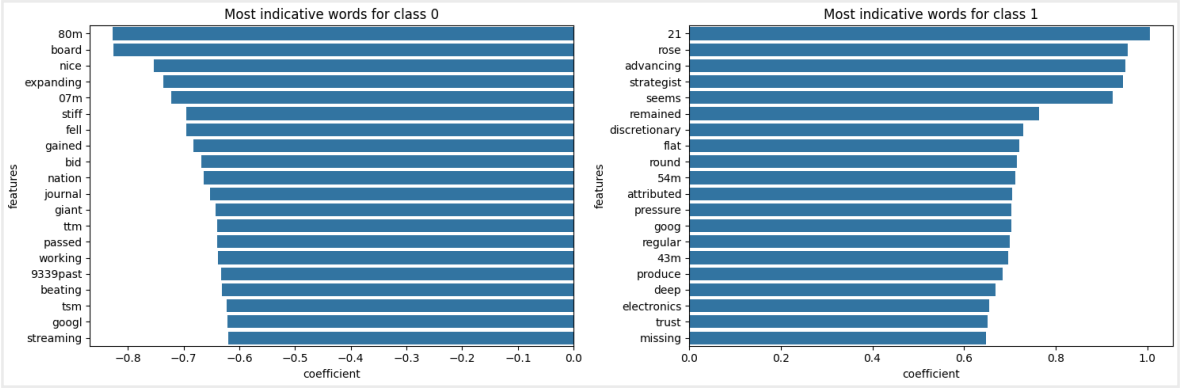
Suggestions for Improvement
1. Pretrained Models: Use models like BERT or GPT to improve text representation and capture nuances.
2. Additional Features: Include stock data, social media sentiment, or technical indicators for more accurate predictions.[5][4]
3. Regression Models: Use regression to predict stock price changes, not just binary movement.
4. Hierarchical Models: Use multi-stage models for better accuracy in predicting movement and magnitude.

Model Tuning and Performance
- Naive Bayes: After tuning, the best smoothing value was 10, resulting in a 53.38% cross-validated score.
- Gradient Boosting: After hyperparameter tuning, the best model achieved an F1 score of 0.548, outperforming other models. [6]. Further improvements could come from using domain-specific embeddings or transformer models.

Appendex


Label Distribution
count
label
0
1
0
200
400
600
800
1000
1200
1400
1600


Most common words
count
word
0
2000
4000
6000
8000
10000
12000
14000
year
company
stock
nasdaq
zacks
market
earnings
nvidia
quarter
share
revenue
growth
billion
also
nyse
technology
estimate
new
investor
million
last
inc
rank
expected
week
day
price
nvda
report
trade
one
time
data
buy
high
investment
industry
like
strong
per
chip
index
top
analyst
intel
said
semiconductor
see
sector
product



References

1. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., ... & Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. arXiv preprint arXiv:1309.0238. https://doi.org/10.5281/zenodo.2783190
2. All Things AI. (2018, August 17). In-depth parameter tuning for gradient boosting. *Medium*. https://medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae
3. OpenCV.org. (n.d.). Introduction to Support Vector Machines (SVM). *OpenCV Documentation*. https://docs.opencv.org/4.x/d1/d73/tutorial_introduction_to_svm.html
4. Baker, M. J., & Wurgler, J. (2006). Investor sentiment and the cross-section of stock returns. *The Journal of Finance, 61*(4), 1645-1680. https://doi.org/10.1111/j.1540-6261.2006.00885.x
5. Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science, 2*(1), 1-8. https://doi.org/10.1016/j.jocs.2010.12.007
6. Gdebiri, D., Alamro, M., & Khashman, A. (2020). Financial stock market prediction using artificial neural networks and evolutionary techniques. *Procedia Computer Science, 170*, 175-182. https://doi.org/10.1016/j.procs.2020.03.027
7. [NVIDIA Corporation] NVIDIA Corporation. Official website of nvidia. https://www.nvidia.com/en-us/. Accessed: 2024-10-02.