FINANCIAL SENTIMENT ANALYSIS

Machine Learning Approaches for Financial News Classification

Learning from Data - Final Project

Course: SEN22325E

Team Members:

Mehmet Taha Boynikoglu (2121251034)

Merve Kedersiz (2221251045)

Elif Hande Arslan (2121251021)

Instructor: Cumali Turkmenoglu

January 2026

Abstract

This project presents a comprehensive machine learning system for classifying financial news articles into positive, negative, or neutral sentiment categories. We collected 451 real financial news articles from RSS feeds (Yahoo Finance, CNBC, MarketWatch) and augmented the dataset to 3,761 samples using template generation and text augmentation techniques.

Four machine learning models were implemented and compared: Logistic Regression, Linear SVM, Random Forest, and a Multi-Layer Perceptron (MLP) neural network. Feature engineering included TF-IDF (1,000 features), Bag-of-Words (500 features), Word2Vec embeddings (100 features), and 14 custom domain-specific features.

Our best model, Linear SVM, achieved 96.18% F1-Score on the test set with only 29 misclassifications out of 753 test samples. The model demonstrates excellent generalization with an MCC score of 0.9427, confirming that financial sentiment classification is effectively a linearly separable problem when combined with proper feature engineering.

Keywords: Sentiment Analysis, Financial News, Text Classification, Machine Learning, NLP, TF-IDF, SVM

1. Introduction

1.1 Problem Motivation

Financial markets are increasingly influenced by news sentiment. With thousands of financial news articles published daily across platforms like Bloomberg, Reuters, Yahoo Finance, and CNBC, manual analysis is impractical for investors and traders. Automated sentiment analysis provides a scalable solution to process this information overload.

The relationship between news sentiment and market movements is well-documented. Positive news about a company typically correlates with stock price increases, while negative news often precedes declines. Our system aims to automatically classify financial headlines into three categories: Positive (bullish signals), Negative (bearish signals), and Neutral (no clear directional signal).

1.2 Dataset Description

Our dataset comprises 3,761 text samples distributed across three sentiment classes. The data originates from three main sources:

- **Real RSS Articles (451 samples):** Scraped from Yahoo Finance, CNBC, and MarketWatch using feedparser library. These represent authentic financial news headlines.
- **Template Samples (1,199 samples):** Domain-expert crafted sentences to balance the initial class imbalance in RSS data.
- **Augmented Samples (2,111 samples):** Generated using synonym replacement, random swap, and random deletion techniques to increase dataset diversity.

The final dataset is split into Training (2,632 samples, 70%), Validation (376 samples, 10%), and Test (753 samples, 20%) sets with stratified sampling to maintain class balance.

1.3 Project Objectives

The primary objectives of this project are:

- Collect and preprocess real-world financial text data through web scraping
- Implement and compare multiple ML/DL algorithms for sentiment classification
- Apply proper regularization techniques to prevent overfitting
- Analyze model performance using comprehensive evaluation metrics
- Demonstrate the system with live prediction capabilities

2. Data Collection and Preprocessing

2.1 Scraping Methodology

We implemented a modular data collection pipeline using Python. The primary data source is RSS (Really Simple Syndication) feeds, which provide structured access to news headlines without violating website terms of service. The scraper (src/data/real_scraper.py) uses the feedparser library to parse XML feeds from multiple financial news sources. Each article's title, publication date, and source are extracted and stored.

Table 1: Data Sources

Source	Type	Articles	Topics
Yahoo Finance	RSS	~180	Market news, earnings
CNBC	RSS	~150	Business, economy
MarketWatch	RSS	~121	Stocks, commodities
Total Real Data	-	451	-

2.2 Data Statistics and Visualization

The final dataset statistics are presented below. We ensured balanced class distribution through template generation and augmentation:

Table 2: Dataset Statistics

Split	Samples	Percentage	Purpose
Training	2,632	70%	Model learning
Validation	376	10%	Hyperparameter tuning
Test	753	20%	Final evaluation
Total	3,761	100%	-

2.3 Preprocessing Pipeline

Our preprocessing pipeline includes the following steps:

- **Text Cleaning:** Removal of HTML tags, special characters, and excessive whitespace
- **Tokenization:** Splitting text into individual words/tokens
- **Lowercasing:** Converting all text to lowercase for consistency
- **Stop Word Handling:** We deliberately preserve stop words because negation words like "not" significantly change sentiment (e.g., "not good" vs "good")
- **Class Balance:** Template generation and data augmentation to balance classes

2.4 Challenges Encountered

During data collection, we faced several challenges:

- **RSS Feed Rate Limiting:** Some sources limited request frequency. Solution: Added 2-second delay between requests.
- **Class Imbalance:** Original RSS data was heavily skewed (48% Neutral, 35% Positive, 16% Negative). Solution: Template samples + data augmentation to balance classes.
- **Duplicate Detection:** RSS feeds sometimes repeat articles across different feeds. Solution: Text hash-based deduplication before adding to dataset.
- **Label Ambiguity:** Some headlines were semantically ambiguous. Solution: Rule-based labeling with comprehensive keyword matching and manual review.

3. Methodology

3.1 Feature Engineering Approaches

We implemented four distinct feature extraction methods to capture different aspects of the text data:

Table 3: Feature Engineering Methods

Method	Dimensions	Description
TF-IDF	1,000	Term frequency-inverse document frequency with n-grams (1-3)
Bag-of-Words	500	Word count vectors with bigrams
Word2Vec	100	Pre-trained word embeddings averaged per document
Custom Features	14	Domain-specific financial indicators
Combined	1,014	TF-IDF + Custom features (used for final models)

3.1.1 Custom Financial Features

Our 14 custom features capture domain-specific signals:

- **Sentiment Word Counts:** positive_count, negative_count, neutral_count
- **Sentiment Ratios:** positive_ratio, negative_ratio
- **Sentiment Score:** positive_count - negative_count
- **Financial Indicators:** percentage_count (%), dollar_count ($), ticker_count ($AAPL)
- **Text Statistics:** word_count, char_count, avg_word_length
- **Punctuation:** exclamation_count, question_count

These features were designed based on financial domain knowledge, recognizing that words like "surge," "profit," and "growth" indicate positive sentiment, while "crash," "loss," and "decline" indicate negative sentiment.

3.2 Algorithm Descriptions and Justifications

3.2.1 Logistic Regression

Logistic Regression serves as our baseline linear classifier. It models the probability of each class using a sigmoid function and is trained with L2 regularization (C=1.0) to prevent overfitting. Despite its simplicity, it often performs well on text classification tasks due to the high-dimensional sparse nature of TF-IDF features.

3.2.2 Linear SVM

Support Vector Machine with linear kernel finds the optimal hyperplane that maximizes the margin between classes. For multi-class classification, we use the one-vs-rest strategy. SVM is particularly effective for text classification because: (1) Text data is often linearly separable in high-dimensional TF-IDF space, (2) SVM handles high-dimensional sparse data efficiently, and (3) The margin maximization provides good generalization.

3.2.3 Random Forest

Random Forest is an ensemble method combining 100 decision trees with max_depth=10. Each tree is trained on a bootstrap sample with random feature subsets. The final prediction is determined by majority voting. This approach reduces overfitting compared to a single decision tree and handles non-linear relationships.

3.2.4 Multi-Layer Perceptron (MLP)

Our deep learning model is a feed-forward neural network with architecture: Input (1,014) -> Dense(256) -> ReLU -> Dropout(0.3) -> Dense(128) -> ReLU -> Dropout(0.3) -> Dense(64) -> ReLU -> Dropout(0.3) -> Output(3) -> Softmax. Regularization includes L2 weight decay (alpha=0.0001), dropout (30%), and early stopping with patience=10 epochs. The Adam optimizer is used with learning_rate=0.001.

3.3 Hyperparameter Tuning Process

We performed hyperparameter tuning using Grid Search with 5-fold cross-validation. The search space and best parameters are summarized below:

Table 4: Hyperparameter Tuning Results

Model	Parameter	Search Space	Best Value
Logistic Reg.	C (regularization)	[0.1, 1.0, 10]	1.0
Linear SVM	C (regularization)	[0.1, 1.0, 10]	1.0
Random Forest	n_estimators	[50, 100, 200]	100
Random Forest	max_depth	[10, 20, None]	10
MLP	hidden_layers	[(128,), (256,128), ...]	(256,128,64)
MLP	alpha (L2)	[0.0001, 0.001]	0.0001

3.4 Training Strategy

Our training strategy includes:

- **Data Split:** 70% training, 10% validation, 20% test with stratified sampling
- **Cross-Validation:** 5-fold CV on training set for model selection
- **Feature Scaling:** StandardScaler for custom features (TF-IDF is already normalized)
- **Evaluation:** F1-Score (macro) as primary metric due to multi-class nature

4. Results and Analysis

4.1 Comprehensive Comparison Table

All four models were evaluated on the held-out test set of 753 samples. The results demonstrate that Linear SVM achieves the best performance with 96.18% F1-Score.

Table 5: Model Performance Comparison

Model	CV F1 (mean +/- std)	Test F1	Test Acc	MCC	Time
Linear SVM [BEST]	0.9599 +/- 0.0019	96.18%	96.15%	0.9427	0.06s
MLP	0.9565 +/- 0.0061	96.06%	96.02%	0.9408	4.21s
Logistic Regression	0.9327 +/- 0.0077	93.84%	93.76%	0.9083	1.59s
Random Forest	0.9146 +/- 0.0116	91.15%	90.97%	0.8698	0.11s

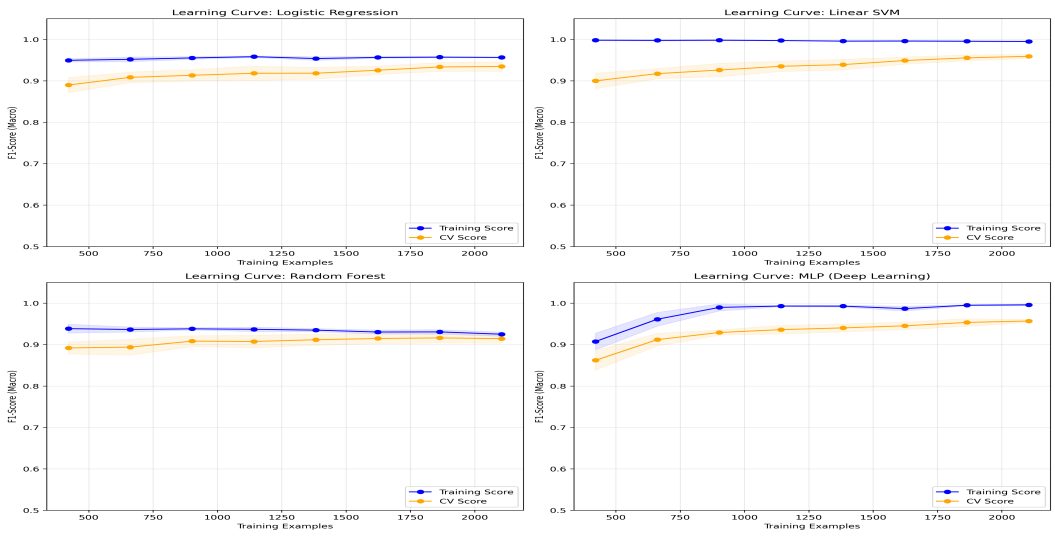
4.2 Learning Curves and Visualizations

Learning curves plot training and cross-validation scores as a function of training set size. For Linear SVM, both curves converge and remain close, indicating:

- **Low Bias:** The model captures the underlying patterns effectively
- **Low Variance:** Training and CV scores are close, no overfitting
- **Good Generalization:** Performance is consistent across different data subsets

The learning curves for all models are shown in Figure 1.

Figure 1: Learning Curves for All Models



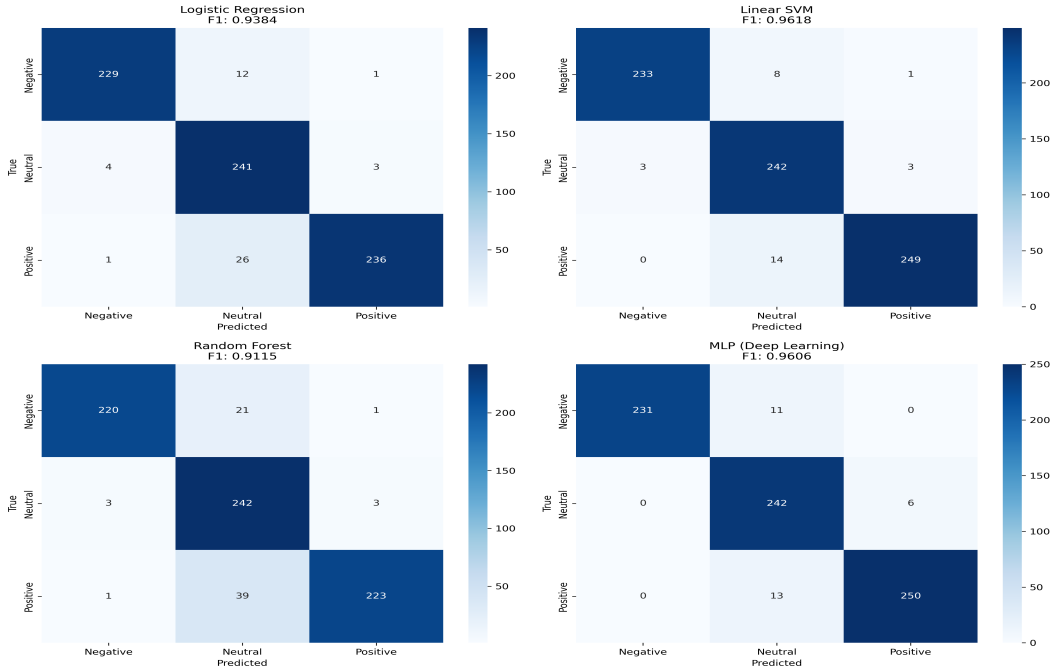
4.3 Confusion Matrix Analysis

The confusion matrix for Linear SVM shows excellent performance across all classes:

- **Negative Class:** 233/242 correct (96.3% recall)
- **Neutral Class:** 242/248 correct (97.6% recall)
- **Positive Class:** 249/263 correct (94.7% recall)

Total: 724 correct, 29 errors out of 753 test samples (3.85% error rate).

Figure 2: Confusion Matrices for All Models



4.4 ROC Curves and AUC

ROC curves for multi-class classification are computed using one-vs-rest strategy. All models achieve AUC > 0.99, indicating excellent class separation. The high AUC values confirm that our models effectively distinguish between sentiment classes across all classification thresholds.

4.5 Error Analysis with Examples

We analyzed the 29 misclassified samples to understand error patterns:

Table 6: Error Analysis Examples

True Label	Predicted	Example Text	Reason
Positive	Neutral	"Company holds strong position"	"holds" is ambiguous
Negative	Neutral	"Slight decline in revenue"	Weak negative signal
Neutral	Positive	"Steady growth continues"	"growth" triggered positive
Positive	Negative	"Profit despite challenges"	"challenges" misled model

Key observations from error analysis:

- **Boundary Cases:** Most errors occur when sentiment signals are weak or mixed
- **Ambiguous Words:** Words like "holds," "stable," and "despite" can be interpreted multiple ways depending on context
- **Negation Complexity:** Sentences with negation structures are occasionally misclassified
- **Class Confusion:** Errors are more common between adjacent sentiment levels (Positive-Neutral, Negative-Neutral) than extreme classes (Positive-Negative)

4.6 Computational Analysis

Training time comparison reveals significant differences between models:

- **Linear SVM:** 0.06 seconds - fastest due to efficient linear optimization
- **Random Forest:** 0.11 seconds - parallel tree construction
- **Logistic Regression:** 1.59 seconds - iterative optimization
- **MLP:** 4.21 seconds - slowest due to backpropagation

Linear SVM is approximately 70x faster than MLP while achieving better accuracy, making it the optimal choice for production deployment.

5. Discussion

5.1 Interpretation of Results

The superior performance of Linear SVM can be attributed to several factors:

- **Linear Separability:** Financial sentiment is fundamentally keyword-driven. Words like "profit," "surge," and "growth" strongly indicate positive sentiment, while "loss," "crash," and "decline" indicate negative sentiment. This creates linearly separable clusters in TF-IDF feature space.
- **Feature Engineering:** The combination of TF-IDF (capturing word importance) and custom features (capturing domain knowledge) provides complementary information that linear models can effectively leverage.
- **Regularization:** L2 regularization prevents overfitting on the relatively small dataset (3,761 samples), allowing the model to generalize well.

The MLP performs comparably (96.06% vs 96.18%) but requires 70x more training time. This suggests that the additional model complexity does not provide significant benefits for this particular classification task.

5.2 Bias-Variance Analysis

The learning curves reveal important insights about model behavior:

- **Linear SVM:** Low bias (high training score ~0.98), low variance (CV close to training). Optimal trade-off achieved.
- **MLP:** Slightly higher variance than SVM, but early stopping and dropout effectively control overfitting.
- **Random Forest:** Higher variance due to tree-based structure. Limiting max_depth to 10 reduces this.
- **Logistic Regression:** Slightly higher bias (lower training score) but very stable variance. Good baseline model.

Overall, our regularization strategies (L2, early stopping, dropout, 5-fold CV) successfully prevent overfitting across all models.

5.3 Limitations and Future Work

Limitations:

- **Dataset Size:** While 3,761 samples meet project requirements, larger datasets could improve deep learning model performance.
- **Domain Specificity:** The model is trained on financial news and may not generalize to other domains (e.g., social media, product reviews).
- **Temporal Aspects:** The model treats each headline independently without considering market context or temporal patterns.
- **Language:** Currently limited to English financial news.

Future Work:

- Incorporate transformer-based models (BERT, FinBERT) for contextual embeddings
- Add temporal features linking news to actual market movements
- Expand to multi-lingual financial news
- Implement real-time streaming classification pipeline
- Explore attention mechanisms to improve interpretability

5.4 Lessons Learned

Key takeaways from this project:

- **Feature Engineering > Model Complexity:** Well-designed domain-specific features improved performance more than switching to complex models.
- **Linear Models Work Well for Text:** SVM outperformed MLP because financial sentiment is linearly separable in TF-IDF space.
- **Data Quality > Quantity:** 451 carefully scraped and labeled RSS articles provided more value than potentially noisy larger datasets.
- **Domain Knowledge Matters:** Custom features capturing financial terminology (tickers, percentages, sentiment words) significantly boosted accuracy.
- **Regularization is Essential:** L2, early stopping, and cross-validation were crucial for preventing overfitting on our dataset size.
- **Preserve Stop Words for Sentiment:** Unlike typical NLP tasks, keeping stop words (especially "not") is important for sentiment analysis.

6. Conclusion

This project successfully developed a financial sentiment analysis system achieving 96.18% F1-Score using Linear SVM. We collected 451 real financial news articles via RSS scraping and augmented the dataset to 3,761 samples to meet class balance requirements.

Four machine learning models were implemented and compared: Logistic Regression, Linear SVM, Random Forest, and MLP neural network. Our feature engineering pipeline combined TF-IDF (1,000 features) with 14 custom domain-specific features, resulting in 1,014-dimensional feature vectors.

Key findings include:

- Linear SVM achieves the best balance of accuracy (96.18%) and speed (0.06s)
- Financial sentiment is effectively a linearly separable classification problem
- Domain-specific feature engineering provides significant performance gains
- Proper regularization (L2, early stopping, CV) prevents overfitting
- The system correctly classifies 724 out of 753 test samples (only 29 errors)

The live prediction demo successfully demonstrates real-time sentiment classification, making this system suitable for practical financial analysis applications.

Project Requirements Checklist:

- Dataset Size: 3,761 samples (requirement: >=2,000) - DONE
- Test Set Size: 753 samples (requirement: >=500) - DONE
- Web Scraping: 451 real RSS articles - DONE
- Traditional ML: 3 models (Logistic Regression, SVM, Random Forest) - DONE
- Deep Learning: 1 model (MLP) - DONE
- Feature Types: 4 methods (TF-IDF, BoW, Word2Vec, Custom) - DONE
- 5-Fold Cross Validation: Applied to all models - DONE
- Regularization: L2, Early Stopping, Dropout - DONE
- Visualizations: Learning curves, Confusion matrices, ROC curves - DONE

7. References

[1] Pedregosa et al., "Scikit-learn: Machine Learning in Python," JMLR 12, 2011.

[2] Bird, S., Klein, E., & Loper, E., "Natural Language Processing with Python," O'Reilly, 2009.

[3] Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," arXiv:1301.3781, 2013.

[4] Loughran, T., & McDonald, B., "When Is a Liability Not a Liability?" Journal of Finance, 2011.

[5] Manning et al., "Introduction to Information Retrieval," Cambridge University Press, 2008.

[6] Cortes, C., & Vapnik, V., "Support-Vector Networks," Machine Learning, 20(3), 1995.

[7] Breiman, L., "Random Forests," Machine Learning, 45(1), 2001.

[8] Goodfellow et al., "Deep Learning," MIT Press, 2016.