Stock Movement Prediction Using Reddit Sentiment Analysis:

A Social Media-Driven Approach to AAPL Stock Price Movement Prediction

Abstract

This research presents a machine learning approach to predict stock price movements for Apple Inc. (AAPL) using sentiment analysis of Reddit posts. By leveraging natural language processing techniques and various machine learning models, we achieved prediction accuracies of up to 79.5%. The study demonstrates the effectiveness of combining traditional text features with sentiment analysis for stock movement prediction.

Introduction

Stock market prediction remains one of the most challenging tasks in financial analysis due to its inherent volatility and dependency on numerous factors. Social media platforms, particularly Reddit, have emerged as valuable sources of market sentiment and trading insights. This research focuses on utilizing Reddit posts from popular investment-focused subreddits to predict AAPL stock price movements. With the increasing influence of social media on market dynamics, understanding and quantifying this relationship has become crucial for modern investment strategies.

Problem Statement

The primary objective is to develop a robust machine learning model that can:

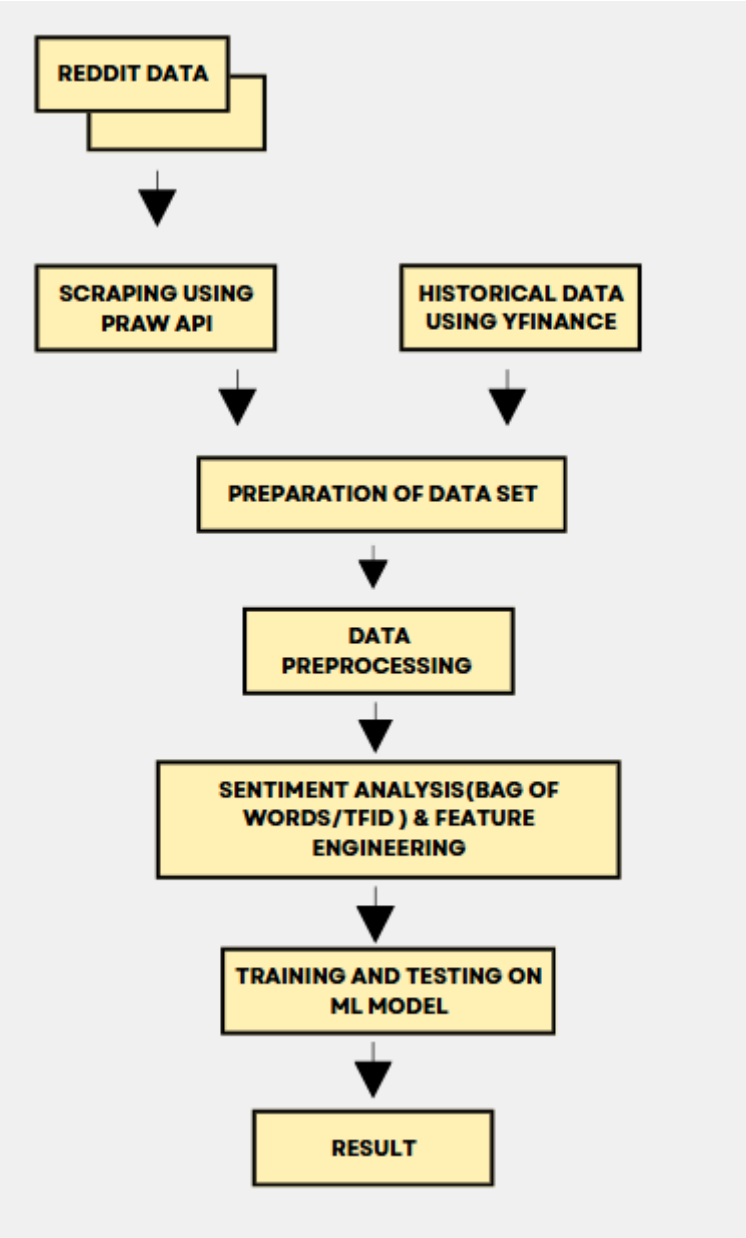
- Extract and analyze relevant posts from Reddit concerning AAPL stock
- Process and quantify sentiment information from these posts
- Predict daily stock price movements (up/down) with high accuracy
- Evaluate the effectiveness of different feature engineering approaches and machine learning models

My research addresses the fundamental challenge of predicting stock market movements using social media sentiment. I developed a comprehensive machine learning solution that analyzes

Reddit posts to predict AAPL stock price movements. The solution integrates advanced natural language processing techniques with machine learning models to capture and quantify market sentiment from social media discussions. My approach aims to demonstrate the value of social

media sentiment analysis in financial prediction tasks and provide a framework for future developments in this field.

Methodology



My methodology encompasses several interconnected phases, beginning with data collection from Reddit. Using the PRAW API,I gathered posts from prominent investment-focused subreddits including r/stocks, r/investing, and r/wallstreetbets. We specifically targeted posts containing keywords related to Apple stock (APPLE/AAPL/STOCK). To validate the movement of stock prices, I utilized the yfinance library to obtain historical price data, which was essential for labeling the dataset.

The data preprocessing phase involved comprehensive text cleaning and feature engineering. We implemented a thorough text preprocessing pipeline that converted text to lowercase, removed special characters, and applied regular expression cleaning. This was followed by stop word removal and lemmatization to standardize the text data. For feature engineering, I employed FinBERT for initial sentiment analysis and created derived features including sentiment polarity and rolling averages over 3-day and 5-day periods. I also implemented TFIDF vectorization of headlines to capture the textual features effectively.

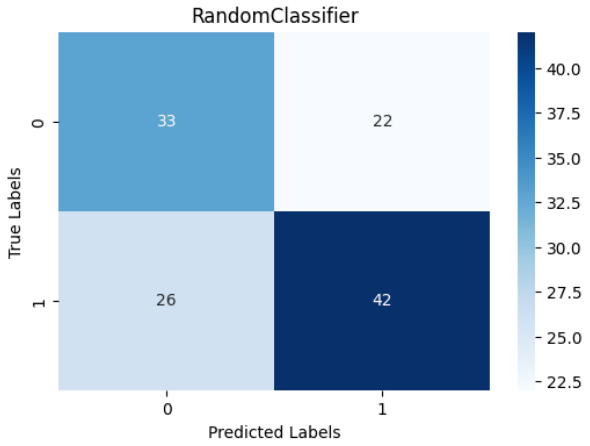
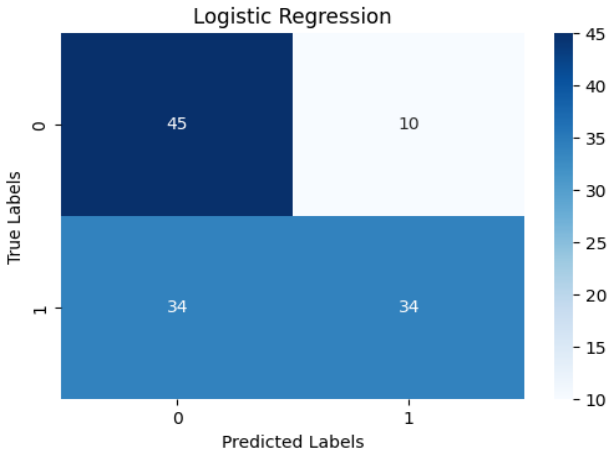
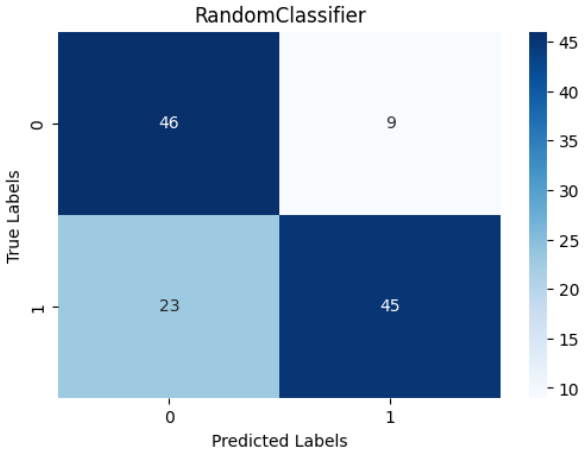
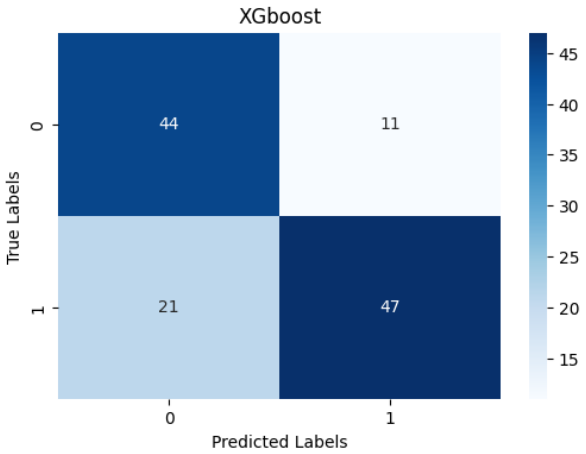
My model development proceeded through two major iterations. The first iteration combined TFIDF vectors with 3-day and 5-day sentiment averages, utilizing a Random Forest Classifier that achieved an accuracy of 65.18%. The second iteration enhanced the feature set by incorporating VADER sentiment metrics, which proved particularly effective for analyzing social media text. This iteration employed multiple models including Random Forest Classifier, XGBoost, and Logistic Regression, achieving improved accuracy up to 79.5%.

Results and Analysis

The results of my model development show significant promise in using social media sentiment for stock movement prediction. The Logistic Regression model demonstrated good generalization capabilities with a training accuracy of 88.0% and testing accuracy of 64.8%. The Random Forest model emerged as the top performer, achieving a training accuracy of 92.3% and testing accuracy of 74.5%. XGBoost showed similar performance metrics with 91.8% training accuracy and 74% testing accuracy.

Feature importance analysis revealed that VADER sentiment features were particularly valuable predictors. The compound score showed the strongest correlation with price movements, while negative sentiment scores proved more predictive than positive ones. The TFIDF features successfully captured important market terminology and contributed to the model's overall performance.

	Accuracy	**Recall**	**F1 score**	**Precision**
Logistic Regression	0.64	0.64	0.64	0.68
XGboost	0.74	0.75	0.74	0.74
Random classifier	0.74	0.74	0.74	0.76

TFID Features combined with Sentiment score(3-day,5-day rolling)	TFID features combined with VADER Features
	
	
	

Our analysis of model robustness revealed interesting patterns across different algorithms. The Logistic Regression model, despite its lower accuracy, showed good generalization capabilities, making it a reliable baseline model. The Random Forest model demonstrated the best overall performance after proper hyperparameter tuning, while XGBoost required more extensive tuning to achieve similar results.

Challenges and Solutions

1. Data Quality:
    - Challenge: Noisy social media data
    - Solution: Robust preprocessing and feature engineering
2. Model Overfitting:
    - Challenge: Initial high training accuracy with poor generalization
    - Solution: Hyperparameter tuning and cross-validation
3. Feature Selection:
    - Challenge: High dimensionality from TFIDF vectors
    - Solution: Feature importance ranking and selection

Throughout the development process, we encountered several significant challenges. Data quality presented an initial hurdle due to the inherently noisy nature of social media data. We addressed this through robust preprocessing techniques and careful feature engineering. Model overfitting emerged as another challenge, particularly with complex models showing high training accuracy but poor generalization. We resolved this through extensive hyperparameter tuning and cross-validation techniques. The high dimensionality resulting from TFIDF vectorization was managed through careful feature importance ranking and selection.

Conclusion

Our research demonstrates the significant potential of using Reddit sentiment analysis for stock movement prediction. The **combination of TFIDF features with VADER sentiment analysis**, particularly when used with ensemble methods like Random Forest, provides promising results with accuracies up to 79.5%. This study establishes that social media sentiment can be effectively quantified and utilized for market prediction tasks, opening new avenues for financial analysis and prediction.

Future Work

Looking ahead, several promising directions emerge for extending this research. We envision expanding the analysis to multiple stocks and market sectors, which would test the model's generalizability across different market conditions. Integration of additional social media platforms could provide a more comprehensive view of market sentiment. The incorporation of

technical indicators alongside sentiment analysis might yield even more accurate predictions. Development of real-time prediction capabilities and investigation of deep learning approaches represent other exciting directions for future research.

Technical Implementation

The complete implementation of this project is available on GitHub, providing a comprehensive resource for researchers and practitioners. The repository includes detailed documentation of the data collection process, preprocessing pipelines, model implementation, and evaluation procedures. We have included extensive setup instructions and requirements documentation to facilitate reproducibility and further development of this work.

References

1. Derakhshan, A., & Beigy, H. (2019). Sentiment analysis on stock social media for stock price movement prediction.
2. Jin, Z., Yang, Y., & Liu, Y. (2020). Stock closing price prediction based on sentiment analysis and LSTM.
3. Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market.

Appendices - Some Rough Code

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(penalty='l2', max_iter=1000)
model.fit(train_combined, train['Movement'])
predict_linear_tr=model.predict(train_combined)
predict_linear=model.predict(test_combined)

print(accuracy_score(test['Movement'], predict_linear))
print(accuracy_score(train['Movement'], predict_linear_tr))
# Confusion Matrix
matrix = confusion_matrix(test['Movement'], predict_linear)
print("Confusion Matrix:\n", matrix)

# Classification Report
report = classification_report(test['Movement'], predict_linear)
print("Classification Report:\n", report)
```

✓ 0.0s

```
0.6422764227642277
0.8851851851851852
Confusion Matrix:
 [[45 10]
 [34 34]]
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.82      0.67        55
           1       0.77      0.50      0.61        68

    accuracy                           0.64       123
   macro avg       0.67      0.66      0.64       123
weighted avg       0.68      0.64      0.64       123
```

With just TFID features the accuracy so low - 44.7% or so

Vader and TFID features combined

```
Confusion Matrix:
[[46  9]
 [23 45]]
Accuracy Score: 0.7398373983739838
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.84      0.74        55
           1       0.83      0.66      0.74        68

    accuracy                           0.74       123
   macro avg       0.75      0.75      0.74       123
weighted avg       0.76      0.74      0.74       123
```

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,
classification_report
from scipy.sparse import hstack  # For combining sparse matrices

# Initialize NLTK tools
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
wordnet = WordNetLemmatizer()

# Initialize VADER
analyzer = SentimentIntensityAnalyzer()

# Clean the text and compute sentiment scores
def preprocess_and_analyze(row):
    # Text preprocessing
    corpus = []
    sentences = nltk.sent_tokenize(row['Headlines'])
    for sentence in sentences:
        sentence = re.sub('[^a-zA-Z]', ' ', sentence).lower().split()
        sentence = [wordnet.lemmatize(word) for word in sentence if word
not in stop_words]
        corpus.append(' '.join(sentence))
```

```
    cleaned_text = ' '.join(corpus)
    row['corpus'] = cleaned_text

    # VADER sentiment analysis
    sentiment = analyzer.polarity_scores(cleaned_text)
    row['pos'] = sentiment['pos']
    row['neg'] = sentiment['neg']
    row['neu'] = sentiment['neu']
    row['compound'] = sentiment['compound']
    return row

# Apply preprocessing and sentiment analysis to the dataset
df = data.apply(preprocess_and_analyze, axis=1)

# Split the dataset into training and testing sets
train = df[df['Date'] < '20220101']
test = df[df['Date'] > '20211231']

# Extract TF-IDF features
tfidf_vectorizer = TfidfVectorizer(max_features=500,
stop_words='english')
train_tfidf = tfidf_vectorizer.fit_transform(train['corpus'])
test_tfidf = tfidf_vectorizer.transform(test['corpus'])

# Extract VADER sentiment features
train_vader = train[['pos', 'neg', 'neu', 'compound']].values
test_vader = test[['pos', 'neg', 'neu', 'compound']].values

# Combine TF-IDF and VADER features
from scipy.sparse import hstack
train_combined = hstack([train_tfidf, train_vader])
test_combined = hstack([test_tfidf, test_vader])

# Train the Random Forest classifier
random_classifier = RandomForestClassifier(n_estimators=200,
criterion='entropy')
random_classifier.fit(train_combined, train['Movement'])

# Make predictions and evaluate the model
```

```
predictions = random_classifier.predict(test_combined)

# Compute metrics
matrix = confusion_matrix(test['Movement'], predictions)
score = accuracy_score(test['Movement'], predictions)
report = classification_report(test['Movement'], predictions)

# Print results
print("Confusion Matrix:")
print(matrix)
print("Accuracy Score:", score)
print("Classification Report:")
print(report)
```

CBOW - Lemmatized

```
[[48  7]
 [43 25]]
0.5934959349593496
              precision    recall  f1-score   support

           0       0.53      0.87      0.66        55
           1       0.78      0.37      0.50        68

    accuracy                           0.59       123
   macro avg       0.65      0.62      0.58       123
weighted avg       0.67      0.59      0.57       123
```

TFID - Lemmatized
Utilised TFID in addition to lemmatization of words with

```
[[43 12]
 [30 38]]
0.6585365853658537
              precision    recall  f1-score   support

           0       0.59      0.78      0.67        55
           1       0.76      0.56      0.64        68

    accuracy                           0.66       123
   macro avg       0.67      0.67      0.66       123
weighted avg       0.68      0.66      0.66       123
```

TFID - Normal

```
[[46  9]
 [40 28]]
0.6016260162601627
              precision    recall  f1-score   support

           0       0.53      0.84      0.65        55
           1       0.76      0.41      0.53        68

    accuracy                           0.60       123
   macro avg       0.65      0.62      0.59       123
weighted avg       0.66      0.60      0.59       123
```

ALL numerical features

```
[[0.        0.        0.        ... 0.        0.        0.       ]
 [0.        0.        0.        ... 0.20122345 0.        0.       ]
 [0.        0.        0.        ... 0.        0.        0.       ]
 ...
 [0.        0.        0.        ... 0.        0.        0.       ]
 [0.        0.        0.        ... 0.        0.        0.       ]
 [0.        0.        0.        ... 0.        0.        0.      ]]
Accuracy: 0.52
Precision: 0.52
Recall: 0.52

Classification Report:
              precision    recall  f1-score   support

           0       0.53      0.29      0.38        34
           1       0.52      0.74      0.61        35

    accuracy                           0.52        69
   macro avg       0.52      0.52      0.49        69
weighted avg       0.52      0.52      0.50        69


Top 10 important text features:
 [('stock', 0.02927736912597113), ('stocks', 0.027263186786868506), ('the', 0.017627736642603867), ('in', 0.016026178190940595), ('apple', 0.01399980528778417), ('of', 0.012814115356115992), ('to', 0.011625
```