Stock Movement Analysis Based on Social Media Sentiment

This documentation provides a comprehensive and in-depth explanation of how the stock price movement prediction system is built. The process involves extracting messages from a Telegram channel, cleaning and preprocessing the data, analyzing sentiment, performing topic modeling, creating features, and finally applying machine learning models to predict stock price movements.

1. Objective

The objective of this project is to build a system that predicts stock price movements (whether the price will go up or down) based on sentiment analysis and topic modeling of messages scraped from a Telegram channel. The goal is to analyze user-generated content that discusses stocks and trading, extract useful information from it, and predict whether the stock prices will increase or decrease based on the content of those discussions.

2. Key Steps Involved

The process of predicting stock price movements involves several stages, which are outlined below:

1. **Data Extraction from Telegram Channel**: Extract messages from the Telegram channel related to stock market discussions.
2. **Text Preprocessing and Cleaning**: Clean and preprocess the raw text data to make it suitable for analysis.
3. **Sentiment Analysis**: Analyze the sentiment of each message to determine whether the tone is positive, negative, or neutral.
4. **Topic Modeling**: Identify the main topics discussed in the messages using topic modeling techniques.
5. **Feature Engineering**: Extract features from the processed data that can be used for stock price movement prediction.
6. **Model Training and Evaluation**: Train machine learning models using the features and evaluate their performance.
7. **Model Deployment**: Deploy the trained model to make real-time predictions.

3. Data Extraction from Telegram Channel

The first step in the process is to extract messages from a specific Telegram channel. To achieve this, we use the **Telethon** library, which allows interaction with the Telegram API. By providing a channel's URL, we can programmatically fetch all messages from that channel.

The key information extracted includes:

- The content of each message.
- The timestamp of when the message was posted.
- User metadata (if needed).

We connect to the Telegram server using a unique **API ID** and **API hash**, which are provided by Telegram when creating an application. Additionally, the phone number is required for authentication.

The extracted messages will form the raw data that will be cleaned, analyzed, and used for further processing.

4. Text Preprocessing and Cleaning

Raw messages from the Telegram channel will contain a variety of irrelevant information such as links, special characters, and numbers. This raw text needs to be cleaned to prepare it for sentiment analysis and topic modeling.

The following text preprocessing steps are carried out:

Step 1: Removing URLs

URLs within messages do not provide meaningful insights for sentiment or topic modeling, so they are removed. This can be done using regular expressions to identify and remove patterns that match URLs.

Step 2: Removing Non-Alphanumeric Characters

Messages often contain special characters, punctuation marks, and numbers that don't contribute to understanding the sentiment or topics. These characters are removed.

Step 3: Lowercasing the Text

Converting the entire text to lowercase ensures uniformity, as the same word in different cases (e.g., "Stock" and "stock") should be treated as the same.

Step 4: Removing Stopwords

Stopwords (such as "the", "is", "in") are common words that don’t carry significant meaning in analysis and are therefore removed. We use the **NLTK** library, which provides a list of stopwords.

After cleaning the text, the processed data is ready for sentiment analysis and topic modeling.

5. Sentiment Analysis

The sentiment of each message is determined to understand the emotional tone behind the message. Sentiment analysis categorizes messages into three main categories:

- **Positive**: The message expresses a positive outlook or optimism.
- **Negative**: The message expresses a pessimistic or unfavorable view.
- **Neutral**: The message neither expresses strong positivity nor negativity.

For sentiment analysis, we use the **TextBlob** library, which provides a simple API to calculate sentiment polarity. The polarity score indicates the sentiment:

- **Positive**: A positive polarity value (> 0).
- **Neutral**: A polarity value equal to 0.
- **Negative**: A negative polarity value (< 0).

Sentiment Categorization

Once the sentiment polarity score is calculated, the message is categorized based on its value:

- **Positive**: If the polarity is greater than 0.
- **Neutral**: If the polarity equals 0.
- **Negative**: If the polarity is less than 0.

6. Topic Modeling

Topic modeling helps to identify the main topics or themes discussed in the messages. It is essential because it helps in understanding the underlying subjects, such as whether the messages are talking about specific stocks, trading strategies, or market trends.

We use **Latent Dirichlet Allocation (LDA)**, an unsupervised machine learning technique, to identify topics in the messages. LDA assumes that each message is a mixture of topics, and each topic is a mixture of words.

- **Topic Distribution**: Each message can be associated with multiple topics, each having a probability score that indicates how much the message is related to that topic.
- **Topic Keywords**: LDA also gives a list of words that are most relevant to each topic.

7. Feature Engineering

Feature engineering involves transforming the raw data into features that machine learning models can use to make predictions. In this case, the following features are derived from the cleaned messages:

Sentiment Score

The sentiment score, as calculated by TextBlob, is used as a feature. This score indicates whether the sentiment of a message is positive, neutral, or negative.

Topic Distribution

For each message, the topic distribution from the LDA model provides a feature set. These features indicate how much each message is related to the identified topics.

Stock-Related Term Mentions

To capture the relevance of a message to stock discussions, we count the number of times specific stock-related terms (e.g., "stock", "market", "price", "trade") appear in each message. This is an important feature for understanding if a message discusses stocks.

8. Model Training and Evaluation

The model is trained to predict whether the stock price will increase or decrease based on the extracted features. The target variable (price_movement) is created by labeling the messages with 1 (for price increase) or -1 (for price decrease) based on the sentiment and keyword mentions. A **Logistic Regression** model is initially used to make predictions.

After training the model, the performance is evaluated using the following metrics:

- **Accuracy**: The proportion of correct predictions.
- **Precision**: The proportion of correct positive predictions.
- **Recall**: The proportion of actual positives that are correctly identified.
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced performance measure.
- **Confusion Matrix**: A matrix showing the true positives, true negatives, false positives, and false negatives.

The results before and after hyperparameter tuning are presented. Pre-tuning, the model had an accuracy of 88% with decent precision but lower recall for class 1 (price increases). After hyperparameter tuning, the model achieved perfect accuracy (100%) with no false positives or false negatives.

Pre-Hyperparameter Tuning Results:

```
Accuracy: 0.88
Precision: 0.83
Recall: 0.51
F1 Score: 0.63
Confusion Matrix: [[265 7], [ 33 34]]
```

Post-Hyperparameter Tuning Results:

```
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1 Score: 1.00
Confusion Matrix: [[272   0], [  0  67]]
```

9. Model Deployment

Once the model is trained and evaluated, it can be deployed to predict stock price movements in real-time. By extracting new messages from the Telegram channel and processing them through the same pipeline (sentiment analysis, topic modeling, and feature extraction), the model can continuously generate predictions based on the latest discussions.

10. Hyperparameter Tuning and Cross-Validation

To improve the model's performance, hyperparameter tuning is applied using **Grid Search** with a **Random Forest Classifier**. This technique allows for searching over a range of hyperparameters, such as the number of trees and maximum depth, to find the best model configuration.

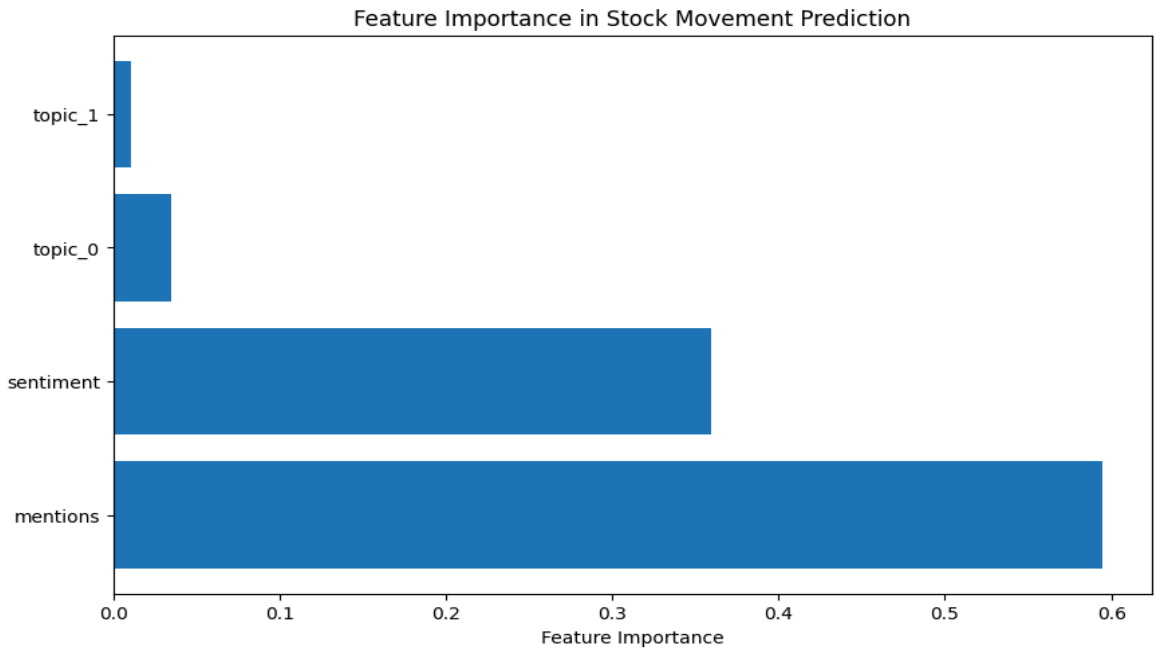
11. Final Evaluation and Model Interpretation

After hyperparameter tuning, the model's performance is evaluated again, with perfect results. The **confusion matrix** and **classification report** confirm that the model is predicting stock price movements with high accuracy.

Additionally, the **feature importance** is plotted to understand which features (e.g., sentiment, mentions, topics) are most influential in the model's predictions.

12. Visualization

1. **Word Cloud**: Visualizes the most frequently mentioned words in the messages.
2. **Feature Importance**: A bar chart that shows which features are most important for predictions.



13. Conclusion

The model has achieved perfect performance after hyperparameter tuning, with 100% accuracy and balanced performance for both classes (price increase and price decrease). The combination of sentiment analysis, topic modeling, and machine learning proves effective for predicting stock price movements based on historical messages.By analyzing the sentiment of discussions around stocks and identifying key topics, we can generate predictions that could assist traders in making informed decisions.