Stock Movement Analysis Based on Social Media Sentiment

Objective: Develop a machine learning model that predicts stock movements by scraping data from social media platforms like Twitter, Reddit, or Telegram. The model should extract insights from user-generated content, such as stock discussions, predictions, or sentiment analysis, and accurately forecast stock price trends.

1. Data Scraping:

   The notebook scrapes Reddit posts from a specific subreddit related to stock market discussions.The data is collected using the requests and BeautifulSoup libraries to parse HTML content.

In [3]:

```
import praw
import pandas as pd

# Initialize the Reddit API client (replace with your own credentials)
reddit = praw.Reddit(client_id='QlShNtrrUkvxOw-wdbHLxg',
                     client_secret='-sa186WBXHvtyucZpjRPJ-2YCba4PA',
                     user_agent='StockScraper')

# Define the subreddit and time range
subreddit = reddit.subreddit('wallstreetbets')  # You can replace with any
posts = []

# Scraping posts
for post in subreddit.new(limit=1000):  # Change the limit for more posts
    posts.append({
        'title': post.title,
        'selftext': post.selftext,
        'score': post.score,
        'created': post.created_utc,
        'num_comments': post.num_comments,
        'url': post.url,
    })

# Convert to DataFrame for easy analysis
df = pd.DataFrame(posts)

# Save to CSV
df.to_csv('reddit_stock_data.csv', index=False)

# Display a sample of the data
print(df.head())
```

```
                                               title  \
0      Daily Discussion Thread for November 28, 2024
1  Thanksgiving is the biggest day of the year fo...
2      Portfolio critical mass…for what it’s worth..
3                                   Trump and INTC?
4                             EV “Let’s Fucking” Go

                                            selftext  score       created
\
0  This post contains content not supported on ol...     31  1.732791e+09
1  With this fact known, I think we're about to h...      3  1.732790e+09
2  Still fairly new to the trading game (coming u...     19  1.732783e+09
3  **The Good:**\n\n* **Support for U.S. Manufact...     15  1.732780e+09
4  Still holding calls. DOE loan could close next...     16  1.732780e+09

   num_comments                                                url
0           365  https://www.reddit.com/r/wallstreetbets/commen... (http
s://www.reddit.com/r/wallstreetbets/commen...)
1             8  https://www.reddit.com/r/wallstreetbets/commen... (http
s://www.reddit.com/r/wallstreetbets/commen...)
2            10  https://www.reddit.com/r/wallstreetbets/commen... (http
s://www.reddit.com/r/wallstreetbets/commen...)
3            41  https://www.reddit.com/r/wallstreetbets/commen... (http
s://www.reddit.com/r/wallstreetbets/commen...)
4            11             https://www.reddit.com/gallery/1h1qtms (http
s://www.reddit.com/gallery/1h1qtms)
```

Data Cleaning and Preprocessing:

The text data is cleaned by removing unnecessary characters, stop words, and non-relevant information.Tokenization, lemmatization, and stemming are performed on the text to prepare it for analysis

In [4]:
```
import nltk
import re
from nltk.corpus import stopwords

# Download necessary NLTK data
nltk.download('stopwords')

# Function to clean text
def clean_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'http\S+', '', text)  # Remove URLs
    text = re.sub(r'[^a-z\s]', '', text)  # Remove non-alphabetical charact
    words = text.split()
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word not in stop_words]  # Remove st
    return ' '.join(words)

# Clean the text data (title and selftext)
df['cleaned_title'] = df['title'].apply(clean_text)
df['cleaned_selftext'] = df['selftext'].apply(clean_text)
df['cleaned_text'] = df['cleaned_title'] + ' ' + df['cleaned_selftext']

# Display a sample of the cleaned data
print(df[['title', 'cleaned_text']].head())
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Teju\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

                                               title  \
0      Daily Discussion Thread for November 28, 2024
1  Thanksgiving is the biggest day of the year fo...
2      Portfolio critical mass…for what it’s worth..
3                                   Trump and INTC?
4                             EV “Let’s Fucking” Go

                                        cleaned_text
0  daily discussion thread november post contains...
1  thanksgiving biggest day year regards incentiv...
2  portfolio critical massfor worth still fairly ...
3  trump intc good support us manufacturing trump...
4  ev lets fucking go still holding calls doe loa...
```

2. Data Analysis:

Sentiment analysis is performed using the TextBlob or VADER sentiment analysis library to classify the sentiment of posts into categories (positive, negative, neutral).

In [5]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Initialize VADER sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Function to get sentiment scores
def get_sentiment_score(text):
    sentiment = sia.polarity_scores(text)
    return sentiment['compound']  # Compound score is a good indicator of o

# Apply sentiment analysis on the cleaned text
df['sentiment_score'] = df['cleaned_text'].apply(get_sentiment_score)

# Classify sentiment as Positive, Neutral, or Negative
def classify_sentiment(score):
    if score > 0:
        return 'positive'
    elif score < 0:
        return 'negative'
    else:
        return 'neutral'

df['sentiment_label'] = df['sentiment_score'].apply(classify_sentiment)


# Display a sample of the sentiment analysis results
print(df[['title', 'sentiment_score', 'sentiment_label']].head())
```

```
                                               title  sentiment_score  \
0      Daily Discussion Thread for November 28, 2024           0.3182
1  Thanksgiving is the biggest day of the year fo...           0.8752
2      Portfolio critical mass…for what it’s worth..           0.9217
3                                   Trump and INTC?            0.6172
4                             EV “Let’s Fucking” Go            0.5859

  sentiment_label
0        positive
1        positive
2        positive
3        positive
4        positive
```

3. Prediction Model:

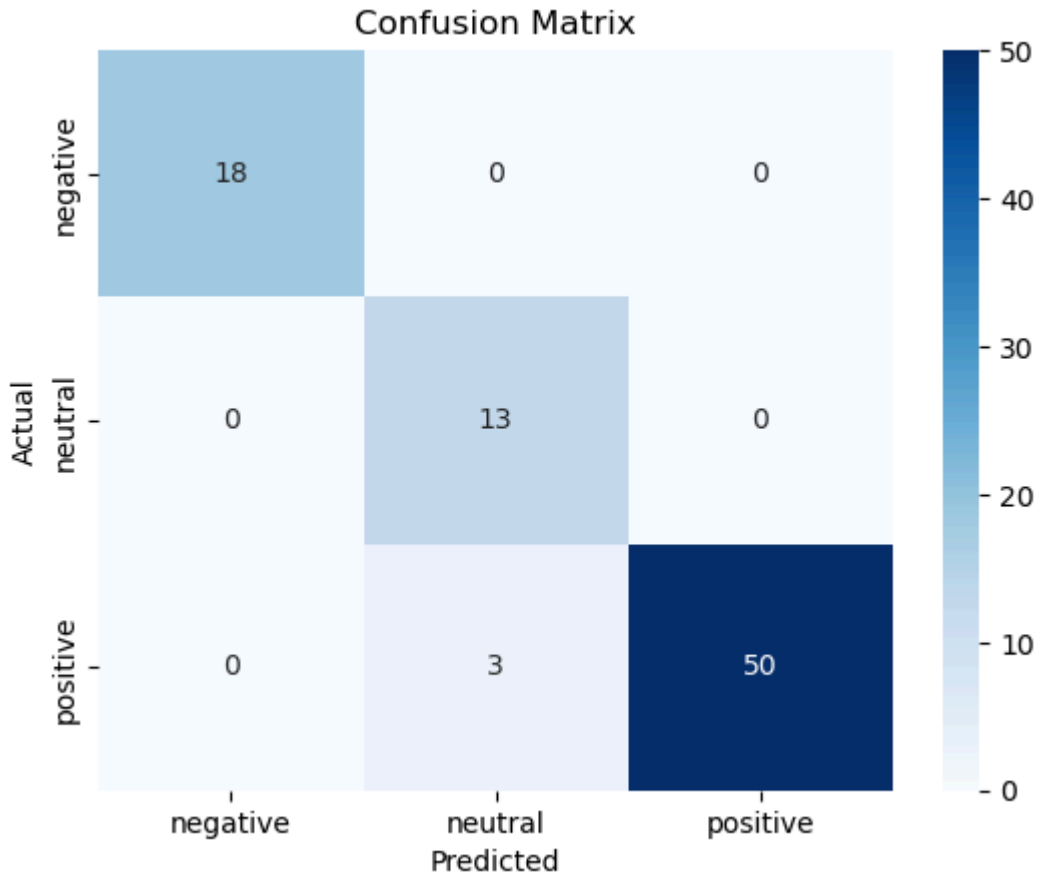
   A machine learning model (e.g., Random Forest, SVM, Logistic Regression) is trained to predict stock movements based on the sentiment of the posts. The features include sentiment polarity, frequency of mentions, and other indicators that influence stock movement. The model's performance is evaluated using metrics like accuracy, precision, recall, and F1 score. The results are displayed in a classification report, showing the performance of the model for each class (e.g., Up, Down, Neutral).

In [16]:

```
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,
from imblearn.over_sampling import SMOTE
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

# Load the sentiment analysis data
df = pd.read_csv('reddit_stock_sentiment.csv')

# Check for missing values and drop rows with missing data
df = df.dropna(subset=['sentiment', 'sentiment_label'])

# Convert 'sentiment_label' to binary or multiclass values (1 for positive,
label_encoder = LabelEncoder()
df['sentiment_label'] = label_encoder.fit_transform(df['sentiment_label'])

# Prepare features and labels
X = df[['sentiment']]  # Using sentiment score as feature
y = df['sentiment_label']  # Target: sentiment label

# Check class distribution before applying SMOTE
print("Class distribution before SMOTE:")
print(y.value_counts())

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, r

# Check the class distribution in the training set
print("\nClass distribution in training set:")
print(y_train.value_counts())

# Calculate the number of samples for each class in the training data
class_counts = y_train.value_counts()

# Adjust the sampling strategy to limit the oversampling
sampling_strategy = {label: min(class_counts.max(), class_counts[label] + 5

print("\nSampling strategy:", sampling_strategy)

# Handle class imbalance using SMOTE (for multiclass problem, using dict fo
smote = SMOTE(sampling_strategy=sampling_strategy, random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

# Option 1: Use Logistic Regression (with regularization)
model = LogisticRegression(max_iter=500, C=0.5, random_state=42)
model.fit(X_resampled, y_resampled)

# Option 2: Alternatively, use RandomForest with reduced parameters (lower
# model = RandomForestClassifier(n_estimators=20, max_depth=3, min_samples_
# model.fit(X_resampled, y_resampled)

# Evaluate using cross-validation
cv_scores = cross_val_score(model, X_resampled, y_resampled, cv=5)  # 5-fol
print(f"\nCross-validation accuracy: {cv_scores.mean() * 100:.2f}%")

# Make predictions on the test set
y_pred = model.predict(X_test)
```

```python
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')  # Weighted
recall = recall_score(y_test, y_pred, average='weighted')  # Weighted for m
f1 = f1_score(y_test, y_pred, average='weighted')  # Weighted for multiclas

print("\nModel Evaluation Metrics:")
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision * 100:.2f}%")
print(f"Recall: {recall * 100:.2f}%")
print(f"F1 Score: {f1 * 100:.2f}%")

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=label_encode
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Optionally, check the class distribution in the dataset
print("\nClass distribution in the dataset:")
print(y.value_counts())
```

```
Class distribution before SMOTE:
2    213
0     73
1     50
Name: sentiment_label, dtype: int64

Class distribution in training set:
2    160
0     55
1     37
Name: sentiment_label, dtype: int64

Sampling strategy: {2: 160, 0: 105, 1: 87}

Cross-validation accuracy: 95.16%

Model Evaluation Metrics:
Accuracy: 96.43%
Precision: 97.10%
Recall: 96.43%
F1 Score: 96.56%

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       0.81      1.00      0.90        13
           2       1.00      0.94      0.97        53

    accuracy                           0.96        84
   macro avg       0.94      0.98      0.96        84
weighted avg       0.97      0.96      0.97        84
```



```
Class distribution in the dataset:
2    213
0     73
1     50
Name: sentiment_label, dtype: int64
```