# Stock Sentiment Analysis Using News Headlines


## Pradipta Sundar Sahoo  (22118054)


Github Repo Link

## 1. Introduction

### 1.1 Background

Stock sentiment analysis is used to understand the mood or sentiment expressed in news headlines and social media mentions about a particular stock. By analyzing these sentiments, traders and investors can predict stock market movements and make informed trading decisions. This project utilizes machine learning algorithms to analyze news headlines and generate trading signals for stock market investments.

### 1.2 Objectives

The main objectives of this project are:

- To collect and preprocess news headlines related to a specific stock.
- To analyze the sentiment of these headlines using sentiment analysis algorithms.
- To generate trading signals based on the aggregated sentiment scores.
- To simulate trades based on these signals and evaluate the trading strategy's performance.

## 2. Data Collection

### 2.1 Stock Data Collection

We use the Yahoo Finance API to fetch historical stock prices for the desired ticker symbol. This data includes the closing prices of the stock, which are essential for evaluating the performance of our trading strategy.

```
def fetch_stock_data(ticker):
    print(f'Fetching stock prices for {ticker} ...')
    stock_data = yf.download(ticker, period = "max")
    stock_data = stock_data[['Close']]
    return stock_data
```

### 2.2 News Headlines Collection

News headlines are scraped from the Business Insider website. The headlines are relevant to the analysed stock and provide the necessary context for sentiment analysis. API from New York Times can also be used but it has limitations for only a limited period of time. Here we are using requests and BeautifulSoup library to scrape the web pages.

```
def fetch_news(ticker):
    print(f'Fetching news for {ticker}...')
    columns = ['datetime','ticker','source', 'headline' ]
    df = pd.DataFrame(columns=columns)
    counter = 0
    ticker = str(ticker).lower()
    print("Fetching news headlines...")
    for page in tqdm(range(1,10)):
        url = f'https://markets.businessinsider.com/news/{ticker}-stock?p={page}'
        response = requests.get(url)
        html = response.text
        soup = BeautifulSoup (html, 'lxml')
        articles = soup.find_all('div', class_ = 'latest-news__story')
        for article in articles:
            datetime = article.find('time', class_ = 'latest-news__date').get('datetime')
            title = article.find('a', class_ = 'news-link').text
            source = article.find('span', class_ = 'latest-news__source').text
            ticker=ticker
            df = pd.concat([pd.DataFrame([[datetime,ticker, source,title]], columns=df.columns), df], ignore_index=True)
            counter += 1

    df['datetime'] = pd.to_datetime(df['datetime'])
    df['date'] = df['datetime'].dt.date
    df['time'] = df['datetime'].dt.time
    df.drop(columns=['datetime'], inplace=True)
    print (f'{counter} headlines scraped from {page+1} pages')
    return df
```

## 3. Sentiment Analysis

### 3.1 Sentiment Calculation

We use the VADER (Valence Aware Dictionary and sentiment Reasoner) sentiment analysis tool from the NLTK library to calculate sentiment scores for each headline. The compound score is used as it provides a single value representing the overall sentiment.

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
analyzer = SentimentIntensityAnalyzer()

def calculate_sentiment_scores(headlines_df):
    headlines_df['sentiment'] = headlines_df['headline'].apply(lambda x: analyzer.polarity_scores(x)['compound'])
    headlines_df['date'] = pd.to_datetime(headlines_df['date'])

    return headlines_df
```

### 3.2 Sentiment Aggregation

We aggregate the sentiment scores by date to generate a daily sentiment score, which is used to generate trading signals.

```
def aggregate_sentiment_scores(headlines_df):
    sentiment_summary = headlines_df.groupby('date')['sentiment'].mean()
    return sentiment_summary
```

## 4. Trading Signal Generation

Based on the aggregated sentiment scores, we generate trading signals. A positive sentiment score above a threshold triggers a buy signal, while a negative score below a threshold triggers a sell signal.

```
def generate_trading_signals(sentiment_summary):
    signals = sentiment_summary.apply(lambda x: 1 if x > 0.15 else (-1 if x < -0.15 else 0))
    return signals
```

### 5. Trade Simulation

We simulate trades based on the generated trading signals. The portfolio's performance, including metrics such as total trades, win percentage, total profit, Sharpe ratio, and maximum drawdown, is calculated.

```
import pandas as pd
from b_data_collection import fetch_news, fetch_stock_data
def simulate_trades(stock_data, trading_signals,capital):
    portfolio = []
    position = 0
    buy_price = 0
    quantity =0
    for date, price in (stock_data['Close'].items()):
        if date in trading_signals.index:
            signal = trading_signals.loc[date]
            if signal == 1 and position == 0:  # Buy signal
                position = 1
                buy_price = price
                quantity = capital/price
                capital = capital%price
                portfolio.append({"date": date, "type": "buy", "price": buy_price, "capital":capital})
            elif signal == -1 and position == 1 and (price>buy_price) :  # Sell signal
                position = 0
                sell_price = price
                profit = (sell_price - buy_price)*quantity
                capital = capital+ quantity*sell_price
                portfolio.append({"date": date, "type": "sell", "price": sell_price,"capital":capital, "profit": profit})

    return pd.DataFrame(portfolio)
```

# 6. Performance Evaluation

## 6.1 Metrics Calculation

We evaluate the performance of the trading strategy using various metrics.

```
def calculate_portfolio_metrics(portfolio):
    total_trades = len(portfolio) // 2
    wins = portfolio[portfolio['type'] == 'sell']['profit'] > 0
    win_percentage = wins.mean() * 100
    total_profit = portfolio[portfolio['type'] == 'sell']['profit'].sum()

    return total_trades, win_percentage, total_profit

def calculate_sharpe_ratio(portfolio, risk_free_rate=0.01):
    daily_returns = portfolio[portfolio['type'] == 'sell']['profit']
    excess_returns = daily_returns - risk_free_rate
    sharpe_ratio = excess_returns.mean() / excess_returns.std()
    return sharpe_ratio

def calculate_max_drawdown(portfolio):
    portfolio['cumulative_profit'] = portfolio['profit'].cumsum()
    cumulative_max = portfolio['cumulative_profit'].cummax()
    drawdown = portfolio['cumulative_profit'] - cumulative_max
    max_drawdown = drawdown.min()
    return max_drawdown
```

## 6.2 Visualization

We visualize the stock prices along with the buy and sell signals for better understanding and analysis.

```python
import plotly.graph_objects as go
def plot_signals(stock_data, portfolio, ticker):
    start_date = portfolio['date'].min()
    end_date = portfolio['date'].max()

    stock_data = stock_data[(stock_data.index >= start_date) & (stock_data.index <= end_date)]

    buy_signals = portfolio[portfolio['type'] == 'buy']
    sell_signals = portfolio[portfolio['type'] == 'sell']
    fig = go.Figure()
    # Add stock price trace
    fig.add_trace(go.Scatter(
        x=stock_data.index,
        y=stock_data['Close'],
        mode='lines',
        name='Stock Price',
        line=dict(color='blue')
    ))

    # Add buy signals
    fig.add_trace(go.Scatter(
        x=buy_signals['date'],
        y=buy_signals['price'],
        mode='markers',
        name='Buy Signal',
        marker=dict(symbol='triangle-up', color='green', size=10)
    ))

    # Add sell signals
    fig.add_trace(go.Scatter(
        x=sell_signals['date'],
        y=sell_signals['price'],
        mode='markers',
        name='Sell Signal',
        marker=dict(symbol='triangle-down', color='red', size=10)
    ))
    # Update Layout for better presentation
    fig.update_layout(
        title=f'Stock Price with Buy and Sell Signals for {ticker}',
        xaxis_title='Date',
        yaxis_title='Price',
        legend_title='Legend',
        hovermode='x'
    )
    fig.show()
```

# 7. Results

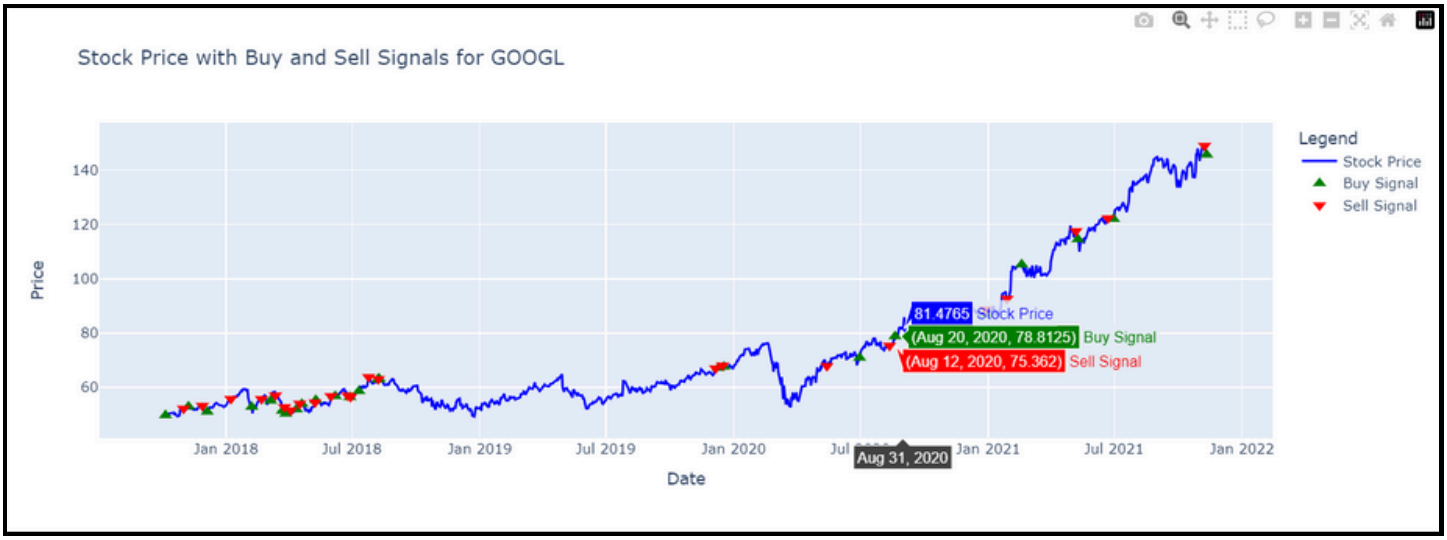
## 7.1 Portfolio Performance

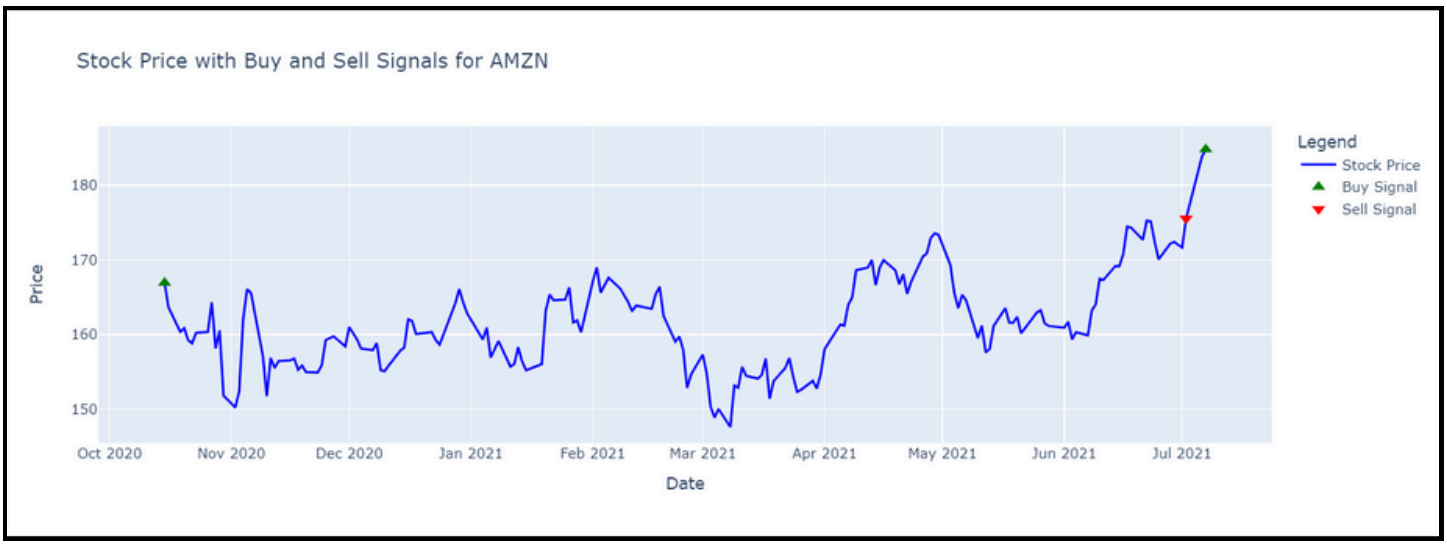
The portfolio performance is evaluated based on total trades, win percentage, total profit, Sharpe ratio, and maximum drawdown.

```python
def final(ticker, rate, intial, df):
    print(f'printing for {ticker} ...')
    # Fetch stock data
    stock_data = fetch_stock_data(ticker)

    # Fetch news data
    news_data = df
    # Calculate sentiment scores
    news_data_with_scores = calculate_sentiment_scores(news_data)
    # Aggregate sentiment scores by date
    sentiment_summary = aggregate_sentiment_scores(news_data_with_scores)
    # Generate trading signals
    trading_signals = generate_trading_signals(sentiment_summary)
    # Simulate trades
    print("\nSimulating trades...")
    portfolio = simulate_trades(stock_data, trading_signals, intial)
    # Calculate portfolio metrics
    total_trades, win_percentage, total_profit = calculate_portfolio_metrics(portfolio)
    # Print the portfolio
    print(portfolio)
    print(f"\nInitial capital: ${intial}")
    print(f"Total Trades: {total_trades}")
    print(f"Win Percentage: {win_percentage:.2f}%")
    print(f"Total Portfolio Returns: ${total_profit:.2f}")
    print(f"Sharpe ratio:{calculate_sharpe_ratio(portfolio, rate):.2f} with risk free rate of {rate}")
    print(f"Max drawdown: {calculate_max_drawdown(portfolio)}")
    # Plot buy and sell signals
    plot_signals(stock_data, portfolio, ticker)
```

## 7.2 Simulations




## 7.3 Machine Learning Algorithms

On GOOGL stocks :

| | Accuracy | Recall | Precision | F1-Score | time to train | time to predict | total time |
|---|---|---|---|---|---|---|---|
| KNN | 75.00% | 75.00% | 82.14% | 70.83% | 0.0 | 0.0 | 0.0 |
| SVC | 62.50% | 62.50% | 39.06% | 48.08% | 0.0 | 0.0 | 0.0 |
| Logistic | 62.50% | 62.50% | 39.06% | 48.08% | 0.0 | 0.0 | 0.0 |
| Decision Tree | 50.00% | 50.00% | 50.00% | 50.00% | 0.0 | 0.0 | 0.0 |
| Extra Trees | 75.00% | 75.00% | 82.14% | 70.83% | 0.2 | 0.0 | 0.2 |
| Random Forest | 50.00% | 50.00% | 50.00% | 50.00% | 0.2 | 0.0 | 0.3 |
| Gradient Boosting Classifier | 62.50% | 62.50% | 39.06% | 48.08% | 0.1 | 0.0 | 0.1 |
| MLP | 62.50% | 62.50% | 39.06% | 48.08% | 0.0 | 0.0 | 0.0 |
| MLP (Keras) | 62.50% | 62.50% | 62.50% | 62.50% | 13.9 | 1.2 | 15.0 |
| GRU (Keras) | 62.50% | 62.50% | 62.50% | 62.50% | 26.3 | 3.5 | 29.8 |
| LSTM (Keras) | 62.50% | 62.50% | 62.50% | 62.50% | 18.9 | 1.7 | 20.6 |

On AMZN stocks :

| | Accuracy | Recall | Precision | F1-Score | time to train | time to predict | total time |
|---|---|---|---|---|---|---|---|
| MultiNB | 50.00% | 50.00% | 25.00% | 33.33% | 0.0 | 0.0 | 0.0 |
| KNN | 37.50% | 37.50% | 36.67% | 36.51% | 0.0 | 0.0 | 0.0 |
| SVC | 50.00% | 50.00% | 25.00% | 33.33% | 0.0 | 0.0 | 0.0 |
| Logistic | 50.00% | 50.00% | 50.00% | 50.00% | 0.0 | 0.0 | 0.0 |
| Decision Tree | 37.50% | 37.50% | 21.43% | 27.27% | 0.0 | 0.0 | 0.0 |
| Extra Trees | 37.50% | 37.50% | 36.67% | 36.51% | 0.2 | 0.0 | 0.2 |
| Random Forest | 37.50% | 37.50% | 21.43% | 27.27% | 0.2 | 0.0 | 0.2 |
| Gradient Boosting Classifier | 37.50% | 37.50% | 21.43% | 27.27% | 0.1 | 0.0 | 0.1 |
| MLP | 50.00% | 50.00% | 25.00% | 33.33% | 0.0 | 0.0 | 0.0 |
| MLP (Keras) | 50.00% | 50.00% | 50.00% | 50.00% | 11.3 | 1.2 | 12.5 |
| GRU (Keras) | 50.00% | 50.00% | 50.00% | 50.00% | 16.8 | 3.1 | 19.8 |
| LSTM (Keras) | 50.00% | 50.00% | 50.00% | 50.00% | 28.6 | 4.3 | 32.9 |

# 8. Conclusion

## 8.1 Summary

This project demonstrates the application of sentiment analysis to stock trading. By analyzing news headlines and generating sentiment scores, we can create trading signals that help in making informed trading decisions. The performance evaluation metrics provide insights into the effectiveness of the strategy.

## 8.2 Future Work

Future work can include:

- Enhancing the sentiment analysis model by incorporating more advanced NLP techniques.
- Expanding the dataset to include more news sources and social media mentions.
- Implementing a more sophisticated trading strategy with additional features such as stop-loss and take-profit levels.