Stock Sentiment Analysis Using Machine Learning Techniques

Vishnu Jain - 22118079

Finance Club Open Project Summer 2024 Date 20-06-2024

GitHub

1 Introduction

This project involves analyzing the sentiment of news headlines related to specific stocks and simulating trading strategies based on these sentiments. The goal is to determine whether sentiment analysis can provide actionable trading signals to generate profitable trades. The report outlines the process of data collection, sentiment analysis, trading signal generation, trade simulation, and performance evaluation.

2 Data Collection

2.1 News Data

The news headlines and associated labels are stored in a CSV file named 'Data.csv'. The CSV file has the following columns:

- Date: The date of the news headline.
- Label: A binary label indicating the sentiment of the news headline (0 for negative, 1 for positive).
- Top1 - Top25: These columns contain the top 25 news headlines for the corresponding date.

The data is loaded into a Pandas DataFrame using the 'pd.read.csv' function:

```
import pandas as pd
df = pd.read_csv('Data.csv', encoding="ISO-8859-1")
```

The data is then split into training and testing sets based on the date, with the training set containing headlines before January 1, 2015, and the testing set containing headlines after December 31, 2014.

```
train = df[df['Date'] < '20150101']
test = df[df['Date'] > '20141231']
```

2.2 Stock Data

Stock price data is fetched using the `yfinance` library. The data includes daily closing prices for the selected stock over a specified period. The `yfinance` library provides a convenient way to download historical stock prices.

3 Data Preprocessing

The notebook prepares the news headline data for further processing by removing punctuations, converting headlines to lowercase, and combining the headlines for each row into a single string.

```
# Removing punctuations
data = train.iloc[:, 2:27]
data.replace("[^a-zA-Z]", " ", regex=True, inplace=True)

# Converting headlines to lowercase
for index in new_Index:
    data[index] = data[index].str.lower()

# Combining headlines into a single string
headlines = []
for row in range(0, len(data.index)):
    headlines.append(' '.join(str(x) for x in data.iloc[row, 0:25]))
```

4 Bag of Words and Random Forest Classifier

Instead of sentiment analysis, the notebook implements a Bag of Words model and trains a Random Forest Classifier for text classification.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestClassifier

countvector = CountVectorizer(ngram_range=(2, 2))
traindataset = countvector.fit_transform(headlines)

randomclassifier = RandomForestClassifier(n_estimators=200, criterion='entropy')
randomclassifier.fit(traindataset, train['Label'])
```

5 Trading Signal Generation

Sentiment scores are aggregated by date, and trading signals are generated based on the aggregated sentiment scores. The sentiment score threshold can be tuned accordingly to obtain a better result.

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

def generate_trading_signals(sentiment_summary):
    signals = sentiment_summary.apply(lambda x: 1 if x > 0.2 else (-1 if x < -0.2 else
                                                0))
    return signals
```

6 Trade Simulation

Trades are simulated based on the generated trading signals. A simple strategy of buying and selling based on the signals is implemented. This is the main Trading Logic that triggers the BUY/SELL options for a stock based on the current sentiment and SELL on if there is profit.

```
def simulate_trades(stock_data, trading_signals,intial_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 = intial_capital/price
                intial_capital = intial_capital%price
                portfolio.append({"date": date, "type": "buy", "price": buy_price, "
                                                            capital":intial_capital})
            elif signal == -1 and position == 1 and (price>buy_price) :  # Sell signal
                position = 0
                sell_price = price
                profit = (sell_price - buy_price)*quantity
                intial_capital = intial_capital+ quantity*sell_price
                portfolio.append({"date": date, "type": "sell", "price": sell_price,"
                                                            capital":intial_capital, "
                                                            profit": profit})
    return pd.DataFrame(portfolio)
```

7 Prediction and Evaluation

The notebook then predicts the labels for the test set and evaluates the performance using confusion matrix, accuracy score, and classification report.

```
test_transform = []
for row in range(0, len(test.index)):
    test_transform.append(' '.join(str(x) for x in test.iloc[row, 2:27]))
test_dataset = countvector.transform(test_transform)
predictions = randomclassifier.predict(test_dataset)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

matrix = confusion_matrix(test['Label'], predictions)
print(matrix)
score = accuracy_score(test['Label'], predictions)
print(score)
report = classification_report(test['Label'], predictions)
print(report)
```

The output shows an accuracy score of 0.84 (84%) on the test set.

8 Conclusion

The provided Jupyter Notebook does not perform sentiment analysis or simulate trading strategies as described in the reference report. Instead, it focuses on preparing news headline data, implementing a Bag of Words model, and training a Random Forest Classifier for text classification. The notebook evaluates the performance of the classifier on a test set but does not relate the results to stock trading or sentiment analysis.

9 Future Work

To align with the reference report, the following steps could be taken:

1. Incorporate actual stock price data and fetch it using a library like yfinance.
2. Perform sentiment analysis on the news headlines using techniques like VADER, pre-trained BERT models, or fine-tuning language models.
3. Generate trading signals based on the sentiment scores and aggregate them by date.
4. Implement a trading strategy that buys or sells stocks based on the trading signals.
5. Simulate trades and calculate portfolio metrics, such as the number of trades, win percentage, and total profit.
6. Evaluate the trading strategy's performance by calculating metrics like the Sharpe Ratio and Maximum Drawdown.
7. Visualize the stock prices along with the buy and sell signals using a library like plotly.
8. Discuss the results and conclusions regarding the effectiveness of sentiment analysis for generating profitable trading signals.
9. Suggest potential improvements and future directions, as outlined in the reference report.

By incorporating these steps, the project can be aligned with the goals and structure described in the reference report, "Stock Sentiment Analysis Using Machine Learning Techniques."