Text Mining Assignment 2022 - Sentiment Analysis on Business News

With this project, we try to understand if the stock market is influenced by the news and if so, by what magnitude.

To do so, we used relevant stocks news headlines for the Tesla stock, on which we apply sentiment analysis, using Vader. Then, we compare the results with the TSLA stock price.

1. Import all necessary libraries as well as the dictionary used

In [1]:
```
import pandas as pd
import re
from datetime import datetime, timedelta

###pip install yfinance
import yfinance as yf

import matplotlib.pyplot as plt
import seaborn as sns

import nltk
nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/nikolaosroumpos/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

2. News Headlines

2.1 Import Data

In [ ]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files no files selected Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

2.2 Construct and Clean DataFrame

In [3]:
```
news_data = pd.read_csv("/Users/nikolaosroumpos/Desktop/teslahead.csv", encoding='windows-1250', header=None)
news_data.head()
```

Out[3]:

	0	1
0	Tesla: Nothing Matters, Until Everything Matters	Montana Skeptic•Mon, Dec. 9•600 Comments
1	Tesla: The Hype Does Not Reflect Reality	The European View•Fri, Dec. 6•742 Comments
2	Tesla Will Likely Achieve Record Deliveries In...	iBEV•Wed, Dec. 4•340 Comments
3	Tesla: Automatic Labeling For Computer Vision	Trent Eady•Tue, Dec. 3•240 Comments
4	Tesla: Ready To Re-Enter The Fast Lane	DoctoRx•Tue, Dec. 3•229 Comments

In [4]:
```
news_data.columns = ['Headlines', 'Metadata'] # column names
news_data.head()
```

Out[4]:

	Headlines	Metadata
0	Tesla: Nothing Matters, Until Everything Matters	Montana Skeptic•Mon, Dec. 9•600 Comments
1	Tesla: The Hype Does Not Reflect Reality	The European View•Fri, Dec. 6•742 Comments
2	Tesla Will Likely Achieve Record Deliveries In...	iBEV•Wed, Dec. 4•340 Comments
3	Tesla: Automatic Labeling For Computer Vision	Trent Eady•Tue, Dec. 3•240 Comments
4	Tesla: Ready To Re-Enter The Fast Lane	DoctoRx•Tue, Dec. 3•229 Comments

2.3 Converting Metadata Strings into a Usable Datetime Format

2.3.1 Extracting the Date

The metadata contains the date of posting in the string, but has to be extracted using regular expressions. The format of the date in the metadata is slightly irregular, but can be easily extracted.

In [5]:
```
print(news_data['Metadata'][1000]) # original metadata
match = re.search(r'\w{3}\.\s\d{1,2}\,\s\d{4}', news_data['Metadata'][1000])
print(match[0]) # extracted datastring
```

```
Donn Bailey•Mar. 27, 2018•128 Comments
Mar. 27, 2018
```

There are four possible formats for the dates in the metadata. The days are displayed as a single or double digit and the months as a three letter abbreviation with a dot behind them, except for the dates in the month of May, they do not have a dot behind them. The 2019 dates do not contain the year in the string, the 2018 ones do.

This that we have to add the year 2019 to the string to bring them all into the same format.

In [6]:
```
print(news_data['Metadata'][200]) # display original (without year)
match = re.search(r'\w{3}\s\d{1,2}', news_data['Metadata'][200])
modifiedDate = match[0] + ", 2019"
print(modifiedDate) # display modified

print(news_data['Metadata'][300]) # display original (without year)
match = re.search(r'\w{3}\.\s\d{1,2}', news_data['Metadata'][300])
modifiedDate = match[0] + ", 2019"
print(modifiedDate) # display modified

print(news_data['Metadata'][850]) # display original (with year)
match = re.search(r'\w{3}\s\d{1,2}\,\s\d{4}', news_data['Metadata'][850])
print(match[0]) # display modified

print(news_data['Metadata'][1000]) # display original (with year)
match = re.search(r'\w{3}\.\s\d{1,2}\,\s\d{4}', news_data['Metadata'][1000])
print(match[0]) # display modified
```

```
Acolyte of Value•Mon, May 13•351 Comments
May 13, 2019
Bill Maurer•Mon, Mar. 4•599 Comments
Mar. 4, 2019
Quoth the Raven•May 17, 2018•123 Comments
May 17, 2018
Donn Bailey•Mar. 27, 2018•128 Comments
Mar. 27, 2018
```

2.3.2 Converting into Useable Datetime Format

Since the May months do not have a dot after their abbreviation, we have two possible datetime formats. To label all headlines with a uniform datetime string, we need to account for this.

In [7]:
```
new_date_1 = datetime.strptime('Dec. 6, 2019', '%b. %d, %Y').date()
print(new_date_1)

new_date_2 = datetime.strptime('May 17, 2018', '%b %d, %Y').date()
print(new_date_2)
```

```
2019-12-06
2018-05-17
```

2.3.3 Combining Everything

In [8]:
```
new_dates = [] # create a list to store the cleaned dates

for article_metadata in news_data['Metadata']: # loop every row in the metadata
    match = re.search(r'\w{3}\.\s\d{1,2}\,\s\d{4}|May\s\d{1,2}\,\s\d{4}|\w{3}\.\s\d{1,2}|May\s\d{1,2}',
                      article_metadata) # match e.g. Nov. 3, 2018 or May 03, 2018 or Jan. 02 or May 03 (May apparently does not get a dot behind it)

    if re.search(r'\w{3}\.\s\d{1,2}\,\s\d{4}|\w{3}\s\d{1,2}\,\s\d{4}',match[0]):
        complete_date = match[0] # don't append year to string, already included
    else:
        complete_date = match[0] + ", 2019" # append year to string, year not included yet

    for date_format in ('%b. %d, %Y', '%b %d, %Y'): # In mei leggen alle vogels een ei, behalve de koekoek en de griet, want die leggen in de meimaand
niet
        try:
            new_date = datetime.strptime(complete_date, date_format).date()
            break # stop when the format is fixed
        except ValueError:
            continue

    new_dates.append(new_date) # add new date to the list

news_data['Date'] = new_dates # make a column with the formatted dates

news_data.head()
```

Out[8]:

	Headlines	Metadata	Date
0	Tesla: Nothing Matters, Until Everything Matters	Montana Skeptic•Mon, Dec. 9•600 Comments	2019-12-09
1	Tesla: The Hype Does Not Reflect Reality	The European View•Fri, Dec. 6•742 Comments	2019-12-06
2	Tesla Will Likely Achieve Record Deliveries In...	iBEV•Wed, Dec. 4•340 Comments	2019-12-04
3	Tesla: Automatic Labeling For Computer Vision	Trent Eady•Tue, Dec. 3•240 Comments	2019-12-03
4	Tesla: Ready To Re-Enter The Fast Lane	DoctoRx•Tue, Dec. 3•229 Comments	2019-12-03

2.4 Getting the Sentiment Score for the Headlines

A sentiment score is calculated for every headline. A probability for negative, neutral or positive sentiment is given together with a compound score.

In [9]:
```
results = []

for headline in news_data['Headlines']:
    polarity_score = SIA().polarity_scores(headline) # run analysis
    polarity_score['headline'] = headline # add the headline to compare with the result
    results.append(polarity_score)

results[:3]
```

Out[9]:
```
[{'neg': 0.349,
  'neu': 0.651,
  'pos': 0.0,
  'compound': -0.0382,
  'headline': 'Tesla: Nothing Matters, Until Everything Matters'},
 {'neg': 0.0,
  'neu': 1.0,
  'pos': 0.0,
  'compound': 0.0,
  'headline': 'Tesla: The Hype Does Not Reflect Reality'},
 {'neg': 0.0,
  'neu': 1.0,
  'pos': 0.0,
  'compound': 0.0,
  'headline': 'Tesla Will Likely Achieve Record Deliveries In Q4 Despite Headwinds'}]
```

In [10]:
```
news_data['Score'] = pd.DataFrame(results)['compound'] # add the sentiment score to the dataframe of the headlines
news_data.head()
```

Out[10]:

	Headlines	Metadata	Date	Score
0	Tesla: Nothing Matters, Until Everything Matters	Montana Skeptic•Mon, Dec. 9•600 Comments	2019-12-09	-0.0382
1	Tesla: The Hype Does Not Reflect Reality	The European View•Fri, Dec. 6•742 Comments	2019-12-06	0.0000
2	Tesla Will Likely Achieve Record Deliveries In...	iBEV•Wed, Dec. 4•340 Comments	2019-12-04	0.0000
3	Tesla: Automatic Labeling For Computer Vision	Trent Eady•Tue, Dec. 3•240 Comments	2019-12-03	0.2500
4	Tesla: Ready To Re-Enter The Fast Lane	DoctoRx•Tue, Dec. 3•229 Comments	2019-12-03	0.3612

2.5 Create Daily Sentiment Score

The sentiment scores of all the news headlines for an individual day are summed up to create a daily score.

In [11]:
```
stock_series = news_data.groupby(['Date']).sum()
stock_series.head()
```

Out[11]:

Date	Score
2018-01-11	0.0000
2018-01-15	0.5719
2018-01-16	-0.1027
2018-01-17	-0.1280
2018-01-18	0.0000

3. Stock Price and Return

3.1 Importing Stock Price Data

In [12]:
```
stock = yf.Ticker("TSLA")
stock_prices = stock.history(start=stock_series.index[0] - timedelta(1), end=stock_series.index[-1] + timedelta(1))
stock_prices["Close"] = stock_prices["Close"].interpolate("time") # interpolate over time in case of missing values
stock_prices.head()
```

Out[12]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2018-01-09	67.031998	67.760002	65.480003	66.737999	35733000	0	0
2018-01-10	66.440002	67.400002	66.000000	66.959999	21549500	0	0
2018-01-11	67.047997	68.961998	66.652000	67.589996	33227500	0	0
2018-01-12	67.725998	68.082001	66.734001	67.244003	24125500	0	0
2018-01-16	67.508003	69.000000	66.959999	68.012001	32371500	0	0

3.2 Calculating Returns

In [13]:
```
returns = stock_prices['Close'] / stock_prices['Close'].shift(1) - 1 # calculate daily returns
returns = returns.rename("Return")

#merge on right so you keep the trading dates and not the news dates as index
stock_series = stock_series.merge(returns, left_index=True, right_index=True, how='right')
stock_series.head()
```

Out[13]:

Date	Score	Return
2018-01-09	NaN	NaN
2018-01-10	NaN	0.003326
2018-01-11	0.0000	0.009409
2018-01-12	NaN	-0.005119
2018-01-16	-0.1027	0.011421

3.3 Delayed Sentiment Score

The news headlines will have a delayed effect on the stock price, so we shift the sentiment score one day in the future, when it will have an effect.

In [14]:
```
stock_series["Shifted Score"] = stock_series["Score"].shift(1)
stock_series.head()
```

Out[14]:

Date	Score	Return	Shifted Score
2018-01-09	NaN	NaN	NaN
2018-01-10	NaN	0.003326	NaN
2018-01-11	0.0000	0.009409	NaN
2018-01-12	NaN	-0.005119	0.0
2018-01-16	-0.1027	0.011421	NaN

4. Analysis

4.1 Scatter Plots of Shifted Score

4.1.1 Calculate Variables

In [16]:
```
shifted_return = stock_series.drop("Score", axis=1) # drop the score column
shifted_return = shifted_return.dropna() # drop useless rows with NaN values
shifted_return = shifted_return[(shifted_return['Shifted Score'] > 0.5) | (shifted_return['Shifted Score'] < -0.5)] # only incorporate the heavy sentiment
values
shifted_return.head()
```

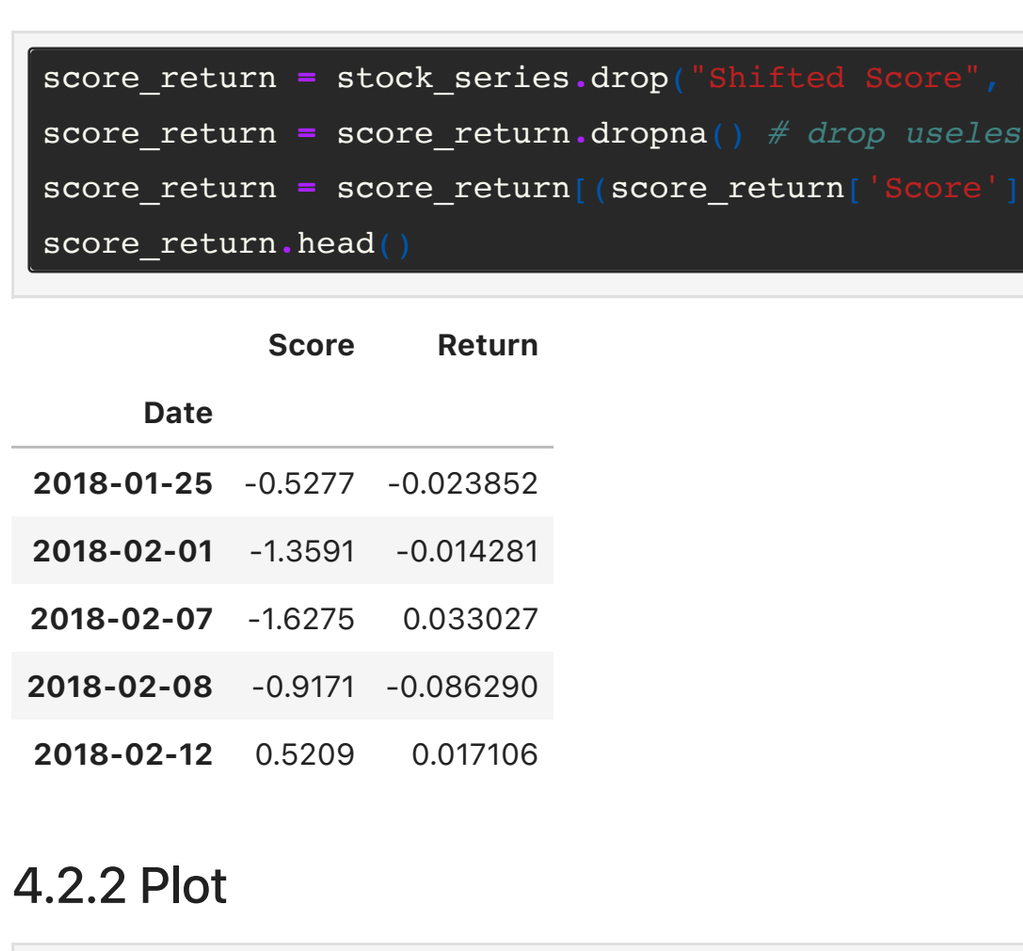
Out[16]:

Date	Return	Shifted Score
2018-01-26	0.015431	-0.5277
2018-02-02	-0.015748	-1.3591
2018-02-08	-0.086290	-1.6275
2018-02-09	-0.015259	-0.9171
2018-02-13	0.025116	0.5209

4.1.2 Plot

In [17]:
```
shifted_return.plot(kind="scatter", x="Shifted Score", y="Return", style="o")
```

Out[17]:
```
<AxesSubplot:xlabel='Shifted Score', ylabel='Return'>
```



4.2 Scatter Plot of Score

4.2.1 Calculate Variables

In [18]:
```
score_return = stock_series.drop("Shifted Score", axis=1) # drop the score column
score_return = score_return.dropna() # drop useless rows with NaN values
score_return = score_return[(score_return['Score'] > 0.5) | (score_return['Score'] < -0.5)] # only incorporate the heavy sentiment values
score_return.head()
```

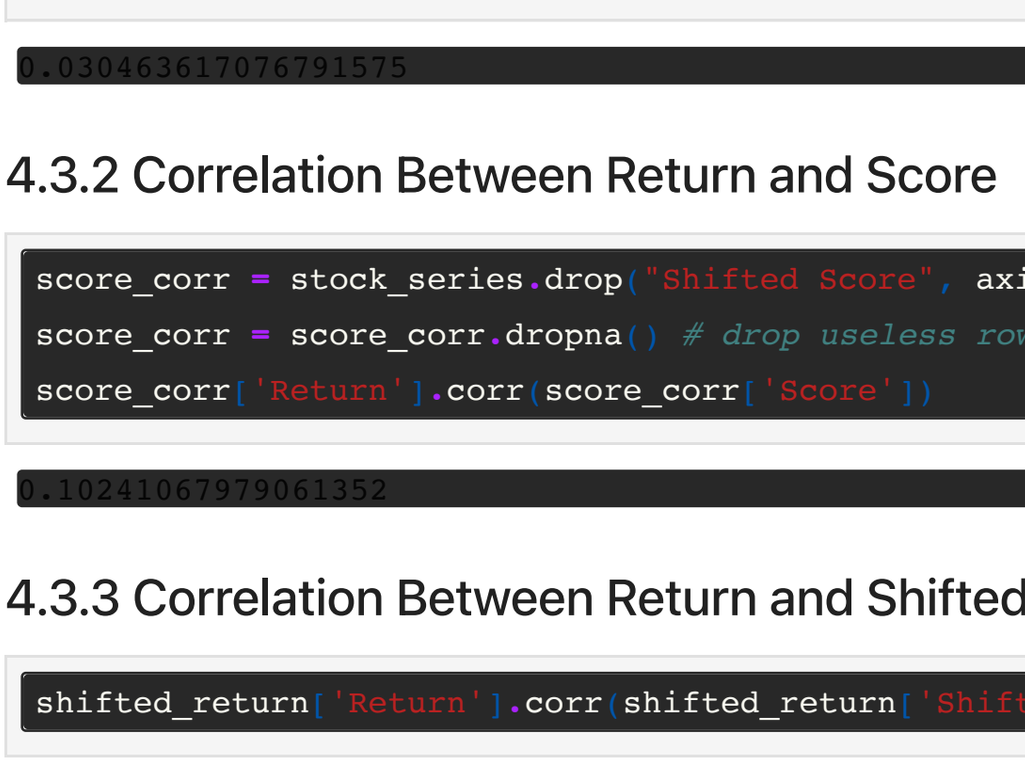
Out[18]:

Date	Score	Return
2018-01-25	-0.5277	-0.023852
2018-02-01	-1.3591	-0.014281
2018-02-07	-1.6275	0.033027
2018-02-08	-0.9171	-0.086290
2018-02-12	0.5209	0.017106

4.2.2 Plot

In [19]:
```
score_return.plot(kind="scatter", x="Score", y="Return", style="o")
```

Out[19]:
```
<AxesSubplot:xlabel='Score', ylabel='Return'>
```



4.3 Correlation

4.3.1 Correlation Between Return and Shifted Score

In [20]:
```
shifted_corr = stock_series.drop("Score", axis=1) # drop the score column
shifted_corr = shifted_corr.dropna() # drop useless rows with NaN values
shifted_corr['Return'].corr(shifted_corr['Shifted Score'])
```

Out[20]:
```
0.030463617076791575
```

4.3.2 Correlation Between Return and Score

In [21]:
```
score_corr = stock_series.drop("Shifted Score", axis=1) # drop the score column
score_corr = score_corr.dropna() # drop useless rows with NaN values
score_corr['Return'].corr(score_corr['Score'])
```

Out[21]:
```
0.10241067979061352
```

4.3.3 Correlation Between Return and Shifted Score For Large Scores

In [22]:
```
shifted_return['Return'].corr(shifted_return['Shifted Score'])
```

Out[22]:
```
0.07193395314934745
```

4.3.4 Correlation Between Return and Score For Large Scores

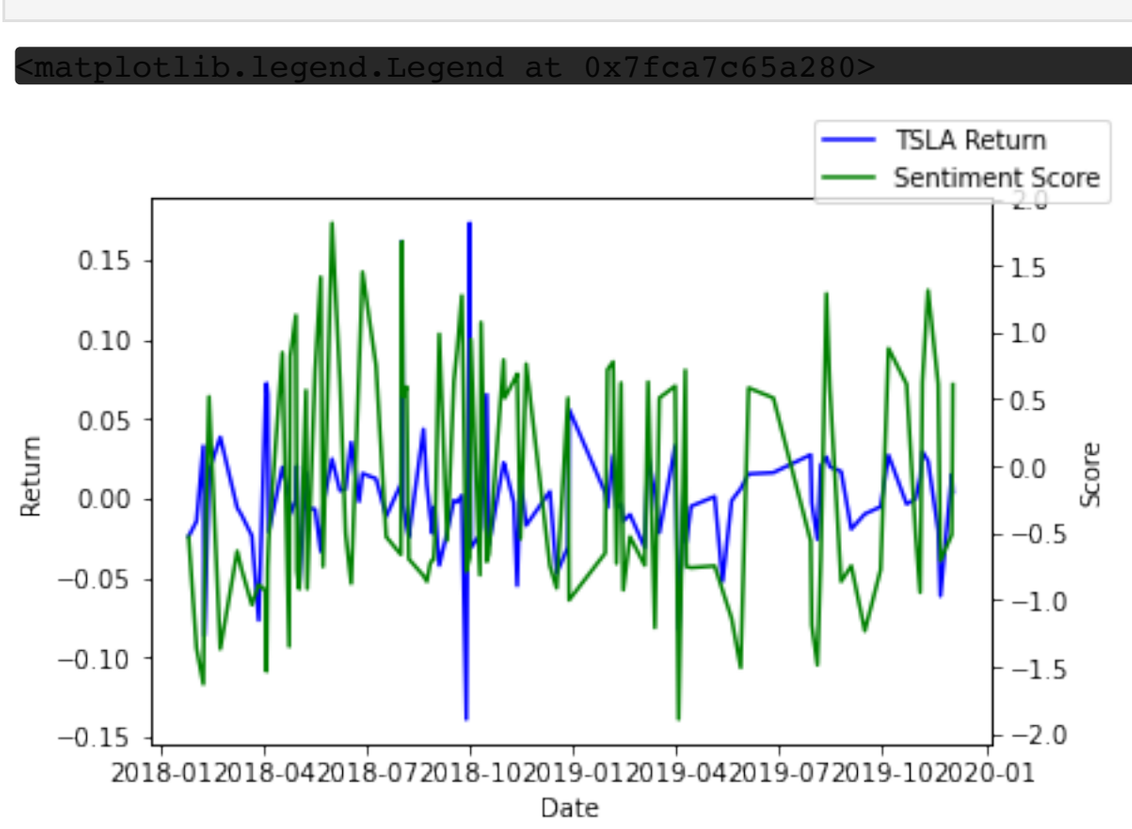
In [23]:
```
score_return['Return'].corr(score_return['Score'])
```

Out[23]:
```
0.15060219912549347
```

4.4 Return and Shifted Sentiment Plot

In [24]:
```
ax=sns.lineplot(data=shifted_return['Return'], color="b", legend=False, label='TSLA Return')
ax2 = plt.twinx()
sns.lineplot(data=shifted_return["Shifted Score"], color="g", ax=ax2, legend=False, label='Shifted Sentiment Score')
ax2.figure.legend()
```

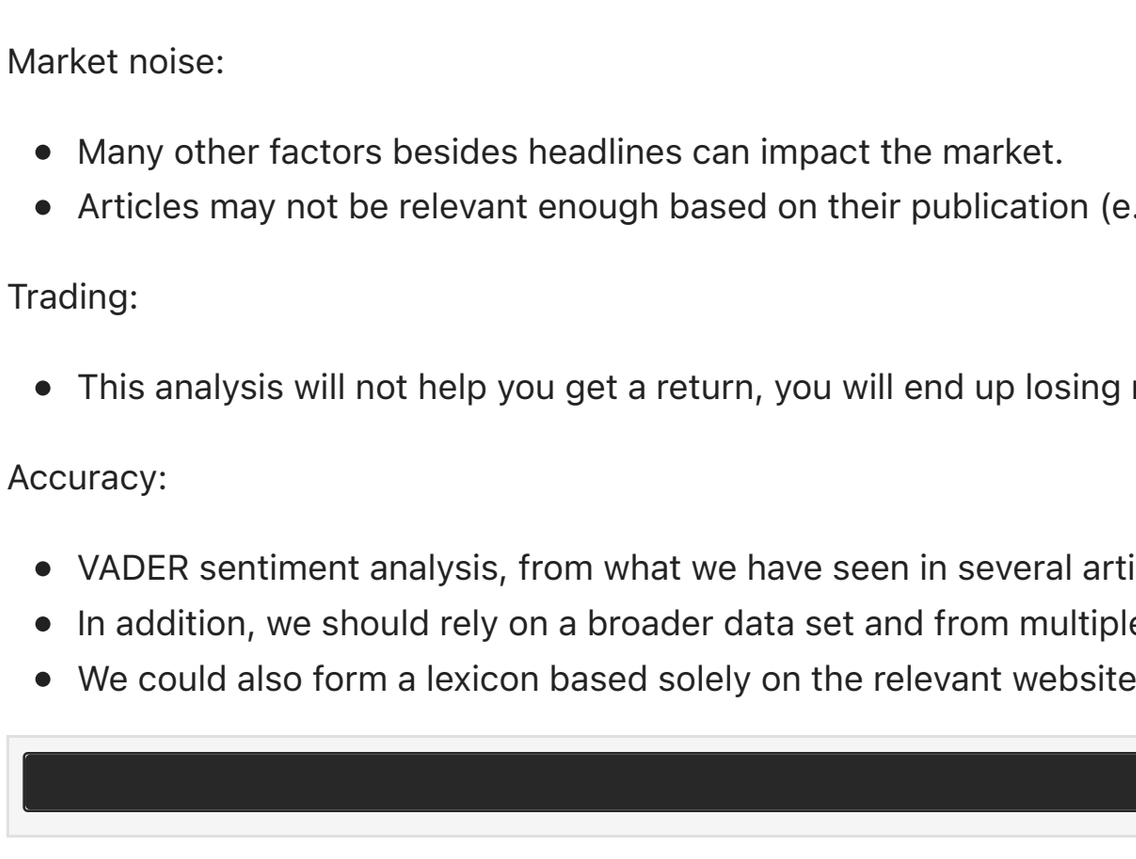
Out[24]:
```
<matplotlib.legend.Legend at 0x7fca58600160>
```



4.5 Return and Sentiment Plot

In [25]:
```
ax=sns.lineplot(data=score_return['Return'], color="b", legend=False, label='TSLA Return')
ax2 = plt.twinx()
sns.lineplot(data=score_return["Score"], color="g", ax=ax2, legend=False, label='Sentiment Score')
ax2.figure.legend()
```

Out[25]:
```
<matplotlib.legend.Legend at 0x7fca7c65a280>
```



5. Discussion

Time:

- A new headline can have a greater impact than the older one, so we should decrease the weight of older headlines.
- Incorporate the time the news was posted (after or before closing).

Market noise:

- Many other factors besides headlines can impact the market.
- Articles may not be relevant enough based on their publication (e.g., an article that is not related to earnings week may have no impact).

Trading:

- This analysis will not help you get a return, you will end up losing money by high frequency traders and institutional investors.

Accuracy:

- VADER sentiment analysis, from what we have seen in several articles, is not accurate enough.
- In addition, we should rely on a broader data set and from multiple sources, including social media.
- We could also form a lexicon based solely on the relevant website we used for our headlines.

In [ ]: