DEDA Digital Economy & Decision Analytics

Sentiment Analysis of CDC COVID-19 Updates and their Market Impact

Ruben Bosch

Fudan University

DDM China and the World Economy
Rijksuniversiteit Groningen
MSc Econometrics, Operations Research & Actuarial Studies
https://www.linkedin.com/in/r-bosch/

Outline

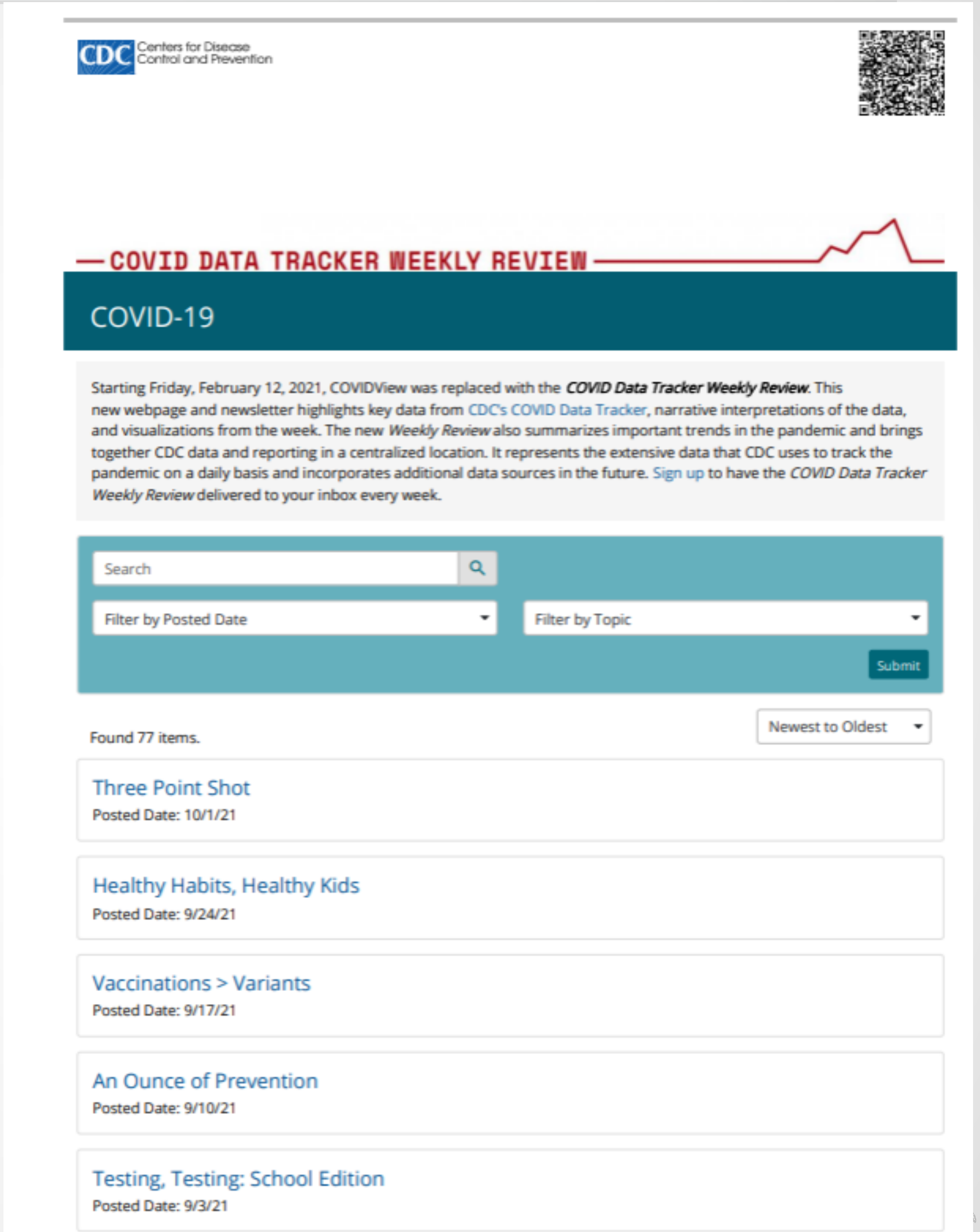
1. Motivation
2. Scraping of CDC
3. Preprocessing the Data
4. Calculation of Sentiment
5. Scaling the Sentiment
6. Sentiment and Nasdaq Returns
7. Conclusions

Motivation

- Stock markets highly influenced by the COVID pandemic (McKinsey, 2021)
- Volatile periods following the news closely
- CDC followed situation closely
- Weekly updates
- Influence of CDC's sentiment on stock markets
- Can CDC updates predict market movements?

Scraping of CDC

- Scrape the CDC news page
- Dive into HTML code to find appropriate links
- Search widget makes it hard to retrieve links
- Exploit URLs



Scraping of CDC

- Example: https://www.cdc.gov/coronavirus/2019-ncov/covid-data/covidview/past-reports/10012021.html

- Format: /MMDDYYYY.html

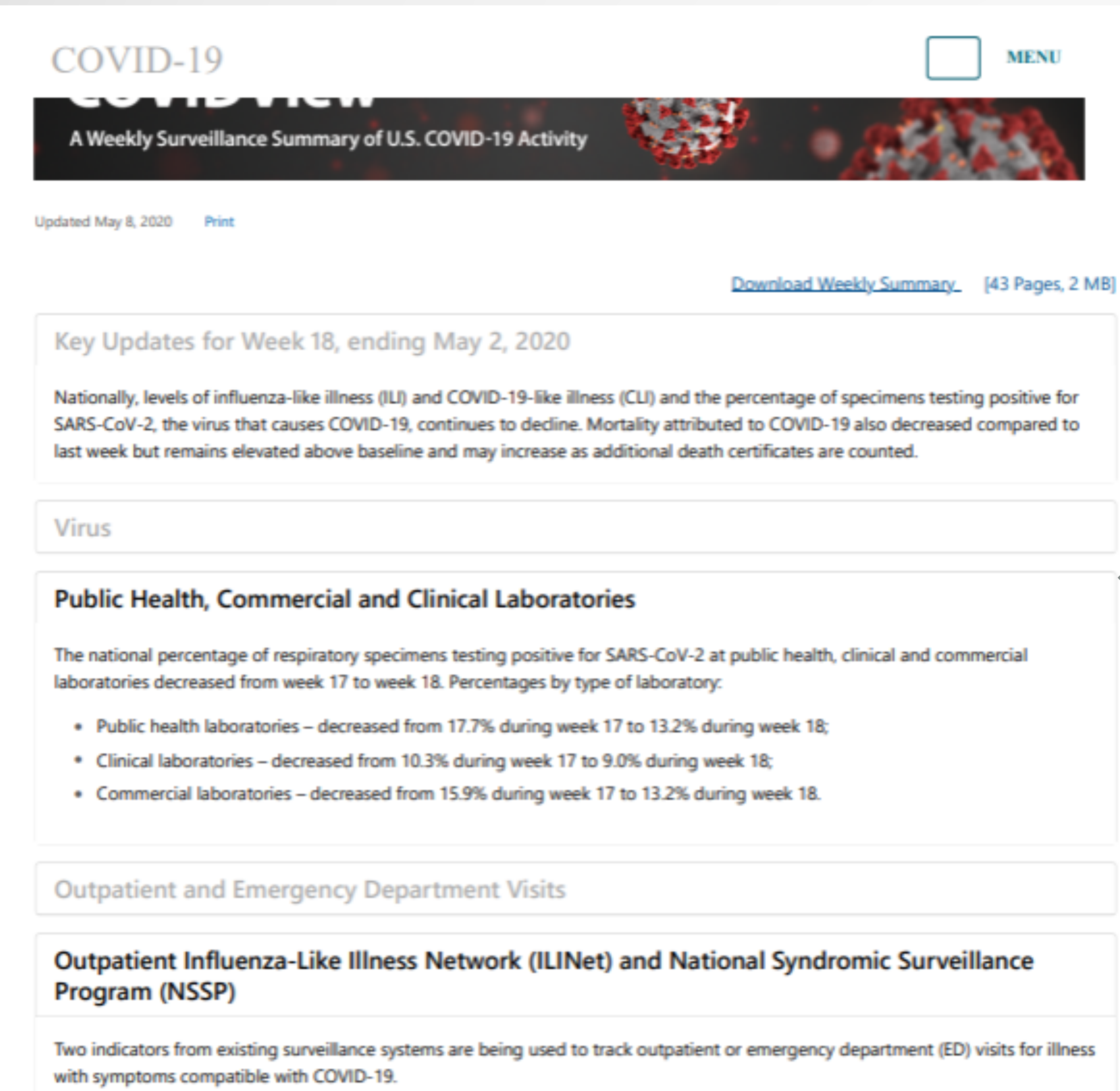
```
#Make the links
dates = [dt.datetime(2020, 4, 10, 0, 0)]
date_list = []
links = []
time_add = dt.timedelta(days=7)
for i in range(1,78):
    new_date = dates[i-1]+time_add
    dates.append(new_date)
    date_list.append(dates[i-1].strftime("%m%d%Y"))
    links.append("https://www.cdc.gov/coronavirus/2019-ncov/covid-data/covidview/past-reports/"+date_list[i-1]+".html")
```

Scraping of CDC

```
#Manual correction for Thanksgiving, Christmas and New Years Eve:
date_list[33] = '11302020'
links[33] = "https://www.cdc.gov/coronavirus/2019-ncov/covid-data/covidview/past-reports/"+date_list[33]+".html"
date_list[37] = '12282020'
links[37] = "https://www.cdc.gov/coronavirus/2019-ncov/covid-data/covidview/past-reports/"+date_list[37]+".html"
date_list[38] = '01042021'
links[38] = "https://www.cdc.gov/coronavirus/2019-ncov/covid-data/covidview/past-reports/"+date_list[38]+".html"

#And a rather unfortunate mistake by the CDC, where they misspelled 2021 into 2121
date_list[53] = '04162121'
links[53] = "https://www.cdc.gov/coronavirus/2019-ncov/covid-data/covidview/past-reports/"+date_list[53]+".html"

#And a day off
date_list.remove('06182021')
links.remove("https://www.cdc.gov/coronavirus/2019-ncov/covid-data/covidview/past-reports/06182021.html")

#Scrape the articles
df_old = [get_info_old(url) for url in links[:44]]
df_new =  [get_info_new(url) for url in links[44:] ]
df = df_old+df_new
news_df = pd.DataFrame(df) #Final dataframe with the articles and dates
```

Scraping of CDC

- Scrape the websites: transition of layout


COVID-19 MENU

A Weekly Surveillance Summary of U.S. COVID-19 Activity

Updated May 8, 2020 Print

Download Weekly Summary [43 Pages, 2 MB]

Key Updates for Week 18, ending May 2, 2020

Nationally, levels of influenza-like illness (ILI) and COVID-19-like illness (CLI) and the percentage of specimens testing positive for SARS-CoV-2, the virus that causes COVID-19, continues to decline. Mortality attributed to COVID-19 also decreased compared to last week but remains elevated above baseline and may increase as additional death certificates are counted.

Virus

Public Health, Commercial and Clinical Laboratories

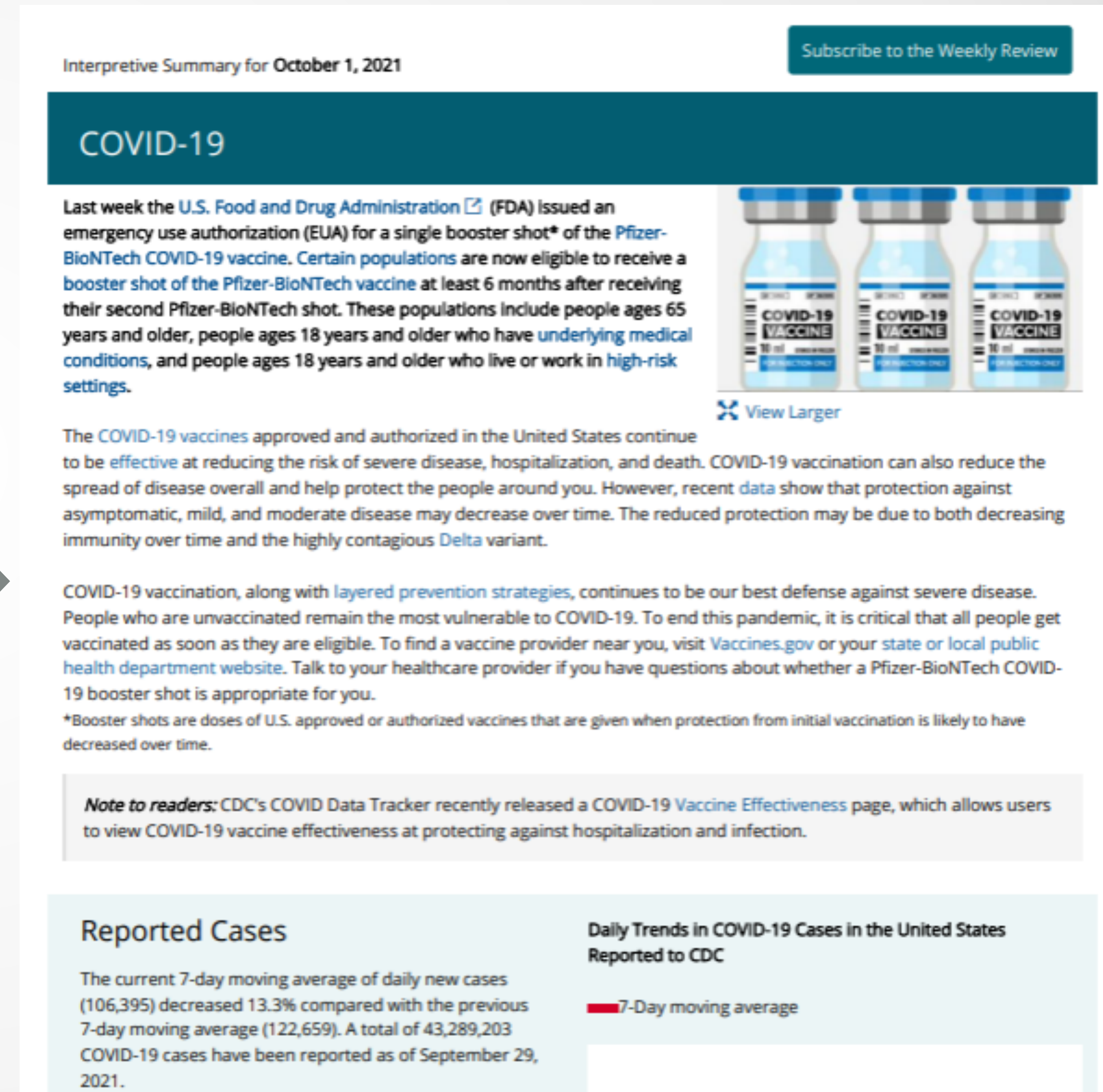
The national percentage of respiratory specimens testing positive for SARS-CoV-2 at public health, clinical and commercial laboratories decreased from week 17 to week 18. Percentages by type of laboratory:

- Public health laboratories – decreased from 17.7% during week 17 to 13.2% during week 18;
- Clinical laboratories – decreased from 10.3% during week 17 to 9.0% during week 18;
- Commercial laboratories – decreased from 15.9% during week 17 to 13.2% during week 18.

Outpatient and Emergency Department Visits

Outpatient Influenza-Like Illness Network (ILINet) and National Syndromic Surveillance Program (NSSP)

Two indicators from existing surveillance systems are being used to track outpatient or emergency department (ED) visits for illness with symptoms compatible with COVID-19.


Interpretive Summary for October 1, 2021

Subscribe to the Weekly Review

COVID-19

Last week the U.S. Food and Drug Administration (FDA) issued an emergency use authorization (EUA) for a single booster shot* of the Pfizer-BioNTech COVID-19 vaccine. Certain populations are now eligible to receive a booster shot of the Pfizer-BioNTech vaccine at least 6 months after receiving their second Pfizer-BioNTech shot. These populations include people ages 65 years and older, people ages 18 years and older who have underlying medical conditions, and people ages 18 years and older who live or work in high-risk settings.

View Larger

The COVID-19 vaccines approved and authorized in the United States continue to be effective at reducing the risk of severe disease, hospitalization, and death. COVID-19 vaccination can also reduce the spread of disease overall and help protect the people around you. However, recent data show that protection against asymptomatic, mild, and moderate disease may decrease over time. The reduced protection may be due to both decreasing immunity over time and the highly contagious Delta variant.

COVID-19 vaccination, along with layered prevention strategies, continues to be our best defense against severe disease. People who are unvaccinated remain the most vulnerable to COVID-19. To end this pandemic, it is critical that all people get vaccinated as soon as they are eligible. To find a vaccine provider near you, visit Vaccines.gov or your state or local public health department website. Talk to your healthcare provider if you have questions about whether a Pfizer-BioNTech COVID-19 booster shot is appropriate for you.

*Booster shots are doses of U.S. approved or authorized vaccines that are given when protection from initial vaccination is likely to have decreased over time.

Note to readers: CDC's COVID Data Tracker recently released a COVID-19 Vaccine Effectiveness page, which allows users to view COVID-19 vaccine effectiveness at protecting against hospitalization and infection.

Reported Cases

The current 7-day moving average of daily new cases (106,395) decreased 13.3% compared with the previous 7-day moving average (122,659). A total of 43,289,203 COVID-19 cases have been reported as of September 29, 2021.

Daily Trends in COVID-19 Cases in the United States Reported to CDC

7-Day moving average

Scraping of CDC

```
def get_info_old(url):
    #send request
    response = requests.get(url)
    #parse
    soup = BeautifulSoup(response.text)
    #get information we need
    news = soup.find('div', attrs={'class': 'card-body bg-white'}).text

    parse_result = parse.urlparse(url)
    parse_splitted = parse_result.path.split("/")
    date_raw = parse_splitted[len(parse_splitted)-1]
    date = datetime.strptime(date_raw.split(".")[0], "%m%d%Y")

    columns = [news,date]
    column_names = ['News','Date']
    return dict(zip(column_names, columns))

def get_info_new(url):
    #send request
    response = requests.get(url)
    #parse
    soup = BeautifulSoup(response.text)
    #get information we need
    news = soup.find('div', attrs={'class': 'row mb-3 bg-white'}).text

    parse_result = parse.urlparse(url)
    parse_splitted = parse_result.path.split("/")
    date_raw = parse_splitted[len(parse_splitted)-1]
    date = datetime.strptime(date_raw.split(".")[0], "%m%d%Y")
    columns = [news,date]
    column_names = ['News','Date']
    return dict(zip(column_names, columns))
```

Preprocessing the Data

- Stopwords from various sources: University of Notre Dame, NLTK module

```
#######  preprocessing the data  #######
nlp = spacy.load("en_core_web_sm", disable=['parser', 'ner'])

#import other lists of stopwords
with open('StopWords_GenericLong.txt', 'r') as f:
 x_gl = f.readlines()
with open('StopWords_DatesandNumbers.txt', 'r') as f:
 x_d = f.readlines()
#import nltk stopwords
stopwords = nltk.corpus.stopwords.words('english')
#combine all stopwords
[stopwords.append(x.rstrip()) for x in x_gl]
[stopwords.append(x.rstrip()) for x in x_d]
#change all stopwords into lowercase
stopwords_lower = [s.lower() for s in stopwords]
```

Preprocessing the Data

- Preprocessing function to get rid of stopwords, websites, mail addresses, punctuation

```
def text_preprocessing(str_input):
    #tokenization, remove punctuation, lemmatization
    words=[token.lemma_ for token in nlp(str_input) if not token.is_punct]

    # remove symbols, websites, email addresses
    words = [re.sub(r"[^A-Za-z@]", "", word) for word in words]
    words = [re.sub(r"\S+com", "", word) for word in words]
    words = [re.sub(r"\S+@\S+", "", word) for word in words]
    words = [word for word in words if word!=' ']
    words = [word for word in words if len(word)!=0]

    #remove stopwords
    words=[word.lower() for word in words if word.lower() not in stopwords_lower]
    #combine a list into one string
    string = " ".join(words)
    return string

news_df['news_cleaned']=news_df['News'].apply(text_preprocessing)
```

Calculation of Sentiment

- Sentiment is calculated by counting positive and negative words
- Negated words accounted for
- Adjustments because of COVID-19 news:
  - decreases are positive
  - increases are negative
  - spike is negative
  - elevated is negative
- Loughran & McDonald positive and negative dictionaries

Calculation of Sentiment

```
def wordcount(words, dct): #to count the amount of words that are in a positive/negative dictionary
    counting = Counter(words)
    count = []
    for key, value in counting.items():
        if key in dct:
            count.append([key, value])
    return count

def negwordcount(words, dct, negdct, lngram): #to count the amount of negation words
    mid = int(lngram / 2)
    ng = ngrams(words, lngram)
    nglist = []
    for grams in ng:
        nglist.append(grams)
    keeper = []
    n = len(nglist)
    i = 1
    for grams in nglist:
        if n - i < int(lngram / 2):
            mid = mid + 1
        if grams[mid] in dct:
            for j in grams:
                if j in negdct:
                    keeper.append(grams[mid])
                    break
        i = i + 1
    count = wordcount(keeper, dct)

    return count
```

```
def lexcnt(txt, txt_raw, pos_dct, neg_dct, negat_dct, lngram):
    #txt is the preprocessed text to save computation time. The raw text is only used for seeing if negations are present.
    txt = word_tokenize(txt)
    # Count words in lexicon
    pos_wc = wordcount(txt, pos_dct)
    pos_wc = [cnt[1] for cnt in pos_wc]
    pos_wc = sum(pos_wc)

    neg_wc = wordcount(txt, neg_dct)
    neg_wc = [cnt[1] for cnt in neg_wc]
    neg_wc = sum(neg_wc)

    # Count negated words in lexicon
    pos_wcneg = negwordcount(txt_raw, pos_dct, negat_dct, lngram)
    pos_wcneg = [cnt[1] for cnt in pos_wcneg]
    pos_wcneg = sum(pos_wcneg)

    neg_wcneg = negwordcount(txt_raw, neg_dct, negat_dct, lngram)
    neg_wcneg = [cnt[1] for cnt in neg_wcneg]
    neg_wcneg = sum(neg_wcneg)

    pos = pos_wc - (pos_wcneg) + neg_wcneg
    neg = neg_wc - (neg_wcneg) + pos_wcneg

    if pos > neg:
        out = 1
    elif pos < neg:
        out = -1
    else:
        out = 0

    return pos, neg, out
```

Calculation of Sentiment

```
negat_dct = ["n't", "not", "never", "no", "neither", "nor", "none"]
lngram = 7

# Dictionaries
# negative dictionary
neg_dct = ""
with io.open("negativemaster.txt", "r", encoding = "utf-8", errors = "ignore") as infile:
    for line in infile:
        neg_dct = neg_dct + line
# saved the lm_negative dictionary in variable neg_dct
neg_dct = neg_dct.split("\n")
neg_dct = [e.lower() for e in neg_dct]   # converted uppercase words to lowercase

# positive dictionary
pos_dct = ""
with io.open("positivemaster.txt", "r", encoding = "utf-8", errors = "ignore") as infile:
    for line in infile:
        pos_dct = pos_dct + line

pos_dct = pos_dct.split("\n")
pos_dct = [e.lower() for e in pos_dct]

pred = [lexcnt(news_df["News"][i], news_df["news_cleaned"][i], pos_dct, neg_dct, negat_dct, lngram) for i in
range(0,news_df.shape[0])]
pred = pd.DataFrame(pred, columns=('p','n', 'out'))

news_df['Sentiment']= pred['out']
```

Scaling the Sentiment

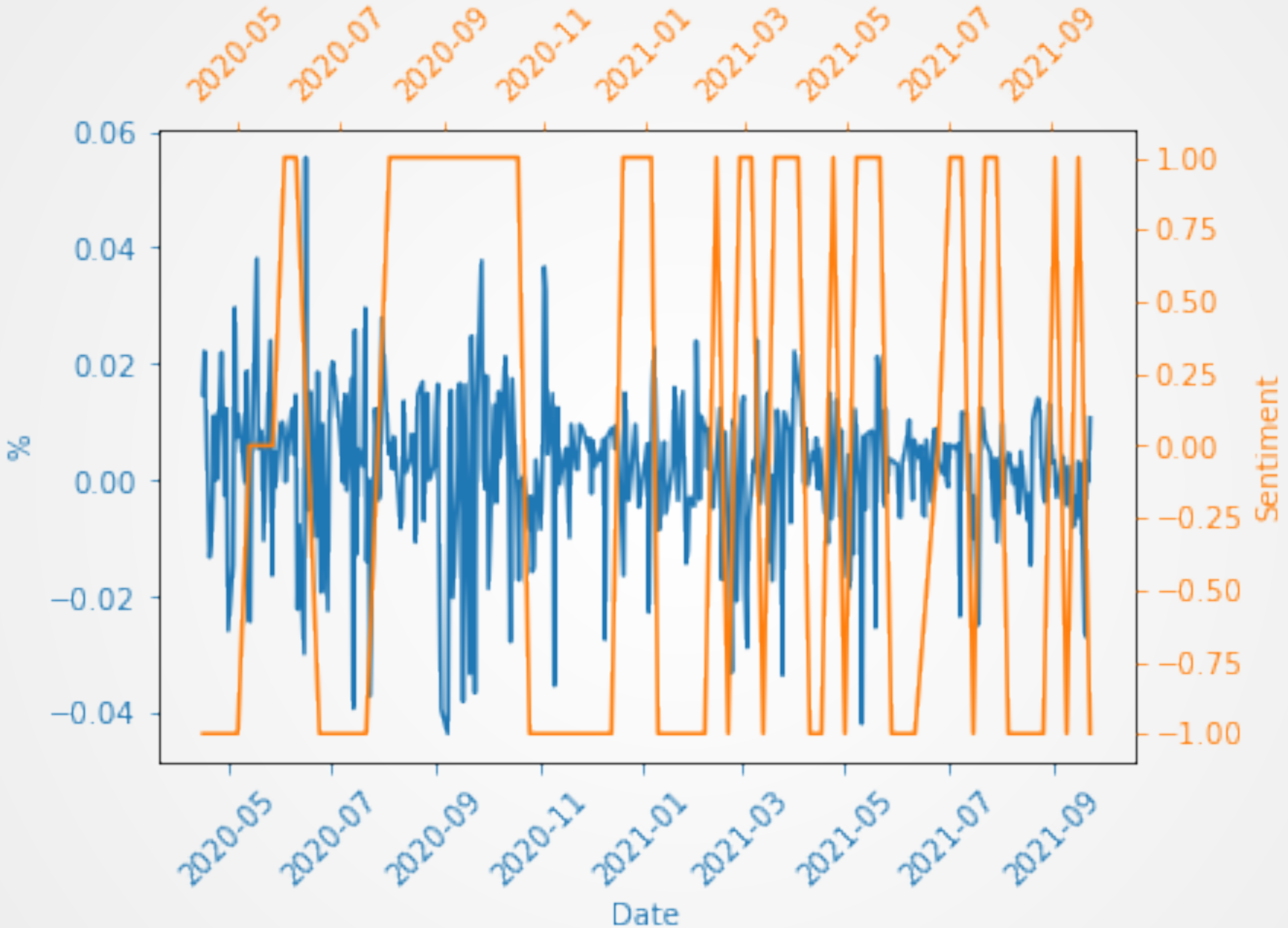
- Scaled version of sentiment based on frequencies of positive and negative words

- Conveys more information on sentiment

Scaling the Sentiment

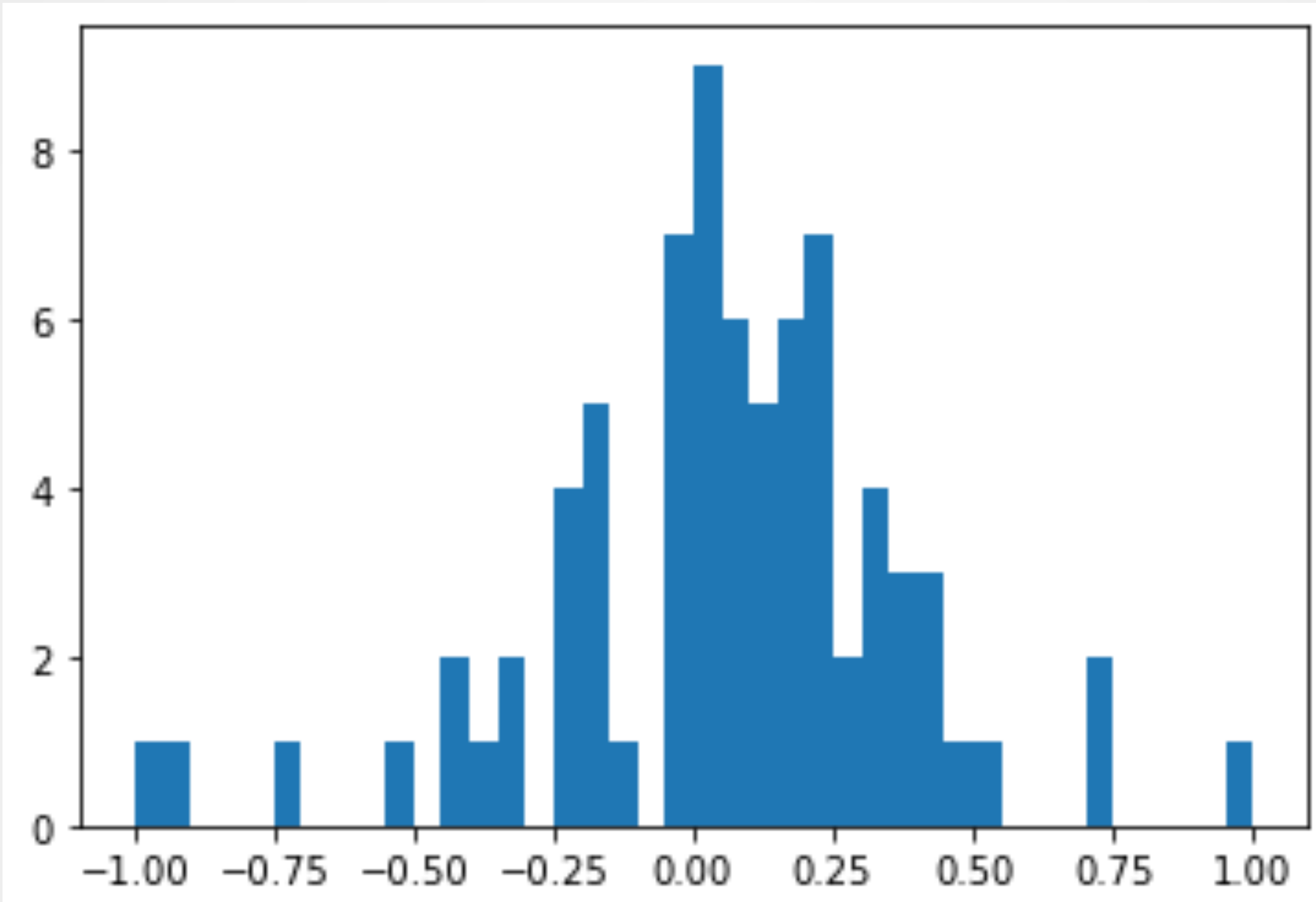
```
#Scaling the articles' sentiment
p_train = np.array(pred['p'])
p_train = p_train.reshape(-1,1)
scaler = MinMaxScaler(feature_range=(0,1))
p_scaled = scaler.fit_transform(p_train)
pred['p_scaled'] = p_scaled

n_train = np.array(pred['n'])
n_train = n_train.reshape(-1,1)
scaler = MinMaxScaler(feature_range=(0,1))
n_scaled = scaler.fit_transform(n_train)
pred['n_scaled'] = -n_scaled

pred['sent_body'] = pred[['p_scaled','n_scaled']].mean(axis=1)

s_train = np.array(pred['sent_body'])
s_train = s_train.reshape(-1,1)
scaler = MinMaxScaler(feature_range=(-1,1))
s_scaled = scaler.fit_transform(s_train)
pred['sent_scaled'] = s_scaled

pred.hist(column='sent_scaled',bins=40)

df_comp = news_df.join(pred['sent_scaled'], how='outer')
df_comp = df_comp.join(pred['p'], how='outer')
df_comp = df_comp.join(pred['n'], how='outer')
```

Sentiment and Nasdaq Returns



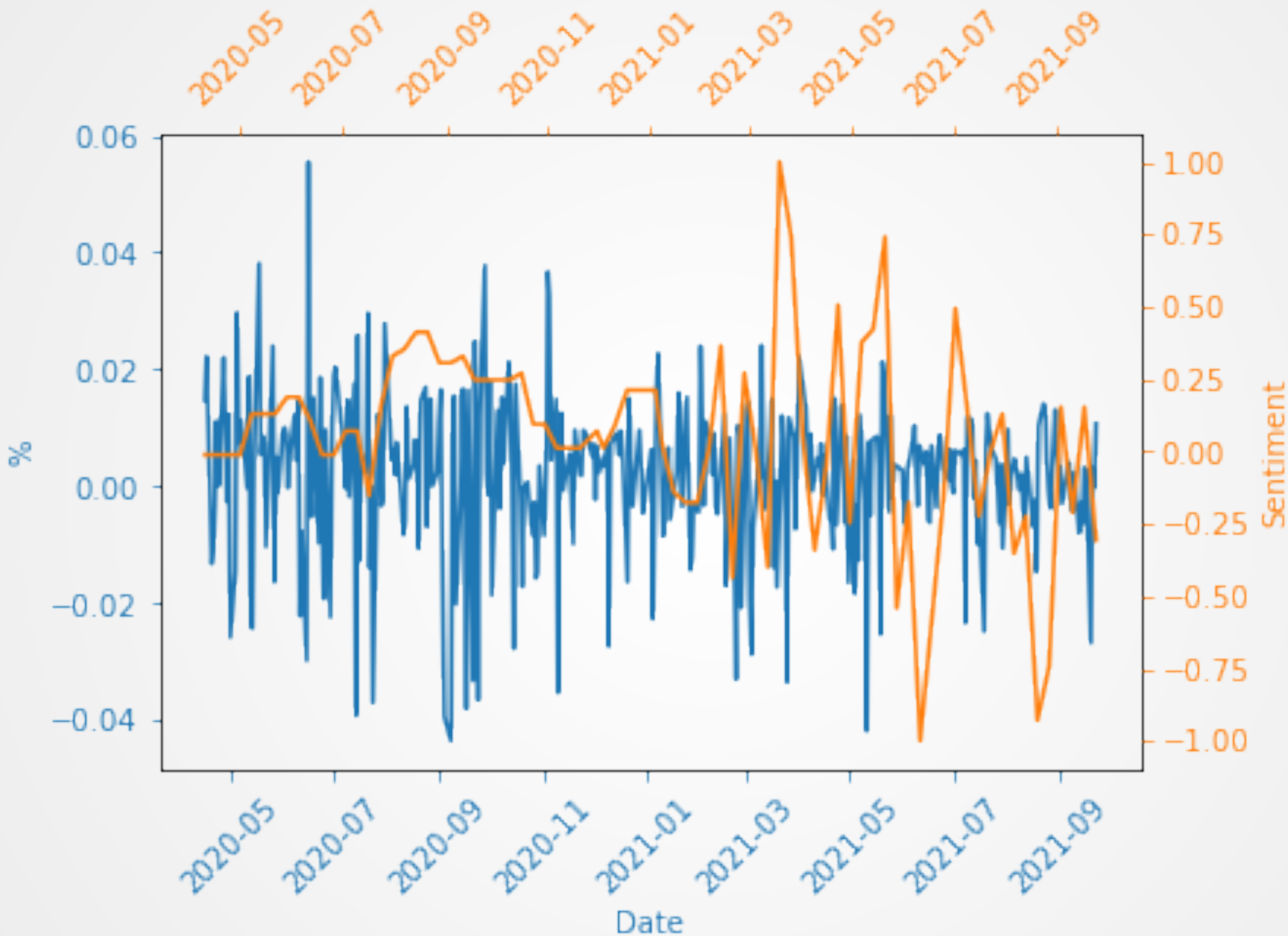
- No clear relation between sentiment and returns
- Appears that positive sentiment induces larger volatility

Sentiment and Nasdaq Returns



- The distribution of scaled sentiment shows a negatively skewed distribution, with mean/median slightly positive

Sentiment and Nasdaq Returns



- Again, a clear pattern cannot be distinguished

Sentiment and Nasdaq Returns

```
corr, _ = kendalltau(df_comp['Sentiment'][:75], nq_daily_returns)
corr
# -0.061

corr, _ = kendalltau(df_comp['sent_scaled'][:75], nq_daily_returns)
corr
# -0.028

corr, _ = kendalltau(df_comp['Sentiment'][:75], nq_var)
corr
# 0.129

corr, _ = kendalltau(df_comp['sent_scaled'][:75], nq_var)
corr
# 0.197
```

- Kendall's Tau is negative and small for sentiment and the succeeding returns
- Kendall's Tau is positive and slightly larger for sentiment and the succeeding volatility

Sentiment and Nasdaq Returns

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0084	0.003	2.700	0.009	0.002	0.015
x1	-0.0024	0.003	-0.750	0.456	-0.009	0.004

- Regression on Nasdaq returns after CDC update on the raw sentiment of that update: negative but insignificant

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0089	0.003	2.819	0.006	0.003	0.015
x1	-0.0068	0.009	-0.720	0.474	-0.026	0.012

- Regression on Nasdaq returns after CDC update on the scaled sentiment of that update: negative but insignificant

Sentiment and Nasdaq Returns

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0002	1.93e-05	8.240	0.000	0.000	0.000
x1	2.654e-05	2.02e-05	1.317	0.192	-1.36e-05	6.67e-05

- Regression on Nasdaq volatility the period after CDC update on the raw sentiment of that update: positive but insignificant

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0002	1.93e-05	7.791	0.000	0.000	0.000
x1	0.0001	5.77e-05	2.146	0.035	8.84e-06	0.000

- Regression on Nasdaq volatility the period after CDC update on the scaled sentiment of that update: positive and significant

Return and Volatility Prediction

- Can sentiment predict whether return is positive/negative and volatility is higher/lower?
- Create dummy variables
- Logistic regression and model training
- Returns and volatility

Return and Volatility Prediction

```
X = df_comp['sent_scaled'][:75]
X = X.tolist()
y = nq_daily_returns
y = y.tolist()
y1= np.asarray(nq_var)

y_dummy = []
for i in range(0,len(nq_daily_returns)):
    if(y[i]>0):
        y_dummy.append(1)
    else:
        y_dummy.append(0)

X_train,X_test,y_train,y_test=train_test_split(X,y_dummy,test_size=0.40,random_state=0
)

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
X_train = np.asarray(X_train)
y_train = np.asarray(y_train)

logreg.fit(X_train.reshape(-1,1),y_train)
```

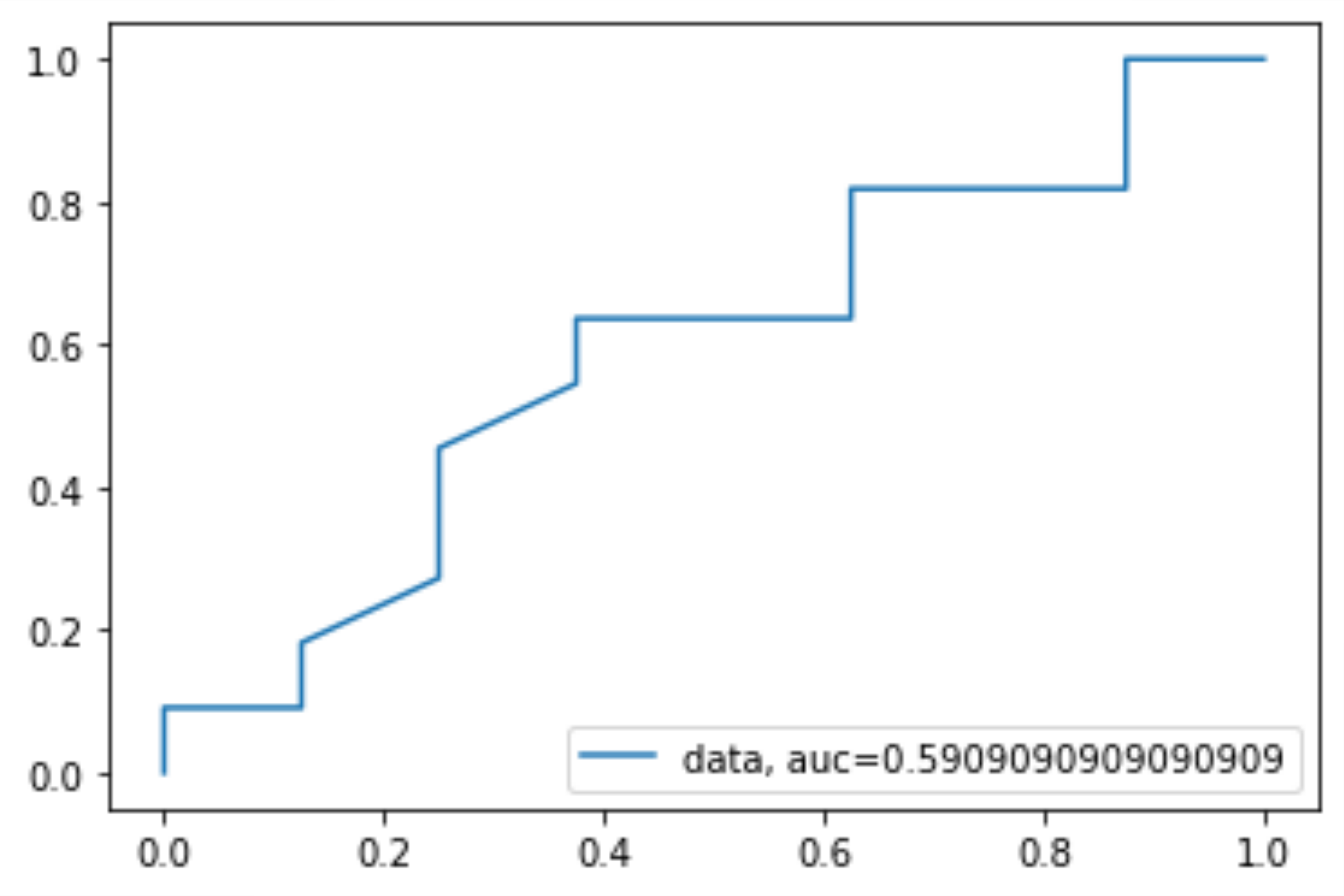
Return and Volatility Prediction

```
X_test = np.asarray(X_test)
y_pred=logreg.predict(X_test.reshape(-1,1))

cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix

[ 0,  8],
[ 0, 11]
```

- Model always predict higher returns
- True for only 11 cases out of the 19 test cases

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
#0.5789 : 57.89% of the predictions are correct
print("Precision:",metrics.precision_score(y_test, y_pred))
#0.5789 : 57.89% of the predictions of positive returns are correct
print("Recall:",metrics.recall_score(y_test, y_pred))
#1.000 : If returns are higher, the model correctly predicts this 100% of the time
```

Return and Volatility Prediction



- Area under the curve marginally better than 0.5 line
- Small sample

Return and Volatility Prediction

- Repeat for volatility:

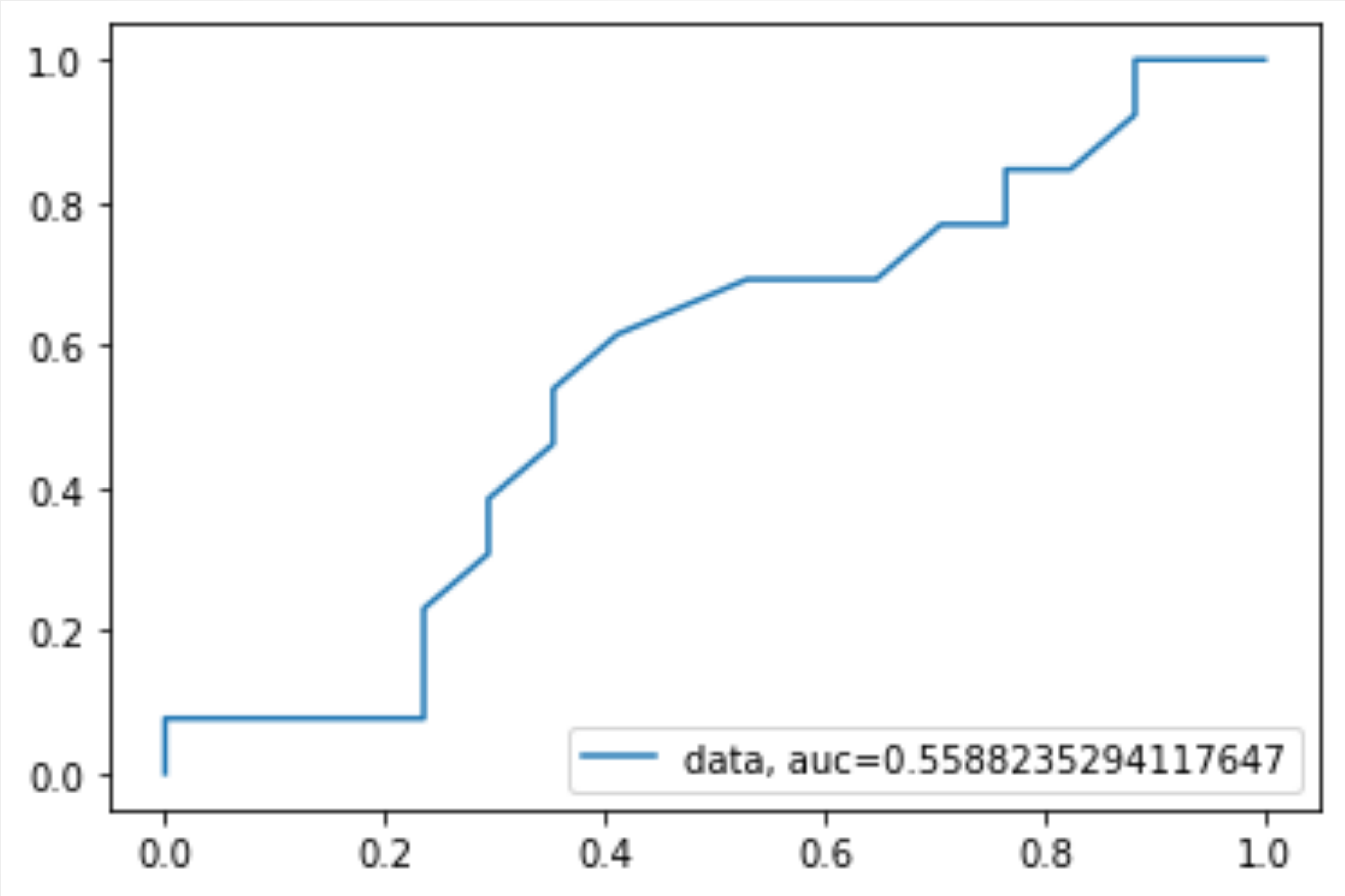
```
cnf_matrix
[ 6, 11],
[ 4,  9]
```

- 6 correct predictions of lower volatility, 9 correct predictions on higher volatility in test

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
#0.5 : 50% of the predictions are correct
print("Precision:",metrics.precision_score(y_test, y_pred))
#0.45 : of the predictions of higher volatility are correct
print("Recall:",metrics.recall_score(y_test, y_pred))
#0.69: If volatility is higher, the model correctly predicts this 69% of the time
```

- Possibly, small sample size results in mediocre statistics

Return and Volatility Prediction



- Again, area under the curve marginally better than 0.5 line

Return and Volatility Prediction

- Test sample rather large
- Set test sample at 25% and perform 10,000 times
- Note that returns are positive 64% of the time

Returns

	Mean	St. Dev.
Accuracy	0,639	0,082
Precision	0,639	0,084
Recall	0,996	0,022
AUC	0,461	0,112

Volatility

	Mean	St. Dev.
Accuracy	0,438	0,076
Precision	0,191	0,232
Recall	0,254	0,363
AUC	0,442	0,089

Conclusions

- Scraping the CDC News page brought challenges
- Adjusted the code to the specifics
- Raw sentiment provides inconclusive results
- Scaled sentiment of CDC's COVID updates did not have a significant relationship with returns following the update
- Scaled sentiment of CDC's COVID updates has a significant positive relationship with the volatility following the update
- Prediction results are less positive

DEDA Digital Economy & Decision Analytics

Sentiment Analysis of CDC COVID-19 Updates and their Market Impact

Thank you for your attention! And thanks for the intensive but very productive course on machine learning & fintech!

Ruben Bosch