Final_Project

August 5, 2024

[441]:
```python
# [1] Load articles datasets

import pandas as pd

# Read Excel files into DataFrames
articles = pd.read_csv("company_articles_updated.csv")
articles
```

[441]:
```
      company_code                                                url  \
0          4998306  https://www-capitaliq-spglobal-com.uaccess.uni…
1          4349418  https://www-capitaliq-spglobal-com.uaccess.uni…
2          6613555  https://www-capitaliq-spglobal-com.uaccess.uni…
3          6613555  https://www-capitaliq-spglobal-com.uaccess.uni…
4          6613555  https://www-capitaliq-spglobal-com.uaccess.uni…
…              …                                              …
4300       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4301       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4302       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4303       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4304       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…

                                                  title  \
0     European banks' capital offerings rebound to b…
1     Condor Gold Says It Has Received Offers for Ni…
2     *Calidus Resources Price Target Raised 3.6% to…
3     *Calidus Resources Price Target Cut 10% to A$0…
4     *Calidus Resources Upgraded to Speculative Buy…
…                                                  …
4300  UK Growth Is a Headache for the BOE; Friday, A…
4301  FTSE 100 Falls On Trader Caution After Strong …
4302  FTSE 100 Seen Opening Lower as Traders Weigh U…
4303  Chaarat Gold Holdings' Kapan Production Falls,…
4304  Iron ore prices slightly increase as China's p…

                             publication_date  \
0          Wednesday, July 19, 2023 5:48 AM ET
1             Friday, July 21, 2023 4:02 AM ET
2        Wednesday, April 17, 2024 11:05 PM ET
```

```
3       Thursday, February 29, 2024 4:12 PM ET
4     Wednesday, November 15, 2023 12:48 AM ET
…                                           …
4300        Friday, August 11, 2023 6:31 AM ET
4301        Friday, August 11, 2023 4:44 AM ET
4302        Friday, August 11, 2023 2:44 AM ET
4303        Friday, August 11, 2023 2:23 AM ET
4304           Monday, May 22, 2023 3:17 PM ET

                                                article
0     Capital offerings by banks in Europe recovered…
1     By Christian Moess Laursen\nCondor Gold said F…
2     (END) Dow Jones Newswires\nApril 17, 2024 23:0…
3     (END) Dow Jones Newswires\nFebruary 29, 2024 1…
4     (END) Dow Jones Newswires\nNovember 15, 2023 0…
…                                                 …
4300    UK Growth Is a Headache for the BOE\n0850 GM…
4301    FTSE 100 Falls On Trader Caution After Stron…
4302    FTSE 100 Seen Opening Lower as Traders Weigh…
4303    By Michael Susin\nChaarat Gold Holdings has …
4304    Iron ore prices slightly increased during th…

[4305 rows x 5 columns]
```

[442]:
```python
# [2.1] We need to convert the date string into dates

import numpy as np

type(articles['publication_date'][1])
```

[442]:
```
str
```

[443]:
```python
# [2.2] Parse the publication_date column to date format (also remove NaN dates)
articles['publication_date'] = pd.to_datetime(articles['publication_date'],␣
 ↪format='%A, %B %d, %Y %I:%M %p ET').dt.date
articles = articles.dropna(subset=['publication_date'])
articles
```

[443]:
```
      company_code                                                url  \
0          4998306  https://www-capitaliq-spglobal-com.uaccess.uni…
1          4349418  https://www-capitaliq-spglobal-com.uaccess.uni…
2          6613555  https://www-capitaliq-spglobal-com.uaccess.uni…
3          6613555  https://www-capitaliq-spglobal-com.uaccess.uni…
4          6613555  https://www-capitaliq-spglobal-com.uaccess.uni…
…              …                                                …
4300       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4301       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
```

```
4302      4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4303      4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4304      4349065  https://www-capitaliq-spglobal-com.uaccess.uni…

                                                  title publication_date  \
0     European banks' capital offerings rebound to b…       2023-07-19
1     Condor Gold Says It Has Received Offers for Ni…       2023-07-21
2     *Calidus Resources Price Target Raised 3.6% to…       2024-04-17
3     *Calidus Resources Price Target Cut 10% to A$0…       2024-02-29
4     *Calidus Resources Upgraded to Speculative Buy…       2023-11-15
…                                                   …              …
4300  UK Growth Is a Headache for the BOE; Friday, A…       2023-08-11
4301  FTSE 100 Falls On Trader Caution After Strong …       2023-08-11
4302  FTSE 100 Seen Opening Lower as Traders Weigh U…       2023-08-11
4303  Chaarat Gold Holdings' Kapan Production Falls,…       2023-08-11
4304  Iron ore prices slightly increase as China's p…       2023-05-22

                                                article
0     Capital offerings by banks in Europe recovered…
1     By Christian Moess Laursen\nCondor Gold said F…
2     (END) Dow Jones Newswires\nApril 17, 2024 23:0…
3     (END) Dow Jones Newswires\nFebruary 29, 2024 1…
4     (END) Dow Jones Newswires\nNovember 15, 2023 0…
…                                                   …
4300    UK Growth Is a Headache for the BOE\n0850 GM…
4301    FTSE 100 Falls On Trader Caution After Stron…
4302    FTSE 100 Seen Opening Lower as Traders Weigh…
4303    By Michael Susin\nChaarat Gold Holdings has …
4304    Iron ore prices slightly increased during th…

[4304 rows x 5 columns]
```

[444]:

```python
# [2.3] ... and also account for some articles being published on non-trading dates
articles_1 = articles.copy()

for index, row in articles_1.iterrows():
    if row['publication_date'].weekday() + 1 == 6:    # Saturday
        articles_1.loc[index, 'publication_date'] = row['publication_date'] + pd.Timedelta(days=2)
    elif row['publication_date'].weekday() + 1 == 7:  # Sunday
        articles_1.loc[index, 'publication_date'] = row['publication_date'] + pd.Timedelta(days=1)

articles_1
```

[444]:
```
      company_code                                                url  \
0          4998306  https://www-capitaliq-spglobal-com.uaccess.uni…
1          4349418  https://www-capitaliq-spglobal-com.uaccess.uni…
2          6613555  https://www-capitaliq-spglobal-com.uaccess.uni…
3          6613555  https://www-capitaliq-spglobal-com.uaccess.uni…
4          6613555  https://www-capitaliq-spglobal-com.uaccess.uni…
…              …                                              …
4300       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4301       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4302       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4303       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…
4304       4349065  https://www-capitaliq-spglobal-com.uaccess.uni…

                                              title publication_date  \
0     European banks' capital offerings rebound to b…       2023-07-19
1     Condor Gold Says It Has Received Offers for Ni…       2023-07-21
2     *Calidus Resources Price Target Raised 3.6% to…       2024-04-17
3     *Calidus Resources Price Target Cut 10% to A$0…       2024-02-29
4     *Calidus Resources Upgraded to Speculative Buy…       2023-11-15
…                                                …                …
4300  UK Growth Is a Headache for the BOE; Friday, A…       2023-08-11
4301  FTSE 100 Falls On Trader Caution After Strong …       2023-08-11
4302  FTSE 100 Seen Opening Lower as Traders Weigh U…       2023-08-11
4303  Chaarat Gold Holdings' Kapan Production Falls,…       2023-08-11
4304  Iron ore prices slightly increase as China's p…       2023-05-22

                                             article
0     Capital offerings by banks in Europe recovered…
1     By Christian Moess Laursen\nCondor Gold said F…
2     (END) Dow Jones Newswires\nApril 17, 2024 23:0…
3     (END) Dow Jones Newswires\nFebruary 29, 2024 1…
4     (END) Dow Jones Newswires\nNovember 15, 2023 0…
…                                                …
4300    UK Growth Is a Headache for the BOE\n0850 GM…
4301    FTSE 100 Falls On Trader Caution After Stron…
4302    FTSE 100 Seen Opening Lower as Traders Weigh…
4303    By Michael Susin\nChaarat Gold Holdings has …
4304    Iron ore prices slightly increased during th…

[4304 rows x 5 columns]
```

[445]:
```python
# [3] Concatenate articles and titles falling on the same dates for the same␣
 ↪companies
# Replace NaN values with empty strings
articles_1['title'] = articles_1['title'].fillna('')
articles_1['article'] = articles_1['article'].fillna('')
```

```python
# Group by 'company_code' and 'publication_date', then concatenate 'title' and␣
↪'article'
articles_1 = articles_1.groupby(['company_code', 'publication_date']).agg({
    'title': ' '.join,
    'article': ' '.join
}).reset_index()

articles_1
```

[445]:

```
      company_code publication_date  \
0           100607       2023-08-31
1           100607       2023-09-14
2           100669       2022-08-02
3           100669       2022-10-04
4           100669       2023-03-02
…              …              …
3699     112934797       2023-10-12
3700     112934797       2023-11-09
3701     112934797       2024-01-05
3702     112934797       2024-01-10
3703     112934797       2024-01-17

                                                  title  \
0     Citigroup's CFO Mark Mason reclaims spot as hi…
1     US bank branch M&A activity muted with only 9 …
2     US bank stocks record best 2022 performance in…
3     Fed's aggressive tightening continues to weigh…
4     US bank stocks post nearly flat median return …
…                                                   …
3699  State of the Pipeline - as of Oct. 11, 2023; R…
3700  State of the Pipeline - as of Nov. 8, 2023; Ro…
3701  Luse Gorman dominates 2023 mutual bank convers…
3702  2023 conversion class features 2nd-largest sta…
3703  US banks' capital offerings rose 61.7% year ov…

                                                article
0     After losing his position to Bank of America C…
1     US whole-bank M&A might have sputtered back to…
2     The U.S. banking industry recorded its best st…
3     U.S. bank stocks continued to take a beating i…
4     U.S. bank stocks ended February with a median …
…                                                   …
3699  This feature has the latest news from the mutu…
3700  This feature has the latest news from the mutu…
3701  Luse Gorman PC nearly swept the legal counsel …
3702  The mutual bank conversions that closed in 202…
3703  US banks' capital issuances rose through 2023,…
```

```
[3704 rows x 4 columns]
```

[446]:
```python
# [4.1] Load stocks dataset

import pandas as pd

# Read Excel files into DataFrames
stocks = pd.read_csv("stocks.csv")
stocks.head()
```

[446]:
```
   index                 Date     Close     Volume  Industry Group  \
0      0  2024-07-25 22:51:00  0.560000        0.0             NaN
1      1  2024-07-25 00:00:00  0.515704  1230694.0      308.820563
2      2  2024-07-24 00:00:00  0.571206  5761410.0      312.439291
3      3  2024-07-23 00:00:00  0.891722   786094.0      314.341423
4      4  2024-07-22 00:00:00  0.716205    62807.0      313.255752

          Ticker  Company code
0  NASDAQCM:LITM      10992240
1  NASDAQCM:LITM      10992240
2  NASDAQCM:LITM      10992240
3  NASDAQCM:LITM      10992240
4  NASDAQCM:LITM      10992240
```

[447]:
```python
# [4.2] ... to date format
stocks['Date'] = pd.to_datetime(stocks['Date'], format='mixed').dt.date
stocks.head()
```

[447]:
```
   index        Date     Close     Volume  Industry Group         Ticker  \
0      0  2024-07-25  0.560000        0.0             NaN  NASDAQCM:LITM
1      1  2024-07-25  0.515704  1230694.0      308.820563  NASDAQCM:LITM
2      2  2024-07-24  0.571206  5761410.0      312.439291  NASDAQCM:LITM
3      3  2024-07-23  0.891722   786094.0      314.341423  NASDAQCM:LITM
4      4  2024-07-22  0.716205    62807.0      313.255752  NASDAQCM:LITM

   Company code
0      10992240
1      10992240
2      10992240
3      10992240
4      10992240
```

[448]:
```python
# [5] filter out the companies not included in stocks:
len(stocks['Company code'].unique())
```

[448]: 493

[449]:

```python
# [5] -cont.-
# Filter articles data to keep only those company codes present in stocks data
articles_2 = articles_1[articles_1['company_code'].isin(stocks['Company code'])]
articles_2
```

[449]:

```
     company_code publication_date  \
0          100607       2023-08-31
1          100607       2023-09-14
2          100669       2022-08-02
3          100669       2022-10-04
4          100669       2023-03-02
…             …              …
3699    112934797       2023-10-12
3700    112934797       2023-11-09
3701    112934797       2024-01-05
3702    112934797       2024-01-10
3703    112934797       2024-01-17

                                                  title  \
0     Citigroup's CFO Mark Mason reclaims spot as hi…
1     US bank branch M&A activity muted with only 9 …
2     US bank stocks record best 2022 performance in…
3     Fed's aggressive tightening continues to weigh…
4     US bank stocks post nearly flat median return …
…                                                 …
3699  State of the Pipeline - as of Oct. 11, 2023; R…
3700  State of the Pipeline - as of Nov. 8, 2023; Ro…
3701  Luse Gorman dominates 2023 mutual bank convers…
3702  2023 conversion class features 2nd-largest sta…
3703  US banks' capital offerings rose 61.7% year ov…

                                                article
0     After losing his position to Bank of America C…
1     US whole-bank M&A might have sputtered back to…
2     The U.S. banking industry recorded its best st…
3     U.S. bank stocks continued to take a beating i…
4     U.S. bank stocks ended February with a median …
…                                                 …
3699  This feature has the latest news from the mutu…
3700  This feature has the latest news from the mutu…
3701  Luse Gorman PC nearly swept the legal counsel …
3702  The mutual bank conversions that closed in 202…
3703  US banks' capital issuances rose through 2023,…

[3638 rows x 4 columns]
```

[450]:
```
# [*] check how many companies have how many articles
# Group by company_code and count the number of articles for each company
company_article_counts = articles_2.groupby('company_code').size().
 ↪reset_index(name='article_count')

# Get the summary of how many companies have how many articles
summary = company_article_counts.groupby('article_count').size().
 ↪reset_index(name='number_of_companies')

summary
```

[450]:

	article_count	number_of_companies
0	1	174
1	2	58
2	3	32
3	4	31
4	5	23
5	6	17
6	7	19
7	8	19
8	9	11
9	10	11
10	11	10
11	12	10
12	13	6
13	14	5
14	15	10
15	16	7
16	17	9
17	19	3
18	20	3
19	21	1
20	22	2
21	23	2
22	24	2
23	25	1
24	26	1
25	27	1
26	28	3
27	29	1
28	30	3
29	36	3
30	37	2
31	41	1
32	46	1
33	48	1
34	53	1

35	63	1
36	65	1
37	72	2
38	76	1
39	82	1
40	83	1
41	89	1
42	95	1

[451]:

```python
# [6] Let's assign a group for each cluster

# Group by company_code and count the number of articles for each company
company_article_counts = articles_2.groupby('company_code').size().
↪reset_index(name='article_count')

# Define the function to determine the group
def assign_group(article_count):
    if article_count == 1:
        return 1
    elif 2 <= article_count <= 5:
        return 2
    elif 6 <= article_count <= 15:
        return 3
    else:
        return 4

# Apply the function to the article_count column to create a new group column
company_article_counts['group'] = company_article_counts['article_count'].
↪apply(assign_group)

# Merge the group information back into the original dataframe
articles_3 = articles_2.merge(company_article_counts[['company_code',␣
↪'group']], on='company_code', how='left')

# ALSO, ALSO add a unique article identifier (will be needed later):
articles_3['article_code'] = articles_3['company_code'].astype(str) + '_' +␣
↪articles_3['publication_date'].astype(str)

# Display the first few rows of the updated dataframe
articles_3
```

[451]:

	company_code	publication_date \
0	100607	2023-08-31
1	100607	2023-09-14
2	100669	2022-08-02
3	100669	2022-10-04
4	100669	2023-03-02

```
…           …             …
3633    112934797      2023-10-12
3634    112934797      2023-11-09
3635    112934797      2024-01-05
3636    112934797      2024-01-10
3637    112934797      2024-01-17

                                                  title  \
0     Citigroup's CFO Mark Mason reclaims spot as hi…
1     US bank branch M&A activity muted with only 9 …
2     US bank stocks record best 2022 performance in…
3     Fed's aggressive tightening continues to weigh…
4     US bank stocks post nearly flat median return …
…                                                  …
3633  State of the Pipeline - as of Oct. 11, 2023; R…
3634  State of the Pipeline - as of Nov. 8, 2023; Ro…
3635  Luse Gorman dominates 2023 mutual bank convers…
3636  2023 conversion class features 2nd-largest sta…
3637  US banks' capital offerings rose 61.7% year ov…

                                                article  group  \
0     After losing his position to Bank of America C…      2
1     US whole-bank M&A might have sputtered back to…      2
2     The U.S. banking industry recorded its best st…      3
3     U.S. bank stocks continued to take a beating i…      3
4     U.S. bank stocks ended February with a median …      3
…                                                  …    …
3633  This feature has the latest news from the mutu…      3
3634  This feature has the latest news from the mutu…      3
3635  Luse Gorman PC nearly swept the legal counsel …      3
3636  The mutual bank conversions that closed in 202…      3
3637  US banks' capital issuances rose through 2023,…      3

              article_code
0        100607_2023-08-31
1        100607_2023-09-14
2        100669_2022-08-02
3        100669_2022-10-04
4        100669_2023-03-02
…                      …
3633  112934797_2023-10-12
3634  112934797_2023-11-09
3635  112934797_2024-01-05
3636  112934797_2024-01-10
3637  112934797_2024-01-17

[3638 rows x 6 columns]
```

```
[452]: # [7] Count the number of unique company_codes for each group
unique_company_counts_per_group = articles_3.groupby('group')['company_code'].
 ↪nunique().reset_index(name='unique_company_count')
unique_company_counts_per_group
```

```
[452]:    group  unique_company_count
0      1                   174
1      2                   144
2      3                   118
3      4                    57
```

```
[453]: # [7] very good, it looks like we have a good distribution of companies/articles
# ... although, we might want to twitch the distribution a little later
unique_company_counts_per_group = articles_3.groupby('group')['company_code'].
 ↪count().reset_index(name='unique_company_count')
unique_company_counts_per_group
```

```
[453]:    group  unique_company_count
0      1                   174
1      2                   451
2      3                  1124
3      4                  1889
```

```
[454]: # [8] Assign a publication_date to the stocks df that will track if that␣
 ↪observation should be the beginning of a new delay point
# Create a set of tuples (company_code, publication_date) for fast lookup
article_dates = set(zip(articles_3['company_code'],␣
 ↪articles_3['publication_date']))

# Create the new column in the stocks DataFrame
stocks['publication_date'] = stocks.apply(
    lambda row: 1 if (row['Company code'], row['Date']) in article_dates else 0,
    axis=1
)
```

```
[455]: # [8] check
stocks['article_code'] = stocks['Company code'].astype(str) + "_" +␣
 ↪stocks['Date'].astype(str)
stocks.loc[stocks['publication_date'] == 0, 'article_code'] = ""

stocks.head()
```

```
[455]:    index        Date     Close     Volume  Industry Group         Ticker  \
0      0  2024-07-25  0.560000        0.0             NaN  NASDAQCM:LITM
1      1  2024-07-25  0.515704  1230694.0      308.820563  NASDAQCM:LITM
2      2  2024-07-24  0.571206  5761410.0      312.439291  NASDAQCM:LITM
3      3  2024-07-23  0.891722   786094.0      314.341423  NASDAQCM:LITM
```

```
4      4  2024-07-22  0.716205    62807.0      313.255752  NASDAQCM:LITM

   Company code  publication_date article_code
0      10992240                 0
1      10992240                 0
2      10992240                 0
3      10992240                 0
4      10992240                 0
```

[456]:
```python
# [*] Perfect
len(stocks[stocks['publication_date'] == 1]['publication_date'])
```

[456]: 3638

[457]:
```python
# ^^^
len(articles_3['publication_date'])
```

[457]: 3638

[502]:
```python
# [9] Now, we create a new column with unique article_codes for each period for␣
↪each company
stocks2 = stocks.copy()

# Assuming stocks2 DataFrame is already sorted by 'Company code' and 'Date'
# We will create a helper function to apply the filling logic for each company
def fill_article_code(group):
    group['article_code'] = group['article_code'].replace("", pd.NA).bfill().
↪fillna("")  # Forward fill the article codes
    return group  # Return the group sorted by Date in ascending order

# Apply the function to each company group
stocks2 = stocks2.groupby('Company code').apply(fill_article_code).
↪reset_index(drop=True)

# Display the updated DataFrame
stocks2[stocks2['Company code'] == 100607][200:260]
```

[502]:
```
     index        Date      Close   Volume  Industry Group      Ticker  \
200   4120  2023-10-19  10.495050    100.0      126.501201  OTCQX:JUVF
201   4121  2023-10-18  10.518244  26108.0      127.547726  OTCQX:JUVF
202   4122  2023-10-17  10.251080    461.0      129.932937  OTCQX:JUVF
203   4123  2023-10-16  10.240560   2090.0      128.279581  OTCQX:JUVF
204   4124  2023-10-13  10.953750   2910.0      127.069482  OTCQX:JUVF
205   4125  2023-10-12  11.847500   4100.0      126.989873  OTCQX:JUVF
206   4126  2023-10-11  11.869200   1149.0      127.806471  OTCQX:JUVF
207   4127  2023-10-10  12.025800    840.0      127.618061  OTCQX:JUVF
208   4128  2023-10-09  12.090825   1772.0      125.932682  OTCQX:JUVF
```

209	4129	2023-10-06	12.291500	NaN	125.766749	OTCQX:JUVF
210	4130	2023-10-05	12.347400	200.0	124.843848	OTCQX:JUVF
211	4131	2023-10-04	12.369500	NaN	124.056533	OTCQX:JUVF
212	4132	2023-10-03	12.424100	NaN	123.406476	OTCQX:JUVF
213	4133	2023-10-02	12.369500	110.0	125.492548	OTCQX:JUVF
214	4134	2023-09-29	12.046200	1270.0	128.015616	OTCQX:JUVF
215	4135	2023-09-28	12.312664	NaN	128.389941	OTCQX:JUVF
216	4136	2023-09-27	12.367306	NaN	127.157972	OTCQX:JUVF
217	4137	2023-09-26	12.302256	1000.0	126.967560	OTCQX:JUVF
218	4138	2023-09-25	12.049268	531.0	128.918835	OTCQX:JUVF
219	4139	2023-09-22	11.964600	1983.0	127.992257	OTCQX:JUVF
220	4140	2023-09-21	12.125754	850.0	129.501635	OTCQX:JUVF
221	4141	2023-09-20	12.127700	NaN	131.131589	OTCQX:JUVF
222	4142	2023-09-19	12.170600	NaN	131.883747	OTCQX:JUVF
223	4143	2023-09-18	12.170600	NaN	132.222638	OTCQX:JUVF
224	4144	2023-09-15	12.183600	155.0	132.823468	OTCQX:JUVF
225	4145	2023-09-14	12.187500	155.0	133.727489	OTCQX:JUVF
226	4146	2023-09-13	11.866425	NaN	131.370515	OTCQX:JUVF
227	4147	2023-09-12	11.884275	3095.0	132.356547	OTCQX:JUVF
228	4148	2023-09-11	12.161635	833.0	130.086058	OTCQX:JUVF
229	4149	2023-09-08	12.460890	3512.0	129.698739	OTCQX:JUVF
230	4150	2023-09-07	12.428850	1309.0	128.794832	OTCQX:JUVF
231	4151	2023-09-06	12.598200	1467.0	130.349279	OTCQX:JUVF
232	4152	2023-09-05	12.834250	3813.0	131.834524	OTCQX:JUVF
233	4153	2023-09-04	NaN	NaN	133.521395	OTCQX:JUVF
234	4154	2023-09-01	13.618080	11907.0	133.521395	OTCQX:JUVF
235	4155	2023-08-31	13.461200	1652.0	132.030725	OTCQX:JUVF
236	4156	2023-08-30	13.094786	4815.0	132.716861	OTCQX:JUVF
237	4157	2023-08-29	13.279680	NaN	133.549929	OTCQX:JUVF
238	4158	2023-08-28	13.327200	1782.0	132.184920	OTCQX:JUVF
239	4159	2023-08-25	13.551720	5300.0	130.985185	OTCQX:JUVF
240	4160	2023-08-24	13.263205	450.0	131.507104	OTCQX:JUVF
241	4161	2023-08-23	13.224960	NaN	131.418176	OTCQX:JUVF
242	4162	2023-08-22	13.223525	NaN	130.465357	OTCQX:JUVF
243	4163	2023-08-21	13.180475	250.0	133.739378	OTCQX:JUVF
244	4164	2023-08-18	13.194825	342.0	133.893374	OTCQX:JUVF
245	4165	2023-08-17	13.329850	300.0	134.056556	OTCQX:JUVF
246	4166	2023-08-16	13.292150	200.0	134.413107	OTCQX:JUVF
247	4167	2023-08-15	13.271850	1650.0	135.811714	OTCQX:JUVF
248	4168	2023-08-14	13.101660	2912.0	139.775679	OTCQX:JUVF
249	4169	2023-08-11	13.000275	NaN	141.156727	OTCQX:JUVF
250	4170	2023-08-10	12.936150	NaN	140.452412	OTCQX:JUVF
251	4171	2023-08-09	12.978900	680.0	140.514563	OTCQX:JUVF
252	4172	2023-08-08	13.063050	1295.0	142.709082	OTCQX:JUVF
253	4173	2023-08-07	13.086720	NaN	144.305116	OTCQX:JUVF
254	4174	2023-08-04	13.044960	123.0	143.041925	OTCQX:JUVF
255	4175	2023-08-03	13.105855	NaN	143.413919	OTCQX:JUVF

256	4176	2023-08-02	13.121640	100.0	142.494044	OTCQX:JUVF
257	4177	2023-08-01	13.014240	NaN	143.938259	OTCQX:JUVF
258	4178	2023-07-31	12.948598	440.0	145.320379	OTCQX:JUVF
259	4179	2023-07-28	13.150050	497.0	144.824231	OTCQX:JUVF

	Company code	publication_date	article_code
200	100607	0	100607_2023-09-14
201	100607	0	100607_2023-09-14
202	100607	0	100607_2023-09-14
203	100607	0	100607_2023-09-14
204	100607	0	100607_2023-09-14
205	100607	0	100607_2023-09-14
206	100607	0	100607_2023-09-14
207	100607	0	100607_2023-09-14
208	100607	0	100607_2023-09-14
209	100607	0	100607_2023-09-14
210	100607	0	100607_2023-09-14
211	100607	0	100607_2023-09-14
212	100607	0	100607_2023-09-14
213	100607	0	100607_2023-09-14
214	100607	0	100607_2023-09-14
215	100607	0	100607_2023-09-14
216	100607	0	100607_2023-09-14
217	100607	0	100607_2023-09-14
218	100607	0	100607_2023-09-14
219	100607	0	100607_2023-09-14
220	100607	0	100607_2023-09-14
221	100607	0	100607_2023-09-14
222	100607	0	100607_2023-09-14
223	100607	0	100607_2023-09-14
224	100607	0	100607_2023-09-14
225	100607	1	100607_2023-09-14
226	100607	0	100607_2023-08-31
227	100607	0	100607_2023-08-31
228	100607	0	100607_2023-08-31
229	100607	0	100607_2023-08-31
230	100607	0	100607_2023-08-31
231	100607	0	100607_2023-08-31
232	100607	0	100607_2023-08-31
233	100607	0	100607_2023-08-31
234	100607	0	100607_2023-08-31
235	100607	1	100607_2023-08-31
236	100607	0	
237	100607	0	
238	100607	0	
239	100607	0	
240	100607	0	

```
241         100607                  0
242         100607                  0
243         100607                  0
244         100607                  0
245         100607                  0
246         100607                  0
247         100607                  0
248         100607                  0
249         100607                  0
250         100607                  0
251         100607                  0
252         100607                  0
253         100607                  0
254         100607                  0
255         100607                  0
256         100607                  0
257         100607                  0
258         100607                  0
259         100607                  0
```

[562]:

```
# [10] And now, we calculate the number of days when the market had the same␣
↪publically available information for each trading day
interday_counts = stocks2.groupby('article_code').size().
↪reset_index(name='count')[1:]
interday_counts
```

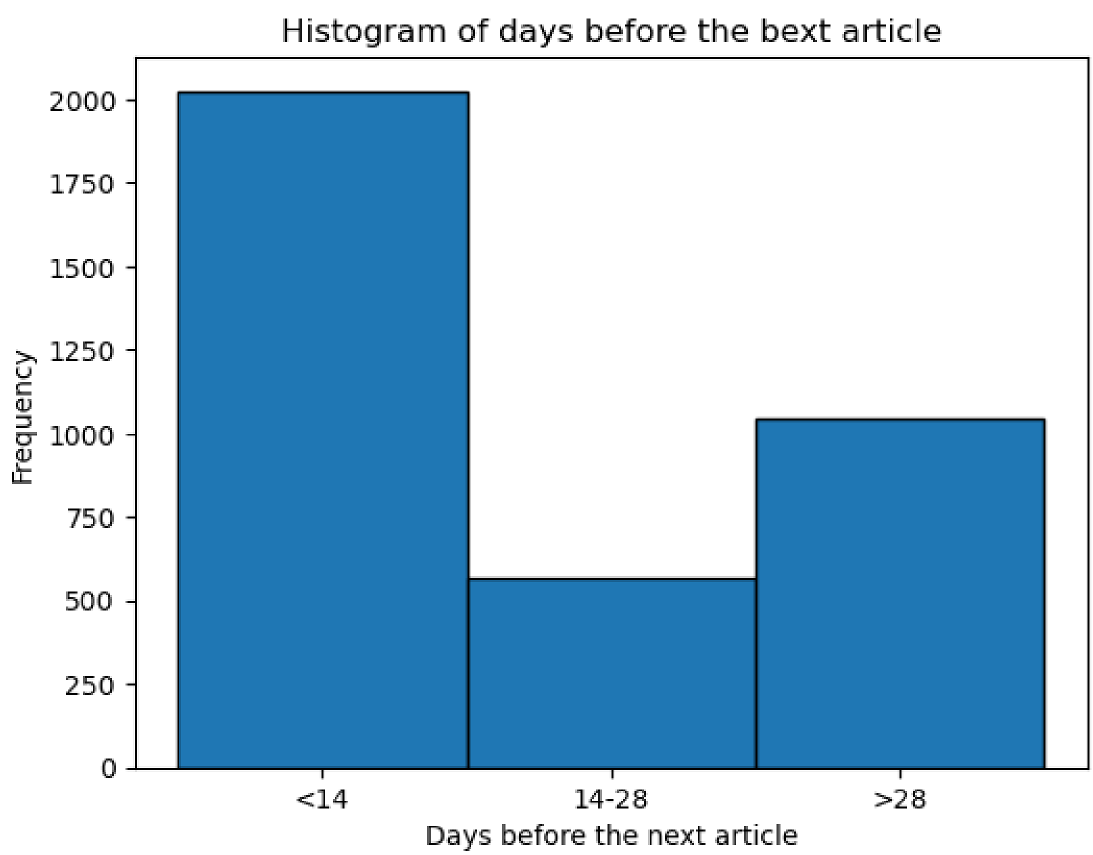
[562]:

```
                article_code  count
1     100034886_2023-08-29     23
2     100034886_2023-09-29     19
3     100034886_2023-10-26      3
4     100034886_2023-10-31      8
5     100034886_2023-11-10      1
…                        …    …
3634    9756394_2023-12-12      6
3635    9756394_2023-12-20     33
3636    9756394_2024-02-05    124
3637    9915809_2023-11-06    168
3638    9915809_2024-06-27     23

[3638 rows x 2 columns]
```

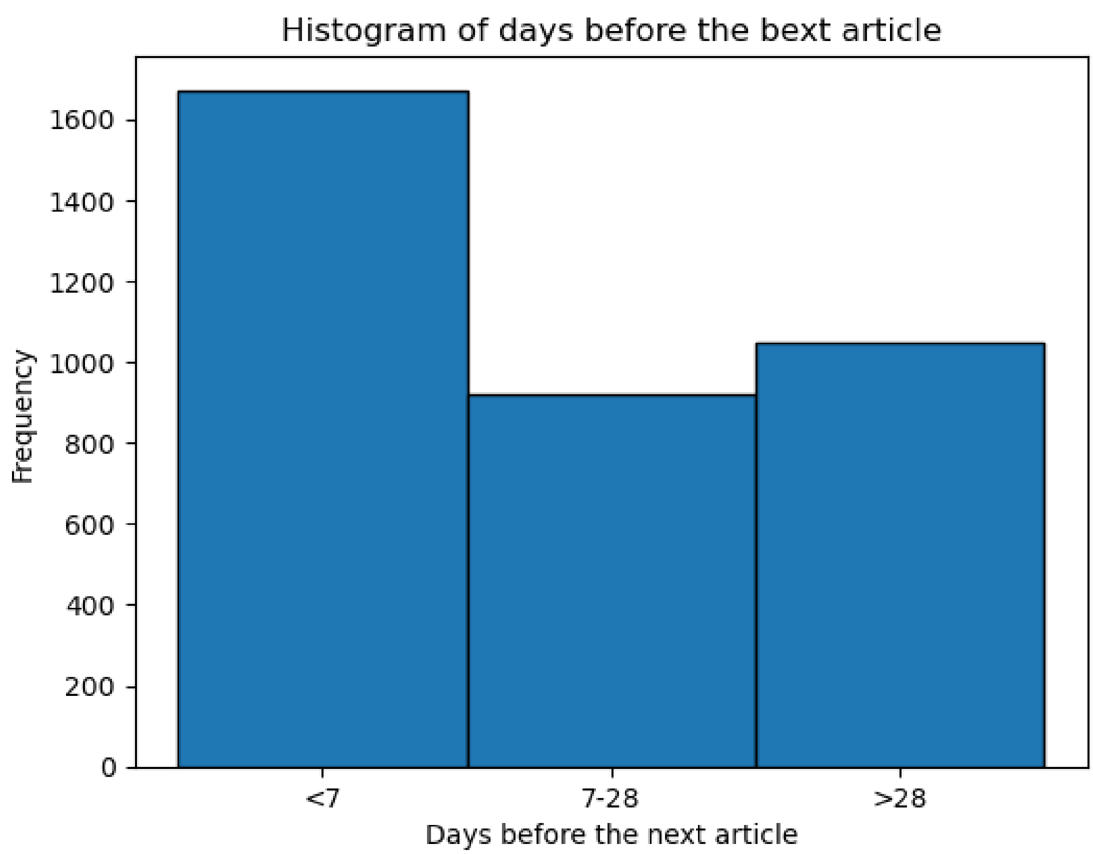
[504]:

```
# [11] It seems that more than 2,000 articles are not so useful for the␣
↪strictest private info leakage impact reduction criteria
# (so that the period before the next article should be more than 2 weeks)

import matplotlib.pyplot as plt
```

```
# Plot a histogram of the counts
bins = [0, 14, 28, 1000]
labels = ['<14', '14-28', '>28']

# Create a new column for bin labels
code_counts['bin'] = pd.cut(code_counts['count'], bins=bins, labels=labels,␣
↪right=False)

# Get the counts for each bin
bin_counts = code_counts['bin'].value_counts().sort_index()

# Plot using bar to ensure equal width bars
plt.bar(labels, bin_counts.values, width=1, edgecolor='black')
plt.xlabel('Days before the next article')
plt.ylabel('Frequency')
plt.title('Histogram of days before the bext article')
plt.show()
```



[505]:
```
# [11] The less stricter condition of only 7 days gives us ~400 more articles␣
 ↪to work with

import matplotlib.pyplot as plt

# Plot a histogram of the counts
bins = [0, 7, 28, 1000]
labels = ['<7', '7-28', '>28']

# Create a new column for bin labels
code_counts['bin'] = pd.cut(code_counts['count'], bins=bins, labels=labels,␣
 ↪right=False)

# Get the counts for each bin
bin_counts = code_counts['bin'].value_counts().sort_index()

# Plot using bar to ensure equal width bars
plt.bar(labels, bin_counts.values, width=1, edgecolor='black')
plt.xlabel('Days before the next article')
plt.ylabel('Frequency')
plt.title('Histogram of days before the bext article')
plt.show()
```



[578]:
```python
# [12] Calculate returns
stocks3 = stocks2.dropna(subset=['Close'])

def reverse_pct_change(group):
    group = group.iloc[::-1]
    group['returns'] = group['Close'].pct_change()
    return group.iloc[::-1]

# Apply the function to each group
stocks3 = stocks3.groupby('Company code', group_keys=False).
 ↪apply(reverse_pct_change)
stocks3['returns_abs'] = stocks3['returns'] + 1
stocks3
```

[578]:
```
         index        Date      Close   Volume  Industry Group       Ticker  \
0         3920  2024-07-25  10.360125   1150.0      186.132106  OTCQX:JUVF
1         3921  2024-07-24  10.152726      NaN      184.602780  OTCQX:JUVF
2         3922  2024-07-23  10.151624    600.0      186.409479  OTCQX:JUVF
3         3923  2024-07-22  10.161928      NaN      185.149417  OTCQX:JUVF
4         3924  2024-07-19  10.158610      NaN      184.466579  OTCQX:JUVF
…          …         …          …       …                …          …
385115  308375  2023-10-24   8.405160   2600.0      121.327170  OTCQB:PFSB
385116  308376  2023-10-23   8.366000   7600.0      121.456937  OTCQB:PFSB
385117  308377  2023-10-20   8.402490   5312.0      123.077113  OTCQB:PFSB
385118  308378  2023-10-19   8.414950  18323.0      126.501201  OTCQB:PFSB
385119  308379  2023-10-18   8.543700  43807.0      127.547726  OTCQB:PFSB

        Company code  publication_date          article_code   returns  \
0             100607                 0     100607_2023-09-14  0.020428
1             100607                 0     100607_2023-09-14  0.000109
2             100607                 0     100607_2023-09-14 -0.001014
3             100607                 0     100607_2023-09-14  0.000327
4             100607                 0     100607_2023-09-14  0.001854
…                …                 …                    …         …
385115     112934797                 0  112934797_2023-10-12  0.004681
385116     112934797                 0  112934797_2023-10-12 -0.004343
385117     112934797                 0  112934797_2023-10-12 -0.001481
385118     112934797                 0  112934797_2023-10-12 -0.015070
385119     112934797                 0  112934797_2023-10-12       NaN

        returns_abs
0          1.020428
1          1.000109
2          0.998986
```

```
3          1.000327
4          1.001854
…               …
385115     1.004681
385116     0.995657
385117     0.998519
385118     0.984930
385119          NaN

[363945 rows x 11 columns]
```

[580]:
```python
# [12] Clean the data
stocks3 = stocks3[stocks3["article_code"] != ""]
stocks3 = stocks3.dropna(subset=['returns'])
stocks3 = stocks3.dropna(subset=['Volume'])
stocks3['sign'] = stocks3['returns'].apply(lambda x: 1 if x >= 0 else -1)
stocks3
```

[580]:
```
         index        Date      Close   Volume  Industry Group       Ticker  \
0         3920  2024-07-25  10.360125   1150.0      186.132106  OTCQX:JUVF
2         3922  2024-07-23  10.151624    600.0      186.409479  OTCQX:JUVF
6         3926  2024-07-17  10.114370   2148.0      189.017020  OTCQX:JUVF
8         3928  2024-07-15  10.655406    350.0      180.627622  OTCQX:JUVF
9         3929  2024-07-12  10.545500    500.0      177.113297  OTCQX:JUVF
…          …         …         …        …               …          …
385114  308374  2023-10-25   8.406940   2323.0      120.999855  OTCQB:PFSB
385115  308375  2023-10-24   8.405160   2600.0      121.327170  OTCQB:PFSB
385116  308376  2023-10-23   8.366000   7600.0      121.456937  OTCQB:PFSB
385117  308377  2023-10-20   8.402490   5312.0      123.077113  OTCQB:PFSB
385118  308378  2023-10-19   8.414950  18323.0      126.501201  OTCQB:PFSB

        Company code  publication_date           article_code   returns  \
0             100607                 0      100607_2023-09-14  0.020428
2             100607                 0      100607_2023-09-14 -0.001014
6             100607                 0      100607_2023-09-14 -0.053565
8             100607                 0      100607_2023-09-14  0.010422
9             100607                 0      100607_2023-09-14 -0.003153
…                  …                …                    …         …
385114     112934797                 0   112934797_2023-10-12  0.000212
385115     112934797                 0   112934797_2023-10-12  0.004681
385116     112934797                 0   112934797_2023-10-12 -0.004343
385117     112934797                 0   112934797_2023-10-12 -0.001481
385118     112934797                 0   112934797_2023-10-12 -0.015070

        returns_abs  sign
0          1.020428     1
2          0.998986    -1
```

```
6          0.946435   -1
8          1.010422    1
9          0.996847   -1
…              …    …
385114     1.000212    1
385115     1.004681    1
385116     0.995657   -1
385117     0.998519   -1
385118     0.984930   -1

[98860 rows x 12 columns]
```

[695]:

```
# [13] Remove miniscule price movements
import math

def consecutive_trades(df, interday_counts, days_check_period = 3, min_days =␣
↪14):



    interday_counts_with_periods = pd.DataFrame(columns=['article_code',␣
↪'count', 'stabilization_period'])      # [!] empty df to fill in later

    for p, i in enumerate(interday_counts[interday_counts['count'] >␣
↪min_days]['article_code']):                    # [!] loop over all article_codes␣
↪w. >14 days period until the next one

        returns_over_the_days_check_period = []
        returns_cumulative = []

        all_returns_for_X = df[df['article_code'] == i]['returns'][::-1]

        start_date = 0      # placeholders
        end_date = 0        # placeholders

        for x, return_for_that_day in enumerate(all_returns_for_X):             ␣
↪                               # [!] loop over all dates for that␣
↪particular article_code (in stocks df)
            start_date = df[df['article_code'] == i]['Date'][::-1].iloc[0]
            returns_over_the_days_check_period.append(return_for_that_day)
            moving_prod = math.prod([num + 1 for num in␣
↪returns_over_the_days_check_period[-days_check_period:]])    # [!] value of␣
↪3-days-moving prod
            returns_cumulative.append(math.prod([num + 1 for num in␣
↪returns_over_the_days_check_period]))
            if returns_over_the_days_check_period[0] < 0:                        ␣
↪                               # if first day's trade is <0
```

```
                if moving_prod >= 1:
                    end_date = df[df['article_code'] == i]['Date'][::-1].iloc[x␣
↪- 1]                          #># checks if the sign of 3d moving prod␣
↪changes
            else:                                                                ␣
↪                              # if first day's trade is >=0
                if moving_prod < 1:
                    end_date = df[df['article_code'] == i]['Date'][::-1].iloc[x␣
↪- 1]                          #># checks if the sign of 3d moving prod␣
↪changes
            if end_date == 0 & x == len(all_returns_for_X) - 1:
                end_date = df[df['article_code'] == i]['Date'][::-1].iloc[x]    ␣
↪                              # [!] sets end_date to the final date if␣
↪there was a consistent incr. or decline over the entire period
            if end_date != 0:
                break  # Exit the loop if end_date has been set

        days = (end_date - start_date + pd.Timedelta(days=1)).days


        print(start_date, "   -   ", end_date)
        #print((end_date - start_date + pd.Timedelta(days=1)).days)
        print("\n"*1, returns_cumulative)                                        ␣
↪                              # [!] list of cum product
        print(returns_over_the_days_check_period, "\n"*3)


    interday_counts_with_periods




    #return interday_counts_with_periods





consecutive_trades(stocks3, interday_counts[20:30], days_check_period = 3,␣
↪min_days = 7)
```

```
2023-10-12    -    2023-10-12

 [1.0061571125265394, 0.9962766953915325]
[0.006157112526539388, -0.009819954569715628]
```

2024-01-26 - 2024-01-30

[1.3560491015759146, 1.334502418474157, 1.3493975903614457, 1.3539527134222378]
[0.35604910157591463, -0.015889308931894552, 0.011161592276707655,
0.0033756715539798865]

2024-05-10 - 2024-05-21

[1.0111575533157082, 1.0118622996529538, 1.0013321044249923,
1.0182653741196832, 1.018044012081831, 1.015535073013949, 1.0095261081250984]
[0.011157553315708224, 0.0006969698588856765, -0.010406747273391859,
0.016910742819351343, -0.00021739130434783593, -0.0024644701389201495,
-0.005917043190853954 5]

2024-06-10 - 2024-06-20

[1.006410771240382, 1.008991299567923, 1.0238972041519834, 1.0339901317386968,
1.0327238488800745, 1.01978915541671]
[0.006410771240382029, 0.0025640905297155125, 0.01477307543726436,
0.009857364143378566, -0.0012246566188141017, -0.012524832729864266]

2024-01-22 - 2024-01-30

[1.07678696076695, 1.0407211944017778, 1.0399360579025698, 1.0816613786501372,
1.0475503903167644, 1.0433054041976222, 0.9673977717699066]
[0.07678696076695002, -0.03349387360660849, -0.0007544157872746426,
0.04012296759064449, -0.03153573660542619, -0.004052297778113223,
-0.07275686689852245]

2024-06-17 - 2024-06-25

[0.9693659588420922, 0.8956005950139866, 0.9253478781746187,
0.8727278288986893, 0.8720953321049107, 0.898532935670598, 0.8713204237383113,
0.8744627034889544]
[-0.0306340411579078, -0.07609650736675178, 0.03321489883575568,
-0.056865153654137135, -0.0007247354476787038, 0.03031503849685424,
-0.03028549188570062, 0.003606342356995862]

```
2023-08-16    -    2023-08-22

 [1.0372936532827313, 1.03993070772683, 1.0390576161154488, 1.0377293800806815,
1.0411453165948243, 1.0390359051376334]
[0.037293653282731265, 0.0025422448462431024, -0.0008395671027829898,
-0.0012783083576568544, 0.0032917411607613634, -0.0020260490284775834]



2023-09-14    -    2023-09-21

 [0.5335363645119472, 0.5463336392355067, 0.5427250257880072,
0.48530952953122924, 0.4838266327890156, 0.5023212655942612, 0.5010102117004506]
[-0.4664636354880528, 0.023985759124901973, -0.006605145992015338,
-0.10579113460525202, -0.003055568976042977, 0.03822574358636133,
-0.002609990823819852]



2023-11-10    -    2023-11-13

 [0.99764101878619, 0.9999645373102597, 1.0027404868044625]
[-0.002358981213809952, 0.0023290126210895323, 0.0027760479403295957]
```

[760]:

```python
# [13] Define a function to calculate the stabilization period
import math

def consecutive_trades(df, interday_counts, days_check_period = 3, min_days =␣
↪14):


    op = 0
    interday_counts_with_periods = pd.DataFrame(columns=['article_code',␣
↪'count', 'stabilization_period'])       # [!] empty df to fill in later

    limited_df = interday_counts

    for p, i in enumerate(limited_df['article_code']):                # [!]␣
↪loop over all article_codes w. >14 days period until the next one

        op = op + 1
        returns_over_the_days_check_period = []
        returns_cumulative = []
```

```
        all_returns_for_X = df[df['article_code'] == i]['returns'][::-1]

        start_date = 0      # placeholders
        end_date = 0        # placeholders

        for x, return_for_that_day in enumerate(all_returns_for_X):                                 # [!] loop over all dates for that particular article_code (in stocks df)
            start_date = df[df['article_code'] == i]['Date'][::-1].iloc[0]
            returns_over_the_days_check_period.append(return_for_that_day)
            moving_prod = math.prod([num + 1 for num in returns_over_the_days_check_period[-days_check_period:]])   # [!] value of 3-days-moving prod
            returns_cumulative.append(math.prod([num + 1 for num in returns_over_the_days_check_period]))
            if (returns_over_the_days_check_period[0] < 0):                                  # if first day's trade is <0
                if (moving_prod >= 1):
                    end_date = df[df['article_code'] == i]['Date'][::-1].iloc[x - 1]                             #># checks if the sign of 3d moving prod changes
            else:                                                                            # if first day's trade is >=0
                if (moving_prod < 1):
                    end_date = df[df['article_code'] == i]['Date'][::-1].iloc[x - 1]                             #># checks if the sign of 3d moving prod changes
            if (end_date == 0) & (x == len(all_returns_for_X) - 1):
                end_date = df[df['article_code'] == i]['Date'][::-1].iloc[x - 1]                                   # [!] sets end_date to the final date if there was a consistent incr. or decline over the entire period
            if (end_date != 0):
                break                                                                        # Exit the loop if end_date has been set

        if (len(df[df['article_code'] == i])) == 0:
            continue
        days = (end_date - start_date + pd.Timedelta(days=1)).days
        if (days < min_days):
            continue
        if (len(df[df['article_code'] == i])) == 0:
            continue
        print("[", op, "] - ", start_date, "   -   ", end_date)
        article_code = limited_df[limited_df['article_code'] == i]['article_code'].tolist()
        count = limited_df[limited_df['article_code'] == i]['count'].tolist()
```

```python
        interday_counts_with_periods.loc[i] = [article_code[0], count[0], days]

        #print(start_date, "    -    ", end_date)
        #print((end_date - start_date + pd.Timedelta(days=1)).days)
        #print("\n"*1, returns_cumulative)
                                     # [!] list of cum product
        #print(returns_over_the_days_check_period, "\n"*3)


    return interday_counts_with_periods
```

[761]:
```python
# [14] Run the function and create a df containing article_codes and their
 respective stabilization periods (in days)
stabilization_table = consecutive_trades(stocks3, interday_counts,
 days_check_period = 3, min_days = 7)
```

```
[ 6 ] -  2023-11-13    -    2023-11-29
[ 10 ] -  2024-03-27    -    2024-04-19
[ 11 ] -  2024-04-23    -    2024-05-02
[ 18 ] -  2023-08-02    -    2023-08-10
[ 23 ] -  2024-05-10    -    2024-05-21
[ 25 ] -  2024-06-10    -    2024-06-20
[ 26 ] -  2024-01-22    -    2024-01-30
[ 27 ] -  2024-06-17    -    2024-06-25
[ 28 ] -  2023-08-16    -    2023-08-22
[ 29 ] -  2023-09-14    -    2023-09-21
[ 32 ] -  2023-09-21    -    2023-10-03
[ 33 ] -  2023-01-19    -    2023-01-30
[ 35 ] -  2024-04-25    -    2024-05-23
[ 36 ] -  2024-05-29    -    2024-06-28
[ 38 ] -  2023-09-14    -    2023-09-21
[ 40 ] -  2022-10-04    -    2022-10-13
[ 42 ] -  2023-07-06    -    2023-07-14
[ 43 ] -  2023-08-31    -    2023-09-08
[ 45 ] -  2024-01-25    -    2024-02-07
[ 60 ] -  2024-01-03    -    2024-01-09
[ 66 ] -  2022-12-13    -    2022-12-19
[ 68 ] -  2024-03-26    -    2024-04-01
[ 69 ] -  2024-04-11    -    2024-04-19
[ 73 ] -  2024-05-28    -    2024-06-17
[ 74 ] -  2024-06-28    -    2024-07-15
[ 75 ] -  2024-02-01    -    2024-02-13
[ 76 ] -  2024-03-05    -    2024-03-13
[ 81 ] -  2023-08-10    -    2023-08-16
[ 91 ] -  2024-04-23    -    2024-04-29
[ 94 ] -  2023-08-28    -    2023-09-11
```

[100] - 2024-02-02 - 2024-03-06
[103] - 2023-03-15 - 2023-03-27
[113] - 2023-12-01 - 2023-12-20
[119] - 2023-03-24 - 2023-03-30
[120] - 2024-04-19 - 2024-04-29
[121] - 2023-10-12 - 2023-10-20
[124] - 2023-10-06 - 2023-11-02
[137] - 2024-02-09 - 2024-02-27
[139] - 2023-03-09 - 2023-03-23
[140] - 2023-08-16 - 2023-09-19
[145] - 2023-09-20 - 2023-09-29
[147] - 2023-11-22 - 2023-11-28
[150] - 2022-12-13 - 2022-12-23
[152] - 2023-10-19 - 2023-12-22
[153] - 2023-10-31 - 2023-11-13
[160] - 2024-04-25 - 2024-05-09
[165] - 2022-12-06 - 2022-12-19
[166] - 2023-01-04 - 2023-01-10
[173] - 2023-10-03 - 2023-10-11
[186] - 2023-12-19 - 2023-12-26
[199] - 2023-11-22 - 2023-11-28
[200] - 2023-12-14 - 2023-12-20
[201] - 2024-02-08 - 2024-02-20
[203] - 2024-05-09 - 2024-05-15
[206] - 2023-09-18 - 2023-09-27
[211] - 2024-04-29 - 2024-05-07
[217] - 2024-06-25 - 2024-07-02
[224] - 2024-01-25 - 2024-01-31
[227] - 2024-03-15 - 2024-03-22
[230] - 2024-07-01 - 2024-07-11
[232] - 2023-09-26 - 2023-10-03
[265] - 2023-08-08 - 2023-08-17
[268] - 2023-10-23 - 2023-11-02
[270] - 2023-10-17 - 2023-10-30
[274] - 2023-12-20 - 2023-12-27
[277] - 2024-03-18 - 2024-04-01
[282] - 2023-08-03 - 2023-08-18
[283] - 2023-09-06 - 2023-09-12
[292] - 2024-04-16 - 2024-04-22
[294] - 2024-05-20 - 2024-05-28
[295] - 2024-07-01 - 2024-07-10
[299] - 2023-09-29 - 2023-10-05
[300] - 2023-12-20 - 2023-12-28
[302] - 2023-12-05 - 2023-12-15
[304] - 2023-08-14 - 2023-08-23
[309] - 2024-01-09 - 2024-01-15
[314] - 2024-04-09 - 2024-04-17
[315] - 2024-05-07 - 2024-05-21

[316] - 2024-06-07 - 2024-06-25
[322] - 2023-12-29 - 2024-01-05
[324] - 2024-02-08 - 2024-02-16
[331] - 2023-10-19 - 2023-11-28
[334] - 2023-08-08 - 2023-08-16
[337] - 2023-10-02 - 2023-10-09
[351] - 2022-10-04 - 2022-10-10
[354] - 2023-02-02 - 2023-02-08
[362] - 2023-08-18 - 2023-08-24
[363] - 2023-09-06 - 2023-09-22
[365] - 2023-10-03 - 2023-10-13
[368] - 2023-11-02 - 2023-11-09
[372] - 2023-11-22 - 2023-11-28
[378] - 2024-02-01 - 2024-02-08
[379] - 2024-02-16 - 2024-02-27
[386] - 2024-07-05 - 2024-07-16
[389] - 2023-09-22 - 2023-10-04
[390] - 2024-04-02 - 2024-04-09
[395] - 2023-08-14 - 2023-08-22
[402] - 2024-03-20 - 2024-04-02
[411] - 2024-03-21 - 2024-04-02
[415] - 2023-09-14 - 2023-09-26
[418] - 2023-11-14 - 2023-11-29
[419] - 2023-12-12 - 2023-12-18
[423] - 2022-09-07 - 2022-09-13
[426] - 2022-09-28 - 2022-10-04
[441] - 2023-01-11 - 2023-01-17
[446] - 2023-03-09 - 2023-03-15
[452] - 2023-12-04 - 2023-12-11
[456] - 2024-05-23 - 2024-05-29
[458] - 2022-12-12 - 2022-12-23
[459] - 2023-03-09 - 2023-03-28
[461] - 2023-11-06 - 2023-11-13
[472] - 2024-03-07 - 2024-03-18
[474] - 2024-03-22 - 2024-04-01
[475] - 2024-05-23 - 2024-05-31
[480] - 2023-07-25 - 2023-08-04
[487] - 2024-04-26 - 2024-05-07
[489] - 2024-05-22 - 2024-05-30
[490] - 2024-06-24 - 2024-07-05
[493] - 2023-11-24 - 2023-12-20
[495] - 2023-07-26 - 2023-08-07
[503] - 2024-03-28 - 2024-04-08
[513] - 2024-04-02 - 2024-04-09
[514] - 2024-05-21 - 2024-05-30
[516] - 2024-06-24 - 2024-07-01
[519] - 2023-12-04 - 2023-12-13
[523] - 2024-05-29 - 2024-06-05

[526] - 2023-12-05 - 2023-12-13
[527] - 2024-03-27 - 2024-04-02
[534] - 2023-08-24 - 2023-08-30
[537] - 2023-10-19 - 2023-10-25
[543] - 2024-01-10 - 2024-01-19
[551] - 2024-03-28 - 2024-04-08
[555] - 2023-07-20 - 2023-07-28
[564] - 2023-01-26 - 2023-02-01
[571] - 2023-12-04 - 2023-12-15
[572] - 2024-02-28 - 2024-03-08
[574] - 2024-06-27 - 2024-07-08
[595] - 2023-08-10 - 2023-08-16
[611] - 2023-10-12 - 2023-10-18
[614] - 2024-01-31 - 2024-02-06
[632] - 2023-03-02 - 2023-03-09
[639] - 2023-05-05 - 2023-05-15
[645] - 2024-06-18 - 2024-06-24
[646] - 2024-06-28 - 2024-07-09
[648] - 2023-10-13 - 2023-10-23
[649] - 2023-11-13 - 2023-12-11
[651] - 2024-01-16 - 2024-01-25
[657] - 2024-04-16 - 2024-04-22
[659] - 2023-07-25 - 2023-08-02
[661] - 2023-10-17 - 2023-10-26
[663] - 2024-01-22 - 2024-01-30
[666] - 2023-12-05 - 2023-12-12
[668] - 2024-01-22 - 2024-02-01
[670] - 2024-05-07 - 2024-05-17
[674] - 2023-11-15 - 2023-11-21
[675] - 2023-09-29 - 2023-10-06
[683] - 2023-07-25 - 2023-07-31
[705] - 2023-11-22 - 2023-12-05
[706] - 2024-05-06 - 2024-05-14
[708] - 2023-07-14 - 2023-07-20
[717] - 2024-02-15 - 2024-02-21
[748] - 2023-03-10 - 2023-03-22
[754] - 2023-10-19 - 2023-11-06
[755] - 2024-02-26 - 2024-05-09
[763] - 2022-11-02 - 2022-11-16
[776] - 2024-01-09 - 2024-01-19
[790] - 2023-01-11 - 2023-01-17
[792] - 2023-01-26 - 2023-02-01
[798] - 2023-08-31 - 2023-09-07
[806] - 2023-07-06 - 2023-07-17
[809] - 2023-10-03 - 2023-10-09
[820] - 2023-11-21 - 2023-12-18
[834] - 2023-09-28 - 2023-10-05
[839] - 2023-12-19 - 2024-01-03

[840] - 2024-01-05 - 2024-01-18
[841] - 2024-03-11 - 2024-03-21
[847] - 2024-05-08 - 2024-05-20
[855] - 2022-09-20 - 2022-09-26
[859] - 2022-11-22 - 2022-11-29
[861] - 2023-03-08 - 2023-03-23
[868] - 2023-06-21 - 2023-06-28
[870] - 2023-08-01 - 2023-08-11
[879] - 2023-11-21 - 2023-11-29
[881] - 2023-12-14 - 2023-12-21
[887] - 2024-02-16 - 2024-02-26
[890] - 2024-03-19 - 2024-03-26
[897] - 2024-05-15 - 2024-05-30
[901] - 2024-06-17 - 2024-06-24
[902] - 2024-07-08 - 2024-07-15
[908] - 2024-02-13 - 2024-02-21
[914] - 2023-08-02 - 2023-08-14
[922] - 2022-10-04 - 2022-10-17
[925] - 2023-07-21 - 2023-07-28
[935] - 2023-08-10 - 2023-08-24
[938] - 2023-11-01 - 2023-11-07
[941] - 2023-11-15 - 2023-11-24
[949] - 2022-08-02 - 2022-08-08
[963] - 2022-12-09 - 2022-12-19
[964] - 2022-12-21 - 2022-12-28
[975] - 2023-02-22 - 2023-03-02
[992] - 2023-09-01 - 2023-09-07
[993] - 2023-09-25 - 2023-10-03
[1000] - 2023-12-29 - 2024-01-04
[1005] - 2024-02-15 - 2024-02-22
[1007] - 2024-03-28 - 2024-04-04
[1021] - 2023-11-08 - 2023-11-15
[1029] - 2024-05-01 - 2024-05-07
[1034] - 2024-05-15 - 2024-05-21
[1051] - 2023-08-10 - 2023-08-21
[1061] - 2022-08-17 - 2022-08-24
[1066] - 2023-04-04 - 2023-04-10
[1086] - 2023-08-01 - 2023-08-07
[1089] - 2023-09-06 - 2023-09-13
[1092] - 2024-02-06 - 2024-03-07
[1096] - 2024-05-16 - 2024-05-22
[1097] - 2024-03-14 - 2024-04-02
[1099] - 2024-04-08 - 2024-04-22
[1102] - 2023-07-20 - 2023-08-15
[1104] - 2023-09-15 - 2023-10-03
[1105] - 2023-10-12 - 2023-10-19
[1109] - 2024-06-13 - 2024-06-24
[1111] - 2023-01-19 - 2023-01-25

[1112] - 2022-12-02 - 2022-12-09
[1115] - 2023-01-17 - 2023-01-30
[1117] - 2023-07-13 - 2023-07-26
[1124] - 2024-05-15 - 2024-06-07
[1150] - 2024-05-20 - 2024-06-06
[1151] - 2023-08-23 - 2023-08-31
[1152] - 2023-10-04 - 2023-10-10
[1157] - 2023-12-29 - 2024-01-09
[1159] - 2024-03-20 - 2024-03-26
[1175] - 2024-02-07 - 2024-02-21
[1180] - 2024-01-11 - 2024-01-22
[1185] - 2024-07-11 - 2024-07-23
[1191] - 2024-03-21 - 2024-04-22
[1200] - 2023-10-27 - 2023-11-28
[1202] - 2024-01-19 - 2024-01-29
[1205] - 2024-04-23 - 2024-05-01
[1224] - 2023-10-24 - 2023-11-24
[1225] - 2024-02-16 - 2024-03-01
[1226] - 2023-07-31 - 2023-08-17
[1234] - 2023-11-14 - 2023-11-20
[1243] - 2024-04-16 - 2024-04-23
[1260] - 2023-07-25 - 2023-07-31
[1261] - 2024-01-24 - 2024-02-22
[1263] - 2023-09-15 - 2023-10-13
[1264] - 2023-10-20 - 2023-11-10
[1265] - 2024-01-18 - 2024-01-25
[1267] - 2024-04-22 - 2024-05-16
[1268] - 2022-12-21 - 2022-12-27
[1287] - 2024-01-18 - 2024-01-24
[1289] - 2023-11-30 - 2023-12-19
[1291] - 2024-03-14 - 2024-04-09
[1300] - 2024-04-22 - 2024-05-01
[1338] - 2023-11-16 - 2023-11-30
[1382] - 2024-05-16 - 2024-05-27
[1384] - 2023-08-10 - 2023-08-17
[1392] - 2024-07-02 - 2024-07-16
[1411] - 2023-08-22 - 2023-08-30
[1457] - 2024-06-12 - 2024-06-28
[1458] - 2022-10-17 - 2022-10-24
[1459] - 2023-09-20 - 2023-10-09
[1460] - 2023-10-11 - 2023-10-25
[1461] - 2023-11-14 - 2023-11-24
[1470] - 2023-04-13 - 2023-04-20
[1472] - 2023-06-26 - 2023-07-07
[1474] - 2023-08-21 - 2023-08-28
[1476] - 2023-09-13 - 2023-09-19
[1484] - 2024-05-10 - 2024-05-16
[1486] - 2023-06-20 - 2023-06-27

[1488] - 2023-09-25 - 2023-10-03
[1491] - 2024-02-29 - 2024-03-11
[1492] - 2024-05-13 - 2024-05-20
[1515] - 2023-08-31 - 2023-09-06
[1533] - 2023-11-30 - 2023-12-11
[1583] - 2024-03-15 - 2024-03-21
[1593] - 2024-02-14 - 2024-02-23
[1648] - 2024-02-02 - 2024-02-13
[1689] - 2023-08-08 - 2023-10-09
[1690] - 2023-09-15 - 2023-09-26
[1691] - 2023-10-12 - 2023-10-18
[1703] - 2023-09-22 - 2023-10-05
[1730] - 2023-11-30 - 2023-12-07
[1737] - 2024-03-12 - 2024-03-22
[1738] - 2024-04-08 - 2024-04-15
[1743] - 2023-09-28 - 2023-10-10
[1749] - 2023-12-21 - 2023-12-27
[1750] - 2024-01-09 - 2024-01-23
[1752] - 2024-02-26 - 2024-03-06
[1756] - 2024-04-10 - 2024-04-26
[1762] - 2024-01-10 - 2024-02-02
[1763] - 2024-02-06 - 2024-02-12
[1764] - 2024-02-16 - 2024-02-27
[1765] - 2024-06-03 - 2024-06-12
[1768] - 2023-10-11 - 2023-10-18
[1769] - 2023-11-08 - 2023-11-20
[1795] - 2023-11-08 - 2023-11-14
[1872] - 2024-04-30 - 2024-05-06
[1882] - 2024-06-04 - 2024-06-11
[1884] - 2023-09-12 - 2023-09-18
[1887] - 2023-12-05 - 2023-12-21
[1889] - 2024-03-07 - 2024-03-18
[1891] - 2023-07-28 - 2023-08-04
[1894] - 2023-10-06 - 2023-10-13
[1896] - 2023-11-09 - 2023-11-16
[1897] - 2024-01-17 - 2024-01-24
[1911] - 2024-07-09 - 2024-07-18
[1912] - 2022-08-02 - 2022-08-15
[1914] - 2022-09-02 - 2022-09-12
[1929] - 2024-01-26 - 2024-02-02
[1931] - 2024-03-26 - 2024-04-03
[1938] - 2022-10-04 - 2022-10-11
[1944] - 2023-01-18 - 2023-01-25
[1952] - 2023-05-18 - 2023-05-24
[1956] - 2023-07-10 - 2023-07-18
[1958] - 2023-08-10 - 2023-08-16
[1970] - 2024-02-27 - 2024-03-04
[1973] - 2024-05-10 - 2024-05-16

[1983]	-	2023-12-18	-	2023-12-27
[1984]	-	2024-02-13	-	2024-02-22
[1989]	-	2024-05-03	-	2024-05-13
[1991]	-	2022-08-01	-	2022-08-15
[1992]	-	2022-09-19	-	2022-09-27
[1993]	-	2023-01-11	-	2023-01-19
[1994]	-	2023-03-27	-	2023-04-04
[1997]	-	2023-09-18	-	2023-09-26
[2000]	-	2023-10-17	-	2023-10-24
[2009]	-	2022-09-01	-	2022-09-09
[2010]	-	2022-10-04	-	2022-10-11
[2012]	-	2023-02-01	-	2023-02-08
[2018]	-	2023-04-04	-	2023-04-17
[2020]	-	2023-05-12	-	2023-05-18
[2032]	-	2024-04-25	-	2024-05-01
[2035]	-	2024-05-29	-	2024-06-05
[2036]	-	2024-06-11	-	2024-06-18
[2057]	-	2023-11-14	-	2023-11-20
[2061]	-	2024-05-22	-	2024-06-04
[2063]	-	2023-08-01	-	2023-08-10
[2064]	-	2023-08-23	-	2023-08-31
[2066]	-	2023-11-14	-	2023-11-29
[2067]	-	2024-02-15	-	2024-03-06
[2071]	-	2023-08-14	-	2023-08-24
[2072]	-	2023-08-30	-	2023-09-06
[2085]	-	2023-08-02	-	2023-08-09
[2087]	-	2024-04-24	-	2024-05-01
[2088]	-	2022-08-03	-	2022-08-09
[2089]	-	2022-08-03	-	2022-08-12
[2092]	-	2023-09-25	-	2023-10-02
[2094]	-	2023-10-26	-	2023-11-01
[2110]	-	2024-01-16	-	2024-01-22
[2126]	-	2024-01-10	-	2024-01-19
[2138]	-	2023-08-25	-	2023-09-01
[2139]	-	2023-09-05	-	2023-09-13
[2144]	-	2023-10-03	-	2023-10-12
[2168]	-	2024-02-29	-	2024-03-06
[2171]	-	2023-09-12	-	2023-09-20
[2173]	-	2023-10-11	-	2023-10-18
[2178]	-	2024-04-17	-	2024-04-25
[2181]	-	2023-09-12	-	2023-09-20
[2188]	-	2024-03-28	-	2024-04-03
[2229]	-	2024-02-08	-	2024-02-15
[2263]	-	2023-10-19	-	2023-10-31
[2267]	-	2023-09-07	-	2023-09-15
[2269]	-	2023-10-10	-	2023-10-19
[2270]	-	2023-11-01	-	2023-11-15
[2280]	-	2023-11-14	-	2023-11-20

[2281] - 2023-12-19 - 2023-12-26
[2282] - 2023-12-28 - 2024-01-04
[2284] - 2024-01-10 - 2024-01-26
[2286] - 2024-02-28 - 2024-03-07
[2287] - 2024-04-30 - 2024-05-07
[2291] - 2023-07-21 - 2023-07-31
[2295] - 2024-05-16 - 2024-05-22
[2298] - 2023-09-13 - 2023-11-28
[2305] - 2023-08-09 - 2023-08-17
[2307] - 2023-12-11 - 2023-12-18
[2310] - 2023-10-24 - 2023-11-01
[2312] - 2023-10-13 - 2023-10-20
[2314] - 2024-01-31 - 2024-02-08
[2316] - 2024-04-23 - 2024-04-29
[2317] - 2024-05-20 - 2024-05-27
[2325] - 2023-09-18 - 2023-09-25
[2326] - 2023-10-05 - 2023-10-12
[2329] - 2024-06-26 - 2024-07-05
[2330] - 2023-10-17 - 2023-10-24
[2338] - 2024-06-20 - 2024-07-02
[2344] - 2023-08-18 - 2023-08-24
[2367] - 2023-07-31 - 2023-08-08
[2372] - 2023-08-15 - 2023-08-23
[2379] - 2023-08-10 - 2023-08-18
[2380] - 2023-09-05 - 2023-09-11
[2399] - 2023-08-29 - 2023-09-04
[2433] - 2024-02-01 - 2024-02-09
[2476] - 2024-06-04 - 2024-06-10
[2479] - 2023-10-25 - 2023-10-31
[2484] - 2024-04-18 - 2024-04-25
[2485] - 2023-08-02 - 2023-08-11
[2487] - 2024-03-13 - 2024-03-20
[2492] - 2023-08-09 - 2023-08-18
[2499] - 2023-11-21 - 2023-11-30
[2505] - 2024-03-28 - 2024-04-11
[2513] - 2024-02-28 - 2024-03-05
[2514] - 2023-09-29 - 2023-10-05
[2516] - 2022-09-07 - 2022-09-20
[2518] - 2023-09-19 - 2023-09-28
[2519] - 2024-01-31 - 2024-02-07
[2521] - 2024-05-28 - 2024-06-06
[2526] - 2024-04-09 - 2024-04-15
[2529] - 2023-10-05 - 2023-10-11
[2534] - 2024-04-02 - 2024-04-08
[2535] - 2023-07-03 - 2023-07-11
[2542] - 2023-11-22 - 2023-11-30
[2543] - 2024-01-17 - 2024-01-30
[2548] - 2023-08-22 - 2023-08-28

[2552] - 2024-04-04 - 2024-04-12
[2557] - 2023-09-12 - 2023-09-18
[2558] - 2023-10-05 - 2023-10-17
[2571] - 2024-01-18 - 2024-01-24
[2579] - 2024-03-14 - 2024-03-27
[2581] - 2024-07-05 - 2024-07-11
[2582] - 2023-10-06 - 2023-10-31
[2583] - 2023-11-09 - 2023-11-23
[2585] - 2023-07-27 - 2023-08-07
[2586] - 2024-05-16 - 2024-05-22
[2594] - 2023-11-02 - 2023-11-20
[2596] - 2023-09-14 - 2023-10-05
[2601] - 2024-03-25 - 2024-04-08
[2604] - 2024-05-09 - 2024-05-15
[2621] - 2023-12-12 - 2023-12-27
[2623] - 2024-07-05 - 2024-07-18
[2624] - 2024-01-10 - 2024-01-24
[2635] - 2023-11-08 - 2023-11-14
[2643] - 2024-01-23 - 2024-01-29
[2662] - 2024-04-03 - 2024-04-10
[2664] - 2023-10-03 - 2023-10-16
[2667] - 2024-03-26 - 2024-04-16
[2669] - 2023-10-27 - 2023-11-08
[2672] - 2024-03-13 - 2024-03-19
[2678] - 2024-06-12 - 2024-06-19
[2680] - 2023-07-20 - 2023-07-28
[2685] - 2023-10-24 - 2023-11-03
[2692] - 2023-10-13 - 2023-10-25
[2694] - 2023-09-11 - 2023-09-18
[2697] - 2023-09-21 - 2023-09-27
[2700] - 2023-11-08 - 2023-11-16
[2704] - 2024-06-11 - 2024-07-03
[2707] - 2023-10-02 - 2023-10-11
[2711] - 2023-07-26 - 2023-08-15
[2716] - 2024-01-29 - 2024-02-05
[2717] - 2024-04-11 - 2024-04-18
[2722] - 2023-10-31 - 2023-11-07
[2723] - 2024-02-27 - 2024-03-04
[2744] - 2023-12-28 - 2024-01-08
[2748] - 2023-08-01 - 2023-08-08
[2750] - 2023-10-10 - 2023-10-23
[2757] - 2024-01-03 - 2024-01-09
[2760] - 2024-04-16 - 2024-04-23
[2762] - 2024-05-17 - 2024-05-29
[2777] - 2023-11-22 - 2023-11-28
[2786] - 2024-07-05 - 2024-07-16
[2787] - 2024-06-26 - 2024-07-05
[2790] - 2023-07-27 - 2023-08-02

[2793]	-	2023-09-20	-	2023-10-03
[2802]	-	2024-06-17	-	2024-06-24
[2806]	-	2023-08-02	-	2023-08-08
[2815]	-	2024-03-22	-	2024-04-01
[2816]	-	2024-05-07	-	2024-05-15
[2824]	-	2023-08-29	-	2023-09-05
[2829]	-	2023-11-13	-	2023-11-20
[2837]	-	2023-08-23	-	2023-08-30
[2850]	-	2023-08-17	-	2023-08-24
[2852]	-	2023-10-02	-	2023-10-13
[2857]	-	2023-12-21	-	2023-12-27
[2858]	-	2023-12-29	-	2024-01-04
[2859]	-	2024-02-09	-	2024-02-16
[2872]	-	2023-12-15	-	2023-12-21
[2873]	-	2024-05-28	-	2024-06-11
[2874]	-	2024-04-15	-	2024-04-22
[2878]	-	2023-08-14	-	2023-08-22
[2886]	-	2022-09-26	-	2022-10-04
[2887]	-	2022-10-31	-	2022-11-08
[2891]	-	2022-12-27	-	2023-01-05
[2905]	-	2023-05-15	-	2023-05-31
[2911]	-	2023-07-20	-	2023-07-31
[2912]	-	2023-08-14	-	2023-08-21
[2917]	-	2023-09-13	-	2023-09-19
[2919]	-	2023-09-29	-	2023-10-06
[2923]	-	2023-11-15	-	2023-11-27
[2943]	-	2024-05-21	-	2024-05-30
[2947]	-	2024-06-21	-	2024-06-28
[2982]	-	2024-03-25	-	2024-04-02
[3016]	-	2023-10-09	-	2023-10-16
[3018]	-	2023-08-03	-	2023-08-21
[3026]	-	2023-11-02	-	2023-11-13
[3028]	-	2024-01-09	-	2024-01-16
[3033]	-	2024-03-01	-	2024-03-07
[3039]	-	2024-03-26	-	2024-04-01
[3045]	-	2023-12-13	-	2023-12-20
[3055]	-	2023-08-14	-	2023-08-24
[3064]	-	2023-09-27	-	2023-10-03
[3067]	-	2023-11-09	-	2023-11-15
[3075]	-	2024-06-26	-	2024-07-10
[3077]	-	2023-09-25	-	2023-10-03
[3078]	-	2023-08-01	-	2023-08-08
[3083]	-	2023-02-01	-	2023-02-13
[3084]	-	2023-08-04	-	2023-08-15
[3085]	-	2023-09-05	-	2023-09-18
[3092]	-	2022-10-19	-	2022-10-27
[3100]	-	2023-05-25	-	2023-06-02
[3101]	-	2023-06-20	-	2023-06-26

[3103] - 2023-08-10 - 2023-08-21
[3104] - 2023-11-14 - 2023-11-29
[3106] - 2024-03-27 - 2024-04-02
[3114] - 2024-05-31 - 2024-06-07
[3116] - 2023-09-12 - 2023-10-04
[3117] - 2023-10-12 - 2023-10-27
[3135] - 2023-09-20 - 2023-09-28
[3137] - 2023-11-08 - 2023-11-15
[3138] - 2024-02-13 - 2024-02-22
[3141] - 2024-01-31 - 2024-02-09
[3145] - 2023-07-04 - 2023-07-11
[3148] - 2024-07-02 - 2024-07-10
[3150] - 2023-08-28 - 2023-09-14
[3158] - 2023-12-28 - 2024-01-04
[3167] - 2023-12-27 - 2024-01-04
[3168] - 2024-01-26 - 2024-02-16
[3169] - 2024-05-28 - 2024-06-04
[3173] - 2022-08-11 - 2022-08-19
[3176] - 2023-11-15 - 2023-11-22
[3177] - 2023-12-28 - 2024-01-03
[3184] - 2024-04-09 - 2024-04-17
[3187] - 2024-05-16 - 2024-05-24
[3196] - 2023-09-27 - 2023-10-04
[3200] - 2024-02-15 - 2024-03-06
[3201] - 2024-05-07 - 2024-05-17
[3209] - 2023-10-10 - 2023-10-17
[3211] - 2023-08-14 - 2023-08-21
[3215] - 2023-10-24 - 2023-10-30
[3216] - 2023-11-14 - 2023-11-22
[3232] - 2023-09-15 - 2023-09-26
[3233] - 2023-10-12 - 2023-11-02
[3235] - 2024-06-11 - 2024-06-18
[3239] - 2022-11-10 - 2022-11-16
[3243] - 2023-02-03 - 2023-02-15
[3247] - 2023-12-01 - 2023-12-13
[3250] - 2024-05-08 - 2024-05-20
[3253] - 2023-08-22 - 2023-08-28
[3254] - 2022-09-21 - 2022-09-27
[3258] - 2023-04-28 - 2023-05-10
[3259] - 2023-07-21 - 2023-07-27
[3263] - 2023-12-05 - 2023-12-11
[3271] - 2024-05-10 - 2024-05-17
[3276] - 2022-09-28 - 2022-10-21
[3280] - 2024-04-17 - 2024-04-24
[3282] - 2024-06-27 - 2024-07-12
[3284] - 2023-09-12 - 2023-09-18
[3287] - 2023-08-17 - 2023-09-11
[3293] - 2024-03-15 - 2024-03-26

[3299]	-	2024-06-07	-	2024-06-13
[3304]	-	2023-10-20	-	2023-11-06
[3315]	-	2022-10-11	-	2022-10-18
[3316]	-	2022-10-31	-	2022-11-07
[3322]	-	2023-01-26	-	2023-02-01
[3337]	-	2024-04-30	-	2024-05-07
[3338]	-	2024-07-03	-	2024-07-10
[3339]	-	2022-09-21	-	2022-09-29
[3342]	-	2023-12-20	-	2023-12-27
[3343]	-	2024-03-26	-	2024-04-11
[3348]	-	2024-05-30	-	2024-06-17
[3349]	-	2024-06-24	-	2024-07-02
[3350]	-	2024-05-30	-	2024-06-14
[3356]	-	2023-04-13	-	2023-04-21
[3357]	-	2023-07-13	-	2023-07-19
[3360]	-	2023-08-09	-	2023-08-29
[3361]	-	2023-09-05	-	2023-09-15
[3362]	-	2023-09-19	-	2023-09-26
[3375]	-	2023-12-04	-	2023-12-12
[3378]	-	2024-03-20	-	2024-03-26
[3380]	-	2023-12-04	-	2023-12-13
[3383]	-	2022-09-15	-	2022-09-21
[3391]	-	2024-07-11	-	2024-07-18
[3394]	-	2022-11-15	-	2022-11-28
[3395]	-	2022-12-20	-	2022-12-28
[3399]	-	2023-06-12	-	2023-06-21
[3421]	-	2023-08-01	-	2023-08-08
[3425]	-	2023-09-28	-	2023-10-11
[3426]	-	2023-08-14	-	2023-08-21
[3434]	-	2023-08-24	-	2023-08-31
[3437]	-	2023-12-12	-	2023-12-18
[3446]	-	2023-09-13	-	2023-09-19
[3448]	-	2024-04-19	-	2024-04-26
[3455]	-	2023-11-30	-	2023-12-13
[3463]	-	2023-11-13	-	2023-12-12
[3467]	-	2024-01-24	-	2024-01-30
[3474]	-	2024-03-28	-	2024-04-03
[3483]	-	2023-10-20	-	2023-10-30
[3495]	-	2024-01-04	-	2024-01-11
[3512]	-	2023-12-22	-	2023-12-28
[3516]	-	2024-04-11	-	2024-04-25
[3518]	-	2024-05-15	-	2024-05-30
[3536]	-	2024-02-16	-	2024-02-23
[3541]	-	2024-03-13	-	2024-03-20
[3548]	-	2024-07-11	-	2024-07-17
[3553]	-	2024-03-01	-	2024-03-11
[3555]	-	2023-12-21	-	2024-01-02
[3567]	-	2024-05-06	-	2024-05-14

```
[ 3592 ] -  2024-01-11    -    2024-01-23
[ 3593 ] -  2023-08-10    -    2023-08-16
[ 3603 ] -  2023-10-18    -    2023-10-30
[ 3609 ] -  2024-03-04    -    2024-03-11
[ 3610 ] -  2024-03-19    -    2024-03-26
[ 3612 ] -  2024-04-03    -    2024-04-23
[ 3614 ] -  2024-06-05    -    2024-06-19
[ 3615 ] -  2024-07-03    -    2024-07-10
[ 3617 ] -  2023-08-14    -    2023-08-30
[ 3620 ] -  2023-08-21    -    2023-08-28
[ 3627 ] -  2024-02-21    -    2024-02-27
[ 3635 ] -  2023-12-20    -    2024-01-04
[ 3636 ] -  2024-02-05    -    2024-02-16
```

[763]:
```
# [*] So, it seems we have a little over 600 observations (with a restriction␣
↪that min_days = 7)
stabilization_table
```

[763]:
```
                                  article_code  count  stabilization_period
100034886_2023-11-13  100034886_2023-11-13     15                    17
100034886_2024-03-27  100034886_2024-03-27     19                    24
100034886_2024-04-23  100034886_2024-04-23     23                    10
1001743_2023-08-02      1001743_2023-08-02     16                     9
1001743_2024-05-10      1001743_2024-05-10     18                    12
…                                        …      …                     …
8762043_2023-08-14      8762043_2023-08-14    122                    17
8999021_2023-08-21      8999021_2023-08-21     30                     8
8999021_2024-02-21      8999021_2024-02-21    112                     7
9756394_2023-12-20      9756394_2023-12-20     33                    16
9756394_2024-02-05      9756394_2024-02-05    124                    12

[619 rows x 3 columns]
```

[799]:
```
# [15] Preparing the articles data
# Filter articles data to keep only those company codes present in␣
↪stabilization_table
articles_4 = articles_3.copy()
articles_4 = articles_4[articles_4['article_code'].
↪isin(stabilization_table['article_code'])]
articles_4
```

[799]:
```
      company_code publication_date  \
1           100607       2023-09-14
3           100669       2022-10-04
5           100669       2023-07-06
6           100669       2023-08-31
8           100669       2024-01-25
```

```
…           …             …
3568     105712952       2023-09-26
3601     106265632       2023-10-17
3605     106265632       2023-12-20
3608     106265632       2024-03-18
3618     110299664       2023-10-18

                                                  title  \
1     US bank branch M&A activity muted with only 9 …
3     Fed's aggressive tightening continues to weigh…
5     US bank stocks log positive median return afte…
6     Citigroup's CFO Mark Mason reclaims spot as hi…
8     Press Release: First US Bancshares, Inc. Repor…
…                                                 …
3568  Cleantech Lithium Shares Drop on Shorter-Than-…
3601  Press Release: Pan American Energy Corp: Explo…
3605  Press Release: Pan American Announces $900,000…
3608  Press Release: Pan American Energy Collaborate…
3618  US bank capital offerings up sequentially in Q…

                                                article  group  \
1     US whole-bank M&A might have sputtered back to…      2
3     U.S. bank stocks continued to take a beating i…      3
5     US bank stocks recorded their first positive m…      3
6     After losing his position to Bank of America C…      3
8     (MORE TO FOLLOW) Dow Jones Newswires\nJanuary …      3
…                                                 …    …
3568  1015 GMT - Cleantech Lithium's much-anticipate…      2
3601    (MORE TO FOLLOW) Dow Jones Newswires\nOctobe…      3
3605    (MORE TO FOLLOW) Dow Jones Newswires\nDecemb…      3
3608    (MORE TO FOLLOW) Dow Jones Newswires\nMarch …      3
3618  The US banking industry recorded a sequential …      3

              article_code
1        100607_2023-09-14
3        100669_2022-10-04
5        100669_2023-07-06
6        100669_2023-08-31
8        100669_2024-01-25
…                      …
3568  105712952_2023-09-26
3601  106265632_2023-10-17
3605  106265632_2023-12-20
3608  106265632_2024-03-18
3618  110299664_2023-10-18

[619 rows x 6 columns]
```

[800]:

```python
# [16] Clean, tokenize, and lemmatize the article and title texts
import nltk
from nltk.corpus import stopwords, wordnet
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
import string
import re


# Downloading necessary NLTK data:
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()


# Define a function to clean, tokenize, and lemmatize the text
def clean_tokenize_lemmatize(text):

    text = text.lower()                                          # Convert to␣
 ↪lowercase
    # Remove financial amounts and dates
    text = re.sub(r'\b\d+(?:,\d{3})*(?:\.\d+)?\b', '', text)    # Removes␣
 ↪numbers and financial amounts
    text = re.sub(r'\b\d{1,2}/\d{1,2}/\d{2,4}\b', '', text)     # Removes dates␣
 ↪in formats like 12/31/2021
    text = re.sub(r'\b\d{1,2}.\d{1,2}.\d{2,4}\b', '', text)     # Removes dates␣
 ↪in formats like 12.31.2021
    text = re.sub(r'\b\d{4}-\d{2}-\d{2}\b', '', text)           # Removes dates␣
 ↪in formats like 2021-12-31
    text = re.sub(r'\b\d{2}-\d{2}-\d{4}\b', '', text)           # Removes dates␣
 ↪in formats like 12-31-2021



    text = text.translate(str.maketrans('', '', string.punctuation))    #␣
 ↪Remove punctuation
    words = word_tokenize(text)                                         #␣
 ↪Tokenize
    words = [word for word in words if word not in stop_words]          #␣
 ↪Remove stop words
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]   #␣
 ↪Lemmatize
```

```
    return lemmatized_words




# Apply:
articles_4['cleaned_title_tokens'] = articles_4['title'].
 ↪apply(clean_tokenize_lemmatize)
articles_4['cleaned_article_tokens'] = articles_4['article'].
 ↪apply(clean_tokenize_lemmatize)
articles_4
# Save the cleaned, tokenized, and lemmatized data
articles_4.to_csv('cleaned_tokenized_lemmatized_articles_4.csv', index=False)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\panov\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\panov\AppData\Roaming\nltk_data…
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\panov\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\panov\AppData\Roaming\nltk_data…
[nltk_data]   Package omw-1.4 is already up-to-date!
```

[802]:
```
# [17] Combine with the y variable
articles_4 = articles_4.merge(stabilization_table, on='article_code',␣
 ↪how='left')
articles_4
```

[802]:
```
     company_code publication_date  \
0          100607       2023-09-14
1          100669       2022-10-04
2          100669       2023-07-06
3          100669       2023-08-31
4          100669       2024-01-25
..            …               …
614     105712952       2023-09-26
615     106265632       2023-10-17
616     106265632       2023-12-20
617     106265632       2024-03-18
618     110299664       2023-10-18

                                                   title  \
```

```
0    US bank branch M&A activity muted with only 9 …
1    Fed's aggressive tightening continues to weigh…
2    US bank stocks log positive median return afte…
3    Citigroup's CFO Mark Mason reclaims spot as hi…
4    Press Release: First US Bancshares, Inc. Repor…
..                                                …
614  Cleantech Lithium Shares Drop on Shorter-Than-…
615  Press Release: Pan American Energy Corp: Explo…
616  Press Release: Pan American Announces $900,000…
617  Press Release: Pan American Energy Collaborate…
618  US bank capital offerings up sequentially in Q…

                                               article  group  \
0    US whole-bank M&A might have sputtered back to…      2
1    U.S. bank stocks continued to take a beating i…      3
2    US bank stocks recorded their first positive m…      3
3    After losing his position to Bank of America C…      3
4    (MORE TO FOLLOW) Dow Jones Newswires\nJanuary …      3
..                                                …    …
614  1015 GMT - Cleantech Lithium's much-anticipate…      2
615    (MORE TO FOLLOW) Dow Jones Newswires\nOctobe…      3
616    (MORE TO FOLLOW) Dow Jones Newswires\nDecemb…      3
617    (MORE TO FOLLOW) Dow Jones Newswires\nMarch …      3
618  The US banking industry recorded a sequential …      3

             article_code                               cleaned_title_tokens  \
0       100607_2023-09-14  [u, bank, branch, activity, muted, deal, far, …
1       100669_2022-10-04  [fed, aggressive, tightening, continues, weigh…
2       100669_2023-07-06  [u, bank, stock, log, positive, median, return…
3       100669_2023-08-31  [citigroups, cfo, mark, mason, reclaims, spot,…
4       100669_2024-01-25  [press, release, first, u, bancshares, inc, re…
..                    …                                                 …
614  105712952_2023-09-26  [cleantech, lithium, share, drop, shorterthane…
615  106265632_2023-10-17  [press, release, pan, american, energy, corp, …
616  106265632_2023-12-20  [press, release, pan, american, announces, cha…
617  106265632_2024-03-18  [press, release, pan, american, energy, collab…
618  110299664_2023-10-18  [u, bank, capital, offering, sequentially, q3,…

                               cleaned_article_tokens  count_x  \
0    [u, wholebank, might, sputtered, back, life, b…      226
1    [u, bank, stock, continued, take, beating, sep…      107
2    [u, bank, stock, recorded, first, positive, me…       40
3    [losing, position, bank, america, corp, alasta…       54
4    [follow, dow, jones, newswires, january, et, g…      131
..                                                …        …
614  [gmt, cleantech, lithium, muchanticipated, sco…      218
615  [follow, dow, jones, newswires, october, et, g…       11
```

```
616  [follow, dow, jones, newswires, december, et, …       10
617  [follow, dow, jones, newswires, march, et, gmt…       79
618  [u, banking, industry, recorded, sequential, i…      202

     stabilization_period_x  count_y  stabilization_period_y
0                         8      226                       8
1                        10      107                      10
2                         9       40                       9
3                         9       54                       9
4                        14      131                      14
..                      …        …                      …
614                       8      218                       8
615                      14       11                      14
616                       8       10                       8
617                      15       79                      15
618                      41      202                      41

[619 rows x 12 columns]
```

[875]:

```
# Random Forest Regression
# Random Forests are an ensemble learning method that combines multiple␣
↪decision trees to improve predictive performance.
# They handle high-dimensional data well and are less likely to overfit (big␣
↪lie, this model faired worse than all others)
# compared to single decision trees due to their use of bootstrapping and␣
↪feature randomness.

# n_estimators = 100: This parameter specifies the number of trees in the␣
↪forest. 100 is a standard balance b.w. performance/computational efficiency
# R2 maximized
```

[804]:

```
# [18] Model time
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# [18.1] Splitting the data:
X_title = articles_4['cleaned_title_tokens'].apply(lambda x: ' '.join(x))
X_article = articles_4['cleaned_article_tokens'].apply(lambda x: ' '.join(x))
y = articles_4['stabilization_period_x']

X_title_train, X_title_test, y_train, y_test = train_test_split(X_title, y,␣
↪test_size=0.3, random_state=42)
X_article_train, X_article_test, _, _ = train_test_split(X_article, y,␣
↪test_size=0.3, random_state=42)
```

[805]:
```
# [18.2] Transforming texts into numerical format:

vectorizer = CountVectorizer()

X_title_train_vec = vectorizer.fit_transform(X_title_train)
X_title_test_vec = vectorizer.transform(X_title_test)

X_article_train_vec = vectorizer.fit_transform(X_article_train)
X_article_test_vec = vectorizer.transform(X_article_test)
```

[806]:
```
# [18.3] Random forest training:
model_title = RandomForestRegressor(n_estimators = 100, random_state = 42)
model_article = RandomForestRegressor(n_estimators = 100, random_state = 42)

model_title.fit(X_title_train_vec, y_train)
model_article.fit(X_article_train_vec, y_train)
```

[806]: RandomForestRegressor(random_state=42)

[807]:
```
# [18.4] Run and check how accurate it is:
y_title_train_pred = model_title.predict(X_title_train_vec)
y_article_train_pred = model_article.predict(X_article_train_vec)

title_train_mse = mean_squared_error(y_train, y_title_train_pred)
article_train_mse = mean_squared_error(y_train, y_article_train_pred)

title_train_r2 = r2_score(y_train, y_title_train_pred)
article_train_r2 = r2_score(y_train, y_article_train_pred)

print(f"Title Model - Training MSE: {title_train_mse}, R2: {title_train_r2}")
print(f"Article Model - Training MSE: {article_train_mse}, R2:␣
↪{article_train_r2}")
```

```
Title Model - Training MSE: 8.234510940751127, R2: 0.830217417550581
Article Model - Training MSE: 17.11030703352929, R2: 0.6472125502586226
```

[808]:
```
# [*] Well, R2 are high for training, hopefully it will show similar results␣
↪for the test data:
y_title_test_pred = model_title.predict(X_title_test_vec)
y_article_test_pred = model_article.predict(X_article_test_vec)

title_test_mse = mean_squared_error(y_test, y_title_test_pred)
article_test_mse = mean_squared_error(y_test, y_article_test_pred)

title_test_r2 = r2_score(y_test, y_title_test_pred)
article_test_r2 = r2_score(y_test, y_article_test_pred)

print(f"Title Model - Test MSE: {title_test_mse}, R2: {title_test_r2}")
```

```
print(f"Article Model - Test MSE: {article_test_mse}, R2: {article_test_r2}")
```

```
Title Model - Test MSE: 55.320083607712796, R2: -0.11156235969658646
Article Model - Test MSE: 59.21577049870747, R2: -0.1898395174807328
```

[868]:
```
# [*] Literally couldn't have been worse
# Let's try optimizing the number of estimators and depth in our random forest
```

[873]:
```
# Optimized Random Forest (we use Grid Search)
# Hyperparameter tuning using Grid Search hoping for better performance␣
↪(spoiler: it's still bad)
# Use a wider range of parameters to Hail Mary a decent model (failed)

# n_estimators: 100 and now also 200 and 300
# max_depth: We tested values of 10, 20, 30 (and -) controls the depth of the␣
↪trees (more would overfit).
# min_samples_split / min_samples_leaf: control the min number of samples␣
↪required to split an internal node and
#     the min number of samples required to be at a leaf node, respectively. We␣
↪used values of 2, 5, and 10 to balance complexity/overfitting.
# bootstrap: both True and False
#                    (resampling with replacement from the original dataset)
# R2 maximized
```

[812]:
```
# [19] First, we'll TF-IDF transform:
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # Limiting to top 5000␣
↪features for computational efficiency (I don't understand this␣
↪recommendation, but OK)

X_title_train_tfidf = tfidf_vectorizer.fit_transform(X_title_train)
X_title_test_tfidf = tfidf_vectorizer.transform(X_title_test)

X_article_train_tfidf = tfidf_vectorizer.fit_transform(X_article_train)
X_article_test_tfidf = tfidf_vectorizer.transform(X_article_test)
```

[813]:
```
# [20] Hyperparameter Tuning for RandomForestRegressor:

param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
```

```
grid_search_title =␣
 ↪GridSearchCV(estimator=RandomForestRegressor(random_state=42),
                                 param_grid=param_grid,
                                 cv=5,
                                 n_jobs=-1,
                                 scoring='r2',
                                 verbose=2)
grid_search_article =␣
 ↪GridSearchCV(estimator=RandomForestRegressor(random_state=42),
                                   param_grid=param_grid,
                                   cv=5,
                                   n_jobs=-1,
                                   scoring='r2',
                                   verbose=2)

grid_search_title.fit(X_title_train_tfidf, y_train)
grid_search_article.fit(X_article_train_tfidf, y_train)

best_model_title = grid_search_title.best_estimator_
best_model_article = grid_search_article.best_estimator_
```

```
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
```

[814]:
```
# [21] Evaluate the model's accuracy on the training data:
y_title_train_pred = best_model_title.predict(X_title_train_tfidf)
y_article_train_pred = best_model_article.predict(X_article_train_tfidf)

title_train_mse = mean_squared_error(y_train, y_title_train_pred)
article_train_mse = mean_squared_error(y_train, y_article_train_pred)

title_train_r2 = r2_score(y_train, y_title_train_pred)
article_train_r2 = r2_score(y_train, y_article_train_pred)

print(f"Title Model - Training MSE: {title_train_mse}, R2: {title_train_r2}")
print(f"Article Model - Training MSE: {article_train_mse}, R2:␣
 ↪{article_train_r2}")
```

```
Title Model - Training MSE: 27.395336939143906, R2: 0.4351515122068025
Article Model - Training MSE: 30.84736101042253, R2: 0.3639762395383591
```

[815]:
```
# [22] Test again
y_title_test_pred = best_model_title.predict(X_title_test_tfidf)
y_article_test_pred = best_model_article.predict(X_article_test_tfidf)

title_test_mse = mean_squared_error(y_test, y_title_test_pred)
article_test_mse = mean_squared_error(y_test, y_article_test_pred)
```

```
title_test_r2 = r2_score(y_test, y_title_test_pred)
article_test_r2 = r2_score(y_test, y_article_test_pred)

print(f"Title Model - Test MSE: {title_test_mse}, R2: {title_test_r2}")
print(f"Article Model - Test MSE: {article_test_mse}, R2: {article_test_r2}")
```

```
Title Model - Test MSE: 53.652910239851266, R2: -0.07806336544443204
Article Model - Test MSE: 54.13424175633617, R2: -0.08773489812059942
```

[872]:
```
# [23] A big fail again
# [23] Let's try a different approach with Ridge Regression

# Ridge Regression
# Ridge Regression linear regression with regularization (prevents overfitting␣
↪by penalizing large coefficients)
# with L2 regularizatio the model adds the squared coefficients' values to the␣
↪loss function (that it tries to minimize during training)
# It is suitable for high-dimensional data like text in TF-IDF vectors (text␣
↪data transformed into numerical format)

# alpha of 0.1, 1.0, 10.0, and 100.0: This parameter controls the strength of␣
↪the regularization (bias vs. variance)
# R2 maximized
```

[825]:
```
# [23] A big fail again
# [23] Let's try a different approach with Ridge Regression

from sklearn.linear_model import Ridge


# Split
X_title = articles_4['cleaned_title_tokens'].apply(lambda x: ' '.join(x))
X_article = articles_4['cleaned_article_tokens'].apply(lambda x: ' '.join(x))
y = articles_4['stabilization_period_x']

X_title_train, X_title_test, y_train, y_test = train_test_split(X_title, y,␣
↪test_size=0.3, random_state=42)
X_article_train, X_article_test, _, _ = train_test_split(X_article, y,␣
↪test_size=0.3, random_state=42)


# Text transformation with TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features = 5000)                    ␣
↪  # Limiting to top 5000 features for computational efficiency

X_title_train_tfidf = tfidf_vectorizer.fit_transform(X_title_train)
X_title_test_tfidf = tfidf_vectorizer.transform(X_title_test)
```

```python
X_article_train_tfidf = tfidf_vectorizer.fit_transform(X_article_train)
X_article_test_tfidf = tfidf_vectorizer.transform(X_article_test)

# Train ridge model
ridge_model_title = Ridge()
ridge_model_article = Ridge()


param_grid = {'alpha': [0.1, 1.0, 10.0, 100.0]}
                        # Hyperparameter tuning with:
grid_search_title = GridSearchCV(ridge_model_title, param_grid, cv=5,
 scoring='r2')                           # ... Grid Search
grid_search_article = GridSearchCV(ridge_model_article, param_grid, cv=5,
 scoring='r2')                       # ... Grid Search

grid_search_title.fit(X_title_train_tfidf, y_train)
grid_search_article.fit(X_article_train_tfidf, y_train)

best_model_title = grid_search_title.best_estimator_
best_model_article = grid_search_article.best_estimator_


# Accuracy of training
y_title_train_pred = best_model_title.predict(X_title_train_tfidf)
y_article_train_pred = best_model_article.predict(X_article_train_tfidf)

title_train_mse = mean_squared_error(y_train, y_title_train_pred)
article_train_mse = mean_squared_error(y_train, y_article_train_pred)

title_train_r2 = r2_score(y_train, y_title_train_pred)
article_train_r2 = r2_score(y_train, y_article_train_pred)

print(f"Title Model - Training MSE: {title_train_mse}, R2: {title_train_r2}")
print(f"Article Model - Training MSE: {article_train_mse}, R2: 
 {article_train_r2}")


# Accuracy of test
y_title_test_pred = best_model_title.predict(X_title_test_tfidf)
y_article_test_pred = best_model_article.predict(X_article_test_tfidf)

title_test_mse = mean_squared_error(y_test, y_title_test_pred)
article_test_mse = mean_squared_error(y_test, y_article_test_pred)

title_test_r2 = r2_score(y_test, y_title_test_pred)
article_test_r2 = r2_score(y_test, y_article_test_pred)
```

```
print(f"Title Model - Test MSE: {title_test_mse}, R2: {title_test_r2}")
print(f"Article Model - Test MSE: {article_test_mse}, R2: {article_test_r2}")
```

```
Title Model - Training MSE: 40.95873160532772, R2: 0.1554957816145851
Article Model - Training MSE: 42.462716439476125, R2: 0.12448599476218158
Title Model - Test MSE: 49.04259095026146, R2: 0.014573106778408895
Article Model - Test MSE: 49.55915593750432, R2: 0.004193617834971186
```

[876]:
```
# final attempt with PyTorch Neural Network
# Neural Networks can capture complex patterns in the data due to their␣
↪non-linear nature and multiple layers.
# PyTorch provides flexibility and control over the model architecture,␣
↪allowing for experimentation with different network structures.

# Input Layer: 5,000 - how many TF-IDF vectors' features we have (higher or␣
↪lower didn't prove much better)
# Hidden Layers: Two hidden layers with 128 and 64 neurons using ReLU␣
↪activation functions (for complex patterns) to process the features
# Output Layer: A single neuron output layer for prediction of a single value
# Optimizer: Adam optimizer (handles large datasets and noisy data by adapting␣
↪the learning rate while training)
# Learning Rate: 0.001 - slower learning, but less swings

# criterion = nn.MSELoss() - Mean Square Error for our loss function
# epochs: 10 (tried other values, but none were much better) - the number of␣
↪times it iterates over the entire training data set
```

[859]:
```
# [*] Well, at least now the model just matched random guessing based on␣
↪averages method
# [24] Final attempt with PyTorch
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset


# [24] Prepare:
X_title = articles_4['cleaned_title_tokens'].apply(lambda x: ' '.join(x))
X_article = articles_4['cleaned_article_tokens'].apply(lambda x: ' '.join(x))
y = articles_4['stabilization_period_x']

X_title_train, X_title_test, y_train, y_test = train_test_split(X_title, y,␣
↪test_size=0.3, random_state=42)
X_article_train, X_article_test, _, _ = train_test_split(X_article, y,␣
↪test_size=0.3, random_state=42)
```

```
tfidf_vectorizer = TfidfVectorizer(max_features=5000)

X_title_train_tfidf = tfidf_vectorizer.fit_transform(X_title_train)
X_title_test_tfidf = tfidf_vectorizer.transform(X_title_test)

X_article_train_tfidf = tfidf_vectorizer.fit_transform(X_article_train)
X_article_test_tfidf = tfidf_vectorizer.transform(X_article_test)

# [24] Convert to pytorch tensos:
X_title_train_tfidf = torch.tensor(X_title_train_tfidf.toarray(), dtype=torch.
 ↪float32)
X_title_test_tfidf = torch.tensor(X_title_test_tfidf.toarray(), dtype=torch.
 ↪float32)

X_article_train_tfidf = torch.tensor(X_article_train_tfidf.toarray(),␣
 ↪dtype=torch.float32)
X_article_test_tfidf = torch.tensor(X_article_test_tfidf.toarray(), dtype=torch.
 ↪float32)

y_train = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)
y_test = torch.tensor(y_test.values, dtype=torch.float32).view(-1, 1)
```

[860]:
```
# [25] Setting up a neural network
class NeuralNetwork(nn.Module):
    def __init__(self, input_dim):
        super(NeuralNetwork, self).__init__()
        self.layer1 = nn.Linear(input_dim, 128)
        self.layer2 = nn.Linear(128, 64)
        self.layer3 = nn.Linear(64, 1)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.layer1(x))
        x = self.relu(self.layer2(x))
        x = self.layer3(x)
        return x
```

[861]:
```
# [26] Initialize the model, loss function, and optimizer
input_dim = X_title_train_tfidf.shape[1]

model_title = NeuralNetwork(input_dim)
model_article = NeuralNetwork(input_dim)

criterion = nn.MSELoss()
optimizer_title = optim.Adam(model_title.parameters(), lr=0.001)
optimizer_article = optim.Adam(model_article.parameters(), lr=0.001)
```

[862]:
```python
# [27] Dataloader preparations:
train_dataset_title = TensorDataset(X_title_train_tfidf, y_train)
test_dataset_title = TensorDataset(X_title_test_tfidf, y_test)

train_loader_title = DataLoader(train_dataset_title, batch_size=32,
 ↪shuffle=True)
test_loader_title = DataLoader(test_dataset_title, batch_size=32, shuffle=False)

train_dataset_article = TensorDataset(X_article_train_tfidf, y_train)
test_dataset_article = TensorDataset(X_article_test_tfidf, y_test)

train_loader_article = DataLoader(train_dataset_article, batch_size=32,
 ↪shuffle=True)
test_loader_article = DataLoader(test_dataset_article, batch_size=32,
 ↪shuffle=False)
```

[863]:
```python
# [28] Train:
def train_model(model, train_loader, criterion, optimizer, epochs=20):
    model.train()
    for epoch in range(epochs):
        running_loss = 0.0
        for inputs, targets in train_loader:
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, targets)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
        print(f"Epoch {epoch+1}, Loss: {running_loss/len(train_loader)}")

train_model(model_title, train_loader_title, criterion, optimizer_title,
 ↪epochs=10)
train_model(model_article, train_loader_article, criterion, optimizer_article,
 ↪epochs=10)
```

```
Epoch 1, Loss: 171.75585610525948
Epoch 2, Loss: 164.23278754098075
Epoch 3, Loss: 143.5751598903111
Epoch 4, Loss: 108.35115078517369
Epoch 5, Loss: 71.67182513645717
Epoch 6, Loss: 57.602366992405486
Epoch 7, Loss: 51.149588448660715
Epoch 8, Loss: 46.28866229738508
Epoch 9, Loss: 42.286061320986065
Epoch 10, Loss: 39.10341491018023
Epoch 1, Loss: 181.5685659136091
Epoch 2, Loss: 169.64338084629603
Epoch 3, Loss: 142.94645363943917
```

```
Epoch 4, Loss: 103.01326669965472
Epoch 5, Loss: 69.09234060559955
Epoch 6, Loss: 59.97426550728934
Epoch 7, Loss: 49.32490314756121
Epoch 8, Loss: 44.749057974134175
Epoch 9, Loss: 42.55326110976083
Epoch 10, Loss: 39.49230582373483
```

[866]:

```
# [29] Test:
def evaluate_model(model, test_loader, criterion):
    model.eval()
    test_loss = 0.0
    predictions, actuals = [], []
    with torch.no_grad():
        for inputs, targets in test_loader:
            outputs = model(inputs)
            loss = criterion(outputs, targets)
            test_loss += loss.item()
            predictions.append(outputs.numpy())
            actuals.append(targets.numpy())
    predictions = np.concatenate(predictions, axis=0)
    actuals = np.concatenate(actuals, axis=0)
    mse = mean_squared_error(actuals, predictions)
    r2 = r2_score(actuals, predictions)
    return mse, r2

title_test_mse, title_test_r2 = evaluate_model(model_title, test_loader_title,␣
↪criterion)
article_test_mse, article_test_r2 = evaluate_model(model_article,␣
↪test_loader_article, criterion)

print(f"Title Model - Test MSE: {title_test_mse}, R2: {title_test_r2}")
print(f"Article Model - Test MSE: {article_test_mse}, R2: {article_test_r2}")
```

```
Title Model - Test MSE: 50.76136016845703, R2: -0.019962700456879645
Article Model - Test MSE: 50.216182708740234, R2: -0.009008368016631518
```

[877]:

```
# [30] Conclusion:
# Too many features, too small a sample? (possible)
# Bad parameters (nah, we tested different variations, all were bad)
# Bad measurement of stabilization period (we feel our metric is pretty␣
↪reasonable, so 60/40 it's not this one)
# Noisy/duplicate text data? (very likely, almost guaranteed this is the case,␣
↪but does it cause issues like we had?)
# Should've included the entire sample with <7 days in between articles (would␣
↪increase the sample 6-fold but subject to news spill - bad compromise)
```

```
# Final: we can't predict the stabilization period using ML methods on tokenized and lemmatized news articles data
```