In [1]:
```
! pip install -q kaggle
```

In [2]:
```
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Out[2]:
```
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"ronggurmwp","key":"659addeda6fe20f4cdbb154d65d80c1
b"}'}
```

In [3]:
```
!rm -r ~/.kaggle
!mkdir ~/.kaggle
!mv ./kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
rm: cannot remove '/root/.kaggle': No such file or directory
```

In [4]:
```
! kaggle datasets download -d miguelaenlle/massive-stock-news-analysis-db-for-nlpba
```

```
Downloading massive-stock-news-analysis-db-for-nlpbacktests.zip to /content
 99% 208M/210M [00:06<00:00, 38.6MB/s]
100% 210M/210M [00:06<00:00, 36.2MB/s]
```

In [5]:
```
!unzip massive-stock-news-analysis-db-for-nlpbacktests.zip
```

```
Archive:  massive-stock-news-analysis-db-for-nlpbacktests.zip
  inflating: analyst_ratings_processed.csv
  inflating: raw_analyst_ratings.csv
  inflating: raw_partner_headlines.csv
```

EDA Dataset Berita Saham

In [6]:
```
# Import library
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import re
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

In [7]:
```
# Setup for VADER
nltk.download('vader_lexicon')
```

Out[7]:
```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

In [8]:
```
# Ubah CSV ke dataframe
df = pd.read_csv("analyst_ratings_processed.csv")
```

In [9]:
```
df.head()
```

Out[9]:

	Unnamed: 0	title	date	stock
0	0.0	Stocks That Hit 52-Week Highs On Friday	2020-06-05 10:30:00-04:00	A
1	1.0	Stocks That Hit 52-Week Highs On Wednesday	2020-06-03 10:45:00-04:00	A
2	2.0	71 Biggest Movers From Friday	2020-05-26 04:30:00-04:00	A
3	3.0	46 Stocks Moving In Friday's Mid-Day Session	2020-05-22 12:45:00-04:00	A
4	4.0	B of A Securities Maintains Neutral on Agilent...	2020-05-22 11:38:00-04:00	A

In [10]:
```
# Identifikasi informasi umum data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1400469 entries, 0 to 1400468
Data columns (total 4 columns):
 #   Column      Non-Null Count    Dtype  
---  ------      --------------    -----  
 0   Unnamed: 0  1399180 non-null  float64
 1   title       1400469 non-null  object 
 2   date        1399180 non-null  object 
 3   stock       1397891 non-null  object 
dtypes: float64(1), object(3)
memory usage: 42.7+ MB
```

Dari hasil identifikasi terhadap informasi umum data, diketahui bahwa terdapat 4 kolom data. Dari keempat kolom tersebut, kolom unnamed tidak relevan karena hanya menyatakan urutan. Oleh karena itu, kolom ini dapat dihapus pada proses selanjutnya.

In [11]:
```
# Hapus kolom Unnamed: 0
df = df.drop(columns=["Unnamed: 0"])
df
```

Out[11]:

	title	date	stock
0	Stocks That Hit 52-Week Highs On Friday	2020-06-05 10:30:00-04:00	A
1	Stocks That Hit 52-Week Highs On Wednesday	2020-06-03 10:45:00-04:00	A
2	71 Biggest Movers From Friday	2020-05-26 04:30:00-04:00	A
3	46 Stocks Moving In Friday's Mid-Day Session	2020-05-22 12:45:00-04:00	A
4	B of A Securities Maintains Neutral on Agilent...	2020-05-22 11:38:00-04:00	A
...	...	...	...
1400464	Top Narrow Based Indexes For August 29	2011-08-29 10:41:00-04:00	ZX
1400465	Recap: Wednesday's Top Percentage Gainers and ...	2011-06-22 16:44:00-04:00	ZX
1400466	UPDATE: Oppenheimer Color on China Zenix Auto ...	2011-06-21 08:26:00-04:00	ZX
1400467	Oppenheimer Initiates China Zenix At Outperfor...	2011-06-21 05:59:00-04:00	ZX
1400468	China Zenix Auto International Opens For Tradi...	2011-05-12 09:36:00-04:00	ZX

1400469 rows × 3 columns

Terdapat baris dengan nilai null pada kolom tertentu kecuali kolom title. Data akan dipasangkan dengan time-series harga saham sehingga kolom title, date, dan stock tidak

boleh bernilai null. Oleh karena itu, baris dengan nilai null harus dihilangkan pada proses selanjutnya.

In [12]:
```
# Hapus baris dengan nilai null
df = df.dropna()
df
```

Out[12]:

	title	date	stock
0	Stocks That Hit 52-Week Highs On Friday	2020-06-05 10:30:00-04:00	A
1	Stocks That Hit 52-Week Highs On Wednesday	2020-06-03 10:45:00-04:00	A
2	71 Biggest Movers From Friday	2020-05-26 04:30:00-04:00	A
3	46 Stocks Moving In Friday's Mid-Day Session	2020-05-22 12:45:00-04:00	A
4	B of A Securities Maintains Neutral on Agilent...	2020-05-22 11:38:00-04:00	A
...	...	...	...
1400464	Top Narrow Based Indexes For August 29	2011-08-29 10:41:00-04:00	ZX
1400465	Recap: Wednesday's Top Percentage Gainers and ...	2011-06-22 16:44:00-04:00	ZX
1400466	UPDATE: Oppenheimer Color on China Zenix Auto ...	2011-06-21 08:26:00-04:00	ZX
1400467	Oppenheimer Initiates China Zenix At Outperfor...	2011-06-21 05:59:00-04:00	ZX
1400468	China Zenix Auto International Opens For Tradi...	2011-05-12 09:36:00-04:00	ZX

1397891 rows × 3 columns

Kolom date yang menyatakan tanggal munculnya berita memiliki tipe objek. Tipe data kolom perlu diubah menjadi tipe datetime agar lebih mudah diproses.

In [39]:
```
import datetime

def convert_date(date_string):
    regex = r'(\d{4}-\d{2}-\d{2}) (.*)$'
    match = re.match(regex, date_string)
    if match:
        date_string = match.group(1)

        # Convert the date string to a datetime object.
        date = datetime.datetime.strptime(date_string, '%Y-%m-%d')

        return date
    else:
        raise ValueError('Invalid date string: {}'.format(date_string))


df['date'] = df['date'].apply(convert_date)
df
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-d211ebe1950d> in <cell line: 17>()
     15
     16
---> 17 df['date'] = df['date'].apply(convert_date)
     18 df

/usr/local/lib/python3.10/dist-packages/pandas/core/series.py in apply(self, func,
convert_dtype, args, **kwargs)
   4769         dtype: float64
   4770         """
-> 4771         return SeriesApply(self, func, convert_dtype, args, kwargs).apply
()
   4772
   4773     def _reduce(

/usr/local/lib/python3.10/dist-packages/pandas/core/apply.py in apply(self)
   1121
   1122         # self.f is Callable
-> 1123         return self.apply_standard()
   1124
   1125     def agg(self):

/usr/local/lib/python3.10/dist-packages/pandas/core/apply.py in apply_standard(sel
f)
   1172             else:
   1173                 values = obj.astype(object)._values
-> 1174                 mapped = lib.map_infer(
   1175                     values,
   1176                     f,

/usr/local/lib/python3.10/dist-packages/pandas/_libs/lib.pyx in pandas._libs.lib.m
ap_infer()

<ipython-input-39-d211ebe1950d> in convert_date(date_string)
      3 def convert_date(date_string):
      4     regex = r'(\d{4}-\d{2}-\d{2}) (.*)$'
----> 5     match = re.match(regex, date_string)
      6     if match:
      7         date_string = match.group(1)

/usr/lib/python3.10/re.py in match(pattern, string, flags)
    188     """Try to apply the pattern at the start of the string, returning
    189     a Match object, or None if no match was found."""
--> 190     return _compile(pattern, flags).match(string)
    191
    192 def fullmatch(pattern, string, flags=0):

TypeError: expected string or bytes-like object
```

In [ ]:

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1397891 entries, 0 to 1400468
Data columns (total 3 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   title   1397891 non-null  object
 1   date    1397891 non-null  datetime64[ns]
 2   stock   1397891 non-null  object
dtypes: datetime64[ns](1), object(2)
memory usage: 42.7+ MB
```

In [15]:
```
# Hitung jumlah berita berdasarkan stock ticker symbol
df_stock_count = df.groupby("stock")["stock"].count().sort_values(ascending=False)
df_stock_count
```

Out[15]:
```
stock
MRK      3334
MS       3242
MU       3144
NVDA     3133
QQQ      3100
         ...
HAWKB       1
GYEN        1
GWX         1
GWL         1
FOIL        1
Name: stock, Length: 6192, dtype: int64
```

Dari hasil di atas, diketahui bahwa 6192 saham berbeda dalam dataset. Setiap saham memiliki jumlah berita yang beragam. Agar sentimen nantinya dapat digunakan pada analisis time series saham dengan baik, diperlukan jumlah berita yang cukup banyak. Oleh karena itu, perlu dihapus data dengan jumlah berita sedikit.

In [16]:
```
# Cari saham yang jumlah beritanya kurang dari 2000
low_news_count_stocks = df_stock_count[df_stock_count < 2000].index.tolist()

# Hapus baris dengan saham yang jumlah beritanya kurang dari 2000
df_hn = df[~df["stock"].isin(low_news_count_stocks)]
df_stock_count = df_hn.groupby("stock")["stock"].count().sort_values(ascending=Fals
df_stock_count
```

Out[16]:
```
stock
MRK     3334
MS      3242
MU      3144
NVDA    3133
QQQ     3100
M       3078
EBAY    3021
NFLX    3009
GILD    2969
VZ      2937
DAL     2929
JNJ     2927
QCOM    2915
BABA    2820
KO      2785
ORCL    2695
FDX     2630
HD      2617
WFC     2612
BBRY    2570
BMY     2517
JCP     2498
LLY     2401
AGN     2400
CMG     2365
CAT     2299
GPRO    2292
TWX     2271
CHK     2270
FSLR    2232
NOK     2217
P       2216
LMT     2213
MCD     2208
MA      2188
EA      2166
FCX     2162
EWU     2152
GPS     2133
PEP     2131
XLF     2113
GRPN    2093
TM      2065
GLD     2034
EWP     2027
HAL     2021
LOW     2012
Name: stock, dtype: int64
```

Filter saham berdasarkan jumlah beritanya berhasil mengurangi jumlah kandidat saham yang akan dianalisis secara signifikan. Untuk kasus ini, akan diambil sampel beberapa saham perusahaan teknologi yaitu NVDA (Nvidia), EBAY (Ebay), NFLX (Netflix), dan ORCL (Oracle)

In [17]:
```
df_nvda = df[df["stock"] == "NVDA"]
df_ebay = df[df["stock"] == "EBAY"]
df_nflx = df[df["stock"] == "NFLX"]
df_orcl = df[df["stock"] == "ORCL"]
```

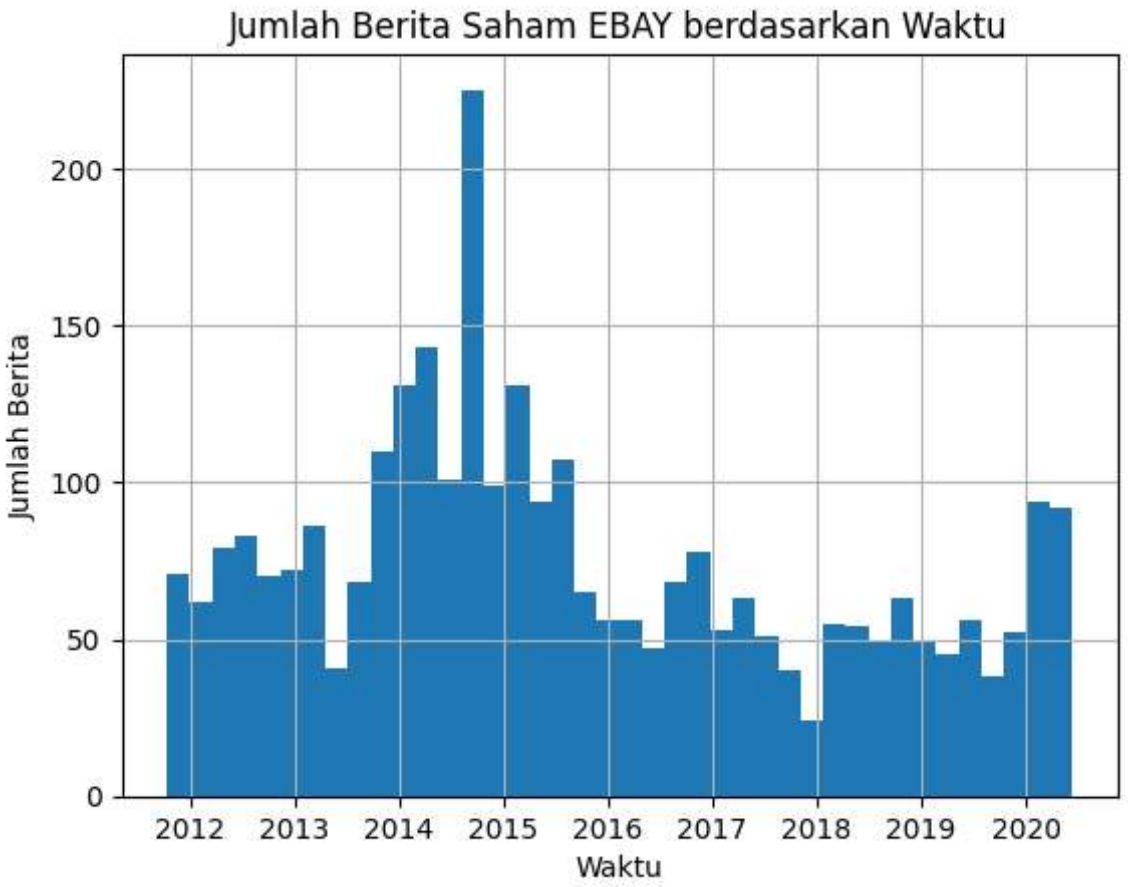
In [18]:
```
# Tampilkan persebaran berita berdasarkan waktu
df_nvda["date"].hist(bins=40)
```

```
plt.title("Jumlah Berita Saham NVDA berdasarkan Waktu")
plt.xlabel("Waktu")
plt.ylabel("Jumlah Berita")
```

Out[18]:
```
Text(0, 0.5, 'Jumlah Berita')
```

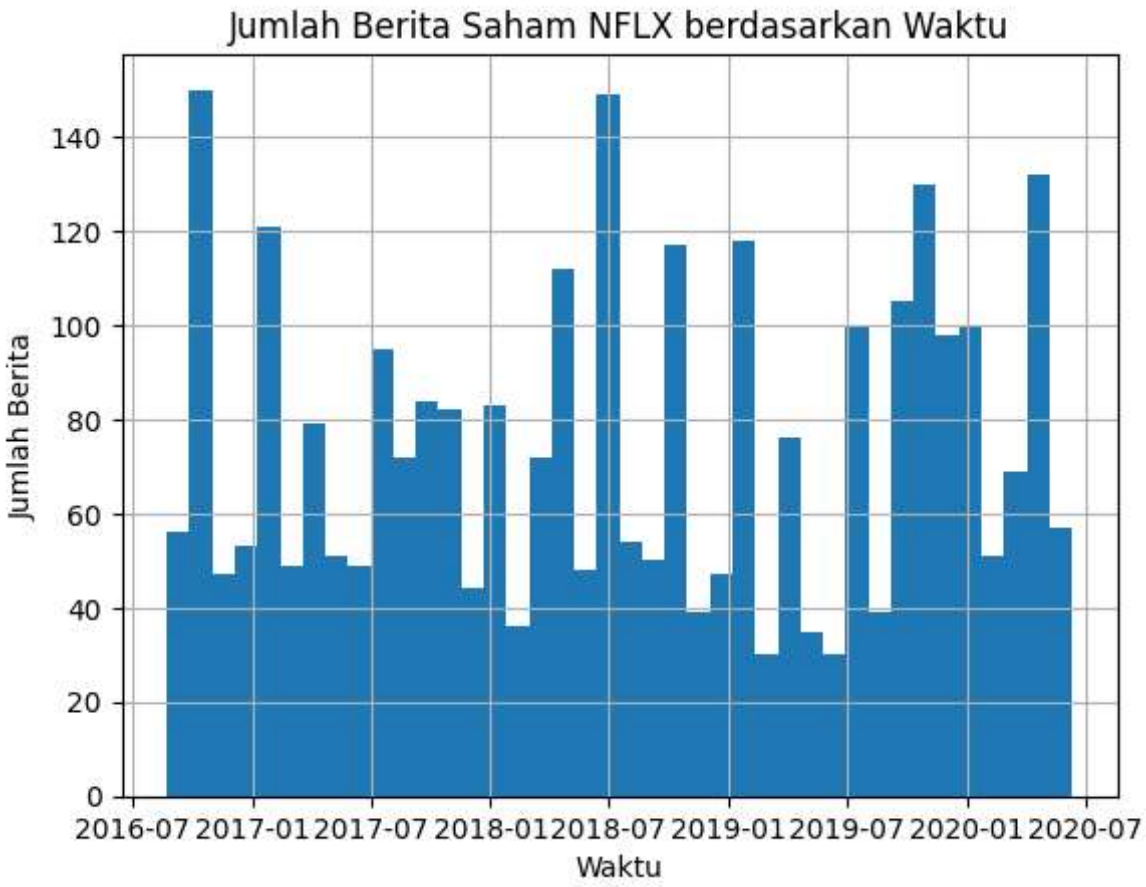
In [19]:
```
df_ebay["date"].hist(bins=40)

plt.title("Jumlah Berita Saham EBAY berdasarkan Waktu")
plt.xlabel("Waktu")
plt.ylabel("Jumlah Berita")
```

Out[19]:
```
Text(0, 0.5, 'Jumlah Berita')
```



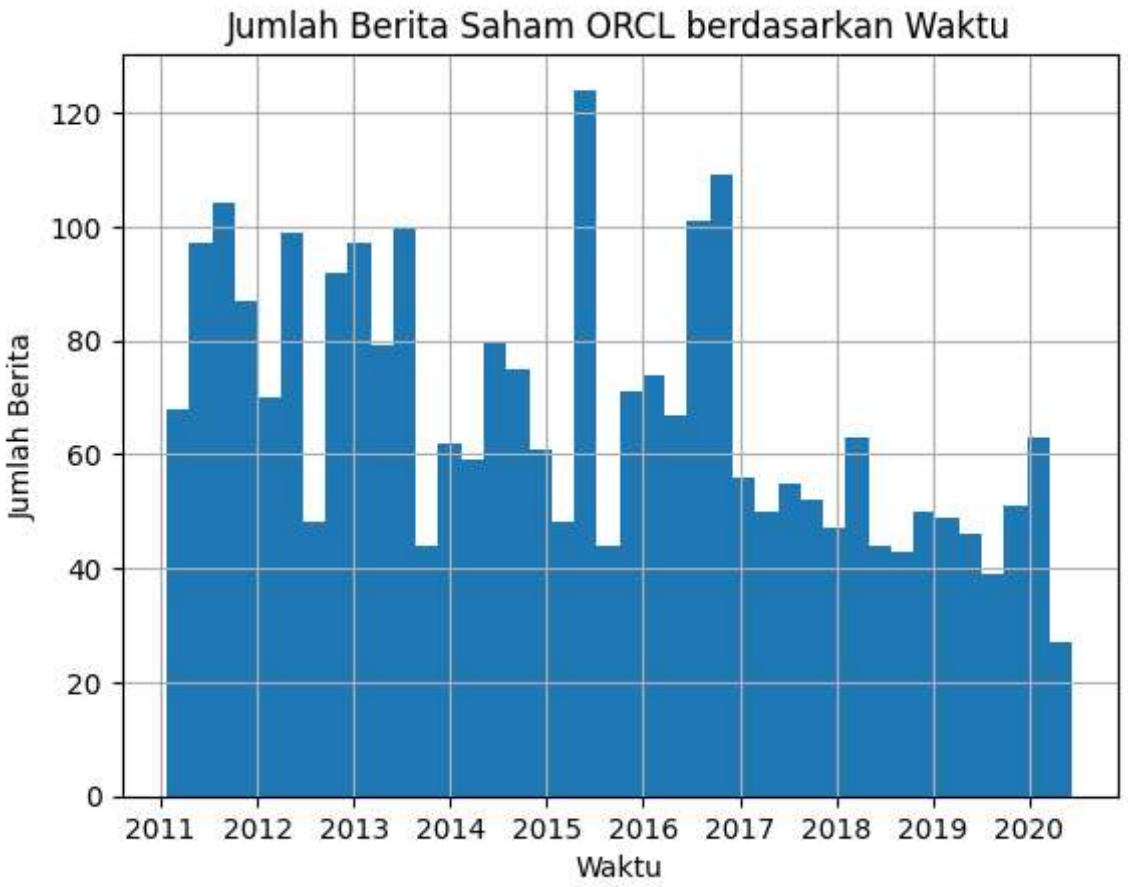
In [20]:
```
df_nflx["date"].hist(bins=40)

plt.title("Jumlah Berita Saham NFLX berdasarkan Waktu")
plt.xlabel("Waktu")
plt.ylabel("Jumlah Berita")
```

Out[20]:
```
Text(0, 0.5, 'Jumlah Berita')
```



In [21]:
```
df_orcl["date"].hist(bins=40)

plt.title("Jumlah Berita Saham ORCL berdasarkan Waktu")
plt.xlabel("Waktu")
plt.ylabel("Jumlah Berita")
```

Out[21]:
```
Text(0, 0.5, 'Jumlah Berita')
```



Berdasarkan diagram di atas, diketahui bahwa data berita tidak terkonsentrasi pada suatu periode pendek tertentu saja. Namun periode pengambilan data untuk setiap perusahaan ternyata berbeda. Contohnya data berita ORCL tersedia pada periode tahun 2011 - 2020 sedangkan data berita NFLX hanya tersedia pada periode tahun 2016 - 2020. Fakta ini perlu menjadi perhatian saat data berita diproses bersama data harga. Tidak adanya data berita pada periode tertentu belum tentu berarti sentimen terhadap saham netral, melainkan bisa karena tidak dilakukan pengumpulan data berita pada periode tersebut.

Untuk demonstrasi analisis sentimen berita, akan digunakan sebuah library untuk analisis sentimen yaitu VADER (Valence Aware Dictionary and Sentiment Reasoner). Demonstrasi analisis sentimen dilakukan untuk NVDA.

In [22]:
```
# Tambahkan kolom sentiment
vader = SentimentIntensityAnalyzer()
calculate_sentiment = lambda title: vader.polarity_scores(title)["compound"]
df_nvda_sent = df_nvda
df_nvda_sent["sentiment"] = df["title"].apply(calculate_sentiment)
df_nvda_sent
```

```
<ipython-input-22-8174c213a4ea>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_nvda_sent["sentiment"] = df["title"].apply(calculate_sentiment)
```

Out[22]:

	title	date	stock	sentiment
917098	Shares of several technology companies are tra...	2020-06-10	NVDA	0.7296
917099	Afternoon Market Stats in 5 Minutes	2020-06-10	NVDA	0.0000
917100	Morning Market Stats in 5 Minutes	2020-06-10	NVDA	0.0000
917101	Shares of several technology companies are tra...	2020-06-09	NVDA	0.7296
917102	Afternoon Market Stats in 5 Minutes	2020-06-08	NVDA	0.0000
...	...	...	...	...
920232	J.P. Morgan Upgrades NVIDIA Corporation To Neu...	2011-03-08	NVDA	0.0000
920233	JP Morgan Upgrades NVIDIA To Neutral, $21 PT	2011-03-08	NVDA	0.3680
920234	Goldman Sachs Gives Color On Semiconductors (N...	2011-03-07	NVDA	0.0000
920235	Auriga Still Not Sure Where Reality Lies For N...	2011-03-07	NVDA	-0.5806
920236	Nvidia Goes Negative (NVDA)	2011-03-03	NVDA	-0.5719

3133 rows × 4 columns

In [23]:

```
# Tampilkan rata-rata sentimen
df_nvda_sent_mean = df_nvda_sent.drop(columns=["title", "stock"]).groupby("date")['

fig, ax = plt.subplots()

ax.bar(df_nvda_sent_mean.index, df_nvda_sent_mean.values)

ax.xaxis.set_major_locator(mdates.MonthLocator(interval=12))
ax.xaxis.set_major_formatter(mdates.DateFormatter("%d-%m-%Y"))

fig.autofmt_xdate()

plt.title("Nilai Sentimen Berita Saham NVDA berdasarkan Waktu")
plt.xlabel("Waktu")
plt.ylabel("Nilai Sentimen")

plt.show()
```

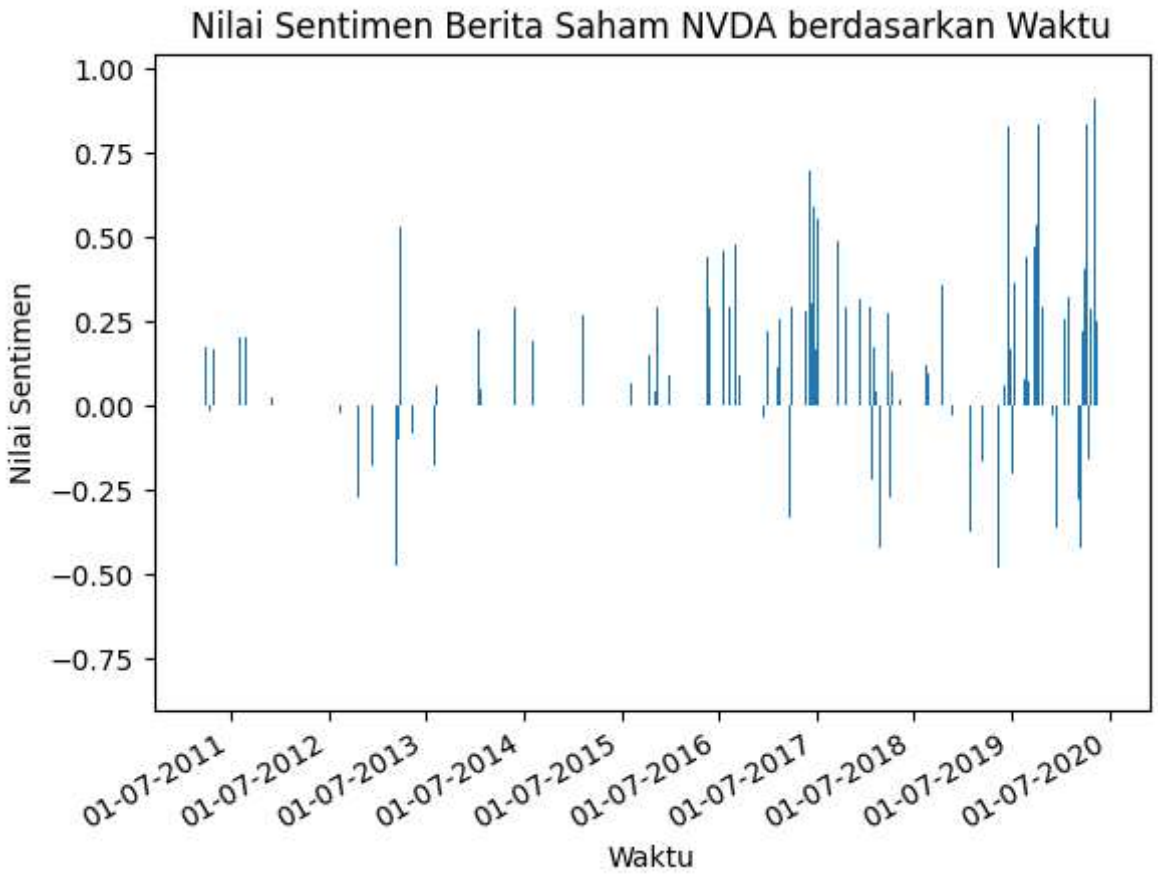


Diagram di atas menunjukkan bahwa berita pada dataset memiliki sentimen yang beragam dan dapat diteksi oleh VADER. Hasil identifikasi sentimen ini dapat ditambahkan sebagai salah satu fitur dalam proses prediksi trend harga saham dengan menyocokkan kolom date dan ticker symbol dari stock.

combine/concat dengan dataset saham

In [121…

```
import yfinance
msft = yfinance.Ticker("NVDA")
start_date = "2011-08-01"
end_date = "2020-06-01"

stock_df = msft.history(start=start_date, end=end_date)

stock_df.head()
```

Out[121]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2011-08-01 00:00:00-04:00	3.245394	3.350898	3.217872	3.341724	101962000	0.0	0.0
2011-08-02 00:00:00-04:00	3.318788	3.527503	3.307321	3.307321	135643200	0.0	0.0
2011-08-03 00:00:00-04:00	3.314201	3.408237	3.282092	3.396770	76523600	0.0	0.0
2011-08-04 00:00:00-04:00	3.334843	3.334843	3.073376	3.073376	97973200	0.0	0.0
2011-08-05 00:00:00-04:00	3.144477	3.183468	2.889892	2.970166	127058800	0.0	0.0

In [122…

```
stock_df.head()
```

Out[122]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2011-08-01 00:00:00-04:00	3.245394	3.350898	3.217872	3.341724	101962000	0.0	0.0
2011-08-02 00:00:00-04:00	3.318788	3.527503	3.307321	3.307321	135643200	0.0	0.0
2011-08-03 00:00:00-04:00	3.314201	3.408237	3.282092	3.396770	76523600	0.0	0.0
2011-08-04 00:00:00-04:00	3.334843	3.334843	3.073376	3.073376	97973200	0.0	0.0
2011-08-05 00:00:00-04:00	3.144477	3.183468	2.889892	2.970166	127058800	0.0	0.0

In [123…

```
stock_df.tail()
```

Out[123]:

Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
2020-05-22 00:00:00-04:00	87.969295	90.638201	86.852886	89.972839	103876400	0.0	0.0
2020-05-26 00:00:00-04:00	91.273663	91.522861	86.441722	86.897751	77078000	0.0	0.0
2020-05-27 00:00:00-04:00	85.973216	86.050467	79.710876	84.978920	117589200	0.0	0.0
2020-05-28 00:00:00-04:00	83.852542	87.323872	83.523607	84.597649	73489600	0.0	0.0
2020-05-29 00:00:00-04:00	85.270493	88.470192	84.577724	88.470192	74525600	0.0	0.0

In [124…
```
stock_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2222 entries, 2011-08-01 00:00:00-04:00 to 2020-05-29 00:00:00-04:0
0
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Open          2222 non-null   float64
 1   High          2222 non-null   float64
 2   Low           2222 non-null   float64
 3   Close         2222 non-null   float64
 4   Volume        2222 non-null   int64  
 5   Dividends     2222 non-null   float64
 6   Stock Splits  2222 non-null   float64
dtypes: float64(6), int64(1)
memory usage: 138.9 KB
```

In [125…
```
# Reset the index to convert Date to another column
stock_df = stock_df.reset_index()
```

In [126…
```
stock_df = stock_df.rename(columns={'index': 'date'})
```

In [127…
```
stock_df.head()
```

Out[127]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2011-08-01 00:00:00-04:00	3.245394	3.350898	3.217872	3.341724	101962000	0.0	0.0
1	2011-08-02 00:00:00-04:00	3.318788	3.527503	3.307321	3.307321	135643200	0.0	0.0
2	2011-08-03 00:00:00-04:00	3.314201	3.408237	3.282092	3.396770	76523600	0.0	0.0
3	2011-08-04 00:00:00-04:00	3.334843	3.334843	3.073376	3.073376	97973200	0.0	0.0
4	2011-08-05 00:00:00-04:00	3.144477	3.183468	2.889892	2.970166	127058800	0.0	0.0

In [128…
```
df_nvda_sent.head()
```

Out[128]:

	title	date	stock	sentiment
917098	Shares of several technology companies are tra...	2020-06-10	NVDA	0.7296
917099	Afternoon Market Stats in 5 Minutes	2020-06-10	NVDA	0.0000
917100	Morning Market Stats in 5 Minutes	2020-06-10	NVDA	0.0000
917101	Shares of several technology companies are tra...	2020-06-09	NVDA	0.7296
917102	Afternoon Market Stats in 5 Minutes	2020-06-08	NVDA	0.0000

In [129…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1397891 entries, 0 to 1400468
Data columns (total 3 columns):
 #   Column  Non-Null Count    Dtype
---  ------  --------------    -----
 0   title   1397891 non-null  object
 1   date    1397891 non-null  datetime64[ns]
 2   stock   1397891 non-null  object
dtypes: datetime64[ns](1), object(2)
memory usage: 74.9+ MB
```

dari index date(karena sudah saham nvdia) yang bersinggunan antara 2 dataset

In [130…
```
df_nvda_sent.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3133 entries, 917098 to 920236
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   title      3133 non-null   object
 1   date       3133 non-null   datetime64[ns]
 2   stock      3133 non-null   object
 3   sentiment  3133 non-null   float64
dtypes: datetime64[ns](1), float64(1), object(2)
memory usage: 122.4+ KB
```

In [131…
```
stock_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2222 entries, 0 to 2221
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Date          2222 non-null   datetime64[ns, America/New_York]
 1   Open          2222 non-null   float64
 2   High          2222 non-null   float64
 3   Low           2222 non-null   float64
 4   Close         2222 non-null   float64
 5   Volume        2222 non-null   int64
 6   Dividends     2222 non-null   float64
 7   Stock Splits  2222 non-null   float64
dtypes: datetime64[ns, America/New_York](1), float64(6), int64(1)
memory usage: 139.0 KB
```

In [132…
```
stock_df = stock_df.rename(columns={'Date': 'date'})
```

In [133…
```
stock_df['date'] = pd.to_datetime(stock_df['date']).dt.date
```

In [134…
```
stock_df.head()
```

Out[134]:

	date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2011-08-01	3.245394	3.350898	3.217872	3.341724	101962000	0.0	0.0
1	2011-08-02	3.318788	3.527503	3.307321	3.307321	135643200	0.0	0.0
2	2011-08-03	3.314201	3.408237	3.282092	3.396770	76523600	0.0	0.0
3	2011-08-04	3.334843	3.334843	3.073376	3.073376	97973200	0.0	0.0
4	2011-08-05	3.144477	3.183468	2.889892	2.970166	127058800	0.0	0.0

In [135…
```
df_nvda_sent_mean.head()
```

Out[135]:

	date	sentiment
0	2011-03-03	-0.5719
1	2011-03-07	-0.2903
2	2011-03-08	0.0234
3	2011-03-09	0.0000
4	2011-03-10	0.2023

In [136…
```
df_nvda_sent_mean.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1194 entries, 0 to 1193
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype         
---  ------     --------------  -----         
 0   date       1194 non-null   datetime64[ns]
 1   sentiment  1194 non-null   float64       
dtypes: datetime64[ns](1), float64(1)
memory usage: 18.8 KB
```

In [137…
```
df_nvda_sent_mean = df_nvda_sent_mean.reset_index()
```

In [138…
```
df_nvda_sent_mean.head()
```

Out[138]:

	index	date	sentiment
0	0	2011-03-03	-0.5719
1	1	2011-03-07	-0.2903
2	2	2011-03-08	0.0234
3	3	2011-03-09	0.0000
4	4	2011-03-10	0.2023

In [139…
```
df_nvda_sent_mean_temp = df_nvda_sent_mean.copy()
df_nvda_sent_mean_temp['date'] = pd.to_datetime(df_nvda_sent_mean_temp['date']).dt.
```

In [156…
```
len(df_nvda_sent_mean)
```

Out[156]:
```
1194
```

In [140…
```
# Merge the two DataFrames on the common column
# TODO : jangan pake inner pake left(stock_df) join
merged_df = pd.merge(stock_df, df_nvda_sent_mean_temp, on=['date'], how='left')

# Count the number of rows in the merged DataFrame to find the common values
common_values_count = len(merged_df)

# Print the count of common values
print(f"Number of common values: {common_values_count}")
```

```
Number of common values: 2222
```

In [141…
```
print("number of day with news", len(merged_df['date'].unique()))
```

```
number of day with news 2222
```

In [142…
```
merged_df.head()
```

Out[142]:

	date	Open	High	Low	Close	Volume	Dividends	Stock Splits	index	sentiment
0	2011-08-01	3.245394	3.350898	3.217872	3.341724	101962000	0.0	0.0	49.0	0.0000
1	2011-08-02	3.318788	3.527503	3.307321	3.307321	135643200	0.0	0.0	50.0	0.0000
2	2011-08-03	3.314201	3.408237	3.282092	3.396770	76523600	0.0	0.0	51.0	0.2023
3	2011-08-04	3.334843	3.334843	3.073376	3.073376	97973200	0.0	0.0	NaN	NaN
4	2011-08-05	3.144477	3.183468	2.889892	2.970166	127058800	0.0	0.0	NaN	NaN

In [145…
```
merged_df.tail()
```

Out[145]:

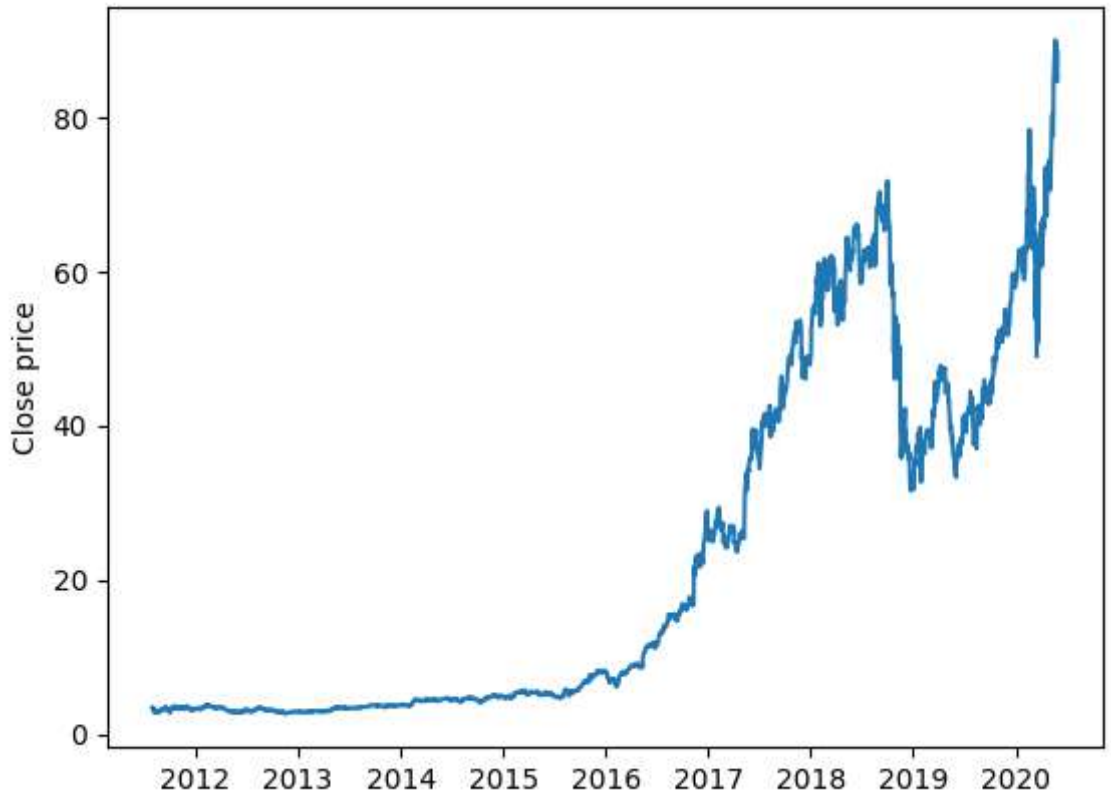
	date	Open	High	Low	Close	Volume	Dividends	Stock Splits	index	se
2217	2020-05-22	87.969295	90.638201	86.852886	89.972839	103876400	0.0	0.0	1184.0	0
2218	2020-05-26	91.273663	91.522861	86.441722	86.897751	77078000	0.0	0.0	1186.0	-0
2219	2020-05-27	85.973216	86.050467	79.710876	84.978920	117589200	0.0	0.0	1187.0	0
2220	2020-05-28	83.852542	87.323872	83.523607	84.597649	73489600	0.0	0.0	1188.0	0
2221	2020-05-29	85.270493	88.470192	84.577724	88.470192	74525600	0.0	0.0	NaN	

In [143…
```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2222 entries, 0 to 2221
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   date          2222 non-null   object 
 1   Open          2222 non-null   float64
 2   High          2222 non-null   float64
 3   Low           2222 non-null   float64
 4   Close         2222 non-null   float64
 5   Volume        2222 non-null   int64  
 6   Dividends     2222 non-null   float64
 7   Stock Splits  2222 non-null   float64
 8   index         1070 non-null   float64
 9   sentiment     1070 non-null   float64
dtypes: float64(8), int64(1), object(1)
memory usage: 191.0+ KB
```

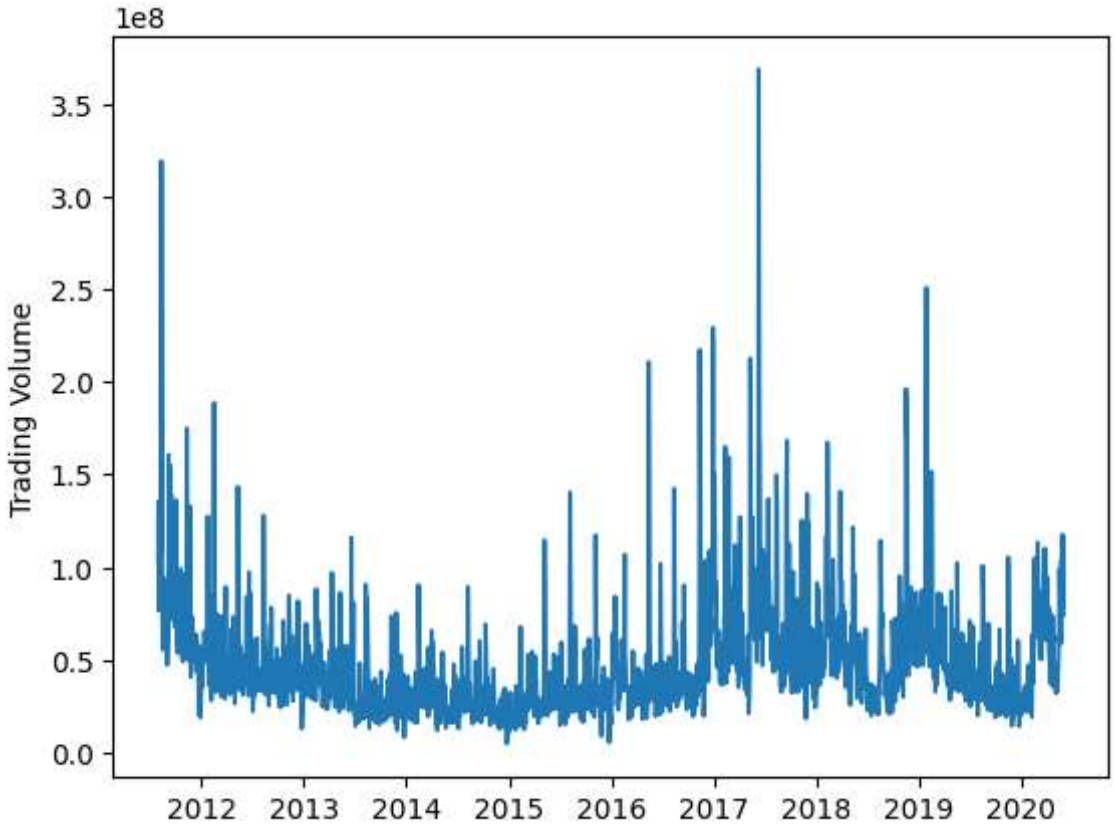
In [152…

```
plt.plot(merged_df['date'], merged_df['Close'])
plt.ylabel('Close price')
plt.show()
```



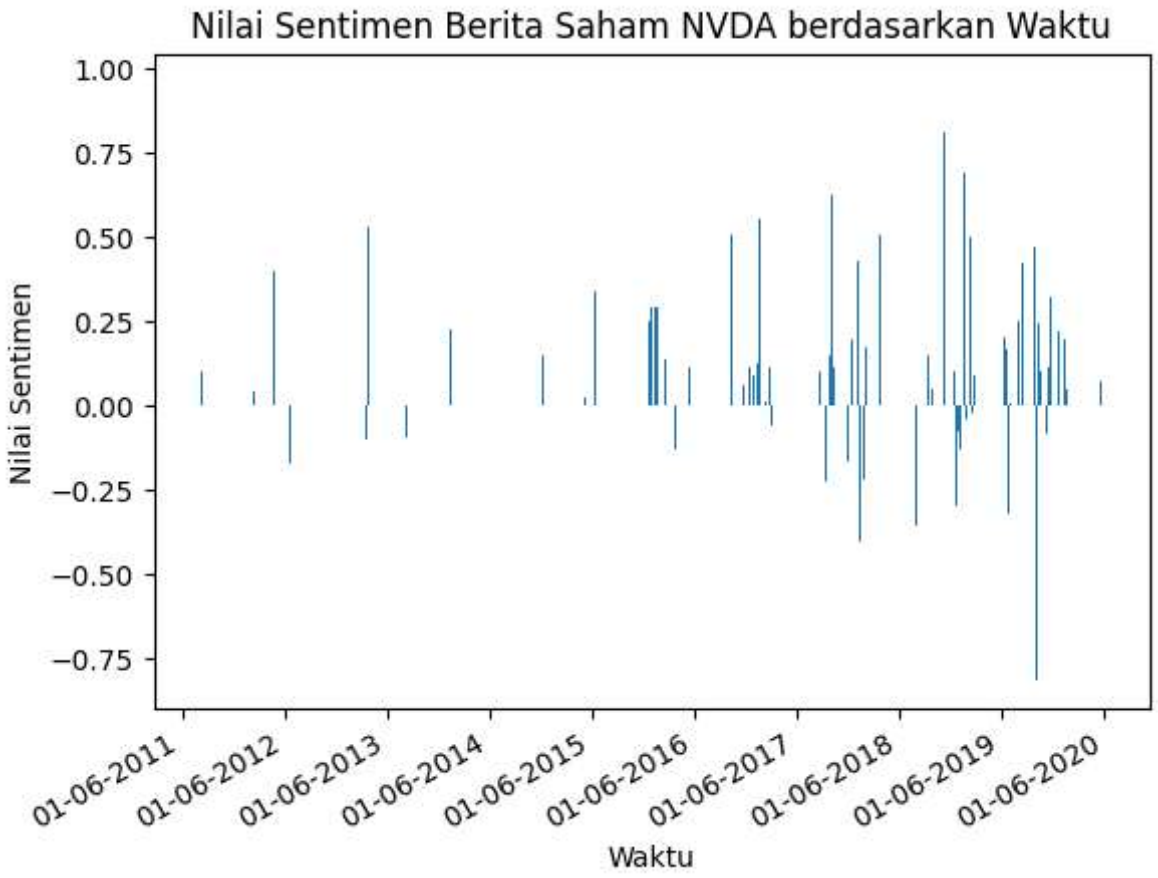
In [153…

```
plt.plot(merged_df['date'], merged_df['Volume'])
plt.ylabel('Trading Volume')
plt.show()
```



In [157…

```
# Tampilkan rata-rata sentimen

fig, ax = plt.subplots()

ax.bar(merged_df["date"], merged_df["sentiment"])

ax.xaxis.set_major_locator(mdates.MonthLocator(interval=12))
ax.xaxis.set_major_formatter(mdates.DateFormatter("%d-%m-%Y"))

fig.autofmt_xdate()

plt.title("Nilai Sentimen Berita Saham NVDA berdasarkan Waktu")
plt.xlabel("Waktu")
plt.ylabel("Nilai Sentimen")

plt.show()
```

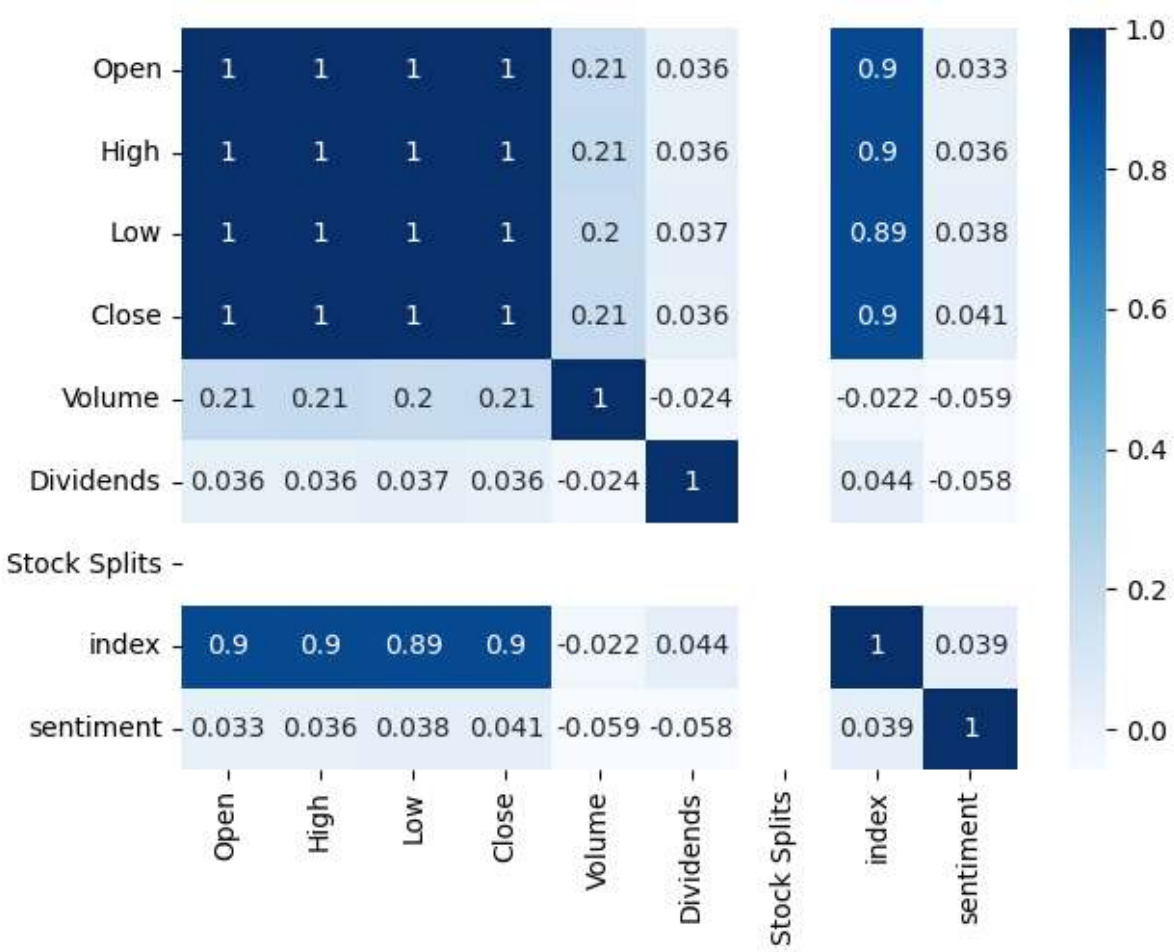


In [159…
```
import seaborn as sns
```

In [161…
```
merged_df2 = merged_df.copy()
plt.figure(figsize=(7,5))
sns.heatmap(merged_df2.corr(),cmap='Blues',annot=True)
```

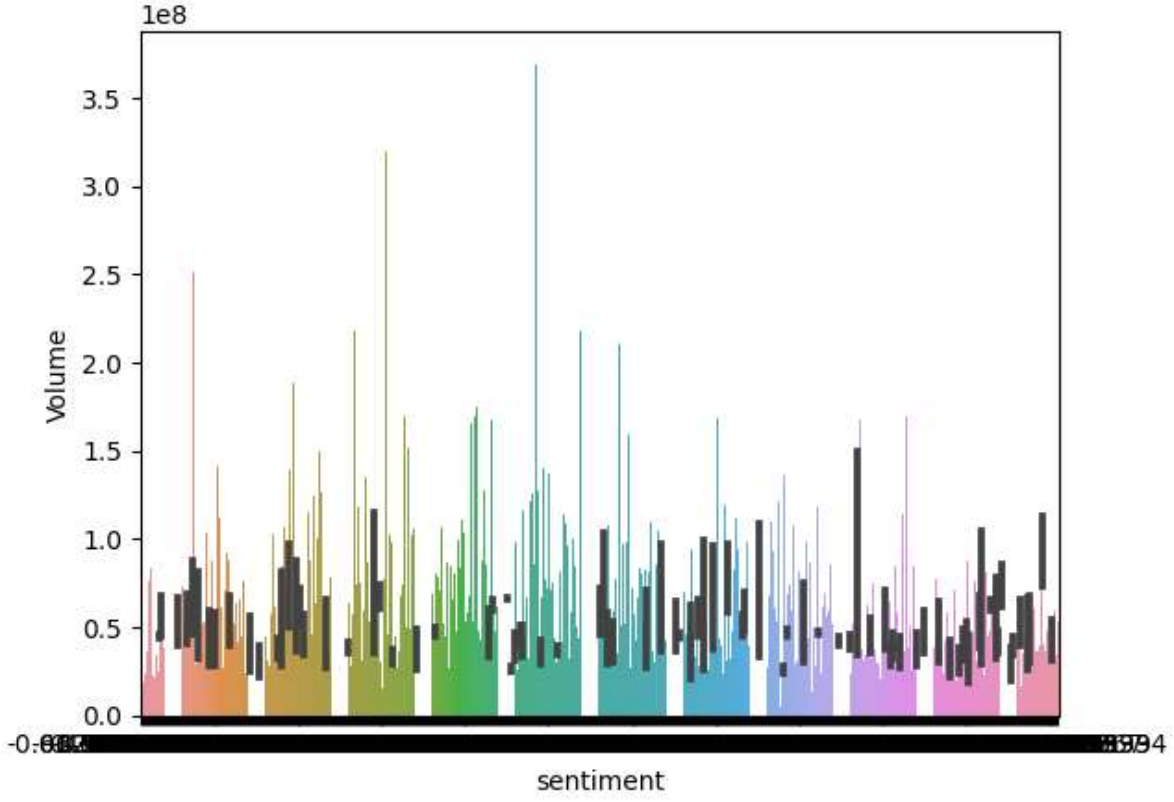
```
<ipython-input-161-c5c5be8e9604>:3: FutureWarning: The default value of numeric_on
ly in DataFrame.corr is deprecated. In a future version, it will default to False.
Select only valid columns or specify the value of numeric_only to silence this war
ning.
  sns.heatmap(merged_df2.corr(),cmap='Blues',annot=True)
```

Out[161]:
```
<Axes: >
```



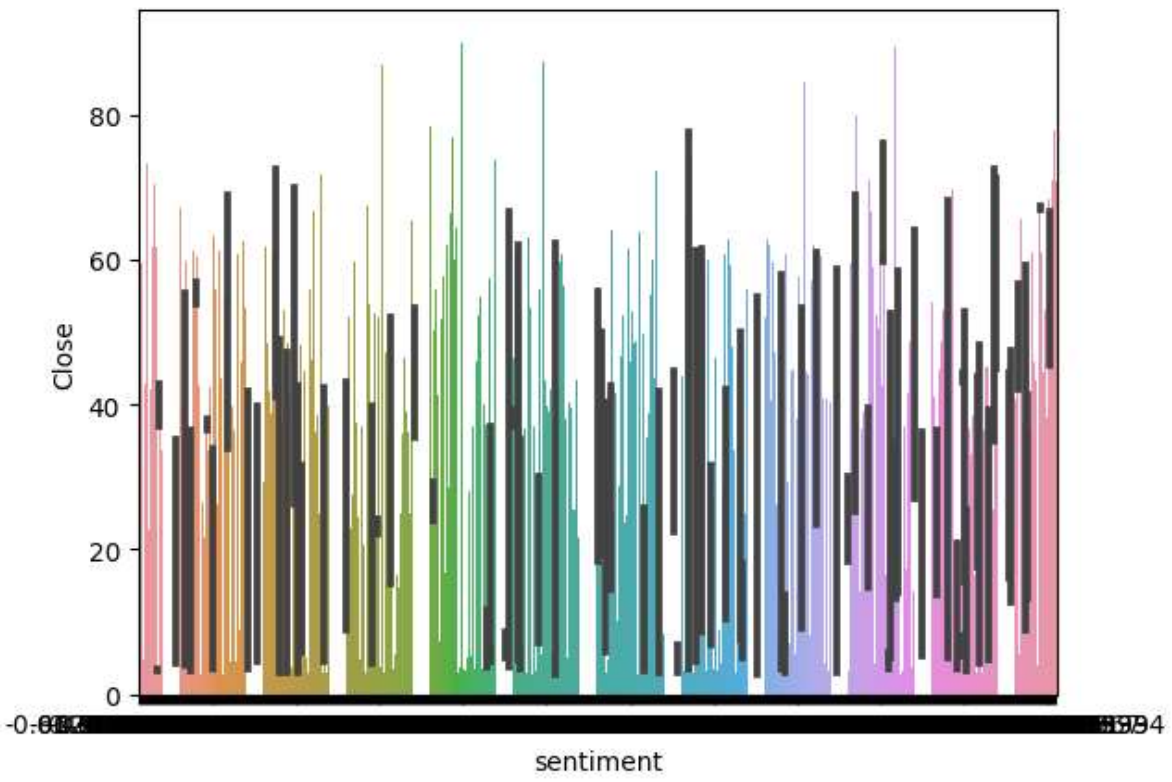
In [162…
```
sns.barplot(data=merged_df, x = "sentiment", y="Volume")
```

Out[162]:
```
<Axes: xlabel='sentiment', ylabel='Volume'>
```



In [163…
```
sns.barplot(data=merged_df, x = "sentiment", y="Close")
```

Out[163]: <Axes: xlabel='sentiment', ylabel='Close'>



Pembuatan Model

In [158…
```
feature = ["date", "Open", "High", "Low", "Close", "Volume", "Dividends", "sentimer
```

LSTM Model 1

In [ ]:
```
#
```

Evaluasi

In [ ]:
```

```