L3: Applications of Alternate Data

Applications of Alternative Data

In this lesson we consider applications of alternative data. Alternative Data in finance refers to non-traditional data sources used to gain insights into market trends, asset pricing, and investment decisions. This contrasts with traditional data like financial statements, company filings, and market prices. Text data from social media falls under this category as it reflects public sentiment, market chatter, and news dissemination that can influence asset prices. Specifically, Twitter and StockTwits data are considered valuable sources of alternative data in financial engineering due to their real-time nature and focus on financial discussions and stock market activity.

```
# Load libraries
import datetime
import matplotlib.pyplot as plt
import os
import pandas as pd
import plotly.express as px
import re
import requests
import zipfile

from datetime import datetime, timedelta
```

1.1 Getting Twitter Data

In this lesson we explore Stock Market Tweets Data from IEEE Dataport available at https://dx.doi.org/10.21227/g8vy-5w61. IEEE DataPort is a curated repository for research data, primarily focusing on engineering and technology-related fields. It is operated by the Institute of Electrical and Electronics Engineers (IEEE) and encourages researchers to share their datasets to advance knowledge in their respective domains. This dataset contains a collection of tweets related to the stock market, specifically focusing on the S&P 500 index. It aims to provide data for researchers and practitioners interested in analyzing the relationship between social media sentiment and stock market movements.

This dataset is available under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. This license allows for the free use, sharing, adaptation, and redistribution of the data, as long as appropriate credit is given to the original creators.

The dataset covers tweets from April 9th, 2020 to July 16th, 2020. This time range provides a substantial amount of data for analysis and covers various market conditions. The data was colected using the S&P 500 tag (#SPX500), the references to the top 25 companies in the S&P 500 index, and the Bloomberg tag (#stocks). The data is packaged in ZIP archive in two CSV files:
'tweets_labelled_09042020_16072020.csv' consists of 5,000 tweets selected using random sampling. Out of those 5,000 tweets, 1,300 were manually annotated in positive, neutral, or negative classes. The second file 'tweets_remaining_09042020_16072020.csv' contains remainder of tweets.

The following code snippet is designed to extract a ZIP file containing Twitter data and then locate all the CSV files within the extracted folder for further processing. The code iterates through the list of CSV files containing Twitter data, reads each file into a pandas DataFrame, standardizes the column name for tweet text, and then combines all the DataFrames into a single DataFrame called `twitter_df`:

```
    # Rename 'full_text' or 'text' column to a 'tweet_text' common n
    if 'full_text' in df.columns:
        df = df.rename(columns={'full_text': 'tweet_text'})
    elif 'text' in df.columns:
        df = df.rename(columns={'text': 'tweet_text'})

    dfs.append(df)

# Concatenate all DataFrames into a single DataFrame and display it
twitter_df = pd.concat(dfs, ignore_index=True)
twitter_df
```

[6]:

	id	created_at	tweet_text	sentiment
0	1	2020-04-09 23:59:51+00:00	@KennyDegu very very little volume. With $10T ...	NaN
1	2	2020-04-09 23:58:55+00:00	#ES_F achieved Target 2780 closing above 50% #...	NaN
2	3	2020-04-09 23:58:52+00:00	RT @KimbleCharting: Silver/Gold indicator crea...	NaN
3	4	2020-04-09 23:58:27+00:00	@Issaquahfunds Hedged our $MSFT position into ...	NaN
4	5	2020-04-09 23:57:59+00:00	RT @zipillinois: 3 Surprisingly Controversial ...	NaN
...	...	...	...	...
928668	411380	2020-06-04 18:14:57+00:00	With ad revenues falling, what's the impact on...	NaN
928669	62318	2020-04-14 02:15:01+00:00	RT @KelvinSCWong: Well another point to add to...	NaN
928670	627230	2020-06-23 14:08:15+00:00	$ITOX working on a contract with a fortune 500...	NaN
928671	890123	2020-07-14 23:18:34+00:00	$DIS it could break the 120 pin, then 125> ...	NaN
928672	301411	2020-05-06 04:22:19+00:00	Amedisys Inc $AMED COO Christopher Gerard Sell...	NaN

928673 rows × 4 columns

1.3 Discovering retweets in Twitter data

On Twitter, "RT" is a common abbreviation for "retweet". When a user wants to share someone else's tweet with their own followers, they can retweet it. "RT" is a key indicator of retweets on Twitter. It's used in manual retweets, quote retweets, and is a convention that has become part of Twitter's culture. By searching for "RT" at the beginning of tweets, we can effectively identify retweets in data analysis.

Let's now take our collection of tweets and separate them into two groups: one for retweets and one for original tweets (or other types of tweets that are not retweets). The following code snippet takes `twitter_df` DataFrame containing Twitter data and splits it into two new DataFrames by filtering 'tweet_text' column. Then the code displays both `retweets_df` and `remainder_df` using the `display()` function:

[8]:
```
# Create a DataFrame containing only retweets
retweets_df = twitter_df[twitter_df['tweet_text'].str.startswith('RT')]

# Create a DataFrame containing tweets that are not retweets
remainder_df = twitter_df[~twitter_df['tweet_text'].str.startswith('RT')]

# Display the DataFrames
print("## Retweets DataFrame:")
print("This DataFrame contains only retweets from the Twitter data.")
display(retweets_df)

print("\n## Remainder DataFrame:")
print("This DataFrame contains tweets that are not retweets.")
display(remainder_df)
```

```
## Retweets DataFrame:
This DataFrame contains only retweets from the Twitter data.
```

```
## Retweets DataFrame:
This DataFrame contains only retweets from the Twitter data.
```

	id	created_at	tweet_text	sentiment
2	3	2020-04-09 19:58:52-04:00	RT @KimbleCharting: Silver/Gold indicator crea...	NaN
4	5	2020-04-09 19:57:59-04:00	RT @zipillinois: 3 Surprisingly Controversial ...	NaN
5	6	2020-04-09 19:57:33-04:00	RT @Crypto____World: 🇿🇼ZIMBABWE🇿🇼\n\nCRYPTO NE...	NaN
7	10	2020-04-09 19:57:08-04:00	RT @NorthmanTrader: I repeat: The Fed is reckl...	NaN
9	13	2020-04-09 19:56:58-04:00	RT @TDANetwork: 🐿️ #TheWatchList panel assesse...	NaN
...	...	...	...	...
928656	853677	2020-07-09 21:55:41-04:00	RT @MarketRealist: Facebook Ad Boycott: Zucker...	NaN
928659	933161	2020-07-16 09:08:58-04:00	RT @MarkNewtonCMT: $IHI Ishares Medical Dvcs E...	NaN
928663	580427	2020-06-19 13:57:42-04:00	RT @HedgehogTrader: HHT's Venture/Microcap Sto...	NaN
928667	592492	2020-06-20 16:34:07-04:00	RT @smtraderCA: "Is A Big Moving Coming?" for ...	NaN
928669	62318	2020-04-13 22:15:01-04:00	RT @KelvinSCWong: Well another point to add to...	NaN

350752 rows × 4 columns

```
## Remainder DataFrame:
This DataFrame contains tweets that are not retweets.
```

	id	created_at	tweet_text	sentiment
0	1	2020-04-09 19:59:51-04:00	@KennyDegu very very little volume. With $10T ...	NaN
1	2	2020-04-09 19:58:55-04:00	#ES_F achieved Target 2780 closing above 50% #...	NaN
3	4	2020-04-09 19:58:27-04:00	@Issaquahfunds Hedged our $MSFT position into ...	NaN

1.4 Understanding distribution of tweets

Let's now try to understand the distribution of tweets over time. The following code takes the 'created_at' column as the index for resampling. It then resamples the data weekly and calculates the count of tweets within each weekly interval. Finally, it counts the number of tweets that occurred within each week:

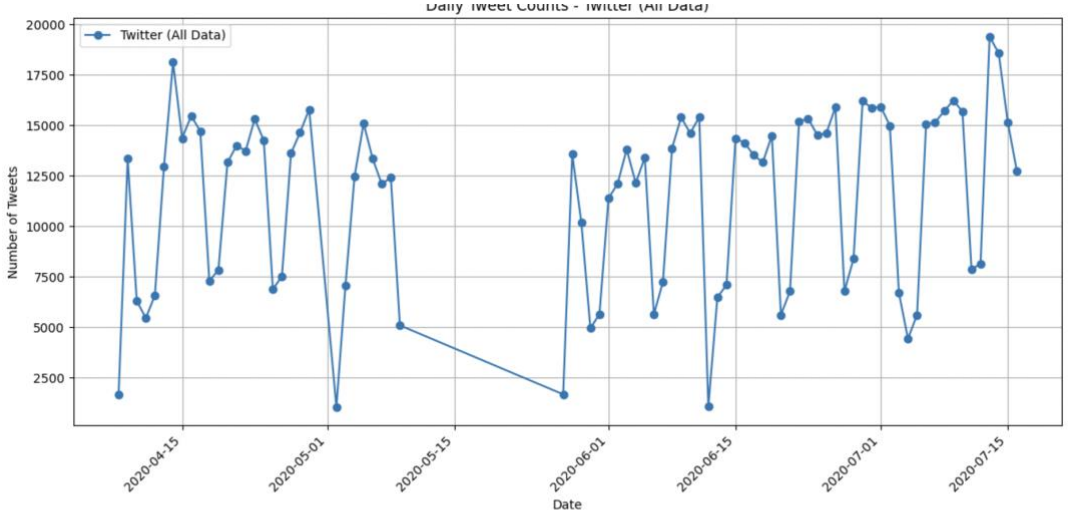
[9]:
```
# Get weekly tweet counts
weekly_counts = twitter_df.set_index('created_at').resample('W-MON', label='left', closed='left').count()
weekly_counts = weekly_counts.rename_axis(index="Week Starting")
display(weekly_counts)
```

Week Starting	id	tweet_text	sentiment
2020-04-06 00:00:00-04:00	33291	33291	39
2020-04-13 00:00:00-04:00	90678	90678	143
2020-04-20 00:00:00-04:00	84788	84788	106
2020-04-27 00:00:00-04:00	52106	52106	71
2020-05-04 00:00:00-04:00	70438	70438	90
2020-05-11 00:00:00-04:00	0	0	0
2020-05-18 00:00:00-04:00	0	0	0
2020-05-25 00:00:00-04:00	35982	35982	53
2020-06-01 00:00:00-04:00	75666	75666	100
2020-06-08 00:00:00-04:00	73845	73845	98
2020-06-15 00:00:00-04:00	81982	81982	124

1.5 Visualising Twitter data

By considering the data volume and applying appropriate visualization techniques, we can gain more meaningful insights from the daily tweet count plot. Let's now group data by date and plot the daily tweet counts for the `twitter_df` DataFrame to investigate it. The followig code first groups the `twitter_df` DataFrame by the date part of the 'created_at' column using `dt.date` to extract the date. This creates groups of tweets for each day. Then it counts the number of tweets in each group using `['tweet_text'].count()`, which counts the non-null values in the 'tweet_text' column. We then plot the data:

[12]:
```
# GroupBy date and count
daily_tweet_counts_twitter_all = twitter_df.groupby(twitter_df['created_at'].dt.date)['tweet_text'].count()

# Plotting
plt.figure(figsize=(12, 6))

plt.plot(daily_tweet_counts_twitter_all.index, daily_tweet_counts_twitter_all.values, label='Twitter (All Data)', mark

plt.xlabel('Date')
plt.ylabel('Number of Tweets')
plt.title('Daily Tweet Counts - Twitter (All Data)')
plt.legend()
plt.grid(True)

plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability

plt.tight_layout()  # Adjust layout to prevent labels from overlapping
plt.show()
```



By examining the plot, we are able to identify any gaps or periods where the tweet counts are significantly less or zero. These gaps might indicate missing data or periods where there was a decrease in tweet activity. If we need to further investigate specific gaps or patterns, we can adjust the date range of the plot or add annotations to highlight areas of interest. In our case we know that there is gap for the period from May 10 to May 27 2020. This means there were either no tweets collected or recorded during that period, or they were filtered out during data processing.

1.6 Discovering cashtags and hashtags in Twitter data

In Twitter tweets, stock symbols can appear as either cashtags (e.g., $AAPL, $GOOG) or hashtags (e.g., #AAPL, #GOOG), both typically representing a company's stock ticker symbol. Cashtags are becoming more common, but hashtags are still used for broader stock-related discussions or events. These symbols can be placed anywhere within a tweet and often provide context for the message, linking it to a specific stock or topic.

Let's now uncover some insights into which stocks are being discussed most frequently in our Twitter data. The following code first extracts stock symbols from the beginning of each tweet in the `twitter_df` DataFrame using regular expressions and stores them in a new 'stock_symbols' column. Then, it explodes this column to create separate rows for each symbol in a tweet. It converts the symbols to lowercase, counts their occurrences, sorts them by frequency in descending order, and finally displays the top 10 most frequently mentioned stock symbols in the dataset:

[7]:
```
# Discover stock symbols and hashtags
twitter_df['cashtags_hashtags'] = twitter_df['tweet_text'].str.findall(r'(?:\$|#)([a-zA-Z]+\.*[a-zA-Z]+)')
twitter_df.explode('cashtags_hashtags')['cashtags_hashtags'].str.lower().value_counts().sort_values(ascending=False).h
```

[7]:
```
cashtags_hashtags
stocks         262381
spx            190857
aapl           123629
spy            115031
amzn           108484
fb              85685
es              66068
stockmarket     65091
msft            62018
trading         60261
Name: count, dtype: int64
```

Here `.str.findall(r'(?:\$|#)([a-zA-Z]+\.*[a-zA-Z]+)')` applies a regular expression to find and extract cashtags and

Here `.str.findall(r'(?:\$|#)([a-zA-Z]+\.*[a-zA-Z]+)')` applies a regular expression to find and extract cashtags and hashtags from the tweet text. Let's break down the regex:

- `(?: ... )` : This creates a non-capturing group. It means this part of the regex is used for matching, but the matched content won't be separately captured or stored.
- `\$|#` : This matches either a literal dollar sign ($) or a hashtag (#). The `|` symbol represents an "or" condition.
- `[a-zA-Z]+` : Matches one or more letters (a-z, A-Z).
- `\.*` : Matches zero or one dot (.), allowing for symbols with dots (e.g., $BRK.B).
- `[a-zA-Z]+` : Matches one or more letters again, capturing the rest of the symbol after the dot (if any).

So the entire regex in the first line in the above code searches for patterns that look like stock symbol cashtags (e.g., $AAPL, $AAPL, $GOOG, $BRK.B) or hashtags and stores them in a new column called 'cashtags_hashtags'.

`twitter_df.explode('cashtags_hashtags')` in the second line of the above code "explodes" the 'cashtags_hashtags' column, transforming each list of cash-/hashtags into separate rows. So, if a tweet had multiple cash-/hashtags, it will now have multiple rows, each with one tag. Then the reminder of second line in the above code takes the extracted cash-/hashtags, counts how many times each tag appears in the tweets, sorts them by frequency, and shows the top 10 most mentioned cash-/hashtags.

Overall, this table highlights that the S&P 500 index (spx) together with SPDR S&P 500 ETF Trust (spy) and E-mini S&P 500 futures (es) (both are financial instruments that track the performance of the S&P 500 index) are among the top 10 mentions in this Twitter dataset. This is not suprising since this Twitter dataset itself was originally collected using S&P 500 tag and other top 25 companies in the S&P 500 index. From the above table we can also gather that other most discussed financial instruments within the analyzed Twitter data are Apple (aapl), Amazon (amzn), FB (fb), and Microsoft (msft). Hashtags "stocks", "stockmarket" and "trading" further emphasize the dataset's focus on stock market-related topics. This information can be valuable for understanding market trends, investor sentiment, and the overall focus of financial discussions on Twitter during the period covered by the dataset.

1.7 Filterig Twitter data

Let's now try to filter some of tweets for specific terms. The following code defines a function called `contains_spy()` that checks if a given text string contains any mention of the SPY ETF (SPDR S&P 500 ETF Trust), considering various textual variations. This time we check entire tweet text instead of just searching for cashtags and hashtags. First we use `text.lower()` which converts the input text to lowercase to ensure case-insensitive matching. `patterns = [...]` defines a list of regular expression patterns representing different ways the SPY ETF might be mentioned. It includes variations like "spy", "$spy", "#spy", "spydr", "s&p 500 etf", with and without word boundaries (\b). Word boundaries ensure that "spy" is not matched within words like "spying". In essence, this function aims to identify text entries in tweets that mention the SPY ETF, regardless of how it is written. Then the code filters the `twitter_df` DataFrame to keep only the tweets that mention the SPY ETF, using the `contains_spy()` function:

[*]:
```
# Define a function to check for S&P 500 mentions with variations
def contains_spy(text):
    # Handle variations in spacing, "&", and case
    text = text.lower()  # Convert to lowercase for case-insensitive matching
    text = re.sub(r"[^a-zA-Z0-9 ]", "", text)  # Remove special characters except spaces

    # Check for different patterns
    patterns = [
        r"\bspy\b",  # Using word boundaries (\b) to avoid capturing part of word (e.g. "spying")
        r"\$spy",  # Cashtag
        r"#spy",  # Hashtag
        r"\b\$spy\b",  # Cashtag with word boundaries
        r"\b#spy\b",  # Hashtag with word boundaries
        r"\bspdr\b",  # Full name part using word boundaries
        r"\bs\&p 500 etf\b",  # Full name part with variations using word boundaries
        r"\bs\&p500 etf\b",  # Full name part with variations using word boundaries
        r"\bsp500 etf\b"  # Full name part with variations using word boundaries
        r"\bspy etf\b"  # Full name part with variations using word boundaries
        # ...add more variations as needed...
    ]
```

```
# Apply the function to filter the DataFrame and display it
filtered_twitter_df = twitter_df[twitter_df['tweet_text'].apply(contains_spy)]
filtered_twitter_df
```

[11]:

	id	created_at	tweet_text	sentiment	cashtags_hashtags
1	2	2020-04-09 19:58:55-04:00	#ES_F achieved Target 2780 closing above 50% #...	NaN	[ES, Fibonacci, SPX, SPY, tradign, futures]
10	14	2020-04-09 19:56:51-04:00	$UMRX bouncing. EXTREMELY OVERSOLD #Coronaviru...	NaN	[UMRX, Coronavirus, DECN, OPGN, CODX, HTBX, TN...
30	34	2020-04-09 19:54:28-04:00	$SPY QQQ $VXX AAPL $BA MSFT\n\nGuys, I figu...	NaN	[SPY, QQQ, VXX, AAPL, BA, MSFT]
35	39	2020-04-09 19:54:01-04:00	$AAPL SPY retest highs before retesting lows....	NaN	[AAPL, SPY]
55	59	2020-04-09 19:48:56-04:00	Traders, did you secure the 💰 this week? $SPY ...	NaN	[SPY, ASTC, ICD, CLMT, ACY, TLSA, NLS, TSLA, B...
...	...	...	...	...	...
928644	644942	2020-06-24 08:15:40-04:00	$SPY QQQ $IWM AAPL <smh> Gonna go gre...	NaN	[SPY, QQQ, IWM, AAPL]
928645	785568	2020-07-06 09:33:50-04:00	RT @hyumialert: [$SPY QQQ $IWM SPX $NDX RU...	NaN	[SPY, QQQ, IWM, SPX, NDX, RUT, AMZN, NFLX, AAP...
928653	71944	2020-04-15 09:05:27-04:00	All these puts printing \n\n$spy nugt $jpm c...	NaN	[spy, nugt, jpm, cvna]
928667	592492	2020-06-20 16:34:07-04:00	RT @smtraderCA: "Is A Big Moving Coming?" for ...	NaN	[SPX, NDX, SPY, QQQ]
928670	627230	2020-06-23 10:08:15-04:00	$ITOX working on a contract with a fortune 500...	NaN	[ITOX, xrp, btc, spy, tsla, msft, goog, ba, fb...

2 What is StockTwits data

StockTwits is a social media platform specifically designed for investors and traders to share ideas and information about the stock market. It's like Twitter, but with a focus on financial discussions. StockTwits aims to provide a platform for investors of all levels to connect, share ideas, and stay informed about the stock market in real time. It's a valuable tool for understanding market sentiment, discovering new investment ideas, and engaging with a community of like-minded individuals.

StockTwits fosters a strong sense of community by enabling users to connect with each other, follow interesting profiles, and engage in discussions about specific stocks or broader market trends. It's a space for collaborative learning and idea exchange. It ofers ability to engage in Stock-Specific Discussions. Conversations are often organized using "cashtags" (similar to hashtags on Twitter), which are denoted by a dollar sign followed by a stock symbol (e.g., $AAPL for Apple stock). This feature makes it easy to find discussions and sentiment related to particular companies.

StockTwits' distinct feature is to allow users to express their attitude and overall emotional tone or opinion regarding a particular stock, market trend, or investment strategy. It's essentially a measure of whether people are feeling bullish (positive), bearish (negative), or neutral about a specific topic. On the StockTwits platform, users can tag their messages as either Bearish or Bullish to indicate their sentiment or outlook on a particular stock or the market in general. These tags provide a quick and visual way for other users to understand the sentiment behind a message. These tags can be levereged for:

- Sentiment Analysis: The Bearish and Bullish tags provide valuable data for sentiment analysis algorithms. These algorithms can use the tags to quickly identify and quantify the overall sentiment expressed by users towards specific stocks or the market as a whole.
- Filtering and Search: Users can filter their StockTwits feed or search for messages based on these tags. This allows them to focus on messages that align with their own sentiment or to get a sense of the prevailing sentiment around a particular stock.
- Community Engagement: The tags encourage users to explicitly express their opinions and engage in discussions with others who share similar or opposing views. This fosters a more dynamic and interactive community.

Important considerations for Bearish and Bullish tags on StockTwits:

This dataset is also available under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. This dataset is provided in CSV format and contains a collection of messages posted on the StockTwits platform related to the SPY ETF (SPDR S&P 500 ETF Trust).

The following code downloads this dataset from figshare, saves it as a CSV file, and then reads the data into `stocktwits_df` pandas DataFrame for further analysis:

[3]:
```
# Define the chunk size (adjust this value as needed)
chunksize = 10000

# Create an empty list to store processed chunks
processed_chunks = []

# Iterate through the CSV file in chunks
for chunk in pd.read_csv("SPY_stocktwits_messages.csv", chunksize=chunksize):
    processed_chunks.append(chunk) # Append the chunk to the list

# Concatenate the processed chunks into a single DataFrame named stocktwits_df
stocktwits_df = pd.concat(processed_chunks, ignore_index=True)
stocktwits_df
```

[3]:

	Unnamed: 0	Sentence	Timestamp	DateTime	Sentiment
0	25	$SPY watching squawk box im losing brain cells	1584964231000	2020/03/23 07:50:31	NaN
1	24	$SPY there's no cure	1584964276000	2020/03/23 07:51:16	Bearish
2	22	$SPY DJIA $AAPL TSLA \n\nIt's only just begi...	1584964278000	2020/03/23 07:51:18	NaN
3	21	$SPY uh oh	1584964308000	2020/03/23 07:51:48	Bullish
4	20	$TVIX Im in a NY hospital with my mom. They do...	1584964331000	2020/03/23 07:52:11	NaN
...	...	...	...	...	...
3261862	2135962	$SPY The sinking love boat sailing south!	1649232079000	2022/04/06 04:01:19	NaN

2.2 Adjusting time zone in StockTwits data

The StockTwits dataset covers much longer period spanning 2 years from 2020 to 2022 than our Twitter dataset. The following code focuses on filtering the StockTwits messages DataFrame (`stocktwits_df`) to include only messages within a specific date range, from April 9, 2020, to July 16, 2020 that corresponds to data availability window for Twitter dataset in previous section of this lesson.

First, we again convert date values to DateTime object. `stocktwits_df['DateTime'] = pd.to_datetime(stocktwits_df['DateTime'])` line converts values in the 'DateTime' column in the `stocktwits_df` DataFrame to pandas DateTime objects. This is essential for performing date-based filtering and comparisons effectively. Then we define the start and end dates of the desired date range and filter the original DataFrame to create a new DataFrame containing only messages posted between April 9, 2020, and July 16, 2020, inclusive:

[4]:
```
nvert 'DateTime' column to DateTime objects
ktwits_df['DateTime'] = pd.to_datetime(stocktwits_df['DateTime'])

lter stocktwits_df DateTime from April 9 to July 16, 2020
t_date = pd.to_datetime('2020-04-09')
date = pd.to_datetime('2020-07-16')

ered_stocktwits_df = stocktwits_df[(stocktwits_df['DateTime'] >= start_date) & (stocktwits_df['DateTime'] <= end_date)
ered_stocktwits_df
```

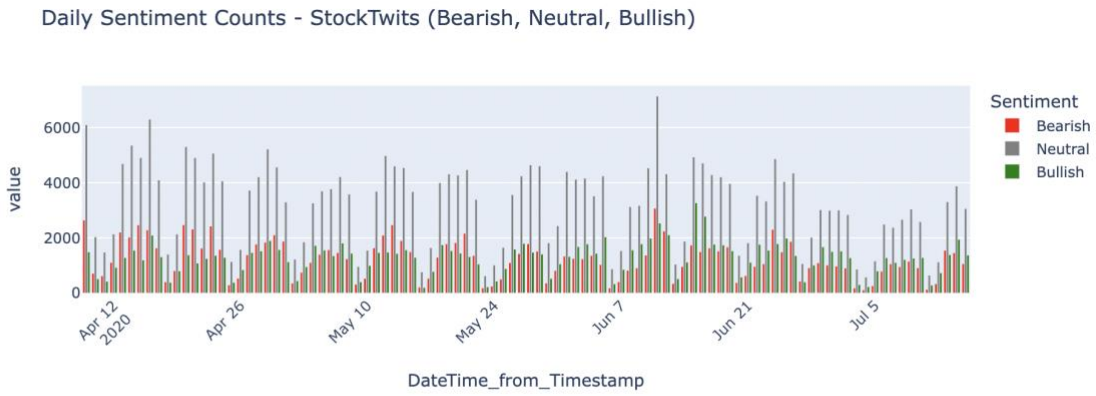
[4]:

	Unnamed: 0	Sentence	Timestamp	DateTime	Sentiment
151678	168399	$SPY I currently have AAPL, $XOM, and JBLU c...	1586404800000	2020-04-09 00:00:00	NaN
151679	168398	$SPY So besides NYC the rest of the world is b...	1586404834000	2020-04-09 00:00:34	Bearish
151680	168397	$SPY bulls thinking numbers are priced in beca...	1586404839000	2020-04-09 00:00:39	Bearish
151681	168396	$SPY plunge protection team is in full effect!	1586404863000	2020-04-09 00:01:03	Bullish

2.3 Visualising StockTwits data

Let's now craft the code to plot the `filtered_stocktwits_df` data as a bar plot, grouped by date and showing the counts of 'Bearish', 'Neutral', and 'Bullish' values in the Sentiment column. The following code processes the `filtered_stocktwits_df` DataFrame to replace NaN values in the 'Sentiment' column with 'Neutral', groups the data by date and sentiment, and then creates an interactive bar plot using Plotly with specific color mapping for different sentiments.

[7]:
```
# Replace 'NaN' with 'Neutral' in the 'Sentiment' column before grouping, using .loc
filtered_stocktwits_df.loc[filtered_stocktwits_df['Sentiment'].isnull(), 'Sentiment'] = 'Neutral'

# Group the data by date and sentiment, then count occurrences
sentiment_counts = filtered_stocktwits_df.groupby([filtered_stocktwits_df['DateTime_from_Timestamp'].dt.date, 'Sentime
sentiment_counts = sentiment_counts[['Bearish', 'Neutral', 'Bullish']] # reorder columns in specific order

# Create the interactive bar plot using Plotly
fig = px.bar(sentiment_counts, x=sentiment_counts.index, y=sentiment_counts.columns,
             labels={'x': 'Date', 'y': 'Count'},
             title='Daily Sentiment Counts - StockTwits (Bearish, Neutral, Bullish)',
             color_discrete_map={'Bearish': 'red', 'Neutral': 'grey', 'Bullish': 'green'})

# Customize the layout for better readability and display the plot
fig.update_layout(barmode='group', xaxis_tickangle=-45)
fig.show()
```



This approach allows to visualize the daily sentiment trends on StockTwits, with clear color-coded representation for Bearish, Neutral, and Bullish sentiments, enhancing the interpretability of the plot. The bar plot visualizes daily sentiment counts for StockTwits messages about the SPY ETF, categorized as Bearish (red), Neutral (grey), or Bullish (green). The x-axis represent dates, while the y-axis represent message counts. Three bars per date show counts for each sentiment. Plotly's interactivity enables zooming, hovering for details, and potential data selection, allowing in-depth exploration of sentiment trends over time.

2.4 Discovering stock symbols in StockTwits data

StockTwits messages often contain stock symbols, also known as tickers or cashtags. Users frequently include them to denote the specific stocks they are discussing. These symbols are typically represented by a dollar sign followed by the stock's ticker symbol (e.g., $AAPL for Apple, $SPY for SPDR S&P 500 ETF Trust).

Similar to how we did for Twitter dataset here again we can extract stock symbols from StockTwits messages. The following code aims to extract stock symbols (tickers) from the 'Sentence' column of the `filtered_stocktwits_df` DataFrame, count their occurrences, and display the top 10 most frequently mentioned tickers:

[8]:
```
# Discovering stock symbols in StockTwits data
filtered_stocktwits_df['tickers'] = filtered_stocktwits_df['Sentence'].str.findall('\$[a-zA-Z]+\.*[a-zA-Z]+')
filtered_stocktwits_df.explode(['tickers'])['tickers'].value_counts().sort_values(ascending=False).head(10)
```

[8]:
```
tickers
$SPY     553357
$QQQ      27869
$spy      12975
$SPX      11136
$AAPL     10631
$DJIA      8977
$DIA       8949
$ES        8062
$TSLA      7627
$BA        5368
Name: count, dtype: int64
```

2.5 Comparing Twitter and StockTwits data

Let's now compare Twitter and StockTwits data. The following code aims to do this by creating a line plot. First we calculate the daily tweet counts from the `filtered_twitter_df` DataFrame by grouping the data by the date part of the 'created_at' column (using `dt.date` to extract the date) and then counting the number of tweets in each group by counting the non-null values in the 'tweet_text' column. We do the similar operation for the `filtered_stocktwits_df` DataFrame, using the 'DateTime_from_Timestamp' column for date grouping and the 'Sentence' column for counting messages. The we create and customize plot:

[*]:
```
# Group Twitter data by date and count tweets
daily_tweet_counts_twitter = filtered_twitter_df.groupby(filtered_twitter_df['created_at'].dt.date)['tweet_text'].coun

# Group StockTwits data by date and count messages
daily_tweet_counts_stocktwits = filtered_stocktwits_df.groupby(filtered_stocktwits_df['DateTime_from_Timestamp'].dt.da

# # Create a figure and plot Twitter and StockTwits data
plt.figure(figsize=(12, 6))
plt.plot(daily_tweet_counts_twitter.index, daily_tweet_counts_twitter.values, label='Twitter', marker='o')
plt.plot(daily_tweet_counts_stocktwits.index, daily_tweet_counts_stocktwits.values, label='StockTwits', marker='x')

# Customize the plot
plt.xlabel('Date')
plt.ylabel('Number of Tweets')
plt.title('Daily Tweet Counts - Twitter vs. StockTwits')
plt.legend()
plt.grid(True)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45, ha='right')

# Adjust layout to prevent overlapping labels and display the plot
plt.tight_layout()
plt.show()
```

First let's observe that for the same time period `filtered_twitter_df` has 99,246 rows, while `filtered_stocktwits_df` has 564,588 rows. This information provides valuable context for interpreting the daily tweet count plots. It highlights the significant difference in the volume of data between the two platforms for the selected period. Here are some additional points to keep in mind:

- Platform Activity: The difference in row counts suggests that StockTwits might have had much higher activity or user engagement compared to Twitter for the given period and filtering criteria. This could be due to various factors, including the platforms' user base, focus, and popularity within the investment community.
- Data Collection and Integrity: The difference in data volume might be influenced by the data collection methods used. Data collectors use different APIs or data sources for Twitter and StockTwits and there might be variations in how data was collected, filtered, or processed, leading to differences in the final row counts. We also need to ensure that the data collection and filtering processes were consistent and reliable for both platforms to minimize potential biases or errors in the analysis.
- Analysis Implications: When comparing trends or patterns between Twitter and StockTwits, keep in mind that the absolute tweet counts might not be directly comparable due to the difference in data volume. We need to consider using normalization or relative metrics to make more meaningful comparisons.
- Data Gap Context: While the gap in Twitter data from May 10 to May 27 2020 is important to acknowledge, it's also essential to consider it in the context of the overall data volume. If the gap represents a small proportion of the total Twitter data, its impact on the overall analysis might be limited. However, if the gap is significant relative to the total data, it could affect the interpretation of trends or patterns.

By considering the overall data volume difference and its potential implications, we can make more informed interpretations of the daily tweet count plots and draw more robust conclusions from analysis. We need to remember to clearly communicate the data volume difference and any normalization or scaling applied in reports or presentations for transparency.

L4: Social Media

2.1 An Example of Quantifying Sentiment Using YouTube Data

In this subsection, we'll explore how to leverage YouTube API to gather and analyze public sentiment around significant financial events. We'll use the 2020 market crash as a case study to understand how sentiment can be quantified and correlated with financial data like the S&P 500 index.

Pre-requisites:

- Google Account: Ensure you have an active Google account. This is necessary to access Google's API services.
- Google Cloud Project: You'll need to create a project in the Google Cloud Console. This project will host your API key, which is essential for accessing YouTube's data.

YouTube API Setup: Follow these steps to get started:

1. Go to the Google Cloud Console.
2. Create a new project.
3. Navigate to the API Library and enable the YouTube Data API v3.
4. Create credentials (API key) for accessing the API.

[1]:
```
import requests
import praw
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import pandas as pd
from collections import defaultdict
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
```

The graph above is an example of how one can quantify sentiment. The easiest way to do so is by counting the number of words in our choice, and given that some words are not easily found unless there is a reason, we can trust that the excess mentioning of such words indicates the sentiment at that time. The green bars represent the sum of keywords like "recession," "economic crisis," and "inflation" found in YouTube comments, indicating heightened public concern. The blue line tracks the S&P 500 index price, highlighting the market's reaction during the same period. As the keyword mentions continue to rise after March 2020, coinciding with the steep decline in the S&P 500, we observe the rising concern of the people and the impact of the crash on overall sentiment.

3. Conclusion

In conclusion, this lesson has demonstrated the powerful role that social media data can play in financial analysis. We've explored how to retrieve and analyze data from major platforms like YouTube and Reddit, using API interactions and employing simple sentiment analysis techniques. These methods provide valuable insights into market sentiment, public opinion, and the trends that can impact financial markets.

The examples given are significant in the sense of what one can achieve with a **free** API. Keep in mind that data for sentiment analysis are hard to find and most likely expensive. But by carefully choosing your period and channels, subreddits, tweets, etc., you will be able to extract, quantify, and visualize the sentiment.