Sentiment

The goal is to find out if trading cryptocurrency based on the overall opinion about a cryptocurrency on Twitter is able to make profits.

Therefore, we need to find the opinions and metrics to objectively compare these opinions from People about a particular topic, in this case Bitcoin. This is where *sentiment analysis* comes in handy.

Sentiment analysis is a part of *Natural Language Processing (NLP)* to systematically identify and classify opinions from text. For every sentence we say or word we use, we kind of know if it has a positive or negative meaning. A program can't do that. It needs validated values for each word, saying if it's positive or negative, to calculate the overall opinion in a sentence. This is why a lot of teams around the world have built lexicons with thousands of words in it and rated each of them with a value to either be positive or negative.

There are a lot of different tools that apply these lexicons for NLP for different use cases, such as calculating and classifying sentiment polarity (positive or negative) from text-input.

Two of the primarily used tools for NLP for python are compared in the next section.

Tools for Natural Language Processing

TextBlob

TextBlob returns polarity and subjectivity of a sentence. Polarity lies between -1 and 1, defining negative and positive sentiment, respectively.

Subjectivity lies between [0,1]. Subjectivity quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains personal opinion rather than factual information. TextBlob has one more parameter — intensity. TextBlob calculates subjectivity by looking at the ‘intensity’. Intensity determines if a word modifies the next word. For English, adverbs are used as modifiers (‘very good’).

Vader

Vader (Valence Aware Dictionary and sEntiment Reasoner) does the same thing as TextBlob, but it is especially good for social media sentiment analysis, because it uses some heuristics to improve their calculations. They not only calculate the polarity (positive or negative), but also the intensity/valence of each sentiment on a scale from -4 to +4. Meaning, that the word "Good" has an intensity of 1.9, the word "Great" of 3.1. This value, coming from observation and experience, will be taken into account for the calculation. They also use intensifiers or degree modifiers to either increase or decrease the intensity. This is more accurate than models who only use a lexicon of words. They also understand emoticons like "😄" or emojis and many slang words like "kinda" (instead of "kind of").

Punctuation ("Good!!!!"), Word Shape ("ALL CAPS WORDS"), Intensifiers ("It is **extremely** hot"), Negation("The weather isn't really that hot") all shift the sentiment intensity.

Comparison

Getting the polarity from a sentence with these tools only requires a few lines of code.

```
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

text = TextBlob("Just came across this pretty good CNBC piece on SpaceX &
Starship")
print(f"TextBlob: {text.sentiment.polarity}")

vader = SentimentIntensityAnalyzer().polarity_scores(text).get("compound")
print(f"Vader: {vader}")
```

Output:

> TextBlob: 0.475
>
> Vader: 0.7684

As can be seen the difference is quite huge between those two values. The sentiment scores for the dataset of tweets were quite similar from both TextBlob and Vader. Considering, that Vader was built for social media sentiment analysis and uses special heuristics to improve the overall performance, compared to other NLP-Tools, Vader will be used to get the sentiment for each tweet in this project ([FR 20]).

Vader gives sentiment score between -1 and 1 for each tweet, normally meaning, that positive values mean a positive sentiment and negative values mean a negative sentiment. To differentiate even further, the decision has been made to classify values above 0.6 to be *"Very Positive"* and values below -0.6 to be '"Very Negative"*.

The final classification:

Sentiment Score	Meaning
> -1 & < - 0.5	Very Negative
>= - 0.5	Negative
0.0	Neutral
<= 0.5	Positive
> 0.5 & < 1	Very Positive

More Filters

To make better trading decisions some more filters and data cleaning after reading the tweet data from the database is needed.

Since sentiment analysis libraries are obviously not able to declare a sentiment polarity to all the words in the world, it tends to classify a lot of the words and thus their tweets as neutral. When going through the acquired database, there were a lot of *neutral* tweets. This would decrease the impact of positive and negative tweets in the calcuation and since most of the tweets were not really useful tweets anyway, they were excluded. Many neutral tweets contained bad grammar, random words, only hashtags or a lot of gibberish. Filtering out neutral tweets was a necessary step and is achieved with just one line of code See here.

With these sentiment values and their meaning at hand some more calculations will be made to build a strategy for signals, when a buy or sell order is made.
