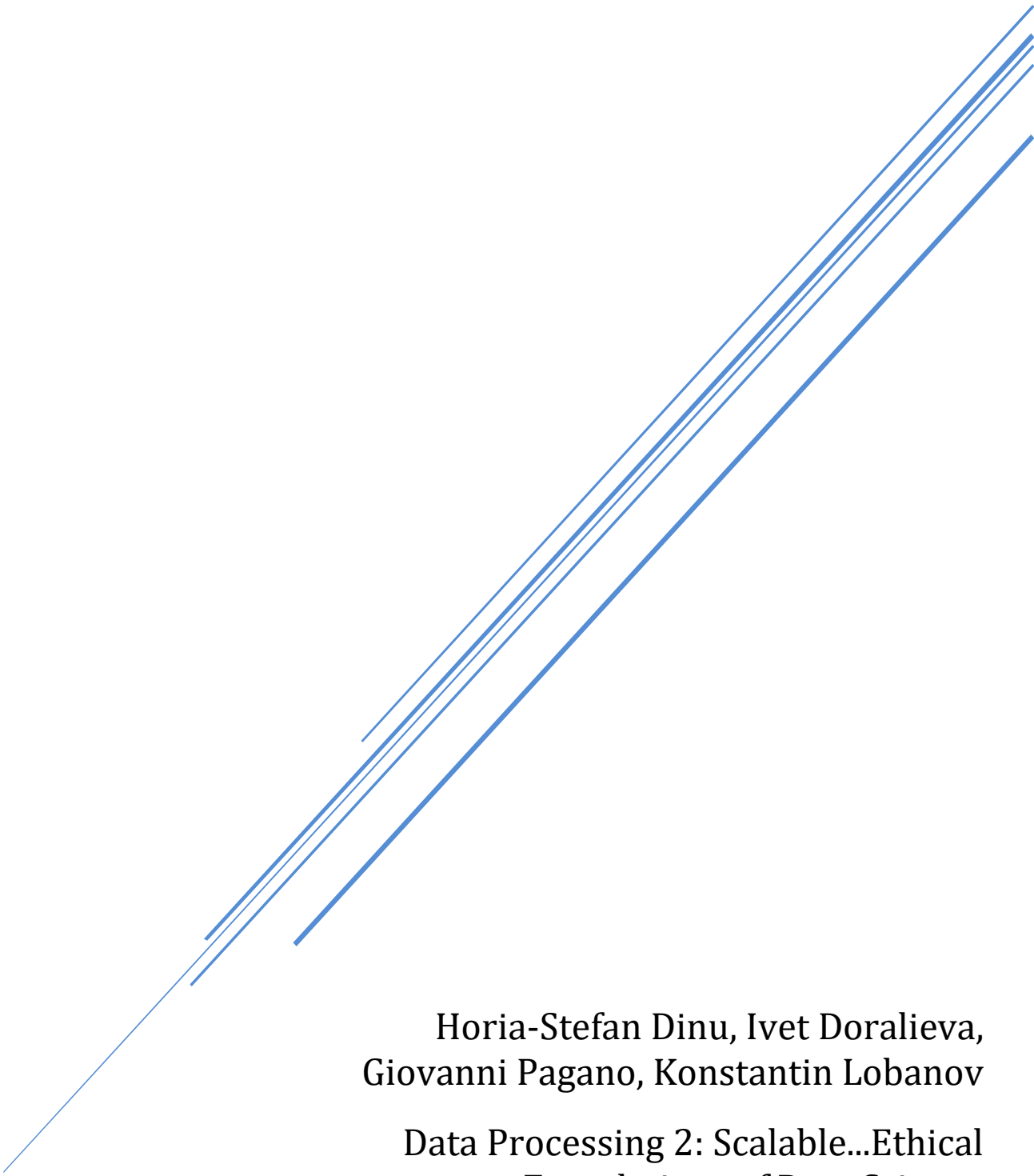
PREDICTION OF CRYPTOCURRENCY PRICES THROUGH TWITTER DATA

Project Report



Horia-Stefan Dinu, Ivet Doralieva,
Giovanni Pagano, Konstantin Lobanov

Data Processing 2: Scalable...Ethical Foundations of Data Science

Contents

Project Overview

Our project overview starts with a project description, in which we explain our goals, our approaches and our means of reaching the set goals. Afterwards, it is followed by a short discussion about Big Data, and why our project qualifies as a big data project. Next, we present our data sources and we explain how we gathered the data for this project. The most important part is the description of the proposed solutions, in which we take a look in detail at how we managed to implement our project. We discuss code, we show you graphs, and we try to explain our notebooks as clearly as possible. Moreover, we also present not only our data architecture diagram, but also the libraries and algorithms used

Project description

Our project aims at predicting the effect on cryptocurrency prices starting from a sentiment analysis of Tweets on the topic. We select only a handful of currencies, including Bitcoin, Etherum, Chainlink, Litecoin and Dogecoin, and by using the Twitter API we will retrieve Tweets on the subject. By conducting the sentiment analysis, we will categorize them into positive, negative or neutral based on the words used by authors. Once we have categorized the Tweets, we will see if they have influenced the transaction prices on the market, by comparing them to data on cryptocurrency prices. Therefore, we will try to predict by looking at the sentiment of Tweets if the market prices for cryptocurrencies have increased or decreased.

Analysis of Tweets

For analyzing the Tweets, we will use in the first place kafka live-streaming in order to gather our data and afterwards we will analyze everything in batches. But, in order to reduce spam, we will only stream tweets from a defined list of influencers which we believe are most likely to influence people into buying or selling cryptocurrencies. We will next conduct a sentiment analysis (a method of text analysis that detects polarity of an input) of the streamed Tweets and we will classify them as positive, negative or neutral and in order to reduce the missing values, we will group them based on timestamps, by constructing an average sentiment value for each minute.

Processing the Cryptocurrency data

We will obtain daily historical data on the cryptocurrency prices from the Nomics API, data which we will filter to only include the above-mentioned currencies: Bitcoin, Etherum, Chainlink, Litecoin and Dogecoin. Since our twitter sentiment analysis provides data for every minute of our stream, and we only have daily price information, we need to address this issue. In order to do so, we came up

with the solution of constructing a mathematical average price difference per minute, which should approximate the daily changes: we will subtract the price of the current day from the price of the next day for each cryptocurrency and then divide by 1440 (number of minutes in one day). Afterwards, we will also categorize this information, only keeping a boolean value for each minute: true if the price has increased and false if the price has decreased.

Machine Learning:

Since we are trying to do a prediction of a binary variable (we want to predict if prices have increased / decreased), we will use the Machine Learning algorithm Logistic Regression. As independent variables we will use the specifically tailored sentiment scores, grouped per minute, and the prices in the form explained above. In order to evaluate the model, we will look at the ROC curve.

Why do we work with big data?

This project qualifies as a Big Data project since it complies with The Five V's of Big Data, a concept defined and used by multiple publications:

- *Value*: The project will provide significant insights into the cryptocurrencies market, by researching one of the factors that influence the prices. Also, by predicting future trends, it could help investors make more informed decisions in the future;
- *Velocity*: New tweets are being generated every second and they require high-speed processing to update the results in real time (for reference, about 6000 Tweets are sent per second). In case we also implement close to real-time streaming through Apache Kafka and a cryptocurrency API, Velocity would be even more relevant for this project;
- *Volume*: By the sheer size of the Twitter community, the amount of generated data is enormous and our code will have to sort through a considerable number of tweets in order to select and classify only the necessary ones for the analysis. Moreover, also the number of cryptocurrency transactions is impressive, and even if we will not record all of them, we will deal with true Big Data in terms of volume;
- *Veracity*: Twitter data is uncertain, being subject to fake news, and with people's opinion being constantly influenced, we expect quick shifts in beliefs that require constant assessment of incoming information;
- *Variety*: by combining Tweets with transaction prices, we will use different types of data, with numerical information, as well as with plain text that will be subject to the sentiment analysis.

Data Sources

One of the primary sources for this project was streamed through the Twitter API data. With multiple attempts and different options discussed, we settled on selecting a rather small number of accounts to follow in order to avoid a large amount of spam and have an opportunity to gather more relevant data prior to reaching the limits. These users were recommended by multiple sources as influencers in the field and are thus credible authors. Unfortunately, we were unable to have a continuous stream over the course of the project (which was our original plan), so instead, we implemented a number of random streams for shorter periods of time in between the second and the fourth of February. Further, in preprocessing we only select a handful of currencies for a detailed analysis.

The other primary data source for this project was obtained from a cryptocurrency API. There are plenty of options available on the market, and considering the goals of our project, we had to choose one wisely. Since we needed historical data, we settled upon Nomics API, a Rest API which does not allow live streaming, but which provides us with the daily closing price of each cryptocurrency on the market. These prices are being taken from the sparkline endpoint, an endpoint accessible for free which calculates them based on OHLCV candles. OHCLV candles are basically datasets which in our case show the timestamps as the start of the day and the closing prices of that day.

While trying to simplify our project, for sentiment analysis, we chose VADER *(Valence Aware Dictionary and Sentiment Reasoner)*, a fully open-sourced lexicon and rule-based sentiment analysis tool, attuned to sentiments expressed in social media (Hutto, 2014). It can be used directly on raw data by assigning a score to each word and then calculating an average score of emotional intensity (Beri, 2020).

Description of the Proposed Solutions

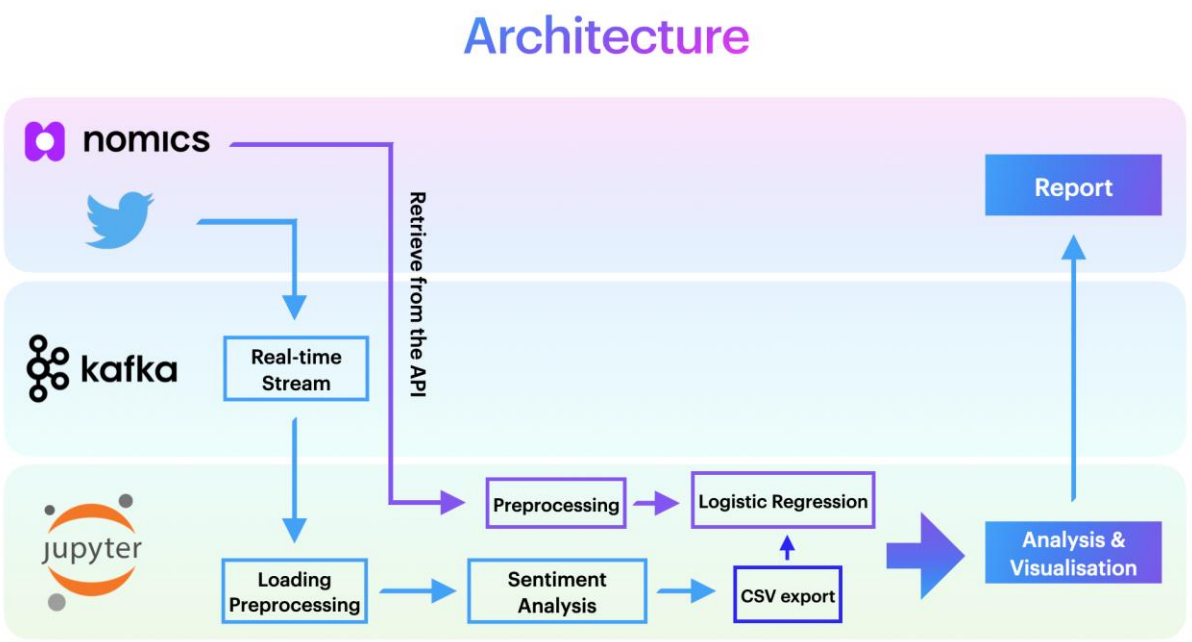
Architecture

In the diagram below, we present our data architecture. We first implement a real time stream from Twitter into kafka, using the *“TwitterProducer”* notebook. This data is later loaded to the *“TwitterSentiment”* notebook, the tweets are preprocessed and put through a sentiment analysis. The results are exported as a csv file for future use.

Additionally, the cryptocurrency data is retrieved from the Nomics API and saved into a json file in the *“CryptoData_Nomics”*. This file is subsequently uploaded into the *“CryptoData_Main”*. It undergoes further processing, at which point it is

joined with the pre-saved results of the sentiment analysis. The resulting dataframe is transformed and used to create a logistic regression model.

Finally, all of the retrieved materials are used to generate visualisations and to make a final analysis. A detailed description can be found in this report.



Twitter Streaming

The first part of the project requires us to stream real-time data from Twitter. The code can be found in the *"TwitterProducer"* notebook. The code was based on the notebook provided in the course materials, but we modified it to suit our needs.

In our first attempts of streaming, we wanted to filter directly by using hashtags that corresponded to the cryptocurrencies in question, however there were a number of problems. Firstly, the flood of tweets received was quickly filling up the available memory, which would prevent us from saving more than a few minutes of tweets, thus rendering the project useless. Secondly, most of the received data could be classified as spam, with no relevant meaning for our analysis. Therefore, we decided to take a couple of steps in order to prevent these difficulties: we added a list of trusted accounts and we manually compiled it into a csv, which we then uploaded in the Jupyter environment. We also modified the filter to follow these accounts, instead of tracking certain hashtags. This helped us in the first place to reduce the amount of spam in our project, but also to stream more data, filling up the available cluster slower.

In the notebook, we begin by initiating a Spark Session. Subsequently, we install and import the necessary packages, including tweepy, a python library used for accessing the Twitter API, and kafka-python, that allowed us to save the

streamed data into a Kafka topic. After setting up the access keys and functions, we uploaded and prepared the list of accounts to follow. Finally, we implement the stream.

Processing of Twitter Data

Further, we work with the streamed data from Twitter in the "Twitter-Sentiment" notebook. Before anything else, we need to set up the connection between spark and kafka. For that purpose, additional packages need to be added to the PySpark Submit Arguments, which we do in the first cell, before setting up a new Spark session.

Next, we install additional packages that will assist us in preprocessing the twitter data. The most important package at this stage is the Natural Language Toolkit. From there we will use a stopwords list and a perception tagger among others. We once again utilised an adapted version of the course materials for strapping our tweets from all unessential information (e.g. hash-signs, tagged accounts, irrelevant words among others). These functions are taken from the modified "preproc.py" file and are registered in the environment as user defined functions (UDF).

We load the entire data from the kafka topic, and extract only the significant fields: text of the tweet and a timestamp for integration with the financial data at the next stage. The aforementioned cleaning functions are applied to the resulting dataframe. Below we can see the dataframe before and after the preprocessing stage.

created_at	text		created_at	text
Tue Feb 02 17:43:...	RT @elonmusk: Off...		Wed Feb 03 08:27:...	dear elon release...
Tue Feb 02 17:43:...	RT @PeterMcCormac...		Wed Feb 03 08:25:...	haha guess bitcoi...
Tue Feb 02 17:43:...	@lopp @CasaHODL S...		Tue Feb 02 19:12:...	honesty matter fa...
Tue Feb 02 17:43:...	RT @twobitidiot: ...		Tue Feb 02 19:11:...	carry world versa...
Tue Feb 02 17:43:...	RT @cz_binance: @...		Tue Feb 02 17:58:...	lmfao
Tue Feb 02 17:43:...	RT @elonmusk: Off...		Tue Feb 02 17:56:...	bitcoin eth amp u...
Tue Feb 02 17:53:...	RT @cz_binance: I...		Tue Feb 02 19:12:...	meme time please
Tue Feb 02 17:53:...	@elonmusk I don't...		Tue Feb 02 17:59:...	stay please
Tue Feb 02 17:53:...	@coinbase Please ...		Wed Feb 03 08:24:...	thanks friend sha...
Tue Feb 02 17:53:...	RT @elonmusk: Off...		Wed Feb 03 08:27:...	thanks tesla team...
Tue Feb 02 17:53:...	RT @davidgokhshte...		Tue Feb 02 19:14:...	possible xrp mark...
Tue Feb 02 17:53:...	@elonmusk Good		Tue Feb 02 17:54:...	supporter bitcoin
Tue Feb 02 17:53:...	RT @elonmusk: Off...		Tue Feb 02 19:14:...	gamestop put bitc...
Tue Feb 02 17:53:...	@elonmusk Why not...		Tue Feb 02 19:13:...	miss know digital...
Tue Feb 02 17:53:...	RT @APompliano: C...		Tue Feb 02 17:57:...	stay busy big daw...
Tue Feb 02 17:53:...	@jack We don't wa...		Tue Feb 02 17:55:...	seem realize late...
Tue Feb 02 17:53:...	RT @100trillionUS...		Tue Feb 02 19:10:...	thanks best blog ...
Tue Feb 02 17:53:...	RT @pierre_rochar...		Tue Feb 02 19:14:...	hello sir uncurba...
Tue Feb 02 17:53:...	RT @elonmusk: Off...		Tue Feb 02 17:56:...	jst btt sun
Tue Feb 02 17:53:...	RT @PeterMcCormac...		Tue Feb 02 19:12:...	man elon cool thi...

Additionally, the timestamp's format was changed. Finally, we apply filters to the processed data, separating the main dataframe into 5 different ones, according to the cryptocurrency mentioned in the tweet.

Sentiment Analysis of Twitter Data

Originally, we intended to use different pre-made sentiment training data packages, but after careful examination, it was clear that we would not be able to present satisfying results with an unspecialized dataset. However, different sources recommended using VADER, an open source lexicon and rule-based sentiment analysis tool. As a part of the Natural Language Toolkit, it was a promising instrument that proved rather user-friendly in its implementation.

After downloading the necessary package and creating a Sentiment Intensity Analyzer, we can retrieve a polarity score of any text in a rather straightforward manner. In order to simplify our further work and achieve stronger results, we decided to rewrite the compound score into ones (for positive and negative, respectively) and zero (for neutral text). With that goal in mind, the custom "sentiment_score" function was added and defined as a UDF. Afterwards, it was applied to all the currency dataframes, adding the sentiment score as an extra column. Lastly, the dataframes were grouped by a time-window of one minute and the averaged sentiment scores were taken.

```
try:
    # grouping the created dataframes by timestamp (each minute) with the average sentiment score and sorting by timestamp
    tweets_btc_grouped = tweets_btc.groupBy(window("created_at", "1 minutes")).avg("sentiment").sort('window')
    tweets_eth_grouped = tweets_eth.groupBy(window("created_at", "1 minutes")).avg("sentiment").sort('window')
    tweets_link_grouped = tweets_link.groupBy(window("created_at", "1 minutes")).avg("sentiment").sort('window')
    tweets_doge_grouped = tweets_doge.groupBy(window("created_at", "1 minutes")).avg("sentiment").sort('window')
    tweets_ltc_grouped = tweets_ltc.groupBy(window("created_at", "1 minutes")).avg("sentiment").sort('window')

    # commented: showing the created dataframe
    tweets_btc_grouped.show()

except:
    # Print the error
    print("Unexpected error:", sys.exc_info()[0])
```

```
+--------------------+--------------------+
|              window|     avg(sentiment)|
+--------------------+--------------------+
|[2021-02-02 17:43...|                 1.0|
|[2021-02-02 17:53...|                 0.4|
|[2021-02-02 17:54...|0.16666666666666666|
|[2021-02-02 17:55...|                 0.2|
|[2021-02-02 17:56...|                 0.4|
|[2021-02-02 17:57...|                0.25|
|[2021-02-02 19:10...|                 0.0|
|[2021-02-02 19:11...|                 0.0|
|[2021-02-02 19:12...|                 0.0|
|[2021-02-02 19:13...|                0.25|
|[2021-02-02 19:14...| 0.3333333333333333|
|[2021-02-03 08:25...|                 0.6|
|[2021-02-03 08:26...|                 0.0|
|[2021-02-03 08:27...|                 0.5|
+--------------------+--------------------+
```

Retrieving cryptocurrency data

After selecting the cryptocurrency API as explained in the Data Sources part, we connect to the Nomics database using our API key. From the Sparkline Endpoint, we retrieve a json file. The Python library that we use, since it is an unofficial community developed library, has only limited parameters for each endpoint and thus we can only specify a time interval from which we want to gather data. The time interval we specify is from 01-01-2021 to the present day. Therefore, we receive much more information than we actually require, and the datafile is in need of preprocessing.

Accessing the endpoint

```
try:
    #getting the data
    data = nomics.Currencies.get_sparkline(
        start = "2021-01-01T00:00:00Z",
    )#returns a jsonfile
    print(data)
except:
    # Print the error
    print("Unexpected error:", sys.exc_info()[0])
```

```
[{'currency': '0241CX', 'timestamps': ['2021-01-04T00:00:00Z', '2021-01-05T00:00:00Z', '2021-01-06T00:00:00Z', '2021-01-08T00:00:00Z', '2021-01-
00Z', '2021-01-18T00:00:00Z', '2021-01-20T00:00:00Z', '2021-01-21T00:00:00Z', '2021-01-22T00:00:00Z', '2021-01-25T00:00:00Z', '2021-01-26T00:00:
21-01-28T00:00:00Z', '2021-01-29T00:00:00Z', '2021-02-02T00:00:00Z', '2021-02-03T00:00:00Z'], 'prices': ['2.82597368', '2.89080993', '2.92850937
96151', '2.89701150', '2.93352423', '3.52828789', '3.39551414', '3.58617564', '3.35608220', '3.34090436', '3.20079790', '3.11748961', '3.4023004
834327']}, {'currency': '0522CX', 'timestamps': ['2021-01-15T00:00:00Z', '2021-01-18T00:00:00Z', '2021-01-19T00:00:00Z', '2021-01-20T00:00:00Z',
-21T00:00:00Z', '2021-01-22T00:00:00Z', '2021-01-25T00:00:00Z', '2021-01-27T00:00:00Z'], 'prices': ['14.32811584', '15.36362424', '15.05358433',
2769', '16.95178850', '16.15905228', '16.40035562', '15.04469731']}, {'currency': '0XBTC', 'timestamps': ['2021-01-01T00:00:00Z', '2021-01-02T00
'2021-01-03T00:00:00Z', '2021-01-04T00:00:00Z', '2021-01-05T00:00:00Z', '2021-01-06T00:00:00Z', '2021-01-07T00:00:00Z', '2021-01-08T00:00:00Z',
09T00:00:00Z', '2021-01-10T00:00:00Z', '2021-01-11T00:00:00Z', '2021-01-12T00:00:00Z', '2021-01-13T00:00:00Z', '2021-01-14T00:00:00Z', '2021-01-
00Z', '2021-01-16T00:00:00Z', '2021-01-17T00:00:00Z', '2021-01-18T00:00:00Z', '2021-01-19T00:00:00Z', '2021-01-20T00:00:00Z', '2021-01-21T00:00:
21-01-22T00:00:00Z', '2021-01-23T00:00:00Z', '2021-01-24T00:00:00Z', '2021-01-25T00:00:00Z', '2021-01-26T00:00:00Z', '2021-01-27T00:00:00Z', '20
00:00:00Z', '2021-01-29T00:00:00Z', '2021-01-30T00:00:00Z', '2021-01-31T00:00:00Z', '2021-02-01T00:00:00Z', '2021-02-02T00:00:00Z', '2021-02-03T
Z', '2021-02-04T00:00:00Z', '2021-02-05T00:00:00Z', '2021-02-06T00:00:00Z', '2021-02-07T00:00:00Z', '2021-02-08T00:00:00Z'], 'prices': ['0.13644
12975086', '0.16400299', '0.15739367', '0.15093108', '0.17405645', '0.14739373', '0.14674646', '0.14912578', '0.17266492', '0.17555944', '0.1967
'0.18705471', '0.21638497', '0.18097649', '0.20923583', '0.22382855', '0.21572725', '0.25250830', '0.28065889', '0.19582160', '0.20715061', '0.2
'0.32975323', '0.30045513', '0.36566399', '0.31668780', '0.35634580', '0.40558086', '0.32875475', '0.30642648', '0.30680052', '0.36225662', '0.3
'0.32101335', '0.35317888', '0.39163359', '0.35604526', '0.38098937']}, {'currency': '0XMR', 'timestamps': ['2021-01-01T00:00:00Z', '2021-01-02T
```

A screenshot with the code for retrieving the data and a peak into the structure of the json file received

Processing cryptocurrency data

After gathering the data from Nomics, we parse it to a .json file which we import into the notebook dedicated to the Crypto Preprocessing and to Machine Learning. Here, we load it into a spark dataframe in order to apply preprocessing methods:

- First of all, using a SQL query we only select the cryptocurrencies which we will use in our analysis, namely Bitcoin, Etherum, Litecoin, Chainlink and Dogecoin
- Afterwards, we format the JSON file to create two new dataframes, one for timestamps and one for prices. We achieve this by exploding the initial column. This step is required in order to achieve a tidy dataset
- We join the two dataframes, by artificially creating a new id column in both of them (the order of the observations has remained unchanged)

- All of the data received from Nomics is in the form of strings and therefore we convert the timestamps to the unix date-time format

- We want to compute the difference in price from one day to another: in order to do so we create a column with the price of the next day and we compute the difference

```
try:
    spark.sql("set spark.sql.legacy.timeParserPolicy=LEGACY")

    # since there is no common column between these two dataframes we add row_index so that it can be joined
    crypto_times=crypto_times.withColumn('row_index', row_number().over(Window.orderBy(monotonically_increasing_id())))
    crypto_prices=crypto_prices.withColumn('row_index', row_number().over(Window.orderBy(monotonically_increasing_id())))

    #we join them
    crypto_final = crypto_times.join(crypto_prices, on=["row_index"]).drop("row_index").drop("currency.price")

    #we convert from a string to the unix date-time format
    crypto_final = crypto_final.withColumn("timestamps",to_timestamp(col("timestamps"), "yyyy-MM-dd'T'HH:mm:ss'Z'"))

    #creating the daily movement in price
    crypto_final = crypto_final.withColumn("id", monotonically_increasing_id())
    my_window = Window.partitionBy().orderBy("id")
    crypto_final = crypto_final.withColumn("prev_price", lag(crypto_final.prices).over(my_window))
    crypto_final = crypto_final.withColumn("difference", crypto_final.prices-crypto_final.prev_price).drop("prev_price")

    crypto_final.show()

except:
    # Print the error
    print("Unexpected error:", sys.exc_info()[0])
```

```
+--------+-------------------+--------------+---+-------------------+
|currency|         timestamps|        prices| id|         difference|
+--------+-------------------+--------------+---+-------------------+
|     BTC|2021-01-01 00:00:00|29602.67118832|  0|               null|
|     BTC|2021-01-02 00:00:00|31732.66795884|  1| 2129.9967705199997|
|     BTC|2021-01-03 00:00:00|32036.63426573|  2| 303.96630689000085|
|     BTC|2021-01-04 00:00:00|32006.14348387|  3|-30.490781860000425|
|     BTC|2021-01-05 00:00:00|33662.25779282|  4| 1656.1143089499965|
|     BTC|2021-01-06 00:00:00|36653.40744260|  5| 2991.1496497800035|
|     BTC|2021-01-07 00:00:00|39449.02099628|  6| 2795.6135536799993|
|     BTC|2021-01-08 00:00:00|40694.50922068|  7| 1245.4882244000037|
|     BTC|2021-01-09 00:00:00|40346.06198103|  8|-348.44723965000594|
```

A part of the dataframe after completing the above-mentioned steps (with the code that appears above)

- Afterwards we split the data into four dataframes, one for each currency and we select a period of time closer to the one we have in the twitter data: from the First of February to the Fifth of February

- We create a timeseries from each day in which we do not have only one price change observation per day, but one price change observation per minute

- We categorize the data as increasing/decreasing by computing a Boolean variable: true if the price increase in the past minute or false if it decreased

```
try:
    crypto_final = crypto_final.withColumn("prev_date", lag(crypto_final.timestamps).over(my_window))

    crypto_minutes = crypto_final.select('id','currency','prices','difference', \
                                         explode(expr('sequence(prev_date, timestamps, interval 1 minute)')).alias("timestamps"))

    crypto_minutes = crypto_minutes.withColumn("difference", crypto_final.difference/1440)

    crypto_minutes = crypto_minutes.withColumn("id", monotonically_increasing_id())

    # creating a categorical variable (to be used in ML)
    crypto_minutes = crypto_minutes.withColumn("category",crypto_minutes.difference>0)

    # commented: showing the resulting dataframe
    crypto_minutes.show()

except:
    # Print the error
    print("Unexpected error:", sys.exc_info()[0])
```

```
+---+--------+--------------+-----------------+-------------------+--------+
| id|currency|        prices|       difference|         timestamps|category|
+---+--------+--------------+-----------------+-------------------+--------+
|  0|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:00:00|    true|
|  1|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:01:00|    true|
|  2|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:02:00|    true|
|  3|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:03:00|    true|
|  4|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:04:00|    true|
|  5|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:05:00|    true|
|  6|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:06:00|    true|
|  7|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:07:00|    true|
|  8|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:08:00|    true|
|  9|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:09:00|    true|
| 10|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:10:00|    true|
| 11|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:11:00|    true|
| 12|     BTC|31732.66795884|1.479164423972222|2021-01-01 00:12:00|    true|
```

Shows a part of the dataset after creating price increase/decrease category and after exploding the timestamp

- As the last step of the preprocessing, we filter by currency, in this case by bitcoin (because we have most of the data for bitcoin)

After computing all the above steps, we are finally ready to start machine learning and combine the cryptocurrency prices dataframe with the sentiment analysis.

Merging Data & Logistic Regression

At this late stage in the project, the generated dataframe, containing prices for each minute is joined with the results of the sentiment analysis. An inner-join is applied, so that the prices and sentiment averages are matched by the timestamps, while all the missing values (since we don't have a high enough number of minutes streamed) are deleted.

A number of transformations are applied to the resulting dataset:

- The boolean variable (indicating an increase in the currency price for the period) is converted to an integer;
- The relevant for the logistic regression columns are selected.

In the “Logistic Regression” section of the notebook we import various PySpark ML modules (‘linalg’, ‘feature’, ‘classification’ and ‘evaluation’). After, the following steps are applied:

- A dense vector is created, the columns are renamed into “label” and “features”;
- The features are scaled;
- The data is separated into 70% training and 30% test sets;
- The logistic regression model is created and trained;
- The predictions on the test data are made and inspected.
- Two evaluation methods are used:
  - the area under the *ROC curve*,
  - manually calculated *accuracy*.

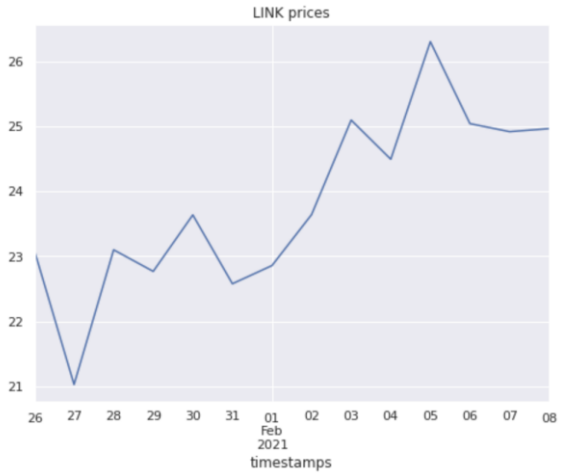
Interpretation and conclusions

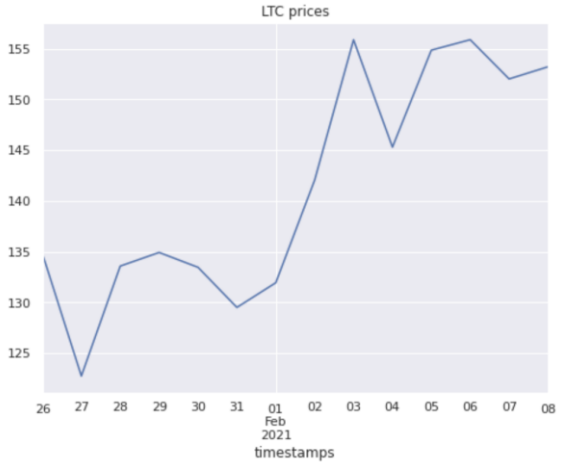
The evaluation results of our Logistic Regression can be seen below.

```
Test Area under ROC:  0.696969696969697
Accuracy :  0.88
```

The area under the ROC curve, that shows the relation of true and false positives rate, is almost 0.7, which indicates a fairly good value. The accuracy of the algorithm is at 88%, which further confirms a good model. It is, however, important to mention that other than having a small set of data, we also attempted a number of simplifications that opened us up to a multitude of biases (discussed further). Although the evaluation is overall positive, we need to be diligent in our process, before applying this model further.

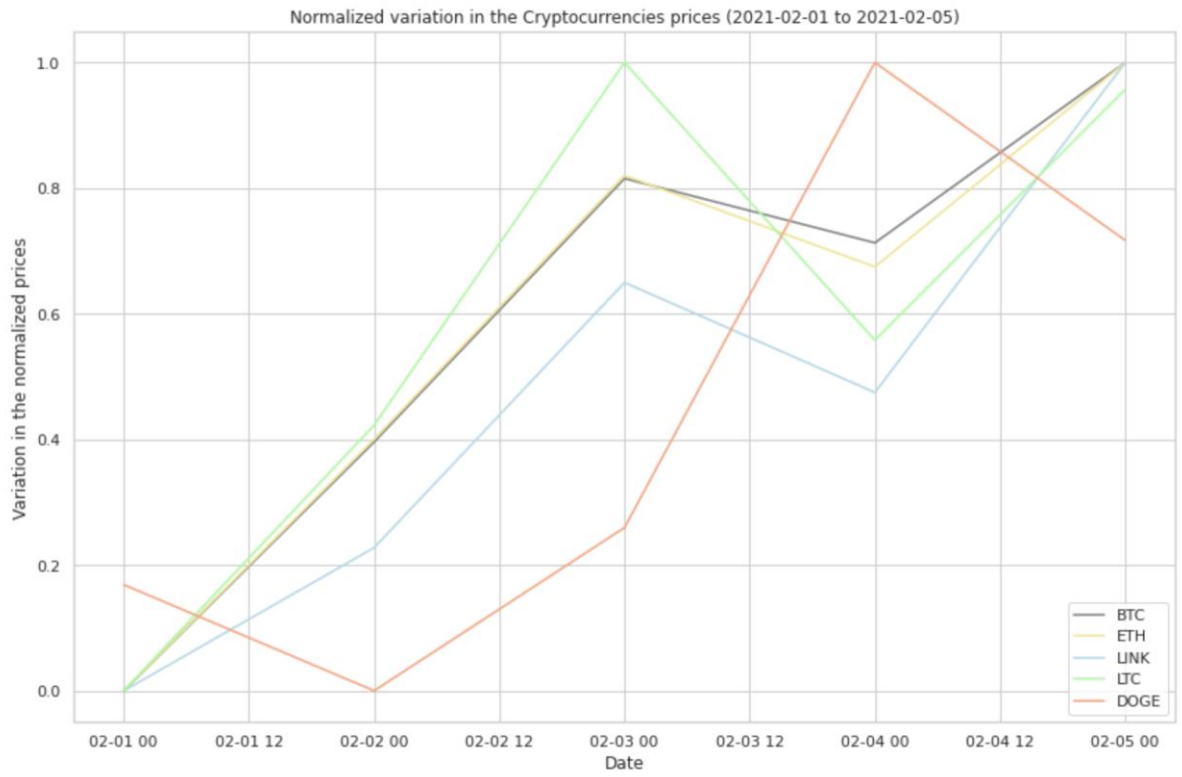
We can further look at the visualizations from the different notebooks. As we can see on the charts showing the price of each cryptocurrency in the period from January 26 to February 8, the price behavior is approximately the same. That is, we see not only general growth, but also movements in certain directions (growth or decline) at the same time, the charts are very similar to each other. This may indicate several things, for example, some general trends and events affecting all cryptocurrencies, as well as the dependence of some cryptocurrencies on others.



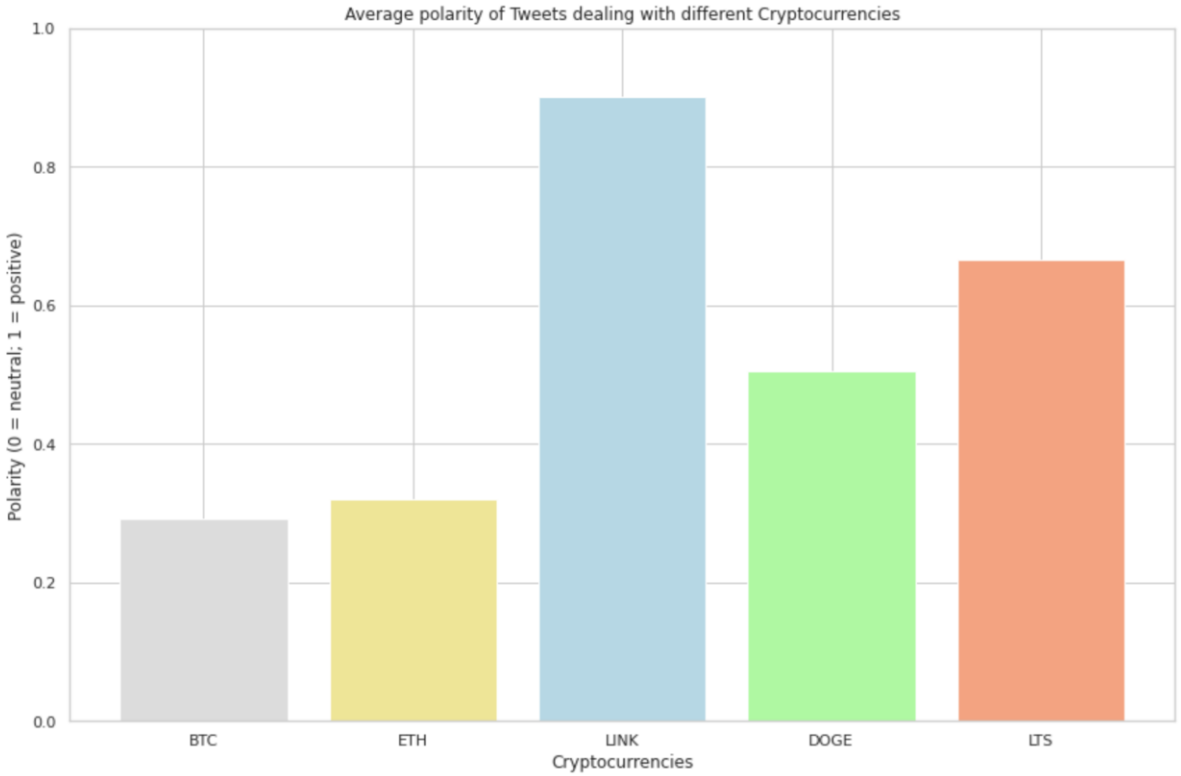


The normalized graph below confirms our observations regarding the similar behaviour of each of the cryptocurrencies, almost all cryptocurrencies not only move in a similar direction, but also start at about the same point and almost all end at the same point. The only thing that stands out here is the DOGE cryptocurrency, which has similar trends, but at some points behaves differently. This can be caused by several reasons. First, less dependence on

other cryptocurrencies or some common variables. Secondly, the growth against the background of the fall of other cryptocurrencies is also caused by the tweet of Elon Musk (Jolly, 2021).



When looking at the normalized chart of variations in cryptocurrencies, we can see the average upward trend. Despite a slight decline on February 4, after which prices not only recover, but also grow, there is a positive trend from February 1 to February 5. This time period is chosen to be slightly longer than our streaming timeframe, to observe possible correlations. As we can see on the graph showing the average tweet polarity for all cryptocurrencies is positive, that is, the trend is the same as in the change in the price of cryptocurrencies. For example, DOGE with a fairly high polarity of 0.5 shows the largest increase from February 3 to February 4. Although other cryptocurrencies have a higher polarity score, DOGE was heavily influenced by Elon Musk's tweet, that's why variation is higher.



Legal and Ethical Issues

In this section the focus shifts on the legal and ethical aspect of the project. The challenges faced are outlined and addressed one by one, by indicating both the means used to overcome them, and the legal and ethical guidelines followed along the process.

Finding meaningful information concerning our project on the intricate legal framework of the Twitter API, comprising all the different agreements, rules, policies, regulations, guidelines.

To overcome our concern, we relied upon the latest version (effective as of March 10, 2020) of the *Developer Agreement and Policy* (Twitter Inc., 2020a), which can be retrieved from the Twitter website for developers. In this text, all the specifications on, inter alia, the restrictions on use, the license we receive from Twitter, and the further terms to which the use of the Licensed Material is subject to (*Incorporated Developer Terms*) are listed. We have focused on the most important aspects that concern our project.
As per *section I paragraph B* of the *Developer Agreement and Policy*, we receive from Twitter a non-exclusive, royalty free, non-transferable, non-sublicensable, revocable license. The coverage of this license is rather broad and, given the

non-commercial purpose of our project, it leaves us enough leeway to implement the desired analyses.
Amongst the specifications on the permissible use of the licensed material, it is stated that we can:

1) *"Use the Twitter API to integrate Twitter Content into your Services or conduct analysis of such Twitter Content, as explicitly approved by Twitter;*
2) *Copy a reasonable amount of and display the Twitter Content on and through your Services to End Users, as permitted by this Agreement;*
3) *Modify Twitter Content only to format it for display on your Services".*

(Twitter Inc., 2020a, Section I par. B)

We have complied with all these points, and our usage of the Twitter API does not exceed these limitations. For example, we did not attempt to carry out any prohibited use of the API, as per section II paragraph A, such as reverse engineering, let alone trying to disrupt the integrity or performance of the Twitter Applications. Moreover, of particular importance are the "Rate Limits" (*section II paragraph D*) and accordingly we did not circumvent them in any way (Twitter Inc., n.d.-c, Rate limits). Another important aspect we have complied with is not removing or altering any proprietary notices or marks on Twitter Content received via the Twitter API. In the *Developer Policy* it is specifically asked to comply with this rule, so that people know where Twitter Content is coming from, and who it belongs to. Hence, we always openly state that a large portion of the data we have analysed has been retrieved from the Twitter API.
Arguably one of the crucial points that many Twitter policies, agreements and guidelines cover, is the respect of the users' privacy. We will have an in-depth look on this crucial issue while we answer how we overcame the next challenge faced.

Understanding in what way the data protection regulation/rules and twitter's privacy policy apply in the case of our project. We are initially using the user's id but then selecting only the text of the tweets, does this still pose a threat to the user's privacy? Do we have to handle the personal data that appears in tweets?

We have thoroughly read all the relevant information on the Privacy Policy of Twitter (Twitter Inc., 2020b), and on certain especially important parts of the *Developer Agreement and Policy*, namely *section XII paragraphs B* (User Protection) and *E* (Data Protection Addendum). Paragraph B focuses, as the title suggests, on the prohibited uses of Twitter Content that could threaten the user's protection and its rights; paragraph E provides a specification on the use of "Twitter European data" which shall follow the Twitter Controller-to-Controller Data Protection Addendum, so as to comply with the EU Directive 95/46/EC, the General Data Protection Regulation (GDPR) (Twitter Inc., n.d.-a) and any implementing legislation. Moreover, we have given a further regard to

the restrictions on use aimed at preventing the developers from taking any action that might cause harm to the users (Twitter Inc., n.d.-b).
With all the information we have acquired from these texts, we have come to the conclusion that in our project we do not face any risk on this matter, i.e. we do not pose any threat to the users' privacy. Apart from the initial filtering of the tweets, where we take into consideration just a preselected number of accounts (more on this below), we are only using the (preprocessed) text of the tweets and the time of publication. Moreover, we also do not face the risk of the users being traced back by combinedly looking at the timestamp and the text, because we are not using the single tweets but the aggregation of them. Hence, there is no way, or at least it would be extremely hard, to trace back the tweets to the single user. This means that we also do not have to be concerned about the possibility that in some tweets there might be personal data of the users.
Other reasons why we should not be worried in this regard are that we do not use (private) direct messages that the users exchange, and we also do not use any sort of location data (Twitter Inc., 2020a, Section II par. E) which, even if it would be allowed under certain restrictions, might pose a higher risk of the users being traced and their information being misused.
Furthermore, when users post a Tweet, they are aware that they are making a public statement and other users can interact with that statement. We have a further reassurance on this matter since we selected only "crypto influencers", hence, popular accounts of users who have a considerable number of followers, which makes us assume with reasonable belief that the people behind these accounts are well aware that when they post a Tweet, their statement will be available to the public.

Nomics API and Python wrapper: what are their limitations on use? What licences apply?

The limitations and the license applying to the Nomics API can be found in the *API & Data License Agreement* on Nomics' website (Nomics, 2020).
The type of license received and its scope are outlined in Paragraph 2:

"*limited, revocable, non-exclusive, non-transferable, non-sublicensable license during the term of the Agreement to: access our Data by using the API solely for your internal business purposes in developing Your Application"* (Nomics, 2020, par. 2).

Therefore, we understood that our use of the API is permitted and legal since we are using it for educational purposes which fall under the broader definition of "internal business purposes". In the same paragraph it is also stated that:

"*In order to use and access the Data and API, you must obtain an API Key through the registration process available at https://Nomics.com*" (Nomics, 2020, par. 2).

We followed this instruction and registered for a free API key. This plan comes with certain restrictions, the most significant being the limit on the endpoints, since we are only able to use the ones for: Sparkline graphs, Exchange Rates, ATH, & Supply. However, for our project this was not a great concern since we only needed the Sparkline endpoint.

Another important paragraph of the aforementioned agreement is number 18 concerning Rate Limits:

"*User is restricted to 100 requests per second. If user exceeds this, the Nomics API will send an HTTP 429 error message indicating that the rate limit has been exceeded"* (Nomics, 2020, par. 18).

This limitation also does not pose a significant limit in our analysis since we were not planning and we did not exceed the 100 requests per second, in fact we never encountered the HTTP 429 error.

Lastly, a further paragraph which must be taken into careful consideration is number 3, on the Use Restrictions. Similarly to the restrictions on the use of the Twitter API, it is prohibited to e.g. reverse engineer and remove any proprietary notices from the API. Moreover, is also forbidden to:

"use the Data or API in any manner or for any purpose that infringes, misappropriates, or otherwise violates any intellectual property right or other right of any person, or that violates any applicable law" (Nomics, 2020, par. 3).

Again, we complied with all these restrictions, especially did not violate any IPR, and we paid special caution to the correct attribution of the Nomics API in our notebooks and in our project.

In order to use the Nomics Crypto Market Data API in Python we relied on the unofficial Nomics Python API wrapper found on GitHub (Facen, 2020), and built by Ms. Taylor Facen from the Massachusetts Institute of Technology (MIT). This wrapper is covered by the MIT License (Facen, 2020, License), which fortunately does not pose any significant limitations. In fact, this is a permissive license allowing both private and commercial use, free of charge, and also free modification and distribution. The only conditions that apply are the preservation of the copyright (*Copyright (c) 2019 The Python Packaging Authority*) and the license notices. Hence, the scope of this license goes as far as exceeding the purpose of our usage, not posing any legal threat to the licit implementation of the project.

With what kind of restrictions comes the use of the VADER Sentiment Analysis tool?

We have found information on the Valence Aware Dictionary and Sentiment Reasoner (VADER) on GitHub (Hutto, 2014). This tool has been developed by Dr. CJ Hutto from the Georgia Institute of Technology, and similarly to the Nomics

Python wrapper is fully open sourced under the MIT License (Hutto, 2014, License). Thus, we are free (of charge) to use it for our analysis without limitations. The only difference with the Nomics wrapper is the copyright notice that must be preserved. In fact, given the different author, we now must mention, along with the MIT License, also: *Copyright (c) 2016 C.J. Hutto*.

Concerning ethical guidelines, certain codes of ethics and of conduct are meant for data scientists that work on projects for commercial purposes not for educational/research. Which ones should we follow? Are we risking to not follow certain of them throughout the project? How can we make our conclusions as ethical and honest as possible?

With the aid of the course's slides (Kirrane, S., 2020-2021) we have observed how certain ethical guidelines such as the *IEEE Code of Ethics* are primarily centred on the commercial spectrum of Data Science, while we are more interested in the educational part. Therefore, we have focused on specific guidelines presented in the *ACM Code of Ethics*, KDnuggets' *Key Ethics Principles for Big Data and Data Science*, and Emanuel Derman's *Hippocratic Oath of Modeling*. In particular, we have been keen on avoiding harm to others and respecting their privacy, as mentioned in the data protection part. Moreover, we have always paid particular attention to give proper credit, to respect licences and copyrights, and to collect minimal data by selecting only texts and dates of the tweets, which also allowed us to not incur in the risk of (unintentionally) discriminating users by e.g. gender, religion, race.
The collection of minimal data is one of the central causes of the biases that our project is exposed to, and this is why it is of crucial importance that we are clearly stating the limits of our analyses by specifying on which assumptions they are based on (Emmanuel Derman). Equally critical in this regard is the use of appropriate terminology when making our conclusions, following the guidelines of the Data Science Association's *Code of Conduct* (Data Science Association, n.d.). We were careful in using, e.g., correlation or reasonable belief, when it was the case, instead of stronger and more reassuring words that would not have represented the truthful conclusions that can be drawn from our analyses.

Did we have legal permission to take the most famous crypto Twitter accounts from Master The Crypto?

To get meaningful data from the Twitter API and avoid noise, we selected only a set of "crypto influencers" mentioned in the article "Most Popular Cryptocurrency Ifluencers: Top 236 Experts in 2020" from the website Master The Crypto. The latter is a knowledge hub and a resource center covering Cryptocurrencies and Blockchain. To answer our question on the legality of our action, we read carefully the Terms of Use of the website (Master The Crypto, 2018). The "Content" section is where we found the answer. In fact, it is stated

how the users retain any and all of the rights to any content posted and that they are responsible for protecting those rights. Hence, we would believe that the rights belong to the author of the article, namely Aziz Zainuddin. However, one matter to highlight is that Mr. Zainuddin is not just any user but the founder of the website. This could mean that the rights of this article are not with the author but of the exclusive property of Master The Crypto (as mentioned in the Terms of Use). Since it is not specified explicitly, we will assume that Mr. Zainuddin, although being the founder, is still considered as a regular user on the website. Hence, to strictly follow our ethical guidelines we have contacted the author privately and received his permission on the use of the information present in his article for the purpose of our project. Since we have simply taken the name of certain Twitter users and we are doing it for educational ends and not commercial ones, we believe that this would fall within the fair use. Hence, we could have limited ourselves in mentioning the author, the title and the website. However, to again abide by our ethical principles, we believed it was appropriate to ask the author himself.

What is the appropriate way to reference and cite all the sources (e.g. code/data) we used?

To overcome this challenge, we followed the valuable advice received by our professor Dr. Sabrina Kirrane (2020-2021) throughout the course, especially on how to appropriately list attributions and cite sources at the beginning of each notebook.
Moreover, for the Nomics API we have been required to give attribution in a specific manner as mentioned on their website (Nomics, n.d.), namely: *Crypto Market Cap & Pricing Data Provided By Nomics*, with a link to the Nomics' homepage. Likewise, the author(s) of the VADER sentiment analysis tool kindly request to reference them in a specific way, namely by citing their paper *A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text* by C.J. Hutto and Eric Gilbert.
For all the other sources used, where we were not asked to cite in a certain manner, we followed APA format.

Biases

The project is subject to numerous biases, many of which are intertwined amongst themselves.

Firstly, we are considering only a limited amount of Twitter users ("crypto influencers"). On the one hand, this could constitute a bias because it could be a non-representative sample of the sentiment among all users. On the other hand, we reasonably believe that selecting only the popular users makes our analysis stronger and that it better reflects reality since those users drive the sentiment of many others through their fame.

Secondly, a more pressing issue is that the technical constraints allowed us to stream the Tweets only for a limited time span. This might be the greatest source of bias in the whole project, because it could be the case that, in this small period, more (or less) events than usual occurred, making our analysis not reproducible at other times. An example of this can be seen for the striking increase in the prices of Dogecoin (Jolly, 2021) which happened during the small period of time streamed.

Thirdly, as good as the sentiment analysis tool can be, the tweets polarity might still be misleading. One reason could be that today's language is constantly developing and words that used to have a positive connotation might now be interpreted differently. Moreover, a user might use sarcasm which could be understood wrongly by the sentiment analysis tool, hence, negatively affecting our outcome.

Fourthly, an issue that intertwines two of the aforementioned biases is for the plots where we display the average polarity of the tweets for each cryptocurrency. This visualization might be heavily biased since we do not have the same number of tweets for each crypto. Hence, it might be that for one crypto we see a relatively high value only because we had a small number of tweets which were all "positive", leading into a misleading result.

To follow our ethical guidelines, we believe in the importance of specifying the biases and not deceiving a third party on the actual statistical power of our results.

Experience Gained

Discussion on the challenges encountered.

Our main challenges could be explained by the time and processing constraints. In the dedicated timeframe of the project we were not able to research the applications and tools we used to our satisfaction. Mainly our work was concentrated on setting an objective, trying to implement it using our personal knowledge base, encountering various problems and errors, searching for ways to solve them and, occasionally, getting stuck in that loop. Our belief is that with more time and experience, we could use a wider selection of methods and built-in capabilities of the programs.

The second largest constraint can be expressed by the limits that we had to work with, regarding streaming data and using the Twitter API, extracting data from Crypto data APIs, the jupyter environment and processing capacities specified by system administrators. With the kafka restriction we were only able to save a couple hours of tweets, which we later tried to overcome by pre-processing and saving the cleaned data in MongoDB, which as explained in the next subsection, did not work. It should be recognised that all those factors

potentially influenced our results and might have led to unanticipated biases and skews of data.
When facing the aforementioned restrictions, we had to make a number of simplifications in order to present a working version of the project, in order to meet the deadline.

- The first one of which, was the limit of accounts that the stream connected to. That also allowed us to decrease the quantity of “spam”, since in primary attempts to retrieve data (using hashtags, or search queries), only a small number of meaningful tweets was received;

- Second, we had to work with a slightly more than one-hour total stream time. Once we realized that our earliest tweets were disappearing, we could no longer stream new data. We tried saving the older tweets into text and csv files, sending processed data back to kafka and mongoDB, but all cases proved to be non-operational due the specifics of our data;

- Thirdly, due to a small sample size, we only decided to proceed with machine learning and detailed analysis of Bitcoin data. However, the principle remains the same, and we could easily use the same code to analyse the other currencies with more data.

Another challenge was adapting the course materials to suit our needs. For instance, we spent a considerable amount of time on switching the kafka parameters from reading a live stream, to retrieving all data from a certain topic (to be put through pre-processing in one batch).

Additionally, we experienced some malfunctioning on JupyterHub’s side. While we cannot reasonably assert so, our assumption is that during the last week of the project, the environment was due to all groups intensively working on their ideas. That resulted in slow loading, many timeout errors, restarts of kernels and servers, which significantly slowed our progress and we feel impacted the final results.

Another important problem was the lack of availability of free detailed historical data on cryptocurrency prices. We did not expect this to be a problem, but we were not able to find any hourly data on the evolution of the prices in the last few months. In order to overcome the crypto data problem, we had to resort to the mathematical approximation explained in the project description which for sure hampered our accuracy and our real-world applications of the project.

Finally, it is important to mention that due to the specifics of streaming data and the limitations of our project, it is hard to estimate the degree of confidence that we would be able to reproduce our analysis in a different period (since we are considering only a small timespan) and expect to have similar results. Many factors could have an influence on our data, and we are fully aware of all the biases that resulted from the aforementioned simplifications. Our hope is that

one day, we will have an opportunity to conduct a similar project without locking ourselves in a myriad of limitations.

The MongoDB problem:

Our plan for MongoDB was to save all the pre-processed data in a database, in different collections, in order to be easier to access when putting everything together in a new notebook dedicated to Machine Learning. Obviously, this did not work as intended and we once again encountered problems and errors, but this time we did not manage to solve all of them.

Connecting to MongoDB and uploading the data frames proved to be relatively easy to do, using the code that can be seen in the below printscreen. Still, a challenge here was sending the data frames to a dictionary, a process for which we had to resort to pandas. For the lengthy Twitter data this code was running for a significant period of time before showing any results. But, at least sending data to Mongo was completed successfully.

```
#creating a dictionary so it can be moved to mongo
crypto_final = crypto_final.toPandas().set_index('id').T.to_dict()
```

Sending the data to MongoDB

```
#installing pymongo
!pip install --user pymongo
```

```
import pymongo
import json
import os

#setting up mongodb and connecting to the wu servers

USERNAME = os.environ.get('JUPYTERHUB_USER')
POSTGRESQL_HOSTNAME = os.environ.get('POSTGRESQL_HOSTNAME')
MONGODB_HOSTNAME = os.environ.get('MONGODB_HOSTNAME')
DATABASES = json.loads(os.environ.get('DATABASES'))
n = 1 #selecting the first db to write in

SELECTED_DATABASE = DATABASES[n-1]['name']

CONNECTION_STRING = 'mongodb://{0}:{0}@{1}/{2}'.format(USERNAME, MONGODB_HOSTNAME, SELECTED_DATABASE)

mc = pymongo.MongoClient(CONNECTION_STRING)
db = mc[SELECTED_DATABASE]

# #creating collections for the data
mycol_crypto = db["crypto_data"]

#sending to mongo
x_btc = mycol_crypto.insert_one(crypto_final)
```

Code showing the way we sent the data to a dictionary and afterwards the connection to MongoDB and the insertion of the data in the database

Now, the real challenge came when we had to read from MongoDB back to a spark dataframe, in order to get going with the Machine Learning part. From MongoDB we were receiving a json file and we spent a significant amount of time trying to convert it to a Spark dataframe. For this purpose, we tried two different approaches, and neither of them were successful:

- Our main idea was to create a PySpark - MongoDB connection and to use the spark.read function to directly convert the mongo collection to a dataframe. This proved to be extremely error prone and it felt like every single time when we were making at least a bit of progress, a new error was thrown. We tried solving the error by using different ways to connect MongoDB and PySpark but in the end we failed to solve the issues.

```
import os
os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages org.mongodb.spark:mongo-spark-connector_2.11:2.3.1 pyspark-shell'
```

an attempt at establishing the connection

```
Py4JJavaError: An error occurred while calling o62.load.
: java.lang.NoClassDefFoundError: scala/Product$class
        at com.mongodb.spark.rdd.partitioner.DefaultMongoPartitioner$.<init>(DefaultMongoPartitioner.scala:64)
        at com.mongodb.spark.rdd.partitioner.DefaultMongoPartitioner$.<clinit>(DefaultMongoPartitioner.scala)
        at com.mongodb.spark.config.ReadConfig$.<init>(ReadConfig.scala:48)
        at com.mongodb.spark.config.ReadConfig$.<clinit>(ReadConfig.scala)
        at com.mongodb.spark.sql.DefaultSource.constructRelation(DefaultSource.scala:91)
        at com.mongodb.spark.sql.DefaultSource.createRelation(DefaultSource.scala:50)
        at org.apache.spark.sql.execution.datasources.DataSource.resolveRelation(DataSource.scala:344)
        at org.apache.spark.sql.DataFrameReader.loadV1Source(DataFrameReader.scala:297)
        at org.apache.spark.sql.DataFrameReader.$anonfun$load$2(DataFrameReader.scala:286)
        at scala.Option.getOrElse(Option.scala:189)
        at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:286)
        at org.apache.spark.sql.DataFrameReader.load(DataFrameReader.scala:221)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:566)
        at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
        at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
        at py4j.Gateway.invoke(Gateway.java:282)
        at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
        at py4j.commands.CallCommand.execute(CallCommand.java:79)
        at py4j.GatewayConnection.run(GatewayConnection.java:238)
        at java.base/java.lang.Thread.run(Thread.java:834)
Caused by: java.lang.ClassNotFoundException: scala.Product$class
        at java.base/java.net.URLClassLoader.findClass(URLClassLoader.java:471)
        at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:589)
        at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
        ... 23 more
```

One of the errors shown by the code

```
data = spark.read.format("com.mongodb.spark.sql.DefaultSource").option(CONNECTION_STRING,"%s.%s?authSource=admin"%(CONNECTION_STRING,'bitcoin')

data.load()
```

The code that throws the error

- We also tried the pythonic way of reading a json file, which only worked partially. It did not throw back any errors, but it was only saving the first column in the spark dataframe. We tried to improve the results by

specifying a custom structure for the dictionary, using the StructType() and StructField() functions, but without any success.

Therefore, since MongoDB clearly did not work in our case, in order to move on with the project we decided to ditch this plan of action and resort to something more rudimentary for the purpose of this project. Even if using a database would have been a lot easier, we used in the end a csv parser for the Twitter data.

Summary of experience gained by each team member.

Ivet Doralieva

I can confidently say that I am not the same person I was when I started the project. Naturally, I am happy to have had the opportunity to apply the theoretical concepts of the course in a practical setting. Personally, I had the opportunity to work with Kafka, Sentiment Analysis, Machine Learning as well as connecting the different parts of the project. Regardless of how trivial in some, and burdensome in other times this experience was, it is rewarding to see how separate ideas form into a whole, how each part flows and connects with another. There is nothing quite like the feeling of satisfaction one gets, once the error that you have been working on day and night, finally disappears and you can see the result of your hard work.

That being said, this experience did not bring as much joy, as one would hope. I'd say the underlying problem is that during this course, we were only vaguely familiarizing ourselves with the tools we were meant to use and had too little time to remedy that before setting on a project theme. We had to shoot in the dark and attempt to predict the skills that we would hope to acquire in the next few weeks, without having too much detail as to how this project would unravel. More specifically, our team was faced with various restrictions and limitations, that we were not well-equipped to handle.

This resulted in a series of pivots of the main idea with the goal of finishing in time. At the end, the result was very far from what we originally envisioned. In no way this means that the experience was less valuable. I definitely improved my problem-solving skills, as well as working under pressure. However, it did not leave us fully satisfied with our performance. If at the beginning of the project, I knew what I know today, in terms of the concrete blocks that stood in the way of implementing our original idea, I would do my best to adapt it early on and spend the majority of the time thinking of ways to enrich the project, rather than simply make it work.

Nevertheless, all experiences are good experiences, since we constantly learn from our mistakes. I have many ideas for my future work and the insights from this course will play a crucial role, when making them into something I can truly be proud of.

Giovanni Pagano

Completing this project has been one of the most challenging and at the same time exciting tasks I have faced in my university studies. It has been a rollercoaster of emotions, which I am glad to have experienced alongside my group peers.

At the beginning I felt overwhelmed by the many new terms and concepts, which made me be rather pessimistic on the foreseeable outcome of the project. This sentiment has unfortunately come back multiple times in different stages of the assignment, making it extremely difficult to keep a positive mindset and not giving up. However, during these delicate times, the great chemistry among all group members played a decisive role. I felt that I was not alone and that I could count on their support anytime I needed it, and likewise they could count on me.

Set aside the hardships, throughout this project I have been able to get a better sense of how Spark and Kafka work since I had not fully grasped them until then. This alone, is a sufficient reason to say that the project has been a valuable experience. Nevertheless, claiming that now I am an expert on the two softwares would be a lie, there are still certain aspects that I have not fully understood and that I will hopefully explore in my future studies.

The outcome of the project is not what I was hoping for, but I have no regrets because I know that we have put our heart and soul into it considering the time constraint and the many other exams we had to prepare. Moreover, on several occasions, especially in the last days before the deadline, I had serious doubts that we would even have anything meaningful to present. However, collectively as a united group we managed to overcome these struggles.

Finally, I cannot say that after having completed the project, I look back at all the challenges faced with great serenity, and with the thought that after all it was not as hard as initially predicted. In fact, it was possibly even harder than foreseen. However, this difficulty is what made this project so unique and stimulating, and what united the group even further.

Horia-Stefan Dinu

One of my responsibilities was setting up the cryptocurrency data for machine learning and one of the challenges I faced in the beginning of the project was picking the right API. I didn’t expect the “API industry” to be so well developed and since there were plenty of options to choose from, I had to analyse and research everything to find the best fit for our purposes, while of course using a free licence.

But without a doubt, the biggest problem I encountered was setting up MongoDB for storing the processed Twitter data and for moving the datasets between notebooks with ease. I think I spent around two days just trying to fix the Errors thrown at me by the code lines, and I got stuck in the end with a faulty JSON file which I couldn't convert to pyspark. Honestly, after spending such a long time on the same project part, without making any progress, I seriously considered just giving up. In the end, I am glad I found the required motivation to see everything through and now finishing up the project, the satisfaction I feel is worth losing a few nights over the same recurring issues.

Overall, in my opinion me and my team have come a long way in understanding the contents covered in this class: we worked and experimented with PySpark, with Kafka, with MongoDB and with Stack Overflow ... we did a lot of debugging with the help of Stack Overflow. And, our path to this final version of the project was filled with problems, bugs and just plenty of other difficulties that we had to overcome. In the end, I am not fully satisfied with how the project turned out, I think there is plenty of room for improvement both in terms of the technologies used and in terms of the approaches we took to reach our conclusions. On the bright side at times I was sure we would not have a working version in time for the deadline, but our determination helped us solve everything as a team. It was truly pleasing to see every team member stepping in for the others when needed.

To conclude, this was one of the most complex and difficult projects I was involved in, but I am glad I was part of it. I gathered knowledge and experience that will for sure help me in the long-term if I decide to work in the Data Science industry. Lastly, I would like to mention that if I would have to choose once more, even knowing the struggles and problems that I had to overcome, I would take on this project again.

Konstantin Lobanov

Before the start of this course and project, I could not even think about what things we would do and what difficulties we would have to face. It was during this course that I felt all the truthfulness of the video joke on the Internet, where the situation is shown, how data scientists write code in the view of others, and there is shown writing code at lightning speed and then how it happens, where it shows how a person gets an error, googles, reads on StackOverflow and the cycle repeats. During this project, I had to read a lot of posts on StackOverflow, as well as a lot of documentation for various libraries. However, it certainly gives a lot of knowledge and understanding.

To be honest, before the start of the project, I still did not fully understand how Kafka and Spark work and why they are needed at all. During the project, I figured out how it works, what happens to the data. My role in the project was preprocessing tweets, that is, turning raw tweets received using the API into text

ready for sentiment analysis. Also, I have been preparing cryptocurrency data and sentiment analysis for their visualization. Here it was necessary to prepare them in the correct format of the data frame, change the column type, divide the columns, filter, group, and all this with the help of different libraries. In the case of PySpark, some methods are used, in the case of Pandas-others, besides, there are restrictions everywhere. This project allowed me to learn how to understand what is best to use, how to properly prepare the data, how to change the format, how to filter the text, and prepare it for sentiment analysis.

Perhaps due to various limitations and problems that we encountered; our project did not turn out exactly as we imagined it. However, in my opinion, we have done the main work in terms of creating the architecture and code, then we just need to get more data, as well as other improvements described in recommendations for future work. But I still have learned a lot from this project, and I think that our team has done a very good job, despite the complexity and the huge number of limitations, difficulties, and bugs.

Recommendations for future work.

We can firmly say that this project has taught us a lot, and nevertheless, we see much room for improvement. To begin with, in order to make our results as accurate as possible, a much bigger data sample has to be compiled. Ideally, the stream should run for a few weeks and the rest of the algorithm should be modified to update in real time. Naturally, to make this possible some of the other restrictions will also have to be overcome, including twitter's policies and limits to new apps, the lack of processing power, among others.

It is important to understand that different influencers affect the market with different forces, that is, the force with which the tweet of a popular investor and the tweet of Elon Musk affects the change in the cryptocurrency is different. For example, one tweet by Elon Musk raised the price of dogecoin by 50% (Jolly, 2021). In order to correct for this, we could enter multipliers for each of the influencers, which will display how much the tweets of this account affect the market. In addition, we can also take into account retweets and likes to understand how popular a particular tweet is, which means it affects public opinion and may have an impact on the market. For example, a tweet from the same account, but with different popularity, will affect changes in the cryptocurrency in different ways.

Our list of accounts will have to be updated & expanded regularly. Alternatively, a spam-filtering algorithm could be implemented that would assist in working with a search-based stream, e.g. hashtags (instead of user-following).

The sentiment analysis has to be modified, here we can outline a couple of options, such as:

1. manually labeling as positive or negative a sample of topic-related tweets;
2. compiling an industry-related dictionary with specific terms, since the ones used in the sentiment analysis are not specific to crypto data in any way;
3. exploring various market alternatives for sentiment analysis and comparing their results.

Another area that could use improvement is the cryptocurrency data itself. Firstly, more accurate and timely data should be acquired (e.g. hourly) in order to improve the quality of the predictions of the model. Secondly, more metrics can be considered such as market cap, growth rates in addition to prices or OHLC candles. Thirdly, the dependence of cryptocurrencies on each other can be taken into account since the change of some can indirectly affect the change of others.

In terms of analysis, with a large enough sample we could experiment with building models that predict rises and drops in future financial performance, based on a live-sentiment analysis of twitter queries. A certain online dashboard that shows the prediction of the change of the cryptocurrency in the nearest period of time (i.e. in 5 minutes), and also displays graphs of accuracy, that is, the correspondence of predictions and reality. Based on this, it would be possible to display the best predictive cryptocurrencies at the moment. Moreover, we could also determine the time it takes for the prices of different cryptocurrencies to react to the changes in the sentiment of Tweets. For example, the longest-reacting cryptocurrencies will be better for the investor, since during this time it is possible to react and make a particular investment.

References:

Beri, A. (2020, May 27). *Sentimental Analysis Using VADER*. Towards Data Science. https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664

Data Science Association. (n.d.). *Code of Conduct | Data Science Code Of Professional Conduct*. http://www.datascienceassn.org/code-of-conduct.html

Facen, T. (2020, September 19). *Nomics-python*. GitHub/TaylorFacen. https://github.com/TaylorFacen/nomics-python

Jolly, J. (2021, February 4). *Price of dogecoin rises by 50% following Elon Musk tweet*. The Guardian.

https://www.theguardian.com/business/2021/feb/04/price-of-dogecoin-rises-by-50-following-elon-musk-tweet

Hutto, C. J. (2014, November 17). *VADER-Sentiment-Analysis*. GitHub/Cjhutto. https://github.com/cjhutto/vaderSentiment

Kirrane, S. (2020-2021). *Course materials*. Data Processing 2: Scalable Data Processing, Legal & Ethical Foundations of Data Science. Vienna University of Economics and Business.

Master The Crypto. (2018, May 25). *Terms of Use*. https://masterthecrypto.com/terms-conditions/

Nomics. (2020, September). *Nomics API & Data License Agreement*. https://p.nomics.com/api-liscense

Nomics. (n.d.). *Plans To Grow With You - FAQ*. https://p.nomics.com/pricing#attribution

Twitter Inc. (2020a, March 20). *Developer Agreement and Policy – Twitter Developers*. Twitter Developer. https://developer.twitter.com/en/developer-terms/agreement-and-policy

Twitter Inc. (2020b, June 18). *Privacy Policy*. Twitter Privacy Policy. https://twitter.com/en/privacy

Twitter Inc. (n.d.-a). *GDPR*. Twitter GDPR. https://gdpr.twitter.com/en.html

Twitter Inc. (n.d.-b). *More on restricted use cases – Twitter Developers*. Twitter Developer. https://developer.twitter.com/en/developer-terms/more-on-restricted-use-cases

Twitter Inc. (n.d.-c). *Rate limits*. Docs | Twitter Developer. https://developer.twitter.com/en/docs/twitter-api/rate-limits