Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY

Predicting Cryptocurrencies Exchange Rates Based on Cryptocurrencies News Sentiment Analysis

Big Data Project

MS3 Analytical Module

Authors:
Maciej Pawilkowski
Hubert Ruczyński
Bartosz Siński
Adrian Stańdo

Warsaw University of Technology
Faculty of Mathematics and Information Science
18 December, 2023

Contents

1 Business goal and potential benefits

The goal of this project is to create a system that enables its users to investigate the influence of the latest news articles on the exchange rates of cryptocurrencies. Our tool scraps current exchange rates, and recent news regarding cryptocurrencies in order to perform a sentiment analysis of those messages. Extracted features are provided in the time-series predictive model that will present estimated exchange rates of selected cryptocurrencies, based on archival data, the most recent trends, and sentiment.

The end users will be able to track current exchange rates, our predictions, and the latest news concerning selected cryptocurrencies. This information might help them make the best decision about when to exchange their money. As a result, they may save a lot of money or even find a way to influence main news sources to publish articles that would have a positive impact on selected exchange rates.

2 Description of data sources

In this section, we provide a detailed description of the data sources used in the implementation of our solution. In the **Record description** field, if the feature name is provided with `such formatting`, it means that we use it later on for model training.

2.1 Coincap

Description: Coincap provides API with real-time cryptocurrency data without an API key.
Category: Real-time cryptocurrencies values.
Number of records and inflow frequency: Inflow frequency: 1 per 10 seconds.
Record description: id, **symbol**, **currencySymbol** (Symbol of cryptocurrency, ex. BTC - bitcoin), type, **rateUsd** (the exchange rate in USD), and **timestamp**.
Collection time period: We will collect data during the winter holidays break for one week.
Data quality: Very high, no missing values.
Source format: JSON.
ML preprocessing: -
Requests Limits: 200 requests per minute, exchange values update each 10 seconds.
Link: https://coincap.io/

2.2 Alpha Vantage

Description: Alpha Vantage website provides API with financial market data such as time series stock data, digital & and cryptocurrency exchange rates, and most importantly market news data which we use in this project. Market news is both real-time and historical and comes from global news outlets. They cover stocks, cryptocurrencies, forex, and others. News data consists of a title, url to the article, summary, and source. Additionally, API

returns sentiment analysis information. However, we want to perform this analysis by ourselves and therefore we will not use this feature.
Category: Cryptocurrencies related news data.
Number of records and inflow frequency: Inflow frequency: all changes from the last hour (set with request parameters).
Record description: title, url, authors (a list), summary, banner_image (url), source (name of magazine, ex. Forbes), category_within_source, source_domain (web-page of source, ex. www.forbes.com), topics (a list of topics describing the news), overall_sentiment_score, overall_sentiment_label, ticker_sentiment, `time_published`.
Collection time period: We will collect data during the winter holidays break for one week.
Data quality: High, only columns not needed by us contain missing values.
Source format: JSON, does not contain article content.
ML preprocessing: We retrieve the sentiment with spaCy from provided texts, and provide only this information to the model.
Requests Limits: 5 API requests per minute, 100 API requests per day. Possible academic access to a premium plan with up to 1200 API requests per minute.
Link: https://www.alphavantage.co/

2.3 Crypto Compare

Description This API provides various crypto market data such as current price and trading information, blockchain data, and social stats for given coins. On top of that we can get information about the latest news articles related to cryptocurrencies. We can choose categories and source feeds of requested articles. Similarly to the previous data source returned news data consists of title, url to the article, summary, and source.
Category: Cryptocurrencies related news data.
Number of records and inflow frequency: API is called once an hour, it returns the last 100 articles. **Record description**: id, guid, imageurl, title, url, `body` (The first part of the main text), tags, lang (language), upvotes (number of users upvotes), downvotes, categories, source_info, source, `published_on`.
Collection time period: We will collect data during the winter holidays break for one week.
Data quality: Very high, no missing values, does not contain article content.
Source format: JSON.
ML preprocessing: We retrieve the sentiment with spaCy from provided texts, and provide only this information to the model.
Requests Limits: Lifetime - 250 000 calls, Day - 50 000 calls, Minute - 2 500 calls.
Link: https://min-api.cryptocompare.com/

2.4 CryptoPanic

Description JSON API that gives access to the most recent posts, indicators, and sentiment data. It offers news articles and updates from various sources related to cryptocurrencies and blockchain technology. It also returns readers' feedback in the shape of a number of

likes, dislikes, comments, etc.
Category: Cryptocurrencies related news data.
Number of records and inflow frequency: Inflow frequency: called once an hour, returns last 20 articles.
Record description: title, `timestamp`, id, url, `crypto` (a code for cryptocurrency which is talked about), `content` (small part of an article), `datetime`.
Collection time period: We will collect data during the winter holidays break for one week.
Data quality: High, no missing values, does not contain full article content.
Source format: JSON.
ML preprocessing: We retrieve the sentiment with spaCy from provided texts, and provide only this information to the model.
Requests Limits: We are limited to 200 last news posts, which we can request 5 times per second without a max request limit.
Link: https://cryptopanic.com/developers/api/about

2.5 NewsData

Description An API containing news articles regarding cryptocurrencies.
Category: Cryptocurrencies related news data.
Number of records and inflow frequency: Inflow frequency: all changes from the last hour (set by query parameter).
Record description: article_id, title, link, keywords, creator, video_url, description (a one sentence description of article), **content** (full content of article), `timestamp`, image_url, source_id, source_priority, country, category, language.
Collection time period: We will collect data during the winter holidays break for one week.
Data quality: High, only columns not needed by us contain missing values.
Source format: JSON.
ML preprocessing: We retrieve the sentiment with spaCy from provided texts, and provide only this information to the model.
Requests Limits: We are limited to 200 credits per day to search and download the News data. Each credit allows the download of 10 articles.
Link: https://newsdata.io

2.6 NewsAPI

Description It is a simple HTTP REST API for searching and retrieving live articles from all over the web. API doesn't return entire articles only title, description, first 200 signs, author, and source. It allows us to request articles based on language, publication date and even based on keywords or phrases.
Category: Cryptocurrencies related news data.
Number of records and inflow frequency: Inflow frequency: all changes from the last hour (set by query parameter).

Record description: source, author, title, description (2 sentences), url, urlToImage, content (partial, ends with..), `publishedAt`
Collection time period: We will collect data during the winter holidays break for one week.
Data quality: Very high, no missing values, or inconsistencies, article content is not available.
Source format: JSON.
ML preprocessing: We retrieve the sentiment with spaCy from provided texts, and provide only this information to the model.
Requests Limits: We have access to articles with 24-hour delay, and can't pull articles older than one month. We can also make only 100 requests per day.
Link: https://newsapi.org/

2.7 Yahoo news

Description: Searching with Google search engine (News tab) articles with keywords "yahoo finance <cryptocurrency name>" from last hour. In <cryptocurrency name> we put the names of all observed cryptocurrencies in our project. Afterward, the content of the article from the searched Yahoo websites is scrapped.
Category: Cryptocurrencies related news data.
Number of records and inflow frequency: Inflow frequency: all changes from the last hour.
Record description:-
Collection time period: We will collect data during the winter holidays break for one week.
Data quality: Quite high, article content is of high quality, but it includes various languages.
Source format: -
ML preprocessing: We retrieve the sentiment with spaCy from provided texts, and provide only this information to the model.
Requests Limits: ?
Link: -

3 Article content scrapping

Because almost all considered news sources return only links to the articles (without content), we decided to put another layer between NiFi and the data source - our custom HTTP server. Its main aim is to imitate having REST API, which returns also article content. This additional layer is implemented for all news sources (except NewsData, which is the only one to return content). It is implemented in Python using FastAPI and uvicorn libraries for the HTTP server, BeautifulSoup, and Selenium for scrapping. An additional preprocessing step done by this layer is converting dates (from different formats and attribute names) to timestamp format with the same attribute name. Unfortunately, some websites have implemented prevention mechanisms against text scrapers, and they provide meaningless text, informing that you cannot collect this data.

4 Used technologies

Technologies which are used in the project:

1. **Data flow:** `Apache NiFi 1.23.2`
   Raw data from APIs is downloaded and preprocessed using `Apache NiFi`. Cryptocurrency data is sent to both `HDFS Archive`, and `Apache Kafka Buffer`, as archival data is used as a back-up to stream continuous data for the project presentation. Additionally, for the last milestone, we plan to cyclically train the model, and recent data is used to fine-tune it. The crypto-news data is also sent to both places as we treat the batch layer as a kind of backup.

2. **Data storage:** `Apache Hadoop 3.2.1`, `Apache Hive 2.3.2` and `Apache Cassandra 4.0.11`
   As presented in Figure 2, cryptocurrencies and crypto-news data is saved in `HDFS Archive`. Additionally, we redirect the results of sentiment analysis here, whereas the stock predictions are accessible from the serving layer only. `Apache Hive` is used as a serving layer for the Batch Layer data, whereas `Apache Cassandra` serves the current data from a Speed Layer. Initially, we planned to use `Apache HBase` instead of `Apache Cassandra`, however, we could not figure out its proper configuration with `Apache Spark`. In `Apache Cassandra`, time-to-live was set to 6 hours.

3. **Data processing:** `Apache Spark 3.2.0`
   `Apache Spark` scripts are widely used in our project at multiple steps, as presented in Figure 2. First and foremost, it has components dedicated to NLP analysis such as calculating sentiment, thus it is used for the ML part of the project. Moreover, we use it to transform the data coming from `Apache Kafka Buffer`, and `HDFS Archive`, before they are transported to the respective Serving Layer views.

4. **Speed Layer Buffer:** `Apache Kafka 3.4`
   `Apache Kafka` serves as a buffer holding the recent cryptocurrency data used for the modeling stage.

5. **Presentation Layer:** `Tableau` or `Jupyter Notebook` with `Apache Spark`
   We have two strategies concerning the presentation layer. The first one focuses on the usage of Business Intelligence (BI) tools, like `Tableau` to create a clean Dashboard with proper visualizations. The approach is however not certain, as we are not sure how BI systems can interact with our data storage, and if they are enabled in free versions. The second one is a backup, where we prepare visualizations in `Jupyter Notebook`, and transform data via `Apache Spark`. Both approaches include a curve presenting recent exchange rates, and our prediction for the near future for selected cryptocurrencies. Moreover, on these plots, we will mark the timestamps of the latest news occurrences and print those news / their titles.

6. **Technological Stack:** `Docker`, `Tailscale` and `Portainer`
   Initially, we planned to create a fully distributed solution described in Figure **??**. Eventually, we resigned from this approach and decided that all `Dockers` would be run on Hubert's PC. The usage of `Tailscale`, lets us create a zero-configuration VPN, where the docker systems are placed. After logging into this network from the physical devices all team members are able to access (ping) proper containers. Finally, the `Portainer` is the web application that lets us log into each container by using the `Tailscale` network, so everyone can do things on the containers. Unfortunately, this solution still has major development drawbacks, as Hubert is the only one able to configure containers, so in practice, he had to be available all the time, anyone was working on the project.

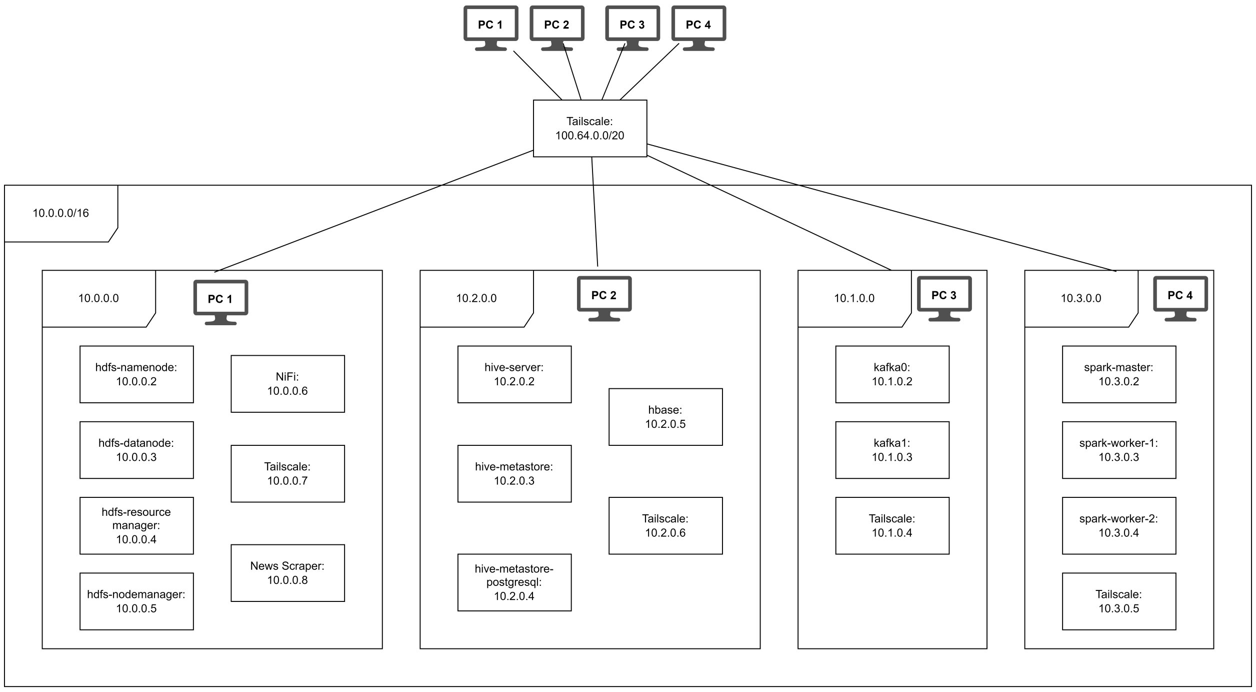


Figure 1: The initial idea of the project deployment. At first, we planned to create separate sub-networks with dockers distributed among the PCs of all our team members to provide more computational power. Eventually, after multiple hours spent on the failed attempts to properly configure such a network, we decided to resign from this approach.

4.1 Architecture Diagram

The figure 2 presents the design of our solution. It differs a bit from the solution proposed in MS2, as due to the complexity of the proposed ML task, we temporarily resigned from the batch training, and streamlined tuning of the model, which is why the ML task marked as red is deleted from the pipeline. Additionally, we changed the ML task in the Speed Layer, where we train the models instead of fine-tuning, and also make predictions. The last change, also marked as orange is the decision to use `Apache Cassandra` instead of `Apache HBase`, as we could not figure out its proper configuration with `Apache Spark`.

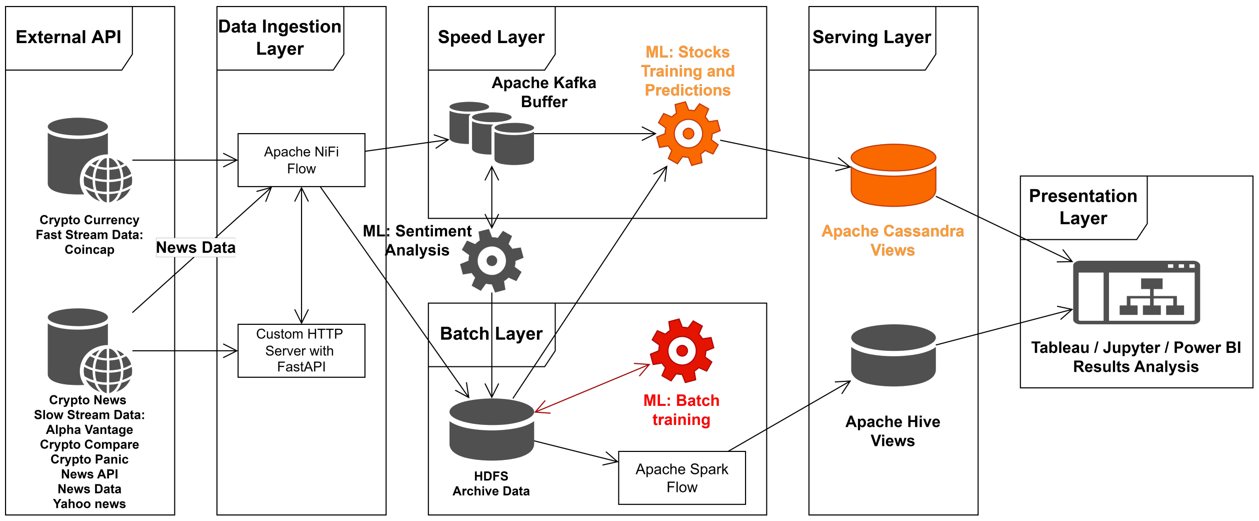
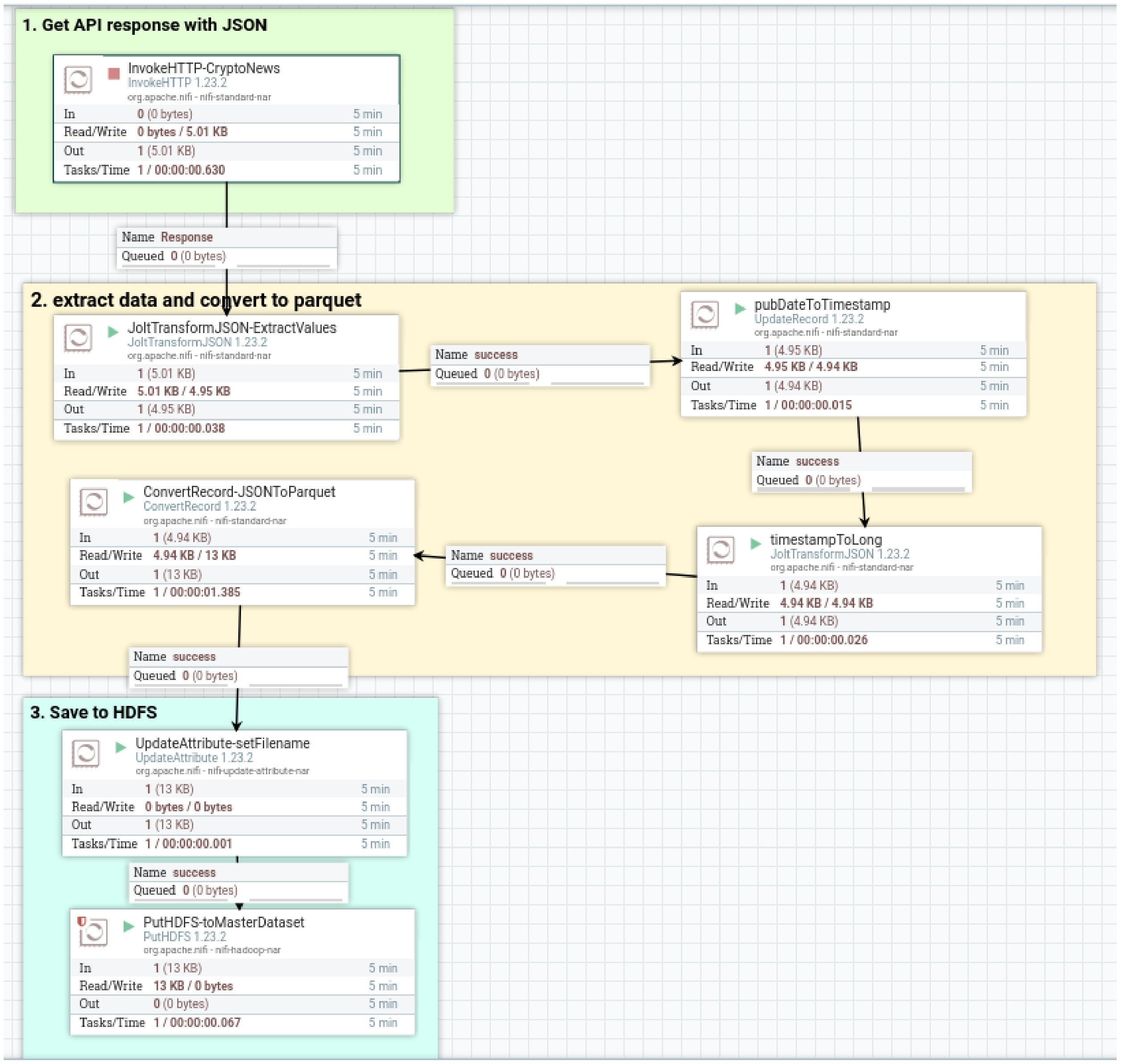

External API
Data Ingestion Layer
Speed Layer
Serving Layer
Presentation Layer
Crypto Currency Fast Stream Data: Coincap
News Data
Crypto News Slow Stream Data: Alpha Vantage Crypto Compare Crypto Panic News API News Data Yahoo news
Apache NiFi Flow
Custom HTTP Server with FastAPI
Apache Kafka Buffer
ML: Stocks Training and Predictions
ML: Sentiment Analysis
Batch Layer
ML: Batch training
HDFS Archive Data
Apache Spark Flow
Apache Cassandra Views
Apache Hive Views
Tableau / Jupyter / Power BI Results Analysis

Figure 2: Revisited architecture of our solution. We simplified the overly complex ML system, by resigning from the tuning step. Thus, we modify the ML component in the Speed Layer. Additionally, we replaced No-SQL `Apache HBase` with `Apache Cassandra`

5 Data Flow

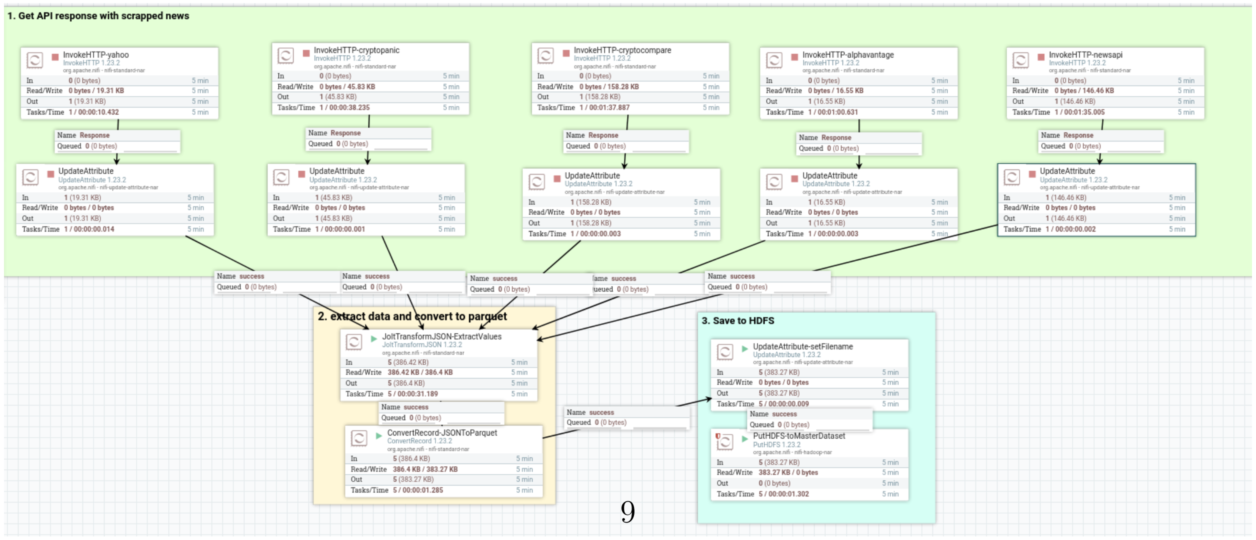
We use `Apache NiFi` to collect data from selected APIs and orchestrate the process of putting it into `HDFS` and forwarding it to `Apache Kafka`. We have designed four different NiFi flows for various sources. The first one considers the CryptoNews platform and is presented in figure 3. As we can see in this case we use InvokeHTTP to get answers from API, later we transform the data, convert it to parquet, and finally put it to `HDFS`.

Another flow from Figure 4 is used for the currencies exchange rate data, where we get the data, accumulate it for 60 seconds, convert it to AVRO format, and later put it into `HDFS`. Finally, the rest of our news article sources have similar data acquisition and transformation structures, presented in Figure 5.



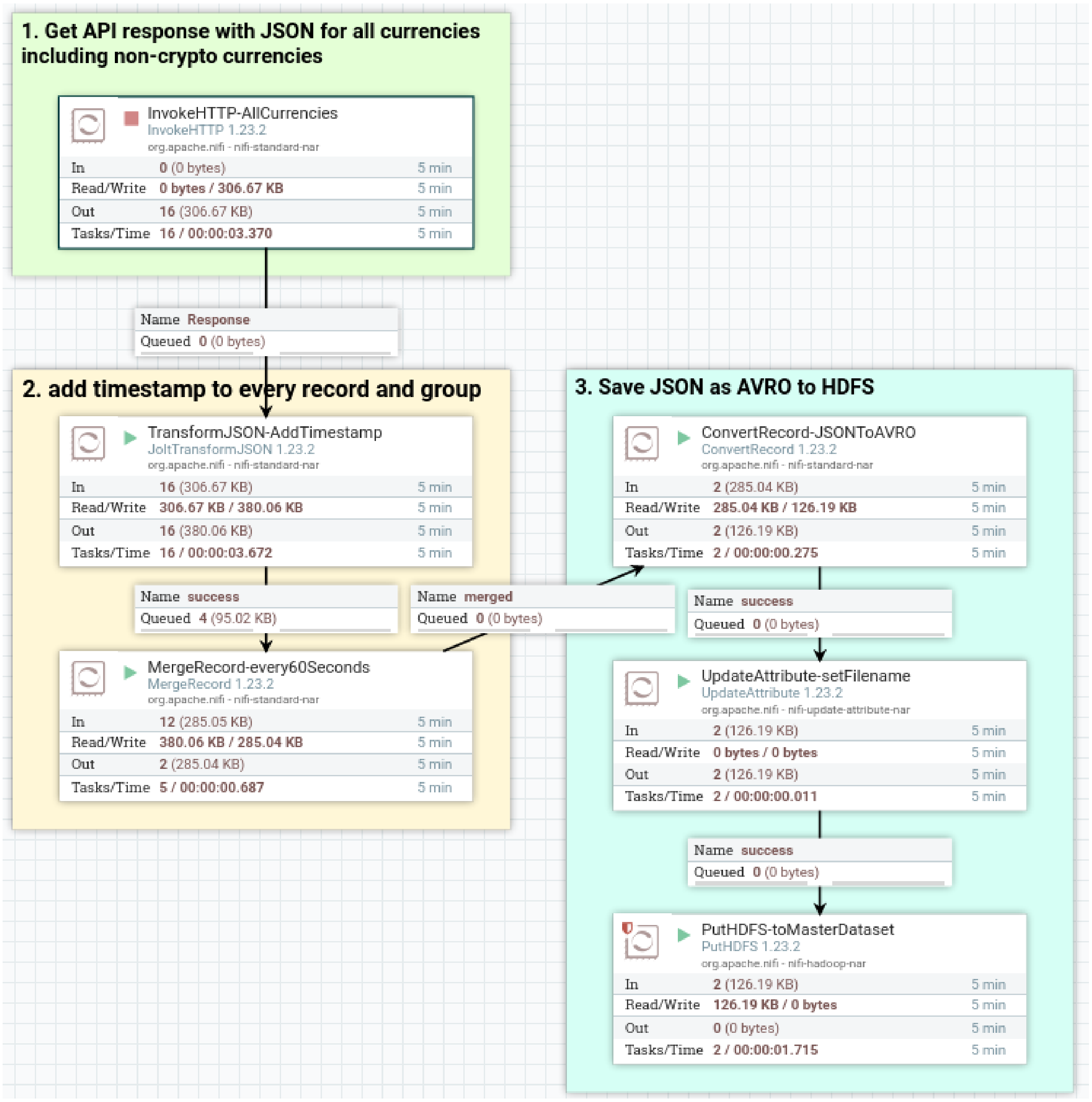
```
1. Get API response with JSON

InvokeHTTP-CryptoNews
InvokeHTTP 1.23.2
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)            5 min
Read/Write  0 bytes / 5.01 KB      5 min
Out         1 (5.01 KB)            5 min
Tasks/Time  1 / 00:00:00.630       5 min

Name    Response
Queued  0 (0 bytes)

2. extract data and convert to parquet

JoltTransformJSON-ExtractValues
JoltTransformJSON 1.23.2
org.apache.nifi - nifi-standard-nar
In          1 (5.01 KB)            5 min
Read/Write  5.01 KB / 4.95 KB      5 min
Out         1 (4.95 KB)            5 min
Tasks/Time  1 / 00:00:00.038       5 min

Name    success
Queued  0 (0 bytes)

pubDateToTimestamp
UpdateRecord 1.23.2
org.apache.nifi - nifi-standard-nar
In          1 (4.95 KB)            5 min
Read/Write  4.95 KB / 4.94 KB      5 min
Out         1 (4.94 KB)            5 min
Tasks/Time  1 / 00:00:00.015       5 min

Name    success
Queued  0 (0 bytes)

timestampToLong
JoltTransformJSON 1.23.2
org.apache.nifi - nifi-standard-nar
In          1 (4.94 KB)            5 min
Read/Write  4.94 KB / 4.94 KB      5 min
Out         1 (4.94 KB)            5 min
Tasks/Time  1 / 00:00:00.026       5 min

Name    success
Queued  0 (0 bytes)

ConvertRecord-JSONToParquet
ConvertRecord 1.23.2
org.apache.nifi - nifi-standard-nar
In          1 (4.94 KB)            5 min
Read/Write  4.94 KB / 13 KB        5 min
Out         1 (13 KB)              5 min
Tasks/Time  1 / 00:00:01.385       5 min

Name    success
Queued  0 (0 bytes)

3. Save to HDFS

UpdateAttribute-setFilename
UpdateAttribute 1.23.2
org.apache.nifi - nifi-update-attribute-nar
In          1 (13 KB)              5 min
Read/Write  0 bytes / 0 bytes      5 min
Out         1 (13 KB)              5 min
Tasks/Time  1 / 00:00:00.001       5 min

Name    success
Queued  0 (0 bytes)

PutHDFS-toMasterDataset
PutHDFS 1.23.2
org.apache.nifi - nifi-hadoop-nar
In          1 (13 KB)              5 min
Read/Write  13 KB / 0 bytes        5 min
Out         0 (0 bytes)            5 min
Tasks/Time  1 / 00:00:00.067       5 min
```

Figure 3: NiFi flow for CryptoNews. DESCRIBE IT MORE



```
1. Get API response with scrapped news

InvokeHTTP-yahoo
InvokeHTTP 1.23.2
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)            5 min
Read/Write  0 bytes / 19.31 KB     5 min
Out         1 (19.31 KB)           5 min
Tasks/Time  1 / 00:00:10.432       5 min

InvokeHTTP-cryptopanic
InvokeHTTP 1.23.2
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)            5 min
Read/Write  0 bytes / 45.83 KB     5 min
Out         1 (45.83 KB)           5 min
Tasks/Time  1 / 00:00:38.235       5 min

InvokeHTTP-cryptocompare
InvokeHTTP 1.23.2
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)            5 min
Read/Write  0 bytes / 158.28 KB    5 min
Out         1 (158.28 KB)          5 min
Tasks/Time  1 / 00:01:37.887       5 min

InvokeHTTP-alphavantage
InvokeHTTP 1.23.2
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)            5 min
Read/Write  0 bytes / 16.55 KB     5 min
Out         1 (16.55 KB)           5 min
Tasks/Time  1 / 00:01:00.631       5 min

InvokeHTTP-newsapi
InvokeHTTP 1.23.2
org.apache.nifi - nifi-standard-nar
In          0 (0 bytes)            5 min
Read/Write  0 bytes / 146.46 KB    5 min
Out         1 (146.46 KB)          5 min
Tasks/Time  1 / 00:01:35.005       5 min

Name    Response
Queued  0 (0 bytes)
(× 5)

UpdateAttribute
UpdateAttribute 1.23.2
org.apache.nifi - nifi-update-attribute-nar
In          1 (19.31 KB)           5 min
Read/Write  0 bytes / 0 bytes      5 min
Out         1 (19.31 KB)           5 min
Tasks/Time  1 / 00:00:00.014       5 min

UpdateAttribute
UpdateAttribute 1.23.2
org.apache.nifi - nifi-update-attribute-nar
In          1 (45.83 KB)           5 min
Read/Write  0 bytes / 0 bytes      5 min
Out         1 (45.83 KB)           5 min
Tasks/Time  1 / 00:00:00.001       5 min

UpdateAttribute
UpdateAttribute 1.23.2
org.apache.nifi - nifi-update-attribute-nar
In          1 (158.28 KB)          5 min
Read/Write  0 bytes / 0 bytes      5 min
Out         1 (158.28 KB)          5 min
Tasks/Time  1 / 00:00:00.003       5 min

UpdateAttribute
UpdateAttribute 1.23.2
org.apache.nifi - nifi-update-attribute-nar
In          1 (16.55 KB)           5 min
Read/Write  0 bytes / 0 bytes      5 min
Out         1 (16.55 KB)           5 min
Tasks/Time  1 / 00:00:00.003       5 min

UpdateAttribute
UpdateAttribute 1.23.2
org.apache.nifi - nifi-update-attribute-nar
In          1 (146.46 KB)          5 min
Read/Write  0 bytes / 0 bytes      5 min
Out         1 (146.46 KB)          5 min
Tasks/Time  1 / 00:00:00.002       5 min

Name    success
Queued  0 (0 bytes)
(× 5)

2. extract data and convert to parquet

JoltTransformJSON-ExtractValues
JoltTransformJSON 1.23.2
org.apache.nifi - nifi-standard-nar
In          5 (386.42 KB)          5 min
Read/Write  386.42 KB / 386.4 KB   5 min
Out         5 (386.4 KB)           5 min
Tasks/Time  5 / 00:00:31.189       5 min

Name    success
Queued  0 (0 bytes)

ConvertRecord-JSONToParquet
ConvertRecord 1.23.2
org.apache.nifi - nifi-standard-nar
In          5 (386.4 KB)           5 min
Read/Write  386.4 KB / 383.27 KB   5 min
Out         5 (383.27 KB)          5 min
Tasks/Time  5 / 00:00:01.285       5 min

Name    success
Queued  0 (0 bytes)

3. Save to HDFS

UpdateAttribute-setFilename
UpdateAttribute 1.23.2
org.apache.nifi - nifi-update-attribute-nar
In          5 (383.27 KB)          5 min
Read/Write  0 bytes / 0 bytes      5 min
Out         5 (383.27 KB)          5 min
Tasks/Time  5 / 00:00:00.009       5 min

Name    success
Queued  0 (0 bytes)

PutHDFS-toMasterDataset
PutHDFS 1.23.2
org.apache.nifi - nifi-hadoop-nar
In          5 (383.27 KB)          5 min
Read/Write  383.27 KB / 0 bytes    5 min
Out         0 (0 bytes)            5 min
Tasks/Time  5 / 00:00:01.302       5 min
```

Figure 5: NiFi flow for news articles data. DESCRIBE IT MORE


1. Get API response with JSON for all currencies including non-crypto currencies
2. add timestamp to every record and group
3. Save JSON as AVRO to HDFS

InvokeHTTP-AllCurrencies
InvokeHTTP 1.23.2
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 306.67 KB	5 min
Out	16 (306.67 KB)	5 min
Tasks/Time	16 / 00:00:03.370	5 min

Name Response
Queued 0 (0 bytes)

TransformJSON-AddTimestamp
JoltTransformJSON 1.23.2
org.apache.nifi - nifi-standard-nar

In	16 (306.67 KB)	5 min
Read/Write	306.67 KB / 380.06 KB	5 min
Out	16 (380.06 KB)	5 min
Tasks/Time	16 / 00:00:03.672	5 min

Name success
Queued 4 (95.02 KB)

MergeRecord-every60Seconds
MergeRecord 1.23.2
org.apache.nifi - nifi-standard-nar

In	12 (285.05 KB)	5 min
Read/Write	380.06 KB / 285.04 KB	5 min
Out	2 (285.04 KB)	5 min
Tasks/Time	5 / 00:00:00.687	5 min

Name merged
Queued 0 (0 bytes)

ConvertRecord-JSONToAVRO
ConvertRecord 1.23.2
org.apache.nifi - nifi-standard-nar

In	2 (285.04 KB)	5 min
Read/Write	285.04 KB / 126.19 KB	5 min
Out	2 (126.19 KB)	5 min
Tasks/Time	2 / 00:00:00.275	5 min

Name success
Queued 0 (0 bytes)

UpdateAttribute-setFilename
UpdateAttribute 1.23.2
org.apache.nifi - nifi-update-attribute-nar

In	2 (126.19 KB)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	2 (126.19 KB)	5 min
Tasks/Time	2 / 00:00:00.011	5 min

Name success
Queued 0 (0 bytes)

PutHDFS-toMasterDataset
PutHDFS 1.23.2
org.apache.nifi - nifi-hadoop-nar

In	2 (126.19 KB)	5 min
Read/Write	126.19 KB / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	2 / 00:00:01.715	5 min

Figure 4: NiFi flow for cryptocurrencies exchange rates. DESCRIBE IT MORE

Additionally, similar data flows were designed to put obtained data into `Apache Kafka`.

It is important to notice, that the input for `HDFS` and `Apache Kafka` are raw data without additional filtering. This step is done directly before the model training, however, in the middle, we invoke an `Apache Spark` script which takes out the data articles from `Apache Kafka` and computes the sentiment scores for it, to later put that information in both `Apache Kafka` and `HDFS`.

The next step of the data flow is getting the outcomes from models present in the speed layer, and forwarding them to `Apache Cassandra` in the serving layer. Additionally, we also forward the data from `HDFS Archive` to `Apache Hive`. The data from both serving layer sources will be finally used in the presentation layer.

6 Exploratory Data Analysis (EDA)

The EDA was conducted on 2 representatives of each data type, the cryptocurrency exchange rates, as well as the texts of news articles related to cryptocurrencies.

In the first dataset, only one column had missing values, however, it was a not important column with the symbol of the given cryptocurrency (we also have its name), so it does not matter. Gathered information regarding 17 currencies: DAI, ZEC, LTC, XPRT, DASH, CRO, QTUM, ETH, DVPN, BTC, USDT, BNB, WAVES, EOS, DOGE, RUNE, BCH, and contain 14 452 unique timestamps.

In cases of news, there are more missing columns, such as keywords, creator, video_url, or image_url, however, as in the previous case, those features are not necessary for our analysis. The dataset contains 508 news articles written in English where 192 of them mention Bitcoin, 63 Ethereum, and 16 Doge Coin.

More interesting analysis, where we choose three exemplary currencies, and plot their value in dollars are presented in the following figures: Figure 6 for Bitcoin, Figure 7 for Ethereum, and Figure 8 for DogeCoin. The lines on those plots represent the moments of publishing articles mentioning given cryptocurrency. If the color is red, it means that the sentiment of this text is negative, whereas the blue lines represent the positive sentiment. It is also important to know that we were collecting the data only when we were developing the solution, so the flat lines indicate the periods when the data was not collected.

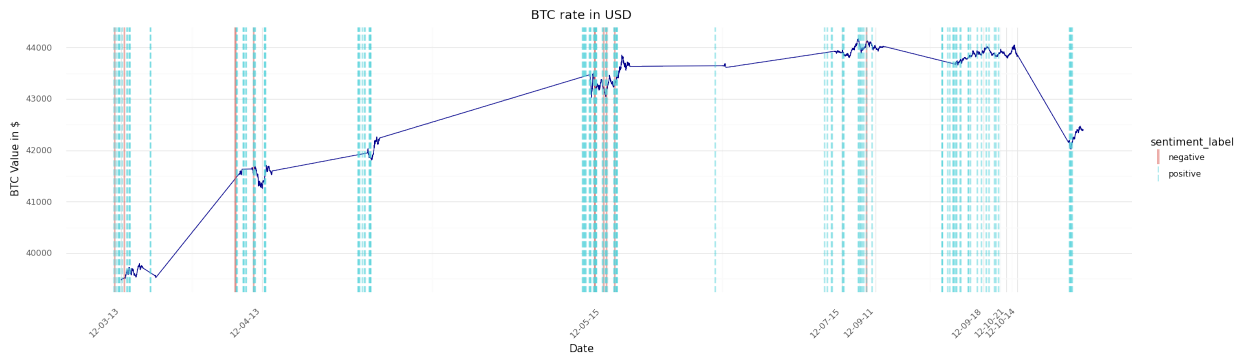


Figure 6: Representation of gathered Bitcoin data. The line plot shows the exchange rates of Bitcoin. Red vertical lines represent the appearance of the article that mentions this currency, and has negative sentiment, whereas the blue line corresponds to positive sentiment.

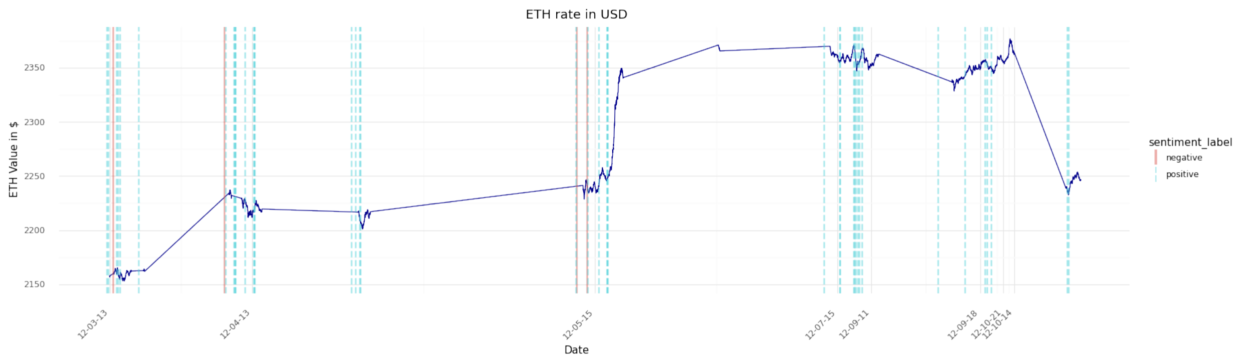


Figure 7: Representation of gathered Ethereum data. The line plot shows the exchange rates of Ethereum. Red vertical lines represent the appearance of the article that mentions this currency, and has negative sentiment, whereas the blue line corresponds to positive sentiment.

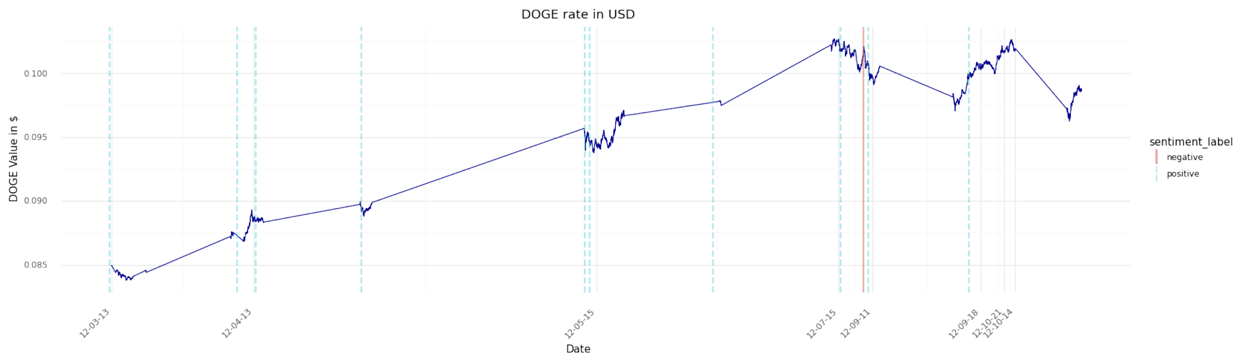
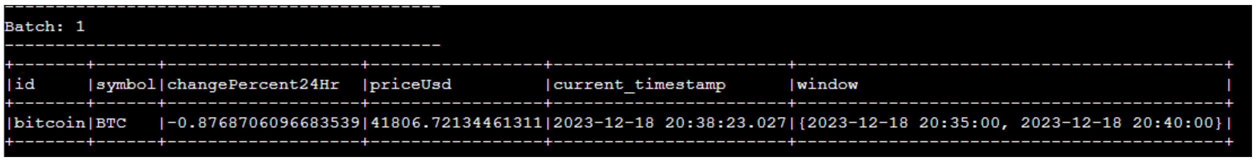


Figure 8: Representation of gathered Doge Coin data. The line plot shows the exchange rates of Doge Coin. Red vertical lines represent the appearance of the article that mentions this currency, and has negative sentiment, whereas the blue line corresponds to positive sentiment.

7 Data Processing, Filtering, and Aggregation

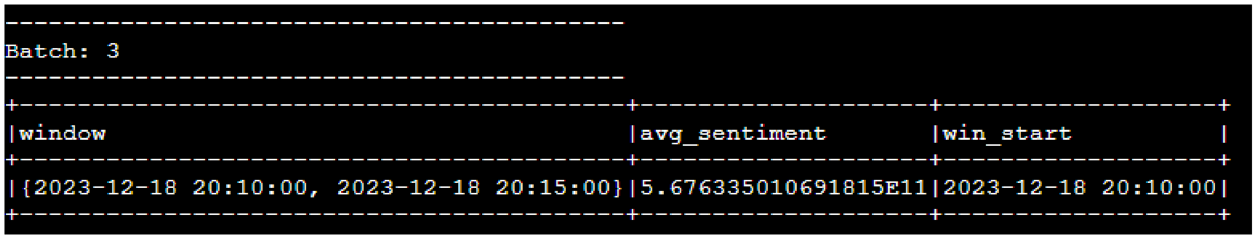
For real-time cryptocurrency rate forecasting, we are using data streams in `Apache Spark`, namely via PySpark. Firstly, we load the data from `Apache Kafka`. Currency rates we load to one stream and news data to another.
For currency rates, we do not have to do much preprocessing as API provides us with high-quality data. We only added a filter so that we have separate streams for each cryptocurrency (for now we only support Bitcoin). We also added timestamps, 5-minute windows, and 6-minute watermarks, both of which are necessary for stream joining. We plan to use those windows as our key based on which we will perform joins and watermarks to enable the system to manage late data.
As we want to focus on each cryptocurrency separately we first searched the contents of each article looking for words like "Bitcoin", "BTC", and "Ethereum", ... to determine which cryptocurrency is given article is about. After that, we need to deal with the frequency problem as articles do not show every 10 seconds as currency rates. To add sentiment to the currency rate data stream we had to group sentiment by 5-minute window visible in Figure

10. In each window, we calculate the average sentiment. After calculating sentiment we join it to the currency rate stream (9) using the time window as a key, which results in the stream presented in Figure 11.



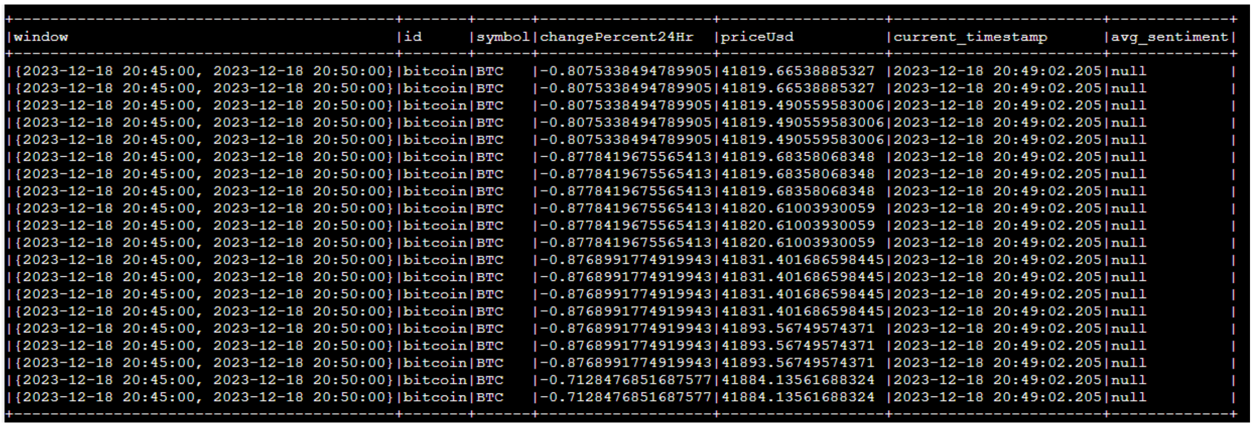
```
-------------------------------------------
Batch: 1
-------------------------------------------
+-------+------+-------------------+-----------------+-----------------------+------------------------------------------+
|id     |symbol|changePercent24Hr  |priceUsd         |current_timestamp      |window                                    |
+-------+------+-------------------+-----------------+-----------------------+------------------------------------------+
|bitcoin|BTC   |-0.8768706096683539|41806.72134461311|2023-12-18 20:38:23.027|{2023-12-18 20:35:00, 2023-12-18 20:40:00}|
+-------+------+-------------------+-----------------+-----------------------+------------------------------------------+
```

Figure 9: Currency rate stream data. A single record contains the cryptocurrency ID, its symbol, how the price changed in the last 24 hours, the price in USD, the observations timestamp, and the window it fell into.



```
-------------------------------------------
Batch: 3
-------------------------------------------
+------------------------------------------+-------------------+-------------------+
|window                                    |avg_sentiment      |win_start          |
+------------------------------------------+-------------------+-------------------+
|{2023-12-18 20:10:00, 2023-12-18 20:15:00}|5.676335010691815E11|2023-12-18 20:10:00|
+------------------------------------------+-------------------+-------------------+
```

Figure 10: Stream data with sentiment. A single record contains the cryptocurrency window it fell into, average sentiment, and the start time of the window.



```
+------------------------------------------+-------+------+-------------------+------------------+-----------------------+-------------+
|window                                    |id     |symbol|changePercent24Hr  |priceUsd          |current_timestamp      |avg_sentiment|
+------------------------------------------+-------+------+-------------------+------------------+-----------------------+-------------+
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8075338494789905|41819.66538885327 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8075338494789905|41819.66538885327 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8075338494789905|41819.490559583006|2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8075338494789905|41819.490559583006|2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8075338494789905|41819.490559583006|2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8778419675565413|41819.68358068348 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8778419675565413|41819.68358068348 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8778419675565413|41819.68358068348 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8778419675565413|41820.61003930059 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8778419675565413|41820.61003930059 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8778419675565413|41820.61003930059 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8768991774919943|41831.401686598445|2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8768991774919943|41831.401686598445|2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8768991774919943|41831.401686598445|2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8768991774919943|41831.401686598445|2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8768991774919943|41893.56749574371 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8768991774919943|41893.56749574371 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.8768991774919943|41893.56749574371 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.7128476851687577|41884.13561688324 |2023-12-18 20:49:02.205|null         |
|{2023-12-18 20:45:00, 2023-12-18 20:50:00}|bitcoin|BTC   |-0.7128476851687577|41884.13561688324 |2023-12-18 20:49:02.205|null         |
+------------------------------------------+-------+------+-------------------+------------------+-----------------------+-------------+
```

Figure 11: Stream data after join. The stream which represents the results of join of streams from Figure 9, and 10.

8 Analytical Module

8.1 Sentiment analysis

We have performed sentiment analysis on streamed scraped articles using Spark-NLP library[1]. For this purpose, we have used a sentence-oriented pretrained pipeline which is

[1]https://sparknlp.org/

available in Spark-NLP Model Hub[2]. It takes text documents and analyzes sentiment in English using naive-bayes. The model returns sentiment in the form of a "positive" and "negative" label for each sentence. Then the mode is calculated to return the overall sentiment for the whole document.

8.2 Machine Learning forecasting

We have performed the forecasting of the rate values by using the curve-fitting approach. For each one-minute window and for each currency, we fitted a linear function. Next, the prediction was estimated by calculating the value of a function at the next point of the curve. The proposed approach deals with missing data out-of-hand as, if there is any, the curve is fitted to the smaller number of training points.

9 Key Performance Measures (KPMs)

Our system requires from 10GB to 16GB of RAM memory in order to work, which means that if we run it on local machine we realistically require 32GB of RAM. Additionally, strong processor is recommended, as the designed system tends to use full computational power of our computing unit. The mentioned PC has 32GB of RAM memory available, and has the following processor: 11th Gen Intel(R) Core(TM) i7-11700KF @3.60GHz.

Additionally, we intend to measure the quality of our models by calculating RMSE, and MAE, by comparing the predicted results with real data that get into the system some time after making the predictions.

The last measure is average processing time, which might be useful, although we cannot precisely measure this value for all observations, as our system gathers the lagged data from 5 minutes periods, so their presence inside the pipeline will vary by this value.

10 Further Works

In this section we provide some plans for the improvement during the last milestone.

10.1 ML Task Improvement

One of the areas with possible further improvement is the sentiment analysis, where sentiment is calculated for the whole document rather than only for each sentence which needs further aggregation. However other tested solutions that process whole documents are based on pretrained Deep Learning models and out-of-the-box pipelines weigh around 0.5 GB. This would mean that whenever we start a new Spark session we would need to download this model repeatedly. This results in out-of-memory errors and is not a practical solution. We have to find different methods or partially implement parts of the pipeline ourselves to overcome this issue.

[2]https://sparknlp.org/models

The Machine Learning forecasting model will be replaced with a more complex approach. We plan to use a tree-based model, for example, LightGBM or XGBoost, which will be periodically retrained using the most recent data.

10.2 Presentation Layer

Our main goal is to deploy the interface with the usage of Business Intelligence (BI) solutions such as Tableau, or Power BI in order to provide a simple-to-use interface, however, we are not sure if `Apache Cassandra` and `Apache Hive` are supported within the free tiers of those solutions. In case they will not work we plan to provide a Jupyter Notebook with proper elements.

Regardless of the technical aspect we plan to provide a line plot that shows us the previous values of a given cryptocurrency and mark the forecasts for at least 3 different time horizons. Additionally, we plan to provide a visualization that shows us the appearances of articles regarding one, or more cryptocurrencies in the given time window, and mark whether their sentiment scores were positive or negative.

11 GitHub Repository

All source codes, documentation, scripts, configuration files, and instructions can be found in our GitHub repository [3]. The main README contains information about the used technological stack, including each component version. We provide detailed information on how to set up and run the project on a singular machine, including the crucial prerequisites. Additionally, we describe where each service is running inside our network, so you can easily modify the solution for your needs or improve its performance. We intend to make it comfortable to use, so you can find lots of scripts that enhance the user experience by running some services and configurations automatically. Additionally, we share a development log, describing encountered issues, in the form of Encountered issues.md file. The most important files stored on the repository are in the COMPOSE_* folders. Each of those represents some part of our solution, its configuration files, and mounted volumes.

12 Work Division

Table 1 presents the division of work among the team members for the whole project. All of us agree, that the division of workload was fair, and balanced.

[3]https://github.com/adrianstando/Big-Data-System-Cryptocurrencies

Table 1: Revisited work division between the team members. The table presents an in-depth work division throughout all phases of the project. The acronyms of particular people are: MP - Maciej Pawlikowski, BS - Bartosz Siński, HR - Hubert Ruczyński, AS - Adrian Stańdo.

Task Name	**Milestone**	**Team Members**
Project Planning	MS 1	AS, HR
Data Source Analysis	MS 1	BS, MP
Writing a report	MS 1	AS, BS, MP, HR
NiFi Flow to HDFS	MS 2	AS
Initial Docker architecture	MS 2	AS
Presentation	MS 2	MP
Testing	MS 2	BS
Writing a report	MS 2	HR
Presentation	MS 3	AS, BS, MP, HR
Docker management support	MS 3	AS
NiFi Flow to Kafka	MS 3	BS
Kafka Configuration	MS 3	BS
Initial ML Implementation	MS 3	AS, BS, MP
Stream data preprocessing	MS 3	BS, MP
Read and write tests	MS 3	MP
Docker management and setup	MS 3	HR
Exploratory Data Analysis	MS 3	HR
Repository documentation	MS 3	HR
Writing a report	MS 3	HR
Presentation	MS 4	AS, BS, MP, HR
Advanced ML Implementation	MS 4	AS, BS, MP
Presentation Layer	MS 4	HR
Writing a report	MS 4	HR