Faculty of Mathematics and Information Science

WARSAW UNIVERSITY OF TECHNOLOGY

Predicting Cryptocurrencies Exchange Rates Based on Cryptocurrencies News Sentiment Analysis

Big Data Project

MS4 Final Report

Authors:
Maciej Pawilkowski
Hubert Ruczyński
Bartosz Siński
Adrian Stańdo

Warsaw University of Technology
Faculty of Mathematics and Information Science
15 January, 2024

Contents

1 Business goal and potential benefits **2**

2 Description of data sources **2**
2.1 Coincap . . . 2
2.2 Alpha Vantage . . . 2
2.3 Crypto Compare . . . 3
2.4 CryptoPanic . . . 4
2.5 NewsData . . . 4
2.6 NewsAPI . . . 4
2.7 Yahoo news . . . 5

3 Article content scrapping **5**

4 Used technologies **6**
4.1 Architecture Diagram . . . 7

5 Data Flow **9**

6 Exploratory Data Analysis (EDA) **11**

7 Data Processing, Filtering, and Aggregation **13**
7.1 Streaming data . . . 13
7.2 Batch processing . . . 14

8 Analytical Module **14**
8.1 Sentiment analysis . . . 14
8.2 Machine Learning forecasting . . . 14

9 Key Performance Measures (KPMs) **15**

10 Results **16**

11 Quality assessment from business perspective **19**

12 Potential Impact **20**

13 Summary **20**

A Resources **21**

B GitHub Repository **21**

C Work Division **21**

This article is not written by the exchange rate experts and has severe limitations. Do not use the provided system, or its idea for any financial decisions. This document is not in the character of a business advisory.

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 to the time-series, online, predictive model that will present estimated exchange rates of selected cryptocurrencies based on recent data. The archival data is also maintained to analyze past trends and article occurrences.

The end users are able to track current exchange rates, our predictions, and current evaluation scores for our models, as well as analyze past data. 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.

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, mar-

ket 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` (a 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 and 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 a 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. Afterwards, 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 also returns 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 3, cryptocurrencies and crypto-news data are 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 3. 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 modelling stage.

5. **Presentation Layer:** `Power BI`
   We had two strategies concerning the presentation layer. The first one focused on the usage of Business Intelligence (BI) tools, like `Tableau`, to create a clean Dashboard with proper visualizations. The approach was, however, not certain, as we were 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 wanted to prepare visualizations in `Jupyter Notebook`, and transform data via `Apache Spark`. Eventually, we decided to

use `Power BI` software, with additional support from ODBC drivers for `Cassandra` and `Hive` acquired from CData with the usage of 30 days trial. We created a dashboard with 3 panels, and the solution offers semi-real-time plotting, as we can refresh the plots at any point in time, but it does not happen automatically.

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.

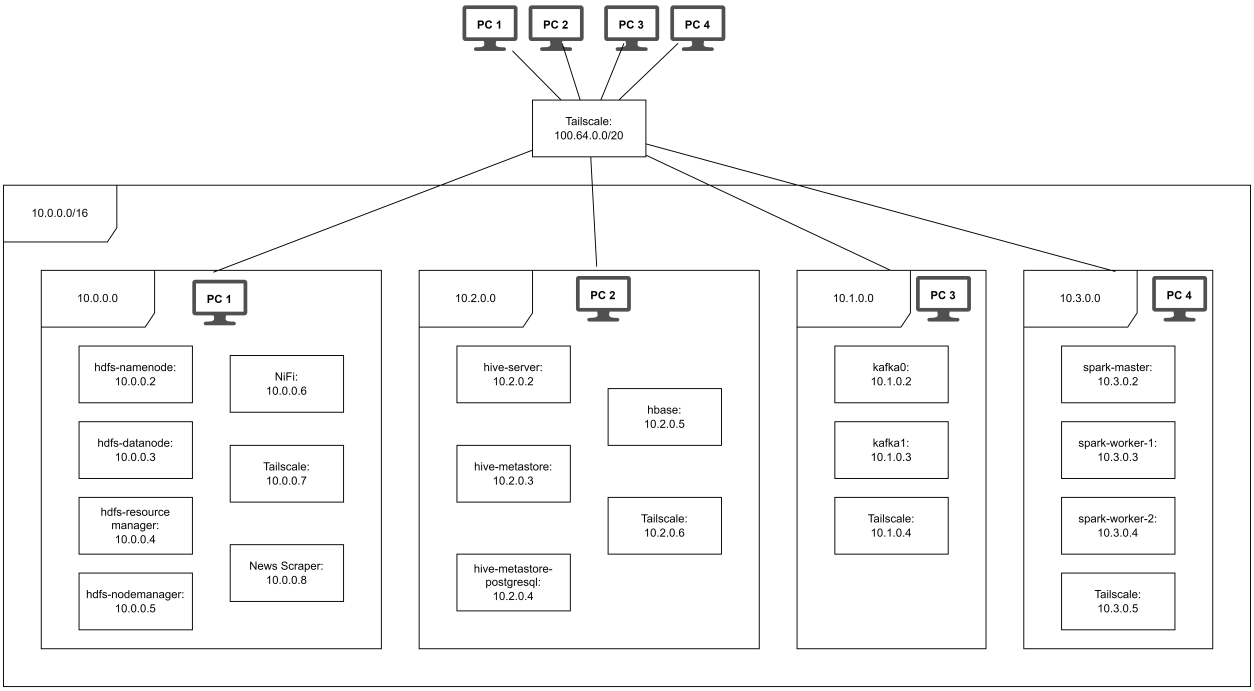


Figure 1: The initial idea of the project deployment. At first, we planned to create separate sub-networks with dockers distributed among the PCs of all our team members to provide more computational power. Eventually, after multiple hours spent on the failed attempts to properly configure such a network,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

4.1 Architecture Diagram

Figure 2 presents the final design of our solution. It differs a bit from the solution proposed in MS3 from Figure 3, as we did not use the sentiment for training the model and creating

predictions. We made that decision due to the hardware limitations, as our workstation, described in Appendix A, was unable to support the execution of both sentiment analysis and model training at the same time. It is still possible to calculate sentiment values in a real-time manner and put the results to proper `Kafka` topic by the execution of proper scripts, however, we want to underline that we finally did not use it. Moreover, the models are eventually calculated in a fully online manner, and we do not use any data from the batch layer in the process of training. Eventually, we decided to use Power BI in the presentation layer.

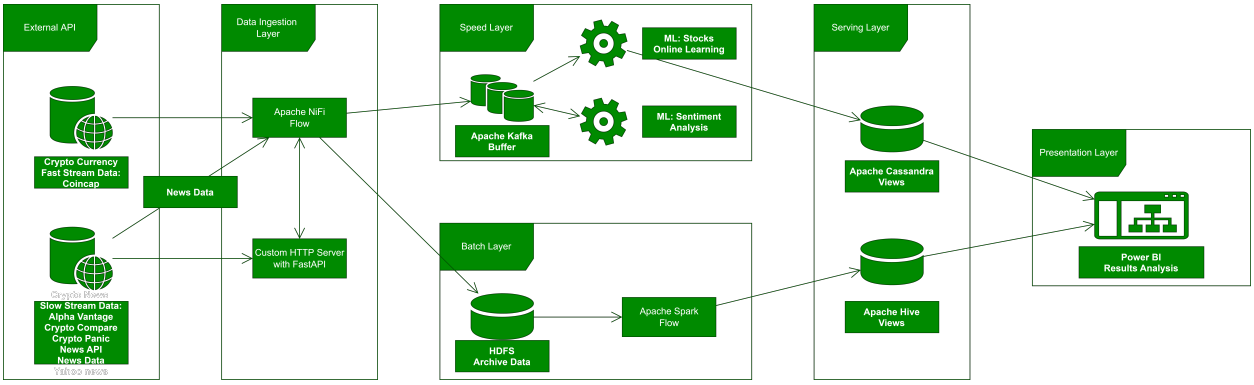


Figure 2: The final architecture of our solution. We simplified the ML system by resigning from using the sentiment for creating the models, as our workstation was unable to make those additional computations in real-time, although it is still possible to calculate the sentiment in a real-time manner, and insert it into `Kafka`. Eventually, we decided to use Power BI in the presentation layer.

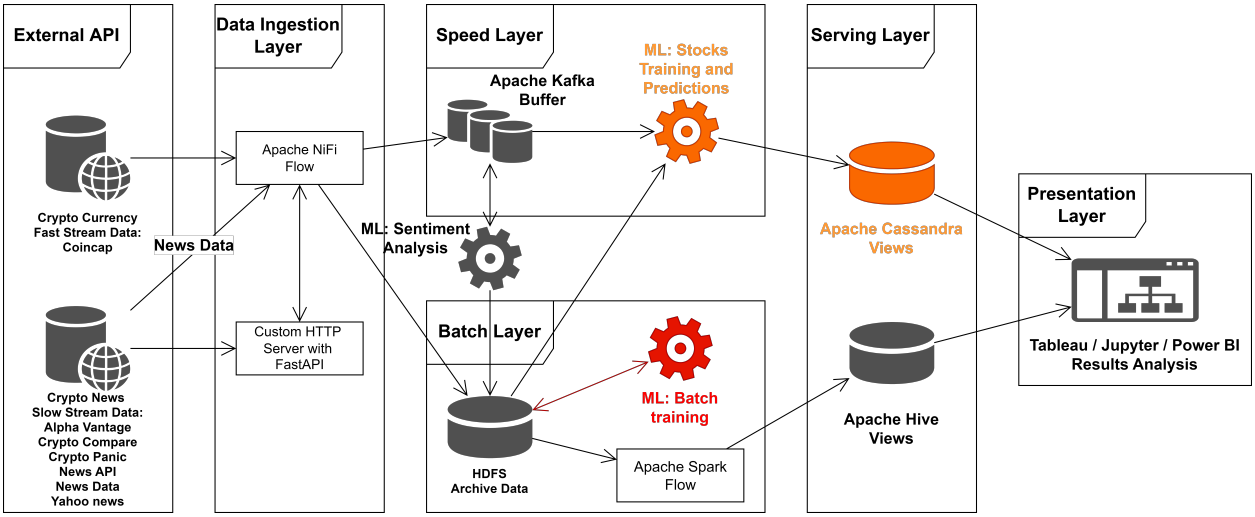


Figure 3: Revisited architecture of our solution during the MS3. We simplified the overly complex ML system, by resigning from the tuning step. Thus, we modify the ML component in the Speed Layer. Additionally, we replaced No-SQL `Apache HBase` with `Apache Cassandra`

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 4. 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`.

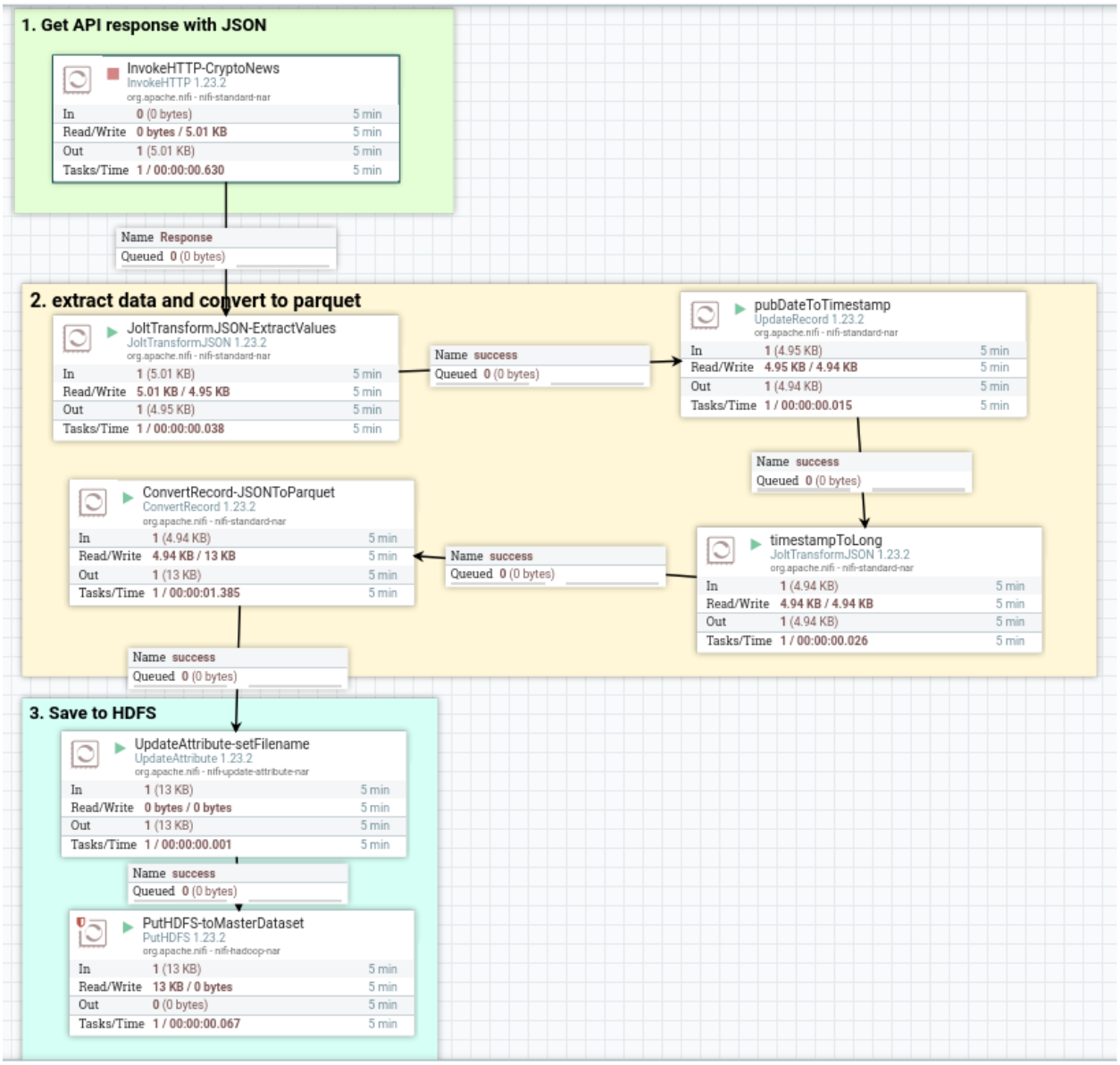


Figure 4: NiFi flow for CryptoNews.

Another flow from Figure 5 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 6.

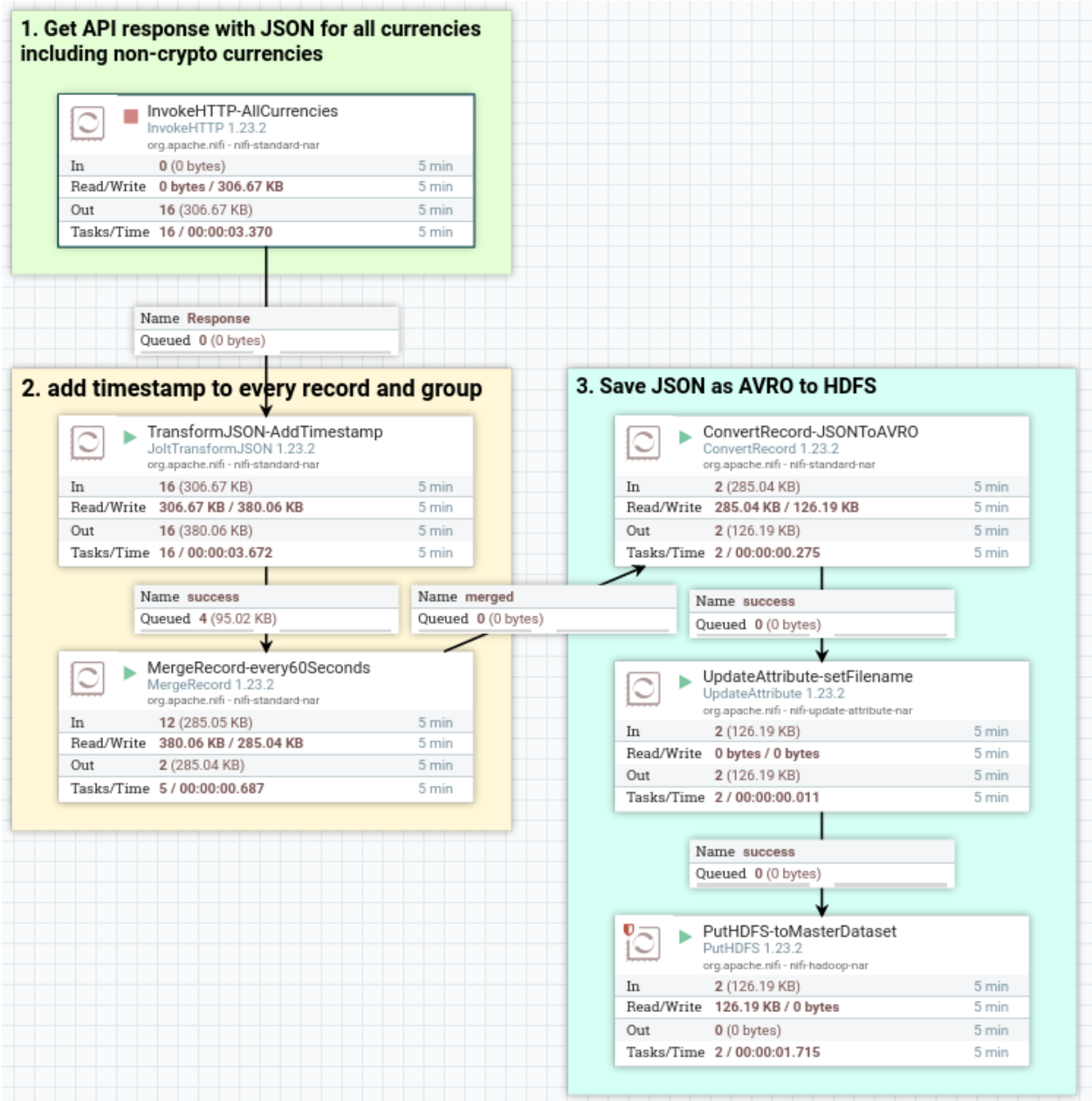
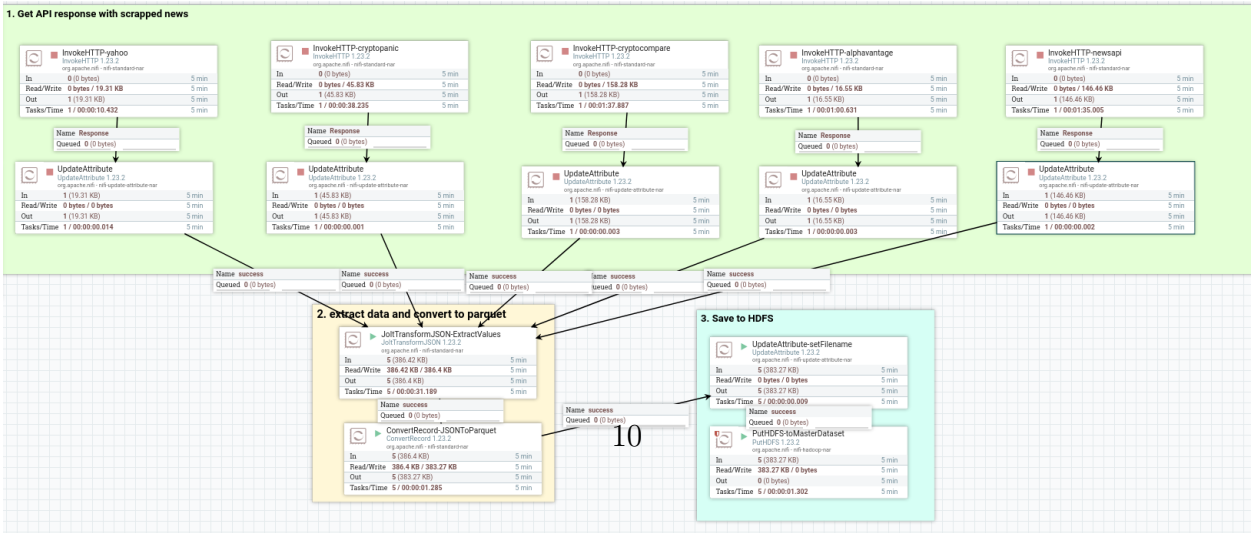


Figure 5: NiFi flow for cryptocurrencies exchange rates.



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 finally be 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 7 for Bitcoin, Figure 8 for Ethereum, and Figure 9 for DogeCoin. The lines on those plots represent the moments of publishing articles mentioning given cryptocurrency. If the colour 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.

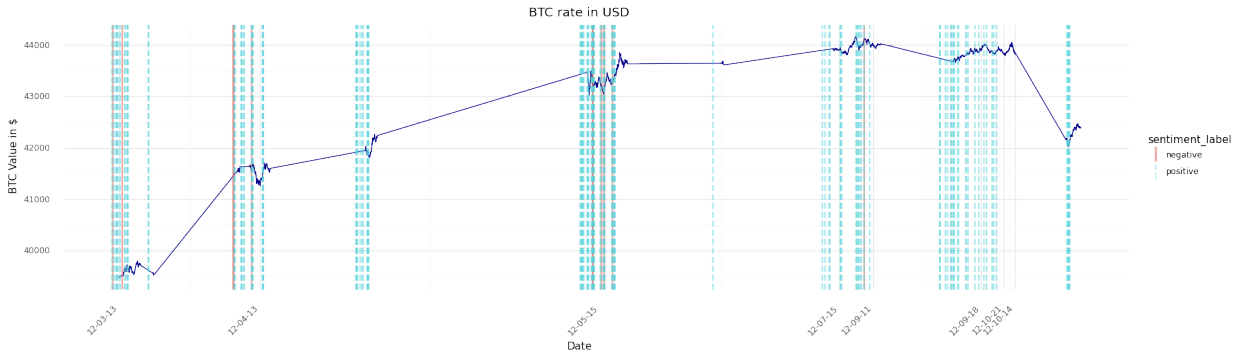


Figure 7: Representation of gathered Bitcoin data. The line plot shows the exchange rates of Bitcoin. Red vertical lines represent the appearance of the article that mentions this currency, and has negative sentiment, whereas the blue line corresponds to positive sentiment.

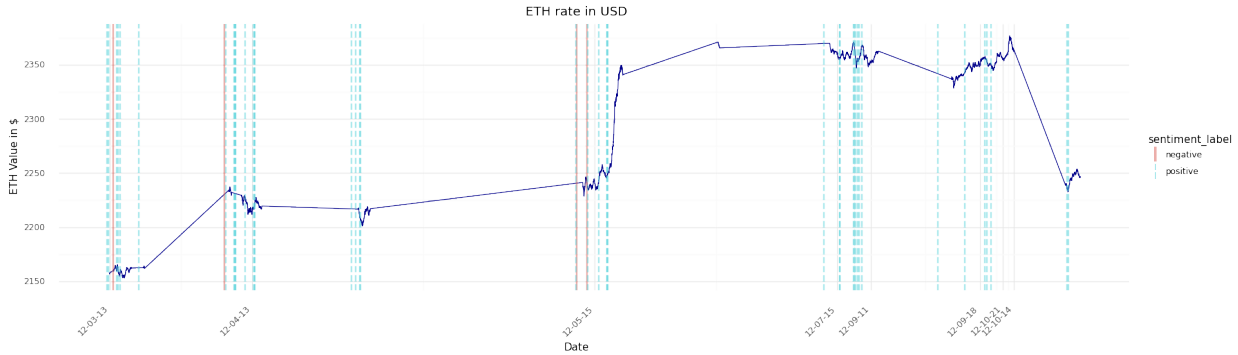


Figure 8: Representation of gathered Ethereum data. The line plot shows the exchange rates of Ethereum. Red vertical lines represent the appearance of the article that mentions this currency, and has negative sentiment, whereas the blue line corresponds to positive sentiment.

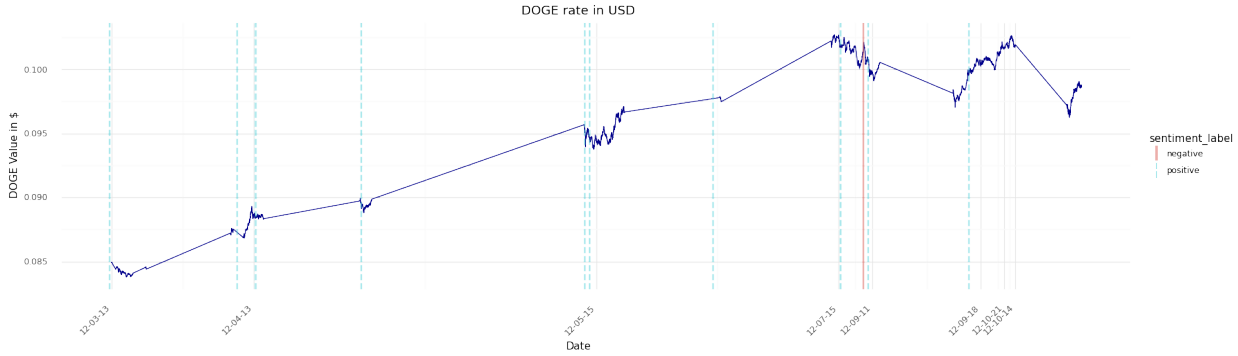


Figure 9: Representation of gathered Doge Coin data. The line plot shows the exchange rates of Doge Coin. Red vertical lines represent the appearance of the article that mentions this currency, and has negative sentiment, whereas the blue line corresponds to positive sentiment.

7 Data Processing, Filtering, and Aggregation

7.1 Streaming data

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 three cryptocurrencies, namely, Bitcoin, Ethereum, and Doge Coin. We also added timestamps, 2-minute windows, and 2-minute watermarks, both of which are necessary for stream joining. We are using those windows as our key, based on which we perform joins and watermarks to enable the system to manage late data. Sample cryptocurrency rates record can be seen in Figure 10.

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 filter out only those articles that contain information about cryptocurrencies that we are analyzing. Then after calculating sentiment for those articles, we grouped data by two 2-minute window and took the average of sentiment from the grouped articles. Aggregated news data which can be seen in Figure 11, is then saved in the `Kafka` topic.

To join two rates and sentiment data, we read sentiment data from `Kafka` topic again with a new 2 minutes window. After that, we join it to the currency rate stream using the time window as a key, which results in the stream presented in Figure 12.

```
-------------------------------------------
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 10: 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 11: Stream data with sentiment. A single record contains the cryptocurrency window it fell into, the average sentiment, and the start time of the window.

```
+-----------------------------------------+-------+------+-------------------+-----------------+----------------------+--------------------+
|window                                   |id     |symbol|changePercent24Hr  |priceUsd         |currency_timestamp    |avg_sentiment       |
+-----------------------------------------+-------+------+-------------------+-----------------+----------------------+--------------------+
|{2024-01-12 17:22:00, 2024-01-12 17:24:00}|bitcoin|BTC   |-5.6425508208902625|43498.94234103732|2024-01-12 17:22:13.549|5.673158458583333E11|
+-----------------------------------------+-------+------+-------------------+-----------------+----------------------+--------------------+

-------------------------------------------
Batch: 35
-------------------------------------------
+-----------------------------------------+-------+------+-------------------+-----------------+----------------------+--------------------+
|window                                   |id     |symbol|changePercent24Hr  |priceUsd         |currency_timestamp    |avg_sentiment       |
+-----------------------------------------+-------+------+-------------------+-----------------+----------------------+--------------------+
|{2024-01-12 17:22:00, 2024-01-12 17:24:00}|bitcoin|BTC   |-5.6455058911320470|43447.90908099249|2024-01-12 17:22:23.156|5.673158458583333E11|
+-----------------------------------------+-------+------+-------------------+-----------------+----------------------+--------------------+
```

Figure 12: Stream data after join. The stream represents the results of joining of streams from Figure 10 and 11.

7.2 Batch processing

Our solution also has modules based on batch processing. In the batch layer, we store the collection of our raw data in HDFS, which is then processed using PySpark on a daily basis. In batch processing, we do not perform major data processing as we only filter currency rates and articles so that we keep only data corresponding to cryptocurrencies we are interested in BTC, ETH, and DOGE. We additionally convert timestamp to date and ingest achieved batch views into Hive tables: one for currency rates (*cryptorates*) and one for news articles (*cryptonews*).

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 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. We map those values to the 0 and 1. Then the average is calculated to return the overall sentiment for the whole document. In the final solution, we provide a script for this feature, although we do not use sentiment values, and the reasons were described before.

8.2 Machine Learning forecasting

Firstly, we implemented a dummy predictor for the cryptocurrency rates as a baseline. It was calculated using the following equation: $\hat{y}_{t+1} = y_t * c_t$, where c_t equals the direction and value change in the last 24 hours (data extracted from API). Nonetheless, it has to be underlined that this was a dummy and a baseline model in our project and we did not stop our work at this step.

We have performed later the forecasting of the rate values using two approaches. The first one was proposed in the previous part of the project and was using the curve-fitting

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

approach. For each sliding window (100 seconds = 10 observations) 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.

The second approach proposed by us is the usage of the Adaptive Random Forest (ARF) model. It is an online learning method which processes data as they arrive (it does not iterate over the same training data many times). The model was proposed in the literature in order to adapt the well-known Random Forest (RF) model to the stream mining needs. On many sides, the ARF model resembles RF with the main exception that the base tree learning algorithm is based on the Hoeffding Tree approach. Additionally, the performance of each tree is tracked using the concept drift detector algorithm called ADWIN (Adaptive Windowing), and if the concept drift is detected, the outdated tree is replaced with a new one. Unfortunately, this approach does not deal well with missing data, as the input to the model needs to be of the same length each time. In our solution, in the case of missing observations, we added the missing number of observations at the beginning of the sliding window by drawing data from a normal distribution (whose parameters and σ are estimated using the observed data).

To sum up, we have proposed two approaches. In the first one, a model is retrained on each new arriving data. In the second approach, there is one ARF model for each observed currency, and because it is an online learning model, it is incrementally updated with each new arriving instance. The input for both models is only a number of the lagged values of the rates. We planned on including the sentiment of the news in the ARF model, however, due to the performance issues at the final stage of the project (the host computer was unable to run all needed containers, sentiment analysis pipeline and forecasting ML pipeline at once), we discarded that information.

The performance of the models was assessed by observing a rolling value of the RMSE metric (10 last errors were included in the calculation in each timestamp). In the task of forecasting, the true values always come to the system with a delay of the time between two windows. This is why our prediction module always took into account a window larger by one value, that is, of length $n + 1$. Firstly, it used observations with indexes $[1, \ldots, n]$ to predict the observation $n + 1$ (in other words, it was recreating the previous prediction). Since that observation has already been observed, the calculation of the error was possible. Next, in the case of the ARF approach, the model was updated using the true value of the $n+1$ observation. Lastly, the prediction for the future value was generated using the window of values of indexes $[2, \ldots, n + 1]$ (in the case of ARF, a new prediction was made, whereas in the case of curve-fitting, a new line was fitted and then a prediction was made).

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 the local machine, we realistically require a machine with 32GB of RAM. Additionally, a strong processor is recommended, as the designed system tends to use the total computational power of our computing unit. The mentioned PC is described in Appendix A.

Additionally, we intended to measure the quality of the forecasting models using the rolling RMSE metric (with a window size equal to 10). Nevertheless, it was possible only with a short delay after making a certain prediction, as the predicted values could not have been observed at the time of making a prediction.

The last measure is the 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-minute periods, so their presence inside the pipeline will vary by this value.

10 Results

Eventually, we implemented the aforementioned system and the results are available in the form of Power BI, an interactive dashboard. Its first page is presented in Figure 13, and it represents the results obtained by the model created for the 3rd milestone. As we can see, the initial solution was very naive, and although it was able to somehow mimic the real trend, its value differed by a huge margin from the reality.

The second page of the dashboard presented in Figure 14 presents us with the plots regarding the Batch Views. Each column of plots presents a different cryptocurrency. The first column shows us the number of articles published in time, regarding the cryptocurrency of our interest. As we can see, most of them mention Bitcoin, the second place belongs to Ethereum, and finally, the smallest amount of text is published regarding the Doge Coin. Quite interestingly, the same order comes in terms of those cryptocurrencies values, where Bitcoin oscillates around 40,000 dollars, Ethereum is around 2,000, and Doge Coin is close to 0.1. Finally, from the plots on the right we can see that the values of cryptocurrencies follow different trends globally, however, they are slightly similar in terms of reaching local peak values. It can be visible when we look at the measured values (dark dots/lines).

Finally, the last page of the dashboard presented in Figure 15 shows us the real-time data flow from 14.01.2024. The plots on the left present us with the real currency value (pink), the prediction with the ARF model (light blue), and the prediction with Logistic Regression (dark blue). The plots on the right, show us the Rolling RMSE Value for those two models. Contrary to the results from Figure 13, this time the models are much better at following the true values. From the plots for ETH, and DOGE, we can clearly see that the more complex Adaptive Random Forest model requires much more time to perform high-quality predictions, and during the first half an hour of training, it was able only to catch a general trend, and omit smaller nuances. After looking at the plots on the right we can also witness, that the ARF model performed worse than the LR, which was surprising for us. Even though, we conclude that the general results obtained by both solutions are very good.

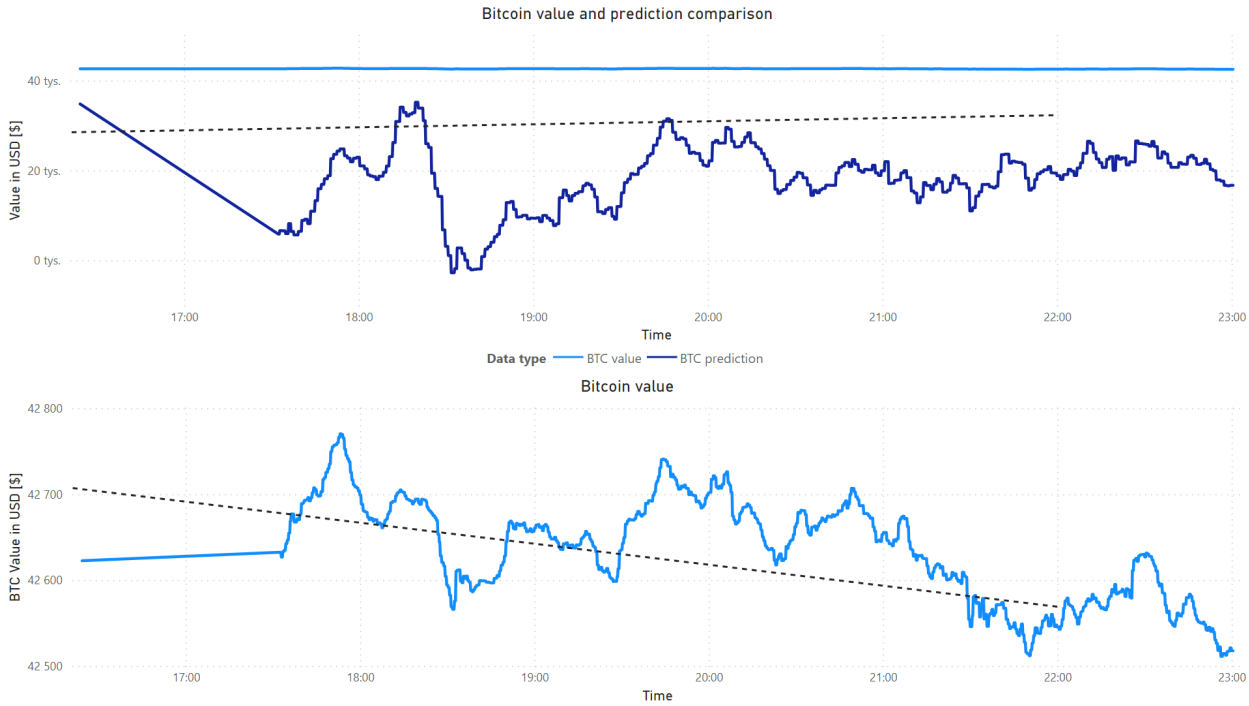


Figure 13: Visualization of Dummy predictor from MS3. The plots present the comparison of real Bitcoin values compared to the predictions of our Dummy predictor from the previous milestone. As we can see, they are terrible, and the differences between them are big.

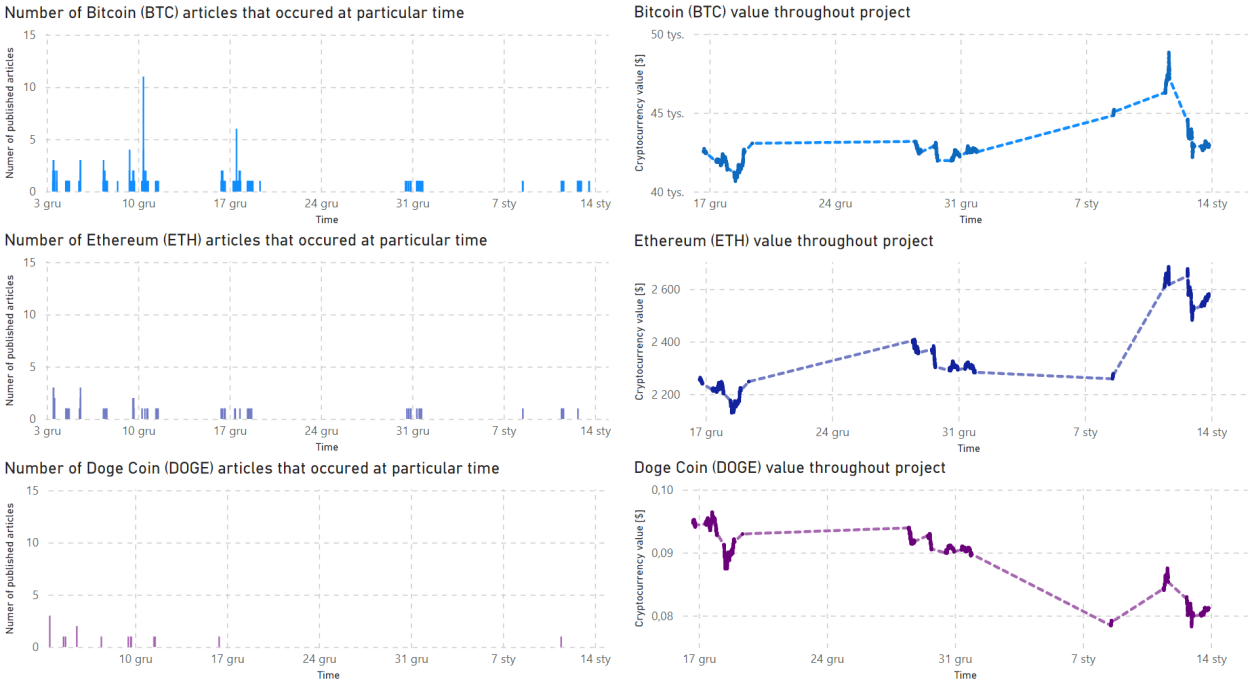


Figure 14: Visualization of Batch Views. The plots on the left present the number of articles that occurred at a particular time regarding the specified cryptocurrency, whereas the ones on the right show their values in dollars (dashed lines mean that we do not have the data). Each row and color corresponds to a different cryptocurrency, namely Bitcoin, Ethereum, and Doge Coin.

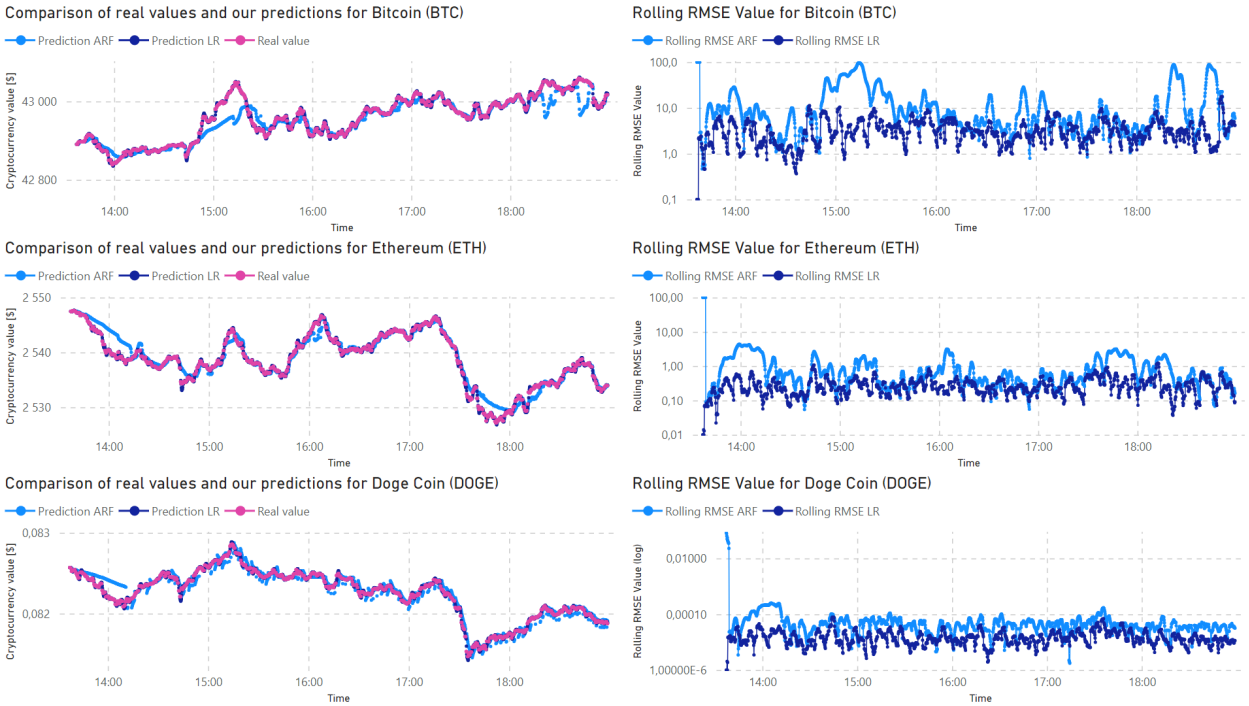


Figure 15: Visualization of Real-time Views. The plots on the left present the real value, prediction with Adaptive Random Forest (ARF), and prediction with Logistic Regression (LR) for a given cryptocurrency, shown in dollars during the last day of the project 14.01.2024. The columns on the right present the evaluation of those models with the usage of Rolling RMSE. Each row corresponds to a different cryptocurrency, namely Bitcoin, Ethereum, and Doge Coin. We can see the improvement in comparison to the Dummy predictor from the previous milestone.

11 Quality assessment from business perspective

Our solution successfully produces an analysis of the rates of three cryptocurrencies: Bitcoin, Ethereum, and Doge coin. This includes both historical data and prediction of future value. Because of the limitations of the approach where our solution runs on a single machine, we did not include the sentiment as an input to the analysis. From a business side, it is a considerable drawback as sentiment from news articles was supposed to bring new information to the analysis. However, our solution could still create an advantage for the user when investing in the cryptocurrency market by analyzing past trends in the data and reevaluating user intuition against model predictions. Furthermore, users can evaluate the accuracy of predictions of our model in the past to better assess its usability.

Unfortunately, there are also some meaningful drawbacks to the proposed solution, as we are making predictions for the next window only, which is an extremely short time horizon. Thus the solution is strongly limited in terms of drawing long-lasting conclusions and future trends. Moreover, the solution is very data-limited, as we do not use too much information, thus it is possible to create better solutions. Furthermore, the predictions made by this system are relatively slow in comparison to SOTA stock-exchange implementations. Thus we strongly recommend not using our solution for this purpose.

12 Potential Impact

Our solution can be used as an advisory tool in making investments in the cryptocurrency market, where in many situations both gains and losses can be sudden and reach very high monetary values. Prediction of the market rates is nearly impossible and people not aware of that could naively use our prediction as ground truth. Therefore, if a user is blindly basing his business decisions on the results of our analysis, it can have a huge impact on the user's economic welfare.

Hypothetically, if our solution gained huge popularity and many people would invest according to our predictions then it could lead to permanent changes in market and cryptocurrency rates on a global scale. Moreover, the solution is as good as its source data. Errors in data collected from the API cause faulty analysis, which could lead to the wrong investments.

13 Summary

Finally, we want to briefly sum up all the things achieved throughout the project:

- We implemented a fully `dockerized`, in-house Big Data system, accessible online from scratch. Unfortunately, we were not able to make it a distributed solution.
- We implemented the Data ingestion layer with the usage of `NiFi`, and wrote an online scrapper that gathers the texts with the usage of `FastAPI`.
- We implemented the Batch Layer storage with the usage of `HDFS`.
- We implemented the Batch Views in Hive, taking data from `HDFS`.
- We implemented the data buffering in a Speed Layer with `Kafka`.
- We implemented the Sentiment Analysis module in `PySpark` that puts the results in a separate `Kafka` topic. Unfortunately, we did not use it in our final solution.
- We managed to merge separate data streams (cryptocurrency, and sentiment `Kafka` topic) into one in a separate `Kafka` topic. Unfortunately, we did not use it in our final solution.
- We implemented online learning models that use the Speed Layer data to train and predict cryptocurrency rates. The results obtained by both models are satisfactory. Unfortunately, we did not manage to follow the initial ideas of (1) using the sentiment data for model training, and (2) we did not combine the batch-training, and speed-tuning concepts.
- We implemented the real-time views in `Cassandra`, which takes the results from the Speed Layer.
- We implemented the Presentation Layer in `Power BI`, as a Dashboard with 3 views: Dummy model results, Batch data, and Real-time data.

A Resources

Even though the real Big Data system should be deployed on multiple machines, it was overly complex for us to do that. The system was deployed on Hubert's PC, and used up all available resources for that sake. Its CPU is: 11th Gen Intel(R) Core(TM) i7-11700KF @ 3.60GHz, GPU: NVIDIA GeForce RTX 3070 Ti, SSD: KINGSTON SA2000M81000G, and it has 32GB of RAM.

B GitHub Repository

All source codes, documentation, scripts, configuration files, instructions, documents, videos, pictures,and so on, 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.

C 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: Final 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
Joining the streams	MS 4	BS
Batch processing	MS 4	MP
Presentation Layer	MS 4	HR
Writing a report	MS 4	HR