High-Frequency Trading Project

Completed by:

Nikita Bogomazov
Vyacheslav Blinov
Roman Vetrin

0. Abstract

The report covers the case study of implementation of high-frequency trading infrastructure utilizing machine learning and distributive data storage. During the report we will describe objectives, data characteristics, data preprocessing, model engineering as well as results evaluation of a proposed Trading platform.

During the project we were able to develop a prototype of signal-generating pipeline which covers order book snapshot preprocessing through to price trend extraction. We were able to support the quality of our pipeline by a custom backtest which was able to generate positive Profit and Loss result (hereinafter – PnL).

Terminology

Binance exchange - is a global company that operates the largest cryptocurrency exchange in terms of daily trading volume of cryptocurrencies. The company provides trading platform as well as historical and current trading data.

Limits Order Book – a collection of stock exchange orders for a particular point in time aggregated by the price levels from the mid price and categorization on bids and asks. The mid price of the order book is calculated as average value of the first level bid and first level ask.

BTCUSDT – a Bitcoin / USD trading pair

Moving mean - is a calculation to analyze data points by creating a series of averages of different selections of the full data set.

CRISP-DM – Cross Industry Standard Process for Data Mining - is an open standard process model that describes common approaches used by data mining experts. CRISP-DM breaks the process of data mining into six major phases: Business Understanding, Data Understanding, Data Preparation, Modeling Evaluation and Deployment.

PCA - Principal component analysis - technique for analyzing large datasets containing a high number of dimensions/features per observation, increasing the interpretability of data while preserving the maximum amount of information, and enabling the visualization of multidimensional data.

1. Business Understanding

1.1. Business Objective

The project was initiated aiming to create a consistent model capable to generate signal on future trend direction utilizing high-frequency limit order book data (hereinafter – LOB). Primary function of the generated signals is to assist to existing portfolio management algorithms to execute their trading strategy. Secondary objective is generation of a signal itself.

Additionally, the purpose of the project is to prepare adequate data storage suitable for placement of high-frequency LOB snapshots. The data storage have to be suitable for quick retrieval of relevant data since the prediction horizon will not exceed 1 second.

In order to ensure credibility of the generated signals, it is required to construct a custom backtest engine in order to validate the signal quality. Positive PnL on a backtest engine will be considered as necessary criteria for a positive project result.

It should be noted that the positive PnL will not grant the model credibility to sustainably generate viable signals. Such problem requires much more thorough research and sophisticated tools in use. Thus, we are limiting our goals to generate a useful feature for another trading algorithm.

1.2. Asses of Situation

The project is passed on to a team of data scientists and engineers. Additional resources were granted by the University of Innopolis and public domain.

Or team comprises of the three specialists. In order to keep principles of task independence, we distributed the available specialists among different CRISP-DM major stages:

- Project Manager – Roman Vetrin, responsible for overall project plan, formulation of business objectives and understanding, team management, presentation, model evaluation and report composition.
- Data Engineer – Vyacheslav Blinov, responsible for data mining, data preparation, feature engineering and backtest construction.

- Data Analyst – Nikita Bogomazov, responsible for the model design, model construction and model deployment.

In order to access data, we used an open source API on Binance exchange. We were able to retrieve Bitcoin / USD LOB snapshots from live updates at rate of appx. 10ms.

We created an isolated virtual machine for our project utilizing a home personal computer. There was no additional computational resources exploited.

Due to limitations in resources, we limited our training data to one trading day as it is not feasible to effectively process high-frequency data for longer period without dedicated equipment. Thus, our model might not represent general trading trend. Additionally, we utilized only a single exchange which may not represent trends of other trade platforms or other asset other than Bitcoin / USD pair.

As the trading trend tends to change rapidly, it is required to provide continuous training of the output model on actual data.

Due to list of our competencies, we were limited only to machine learning techniques ignoring modern statistical random processes methods which are in high regard in the industry.

The project is open-sourced and did not incur any costs. Potential profits from the can be totaled up to **4.33%** per month on condition of continuous training of a model.

1.3. Data Mining Goals

As was stated earlier, primary goal of the project is to generate signals useful for portfolio management i.e. which is capable to produce viable trade signals. Thus, in order to fulfill the given business goal, we will achieve the following goals:

- We will receive LOB BTCUSDT snapshots from the Binance exchange for one trading day with frequency of no more than 10ms.
- We will implement a machine learning algorithm which is capable of calculation of the moving mean direction 1 second ahead. The binary signal should represent direction of the moving mean compared to the current mid price.
- We will construct data preprocessing pipeline which will be able to extract relevant features in a timely manner.
- We will construct custom backtest engine which will utilize data not seen by the trained algorithm and will include constrains such as commission in place.

Achieving such goals will allow us to create promising model for useful signal generation for an existing portfolio management algorithm.

1.4. Project Plan

In order to achieve the goal set up above, we composed a project plan which will address key issues required to deliver the product. In order to keep in track steps which is frequently used in industrial practice, we utilized the CRISP-DM terminology while developing the project steps. The list of proposed project steps is listed below in Table 1:

#	Name	CRISP-DM stage
1	Determine project goals and target product	• Business Understanding
2	Determine trading asset for analysis and data source	• Data Understanding • Business Understanding
3	Implement Exchange – local machine connector	• Data Understanding
4	Implement local-machine – data storage infrastructure	• Data Understanding
5	Conduct Exploratory data analysis	• Data Understanding
6	Create schema for required features / label generation	• Data Preparation
7	Finalize pipeline for data preprocessing	• Data Preparation
8	Model selection and fine-tuning	• Modeling
9	Model evaluation as classification task	• Evaluation • Modeling
10	Backtest architecture development	• Business Understanding
11	Model evaluation as trade signal generation task	• Evaluation
12	Model and pipeline deployment in virtual machine	• Deployment
13	Preparation of reporting materials	• Deployment
14	Preparation of project documentation	• Deployment

In order to complete those tasks we used the following technological stack:

Exchange – local machine connector:
Python web socket;
API interface;
PostgreSQL;
Local-machine – data storage infrastructure:

Apache Sqoop;
Apache Avro;
Apache Hive;
Apache Hadoop;
Apache Tez;

Model selection and fine-tuning:

Apache Spark, Python API;
Pandas, Python;

Model evaluation as trade signal generation task;

Numpy;
Python;

Model and pipeline deployment in virtual machine:

Oracle Virtual Box;
Linux CentOS;

Preparation of reporting materials:

Streamlit;
Altair;
Microsoft office;
LibreOffice;

Such technological stack will allow us to effectively deal with large amount of daily trading datasets which may exceed several millions rows or 3 Gb per trading day. Such infrastructure addresses the risk of a model being stuck at data retrieval stage since the data storage cluster will allow us to quickly access and process the incoming data.

2. Data Understanding

2.1. Initial Data Collection

We have downloaded the BTCUSDT data from Binance exchange using open sourced API. The data was then transferred to the *.csv* file. Downloaded data covers 14th of April trading day.

The data was then imported into PostgreSQL database for further integration. Then we transferred the data into Apache Hive data warehouse system using Apache Sqoop utility and Apache Avro serialization format.

At the end of data collection and integration we established one database Apache Hive with a single table containing LOB snapshots for the trading day.

2.2. Data Description

The data contains information of the LOB state at a particular timestamp.

Field ***"timestamp"*** describes the point in time to which the snapshot relate to.

Field ***"last_update_id"*** *signifies internal id number on the last order thrown into the LOB.*

Fields ***"bid_level_[#]_price"*** reflect bid prices for [#] level from the lowest ask price.

Fields ***"ask_level_[#]_price"*** reflect ask prices for [#] level from the highest bid price.

Fields ***"bid_level_[#]_quantity"*** reflect bid quantity relevant to [#] level of bid prices.

Fields ***"ask_level_[#]_quantity"*** reflect ask quantity relevant to [#] level of ask prices.

We downloaded the LOB snapshots with **5** level depth. Thus in total our dataset contains **22** fields and over **1m** rows.

2.3. Data Exploration

In order to gain better understanding of the data, we conducted Exploratory Data Analysis. Since we are dealing with the time-series data, we utilized trends analysis in order to gain insights into data structure.

In order to have overall view on the data, we reconstructed the popular "candlestick" chart below:

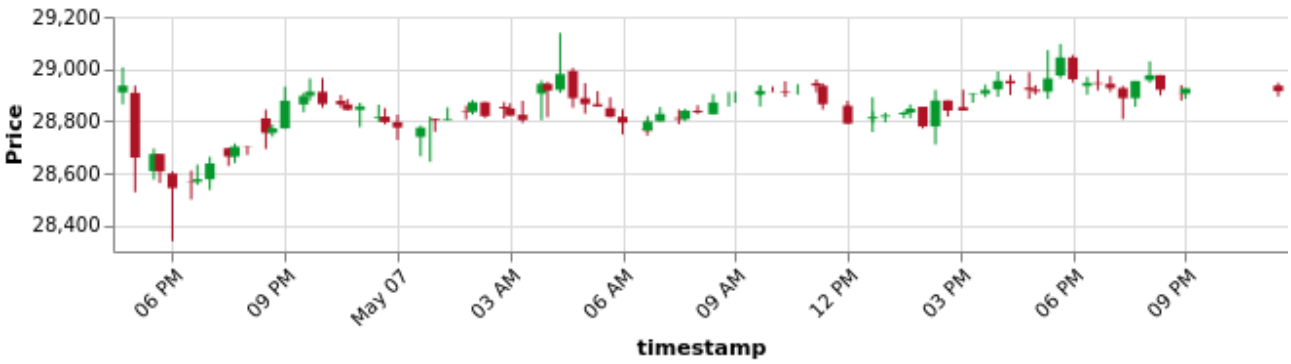


Figure 1: Candlestick chart with description of open, close, high and low prices per 20 minutes step

As we may observe, the price changes within appx. 5% margin within the day which is quite high comparing to less volatile assets. Further we will take a closer look into the order book microstructure.

First, we may look into differences between bid and ask prices on various levels. Potentially, huge difference between prices should indicate unpredictability in future mid price which we could use as a valuable feature.

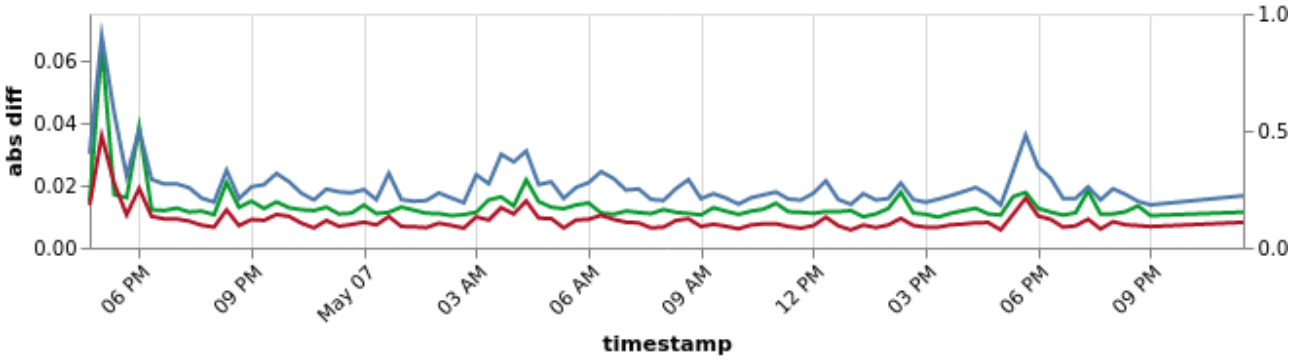


Figure 2: Difference between price levels 1(green, secondary axis), 2(red) and 3(blue)

The plot above demonstrates that few sudden changes in candlestick graph corresponds to jumps in price difference between bids and asks. Such feature can be useful in the final model.

Additionally, we can take a look into difference between amount of coin available at each level. Shortage of the asset on the first levels should signify a price change in the following periods.

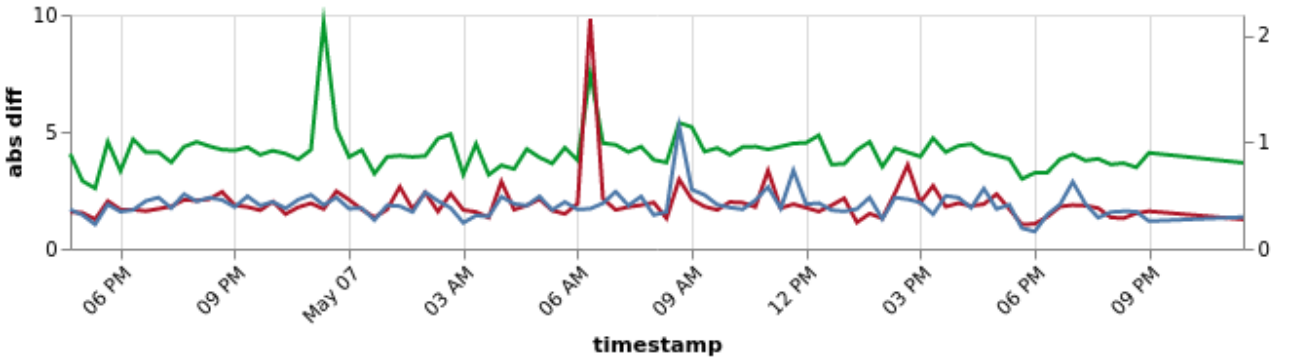


Figure 3: Difference between quantity levels 1(green, secondary axis), 2(red) and 3(blue)

As is shown on the plot, some pikes in quantity difference correspond with future highly volatile displayed on the Plot 1. Further we used such feature during the model training.

Next we will take a look into volatility estimation of a given trend. This is one of the parameters to which should define the current trend and ensure that it is feasible to divide the time-series into train and test dataset saving the overall trend properties.

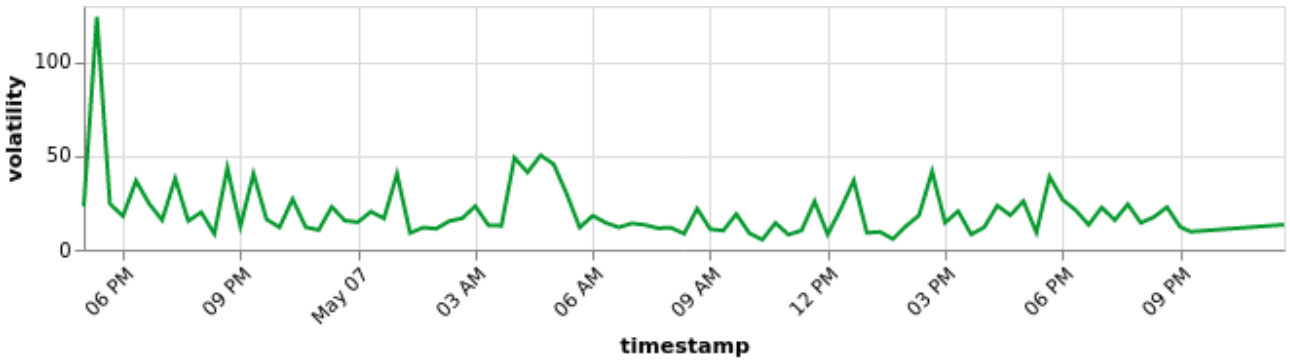


Figure 4: Volatility trends

According to the volatility values, it is feasible to divide the trend into the two parts as the levels seems overall constant through the day.

Our task will be generation of a feature, signifying the overall trend direction. In order to achieve this, we will predict the moving average value 1s ahead of the given tick and output the corresponding binary signal. The overall moving average value is displayed below:

Figure 5: Moving average trends

Prediction of the moving average allow us to output useful feature for overall trend direction.

2.4. Data Quality Verification

According to our data schema established during data collection step, there are cannot be any missing data. Every field have to be filled otherwise the row would have been rejected in PostgeSQL. Thus we can ensure that there is no missing data.

Additionally, we reviewed the trends during the EDA stage. We observed unusual trends at the beginning of the day which might not be representable in relation to the overall trend. Besides that, we did not observed any additional threats to overall data quality.

3. Data Preparation

3.1. Data Selection

In order to make our analysis complete, we believe that every field is relevant for our analysis with an exception of "***last_update_id"*** field which is not relevant to our objective.

3.2. Data Clean

According to our Hive database schema, there cannot be any data except the timestamp type and the Float type. Thus, there was no need in additional steps on cleaning up the data.

3.3. Data Construction

According to our EDA, we revealed some potential useful features, which may be distinctive in the current trade trend. Thus we derived a number of features for further analysis.

mid_price_value

Since we did not have any actual labels, we need to construct our own using the LOB data. The raw LOB snapshot do not contain mid price information since it is essentially calculated during the transaction itself. However, we may approximate it closely enough using average value of the first price levels if bids and asks (fields ***bid_level_1_price*** and ***ask_level_1_price***).

label

As was stated earlier, our goal will be to predict direction of moving mean value ahead for 1 second. In order to create the appropriate label for training, we calculated the moving mean using average value of ***mid_price*** for 80 ticks ahead of the current timestamp. In case such value exceeds current mid_price, we marked it as **0** – upward trend. Otherwise we marked it as **1** – downward trend. Note that we were unable to mark the last 80 ticks since the future data is not available to us. Thus we excluded the last 80 observations from out training data.

level_[#]_diff

In order to access information on LOB imbalance, we calculated price difference between asks and bids between each of the **5** levels. This way, we generated **5** additional fields.

ask_level_[#-1]_price – ask_level_[#]_price

In order to understand distribution of ask prices, we calculated difference of subsequent pairs of asks levels. For instance, we calculated price difference between

Level 1 and Level 2, than between Level 2 and Level 3, and so forth. This way we calculated additional **4** fields.

bid_level_[#]_price – bid_level_[#-1]_price

In order to understand distribution of bid prices, we calculated difference of subsequent pairs of bids levels. For instance, we calculated price difference between Level 2 and Level 1, than between Level 3 and Level 2, and so forth. This way we calculated additional **4** fields.

ask_level_[#]_price – ask_level_1_price

In order to understand distribution of ask prices, we calculated difference of each level of asks compared to the first level. This way we calculated additional **4** fields.

bid_level_1_price – bid_level_{}_price

In order to understand distribution of bid prices, we calculated difference of each level of bids compared to the first level. This way we calculated additional **4** fields.

avg_asks_price

In order to gain insight into asks prices distribution, we calculated the average price among all asks levels.

avg_asks_quantity

In order to gain insight into asks quantity distribution, we calculated the average quantity among all asks levels.

avg_bids_price

In order to gain insight into bids prices distribution, we calculated the average price among all bids levels.

avg_bids_quantity

In order to gain insight into bids quantity distribution, we calculated the average quantity among all bids levels.

level_diff_price_sum

In order to assess LOB imbalance, we calculated sum of price differences between asks and bids for each level of LOB.

level_diff_quantity_sum

In order to assess LOB imbalance, we calculated sum of quantity differences between asks and bids for each level of LOB.

In order to evaluate existing and generated features we trained a Gradient Boosting Classifier Tree and extracted information on feature importance. The result is presented below:

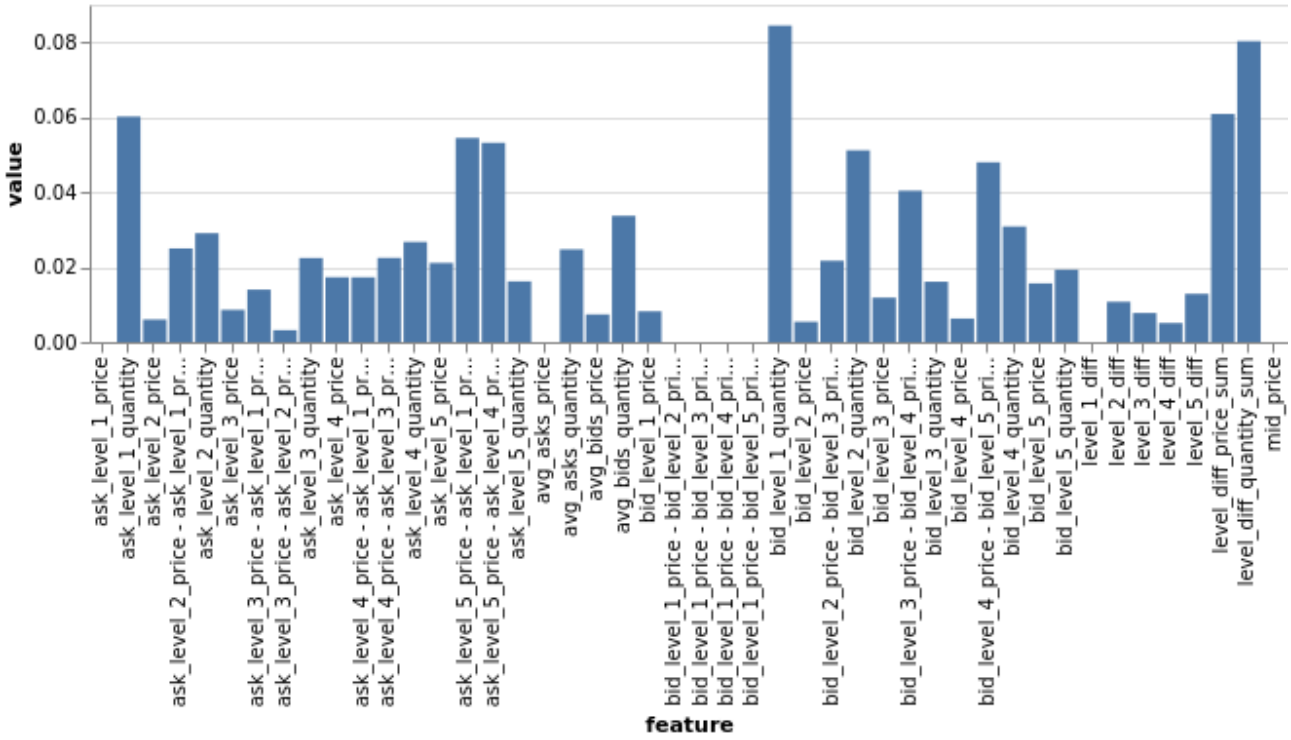
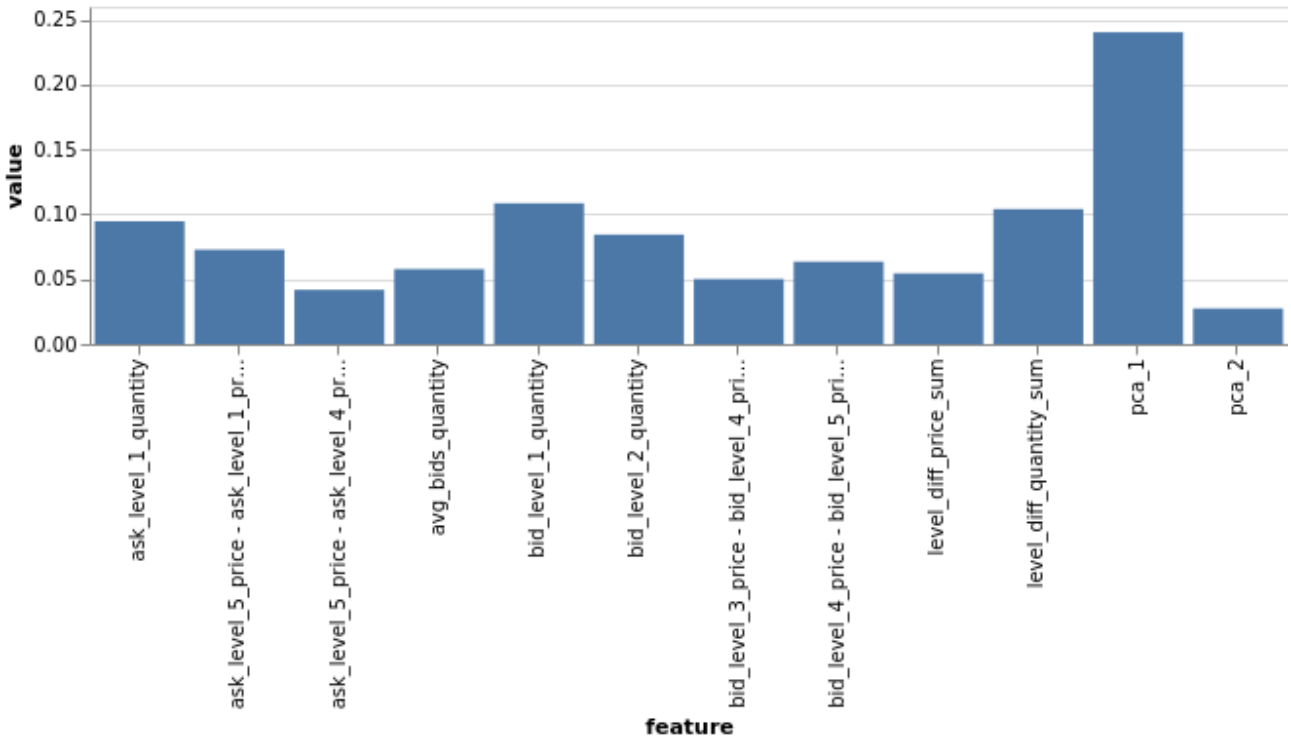


Figure 6: Features importance bar chart

We may observe that a number of proposed features indeed evaluated as quite important. However, a lot of features seems not to be important for our goals. In order to reduce dimensionality, we will extract key vectors from those features via PCA algorithm. We selected 10 top features for direct usage and PCAed the rest of features transforming them into two dimensions. Then we reevaluated importance of the new features and ended up with the following importance distribution:



Thus, we kept the mos important features and combined the rest in the PCA vectors. As we may observe from the plot, the first PCA vector should represent an important feature.

3.4. Data Integration

Since we had a single data table for our project, there was no need in additional steps in integrating our data structure. We may note that all of our data was stored in the Apache Hive database. Thus, in order to effectively access the data in a timely manner, we utilized Apache Spark technology during transformation and modeling construction phases.

3.5. Data Formatting

In order to finalize the data before the modeling stage, we separated it into train and test dataset. Since we are dealing with the time-series, it is not viable to shuffle the data before the split since each data point contains information about the subsequent entries. So shuffling the data will lead to effectively mixing the test and train data together without a way to properly divide them.

In order to avoid this, we separated the data by a certain timestamp which divides the data by 80% and 20% without breaking the order of data.

4. Modeling

4.1. Modeling Technique Selection

We considered the three classification models for our task:

- Gradient Boosting Classifier Tree
- Supported Vector Machine
- Multilayer Perceptron Classification model

We assembled each model, derived the optimal parameters and evaluated final performance in the following stage.

4.2. Generation of Test Design

Since we transformed our task into the classification problem, we designed the test procedures considering typical classification problem issues. First of all, we evaluated label balance in order to reveal the threat of imbalanced features. As demonstrated on the plot below, the labels are balanced thus there is limited threat of our metrics' validity.

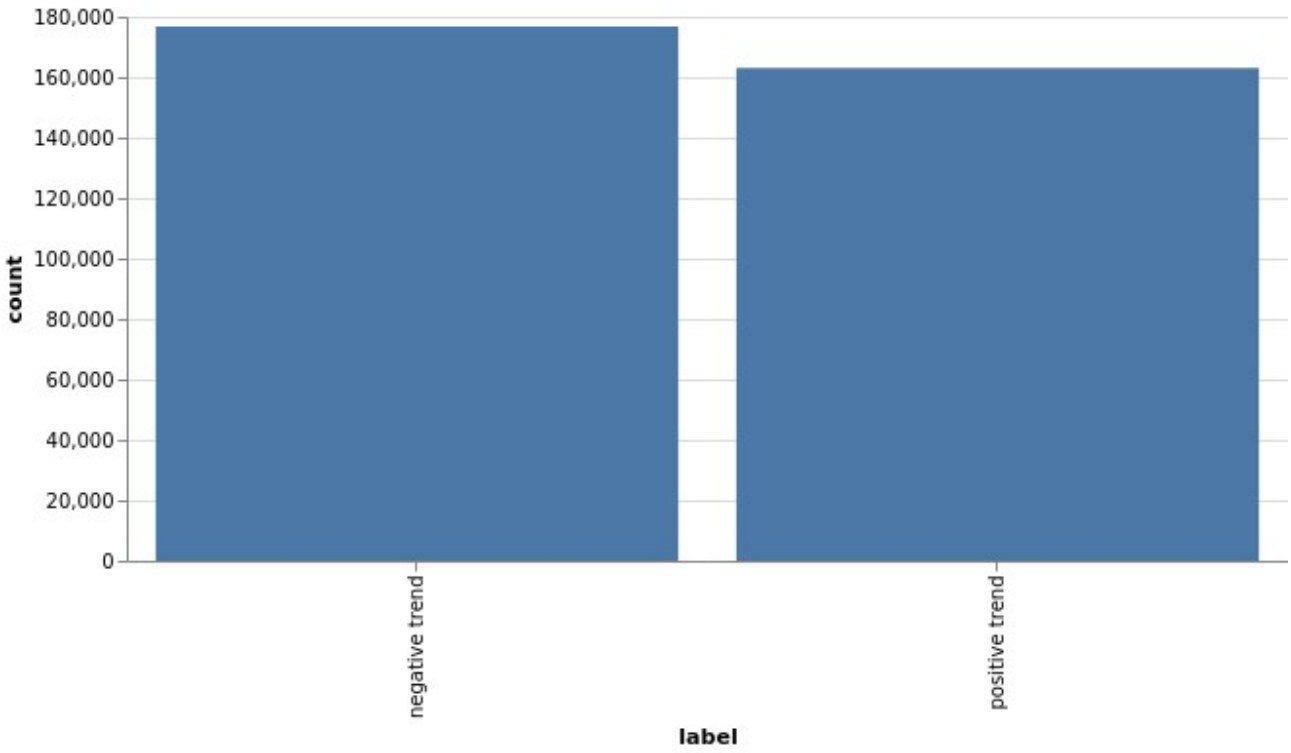


Figure 7: Label count graph

Next, we were considering the standard classification task metrics like Accuracy, Precision and Recall. In order to implement a weighted score which would consider every fo the mentioned metrics, we will measure our model by the Precision/Recall curve area. Such metric ranges from 0 to 1 and considers all of the metrics’ values.

4.3. Model Building

In order to keep speed efficiency of our model, we used integrated Apache Spark tools to build the models. Thus we saved the efficiency of data access from the Apache Hive storage.

We used the following parameters for our models:

Gradient Boosting Classifier Tree

Parameter	Value
maxDepth	5
maxMemoryInMB	512
maxBins	64
stepSize	0.05

Supported Vector Machine

Parameter	Value
maxIter	30
aggregationDepth	4

Multilayer Perceptron Classification model

Parameter	Value
maxIter	40
aggregationDepth	[12, 512,128,2]

4.4. Model Assessment

After fin-tuning the models we compared the Precision/Recall curve area among each of them. The **MPC** model ended up being the most promising achieving 64% of area square and apprx. 67% of accuracy on train data.

5. Evaluation

5.1. Evaluation Results

We utilized a two step evaluation process in order to validate the model overall quality.

First, evaluated our model on the test data and plotted the confusion matrix. The matrix displays the correct predictions on the diagonal (1-1 and 0-0 match) and errors on the other squares (0-1 and 1-0) match. As we may observe from the confusion matrix below, we achieved stable result without imbalance in recall or precision metrics.

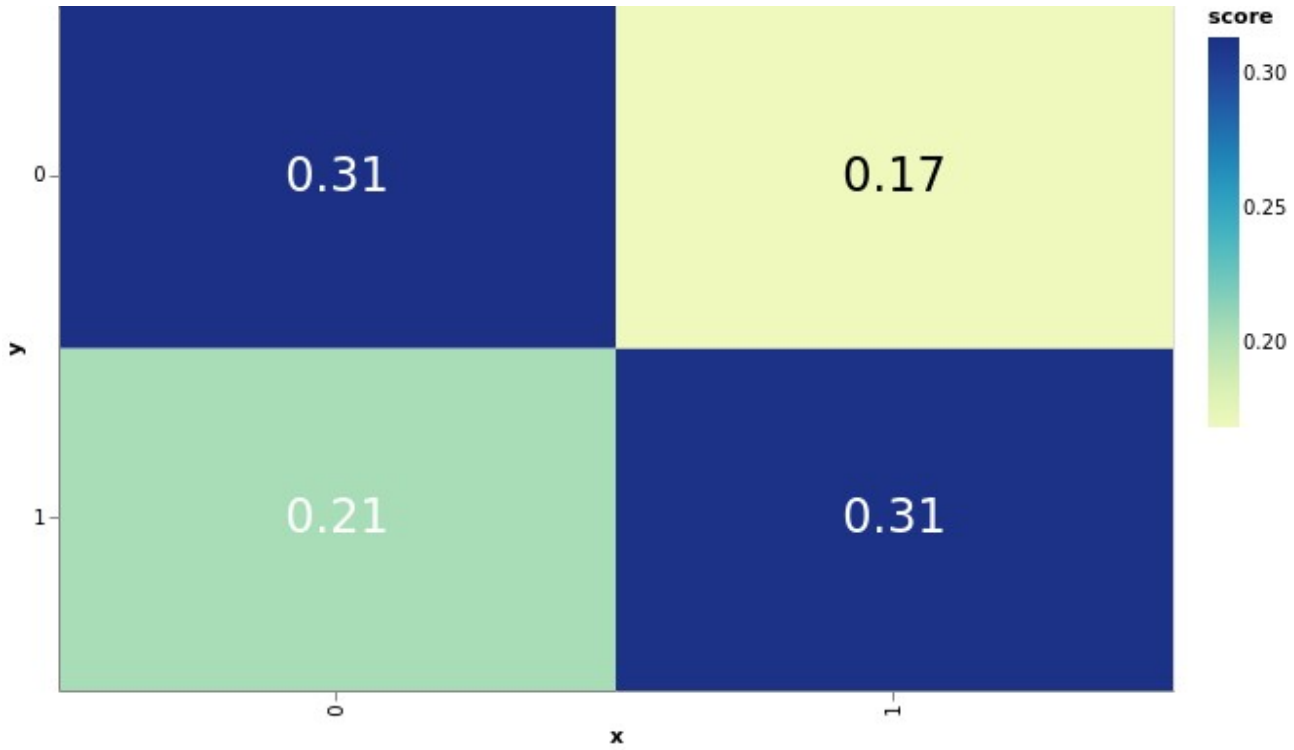


Figure 8: Confusion matrix

Then, we constructed the backtest engine and evaluated trading signals using the test time period. In our simulation we made the following assumptions:

- The commission for each transaction is 0.015% (which is available on Binance exchange)
- Our model is able to instantly generate the signal and the infrastructure is able instantly post the order
- The total price of the posted order is matching with the calculated mid price
- There is USD 1m starting capital and the price of capital is 0%

Overall we achieved apprx. 0.14% of daily return rate which corresponds to 51% of annual income. Of course this limited test is not enough to prove the model itself is viable, however, it does indicate viability of the generated trend indicator.

Sd

Thus we may conclude that our signal have respectable strength and can be used for the trading purposes.

5.2. Review of Process

The process of data mining is fairly straight forward and is not prune for much threats of validity. However, it should be noted that it is important to note the latency of the accepted data. Since the project is related to high-frequency trading, the latency which can be afforded is limited to sub-second values.

The model itself is fairly robust, however it is possible that it learned strictly the local trend which may be changing over the next weeks or months. It is important to further train and evaluate the model on additional data.

5.3. Determination of Next Steps

Currently the system is separated from the data scrapping stage – it does not have direct access to the API and reads the data from .csv file. It is vital to create the infrastructure to directly mine the trading data into the Apache Hive data storage directly to minimize latency.

Next issue is that the currently lives in the virtual machine. It is required to create an machine-to-machine interface with the VM in order to automate cooperation with the built infrastructure.

6. Deployment

6.1. Deployment Plan

It is recommended to fund a dedicated server which can be placed in the location of Binance exchange and where the virtual machine will be placed. In this way the latency loss will be minimized and such system will produce more accurate data.

Utilizing the machine-to-machine interface with the environment, it is possible to receive access to live predictions from the Binance exchange.

6.2. Monitoring and Maintenance Plan

The model have to be constantly reviewed using the metrics described above. It is expected for the model to become outdated fast given that the cryptocurrency trends tends to change very quickly over time. Thus, an engineer should be assigned to the task of monitoring performance and retraining of the model using the previously precessed data.

6.3. Production of Final Report

The final report was assembled upon the project's end. The final project documentation along with source code and relevant EDA dashboards is placed in the github repository: https://github.com/RVdeported/bigdata-final-project-rv-vb-nb.git