ALLDATA 2022

The Eighth International Conference on Big Data, Small Data, Linked Data and Open Data

ISBN: 978-1-61208-945-4

April 24 - 28, 2022

Barcelona, Spain

ALLDATA 2022 Editors

Anna Wałek, Gdańsk University of Technology, Poland

ALLDATA 2022

Forward

The Eighth International Conference on Big Data, Small Data, Linked Data and Open Data (ALLDATA 2022) continued a series of events bridging the concepts and the communities devoted to each of data categories for a better understanding of data semantics and their use, by taking advantage from the development of Semantic Web, Deep Web, Internet, non-SQL and SQL structures, progresses in data processing, and the new tendency for acceptance of open environments.

The volume and the complexity of available information overwhelm human and computing resources. Several approaches, technologies and tools are dealing with different types of data when searching, mining, learning and managing existing and increasingly growing information. From understanding Small data, the academia and industry recently embraced Big data, Linked data, and Open data. Each of these concepts carries specific foundations, algorithms and techniques, and is suitable and successful for different kinds of applications. While approaching each concept from a silo point of view allows a better understanding (and potential optimization), no application or service can be developed without considering all data types mentioned above.

We take here the opportunity to warmly thank all the members of the ALLDATA 2022 technical program committee, as well as all the reviewers. The creation of such a high-quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to ALLDATA 2022. We truly believe that, thanks to all these efforts, the final conference program consisted of top-quality contributions. We also thank the members of the ALLDATA 2022 organizing committee for their help in handling the logistics of this event.

ALLDATA 2022 Chairs

ALLDATA 2022 Steering Committee
Yoshihisa Udagawa, Tokyo University of Information Sciences, Japan
Mamadou H. Diallo, Naval Information Warfare Center (NIWC) Pacific, U.S. Department of Defense, San Diego, CA, USA
Fernando Perales, JOT INTERNET MEDIA, Madrid, Spain
Dong Quan Ngoc Nguyen, University of Notre Dame, USA

ALLDATA 2022 Publicity Chairs
Javier Rocher, Universitat Politècnica de València, Spain
Lorena Parra, Universitat Politècnica de València, Spain

ALLDATA 2022
Committee

Table of Contents

Developing a Stock Trading Simulator Using Candlestick Chart Patterns for Estimating Profitability of Global Markets

Yoshihisa Udagawa
Faculty of Informatics, Tokyo University of Information Sciences
Chiba-city, Chiba, Japan
e-mail: yu207233@rsch.tuis.ac.jp

***Abstract*— This paper deals with the development of a stock trading simulator. It uses candlestick chart patterns and the 5-day moving average for deciding trade opportunities, and tries to make profits on both trades that start with buying stocks and those that start with selling stocks, i.e., long and short trades. Experiments are performed using the seven world markets consisting of Euro Stoxx 50 (the European Union), DJI and NASDAQ (the United Sates), Bovespa (Brazil), Sensex (India), SSEC (China), and Nikkei 225 (Japan). Profits are simulated using ten-year daily stock data of the seven global stock markets with and without a trading fee. The results of experiments are generally consistent in the seven markets, showing the wide range of applicability of the simulator. The average success rates for trades that are started with buying stocks with a trading fee remains in the middle of 60%, while it is at the beginning of 20% for the trades that start with selling stocks because margin interest to loan stocks squeezes profits. Although the developed simulator has some limitations, it provides a benchmark for measuring profitability of trading techniques and helps to analyze stock trade timings that lead to significant profits or losses.**

Keywords*— *Stock trading simulator; Global market comparison; Technical analysis; Candlestick chart; Moving average.

I. Introduction

The stock market has fascinated people for extra income. Many make profits, others losses thorough trading on a stock market. Stock price forecasts play an important role to mitigate the risks associated with investment.

Generally, there are two main types of methods that allow investors to make an investment decision, namely fundamental analysis and technical analysis [1].

Fundamental analysis is an investing strategy that attempts to determine a security's intrinsic value by focusing on microeconomic indicators and macroeconomic factors. Examples of the former are a company's historical earnings and profit margins as well as future growth prospects. Examples of the latter include the state of industry conditions and market sentiments. The primary assumption of the fundamental analysis is that the stock price of a company reflects the intrinsic value in the long run. Because of this nature, the fundamental analysis is considered to be effective for a long-term trade.

In contrast, technical analysis is an investing strategy that tries to identify trading signals by analyzing the statistics of preceding stock price data and volume. Typically, the technical analysis uses charts and indicators to forecast price trends and patterns in the near future.

This study focuses on candlestick patterns to spot the timing of a trade [2][3]. Because candlestick patterns are typically formed by one to three candlesticks, they are believed to be suitable for a short-term prediction. While the candlestick charting technique was probably established sometime after 1850, the candlestick technique became popular worldwide in the 1990s. Despite its long history and popularity, mixed results are obtained in the studies on the technique. Negative conclusions to the predictability of candlestick charts are reported [4]-[6], while positive evidences are provided for several candlestick patterns by experiments using the U.S. and Asian stock markets' historical data [7]-[11]. Most of the papers on stock price prediction focus on discussions about the accuracy of predictions concerning stock price increases using specific chart patterns and/or indicators. Since stock trading fundamentally consists of buying and selling stocks, the criteria for deciding when to buy and sell stocks, and the resulting profits must be estimated before they can be applied to actual trading.

The purpose of this study is to develop a simulator for stock trading and to compare the performances of global markets in terms of profits. The developed simulator uses candlestick patterns and the 5-day moving average to find the opportunities to buy and sell stocks. It is designed to make profits on both trades that start with buying stocks, i.e., trades in a long position, and those that start with selling stocks, i.e., trades in a short position. Experiments are performed using the seven international markets consisting of Euro Stoxx 50 (the European Union), Dow Jones Industrial Average (DJI) and NASDAQ (the United Sates), Bovespa (Brazil), Sensex (India), SSEC (China), and Nikkei 225 (Japan) [12].

The contributions of this study are as follows:

I. Developing a stock trade simulator that continuously tries to make profits by buying and selling a stock.
II. Profits are simulated over the seven global stock markets for long and short trades, with and without trade fees.
III. Detailed analysis on big trades for demonstrating usefulness and limitation of the developed simulator.

The remainder of the paper is organized as follows. Section II gives the background of the candlestick chart, and overviews the algorithm of the developed simulator. Section

III shows experimental results including success rates and cumulative profits in seven markets. Section IV concludes the paper with our plans for future work.

II. STRUCTURE OF STOCK SIMULATOR

This section describes formations of a candlestick chart, criteria for finding trade opportunities, and outlines the developed stock trading simulator.

A. *Formation of Candlestick*

As depicted in Figure 1, a daily candlestick is formed with the market's opening, high, low, and closing prices of a market day [2][3]. The candlestick has a wide part, which is called *real body* representing the range between the opening and closing prices of that day's trading. The color of the *real body* represents whether the opening price or the closing price is higher. If the price rises, a hollow body is drawn indicating *bullish* or buying pressure. Otherwise, a filled body is drawn showing *bearish* or selling pressure.

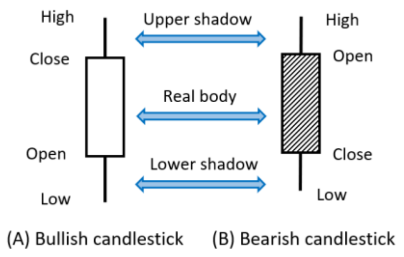


Figure 1. Candlestick formation.

The thin lines above and below the body, which are named *shadows*, represent the range of prices traded in a day. The high is marked by the top of the upper shadow and the low by the bottom of the lower shadow.

B. *Criteria for Spotting Trade Timings*

A candlestick chart is a graph in which candlesticks are presented in chronological order. It is used as a tool to get information on whether the current price is higher or lower than the historical stock price movements. Moving averages form a line graph by connecting the average of closing prices over a certain period of time. As for a period of time to compute an average, each country uses its own. For example, the short-term average is often calculated for 5 days, the medium-term average is for 25 days in Japan. The moving average graph is useful to decide whether a stock price is at the beginning stage of a rising trend or a falling one.

Candlestick patterns are usually described in a context of stock price trends. In this study, we focus on the difference between the closing price and the 5-day average. Figure 2 depicts a bullish reversal pattern that is commonly observed at the start of a rising trend. The pattern has two parameters, i.e., δ for the length of the candlestick body and ε for the difference between the closing price and the 5-day moving average. The optimum values for δ and ε vary for each stock market, which is discussed in the next subsection.

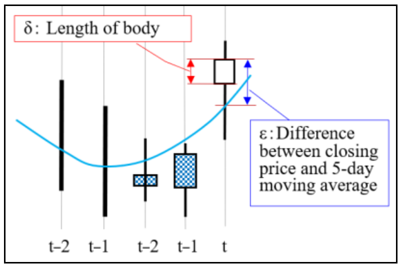

δ : Length of body
ε : Difference between closing price and 5-day moving average
t–2
t–1
t–2
t–1
t

Figure 2. Bullish reversal pattern.

Figure 3 illustrates a bearish reversal pattern that is generally observed at the beginning of a falling trend. The pattern also has two parameters, i.e., δ and ε. In this study, the perfect symmetry of the bullish and bearish reversal patterns is presumed.

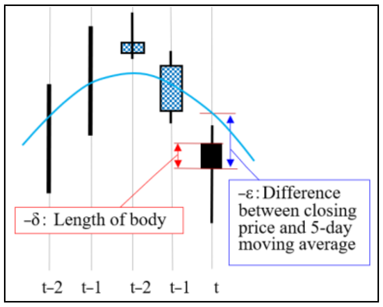
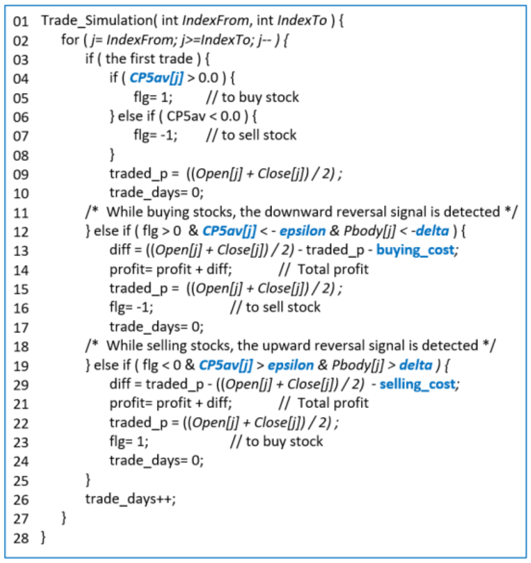

–δ : Length of body
–ε : Difference between closing price and 5-day moving average
t–2
t–1
t–2
t–1
t

Figure 3. Bearish reversal pattern.

Figure 4 shows the pseudocode that outlines the algorithms of the developed simulator. Stock data are stored in arrays, such as *Open[]* for the opening prices, *Close[]* for the closing prices, *Pbody[]* for the lengths of candlesticks, and *CP5av[]* for the difference between the closing price and the 5-day moving average. Stock data are stored in chronological order, i.e., the larger index values of an array represent the further past trading dates.



```
Trade_Simulation( int IndexFrom, int IndexTo ) {
    for ( j= IndexFrom; j>=IndexTo; j-- ) {
        if ( the first trade ) {
            if ( CP5av[j] > 0.0 ) {
                flg= 1;        // to buy stock
            } else if ( CP5av < 0.0 ) {
                flg= -1;       // to sell stock
            }
            traded_p =  ((Open[j] + Close[j]) / 2) ;
            trade_days= 0;
        /*  While buying stocks, the downward reversal signal is detected */
        } else if ( flg > 0  & CP5av[j] < - epsilon & Pbody[j] < -delta ) {
            diff = ((Open[j] + Close[j]) / 2) - traded_p - buying_cost;
            profit= profit + diff;          //  Total profit
            traded_p =  ((Open[j] + Close[j]) / 2) ;
            flg= -1;                  // to sell stock
            trade_days= 0;
        /*  While selling stocks, the upward reversal signal is detected */
        } else if ( flg < 0 & CP5av[j] > epsilon & Pbody[j] > delta ) {
            diff = traded_p - ((Open[j] + Close[j]) / 2)  - selling_cost;
            profit= profit + diff;          //  Total profit
            traded_p = ((Open[j] + Close[j]) / 2) ;
            flg= 1;                   // to buy stock
            trade_days= 0;
        }
        trade_days++;
    }
}
```

Figure 4. Outline of developed simulator algorithms.

The method *Trade_simulation* takes two integer parameters, i.e., *IndexFrom*, and *IndexTo*. These variables specify the start and end of a simulation, respectively.

The position of a trade is indicated by the value of a variable named *flg*, i.e., 1 for a long position, and –1 for a short position. At the start of a trading simulation, the position is determined by the value of *CP5av[j]*. If *CP5av[j]>0* then the variable *flg* is assigned to 1, otherwise to –1.

If the bearish or downward reversal pattern shown in Figure 3 is detected during the period of buying a stock, the simulator sells the stock and calculates the profit *diff* of the trade and the cumulative profit *profit*, updates the trade price *trade_p*, and assigns the variable *flg* to –1 to enter a short position, etc. The profit *diff* is calculated as the profit of one stock at the market price. The trade price *trade_p* is calculated by adding the opening and closing prices, and dividing by 2.

If the bullish or upward reversal pattern shown in Figure 2 is found during the period of selling a stock, the simulator buys a stock back and assigns the variables in the symmetrically same way as the detection of the bearish reversal pattern.

C. Calculating trading fee

Purchasing stocks comes with a trading fee. The trading fee varies across brokers and depends on the amount of the trade. Table I summarizes trading fees of some Japanese online brokers. This study assumes the trading fee is 0.1% of the amount of a traded stock price due to the fact that the fee is reasonable for the trade amount between 1M JPY (≒ 8,700 USD) to 3M JPY (≒26,000 USD).

TABLE I. TYPICAL TRADING FEE OF JAPANESE ONLINE BROKERS

	～ ¥1M	¥1M ～ ¥3M	¥3M ～ ¥5M
SBI	0	0	¥2,576
Rakuten	0	¥3,300	¥5,500
Manex	¥550	¥2,750	¥5,500

In a short sale, a trader must borrow the stock to sell from an investment firm with a payment of margin interest. The leverage of trades is assumed to be one, because the symmetry of long and short trades is presumed in this study. Margin interest slightly varies across brokers. Typically, it consists of an annual interest and a fixed one. In this study, the margin interest is calculated using the following formula:

$$(\text{Stock price}) * (2.80\% / 365 * N + 1.15\%) \qquad (1)$$

N is the total number of days for borrowing stocks. 1.15% indicates the fixed interest, which greatly affects short sale profits, as described in Section III.

D. Determining Optimal δ and ε

All market data used in this study are downloaded from the *Investing.com*'s Web site [12]. Price data are converted into percentage ratios. For example, let *Close[j]* indicate the closing price of the *j*-th trading day stored in the simulator. The price change ratio in percentage is calculated by the following formula:

$$\mathit{PriceChangeRatio[j]} = (\mathit{Close[j]} - \mathit{Close[j+1]}) * 100 / \mathit{Close[j+1]} \ (1 \leqq j \leqq n) \qquad (2)$$

Since the proposed model of stock trading has two parameters, i.e., δ and ε, an exhaustive search method or a brute-force approach is applied to find the optimum combination of the parameters in the sense of maximizing profits. Table II shows the result of the exhaustive search on δ and ε using NASDAQ stock price data for ten years, i.e., strictly between Nov. 30, 2011 and Dec. 1, 2021 in a long position. The maximum profit is achieved when the value of δ= 0.2% and ε= –2.4%, i.e., between –2.3% and –2.5%. The profit values in US dollar in Table II are those without the trading fee.

TABLE II. RESULT OF EXHAUSTIVE SEARCH ON δ AND ε

ε＼δ	0.1	0.15	0.2	0.25	0.3
-2.7	50383.1	51545.6	51784.2	50117.3	49072.6
-2.6	49477.1	50639.6	51784.2	50117.3	49072.6
-2.5	49957.6	51120.1	52264.7	50597.8	49553.1
-2.4	49949.1	51120.1	52264.7	50597.8	49553.1
-2.3	49949.1	51120.1	52264.7	50597.8	49553.1
-2.2	49055	50226	51370.6	50297.4	49252.7
-2.1	49393.4	50564.4	51709	50635.8	49591.1
-2	48837.7	50008.7	51153.3	49968.1	48944.5

The optimal combinations of δ and ε concerning the seven global markets are listed in Table III. The value of the parameter δ is positive except for SSEC, which indicates that the candlestick body is hollow or Doji [2], i.e., the opening price and the closing prices are virtually equal, when the price is rising. Meanwhile, the candlestick body is filled or Doji when the price is falling.

TABLE III. OPTIMAL COMBINATIONS OF δ AND ε FOR SEVEN GLOBAL MARKETS

Market	δ	ε
Euro Stoxx 50	0.05	–2.4
DJI	0	–1.7
NASDAQ	0.2	–2.4
Bovespa	0	–3.2
Sensex	0	–2.9
SSEC	–0.2	–2.3
Nikkei 225	0.05	–2.8

The value of the parameter ε is negative in all markets. This indicates that the closing price is before crossing the 5-day average upward during a price increase, whereas before crossing downward during a price decrease.

III. EXPERIMENTAL RESULTS

This section discusses simulated success rates and profitability of stock trading in the seven markets under consideration. It also includes analyses of the cumulative profits over ten-year trade simulations, which show strong liner dependency upon the number of trades.

A. *Success Rate and Profitability*

Stock trade simulations are performed over the ten years' daily stock data using the seven markets, i.e., Euro Stoxx 50 (the European Union), Dow Jones Industrial Average and NASDAQ (the United Sates), Bovespa (Brazil), Sensex (India), SSEC (China), and Nikkei 225 (Japan) [12]. The success rate of stock trading in the study is defined by the following formula:

$$\text{Success rate} = \text{the number of profitable trades} / \text{the total number of trades} \qquad (3)$$

Strictly, the number of profitable trades indicates the number of trades that result in a positive profit.

A trade profit is calculated by the subtraction of a purchasing price from a selling price. In a long position, a trader firstly purchases a stock at a market price, and eventually sells it to fix profits and/or losses. A profit is generated when a market price increases. Meanwhile, in a short position, a trader firstly borrows a stock with margin interest from a broker and sells the stock at a market price, and eventually buys them back to fix profits. A profit is generated when a market price declines.

Table IV summaries the success rates for each market in both long and short positions, with and without the trading fee or interest. Cells with the largest success rates among the markets are highlighted with a yellow background color, and cells with the smallest value are filled with a light blue color. The success rates in a short position with interest are notably low compared to other success rates. This indicates that the margin interest (1) has a significant influence on the success rates.

TABLE IV. SUMMARY OF THE SUCCESS RATES

	Success rate in long position		Success rate in short position	
	Without fee	With fee	Without interest	With interest
DowJ	72.0%	64.3%	64.0%	15.9%
NASDAQ	73.1%	70.0%	61.3%	23.2%
Bovespa	70.8%	65.9%	69.6%	31.0%
Euro Stoxx 50	70.8%	64.6%	62.3%	21.3%
Sensex	79.4%	73.2%	57.4%	20.7%
SSEC	67.4%	62.1%	68.3%	22.0%
Nikkei 225	66.1%	62.9%	59.6%	22.8%
Average	71.4%	66.1%	63.2%	22.4%

Table V shows a summary of the profitability for each market in both long and short positions, with and without the fee or interest. Like Table IV, cells with the largest success rate are highlighted with a yellow color, and cells with the smallest are filled with a light blue color.

The profit ratio is calculated by the following formula:

$$\text{Profit ratio} = \text{Simulated profits} / \text{Base stock price} \qquad (4)$$

The base stock price is a 25-day average at the beginning of the simulation, i.e., the 25-day average on Nov. 30, 2011, for each market.

TABLE V. SUMMARY OF PROFIT RATIOS

	Ratio in long position		Ratio in short position		Stock price increase ratio
	Without fee	With fee	Without interest	With interest	
DowJ	7.35	6.24	5.39	-7.55	3.02
NASDAQ	12.45	11.24	7.48	-6.76	6.01
Bovespa	8.75	7.93	7.94	-1.69	1.82
Euro Stoxx 50	6.71	5.85	5.89	-4.24	1.9
Sensex	8.93	7.80	6.46	-6.80	3.54
SSEC	6.60	5.74	6.11	-3.97	1.44
Nikkei 225	9.62	8.31	7.35	-8.03	3.41
Average	8.63	7.59	6.66	-5.58	3.02

NASDAQ has the maximum profit ratio of 11.24 times the 25-day average on Nov. 30, 2011 in ten years in a long position with the trading fee. Meanwhile, profit ratios in a short sale are catastrophic. Losses reach 6.76 times the base stock price in NASDAQ. Again, the simulations prove that the margin interest (1) has a heavy negative impact on profits.

B. *Analysis of Cumulative Profits*

One of the important aspects of stock trading is stability of investment returns on an annual basis. Figure 5 shows the graphs of cumulative profits in a long position with the trading fee for each of the seven markets being discussed. The cumulative profits' graphs without the fee have the same shapes except for the slopes of increase are 1.001 times of those in Figure 5.

As the graphs show, the cumulative profits almost constantly increase as stock trades continue. NASDAQ achieves the best ratio of 11.24 times at the end of a simulation. Linear regression equations with the R-squared (R^2) are added near the graphs. R^2 is a statistical measure of how close the original data are to the approximate regression line. All R^2s of the seven markets are greater than 0.9302 which indicates significant correlation between the cumulative profits and the number of trades.

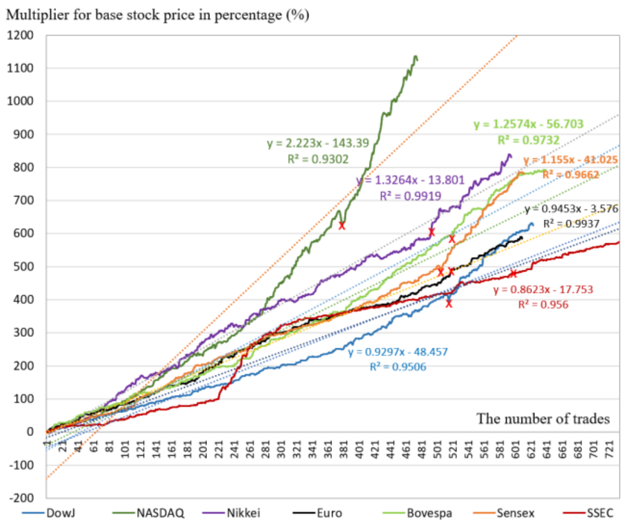


Figure 5. Cumulative profits in long position with fee.

Scarlet “X” marks on the graphs indicate the trade in the vicinity of the lowest price day, i.e., around March 18, 2020 [13]. Since that day, most stock markets have seen obvious increases in cumulative profits.

Figure 6 shows the line graphs of cumulative profits in a short position without the margin interest. All R^2s are greater than 0.9486 indicating significant correlation between the cumulative profits and the number of trades. In other words, the proposed reversal criterion in Figure 3 fairly predicts the bearish trend in all seven markets on an annual basis.

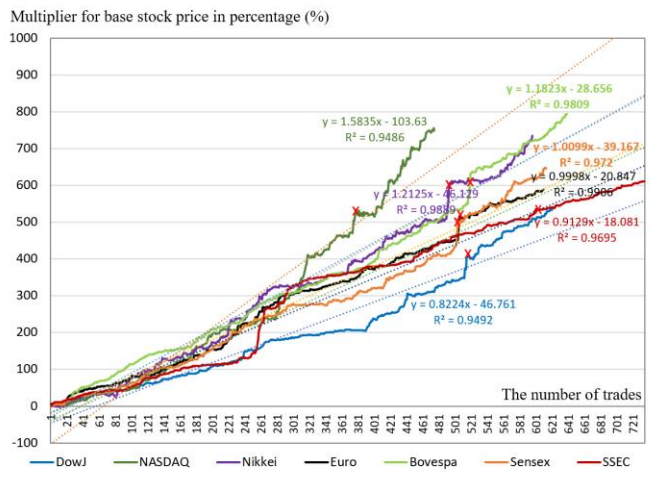


Figure 6. Cumulative profit in short position without margin interest.

Figure 7 shows the line graphs of cumulative profits including the margin interest. All lines are in the negative region, which means losses. The margin interest defined by (1) overwhelms the profits generated by the simulator.

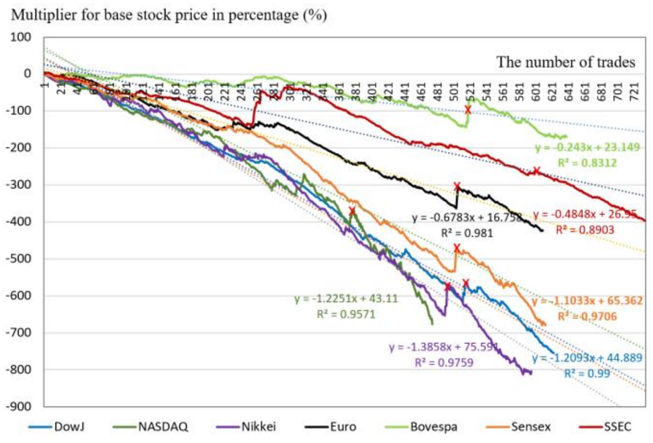


Figure 7. Cumulative profit in short position with margin interest.

Table VI shows the number of stock-holding days for all trades using NASDAQ historical data in long and short positions. Bullish reversals of 158 out of 476, i.e., 33.2%, are proved to be fail, which causes to change from a long position to a short position in one market day. On the other hand, bearish reversals of 227 out of 475, i.e., 47.8%, occurred in one market day. In this case, most trades yield losses for the payment of the margin interest of slightly greater than 1.15% according to (1).

TABLE VI. THE NUMBER OF HOLDING DAYS IN NASDAQ

Holding Days	Long position		Short position	
	Number of occurrences	Ratio (%)	Number of occurrences	Ratio (%)
1	158	33.2%	227	47.8%
2	111	23.3%	121	25.5%
3	69	14.5%	50	10.5%
4	39	8.2%	32	6.7%
5	24	5.0%	18	3.8%
6	20	4.2%	7	1.5%
7	15	3.2%	11	2.3%
8	15	3.2%	3	0.6%
9	7	1.5%	3	0.6%
10	6	1.3%	2	0.4%
11	4	0.8%	1	0.2%
12	4	0.8%	0	0.0%
13	1	0.2%	0	0.0%
14	2	0.4%	0	0.0%
15	1	0.2%	0	0.0%
Total	476	100.0%	475	100.0%

In summary, the developed simulator has lower accuracy in a bearish trend prediction than a bullish trend prediction. Furthermore, the margin interest for borrowing stocks for taking a short position worsens profits, leading to notable losses.

C. *Trading Frequency and Profits*

As shown in Table VI, 33.2% of the trades based on the proposed method have one-day stock holding in a long position and 47.8% in a short position. Figure 8 shows a candlestick chart that includes a trade in which the number of days to hold stock is one, causing high frequency of trading. Since the stock price rises on the market day *t*, the simulator buys a stock in anticipation of a price increase. Since the stock price remarkably fall on the day *t+3*, the simulator sells the stock short. However, on the next day *t+4*, the price of the stock notably goes up, signaling the simulator to change positions and buy a stock. The number of holding days is one for this trade. Because the traded price of the stock on the day *t+4* is higher than the traded price on the day *t+3* in Figure 8, a loss, i.e., –SP1, is incurred.

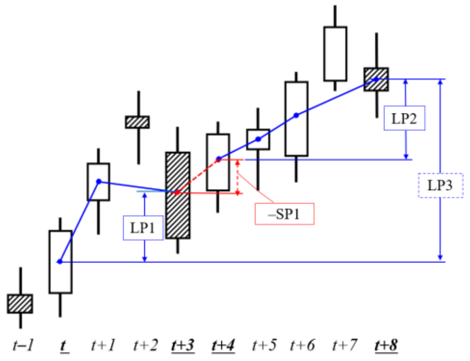


Figure 8. Candlestick chart including one-day stock holding.

The losses of –SP1 also worsen profits in a long position. If the short sale on the day *t+3* was avoided, the simulator could earn a profit LB3 calculated by the following formula:

$$LP3= LP1 + SP1 + LP2 \quad (5)$$

Figure 9 shows a candlestick chart consisting of favorable trades whose holding days of a stock are longer than one. The developed simulator sells the stock in a short position because a stock price falls on the market day *t+3*. The stock price continues to fall on the next day *t+4*.

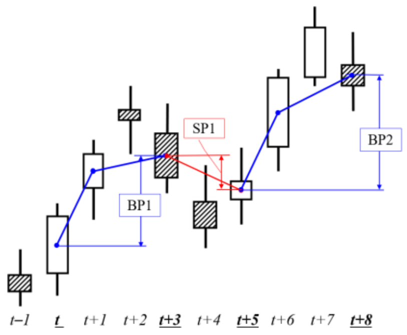


Figure 9. Candlestick chart consisting of favorable trades.

Since the stock price rises on the day *t+5*, the simulator takes a long position and buys the stock back. In this case, the traded price of the stock on the day *t+5* is fortunately lower than that on the day *t+3* as shown in Figure 9, a profit, i.e., SP1, is generated.

It is important to note that a trade with a one-day stock holding frequently results in losses, and causes low success rates. Methods using candlestick patterns are effective to avoid unsuccessful trading, which is discussed in [13].

D. Analysis of Trades in Long Position

Figure 10 is a scatter plot of all trades in a long position with the trading fee using the NASDAQ historical stock data. The Y-axis represents profits and the X-axis shows the number of days to hold a stock. As R^2 is 0.2161, there is an insignificant correlation between profits and holding days.

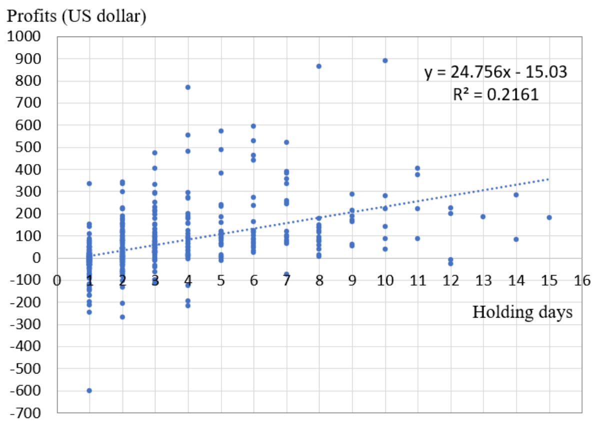


Figure 10. Scatter plot of profits and holding days in long position.

Points with a negative value on the Y-axis yield losses. They mostly occur in the range of one to four on the X-axis. In other words, trades with more than four holding days are generally profitable.

Figure 11 shows the candlestick chart of the most profitable trade whose point is located at the upper center in Figure 10. The trade begins on Mar. 30 and ends on Apr. 14, 2021 with ten holding days. The simulator makes sensible buying and selling decisions as far as this trade is concerned. The profits are UD$890.80 with the fee, and UD$903.90 without the fee.

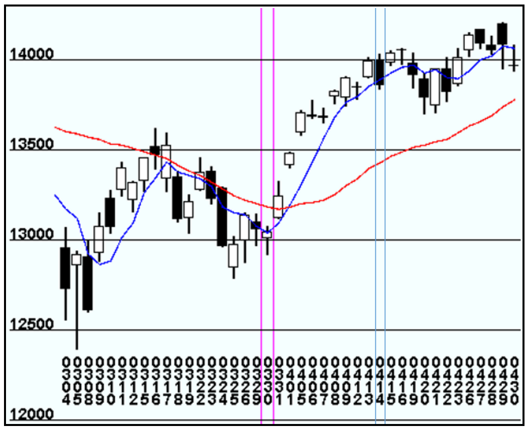


Figure 11. Candlestick chart on the most profitable trade occurred in 2021.

The trade that ends in the least profit is located at the lower left corner in Figure 10. The magenta vertical lines in Figure 12 indicate the traded date.

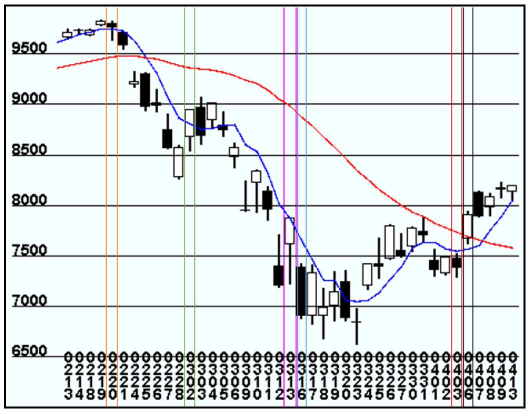


Figure 12. Candlestick chart on the least profitable trade.

The simulator takes a long position on Friday, Mar. 13, 2020. But the stock price plummets on the very next market day, i.e., Monday, Mar. 16, 2020. The simulator reverses the position, causing a loss of US$601.70 with the trading fee or US$594.00 without the fee. During the periods when prices rise and fall frequently, as in this example, it is difficult to make good decisions based on the proposed criterion in Figure 2. The use of sophisticated candlestick patterns is discussed in [13] for achieving profitable trades.

E. Analysis of Trades in Short Position

Figure 13 shows a scatter plot of all trades in a short position with the margin interest (1). The most lucrative trade is located at the upper center in Figure 13. The trade begins on Feb. 20 and ends on Mar. 2, 2020 with seven holding days as the orange and green vertical lines indicate in Figure 12. The profits are US$847.80 with the interest or US$965.40 without the interest.

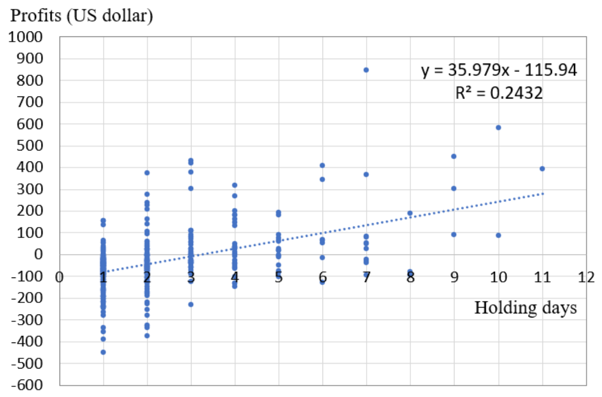


Figure 13. Scatter plot of profits and holding days in short position.

The most unsuccessful trade begins on Apr. 3, 2020 and ends on the following day, which is located at the lower left corner. Coincidentally, the trade with the worst losses is included in Figure 12, and is indicated by the red and dark gray vertical lines. As is the case for a long position, the simulator changes to the opposite position only in one market day. The losses are US$447.50 with the interest or US$361.50 without the interest.

IV. Conclusion and Future Work

This study deals with a stock trade simulator that tries to make profits in both long and short positions. The trade opportunities are decided based on the criteria using two parameters, i.e., the length of the candlestick body, and the difference between the closing price and the 5-day moving average. The symmetry of bullish and bearish reversal patterns is presumed. The optimal values of the parameters are determined thorough an exhaustive search method for seven global markets, i.e., Dow Jones Industrial Average, NASDAQ, Bovespa, Euro Stoxx 50, Sensex, SSEC, and Nikkei 225.

Stock trade simulations with and without stock trading fees are performed using the daily historical stock data of the seven markets for a period of ten years. The profits simulated in each market in a long position are noteworthy. The best performance is recorded in NASDAQ with 11.24 times the stock price at the beginning of the trade in ten-year simulation with fees. However, significant losses occur in each market in a short position. Margin interest to loan stocks is generally greater than a simulated profit in a short trade, especially when the trading period is one or two days.

Through a series of experiments, we find that the developed simulator has some limitations. The average of success rates in a long position remains in the middle of 60%, while it is at the beginning of 20% in a short position. The improvement of success rates is a major issue. In a highly volatile period of price fluctuation, the proposed criteria tend to result in a false prediction, incurring notable losses. There is an idea to refrain from trading stocks during the periods of market turbulence. This means developing a simulator with three states, i.e., buying, selling, and no trading.

Another interesting study subject is to simulate trades using individual stocks. This type of research is expected to be an important contribution toward practical stock trading.

References

[1] W. Utami, L. Nugroho, and A. Farida, “Fundamental Versus Technical Analysis of Investment: Case Study of Investors Decision in Indonesia Stock Exchange,” Journal of Internet Banking and Commerce, vol. 22, pp. 1-18, June 2017.

[2] “Technical Analysis,” Cambridge Univ., Available from: http://www.mrao.cam.ac.uk/~mph/Technical_Analysis.pdf, pp. 1-179, February 2011.

[3] M. C. Thomsett, “Bloomberg Visual Guide to Candlestick Charting,” John Wiley & Sons, Inc., Hoboken, New Jersey, 2012.

[4] M. J. Horton, “Stars, crows, and doji: The use of candlesticks in stock selection,” Quarterly Review of Economics and Finance, vol. 49, pp. 283–294, November 2007.

[5] R. B. Marshall, R. M. Young, and R. Cahan, “Are candlestick technical trading strategies profitable in the Japanese equity market?” Review of Quantitative Finance and Accounting, vol. 31, pp. 191–207, August 2008.

[6] P. Tharavanij, V. Siraprapasiri, and K. Rajchamaha, “Profitability of Candlestick Charting Patterns in the Stock Exchange of Thailand,” SAGE journals, pp. 1–18, October 2017.

[7] G. Caginalp and H. Laurent, “The predictive power of price patterns,” Applied Mathematical Finance, vol. 5, pp. 181-206, June 1998.

[8] Y.-J. Goo, D.-H. Chen, and Y.-W. Chang, “The application of Japanese candlestick trading strategies in Taiwan,” Investment Management and Financial Innovations, vol. 4, pp. 49–79, January 2007.

[9] N. Goumatianos, I. T. Christou, and P. Lindgren, “Useful pattern mining on time series: Applications in the stock market,” Proceedings of the 2nd International Conference on Pattern Recognition Applications and Methods (ICPRAM 2013), pp. 608-612, February 2013.

[10] M. Zhu, S. Atri, and E. Yegen, “Are candlestick trading strategies effective in certain stocks with distinct features?” Pacific Basin Finance Journal, vol. 37, pp. 116–127, April 2016.

[11] T.-H. Lu, Y.-. Chen, and Y.-C. Hsu, “Trend definition or holding strategy: What determines the profitability of candlestick charting?” Journal of Banking & Finance, vol. 61, pp. 172–183, December 2015.

[12] “Major World Market Indices,” Available from: https://www.investing.com/indices/major-indices/, December 2021.

[13] Y. Udagawa, “Stock Price Prediction Using Average Based Trend Indicator Derived from Analysis of Impact of COVID-19 Pandemic,” International Journal on Advances in Systems and Measurements, v 15 n 1&2 2022., in press.

Machine Learning Stacking Ensemble Model for Predicting Heart Attacks

Muath A. Obaidat
Department of Computer Science
City University of New York
New York, NY 10019
email: muobaidat@ccny.cuny.edu

Alex Alexandrou
Security and Emergency Management
John Jay College of Criminal Justice
New York, NY 10019
email: aalexandrou@jjay.cuny.edu

Samantha Sanacore
Department of Computer Science
City University of New York
New York, NY 10019
email: ssanacore@jjay.cuny.edu

***Abstract*— To mitigate the extent of one of the world's leading causes of death, heart attacks, there needs to be an improvement in the technological aspect to predict this disease more accurately. Machine learning methods have come very far in increasing prediction accuracy based on patient data. Ensemble methods have exhibited improvement compared to individual classifier models. For this study, the goal is to develop a Machine Learning model to reach a very high level of accuracy for predicting myocardial infarction, otherwise known as a heart attack. A stacked ensemble model is used in this study and combines a group of three base-level classifiers such as Naïve Bayes, Random Forest, and Extreme Gradient Boosting (XGBoost). This model will help identify those who are at risk and prevent heart attacks, therefore, lowering the mortality rate globally. Diversity among strong classifiers used in this model will be a more effective way to achieve the highest accuracy. The metrics used to evaluate the prediction performances are accuracy, Area Under the Curve (AUC), specificity, precision, and sensitivity. This process is carried out using RStudio and the results indicate that the proposed stacked ensemble method had a better performance under every evaluation metric compared to the individual base-level classifiers that were utilized.**

Keywords-machine learning; naïve bayes; random forest; extreme gradient boosting; ensemble; heartattack; accuracy.

I. INTRODUCTION

Heart attacks, also known as myocardial infarction, have been one of the leading causes of death worldwide. The number of heart attacks has gone up by millions over the past decade [1]. In The U.S. alone, someone has a heart attack every 40 seconds and 1 in 5 attacks are silent, which results in damage being done to the person even if that person is unaware [2]. These staggering numbers are disturbing and will only get worse if no action is taken. Aside from the conventional dieting and exercise, something else that can be proactively done is to improve how we detect and predict heart attacks.

A heart attack is caused when a blood clot prevents oxygen from entering the heart, causing enough damage to the muscle cells that they start dying. Blood clots are formed when a buildup of plaque, made up of deposits, cholesterol, and other substances, ruptures and causes a blockage in the arteries [3]. It is important to evaluate and diagnose the patients early on and take steps to reduce or eliminate any of the risk factors, whether it be genetic or acquired. This should be done before there is any irreversible damage, which would eradicate the ability to provide any treatment to the patient. If a patient has a blockage in a heart artery, the most common and effective procedure to reduce the risk of heart attack and improve the supply of blood, oxygen and reduce blockage in a coronary artery to the heart is coronary bypass surgery [4], which is relatively very rarely used nowadays. However, the most used procedure is angioplasty where stents are inserted to enlarge the artery; they are further absorbed by the artery wall [5].

Predicting and diagnosing patients early can possibly save someone's life. This is where Machine Learning (ML) can have a major impact in the medical field. ML methods can be implemented to determine who is at risk of suffering a heart attack and get treatment.

ML implements data that is used as input and utilizes algorithms that can be trained to predict certain outcomes based on features from a provided dataset. ML continues to constantly evolve and has become beneficial in programming tasks that can predict or classify data. Classifiers separate data into classes and the prediction functions create a trend line, also known as the line of best fit, to fit a shape to get the closest to the data points. ML can fall into three categories: supervised machine learning, unsupervised machine learning, and semi-supervised learning. The classifiers used in this study fall under supervised machine learning [6].

An ensemble model is a valuable ML algorithm and can provide a variety of techniques for classification and regression. This study focuses on classification techniques to improve the prediction accuracy of heart attacks from a dataset. There are several ensemble techniques available such as bagging, boosting, stacking, and blending. Stacking is the technique that is relevant in our proposed model. The stacking ensemble model creates a strong meta-classifier, which is trained on features that are outputs from the combination of weak or base level classifiers [7].

With the use of ML, algorithms are trained to find patterns using a large dataset to make predictions. Some past researchers have used ML models to predict heart attacks using single classifiers and have had some high success rates. This proposed ML model plans to use a combination of three classifiers to achieve a high accuracy percentage, which would then result in a high success rate. This project will use a combination of datasets, such as Cleveland database [8] and Statlog (Heart) [9], which consists of 573 patients, 13 features and 1 target column. The target features determines whether there is a “presence” or “absence” of heart disease, which is scaled as 0 or 1.

The 13 features in each patient’s data consist of age, sex, chest pain type (4 values) (cp), resting blood pressure (trestbps), serum cholesterol in mg/dl (chol), fasting blood sugar > 120 mg/dl (fps), resting electrocardiographic results (values 0, 1, 2) (restecg), maximum heart rate achieved (thalach), exercise induced angina (exang), oldpeak = ST depression induced by exercise relative to rest, the slope of the peak exercise ST segment (slope), number of major vessels colored by fluoroscopy (CA), thal: 1=normal; 2=fixed defect; 3=reversible defect.

This study will use the analysis and evaluation of the features, with some risk factors incorporated in the patient’s data to help identify and accurately predict heart attacks. Out of the data, 70% is the training dataset and the remainder 30% of the data is considered the testing dataset. The training dataset is used to train the three classifier models. The testing dataset is used to test and evaluate the stacked ensemble model using the three classifier models. The three models used for classification are Random Forest, Naïve Bayes, and extreme gradient boosting (XGBoost). These classifiers are included in the ensemble model using the stacking technique. This ensemble method is proposed to achieve the best prediction accuracy of heart attacks and is evaluated by calculating the performances of accuracy, AUC, specificity, precision, and sensitivity.

The contributions made to this study include a unique combination of three base-level classifiers in the stacked ensemble model. The train control uses the 10-fold cross validation method for resampling the data. The meta-classifier uses a generalized linear model method.

The following sections make up the rest of the paper. Section II discusses the related work of this topic. The description of the proposed solution, along with the methodology, equations and implementation are presented in Section III. The results and discussion are reported in Section IV. Section V concludes the paper and provides some future work directions.

II. RELATED WORK AND BACKGROUND

There are many studies involving different machine learning techniques utilized to improve the prediction accuracy on heart attacks or heart disease. There are also some studies discussed in this paper that have used ensemble methods for prediction using various other datasets. The ensuing reviews of these methods and results are portrayed in this section.

Gao et al. [10] used K-nearest Neighbor, Support Vector Machine, Decision Tree, Random Forest, and Naïve Bayes to classify heart disease. The models were compared by using a boosting and bagging ensemble method with feature extractions algorithms such as linear discriminant analysis and principal component analysis. The results concluded that the bagging ensemble method with the principal component analysis feature extraction and Decision Tree achieved the best overall performance as compared to other models.

Parthasarathy et al. [11] used data from Cleveland UCI repository to perform heart disease classification by using Random Forest, Decision Tree, Support Vector Machines and Naïve Bayes based on only 9 of the 13 features provided. This study concluded that Random Forest classifier provided the best precision and accuracy when training the model using feature selection. Their model resulted in a prediction accuracy of 79.47%.

Obasi and Shafiq [12] compared Naïve Bayes, Logistic Regression and Random Forest using a dataset of 4838 observations combining Cleveland heart disease dataset, cardiovascular disease dataset and Framingham Heart study dataset. The study determined that Random Forest was the most accurate with 92.44%, followed by Naïve Bayes and Logistic Regression with accuracies of 61.96% and 59.70%, respectively.

Fang et al. [13] discussed how genes are important risk factors for myocardial infarction. The Recursive Feature Elimination (RFE) algorithm was implemented to find the 15 genes with the highest prediction accuracy. They integrated these genes using the GSE61144 dataset to construct a Support Vector Machine to predict the patients who have a high risk for myocardial infarction or heart attack. The outcome of the proposed model resulted in a 92% prediction accuracy.

Alaa et al. [14] developed a machine learning based model using AutoPrognosis to predict cardiovascular disease. The dataset contained more than 400,000 participants from the UK Biobank and included 473 features for each participant. The proposed AutoPrognosis model had a better AUC performance compared to the standard Framingham score and Cox PH models.

Revathi and Kavitha [15] compared Naïve Bayes, Instance-Based learning with parameter k (IBK) and Random Forest using the University of California, Irvine (UCI) heart disease dataset containing 270 observations. The study concluded that Naïve Bayes had the best performance with an 83.70% accuracy followed by Random Forest with 81.48% and IBK with 75.18%.

Gupta et al. [16] compared several models using the Cleveland heart disease dataset. The top 3 models with the highest accuracy were Naïve Bayes, AdaBoost and Boosted Tree with results of 86.42%, 86.21% and 85.75%, respectively. These top 3 models are implemented in the ensemble model and the accuracy of this model was the highest of all the models compared, with 87.91%.

Ali et al. [17] applied multiple techniques using the Hungarian and Cleveland datasets to their proposed ensemble approach to predict heart disease. This model is compared with classifiers such as Support Vector Machine, Logistic Regression, Multilayer perceptron, Random Forest, Decision Tree and Naïve Bayes, based on feature fusion, feature selection and weighing techniques. The model used the LogitBoost boosting algorithm and the proposed feature fusion approach to obtain results higher than existing ones with 98.5% accuracy.

Palaniappan and Awang [18] developed a prototype Intelligent Heart Disease Prediction System (IHDPS) using Decision Trees, Naïve Bayes, and Neural Network. This system was built using CRISP-DM to build the models and used patient records from the Cleveland dataset. The Naïve Bayes model gives the highest accuracy of predicting heart disease with 95% followed by Decision Tree with 94.93% accuracy and Neural Network with 93.54% accuracy.

Tama et al. [19] used a stacked ensemble model in their research to predict coronary heart disease. This ensemble is a combination of classifiers such as Random Forest, Gradient Boosting Machine and Extreme Gradient Boosting, while applying Particle Swarm Optimization (PSO)-based feature selection and using datasets from Z-Alizadeh Sani, Cleveland, Hungarian and Statlog. This proposed model achieved a prediction accuracy of 98.13%, 93.55% and 91.18% which is the highest among other studies. It is compared with using Z-Alizadeh Sani, Statlog and Hungarian datasets, respectively.

Zhang et al. [20] compared the performances of eight base classifiers and chose the three with the best AUC results to be used for the stacking-based ensemble model to predict the risk of 30-day re-admission in patients with acute myocardial infarction. The proposed stacked model used an under-sampling method of neighborhood cleaning rule and a feature selection method of SelectFrom Model (SFM). The proposed stacked model had an AUC of 0.72, which was better than all the other classifiers the study compared.

Muhammad et al. [21] have researched which models and techniques result in an improved performance for the prediction of heart disease. Ten classifiers were compared and implemented by applying different feature selection algorithms to achieve the best performance possible. The top two classifiers without applying feature selection algorithms were Extra-Tree and Gradient Boost which had 92.09% and 91.34% accuracies, respectively. After experimenting with the feature selection algorithms, Extra-Tree had the highest accuracy, 94.41%, when the Relief algorithm was applied, and Gradient Boost had the highest increase to 93.36% with the Fast Correlation Based Filter algorithm.

To this end, we propose an ensemble model that uses fewer factors and variables to predict heart attacks. Nonetheless, the proposed model has better performance metrics than many previously discussed research studies while some of these studies have better performance in other metrics. The difference, however, is not very significant, which means the proposed scheme is well suited to predict heart attacks with a high degree of accuracy and proactively.

III. METHODOLOGY AND THE PROPOSED SCHEME

The objective of the proposed stacked algorithm is to improve the performance of being able to predict heart attacks. Fig. 1 displays the flow of the proposed scheme. This includes data splitting, training the base-level models, meta-classifier, and the evaluation metrics.

First, the data used for this model is collected from two databases. The Cleveland heart disease dataset has 303 observations and the Statlog (heart) dataset has 270 observations. Both datasets have the same number, 14, types of variables which allows them to be combined for a larger dataset to be used for this study. Once the data is imported, it is split into a training set and a testing set. The training set contains 70% of the data and the testing set contains 30% of the data.

The training set is used to train the models to learn about the features of the data. This is essential for future use to accurately predict the new data from the testing set. The dependent variable, y, is the target variable used for the prediction of heart attacks. The independent variable, x, is the variable or feature from the dataset that is used to determine the prediction results.

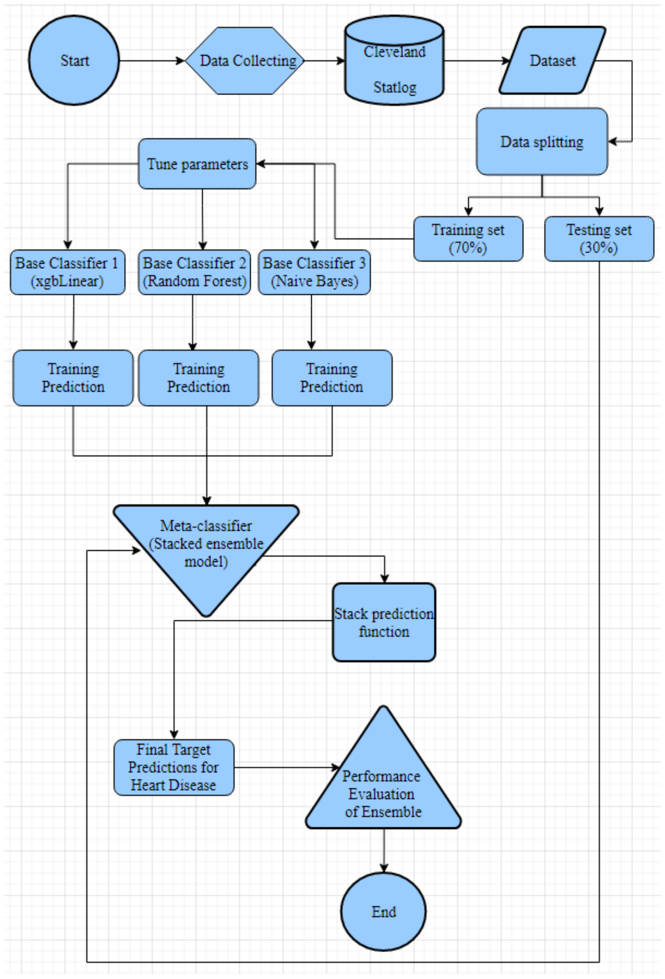

Start
Data Collecting
Cleveland
Statlog
Dataset
Data splitting
Tune parameters
Training set (70%)
Testing set (30%)
Base Classifier 1 (xgbLinear)
Base Classifier 2 (Random Forest)
Base Classifier 3 (Naive Bayes)
Training Prediction
Training Prediction
Training Prediction
Meta-classifier (Stacked ensemble model)
Stack prediction function
Final Target Predictions for Heart Disease
Performance Evaluation of Ensemble
End

Figure 1. Scheme of proposed model to predict heart attacks.

The next step is to tune the parameters with 'trainControl' before training the base-level classifiers: Random Forest, XGBoost and Naïve Bayes. The tuning parameters would cross-validate (cv) the results 10-fold and save the predictions for later use with the class probabilities set to

'True'. The 'trainControl' is utilized in the three base-level classifiers.

The three base-level classifiers use the training set and each are resampled. This set of data is used to train the models to learn the features to predict the target variable, the factor determining the possibility of suffering a heart attack. After each model is optimized with the parameters and trained, the prediction accuracy results are shown. The results of each model are averaged and merged into a single array by implementing the 'caretList' function. This function includes the target variable, the training set and the 'trainControl' parameters. This will be the input data for the meta-classifier/stacked model.

The Naive Bayes classifier is based on Bayes Theorem [22]:

$$\mathrm{P(A|B)} = \frac{\mathrm{P(B|A)*P(A)}}{P(B)} \tag{1}$$

For this model, A and B are replaced with y, which is the output prediction variable, and x, which is the input variable. This equation solves for the probability of y given the input features, x, from the training set. The equation is re-written to adapt to the independent variables:

$$\mathrm{P(X|y) = P(x_1|y) * P(x_2|y) * \ldots * P(x_n|y)} \tag{2}$$

The probability of X, P(X), is a constant so it can be removed and replaced with a proportionality:$P(y|X) \propto P(X|y) * P(y)$. The last step of the Naïve Bayes classifier is to choose the class y with the maximum probability. This is accomplished by using "argmax" operation which finds the argument that results in the maximum y value:

$$\mathrm{y = \ argmax_y[P(y) * \textstyle\prod_{i=1}^{n} P(x_i|y)]} \tag{3}$$

The Random Forest Classifier is made up of multiple decision trees for each patient observation. The algorithm starts with using random observations from the training set. The next step in the algorithm is to create a decision tree for every one of these observations to produce results containing the prediction for heart attacks. For each variable, a prediction is made, and voting is performed to determine the final prediction result for that observation [23].

The stacked ensemble model, 'stack.model', is implemented by using the 'caretStack' function with a generalized linear model (glm). The model uses the merged prediction results from the previous step, as well as the 'Accuracy' metric. The same tuning parameters to train the base-level models are utilized for this model under 'stackControl'. The next step is to apply the 'stack.model' to the stack prediction function. This function produces the probability and uses 'as.data.frame', which returns a frame containing columns comparing the estimated prediction results using the input data from 'stack.model' and the actual results from the testing set.

The stacked ensemble model is constructed by using the dataset as the input. The dataset is $\mathrm{D = \{x_i, y_i\}_{i=1}^{m}}$, where x_i represents the feature vectors and yi represents their classifications. The first phase of the model consists of a set of three base-level classifiers, h_t. The XGBoost classifier is h_1, Random Forest classifier is h_2 and Naïve Bayes classifier is h_3. Each base-level classifier, $h_{t,}$ is trained by applying level 0. Level 0 is known as the data from the training set inputs which the level 0 classifiers make predictions from [24]. After the classifiers are trained, the next step is to construct a new dataset that contains $\{x_i^{new}, y_i\}$ [25], where:

$$X_i^{new} = \{h_1(x_i), h_2(x_i), \ldots, h_T(x_i)\} \tag{4}$$

This step uses the predictions from the previous step as the new input for the meta-classifier. The final step is to learn the meta-classifier by applying a generalized linear model (glm) and training it. This new classifier, h^{new}, is based on the recently constructed dataset. The final output is displayed with H being the stacked ensemble model:

$$H(x) = h^{new}\,(h_1(x), h_2(x), \ldots, h_T(x)) \tag{5}$$

The last step is to evaluate the metrics for the stacked ensemble model of the predicted results and the actual results using a confusion matrix. The metrics used to evaluate the prediction performance are accuracy, AUC, specificity, precision, and sensitivity. A confusion matrix shows the prediction results which are categorized into four sections. The True Positive (TP) outcome is defined as predicted true and true in reality. Another outcome is True Negative (TN), which means it is predicted false and false in reality. The False Positive (FP) outcome is defined as predicted true and false in reality, whereas the False Negative (FN) means that it is predicted false and true in reality [26]. Accuracy is defined as the number of correct predictions divided by the total number of predictions made. The closer the accuracy is to 100 percent, the stronger the performance of the model; the equation follows [26]:

$$\mathrm{Acc} = \frac{\mathrm{(TP+TN)}}{\mathrm{(TP+TN+FP+FN)}} \tag{6}$$

For AUC, the closer the score is to 1, the better the model would be at differentiating between positive and negative predicted and actual values, and vice versa, with an underperforming model. Precision is defined as the number of correct positive results divided by the number of positive results predicted by the model. The precision equation is [27]:

$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{(TP+FP)}} \tag{7}$$

The specificity metric gives us the true negative rate and is defined as the proportion of actual negative results that are correctly identified, also known as true negatives, to everything classified as negative. The equation for specificity follows [28], where TNR is true negative rate.

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{(TN+FP)}} \tag{8}$$

Sensitivity gives us the true positive rate, which is the proportion of actual positive predictions that are correctly identified to everything classified as positive [29], where TPR is true positive rate.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{(\mathrm{FN}+\mathrm{TP})} \tag{9}$$

IV. Results and Discussion

This section discusses the results of the proposed model's performance along with the performance of the three individual classifier models. The overall performance of the ensemble model outperforms Random Forest, XGBoost and Naïve Bayes. The stacked ensemble model had the highest performance in accuracy, AUC, specificity, precision, and sensitivity with 80.11%, 85.53%, 84.21%, 85.89% and 76.84%, respectively.

In Fig. 2, the ensemble model achieved the highest accuracy performance at 80.12%. The second-best model was the Random Forest classifier model with an accuracy of 78.95%. The third best model was XGBoost with an accuracy of 74.85%. The worst accuracy performance of the group was Naïve Bayes with 73.1%.

Displayed in Fig. 3 is the AUC performance of each model. The ensemble model had the best results with 85.53% AUC followed by Random Forest with 78.97% and XGBoost with 78.98% and then Naïve Bayes with the lowest AUC of 74.15%. Fig. 5 shows the specificity results of the ensemble model, Random Forest, XGBoost and Naïve Bayes, which are 84.21%, 82.05%, 78.67% and 77.78%, respectively.

The specificity performance metric is presented in Fig. 4. As can be seen from Fig. 4, the ensemble model has the highest specificity, which reaches around 84.21%. It is clear that the proposed model achieves a very significant specificity as compared to other benchmark models in the study.

The precision metric is presented in Fig. 5. It shows that the best result is at 85.89%, which comes from the ensemble model. The next best result is at 83.53% from Random Forest and the worst performing models based on precision are both XGBoost and Naïve Bayes with 81.18%.

The sensitivity results conclude the evaluation of the performing models in Fig. 6, with the ensemble model having the best result with 78.84%; the next highest sensitivity result is Random Forest with 76.34%. The last two-classifier models, XGBoost and Naïve Bayes, have results of 71.88% and 69.7%, respectively.

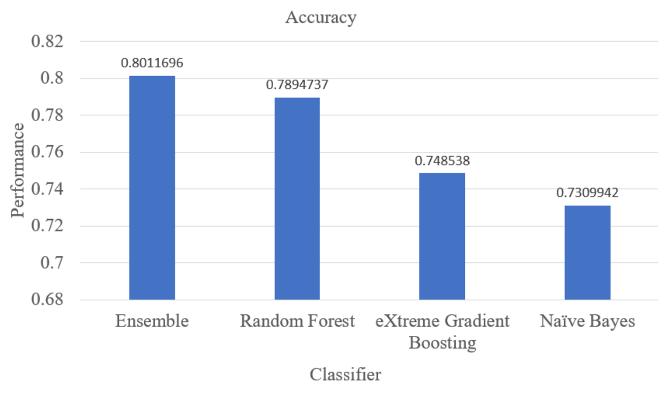


Figure 2. Accuracy

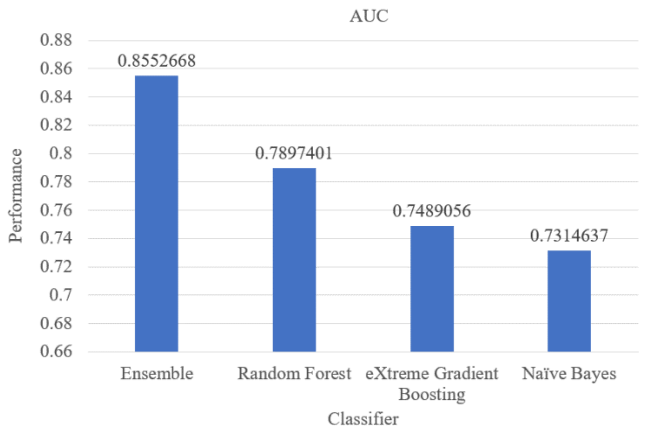


Figure 3. Area under the curve (AUC)

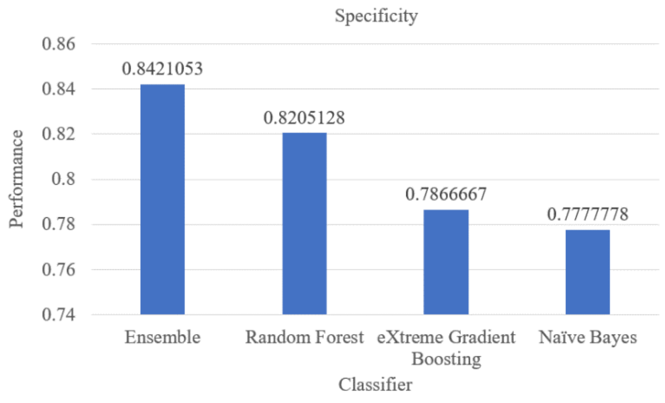


Figure 4. Specificity

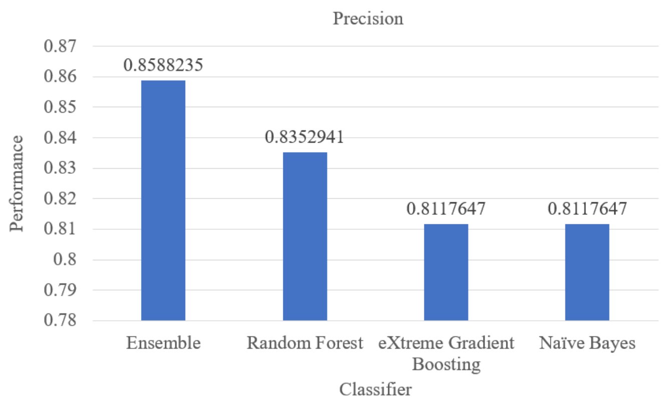


Figure 5. Precision

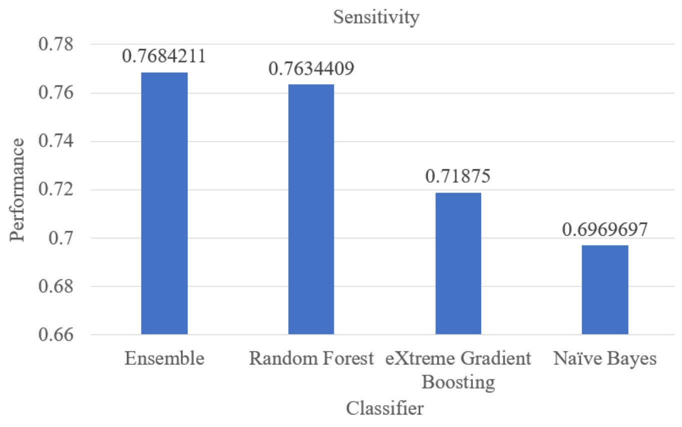


Figure 6. Sensitivity

V. CONCLUSION AND FUTURE WORK

In this work, we proposed a stacked ensemble model to improve the level of accuracy for the prediction of heart attacks. The stacked ensemble model achieved the highest results when evaluating the accuracy, AUC, specificity, precision, and sensitivity metrics compared to other schemes used in the study. The method had the highest accuracy performance, which was 1.16% higher than the next best performance from Random Forest. The performances of the three classifiers such as, Random Forest, Naïve Bayes and XGBoost, underperformed the ensemble model. Although these classifiers had lower performances individually, when they were combined, the performance improved significantly.

The proposed model can help reduce the number of deaths caused by heart attacks worldwide. Patients would be able to get early diagnoses and receive treatment by medical professionals in a timely manner. If the risk factors associated with the patient are determined to be genetic, treatments can help reduce the risk. An acquired risk factor due to lifestyle can be resolved by making changes to the patient’s daily routine, which can decrease the probability of a heart attack.

This study can be enhanced by incorporating a larger dataset or by applying feature selection algorithms and adjusting the hyper parameters of the base-level classifier models. More classification algorithms can be compared to determine which outcome produces the best performance model that more accurately predicts heart attacks. In addition, investigating when these ML models fail would be of interest focus of research.

REFERENCES

[1] World Health Organization, “The top 10 causes of death,” World Health Organization: WHO, Dec. 09, 2020. https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death. [retrieved: Jan, 2022]

[2] Centers for Disease Control and Prevention, “Heart disease facts & statistics,” Centers for Disease Control and Prevention, Sep. 08, 2020. https://www.cdc.gov/heartdisease/facts.htm [retrieved: Jan, 2022]

[3] “Heart Attack,” *Hopkins Medicine*, 2019. https://www.hopkinsmedicine.org/health/conditions-and-diseases/heart-attack [retrieved: Jan, 2022]

[4] “Cardiac Procedures and Surgeries,” www.heart.org, 2017. https://www.heart.org/en/health-topics/heart-attack/treatment-of-a-heart-attack/cardiac-procedures-and-surgeries [retrieved: Jan, 2022]

[5] J. R. Stevens, A. Zamani, and J. I. A. Osborne, “Critical evaluation of stents in coronary angioplasty: a systematic review”, BioMed Eng OnLine 20, 46 (2021). https://doi.org/10.1186/s12938-021-00883-7

[6] IBM Cloud Education, “What is Machine Learning?,” www.ibm.com, Jul. 15, 2020. https://www.ibm.com/cloud/learn/machine-learning [retrieved: Jan, 2022]

[7] M. Alhamid, “Ensemble Models,” Medium, Mar. 15, 2021. https://towardsdatascience.com/ensemble-models-5a62d4f4cb0c [retrieved: Jan, 2022]

[8] N. Bhat, “Health care: Dataset on Heart attack possibility,” kaggle.com. https://www.kaggle.com/nareshbhat/health-care-data-set-on-heart-attack-possibility [retrieved: Jan, 2022]

[9] “UCI Machine Learning Repository: Statlog (Heart) DataSet,” Uci.edu, 2019. https://archive.ics.uci.edu/ml/datasets/Statlog+%28Heart%29 [retrieved: Jan, 2022]

[10] X.-Y. Gao, A. A. Ali, H. S. Hassan, and E. M. Anwar, “Improving the Accuracy for Analyzing Heart Diseases Prediction Based on the Ensemble Method,” Complexity, pp. 1–10,
Feb. 2021, doi: 10.1155/2021/6663455.

[11] G. Parthasarathy, L. S. Kesaragopp, M. M. Vishal, S. Manigandan and K. Kulkarni, “Analysis of machine learning algorithm for prediction of heart disease”, International Journal of Advanced Research in Computer Science, vol. 11, no. 3, pp. 42–46, Jun. 2020, doi: 10.26483/ ijarcs.v11i3.6532

[12] T. Obasi and M. O. Shafiq, "Towards comparing and using Machine Learning techniques for detecting and predicting Heart Attack and Diseases," 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 2393-2402, doi: 10.1109/BigData47090.2019.9005488.

[13] H. Fang, D. Hu, Q. Li, and S. Tu, “Risk gene identification and support vector machine learning to construct an early diagnosis model of myocardial infarction,” Molecular Medicine Reports, vol. 22, no. 3, pp. 1775–1782, Jun. 2020, doi: 10.3892/mmr.2020.11247.

[14] A. Alaaid, T. Bolton, E. Angelantonio, J. Ruddid, and M. V. Der Schaar, “Cardiovascular disease risk prediction using

automated machine learning: A prospective study of 423,604 UK Biobank participants," May 2019. [retrieved: Jan, 2022]

[15] K. K. Revathi, and K. K. Kavitha "Comparison of classification techniques on heart disease dataset," International Journal of Advanced Research in Computer Science, vol. 8, no. 9, pp. 276–280, Sep. 2017, doi: 10.26483/ijarcs.v8i9.4870.

[16] N. Gupta, N. Ahuja, S. Malhotra, A. Bala, and G. Kaur, "Intelligent heart disease prediction in cloud environment through ensembling," Expert Systems, vol. 34, no. 3, p. e12207, Apr. 2017, doi: 10.1111/exsy.12207.

[17] Ali Farman et al., "A Smart Healthcare Monitoring System for Heart Disease Prediction Based on Ensemble Deep Learning and Feature Fusion." Information Fusion, vol. 63, 2020, pp. 208–222., doi:10.1016/j.inffus.2020.06.008.

[18] S. Palaniappan and R. Awang, "Intelligent heart disease prediction system using data mining techniques," 2008 IEEE/ACS International Conference on Computer Systems and Applications, 2008, pp. 108-115, doi: 10.1109/AICCSA.2008.4493524.

[19] A. Tama, B. I. Sun, and S. Lee, "Improving an Intelligent Detection System for Coronary Heart Disease Using a Two-Tier Classifier Ensemble." BioMed research international 2020 (2020): 9816142–10.

[20] Z. Zhen et al., "A Stacking-Based Model for Predicting 30-Day All-Cause Hospital Readmissions of Patients with Acute Myocardial Infarction." BMC medical informatics and decision making 20.1 (2020): 335–335.

[21] Y. Muhammad, M. Tahir, M. Hayat, and K. T. Chong, "Early and accurate detection and diagnosis of heart disease using intelligent computational model," Scientific Reports, vol. 10, no. 1, Nov. 2020, doi: 10.1038/s41598-020-76635-9.

[22] S. Terence, "A Mathematical Explanation of Naive Bayes in 5 Minutes." Medium, Towards Data Science, 6 June 2020, towardsdatascience.com/a-mathematical-explanation-of-naive-bayes-in-5-minutes-44adebcdb5f8.

[23] Classification Algorithms Random Forest, www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_random_forest.htm.

[24] Johnny's Machine Learning Blog, "How to Develop a stacking Ensemble for Deep Learning Neural Networks in Python with Keras." 18 Jan. 2020, johdev.com/ensemble/2020/01/18/Ensemble_Keras.html.

[25] "Stacked Ensemble Learning, "web.njit.edu. https://web.njit.edu/~avp38/projects/multi_projects/ensemble.html [retrieved: Jan, 2022]

[26] N. S. Chauhan, "Model Evaluation Metrics in Machine Learning," KDnuggets, May 2020. https://www.kdnuggets.com/2020/05/model-evaluation-metrics-machine-learning.html [retrieved: Jan, 2022]

[27] A. Emma, "Machine Learning Evaluation Metrics." Medium, ML Cheat Sheet, 3 Mar. 2020, medium.com/ml-cheat-sheet/machine-learning-evaluation-metrics-b89b8832e275.

[28] S. Tavish, "Evaluation Metrics machine Learning" Analytics Vidhya, 5 Aug. 2020, https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

Big Data for Monitoring Mobile Applications

Collection and Indexing Results for Analysis and Decision

Mourlin Fabrice
Algorithmic, Complexity and Logic Laboratory
UPEC University
Cretéil, France
e-mail: fabrice.mourlin@u-pec.fr

Djiken Guy Lahlou
Applied Computer Science Laboratory
Douala University
Douala, Cameroon
e-mail: gdjiken@fs-univ-douala.cm

Laurent Nel
Leuville Objects
Paris-Saclay University
Paris, France
e-mail: laurent.nel@universite-paris-saclay.fr

Abstract—**Behavior monitoring is an activity that cuts across all software. It ensures that the application proceeds as expected. Our Big Data workflow supports data streams of varying rates. The collected data sometimes contains errors that we want to explain. To this end, we seek to trace the important events of our calculations in order to qualify the anomalies in our processing and then easily trace the origin of the problem. We have implemented a monitoring layer within mobile applications in order to perform smart control over a set of mobile devices. We have defined a Big Data workflow to collect, index and store the log data in order to submit it to an Artificial Intelligence (AI) model. We detect behavioural anomalies through the analysis of software logs deployed on embedded devices. Based on the patterns recognised in the logs, our AI model provides us with a sequence of system operations. These operations are scheduled to re-deploy a service, change a driver, perform a library update, etc. The critical points concern the management of the Android APIs with respect to the deployed software; we must manage with precision the software updates with respect to the firmware versions, among other things. In the end, management reports are built every week and issued to the maintenance team. These documents are the record of maintenance activities. They provide an explanation for periods of non-availability of equipment and for the withdrawal of obsolete equipment.**

Keywords— Big Data; indexing; log analysis; distributed application; AI model; storage server; anomaly detection.

I. INTRODUCTION

With multiple sources of information, the supervision of mobile applications becomes essential and requires administration of the different applications. Software administration is an essential facet of any application these days, from application servers to embedded applications. We want to detect anomalies by information visualization, borderline behavior or even fraudulent actions. In order to have this information, developers use log Application Programming Interfaces (APIs) to transmit all behavioral data. To make the information usable, log messages generally have a format as well as a level or hierarchy of severity. The application administrator manages the level of expressiveness per module in order not to suffer from too much information.

There are many areas of application for log messages, from system and network aspects to business applications. Database servers like Postgresql have log files containing the history of activity, from the creation of databases and triggers of application connections. Embedded applications share the same need. An Android application has access to a log API whether it is written in Java, Kotlin or C ++. An Android logger corresponds to a variable in memory. It can be stored in a file or sent to a socket.

Log messages play a key role in several phases, not only at runtime but also in all the test phase: unit, integration, system and functional. During the development phase, they provide a view of the state of the application, in terms of network, security, business, etc. In the case of an on-board application, these data cannot be displayed because the peripheral does not necessarily have a screen and it is useful to redirect the information into a persistence unit (memory card, memory stick, etc.). During the debugging phase, these same messages have a tag that allows them to be filtered, or even to specify a different level of severity from one package to another to configure the feedback. As part of this study, this effort is focused on the analysis of log messages in use.

The difficulties associated with log analysis relate first to the volume of messages received. Indeed, it grows rapidly with the number of sources. Thus, in the case of on-board application monitoring, when the fleet of peripherals is enriched with new equipment, it is no longer possible in human terms to analyze the logs with serenity. The automation of this process is required. A second difficulty arises with the flow of these messages, which depends on the use of the monitored applications. When the number of users grows, then it is necessary to set up information sampling. Finally, it is not uncommon for each embedded application to have its own log format, even if it is generated by the same API. In this case, it is necessary to consider a standardization of formats during the collection in order to be able to extract information from different sources and put them in causal relation.

All the properties linked to the processing of log messages naturally lead us to consider this work as a Big Data workflow applied to a time scale. Indeed, it is essential to react to the problems detected before they become even more serious. In this document, we present the results of our work, which began more than a year ago with the Big Data prototyping of a solution to the anomaly prediction linked to Android applications. These mobile applications are used for taking pictures of experiments in biology. Users can annotate each photo and rearrange the photos within a document linked to an experiment. Users export their final document from the mobile device to a Web server accessible from the WiFi network at the place of the experiment.

The rest of this paper is structured as follows. Section II describes the works close to our domain. Section III provides a precise description of our use case. Section IV addresses the software architecture of our distributed platform. Section V goes into finer details on our streaming approach, which includes an indexing step. Section VI focuses on our results and the impact on the maintenance task. The acknowledgement and conclusions close the article.

II. Related Works

Predictive log analysis is a widely studied topic. Part of the work focuses on enhancing the information itself. A second part concerns the use of this data to react, alert, and more generally automates a process.

A. *Log analysis methods*

Adam Oliner et al. describe, through several use cases, the information useful to report during execution for software monitoring [1]. They stress, among other things, the importance of adopting a consistent format throughout an application. They make the analogy between the follow-up of manufacturing on one meeting on a production line and the follow-up of the software activity, which is the subject of this work.

T. Yen et al. describe how to leverage distributed application logs for the detection of suspicious activity on corporate networks [2]. Their work highlights the use of the beehive tool for extracting information and producing easily exploitable messages. Analysis against a signature database is then possible.

S. He et al. present six methods for log analysis of distributed systems: three of them are supervised and three others are unsupervised. The authors make a comparative evaluation of these methods on a significant volume of log messages. They emphasize the strengths of software monitoring task automation [3].

In more constrained fields such as real time, log analysis systems must be able to detect an anomaly in a limited time. B. Debnath et al. present LogLens that automates the process of anomaly detection from logs with minimal target system knowledge [4]. LogLens presents an extensible index process based on new metrics (term frequency and boost factors). The use of temporal constraint also intervenes in the recognition of behavior pattern. Therefore, abnormal events are defined as visible in a time window while other events are not. This allows semi-automatic real-time device monitoring.

B. *Log analysis and machine learning*

The development of machine learning has greatly impacted the use of logs. Depending on the work, studies lead to the detection of anomalies or the discovery of software attacks.

Q. Cao et al. presented a work on web server log analysis for intrusion detection and server load reduction. The use of two-level AI model allowed them to increase the efficiency of their detection system. In this approach, the use of decision trees structures the log data [5].

W. Li considers that logs are a complement to the software-testing phase. Since the time allocated to testing is insufficient, he presents a failure diagnosis strategy based on the use of an AI model [6]. He provides a comparative study between several models.

There is a large body of work on network log analysis for various protocols including HTTP [7] or data-centric protocols such as Named Domain Networking (NDN) [8]. In all cases, the strategy is based on formatted messages where part of the information is filtered and then submitted to a model for prediction.

C. *Reporting of artificial intellgence prediction model*

In order to obtain a set of guidelines for the use of predictive machine learning models, it is essential to build regular reports on the quality of predictions. In the context of clinical experiments, W. Luo et al. published a rulebook for AI model development [9].

P. Henderson et al. present a systematic reporting of the energy and carbon footprints of machine learning. The authors' goal is to adapt an efficient reinforcement learning strategy and explain the reinforcement learning events [10].

L.M. Stevens et al. present a recommendation for transparent and structured reporting of Machine Learning (ML) analysis results specifically directed at clinical researchers [11]. Their goal is to convince many clinicians and researchers who are not yet familiar with evaluating and interpreting ML analyses.

D.P. Dos Santos et al. take a similar approach to the analysis of radiological images. Their quality is uneven and it becomes difficult to provide a reproducible analysis approach. It then becomes essential to build reports to explain the state of the AI model that led to certain predictions. The authors explain how to structure to help build a post analysis explanation [12].

III. Use Case Description

A. *Context Description*

In biology trainings, many experiments are done where students are asked to prepare, perform and follow up. In this context, mobile devices are provided to take pictures, record sounds, or even use the device's sensors to collect data. To save different documents in the memory of the mobile device, a software suite is installed. It allows the authentication of the user, the dating of each collected information and the transfer at the end of the experiment to a server for validation.

During an experiment, all the peripherals are connected to the laboratory WiFi network. This connection authorizes data exchange with the laboratory server, which will receive all the students' data at the end of the experiment for validation by the supervisor. This network connection is also used to send log messages to monitor the activity of each mobile device. This concerns the capture of information: taking a picture or recording a single comment or a short video. This type of recording is not often used during an experiment because several students are monitoring the same experiment and this leads to noise pollution for the other participants.

The analysis of an experiment by a group of twenty students takes place over a period of one-day maximum in the same experimentation room. This means that the connection is made with the same access point for all devices. Even if the batteries are initially charged, it is possible at any time to have a recharging point in the room.

Laboratory observation sessions can be short in the early grades, such as showing the release of gas bubbles by an aquatic plant. Students construct a document to highlight the conditions of this phenomenon and then make a video to support their comments. Then, they observe the role of light and measure its value with the light sensor of the Android tablet. A second video will show the release of gas bubbles by an aquatic plant in the presence of light. In the absence of light, the students make a comparison with pictures.

In the lab room, a group of students follows an experiment with one tablet per student. A typical scenario consists of one WiFi access point per room, a set of mobile devices and a remote storage server for document backups at the end of the session. This scenario is to be multiplied by the number of groups, possibly in parallel in different lab rooms. Two properties are thus highlighted: on the one hand, a local authentication phase on the mobile device, on the other hand a centralized storage server (see Figure 1). In addition, the lab room has a laptop for the teacher and a shared printer.

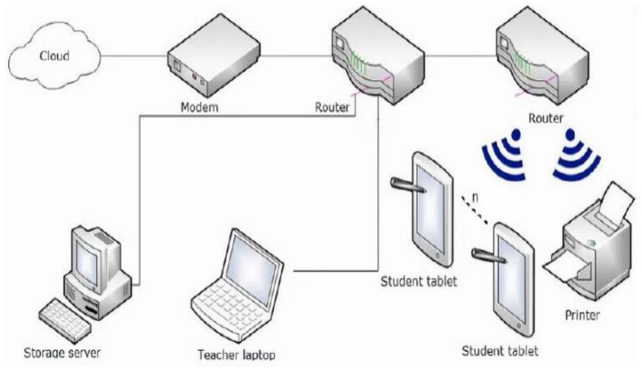


Figure 1. Network diagram of a laboratory room.

B. Scenario description

In order to describe a nominal scenario more precisely, let us take the case of a student from the beginning of a lab session to the submission of a document at the end of the session.

Figure 2 describes the general flow of the scenario in the Unified Modeling Language (UML) notation; it concerns an observation session (Lab Session). The biology teacher manages this session. Each student manages their own documents locally on their local device. Thus, the student takes notes, photos, videos and measurements via the available sensors. When his work is finished or the teacher has closed the session, a student prepares his final document, signs it and deposits it on the storage server.

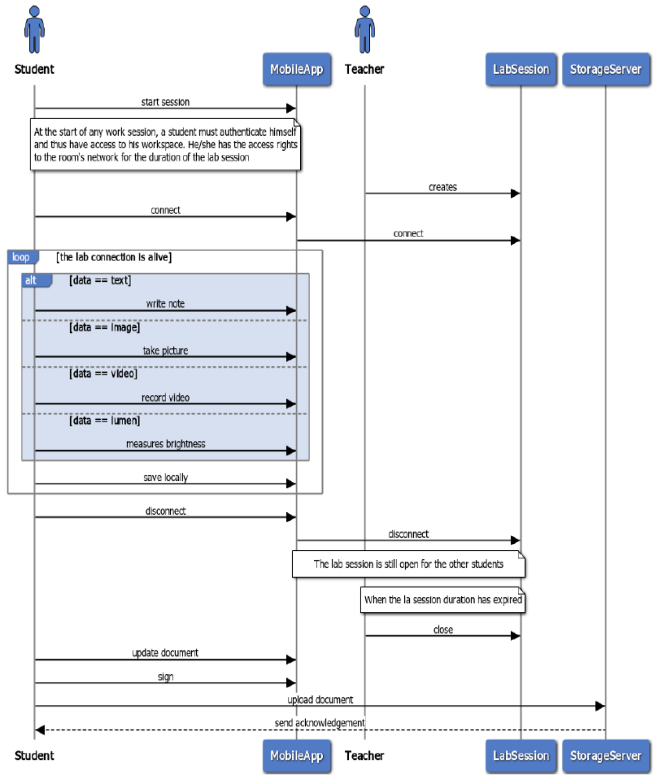


Figure 2. Nominal scenario during an experiment in a lab room.

IV. Software Architecture

If the software architecture of the business part is very simple, it is only the entry point of the information collection, which gathers the log data on the storage server. The analysis of these logs is more complex because it takes into account additional constraints: the arrival of log data continuously, the need to impose a data schema to index the information, refine the search for information and the detection of anomalies.

A. *Client application*

In order to collect information from the activity of the actors in the scenario in Figure 2, the log system of the devices is used by the students and the teacher. Our goal is to collect and cross-reference information from the various sources. Thus, it is essential to monitor the events related to the management of the laboratory sessions. In addition, any event related to an information capture or modification is useful.

1) *Mobile application*

The MobileApp instance in Figure 2 is an Android component installed on each tablet. The set of class is written in Java using the log API specific to this system. In the business part, we have defined a message format in order to easily extract the information. The creation of the log messages is done by using the android.util.Log class, which allows not only to prefix with a semantic tab, but also to add a severity level. Thus, from a set of messages, a regular expression filters the relevant results to focus on the essential data.

In addition to the business events, in this effort, we have traced the memory events provided by the garbage collector, the transmissions and receptions of information from the http network. Moreover, we used the Android Mobility Management API to define usage profiles such as Student profile. It allows business apps and data to be stored in a separate, self-contained space within a device. The teacher has full management control over the applications, data, and Student profile settings on the device, but has no visibility or access to the device's personal profile. This strong separation allows teachers to control MobileApp data and security without compromising student privacy if they are using applications other than those intended for the biology course.

We have developed a Device Policy Controller (DPC), which logs network activity. Network activity logging helps us detect and track the spread of malware on tablets. Our DPC uses network logging APIs to report the Transmission Control Protocol (TCP) connections and Domain Name System (DNS) lookups from system network calls.

To further process the logs on our Big Data cluster, we have configured DNS deny lists to detect and alert for suspicious behavior. We have enabled Android network logging to record every event from the MobileApp application. It uses the system's network libraries. Network logging records two types of events: DNS lookups and network connections. The logs capture every DNS query that resolves a host name to an IP address. Other supporting DNS queries, such as name server discovery are not logged. The Network Activity Logging API presents each DNS lookup as a DnsEvent instance. Network logging also records an event for each connection attempt that is part of a system network request. The logs capture successful and failed TCP connections, but User Datagram Protocol (UDP) transfers are not recorded. The Network Activity Logging API presents each connection as a ConnectEvent instance. All this network log configuration is complex, but grouped in a specific concrete class named DevicePolicyManager. The configuration is taken into account asynchronously and it is important to validate it before distributing the tablets to students at each software update.

2) Mobile component

The component deployed on the teacher's laptop is a traditional Java component also configured with a message format and a log level. This provides a trace of important events that take place on this workstation. Log analysis is the fastest way to detect what went wrong, which makes logging in Java essential to ensure the performance and health of our distributed application. The goal is to minimize and reduce any downtime, to reduce the mean time to repair.

We used the slf4j library because it represents a simple and highly configurable API. In particular, we have configured the directory where the log messages are saved as well as the expression to generate the file names with the date. The size of the messages is voluntarily limited, so that the subsequent collection is always done within a reasonable time. In addition, the stack trace is provided for any anomaly. Finally, the structure of all logging events follows a pattern consistent with the MobileApp component. We have added a log converter to hide some information such as student IDs. It is important that sensitive information is not traced because this data is then transmitted to our Big Data cluster for analysis.

B. Server application

The server application part is deployed on the storage server. This component, also written in Java, contains the implementation of Web service allowing on the one hand to receive the documents of the students but also to acknowledge the receptions. This part is developed with the Spring Boot library. We use intensively the Spring configuration for the logs, but also for the persistence aspects. The database is Postgresql version 10. As in the previous section, the location of the log files for our server component or for the Postgresql server is imposed. As an example, we record the trace of any http request received by our Web services. The headers are kept as well as the response headers. The version of the http protocol used is http/1.1. In the same way as for the Laptop component, we have imposed a log message format.

C. Big Data architecture

This section focuses on describing our Big Data workflow from collection to building our AI model. We wanted to automate our approach as much as possible because any human intervention leads to blockages or even loss of information.

1) Data collection

This part deals with the collection of log files in order to send them to a Kafka queue. These Kafka files are the entry points of our Big Data cluster. Because there are 3 different types of components, our best choice was to build an event-based collection based on scenario actions. For the MobileApp part, the logs are recorded locally on the device. The sending of the information to a Kafka topic is done when the student sends his final document to the storage server. This approach reduces the number of accesses to the Kafka topic server. Thus, the access point of the lab room has been used to send an http request with an attachment part (the document). This sending is also present in the logs so that the next time only a request is sent, not the same data but only the new ones.

The same approach is used for the laptop component. When the lab session is closed, the logs on the laptop are sent to a Kafka file of the same topic. The message volume is lower, but the information is essential when associating with the logs of the mobile devices.

For the storage server, a repetitive task was our best bid because this central point does not reveal any particular interaction but a continuous flow of data. A cron table was used to collect logs from the Postgresql server and the server component to a Kafka file of the concerned topic. Data are automatically routed to the Kafka server where the topics are managed.

2) Big Data analysis

A Big Data cluster that is built from Hortonworks Data Platform (HDP) 3.0.1 virtual machines is used to perform log analysis. This solution offers the advantage of deploying software from the Hadoop ecosystem while remaining open to other installations. Moreover, the Ambari console allows a simple configuration of servers such as Kafka for topics or Flume for routes. Our software architecture for Big Data is based on two software routes from Kafka topics to the persistence system.

Figure 3 shows the layer components of our project. Deployed on the lab room platforms, we developed Kafka producers for the peripherals, the teaching laptop and the storage server. All these producers issue log messages in a Kafka topic that is partitioned on the server. This improves the access time to the information.

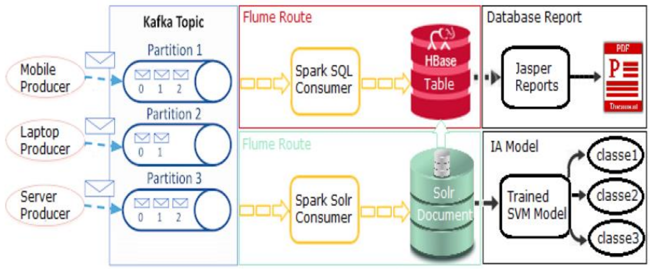


Figure 3. Big Data Software architecture.

The topic partition is the unit of parallelism in Kafka. On both the producer and the broker side, writes to different partitions are done fully in parallel. At the output of the Kafka topics, two Flume routes have been defined within this experiment, each managed by an agent. A first route (red on Figure 3) consumes the messages in order to transform them for some residual format differences and store them in a column-oriented database, HBase, installed on the cluster. A second route (green on figure 3) consumes the messages to index them according to a Solr data schema. Each persistence system has its own role: HBase keeps the log data and Solr keeps the indexes on this data to enrich the searches. We consider HBase and Solr as two data sources accessible from Spark components. The Spark SQL API is easily used to write to HBase column families on a Hadoop cluster. In contrast, our Spark to Solr consumer does not have such an easily accessible API and we used Solr Cloud REST services for our updates.

The data indexed by Solr enables our system to classify the messages in order to carry out maintenance operations on the various materials. A relevant option here was a linear classifier with margin calculation. In fact, in several evaluations of AI models, it is established that in the category of linear classifiers, the Support Vector Machine (SVM) are those that obtain the best results. Another advantage of SVMs, and one that is important to note, is that they are very efficient when there is little training data: while other algorithms would fail to generalize correctly, SVMs are observed to be much more efficient. However, when there is too much data, the SVM tends to decrease in performance.

In order to understand the HBase events and their distribution on the cluster, we have defined a report template to generate a pdf report. It summarizes the activities by table, their events, in particular the use of locks. The use of a template guarantees the scalability of these reports according to the evolutions of the consumer SQL Spark. We added a page header with a table name and the current edition date and a page footer with the page number. The column header band is printed at the beginning of each detail column with the column names in a tabular report. This means the part name of a log message.

3) Log Data storage

A first Spark consumer (named "Spark SQL consumer") has an essential task to recognize and process the contents of the file and load them into an SQL table in memory, perform filter operations and put them in a common format. Then, the route continues with a backup of these data in HBase tables. The role of this Flume route is to store structured information in a column-oriented database (the red route in Figure 3). In this effort, we experimented keeping software routes with Flume for event routing and defined Kafka topics to ensure decorrelation between components. This makes it possible to simplify the management of components, among other things for software updates. In addition, the Kafka API allows more controls on the management of messages associated with a topic; for example time management. We have added rules to ensure that a received message is processed within an hour (from a configuration file). In that case, the system raises an alert and the data saved in the local file system.

A Flume agent is an independent daemon process, which manages the red route. The Flume agent ingests the streaming data from the Kafka topic source to the Spark SQL sink. The channel between the source and the sink is a temporary storage. It receives the events from the Kafka source and buffers them until they are consumed by Spark sinks. It acts as a bridge between the source and the sinks. We have added a Flume interceptor to decide what sort of data should pass through to the channel. It plays first a filter role in case of unsuitable data from the Kafka source and inserts the time in nanosecond into the event header. If the event already contains a timestamp, it will be overwritten with the current time.

We wrote the script for creating tables structured in families of columns to keep the information from the log files. The column families are logical and physical groups of columns. The columns in one family are stored separately from the columns in another family. We assign that data to separate column family when they are not often queried. Because the column families are stored in separate HFiles, we keep the number of column families as small as possible. A HFile is a specific map file implementation for HBase. It contains key/value data. Moreover, one of our objectives was to reduce the number of column families to reduce the frequency of mem-store flushes, and the frequency of compactions. Moreover, by using the smallest number of column families possible, we improve the load time and reduce disk consumption.

Our Spark SQL consumer uses the Spark SQL module to store data in an HBase database whose schema is structured in family of columns. The labels of these families of columns are involved in the data schema of the second Spark consumer. HBase is a database distributed on the nodes of our Hadoop cluster, which allows having a persistence system where the data are highly available because the replicated rate on separate nodes is set to three.

4) Log Data indexing

In parallel, another route has the role of indexing the data from the logs (green route in Figure 3). From the same Kafka source, a second Spark consumer (named "Spark Solr consumer") takes care of data indexing while respecting the Solr schema. The index is updated for the query steps and then the use of a model for the prediction of maintenance tasks. Solr Cloud is the indexing and search engine. It is completely open and allows us to personalize text analyzes. It allows a close link with HBase, database so the schemas used by both tools are designed in a closely related way. On our Big Data cluster, the Solr installation is also distributed. In that context, we have four shards with a replication rate equals to three. This allows us to distribute operations by reducing blockages due to frequent indexing. We have configured, not only the schema, but also the data handlers (schema.xml and solrconfig.xml files).

Our schema defines the structure of the documents that are indexed into Solr. This means the set of fields that they contain, and define the datatype of those fields. It configures also how field types are processed during indexing and querying. This allows us to introduce our own parsing strategy via class programming.

The Spark Solr consumer uses the Spring Data and SolrJ library to index the data read from the Kafka topic. It splits the data next to the Solr schema where the description of each type includes a "docValue" attribute, which is the link to the HBase column family. For each Solr type, our configuration provides a given analyzer. We have developed some of the analyzers in order to keep richer data than simple raw data from log files. Finally, the semantic additions that we add in our analysis are essential for the evaluation of Solr query. Likewise, we store the calculated metrics in HBase for control. SolrCloud is deployed on the cluster through the same Zookeeper agents. Thus, the index persistence system is also replicated. We therefore separate the concepts of backup and search via two distinct components. This reduces the blockages related to frequent updates of our HBase database [13].

At the beginning of our Solr design, we have built our schema based on our data types. Some of them were already defined, but some others are new. In addition, we have implemented new data classes for the new field types. For example, we used RankFieldType as a type of some fields in our schema. It allows us to manage enumeration values from the log message. Then, it becomes a sub class of FieldType in our Solr plugin. We have redesigned Solr filters so that they can be used in our previous setups. Our objective was to standardize the values present in the logs coming from different servers. Indeed, the messages provide information of the form <attribute, value> where the values certainly have units. However, the logs do not always provide the same units for the same attribute calculation. The analysis phase is the place to impose a measurement system in order to be able to compare the results later. The development pattern proposed by SolrJ is simple because it proposes abstract classes like TokenFilter and TokenFilterFactory then to build inherited classes. Then we have to build a plugin for Solr and drop it in the technical directory agreed in the installation of the tool [14].

5) Model factory

In Artificial Intelligence, Support Vector Machine (SVM) models are a set of supervised learning techniques designed to solve discrimination and regression problems. SVMs have been applied to a large number of fields (bioinformatics, information research, computer vision, finance, etc.) [15]. SVM models are classifiers, which are based on two key ideas, which allow to deal with nonlinear discrimination problems, and to reformulate the ranking problem as a quadratic optimization problem. In our project, SVMs can be used to decide to which class of problem a recognized sample belongs. The weight of these classes if linked to the Solr metrics on these names. This amounts to predicting the value of a variable, which corresponds to an anomaly.

All filtered log entries are potentially useful input data if it is possible that there are correlations between informational messages, warnings, and errors. Sometimes the correlation is strong and therefore critical to maximizing the learning rate. We have built a specific component based on Spark MLlib. It supports binary classification with linear SVM. Its linear SVMs algorithm outputs an SVM model [16]. We applied prior processing to the data from our HBase tables before building our decision modeling. These processes are grouped together in a pipeline, which leads to the creation of the SVM model with the configuration of its hyper-parameters such as weightCol. Part of the configuration of these parameters comes from metrics calculated by our indexing engine (Figure 2). Once created and tested, the model goes into action to participate in the prediction of incidents. We use a new version of the SVM model builder based on distributed data augmented. This comes from an article written Nguyen, Le and Phung [17].

6) Report generation

Jasper Report library allows us to build weekly graphical reports on indexing activity. HBase events are collected for display. The goal is to correlate the volumes of data saved in the database with the updates of the AI model. We would like to refine this report template in order to have metrics to decide on the model update. Currently, only HBase movements are represented graphically. Based on an HBase handler, we handle the change events at runtime and send data beans to the Jasper Report Server.

V. DATA STREAMING PART

A. Filtered log strategy

Our component called Spark SQL Consumer contains a Kafka receiver class, which runs an executor as a long-running task. Each receiver is responsible for exactly one input discretized stream (called DStream). In the context of the first

Flume route, this stream connects the Spark streaming to the external Kafka data source for reading input log data.

Because the log data rate is high, our component reads from Kafka in parallel. Kafka stores the data logs in topics, with each topic consisting of a configurable number of partitions. The number of partitions of a topic is an important key for performance considerations as this number is an upper bound on the consumer parallelism. If a topic has N partitions, then our component can only consume this topic with a maximum of N threads in parallel. In our experiment, the Kafka partition number is set to four.

Since log data are collected from a variety of sources, data sets often use different naming conventions for similar informational elements. The Spark SQL Consumer component aims to apply name conventions and a common structure. The ability to correlate the data from different sources is a crucial aspect of log analysis. Using normalization to assign the same terminology to similar aspects can help reduce confusion and error during analysis [17]. This case occurs when log messages contain values with different units or distinct scales. The log files are grouped under topics. We apply transformations depending on the topic the data come from. The filtered logs are cleaned and reorganized and then are ready for an export into an HBase instance.

In the next step, the Spark SQL Consumer component inserts the cleaned log data into memory data frames backed to a schema. We have defined a mapping between HBase and Spark tables, called Table Catalog. There are two main difficulties of this catalog.

a*) The row key definition implies the creation of a specific key generator in our component.*

b) The mapping between table column in Spark and the column family and column qualifier in HBase needs a declarative name convention.

The HBase sink exploits the parallelism on the set of Region servers, which are under control of the HBase master. The HBase sink treats both Put operation and Delete operation in a similar way, and both actions are performed in the executors. The driver Spark generates tasks per region. The tasks are sent to the preferred executors collocated with the region server, and are performed in parallel in the executors to achieve better data locality and concurrency. By the end of an exportation, a timed window a log data are stored into HBase tables.

B. *Index construction and query*

The strategy of the Spark Solr Consumer component deals with the ingestion of the log data into Apache Solr for search and query. The pipeline is built with Apache Spark and Apache Spark Solr connector. Spark framework is used for distributed in memory compute, transform and ingest to build the pipeline.

The Apache HBase role is the log storage and the Apache Solr role is the log indexing. Both are configured in cloud mode Multiple Solr servers are easily scaled up by increasing server nodes. The Apache Solr collection, which plays the role of a SQL table, is configured with shards. The definition of shard is based on the number of partitions and the replicas rate for fault tolerance ability. The Spark executors run a task, which transforms and enriches each log message (format detection). Then, the Solr client takes the control and send a REST request to Solr Cloud Engine. Finally, depending on the Solr leader, a shard is updated.

We use also Solr Cloud as a data source Spark when we create our ML model. We send requests from Spark ML classes and read results from Solr (with the use of Solr Resilient Distributed Dataset (SolrRDD class). The pre statement of the requests is different from the analysis of the log document. Their configuration follows another analysis process. With Spark SQL, we expose the results as SQL tables in the Spark session. These data frames are the base of our ML model construction. The metrics called Term Factor (TF) and Inverse Document Frequency (IDF) are key features for the ML model. We have also used boost factor for customizing the weight of part of the log message.

VI. RESULTS AND TASK MAINTENANCE

We have several kinds of results. A part is about our architecture and the capacity to treat log messages over time. Another part is about the classification of log messages. The concepts behind SVM algorithm are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. In our working context, the margin between log patterns is a suitable discriminant.

A. *Data features*

For our tests, we used previously saved log files from a month of application server and database server operations. We were interested in the performance of the two Spark consumers: For Spark SQL Consumer, the volume of data to analyze is 102.9 M rows in HBase. To exploit this data, we used a cluster of eight nodes on which we deployed Spark and HBase. The duration of the tests varies between 32 minutes and 3 hours and 30 minutes.

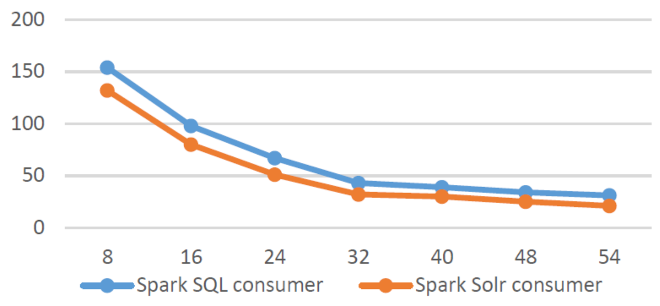


Figure 4. Spark consumer runtime versus number of partitions.

For Spark Solr Consumer, the volume of data indexed is 100.5M rows indexed in about an hour. The number of documents indexed per second is 34k. We only installed Solr on four nodes with four shards and a replication rate of three. We have seen improved results by increasing the number of Spark partitions (RangePartitioner). At runtime for our data set based on a unique log format, the cost of Spark SQL consumer decreases when the partitioning of the dataset increases, an illustrated in Figure 4. The X-axis represents the partition number and the Y-axis represents the time consuming. We have to oversize the partitions and the gains are much less interesting.

SVM offers very high accuracy compared to other classifiers such as logistic regression, and trees. There are several modes of assessment. The first is technical; it is obtained thanks to the framework used for the development (Spark MLlib). The second is more empirical because it relates to the use of this model and the anomaly detection rate on a known dataset. The analytical expression of the features precision, recall of retrieved log messages that are relevant to the find: Precision (1) is the fraction of retrieved log messages that are relevant to the find:

$$precision = \frac{|\{relevant \log messages\} \cap \{retrieved\ log\ messages\}|}{|\{retrieved \log messages\}|} \quad (1)$$

Recall (2) is the fraction of log messages that are relevant to the query that are successfully retrieved:

$$recall = \frac{|\{relevant \log messages\} \cap \{retrieved\ log\ messages\}|}{|\{relevant \log messages\}|} \quad (2)$$

$$F_\beta = (1+\beta^2) * \frac{precision*recall}{(\beta^2*precision)+recall} \quad (3)$$

In Table 1, we have four classes and for each class we compute three metrics: true positive (tp), false positive (fp) and false negative (fn). For instance, for the third class, we note these numbers tp3, fp3 and fn3. From these values, we compute precision by label, recall by label and F-score by label.

TABLE I. SVM MODEL MEASURES

Class number	Metrics		
	Precision by label	***Recall by label***	***F1 score by label***
0.000000	0.815846	0.890100	0.896616
1.000000	0.911000	0.981000	0.991000
2.000000	0.854461	0.785714	0.851481
3.000000	0.852446	0.7589148	0.833129

Our prediction models are similar to a multiclass classification. We have several possible anomaly classes or labels, and the concept of label-based metrics is useful in our case. Precision is the measure of accuracy on all labels. This is the number of times a class of anomaly has been correctly predicted (true positives) normalized by the number of data points. Label precision takes into account only one class and measures the number of times a specific label has been predicted correctly normalized by the number of times that label appears in the output. The last observations are:

- Weighted precision = 0.901742
- Weighted recall = 0.931803
- Weighted F1 score = 0.981731
- Weighted false positive rate = 0.040009

Our results for four classes are within acceptable ranges of values for the use of the model to be accepted.

The test empirical phase on the SVM model was not extensive enough to be conclusive. However, our results suggest that increasing the number of log patterns deteriorates the performance. In addition, we defined a finite set of log patterns for a targeted anomaly detection approach.

B. *Reporting*

We have created a custom data source to connect to Apache Solr, therefore we are able to retrieve data and provide them back in following the JRDataSource interface of Jasper Report. With this access point, we have extracted metrics about the document cache and Query result cache. Both give an overview of the Solr activities and is meaningful for the analysts. We have deployed the CData JDBC Driver on Jasper Reports to provide real-time HBase data access from reports. We have found that running the underlying query and getting the data to our report takes the most time. When we generate many pages per report, there is overhead to send that to the browser.

For the reporting phase, we have developed two report templates based on the use of a JDBC adapter. With system requests, we collect data about the last events (Get, Put, Scan, and Delete). From these HBase view, we have designed the report templates with cross tables. For the storage phase, we compute and display the number of Put events per timed window or grouped over a period. We periodically updated the data across report runs. We export the PDF files to the output repository where a web server manages them.

VII. CONCLUSION AND FUTURE WORK

We have presented our approach on log analysis and maintenance task prediction. We showed how an index engine is crucial for a suitable query engine. We have developed specific plugin for customizing the field types of our documents, but also for filtering the information from the log message. Because indexing and storage are the two sides of our study, we have separated the storage into a Hadoop database. We have stressed the key role of our Spark components, one per data source. The partition management is the key concept for improving the performance of the Spark SQL component. The data storage into data frames during the micro batches is particularly suitable for the management of flows originating from Kafka files. We observed that our approach supported a large volume of logs.

From the filtered logs, we presented the construction of our SVM model based on work from the Center for Pattern Recognition and Data Analytics, Deakin University, (Australia). We were thus able to classify the recognized log patterns into classes of anomalies. This means that we can identify the associated maintenance operations. Finally, to measure the impact of our distributed analysis system, we wanted to build automatically reports based on templates and highlight indexing and storage activity.

Our study also shows the limits that we want to push back, such as the management of log patterns. The use of an AI model is not the guarantee of an optimal result. We want to make more use of indexing metrics to give more weight to some information in the analyzed logs. We are, therefore, thinking of improving the classification model of log data.

A first perspective will be to improve the indexing process based on a custom schema. We think that the use of DisMax query parser could be more suitable in log requests where messages are simple structured sentences. The similarity detection makes DisMax the appropriate query parser for short structured messages.

The log format has a deep impact on the Solr schema definition and on the anomaly detection. We are going to evolve our approach. In the future, we want to extract dynamically the log format instead of the use of a static definition. We think also about malicious messages, which can perturb the indexing process and introduce bad requests in our prediction step. The challenge is to manage a set of malicious patterns and the quarantine of some message sequences.

References

[1] A. Oliner, A. Ganapathi, and W. Xu, “Advances and challenges in log analysis,” Communications of the ACM, 2012, 2nd ed., vol. 55, pp. 55-61.

[2] T. F. Yen et al., “Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks,” In Proceedings of the 29th Annual Computer Security Applications Conference, pp. 199-208, December 2013.

[3] S. He, J. Zhu, P. He, and M. R. Lyu, “Experience report: System log analysis for anomaly detection,” In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 207-218, October 2016.

[4] B. Debnath et al., “Loglens: A real-time log analysis system,” In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 1052-1062, July 2018.

[5] Q. Cao, Y. Qiao, and Z. Lyu, “Machine learning to detect anomalies in web log analysis,” In 2017 3rd IEEE International Conference on Computer and Communications (ICCC), pp. 519-523, December 2017.

[6] W. Li, “Automatic log analysis using machine learning: awesome automatic log analysis version 2.0.,” 3 edition, November 2013.

[7] A. Juvonen, T. Sipola, and T. Hämäläinen, “Online anomaly detection using dimensionality reduction techniques for HTTP log analysis,” Computer Networks 91, pp. 46-56, November 2015.

[8] J. Dongo, C. Mahmoudi, and F. Mourlin, “Ndn log analysis using big data techniques: Nfd performance assessment,” In 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), pp. 169-175, March 2018.

[9] W. Luo et al., “Guidelines for developing and reporting machine learning predictive models in biomedical research: a multidisciplinary view,” Journal of medical Internet research, 12 ed., vol.18, 2016.

[10] P. Henderson et al., “Towards the systematic reporting of the energy and carbon footprints of machine learning,” Journal of Machine Learning Research, 2020, 248 ed., vol. 21, pp. 1-43.

[11] L. M. Stevens, B. J. Mortazavi, R. C. Deo, L. Curtis, and D. P. Kao, “Recommendations for reporting machine learning analyses in clinical research. Circulation: Cardiovascular Quality and Outcomes,” 2020, 10 ed., vol. 13.

[12] D. P. Dos Santos and B. Baeßler, “Big data, artificial intelligence, and structured reporting,” European radiology experimental, 2018, 1rd ed., vol. 2, pp. 1-5.

[13] K. Koitzsch, “Advanced Search Techniques with Hadoop, Lucene, and Solr,” Pro Hadoop Data Analytics, Apress, Berkeley, CA, 2017, pp. 91-136.

[14] J. Kumar, “Apache Solr search patterns,” Packt Publishing Ltd, 2015.

[15] M. F. Ghalwash, D. Ramljak, and Z. Obradović, “Early classification of multivariate time series using a hybrid HMM/SVM model,” 2012 IEEE International Conference on Bioinformatics and Biomedicine, IEEE, pp. 1-6, 2012.

[16] M. Assefi, E. Behravesh, G. Liu, and A. P. Tafti, “Big data machine learning using apache Spark MLlib,” 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 3492-3498.

[17] T. D. Nguyen, V. Nguyen, T. Le, and D. Phung, “Distributed data augmented support vector machine on Spark,” 23rd International Conference on Pattern Recognition (ICPR), 2016, IEEE.