Original Scientific Paper
Received: 2022-03-08
Accepted: 2022-04-20

STREAMING DATA CLASSIFICATION USING NAIVE BAYES ALGORITHM VS LOGISTIC REGRESSION

Belma JAKUPOVIĆ, RIT Croatia, bxj4159@g.rit.edu

Abstract

With the increase in popularity of Big Data and machine learning algorithms, gathering and processing large amounts of available data became common practice. Usually, while developing machine learning algorithms data would be gathered first and then machine learning models would use that data for prediction or classification. More often than not, analysis or prediction of data is needed in real-time. The paper demonstrates a data streaming of tweets and their sentiment classification in real-time. Essentially, the text will be classified as negative or positive, as soon as it arrives in the system. The data that will be used for the classification model will be a new incoming tweet in combination with the data of the old already labeled tweets. New tweets will be arriving into the system via the Kafka framework. In this paper, we will focus on evaluating the model that is being built upon streaming data rather than the sentiment analysis itself. We will write about the challenges of evaluating a model while data is streaming, which metrics are used and why we chose such solution. This paper proposes a system solution for problems with streaming data and it will use a machine learning models that are being updated given continuous input of data in real-time. We will present the similarities as well as differences between Naïve Bayes algorithm and Logistic regression.

Keywords: classification, Twitter, sentiment analysis, data streaming, Kafka

1. INTRODUCTION

With the rapid development of technology and its usage in a wide specter of businesses, there is a large fluctuation of data. Many digital solutions rely on the input or output data of the users, especially in real-time, for example predicting air quality [2]. Lately, it became very important to provide almost instant results given the data users provide via digital systems. There are many examples where a prediction is needed as soon as the data arrives: prediction of covid positive cases or hospitalizations or rather less negative example is sentiment prediction in social media. Machine learning are very useful in order to prepare the hospital for new patients, fraud detection with credit cards, anomaly detection, or in the case of sentiment analysis to prevent hate speech from appearing on a platform.

In literature there are several names indicating data streaming, like online streaming or learning, incremental learning, real-time learning. And although they all might sound similar; they have some

fundamental differences. Putatunda, in his book, dedicates a chapter to explain the differences between incremental and online learning. As the author explains, incremental learning algorithms work with batches of predetermined size, and update the model parameters as soon as new batch of data arrives. While with online learning algorithms parameters are updated as soon as new observation arrives [14]. By Putatunda's definition, unlike incremental, online learning algorithm actually allows us real-time classification.

In this paper, we developed a system that will handle streaming of the Twitter data with Kafka streams, create the machine learning classification model for continuous streaming data into positive and negative text, evaluate the model and perform sentiment classification for each incoming tweet. There are several steps involved in developing such a system. Although data preparation is a crucial part of developing machine learning algorithms, in this case, we have to take special care of our data. Given that we are handling the classification of textual data, in order to be properly classified, data has to be clean since machine learning algorithms cannot process textual data in a raw format. After we clean the data, we fit the model and calculate the scores for accuracy, precision, and recall. In this section, we have to also keep track of the model performance, in case the model needs refitting. We classify the incoming tweet and store it with the rest of the already labeled tweets. For the classification of the text, we used the Gaussian Naïve Bayes algorithm and Logistic regression.

2. MOTIVATION AND PREVIOUS WORK

Given the amount of data that is fluctuating over the internet and many digital systems, it is surpassing businesses, and entering the everyday life of ordinary people. From suggesting traffic routes based on traffic jams to learning people's shopping habits to advance sales. Data has great power. Real-time detection, prevention, or prediction can be very useful in everyday business. Its applicability can be seen in various branches of businesses like credit card fraud detection, speech to text and text to speech conversion, real-time transcription or translation, anomaly detection in pictures, sound, or just plain numbers, as well as classification issues in spam email detection, and sentiment analysis of the text.

This work is mainly focusing on the system handling data streams and performing a classification as the data comes into a system. There are many libraries that can be used to load and consume data streams in Python: Crème, scikit-learn [17], scikit-multiflow[1], PySpark[2], and probably many more. For this system solution to be applicable not only for Python projects but for any language as well, we have used Kafka framework that will perform streaming the data into the system.

Although Twitter sentiment analysis is not a new topic and has been extensively worked on for many years, it served as a good example where almost immediate classification is possible. There are a lot of research papers on this topic, like the works of Wongkar and Angdresey, Muthuswamy, Ranjan et al., Goel et al, but not all of them talk about streaming the data itself, or model performance tracking

[1] https://scikit-multiflow.github.io/

[2] https://spark.apache.org/docs/latest/api/python/#:~:text=PySpark%20is%20an%20interface%20for,data%20in%20a%20distributed%20environment.

which is useful for these kinds of problems [19][16][12][6]. The majority of them use Naïve Bayes algorithms, as well as Support Vector Machines, Logistic Regression, and Neural Networks [19][6][11]. Many of the papers mentioned show extensive explanation and results of the performance of these algorithms and their models, however, we have focused on tracking the model during the data stream, and detecting when does the model needs to be refitted.

A. Challenges of data streaming

Given the nature of the data that is produced in real world, and as such streamed to the systems, data pre-processing can be a challenging task. Some trivial pre-processing, like removing noisy data, ignoring null values, etc., that are typically done for machine learning with static data, can be complicated. And this is still unknown territory, which has been not explored in detail [7]. However, we still have to deal with invalid data, feature selection and scaling, only in real-time, as the data arrives. Some other challenges mentioned in the literature are problems with system memory and speed, as well as the dynamics of data processing [15]. It is not possible to go through data more than once.

B. Model evaluation

Another problem that arises when we start training a model on streaming data is to determine whether a model needs to be refitted. Depending on how we are gathering the data, data streams can manifest a behavior called concept drift. We have to be aware of this issue while mining the data, so we can adequately update the machine learning model [15].

Concept drift is considered as a sudden change in the data's mean, variance. It is important to keep track of these changes, because they are the indicators for many problems we are trying to solve. Several algorithms have been invented to detect the concept drift in data: Drift Detection Method (DDM), and Early Drift Detection Method (EDDM) for example [15]. These algorithms for drift detection can be applied to any learning algorithm, whether it is for online or static learning. Main idea behind these drift detection algorithms is to detect the learning algorithm's error rate [15].

From these problems evolved ensemble learning, whose reactive strategies can cope with the concept of drift. Gomes et al. (2019) in their paper talk about how reactive strategies help with model updates, "often assigning different weights to base models according to their prediction performance" [7]. Another use case for ensemble learning is addressing feature drift, especially in classification. It helps with identification of feature influence, meaning that if considered irrelevant its classifier that is associated with that feature could be removed [7]. Given all information gathered, we can conclude that there are a lot of methods and algorithms that can detect change in data. This is a very important step in machine learning from streaming data, as the data can be unpredictable. Furthermore, observation of data is crucial in pinpointing the moment when the model should be retrained.

3. DATA PREPARATION

Dataset used in this project is taken from Kaggle[3], containing 31962 rows. There are three columns: id, sentiment label, and text. Very simple, and straightforward. But we can't just use it as it is, some transformation is needed.

Table 1 Example of the data

ID	SENTIMENT LABEL	TEXT
1	0	@user when a father is dysfunctional and is so selfish he drags his kids into his dysfunction. #run
2	0	@user @user thanks for #lyft credit i can't use cause they don't offer wheelchair vans in pdx. #disapointed #getthanked
3	0	bihday your majesty
4	0	#model i love u take with u all the time in urð ±!!! ð ð ð ð ð ¦ð ¦ð ¦
5	0	factsguide: society now #motivation

The machine learning models do not accept textual data in a raw format. Textual data used for this project is even more specific. Since we are using tweets, the text of a tweet can contain many special characters, numbers and be full of spelling mistakes or just shortened words. As seen in Table 1, Twitter data in textual format can contain a lot of special character, and sometimes can consist of only special characters or even emojis. Apart from textual data being very tricky to work with, Twitter data does not even have to be meaningful, can include a lot of slang vocabulary, which is challenging when one tries to classify that text. In order to perform a successful sentiment analysis of this kind of text, we have to polish our data using the following steps:

1. Transform text to lowercase
2. Tokenize sentences (split the sentence in the list of words)
3. Remove numbers
4. Remove punctuation
5. Remove stop words (is, a, the, an, etc…)
6. Normalization of words

The last step mentioned is needed because in the English language, words can take many forms

[3] https://www.kaggle.com/arkhoshghalb/twitter-sentiment-analysis-hatred-speech?select=train.csv

depending on the usage. Therefore, when tokenized (separated from the sentence), the word can mean something different than intended within the sentence. Even though we know they share a meaning, the system doesn't. There are two ways to normalize words: stemming and lemmatization. In this project, we have used lemmatization. This technique is much better than stemming and uses vocabulary and morphological analysis, which results in returning the dictionary form of a word. Because of this, the words contain more meaning, which helps with analysis. For lemmatization, we have used Natural Language Toolkit Python package which enables many libraries for natural language processing.

After performing these steps, data from the Table 1, will be free of punctuation, numbers, stop words, leaving only textual characters to work with. The text is then split into a list of words, and forwarded to further processing. As mentioned in the beginning of this section, since we cannot feed raw textual data to machine learning algorithm, we must convert these characters into something machine learning algorithms can interpret: numbers. For this step of preprocessing of the data we have used Count Vectorizer[4] from sklearn's Feature Extraction library. Count Vectorizer is able to convert a textual data to a vector of term counts. As Count Vectorizer's analyzer we have provided a method that involve all the steps described above, which gives our data highly flexible feature representation. This process helps us to identify and give a value to a sentence, in order to be able to classify it properly.

4. SYSTEM ARCHITECTURE

New tweets are being sent into the system via Kafka producer, with the rate of 1 tweet per second. As soon the new tweet enters the system, Kafka consumer will initiate classification. Since we have a dataset of already classified tweets, our model is fitted and trained before we do a prediction of an incoming tweet. Once a prediction has been made, we save the classified tweet into the existing dataset.

However, new data will be stored in our dataset, and the original model might not perform the same as with the new data we classified. Model evaluation is very important while streaming, so we don't start classifying incorrectly. Although, the accuracy of a model might be good (80%+), we can still track its performance via other metrics. For evaluation of our models, we decided to use Welford's algorithm for computing the running variance. While calculating variance we deal with floating-point arithmetic, and it is therefore important that we use algorithms that are numerically stable, and with which we avoid overflows. The good thing about Welford's algorithm is that we can feed it a list of values, and it will compute its standard deviation. Based on standard deviation, it is decided whether the model needs refitting.

4https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

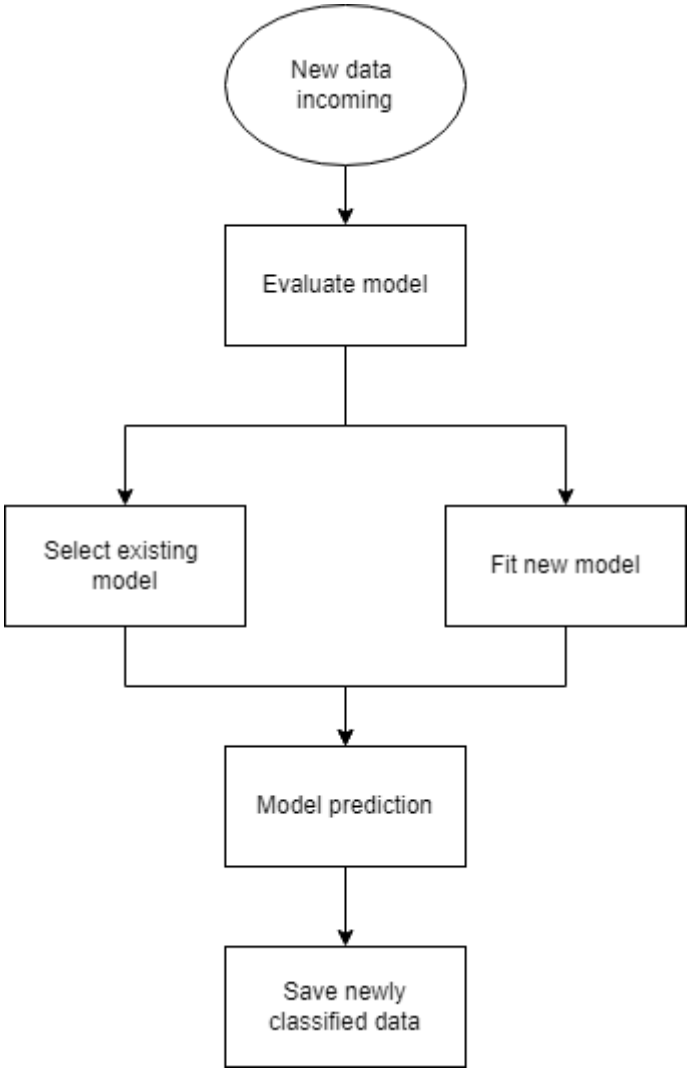

Figure 1 System Architecture

A. Welford's Algorithm[5]

Standard deviation plays an important role in the measurements of the dataset. It tells us how dispersed the data is. This number can help us understand the data that we have as well as the dataset including newly classified tweets. Higher values of standard deviation will help us notice the extreme values and how frequent they are. Since we are calculating a variance of a list of tweets, Welford's algorithm is a very simple and stable algorithm for calculating a running variance [3]. This algorithm provides numerical stability as well as speed. Since we are working with floating point numbers, we are bound to lose some precision while subtracting a small number from a larger number. Most importantly this algorithm allows computation of the mean, median, and variance in real-time. This algorithm is not widely used, as other mentioned algorithms in previous sections (DDM, EDDM), but it can be useful for smaller scopes.

[5] https://www.embeddedrelated.com/showarticle/785.php

5. CLASSIFICATION

Classification is a process of data categorization into classes, based on the available information that can determine the sentiment of the data. For this project we classify the text into two classes: positive and negative tweets. For the classification of the tweets, we used Gaussian Naïve Bayes algorithm and Logistic regression algorithm. We created an initial model out of the data containing the tweets that have been already classified and labeled as positive or negative. Dataset consists of three columns: id, text, and sentiment. Since we can only use one feature, which is the text of the tweet, for training the model we clean textual data following the steps explained in the data preparation section before we can train and fit our models.

A. Gaussian Naïve Bayes Algorithm

One of the machine learning algorithms we have used for classification is Gaussian Naïve Bayes algorithm. Naïve Bayes algorithm, as a probabilistic machine learning algorithm, is based on Bayes Theorem. Even though this is a simple algorithm it is very powerful for many classification problems. Most importantly, predictions made with this algorithm are almost instant. Overall, according to scikit-learn documentation, Naïve Bayes methods work with the "assumption of conditional independence between every pair of features given the value of the class variable" [16]. There are different variations of Naïve Bayes algorithms, but all of them work exceptionally well for classification problems. They are fast, and don't need a lot of data for training the model [16].

Equation 1 GaussianNB likelihood of features

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

B. Logistic regression

Logistic regression is a statistical analysis, that is used for prediction of values. In this case, the dependent variable is categorical, making this a binary or multinomial regression. Hence, it can be used in classification problems. Although, logistic regression is mainly used for classification, some other models can have better predictive performance. This is because with logistic regression complete separation can happen. That means that when the feature that perfectly separates two classes, logistic regression model cannot be trained anymore. This happens mainly because logistic regression weights the features, and this perfect feature so to say will have infinite optimal weight. However, for these reasons logistic regression can also be used for determining probabilities [17].

Equation 2 Logistic function

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$

6. RESULTS

Training of the models has been done on 564 tweets, which are already correctly labeled as positive or negative tweets. To evaluate the model, several metrics have been computed: accuracy, precision, recall and F1 score. The data has been split into train and test sections, where 20% of the data has been used for testing the fitted model. Data has been randomly assigned to either test or train data each time. We have trained the models four times, and below is the table showing the results:

Table 2 Model evaluation

#	Gaussian Naïve Bayes model		Logistic regression	
1	Recall score	0.858	Recall score	0.946
	Accuracy score	0.858	Accuracy score	0.946
	Precision score	0.858	Precision score	0.946
	F1 score	0.858	F1 score	0.946
2	Recall score	0.893	Recall score	0.876
	Accuracy score	0.893	Accuracy score	0.876
	Precision score	0.893	Precision score	0.876
	F1 score	0.893	F1 score	0.876
3	Recall score	0.893	Recall score	0.876
	Accuracy score	0.893	Accuracy score	0.876
	Precision score	0.893	Precision score	0.876
	F1 score	0.893	F1 score	0.876
4	Recall score	0.867	Recall score	0.946
	Accuracy score	0.867	Accuracy score	0.946
	Precision score	0.867	Precision score	0.946
	F1 score	0.867	F1 score	0.946

Both models have the accuracy above 85%, although logistic regression two times out of four had accuracies above 90%. Even though logistic regression was somewhat more accurate, they performed almost equally well. The dataset itself has small number of tweets labeled as negative, so few of negative incoming tweets through data stream has been correctly labeled by both models. Of course, there have been several tweets that have been wrongly labeled, but that is expected. To support my claim, we have also computed confusion matrixes for two models, for four times as well. From the graphs below we can see that majority of the data has been correctly labeled, with few

mistakes.

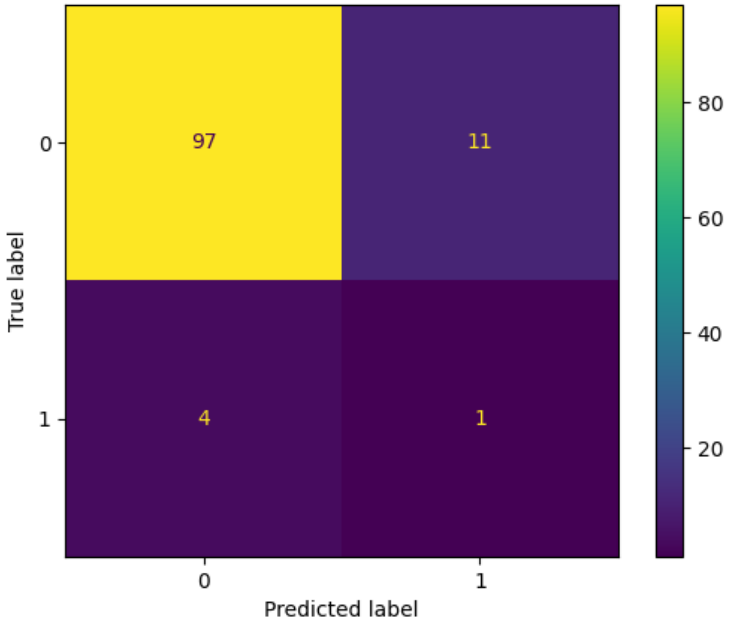

Figure 2 Gaussian NB confusion matrix

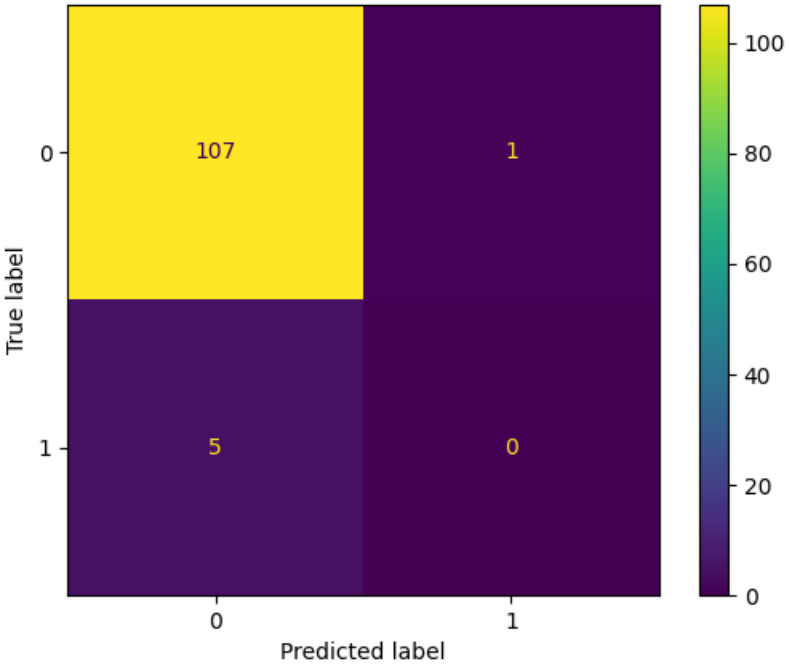

Figure 3 Logistic Regression confusion matrix

Pictures show the best outcome, out of four models trained for each machine learning algorithm.

Below we have a complete table showing the results of confusion matrix for all iterations of each model.

Table 3 Confusion matrixess

#	Gaussian Naïve Bayes model		Logistic regression model	
1	True negative	94	True negative	107
	False positive	4	False positive	1
	False negative	8	False negative	5
	True positive	7	True positive	0
	True positive rate	0.4667	True positive rate	0.0
	True negative rate	0.9592	True negative rate	0.9907
2	True negative	94	True negative	98
	False positive	4	False positive	0
	False negative	8	False negative	14
	True positive	7	True positive	1
	True positive rate	0.4667	True positive rate	0.0667
	True negative rate	0.9592	True negative rate	1.0
3	True negative	94	True negative	99
	False positive	4	False positive	0
	False negative	8	False negative	14
	True positive	7	True positive	0
	True positive rate	0.4667	True positive rate	0.0
	True negative rate	0.9592	True negative rate	1.0
4	True negative	97	True negative	107
	False positive	11	False positive	1
	False negative	4	False negative	5

	True positive	1	True positive	0
	True positive rate	0.2	True positive rate	0.0
	True negative rate	0.8981	True negative rate	0.9907

CONCLUSION

This paper provided an example of a system architecture that consumes data streams and performs an analysis of the data in real-time. Various technologies have been used to build the architecture, so this can serve as an example of a solution for different problems including streams of data, and not necessarily for strictly python projects. The goal was to provide a solution with technologies that are available for integration with other programming languages. Aside from the technologies used, this paper provided an insight into a data streaming, what does it mean and how to handle it in real-time. As mentioned in data preparation section, we have explained why it is difficult to work only textual data as features. A lot of preprocessing is needed, so we can make some conclusions out the text. We have explored two machine learning algorithms that are used for classification of the data: Gaussian Naïve Bayes and Logistic regression. Both have a good performance; they are fast and easy to implement. The results showed that majority of the data will be 85%, or more, accurately labeled. Furthermore, the data is unpredictable in real world, and detection of drifts in the incoming data is very important. This could be a starting point into widening this topic, as this deserves a more in-depth analysis, especially dealing with evolving data such as text, images, or graphs.

References

[1] Al shammari, A. S. (n.d.). *Real-time twitter sentiment analysis using 3-way classifier* [Paper presentation]. 2018 21st Saudi Computer Society National Computer Conference (NCC).

[2] Ameer, S., Shah, M. A., Khan, A., Song, H., Maple, C., Islam, S. U., & Asghar, M. N. (2019). Comparative analysis of machine learning techniques for predicting air quality in smart cities. *IEEE Access*, *7*, 128325-128338. https://doi.org/10.1109/ACCESS.2019.2925082

[3] Chen, C. (n.d.). Welford algorithm for updating variance. https://changyaochen.github.io/welford/

[4] Efanov, A. A., Ivliev, S. A., & Shagraev, A. G. (n.d.). *Welford's algorithm for weighted statistics* [Paper presentation]. 2021 3rd International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE).

[5] Gama, J., Sebastião, R., & Rodrigues, P. P. (2012). On evaluating stream learning algorithms. *Machine Learning*, *90*(3), 317-346. https://doi.org/10.1007/s10994-012-5320-9

[6] Goel, A., Gautam, J., & Kumar, S. (n.d.). *Real time sentiment analysis of tweets using naive bayes* [Paper presentation]. 2016 2nd International Conference on Next Generation

Computing Technologies (NGCT).

[7] Gomes, H. M., Read, J., Bifet, A., Barddal, J. P., & Gama, J. (2019). Machine learning for streaming data. *ACM SIGKDD Explorations Newsletter*, *21*(2), 6-22. http://dx.doi.org/10.1145/3373464.3373470

[8] Gupta, B., Negi, M., Vishwakarma, K., Rawat, G., & Badhani, P. (2017). Study of twitter sentiment analysis using machine learning algorithms on python. *International Journal of Computer Applications*, *165*(9), 29-34. http://dx.doi.org/10.5120/ijca2017914022

[9] Jakob. (2021, May 6). Calculating Variance: Welford's Algorithm vs NumPy. https://natural-blogarithm.com/post/variance-welford-vs-numpy/

[10] Molnar, C. (2019). *Interpretable machine learning: A guide for making black box models explainable*. Leanpub.

[11] Munir, M., Chattha, M. A., Dengel, A., & Ahmed, S. (2019, December). *A comparative analysis of traditional and deep learning-based anomaly detection methods for streaming data* [Paper presentation]. 2019 18th IEEE international conference on machine learning and applications (ICMLA), Boca Raton, FL, United States.

[12] Muthuswamy, S. (2018, February). *Sentiment Analysis on Twitter Data Using Machine Learning Algorithms in Python* [Paper presentation]. International Conference on Advances in Computing Applications(ICACA-18)At: NIT Uttarakhand. https://www.researchgate.net/publication/327160507_Sentiment_Analysis_on_Twitter_Data_Using_Machine_Learning_Algorithms_in_Python

[13] Nadungodage, C. H., Xia, Y., Li, F., Lee, J. J., & Ge, J. (2011). StreamFitter: A real time linear regression analysis system for continuous data streams. *Database Systems for Advanced Applications*, 458-461. http://dx.doi.org/10.1007/978-3-642-20152-3_39

[14] Nair, L. R., Shetty, S. D., & Shetty, S. D. (2018). Applying spark based machine learning model on streaming big data for health status prediction. *Computers & Electrical Engineering*, *65*, 393-399. http://dx.doi.org/10.1016/j.compeleceng.2017.03.009

[15] Putatunda, S. (2021). *Practical machine learning for streaming data with Python: Design, develop, and validate online learning models*. Springer. https://doi.org/10.1007/978-1-4842-6867-4

[16] Ranjan, S., Sood, S., & Verma, V. (n.d.). *Twitter sentiment analysis of real-time customer experience feedback for predicting growth of indian telecom companies* [Paper presentation]. 2018 4th International Conference on Computing Sciences (ICCS).

[17] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[18] Singla, A., & Jangir, H. (2020, February 21). *A comparative approach to predictive analytics with machine learning for fraud detection of realtime financial data* [Paper presentation]. 2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3), Lakshmangarh, India.

[19] Wongkar, M., & Angdresey, A. (n.d.). *Sentiment analysis using naive bayes algorithm of the*

data crawler: Twitter [Paper presentation]. 2019 Fourth International Conference on Informatics and Computing (ICIC). https://ieeexplore.ieee.org/document/8985884