

An intelligent artificial intelligence framework for business insight generation from streaming data

Benymol Jose¹, Sajimon Abraham², Nikitha Sajith³

¹Department of Computer Applications, Marian College Kuttikkanam Autonomous, Kuttikkanam, India

²School of Management and Business Studies, Mahatma Gandhi University, Kottayam, India

³LTM Limited, Bengaluru, India

Article Info

Article history:

Received Dec 15, 2025

Revised Aug 1, 2026

Accepted Aug 12, 2026

Keywords:

Champion model

Gini coefficient

Machine learning pipeline

Natural language processing techniques

Receiver operating characteristic curve

Streaming data

ABSTRACT

Big data analytics helps organizations leverage their data to uncover new opportunities, leading to smarter decisions, more efficient operations, and ultimately higher profits. Effective use of big data and creating important predictive models make it more straightforward for a business to realize what the consumer likes and what he or she does not. This will enable the business to propose better offers and administrations with their items when contrasted to its competitors. In this paper, the authors suggest the use of a big data analytics platform combined with natural language processing (NLP) techniques to do the intelligent processing of unstructured streaming data. A new method utilizing Apache Spark pipelining is proposed to achieve predictive analytics for unstructured data streams. With this predictive model, mining of the unstructured streaming data can be done and intelligent predictions are made possible in real time using artificial intelligence techniques.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Benymol Jose

Department of Computer Applications, Marian College Kuttikkanam Autonomous

Kuttikkanam P.O, Peermade, Idukki District, Kerala, India

Email: benymol.jose@mariancollege.org

1. INTRODUCTION

Stream computing represents a modern paradigm driven by emerging data-generation scenarios such as the internet of things (IoT), sensor networks, financial market feeds, mobile devices, geospatial services, and social media platforms [1]. This approach is based on the key premise that the value of data is highest when it is most recent [2]. Data produced continuously is known as streaming data, and it must be analyzed immediately as it arrives to generate timely insights. To handle such processing demands, robust parallel architectures, and scalable computing systems are essential [3]. Stream computing enables ongoing evaluation of static queries against data that is constantly in motion [4]. Although vast amounts of unstructured data are produced, extracting relevant and meaningful information from these streams is critical for making prompt decisions. Such timely decision-making is possible only when incoming data is processed, analyzed, and visualized in real time [5].

Demirezen and Navruz [6] suggest employing the lambda (λ) architecture, built on top of hadoop, to support near real-time data processing that accommodates both streaming and batch analytics. This architecture comprises three primary layers known as the batch layer, the speed layer, and the serving layer. The batch layer generates precomputed results using a distributed processing framework and ensures high accuracy by handling the full dataset [7]. The speed layer manages incoming data streams in real time, compensating for the latency inherent in the batch layer. Because it focuses on rapid output, its results may be less precise or incomplete compared to those produced by the batch layer, and these fast outputs are

frequently stored in not only structured query language (NoSQL) systems [8], such as MongoDB, VoltDB, and Cassandra. The results from the batch and speed layers are kept in the serving layer, which answers queries by returning pre-computed interpretations from the processed data [9], and reprocessing every batch cycle is not beneficial in many scenarios [10].

Apache Spark is a robust open-source engine capable of handling both batch and stream processing tasks [10]. In their recent study, Kamal *et al.* [11] employ Apache Spark for real-time data processing combined with text mining and natural language processing (NLP). Their work aims to illustrate how business intelligence can be enhanced by gathering and analyzing web data in real time. Another investigation by Svyatkovskiy *et al.* [12] assesses the Apache Spark framework for data-intensive machine learning (ML) workloads. They propose a text-processing pipeline that incorporates the Avro serialization framework, Spark ML, GraphFrames, and the histogrammar suite of data aggregation tools, applying it to a policy diffusion problem. In a survey conducted by Ahmed *et al.* [13], the authors discuss the idea of big data and the numerous forms of analytics that researchers can perform using large-scale data. Their work outlines strategies for improving the performance of big data processing and explains different big data analytics tools. In another approach, Azhir *et al.* [14] investigate big data analysis and highlight the importance of Apache Spark in real-time processing by extracting meaningful insights from X (formerly Twitter) data.

The work presented by Nair *et al.* [15] centers on developing a real-time system for forecasting remote health conditions, which is implemented using the open-source big data processing framework Apache Spark and hosted within a cloud-based infrastructure. In this scalable framework, a ML model determines a user's health condition based on health-related tweets that the application receives instantly. Kolajo *et al.* [16] present a systematic review of big data stream analysis and observe that, although substantial research has concentrated on batch processing, there is growing interest in examining data in motion. Their review provides a comprehensive assessment of the current state of stream processing and recommends that researchers devote greater attention to empirical evaluations of big data streaming tools and technologies. In another contribution, Matteussi *et al.* [17] successfully developed and demonstrated a scalable system for real-time analysis and filtering of job-related tweets from among millions of incoming streaming tweets. These tweets are then categorized into various job classifications, thereby enhancing the efficiency of job searching through Spark streaming.

This paper showed that NLP methods can be applied in big data platforms such as Spark to extract insights from unstructured text and uncover valuable and meaningful patterns for business intelligence. In this work, predictive analytics is applied to unstructured streaming data using various ML models within a Spark ML pipeline. The models' performances are then compared using evaluation metrics such as receiver operating characteristic (ROC) curves and Gini coefficients. The best-performing model is selected as the champion model which can be deployed for smarter business intelligence because insights gained from this analysis offer organizations a clear view of various dimensions of their operations and customer behavior, ranging from service consumption and server performance to website interactions, device locations, user movements, and the flow of physical goods, allowing them to react quickly to new emerging developments. The extra benefits of stream-processing capability, which enhances the velocity aspect of big data, are more effectively realized with the proposed model.

The paper is organized as follows. Section 2 introduces the methodology, the process flow diagram, and the algorithm employed. The dataset utilized, along with all experimental assessments are described in section 3. Lastly, section 4 provides the summary and the conclusion.

2. METHOD

2.1. Intelligent predictions from unstructured streaming data using Spark machine learning pipelines

Spark supports both batch and stream processing through its pipelined architecture, which is a fundamental element of computer system design. Normally, pipelines are divided into stages, and each stage is either a 'transformer' or an 'estimator' that is related to one another to form a pipe-like structure [10], [18]. It refers to a sequence of data-processing components arranged so that each stage passes its output directly to the next stage [19], [20]. Data or instructions enter from one end and exit from the other, thereby increasing overall instruction throughput. In this process, a transformer refers to an algorithm that converts one DataFrame into another, while an estimator is a component that can be fitted on a DataFrame to produce a transformer. In this context, a learning algorithm functions as an estimator that learns patterns from the input DataFrame and generates a trained model as output. For estimator stages, the `fit()` method is called to produce a transformer, and that transformer's `transform()` method is called [21] on the data frame. The pipeline for predictive analytics comprises several sequential stages, including tokenization, feature extraction using HashingTF, term weighting through inverse document frequency (IDF), and application of ML algorithms

for model training and prediction. The stages execute sequentially, and the input DataFrame retrieved from the distributed data store is progressively transformed as it flows through each stage.

To make intelligent predictions with the unstructured textual reviews, to decide the 'star_rating' assigned to each product [22], the Spark pipelining and ML classification algorithms were used. The original problem was a multiclass classification with five star-rating values 1, 2, 3, 4, and 5, which made the results difficult to interpret and complicated the evaluation of the models. Therefore, the problem was reformulated as a binary classification task to facilitate a more straightforward evaluation. Using the proposed method, the attempt was to classify the star_rating into '1' with 4 or 5 star_ratings and '0', consisting of 1 or 2 star_ratings, instead of trying to predict the exact 'star_rating' associated with the review body. All reviews with a star_rating of 3 were excluded since they reflected neutral or indifferent feedback. The different text pre-processing steps were done with this unstructured textual data with NLP and text mining techniques before passing it through the pipeline. The model was trained with the batch data, which is stored in distributed file storage, such as a document-based NoSQL and the data was moved through the pipeline continuously.

The following steps were executed while pre-processing the unstructured textual reviews such as stripping, removing punctuation and alphanumeric characters, removing all stop words, tagging using part-of-speech (POS) tagging, and lemmatization. Lemmatization involves the identification of underlying lemmas of the words in a text, whereas with tagging, a part of speech was assigned to each word in a list using context clues. With a view to applying ML classification with the streaming pipeline, the cleaned dataset was split into training and testing portions using a 70:30 ratio, and a range of methods including logistic regression (LR), support vector machines (SVM), and naïve Bayes (NB) were employed on the dataset to estimate the probability that a product would be accepted. The model training was done with the training dataset and tested with the testing set. The 'review_body' and 'star_rating' fields were selected as the 'feature' and 'label' fields.

The lemmatized textual data in the training set was passed through the pipeline for fitting with the different ML models, where the operations of tokenization, hashing term frequency (TF) and IDF calculations [23] were performed on it. With tokenization and hashing, the data was converted and stored as a two-dimensional vector. With tokenization, the individual review texts are broken down into separate words known as tokens. Usually, a simple tokenizer class in Spark provides this functionality. The regular expression (Regex) tokenizer used enables more sophisticated text splitting by using Regex patterns to identify tokens. Instead of the TF-IDF technique, the hashing trick was used here [24]. Since TF-IDF calculations tend to get complicated in a general scenario, it will be far worse in a distributed system setup. Hence, to reduce the memory and computational overheads, hashing was implemented with Spark. Hashing is a type of transformer that converts sets of terms into fixed-length feature vectors. In text processing, these sets of terms could represent a bag of words, which includes all words in a text, but hashing applies the hashing trick to achieve this transformation.

A raw attribute was converted into an index by applying a hashing procedure. The hashing mechanism employed was MurmurHash3 [12], after which term frequencies were calculated using the resulting indices. This approach removes the requirement for generating a global term-to-index dictionary which can be computationally costly for very big datasets. But it introduces the possibility of hash collisions, where unrelated input features may be assigned the same hashed term and to minimize collision risks, the number of buckets in the hash table can be increased. Because a simple modulo operation was used to map the hash output to a column index, the feature dimension was chosen as a power of two to ensure a uniform distribution across columns. The default dimensionality is set to $2^{18} = 262,144$, and a configurable binary parameter determines whether term frequencies are counted in binary form. Normally, *idf* of a term 't' is calculated using (1).

$$idf(t) = \frac{n}{df(t)} \quad (1)$$

Where n is the total number of terms and $df(t)$ is the frequency of 't' in the document [25]. But when a word that was not in the vocabulary occurs, the corresponding $df(t)$ will be zero, which was obtained from the hashing trick. As division by zero was not possible, the value was smoothed by adding 1 to the denominator. Hence, (2) was used to calculate the IDF, where n is the total number of terms.

$$idf(t) = \log\left(\frac{n}{df(t)+1}\right) \quad (2)$$

Then, model training was carried out using classification algorithms such as LR, SVM, and NB, which are available within the Spark ML library. The operation is an estimator and an additional column,

'prediction' was getting added to it. Then, the pipelined ML model is tested with the test data. Whenever an estimator function such as a ML algorithm was triggered in a pipeline, it's preceding transformation functions which were pending to be executed are enabled, before it estimated the predictive labels. It is transforming our resultant data frame with an additional feature called 'prediction', apart from a label which represents the expected predictions. The accuracy of these predictions made by the different ML models using the ML pipeline was evaluated using the metrics: the ROC-area under the curve (AUC) curves and the Gini coefficient. The best-performing model was selected as champion model and it is deployed for production.

2.2. Process flow diagram and algorithm

Figure 1 illustrates the overall flowgraph of the proposed approach, outlining the sequential processes and interactions among its key components. The streaming data is prepared first, and the pre-processing up to lemmatization is applied with the textual reviews. Then this data is passed through a streaming pipeline, which consists of the stages, tokenization, hashing, IDF calculation, and finally the ML modelling. The proposed approach for making the predictive analysis with the streaming pipeline is detailed in the Algorithm 1 predictive analytics pipeline.

Algorithm 1. Predictive analytics pipeline

```

Input: The Data frame 'data' consisting of unstructured review_body and star_rating
Output: The deployable machine learning model for intelligent prediction
//Preparing the dataset for modelling by removing neutral comments
1: Load the data frame 'data' from the distributed storage facility.
2: For each row 'i' in the data frame 'data'
3:   if (i[star_rating] == '3') filter it
4:   else if ((i[star_rating] == '1') || (i[star_rating] == '2')), i[y]=0, where y
      ∈ Target
5:   else i[y] = 1, where y ∈ Y /* Y is a new column 'Target' */
6:   End of if
7: End of for
//Pre-processing the entire set of 'review_body' in data, which is the unstructured textual
data
8: For each 'review_body', 'X' in data, perform the following transformations from step 9
   to step 16 as pre- processing, and with each step a new column will be added to 'data'
9:   lower_text = lower(X)
10:  text_non_ascii = strip_non_ascii(lower_text)
11:  fixed_abbrev = fix_abbreviation(text_non_ascii)
12:  removed_features = remove_punctuations_special_symbols(fixed_abbrev)
13:  removed_stops = remove_stops(removed_features)
14:  tag_and_remove_pos = remove_tags(removed_stops)
15:  lemmatized_text = lemmatize(tag_and_remove_pos)
16: End of for
17: Construct the final data frame 'data' with the result of previous step 15 as
18:   data = data.select(lemmatized_text, date, Target) and
19:   set 'text' = lemmatized_text and 'label' = Target
20: Divide the data for training and testing stages as 70% data for training and 30% for
   test as train_data=random_select(data,70) and test_data=random_select(data,30)
21: Train and fit the model by passing the train data through the streaming pipeline as in
   step 22 with the following stages, where ML is the machine learning algorithm used.
22: Streaming_pipeline = Pipeline(stages = [tokenizer, hashing, IDF, ML])
23: For each stage in the pipeline, perform the following Steps 25 to Step 37
24:   switch(stage)
25:     Case 'tokenizer':
26:       tokenize('text') = {t: t ∈ T, where T is the text in the dataset}
27:     Case 'hashing':
28:       //Deciding the term frequency by applying hashing trick to each term in the {t}
29:       For each token {t: t ∈ T}
30:         hashing(t) = {index(t, V), where V is the fixed length feature
           vector}
31:       End for
32:       //Hashing is a transformer which takes sets of terms and converts them into fixed-length
       feature vectors
33:       Case 'IDF':  $idf(t) = \log(n/(hashing(t) + 1))$ 
34:       //where n → total no of terms in V
35:        $hashing\_idf(t, T) = hashing(t) * \log(n/(hashing(t) + 1))$ 
36:       //whatever we are processing now is T
37:     Case 'ML':
38:       Carry out model construction using the following supervised machine learning
       classification techniques, ML = {LR, SVM, NB}, where LR = logistic regression, SVM =
       support vector machines, and NB = naïve Bayes
39:     End switch
40: End For

```

37: Test the models using the test data and evaluate the results with AUC-ROC and Gini Coefficient.
 38: Select the champion model which is the deployable model that can work with stream data
 39: End

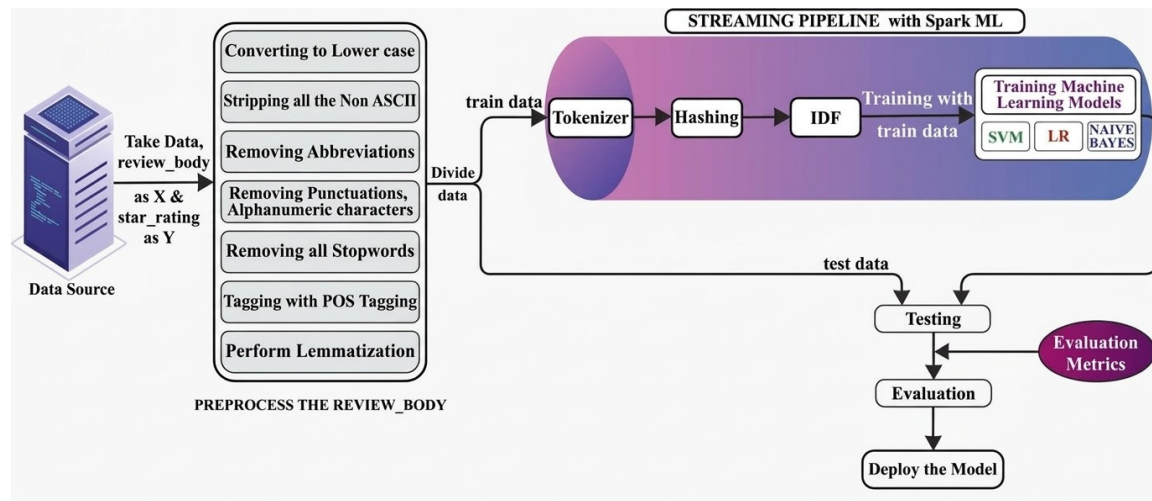


Figure 1. Flow diagram used with the data processing platform

3. RESULTS AND DISCUSSION

3.1. Dataset used and experimental setup

The dataset utilized in this is the Amazon reviews for sports and outdoors, downloaded from Kaggle. It contains 4,849,945 records, approximately 2 GB in size, and includes details of sports products along with customer feedback and summary information. The analysis focuses on the unstructured textual reviews `review_body` provided by customers, along with the corresponding star rating for various sports products [22]. The experiments were done using the Databricks community edition within the Apache Spark ecosystem, configured with 6 GB of RAM, 0.88 cores, Databricks runtime (DBR) 6.4 with Spark 2.4.5, and Scala 2.11. Additionally, Google Colab, a free cloud platform by Google, was used to encourage research leveraging the Google Cloud Platform (GCP) for computations and evaluations, offering benefits such as integration with reporting and extract, transform, and load (ETL) tools. The Spark pipelining implementations were executed using PySpark on a GCP compute engine backend, with 12.72 GB RAM and 107.77 GB of disk space.

3.2. Comparative performance evaluation of artificial intelligence models for champion model selection

To activate and perform the intelligent processing of the unstructured reviews, stored in the distributed file system, the data is passed through a streaming pipeline. Spark streaming and Spark ML are used together to perform the predictive analytics, and in the final stage of the pipeline, different ML classification models like LR, SVM, and NB, are implemented. These three algorithms represent various aspects of ML such as regression analysis, which is based on statistical modelling, whereas SVM plays with the idea of state-based analogy, with a unique way of evaluation [26]. The Bayesian theorem backs the NB algorithm that works with probabilistic analysis [27].

The training data is passed through the pipeline stages, such as tokenization, hashing, IDF, and finally the different ML algorithms are applied, where the pipelines perform differently with each of the algorithms used. The performance of the pipelined model in making predictions with the test data is measured by considering the metrics AUC-ROC and the Gini coefficient. The ROC graph is a standard way of analyzing the performance of ML algorithms based on its ability to do predictions. Gini coefficient is another performance metric used in classification problems. Gini index can be derived straight away from the AUC-ROC, and it is an indicator of how well the model is performing. The Gini index for the proposed models can be calculated as in (3). The performance of the pipeline with the following ML algorithms is evaluated.

$$\text{Gini Coefficient (Model)} = (2 \times \text{AUC}) - 1 \quad (3)$$

3.2.1. Logistic regression model

It represents a specialized form of generalized linear model intended to compute the likelihood of possible outcomes. Within Spark ML, LR provides support for both two-class prediction using binomial LR and multi-category prediction using multinomial LR [28]. Here, this ML model is used for making binary class predictions regarding the acceptance of a product, based on the reviews. With a prediction value '1', the product is assumed to be acceptable, and prediction value '0' denotes its unacceptability.

- AUC-ROC: when this statistical model was applied using that workflow, the results showed an AUC of 61%. The ROC curve of LR pipeline is shown in Figure 2. The ROC-AUC for the LR model shows 61% of accuracy, which indicates its confidence in predicting labels accurately with an error chance of 39%. It indicates that when the predictions are made using this ML pipeline, 61% of predictions are made correctly and 39% of predictions were wrong.
- Gini coefficient: evaluating the LR model with the Gini coefficient gave a score of 0.22, which implies that the model is not suitable for production. Since LR works based on likelihood and is prone to outliers, it is conveniently eliminated in the champion model selection.

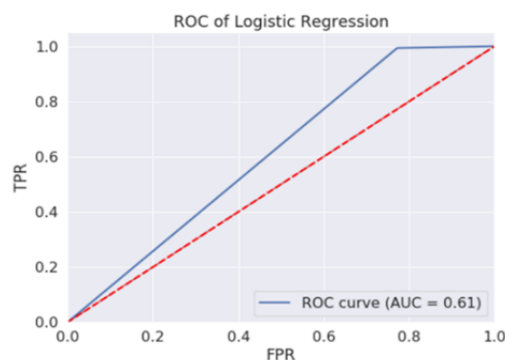


Figure 2. The AUC-ROC of LR

3.2.2. Naïve Bayes model

NB classifiers are a group of straightforward probabilistic multiclass models that rely on Bayes' theorem, assuming that all features are conditionally independent of one another. With just one pass through the training dataset, the algorithm calculates the conditional probability distribution of each feature for every label. During prediction, it uses Bayes' theorem to determine the conditional probability distribution of each label given a new observation [28].

- AUC-ROC: with NB classifiers the streaming pipelining gave an AUC value of 81%. AUC value 0.81 means that 81% of the predictions are correct and that only 19% of predictions are wrong. The ROC curve of the pipeline is shown in Figure 3.
- Gini coefficient: NB algorithm gave a value of 0.62 as its Gini coefficient, which is a promising outcome, since the strategy of Gini coefficient technique demands the performance of the model must be greater than 0.60 [29], [30].

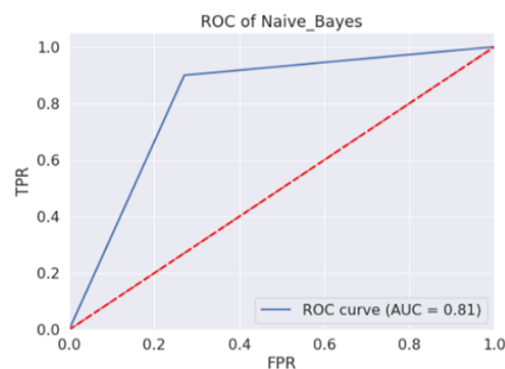


Figure 3. The AUC-ROC of NB

3.2.3. Support vector machine model

An SVM creates a hyperplane, or multiple hyperplanes, within a high-dimensional or even infinite-dimensional space, which can be useful for classification, regression, and other tasks. In essence, an effective separation is obtained by selecting the hyperplane that maximizes the distance to the closest training samples from each class. Generally, a larger margin results in lower generalization error for a linear SVM classifier [31].

- AUC-ROC: the ROC curve of the SVM pipeline is shown in Figure 4. The pipeline with the SVM model gave an AUC value of 72%. The model's probabilistic metrics indicate an accuracy of 72%, reflecting the SVM model's capability to correctly classify most labels, while leaving a remaining 28% margin for potential misclassification.
- Gini coefficient: SVM scores 0.44 for Gini coefficient, which implies that the SVM model is not up to the notch for production in this context since it does not breach the threshold of 0.60.

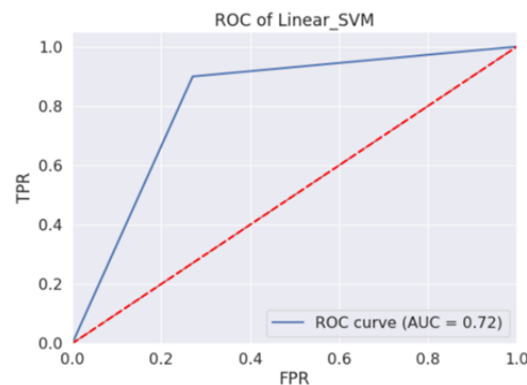


Figure 4. The AUC-ROC of SVM

3.2.4. Champion model selection

The ROCs of the three algorithms used for making intelligent predictions with the pipelined modelling are combined and are shown in Figure 5. From Figure 5, it is evident that the NB ML model performs better in the Spark ML and pipelining environment for unstructured textual data analysis with an AUC of 81%. i.e., when the entire test data is evaluated, this study got a harmonized prediction and label for 81% of the data, whereas it contradicted the label with prediction for the remaining 19%. The SVM and LR models also perform with a competing AUC of 72% and 61% which does not imply that these models are incompetent, instead, these algorithms are also adaptable since the models show AUC over the threshold value, which is 50%. With this analysis, this study selected the NB model as the champion, which is used for deployment purposes.

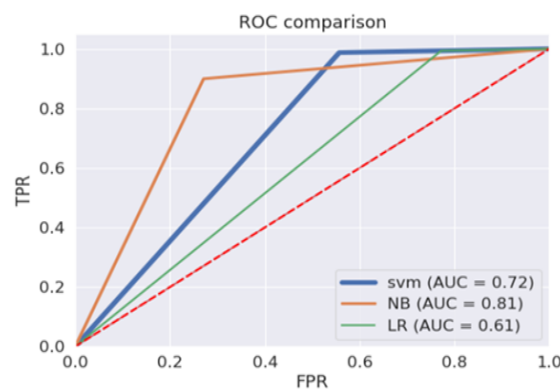


Figure 5. Comparison of ROCs of different ML models

Models which are having the Gini coefficient above 60% are considered as good models [29], [30] which can be taken as the champion model and be deployed. The following graph in Figure 6 demonstrates a

comparison of Gini coefficient obtained for different ML algorithms. For NB model, this study got Gini coefficient as 62%, which is above the margin of 60% and hence it can be considered as the champion model. According to Chen *et al.* [32], the two key obstacles in tapping big data are its unstructured nature and the volume of the data to be analyzed. This paper proposes that NLP techniques can be applied within big data platforms such as Spark to analyze unstructured streaming data and uncover valuable patterns that support business intelligence.

According to Kamal *et al.* [11], the huge data can be processed in real time by using Spark, and they have used few structured query language (SQL) queries to mine information from the big data. Here, text mining and NLP are implemented to apprehend the unstructured textual data of streaming nature. The data of streaming nature is processed and then the currently discovered data are appended to it.

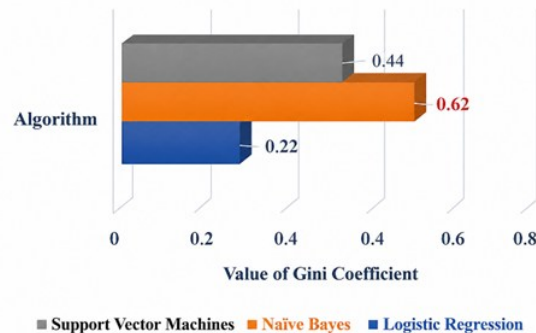


Figure 6. Gini coefficient value of different ML models

4. CONCLUSION

In this paper, to take advantage of intelligent processing on unstructured textual data with the velocity factor, a model is developed using Spark ML pipelining concepts. The proposed model has the capabilities to process unstructured data with procedural processing flexibility for implementing artificial intelligence. The data, which is spread globally over various clusters, can be mined in a real-time fashion, since it is possible that the data can be from a live source or from a streamable source. It helps to implement predictive analytics with procedural-level programming flexibility over stream data without any additional engineering upgrades. The authors suggest that business intelligence can be enhanced by capturing and analyzing large volumes of data in real time. Stream processing is inarguably one of the most valuable components of the proposed system. By enabling real-time computations, it equips businesses with the ability to generate immediate insights, optimize operations, and make proactive decisions based on live data streams.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Benymol Jose	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Sajimon Abraham	✓	✓		✓		✓		✓	✓	✓	✓			
Nikitha Sajith	✓	✓	✓					✓		✓	✓			

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study are openly available in Kaggle at <https://www.kaggle.com/datasets/beaglelee/amazon-reviews-us-sports-v1-00-tsv>, reference [22].

REFERENCES

- [1] G. Mohanraj, R. K. Nadesh, J. Jagannathan, S. Adityan, and V. Sathiyamoorthi, "Monitoring incident response using real-time analytics," in *2nd International Conference on IT Innovations and Knowledge Discovery, ITIKD 2024*, 2025, pp. 1–8, doi: 10.1109/ITIKD63574.2025.11004824.
- [2] Md A. Alam, A. R. Nabil, A. A. Mintoo, A. Islam, "Real-time analytics in streaming big data: techniques and applications," *Journal of Science and Engineering Research*, vol. 1, no. 1, Oct. 2024, doi: 10.70008/jeser.v1i01.56.
- [3] A. Vogel, S. Henning, E. Wohlfeil, O. Ertl, and R. Rabiser, "A comprehensive benchmarking analysis of fault recovery in stream processing frameworks," in *DEBS 2024 - 18th ACM International Conference on Distributed and Event-Based Systems*, Jun. 2024, pp. 171–182, doi: 10.1145/3629104.3666040.
- [4] S. Henning and W. Hasselbring, "Benchmarking scalability of stream processing frameworks deployed as microservices in the cloud," *Journal of Systems and Software*, vol. 208, Oct. 2024, doi: 10.1016/j.jss.2023.111879.
- [5] R. N. S. Widodo, H. Abe, and K. Kato, "Hadoop data reduction framework: applying data reduction at the DFS layer," *IEEE Access*, vol. 9, pp. 152704–152717, 2021, doi: 10.1109/ACCESS.2021.3127499.
- [6] M. U. Demirezen and T. S. Navruz, "Performance analysis of lambda architecture-based big-data systems on air/ground surveillance application with ADS-B data," *Sensors*, vol. 23, no. 17, Aug. 2023, doi: 10.3390/s23177580.
- [7] M. Fragkoulis, P. Carbone, V. Kalavri, and A. Katsifodimos, "A survey on the evolution of stream processing systems," *VLDB Journal*, vol. 33, no. 2, pp. 507–541, 2024, doi: 10.1007/s00778-023-00819-8.
- [8] A. A. Munshi and Y. A. -R. I. Mohamed, "Data lake lambda architecture for smart grids big data analytics," *IEEE Access*, vol. 6, pp. 40463–40471, 2018, doi: 10.1109/ACCESS.2018.2858256.
- [9] Z. Hasan, H. J. Xing, and M. I. Magray, "Big data machine learning using Apache Spark mllib," *Mesopotamian Journal of Big Data*, vol. 2022, pp. 1–11, 2022, doi: 10.58496/MJBD/2022/001.
- [10] U. Suthakar, L. Magnoni, D. R. Smith, and A. Khan, "Optimised lambda architecture for monitoring scientific infrastructure," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 6, pp. 1395–1408, 2021, doi: 10.1109/TPDS.2017.2772241.
- [11] M. V. Kamal, P. Dileep, and D. Vasumati, "Spark streaming for predictive business intelligence," in *Soft Computing and Signal Processing*, Singapore: Springer, 2019, pp. 289–298, doi: 10.1007/978-981-13-3393-4_30.
- [12] A. Svyatkovskiy, K. Imai, M. Kroeger, and Y. Shiraito, "Large-scale text processing pipeline with Apache Spark," in *2016 IEEE International Conference on Big Data, Big Data 2016*, Dec. 2016, pp. 3928–3935, doi: 10.1109/BigData.2016.7841068.
- [13] A. Ahmed, R. Xi, M. Hou, S. A. Shah, and S. Hameed, "Harnessing big data analytics for healthcare: a comprehensive review of frameworks, implications, applications, and impacts," *IEEE Access*, vol. 11, pp. 112891–112928, 2023, doi: 10.1109/ACCESS.2023.3323574.
- [14] E. Azhir, M. Hosseinzadeh, F. Khan, and A. Mosavi, "Performance evaluation of query plan recommendation with Apache Hadoop and Apache Spark," *Mathematics*, vol. 10, no. 19, 2022, doi: 10.3390/math10193517.
- [15] L. R. Nair, S. D. Shetty, and S. D. Shetty, "Applying spark-based machine learning model on streaming big data for health status prediction," *Computers and Electrical Engineering*, vol. 65, pp. 393–399, 2018, doi: 10.1016/j.compeleceng.2017.03.009.
- [16] T. Kolajo, O. Daramola, and A. Adebisi, "Big data stream analysis: a systematic literature review," *Journal of Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0210-7.
- [17] K. J. Matteussi, J. C. S. D. Anjos, V. R. Q. Leithardt, and C. F. R. Geyer, "Performance evaluation analysis of Spark Streaming backpressure for data-intensive pipelines," *Sensors*, vol. 22, no. 13, Jun. 2022, doi: 10.3390/s22134756.
- [18] Apache Spark, "Apache Spark," spark.apache.org. Accessed: Jun. 19, 2025. [Online]. Available: <https://spark.apache.org/>
- [19] F. Cerezo, C. E. Cuesta, J. C. M.-Herranz, and B. Vela, "Deconstructing the lambda architecture: an experience report," in *2019 IEEE Int. Conf. Software Architecture Companion (ICSA-C)*, 2019, pp. 196–201, doi: 10.1109/ICSA-C.2019.00042.
- [20] Amazon Web Services, "Streaming data," [aws.amazon.com](https://aws.amazon.com/streaming-data/). Accessed: Jun. 21, 2025. [Online]. Available: <https://aws.amazon.com/streaming-data/>
- [21] Apache Spark, "ML pipelines - Spark 3.0.0 documentation," [spark.apache.org](https://spark.apache.org/docs/latest/ml-pipeline.html). Accessed: Jun. 21, 2025. [Online]. Available: <https://spark.apache.org/docs/latest/ml-pipeline.html>
- [22] B. Lee, "Amazon reviews US - sports v1.00 TSV," [kaggle.com](https://www.kaggle.com/datasets/beaglelee/amazon-reviews-us-sports-v1-00-tsv). Accessed: Jan. 8, 2025. [Online]. Available: <https://www.kaggle.com/datasets/beaglelee/amazon-reviews-us-sports-v1-00-tsv>
- [23] R. Ranjan, S. Garg, A. R. Khoskbar, E. Solaiman, P. James, and D. Georgakopoulos, "Orchestrating big data analysis workflows," *IEEE Cloud Computing*, vol. 4, no. 3, pp. 20–28, May 2017, doi: 10.1109/MCC.2017.55.
- [24] J. C.-Rodríguez et al., "Apache hive: from mapreduce to enterprise-grade big data warehousing," in *SIGMOD '19: 2019 International Conference on Management of Data*, 2019, pp. 1773–1786, doi: 10.1145/3299869.3314045.
- [25] K. Papineni, "Why inverse document frequency?," in *2nd Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL 2001*, 2001, pp. 1–8, doi: 10.3115/1073336.1073340.
- [26] N. A.-Tejera, M. Gamarra, J. I. Vélez, and E. Zurek, "A distance-based kernel for classification via support vector machines," *Frontiers in Artificial Intelligence*, vol. 7, Feb. 2024, doi: 10.3389/frai.2024.1287875.
- [27] Z. Yang et al., "A new three-way incremental naive Bayes classifier," *Electronics*, vol. 12, no. 7, 2023, doi: 10.3390/electronics12071730.
- [28] A. K. Gárateescamilla, A. H. El Hassani, and E. Andres, "Big data execution time based on spark machine learning libraries," in *ICCBDC '19: 2019 3rd International Conference on Cloud and Big Data Computing*, Aug. 2019, pp. 78–83, doi: 10.1145/3358505.3358519.
- [29] E. Schechtman and G. Schechtman, "The relationship between Gini terminology and the ROC curve," *Metron*, vol. 77, no. 3, pp. 171–178, Dec. 2019, doi: 10.1007/s40300-019-00160-7.
- [30] C. Gini, "On the measurement of concentration and variability of characters," *Metron - International Journal of Statistics*, vol. 63, no. 1, pp. 1–38, 2005.

- [31] S. Ghosh, A. Dasgupta, and A. Swetapadma, "A study on support vector machine based linear and non-linear pattern classification," in *2019 International Conference on Intelligent Sustainable Systems (ICISS)*, 2019, pp. 24–28, doi: 10.1109/ISS1.2019.8908018.
- [32] Y. Chen, C. Li, and H. Wang, "Big data and predictive analytics for business intelligence: a bibliographic study (2000–2021)," *Forecasting*, vol. 4, no. 4, pp. 767–786, Sep. 2022, doi: 10.3390/forecast4040042.

BIOGRAPHIES OF AUTHORS



Dr. Benymol Jose     pursued masters in Computer Science from Bharathidasan University, Thiruchirappalli, Tamil Nadu in 1999; an M.Phil. in Computer Science from Madurai Kamaraj University, Madurai, Tamil Nadu in the year 2014; and Ph.D. in Computer Science from Mahatma Gandhi University, Kottayam, Kerala, India in 2021 on the topic "Unstructured data mining in big data: a NoSQL perspective". She is currently working as an associate professor in the Department of Computer Applications, Marian College Kuttikkanam Autonomous, Idukki, Kerala, India. She has published many papers in journals and conference proceedings, including IEEE, Elsevier, and ACM, and most of them are indexed by Scopus. Her main research work focuses on unstructured data mining, machine learning, NoSQL databases, and big data analytics. She has 26 years of teaching experience and 8 years of research experience. She can be contacted at email: benymol.jose@mariancollege.org.



Dr. Sajimon Abraham     is a MCA, M.Sc. Mathematics, MBA, Ph.D. in Computer Science. He has been working as a faculty member in Computer Applications and IT, School of Management and Business Studies, Mahatma Gandhi University, Kottayam, Kerala, India. He currently holds the additional charge of director (Hon), university center for international cooperation. He was previously working as a systems analyst in the Institute of Human Resource Development, a faculty member of Computer Applications in Marian College, Kuttikkanam, and a database architect in Royal University of Bhutan under the Colombo plan on deputation through the Ministry of External Affairs, Govt. of India. His research area includes data science, spatio-temporal databases, mobility mining, sentiment analysis, big data analytics, e-learning, database and data mining, data management, web clickstream analysis, and business analytics. He has published 110 articles in national, international journals, and conference proceedings. He can be contacted at email: sajimabraham@rediffmail.com.



Nikitha Sajith     is a quality engineer at LTM Limited in Bengaluru, specializing in software quality assurance and emerging technologies. She holds a Bachelor of Technology in Computer Science and Engineering from Christ Deemed to be University, Bengaluru, Karnataka where she also served as the vice chair of the IEEE Women in Engineering Student Chapter. While working as a quality engineer, she is also pursuing a hybrid Master of Technology in Artificial Intelligence and Data Science Engineering from Indian Institute of Technology Patna, India. Her technical interests include computer vision, machine learning, and the application of intelligent automation in quality engineering. She has contributed to multiple projects involving model development, performance evaluation, and the integration of AI-driven solutions into software workflows. She continues to pursue research at the intersection of software quality, AI, and emerging digital technologies. She can be contacted at email: nikithasajith2020@gmail.com.