Assessing naive Bayes and support vector machine performance in sentiment classification on a big data platform

Redouane Karsi, Mounia Zaim, Jamila El Alami
LASTIMI Laboratory, Higher School of Technology of Sale, Mohammed V University, Morocco

Article Info

Article history:

Received Feb 8, 2021
Revised May 19, 2021
Accepted Jul 28, 2021

Keywords:

Apache Spark
Big data
Naive Bayes
Sentiment analysis
Support vector machine

ABSTRACT

Nowadays, mining user reviews becomes a very useful mean for decision making in several areas. Traditionally, machine learning algorithms have been widely and effectively used to analyze user's opinions on a limited volume of data. In the case of massive data, powerful hardware resources (CPU, memory, and storage) are essential for dealing with the whole data processing phases including, collection, pre-processing, and learning in an optimal time. Several big data technologies have emerged to efficiently process massive data, like Apache Spark, which is a distributed framework for data processing that provides libraries implementing several machine learning algorithms. In order to evaluate the performance of Apache Spark's machine learning library (MLlib) on a large volume of data, classification accuracies and processing time of two machine learning algorithms implemented in spark: naive Bayes and support vector machine (SVM) are compared to the performance achieved by the standard implementation of these two algorithms on large different size datasets built from movie reviews. The results of our experiment show that the performance of classifiers running under spark is higher than traditional ones and reaches F-measure greater than 84%. At the same time, we found that under spark framework, the learning time is relatively low.

Corresponding Author:

Redouane Karsi
Laboratory of System Analysis, Information Processing and Integrated Management
Higher School of Technology of Sale
Mohammed V University
Morocco
Email: rdkarsi@yahoo.fr

1. INTRODUCTION

Following the explosion of subjective textual information in social networks, forums, and blogs in the form of opinions freely written by internet users, sentiment analysis has emerged as a discipline of data mining that aims to extract an opinion from unstructured textual data. It allows, for example, managing the marketing strategy of a company based on the analysis of consumer feedback towards a product [1], [2]. Tackling sentiment analysis issues is done according to several approaches, a lexical approach [3] that uses a dictionary to identify the text's sentiment from its constituents' polarity, whether they are words or sentences. However, this approach is not always the best solution because a word can have different orientations depending on the domain where it appears. Indeed, "a dangerous player" has a positive polarity in the sports domain, but "dangerous animal" has a negative polarity in the animal domain. Besides the lexical approach, there is an approach using machine learning methods [4], and for comparison, research has shown that machine learning methods are more accurate than lexical-based methods [5].

As experiments have shown that machine learning algorithms outperform lexical-based algorithms, they are a preferred choice for sentiment classification problems. In Research works that address sentiment analysis problems with machine learning algorithms, we often use small or medium-sized learning data that do not require much hardware resources. In these conditions, these algorithms reach high accuracies and very low latency. When the training data is large, machine learning algorithms face many challenges. Their design must accommodate limited memory resources and ensure adequate execution time [6].

The concept of big data has emerged to bring together all technologies for the collection, storage, and processing of massive data that traditional tools can not process [7], [8]. The purpose of this paper is to evaluate the performance of two machine learning methods embedded into a big data framework named Apache Spark on a large dataset by comparing them to the performance of these same methods executed according to a traditional approach. By testing on several hardware configurations of the spark cluster, we found that the classification performances of naive Bayes and support vector machine (SVM) under spark platform are better than those achieved in a single machine with F-measure beyond 84%. We also observe that support vector machine (SVM) and naive Bayes are scalable machine learning algorithms on the spark platform.

The remainder of the paper is organized as follows. In section 2, related work is presented. In section 3, the adopted methodology is detailed. Experimental results are presented and discussed in section 4. Finally, in section 5, the paper is concluded, and future research issues are underlined.

2. RELATED WORK

In this section, we discuss different works on sentiment analysis, big data frameworks, and distributed machine learning methods.

2.1. Sentiment analysis

Sentiment analysis is a set of techniques including text analytics, computational linguistics, and natural language processing for classifying texts into positive, negative, or neutral. Pang *et al.* [9] are the origin of the first studies on sentiment analysis. They used a machine learning approach to classify movie reviews. Kim *et al.* [10] had tested feature selection on the support vector machine (SVM) algorithm. The authors concluded that SVM outperforms all other machine learning algorithms for sentiment classification tasks. Jeong *et al*. [11] use sentiment analysis to identify customer preferences and trends. Wu *et al*. [12] had explored tweets to predict stock market price. They used both lexicon and machine learning approaches. The authors found that machine learning is better than the lexicon approach. Kumari *et al.* [13] collected tweets in all languages, then translate them online to English. Afterwards, tweets are classified as positive or negative using a machine learning algorithm to serve as naive Bayes classifier's training data. This approach provides good classification results.

2.2. Distributed machine learning

Distributed machine learning can deal with computational complexity algorithms and memory restrictions in large datasets [14]. To solve the problem of algorithms' inability to process a large volume of data, they must run on several machines or processors [15]. Besides the prediction efficiency by parallel data processing, the distributed machine learning algorithms provide fault tolerance by copying the data on several machines. Moreover learning from distributed data using different algorithms produces good precisions, especially in large domains [16]. The distributed algorithms can be integrated with other data processing systems [17]. However, designing and implementing distributed algorithms is a hard task [18]. Also, the distributed algorithms are effective when the nodes dedicated to the data processing communicate directly. However, communication across the network between the nodes entails a longer data processing time [19].

2.3. Machine learning tools

Spark MLlib [20] and Mahout [21] are two open-source tools that include several scalable machine learning algorithm implementations. The implemented algorithms perform classification, regression, clustering, collaborative filtering, and dimensionality reduction tasks. They are independent of the big data engine, so they are portable, and we can easily implement them in another big data platform. Mahout supports Hadoop, spark and H2O. Further, although these algorithms are mainly intended for processing large data in a distributed environment, they are also used to process small data on a single machine. There are also frameworks for large-scale data learning, such as SAMOA, but it is a project in its beginnings [22].

3. METHODOLOGY

As shown in Figure 1, we constructed a sentiment classification system from a dataset called Amazon Movie Reviews containing over 8 million reviews. To test our system's resilience and its ability to scale up, we worked with five datasets extracted from the Amazon Movie Reviews dataset, whose size varies between 10,000 and 200,000 reviews. The processing of this large data is made using the Apache Spark framework that incorporates machine learning libraries and relies on a distributed processing system to execute preprocessing and learning tasks. We chose two algorithms for our experiment: naive Bayes and support vector machines. The choice of these two algorithms is argued by the fact that they are the best algorithms in terms of precision, as it has been proven in various research works [23].

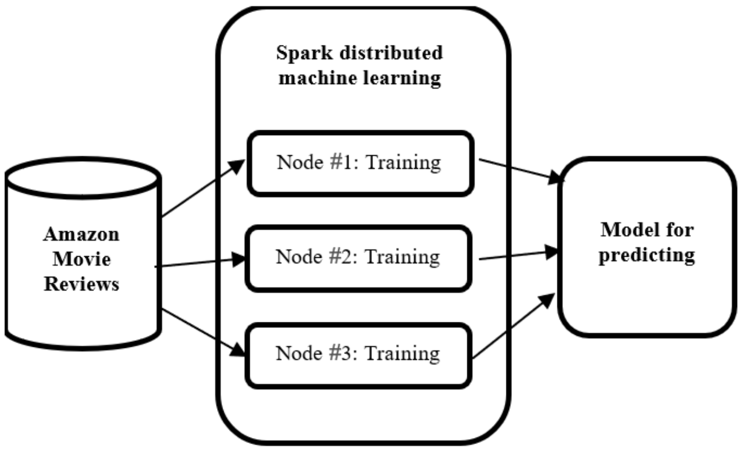

Figure 1. Distributed machine learning architecture in spark platform

3.1. The dataset

Our experimental study dataset: Amazon Movie Reviews Dataset [24] is part of the Stanford Network Analysis Project. It is a collection of opinions collected from Amazon over a period of 10 years until October 2012. It has about 8 million reviews. In our experiment's case, five disjoint subsets with respective sizes of 10 k, 50 k, 100 k, 150 k, 200 k reviews have been extracted from this dataset. Each review is composed of eight features. In our study, we extracted only two features, which are the review text and the score. The review text is transformed into a vector using the bag of words model. The scores are converted to 0's and 1's to assign polarities to reviews by applying the following conversion rule: Reviews with scores between 1 and 3 are considered negative and are assigned the value 0, while reviews with scores between 4 and 5 are regarded as having positive sentiment and are given the value 1. In our experiment, we used balanced training data between the positive and negative classes.

3.2. Feature selection

The extracted datasets have undergone preprocessing operations through three stages:

- Tokenisation: Each review is segmented by splitting the text into words separated by spaces and punctuations.
- Stop words removal: The removal of empty words such as articles and punctuation.
- Stemming: Each word is converted into its stem.
- Feature selection: The selected features correspond to sequences of a single word called unigrams, previous studies have argued that classification accuracy is more accurate when using unigrams in movie domain, then, the selected features are weighted according to the TF-ITF scheme following the formula.

$$\mathrm{TFIDF} = \mathrm{TF} * \log(\frac{\mathrm{N}}{\mathrm{df}}) \quad (1)$$

N is the total number of documents.
df is the number of documents in which the term appears.
TF is the number of times a term appears in a document.

3.3. Training the classifier

Referring to the literature, support vector machines and naive Bayes are the classifiers that bring the best performances in the movie domain. Our experimental study focuses on these two algorithms.

– Support vector machines:

Support vector machines (SVM) is a supervised learning algorithm used to perform classification and regression tasks [25]. SVM is a classifier based on a statistical approach for either learning or prediction. It is developed by Vapnik [26]. Its operating principle consists of performing a set of computations to determine a hyperplane that separates the data into two different classes, so that the distance between the two classes is maximum. SVM seeks to perform a binary classification by defining a hyperplane that separates the two classes' data. This can be achieved by expressing the data in a multidimensional space that makes the data's linear separation quite possible. What makes SVM a complex algorithm is that it uses a kernel function that relies on the projection of data into a higher-dimensional space in which the problem becomes linear. The algorithm must go through several iterations to select the only hyperplane among all those who separate learning data according to their class. The particularity of this hyperplane is that it is located at a maximum distance from different learning instances.

– Naive Bayes:

Naive Bayes [27] is a classifier based on the Bayes theorem. In this model, the random variables are statistically independent given a class c. This assumption of data independence will reduce the computation time. To predict the class c_i of a random variable X by applying the Bayes theorem, we calculate the conditional probability that the variable X belongs to the class c_i by this formula:

$$P(C = c_i/X) = P(C = c_i) \times \frac{P(X/C = c_i)}{P(X)} \tag{2}$$

$P(C = c_i/X)$ is the probability of class c_i conditioned on X.
$P(C = c_i)$ is the probability of class c_i.
$P(X/ C = c_i)$ is the probability of X conditioned on c_i.
P(X) is the probability of X.
The random variable X will be assigned the class c_i which maximizes the conditional probability $P(C = c_i/X)$.

– Apache Spark:

The first Big Data platforms like Hadoop based on the MapReduce framework were mainly designed for batch data processing which requires frequent access to the storage space, but in the case of iterative computing, the performance of the MapReduce frameworks decreases considerably. With the widespread use of machine learning algorithms for data analysis, and in order to overcome the problem of intensive computations performed by machine learning algorithms, several techniques have been developed, especially for the fast processing of massive data. Among these techniques, Apache Spark is positioned as an efficient solution that provides a higher-level programming interface to develop distributed applications. In this platform, the data and the intermediate results are loaded and stored in the memory of cluster machines using a data abstraction system called Resilient Distributed Dataset providing data processing in parallel.

4. EXPERIMENTS AND RESULTS

Several experiments were conducted to highlight the performance of two classifiers: svm and naive Bayes in a distributed environment such as spark. Our evaluation was done from several angles by observing indicators such as (classification F-measure and time needed to complete the learning job) while varying the dataset size and the number of cluster slave nodes.

4.1. Setup spark cluster

To set up the environment to train the classification algorithms, we have implemented a multi-node cluster architecture. Our system is composed of a master node, and three slave nodes, each node of the cluster has a configuration with a 3.4 GHz processor, 8 GB memory, and 500 GB hard disk. These different nodes are interconnected with a local network with a speed of 100 Mbps. We opted for this configuration to provide the same conditions in which traditional algorithms have been experimented on a single machine.

4.2. Training and classification algorithm

After extracting five datasets of respective sizes of 10 k, 50 k, 100 k, 150 k and 200 k from Amazon Movie reviews dataset, we have written a java program which exploits the spark's machine learning library MLlib, our program receives as input a dataset of movie reviews. Next, it carries out various preprocessing operations, including selecting tokens, suppressing stop words and feature selection (Unigrams weighted as

TF-IDF), and then performing learning, classification, and computation of performance indicators. Below, the sentiment classification algorithm using naive Bayes.

Algorithm: Sentiment classification using naive Bayes on spark platform

```
// Load trainning data
Dataset movieData = load("c:/data/movie.txt");
// Split the data into tokens
new Tokenizer().transform(movieData)
//Stop words removal
StopWordsRemover().loadDefaultStopWords("english")
remover.transform(movieData).show(false);
// Select unigrams
NGram ngramT = new NGram().setN(1);
ngramT.transform(movieData);
// Set TF-IDF as a Weight scheme
new hashingTF().transform(movieData);
new idf().fit(movieData);
// Split data into training (90%) and test (10%)
splits = movieData.randomSplit(new double[]{0.9, 0.1});
Dataset<Row> nbTrain = splits[0];
Dataset<Row> nbTest = splits[1];
// create the Naive Bayes classifier
NaiveBayes nBayes = new NaiveBayes();
// train the model
NaiveBayesModel nbModel = nBayes.fit(nbTrain);
// Test the model
Dataset<Row> results = nbModel.transform(nbTest);
results.show();
// compute F-measure on the test data
evaluator = new BinaryClassificationEvaluator() ;
Double f1 = evaluator.evaluate(results);
```

4.3. Results and discussion

The designed program provides several statistics measuring the classification F-measure and the processing time according to several parameters such as the dataset size and the number of slave nodes constituting the spark cluster. To evaluate the performance of our algorithms, we use the classification recall, precision and F-measures defined as (3)-(5):

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$F - measure = 2.\frac{Precision\,.Recall}{Precision + Recall} \tag{5}$$

where

- TP (True Positive): correctly classified as positive.
- FP (False Positive): incorrectly classified as positive.
- TN (True Negative): correctly classified as negative.
- FN (False Negative): incorrectly classified as negative.

In Table 1, we find that the classification F-measure of SVM and naive Bayes under spark framework is greater than 84% and consistently exceeds baseline results obtained on a single machine regardless of the dataset size. On the other hand, if the classification process is performed in a single machine configuration, performance is poor from 10k dataset size. In larger sizes, the system fails to complete the learning task and generates an out-of-memory error. Unlike the results achieved by traditional machine learning techniques, naive Bayes is more accurate than SVM when using spark components. Otherwise, we observe that the classification F-measure increases until it stabilizes from dataset sizes greater than 150 k. This is because the model gains enough knowledge from many training examples.

To test our methods' scalability, the time required for both SVM and naive Bayes algorithms executed on three slave nodes to complete preprocessing and learning operations was calculated as illustrated in Figure 2. We deduce that these two algorithms' running time rises proportionally to the dataset size while maintaining better classification performance, confirming that SVM and naive Bayes are scalable machine learning algorithms on spark platform. This is due to spark's capabilities in reducing latency by caching dataset in memory for fast processing and sharing data during iterative computations. Furthermore, if we add nodes to the cluster, we note that the running time decreases considerably. Indeed, the master node distributes data processing between the different slave nodes as illustrated in Figure 3.

Table 1. Sentiment classification F-measure of SVM and naive Bayes under spark platform compared to baseline performance on single machine

	10k	50k	100k	150k	200k
SVM spark	84.79	84.99	85.65	86.88	87.13
SVM Baseline	84.51	-	-	-	-
NB spark	85.58	86.31	86.68	87.51	87.82
NB Baseline	83.39	-	-	-	-

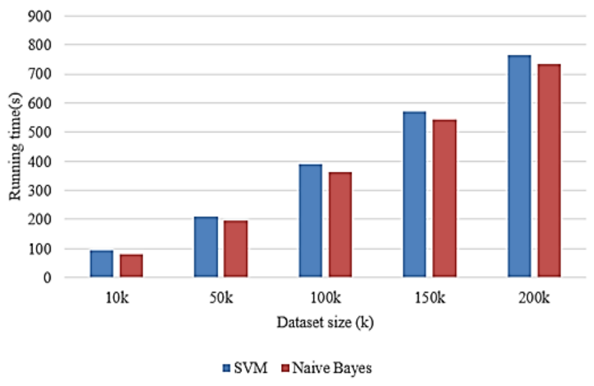

Figure 2. Running time of SVM and naive Bayes algorithms when dataset size increases using 3 slave nodes

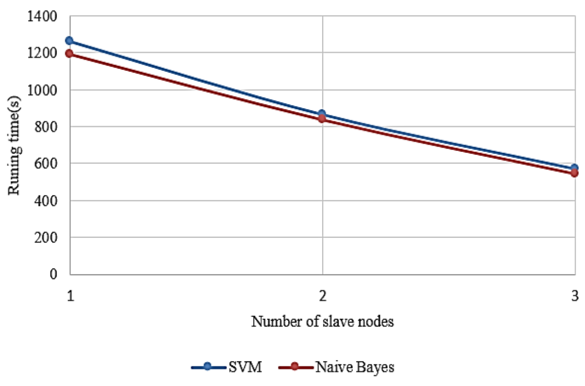

Figure 3. Running time of SVM and naive Bayes algorithms when adding nodes to the spark cluster on 150 k dataset size

5. CONCLUSION

Experiments have shown that machine learning algorithms are very effective in dealing with different issues of sentiment analysis. However they have some weaknesses, among which their inability to scale up when the volume of data increases as in big data context. Through this paper, we conducted a sentiment analysis approach that exploits machine learning components of spark as a big data framework. In our experimental study, we wrote a program based on Apache Spark's machine learning library (MLlib) to observe the behavior of two machine learning algorithms: SVM and naive Bayes for sentiment classification using large training datasets whose size varies between 10 k and 200 k. From the results of our experiments, it appears that the classification performance under spark is much better compared to traditional approaches. Moreover, in terms of scalability, the running time is proportional to the training dataset size. Besides, it has been found that adding slave nodes to the cluster significantly reduces latency. In our future work, we will investigate ways to train classifiers from various heterogeneous data sources.

REFERENCES

[1] E. Park, J. Kang, D. Choi, and J. Han, "Understanding customers' hotel revisiting behaviour: A sentiment analysis of online feedback reviews," *Curr. Issues Tour*, vol. 23, no. 5, pp. 605–611, Mar. 2020, doi: 10.1080/13683500.2018.1549025.

[2] E. Park, Y. Jang, J. Kim, N. J. Jeong, K. Bae, and A. P. del Pobil, "Determinants of customer satisfaction with airline services: An analysis of customer feedback big data," *J. Retail. Consum. Serv*, vol. 51, pp. 186–190, Nov. 2019, doi: 10.1016/j.jretconser.2019.06.009.

[3] E. M. Alshari, A. Azman, S. Doraisamy, N. Mustapha, and M. Alkeshr, "Effective method for sentiment lexical dictionary enrichment based on word2vec for sentiment analysis," in *2018 Fourth International Conference on*

Information Retrieval and Knowledge Management (CAMP), Mar. 2018, pp. 1–5, doi: 10.1109/INFRKM.2018.8464775.

[4] A. Rahman and M. S. Hossen, “Sentiment analysis on movie review data using machine learning approach,” in *2019 International Conference on Bangla Speech and Language Processing (ICBSLP)*, Sep. 2019, pp. 1–4, doi: 10.1109/ICBSLP47725.2019.201470.

[5] O. Kolchyna, T. T. P. Souza, P. Treleaven, and T. Aste, “Twitter sentiment analysis: Lexicon method, machine learning method and their combination,” *ArXiv150700955 Cs Stat*, Sep. 2015.

[6] J. López Belmonte, A. Segura-Robles, A.-J. Moreno-Guerrero, and M. E. Parra-González, “Machine learning and big data in the impact literature. a bibliometric review with scientific mapping in web of science,” *Symmetry*, vol. 12, no. 4, Art. no. 4, Apr. 2020, doi: 10.3390/sym12040495.

[7] I. Lee, “Big data: Dimensions, evolution, impacts, and challenges,” *Bus. Horiz*, vol. 60, no. 3, pp. 293–303, May 2017, doi: 10.1016/j.bushor.2017.01.004.

[8] A. L’Heureux, K. Grolinger, H. F. Elyamany, and M. A. M. Capretz, “Machine learning with big data: Challenges and approaches,” *IEEE Access*, vol. 5, pp. 7776–7797, 2017, doi: 10.1109/ACCESS.2017.2696365.

[9] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? Sentiment classification using machine learning techniques,” in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, Jul. 2002, pp. 79–86.

[10] H. Kim, P. Howland, H. Park, and N. Christianini, “Dimension reduction in text classification with support vector machines,” *J. Mach. Learn. Res.*, vol. 6, no. 1, 2005.

[11] B. Jeong, J. Yoon, and J.-M. Lee, “Social media mining for product planning: A product opportunity mining approach based on topic modeling and sentiment analysis,” *Int. J. Inf. Manag*, vol. 48, pp. 280-290, 2019, doi: 10.1016/j.ijinfomgt.2017.09.009.

[12] D. D. Wu, L. Zheng, and D. L. Olson, “A decision support approach for online stock forum sentiment analysis,” *IEEE Trans. Syst. Man Cybern. Syst*, vol. 44, no. 8, pp. 1077–1087, 2014, doi: 10.1109/TSMC.2013.2295353.

[13] P. Kumari, S. Singh, D. More, D. Talpade, and M. Pathak, “Sentiment analysis of tweets,” *Int. J. Sci. Technol. Eng.*, vol. 1, no. 10, pp. 130–134, 2015, doi: 10.1007/s00484-018-1574-7.

[14] D. Peteiro-Barral and B. Guijarro-Berdiñas, “A survey of methods for distributed machine learning,” *Prog. Artif. Intell.*, vol. 2, no. 1, pp. 1–11, 2013.

[15] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, “A survey on distributed machine learning,” *ACM Comput. Surv. CSUR*, vol. 53, no. 2, pp. 1–33, 2020, doi: 10.1145/3377454.

[16] A. Qiao *et al.*, “Litz: Elastic framework for high-performance distributed machine learning,” *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, 2018, pp. 631-644.

[17] M. Yui and I. Kojima, “A Database-hadoop hybrid approach to scalable machine learning,” in *2013 IEEE International Congress on Big Data*, Jun. 2013, pp. 1–8, doi: 10.1109/BigData.Congress.2013.10.

[18] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein, “Distributed graphLab: A framework for machine learning in the cloud,” *ArXiv12046078 Cs*, Apr. 2012.

[19] K. Luu, C. Zhu, and M. Savvides, “Distributed class dependent feature analysis-A big data approach,” *2014 IEEE Int. Conf. Big Data Big Data*, 2014, doi: 10.1109/BigData.2014.7004233.

[20] M. Assefi, E. Behravesh, G. Liu, and A. P. Tafti, “Big data machine learning using apache spark MLlib,” in *2017 IEEE International Conference on Big Data (Big Data)*, Dec. 2017, pp. 3492–3498, doi: 10.1109/BigData.2017.8258338.

[21] R. Anil *et al.*, “Apache mahout: Machine learning on distributed dataflow systems,” *J. Mach. Learn. Res.*, vol. 21, no. 127, pp. 1–6, 2020.

[22] N. Kourtellis, G. De Francisci Morales, and A. Bifet, “Large-scale learning from data streams with apache SAMOA,” in *Learning from Data Streams in Evolving Environments: Methods and Applications*, M. Sayed-Mouchaweh, Ed. Cham: Springer International Publishing, 2019, pp. 177–207,

[23] J. Huang, J. Lu, and X. Ling, “Comparing naive bayes, decision trees, and SVM with AUC and accuracy,” in *Third IEEE International Conference on Data Mining*, Nov. 2003, pp. 553–556, doi: 10.1109/ICDM.2003.1250975.

[24] ‘SNAP: Web data: Amazon movie reviews. [Online]. Available: https://snap.stanford.edu/data/web-Movies.html (accessed Feb. 07, 2021).

[25] R. Karsi, M. Zaim, and J. El Alami, “Impact of corpus domain for sentiment classification: An evaluation study using supervised machine learning techniques,” in *Journal of Physics: Conference Series*, vol. 870, no. 1, 2017.

[26] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[27] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, “Supervised machine learning: A review of classification techniques,” *Emerg. Artif. Intell. Appl. Comput. Eng.*, vol. 160, no. 1, pp. 3–24, 2007.