AI

MDPI

Article

Filtering Useful App Reviews Using Naïve Bayes—Which Naïve Bayes?

Pouya Ataei [1], Sri Regula [1], Daniel Staegemann [2,*] and Saurabh Malgaonkar [3]

[1] School of Engineering Computer and Mathematical Sciences, Auckland University of Technology, Auckland 1010, New Zealand; pouya.ataei@aut.ac.nz (P.A.); sri.regula@autuni.ac.nz (S.R.)
[2] Faculty of Computer Science, Otto-von-Guericke University Magdeburg, 39106 Magdeburg, Germany
[3] IDEXX, Auckland 1011, New Zealand; saurabhmalgaonkar@gmail.com
* Correspondence: daniel.staegemann@ovgu.de

Abstract: App reviews provide crucial feedback for software maintenance and evolution, but manually extracting useful reviews from vast volumes is time-consuming and challenging. This study investigates the effectiveness of six Naïve Bayes variants for automatically filtering useful app reviews. We evaluated these variants on datasets from five popular apps, comparing their performance in terms of accuracy, precision, recall, F-measure, and processing time. Our results show that Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing performed best overall, achieving up to 89.2% accuracy and 0.89 F-measure. Complement Naïve Bayes with Laplace smoothing demonstrated particular effectiveness for imbalanced datasets. Generally, incorporating Laplace smoothing and Expectation Maximization improved performance, albeit with increased processing time. This study also examined the impact of data imbalance on classification performance. Our findings suggest that these advanced Naïve Bayes variants hold promise for filtering useful app reviews, especially when dealing with limited labeled data or imbalanced datasets. This research contributes to the body of evidence around app review mining and provides insights for enhancing software maintenance and evolution processes.

Keywords: app review classification; naïve bayes variants; text mining; software maintenance; machine learning; expectation maximization; laplace smoothing; imbalanced data; information retrieval; user feedback analysis

Citation: Ataei, P.; Regula, S.; Staegemann, D.; Malgaonkar, S. Filtering Useful App Reviews Using Naïve Bayes—Which Naïve Bayes? *AI* **2024**, *5*, 2237–2259. https://doi.org/10.3390/ai5040110

Academic Editor: Gianni D'Angelo

Received: 16 July 2024
Revised: 23 October 2024
Accepted: 28 October 2024
Published: 5 November 2024

1. Introduction

It is predicted that the app market will be a $200B industry with more than ten million apps hosted on Online Application Distribution Platforms (OADPs) [1]. This is due to rapid increases in the usage and popularity of smart devices worldwide [2]. App developers use relevant OADPs such as Google Play (https://play.google.com/store) or Apple App store (https://www.apple.com/nz/ios/app-store/) to launch their app for end-users to access on their mobile devices. In addition, OADPs provide feedback from end-users in the form of reviews [3].

The majority of feedback points towards requests for new features, bugs, or suggestions for improvements to the app [4], which is useful for software maintenance and product evolution. However, OADPs host many reviews [3], accessible to the public in informing future decisions concerning potential app use. Thus, in meeting the expectations of end-users, app developers benefit if they extract and address the necessary useful reviews reflecting end-users' concerns about their app [5]. Such knowledge greatly helps app developers in their user-driven software quality assessments, marketing, and maintenance processes [3,5]. However, manually obtaining valuable reviews from large review datasets demands substantial levels of cognitive load, effort, and time.

The manual review extraction process also lacks scalability. In fact, the burden of manual review extraction may be compounded due to non-essential information present in

the app reviews [6]. Avoiding non-useful reviews that do not depict app concerns (i.e., non-essential information) is crucial, as such reviews can be misleading to app developers [7]. For instance, consider non-useful reviews such as 'The app is ok!' and 'a good app'. Usually, there are numerous such irrelevant reviews found in the app review repository of an app [6]. App developers must focus on filtering the useful reviews between these inconsequential ones to address the most pressing user concerns. For instance, word cloud analysis of the most frequent words reflecting app concerns mentioned by the end-users can be used. In such analysis, if the non-useful reviews are not removed, the word cloud analysis would be biased towards irrelevant words such as 'app', 'ok', and 'good' over the words reflecting app concerns such as 'inaccurate', 'update', or 'crash'.

Filtering out non-relevant reviews assures the quality of information (i.e., useful reviews) that needs to be manually or automatically processed by app developers to gain actionable knowledge [7]. For instance, with regards to the previously mentioned word cloud analysis, if only useful reviews were extracted and analyzed, then the app developers would be able to achieve a prioritized list of words (i.e., words occurring in descending order of their frequency) that would reflect significant app concerns [8]. Thus, the majority of app developers are shifting towards automated IR approaches for extracting useful reviews [9].

We explored such approaches in this work, where deficiencies were observed [10,11]. Most significant in our observations was that previous approaches, which were designed to extract or filter useful reviews, missed crucial reviews [10]. Further, while the Naïve Bayes method is one of the best suited for software engineering research and applications involving data filtering [12], we have not seen published attempts focused on assessing the performances of particular variants of this method for the filtering of app reviews. To address this gap in knowledge, we formulate the following research questions:

RQ1. *What are the performances of Naïve Bayes variants when extracting useful reviews?*

RQ2. *Are there differences in outcomes for different Naïve Bayes implementations, particularly when considering data imbalances?*

These research questions guide our investigation into the effectiveness of Naïve Bayes variants for filtering useful app reviews. Through this study, we provide contributions to the body of evidence around app review mining and software maintenance. Firstly, we empirically evaluated Naïve Bayes variants and benchmarked their performances, including various measures of accuracy and the time taken for filtering (i.e., via classification) useful reviews. Second, we differentiate useful from non-useful reviews for five datasets, ultimately providing recommendations for the conditions under which various Naïve Bayes variants may be selected for the review extraction process.

Naïve Bayes variants were selected for this study due to their proven effectiveness in software engineering research and applications involving data filtering [12]. Their computational efficiency and ability to handle imbalanced datasets make them particularly suitable for real-time app review analysis. Overall, our contributions provide insights (and recommendations) for an important software engineering problem.

The remaining sections of this paper are organized as follows. Section 2 presents studies concerning the mining of valuable reviews. Section 3 outlines the methods and ideas that help in creating the versions of the Naïve Bayes experimental setup for the evaluation of the variants of Naïve Bayes. Section 4 provides the results of our experiments. We document our discussions and the implications of our findings in Section 5 before considering this study's limitations in Section 6. Finally, we present concluding remarks in Section 7.

2. Related Work

App reviews expressed in the form of natural language are a common mechanism for gathering end-user feedback [3] for software maintenance and evolution after apps are released online [5]. Due to the nature of app reviews, traditional information retrieval

approaches lack the ability to conduct filtering based on the contextual meaning of the review contents [4]. Keertipati et al. [10] have identified features from filtered reviews with ratings < 3, thus missing out on the features requiring attention that were mentioned in reviews with higher ratings. Similarly, Fu et al. [11] conducted sentiment analysis with logistic regression to extract the reviews reflecting negative end-user sentiments assuming that negative reviews indicate serious app problems, missing out on useful positive reviews. In a different study, Shah et al. [13] assessed the Bag-of-Words (BoW) approach against Convolutional Neural Network (CNN) for extracting app features and found the former approach to perform better. However, given that BoW is a simple approach, it tends to overfit the learning data [14].

Similarly, Johann et al. [15] have utilized the parts of the speech pattern evaluation approach to identify and extract app features. However, this approach requires manual efforts to extract app features after the parts of speech pattern evaluation have been initiated. Gao et al.'s [16] work highlights some of the disadvantages of various techniques such as Pointwise Mutual Information (PMI), Adaptively Online Latent Dirichlet Allocation (OLDA), and Anomaly Discovery (AD). For instance, PMI is assessed as highly biased towards infrequent content expressed in the reviews, the absence of discriminatory information along with generally large sample sizes of reviews affect the performance of OLDA, and the complexity of the AD method makes it difficult to identify the appropriate threshold parameters necessary for tuning this method to produce accurate results. Furthermore, AD often frequently generates false positive results [16]. Nevertheless, it is to be noted that app developers usually prefer the full form of useful reviews over specific app features as these reviews portray detailed information related to requests, bugs, or recommendations related to the app features [17] (e.g., description of what is wrong with the feature).

Furthermore, the IR approaches mentioned above miss out on crucial information or capture unwanted information that reflect irrelevant or noisy data [18]. For example, consider the useful review that is filtered (extracted) on the basis of a lower rating (<3) and negative sentiment, "(i) *Very angry, this app is useless, uninstalling, will try in my next life perhaps lol!!*", and another review labeled non-useful by the filtering process because of its higher rating (>3), "(ii) *Great app, works fine but the user interface appears broken at Home Page on Nexus 7* ". Review (i) may be termed futile by app developers, as it does not provide any useful information that may lead to app improvement (i.e., an actionable insight). However, review (ii) may lead to the fixing of a user interface issue, which would be useful to app developers. Therein lies the challenge of discriminating between useful and non-useful reviews based on such subjectivities.

Certain research studies from the app domain have utilized classification as an approach to extract app reviews of interest (i.e., useful reviews) to address the above-mentioned challenge. This method groups app reviews with similar attributes into distinct categories (e.g., Pricing, Rating, and so on) based on a manually derived taxonomy from domain expertise, as the literature review shows all classification methods for app reviews depend on domain knowledge obtained through extensive research or domain experts. For instance, Panichella et al. [19] adopted r by domain experts. For instance, Panichella et al. [19] have inherited a taxonomy from the taxonomy proposed by Pagano et al. [4] and have evaluated the classification performance of SVM (Support Vector Machines), Naïve Bayes, Decision Trees, and Logistic Regression. Pagano et al. [4] manually created categories forming a taxonomy for classifying app reviews. Similarly, Maalej et al. [9] manually created four categories for app review classification using methods such as keyword lookup, Decision Trees, Naïve Bayes, and Maximum Entropy.

Beyond the approaches mentioned above, rule-based linguistic approaches are assessed as valuable for filtering useful reviews. For instance, Iacob et al. [20] identified a set of language rules to extract app feature requests from reviews. Similarly, Sutino et al. [21] have come up with extraction rules that are based on different concepts of similarity to extract app features. However, the rule-based extraction approaches are only limited to app features (excluding bugs and suggestions for improvements). Hence, rule-based

approaches similar to these are combined with suitable machine learning methods to address such drawbacks, and may help with scalability challenges. For example, Huang et al. [22] have developed a probabilistic classifier that learns from a training set of manually pre-labeled requirements to predict suitable labels (i.e., availability, look and feel, legal, maintainability, operational, performance, scalability, security, and usability) of the remaining set of non-functional requirements.

In recent research, Panichella et al. [23] have developed a tool named '*Requirements-Collector*', which automates the task of requirements specification and user feedback analysis through means of classification using a predefined taxonomy that was manually derived. The authors have utilized and evaluated the performance of machine learning (Sequential Minimal Optimization, F-measure: 0.77) and deep learning (F-measure: 0.33) methods toward automation of tasks. However, the machine learning or deep learning approach that is used here (similar to some others) requires a vast amount of pre-classified training data to attain substantial levels of prediction accuracy [24].

Nonetheless, Multinomial Naïve Bayes is a well-known supervised machine learning method empirically proven to be suitable for text-related software engineering applications, as it operates with the knowledge of word frequency information extracted from a text corpus [25,26]. Accordingly, this method significantly outperforms other machine learning methods [25]. We reviewed the Naïve Bayes techniques and principles specialized in text classification operations [27,28], identifying six variants that have potential for filtering useful reviews.

However, these variants have not been investigated for their utility in extracting app reviews, a gap that needs to be addressed in supporting the software engineering community's maintenance and evolution efforts. Hence, in this study, we formulate the design and configuration of basic Naïve Bayes variants that are specialized in text classification. We then utilize Laplace Smoothing and Expectation Maximization to develop additional variations of the Naïve Bayes method prime objective of examining the Naïve Bayes variants proposed in this research is to help app developers in the accurate extraction of valuable reviews for software maintenance and development and extend academic knowledge around the application of IR approaches in software engineering.

The performance of app review filtering methods is of prime importance to app developers in their drive to target the correct and most pressing app maintenance and evolution tasks [29]. To measure the performance (RQ1) of Naïve Bayes variants, we utilize accuracy, precision, recall, and F-measure metrics [24,30]. In addition to these metrics, we also examine the time taken by each variant for learning and prediction purposes [24]. Since app developers must promptly address valuable reviews, time becomes a crucial factor in the assessment of information retrieval methods. We use various statistical procedures to examine differences (RQ2), while controlling for data imbalances in our datasets.

It is to be noted that the studies reviewed in this Section have various experimental setups (i.e., research methods, data for testing, validation procedures, and outcomes), and have used non-identical metrics when evaluations were performed. For example, Keertipati et al. [10] have used the rating as a criterion to filter reviews for prioritizing app features in those reviews but have not reported accuracy and time statistics for their filtering approach. In another study, recall and Matthews Coefficient Constant (MCC) metrics were used to validate the filtering approach, which was restricted to specific app features (i.e., unigrams of interest) and not to entire app reviews [11]. Similarly, some studies are confined to the identification of functional and non-functional requirements, where each review is assigned one out of many labels (i.e., non-binary classification) through means of classification approaches. Such works provide limited details on accuracy and time metrics [16–19]. However, the differences in outcomes reported for these works are fitting, given the differences in the objectives and experimental settings.

3. Methods and Concepts

This section introduces the methods and ideas that helped us create the respective variants of Naïve Bayes. The main goal of the variants is to automatically filter (through classification) valuable and irrelevant reviews present in a vast app reviews corpus expressed in natural language.

An initial collection of valuable and irrelevant reviews can be manually recognized using a predefined set of filtering rules proposed in [12]. Rules pertaining to valuable reviews reflect feature requests (e.g., 'please add feature A'), issues or bugs related to the app (for example, 'the app crashes at the checkout screen'), or suggestions for app improvements (for example, 'I suggest you increase the font size for a better view'). On the other hand, irrelevant reviews contain unnecessary and unwanted information (e.g., 'this app is useless, uninstalling asap!'). Once the specific variant of Naïve Bayes has been trained, it can separate valuable reviews from irrelevant reviews by classifying each review into the appropriate category. Thus, for the given problem of classifying useful and non-useful reviews, the purpose of the specific Naïve Bayes variant is to allocate a set of reviews to one of the two defined categories (useful and non-useful reviews, wherein each category is expected to contain reviews with properties reflecting the filtering rules).

In the learning (training) phase, the Naïve Bayes variant is utilized to generate a classifier that forecasts the categories of new reviews during the classification (prediction/testing) phase. In the following sub-sections, we describe how unstructured reviews are transformed into suitable vocabulary to be used as input for various Naïve Bayes variants through the application of text pre-processing. Then, an overview of Naïve Bayes machine learning methods is provided. This is followed by the ideas of Laplace Smoothing and Expectation Maximization. It is worth noting that these are well-known concepts in the machine learning domain that are used as part of information retrieval approaches in software engineering applications (e.g., [31]).

3.1. Pre-Processing of Reviews

Initially, several text pre-processing steps are followed to convert reviews into subsequent word vectors [32]. We perform review pre-processing by eliminating whitespaces, numbers, special characters (e.g., $, #), and punctuation marks (e.g., !, ?) present in the reviews before converting them into lowercase [33]. Finally, we perform the removal of stop words (e.g., is, and) followed by lemmatization of the pre-processed reviews to create the full dictionary form of words in the pre-processed reviews [34]. The aforementioned steps are typical text preprocessing methods used by researchers to avoid the creation of unreliable noisy outcomes and, at the same time, shortlist the reliable features (words) for learning and prediction purposes [9]. For instance, the stop words elimination process removes the most common insignificant words such as 'the', 'a', 'on', 'is', and so on that do not reflect unique information that can be used by any machine learning algorithm [33]. Finally, these pre-processed reviews form the Vocabulary (V) that gives essential word frequency data for the Naïve Bayes variants [27].

3.2. Multinomial Naïve Bayes

Multinomial Naïve Bayes is a customized version of the basic Naïve Bayes method, which is specialized for text classification [27]. This method works on the principle of maximum likelihood estimates. This means it uses the information on word frequencies extracted from a text corpus for the required learning and prediction tasks. For the given problem statement, the objective of the Multinomial Naïve Bayes is to compute the probability of a review belonging to a particular category (c_n), which is given as:

$$P\ (c_n) = N_{reviews}\ (r = c_n)/N_{reviews} \quad (1)$$

$N_{reviews}$ indicates the number of reviews present in the app reviews corpus, and $N_{reviews}$ (r = c_n) shows the number of reviews in a category c_n. The maximum likelihood estimation is given as:

$$P(w_i \mid c_n) = \text{count}(w_i, c_n) / \sum_{w \in V} \text{count}(w, c_n) \quad (2)$$

$P(w_i \mid c_n)$ denotes the conditional probability of the word w_i given the probability of category c_n that is expressed as the ratio of the total occurrences of a word w_i in category c_n to the total words w in the reviews of category c_n. Thus, the fraction of times word w_i appears among all words (V) in the reviews of category c_n. Thus, Multinomial Naïve Bayes constructs a word space for category c_n by forming a dictionary of words from the reviews of category c_n by utilizing the frequency of each word w. Finally, using Equations (1) and (2), the category of a review R can be determined using:

$$C_{MAP}(R) = \text{argmax}_{cn}(P(c_n) * \Pi_i P(w_i \mid c_n)) \quad (3)$$

C_{MAP} (R) denotes the most probable category termed as maximum a posteriori (MAP), i.e., most likely category c_n for a review R, which is given as the arguments of the maxima over all the categories of the priori times the likelihood. Based on this, we provide the learning phase for Multinomial Naïve Bayes for classifying app reviews into relevant categories [24] in Algorithm 1.

Algorithm 1. Learning phase of Multinomial Naïve Bayes for performing predictions

Begin
1. From the manually classified pre-processed app reviews, extract Vocabulary (V)
2. Calculate P(c_n) terms
   2.1 For each c_n in C do:
      2.1.1 $reviews_n \leftarrow$ all reviews with category = c_n
      2.1.2 P(c_n) $\leftarrow$ $\mid reviews_n \mid$ / $\mid$ Total reviews $\mid$
3. For every word w_i, given every category c_n
   3.1 Calculate P($w_i \mid c_n$) (maximum likelihood estimates)
      3.1.1 Word $space_n \leftarrow$ words belonging to $reviews_n$
      3.1.2 For each word w_i in the Vocabulary (V)
         3.1.2.1 $n_i \leftarrow$ Total occurrences of w_i in Word $space_n$ consisting of a total of n words
         3.1.2.2 P($w_i \mid c_n$) $\leftarrow n_i/n$
End

3.3. Complement Naïve Bayes

In this sub-section, we discuss the Complement Naïve Bayes, which is a modified version of the Multinomial Naïve Bayes. Complement Naïve Bayes deals with the limitations of Multinomial Naïve Bayes's to perform well when trained with imbalanced data [35], i.e., the training data do not comprise of approximately equal proportion of reviews belonging to different types of categories. Complement Naïve Bayes addresses this drawback by estimating the probability of a category c_n using training data from the other category(ies) $\overline{c_n}$ (except c_n). In the case of Complement Naïve Bayes, the initial probability is computed using Equation (1). Dissimilar to Multinomial Naïve Bayes, Complement Naïve Bayes computes the likelihood of a word w_i by considering its occurrences in categories $\overline{c_n}$ other than c_n (i.e., computing the likelihood of w_i occurring in other category(ies)). Hence, the maximum likelihood estimation is given as:

$$P(w_i \mid \overline{c_n}) = \text{count}(w_i, \overline{c_n}) / \sum_{w \in V} \text{count}(w, \overline{c_n}) \quad (4)$$

$P(w_i \mid \overline{c_n})$ denotes the probability of the word w_i given it belongs to category(ies) $\overline{c_n}$. It is given as the ratio of the total number of times a word w_i occurs in category(ies) $\overline{c_n}$ to the total number of words w in the reviews of category(ies) $\overline{c_n}$. Thus, in contrast to Multinomial Naïve Bayes, the Complement Naïve Bayes creates a word space for category

c_n by creating a dictionary of words belonging to the reviews of category(ies) $\overline{c_n}$ by utilizing the frequency of w. Finally, using Equations (1) and (4), the category of a review R can be determined using:

$$C_{MAP}\ (R) = \text{argmin}_{cn}\ (P\ (c_n)\ *\ \Pi_i\ (1/(P\ (w_i|\overline{c_n}))))\quad (5)$$

C_{MAP} (R) denotes the most likely category c_n for a review R, which is given as the argument of the minimum likelihood estimates of the category calculated as prior times the inverse likelihood. Based on this, we provide the learning phase for Complement Naïve Bayes for classifying the app reviews into the relevant categories [35] in Algorithm 2.

Algorithm 2. Learning phase of Complement Naïve Bayes for performing predictions

Begin
1. From the manually classified pre-processed app reviews, extract Vocabulary (V)
2. Calculate P (c_n) terms
   2.1 For each c_n in C do:
      2.1.1 $reviews_n$ ← all reviews with category = c_n
      2.1.2 P (c_n) ← $|reviews_n| / |$ Total reviews $|$
3. For every word w_i, given every category c_n
   3.1 Calculate P (w_i-$|\overline{c_n}$) (maximum likelihood estimates)
      3.1.1 Word $space_n$ ← words belonging to reviews of category(ies) $\overline{c_n}$
      3.1.2 For each word w_i in the Vocabulary (V)
         3.1.2.1 n_i ← Total occurrences of w_i in Word $space_n$ consisting of a total of *n* words
         3.1.2.2 P ($w_i | \overline{c_n}$) ← n_i/n

End

3.4. Laplace Smoothing

From Equations (2) and (4), it is evident that the parameters that generate the maximum likelihood estimate are unable to handle any zero probabilities effectively [36]. For example, if a word has not been observed in the learning phase, both Naïve Bayes (Multinomial and Complement) methods would generate a zero-probability value for that word, which subsequently impacts classification accuracy. This issue is addressed by applying Laplace Smoothing to the parameters [28,37], which instructs the parameters to add 1 to handle the zero counts of words efficiently, thus allowing the particular Naïve Bayes method to monitor the word count in identifying the relevant category. Therefore, such a strategy is crucial, particularly when the specific Naïve Bayes method encounters a word during the classification phase (prediction/testing) that was not present during the learning (training) phase. Thus, we modify the parameters of the Multinomial and Complement Naïve Bayes methods that perform the maximum likelihood estimation to incorporate the Laplace smoothing functionality for handling information related to missing word w_i. For the Multinomial Naïve Bayes method, using Equation (2), we generate its new parameter that performs maximum likelihood estimation based on Laplace smoothing, given as:

$$P\ (w_i\ |\ c_n) = (\text{count}\ (w_i, c_n) + 1)/(\textstyle\sum_{w\in V}\ (\text{count}(w,c_n) + |V|))\quad (6)$$

Similarly, for Complement Naïve Bayes, using Equation (4), we generate its new parameter that performs maximum likelihood estimation using Laplace smoothing, given as:

$$P\ (w_i|\overline{c_n}) = (\text{count}\ (w_i, \overline{c_n}) + 1)/(\textstyle\sum_{w\in V}\ (\text{count}\ (w, \overline{c_n})\ +\ |V|))\quad (7)$$

It is to be noted that, as a count of one has been added to the numerator, the size of the vocabulary ($|V|$) is added to the denominator, indicating the addition of one for every vocabulary word in the denominator. Based on Equations (6) and (7), the learning phases of Multinomial Naïve Bayes (refer to Section 3.2) and Complement Naïve Bayes (refer to Section 3.3) can be updated accordingly.

3.5. Expectation Maximization

Both methods highlighted in Sections 3.2 and 3.3 are supervised learning methods and thus need a significant amount of manually labeled reviews to train a classifier that can accurately predict the category of a new review. Accordingly, manually labeling (categorizing) adequate amounts of reviews can be a time-consuming task prone to potential errors, as it must be manually performed by app developers. Semi-supervised learning approaches assist in addressing this drawback by lessening the labeling effort required from app developers. One of the common semi-supervised learning concepts comprises learning from labeled as well as unlabeled information, and Expectation Maximization (EM) is an example of one such concept [38,39]. EM primarily consists of two steps: Expectation (E) and Maximization (M). The Expectation step predicts and generates the missing information based on the current maximum likelihood estimation parameters set by the method in question (Multinomial Naïve Bayes), while the Maximization step iteratively recalculates the parameters, thereby maximizing the overall likelihood [40].

Hence, EM enables the Multinomial Naïve Bayes method to run repeatedly until the parameters that estimate the total likelihood become constant [41]. We utilize the EM strategy to develop the semi-supervised version of the Multinomial Naïve Bayes method discussed in Sections 3.2 and 3.4. The EM concept for this study was devised based on the algorithm mentioned in [40,41]. The primary steps of EM involve training the Multinomial Naïve Bayes method on known review categories and then using the learned information to predict the categories of uncategorized reviews. Hence, these predictions can later be transformed into categories and, therefore, can be utilized for subsequent training of the Multinomial Naïve Bayes method using the uncategorized reviews with the previously generated categories.

Finally, the entire procedure is repeated until the value of the Multinomial Naïve Bayes method's total likelihood stabilizes (likelihood is calculated using the whole collection of app reviews). The detailed explanation of the process mentioned above is as follows: consider app reviews set AR consisting of reviews where each review R is labeled with a category C (useful or non-useful). The main goal of EM is to determine the categories of uncategorized reviews using the Multinomial Naïve Bayes method's prediction mechanism. In every cycle, EM calculates the relevant probabilistic category and assigns it to the uncategorized review, that is $P(c_n \mid R_i)$, which is estimated to be 0 or 1. Here, c_n denotes the category, and R_i indicates the particular review. The categorized reviews having a specific category (x) are known prior; hence, $P(c_x \mid R_i) = 1$ and $P(c_y \mid R_i) = 0$ for $x \neq y$. Using the information of categorized reviews and $P(c_n \mid R_i)$, a new version of the Multinomial Naïve Bayes classifier is generated, which works in a recurring fashion until $P(w_i \mid c_n)$ and $P(c_n)$ become constant. We provide the pseudo-code of the EM concept [41] in Algorithm 3.

Algorithm 3. Expectation Maximization concept for semi-supervised learning

Begin
1. Train the Multinomial Naïve Bayes method mNB using the manually categorized set of reviews R.
2. Expectation (E):
   2.1 For each review Ri in the review set AR
      2.1.1 Using the method mNB, calculate $P(c_n \mid Ri)$
3. Maximization (M):
   3.1 Train an updated version of mNB from $R \cup AR$ by calculating $P(c_n)$ and $P(w_i \mid c_n)$
4. Repeat steps 2 and 3 until mNB's parameters (maximum likelihood estimators) become constant.
5. Return mNB after completion of step 4.

End

That said, as the Complement Naïve Bayes method does not allow for generative interpretations, hence creating its EM variant is not feasible [35].

3.6. Summary of Naïve Bayes Variants

In this subsection, we review the Naïve Bayes variants because of the methods (refer to Sections 3.2 and 3.3) and concepts (refer to Sections 3.4 and 3.5) that were documented prior. Table 1 provides an overview of the particular Naïve Bayes variants. The main goal in developing these variants is to investigate their performance related to the prediction of review categories for a set of reviews related to an app. To begin, we first formulate the Naïve Bayes variants of the Multinomial Naïve Bayes method. Based on the method described in Section 3.2 and the concepts discussed in Sections 3.4 and 3.5, there are four possible variants concerning the Multinomial Naïve Bayes method. We present the first Naïve Bayes variant (I) that incorporates the functionality of the Multinomial Naïve Bayes method discussed in Section 3.2.

Table 1. Naïve Bayes variants for experimental evaluation.

Variant	Name	Description
I	Multinomial Naïve Bayes	This variant is the Multinomial Naïve Bayes method described in Section 3.2.
II	Expectation Maximization—Multinomial Naïve Bayes	The Expectation Maximization concept described in Section 3.5 has been incorporated in I. Thus, this variant is the semi-supervised version of I.
III	Multinomial Naïve Bayes with Laplace smoothing	The Multinomial Naïve Bayes method has been incorporated with the concept of Laplace smoothing as described in Section 3.4. Thus, this variant is the post version of I.
IV	Expectation Maximization—Multinomial Naïve Bayes with Laplace smoothing	The Multinomial Naïve Bayes method has been incorporated with the concept of Laplace smoothing as well as Expectation Maximization. Thus, this variant is the semi-supervised version of III and the post version of II.
V	Complement Naïve Bayes	This variant is the Complement Naïve Bayes method described in Section 3.3.
VI	Complement Naïve Bayes with Laplace smoothing	The Complement Naïve Bayes method has been incorporated with the concept of Laplace smoothing. Thus, this variant is the post version of V.

Next, as the EM mechanism enables the Multinomial Naïve Bayes method to deal with unlabeled reviews, we generate the second variant (II) of Naïve Bayes, which is a semi-supervised version of I. Third, based on Sections 3.2 and 3.4, we introduce the third variant (III) that incorporates Laplace Smoothing with the Multinomial Naïve Bayes method, thus making it a post (i.e., extended) version of I. Finally, we generate the fourth variant (IV) that incorporates the EM mechanism in III, thus making IV the semi-supervised version of III and a subsequent version of II. Next, we highlight the variants of the Complement Naïve Bayes method. Based on the method described in Section 3.3, we develop the Naïve Bayes variant (V) that implements the functionality of the Complement Naïve Bayes method. Next, based on Sections 3.3 and 3.4, we introduce the variant (VI), which incorporates Laplace smoothing in V, thus making VI a post version of V.

3.7. Experimental Setting

In this study, the Naïve Bayes variants described in Table 1 were implemented using the Python (https://www.python.org/) programming language with suitable libraries provided by the Natural Language Tool Kit (NLTK) (https://www.nltk.org/) numpy (https://numpy.org/) and the scikit-learn (https://scikit-learn.org/stable/) packages.

Python and its suitable libraries allow researchers to develop complex algorithms efficiently, as programming in Python is easy to understand and implement. Additionally, Python is flexible as it can be integrated with other languages and has extensive community support. Our own implementation was used to carry out an experimental evaluation of all six Naïve Bayes variants on datasets comprising app reviews belonging to five different apps obtained from the Google Play Store (i.e., public software repository). These datasets belonged to TradeMe, MyTracks, VodafoneNZ, ThreeNow, and Flutter apps. These five datasets belonging to the popular categories of Google Play Store were selected to demonstrate the general applicability of the proposed filtering approach (refer to Appendix A Table A1 for more details on these datasets) [25,42]. All datasets consisted of reviews submitted by end-users written in natural language. TradeMe consisted of 4559 reviews, MyTracks dataset included 4003 reviews, VodafoneNZ had 6583, ThreeNow consisted of 3683 reviews, and the Flutter dataset consisted of 3483 reviews.

Using the set of rules defined in [12] (refer to Section 3), app reviews from the datasets were manually categorized through labeling before reliability checks were performed [43]. The labels associated with the app reviews indicated whether the particular app review was useful or non-useful. The inherited rules associated with the specific label are described in Table 2. Here, the first column indicates the label, the second column indicates the rules associated with the particular label and the third column shows the examples of app reviews that are covered by the relevant rule.

Table 2. Rules for Manually Tagging App Reviews as Valuable or Irrelevant.

Label	Rule	App Review Examples
Useful	Request • Requests to add or modify features • Request to remove advertisements or notifications	a. Please make the user interface more friendly and simple. b. The advertisements play continuously, they need to be stopped. c. I need a feature to compare several products.
	Bug • Bug that generates incorrect or unexpected output • Bug that affects the performance of the app • Bug that causes app failure	a. The app lags a lot after new update and does not respond to many touch inputs! b. The images of the products fail to load on the main screen. c. The app crashes after the butterflies and forest comes on the screen, poor job by app developers!
	Suggestion • Suggestions that indicate a need for app improvement	a. I wish there were more skins to choose from. b. Suggestion to include a $5 free voucher add-on. c. The app interface would look great in a black and white theme.
Non-Useful	Irrelevant and unwanted information	a. This app is useless. b. I have changed my rating from 4 to 2 star. c. This app is great, I love it!

Following the suggested validation practices of the software engineering field, this task was undertaken to empirically evaluate the performance of six Naïve Bayes variants based on human judgments and evaluations [43,44]. A cross-validation (i.e., comparison of results generated from human decisions with the results generated by the respective Naïve Bayes variant) approach is deemed reliable, and the human feedback in such cases

acts as the concrete ground truth [43,44]. Based on the manual labeling task and after performing the necessary reliability assessments, the TradeMe dataset contained 1154 (25%) valuable reviews and 3405 (75%) irrelevant reviews, making it imbalanced (imbalance scale: 0.7) [45,46]. MyTracks dataset included 1638 (41%) valuable reviews and 2365 (59%) irrelevant reviews (imbalance scale: 0.3), whereas VodafoneNZ consisted of 1120 (17%) valuable reviews and 5463 (83%) irrelevant reviews, making it imbalanced (imbalance scale: 0.8) [47]. ThreeNow consisted of 1760 (48%) useful reviews and 1923 (52%) non-useful reviews (imbalance scale: 0.1), and the Flutter dataset included 2433 (70%) valuable reviews and 1063 (30%) irrelevant reviews, making it imbalanced (imbalance scale: 0.7) [45,46]. It is to be noted that we followed the guidelines mentioned in [45,46] to derive the necessary imbalance scales. The measure of the imbalance scale is holistic in terms of the imbalanced categories (i.e., balance to imbalance or vice-versa). It is calculated based on the number of reviews in both classes and not as a percentage (%). For instance, if app 1 has 30 non-useful reviews and 70 useful reviews, the imbalance scale is calculated as $1 - (30/70) = 0.67$ (which is rounded to 0.7). If app 2 has 9 non-useful reviews and 91 useful reviews, then the imbalance scale is computed as $1 - (9/91) = 0.9$. Similarly, if app 3 has 23 non-useful reviews and 32 useful reviews, then the imbalance scale is computed as $1 - (23/32) = 0.3$. As can be seen from the examples, the higher imbalance scale represents the dominating class (i.e., useful or non-useful). For approximately equally proportionate cases, the imbalance scale is lower. Hence, values closer to 0 indicate a lower imbalance and values closer to 1 indicate a larger imbalance.

Of note is that these app reviews were independently labeled valuable or irrelevant by the three authors (refer to Table 2 and [12] for rules). To perform the reliability assessments, we utilized Fleiss' Kappa, an extended version of Cohen's Kappa, which supports the evaluations of three or more human assessors [48]. The Fleiss coefficients were 0.68 (substantial agreement), 0.74 (substantial agreement), 0.71 (substantial agreement), 0.65 (substantial agreement), and 0.78 (substantial agreement) for TradeMe, MyTracks, VodafoneNZ, ThreeNow, and Flutter datasets, respectively [49]. Follow-up discussions were conducted among the authors to resolve any disagreements and reach a consensus for a reliable manual labeling process that led to 100% agreement.

The objective of app review classification using the specific Naïve Bayes variant is to accurately identify the type of each review, i.e., to predict the label—useful or non-useful. As mentioned above, the performance results of the classification task were evaluated using the standard definitions of accuracy, recall, precision, F-measure, and time metrics. Accuracy, defined as a metric, determines the correctness of the particular Naïve Bayes, defined as the number of correctly classified reviews out of the total classified reviews. In the field of machine learning the accuracy metric is interpreted as the sum of true positives and true negatives to the total number of entries [30]. Next, we evaluate the precision metric, which indicates the true positives to the total number of true positives and false positives [30]. Recall is defined as the correctly classified valuable reviews to the total number of reviews that were useful. Therefore, recall indicates the true positives to the total number of true positives and false negatives [30]. Finally, F-measure is calculated as the harmonic mean of precision and recall, which confirms the robustness of the variants [30]. Furthermore, the time metric measures the time (in seconds) required for a particular Naïve Bayes variant to learn from a set of manually labeled reviews (training data) to predict the category of unknown reviews (test data) [50]. The computer used for our experiments had 14 GB RAM and a CORE i5 CPU. For each experiment, we randomly divided the respective dataset into a training set (90%) used to train the relevant Naïve Bayes variant for reviews and a testing set (10%) used to evaluate their performance in classifying undisclosed reviews. Each experiment was conducted 100 times using ten-fold cross-validation (i.e., $k = 10$) to obtain average scores for the metrics mentioned above [51]. This process is commonly used by researchers to verify the stability of the methods [52]. That said, the same pattern of results was observed for every execution of our algorithms

(all 100), and thus, even a single- or ten-times execution of our methods would support our stated conclusions.

We measure the performances of the various Naïve Bayes approaches to answer RQ1. We then experimented with the different Naïve Bayes implementations, particularly considering data imbalances when answering RQ2. These results are provided in the next section.

4. Results

RQ1. *What are the performances of Naïve Bayes variants when extracting useful reviews?*

We display the results of the experiments performed on the five datasets in Table 3, wherein we present the average results of 100 ten-fold cross-validation operations performed on the TradeMe, MyTracks, Vodafone NZ, ThreeNow, and Flutter datasets according to the metrics listed in Section 4. It is to be noted that in examining statistically significant differences among our outcomes, we ran the Shapiro–Wilk test to examine the distribution of the results produced by each Naïve Bayes variant for normality assumption [53], finding no evidence confirming normality (p-value < 0.01). Consequently, we conducted the Kruskal–Wallis non-parametric test to detect potential statistically significant differences between the results of the Naïve Bayes variants [53]. On finding statistically significant differences (p-value < 0.01), pairwise Wilcoxon testing was carried out to assess pairwise comparisons, with corrections for multiple comparisons [54], finding statistically significant differences for all comparisons (p-value < 0.01).

That said, diverse performances shown by the Naïve Bayes variants can be observed in Table 3. Initially, we tested the six Naïve Bayes variants on the TradeMe dataset and evaluated their performances accordingly. Overall, Variant I showed the lowest accuracy (59.3%) and F-measure (0.57) compared to others, while VI demonstrated the highest accuracy (80.2%) and F-measure (0.65). Variant VI also needed the least amount of time for learning and prediction (0.10 s), whereas Variant II took the most time (0.29 s). Next, we tested the six variants on the MyTracks dataset and evaluated their performances accordingly. Overall, Variant I had the lowest accuracy (68.1%) and F-measure (0.71) compared to others, while Variant IV exhibited the highest accuracy (89.2%) and F-measure (0.89). That said, Variant VI required the least time for learning and prediction purposes (0.10 s), while Variant II required the most time (0.30 s).

Similarly, we tested the six variants on the Vodafone NZ dataset and evaluated their performances accordingly. Overall, Variant I showed the lowest accuracy and F-measure (56.9% and 0.43, respectively), while Variant VI was seen to exhibit the highest accuracy and F-measure (79.6% and 0.58, respectively) while also taking the least time (0.17 s). We observe that Variant II required the most time (0.40 s), and Variant IV was noted to be the second highest in terms of its performance based on accuracy (78.2%) and F-measure (0.55). Further, based on the observations in Table 3, the results for Variants II and V show substantial differences in accuracy magnitude (even though differences were statistically significant $p < 0.01$). In following the trend of analyses above, we tested the six variants on the ThreeNow dataset and evaluated their performances accordingly. Overall, Variant I showed the lowest accuracy (60.2%) and F-measure (0.72) compared to others, while Variant IV showed the highest accuracy (78.2%) and F-measure (0.81). However, Variant VI required the least time requirements (0.11 s), whereas Variant II needed more time than others (0.28 s).

Finally, we tested the six Naïve Bayes variants on the Flutter dataset and evaluated their performances accordingly. Overall, Variant I displayed the lowest accuracy (76.2%), whereas VI was seen to exhibit the highest F-measure (0.89) with the least time (0.08 s) requirement. We observe that Variant II had the highest time requirement (0.23 s), and Variant IV was noted to be the second highest in terms of its performance based on the accuracy (82.3%) and F-measure (0.88). That said, based on the observations in Table 3, the results

for Variants II, III, and V did not show substantial differences in accuracy and F-measure magnitude (although these differences were statistically significant, p-value < 0.01).

Table 3. Naïve Bayes Variants' Performance on Five Datasets.

Dataset	Category Labels	Imbalance Scale (0–1)	Variant	Accuracy (%)	Precision (0–1)	Recall (0–1)	F (0–1)	Time (Seconds)
TradeMe	Imbalanced	0.7	I	59.3	0.40	**0.98**	0.57	0.23
			II	74.9	0.58	0.62	0.60	0.29
			III	78.1	0.64	0.63	0.63	0.14
			IV	79.0	**0.65**	0.64	0.64	0.17
			V	74.1	0.54	0.71	0.61	0.12
			VI	**80.2**	0.56	0.78	**0.65**	**0.10**
MyTracks	Balanced	0.3	I	68.1	0.56	**0.98**	0.71	0.26
			II	80.4	0.73	0.88	0.80	0.30
			III	87.4	0.81	0.91	0.86	0.12
			IV	**89.2**	**0.84**	0.94	**0.89**	0.19
			V	84.6	0.76	0.90	0.82	0.15
			VI	86.5	0.78	0.91	0.84	**0.10**
Vodafone NZ	Imbalanced	0.8	I	56.9	0.28	**0.93**	0.43	0.32
			II	75.1	0.52	0.41	0.46	0.40
			III	76.6	0.72	0.39	0.51	0.23
			IV	78.2	**0.75**	0.43	0.55	0.29
			V	75.2	0.43	0.57	0.49	0.20
			VI	**79.6**	0.53	0.63	**0.58**	**0.17**
ThreeNow	Balanced	0.1	I	60.2	0.57	**0.97**	0.72	0.24
			II	71.1	0.71	0.76	0.74	0.28
			III	74.1	0.74	0.83	0.78	0.16
			IV	**78.2**	**0.77**	0.86	**0.81**	0.19
			V	69.6	0.70	0.75	0.73	0.14
			VI	72.2	0.73	0.80	0.76	**0.11**
Flutter	Imbalanced	0.7	I	76.2	0.75	**0.97**	0.85	0.19
			II	80.3	0.82	0.91	0.86	0.23
			III	80.5	0.81	0.94	0.87	0.12
			IV	82.3	0.84	0.93	0.88	0.16
			V	80.4	0.83	0.87	0.85	0.10
			VI	**84.4**	**0.87**	0.91	**0.89**	**0.08**

Bold values indicate the best performance.

To summarize the outcomes of this sub-section, Variant I, given its recall (on average: 0.97) is able to correctly classify useful reviews (from all app reviews belonging to the useful category) than the other variants. Similarly, Variant IV was found to be precise (on average: 0.8), indicating correct identification of useful reviews among the app reviews that were classified as useful reviews and robust (average F-measure: 0.8) than the other variants.

RQ2. *Are there differences in outcomes for different Naïve Bayes implementations, particularly when considering data imbalances?*

To answer RQ2, we conducted Spearman's Rho correlation test to explore the relationship between the degree of data imbalance and accuracy, F-measure, and the time each variant took to classify reviews [55]. We report our findings in Table 4. Since there were five datasets involved in the previously conducted experiment, we obtained a total of five hundred observations for each variant (i.e., five datasets subjected to one hundred experiments each), wherein outcomes reflected accuracy, F-measure, and time results of the respective cross-validation operation. The results reported in Table 4 show that the accuracy of Variant I (Multinomial Naïve Bayes) decreased with an increase in data imbalance, whereas the accuracy of Variant II (Expectation Maximization—Multinomial Naïve Bayes) increased slightly with an increase in data imbalance. A similar conclusion can be drawn in cases of Variant IV (Expectation Maximization—Multinomial Naïve Bayes with Laplace smoothing) and V (Complement Naïve Bayes), where accuracy is directly affected by data imbalance. In addition, as the pattern of correlation coefficients observed for the accuracy metric is inconsistent and inconclusive for Variants III (Multinomial Naïve Bayes with Laplace smoothing) and VI (Complement Naïve Bayes with Laplace smoothing), no definitive inferences can be drawn from them.

Table 4. Tradeoff between Data Imbalance and Accuracy, F-measure and Time of Each Naïve Bayes Variant Measured through Spearman's Rho (r).

Variant	Spearman's Rho (r)		
	Accuracy	**F**	**Time**
I	−0.4 *	−0.41 *	0.2 *
II	0.1 *	−0.33 *	0.4 *
III	0.0	−0.17 *	0.2 *
IV	−0.2 *	−0.11 *	0.4 *
V	0.2 *	−0.13 *	0.2 *
VI	0.0	0.15 *	0.2 *

(* p-value < 0.01).

More importantly, such statistical outcomes pertaining to the accuracy metric are commonly observed in cases of imbalanced data, and hence, such outcomes are generally not considered to draw any conclusions by researchers [55]. Furthermore, as the data imbalance increases, the F-measure of Variants I to V decreases. This divergence was particularly pronounced for Variants I and II. On the contrary, the F-measure of Variant VI increases with an increase in data imbalance, indicating that Variant VI (i.e., Complement Naïve Bayes with Laplace Smoothing) is effective at handling imbalanced data. However, the decrease in the F-measure of the expectation maximization variants (II and IV) was lesser compared to their predecessors (I and II). Similarly, the variants incorporated with Laplace Smoothing were effective in handling imbalanced data compared to their earlier versions (III–I, IV–II, and VI–V).

In addition, the expectation maximization variants (II and IV) took more time when handling imbalanced data compared to their predecessors (I and II). That said, these data imbalances do not seem to affect the time needed for learning and prediction. Even though the reported correlations supporting the above-mentioned inferences are weak, they are statistically significant (p-value < 0.01).

Finally, we conducted Spearman's Rho correlation test to explore the relationship between the results of the F-measure and the time of each variant to probe our outcomes further [55]. We report our findings in Table 5. The results reported in Table 5 show that the F-measure of all the variants reduces with an increase in the time required for learning and prediction purposes. For all six cases, there is a statistically significant correlation (trade-off) observed between the F-measure and time.

Table 5. Tradeoff between F-measure and Time of Each Naïve Bayes Variant Measured through Spearman's Rho (r).

Variant	Spearman's Rho (r)
I	−0.7 *
II	−0.7 *
III	−0.7 *
IV	−0.5 *
V	−0.6 *
VI	−0.7 *

(* p-value < 0.01).

To summarize the outcomes of this sub-section, Variant VI exhibits better robustness (r = 0.15) when dealing with imbalanced data in comparison to the other variants, and the robustness of Variant IV (r = −0.5) was found to be reliable with an increase in the time for learning and prediction in comparison to the other variants.

The performance variations observed across datasets and variants can be attributed to differences in review characteristics such as length, vocabulary size, and topic diversity. These factors influence each variant's classification accuracy, reflecting the complex nature of app review data.

5. Discussion and Implications

RQ1. *What are the performances of Naïve Bayes variants when extracting useful reviews?*

Figure 1 provides a summary of performance results (accuracy, F-measure, and time metrics) of the six Naïve Bayes variants for the five datasets in the form of a box plot. The figure allows for meaningful evaluation of trends in our outcomes. When examining the range of results observed for the five datasets (TradeMe, MyTracks, Vodafone NZ, ThreeNow, and Flutter), the six variants exhibited varied performances. This conclusion is drawn based on the results shown by the accuracy, F-measure, and time metrics (refer to Section 5—RQ1 and Figure 1). We suspect that the type of features associated with each label (i.e., category) plays an important role in predicting the relevant label (useful or non-useful). This may explain variations in performances shown by the Naïve Bayes variants when classifying useful and non-useful reviews for the five datasets. Based on this outcome, we believe the variants can reliably predict the label of each review if the features spread across various labels had a higher degree of distinctness (i.e., if the features associated with a label are significantly distinct compared to the features associated with other labels), an aspect needing further empirical investigation. This is because, for some overlapping features (i.e., similar words belonging to different categories), the conditional probability $P(w_i \mid c_n)$ of the specific feature w_i given the category c_n could be normally distributed. In such a scenario, bias and variance of such features belonging to each category in the training data could be computed, and later, utilizing the probability density function of the normal distribution, $P(w_i \mid c_n)$ can be computed for the unlabeled reviews. To generate the probability value of a specific feature w_i from the feature's continuous probability density function, it would be necessary to integrate the probability density function around the probability value of the feature under examination over an interval of width epsilon and compute the limit of the integral as epsilon moves towards zero. This would enable the examination of the ratio of conditional probabilities generated by the particular variant that would ultimately assist in the generation of reliable features for learning purposes [56,57].

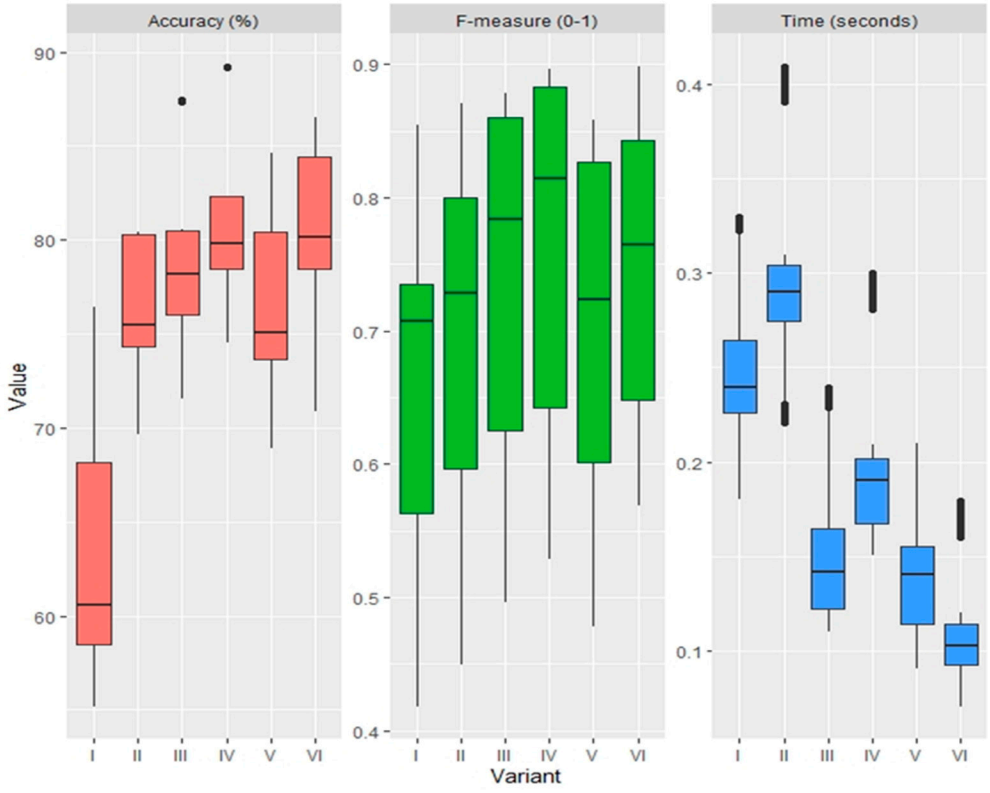

Figure 1. Overall performance of Naïve Bayes variants based on accuracy, F-measure, and time.

More importantly, we noticed that all the Naïve Bayes variants operated on the independence assumption, causing each variant to ignore the meaning of words relative to other words. This, in general, may compromise each variant's ability to calculate probabilities when working with words pertaining to real-world natural language applications [58]. For example, in the review 'the signal fades away', the words 'signal' and 'fades' are related as the word pair 'signal—fades' indicates that there is an issue with the phone signal. However, this is not considered by the Naïve Bayes variants. However, other machine learning algorithms, such as logistic regression, discretize the words or try to fit a normal distribution [59]. In fact, each Naïve Bayes variant assumes the word space is normally distributed with zero variance among words in all categories. This is a questionable assumption for any real-world application as, in some cases, the variant may be unable to generate a reliable discretization of interrelated (continuous) word features. This may potentially compromise prediction accuracy and thus demand a solution. A simple potential solution would be to test for the independence of the words to obtain a tentative estimate of prediction errors to determine the suitability of the application of a particular Naïve Bayes variant, or it may be useful to obtain a zero normal distribution to achieve more efficient results [60,61]. However, some of the measures returned above are significant (e.g., 89% accuracy, 0.87 precision, 0.98 recall, 0.89 F-measure, and 0.08 s time). Therefore, the Naïve Bayes variants examined in this study, individually, show promise for aiding useful review filtering to support software maintenance and evolution practice.

Furthermore, the Naïve Bayes method has been shown to outperform other methods on information retrieval tasks (i.e., tasks involving textual data) [25,26], and Chen et al. [8] reported an F-measure of 0.86 when their approach was evaluated. In the current study, we perform extensive experiments and confirm the value of Naïve Bayes with slightly improved results. In particular, the expectation maximization variants of the Naïve Bayes method produced F-measure as much as 0.86 (i.e., Variant II) and 0.89 (i.e., Variant IV).

RQ2. *Are there differences in outcomes for different Naïve Bayes implementations, particularly when considering data imbalances?*

It is evident in Figure 1 and statistics reported in Table 4 that the expectation maximization variants (II and IV) notably enhanced the basic Multinomial Naïve Bayes variants (I and III). The Expectation Maximization-Multinomial Naïve Bayes and Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing consistently outperformed their predecessors Multinomial Naïve Bayes and Multinomial Naïve Bayes with Laplace smoothing. These customizations resulted in as much as 32% improvement in accuracy in the retrieval of useful reviews over their predecessors. However, the Expectation Maximization-Multinomial Naïve Bayes and Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing variants of Naïve Bayes required more time for learning and prediction purposes (as much as 25% increase in time).

The increase in accuracy and F-measure noted in Section 5 is due to the working mechanism of Expectation Maximization that allows the Multinomial Naïve Bayes variants to gain maximum information about the underlying words in reviews of the same category during its learning phase. This is seen in Section 3.5 when uncategorized and categorized reviews are passed to the Expectation Maximization variant, which allows the Expectation Maximization variant to gain insights into the different types of words related to a particular category during its learning phase. The knowledge gathered during the learning process also leads to higher accuracy and F-measure. That said, the operating structure of the Multinomial Naïve Bayes and Multinomial Naïve Bayes with Laplace smoothing work based on closed-form formulas [62], which allows these variants to generate results quickly. This contrasts with Expectation Maximization-Multinomial Naïve Bayes and Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing, which produces results through an iterative process approach (waiting for likelihood parameters to stabilize), thereby requiring more time. In addition, the Expectation Maximization variants could handle these imbalanced data better than their predecessors even though they needed additional time for learning and prediction purposes (refer to Table 4).

In terms of Laplace smoothing, results show that this enhancement assisted significantly in increasing the accuracy and F-measure and reducing the time requirements for predictions including Multinomial Naïve Bayes, Expectation Maximization-Multinomial Naïve Bayes, and Complement Naïve Bayes (III, IV, and VI). We observe as much as an 18.8% increase in accuracy, a 0.15 improvement in F-measure, and a 0.14 s reduction in time attributed to the Laplace smoothing (all statistically significant outcomes). This concept significantly enhanced the retrieval of useful reviews. As seen from Equations (6) and (7), Laplace smoothing prevents zero counts of words whose information is not known in the training phase, thus maintaining the value of maximum likelihood estimates that are crucial towards the computation of a category of review. Therefore, any maximum likelihood estimate being 0 causes a lapse in the judgment toward determining the relevant category of a review. Consequently, the variants augmented by Laplace smoothing generate faster estimates of the parameters that compute the likelihood [63], hence improving Naïve Bayes prediction performance. In addition, as inferred from the findings reported in Table 4, Laplace smoothing benefited Variants III, IV, and VI in dealing with data imbalance. This effect is particularly pronounced when Variant VI is considered. Thus, concepts such as expectation maximization and Laplace smoothing contribute towards resolving the data imbalance issue.

Figure 1 and statistics reported in Table 4 also show overall, Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing performed well on the datasets in terms of accuracy and F-measure. Therefore, from a practical standpoint, Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing (IV) may be a suitable candidate for the task of retrieving useful reviews when app developers are dealing with limited amounts of manually categorized (or labeled) reviews. On the contrary, Complement Naïve Bayes with Laplace smoothing (VI) showed good performance on the TradeMe, Vodafone NZ, and Flutter datasets. This is because the working methodology of Comple-

ment Naïve Bayes incorporated with Laplace smoothing enables it to perform well when the dataset is imbalanced, as observed in the case of the above-mentioned datasets (refer to Section 4 and Table 4). To elaborate further, Complement Naïve Bayes variants attempt to normalize the word counts to rectify weight bias (i.e., data imbalance) [43]. The overall objective of the Complement Naïve Bayes variants is to make the estimated conditional probabilities insensitive to skewed counts of words (refer to Section 3.3). Hence, if there is a presence of few app reviews in one category (e.g., Useful) and these app reviews are comparable in length to those of the other category (e.g., non-useful) given the fact that certain words appear more often in app reviews of one category, then Complement Naïve Bayes tends to associate these app reviews with app reviews of other categories. Thus, by normalizing the word counts across categories, the weight bias is compensated.

Moreover, concerning the datasets, Complement Naïve Bayes with Laplace smoothing (VI) had the least time requirements (average ~0.11 s). Hence, the application of Complement Naïve Bayes with Laplace smoothing is best suited when app developers have a substantial number of categorized reviews whose labels are imbalanced and, at the same time, are bound by time constraints. However, the Complement Naïve Bayes with Laplace smoothing variant cannot incorporate Expectation Maximization, limiting its use to the previously mentioned application scenario.

Furthermore, as observed from Table 5, the F-measure of all the versions of the Naïve Bayes method decreased as the time required for learning and prediction increased. It is suspected that as the number of features increases, and if the likelihood of these features does not conform to the appropriate distribution required by the Naïve Bayes method, the F-measures of the variants are compromised. In addition, the Naïve Bayes method requires the number of features related to each category to be logarithmic to the size of the training data [59]. These observations further support our theory of generating reliable feature sets (i.e., feature sets consisting of appropriate features) pertaining to each category for the relevant variant, as mentioned earlier (refer to Section 5, RQ1 discussion). One potential solution to address this problem would be to utilize Information Gain (IG) to extract features from the training data and later sort the extracted features in descending order of their computed IG ratio to select the prominent features (e.g., top ‘n’, where n is based on some appropriate threshold) [64].

Our analysis revealed that dataset imbalance, review length, and vocabulary diversity affect variant performance. Complement Naïve Bayes variants (V and VI) showed improved performance on imbalanced datasets, while variants with Laplace smoothing (III, IV, and VI) handled larger vocabularies more effectively. These observations highlight the importance of considering dataset characteristics when selecting a Naïve Bayes variant for app review classification.

6. Threats to Validity

6.1. Internal Validity

In this study, the risks associated with labeling app reviews have been addressed by (1) making use of feedback from app developers, (2) studying and understanding the rules outlined in Chen et al. [12] for labeling app reviews, and (3) thoroughly analyzing various types of app reviews that concern app developers. The rules for labeling app reviews were extensively discussed among the authors to achieve a shared understanding before conducting reliability checks, which resulted in substantial agreements (see Fleiss Kappa statistics in Section 4). Follow-up discussions were held to reach a consensus among the authors before finalizing the labeled reviews. The primary objective of this study was to compare the performance of different Naïve Bayes variants in filtering useful app reviews, addressing data imbalance issues, and identifying potential research opportunities for improving their performance. Therefore, this work does not investigate the performance of other information retrieval approaches or methods for addressing data imbalance. However, potential future work aimed at conducting such an investigation could be planned. This investigation could involve the performance evaluation of popular machine learning

algorithms such as BERT (Bidirectional Encoder Representations from Transformers), Decision Trees, Random Forests, Logistic Regression, SVM, and so on, towards the filtering of useful reviews along with methods such as SMOTE (Synthetic Minority Oversampling Technique), ADASYN (adaptive synthetic sampling approach) which specialize in addressing the data imbalance issue. Potential future work could be planned to investigate these approaches. Finally, this study primarily focused on the performance of Naïve Bayes variants for extracting useful reviews, hence investigating and addressing issues related to the generation of distinct (reliable) features for learning purposes, and independence assumptions made by these variants were outside the scope of this study.

6.2. External Validity

A computer with a specific hardware configuration (detailed in Section 4) was used, which may limit the generalizability of our outcomes; however, the pattern of results was consistent across the datasets, so this was not a threat to the pattern of outcomes observed. We have utilized five datasets to evaluate the utility of the six Naïve Bayes variants for filtering useful reviews. Hence, the generalizability of the results may be limited to these datasets. However, the main objective of this study was to examine the feasibility and performance of the variants in filtering useful reviews and quantifying the evaluation of the results produced by the variants to identify the best-performing variants under certain circumstances. Our analysis was also restricted by the time and human resource constraints related to the manual labeling of the reviews and reliability assessments performed in this study.

6.3. Construct Validity

To construct the ground truth to filter useful reviews, we followed the well-established rules from a prominent study to label the app reviews [12]. In addition, recommended practices from the software engineering discipline guided our decisions (e.g., around reliability assessments and consensus formation). However, another approach to constructing this ground truth would be to contact the app developers of the respective apps to obtain the labeled set of reviews for evaluating the performance of the filtering approach. Such an approach could be a natural next step for future research.

7. Conclusions and Future Work

In this study, we examined Naïve Bayes variants for their usefulness in extracting useful app reviews. In the past, various approaches have been used to extract app reviews, with the method incorporating Expectation Maximization for the Naïve Bayes method showing the most potential. Thus, in this study, we explore the performances of six variants of Naïve Bayes. The findings indicate that, overall, Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing (Variant IV) is the most effective for extracting useful reviews from various datasets, while Complement Naïve Bayes with Laplace smoothing (Variant VI) is better suited for extracting useful reviews from highly imbalanced datasets. Furthermore, the utilization of such variants may provide decision support for software product maintenance and evolution.

That said, this study identifies several further research opportunities. For instance, the potential performance optimization of the Naïve Bayes variants for filtering app reviews provides a useful opportunity for follow-up research. In addition, the generation of discrete (reliable) features for learning purposes and addressing the independence assumption made by the variants are useful avenues for follow-up work. Additional datasets belonging to a wide spectrum of categories (e.g., Banking, Social, Video Players and Editors, and so on) from the app domain can be utilized to evaluate the performance of the proposed versions of Multinomial Naïve Bayes method to verify the application generalizability of the best-performing variants from a broader perspective (i.e., industry or academic settings). Beyond app reviews, however, the usefulness of these variants can be empirically

evaluated on bug reports and requests logged in software repositories such as Jira, GitHub, and others.

Author Contributions: Conceptualization, P.A. and S.M.; methodology, S.M.; software, S.M.; validation, P.A.; formal analysis, P.A. and S.M.; investigation, S.M.; resources, S.M.; data curation, S.M.; writing—original draft preparation, P.A., S.M. and D.S.; writing—review and editing, P.A., S.M., S.R. and D.S.; visualization, S.M. and D.S.; supervision, P.A. and D.S.; project administration, P.A. and D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data and code used in this study are now publicly available. The anonymized datasets for all five apps (TradeMe, MyTracks, VodafoneNZ, ThreeNow, and Flutter) and the Python implementation of the six Naïve Bayes variants can be accessed at https://github.com/Polyhistor/Naive-Bayes-MDPI/tree/main. This repository also includes instructions for running the experiments. We encourage researchers to use these resources for validation and extension of our work.

Acknowledgments: We would like to thank the app developers of Flutter for providing the app reviews and validating our preliminary outcomes.

Conflicts of Interest: Authors S.M. and S.R. were employed by the company IDEXX. Author P.A. was employed by the company Invenco. DS declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. We certify that the submission is original work and is not under review at any other publication venue.

Appendix A

Table A1. Datasets summary.

App Name	Total Number of Reviews Logged Category	Maximum Review Length (Characters)	Minimum Review Length (Characters)	Average Length of Review	Average App Rating	Category
MyTracks	4003	1988	3	136	3.8	Travel
Flutter	3483	2110	2	126	4.2	Casual
ThreeNow	3683	1483	2	132	1.5	Entertainment
TradeMe	4559	1732	3	112	3.2	Shopping
Vodafone NZ	6583	1434	2	123	2.4	Tool

References

1. Iqbal, M. App Revenue Data (2024), Business of Apps. Available online: https://www.businessofapps.com/data/app-revenues/ (accessed on 5 June 2024).
2. Laricchia, F. Topic: Smartphones, Statista. Available online: https://www.statista.com/topics/840/smartphones/ (accessed on 13 June 2024).
3. Malgaonkar, S. Prioritisation of Requests, Bugs and Enhancements Pertaining to Apps for Remedial actions Towards Solving the Problem of Which App Concerns to Address Initially for App Developers. Ph.D. Thesis, University of Otago, Dunedin, New Zealand, 2021.
4. Pagano, D.; Maalej, W. User feedback in the appstore: An empirical study. In Proceedings of the 2013 21st IEEE International Requirements Engineering Conference (RE), Rio de Janeiro, Brazil, 15–19 July 2013; pp. 125–134.
5. Maalej, W.; Nayebi, M.; Johann, T.; Ruhe, G. Toward Data-Driven Requirements Engineering. *IEEE Softw.* **2016**, *33*, 48–54. [CrossRef]

6. Fawareh, H.M.A.; Jusoh, S.; Osman, W.R.S. Ambiguity in text mining. In Proceedings of the 2008 International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia, 13–15 May 2008; pp. 1172–1176.
7. Corbett, J.; Savarimuthu, B.T.R.; Lakshmi, V. *Separating Treasure from Trash: Quantifying Data Waste in App Reviews*; University of Otago: Dunedin, New Zealand, 2020.
8. Licorish, S.A.; Savarimuthu, B.T.R.; Keertipati, S. Attributes that Predict which Features to Fix. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden, 15–16 June 2017; pp. 108–117.
9. Maalej, W.; Kurtanović, Z.; Nabil, H.; Stanik, C. On the automatic classification of app reviews. *Requir. Eng.* **2016**, *21*, 311–331. [CrossRef]
10. Keertipati, S.; Savarimuthu, B.T.R.; Licorish, S.A. Approaches for prioritizing feature improvements extracted from app reviews. In Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, Limerick, Ireland, 1–3 June 2016; pp. 1–6.
11. Fu, B.; Lin, J.; Li, L.; Faloutsos, C.; Hong, J.; Sadeh, N. Why people hate your app. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013; pp. 1276–1284.
12. Chen, N.; Lin, J.; Hoi, S.C.H.; Xiao, X.; Zhang, B. AR-miner: Mining informative reviews for developers from mobile app marketplace. In Proceedings of the 36th International Conference on Software Engineering, Hyderabad, India, 31 May–7 June 2014; pp. 767–778.
13. Shah, F.A.; Sirts, K.; Pfahl, D. Simple App Review Classification with Only Lexical Features. In Proceedings of the ICSOFT, Porto, Portugal, 26–28 July 2018; pp. 146–153.
14. Luo, Q.; Xu, W.; Guo, J. A Study on the CBOW Model's Overfitting and Stability. In Proceedings of the 5th International Workshop on Web-Scale Knowledge Representation Retrieval & Reasoning, Shanghai, China, 3 November 2014; pp. 9–12.
15. Johann, T.; Stanik, C.; Maalej, W. SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews. In Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 4–8 September 2017; pp. 21–30.
16. Gao, C.; Zeng, J.; Lyu, M.R.; King, I. Online App Review Analysis for Identifying Emerging Issues. In Proceedings of the 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), Gothenburg, Sweden, 27 May–3 June 2018; pp. 48–58.
17. Suresh, K.P.; Urolagin, S. Android App Success Prediction based on Reviews. In Proceedings of the 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM), Dubai, United Arab Emirates, 9–10 January 2020; pp. 358–362.
18. Hoon, L.; Vasa, R.; Schneider, J.-G.; Mouzakis, K. A preliminary analysis of vocabulary in mobile app user reviews. In Proceedings of the 24th Australian Computer-Human Interaction Conference, Melbourne, Australia, 26–30 November 2012; pp. 245–248.
19. Panichella, S.; Sorbo, A.D.; Guzman, E.; Visaggio, C.A.; Canfora, G.; Gall, H.C. How can I improve my app? Classifying user reviews for software maintenance and evolution. In Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bremen, Germany, 29 September–1 October 2015; pp. 281–290.
20. Iacob, C.; Harrison, R. Retrieving and analyzing mobile apps feature requests from online reviews. In Proceedings of the 2013 10th Working Conference on Mining Software Repositories (MSR), San Francisco, CA, USA, 18–19 May 2013; pp. 41–44.
21. Sutino, Q.; Siahaan, D. Feature extraction from app reviews in google play store by considering infrequent feature and app description. *J. Phys. Conf. Ser.* **2019**, *1230*, 012007. [CrossRef]
22. Cleland-Huang, J.; Settimi, R.; Zou, X.; Solc, P. Automated classification of non-functional requirements. *Requir. Eng.* **2007**, *12*, 103–120. [CrossRef]
23. Panichella, S.; Ruiz, M. Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback. In Proceedings of the 2020 IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland, 31 August–4 September 2020; pp. 404–407.
24. Michie, D.; Spiegelhalter, D.J.; Taylor, C. *Machine Learning, Neural and Statistical Classification*; Ellis Horwood: Amsterdam, The Netherlands, 1994; p. 13.
25. Caruana, R.; Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning—ICML '06, Pittsburgh, PA, USA, 25–29 June 2006; pp. 161–168.
26. Wang, C.; Zhang, F.; Liang, P.; Daneva, M.; van Sinderen, M. Can app changelogs improve requirements classification from app reviews? In Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Oulu, Finland, 11–12 October 2018; pp. 1–4.
27. McCallum, A.; Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification. In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization, Madison, Wisconsin, 26–27 July 1998.
28. Yuan, Q.; Cong, G.; Thalmann, N.M. Enhancing naive bayes with various smoothing methods for short text classification. In Proceedings of the 21st International Conference on World Wide Web, Lyon, France, 16–20 April 2012; pp. 645–646.
29. Iacob, C.; Harrison, R.; Faily, S. *Online Reviews as First Class Artifacts in Mobile App Development*; Springer: Cham, Switzerland, 2014; pp. 47–53.
30. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [CrossRef]

31. Wang, T.; Li, W.-h. Naive bayes software defect prediction model. In Proceedings of the 2010 International Conference on Computational Intelligence and Software Engineering, Wuhan, China, 10–12 December 2010; pp. 1–4.
32. Salton, G.; Wong, A.; Yang, C.-S. A vector space model for automatic indexing. *Commun. ACM* **1975**, *18*, 613–620. [CrossRef]
33. Aggarwal, C.; Zhai, C. *Mining Text Data*; Springer: Berlin/Heidelberg, Germany, 2012.
34. Plisson, J.; Lavrac, N.; Mladenic, D. A rule based approach to word lemmatization. *Proc. IS* **2004**, *3*, 83–86.
35. Rennie, J.D.; Shih, L.; Teevan, J.; Karger, D.R. Tackling the poor assumptions of naive bayes text classifiers. In Proceedings of the 20th International Conference on Machine Learning (ICML-03), Washington, DC, USA, 21–24 August 2003; pp. 616–623.
36. Lowd, D.; Domingos, P. Naive Bayes models for probability estimation. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 529–536.
37. He, F.; Ding, X. Improving naive bayes text classifier using smoothing methods. In Proceedings of the European Conference on Information Retrieval, Rome, Italy, 2–5 April 2007; pp. 703–707.
38. Dempster, A.P.; Laird, N.M.; Rubin, D.B. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Ser. B* **1977**, *39*, 1–38. [CrossRef]
39. Liu, B. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*; Springer: Berlin/Heidelberg, Germany, 2007.
40. Collins, M. The Naive Bayes Model, Maximum-Likelihood Estimation, and the EM Algorithm. Lecture Notes 2012. Available online: https://www.cs.columbia.edu/~mcollins/em.pdf (accessed on 13 June 2024).
41. Nigam, K.; McCallum, A.K.; Thrun, S.; Mitchell, T. Text Classification from Labeled and Unlabeled Documents using EM. *Mach. Learn.* **2000**, *39*, 103–134. [CrossRef]
42. Maalej, H.N.W. Bug report, feature request, or simply praise? On automatically classifying app reviews. In Proceedings of the 2015 IEEE 23rd International Requirements Engineering Conference (RE), Ottawa, Canada, 24–28 August 2015; pp. 116–125.
43. Kulesza, T.; Amershi, S.; Caruana, R.; Fisher, D.; Charles, D. Structured labeling for facilitating concept evolution in machine learning. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Toronto, ON, Canada, 26 April–1 May 2014; pp. 3075–3084.
44. Stumpf, S.; Rajaram, V.; Li, L.; Burnett, M.; Dietterich, T.; Sullivan, E.; Drummond, R.; Herlocker, J. Toward harnessing user feedback for machine learning. In Proceedings of the 12th International Conference on Intelligent User Interfaces, Honolulu, HI, USA, 28–31 January 2007; pp. 82–91.
45. He, H.; Garcia, E.A. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284.
46. Leevy, J.L.; Khoshgoftaar, T.M.; Bauder, R.A.; Seliya, N. A survey on addressing high-class imbalance in big data. *J. Big Data* **2018**, *5*, 42. [CrossRef]
47. Almuayqil, S.N.; Humayun, M.; Jhanjhi, N.; Almufareh, M.F.; Javed, D. Framework for improved sentiment analysis via random minority oversampling for user tweet review classification. *Electronics* **2022**, *11*, 3058. [CrossRef]
48. Fleiss, J.L.; Cohen, J. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educ. Psychol. Meas.* **1973**, *33*, 613–619. [CrossRef]
49. Landis, J.R.; Koch, G.G. The measurement of observer agreement for categorical data. *Biometrics* **1977**, *33*, 159–174. [CrossRef] [PubMed]
50. Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
51. Arlot, S.; Celisse, A. A survey of cross-validation procedures for model selection. *Stat. Surv.* **2010**, *4*, 40–79. [CrossRef]
52. Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the IJCAI, Montreal, QC, Canada, 20–25 August 1995; pp. 1137–1145.
53. Sheskin, D.J. *Handbook of Parametric and Nonparametric Statistical Procedures*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2003.
54. Wilcox, R.R. *Introduction to Robust Estimation and Hypothesis Testing*; Academic Press: Cambridge, MA, USA, 2011.
55. Myers, L.; Sirois, M.J. Spearman correlation coefficients, differences between. *Encycl. Stat. Sci.* **2004**, *12*. [CrossRef]
56. Zhu, J.; Wang, H.; Zhang, X. Discrimination-based feature selection for multinomial naïve bayes text classification. In Proceedings of the International Conference on Computer Processing of Oriental Languages, Singapore, 17–19 December 2006; pp. 149–156.
57. Kim, S.-B.; Rim, H.-C.; Yook, D.; Lim, H.-S. Effective methods for improving naive bayes text classifiers. In Proceedings of the Pacific Rim International Conference on Artificial Intelligence, Tokyo, Japan, 18–22 August 2002; pp. 414–423.
58. John, G.H.; Langley, P. Estimating continuous distributions in Bayesian classifiers. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–20 August 1995; pp. 338–345.
59. Ng, A.Y.; Jordan, M.I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 3–8 December 2002; pp. 841–848.
60. Boullé, M. MODL: A Bayes optimal discretization method for continuous attributes. *Mach. Learn.* **2006**, *65*, 131–165. [CrossRef]
61. Boullé, M. A Bayes Optimal Approach for Partitioning the Values of Categorical Attributes. *J. Mach. Learn. Res.* **2005**, *6*, 1431–1452.
62. Ren, J.; Lee, S.D.; Chen, X.; Kao, B.; Cheng, R.; Cheung, D. Naive bayes classification of uncertain data. In Proceedings of the 2009 Ninth IEEE International Conference on Data Mining, Miami, FL, USA, 6–9 December 2009; pp. 944–949.

63. Jung, Y.G.; Kim, K.T.; Lee, B.; Youn, H.Y. Enhanced Naive Bayes Classifier for real-time sentiment analysis with SparkR. In Proceedings of the 2016 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 19–21 October 2016; pp. 141–146.
64. Liu, Y.; Yi, X.; Chen, R.; Zhai, Z.; Gu, J. Feature extraction based on information gain and sequential pattern for English question classification. *IET Softw.* **2018**, *12*, 520–526. [CrossRef]