NaiveBayesWithReferences.docx

by Sri Regula

Submission date: 30-Jun-2024 08:13PM (UTC+1200)
Submission ID: 2410507450
File name: NaiveBayesWithReferences.docx (447.91K)
Word count: 13633
Character count: 76402

AI

MDPI

Article

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

Pouya Ataei [1], Saurabh Malgaonkar [2], Sri Regula [3], Daniel Staegemann [4,*]

1 School of Engineering Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand; pouya.ataei@aut.ac.nz
2 ezyVet, Auckland, New Zealand ; saurabhmalgaonkar@gmail.com
3 School of Engineering Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand; srir@aut.ac.nz
4 Faculty of Computer Science, Otto-von-Guericke University Magdeburg, 39106 Magdeburg, Germany; daniel.staegemann@ovgu.de
* Correspondence: daniel.staegemann@ovgu.de

Abstract: A single paragraph of about 200 words maximum. For research articles, abstracts should give a pertinent overview of the work. We strongly encourage authors to use the following style of structured abstracts, but without headings: (1) Background: Place the question addressed in a broad context and highlight the purpose of the study; (2) Methods: briefly describe the main methods or treatments applied; (3) Results: summarize the article's main findings; (4) Conclusions: indicate the main conclusions or interpretations. The abstract should be an objective representation of the article and it must not contain results that are not presented and substantiated in the main text and should not exaggerate the main conclusions.

Keywords: software maintenance, natural language processing, information retrieval, naïve bayes, expectation maximization, laplace smoothing

1. Introduction

It is predicted that the app market will be a $200B industry by end of 2020, 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].

Citation: To be added by editorial staff during production.

Academic Editor: Firstname Lastname

Received: date
Revised: date
Accepted: date
Published: date

Commented [DS1]: TODO, add Sri (on 3?)

Commented [DS2]: TODO

App developers use relevant OADPs such as Google Play[1] or Apple App [2] store to launch their app for end-users to access on mobile devices. In addition, OADPs facilitate the provision end-users' feedback in the form of reviews. The majority of feedback point towards request for new features, bugs or suggestions for improvements of the app which is useful for software maintenance and product evolution. However, OADPs host numerous reviews [4], which are open to public access in informing future users' 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 significantly assists app developers in their end-users' driven software quality evaluations, product marketing, and software maintenance processes [5]. However, manually extracting useful reviews from vast volumes of reviews demands high 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 non-useful reviews present in the app reviews repository of an app [6]. App developers must focus on filtering the useful reviews between these inconsequential ones in order 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', 'good' over the words reflecting app concerns such as 'inaccurate', 'update', 'crash' and so on. The filtering of non-useful 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 [4].

We explored such approaches in this work, where deficiencies were observed [9, 10]. Most significant in our observations, was that previous approaches which were designed to extract or filter useful reviews miss crucial reviews [9]. Further, while the Naïve Bayes method stands out as one of the most suitable for software engineering research and applications involving data filtering [11], we have not observed published efforts aimed at assessing the performances of particular variants of this method for the filtering of app reviews. We thus conducted this investigation, and explored the suitability of six Naïve Bayes variants for filtering 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. Secondly, 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. Overall, our contributions provide insights (and recommendations) for a critical software engineering problem.

The remaining sections of this paper are organized as follows. Section 2 presents studies related to the extraction of useful reviews. Section 3 describes the methods and concepts that assisted us in formulating the variants of Naïve Bayes. The experimental

[1] https://play.google.com/store

[2] https://www.apple.com/nz/ios/app-store/

setup for the evaluation of the variants of Naïve Bayes is presented in Section 4. Section 5 provides the results for our experiments. We document our discussions and the implications of our findings in Section 6, before considering the study's limitations in Section 7. Finally, we present concluding remarks in Section 8.

2. Related Work

App reviews expressed in the form of natural language is a common mechanism for gathering end-users' feedback 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 perform filtering based on disambiguation (contextual meaning) of the review contents [3]. For instance, Keertipati et al. [9] have extracted app 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. [10] have performed sentiment analysis using logistic regression to extract the reviews reflecting negative end-user sentiments with the assumption that negative reviews reflect severe app issues, missing out on useful positive reviews. In another study, Shah et al. [12] have evaluated 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 [13]. Similarly, Johann et al. [14] have utilized the parts of 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 has been initiated. Gao et al.'s [15] 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 that makes it difficult to identify the appropriate threshold parameters necessary for tuning this method in order to produce accurate results. Furthermore, AD often frequently generates false positive results [15]. 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 suggestions associated with the app features [16] (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 [17]. For example, consider the *useful* review that is filtered (extracted) on the basis of 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 labelled *non-useful* by the filtering process due to 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 with discriminating useful and non-useful reviews based on such subjectivities.

Certain research studies from the app domain have utilized classification as an approach to extract the reviews of interest (i.e., useful reviews) to address the above mentioned challenge. Such approach classifies app reviews having common attributes into specific categories (e.g., Pricing, Rating and so on) based on a taxonomy derived manually from domain knowledge, as a review of the literature shows that all the classification methods for classifying app reviews are dependent on domain knowledge made available manually through means of extensive research or by domain experts. For instance, Panichella et al. [18] have inherited a taxonomy from the taxonomy proposed by Pagano et al. [3] and have evaluated the classification performance SVM (Support Vector Machines),

Naïve Bayes, Decision Tress and Logistic Regression. Pagano et al. [3] have manually assigned categories that constitute a taxonomy for classifying app reviews. Similarly, Maalej et al. [4] manually developed four categories to classify app reviews using methods such as keyword lookup classifying mechanism, Decision Tress, Naïve Bayes and Maximum Entropy. Such studies have provided inspiration for others. For instance, Panichella et al. [19] developed a manual taxonomy that was inherited from the taxonomy created by Panichella et al. [18] to automatically classify reviews using the J48 supervised machine learning method. In another study, Ciurumelea et al. [20] have come up with five sets of categories by taking inspiration from [18] and created a taxonomy to classify reviews using Gradient Boosting supervised machine learning method. Similarly, Sorbo et al. [21] have developed a fine-grained taxonomy from the taxonomy proposed by Panichella et al. [18] which consists of additional categories over the study it is based on. After investigating the classification methods that classified reviews of apps we were able to identify a drawback. This drawback being, that all the classification methods were driven by manually derived taxonomy which is problematic when the domain knowledge is challenging to obtain from domain experts. In addition, the classification based approach using a manually derived taxonomy could be restricted in its application i.e., limited number of categories (e.g., GUI, Pricing and so on) specific to the category of an app (e.g., gaming or entertainment) made available for classification, thus potentially missing out other crucial app reviews which do not get classified into any of the limited categories. Another drawback of utilizing a manually created taxonomy for extracting app reviews of interest is the necessity to update the domain knowledge to create a new version of the taxonomy when the app evolves and new reviews are logged by the end-users [3]. Thus, these drawbacks point towards the requirement of an universal accurate and semi-automated information retrieval approach which extracts app reviews that reflect feature requests, bugs or suggestions for improvements which could be later turned into reliable actionable insights (e.g., classified app reviews based on an automatically generated taxonomy [22], prioritized app reviews for remedial actions [23] and so on).

Beyond the approaches mentioned above, rule-based linguistic approaches are assessed as valuable for filtering useful reviews. For instance, Iacob et al. [24] identified a set of linguistic rules to extract app feature requests from reviews. Similarly, Sutino et al. [25] 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 like these are combined with suitable machine learning methods to address such drawbacks, and may help with scalability challenges. For example, Huang et al. [26] have developed a probabilistic classifier that learns from a training set of manually pre-labelled requirements to predict appropriate labels (i.e., availability, look and feel, legal, maintainability, operational, performance, scalability, security, and usability) of the remaining set of non-functional requirements. In a recent study, Panichella et al [27] 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 which 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 towards automation of the tasks. However, the machine learning or deep learning approach that is used here (like some others) requires a large amount of pre-classified learning data to attain substantial levels of prediction accuracy [28].

That said, Multinomial Naïve Bayes is a well-known supervised machine learning method that has been empirically evaluated to be a suitable choice for text related software engineering applications, as it operates with the knowledge of word frequency information extracted from a text corpus [29,30]. Accordingly, this method significantly outperforms other machine learning methods [29]. For instance, Wang et al. [30] have evaluated the

performance of Naïve Bayes, Decision Trees, Bagging and KNN when classifying functional and non-functional requirements where Naïve Bayes provided the most reliable results. Additionally, the Naïve Bayes method does not tend to overfit the training data due to its handling of generalization towards predictions on new data, further leading towards the requirement of less training data for learning purposes [31]. This indicates that Naïve Bayes converges better than other discriminative machine learning methods like logistic regression and adapts suitably towards the changing or new data [31]. Thus, the Naïve Bayes method is able to efficiently map the data made available for the purpose of learning to relevant predictions [18-20]. In addition, the method does not possess parameters that need repeated tuning each time new predictions are to be performed. This contrasts with other machine learning algorithms such as SVM and KNN which require tuning [18-20]. Furthermore, the task of generating suitable amounts of training data required for Multinomial Naïve Bayes has been addressed by the semi-supervised Expectation Maximization of Multinomial Naïve Bayes [32]. This method has been widely used in software engineering applications such as for spam email filtering and software defect prediction [33,34].

One study has also used this approach to identify useful reviews [11], however its algorithmic and implementation details were not provided. In addition, even though Naïve Bayes was utilized to filter useful reviews belonging to four apps (i.e., SwiftKey, Facebook, TempleRun2 and TapFish) for the primary purposes of performing classification and generating meaningful visualizations; the authors only reported the F-Measure (0.86) for one app (i.e., Facebook) [11]. In fact, the goal of that work was not to investigate the performance of Naïve Bayes variants for their utility towards extracting useful app reviews. This raises the question: *Under what circumstances does Naïve Bayes method deliver the best performance?*

We reviewed the Naïve Bayes methods and concepts that are specialized in text classification operations [35,36], identifying six variants that have potential for filtering useful reviews. However these variants have not been investigated for their utility for 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 variants of the Naïve Bayes method. The prime objective of examining the Naïve Bayes variants proposed in this research is to assist app developers in the accurate extraction of useful reviews for software maintenance and evolution and extend academic knowledge around the application of IR approaches in software engineering. Thus, the research questions (RQs) that drive our study are:

RQ1. *What are the performances of Naïve Bayes variants when extracting useful reviews?*
RQ2. *Are there differences in outcomes for different Naïve Bayes implementations, and particularly when considering data imbalances?*

The performance of app reviews filtering methods is of prime importance to app developers in their drive to target the correct and most pressing app maintenance and evolution tasks [37]. To measure the *performance* (RQ1) of Naïve Bayes variants we utilize **accuracy**, **precision**, **recall**, and **F-Measure** metrics [28,38]. In addition to these metrics, we also examine the **time taken** by each variant for learning and prediction purposes [28]. As app developers need to address useful reviews in a timely manner, 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 different experimental settings (i.e., research methodology, data for experimentation, validation procedures and outcomes), and have used non-identical metrics when evaluations were performed. For example, Keertipati et al. [9] have used 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 for 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., not 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

In this section, we introduce the methods and concepts which assisted us in generating the respective variants of Naïve Bayes. The prime objective of the variants is to automatically filter (via classification) useful and non-useful reviews present in a vast app reviews corpus expressed in natural language. An initial set of useful and non-useful reviews can be manually identified using a pre-defined set of rules for filtering proposed in [11]. Rules pertaining to useful reviews reflect feature requests (e.g., 'please add feature A'), issues or bugs related to the app (e.g., 'the app crashes at the checkout screen'), or suggestions for app improvements (e.g., 'I suggest you increase the font size for a better view'). On the other hand, non-useful reviews contain irrelevant and unwanted information (e.g., 'this app is useless, uninstalling asap!'). Once the particular variant of Naïve Bayes has been trained, it can distinguish useful reviews from non-useful reviews by classifying each review into the appropriate category. Thus, for the given problem of classifying useful and non-useful reviews, the objective of the respective Naïve Bayes variant is to assign 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 particular Naïve Bayes variant is utilized to generate a classifier that predicts the categories of new reviews in 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 concepts 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., [39]).

3.1 Pre-processing of Reviews

Initially, several text pre-processing steps are followed to convert reviews into subsequent word vectors [40]. We perform review pre-processing by removing whitespaces, numbers, special characters (e.g., $, #) and punctuations (e.g., !, ?) present in the reviews, before converting them into lower case [41]. Finally, we perform removal of stop words (e.g., is, and) followed by lemmatization of the pre-processed reviews to generate the complete dictionary form of words present in the pre-processed reviews [42]. The aforementioned steps are standard text preprocessing procedures that are followed by researchers to avoid the generation of unreliable noisy results, and at the same time shortlist the reliable features (words) for learning and prediction purposes [4]. For instance, 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 which can be used by any machine learning algorithm [41]. Finally, these pre-processed reviews form the Vocabulary (V) that provides the necessary word frequency information for the Naïve Bayes variants [35].

3.2 Multinomial Naïve Bayes

Multinomial Naïve Bayes is a customized version of the basic Naïve Bayes method which is specialized for text classification [35]. This method works on the principle of maximum likelihood estimates. That is, 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:

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

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

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

Where, $P(w_i | c_n)$ denotes the conditional probability of the word w_i given the probability of category c_n that is given as the ratio of the total number of times a word w_i occurs in category c_n to the total number of words w in the reviews of category c_n. That is, the fraction of times word w_i appears among all words (V) in the reviews of category c_n. Thus, the Multinomial Naïve Bayes creates a word space for category c_n by creating a dictionary of words belonging to 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:

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

Where 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.

Learn

1. From the manually categorized 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 (1)
   - 3.1 Calculate $P(w_i | c_n)$ (maximum likelihood estimates)
     - 3.1.1 Word $space_n$ ← words belonging to $reviews_n$
     - 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 $P(w_i | c_n)$ ← n_i / n

End

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

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 addresses the inability of Multinomial Naïve Bayes's to perform well when trained with imbalanced data [43], i.e., the training data does not comprise of approximately equal proportion of reviews belonging to different types of categories. Complement Naïve Bayes addresses this drawback by estimating the likelihood of a category of c_n using training data of the other category(ies) $\bar{c_n}$ (except c_n). In case of Complement Naïve Bayes, the prior probability is computed using equation (1). Unlike Multinomial Naive Bayes, Complement Naive Bayes calculates the likelihood of a word w_i by considering its occurrences in category(ies) $\bar{c_n}$ 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:

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

Where, $P(w_i \mid \bar{c_n})$ denotes the probability of word w_i given it belongs to category(ies) $\bar{c_n}$ is given as the ratio of the total number of times a word w_i occurs in category(ies) $\bar{c_n}$ to the total number of words w in the reviews of category(ies) $\bar{c_n}$. 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) $\bar{c_n}$ by utilizing the frequency of w. Finally, using equations (1) and (4), the category of a review R can be determined using:

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

Where $C_{MAP}(R)$ denotes the most likely category c_n for a review R which is given as the argument of the minimum of likelihood estimates of the category computed as priori 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 [45] in Algorithm 2.

Begin

1. From the manually categorized 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 $P(c_n) \leftarrow |reviews_n| / |Total\ reviews|$
3. Calculate every word w_i given every category c_n
   - 3.1 Calculate $P(w_i \mid \bar{c_n})$ (maximum likelihood estimates)
     - 3.1.1 Word $space_n \leftarrow$ words belonging to reviews of category(ies) $\bar{c_n}$
     - 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 $P(w_i \mid \bar{c_n}) \leftarrow n_i / n$

End

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

Laplace Smoothing

From equations (2) and (4) it is evident that the parameters that generate the maximum likelihood estimate are unable to effectively handle any zero probabilities [44]. 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 in turn affects the accuracy of classification. This drawback is addressed by subjecting the parameters to Laplace Smoothing [36, 45], which instructs the parameters to add 1 to handle the zero counts of words efficiently, thus allowing the particular Naïve Bayes method to keep track of the count of words in determining the relevant category. Therefore, such a strategy is of prime importance especially when the particular Naïve Bayes method encounters a word in the classification phase (prediction/testing) whose information was not made available in the learning (training) phase. Hence, 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:

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

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

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

It is to be noted that, as a count of one has been added in the numerator, the size of the vocabulary (|V|) is added in 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 sub-section 3.2), and Complement Naïve Bayes (refer to sub-section 3.3) can be updated accordingly.

3.5 Expectation Maximization

Both methods highlighted in sub-sections 3.2 and 3.3 are supervised learning methods, and thus require a substantial number of manually categorized reviews to learn a classifier that is capable of accurately predicting the category of a new review. Accordingly, manually labelling (categorizing) adequate amounts of reviews might become a time-consuming task associated with potential errors, as it has to be manually performed by app developers. Semi-supervised learning approaches assist in addressing this drawback by reducing the labelling effort demanded from app developers. One of the common semi-supervised learning concepts comprises of learning from labelled as well as unlabeled information, and Expectation Maximization (EM) is an example of one such concept [46, 47]. EM primarily consists of two steps, Expectation (E) and Maximization (M). The *Expectation* step predicts and generates the absent information based on the current maximum likelihood estimation parameters initiated by the method in question (Multinomial Naïve Bayes), while the *Maximization* step iteratively re-estimates the parameters thus maximizing the overall likelihood [48].

Hence, EM allows the Multinomial Naïve Bayes method to run repeatedly until the parameters that estimate the total likelihood become constant [32]. We utilize the EM strategy to develop the semi-supervised version of the Multinomial Naïve Bayes method mentioned in sub-sections 3.2 and 3.4. The EM concept for this study was developed according to the algorithm mentioned in [32, 48]. The primary steps of EM would comprise of training the Multinomial Naïve Bayes method on known categories of reviews, and then later, using the learned information on categories associated with the reviews to make predictions on the 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 becomes constant (likelihood is computed using the entire corpus of app reviews). The detailed elaboration of the process mentioned above is as follows; consider an app reviews set AR consisting of reviews wherein each review R is tagged with a category C (useful or non-useful). The prime objective of EM is to generate the categories of the uncategorized reviews based on the Multinomial Naïve Bayes method's prediction mechanism. In every cycle, EM calculates the relevant probabilistic category and assigns it to the particular uncategorized review, that is $P(c_n|R_i)$ which is estimated to be 0 or 1. Here, c_n denotes the particular category, and R_i indicates the particular review. The categorized reviews having a specific category (x) is known prior, hence $P(c_x|R_i) = 1$ and $P(c_y|R_i) = 0$ for $x \neq y$. Using the information of categorized reviews, and $P(c_n|R_i)$, a new version of the Multinomial Naïve Bayes classifier is generated, which works in a recurring fashion until $P(w_i|c_n)$ and $P(c_n)$ become constant. We provide the pseudo-code of the EM concept [32] in Algorithm 3.

Begin

1. Train the Multinomial Naïve Bayes method mNB from 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 $P(c_n|Ri)$ (3)
3. Maximization (M):
   3.1 Train an updated version of mNB from R ∪ AR by calculating $P(c_n)$ and $P(w_i|c_n)$
4. Repeat steps 2 and 3 until mNB's parameters (maximum likelihood estimators) become constant.
4. Return mNB after completion of step 4.

End

Algorithm 3. Expectation Maximization concept for semi-supervised learning.

That said, as Complement Naïve Bayes method does not support any generative interpretations, thus the creation of its EM variant is not possible [43].

3.6 Summary of Naïve Bayes Variants

In this subsection, we review the Naïve Bayes variants as an outcome of the methods (refer to sub-sections 3.2 and 3.3), and concepts (refer to sub-sections 3.4 and 3.5) that were documented prior. Table 1 provides an overview of the particular Naïve Bayes variants. The prime objective in formulating these variants is to investigate their performance related to the prediction of review categories for a set of reviews pertaining to an app. To begin, we first formulate the Naïve Bayes variants belonging to the Multinomial Naïve Bayes method based on the method mentioned in sub-section 3.2, and the concepts mentioned in sub-section 3.4 and sub-section 3.5, there are four possible variants related to the Multinomial Naïve Bayes method. We introduce the first Naïve Bayes variant (I) that incorporates the functionality of the Multinomial Naïve Bayes method mentioned in sub-section 3.2. Next, as the EM mechanism allows the Multinomial Naïve Bayes method to deal with unlabeled reviews we generate the second variant (II) of Naïve Bayes that is a semi-supervised version of I. Thirdly, based on sub-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 post version of II. Next, we highlight the variants related to the Complement Naïve Bayes method. Based on the method mentioned in sub-section 3.3, we generate the Naïve Bayes variant (V) that implements the functionality of the Complement Naïve Bayes method. Next, based on sub-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.

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 sub-section 3.2.
II	Expectation Maximization - Multinomial Naïve Bayes	The Expectation Maximization concept described in sub-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 sub-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 post version of I.
V	Complement Naïve Bayes	This variant is the Complement Naïve Bayes method described in sub-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.

4. Experimental Setting

In this study, the Naïve Bayes variants described in Table 1 were implemented using the Python[3] programming language with suitable libraries provided by the Natural Language Tool Kit (NLTK)[4], numpy[5] and the scikit-learn[6] 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 using datasets consisting of app reviews belonging to five different apps obtained from 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 Table A for more details on these datasets) [29,49]. All the datasets consisted of reviews submitted by end-users written in natural language. TradeMe consisted of 4559 reviews, MyTracks dataset consisted of 4003 reviews, VodafoneNZ consisted of 6583, ThreeNow consisted of 3683 reviews and Flutter dataset consisted of 3483 reviews.

Using the set of rules defined in [11] (refer to section 3), app reviews from the datasets were manually categorized through labelling before reliability checks were done [50]. 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 Useful or Non-Useful.

Label	Rule	App Review Examples
Useful	Request	a. Please make the user interface more friendly and simple.

[3] https://www.python.org/

[4] https://www.nltk.org/

[5] https://numpy.org/

[6] https://scikit-learn.org/stable/

	• Requests to add or modify features • Request to remove advertisements or notifications	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 recommended validation practices of the software engineering discipline, this task was undertaken to empirically evaluate the performance of six Naïve Bayes variants based on human judgments and evaluations [50,51]. 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 [50,51]. Based on the manual labelling task and after performing the necessary reliability assessments, TradeMe dataset consisted of 1154 (25%) useful reviews and 3405 (75%) non-useful reviews, making it imbalanced (imbalance scale: 0.7) [52,53]. MyTracks dataset consisted of 1638 (41%) useful reviews and 2365 (59%) non-useful reviews (imbalance scale: 0.3) whereas VodafoneNZ consisted of 1120 (17%) useful reviews and 5463 (83%) non-useful reviews making it imbalanced (imbalance scale: 0.8) [54]. ThreeNow consisted of 1713 (48%) useful reviews and 1923 (52%) non-useful reviews (imbalance scale: 0.1), and Flutter dataset consisted of 2433 (70%) useful reviews and 1063 (30%) non-useful reviews, making it imbalanced (imbalance scale: 0.7) [52,53]. It is to be noted that we followed the guidelines mentioned in [52,53] 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). In case of approximately equally proportionate cases, the imbalance scale is lower. Hence, values closer to 0 indicate lower imbalance and values closer to 1 indicate larger imbalance.

Of note is that these app reviews were independently labelled *useful* or *non-useful* by the three authors (refer to Table 2 and [11] for rules). To perform the reliability assessments we

utilized Fleiss' Kappa which is the extended version of Cohen's Kappa to support the evaluations of three or more human evaluators [55]. The Fleiss coefficients were found to be 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 [56]. Follow up discussions were held among the authors to resolve any disagreements and establish consensus for achieving a reliable manual labelling process that led to 100% agreement.

The objective of app review classification using the particular Naïve Bayes variant is to correctly 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 as a metric determines the correctness of the particular Naïve Bayes given as the number of correctly classified reviews among the total number of 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 [38]. Next, we evaluate the precision metric which indicates the true positives to the total number of true positives and false positives [38]. Recall is given as the correctly classified useful reviews to the total number of reviews that were actually useful. Therefore, recall indicates the true positives to the total number of true positives and false negatives [38]. Finally, F-Measure is computed as the harmonic mean of precision and recall, which validates robustness of the variants [38]. Furthermore, the time metric measures the time (in seconds) required for a particular Naïve Bayes variant to learn from a set of manually categorized reviews (training data) to predict the category of unknown reviews (test data) [57]. The computer used for our experiments had 14 GB RAM and a CORE i5 CPU. For each experiment, we randomly split the respective dataset into a training set (90%) that is used to learn the relevant Naïve Bayes variant for reviews, and a testing set (10%), which is used to evaluate their performance in classifying nondisclosed reviews. Every experiment was run 100 times using ten-fold cross-validation (i.e., k=10) to obtain average scores for the metrics mentioned above [58]. This process is traditionally followed by researchers to validate the stability of the methods [59]. 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. The datasets and implementations of the six Naïve Bayes variants are available at https://tinyurl.com/y5zk7m6q for replication research or application purpose.

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.

5. Results

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

We present the results of the experiments conducted on the five datasets in Table 3, wherein, we report the average results of 100 times ten-fold cross-validation operations conducted on the TradeMe, MyTracks, Vodafone NZ, ThreeNow and Flutter datasets based on the metrics mentioned in Section 4. It is to be noted that in examining statistically significant differences among our outcomes, we ran the Shapiro-Wilk test to check the distribution of the results generated by each Naïve Bayes variant for normality assumption [60], finding no evidence confirming normality (p-value<0.01). Thus, we ran the Kruskal-Wallis non-parametric test to identify potential statistical significant differences between the results of the Naïve Bayes variants [60]. On finding statistically significant differences (p-value<0.01), we performed pairwise Wilcox testing to evaluate pairwise comparisons, with corrections for multiple comparisons [61], finding statistically significant differences for all comparisons (p-value<0.01).

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

Commented [DS3]: Maybe add an explanation?

Commented [SR4]: The url is broken

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 best performance.

That said, varied performances exhibited by the Naive 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 had the lowest accuracy (59.3%) and F-Measure (0.57) when compared to others, while VI exhibited the highest accuracy (80.2%) and F-Measure (0.65). Variant VI also required the least amount of time for learning and prediction purposes (0.10 seconds), while variant II required the most time (0.29 seconds). 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 seconds), while variant II required most time (0.30 seconds).

Similarly, we tested the six variants on the Vodafone NZ dataset and evaluated their performances accordingly. Overall, variant I had 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 seconds). We observe that variant II had the highest time requirement (0.40 seconds), 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 III, the results for variants II and exhibit very large differences in magnitude for accuracy (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 had the lowest accuracy (60.2%) and F-Measure (0.72) compared to others, while variant IV exhibited the highest accuracy (78.2%) and F-Measure (0.81). That said, variant VI had the least time requirements (0.11 seconds), while variant II required more time than others (0.28 seconds).

Finally, we tested the six Naïve Bayes variants on the Flutter dataset and evaluated their performances accordingly. Overall, variant I had the lowest accuracy (76.2%), while VI was seen to exhibit the highest F-Measure (0.89) with the least time (0.08 seconds) requirement. We observe that variant II had the highest time requirement (0.23 seconds), and variant IV was noted to be the second highest in terms of its performance based on accuracy (82.3% and F-Measure (0.88). That said, based on the observations in Table III, the results for variants II, III, and V did not exhibit very large differences in magnitude for accuracy and F-Measure (notwithstanding these differences were statistically significant p-value<0.01).

To summarize the outcomes of this sub-section, the 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 identfication of useful reviews among the app reviews what 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, and particularly when considering data imbalances?*

To answer RQ2, we conducted the Spearman's Rho correlation test to investigate the association between the scale of data imbalance and the accuracy, F-Measure, and time each variant took to classify reviews [62]. We report our findings in Table 4. Since there were five datasets involved in the previously conducted experiment, we obtained a total of 500 observations for each variant (i.e., 5 datasets subjected to 100 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 the increase in data imbalance, whereas the accuracy of variant II (Expectation Maximization -Multinomial Naïve Bayes) increased slightly with the 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.

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 decrease. This divergence was particularly pronounced for variants I and II. On the contrary, the F-Measure of variant VI increases with the increase in data imbalance, indicating that variant VI (i.e., Complement Naïve Bayes incorporated with Laplace Smoothing) is effective at handling imbalance data. However, the increase in F-Measure of the the expectation maximization variants (II and IV) was lesser in comparison to their predecessors (I and II). Similarly, the variants incorporated with Laplace Smoothing were effective in handling the imbalanced data in comparison to their previous versions (III-I, IV-II and VI-V).

Commented [DS5]: Maybe add a brief explanation

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

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)			

Finally, we conducted the Spearman's Rho correlation test to investigate the association between the results of the F-Measure and time of each variant to further probe our outcomes [62]. We report our findings in Table 5. The results reported in Table 5 show that the F-Measure of all the variants reduces with increase in time required for learning and prediction purposes. For all the six cases, there is statistically significant correlation (trade-off) observed between the F-Measure and time.

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

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)	

6. 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, ThreeNZ and Flutter), the six variants exhibited varied performances. This conclusion is drawn based on the results conveyed through the accuracy, F-Measure and time metrics (refer to section V- RQ1 and Figure 1). We suspect that the type of features associated with each label (i.e., category) plays significant role in predicting the relevant label (useful or non-useful). This may explain variations in performances exhibited for the Naïve Bayes variants when classifying useful and non-useful reviews for the five datasets. Based on this outcome, we believe that the variants may reliably predict the label associated with each review if the features spread across various labels had higher degree of distinctness (i.e., if the features associated with a label are significantly discrete in comparison to the features associated with the other labels), an aspect that requires further empirical investigation. This is because, for some overlapping features (i.e., similar words belonging to different categories) the conditional probability $P(w_i|c_n)$ 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, $P(w_i|c_n)$ can be computed for the unlabeled reviews. To generate the probability value of a specific feature w_i from a 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 towards the generation of reliable features for learning purposes [63,64].

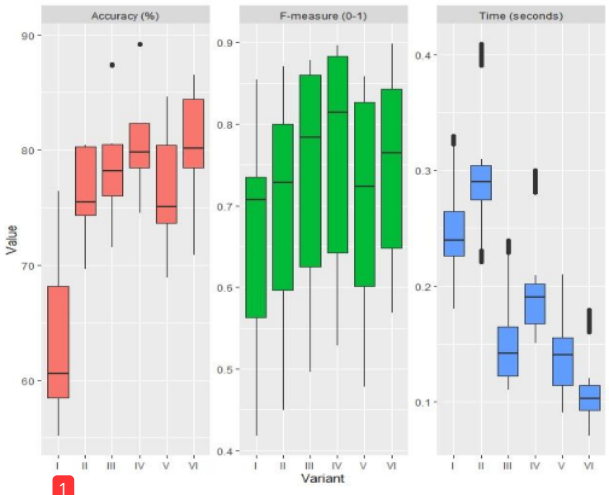

Figure 1. Overall performance of Naïve Bayes variants based on accuracy, F-Measure and time.

More importantly, we noticed that all the Naïve Bayes variants operated on the assumption of independence, which causes each variant to disregard the meaning of words it processed 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 [65]. 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. That said, other machine learning algorithms such as logistic regression discretize the words or attempt to fit a normal curve [31]. In fact, each Naive Bayes variant assumes that the word space is normally distributed with zero variance between words in all categories. This is a questionable assumption for any real-world application as in some cases the particular variant may be unable to generate a reliable discretization of interrelated (continuous) word features. This may potentially compromise prediction accuracy, and thus demands a solution. A simple potential solution would be to test for the independence of the words to get 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 generate a zero normal distribution towards producing more efficient results [66,67]. 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 seconds time). Thus, the **Naïve Bayes variants investigated in this work, on their own, hold promise for aiding useful reviews filtering to support software maintenance and evolution practice.**

Furthermore, Naïve Bayes method has been shown to outperform other methods on information retrieval tasks (i.e., tasks involving textual data) [29,30], and Chen et al. [8] reported 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, and 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) significantly improved 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 present in reviews belonging to the same category during its learning phase. This is seen in the sub-section 3.5 when uncategorized and categorized reviews are passed to the Expectation Maximization variant, which in turn allows the Expectation Maximization variant to gain insights on the different types of words pertaining to a particular category in its learning phase. The knowledge gathered during the learning process also leads towards 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 [68], which allow these variants to generate results quickly. This contrast with Expectation Maximization-Multinomial Naïve Bayes and Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing, which generate results based on an iterative approach (waiting for likelihood parameters to become constant), thus requiring more time.

Commented [SR6]: ?

In addition, the Expectation Maximization variants were capable of handling the 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 involving Multinomial Naïve Bayes, Expectation Maximization-Multinomial Naïve Bayes and Complement Naïve Bayes (III, IV and VI)**. We observe as much as 18.8% increase in accuracy, 0.15 improvement in F-Measure and 0.14 seconds reduction in time that were accounted for by the Laplace smoothing (all statistically significant outcomes). This concept enhanced the retrieval of useful reviews significantly. As observed from equations (6) and (7), Laplace smoothing avoids the zero counts of words whose information are not known in the training phase, thus preserving the value of maximum likelihood estimates that are crucial towards the computation of a category of review. Therefore, any maximum likelihood estimates being 0 causes a lapse in the judgment towards determining the relevant category of a review. Subsequently, the variants augmented by Laplace smoothing generate faster estimates of the parameters that compute the likelihood [69], hence improving Naïve Bayes prediction performance. Besides, as inferred from the findings reported in Table 4, Laplace smoothing assisted 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 that, overall, Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing performed well on the datasets in terms of accuracy and F-Measure. Thus, from a practical perspective, **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 labelled) reviews**. On the contrary, Complement Naïve Bayes with Laplace smoothing (VI) performed well on the TradeMe, Vodafone NZ and Flutter datasets. This is because the **working methodology of Complement Naïve Bayes incorporated with Laplace smoothing allows 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 the sub-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 Naive Bayes tends to associate these app reviews with app reviews of other categories. Thus, by normalizing the word counts across categories, the weight bias gets compensated.

Moreover, concerning the datasets, Complement Naïve Bayes with Laplace smoothing (VI) had the least time requirements (average ~ 0.11 seconds). **Hence, the application of Complement Naïve Bayes with Laplace smoothing is best suited when app developers have a substantial amount of categorized reviews whose labels are imbalanced, and at the same time are bound by time constraints.** However, Complement Naïve Bayes with Laplace smoothing variant's inability to incorporate Expectation Maximization makes its usage restrictive to the prior mentioned application scenario.

Furthermore, as observed from Table 5, the F-Measure of all the variants of the Naïve Bayes method decreased as the time required for learning and prediction increased. We suspect that as the number of features increase and if the likelihood of these features is not following the suitable distribution as required by the Naïve Bayes method, F-Measures of the variants are compromised. Besides, the Naïve Bayes method requires the number of features related to each category to be in logarithmic to the size of the training

data [31]. These observations further supports our theory of generating reliable features sets (i.e., features sets consisting of appropriate features) pertaining to each category for the relevant variant as mentioned earlier (refer to Section 6, 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 sorting 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) [30].

Commented [SR7]: Whats this?

7. Threats to Validity

7.1 Internal Validity

In this study we have mitigated the threats related to labelling of app reviews by: (1) making use of feedback provided by app developers, (2) studying and becoming familiar with the rules mentioned in Chen et al. [11] for labeling app reviews and, (3) rigorously analyzing various types of app reviews that app developers are concerned with. The rules pertaining to the labelling of app reviews were discussed extensively among the authors for shared understanding, before reliability checks were conducted which returned substantial agreements (see Fleiss Kappa statistics in Section 4). Follow up discussions were also held to establish consensus among the authors before finalizing the labelled reviews. That said, the prime objective of this study was to compare the performance of Naïve Bayes variants against each other for their effectiveness towards filtering of useful app reviews, addressing the data imbalance issue and identifying any potential research avenues that point towards the improvement in their performance. Thus, the performance of other IR approaches or machine learning and data imbalance addressing methods is not investigated in this work. 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 purpose and independence assumption made by these variants were beyond the scope of this study.

7.2 External Validity

We used a computer with specific hardware configuration (refer to Section 4), which may limit the generalizability of our outcomes, however, the pattern of results were consistent across the datasets and 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 towards filtering of 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 towards filtering of useful reviews and quantifying the evaluation of the results generated by the variants to identify the best performing variants under certain circumstances. Our analysis was also restricted by the time and human resource constraints associated with the manual labelling of the reviews and reliability assessments performed in this study.

7.3 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 [11]. In addition, recommended practices from the software engineering discipline guided our decisions (e.g., around reliability assessments and consensus formation). However, another alternative to construct

this ground truth would be to approach the app developers of the respective apps to obtain the labelled set of reviews for evaluating performance of the filtering approach. Such an approach could be a natural next step for future research.

8. Conclusion and Future Work

In this study, we investigated Naïve Bayes variants for their utility extracting useful app reviews. In the past, various approaches have been used to extract app reviews, with the approach incorporating Expectation Maximization for the Naïve Bayes method showing the most promise. Thus, in this study, we investigate the performances of six variants of Naïve Bayes. Our outcomes suggest that, overall Expectation Maximization-Multinomial Naïve Bayes with Laplace smoothing (i.e., variant IV) is best suited for extracting useful reviews belonging to different datasets and Complement Naïve Bayes with Laplace smoothing (i.e., variant VI) may be best suited for extracting useful reviews belonging to 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 of 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 wide spectrum of categories (e.g., Banking, Social, Video Players & Editors and so on) from app domain can be utilized to evaluate the performance of the proposed variants of Multinomial Naïve Bayes method to validate the application generalizability of the best performing variants from a broader perspective (i.e., industry or academic settings). Beyond app reviews however, the utility of these variants can be empirically evaluated on bug reports and requests logged in software repositories such as Jira, GitHub and so on.

Author Contributions: For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, X.X. and Y.Y.; methodology, X.X.; software, X.X.; validation, X.X., Y.Y. and Z.Z.; formal analysis, X.X.; investigation, X.X.; resources, X.X.; data curation, X.X.; writing—original draft preparation, X.X.; writing—review and editing, X.X.; visualization, X.X.; supervision, X.X.; project administration, X.X.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript." Please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding: Please add: "This research received no external funding" or "This research was funded by NAME OF FUNDER, grant number XXX" and "The APC was funded by XXX". Check carefully that the details given are accurate and use the standard spelling of funding agency names at https://search.crossref.org/funding. Any errors may affect your future funding.

Data Availability Statement: We encourage all authors of articles published in MDPI journals to share their research data. In this section, please provide details regarding where data supporting reported results can be found, including links to publicly archived datasets analyzed or generated during the study. Where no new data were created, or where data is unavailable due to privacy or ethical restrictions, a statement is still required. Suggested Data Availability Statements are available in section "MDPI Research Data Policies" at https://www.mdpi.com/ethics.

Commented [DS8]: TODO

Commented [DS9]: TODO

Acknowledgments: We would like to thank the app developers of Flutter for providing the app reviews and validating our preliminary outcomes. This work is funded by University of Otago Research Grant (UORG) Award – accessed through the University of Otago Research Committee.

Conflicts of Interest: The authors have no conflicts of interest to declare. All co-authors have seen and agree with the contents of the manuscript and there is no financial interest to report. We certify that the submission is original work and is not under review at any other publication venue.

Appendix A

Table A. 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

Commented [DS10]: Check in the end if formatting is fine

References

Commented [DS11]: Reference 54 in original paper is broken

1. [illegible]iev, A. App Revenues. https://www.businessofapps.com/data/app-revenues/.
2. Statista Smartphones - Statistics & Facts. https://www.statista.com/topics/840/smartphones/.
3. Pagano, D.; Maalej, W. In *User feedback in the appstore: An empirical study*, 2013 21st IEEE International Requirements Engineering Confe[illegible] (RE), 15-19 July 2013; 2013; pp 125-134.
4. Maalej, W.; Kurtanović, Z.; Nabil, H.; Stanik, C., On the automatic classification of app reviews. *Requirements Engineering* **2016,** [illegible] 311-331.
5. Maalej, W.; Nayebi, M.; Johann, T.; Ruhe, G., Toward Data-Driven Requirements Engineering. *IEEE Software* **2016,** 33 (1), 48-54.
6. Fawareh, H. M. A.; Jusoh, S.; Osman, W. R. S. In *Ambiguity in text mining*, 2008 International Conference on Computer and Communication Engineering, 13-15 May 2008; 2008; pp 1172-1176.
7. Corbett, J.; Savarimuthu, B. T. R.; Lakshmi, V., Separating Treasure from Trash: Quantifying Data Waste in App Reviews. **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*, ACM: Karlskrona, Sweden, 2017; pp 108-117.

9. 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*, ACM: Limerick, Ireland, 2016; p 306.
10. 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*, ACM: Chicago, Illinois, USA, 2013; pp 1276-1284.
11. 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*, ACM: Hyderabad, India, 2014; pp 767-778.
12. [illegible], F. A.; Sirts, K.; Pfahl, D. In *Simple App Review Classification with Only Lexical Features*, ICSOFT, 2018; pp 146-153.
13. Luo, Q.; Xu, W.; Guo, J. In *A Study on the CBOW Model's Overfitting and Stability*, Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning, ACM: 2014; pp 9-12.
14. Johann, T.; Stanik, C.; B, A. M. A.; Maalej, W. In *SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews*, 2017 IEEE 25th International Requirements Engineering Conference (RE), 4-8 Sept. 2017; 2017; pp 21-30.
15. Gao, C.; Zeng, J.; Lyu, M. R.; King, I. In *Online App Review Analysis for Identifying Emerging Issues*, 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), 27 May-3 June 2018; 2018; pp 48-58.
16. Suresh, K. P.; Urolagin, S. In *Android App Success Prediction based on Reviews*, 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM), IEEE: 2020; pp 358-362.
17. Hoon, L.; Vasa, R.; Schneider, J.-G.; Mouzakis, K. In *A preliminary analysis of vocabulary in mobile app user reviews*, Proceedings of the 24th Australian Computer-Human Interaction Conference, ACM: 2012; pp 245-248.
18. Panichella, S.; Sorbo, A. D.; Guzman, E.; Visaggio, C. A.; Canfora, G.; Gall, H. C. In *How can i improve my app? Classifying user reviews for software maintenance and evolution*, 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Sept. 29 2015-Oct. 1 2015; 2015; pp 281-290.
19. Panichella, S.; Di Sorbo, A.; Guzman, E.; Visaggio, C. A.; Canfora, G.; Gall, H. C. In *Ardoc: App reviews development oriented classifier*, Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering, ACM: 2016; pp 1023-1027.
20. Ciurumelea, A.; Panichella, S.; Gall, H. C. In *Poster: Automated User Reviews Analyser*, 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), 27 May-3 June 2018; 2018; pp 317-318.
21. Di Sorbo, A.; Panichella, S.; Alexandru, C. V.; Shimagaki, J.; Visaggio, C. A.; Canfora, G.; Gall, H. C., What would users change in my app? summarizing app reviews for recommending software changes. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ACM: Seattle, WA, USA, 2016; pp 499-510.
22. Malgaonkar, S.; Licorish, S. A.; Savarimuthu, B. T. R. In *Towards Automated Taxonomy Generation for Grouping App Reviews: A Preliminary Empirical Study*, Cham, Springer International Publishing: Cham, 2020; pp 120-134.
23. Noei, E.; Zhang, F.; Wang, S.; Zou, Y., Towards prioritizing user-related issue reports of mobile applications. *Empirical Software Engineering* **2019,** *24* (4), 1964-1996.
24. Iacob, C.; Harrison, R. In *Retrieving and analyzing mobile apps feature requests from online reviews*, 2013 10th Working Conference on Mining Software Repositories (MSR), 18-19 May 2013; 2013; pp 41-44.
25. Sutino, Q.; Siahaan, D. In *Feature extraction from app reviews in google play store by considering infrequent feature and app description*, Journal of Physics: Conference Series, IOP Publishing: 2019; p 012007.
26. Cleland-Huang, J.; Settimi, R.; Zou, X.; Solc, P., Automated classification of non-functional requirements. *Requirements Engineering* **2007,** *12* (2), 103-120.
27. Panichella, S.; Ruiz, M. In *Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback*, 2020 IEEE 28th International Requirements Engineering Conference (RE), 31 Aug.-4 Sept. 2020; 2020; pp 404-407.
28. Michie, D.; Spiegelhalter, D. J.; Taylor, C., Machine learning. *Neural and Statistical Classification* **1994,** *13*.

29. Caruana, R.; Niculescu-Mizil, A., An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*, ACM: Pittsburgh, Pennsylvania, USA, 2006; pp 161-168.
30. 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*, ACM: Oulu, Finland, 2018; pp 1-4.
31. Ng, A. Y.; Jordan, M. I. In *On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes*, Advances in neural information processing systems, 2002; pp 841-848.
32. Nigam, K.; McCallum, A. K.; Thrun, S.; Mitchell, T., Text Classification from Labeled and Unlabeled Documents using EM. *Machine learning* **2000,** *39* (2/3), 103-134.
33. Bacchelli, A.; Dal Sasso, T.; D'Ambros, M.; Lanza, M. In *Content classification of development emails*, 2012 34th International Conference on Software Engineering (ICSE), IEEE: 2012; pp 375-385.
34. Calders, T.; Verwer, S., Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery* **2010,** *21* (2), 277-292.
35. McCallum, A.; Nigam, K., *A Comparison of Event Models for Naive Bayes Text Classification*. 2001; Vol. 752.
36. Yuan, Q.; Cong, G.; Thalmann, N. M. In *Enhancing naive bayes with various smoothing methods for short text classification*, Proceedings of the 21st International Conference on World Wide Web, ACM: 2012; pp 645-646.
37. Iacob, C.; Harrison, R.; Faily, S. In *Online Reviews as First Class Artifacts in Mobile App Development*, Cham, Springer International Publishing: Cham, 2014; pp 47-53.
38. Sokolova, M.; Lapalme, G., A systematic analysis of performance measures for classification tasks. *Information processing & management* **2009,** *45* (4), 427-437.
39. Wang, T.; Li, W.-h. In *Naive bayes software defect prediction model*, 2010 International Conference on Computational Intelligence and Software Engineering, Ieee: 2010; pp 1-4.
40. Salton, G.; Wong, A.; Yang, C.-S., A vector space model for automatic indexing. *Communications of the ACM* **1975,** *18* (11), 613-620.
41. Aggarwal, C., & Zhai, C., *Mining Text Data*. Springer Science Business Media: 2012.
42. Nilsson, J.; Lavrac, N.; Mladenic, D., A rule based approach to word lemmatization. *Proceedings of IS-2004* **2004**, 83-86.
43. Rennie, J. D.; Shih, L.; Teevan, J.; Karger, D. R. In *Tackling the poor assumptions of naive bayes text classifiers*, Proceedings of the 20th international conference on machine learning (ICML-03), 2003; pp 616-623.
44. Lowd, D.; Domingos, P. In *Naive Bayes models for probability estimation*, Proceedings of the 22nd international conference on Machine learning, ACM: 2005; pp 529-536.
45. He, F.; Ding, X. In *Improving naive bayes text classifier using smoothing methods*, European Conference on Information Retrieval, Springer: 2007; pp 703-707.
46. Dempster, A. P.; Laird, N. M.; Rubin, D. B., Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)* **1977**, 1-38.
47. Liu, B., *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media: 2007.
48. Collins, M., The naive bayes model, maximum-likelihood estimation, and the em algorithm. *Lecture Notes* **2012**.
49. W. Maalej, H. N., Bug report, feature request, or simply praise? On automatically classifying app reviews. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, Ottawa, Canada, 2015; pp 116-125.
50. Kulesza, T.; Amershi, S.; Caruana, R.; Fisher, D.; Charles, D. In *Structured labeling for facilitating concept evolution in machine learning*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2014; pp 3075-3084.

51. Stumpf, S.; Rajaram, V.; Li, L.; Burnett, M.; Dietterich, T.; Sullivan, E.; Drummond, R.; Herlocker, J. In *Toward harnessing user feedback for machine learning*, Proceedings of the 12th international conference on Intelligent user interfaces, ACM: 2007; pp 82-91.
52. He, H.; Garcia, E. A., Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* **2009,** *21* (9), 1263-1284.
53. Leevy, J. L.; Khoshgoftaar, T. M.; Bauder, R. A.; Seliya, N., A survey on addressing high-class imbalance in big data. *Journal of Big Data* **2018,** *5* (1), 42.
54. **!!!INVALID CITATION !!!**
55. Fleiss, J. L.; Cohen, J., The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement* **1973,** *33* (3), 613-619.
56. Landis, J. R.; Koch, G. G., The measurement of observer agreement for categorical data. *Biometrics* **1977,** *33* (1), 159-74.
57. Huang, G.-B.; Zhu, Q.-Y.; Siew, C.-K., Extreme learning machine: theory and applications. *Neurocomputing* **2006,** *70* (1-3), 489-501.
58. Arlot, S.; Celisse, A., A survey of cross-validation procedures for model selection. *Statistics Surveys* **2010,** *4* (none), 40-79.
59. Kohavi, R. In *A study of cross-validation and bootstrap for accuracy estimation and model selection*, Ijcai, Montreal, Canada: 1995; pp 1137-1145.
60. Sheskin, D. J., *Handbook of parametric and nonparametric statistical procedures*. Chapman and Hall/CRC: 2003.
61. Wilcox, R. R., *Introduction to robust estimation and hypothesis testing*. Academic press: 2011.
62. Myers, L.; Sirois, M. J., Spearman correlation coefficients, differences between. *Encyclopedia of statistical sciences* **2004,** *12*.
63. Zhu, J.; Wang, H.; Zhang, X. In *Discrimination-based feature selection for multinomial naive bayes text classification*, International Conference on Computer Processing of Oriental Languages, Springer: 2006; pp 149-156.
64. Kim, S.-B.; Rim, H.-C.; Yook, D.; Lim, H.-S. In *Effective methods for improving naive bayes text classifiers*, Pacific Rim International Conference on Artificial Intelligence, Springer: 2002; pp 414-423.
65. John, G. H.; Langley, P. In *Estimating continuous distributions in Bayesian classifiers*, Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc.: 1995; pp 338-345.
66. Boullé, M., MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning* **2006,** *65* (1), 131-165.
67. Boullé, M., A Bayes Optimal Approach for Partitioning the Values of Categorical Attributes. *Journal of Machine Learning Research* **2005,** *6*, 1431-1452.
68. Ren, J.; Lee, S. D.; Chen, X.; Kao, B.; Cheng, R.; Cheung, D. In *Naive bayes classification of uncertain data*, 2009 Ninth IEEE International Conference on Data Mining, IEEE: 2009; pp 944-949.
69. Jung, Y. G.; Kim, K. T.; Lee, B.; Youn, H. Y. In *Enhanced Naive Bayes Classifier for real-time sentiment analysis with SparkR*, 2016 International Conference on Information and Communication Technology Convergence (ICTC), 19-21 Oct. 2016; 2016; pp 141-146.
70. Liu, Y.; Yi, X.; Chen, R.; Zhai, Z.; Gu, J., Feature extraction based on information gain and sequential pattern for English question classification. *IET Software* **2018,** *12* (6), 520-526.

NaiveBayesWithReferences.docx

ORIGINALITY REPORT

44%	43%	23%	15%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	hdl.handle.net Internet Source	24%
2	www.steconf.org Internet Source	2%
3	link.springer.com Internet Source	2%
4	Saurabh Malgaonkar, Sherlock A. Licorish, Bastin Tony Roy Savarimuthu. "Prioritizing User Concerns in App Reviews – A Study of Requests for New Features, Enhancements and Bug Fixes", Information and Software Technology, 2021 Publication	1%
5	Submitted to Unizin, LLC Student Paper	1%
6	www.mdpi.com Internet Source	1%
7	Agustin Casamayor, Daniela Godoy, Marcelo Campo. "Identification of non-functional requirements in textual specifications: A semi-	1%

supervised learning approach", Information and Software Technology, 2010
Publication

8	www.researchgate.net Internet Source	1 %
9	www.springerprofessional.de Internet Source	1 %
10	dokumen.pub Internet Source	<1 %
11	journals.plos.org Internet Source	<1 %
12	Tejas Hirave, Saurabh Malgaonkar, Mostafa Alwash, Jithin Cheriyan, Sakshi Surve. "Analysis and Prioritization of App Reviews", 2019 International Conference on Advances in Computing, Communication and Control (ICAC3), 2019 Publication	<1 %
13	Submitted to Nottingham Trent University Student Paper	<1 %
14	export.arxiv.org Internet Source	<1 %
15	Ernesto Rosario Russo, Andrea Di Sorbo, Corrado A. Visaggio, Gerardo Canfora. "Summarizing vulnerabilities' descriptions to support experts during vulnerability	<1 %

assessment activities", Journal of Systems and Software, 2019
Publication

16	Submitted to Universita degli Studi di Torino Student Paper	<1 %
17	Submitted to Ege Üniversitesi Student Paper	<1 %
18	academic.hep.com.cn Internet Source	<1 %
19	www.infocomm-journal.com Internet Source	<1 %
20	d-nb.info Internet Source	<1 %
21	trepo.tuni.fi Internet Source	<1 %
22	mdpi-res.com Internet Source	<1 %
23	Sherlock A. Licorish, Bastin Tony Roy Savarimuthu, Swetha Keertipati. "Attributes that Predict which Features to Fix", Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering - EASE'17, 2017 Publication	<1 %
24	www2.mdpi.com Internet Source	<1 %

25	Submitted to University of Edinburgh Student Paper	<1 %
26	ndltd.ncl.edu.tw Internet Source	<1 %
27	research.monash.edu Internet Source	<1 %
28	ebin.pub Internet Source	<1 %
29	Submitted to Universidade de Sao Paulo Student Paper	<1 %
30	people.cmix.louisiana.edu Internet Source	<1 %
31	www.wseas.org Internet Source	<1 %
32	manu44.magtech.com.cn Internet Source	<1 %
33	P. Priyanga, A. R. Nadira Banu Kamal. "Mobile App Usage Pattern Prediction Using Hierarchical Flexi-Ensemble Clustering (HFEC) for Mobile Service Rating", Wireless Personal Communications, 2021 Publication	<1 %
34	mds.marshall.edu Internet Source	<1 %

35	Submitted to The Robert Gordon University Student Paper	<1 %
36	online-journals.org Internet Source	<1 %
37	www.benthamscience.com Internet Source	<1 %
38	saemobilus.sae.org Internet Source	<1 %
39	Submitted to Sim University Student Paper	<1 %
40	www-di.inf.puc-rio.br Internet Source	<1 %
41	www.jos.org.cn Internet Source	<1 %
42	www.otago.ac.nz Internet Source	<1 %
43	yadda.icm.edu.pl Internet Source	<1 %
44	doc.lagout.org Internet Source	<1 %
45	researchwith.njit.edu Internet Source	<1 %
46	P. Daniel Patterson, Charity G. Moore, Jane H. Brice, Elizabeth G. Baxley. "Use of ED	<1 %

	Diagnosis to Determine Medical Necessity of EMS Transports", Prehospital Emergency Care, 2009 Publication	
47	doi.org Internet Source	<1 %
48	www.ripublication.com Internet Source	<1 %
49	Submitted to National Institute of Technology, Hamirpur Student Paper	<1 %
50	Submitted to RMIT University Student Paper	<1 %
51	Saurabh Malgaonkar, Sherlock A. Licorish, Bastin Tony Roy Savarimuthu. "Understanding requirements prioritisation: literature survey and critical evaluation", IET Software, 2020 Publication	<1 %
52	web.archive.org Internet Source	<1 %
53	Submitted to University of Sydney Student Paper	<1 %
54	aisel.aisnet.org Internet Source	<1 %
55	res.mdpi.com Internet Source	<1 %

56	"Service-Oriented Computing – ICSOC 2020 Workshops", Springer Science and Business Media LLC, 2021 Publication	<1 %
57	biointerfaceresearch.com Internet Source	<1 %
58	c.coek.info Internet Source	<1 %
59	eprints.usq.edu.au Internet Source	<1 %
60	repozitorij.svkst.unist.hr Internet Source	<1 %
61	vdocuments.mx Internet Source	<1 %
62	www.nature.com Internet Source	<1 %
63	xuanhui.me Internet Source	<1 %
64	Grady Ball, Peter Regier, Ricardo González-Pinzón, Justin Reale, David Van Horn. "Wildfires increasingly impact western US fluvial networks", Nature Communications, 2021 Publication	<1 %

65	Shawni Dutta, Samir Kumar Bandyopadhyay. "3 Analysis of divergence aspects of breast in breast cancer patients that change due to COVID-19", Walter de Gruyter GmbH, 2021 Publication	<1 %
66	Submitted to University of Leicester Student Paper	<1 %
67	bmcbioinformatics.biomedcentral.com Internet Source	<1 %
68	www.medrxiv.org Internet Source	<1 %
69	buleria.unileon.es Internet Source	<1 %
70	iris.unitn.it Internet Source	<1 %
71	xin-xia.github.io Internet Source	<1 %
72	Submitted to Student Paper	<1 %
73	Rajni Jindal, Ruchika Malhotra, Abha Jain, Ankita Bansal. "Mining Non-Functional Requirements using Machine Learning Techniques", e-Informatica Software Engineering Journal, 2021 Publication	<1 %

74	Saurabh Malgaonkar, Sherlock A. Licorish, Bastin Tony Roy Savarimuthu. "Prioritizing user concerns in app reviews – A study of requests for new features, enhancements and bug fixes", Information and Software Technology, 2022 Publication	<1 %
75	www.iajit.org Internet Source	<1 %
76	aemps.ewapublishing.org Internet Source	<1 %
77	Hao Lu, Kaize Shi, Yifan Zhu, Yisheng Lv, Zhendong Niu. "Sensing Urban Transportation Events from Multi-Channel Social Signals with the Word2vec Fusion Model", Sensors, 2018 Publication	<1 %
78	www.academypublisher.com Internet Source	<1 %
79	Kato, Bence. "Regional-Scale Analyses on Seismic Site-City Interaction of High-Rise Building Clusters and Shallow Basin Effects", Hong Kong University of Science and Technology (Hong Kong), 2022 Publication	<1 %
80	Rongfang Bie, Zengmei Fu, Qiurui Sun, Chuanliang Chen. "A Comparison Study of	<1 %

Bayesian Classifiers on Web Pages Classification", New Generation Computing, 2010
Publication

81	zone.biblio.laurentian.ca Internet Source	<1 %
82	Broschk, Heidi. "Teaching German as a Mother Tongue and as a Foreign Language at the DSJ", University of Johannesburg (South Africa), 2021 Publication	<1 %
83	Herold, J.. "SpeedSeg: A technique for segmenting pen strokes using pen speed", Computers & Graphics, 201104 Publication	<1 %
84	Hua (Jonathan) Ye, Cecil Eng Huang Chua, Jun Sun. "Enhancing mobile data services performance via online reviews", Information Systems Frontiers, 2017 Publication	<1 %
85	Lecture Notes in Computer Science, 2008. Publication	<1 %
86	Naila Aslam, Waheed Yousuf Ramay, Xia Kewen, Nadeem Sarwar. "Convolutional Neural Network Based Classification of App Reviews", IEEE Access, 2020 Publication	<1 %

87	Saurabh Malgaonkar, Sherlock A. Licorish, Bastin Tony Roy Savarimuthu. "Automatically generating taxonomy for grouping app reviews — a study of three apps", Software Quality Journal, 2021 Publication	<1 %
88	doczz.pl Internet Source	<1 %
89	ipfs.io Internet Source	<1 %
90	vdoc.pub Internet Source	<1 %
91	www.mtome.com Internet Source	<1 %
92	www.scribd.com Internet Source	<1 %
93	Sebastiano Panichella, Marcela Ruiz. "Requirements-Collector: Automating Requirements Specification from Elicitation Sessions and User Feedback", 2020 IEEE 28th International Requirements Engineering Conference (RE), 2020 Publication	<1 %
94	"International Conference on Innovative Computing and Communications", Springer Science and Business Media LLC, 2022	<1 %

Publication

95	"Machine Learning and Data Mining in Pattern Recognition", Springer Nature, 2011 Publication	<1 %
96	"The Essence of Software Engineering", Springer Nature, 2018 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		