Soft Computing
https://doi.org/10.1007/s00500-020-05297-6

METHODOLOGIES AND APPLICATION

Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation

Indika Wickramasinghe[1] · **Harsha Kalutarage**[2]

Abstract
Naïve Bayes (NB) is a well-known probabilistic classification algorithm. It is a simple but efficient algorithm with a wide variety of real-world applications, ranging from product recommendations through medical diagnosis to controlling autonomous vehicles. Due to the failure of real data satisfying the assumptions of NB, there are available variations of NB to cater general data. With the unique applications for each variation of NB, they reach different levels of accuracy. This manuscript surveys the latest applications of NB and discusses its variations in different settings. Furthermore, recommendations are made regarding the applicability of NB while exploring the robustness of the algorithm. Finally, an attempt is given to discuss the pros and cons of NB algorithm and some vulnerabilities, with related computing code for implementation.

Keywords Naïve Bayes · Probabilistic classification · Machine learning vulnerabilities · R code snippets

1 Introduction

Recent advances in low-cost computing and data explosion have democratized machine learning and data analytic (MLDA), allowing developers to apply these technologies almost everywhere in real-world applications. Data classification plays a key role in MLDA. The use of some of the conventional statistical techniques in data classification became plausible as a result of low-cost computational power. Statistical approaches to MLDA problems have become major alternatives to well-known algorithms used in MLDA (John and Langley 1995). Though sophisticated statistical techniques are used in numerous applications, some appealing and effective outcomes have been revealed with the use of much simpler and basic statistical approaches (George et al. 1995). Most importantly, these alternative techniques to conventional machine learning algorithms have been outperformed by the statistical approaches.

Broadly, any classification algorithm can be classified as either probabilistic or non-probabilistic. Probabilistic data classification relies on approximating a distribution. Data classification techniques based on probabilistic approach work well as most of the distributions of related features follow probabilistic nature. Garg and Roth (2001) answer the fundamental question of why probabilistic approach works well theoretically in various real-world applications. Some of the probabilistic classifiers include NB, logistic regression, and multilayer perceptrons. Support vector machines and K-nearest neighbors are examples for the non-probabilistic classifiers.

Out of the above-stated algorithms, NB has an important place in data classification due to many reasons. The simplicity and the accuracy of the algorithm are two of the main reasons. As the name implies, NB algorithm is based on the popular Bayes theorem and is one of the prominent probabilistic classification techniques used in MLDA. The popularity of NB is not only due to the simplicity of it, but also due to the effectiveness and the robustness of the algorithm (Arar and Ayan 2017). According to the literature, NB is one of the top performing algorithms used in data mining (Wu et al. 2008; Settouti et al. 2016). Clark and Niblett (1989) reveal that NB generates similar and accurate results to rule-induction approaches in medical domains, while Langley

Communicated by V. Loia.

✉ Indika Wickramasinghe
iprathnathungalage@pvamu.edu

Harsha Kalutarage
h.kalutarage@rgu.ac.uk

[1] Department of Mathematics, Prairie View A & M University, Prairie View, USA

[2] School of Computing, Robert Gordon University, Aberdeen, United Kingdom

Published online: 09 September 2020

et al. (1992) finds that basic statistical approaches outperform some of the other machine learning algorithms.

The accuracy of classification model can depend on various factors. One of the main factors is the set of assumptions attached to each classifier. Similar to majority of parametric procedures, NB also makes some assumptions about the underline data in order to produce the optimal result. Furthermore, Rish (2001) assesses that NB competes well compared to more sophisticated classification techniques. Rish (2001) investigates the impact of the characteristics of data on the performance of the accuracy of NB using entropy feature distribution. Though the success of the algorithm is appealing in practice (Garg and Roth 2001; Domingos and Pazzani (1996); Elkan 1997), these assumptions become unrealistic in some applications. Rish (2001) finds that NB performs reasonably well, even with feature dependencies. Rennie et al. (2003) see the main assumption of the NB is very straight forward and simple which has helped it to become an easy to use algorithm. At times, these assumptions can adversely affect the output's quality. They address problem of the selection of poor weights for decision boundary when the training sample sizes have different instances for each class. Furthermore, they investigate about the issue of independent assumption and find that if there is a higher dependency among the features, the magnitude of the assigned weights for those classes is higher than that of classes with lower dependency. In addition, there are some variations of NB achieving different levels of accuracy.

The rest of the paper is organized as follows: Section 2 discusses the theoretical background of the NB, and Sect. 3 discusses the existing variations of NB. Applications of NB classifier are discussed in Sect. 4, and Sect. 5 discusses the instances when NB performs well and poorly. In Sect. 6, we discuss how to use NB classifier in various data classification problems and to implement Bayesian poisoning attack using R language. Finally, Sect. 7 concludes the paper.

2 Naïve Bayes classifier

Let $X_1, X_2, \ldots, X_n$ be a n-dimensional vector of random variables (features) from a domain D_X and $x_1, x_2, \ldots, x_n$ be their corresponding instances. Let Y be an unobserved random variable from domain $D_Y = 0, 1$. Though it is unknown, let's assume that there is a function from D_X to D_Y. The aim is to estimate the target value y_i of Y for a given instance x_i of X. In another words, the aim is to select the class Y that maximizes the posterior probability, $P(Y = y|X = x)$. Here, $P(Y = y)$ and $P(X = x|Y = y)$ are prior and class conditional probabilities. As the NB assumes conditional independence, $P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n|Y = y) = \prod_{i=1}^{n} P(X = x_i|Y = y)$. Let C be the number of classes of Y. According to the Bayes' theorem,

$$
\begin{aligned}
&P(Y = y|X = x) \\
&= \frac{P(Y = y, X = x)}{P(X = x)} \\
&= \frac{P(Y = y)P(X = x|Y = y)}{P(X = x)} \\
&= \frac{P(Y = y)P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n|Y = y)}{\Sigma_i^C P(y_i, X = x)} \\
&= \frac{P(Y = y)\prod_{i=1}^{n} P(X = x_i|Y = y)}{\Sigma_i^C P(y_i, X = x)} \qquad (1)
\end{aligned}
$$

In practice, it is not interested about the $P(X = x)$. Instead of estimating $P(X = x)$, it is normalized in order to have the $P(Y = y|X = x) = 1$. Usually, in practical point of view, $P(X = x|Y = y)$ is assumed to follow a Gaussian distribution, though the literature shows some exceptions. George et. al (1995) replace the flexible Gaussian assumption with a kernel density estimation. In addition to the above Gaussian assumption, one of the main assumptions is that the value of any given feature is independent of the value of any other feature. There are different opinions about the practicality of this assumption, which has created the algorithm more viable. Moreover, one can exploit the Naive assumption to speed up the execution of the algorithm. For example, attribute probabilities can be calculated in parallel using different CPUs, machines, or clusters in the real-world deployment of NB-based applications. This success of NB algorithm has motivated research community to search for variations of NB (Langley et al. 1992; Kononenko 1991; Pazzani et al. 1996).

3 Variations of Naive Bayes

As pointed out earlier, real-world applications may not follow the associated assumptions of the NB. Features in practical data sets can be interrelated by violating the independent assumption. For an instance, Naïve Bayes classifiers are a popular statistical method of spam filtering, in which occurrences of terms like "online," "meds," "pharmacy" would not be independent from each other. Therefore, assuming the independence can introduce adverse effects for the accuracy of the classification.

Another major issue with the use of independent feature assumption is the difficulty for the learner to extract available hidden patterns within the data (Arar and Ayan 2017). If someone attempts to apply the NB ignoring the feature dependency, this can cause a considerable detriment in performance. Arar and Ayan (2017) state how performances are hindered due to the violation of Naïve Bayes assumptions. As an alternative approach, authors propose a novel feature-dependent NB approach to overcome the above issue. By applying this novel approach to a problem of predicting software defects, they receive appealing results compared

to the conventional NB. The literature indicates the various attempts to circumvent the above issue by seeking for alternatives. Researchers approach this problem in two directions to rectify the violation of "independent" assumption (Zaidi et al. 2013). The first method uses reduced conditional independent assumption, which is considered as semi-Naive Bayes (Langley and Sage 1994; Friedman and Goldszmidt 1996; Zheng et al. 1999, 2012; Cerquides and M'antaras,2005). The aim of the second approach is to use feature-weighting techniques in order to enhance the influence of highly predictive feature (Hilden and Bjerregaard 1976; Ferreira et al. 2001; Hall 2006; Zaidi et al. 2013). Let's discuss the above approaches in the following section.

3.1 Semi-Naive Bayes

The aim of these algorithms is to improve the accuracy of NB by weakening the conditional assumption. The term "semi-Naive Bayes" was introduced by Kononenko (1991) and aimed to discuss a semi-Naive Bayes classifier. In this study, the author computes conditional probabilities of joint features using training data and uses them to test cases instead of original. Author's work on data sets in medical domains shows slight improvements over the regular NB. In another study, Jin et al. (2004) develop a novel semi-Naive Bayes model for automatic image annotation for content-based image retrieval, using clustering with pairwise constraints. According to their experimental results, the novel model has performed considerably better than the conventional NB model. In a different study conducted by Carvajal et al. (2015), use semi-Naive Bayes models in their research to study the pathogen reduction and operating conditions used in waste water treatment. They point out that the use of these semi-NB models could reduce the costs for pathogen monitoring and reduction. Flores et al. (2014) discuss several semi-Naive Bayes approaches such as averaged one-dependence estimators (AODE), tree augmented Naive Bayes (TAN), and k-dependence Bayesian classifier (KDB). Robles et al. (2003) in their research paper propose semi-Naıve Bayes structures via estimation of distribution algorithms.

3.2 Weighted Naive Bayes

Usually, the aim of attribute weighting is to add more weight on highly predictive attributes and to place less weight on less predictive attributes. Note that in NB, we have $P(Y = y|X = x) = \frac{P(Y=y)\prod_{i=1}^{n} P(X=x_i|Y=y)}{\Sigma_i^C P(y_i, X=x)}$. Though $P(Y = y, X = x)$ is assumed to be independent in NB, in weighted NB this value is estimated by $\hat{P}(Y = y, X = x)$ by $\hat{P}(Y = y)\prod_{i=1}^{n} \hat{P}(X = x_i|Y = y)$. The latter expression is usually estimated by:

$$P(Y = y|X{=}x){=}\hat{P}(Y = y)\prod_{i=1}^{n} \hat{P}(X = x_i|Y = y)^{w_i}, \quad (2)$$

where w_i represents the weights assigned for each attribute. Zaidi et al. (2013) propose a novel approach with the argument of alleviating conditional dependence assumption by attribute weighting. Furthermore, they state that violation of NB assumption can be minimized by selecting the suitable weights w_i. In another study, Frank et al. (2002) use a locally weighted version of NB by relaxing the independence assumption. According to their experimental results, the proposed method performs better than the NB in empirically for the data sets they considered. Furthermore, they show this method exhibits robustness behavior and reduces the computing complexity. Zhang and Sheng (2004) investigate further into weighted NB method in order to rank data accurately. In this study, they utilize different techniques such as the gain ratio method, the hill climbing method, and the Markov Chain Monte Carlo method. Authors find that the hill climbing method outperforms its counterparts. One of the most important steps in weighted NB is the selection of weights that go with attributes. Lee et al. (2011) propose a novel approach for calculating the weights corresponding to the feature set using Kullback–Leibler measure. Empirical results indicate that this feature weighting method gives better performance than the traditional NB for numerous data sets. As the name implies in local learning approach, NB model is formed by taking the neighborhood instances in the test data set instead of considering the entire data set. This training data selection technique weakens the impact of attribute dependency that exists in the entire data set (Jiang 2011; Jiang et al. 2012). With the expectation of weakening the conditional independence assumption, Jiang et al. (2013) combine NB learning with locally weighted learning technique for text classification. According to their experimental results, this locally weighted approach shows a significantly better classification accuracy than the original NB text classifiers.

4 Applications of Naive Bayes classifier

This section introduces a few selected uses of NB. Due to the popularity of NB, its applications can be seen in a variety of different other fields not listed in this article. Characteristics of the NB algorithm such as the simplicity, speed of the execution, and the accuracy are the main reasons to see researchers applying NB in a variety of fields. This section presents some published work in the disciplines of software development, health care, cybersecurity, and education. It should be noted that this list is not exhaustive, and the aim of this paper is to provide a general overview of NB, its varia-

tions, applications, and vulnerabilities rather than reporting an exhaustive list of application domains.

4.1 Software defect prediction

Modern software has become very versatile and sophisticated. Due to this, identifying bugs in software before releasing is done in several stages and given high priority. The prediction of software defects is widely studied because of the enormous importance of it for the software industry. Naive Bayes is considered as the most (47.4 % of all studies) widely used learner group in software defect prediction (Arar and Ayan 2017). Menzies et al. (2006) research on defect predictors using static code attributes, which are easier to find compared to other labor-intensive methods. They expel some of the prior believes against the defect prediction by emphasizing the importance of the selection of attributes to build predictors. Furthermore, they obtain appealing results by using log-filtering preprocessor before applying NB algorithm on the data. Arar and Ayan (2017) use an algorithm called "feature-dependent Naive Bayes (FDNB)" to predict software defects. Implementing their proposed algorithm using a NASA PROMISE data sets, they obtain outstanding results that beat the results given by the conventional approach of NB. In their study, they question the linearity of the independent assumption of NB by evaluating the performance when handling features in pairs instead of individual features. Queiroz et al. (2016) examine how different machine learning techniques can be used to identify the defective features in open-source software development. Their aim is to study the defect prediction models in order to identify defective features in software production. In this study, the researchers use three machine learning classifiers, namely J48, random forest, and NB, and they find NB as the best classifier compared to the other counterparts in accurately predicting defects. In another study about software fault prediction, Catal and Diri (2009) use nine classifiers and find NB as the best prediction algorithm when using for data sets of smaller size. Furthermore, Veni and Srinivasan (2017) classify the type of defect as minor, moderate, or major defects using NB classifier.

4.2 Health care

Machine learning differs from conventional statistical procedures as the former lets the data to talk. With the digital recording of health-related data and patient's information, it creates a wonderful opportunity for researchers in machine learning to exploit hidden sophisticated associations and interactions among the data. The literature indicates the application of NB classifier in various areas related to health. Pattekar and Praveen (2012) develop a decision support system to predict heart disease based on NB classifier. The purpose of this system is to extract the unseen patterns and knowledge from heart disease database to make prediction for novel patients about the status of heart disease in an effective manner. A similar study is conducted by Vembandasamy et al. (2015) to use NB classifier to predict heart disease, and they experience higher success rate of accurate prediction over the traditional approaches. Apart from identifying heart diseases using NB classification, use in the area of liver diseases is another major application of NB in health-related fields. According to the published works related to liver disease, majority attempts are about the prediction of the disease. In order to predict three main liver diseases with the help of their distinct symptoms, Dhamodharan (2014) use NB and FT tree algorithm. By comparing the above two algorithms, authors find NB performs better than its counterpart to predict the three liver diseases: liver cancer, cirrhosis, and hepatitis. In a similar study, Vijayarani and Dhayanand (2015) use both NB and support vector machines (SVM) to predict liver diseases using related attributes such as patients' age, gender, and certain liver enzymes such as TB, DB, ALP, Sgpt, Sgot, TP, ALB, and A/G ratio. According to their experimental results, authors find SVM performs better than NB though the latter is faster in the prediction. Application of NB has not limited only for the detection of a specific disease. Marucci-Wellman et al. (2015) develop human–machine semi-automated system based on NB algorithm to classify injury narratives with the help of large administrative database. They find that the developed system shows overall accuracy of 87%. In another study, Liu et al. (2015) develop a patient-centered clinical decision support system using NB and cloud computing technology to diagnose deceases. A comprehensive overview of the use of NB in medical data mining is discussed by Al-Aidaroos et al. (2012). Furthermore, these authors use the experimental data to illustrate that how NB technique is better suited for the problems in medical domain.

4.3 Cybersecurity

An intruder typically aims to violate the integrity, availability, or confidentiality of the target system which can be a computer resource, computer network, or any other valuable information asset. Intrusion detection can be broadly classified as signature based and signatureless. NB offers great advantages for modern cybersecurity. Its applications can be found in both offensive and defensive aspects in security. Bayesian-based spam filtering, phishing attack detection, malware and ransomware detection, and network traffic monitoring (Kalutarage et al. 2015) are popular applications among them. Machine learning can be considered as popular approach to address signatureless intrusion detection due to the ability of generalizing both malware families and polymorphic strains that have never before seen (Anderson

et al. 2017). The malware detection can be considered as mathematical function in which the domain is the set of executable programs and the range is the status of the program (malicious or benign) (Vinod et al. 2009). Though the initial focus of malware detection algorithms was mainly toward the computers, after the mobile devices became more popular and sophisticated, the malware detection became a major concern for mobile devices as well. Detection of malware is not an easy task due to the vulnerabilities of malicious programs as they can bypass security measures implemented by the user (Ravi and Manoharan 2012). Comprehensive discussion about the applications of machine learning to improve malware detection is presented in Geigel (2013) and Geigel (2014). Sayfullina et al. (2015) apply both Bernoulli Naive Bayes and normalized Bernoulli Naive Bayes to classify Android Malware. They experience that normalized Bernoulli NB performs better than its counterpart in malware classification by giving an improved class separation with a higher accuracy. In another study, Shang et al. (2018) propose a novel algorithm based on improved NB classification for Android malware detection. In order to improve the efficiency of Android malware detection rate, authors use new malware permissions together with training permissions as the weight of the weighted NB algorithm. Their experimental results indicate higher detection rates. By utilizing frequent patterns and weighted NB, He et al. (2015) propose Android malware detection method to obtain improved results in malware detection. Chaba et al. (2017) in their study apply three algorithms, namely NB algorithm, random forest algorithm, and stochastic gradient descent algorithm in malware detection. They focus on studying the behavior of malicious software, while it is actually running on a host system. Kalutarage et al. (2017) utilize NB in detecting colluding attacks on Android applications. The unique features of the NB, such as the simplicity and speed of execution, have made it as the first choice for developers who develop security solutions for resource-constrained ubiquitous devices such as IoT devices.

4.4 Education

Education is no longer limited to textbook-based teaching, memorizing the material, and testing in examinations. Ignorance of testing student's ability to comprehend the information they are taught and apply the concepts in real-life situations (Nafea 2018) is one of the major accusation over the traditional education. Furthermore, the traditional education system does not cater all levels of students as it is more instructor driven and the student has little chance to control the phase of learning. With the dynamic development of the computer technology, there is a drastic change even in the education sector as well. The impact of the technology has influenced various sectors in education including the process of planning, implementing, evaluating, following up, and developing objectives Lv and Li (2015). The influence of advanced machine learning algorithms has changed the education process by introducing a personalized learning environment where student has a more control over the education process. This has facilitated close monitoring of student's understanding, which has helped the student to comprehend concepts by leaving no student behind. Prediction of students' academic success is very important as it gives the educators the opportunity to give extra attention toward students, who predicts unsatisfactory performance in the future. Razaque et al. (2017) attempt to predict the students graduate point average (GPA) based on students' prior performance in quiz, discussion, assignments, attendance, and laboratory work. In another study, Devasia et al. (2016) develop a Web-based system to predict student's end of the semester performance by taking student's academic history such as the details of admission, course, grades, and attendance in to consideration. Their results indicate that NB algorithm provides higher accuracy over other techniques, namely multiple regression, decision tree, neural networks, etc., for comparison and prediction. Identifying dropouts in education process also has a major importance. The aim of the study conducted by Yukselturk et al. (2014) is to make prediction about the dropouts in an online program based on the attributes they collected over students. They use several machine learning techniques including NB in this process.

5 Strengths and weaknesses of Naive Bayes and some guidelines

5.1 Robustness of Naive Bayes

A good classification algorithm features with less errors in data classification. A classification algorithm can exhibit errors mainly due to three reasons: the noise in the initial sample, statistical bias, and the variance (Friedman and Goldszmidt 1996). Robustness of a classification algorithm refers to the tolerance of the algorithm for errors, during the execution (Carbin and Rinard 2010; Danglot et al. 2018). One of the main features of NB classification technique is the robustness. Though the data deviate from the underline assumptions of NB model, still NB can perform really well. As majority of naturally occurring data sets do not have independent features, researchers at times tend to choose more complected models over NB due to violation of assumptions. Surprisingly, basic model comparison studies show that this model performs extremely well irrespective of the violation of assumptions (Hand and Yu 2001).

Using a data from medical domain about severe head injuries, Titterington et al. (1981) find independent model performed better than other counterparts. Practical data sets like these, usually, consist of dependencies and missing val-

ues, but their outcomes do not show any hindrance in the performance with NB. More studies in medical domain such as breast cancer (Mani et al. 1997), heart disease (Russek et al. 1983; Vembandasamy et al. 2015), thyroid (Nordyke et al. 1971), liver disease (Dhamodharan 2014; Croft and Mitchol 1987), and abdomen (Gammerman and Thatcher 1991; Todd and Stamper 1994; Ohmann et al. 1996) confirm that NB is the best model compared to their counterparts. In other applications such as email spam filtering, though the selected features are not independent, the NB works exceptionally well.

Though many studies seek for explanations to understand the robustness of NB, none of them have been able to establish a necessary and sufficient conditions for the optimal behavior of the model (Kuncheva 2006). Even though it is not generalized for more than two binary features, using a two-class two-binary feature problem Kuncheva (2006) shows that when the dependencies on both classes are same with both features, NB reaches optimal. According to the studies (Hand and Yu 2001; Domingos and Pazzani 1996; Zhang 2004; Rish 2001), an interesting yet an important outcome suggests that the conditional independence is only a sufficient condition for the optimal of the NB, but not a necessary condition.

Nevertheless, there are several more reasons behind the success of NB. First reason is the process of preselection of features. In this process as the correlated features are eliminated, the impact of the assumption violation becomes minimal. It is a fact in data analytic, the selection of highly correlated features can degrade the accuracy of the prediction. Secondly, the NB model is less prone to over-training, mainly for small sample data (Kuncheva 2006). Hand (1992) states that the NB needs relatively less parameters to be estimated compared to other alternative methods.

5.2 How does Naive Bayes perform well with missing values?

Missing values in a data set can hinder the classification performance. Fortunately, this issue is negligible for NB. In order to understand how this takes place in NB, let's consider the feature set $X = (X_1, X_2, \ldots, X_k, \ldots, X_n)$. Furthermore, let's assume the value of X_k is missing as well. Consider the conditional probability, $P(X|y) = P(X_1 = x_1, X_2 = x_2, \ldots, X_k = ?, \ldots, X_n = x_n|y)$. Without knowing the value of X_k, it is very difficult to calculate the above conditional probability. As an alternative, one needs to consider all possible cases of X_k in order to complete the probability calculation. Consider $X_k \in \{x_{k1}, x_{k2}, \ldots, x_{km}\}$ for some $m \in N$. Then, we can calculate the above conditional probability as follows:

$$
\begin{aligned}
P(X|y) &= P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n|y) \\
&= P(X_1 = x_1|y)P(X_2 = x_2|y) \\
&\quad \cdots P(X_n = x_n|y) \\
&= \Sigma_{j=1}^{m} P(X_1 = x_1|y) \cdot P(X_2 = x_2|y) \\
&\quad \cdots P(X_k = x_{kj}|y) \cdots P(X_n = x_n|y) \\
&= P(X_1 = x_1|y)P(X_2 = x_2|y) \\
&\quad \cdots \Sigma_{j=1}^{m} P(X_k = x_{kj}|y) \cdots P(X_n = x_n|y) \\
&= P(X_1 = x_1|y)P(X_2 = x_2|y) \\
&\quad \cdots P(X_n = x_n|y)\Sigma_{j=1}^{m} P(X_k = x_{kj}|y) \\
&= P(X_1 = x_1|y)P(X_2 = x_2|y) \\
&\quad \cdots P(X_n = x_n|y) \\
&= \prod_{i=1}^{n} P(X = x_i|y). \qquad (3)
\end{aligned}
$$

5.3 When does the Naive Bayes perform well and poorly?

Using an artificial domains and following a theoretical approach, Domingos and Pazzani (1996) study the NB algorithm, the optimally and other related conditions of it. This detailed study shows that in order to perform optimal under zero loss, attribute independence is not required. Furthermore, the study shows that NB becomes an optimal learner for conjunctive and disjunctive concepts, in spite of the violation of the independence assumption. In addition, Kohavi (1996) observes that Bayesian classifier outperforms on smaller data sets in the range of hundreds to thousands of examples. This goes to confirm the early findings of Domingos and Pazzani (1996). Using Monte Carlo simulations, Rish (2001) studies about the NB and the state of the best and the worst performance criteria. According to the above study, the author confirms that NB performs best when features are completely independent and functionally dependent, though the latter is surprising. In the mean time, the author points out that the classifier has its worst performance between the above extremes.

If zero observations problem occurs, NB will not be reliable. In general, the problem of zero observation is that your training data lack certain cases that are actually possible. In this case, it is recommended to assign some low probabilities to such cases, as it does not seem appropriate to assign a probability of zero elements (Cover and Thomas 2012). If the feature set contains highly correlated features, the performance of NB may also be affected as correlated features voting twice in the model. In such cases, it is recommended to remove strongly correlated features.

5.4 Naive Bayes is vulnerable to attacks

Recent research demonstrates that MLDA algorithms are vulnerable to attacks. NB has no exception. An adversary can launch several attacks against these systems by examining

weaknesses of learning algorithms. If an adversary can determine a particular behavior of the learning algorithm that its developers do not know, then she can use that behavior to get a potential return. Bayesian poisoning is such a technique used by email spammers to reduce the effectiveness of spam filters based on Bayesian spam filtering. To this end, for example, attackers attempt to create a series of spam emails that are identified as ham and endanger the integrity of the system. The spammer hopes that adding random (or carefully selected) words that are not likely to appear in a spam message will cause the spam filter to consider the message legitimate (type II error). As the literature indicates, Bayesian-based spam filters are particularly vulnerable to these types of attacks. Therefore, NB-based classifiers are expected to have rigorous robustness that allows them to be robust enough to deal with these types of hostile forces. Demonstration of Bayesian poisoning attack with relevant code snippets can be found in Sect. 6.4 of this paper. As the security systems have became more sophisticated, attackers are continuously attempting to intrude security systems in many ways. One of the main ways a machine learning-based security system becomes vulnerable is due to the misinterpretation of inputs into the system, and it behaves supportive to attackers. As discussed before, the accuracy of the NB classifier depends on the conditional probability of,

$$P(\text{email is a spam or a non-spam} \mid \text{email has a set of specific words}).$$

Spammers frequently attempt to bypass the spam filters by abusing the above conditional probability. According to Xiao (2017), spammers attempt to obfuscate the spam filter by deliberately adding more legitimate words in their spams. Their aim is to increase the probability of good words so that there is the likelihood of classifying the email as non-spam (ham). Another approach is to alter the words that filters are searching to classify the email as a spam. For an instance, Lowd and Meek (2005) point out that spammers use "vi@gra" instead of "viagra," in order to sneak through the system. A similar study is explored by Wittel and Wu (2004), where the researches consider the replacement of spam words with a list of common English words. According to the outcomes of Wittel and Wu (2004), spammers can get 50% of the blocked spams through the filter by adding a number of easily found words. Further studies about the issues of spam filtering can be found in (Xiao 2017).

6 Code snippets for implementation

Up to now, we have discussed about various applications of NB and its vulnerabilities. This section aims to provide code snippets to demonstrate to the reader how NB classifiers can be trained for selected applications and to show that how vulnerabilities can occur in practice. It should be noted that building and deploying NB-enabled systems in industry settings would be more complicated than these simplified examples (toy examples), and such a discussion is out of the scope of this work.

Here, we present three examples from different areas of application and also demonstrate a Bayesian poisoning attack. These examples have been written in R language. Due to the space constraint, no intermediate outputs generated by each code snippets are shown in the paper. Only the confusion matrix and performance statistics are presented at the end of each example. However, complete documentation with all intermediate outputs can be accessed via https://github.com/HarshaKumaraKalutarage/Naive-Bayes-Applications-and-Vulnerabilities. It also provides both R scripts and data sets so that readers can easily clone and run these examples on their systems if necessary.

6.1 Example 1: Iris data set

In this example, we use well-known Iris data set (included with R) to train a NB classifier. The data set consists of four features (measurements): sepal length, sepal width, petal length, and petal width of 150 Iris owers. It contains information about three types of Iris plants: Setosa, Versicolor, and Virginica. We are going to train a NB model to identify (classify) the species of an Iris flower, based on the values of the four features of a given Iris flower. Note that in this example all the features are numerical variables. In R language, likelihoods for numerical features (e.g., measurements) in the Naive Bayes formula are calculated using probability distributions. Listings 1–3 provide necessary code snippets to complete this classification task as described below.

```
set.seed(12345) #set a seed value to ensure results are
    reproducible
data(iris) #attach the data set to the R environment
myData <- iris #rename the data set as myData
dim(myData) #check dimensions of myData
sapply(myData, class) #check the data types of each
    feature
levels(myData$Species) #different levels (values) in
    label column
head(myData) #look at top data points in mydata
summary(myData) #produce descriptive statistics
```

Listing 1 R code snippet: Load the Iris data and produce summary statistics

Partitioning the data set and training a NB classifier: We are going to follow the convention of 80:20 samples ratio in partitioning the data set to the training (traintSet) and testing (testSet) sets. For this purpose, we use the createDataPartition function in the caret package. Then, we train our NB classifier with the trainSet. TestSet is used to evaluate the performance of the fitted model. For this task, we use the e1071 package in R language and Listing 2 contains the necessary code.

```
library(caret) #load the library
tr_index <- createDataPartition(myData$Species, p=0.80,
    list=FALSE) #list of 80
trainSet <- myData[tr_index,] #select 80
testSet <- myData[-tr_index,] #select the remaining 20
library(e1071)
NBclassfier=naiveBayes(Species~., data=trainSet) #train a
    NB model
print(NBclassfier) #check model details
```

Listing 2 R code snippet: Partition the data set and train a NB classifier

Making predictions: Let's apply the above model to assign labels for test cases. Then, we create the confusion matrix, a table that is often used to describe the performance of a classifier. Listing 3 presents necessary code to perform this task.

```
testPrediction=predict(NBclassfier, newdata=testSet, type
    ="class") #assign labels for test cases
confusionMatrix(testPrediction, testSet$Species) #print
    the confusion matrix
```

Listing 3 R code snippet: Make predictions and produce the confusion matrix

As can be seen in the confusion matrix, the accuracy of the fitted model in this example is 96.67%.

6.2 Example 2: Breast cancer detection

Early detection of breast cancer is crucial for successful treatment. Conventional methods such as breast biopsy are more invasive and must be performed by a human specialist, which would be time consuming and not scalable. However, samples obtained with less invasive techniques like fine needle aspiration can be easily digitized and used for computer-aided diagnosis. To this end, the use of machine learning methods can significantly reduce the cost and time for the diagnostic process. This code snippet shows the reader how to train a Naive Bayes (NB) classifier for breast cancer detection. We utilize the Breast Cancer Wisconsin (Diag-

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         1
##   virginica       0          0         9
##
## Overall Statistics
##
##                Accuracy : 0.9667
##                  95
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 2.963e-13
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9000
## Specificity                 1.0000            0.9500           1.0000
## Pos Pred Value              1.0000            0.9091           1.0000
## Neg Pred Value              1.0000            1.0000           0.9524
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.3000
## Detection Prevalence        0.3333            0.3667           0.3000
## Balanced Accuracy           1.0000            0.9750           0.9500
```

nostic) data set available at http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/. Our aim in this application is to classify a tumor as benign or malignant; hence, machine learning (ML) task in this problem would be a classification. Listings 4-6 provide necessary code snippets to perform this task. Comments are only provided for the newly appeared syntax in each listing. Readers should refer to the previous listings for comments on repetitive syntax.

```
set.seed(12)
myData <- read.csv("CancerData.csv", header=T)
dim(myData)
sapply(myData, class)
levels(myData$label)
head(myData)
summary(myData)
prop.table(table(myData$label)) #look at ratio between
    classes
library(corrplot) #load libraries to plot correlation
library(RColorBrewer)
corrVal <-cor(myData[,3:ncol(myData)])#plot correlations
corrplot(corrVal, type="upper", order="hclust", col=
    brewer.pal(n=10, name="RdYlBu"))
```

Listing 4 R code snippet: Load the data set and produce a correlation matrix

```
## Ratio between two classes
##         B         M
## 0.6274165 0.3725835
```

We noticed that the data in this example are slightly imbalanced and there are some highly correlated features (Fig. 1). As mentioned in Sect. 5.1, highly correlated features can be badly affected on the performance of a NB model and therefore should be treated accordingly. In order to keep our code short and simple, however, in this example, we do not remove correlated features nor deal with the problem of class imbalance. We use the entire data set to train and test the NB model for the purpose of demonstration the code.

Partitioning the data set and training a NB classifier: As in the previous example, we are going to construct a 80:20 partitioning for the training (trainSet) and validation (testSet) sets. After partitioning the data set, we will train a NB classier. Listing 5 provides the code snippet for this task.

```
library(caret)
tr_index <- createDataPartition(myData$label, p=0.80,
    list=FALSE)
trainSet <- myData[tr_index,]
testSet <- myData[-tr_index,]
library(e1071)
NBclassfier=naiveBayes(label~., data=trainSet[,2:ncol(
    trainSet)])
print(NBclassfier)
```

Listing 5 R code snippet: Partition the data set and train a NB classifier

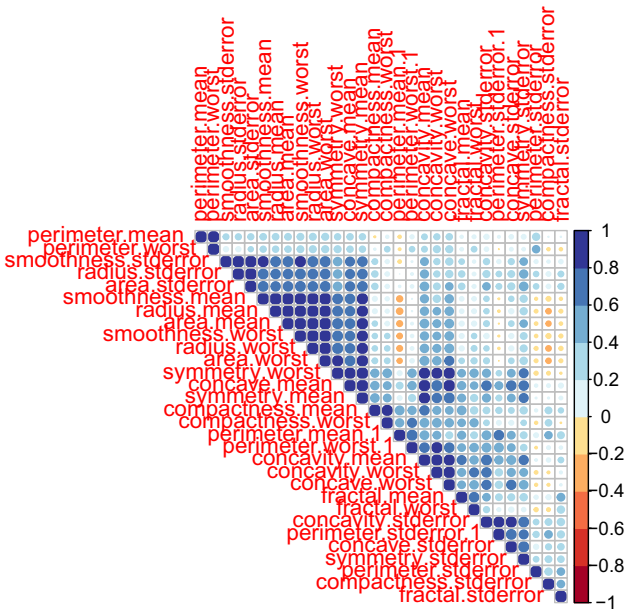

Fig. 1 Correlation matrix: breast cancer diagnostic data set

Making predictions: Listing 6 applies the newly fitted model to the test cases and creates the confusion matrix for performance comparison.

```
testPrediction=predict(NBclassfier, newdata=testSet[,2:
    ncol(testSet)], type="class")
confusionMatrix(testPrediction, testSet$label, positive =
    'M')
```

Listing 6 R code snippet: Make predictions and produce the confusion matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 68  4
##          M  3 38
##
##                Accuracy : 0.9381
##                  95
##     No Information Rate : 0.6283
##     P-Value [Acc > NIR] : 1.718e-14
##
##                   Kappa : 0.8667
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9048
##             Specificity : 0.9577
##          Pos Pred Value : 0.9268
##          Neg Pred Value : 0.9444
##              Prevalence : 0.3717
```

```
##          Detection Rate : 0.3363
##    Detection Prevalence : 0.3628
##       Balanced Accuracy : 0.9313
##
##        'Positive' Class : M
##
```

In this example, we can see that the accuracy is 93.81%. It was a small test set; however, whether this accuracy is sufficient or not depends on the problem context and is based on many other factors, such as the costs of misclassification. For example, the cost of false positives and negatives in medical diagnosis can be fatal to the patient.

6.3 Example 3: Malware classification

This example explains how to apply NB to a problem in the cybersecurity domain, which is malware classification. Here, we show you how to train the NB algorithm using various features in malware. It contains features that obtain from static code analysis and analysis of the dynamic behavior of the malware. The data set used for this analysis can be accessed via https://data.mendeley.com/datasets/w2w8gjsgnt/1. It contains a total of 1944 features that were obtained from static and dynamic code analysis of $\approx$ 19400 malware samples, including malware related to advanced persistent threat (APT) attacks. These malwares were labelled as worm, back door, other type, rootkit, spyware, trojan, and unknown. In our analysis, we preprocess the data set by removing the first three columns and the last seven columns and adding a label column to the end of the data set. The ML task in this example is also a classification. Listings 7–13 contain the code snippets required to accomplish this task using a NB model.

```
set.seed(1234)
myDataOrg <- read.csv("malware.csv", header=T)
dim(myDataOrg)
levels(myDataOrg$label)
```

Listing 7 R code snippet: Load the data set and produce summary statistics

Checking the balance of the data set: As mentioned in the previous example, class imbalance can be negatively affected on the performance of an NB algorithm. Class imbalance is a very common problem in cybersecurity data sets. Due to the scarcity of attack data, a ratio of 1:100000 between attack and normal classes is very common in these data sets. Listing 8 shows the ratio between classes in our data set.

```
print(table(myDataOrg$label))
```

Listing 8 R code snippet: Look at the ratio between classes

```
##  Backdoor OtherType   Rootkit   Spyware    Trojan   Unknown      Worm
##        53      1275       789       709     11034      3957      1640
```

Addressing the class imbalance issue: Obviously, our data set is an enormous imbalance, especially Backdoor:Trojan. This can affect the model performance. For example, we tried a NB model for this original imbalanced data set and the maximum accuracy we could achieve was $\approx$ 34%. To reduce the impact of class imbalance, various methods have been proposed. For example, a mix of oversampling and undersampling methods could be used, i.e., by increasing the size of Backdoor class and reducing the size of Trojan class. However, this can be resulted information lost from the larger (Trojan) class. Therefore, we will split the data set into two subsets and train two NB models separately (ensemble technique). Of course, if NB does not work very well with one data set, you can train another model (e.g., random forest) for that data set and combine NB with the other model for better performance. Listing 9 provides the code snippet for splitting the data set.

```
myDataSet1<-rbind(subset(myDataOrg, label == "Backdoor"),
    subset(myDataOrg, label == "OtherType"),subset(
    myDataOrg, label == "Rootkit"),subset(myDataOrg,
    label == "Spyware"))
myDataSet2<-rbind(subset(myDataOrg, label == "Trojan"),
    subset(myDataOrg, label == "Unknown"),subset(
    myDataOrg, label == "Worm"))
```

Listing 9 R code snippet: Split the data set into two subsets

We split the original data set into two subsets with similar class sizes in each subset. Let's continue with myDataSet1. You can follow the same approach for myDataSet2 or fit a different model as mentioned above. The code in Listing 10 makes class sizes equal in myDataSet1.

```
sample.df <- function(df, n) df[sample(nrow(df), n,
    replace = T), , drop = F] #function to sample data
    from a data frame
classSize<-500 #sample size for each class
myData<-rbind(sample.df(subset(myDataSet1, label = "
    Backdoor"), classSize),sample.df(subset(myDataSet1,
    label == "OtherType"), classSize),sample.df(subset(
    myDataSet1, label == "Rootkit"), classSize),sample.
    df(subset(myDataSet1, label == "Spyware"), classSize
    ))
```

Listing 10 R code snippet: Make class sizes equal via oversampling and undersampling

NB model for malware classification: In order to fit a NB model to this data set, we want to represent the presence or absence of a certain static/dynamic feature in the malware. Hence, we code our data set as shown in Listing 11.

```
convert_counts <- function(x) {#define a function to code
    the data
  x <- ifelse(x > 0, "Y", "N")
}
myData <- data.frame(apply(myData[,1:1934], MARGIN = 2,
    convert_counts),myData[1935] ) #apply above function
    across columns
```

Listing 11 R code snippet: Code the data set

Creating training and validation data sets: Even in this example, we are going to follow the convention of 80:20 samples ratio to partition the data set to the training and validation sets. Listing 12 uses the createDataPartition function in caret package for this purpose.

```
library(caret)
tr_index <- createDataPartition(myData$label, p=0.80,
    list=FALSE)
trainSet <- myData[tr_index,]
testSet <- myData[-tr_index,]
```

Listing 12 R code snippet: Partition the data

Training a NB classifier and making predictions: Now, we will train our NB classier using the above trainSet and apply it to the test set. Listing 13 provides the necessary code for this purpose.

```
library(e1071)
NBclassfier <- naiveBayes(trainSet[,1:1934], trainSet$
    label)
testPrediction <- predict(NBclassfier, testSet[,1:1934])
confusionMatrix(testPrediction, testSet$label)
```

Listing 13 R code snippet: Train a NB classifier and make predictions

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Backdoor OtherType Rootkit Spyware Trojan Unknown Worm
##   Backdoor        86        13      15      14      0       0    0
##   OtherType        0        50       2       4      0       0    0
##   Rootkit         13        24      68      21      0       0    0
##   Spyware          1        13      15      61      0       0    0
##   Trojan           0         0       0       0      0       0    0
##   Unknown          0         0       0       0      0       0    0
##   Worm             0         0       0       0      0       0    0
##
## Overall Statistics
##
##                Accuracy : 0.6625
##                  95
##     No Information Rate : 0.25
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.55
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Backdoor Class: OtherType Class: Rootkit
## Sensitivity                   0.8600           0.5000         0.6800
## Specificity                   0.8600           0.9800         0.8067
## Pos Pred Value                0.6719           0.8929         0.5397
## Neg Pred Value                0.9485           0.8547         0.8832
## Prevalence                    0.2500           0.2500         0.2500
## Detection Rate                0.2150           0.1250         0.1700
## Detection Prevalence          0.3200           0.1400         0.3150
## Balanced Accuracy             0.8600           0.7400         0.7433
##                      Class: Spyware Class: Trojan Class: Unknown
## Sensitivity                  0.6100            NA             NA
## Specificity                  0.9033             1              1
## Pos Pred Value               0.6778            NA             NA
## Neg Pred Value               0.8742            NA             NA
## Prevalence                   0.2500             0              0
## Detection Rate               0.1525             0              0
## Detection Prevalence         0.2250             0              0
## Balanced Accuracy            0.7567            NA             NA
##                      Class: Worm
## Sensitivity                   NA
## Specificity                    1
## Pos Pred Value                NA
## Neg Pred Value                NA
## Prevalence                     0
## Detection Rate                 0
## Detection Prevalence           0
## Balanced Accuracy             NA
```

So far in this example, we have trained and tested a NB model for the first subset (myDataSet1) in our original data set. Readers can repeat the same approach by fitting a NB model for the second subset (myDataSet2) and then combining both models using an ensemble method.

6.4 Example 4: Bayesian Poisoning attack

The purpose of this code snippet is to demonstrate to the reader how a NB classier is vulnerable to Bayesian poisoning attacks. Bayesian poisoning is a technique used by email spammers to compromise the effectiveness of Bayesian-based spam lters by adding non-spamming (ham) like words at the end of a spam message. The aim of spammers is to hamper the probability calculation associated with the NB algorithm. In other words, this increases the type II probability. In hypothesis testing, type II error is referred to as "false negative," which means not rejecting the null hypothesis when it is false. Therefore, as a result of the Bayesian poisoning, the spammers make the spam filtering system to consider a non-legitimate message a legitimate one.

For the implementation of this code, we use the R language and a data set which can be accessed via https://www.kaggle.com/venky73/spam-mails-dataset. This data set consists of a collection of spam and non-spam email messages. For the purpose of demonstration, we will first develop a NB-based spam filter using the above email repository (raw emails) and then conduct a Bayesian poisoning attack to circumvent the security control. To this end, we poison the test data into the spam lter and bypass the security control. Listings 14-19 present code snippets required to develop a NB-based spam filter, while Listings 20-24 present the necessary steps to perform the Bayesian poisoning attack. As in other examples, comments are only provided for the new syntax. Readers should refer to the previous listings for comments on the repeating syntax.

```
library(tm) #load the necessary libraries
library(wordcloud)
library(e1071)
library(gmodels)
library(SnowballC)
library(caret)
myData <- read.csv("spam_ham_dataset.csv")
myData<-myData[,c("label","text")] # select two columns
myData$text<-substring(myData$text,9) #remove constant
    words
```

Listing 14 R code snippet: Load the data and remove constant words

Code snippet in Listing 14 loads the data set to the R environment, and remove the terms like "Subject:." There is no point of keeping common constant terms like "Subject:" as they do not have discrimination power, since appearing in both ham and spam messages.

```
spamMsg<-subset(myData,label=="spam")
hamMsg<-subset(myData,label=="ham")
wordcloud(spamMsg$text,scale=c(4,.5),min.freq=5) #plot if
    the word appears more than 5 times in spam messages
wordcloud(hamMsg$text,scale=c(4,.5),min.freq=5) #plot if
    the word apperas more than 5 times in the ham
    messages
```

Listing 15 R code snippet: Produce word clouds

Listing 15 checks the word distributions (a.k.a. document terms) in each class. For this purpose, we use word clouds in which the size of the word is proportional to its frequency in the text. If the word clouds are different, we can conclude that words that appear frequently in spam differ from words that appear in ham. Therefore, we can use words as features when training a NB model to classify spam from ham.

As we can see in word cloud outputs (Fig. 2), the term distributions differ between spams and hams, so we can train an ML model (in our case, NB) by using words as features.

Building a NB-based spam filter: Listings 16-19 contain the necessary steps to train a spam filter using the above data set. To do this, we first need to create a Document-Term Matrix (DTM)—a matrix that describes the frequency of terms that appear in every message in our collection, and then use it to train and test our models. DTM is usually a sparse matrix with most of its entries which are filled with zeros. Therefore, in this example, we will only use terms whose frequency is $>= 5$, so that we can significantly reduce the number of columns in our DTM matrix, which would lead to a manageable size DTM when training our NB model.

```
myCorpus <- VCorpus(VectorSource(myData$text)) # create a
    corpus
myDTM<- DocumentTermMatrix(myCorpus, control = list(
  tolower=T,
  removeNumbers=T,
  removePunctuation=T,
  stopwords = T,
  stem=T
))
freqWords <- findFreqTerms(myDTM,5) #remove low frequent
    words
myDTM<- myDTM[,freqWords]
```

Listing 16 R code snippet: Create a Document-Term Matrix (DTM)

Once DTM is ready, we need to split it into train and test sets. We are going to use usual 80:20 partitioning for this purpose. Listing 17 provides code snippet for this purpose.

```
tr_index <- createDataPartition(myData$label, p=0.80,
    list=FALSE)
trainSet <- myData[tr_index,]
testSet <- myData[-tr_index,]
```

Listing 17 R code snippet: Partition the data set

As the NB model is created with DTM entries, we should find the corresponding DTM entries for the data in training

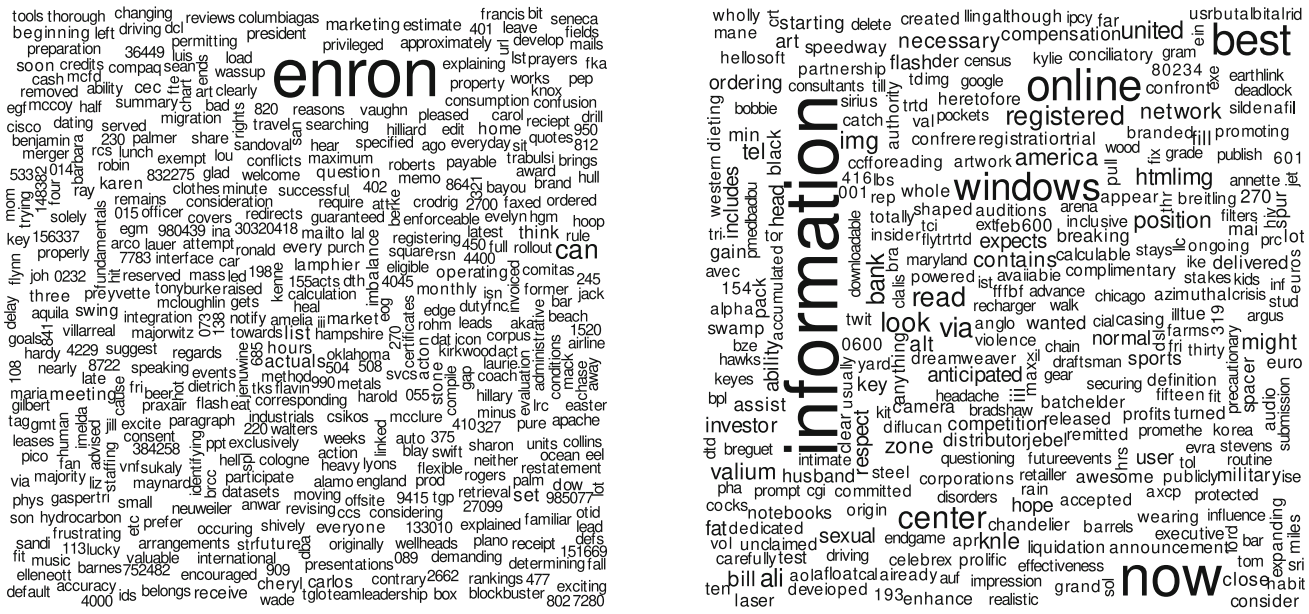

Fig. 2 Word cloud outputs: ham (left) and spam (right) messages

and test sets. Then, we want to represent the presence or absence of a particular word (feature) in a particular message so that we code our DTM entries as shown in Listing 18.

```
myDTMTrain <- myDTM[tr_index,] #corresponding DTM entries
    for training
myDTMTest <- myDTM[-tr_index,] #corresponding DTM entries
    for testing

convert_counts <- function(x) { #function to code DTM
    entries
  x <- ifelse(x > 0, "T", "F")
}
myDTMTrainNew <- apply(myDTMTrain, MARGIN = 2,convert_
    counts) #apply above function across columns
myDTMTestNew <- apply(myDTMTest, MARGIN = 2, convert_
    counts)
```

Listing 18 R code snippet: Partition the DTM and code its entries

Finally, Listing 19 trains and tests the NB-based spam filter.

```
NBbasedSpamFilter <- naiveBayes(myDTMTrainNew, trainSet$
    label)
testPredictMsgLabel <- predict(NBbasedSpamFilter,
    myDTMTestNew)
confusionMatrix(testPredictMsgLabel, testSet$label,
    positive = "spam")
```

Listing 19 R code snippet: Train and test a NB-based spam filter

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ham spam
##        ham  704   67
##       spam   30  232
##
##                Accuracy : 0.9061
##                  95
##     No Information Rate : 0.7106
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.763
##
##  Mcnemar's Test P-Value : 0.0002569
##
##             Sensitivity : 0.7759
##             Specificity : 0.9591
##          Pos Pred Value : 0.8855
##          Neg Pred Value : 0.9131
##              Prevalence : 0.2894
##          Detection Rate : 0.2246
##    Detection Prevalence : 0.2536
##       Balanced Accuracy : 0.8675
##
##        'Positive' Class : spam
##
```

As can be seen in the confusion matrix, the accuracy of the fitted model is 90.61%, and 67 false negatives are also generated. In the rest of this example, we will attempt to increase the false negative number by poisoning the test cases.

Poisoning test data: Let's poison the test data to bypass the above spam filter. To this end, randomly selected non-spam messages are amended at the end of spam messages in our test data set as shown in Listing 20.

```
spamTestCases <- subset(testSet,label=="spam") #select spams in test cases
hamTestCases <- subset(testSet, label=="ham") #select ham in test cases
hamMsgInmyData<-subset(myData, label=="ham") #select ham in our original data set
hamMsg2ammend<-hamMsgInmyData[sample(nrow(hamMsgInmyData), nrow(spamTestCases)), ] #select number of ham messages equal to the spam messages in test cases
spamTestCases$text <-paste(spamTestCases$text, hamMsg2ammend$text,sep = " ") #amend selected ham at the end of spams in test cases
poisTestData<-rbind(hamTestCases,spamTestCases) #create the poisoned test set
myData[-tr_index,]<-poisTestData #replace the test entries in original data set with poisoned test cases
```

Listing 20 R code snippet: Poisoning test data

Create the new DTM with poisoned data and select high-frequency words. Note that we have to create the DTM and split it as we did with original data in the above. Listing 21 provides the necessary code for this task.

```

myCorpusPoisoned <- VCorpus(VectorSource(myData$text))
myDTMPoisoned <- DocumentTermMatrix(myCorpusPoisoned, control = list(
  tolower=T,
  removeNumbers=T,
  removePunctuation=T,
  stopwords = T,
  stem=T
))
freqWords <- findFreqTerms(myDTMPoisoned,5)
myDTMPoisoned <- myDTMPoisoned[,freqWords]
myDTMTestPoisoned <- myDTMPoisoned[-tr_index,] #new test set, note that we don't need a training set as we use the same model created in Listing 19
myDTMTestNewPoisoned <- apply(myDTMTestPoisoned, MARGIN = 2, convert_counts) #use the same convert_counts function, we code our poisoned test DTM as did in Listing 18 above.
```

Listing 21 R code snippet: Create a test DTM with poisoned test data

Listing 22 predicts labels for poisoned test cases; note that we use the same NB model (NBbasedSpamFilter) trained in Listing 19.

```
poisonedTestPredictMsgLabel <- predict(NBbasedSpamFilter, myDTMTestNewPoisoned)
confusionMatrix(poisonedTestPredictMsgLabel, testSet$label, positive = "spam")
```

Listing 22 R code snippet: Predict labels for poisoned test data

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction ham spam
##       ham  641  256
##       spam  93   43
##
##                Accuracy : 0.6621
##                  95
##     No Information Rate : 0.7106
##     P-Value [Acc > NIR] : 0.9997
##
##                   Kappa : 0.0204
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.14381
##             Specificity : 0.87330
##          Pos Pred Value : 0.31618
##          Neg Pred Value : 0.71460
##              Prevalence : 0.28945
##          Detection Rate : 0.04163
##    Detection Prevalence : 0.13166
##       Balanced Accuracy : 0.50855
##
##        'Positive' Class : spam
##
```

As you can see in the confusion matrix, the misclassification of spam as ham (false negative) increases from 67 to 256. By adding more ham-like words to the end of spam messages, the attacker can further increase the false negative number. As you see in this example, identifying Bayesian poisoning is imperative. The impact of the misclassification depends on the domain of the data set. In particular, applications in medicine and security and the costs of false alarms (specially false negatives) would be very high. Therefore, special care should be given to mitigate the risk of Bayesian poisoning attacks in your NB application development.

7 Conclusion

This article discussed novel applications of Naves Bayes in variety of different fields. It is evident that various novel applications in variety of fields include software defect prediction, health, cybersecurity, and education. Naive Bayes performs exceptionally well compared to other classifiers when the assumptions are satisfied. Furthermore, NB shows the robustness due to the higher level of accuracy even with the violation of assumptions. As the literature indicates, NB performs better than the other counterparts under the same circumstances. Empirical research findings suggest that for data sets in smaller size, NB outperforms other alternative techniques. It is evident that most of natural occurring data may not follow the assumptions of NB. In situations like these, one can stick into NB due to its robustness. Alternatively, one can use the variations of NB such as semi-Naive Bayes and weighted Naive Bayes as the classification tech-

niques. Finally, we hope the reader of this manuscript will be able to gather both theoretical and practical aspects of topics of NB and Bayesian poisoning.

Acknowledgements This work has been partly funded by Burroughs Wellcome Fund, and we are thankful to the funder for their support.

Compliance with ethical standards

Conflict of interest Author Indika Wickramasinghe declares that he has no conflict of interest. Author Harsha Kalutarage declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

References

Al-Aidaroos K, Bakar A, Othman Z (2012) Medical data classification with Naive Bayes approach. Inf Technol J 11(9):1166–1174

Anderson HS, Kharkar A, Filar B, Roth P (2017) Evading machine learning malware detection. Black Hat, London

Arar ÖF, Ayan K (2017) A feature dependent Naive Bayes approach and its application to the software defect prediction problem. Appl Soft Comput 59:197–209

Carbin M, Rinard MC (2010) Automatically identifying critical input regions and code in applications. In: Proceedings of the 19th international symposium on software testing and analysis. ACM, pp 37–48

Carvajal G, Roser DJ, Sisson SA, Keegan A, Khan SJ (2015) Modelling pathogen log10 reduction values achieved by activated sludge treatment using naïve and semi naïve bayes network models. Water Res 85:304–315

Catal C, Diri B (2009) A systematic review of software fault prediction studies. Expert Syst Appl 36(4):7346–7354

Chaba S, Kumar R, Pant R, Dave M (2017) Malware detection approach for android systems using system call logs. arXiv:1709.08805

Clark P, Niblett T (1989) The CN2 induction algorithm. Mach Learn 3(4):261–283

Cover TM, Thomas JA (2012) Elements of information theory. Wiley, Hoboken

Danglot B, Preux P, Baudry B, Monperrus M (2018) Correctness attraction: a study of stability of software behavior under runtime perturbation. Empir Softw Eng 23(4):2086–2119

Devasia T, Vinushree T, Hegde V (2016) Prediction of students performance using educational data mining. In: 2016 International conference on data mining and advanced computing (SAPIENCE). IEEE, pp 91–95

Dhamodharan S (2014) Liver disease prediction using Bayesian classification. In: 4th national conference on advanced computing, applications & technologies, pp 1–3

Domingos P, Pazzani M (1996) Beyond independence: conditions for the optimality of the simple Bayesian classifier. In: Saitta L (ed) Proceedings of the Thirteenth International Conference on Machine Learning. Morgan Kaufmann, San Francisco, CA, pp 105–112

Elkan C (1997) Boosting and Naive Bayesian learning. In: Proceedings of the international conference on knowledge discovery and data mining

Ferreira J, Denison D, Hand D (2001) Weighted Naive Bayes modelling for data mining. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.1176&rep=rep1&type=pdf

Flores MJ, Gámez JA, Martínez AM (2014) Domains of competence of the semi-naive Bayesian network classifiers. Inf Sci 260:120–148

Frank E, Hall M, Pfahringer B (2002) Locally weighted naive bayes. In: Proceedings of the nineteenth conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc, pp 249–256

Friedman N, Goldszmidt M (1996) Building classifiers using Bayesian networks. In: Proceedings of the national conference on artificial intelligence, pp 1277–1284

Gammerman A, Thatcher A (1991) Bayesian diagnostic probabilities without assuming independence of symptoms. Methods Inf Med 30(01):15–22

Garg A, Roth D (2001) Understanding probabilistic classifiers. In: European conference on machine learning. Springer, pp 179–191

Geigel A (2013) Neural network trojan. J Comput Secur 21(2):191–232

Geigel A (2014) Unsupervised learning trojan. Ph.D. thesis

Hall M (2006) A decision tree-based attribute weighting filter for Naive Bayes. In: International conference on innovative techniques and applications of artificial intelligence. Springer, pp 59–70

Hand D (1992) Statistical methods in diagnosis. Stat Methods Med Res 1(1):49–67

Hand DJ, Yu K (2001) Idiot's bayes-not so stupid after all? Int Stat Rev 69(3):385–398

He P, Li B, Liu X, Chen J, Ma Y (2015) An empirical study on software defect prediction with a simplified metric set. Inf Softw Technol 59:170–190

Hilden J, Bjerregaard B (1976) Computer-aided diagnosis and the atypical case. Decision making and medical care: can information science help, pp 365–378

Jiang L (2011) Random one-dependence estimators. Pattern Recognit Lett 32(3):532–539

Jiang L, Cai Z, Zhang H, Wang D (2013) Naive bayes text classifiers: a locally weighted learning approach. J Exp Theor Artif Intell 25(2):273–286

Jiang L, Wang D, Cai Z (2012) Discriminatively weighted Naive Bayes and its application in text classification. Int J Artif Intell Tools 21(01):1250007

Jin W, Shi R, Chua TS (2004) A semi-naive Bayesian method incorporating clustering with pair-wise constraints for auto image annotation. In: Proceedings of the 12th annual ACM international conference on multimedia. ACM, pp 336–339

John GH, Langley P (1995) Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence

Kalutarage HK, Nguyen HN, Shaikh SA (2017) Towards a threat assessment framework for apps collusion. Telecommun Syst 66(3):417–430

Kalutarage HK, Shaikh SA, Wickramasinghe IP, Zhou Q, James AE (2015) Detecting stealthy attacks: efficient monitoring of suspicious activities on computer networks. Comput Electr Eng 47:327–344

Kohavi R (1996) Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In: Kdd, vol 96. Citeseer, pp 202–207

Kononenko I (1991) Semi-naive Bayesian classifier. In: European working session on learning. Springer, pp 206–219

Kuncheva LI (2006) On the optimality of Naive Bayes with dependent binary features. Pattern Recognit Lett 27(7):830–837

Langley P, Iba W, Thompson K et al (1992) An analysis of Bayesian classifiers. AAAI 90:223–228

Langley P, Sage S (1994) Induction of selective bayesian classifers. In: Proceedings of the Tenth Conference on Uncertainty in Articial Intelligence Uncertainty. Morgan Kaufmann, Seattle, WA, pp 399–406

Lee CH, Gutierrez F, Dou D (2011) Calculating feature weights in Naive Bayes with Kullback–Leibler measure. In: 2011 IEEE 11th international conference on data mining. IEEE, pp 1146–1151
Liu X, Lu R, Ma J, Chen L, Qin B (2015) Privacy-preserving patient-centric clinical decision support system on Naive Bayesian classification. IEEE J Biomed Health Informatics 20(2):655–668
Lowd D, Meek C (2005) Good word attacks on statistical spam filters. In: CEAS, vol 2005
Lv Z, Li X (2015) Virtual reality assistant technology for learning primary geography. In: International conference on web-based learning. Springer, pp 31–40
Mani S, Pazzani MJ, West J (1997) Knowledge discovery from a breast cancer database. In: Conference on artificial intelligence in medicine in Europe. Springer, pp 130–133
Marucci-Wellman HR, Lehto MR, Corns HL (2015) A practical tool for public health surveillance: semi-automated coding of short injury narratives from large administrative databases using naïve bayes algorithms. Accid Anal Prev 84:165–176
Menzies T, Greenwald J, Frank A (2006) Data mining static code attributes to learn defect predictors. IEEE Trans Softw Eng 33(1):2–13
Nafea IT (2018) Machine learning in educational technology. In: Machine learning-advanced techniques and emerging applications. pp 175–183
Nordyke RA, Kulikowski CA, Kulikowski CW (1971) A comparison of methods for the automated diagnosis of thyroid dysfunction. Comput Biomed Res 4(4):374–389
Ohmann C, Moustakis V, Yang Q, Lang K, Group AAPS et al (1996) Evaluation of automatic knowledge acquisition techniques in the diagnosis of acute abdominal pain. Artif Intell Med 8(1):23–36
Pattekari SA, Parveen A (2012) Prediction system for heart disease using Naïve Bayes. Int J Adv Comput Math Sci 3(3):290–294
Pazzani MJ (1996) Searching for dependencies in Bayesian classifiers. In: Learning from data. Springer, pp 239–248
Provan GM, Singh M (1996) Learning Bayesian networks using feature selection. In: Learning from Data. Springer, New York, NY, pp 291–300
Queiroz R, Berger T, Czarnecki K (2016) Towards predicting feature defects in software product lines. In: Proceedings of the 7th international workshop on feature-oriented software development. ACM, pp 58–62
Rasmussen CE (2003) Gaussian processes in machine learning. In: Summer school on machine learning. Springer, pp 63–71
Ravi C, Manoharan R (2012) Malware detection using windows api sequence and machine learning. Int J Comput Appl 43(17):12–16
Razaque F, Soomro N, Shaikh SA, Soomro S, Samo JA, Kumar N, Dharejo H (2017) Using naïve bayes algorithm to students' bachelor academic performances analysis. In: 2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS). IEEE, pp 1–5
Rennie JD, Shih L, Teevan J, Karger DR (2003) Tackling the poor assumptions of Naive Bayes text classifiers. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp 616–623
Rish I et al (2001) An empirical study of the Naive Bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence, vol 3, pp 41–46
Robles V, Larrañaga P, Peña J, Pérez M, Menasalvas E, Herves V (2003) Bayesian netwoks as consensed voting system in the construction of a multi-classifier for protein secondary structure prediction. Artif Intell Med
Russek E, Kronmal RA, Fisher LD (1983) The effect of assuming independence in applying bayes' theorem to risk estimation and classification in diagnosis. Comput Biomed Res 16(6):537–552
Sayfullina L, Eirola E, Komashinsky D, Palumbo P, Miche Y, Lendasse A, Karhunen J (2015) Improved naive bayes classifier for android malware classification. In: The Proceedings of the 14th IEEE international conference on trust, security and privacy in computing and communications (IEEE TrustCom'15)(Aug. 2015). IEEE
Settouti N, Bechar MEA, Chikh MA (2016) Statistical comparisons of the top 10 algorithms in data mining for classification task. Int J Interact Multimed Artif Intell 4(1):46–51
Shang F, Li Y, Deng X, He D (2018) Android malware detection method based on Naive Bayes and permission correlation algorithm. Cluster Comput 21(1):955–966
Titterington D, Murray G, Murray L, Spiegelhalter D, Skene A, Habbema J, Gelpke G (1981) Comparison of discrimination techniques applied to a complex data set of head injured patients. J R Stat Soc Ser A (Gen) 144(2):145–161
Todd Ba, Stamper R (1994) The relative accuracy of a variety of medical diagnostic programs. Methods Inf Med 33(04):402–416
Vembandasamy K, Sasipriya R, Deepa E (2015) Heart diseases detection using Naive Bayes algorithm. Int J Innov Sci Eng Technol 2(9):441–444
Veni S, Srinivasan A (2017) Defect classification using Naïve Bayes classification. Int J Appl Eng Res 12(22):12693–12700
Vijayarani S, Dhayanand S (2015) Liver disease prediction using svm and Naïve Bayes algorithms. Int J Sci Eng Technol Res (IJSETR) 4(4):816–820
Vinod P, Jaipur R, Laxmi V, Gaur M (2009) Survey on malware detection methods. In: Proceedings of the 3rd Hackers' workshop on computer and internet security (IITKHACK'09), pp 74–79
Wickramasinghe I, Kalutarage H (2020) Naive Bayes: applications, variations and vulnerabilities—a review of literature with code snippets for implementation. https://github.com/HarshaKumaraKalutarage/Naive-Bayes-Applications-and-Vulnerabilities
Wittel GL, Wu SF (2004) On attacking statistical spam filters. In: CEAS
Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Philip SY et al (2008) Top 10 algorithms in data mining. Knowl Inf Syst 14(1):1–37
Xiao H (2017) Adversarial and secure machine learning. Ph.D. thesis, Technische Universit"AT M" Unchen
Yukselturk E, Ozekes S, Türel YK (2014) Predicting dropout student: an application of data mining methods in an online education program. Eur J Open Distance e-Learn 17(1):118–133
Zaidi NA, Cerquides J, Carman MJ, Webb GI (2013) Alleviating Naive Bayes attribute independence assumption by attribute weighting. J Mach Learn Res 14(1):1947–1988
Zhang H (2004) The optimality of Naive Bayes. AA 1(2):3
Zhang H, Sheng S (2004) Learning weighted Naive Bayes with accurate ranking. In: Fourth IEEE international conference on data mining (ICDM'04). IEEE, pp 567–570
Zheng F, Webb GI, Suraweera P, Zhu L (2012) Subsumption resolution: an efficient and effective technique for semi-naive Bayesian learning. Mach Learn 87(1):93–125
Zheng Z, Webb GI, Ting KM (1999) Lazy bayesian rules: A lazy semi-naive bayesian learning technique competitive to boosting decision trees. In: Proceedings of 16th international conference on machine learning. Citeseer