Balancing the trade-off between accuracy and interpretability in software defect prediction, ESE 2019

Rahul Yedida
Department of Computer Science, NC State University
ryedida@ncsu.edu

ABSTRACT

Can we build accurate ML systems that are also interpretable? This poster summarizes the method described in [6]: a novel classification model is proposed that is both accurate and interpretable. It extends the standard Naive Bayes model to create an ensemble, and then creates a linear approximation. The method, superposed Naive Bayes (SNB) produces a balanced output that is both interpretable and accurate, and its results are discussed in the context of software datasets.

1 INTRODUCTION

Recently, interpretability is becoming more important in machine learning [7] in several domains including healthcare, where computational models cannot be blindly trusted without clear reasoning behind decisions. However, it has been observed that there is in general a trade-off between interpretability of models and accuracy, and the proper balance may be different in each domain.

Software engineering is one of the domains where machine learning could potentially be useful. Among other use cases, machine learning techniques have been extensively applied in software defect prediction [5]. This paper outlines a novel classification model that first builds an ensemble of Naive Bayes classifiers and then creates a linear approximation, with the intention that the first step improves accuracy, and the second step improves interpretability.

We aim to answer the following research questions:

(1) **How accurate are the current predictors?** To compare the accuracy of SNB with other classifiers, we use multiple k-fold cross-validation and a double Scott-Knott test [2].
(2) **How do we assess the interpretability of classifiers?** We use a qualitative assessment based on [4], which is comprised of model transparency, component transparency, and algorithmic transparency.
(3) **How can we balance the trade-off between accuracy and interpretability?** To analyze this, we combine the results measuring accuracy and interpretability. Our findings confirm the general assumption that there is a trade-off between the two; nevertheless, SNB gives balanced outputs that satisfy both criteria.

2 METHOD

We focus on binary classification problems, since multi-class classification problems can be converted to binary classification problems by schemes such as one-vs-rest.

[0]Toshiki Mori and Naoshi Uchihira. Balancing the trade-off between accuracy and interpretability in software defect prediction. Empirical Software Engineering, 24(2):779–825, 2019.

2.1 Naive Bayes

Naive Bayes is a special case of Gaussian Discriminant Analysis with the stronger assumption that the predictor variables are conditionally independent given the target variable. We then use Bayes' rule to compute posterior likelihoods:

$$p(y=1|x_1,\dots,x_d) \propto p(y=1)\prod_{i=1}^{d} p(x_i|y=1) \tag{1}$$

From this, we compute the log-odds ratio:

$$\frac{\log p(y=1|x_1,\dots,x_d)}{\log p(y=0|x_1,\dots,x_d)} = \log\frac{p(y=1)}{p(y=0)} + \sum_{i=1}^{d}\log\frac{p(x_i|y=1)}{p(x_i|y=0)} \tag{2}$$

which can be expressed as

$$W(X) = W_0 + \sum W_i(x_i) \tag{3}$$

Each term here is called a weight of evidence. A positive $W_i(x_i)$ indicates that the state of x_i is evidence in favor of the hypothesis that $y=1$. The total sum of the weights of evidence gives a raw score $W(X)$, a higher value indicating a higher posterior likelihood that $y=1$.

There are some issues with the standard Naive Bayes. First is the issue of cases where a dependent variable value and an independent variable value never occur, resulting in an indefinite weight of evidence. This can be fixed by adding a small sample correction to all likelihoods. This is called Laplace smoothing. The second issue is with numeric values. This can be solved by assuming a prior distribution on that independent variable and using it in the posterior likelihood expression. The third issue is with post-processing of the raw score. Class membership probability estimates can be obtained by sigmoid calibration of the raw scores:

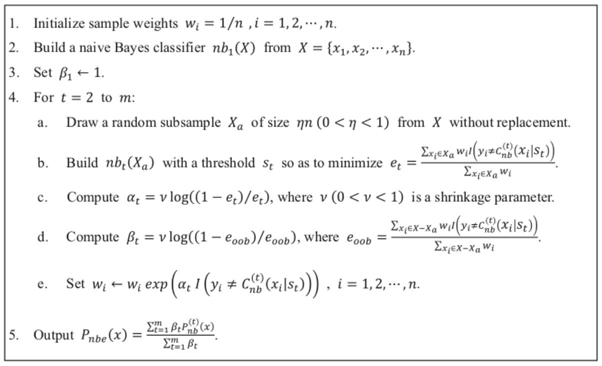
$$p_{nb}(X) = \sigma(c_0 + c_1 W(X)) = \frac{1}{1+\exp(-(c_0 + c_1 W(X)))} \tag{4}$$

Given a discrimination threshold t, we predict $C_{nb}(X|t) = 1$ if $p_{nb}(X) \geq t$.

2.2 Naive Bayes ensemble

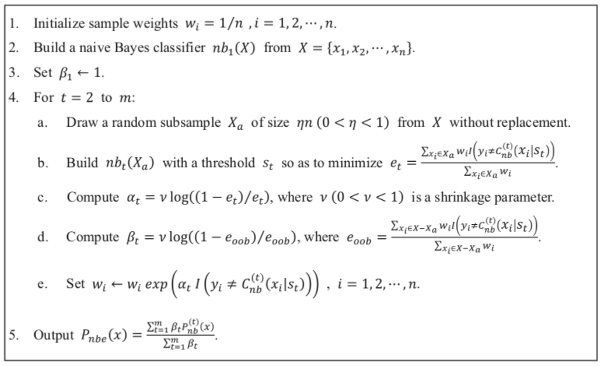
Creating ensembles can be done in several ways involving random selection of feature subsets to train different models. Boosting is one such technique that incrementally builds an ensemble by reweighting each training sample; correctly classified samples are given a lower weight, and incorrectly classified samples are given a higher weight. AdaBoost [1] uses a multiplicative weight update rule. An extension of this called stochastic gradient boosting uses a gradient descent approach at each iteration.

Figure 1 shows the algorithm to create a Naive Bayes ensemble.



```
1. Initialize sample weights $w_i = 1/n$, $i = 1, 2, \cdots, n$.
2. Build a naive Bayes classifier $nb_1(X)$ from $X = \{x_1, x_2, \cdots, x_n\}$.
3. Set $\beta_1 \leftarrow 1$.
4. For $t = 2$ to $m$:
    a. Draw a random subsample $X_a$ of size $\eta n$ $(0 < \eta < 1)$ from $X$ without replacement.
    b. Build $nb_t(X_a)$ with a threshold $s_t$ so as to minimize $e_t = \frac{\sum_{x_i \in X_a} w_i I\left(y_i \neq C_{nb}^{(t)}(x_i|s_t)\right)}{\sum_{x_i \in X_a} w_i}$.
    c. Compute $\alpha_t = \nu \log((1 - e_t)/e_t)$, where $\nu$ $(0 < \nu < 1)$ is a shrinkage parameter.
    d. Compute $\beta_t = \nu \log((1 - e_{oob})/e_{oob})$, where $e_{oob} = \frac{\sum_{x_i \in X - X_a} w_i I\left(y_i \neq C_{nb}^{(t)}(x_i|s_t)\right)}{\sum_{x_i \in X - X_a} w_i}$.
    e. Set $w_i \leftarrow w_i \, exp\left(\alpha_t \, I\left(y_i \neq C_{nb}^{(t)}(x_i|s_t)\right)\right)$, $i = 1, 2, \cdots, n$.
5. Output $P_{nbe}(x) = \frac{\sum_{t=1}^{m} \beta_t P_{nb}^{(t)}(x)}{\sum_{t=1}^{m} \beta_t}$.
```

Figure 1: Naive Bayes ensemble algorithm



```
1. Initialize sample weights $w_i = 1/n$, $i = 1, 2, \cdots, n$.
2. Build a naive Bayes classifier $nb_1(X)$ from $X = \{x_1, x_2, \cdots, x_n\}$.
3. Set $\beta_1 \leftarrow 1$.
4. For $t = 2$ to $m$:
    a. Draw a random subsample $X_a$ of size $\eta n$ $(0 < \eta < 1)$ from $X$ without replacement.
    b. Build $nb_t(X_a)$ with a threshold $s_t$ so as to minimize $e_t = \frac{\sum_{x_i \in X_a} w_i I\left(y_i \neq C_{nb}^{(t)}(x_i|s_t)\right)}{\sum_{x_i \in X_a} w_i}$.
    c. Compute $\alpha_t = \nu \log((1 - e_t)/e_t)$, where $\nu$ $(0 < \nu < 1)$ is a shrinkage parameter.
    d. Compute $\beta_t = \nu \log((1 - e_{oob})/e_{oob})$, where $e_{oob} = \frac{\sum_{x_i \in X - X_a} w_i I\left(y_i \neq C_{nb}^{(t)}(x_i|s_t)\right)}{\sum_{x_i \in X - X_a} w_i}$.
    e. Set $w_i \leftarrow w_i \, exp\left(\alpha_t \, I\left(y_i \neq C_{nb}^{(t)}(x_i|s_t)\right)\right)$, $i = 1, 2, \cdots, n$.
5. Output $P_{nbe}(x) = \frac{\sum_{t=1}^{m} \beta_t P_{nb}^{(t)}(x)}{\sum_{t=1}^{m} \beta_t}$.
```

Figure 2: Superposing Naive Bayes ensemble

2.3 Superposed Naive Bayes

We then create a single Naive Bayes model by superposing the different models, weighted by the weights of evidence. Figure 2 shows an example.

From (3) and (4), the output of the Naive Bayes ensemble is

$$p_{nbe}(X) = \frac{\sum\limits_{t=1}^{m} \beta_t p_{nb}^{(t)}(X)}{\sum\limits_{t=1}^{m} \beta_t} = \frac{\sum\limits_{t=1}^{m} \beta_t \sigma\left(c_0^{(t)} + c_1^{(t)} \sum\limits_{i=0}^{d} W_i^{(t)}(x_i)\right)}{\sum\limits_{t=1}^{m} \beta_t} \tag{5}$$

We now approximate the sigmoid function by its first order Taylor series approximation about c_0:

$$\begin{aligned}\sigma\left(c_0^{(t)} + c_1^{(t)} W^{(t)}(X)\right) &= \sum_{n=0}^{\infty} \frac{\sigma^{(n)}(c_0^{(t)})}{n!}\left(c_1^{(t)} W^{(t)}(X)\right) \\ &\approx \sigma(c_0^{(t)}) + \left(c_1^{(t)} \sigma(c_0^{(t)})\left(1 - \sigma(c_0^{(t)})\right)\right) W^{(t)}(X)\end{aligned} \tag{6}$$

which is of the form $a_t + b_t W^{(t)}(X)$.

3 EXPERIMENTAL SETTINGS

We use 13 datasets from two sources: a cleaned version of the NASA MDP datasets provided by [8], and the JIT datasets provided by [3]. From the first source, 11 datasets that were initially from various NASA systems, are collected. We use 2 datasets from the second source: Bugzilla and Columba.

We evaluate predictive accuracy using 5-fold cross-validation. The entire 5-fold cross-validation is repeated 5 times with different random subsets. We use the area under receiver operating characteristics (AUC-ROC) as the performance measure, and use the Scott-Knott test to partition the predictors into statistically distinct ranks at a significance level of 5%. We rank predictors using a double Scott-Knott test. At the first level, the Scott-Knott test is performed on each dataset. At the second level, we run the test on the results of the first level to find the statistically distinct ranks of the predictors.

To evaluate interpretability, we use Lipton's criteria: model transparency (can a user understand the model?), component transparency (can a user understand each component?), and algorithmic transparency (is the algorithm deterministic?).

4 RESULTS

REFERENCES

[1] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[2] Enio G Jelihovschi, José Cláudio Faria, and Ivan Bezerra Allaman. Scottknott: a package for performing the scott-knott clustering algorithm in r. *TEMA (São Carlos)*, 15(1):3–17, 2014.

[3] Yasutaka Kamei, Emad Shihab, Bram Adams, Ahmed E Hassan, Audris Mockus, Anand Sinha, and Naoyasu Ubayashi. A large-scale empirical study of just-in-time quality assurance. *IEEE Transactions on Software Engineering*, 39(6):757–773, 2012.

[4] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[5] Ruchika Malhotra. A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, 27:504–518, 2015.

[6] Toshiki Mori and Naoshi Uchihira. Balancing the trade-off between accuracy and interpretability in software defect prediction. *Empirical Software Engineering*, 24(2):779–825, 2019.

[7] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. 2016.

[8] Martin Shepperd, Qinbao Song, Zhongbin Sun, and Carolyn Mair. Data quality: Some comments on the nasa software defect datasets. *IEEE Transactions on Software Engineering*, 39(9):1208–1215, 2013.