August 2023

Binary Classifier Selection Based on Precision and Recall Metrics

Richa Gupta

Babu Prasad Elumalai

Vineet Jain

Vidhi Gupta

Subham Mishra

See next page for additional authors

Inventor(s)
Richa Gupta, Babu Prasad Elumalai, Vineet Jain, Vidhi Gupta, Subham Mishra, and Aman Singh

Binary Classifier Selection Based on Precision and Recall Metrics

ABSTRACT

Binary classifiers are used for bug deduplication in software development. However, the precision and recall metrics of individual classifiers may be insufficient for particular use cases. Selecting an appropriate binary classifier can improve bug deduplication, e.g., high precision can ensure that duplicate bugs are identified (and thus not analyzed), while providing sufficient recall (to eliminate redundant analysis of bugs that are duplicates). This disclosure describes automated techniques to identify the most appropriate classification model from a set of models. Two different functions - F-beta score and weighted sum score - are evaluated for different performance thresholds. The model and threshold combination with the highest score is selected and used for bug deduplication.

KEYWORDS

- Binary classifier
- Binary classification
- Classification model
- Machine learning
- Bug deduplication
- Model precision
- Model recall

BACKGROUND

Bug deduplication is a well-known problem in software development. Binary classification techniques are widely used to classify a bug as a duplicate or not a duplicate. However, finding the best machine learning (ML) model among multiple candidates that perform binary classification might not always be possible. Using model precision and recall metrics may not be enough in the context of classifiers that perform bug deduplication since higher precision implies a necessary reduction in recall, and vice-versa. Selecting an optimal threshold that balances precision and recall may not always be possible. A lower threshold necessarily implies that more bugs end up being predicted as duplicates. Conversely, selecting a higher threshold leads to poor recall - fewer bugs being predicted as duplicates.

Use of the *F1* is a standard technique to choose a particular model where balancing precision and recall is important. The *F1* score is often referred to as the harmonic mean of the precision and recall of the model. The *F1* score ranges from a minimum value of 0 to a maximum value of 1. A higher *F1* score corresponds to a better quality model. The *F1* score can be calculated using the formula:

$$F1\ Score\ = \frac{2 * precision * recall}{(precision + recall)}$$

DESCRIPTION

This disclosure describes automated techniques to identify the most appropriate classification model from a set of models with an optimal threshold. It is assumed that each classification model is pre-trained on historical data on bugs. Further, it is assumed that performance metrics from evaluation of each classification model on the historical bug data are available.

Multiple classification models (m_1, m_2, m_3, ..., m_n) are trained to predict whether a pair of bugs are duplicates from a historical dataset of bugs. The most appropriate model among all of the trained models is selected. The choice of the model is based on a threshold that balances the precision and recall metrics for evaluation datasets of bugs (that are not part of the training dataset used to train the classification models).

In a bug deduplication context, balancing precision and recall is important. It is vital that bugs that are marked as duplicates are actually duplicates (“high precision”). However, high precision implies that some duplicate bugs are erroneously classified as non-duplicates (“lower recall”). In this disclosure, two techniques for model selection are described. Both techniques assign a higher weightage to precision than to recall.

F-beta score

Minimum values of precision (p_{min}) and recall (r_{min}) are chosen such that if no model meets the minimum criteria (*precision* >= p_{min} and *recall* >= r_{min}), no model is selected. The models are evaluated as follows:

1. Calculate the precision and recall of each model for each threshold from 0.0 to 1, with a step size of 0.01. Thus, each model is evaluated over the set of thresholds [0.01, 0.02, 0.03, ... 0.98, 0.99, 1].
2. Calculate the F-beta score for each model and threshold where the precision and recall are not equal to 0 using the formula:

$$F_\beta = \frac{(1+\beta*\beta)*precision*recall}{(\beta*\beta*precision+recall)},$$

where β can be any number from 0 to 1.

The model and threshold combination that has the highest F-beta score is selected and used for bug deduplication.

While the above description recites a step size of 0.01 for each iteration, any other suitable step size may be used in various implementations. The value of β can be selected as appropriate, e.g., 0.5.

Weighted sum

Minimum values of precision (p_{min}) and recall (r_{min}) are chosen such that if no model meets the minimum criteria (*precision* >= p_{min} and *recall* >= r_{min}), no model is selected. The models are evaluated as follows:

1. Calculate the precision and recall of each model for each threshold from 0.0 to 1, with a step size of 0.01. Thus, each model is evaluated over the set of thresholds [0.01, 0.02, 0.03, ... 0.98, 0.99, 1]. The step size can be adjusted up or down as appropriate.
2. Calculate the weighted sum score for each model and threshold where the precision and recall are not equal to 0 using the formula:

   weighted_sum = w * precision + recall, where *w* is the weightage for precision.

   For example, *w* can be selected as 5.
3. The model and threshold combination that has the highest weighted_sum score is selected and used for bug deduplication.

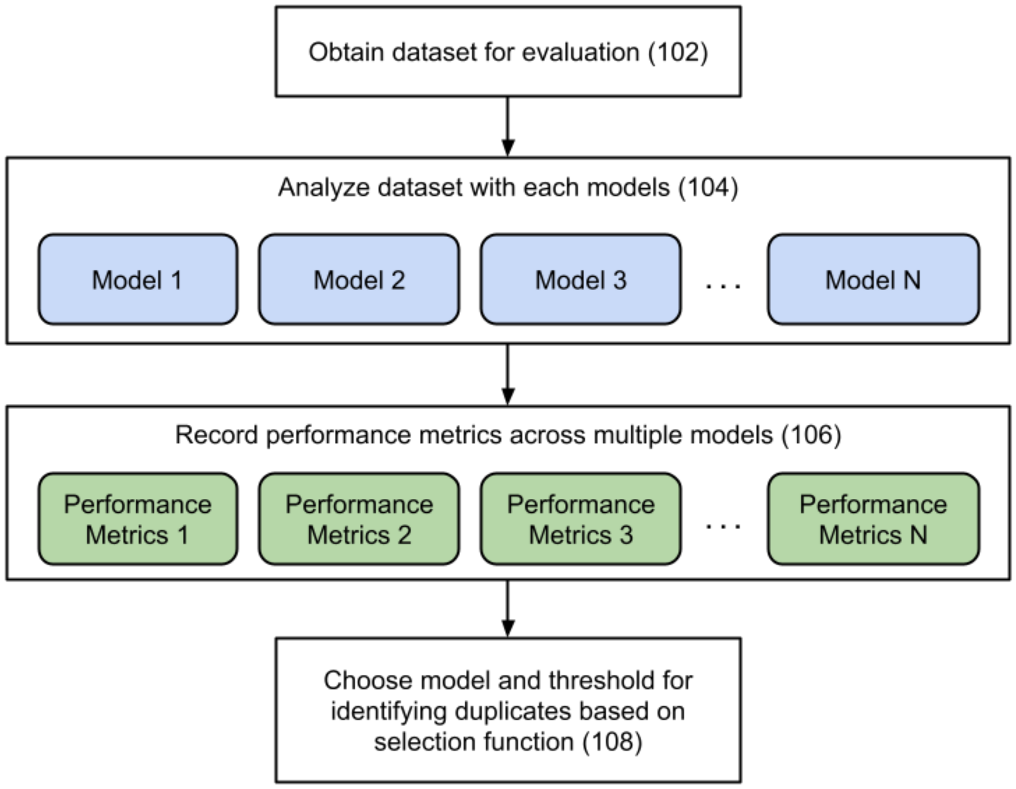
Obtain dataset for evaluation (102)
Analyze dataset with each models (104)
Model 1
Model 2
Model 3
. . .
Model N
Record performance metrics across multiple models (106)
Performance Metrics 1
Performance Metrics 2
Performance Metrics 3
. . .
Performance Metrics N
Choose model and threshold for identifying duplicates based on selection function (108)

Fig. 1: Automated model and threshold selection for machine learning classification models

Fig. 1 illustrates an example technique to automatically select an appropriate model and threshold to select a classification model for bug deduplication (or other applications). A dataset is obtained for evaluation (102). The dataset is analyzed by each of the candidate models (*model1, model 2, model3…modelN*) (104). The performance metrics of each model for the given dataset are recorded (106). One or both of the evaluation criteria - the F-Beta score and the weighted sum score - are calculated. Based on the chosen criteria, a particular combination of model and threshold is chosen (108) to perform bug deduplication.

While the foregoing description refers to selection of a suitable binary classifier in the context of bug deduplication, the described techniques of model selection can be used in any context where ML models are used for classification and where model precision and recall are both important.

CONCLUSION

This disclosure describes automated techniques to identify the most appropriate classification model from a set of models. Two different functions - F-beta score and weighted sum score - are evaluated for different performance thresholds. The model and threshold combination with the highest score is selected and used for bug deduplication.

REFERENCES

1. "Binary classification - Wikipedia" available online at https://en.wikipedia.org/wiki/Binary_classification accessed Jul 28, 2023.
2. "Classification: True vs. False and Positive vs. Negative | Machine Learning | Google for Developers" available online at https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative accessed Jul 28, 2023.
3. "Classification: Thresholding | Machine Learning | Google for Developers" available online at https://developers.google.com/machine-learning/crash-course/classification/thresholding accessed Jul 28, 2023.
4. "Classification: Precision and Recall | Machine Learning | Google for Developers" available online at https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall accessed Jul 28, 2023.