World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(RESEARCH ARTICLE)

Classification of positive and negative test cases using ensemble machine learning methods

Amina Becic *

Department of Information Technology, Faculty of Engineering, Natural and Medical Sciences, International Burch University, Bosnia and Herzegovina.

World Journal of Advanced Engineering Technology and Sciences, 2024, 12(01), 364–371

Publication history: Received on 28 April 2024; revised on 10 June 2024; accepted on 12 June 2024

Article DOI: https://doi.org/10.30574/wjaets.2024.12.1.0232

Abstract

In today's IT industry, fast software delivery is becoming a standard and the race to the market adds up to the pressure. To ensure the quality of the developed features, quality assurance engineers apply both manual and automated techniques, and adapt their approach to optimize the process of verification. Machine learning can help significantly with the automation of some traditionally manual processes. In this research, we are showing how the classification of test cases into positive and negative can be done using ensemble machine learning methods, and whether those have advantages and better results over the basic machine learning methods. The best result was obtained by the Gradient Boosting Classifier with accuracy of 94%. However, since there is just a 1% difference in accuracy when compared to regular Decision Tree, it would not be necessary to use ensemble methods for this specific problem.

Keywords: Quality Assurance; Positive test cases; Negative test cases; Machine learning; Classification; Ensemble learning

1. Introduction

Quality Assurance (QA) in software development became one of the essential parts of the software lifecycle, especially for big, fast-changing software products. Quality Assurance process makes sure that expectations set prior to beginning of development are successfully met, while also keeping software with as few bugs and defects as possible.

Machine learning can be applied in all areas of the software quality assurance process. Different research papers address various ways of applying machine learning to quality assurance activities. Predicting the severity of defects found [1], automating selection of regression tests [2], predicting quality level of a software product [3], self-healing the execution of Selenium tests, automating API test generation and visual validation of automation testing [4] etc.

Paper from 2019., published on the topic of impact that artificial intelligence has on software testing [5] has a good outline of software testing areas and common artificial intelligence algorithms that can be used to resolve problems and make appropriate automated decisions. It divides areas in 18 distinct potential points in which machine learning and artificial intelligence can be applied, starting from specification management, predicting the test coverage, automatically writing automated tests for user interface (UI), along with test cases optimization, classification of test cases, test case management, detection of infeasible tests, generating test data, documentation, planning, scheduling of testing activities [5] etc.

Explored literature tends to dive deeper into the huge potential of the quality assurance area to optimize current processes by using machine learning. Based on conclusions from each of the papers analyzed, this work has globally just

* Corresponding author: Amina Becic

started and there is a lot of space for discussing new applications of machine learning, or optimization of used algorithms to find the best possible solutions for a certain use case. Test case management is an area of interest of this research.

Test cases are a part of every testing process. They outline possible use cases users can make in the application. Every test case includes information about the exact steps that need to be made, any needed data or prior setup, priority of the test case and expected result of the executed case, which is then compared to actual result QA engineer obtained after testing is completed.

Test cases can be positive and negative. Positive test cases refer to a set of actions for which we expect the system to correctly accept valid inputs provided by the user. Negative test cases, on the other hand, show that the system can handle invalid inputs or potential attempts of malicious actions. Example of a positive test case would be providing the correct username and password to a login form and expecting it to pass. Example of a negative test case is to provide either incorrect username or password, and expecting the system to throw an error, which is still expected behavior, considering the input provided.

Classifying test cases into positive and negative can help save time in any quality assurance process by assisting with prioritization of tests, optimization of regression testing, generating new test cases and maintaining both manual and automated test cases. Some basic machine learning methods, such as Decision Tree, can reach up to 93% accuracy when applied to this classification.

Ensemble methods are more complex machine learning methods that combine decisions from several models to achieve better accuracy of results, compared to using only one model. Ensemble methods aim to decrease a chance for errors or biases that might exist in individual models by balancing the collective intelligence of combined methods.

As a part of systematic mapping study research [6] which explored 48 papers with the topic of application of machine learning in quality assurance, it was observed that the most of approaches use supervised learning algorithms, and the artificial Neural Networks and Decision Trees are the most used algorithms.

Similar conclusions can be drawn from the chronological survey research [7] which discusses that Support Vector Machine and some generic machine learning techniques are also popular for solving quality assurance challenges, along with Neural Networks and Decision Trees (especially in more recent papers).

Through the literature, there were examples of test case management, which involved writing new test cases using machine learning, or filtering them based on different criteria to enhance test case selection for regression testing and automation. However, a use case of classification of test cases into positive and negative, with the same goal, did not appear during the investigation. Furthermore, even though applied in many different fields, it was observed that ensemble learning methods are not often used for quality assurance problems, so the goal is to explore how do they result when applied to classification of test cases. Motivation behind this paper is to explore whether ensemble learning methods could achieve even better results than decision tree.

2. Material and Methods

2.1. Dataset

For this research, a publicly available dataset [Kaggle, https://www.kaggle.com/datasets/razailin/positive-and-negative-test-cases, Last accessed on 06/05/2024] was used. Dataset contains 3000 records, which could match the number of test cases for small to medium sized real-world projects.

Initial dataset contains 2 columns, 1 containing a test case description, and the other with positive/negative classification for each of them. There are 2100 positive and 900 negative test cases in the dataset.

The Dataset did not have missing or null values, and all valid values are present. As a part of preprocessing, removal of punctuation marks and conversion of all uppercase to lowercase letters was conducted.

Table 1 Examples of positive and negative test cases from the dataset

Classification	Test case
Neg	Enter an invalid username and valid password
Neg	Enter a valid username and invalid password
Neg	Enter an invalid username and invalid password
Pos	Enter a Valid Username and click next
Pos	Enter Valid Password and click Next

This dataset is composed of textual data, so it was needed to convert text to numerical data using CountVectorizer, which simplifies extracting features from text data, and converting it into vector format used by machine learning models. Results of vectorization of the first Positive test case from Table 1 are shown in Table 2 below.

Table 2 Count Vectorizer output for a single test case

Vectorized output	Matching words
(0, 0) 1	#Enter
(0, 2) 1	# a
(0, 3) 1	# Valid
(0, 4) 1	#Username
(0, 5) 1	# and
(0, 6) 1	# click
(0, 7) 1	# Next

The output is shown in coordinate list format, where the first element represents the row (entire sentence is in a single row, which is why it is always 0), second element represents the column (1 column is equal to the index of the word in the sentence), and third element represents the frequency of each word in the sentence (each word appears once, which is why this value is always 1).

2.2. Ensemble Machine Learning Methods

There is a set of six popular Ensemble methods used in this research:

- **Gradient Boosting Classifier:** This method creates a sequence of decision trees. Each following tree corrects mistakes made by the previous one.
- **Neural Networks:** This method can recognize complex relationships in data and handle large datasets. It is based on interconnected nodes which process the data to provide an output.
- **Bagging Classifier:** This method combines output predictions from several base models through averaging or voting. Each model uses a different subset of training data, which helps obtain generalization in the final prediction.
- **AdaBoost Classifier:** Adaptive Boosting or AdaBoost Classifier sequentially trains chain of base models (such as decision tree). The weights of misclassified instances are adjusted in each iteration.
- **Light GBM:** This gradient boosting framework performs better than traditional gradient boosting classifiers when it comes to larger datasets since it is faster and has a low memory usage.
- **Cat Boost Classifier:** Categorical Boosting or Cat Boost is a gradient boosting library. It is designed for handling categorical features and decreases a need for preprocessing since it automatically handles missing values.

3. Proposed Methodology

The split between training and test data is 70:30. Code for the classification was written in Python.

Entire process can be divided into several distinct phases which can be grouped into:

- Data manipulation and preparation
  - Data Preprocessing
  - Train - Test Split
  - Data Vectorization
- Model executions and iterations
  - Model Selection
  - Baseline Model Performance
  - Model Iterations
  - Best Model Selection

Note that after Baseline Model Performance there could also be a Hyperparameter tuning phase, but it did not give better results, so it was omitted from this research.

The process and phases described in previous sections are visualized in flowchart shown in Figure 1.

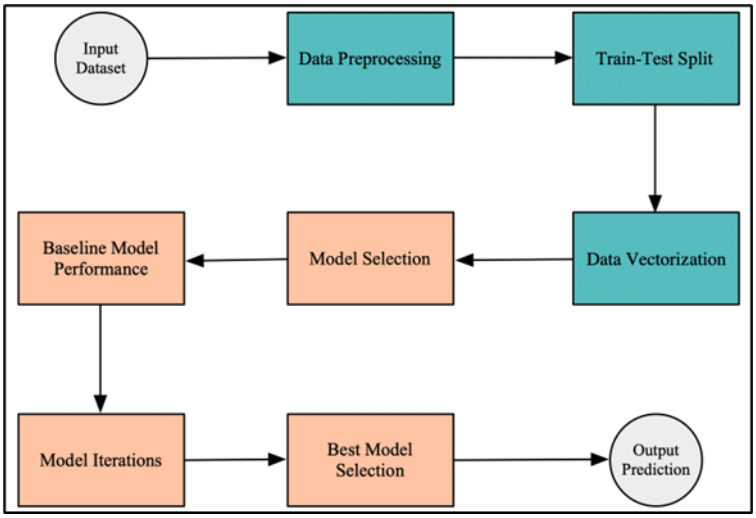

Figure 1 Approach phases for classification data analysis and model selection

4. Results

For each of the previously mentioned methods, results from the tables below were obtained for the following metrics:

- **Accuracy:** Measures how often a machine learning model correctly predicts the outcome. It can be calculated by dividing the number of correct predictions with the total number of predictions.
- **Precision**: This parameter indicates the quality of performance of a machine learning model. When it comes to mean of calculation, it refers to number of real positives divided by the number of positive predictions (real positives + false positives)
- **F1 measure**: Indicator of a performance of a model and how well it keeps high precision and recall. It ranges between 0 and 1, with 0 being the lowest possible score, and 1 being the ideal result where the model accurately predicts each label. Formula for calculating f1 contains results obtained from precision and recall calculations.
- **Confusion Matrix:** Represents a predictive performance of a model on a dataset. For binary dataset (such as the one researched in this paper) it contains 4 essential components:
  - True Positives (TP): Number of samples *correctly* predicted as "positive."
  - False Positives (FP): Number of samples *wrongly* predicted as "positive."
  - True Negatives (TN): Number of samples *correctly* predicted as "negative."
  - False Negatives (FN): Number of samples *wrongly* predicted as "negative."

From the results in Table 3 shown below, the best accuracy was obtained by Gradient Boosting Classifier, even though Bagging and Cat Boost Classifier fall behind for less than 1%.

Similar situation can be observed for precision where the Gradient Boosting Classifier has a precision of 0.9347, followed by Cat Boost Classifier scoring 0.923. In terms of precision Bagging Classifier and Light GBM had almost the same precision obtained (0.001 difference).

As for F1 results, all methods except for Neural Networks scored more than 0.939, with Gradient Boost Classifier having the highest F1 score of 0.9586, followed again by Bagging and Cat Boost Classifier which had the same F1 score of 0.952.

For all three parameters, Neural Networks had the lowest results. It was the only method with accuracy smaller than 90%, and precision and F1 smaller than 0.9.

Table 3 Average Accuracy, Precision and F1 score for 6 chosen Ensemble methods

Ensemble Method Name	**Gradient Boosting Classifier**	**Neural Networks**	**Bagging Classifier**	**AdaBoost Classifier**	**Light GBM**	**Cat Boost Classifier**
Average Accuracy	94%	86%	93.44%	91.4%	92%	93.33%
Precision	0.9347	0.870	0.918	0.909	0.917	0.923
F1	0.9586	0.895	0.952	0.939	0.943	0.952

Comparison of accuracies for all used models is also represented in graph from Figure 2.

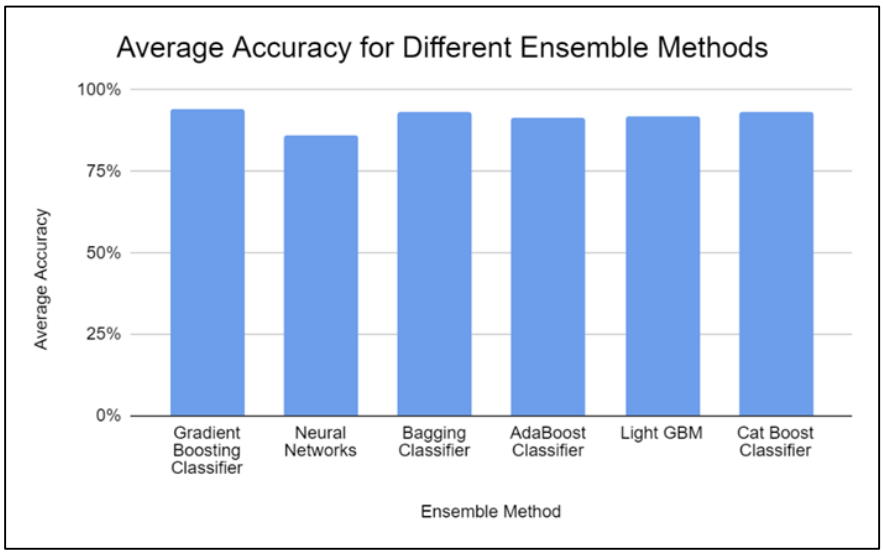

Figure 2 Average Accuracy for 6 distinct Ensemble methods

Confusion matrix results for all used Ensemble methods are listed in 6 tables below.

When comparing actual and predicted values for a test dataset of 900 records, it can be observed that Gradient Boosting Classifier correctly predicted 848 test cases, where 246 were positive test cases and 602 were negative test cases. This method did not have correct results for a total of 52 records, where it classified 42 negative test cases as positive, and 10 positive test cases as negative.

Table 4 Confusion Matrix for Gradient Boosting Classifier

Gradient Boosting Classifier		**Actual Values**	
		Positive	**Negative**
Predicted Values	Positive	246	42
	Negative	10	602

Looking into Neural Networks results, it correctly predicted 768 test cases, where 204 were positive test cases and 564 were negative test cases.

This method did not have correct results for a total of 132 records, where it classified 84 negative test cases as positive, and 48 positive test cases as negative.

Table 5 Confusion Matrix for Neural Networks

Neural Networks		**Actual Values**	
		Positive	**Negative**
Predicted Values	Positive	204	84
	Negative	48	564

Analyzing outputs made by Bagging Classifier, it correctly predicted 839 test cases, where 234 were positive test cases and 605 were negative test cases.

This method did not have correct results for a total of 61 records, where it classified 54 negative test cases as positive, and 7 positive test cases as negative.

Table 6 Confusion Matrix for Bagging Classifier

Bagging Classifier		**Actual Values**	
		Positive	**Negative**
Predicted Values	Positive	234	54
	Negative	7	605

AdaBoost Classifier correctly predicted 823 test cases, where 229 were positive test cases and 594 were negative test cases.

This method did not have correct results for a total of 77 records, where it classified 59 negative test cases as positive, and 18 positive test cases as negative.

Table 7 Confusion Matrix for AdaBoost Classifier

AdaBoost Classifier		**Actual Values**	
		Positive	**Negative**
Predicted Values	Positive	229	59
	Negative	18	594

Results obtained by LightGBM show that this method correctly predicted 828 test cases, where 234 were positive test cases and 594 were negative test cases. This method did not have correct results for a total of 72 records, where it classified 54 negative test cases as positive, and 18 positive test cases as negative.

Table 8 Confusion Matrix for Light GBM Classifier

Light GBM		**Actual Values**	
		Positive	**Negative**
Predicted Values	Positive	234	54
	Negative	18	594

Finally, Cat Boost Classifier correctly predicted 840 test cases, where 238 were positive test cases and 602 were negative test cases.

This method did not have correct results for a total of 60 records, where it classified 50 negative test cases as positive, and 10 positive test cases as negative.

Table 9 Confusion Matrix for Cat Boost Classifier

Cat Boost Classifier		**Actual Values**	
		Positive	**Negative**
Predicted Values	Positive	238	50
	Negative	10	602

5. Discussion

When looking into the results presented in the previous section, it is important to note that in general, all methods achieve accuracy higher than 85%. When making mutual comparison of used Ensemble methods, the highest accuracy, precision and F1 score were achieved using Gradient Boost Classifier.

The lowest score for all three parameters was recorded for Neural Networks. This is one of the scenarios of supervised learning where Gradient Boost Classifier (and other Ensemble methods) outperform Neural Networks, even though it is one of the most used methods in related literature. One of the possible reasons is the limited amount of data (3000 records in our dataset) while Neutral Networks perform better with bigger datasets.

Analysis of the confusion matrix confirms that the greatest number of correctly predicted cases was made by Gradient Boosting Classifier (848), followed by Cat Boost (840) and Bagging Classifier (839). Total number of correctly classified test cases vs. total number of incorrectly classified test cases can be found in Table 10 below. More detailed distribution of positive and negative classifications within correct and incorrect prediction buckets is shown in tables 4-9 from the Results section.

Neural Networks have more than 2.5 times more incorrect predictions compared to Gradient Boosting Classifier. This confirms the conclusion made before, that the best results for this classification and the dataset size, when choosing amongst Ensemble methods, can be obtained by the Gradient Boosting Classifier.

Table 10 Total number of correct and incorrect predictions for 6 distinct Ensemble methods

	Gradient Boosting Classifier	**Neural Networks**	**Bagging Classifier**	**AdaBoost Classifier**	**LightGBM Classifier**	**Cat Boost Classifier**
Correct predictions	848	768	839	823	828	840
Incorrect predictions	52	132	61	77	72	60

Decision Trees are one of the most used methods in the latest published papers on the topic of application of machine learning to quality assurance, and in this particular use case, it was justified to rely more on this method. Even though Gradient Boost Classifier consists of more decision trees within, it did not make much more accurate predictions, when compared to Decision Tree alone (94% vs 93%, respectively).

Given that Gradient Boost Classifier takes more resources than Decision Tree, it can be concluded that for this dataset and use case, there is no need to introduce Ensemble methods, when Decision Tree does the job with the same efficiency.

When comparing results to machine learning methods overviews and comparisons found in systematic mapping study [6] and chronological survey [7], Neural Networks are positioned lower in the hierarchy of methods that could be used

for the classification of test cases on positive and negative, while decision tree was confirm as an efficient approach to different software product quality assurance challenges.

6. Conclusion

Having machine learning as a new approach to solving problems in the quality assurance area, engineers get more space for creativity and freedom to shape their processes in new, revolutionized ways. Repetitive and error prone work can be replaced with algorithms that can often do the job and recognize patterns even better than humans.

Introducing machine learning provides an opportunity to decrease the time needed from the moment a feature is received for testing, until the moment of QA sign-off to production, which also supports continuous deployment initiatives that are becoming common practices. Testing is a crucial part of every software product delivery, and every chance of automating the processes need to be taken to improve performance and build trust in the product that is being shipped to production.

Having the ability to automatically classify test cases into positive and negative can speed up manual work needed for this classification by quality assurance engineers and support automatic selection and optimization of regression tests. Classifying test cases into positive and negative with machine learning algorithms can be used in the real-world projects due to high accuracy that can be achieved.

Machine learning algorithms can also process a larger amount of data, depending on the size of project and number of test cases, which saves a lot of time and increases efficiency of QA teams. Using Ensemble methods does not bring a great advantage to the users, compared to Decision Tree which already shows accurate predictions.

This leads to the conclusion that QA engineers can optimize their processes, achieve better efficiency, consistency and scalability with smaller cost, less time and computational power, which can play a significant role in deciding to use machine learning for automating their processes.

Compliance with ethical standards

Disclosure of conflict of interest

Author declares that there is no conflict of interest regarding this research paper.

References

[1] Rathore S S and Kumar S. (2015). A decision tree logic based recommendation system to select software fault prediction techniques. Springer, 99, 255–285.

[2] Poth A, Beck Q and Riel A. (2019). Artificial Intelligence helps making Quality Assurance processes leaner. EuroSPI 2019: European Conference on Software Process Improvement, 722-730.

[3] Ceran A A and Tanirover Ö Ö. (2020). An experimental study for software quality prediction with machine learning methods. IEEE HORA 2020 Congress Proceedings, 93-97

[4] Mulla N and Jayakumar Dr N. (2021). Role of Machine Learning & Artificial Intelligence Techniques in Software Testing. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(6), 2913 – 2921.

[5] Hourani H, Hammad A and Lafi M. (2019). The Impact of Artificial Intelligence on Software Testing. IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT) Proceedings, 565 – 570.

[6] Durelli V H S, Durelli R S, Borges S S, Endo A T, E M M, Dias D, and Guimaraes M P. (2019). Machine Learning Applied to Software Testing: A Systematic Mapping Study. IEEE Transactions on Reliability, vol 68, 1189 – 1212.

[7] Chen H and Hossain M. (2022). Application of Machine Learning on Software Quality Assurance and Testing: A Chronological Survey. EPiC Series in Computing, Proceedings of 37th International Conference on Computers and Their Applications, vol 82, 42-52.