The Application Of Machine Learning In Test Case Prioritization - A Review

Elinda Kajo Meçe, Hakik Paci, and Kleona Binjaku

***Abstract*—Regression testing is very important but also a very costly and time-consuming activity that ensures the developers that changes in the application will not bring new errors. Retest all, selection of test cases and prioritization of test cases (TCP) approaches are used to enhance the efficiency and effectiveness in regression testing. While test case selection techniques decrease testing time and cost, it can exclude some critical test cases that can detect the faults. On the other hand, test case prioritization considers all test cases and execute them until resources are exhausted or all test cases are executed, while always focusing on the most important ones. Over the years, machine learning has found wide usage in solving different problems in software engineering. Software development and maintenance problems can be defined as learning problems and machine learning techniques have shown to be very effective in solving these problems. In the range of application of machine learning, machine learning techniques have also found usage in solving the test case prioritization problem. In this paper, we investigate the application of machine learning techniques in test case prioritization. We survey some of the most recent studies made in this field and provide information like techniques of machine learning used in TCP process, metrics used to measure the effectiveness of the proposed methods, data used to define the priority of test cases and some advantages or limitations of application of machine learning in TCP.**

***Index Terms*— Machine Learning, Regression Testing, Review, Test Case Prioritization.**

I. Introduction

Software testing has a significant role in the software development process. It is performed to detect the errors or defects of a software system and guarantees that it works according to its specifications. When changes, such as enhancements, are made in the existing system it is important to uncover any new software bug or error. Regression testing solves this issue. Regression testing is an important process in software testing because it ensures the developers that the changed application will not introduce new errors [1], [2]. However, time, cost and resource constraints make it hard to execute all test cases [3]-[5]. Time is limited because of the quick release cycles for modified software which limits the time for performing regression testing [6]. Regression testing is also an activity that is performed very often, especially in large software, so it requires a lot of resources and has maintenance costs [7],[8]. There exist many techniques to test the changed software within time and resource constraints. There are three categories of regression testing techniques [9]: 1. Retest all Retest all technique consider all previous test cases in testing the changed system. Therefore, it is a time-consuming technique. 2. Test case selection The test case selection technique selects some test cases from the test suite based on some criteria. 3. Test case prioritization Test case prioritization technique order test cases based on some calculated priorities. The main aim of this technique is to execute earlier the test cases that have higher priority, to increase the early fault detection rate [10], [11].

While the test case selection techniques decrease testing time and cost, they can exclude some important test cases that can help to detect the faults [12]. On the other hand, TCP techniques decrease the cost of testing using all test cases. The costs are decreased by parallelizing debugging and testing activities [13]. The advantage of the prioritization of test cases is that it allows continuous testing until resources are exhausted or all test cases are executed, while always executing the most important test cases first [14]. Test case prioritization techniques are classified into two groups: code-based and model-based techniques. The code-based TCP techniques use the source code of the application to prioritize test cases. The model-based TCP techniques use the model, which shows the desired behavior of the system, to prioritize test cases. Many techniques are proposed and developed for test case prioritization. Most used approaches are coverage based, fault-based, model-based, history-based, modification based, similarity-based, genetic-based, etc. [13], [15], [16]. Khatibsyarbini et al. [16] show that the number of publications related to the test case prioritization is still growing in recent years. Years of publications of primary studies in this research also show that the interest in the application of machine learning in TCP is greater in recent years. With the use of learning algorithms system can use the knowledge from previous tests and can adapt itself to changed applications. In this paper, we investigate some of the studies that use machine learning in the process of test case prioritization. The rest of the paper is organized as follows: Section 2 gives details about the research methodology. Section 3 describes shortly each of the primary studies gathered. Section 4 gives the results of this review and give the answers to the research questions. Section 5 gives the threads to validity and finally, some conclusions are mentioned in Section 6.

Published on January 11 , 2020.

E. K. M. is with the Department of Computer Engineering, Polytechnic University of Tirana, Tirane, Albania (e-mail: ekajo@fti.edu.al).

H. P. is with the Department of Computer Engineering, Polytechnic University of Tirana, Tirane, Albania (e-mail: hpaci@fti.edu.al).

K. B. is with the Department of Computer Engineering, Polytechnic University of Tirana, Tirane, Albania (e-mail: Kleona.Binjaku@fti.edu.al).

DOI: http://dx.doi.org/10.24018/ejece.2020.4.1.128

II. RESEARCH METHOD

To perform our review, the systematic and structured method shown in Figure 1 was implemented based on [17], [18].

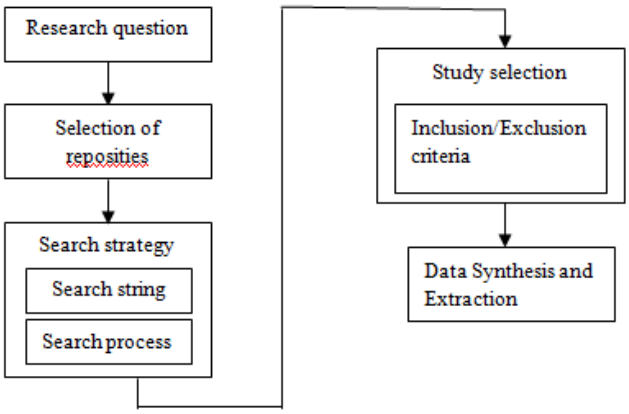

Fig. 1. Phases of the research method

The first step of a review is the definition of the research question that gives the goal of the review. To have reliable answers for the research question confidential sources must be chosen. After that, a strategy of searching the studies that answer the research question must be followed and after selecting studies that fulfill criteria, data synthesis is made.

A. Research questions

As the interest of applying machine learning in test case prioritization is growing, through this review we aim to comprehend and review some of the aspects of this application.

To achieve this goal, we articulated three research questions.

RQ1: What techniques of machine learning are used in TCP?

RQ1.1: What metrics are used to measure the effectiveness of the methods used?

Through these questions, we aim to review all the techniques of machine learning used for test case prioritization to show what the most appropriate techniques are to help researchers for further investigation and developers to pick the appropriate technique. We will also investigate the most used metrics that measure the effectiveness of proposed approaches in TCP.

RQ2: Based on what information is determined the priority of test cases?

In TCP a test case gets priority based on a metric or a combination of metrics. Later on, we will mention what are the metrics that are usually used to archive TCP, but the motivation of this question is to investigate what metrics are used when machine learning techniques are applied in TCP.

RQ3: What are the strengths and limitations of using machine learning in TCP? This research question helps to understand the benefits of applying machine learning in TCP and if there is any limitation

B. Maintaining the Integrity of the Specifications

The strategy of the study is very important as it defines the value of the review. The value of the review is determined from the selected primary studies [16]. The study strategy includes the following steps: -Selection of literature repositories -Determination of the search string -Process of selecting studies.

C. Selection of the literature repositories

To have reliable data about our review choosing reliable sources is very important. For this reason, the chosen repositories to gather the primary studies are: a) SemanticScholar b) ACM Digital Library c) IEEE Xplore d) ScienceDirect e) Springer f) AMA Association for Sensors and Measurement These online databases offer several well-known conferences, symposiums, workshops and journal articles.

D. Determination of the research string

As the focus is to find studies that use machine learning techniques in test case prioritization the used search strings are: "test case prioritization using machine learning", "machine learning in test case prioritization process" and "machine learning in regression testing".

E. Process of selecting the primary studies

A lot of studies were available in each of the repositories. From these papers, we selected relevant ones, based on some criteria.

Inclusion criteria:

-Studies focusing on test case prioritization using machine learning.

-Studies that can answer one of the research questions.

Exclusion criteria:

-Studies that focus on the selection or reduction of test cases using machine learning.

-Studies that focus on TCP but doesn't use any machine learning techniques.

- Studies that don't evaluate the results.

First, the found studies went through inclusion and exclusion criteria. This process is important since there may be studies that are duplicated or are out of the research scope. Then their abstract was studied. Only the papers that had a relation with the research question were selected. Table 1 shows the search procedure.

TABLE I: SELECTION PROCESS

Step	Action	Results
1	Exclusion by title	70 papers
2	Reading the abstract	15 papers
3	Reading full paper	15 papers

As we can see, the number of primary studies that fulfill the inclusion criteria is small. This shows that there are not a lot of studies that focus on the application of machine learning in test case prioritization. Also, the years when these studies are published vary from 2006 to 2018 which shows that this field of research is relatively new. The following graph shows the distribution of the studies over the years.

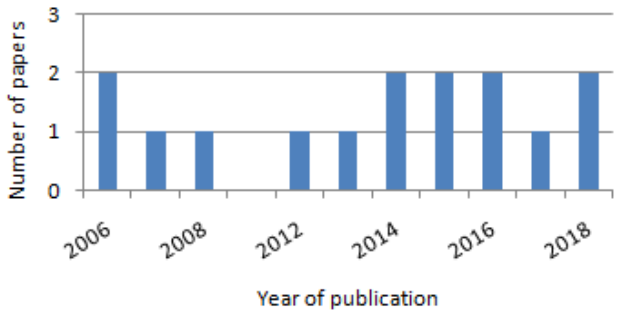

Fig. 2. Distribution of papers over the years

Table 2 shows to which repository belongs each of the primary studies selected. From the primary studies, 3 are from journals, 3 from symposiums and 9 from conferences.

TABLE II: SOURCES OF PRIMARY STUDIES

Source	Paper included
IEEE	6
ACM	2
SemanticScholar	4
ScienceDirect	1
Springer	1
AMA Association for Sensors and Measurement	1

III. Overview of Primary Studies

Before answering our research questions, we will shortly describe how machine learning is applied in each of the studies for the main goal which is test case prioritization. In each description, we try to give only important information that will help us to get answers to our research questions. The studies are classified by the machine learning technique used to get a better overview of what machine learning techniques are used in the TCP process.

A. Neural Network-based TCP

Neural Networks are an imitation of the human brain. They are used to solving different problems that require data processing, classification or regression analysis. They have also found application in solving the test case prioritization problem.

Three of our primary studies use Neural Network-based classification algorithms in the process of test case prioritization. In these studies, the model of the system is used for retrieving the necessary information for ordering test cases. The system model shows the system behavior that interacts with the actions of the user [19]. There are many modeling languages in the literature like UML (Unified Modeling Language) sequences or activity diagrams, state charts, Specification Description Language and ESG (Event Sequence Graphs) [12], [20], [21].

Gökçe et al. [19] order test cases based on their preference degrees, using the model of the system and its specifications. The testing algorithm is based on ESG and generates test suites to minimize the number of tests required for maximum coverage and to also minimize their execution cost.

The approach proposed in this study order the test cases based on their preference degree which is determined by attributes like the number of occurrence layers, the relationship degree, the number of sub-nodes, the number of belongings, the distance, the balancing degree, average frequency of usage.

These attributes are defined using ESG and their classification is done using unsupervised Neural Network clustering.

The performance of the clustering algorithm is evaluated using the mean square error metric (MSE). The results show that the method is very effective in ordering test cases.

This work was extended in [15] where clustering of events is attained using algorithms like unsupervised Neural Network and fuzzy c-means. The proposed technique is based on the specifications of the system and coverage degree. Same as in previous work the model of the system is used to generate test cases and then each test case is assigned with a preference degree.

In this study, a comparison with code-base approaches is made. From the results is concluded that the proposed model-based approach has similar results as considered code-based approaches. Because model-based approaches don't need to access the source code, model-based test case prioritization techniques may be better alternatives over the code-based approaches.

One of the advantages of these two previous techniques is that they don't need prior knowledge, such as source code, number of faults, coverage degree, changed file, etc., which makes these methods different from the most existing approaches.

However, in these model-based approaches, the preference degree of test cases needs to be determined indirectly. As mentioned above, the preference degrees are given by several attributes of events and their frequency of occurrence and then ranking them according to obtained preference degrees. Such as a way test prioritization has high computational cost for a big number of test cases, and it is only capable of ranking the existing test cases without considering the need for updating or extending the test suite.

Gökçe and Eminli [22] present a new model-event based TCP. Test cases are divided into preference groups through the use classification technique. For classification, each test case is presented as a data point with previously obtained index and frequency of occurrence for all events related to a given test case as attributes and its corresponding preference label as class membership. The classification of test cases is done using a multilayer perceptron (MLP) Neural Network.

To verify the accuracy performance of NN mean square error metric (MSE) is used. The results show that the proposed technique improves the effectiveness of the TCP process. Testing time and the capability to predict the label for new test cases are improved.

Spieker et al. [23] propose a new method that prioritizes test cases using reinforcement learning and Neural Networks based on their duration, last execution, and failure history.

By using reinforcement learning, a method that learns from its experience can be designed.

By using a Neural Network, test cases, which have detected faults in previous executions are prioritized and ordered in the way that those which have a higher rate of fault detection are executed first.

Reinforcement learning works based on agents. Agents interact with the environment by perceiving the test case's metadata, which is represented by a state and selects an appropriate action. As an action, the current test cases' priority is returned. The agent learns from the feedback calculated after the test execution and adapts its behavior based on this feedback. In the case of positive feedback previous behavior is. reinforced, while when the feedback is negative, it is discouraged.

It is shown that the results of previous tests are useful in predicting future failures [29].

However, the results of the prediction depend on the length of the test history. A long test history gives more information and a better overview of the distribution of failures in the system. On the other hand, more data must be processed when the history of tests is long because there may be data that might have become inappropriate with previous changes in the system. This adds complexity to the learning process since the agent has to time-weight the results of previous tests.

The performance of the proposed approach is measured, and the results have shown that the performance of network-based agents is very efficient, and it adapts easily because the prioritization of test cases from network-based agents is done with continuous actions.

B. TCP using the genetic algorithm

Genetic algorithms are inspired by the evolution theory of Darvin. It starts from a population which is a set of solutions for a problem and after selecting some solutions using a fitness function, crossover and mutation steps, new solutions are generated. The way the algorithm works is used in the test case prioritization process to order test cases.

Abeleand and Göhner [24] use software agents to prioritize test cases. Software agents are pieces of software that are used to predict the fault-proneness of modules and the fault-revealing probability of each test case. Based on this information the priority of test cases is defined. To estimate the fault-proneness fuzzy logic rules are used. The rules state that modules that have been faulty in the past will be faulty in the future, test cases that have found faults will also find faults in the future or that complex modules have more faults.

The genetic algorithm is used to change the weight of the fuzzy rules. It takes as input the difference between the number of faults that are predicted and the found faults. This requires test cases that can detect a representative number of faults.

The proposed method has some limitations. If test cases cannot be executed due to time restrictions, optimization is not possible. Also, running the algorithm with a small number of faults will not be effective.

On the other hand, the algorithm only optimizes the weights of the fuzzy logic rules and doesn't consider other relations that may ask to change the rules or add new rules.

The authors evaluate the prioritization method comparing it with the classic method that doesn't learn. The comparison is based on a calculated value based on the number of found faults, the severity of faults and the estimation of the developer. The analysis shows that the learning agents improve the prioritization.

In another study, [25] Konsaard and Ramingwong use a modified genetic algorithm to solve the problem of test case prioritization. The advantage of this algorithm is that it is very adaptive and can be customized depending on the problem. Test cases are prioritized based on code coverage.

The approach is compared with five other prioritization techniques: no order, reverse order, random order, optimum order and bee colony optimization using two metrics: Average Percentage Code Coverage (APCC) value and execution time. The results show that the method based on a modified genetic algorithm outperforms other methods. BCO algorithm has also shown to be efficient but it is more complex than the genetic algorithm. The authors show that the approach proposed in this paper offers 100% code coverage.

A more advanced approach based on the genetic algorithm is proposed in [26], where multiple coverage criteria are used. The method is based on the control flow graph and for each test case, three metrics are measured: coverage degree of statements, conditions and multiple conditions. These metrics are integrated through the genetic algorithm and the fitness function of the algorithm outputs one value in each iteration. The calculated values for each test case are used to determine the order of their execution.

From the comparison with other coverage-based techniques, using APFD metric, the authors noticed that this method has lower APFD value, but different from other methods the proposed one takes into consideration the severities of faults, the time of execution for each test case and the structural coverage, while related methods consider only one or two of these features.

C. Bayesian Network-based TCP

Bayesian Networks are graphs used for probability calculation. These graphs show the relationship between different variables. These two sides of Bayesian Networks are utilized for prioritizing test cases in the TCP process.

Mirarab and Tahvildari [27] prioritize test cases based on probability and use Bayesian Networks (BN) to integrate white-box information into a unified model. In the proposed method three kinds of information are used: changes in the source code, coverage degree, and fault-proneness of the software. Bayesian Network is used to relate these data. Based on previous results, a probability is associated with each test case using the probabilistic inference algorithms. Based on these probability values, test cases are ordered.

The technique's performance is evaluated using APFD (Average Percentage of Faults Detected).

The results show that when the number of available faults is representative, the proposed technique achieves high values of APFD.

This work was enhanced in [28] with a feedback mechanism and a new method to gather the information. Also, the authors make a deeper performance evaluation of the approach based on Bayesian Networks, performing different implementation of the method, in many open-source Java objects. Test cases are prioritized based on quality metrics, code change, and coverage degree. Two methods of gathering information are used. The first one compares two versions of a file with code line by line and the other one makes a comparison related to the byte code. The feedback mechanism works in the way that test cases that cover similar modules of the code as a previous test case will have a lower prioritization degree. This way, all test cases will be ordered and test cases that will be executed earlier to cover different parts of the source code.

To evaluate the approach two metrics are used: Average Percentage of Faults Detected (APDF) and running time. From the results, the authors concluded that both methods of gathering information are similar and effective, using Bayesian Networks with the feedback mechanism improves the performance of detecting faults earlier, especially when generated test cases cover the same modules in the code. On the other hand, adding a feedback mechanism is time-consuming.

These previous test case prioritization methods based on Bayesian Networks have good results regarding early failure detection and code coverage, but they don't consider the similarity between test cases. To solve this problem Chen et al. [29] propose a hybrid method that integrates the clustering method based on code coverage and BNs. First test cases are clustered based on their similarity on code coverage and then based on the probability of failure, test cases are prioritized using BN. This way of functioning the method guarantee that similar test cases will not be executed with the same probability. The approach was compared with methods that use BN, BN with feedback and with clustering approach based on code coverage using Average Percentage of Faults Detected (APFD) metric. The results show that the hybrid method outperforms other approaches.

D. Other machine learning techniques for TCP

Besides the above classification that we have done for the techniques of machine learning that are used in the TCP we have found many studies that use other machine learning techniques.

Tonella et al. [30] propose a test case prioritization technique based on the Case-Based Ranking (CBR) machine learning algorithm. CBR takes two inputs: a set of prioritization indexes and some priority information from the user and learns the ranking using this information.

Information like statement coverage and cyclomatic complexity gathered automatically about the test cases, is used to approximate the ordering of test cases, while the user information is used to resolve the cases where there are not enough data, or the data are contradictory. The user has to provide only some local information that is in the form of pairwise comparison. The user should choose between two test cases to define which test case should get higher priority when the algorithm can't decide, for the reasons mentioned above. The prioritization indexes and the information provided from the user are integrated to define test case ordering. This is done in an iterative process. In every learning iteration, the prioritized test suites are saved and the value of the APFD (Average Percentage of Faults Detected) is computed.

The proposed approach is compared to optimal and random prioritization. The results show that this technique is better than optimal and random approaches and user information plays an important role in this improvement.

Lachmann et al. [31] prioritize test cases based on data that don't require access to the source code and in the description of test cases.

The black-box data used are requirements coverage, priority, count and age of failures, and test execution cost.

On the other hand, the test case textual description is parsed. A dictionary of all words in the test cases is computed where each word represents a feature.

These data are used to learn a ranked classification model. The algorithm used for this purpose is ranked support vector machine (SVM Rank), a machine learning algorithm introduced by Joachims [32]. The method is based on the intuition of a human tester who choose the training data by classifying a subset of test cases. Then, a ranked classification model is learned from the SVM RANK algorithm.

To measure the effectiveness of the proposed method authors use the Average Percentage of Faults Detected (APFD) metric [14]. The method is compared with a random order and the results show that the proposed approach increases the failure detection rate.

This work is diversified and extended in [33] using additional ML algorithms to prioritize black-box test cases.

The authors apply machine learning algorithms and an ensemble learning approach. The used machine learning techniques are Neural Networks, k-nearest neighbor (KNN) and logistic regression. They extend their technique by combining the output of several ML algorithms to create an ensemble learner [32].

To create the ensemble learner the authors, consider two approaches: historical and combinatorial ensemble learning. For the first approach, the results of the classifiers of the latest versions are combined to improve the prioritization quality. So, many classifiers are used and are trained based on previous findings. This means that old classifiers are reused. The second ensemble learning approach considers a set of classifiers for the same version. Each of these classifiers determines e priority value for a given test case and the calculated priorities are combined.

The technique effectiveness is measured using the Average Percentage of Faults Detected (APFD) The results show that logistic regression performs better than other ML algorithms. It is also highlighted the importance of the test case description. When the test case description is not available to the ML algorithms, the quality decreases and no algorithm was able to increase its average APFD without using the test case description. Neural Networks showed to be more efficient and the highest failure finding potential is achieved when all four approaches are combined.

Busjaeger and Xie [34] address heterogeneity in industrial environments. The authors apply multiple heuristic techniques like code coverage, textual similarity between tests and changes, test-failure history, fault history, and test age.

To prioritize test cases a ranking model is systematically trained using these data. The authors dealt with the challenge of how to integrate multiple techniques effectively. The features are integrated through a linear model trained using SVMmap.

The accuracy of the proposed method is measured using two metrics: recall (the percentage of relevant items retrieved) and APFD. The method that combines all the techniques has a higher average recall value than any single approach and the APFD value is close to 100%.

Code coverage has shown to be not a strong classifier on its own because a lot of test failures are induced by non-code changes. However, coverage is important when techniques are applied together because it helps predict failures that are not textually or temporally related to changes. The results show that the approach when all of the techniques are applied together is efficient, practical, and convenient for industrial settings.

Chen et al. [35] propose a test case prioritization method of object-oriented systems based on the Adaptive Random Sequences (ARSs) method. ARSs are sequences, in which test cases are spread in the input domain to provide effective failure detection. Authors generate ARSs based on clusters. They propose two methods based on machine learning algorithms for creating clusters with test cases, which have analogous properties. The methods used are method object clustering (MOClustering) and dissimilarity metric clustering (DMClustering). MOClustering uses algorithms like k-means and k-medoids to form clusters, based on properties like the length of method invocation sequences and the number of objects. DMClusterin form clusters using the structure information of tests using K-medoids algorithm. To construct the ARSs within these methods, MSampling (maximum sampling) strategy is used.

The effectiveness of the proposed methods is measured using three metrics: Fm (F-measure) which represents the number of test cases that are executed before the first fault; E represents the number of faults that are detected by test cases and APFD. The proposed methods are compared with RT-ms and Method_Coverage techniques.

The results of the experiment show that the proposed methods perform better than Method_Coverage and RT-ms methods. DMClustering showed to have the best performance and is a good alternative for prioritization of test cases especially when the object-oriented system under test is large.

IV. RESULTS AND DISCUSSION

To get the answers to the research question we have summarized the data gathered in the following table.

TABLE III: EXTRACTED DATA FROM PRIMARY STUDIES

Authors	Type of testing	Machine learning technique used	Information used to determine priority	Advantages / Limitations	Effectiveness metric
Gökçe et *al.* [19]	Black box testing	Unsupervised Neural Networks	Number of occurrence layers, Relationship degree, Number of sub-nodes, Number of belongings, Distance, Balancing degree, Average frequency of usage from ESG	Doesn't need prior information / High computational cost	MSE
Gökçe et *al.* [15]	Black-box Testing	Unsupervised Neural Network, Fuzzy c-means	Number of occurrence layers, Relationship degree, Number of sub-nodes, Number of belongings, Distance, Balancing degree, Average frequency of usage from ESG	Doesn't need prior information / High computational cost	MSE
Gökçe and Eminli [22]	Black –box testing	Multilayer perceptron (MLP) Neural Network.	Index and frequency of occurrence for all events belonging to the given test case as attributes, Preference degree	Divides test cases into preference groups instead of ranking them according to their preference degree	MSE
Spieker [23]	Black –box testing	Neural Networks	Duration of test cases, Last execution, History of failures	The prioritization of test cases is done with continuous actions, Adaptive /Needs a long history of tests, needs more data processing if the test history is long	Normalized APDF
Mirarab and Tahvildari [27]	White-box testing	Bayesian Network	Changes of the source code, Coverage degree, Fault-proneness of the software	Hight value of APFD/ Requires a reasonable number of available faults	APFD

Mirarab and Tahvildari [32]	White-box testing	Bayesian Networks	Quality metrics, code change, coverage degree	Improves performance of detecting faults earlier / Time-consuming	APDF, running time
Chen *et al.* [29]	White-box testing	Bayesian Networks	Similarity on code coverage, Probability of failure	Achieves full coverage, Faster fault detection / Doesn't use real faults	APDF
Abeleand and Göhner [24]	White-box testing	Genetic algorithm	Fault proneness of software modules, Fault –revealing probability of each test case	More efficient than the approach that doesn't learn/There is no optimization if there is no time to execute test cases, There may be other relations, which may require changing the rules	Calculated value based on the number of found faults, the severity of faults and the estimation of the developer
Konsaard and Ramingwong [25]	White-box testing	Genetic algorithm	Code coverage	Adaptive, Simple	APCC, Execution time
Abdel Fatah et al. [26]	White-box testing	Genetic algorithm	Code degree of statements, conditions, multiple conditions	Considers severities of faults, time of execution and structural coverage / Has a low value of APDF	APDF
Tonella [30]	White-box testing	CBR	Statement coverage, Cyclomatic complexity	Integrates multiple ranking indexes with case-based knowledge	APFD
Lachmann et al. [31]	Black-box testing	SVM Rank	Coverage degree of requirements, Failure count, Failure age, Priority of failures, Cost of executing tests, Test case description	Failure detection rate is increased compared to a random order / Is based on the intuition of a human	APFD
Lachmann [33]	Black-box testing	Neural Networks, K-nearest neighbor (KNN), Logistic regression	Coverage degree of requirements, Failure count, Failure age, Priority of failures, Cost of executing tests, Test case description	Achieves high failure finding potential with combined approaches/Challenging in combining approaches in an efficient way	APFD
Busjaeger and Xie [34]	White-box testing	SVMmap	Code coverage, Textual similarity between tests and changes, Test-failure history, Fault history, Test age	Efficient, Practical, Convenient for industrial settings. /Integration of multiple techniques is a challenge	APFD, recall
Chen et al. [35]	Black-box testing	K-means, K-medoids	Fault detection capability	Good choice for test case prioritization, especially for large scale OOS testing	Fm, E, APFD

From the above table, we can easily get answers to our research question.

RQ1: What techniques of machine learning are used in TCP?

A variety of machine learning techniques are used in test case prioritization process. The most used techniques are Neural Networks, Bayesian Networks, and genetic algorithm. Neural networks are very good classifiers and are used to group test cases based on similar properties. This facilitates the ordering of test cases for the TCP process. Bayesian Networks are good in calculating probabilities that are used to order test cases. Also, Genetic algorithm order test cases executing many iterations with three steps each: selecting some of the test cases based on a fitness function, which uses some given features to select them; crossover and mutation. Neural Networks and Bayesian Networks are very adaptive to the problem and very efficient in integrating many features. Bayesian Networks and genetic algorithms have shown to be very simple. A variety of other machine learning techniques have also shown to be efficient in TCP. CBR and SVMmap are also effective in integrating different attributes while K-nearest neighbor, logistic regression and k-means can be used in clustering test cases.

Different machine learning techniques results can also be combined to create an ensemble learner. This approach seems to have good results with the challenge of how to combine the results effectively.

We can conclude that in general machine learning techniques improve the performance for solving TCP problem with a drawback that some of these techniques require a big dataset for training.

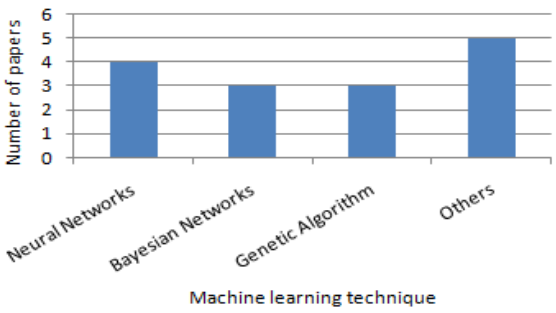

Fig. 3. Distribution of machine learning techniques.

RQ1.1: What metrics are used to measure the effectiveness of the methods used?

To measure the effectiveness of the approaches most of the studies use metrics and compare the result with other approaches using the same metric while some of them only measure the technique performance without comparing with other techniques. The most used metric is APDF (Average Percentage of Faults Detected) because the main aim of prioritizing test cases is ordering them with the purpose of increasing the early fault detection. So, a high APDF value shows that the prioritization of test cases is effective.

RQ2: Based on what information is determined the priority of test cases

Approaches of test case prioritization can be classified as coverage-based, fault-based, history-based, time-based, requirements-based, model-based, risk-based, genetic-based, etc.[16], [36]. In our primary studies, three of them use a model-based approach, three use a coverage base approach, two are fault-based while the others apply multiple approaches together. The results show that combining different approaches brings to better prioritization results but comes with the challenge of combining them efficiently. The ratio between used black-box and white-box features is almost 1. Eight of the studies use features related to the code source like code coverage, but other studies take into account that testers may not have access to the source code, so they use black-box features like failure history, fault detection rate, the textual description of test cases or requirements.

RQ3: What are the strengths and limitations of using machine learning in TCP?

To get the answer to this research question we use the penultimate column of the above table. In this column, the advantages and disadvantages of each approach used in each study are given. From these advantages and disadvantages, we extracted only those related to the machine learning technique. We can say that through machine learning techniques multiple indexes can be integrated for the prioritization of test cases. Also, test cases define priority with continuous actions and machine learning techniques offer easily adaption to changes. The greatest limitation of using machine learning techniques is that they require long data sets to provide reliable results, which means more data processing and massive storage.

V. THREADS TO VALIDITY

The validity of the review may be threatened due to some limitations. Threads of this review may be related to the process of selecting primary studies and the process of data extraction.

A. Process of selecting studies

Even though the research method for selecting relevant studies is presented in detail in section 2, we cannot completely ensure that we have found all the studies related to the application of machine learning in TCP, to answer our RQs. The main reason for this issue is because there may be studies that use different keywords from our research string.

B. Data extraction process

The process of data extraction may be imprecise, and this may be another thread to the validity of this research. There may be two reasons that cause this impreciseness: the data are not extracted systematically, and their classification may be invalid. To reduce the inexact data, we focused only on the data gathered from our primary studies.

VI. CONCLUSIONS

The purpose of this study was to investigate the current state of the application of machine learning in test case prioritization. According to the aim of the study, we formulated three RQs. The motivation of this research was to discover techniques of machine learning that can be applied in TCP, metrics that are used to define the priority and finally some advantages and limitations of using machine learning in TCP. From this study, we concluded that different techniques of machine learning can be used in the test case prioritization process. The most used techniques are Neural Networks, Bayesian Networks, and genetic algorithm. Neural Networks are good classifiers while Bayesian Networks and genetic algorithms are very good at ordering test cases. Neural Networks and Bayesian Networks are very adaptive and can integrate different features. Genetic algorithms and Bayesian Networks are very simple to implement. A variety of other machine learning techniques have also shown to be efficient in TCP. CBR and SVMmap are also effective in integrating different attributes while K-nearest neighbor, logistic regression and k-means can be used in clustering test cases. Creating an ensemble learner that combines results from different machine learning techniques has also resulted effective, but it comes with the challenge of how to combine these results.

To prioritize test cases half of the studies, use white-box features and half of them black-box ones. This may be because of research purpose to give alternatives and solutions for both cases when testers have and when they don't have access to the source code. Better results can be provided using a combination of features. Finally, we can conclude that machine learning techniques are very effective in solving the test case prioritization problem because they offer easily adaption to changes, good classification and prioritization results with the drawback that they require long data sets for training, to have reliable results.

REFERENCES

[1] D. Suleiman, M.Alian, A. Hudaib, "A survey on prioritization regression testing test case", presented at the 8th International Conference on Information Technology (ICIT), Amman, Jordan, May 17-18, 2017.

[2] P. K. Chittimalli, M. J. Harrold, "Recomputing coverage information to assist regression testing", *IEEE Trans. Software Eng*, vol. 35, pp. 452–469, Aug. 2009.

[3] E. Khanna, (November 2016). Regression testing based on genetic algorithm, *International Journal of Computer Applications – IJCA [Online]*. 154(8). *pp. 43-46.* Available: https://www.ijcaonline.org/archives/volume154/number8/26515-2016912201

[4] M. Al-Refai, S.Ghosh, W. Cazzola, "Model-based regression test selection for validating runtime adaptation of software systems", presented at the 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST), Chicago, IL, USA, April 10-15, 2016.

[5] J. S. Rajal, S. Sharma, A review on various techniques for regression testing and test case prioritization, *International Journal of Computer Applications (0975 – 8887)* [Online]. 116 (16). Available: https://pdfs.semanticscholar.org/1601/3e34d8b1ff09c2271195c14fdf31557cb2ec.pdf

[6] T. L. Graves, M. J. Harrold, J. Kim, A. Porter, G. Rothermel, "An empirical study of regression test selection techniques", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 10, pp. 184–208, April 2001.

[7] P. K., A. A. Rao, K. V. G. Rao, S. S. S. Reddy, M. Gopichand, "Cost effective model based regression testing", in *Proceedings of the World Congress on Engineering 2017*, 2017, vol. 1.

[8] P. K.Chittimalli and M. J. Harrold, "Regression test selection on system requirements", in *ISEC '08 Proceedings of the 1st India software engineering conference*, pp. 87–96, India, 2008.

[9] G. Duggal and Mrs. B. Suri, "Understanding regression testing techniques", presented at the COIT, India, 2008.

[10] Varun Kumar, Sujata, and Mohit Kumar, Test case prioritization using fault severity, *International Journal of Computer Science and Technology (IJCST)* [Online]. 1(1). pp. 67–71.

[11] S. Yoo and M. Harman, Regression testing minimisation, selection and prioritisation: a survey, *Journal of Software Testing Verification and Reliability* [Online]. 22(2). pp. 67–120. Available: https://onlinelibrary.wiley.com/doi/10.1002/stvr.430

[12] C. Catal and D. Mishra, Test case prioritization: a systematic mapping study, *Software Quality Journal* [Online]. 21(3). pp. 445–478. Available: http://www.ijirst.org/articles/IJIRSTV1I10014.pdf

[13] H. Do, S. Mirarab, L. Tahvildari, G. Rothermel, "The effects of time constraints on test case prioritization: A series of controlled experiments", *IEEE Trans. Softw. Eng.*, vol. 36, pp. 593–617, Oct. 2010.

[14] G. Rothermel, R.H. Untch, C. Chu, M.J. Harrold, "Prioritizing test cases for regression testing", *IEEE Trans. Software Eng*, vol. 27, pp. 929–948, Oct. 2001.

[15] N. Gökçe, F. Belli, M. Eminli, B. T. Dinçer, Model-based test case prioritization using cluster analysis-a aoft-computing approach, *Turkish Journal of Electrical Engineering Computer Sciences* [Online]. 23 (3), pp. 623 – 640, Available: http://journals.tubitak.gov.tr/elektrik/issues/elk-15-23-3/elk-23-3-1-1209-109.pdf

[16] M. Khatibsyarbini, M. A. Isa, D. N.A. Jawawi, R. Tumeng, Test case prioritization approaches in regression testing: A systematic literature review, *Journal of Information and Software. Technology* [Online]. 93 (1), pp. 74–93, Available: https://www.sciencedirect.com/science/article/abs/pii/S0950584916304888

[17] B. Kitchenham," Procedures for performing systematic reviews", TR/SE-0401, 2005.

[18] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering - A systematic literature review, *Journal of Information and Software. Technology* [Online]. 51(1), pp. 7–15, Available: https://www.sciencedirect.com/science/article/abs/pii/S0950584908001390

[19] N. Gökçe, M. Eminov, F. Belli, "Coverage-based, prioritized testing using neural network clustering", presented at the Computer and Information Sciences – ISCIS, Istanbul, Turkey, November 1-3, 2006.

[20] B. Korel, G. Koutsogiannakis, Experimental comparison of code-based and model-based test prioritization, in *ICSTW '09 Proceedings of the IEEE International Conference on Software Testing, Verification, and Validation Workshops*, pp. 77–84, Washington, DC, USA, 2009.

[21] F. Belli, "Finite-state testing and analysis of graphical user interfaces", in *Proceedings 12th International Symposium on Software Reliability Engineering*, Hong Kong, China, 2001.

[22] N. Gökçe, M. Eminli, Model-based test case prioritization using neural network classification, Computer *Science & Engineering: An International Journal (CSEIJ)* [Online]. 4(1), pp. 15–25. Available: https://www.academia.edu/12477424/MODEL-BASED_TEST_CASE_PRIORITIZATION_USING_NEURAL_NETWORK_CLASSIFICATION

[23] H. Spieker, A. Gotlieb, D. Marijan, D. Marijan, "Reinforcement learning for automatic test case prioritization and selection in continuous integration", in *ISSTA 2017 Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 12–22, Santa Barbara, CA, USA, 2017,

[24] S. Abele, P. Göhner, "Improving proceeding test case prioritization with learning software agents", in *ICAART 2014 - Proceedings of the 6th International Conference on Agents and Artificial Intelligence*, pp. 293–298, vol. 2, France, 2014,

[25] P. Konsaard, L. Ramingwong, "Total coverage based regression test case prioritization using genetic algorithm", presented in 2015 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Hua Hin, Thailand, June 24-27, 2015.

[26] A. A. Ahmed, M. Shaheen, E. Kosba, "Software testing suite prioritization using multicriteria fitness function", in 2012 22nd International Conference on Computer Theory and Applications (ICCTA), Alexandria, Egypt, October 13-15, 2012.

[27] S. Mirarab, L. Tahvildari," A prioritization approach for software test cases based on Bayesian networks", in *FASE'07 Proceedings of the 10th international conference on Fundamental approaches to software engineering*, pp. 276–290, Braga, Portugal, 2007.

[28] S. Mirarab and L. Tahvildari, An empirical study on bayesian network-based approach for test case prioritization, presented in 2008 1st International Conference on Software Testing, Verification, and Validation, Lillehammer, Norway, April 9-11, May 2008.

[29] X. Zhao, Z. Wang, X. Fan, Z. Wang, "A clustering – bayesian network-based approach for test case prioritization", presented in 2015 IEEE 39th Annual Computer Software and Applications Conference,Taichung, Taiwan, July 1-5, 2015.

[30] P. Tonella, P. Avesani, A. Susi, "Using the case-cased ranking methodology for test case prioritization", presented at the 22nd IEEE International Conference on Software Maintenance, Philadelphia, Pennsylvania, 2006.

[31] R. Lachmann, S. Schulze, M. Nieke, C. Seidl, I.Schaefer, "System level test case prioritization using machine learning", presented in 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), California, USA, Dec. 18-20, 2016.

[32] T. Joachims, Optimizing search engines using clickthrough data, in *KDD '02*, 2002, pp. 133–142.

[33] R. Lachmann, "Machine learning-driven test case prioritization approaches for black-box software testing", presented at the The European Test and Telemetry Conference, Nuremberg, Germany, June 26-28, 2018.

[34] B.Busjaeger, T. Xie," Learning for test prioritization: an industrial case study", in *FSE 2016 Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp. 975–980, Seattle, WA, USA, 2016.

[35] J. Chen *et al.*, Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering, Journal of Systems and Software, vol. 135, 107–125, Jan. 2018.

[36] R. Mukherjee, K. Sridhar Patnaik, A survey on different approaches for software test case prioritization, Journal *of King Saud University - Computer and Information Sciences [Online]*, Available: https://www.sciencedirect.com/science/article/pii/S1319157818303616.