International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V71I9P203

Volume 71 Issue 9, 27-35, September 2023
© 2023 Seventh Sense Research Group®

Original Article

Utilizing Machine Learning Algorithms to Automatically Categorize Software Test Cases

Abdullahi Ahmed Abdirahma[1*], Abdirahman Osman Hashi[1], Siti Zaiton Mohd Hashim[2], Mohamed Abdirahman Elmi[1]

[1]*Faculty of Computing, SIMAD University, Mogadishu-Somalia.*
[2]*Department of Computer Science, Faculty of Computing, University Teknologi Malaysia, Skudai, Johor, Malaysia.*

[*]*Corresponding Author : wadani12727@gmail.com*

Received: 04 June 2023 Revised: 31 July 2023 Accepted: 08 August 2023 Published: 03 September 2023

Abstract - *The creation of an efficient strategy to help decrease the problem of manual effort expended by software developers when labelling software test cases has been the focus of many academic researchers. To ensure that all features and applications are fully tested, it is important to have a framework that can effectively match the feature labels and test cases in the correct sequence. Irrelevant labeling of test cases can result in inaccuracy, so avoiding it is a key objective of this paper. As a result, the primary goal of this work is to extend a previous method for doing automatic directory categorization of test cases based on their test case description using the K-nearest-neighbor classifier, Logistic regression, Decision tree and MLP. Bag-of-word (Bow) is used as a vector representation and fits all classifiers. The experimental results reveal that using KNN-BOW and MLP have a higher score than Logistic regression and Decision tree since it outperformed and obtained 77% accuracy vs. 71% for KNN-TF-IDF. Meanwhile, we extended using KNN-BOW and MLP-BOW have scored a good result compared to Logistic regression and Decision tree, as it outperformed and achieved 77% accuracy in comparison with the 67% and 65% that Logistic regression and Decision tree achieved, respectively. As a result, KNN-BOW and MLP-BOW are excellent choices for directory categorization based on test case descriptions. The suggested strategy contributes to the domain by ensuring that machine learning algorithms can easily directly classify test-case descriptions.*

Keywords - *Test cases, K nearest neighbors, Bag of words, Logistic regression, Decision tree.*

1. Introduction

Software testing is the primary factor in effectively designing software and ensuring its accuracy about the operation or functionality that is intended to be done under various input variables [1]. There are several ways or tactics for doing software testing. Black box and white box testing are the most popular. Other categorizations of software testing methodologies may be made based on how the testing is performed. Classification test scenarios, according to this, can be either manual or automated [2].

Manual testing involves manually constructing test cases and is more susceptible to human mistakes, whereas automation testing involves capturing the different test cases based on actions the end user has to take. In contrast, rather than wasting time, it is important to use automated testing when developing test cases to increase efficiency. Furthermore, manual testing is unsuitable for heavy software that has a lot of components, such as that used by firms that manufacture Android phones [3]. The reason is that Android is a multi-layered software platform comprised of apps, drivers, components, a kernel and many more. Obviously, this software is complicated, and Android makers demand extensive testing to ensure that the system is performing as planned and meets the organization's hardware and software requirements [4].

Meanwhile, in the field of software testing, the management and organization of test cases play a crucial role in ensuring the quality and reliability of software systems. As software projects grow in size and complexity, the number of test cases also increases exponentially, making manual organization and classification a challenging and time-consuming task [5]. To address this issue, researchers have turned to machine learning algorithms to automate the process of directory classification for test cases. By leveraging the power of machine learning, it becomes possible to categorize test cases automatically, thus improving efficiency and reducing human effort [6].

Machine learning algorithms offer the capability to learn patterns and relationships from labelled datasets, enabling the development of accurate classification models. These models can be trained to classify test cases into different directories based on their features and attributes. Several studies have explored the application of machine learning algorithms in this context, aiming to enhance the effectiveness and scalability of test case management [7]. One commonly employed approach in automatic directory classification is feature extraction. Test cases can be represented using various features, such as keywords, textual descriptions, or metadata. These features serve as input to machine learning models, which learn to associate them with specific directories based on the provided labels. Researchers have investigated different feature extraction techniques, including natural language processing methods, to capture the semantic meaning and context of test cases [9].

Numerous solutions have been proposed by scholars to address the problem, including manual tagging or feature name assignment and even the implementation of an application lifecycle management system. However, these recommended procedures continue to result in inaccurately labelled test cases. Therefore, the primary objective of this study is to automate the classification of test cases based on their information, with the aim of reducing the time and effort required by software developers to classify them manually. The ultimate goal is to test and validate all features and applications thoroughly. A robust framework capable of matching feature labels with their respective test cases in chronological order is essential to accomplish this. The proposed technique will determine the appropriate directory for each test case based on the component to which it belongs, ensuring that all test cases are correctly categorized.

For the organization of the paper, it is divided into five parts. The background of test cases and topic classification is in the first two sections. The third part outlines how the suggested approach will be applied to our model. Section fourth contains the results of the suggested framework and analysis. Finally, the conclusion are presented in the fifth part.

2. Related Work

As software test-case classification is a vital area of research, scholars have proposed numerous methods, including manual tagging, application-lifecycle management, and automated topic classification. Each approach has its own advantages and disadvantages in terms of the time required and the frequency of incorrect classification. For instance, at a particular Android smartphone manufacturer, tests are tagged manually, and categories and feature names are specified. It is essential that these features are accurately identified; otherwise, the test cases will fail, and the developer will be unable to determine which features and applications have been tested. Another proposed solution to this problem is application lifecycle management (ALM). ALM is a method used to manage a predetermined process that supports software development from the beginning until the end, including the release of the product and post-release support. This process may involve defect-tracking, repair, or testing, and it should be linked through a web interface or a customized window application form [10].

On another note, recent years have seen a significant amount of research being published on autonomous test case development methods [2]. This has led to the classification of test case creation into three types: random-based and requirement-based, program-based, feature extraction, machine learning and test case maintenance and evolution.

2.1. Requirement-Based

The process of creating test cases based on software requirements is commonly known as specification test-case generation, as the test case itself can be a semi-formal or formal definition of the data or function needed for the program being tested [3]. Various software requirement formalisms, including logic programs, finite-state machines, and first-order logic, can drive formal test case specifications. Alternatively, the diagram notation of software systems may be utilized for semi-formal specifications. The structural requirements for the test case benchmark are often presented in a hierarchical form, as described in the dataflow diagram. These approaches, along with others, can help ensure that the test cases effectively meet the requirements of the software being tested [11].

2.2. Random-Based

Random-based test cases refer to a subset of possibility models created during software processes. This method involves selecting test cases from the software's input space using random sampling based on a specified probabilistic distribution. This approach is often used for simple random testing by applying prior software operations at random. However, a more complex random testing method involves using a stochastic model. Sophisticated models like Markov Chains and Bayesian Networks have been utilized to create more advanced testing methods. These advanced techniques have been used for flaw identification, reliability testing, and functional validation and verification. By using probabilistic models and random sampling, these techniques can help ensure that the software functions as intended, even under varying conditions [12].

2.3. Program-Based

Program-based test cases involve analysing the source code of the program being tested without considering its execution. This approach is classified as a static-test creation or generation technique since it does not consider the program's behaviour during execution. Additionally, program-based testing is route-oriented, meaning it always accepts a specific way as input when creating test cases. This method is also referred to as a targeted strategy because it is focused on identifying which way leads to the execution of the statement. While program-based testing may not account for the program's dynamic behaviour during execution, it is still a useful method for ensuring that the source code is correct and that it meets the intended goals. This testing approach can help identify potential issues with the program's logic or syntax and can improve its overall quality [13].

2.4. Feature Extraction

Feature extraction is a critical step in the automatic directory classification of test cases. It involves transforming the raw input data (test cases) into a set of meaningful features that can be used as input for machine learning algorithms. Various feature extraction techniques have been explored in the literature to capture the relevant information from test cases and improve the accuracy of classification models [13]. Natural language processing (NLP) techniques have been widely applied in test case classification to extract features from textual labels. These techniques leverage linguistic and statistical methods to analyze the textual content of test cases and derive meaningful representations. For example, Bag-of-Words (BoW) representation is commonly used, where each test case is represented as a vector of word frequencies. The presence or absence of specific keywords or phrases can also be used as binary features [14].

In addition to BoW, more advanced NLP techniques have been employed for feature extraction. Term Frequency-Inverse Document Frequency (TF-IDF) is a technique that weights words based on their importance in a document collection [15]. This approach considers not only the frequency of words but also their rarity across the entire corpus. It helps capture the discriminatory power of words and enhances the discriminative features used for classification.

Another approach is word embedding, representing words as dense vectors in a continuous space. Word2Vec and GloVe are popular algorithms for learning word embeddings. These techniques capture semantic relationships between words and enable the comparison of word meanings based on their vector representations. Test cases can be represented by averaging or concatenating the word embeddings of the words present in the textual descriptions [16].

In addition to NLP techniques, metadata associated with test cases can also serve as valuable features. Metadata can include information such as the test case author, creation date, associated requirements, and execution status. This metadata can provide insights into the characteristics and context of test cases, which can be leveraged for classification [17].

2.5. Machine Learning

Machine learning methods are critical components of test case automated directory categorization systems. These approaches use labelled datasets to train models that can predict the correct directory for unknown test situations. For this job, many machine learning techniques have been investigated in the literature, with the goal of improving the accuracy and efficiency of categorization models [18].

For test case categorization, decision trees are prominent techniques. Based on the characteristics retrieved from test cases, they build a decision tree-like model. Each internal node in the tree reflects a feature-based judgement, while each leaf node corresponds to a directory name. Decision trees are interpretable and can handle both categorical and continuous variables, making them valuable for understanding the classification process [19, 23].

Random forests are an ensemble learning approach for improving classification accuracy by combining numerous decision trees. Each tree is trained using a randomly selected subset of the training data and features. Individual tree projections are then integrated to determine the final categorization choice. Random forests are well-known for their resilience, capacity to handle large feature spaces and resistance to overfitting [20].

Support Vector Machines (SVMs) are binary classifiers that seek the best hyperplane for separating data points from distinct classes with the least amount of overlap. SVMs translate the input characteristics into a higher-dimensional space to identify a linear or non-linear decision boundary. They have been extensively used in a variety of classification tasks, including test case classification, and are capable of handling both linear and non-linear correlations between data and classes [21].

In recent years, neural networks, especially deep learning models, have received a lot of attention for their capacity to learn complicated patterns and correlations. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are extensively employed architectures in test case classification. CNNs thrive at collecting geographical and local characteristics in test case data, while RNNs excel at capturing temporal relationships in sequential test case data [22].

Researchers have also investigated strategies like hyperparameter tweaking, feature selection, and ensemble learning to increase the performance of machine learning systems. The process of optimising the parameters of machine learning algorithms to identify the optimal configuration for a particular dataset is known as hyperparameter tuning. The goal of feature selection approaches is to discover the most important characteristics for classification, hence lowering the dimensionality of the input data and increasing the model's efficiency. Ensemble learning leverages the variety and capabilities of individual models by combining numerous models to produce collective predictions [24].

2.6. Test Case Maintenance and Evolution

Automatic directory classification of test cases not only aids in organizing and managing test cases efficiently but also supports test case maintenance and evolution throughout the software development lifecycle. As software systems evolve and requirements change, the classification of test cases needs to be updated to reflect these modifications. When changes are made to the directories or new directories are introduced, the classification models need to be adapted to ensure the accurate categorization of test cases. Automatic directory classification simplifies this process by automatically reflecting changes in the classification models. As new test cases are added, the models can predict their appropriate directories based on the updated classification rules [25, 26].

This capability of automatic directory classification greatly benefits test case maintenance. Manual reclassification of test cases can be a time-consuming and error-prone task. However, with automatic classification, the effort required to maintain and update test case classification is significantly reduced, allowing testers and developers to focus on other critical activities [18].

In addition, the evolution of test cases can be better managed through automatic directory classification. As test cases are executed, and their results are recorded, the classification models can be updated to incorporate this feedback. For example, if a test case consistently fails, the model can learn to associate it with directories related to defect areas, facilitating targeted debugging and maintenance efforts [27].

By automating the process of directory classification, the reusability of test cases is also enhanced. Test cases categorized into directories based on their functionalities or characteristics can be efficiently retrieved and reused for specific testing scenarios. This reduces redundant test case creation and promotes efficient utilization of existing test assets. Moreover, automatic directory classification contributes to the traceability and organization of test cases. By accurately categorizing test cases, it becomes easier to track their coverage of specific requirements, modules, or functionalities. This traceability information can be leveraged for test case prioritization, impact analysis, and coverage assessment [28].

Researchers have proposed using feature labelling as a method to tackle the problem of accurately identifying and categorizing software features. However, this approach presents challenges as it is difficult to come up with suitable characteristics that are unique to each feature and team. Additionally, each label must be assigned manually, which requires domain knowledge and extensive investigation. Consequently, the process of feature labelling can be time-consuming, similar to other labelling activities. Despite these challenges, feature labelling remains an important method to help organize and manage software features, which can enhance software development practices.

The proposed method involves summarizing test case descriptions, but assigning feature labels manually poses a risk of human error. Instead, the language used in the test case description should be analysed to determine the appropriate category for each feature label. The test case descriptions used in this approach were collected from Android smartphone providers, and this data was used to develop a text-based directory classification system. This approach is based on the concept of application lifecycle management and builds upon the work of previous researchers who have developed the concept of topic categorization. The goal is to expand the areas of feature labelling that were previously limited to test case descriptions. By doing so, the proposed method can help improve the accuracy and efficiency of test case management, leading to better software development practices.

3. Methodology

The primary objective of this study is to propose a method for automatically classifying test cases based on their descriptions. To achieve this goal, the researchers have developed various research techniques that align with their objectives. The study presents a detailed methodology that outlines the different approaches that can be used to attain the system's intended purpose. By thoroughly examining the methods used, the study aims to provide insights that can inform future research in the area of test case classification.

3.1. Research Framework

The focus of this approach is on test case descriptions, which are divided into two parts for analysis: training and testing. The raw data was split in half, with 50% being allocated to each. We tried using a loop for the data split but found duplicate data, so we followed previous studies and divided the dataset into two subsets. We then worked on vectorizing the training text so that it could be turned into a bag of words. In order to eliminate certain words and terms, keywords with numeric characters and those appearing (<5%) were removed. A flowchart of the entire process was provided for clarity.

The suggested architecture is represented in Figure 1, which shows how each phase is connected and the way in which data is sourced from ALMS to score predicted. Meanwhile, it is important to note that preliminary steps need to be taken before we train the models, which are discussed before training the model:

- Transforming Data
- Cleaning Data
- Steaming and also Lemmatizing

- Doing Vectorization

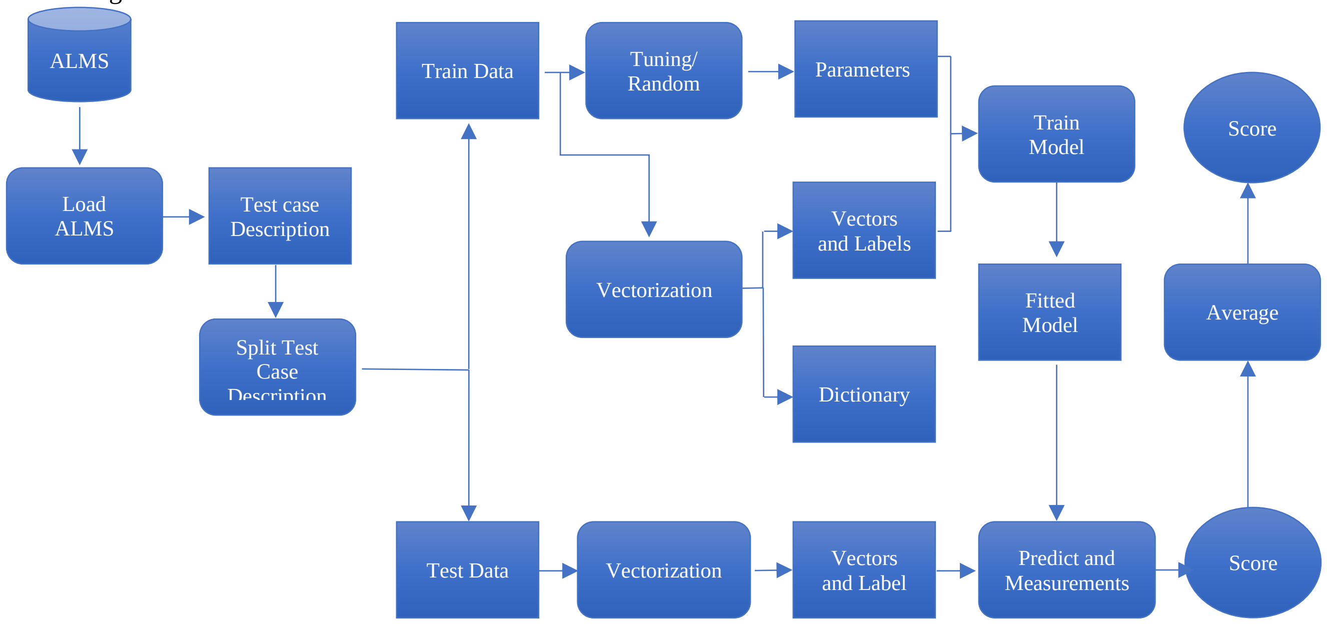
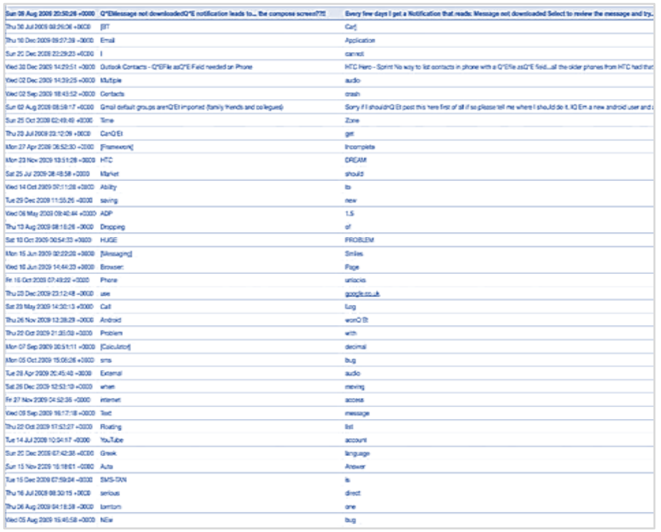


Fig. 1 Proposed framework

3.1.1. Transformation Data

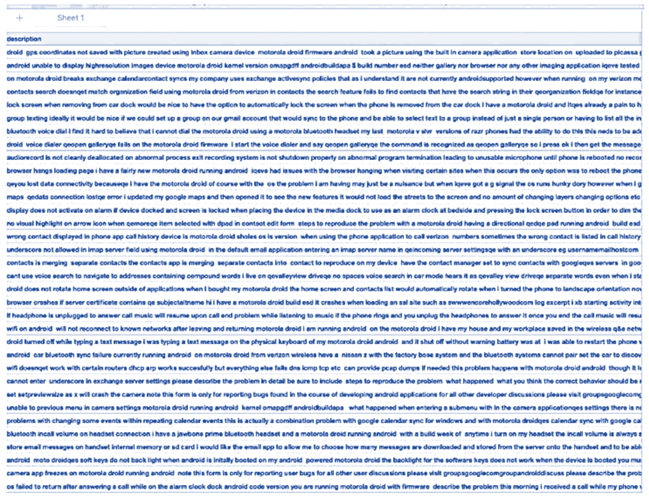
The first thing we did was alter the data. Because structured data most likely increases data quality, it was required. Moreover, the modified data will make fitting the model easier. Here is the original dataset Figure 2 before data modification.



Sun 08 Aug 2009 20:50:28 +0000	Q"EMessage not downloadedQ"E notification leads to... the compose screen?!!	Every few days I get a Notification that reads: Message not downloaded Select to review the message and try...
Thu 30 Jul 2009 08:26:06 +0000	[BT	Car]
Thu 10 Dec 2009 09:07:39 +0000	Email	Application
Sun 20 Dec 2009 22:29:23 +0000	I	cannot
Wed 30 Dec 2009 14:29:51 +0000	Outlook Contacts - Q"EFile asQ"E Field needed on Phone	HTC Hero - Sprint No way to list contacts in phone with a Q"EFile asQ"E field...all the older phones from HTC had that
Wed 02 Dec 2009 14:39:25 +0000	Multiple	audio
Wed 02 Sep 2009 18:43:52 +0000	Contacts	crash
Sun 02 Aug 2009 08:58:17 +0000	Gmail default groups aren'Q'Et imported (family friends and collegues)	Sorry if I shouldn'Q'Et post this here first of all if so please tell me where I should do it. IQ'Em a new android user and a
Sun 25 Oct 2009 02:49:49 +0000	Time	Zone
Thu 23 Jul 2009 23:12:09 +0000	CanQ'Et	get
Mon 27 Apr 2009 08:52:30 +0000	[Framework]	Incomplete
Mon 23 Nov 2009 13:51:28 +0000	HTC	DREAM
Sat 25 Jul 2009 08:48:58 +0000	Market	should
Wed 14 Oct 2009 07:11:28 +0000	Ability	to
Tue 29 Dec 2009 11:55:26 +0000	saving	new
Wed 06 May 2009 09:40:44 +0000	ADP	1.5
Thu 13 Aug 2009 08:18:26 +0000	Dropping	of
Sat 10 Oct 2009 00:54:33 +0000	HUGE	PROBLEM
Mon 15 Jun 2009 02:22:20 +0000	[Messaging]	Smiles
Wed 10 Jun 2009 14:44:23 +0000	Browser:	Page
Fri 16 Oct 2009 07:49:22 +0000	Phone	unlocks
Thu 03 Dec 2009 23:12:48 +0000	use	google.co.uk
Sat 23 May 2009 14:30:13 +0000	Call	Log
Thu 26 Nov 2009 12:38:29 +0000	Android	wonQ'Et
Thu 22 Oct 2009 21:35:03 +0000	Problem	with
Mon 07 Sep 2009 20:51:11 +0000	[Calculator]	decimal
Mon 05 Oct 2009 15:06:26 +0000	sms	bug
Tue 28 Apr 2009 20:45:40 +0000	External	audio
Sat 26 Dec 2009 12:53:10 +0000	when	moving
Fri 27 Nov 2009 04:52:36 +0000	internet	access
Wed 09 Sep 2009 16:17:18 +0000	Text	message
Thu 22 Oct 2009 17:53:27 +0000	Floating	list
Tue 14 Jul 2009 10:04:17 +0000	YouTube	account
Sun 20 Dec 2009 07:42:38 +0000	Greek	language
Sun 15 Nov 2009 18:18:01 +0000	Auto	Answer
Tue 15 Dec 2009 07:59:24 +0000	SMS-TAN	is
Thu 16 Jul 2009 08:30:15 +0000	serious	direct
Thu 06 Aug 2009 04:18:58 +0000	tomtom	one
Wed 05 Aug 2009 15:46:58 +0000	NEw	bug

Fig. 2 Actual dataset

The second image in the research paper shows the actual dataset with test case descriptions separated into columns. This format makes it difficult to match the data with the training model. Therefore, data transformation is needed to make test case descriptions in a manner that can be trained to models, which directly impacts its quality. Figure 3 demonstrates the data after undergoing the transformation process.



Sheet 1

description
droid gps coordinates not saved with picture created using inbox camera device motorola droid firmware android took a picture using the built in camera application store location on uploaded to picassa g
android unable to display highresolution images device motorola droid kernel version omapgdff androidbuildapa $ build number esd neither gallery nor browser nor any other imaging application iqeve tested
on motorola droid breaks exchange calendarcontact syncs my company uses exchange activesync policies that as i understand it are not currently androidsupported however when running on my verizon mo
contacts search doesnqet match organization field using motorola droid from verizon in contacts the search feature fails to find contacts that have the search string in their qeorganization fieldqe for instance
lock screen when removing from car dock would be nice to have the option to automatically lock the screen when the phone is removed from the car dock i have a motorola droid and itqes already a pain to h
group texting ideally it would be nice if we could set up a group on our gmail account that would sync to the phone and be able to select text to a group instead of just a single person or having to list all the in
bluetooth voice dial i find it hard to believe that i cannot dial the motorola droid using a motorola bluetooth headset my last motorola v slvr versions of razr phones had the ability to do this this neds to be ad
droid voice dialer qeopen galleryqe fails on the motorola droid firmware i start the voice dialer and say qeopen galleryqe the command is recognized as qeopen galleryqe so i press ok i then get the message
audiorecord is not cleanly deallocated on abnormal process exit recording system is not shutdown properly on abnormal program termination leading to unusable microphone until phone is rebooted no recor
browser hangs loading page i have a fairly new motorola droid running android iqeve had issues with the browser hanging when visiting certain sites when this occurs the only option was to reboot the phone
qeyou lost data connectivity becauseqe i have the motorola droid of course with the os the problem i am having may just be a nuisance but when iqeve got a g signal the os runs hunky dory however when i g
maps qedata connection lostqe error i updated my google maps and then opened it to see the new features it would not load the streets to the screen and no amount of changing layers changing options etc
display does not activate on alarm if device docked and screen is locked when placing the device in the media dock to use as an alarm clock at bedside and pressing the lock screen button in order to dim the
no visual highlight on arrow icon when qemoreqe item selected with dpad in contact edit form steps to reproduce the problem with a motorola droid having a directional qedqe pad running android build esd
wrong contact displayed in phone app call history device is motorola droid sholes os is version when using the phone application to call verizon numbers sometimes the wrong contact is listed in call history
underscore not allowed in imap server field using motorola droid in the default email application entering an imap server name in qeincoming server settingsqe with an underscore eg usernamemailhostcom
contacts is merging separate contacts the contacts app is merging separate contacts into contact to reproduce on my device have the contact manager set to sync contacts with googleqes servers in goo
cant use voice search to navigate to addresses containing compound words i live on qevalleyview driveqe no spaces voice search in car mode hears it as qevalley view driveqe separate words even when i sta
droid does not rotate home screen outside of applications when i bought my motorola droid the home screen and contacts list would automatically rotate when i turned the phone to landscape orientation nov
browser crashes if server certificate contains qe subjectaltname hi i have a motorola droid build esd it crashes when loading an ssl site such as swwwencorehollywoodcom log excerpt i xb starting activity int
if headphone is unplugged to answer call music will resume upon call end problem while listening to music if the phone rings and you unplug the headphones to answer it once you end the call music will resu
wifi on android will not reconnect to known networks after leaving and returning motorola droid i am running android on the motorola droid i have my house and my workplace saved in the wireless q&e netw
droid turned off while typing a text message i was typing a text message on the physical keyboard of my motorola droid android and it shut off without warning battery was at i was able to restart the phone w
android car bluetooth sync failure currently running android on motorola droid from verizon wireless have a nissan z with the factory bose system and the bluetooth systems cannot pair set the car to discov
wifi doesnqet work with certain routers dhcp arp works succesfully but everything else fails dns icmp tcp etc can provide pcap dumps if needed this problem happens with motorola droid android though it lo
cannot enter underscore in exchange server settings please describe the problem in detail be sure to include steps to reproduce the problem what happened what you think the correct behavior should be r
set setpreviewsize as x will crash the camera note this form is only for reporting bugs found in the course of developing android applications for all other developer discussions please visit groupsgooglecomg
unable to previous menu in camera settings motorola droid running android kernel omapgdff androidbuildapa what happened when entering a submenu with in the camera applicationqes settings there is no
problems with changing some events within repeating calendar events this is actually a combination problem with google calendar sync for windows and with motorola droidqes calendar sync with google cal
bluetooth incall volume on headset connection i have a jawbone prime bluetooth headset and a motorola droid running android with a build week of anytime i turn on my headset the incall volume is always s
store email messages on handset internal memory or sd card i would like the email app to allow me to choose how many messages are downloaded and stored from the server onto the handset and to be abl
android moto droidqes soft keys do not back light when android is initally booted on my android powered motorola droid the backlight for the software keys does not work when the device is booted you mu
camera app freezes on motorola droid running android note this form is only for reporting user bugs for all other user discussions please visit groupsgooglecomgroupandroiddiscuss please describe the prob
os failed to return after answering a call while on the alarm clock dock android code version you are running motorola droid with firmware describe the problem this morning i received a call while my phone w

Fig. 3 Data transformed

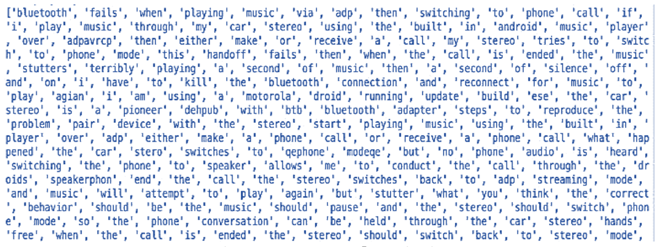
3.1.2. Cleaning Data

To ensure the success of the proposed model, the test case descriptions were cleaned up by removing digits, punctuation, and capitalization. The data cleaning process is crucial to the project's success since it removes any irrelevant information, making the data more useful for analysis. The author emphasizes the importance of having clean data and takes extra precautions to remove unnecessary information, such as number phrases or alphanumeric letters.

3.1.3. Steaming and Lemmatizing

The primary objective of stemming is to identify the essential base root of a word of the description, while lemmatizing aims to decrease inflected base words to their base form. Prior to transforming the words into vectors, it is necessary to extract the base form of each verb, which is the root word. Figure four displays the initial form of the test case description, and the subsequent images illustrate how it was transformed through lemmatization.

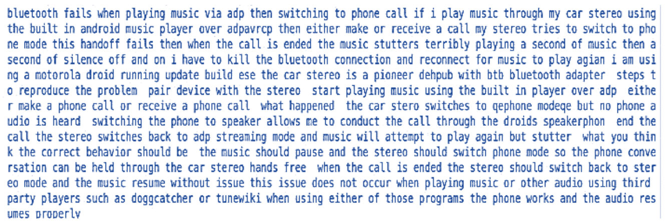
The context of this passage appears to be focused on the pre-processing of text data for natural language processing. The author describes the steps taken to prepare the test case descriptions for further analysis, specifically the application of stemming and lemmatizing to the text. The goal is to transform the text into a format that can be more easily processed and analysed by a machine learning algorithm.



```
['bluetooth', 'fails', 'when', 'playing', 'music', 'via', 'adp', 'then', 'switching', 'to', 'phone', 'call', 'if',
'i', 'play', 'music', 'through', 'my', 'car', 'stereo', 'using', 'the', 'built', 'in', 'android', 'music', 'player'
, 'over', 'adpavrcp', 'then', 'either', 'make', 'or', 'receive', 'a', 'call', 'my', 'stereo', 'tries', 'to', 'switc
h', 'to', 'phone', 'mode', 'this', 'handoff', 'fails', 'then', 'when', 'the', 'call', 'is', 'ended', 'the', 'music'
, 'stutters', 'terribly', 'playing', 'a', 'second', 'of', 'music', 'then', 'a', 'second', 'of', 'silence', 'off', '
and', 'on', 'i', 'have', 'to', 'kill', 'the', 'bluetooth', 'connection', 'and', 'reconnect', 'for', 'music', 'to',
'play', 'agian', 'i', 'am', 'using', 'a', 'motorola', 'droid', 'running', 'update', 'build', 'ese', 'the', 'car', '
stereo', 'is', 'a', 'pioneer', 'dehpub', 'with', 'btb', 'bluetooth', 'adapter', 'steps', 'to', 'reproduce', 'the',
'problem', 'pair', 'device', 'with', 'the', 'stereo', 'start', 'playing', 'music', 'using', 'the', 'built', 'in', '
player', 'over', 'adp', 'either', 'make', 'a', 'phone', 'call', 'or', 'receive', 'a', 'phone', 'call', 'what', 'hap
pened', 'the', 'car', 'stero', 'switches', 'to', 'qephone', 'modeqe', 'but', 'no', 'phone', 'audio', 'is', 'heard',
'switching', 'the', 'phone', 'to', 'speaker', 'allows', 'me', 'to', 'conduct', 'the', 'call', 'through', 'the', 'dr
oids', 'speakerphon', 'end', 'the', 'call', 'the', 'stereo', 'switches', 'back', 'to', 'adp', 'streaming', 'mode',
'and', 'music', 'will', 'attempt', 'to', 'play', 'again', 'but', 'stutter', 'what', 'you', 'think', 'the', 'correct
', 'behavior', 'should', 'be', 'the', 'music', 'should', 'pause', 'and', 'the', 'stereo', 'should', 'switch', 'phon
e', 'mode', 'so', 'the', 'phone', 'conversation', 'can', 'be', 'held', 'through', 'the', 'car', 'stereo', 'hands',
'free', 'when', 'the', 'call', 'is', 'ended', 'the', 'stereo', 'should', 'switch', 'back', 'to', 'stereo', 'mode',
```

Fig. 4 Text case description

Figure 4 displays a test case description from their data set. The description has not yet been fully lemmatized, but it has already undergone data transformation and purification, making it clearer to understand. The next image shows how the same description would appear after undergoing lemmatization. However, before the lemmatization process can occur, the words in the description must first be tokenized, meaning they must be separated into individual units or tokens.

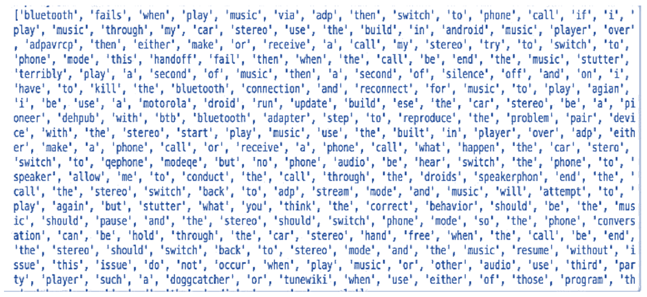


```
bluetooth fails when playing music via adp then switching to phone call if i play music through my car stereo using
the built in android music player over adpavrcp then either make or receive a call my stereo tries to switch to pho
ne mode this handoff fails then when the call is ended the music stutters terribly playing a second of music then a
second of silence off and on i have to kill the bluetooth connection and reconnect for music to play agian i am usi
ng a motorola droid running update build ese the car stereo is a pioneer dehpub with btb bluetooth adapter  steps t
o reproduce the problem  pair device with the stereo  start playing music using the built in player over adp  eithe
r make a phone call or receive a phone call  what happened  the car stero switches to qephone modeqe but no phone a
udio is heard  switching the phone to speaker allows me to conduct the call through the droids speakerphon  end the
call the stereo switches back to adp streaming mode and music will attempt to play again but stutter  what you thin
k the correct behavior should be  the music should pause and the stereo should switch phone mode so the phone conve
rsation can be held through the car stereo hands free  when the call is ended the stereo should switch back to ster
eo mode and the music resume without issue this issue does not occur when playing music or other audio using third
party players such as doggcatcher or tunewiki when using either of those programs the phone works and the audio res
umes properly
```

Fig. 5 Text case description tokenized

In Figure 5, we can see a test case description that has been tokenized, meaning it has been broken down into individual words. The purpose of tokenization is to prepare the text for further processing, such as lemmatization or stemming. In this example, it can be noted that the words in the test-case description are not yet in their root form, as is the case with the word 'playing'. The next step is to perform either lemmatization or stemming, which will reduce the inflected words to their base form.

Figure 6 depicts the test-case description that has been done to be lemmatized. Every word has been explicitly guided by its fundamental root. For example, the word 'play' used to be 'playing' before we lemmatized it. However, after applying the Corpus dictionary's WordNet Lemmatized and Snowball Steamer, each word can now be recognized by its fundamental form.



```
['bluetooth', 'fails', 'when', 'play', 'music', 'via', 'adp', 'then', 'switch', 'to', 'phone', 'call', 'if', 'i', '
play', 'music', 'through', 'my', 'car', 'stereo', 'use', 'the', 'build', 'in', 'android', 'music', 'player', 'over'
, 'adpavrcp', 'then', 'either', 'make', 'or', 'receive', 'a', 'call', 'my', 'stereo', 'try', 'to', 'switch', 'to',
'phone', 'mode', 'this', 'handoff', 'fail', 'then', 'when', 'the', 'call', 'be', 'end', 'the', 'music', 'stutter',
'terribly', 'play', 'a', 'second', 'of', 'music', 'then', 'a', 'second', 'of', 'silence', 'off', 'and', 'on', 'i',
'have', 'to', 'kill', 'the', 'bluetooth', 'connection', 'and', 'reconnect', 'for', 'music', 'to', 'play', 'agian',
'i', 'be', 'use', 'a', 'motorola', 'droid', 'run', 'update', 'build', 'ese', 'the', 'car', 'stereo', 'be', 'a', 'pi
oneer', 'dehpub', 'with', 'btb', 'bluetooth', 'adapter', 'step', 'to', 'reproduce', 'the', 'problem', 'pair', 'devi
ce', 'with', 'the', 'stereo', 'start', 'play', 'music', 'use', 'the', 'built', 'in', 'player', 'over', 'adp', 'eith
er', 'make', 'a', 'phone', 'call', 'or', 'receive', 'a', 'phone', 'call', 'what', 'happen', 'the', 'car', 'stero',
'switch', 'to', 'qephone', 'modeqe', 'but', 'no', 'phone', 'audio', 'be', 'hear', 'switch', 'the', 'phone', 'to', '
speaker', 'allow', 'me', 'to', 'conduct', 'the', 'call', 'through', 'the', 'droids', 'speakerphon', 'end', 'the', '
call', 'the', 'stereo', 'switch', 'back', 'to', 'adp', 'stream', 'mode', 'and', 'music', 'will', 'attempt', 'to', '
play', 'again', 'but', 'stutter', 'what', 'you', 'think', 'the', 'correct', 'behavior', 'should', 'be', 'the', 'mus
ic', 'should', 'pause', 'and', 'the', 'stereo', 'should', 'switch', 'phone', 'mode', 'so', 'the', 'phone', 'convers
ation', 'can', 'be', 'hold', 'through', 'the', 'car', 'stereo', 'hand', 'free', 'when', 'the', 'call', 'be', 'end',
'the', 'stereo', 'should', 'switch', 'back', 'to', 'stereo', 'mode', 'and', 'the', 'music', 'resume', 'without', 'i
ssue', 'this', 'issue', 'do', 'not', 'occur', 'when', 'play', 'music', 'or', 'other', 'audio', 'use', 'third', 'par
ty', 'player', 'such', 'a', 'doggcatcher', 'or', 'tunewiki', 'when', 'use', 'either', 'of', 'those', 'program', 'th
```

Fig. 6 Text-case-description lemmatized

3.2. *Training all Models with Best-Parameters*

In this proposed model, multiple machine learning models, namely the K-nearest neighbor (KNN) classifier, logistic regression, decision tree, and multilayer perceptron (MLP), were employed for the automatic directory categorization of test cases based on their descriptions. The feature representation used was the Bag-of-Words (BoW) approach, applied to all the classifiers. The research's primary objective was to compare these models' performance in accurately categorizing test cases and reducing manual effort. For the KNN model, the study utilized the optimal parameters obtained from the BoW or TF-IDF vector representations. The scikit-learn library in Python was employed to develop the model. In the BoW approach, the Minkowski metric was employed to calculate the distance between a vector and other vectors in the training dataset. Notably, the study found that the cosine distance was unnecessary in BoW since the document sizes were nearly equal. On the other hand, in the TF-IDF approach, the cosine metric was used to compare the distance of a vector to others in the training dataset, as size normalization is required when computing the cosine distance.

3.3. *Vectorizing the Test Dataset*

In the context of machine learning, the statement implies that during the testing phase of the algorithm, the correctness of words is not important to examine because our models only learned words that have appeared during the training dataset. Therefore, during training, models were fitted with words that did not occur in the test dataset, and the word count vector was based on the corpus dictionary. As a result, the testing dataset does not require examining the correctness of words since the model has already learned them from the training dataset.

3.4. *Predict and Measurement*

In order to confirm the accuracy of our classification, we need to review the results of our model predictions and verify the data. Our statistical classification method is K-nearest neighbour, Logistic regression, Decision Tree and MLP classifier, which identifies the most commonly occurring categories as predicted labels. There may be instances when the classifier is unable to accurately assign labels, resulting in unexpected outputs that do not affect the model's overall performance. The results section will provide an analysis of the model's performance.

4. Results and Discussions

The next section will present the results and discussions of the proposed models, including the accuracy of each model measured by the various metrics we used. We found that Knn-BOW and MLP had the highest accuracy compared to Logistic Regression and Decision Trees. Here are the specific details for each model.

4.1. Dataset Description

The data utilized in this research was obtained from two Android smartphone manufacturers, who requested that their identities remain confidential. The dataset contains a large number of test cases, but for this study, six test cases were selected from six different teams within the organization. The ALMS (application lifecycle management system) was selected by the previous researchers as it handled these test scenarios. The table below lists the domains that are represented in the data.

Table 1. Dataset description

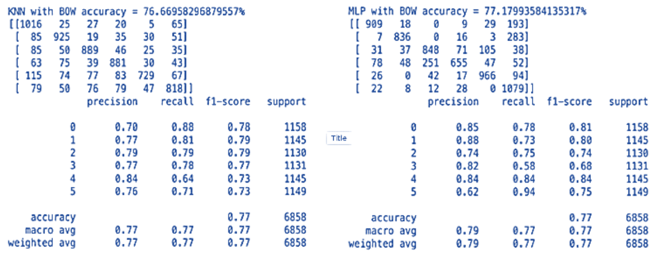
Team ID	Test Domain	Test Cases
0	Multimedia	2286
1	Multimedia	2286
2	Android OS & Linux Kernel	2286
3	Android OS & Linux Kernel	2286
4	Cellular & Connectivity	2286
5	Cellular & Connectivity	2286
	Total Test Cases	13716

4.2. Result and Discussions

Several models were proposed to classify test cases based on categories, including a K nearest neighbor classifier, logistic regression, DecisionTree, and MLP classifier. These models use different metrics to calculate the distance between the real vectors by adding their absolute differences. In this study, the Minkowski metric was utilized as the metric for the K nearest neighbor classifier.

After performing data cleaning, lemmatizing, and stemming, the data was applied to a K-nearest neighbor model with bag-of-words. Different metrics were tested for this model, including Minkowski and cosine. It was discovered that Minkowski was a better metric option, outperforming cosine. The output of this process is presented below.

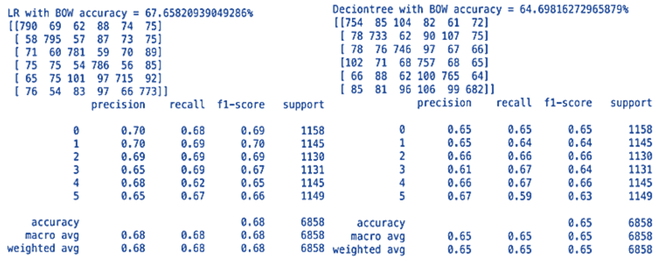
Figure 6 indicate that using lemmatization is a better option in terms of accuracy when compared to stemming, particularly when using Minkowski as a metric. However, it should be noted that the F1 results for both models are similar, suggesting that they are comparable in performance. Furthermore, the following figure will show the results obtained after applying Logistic Regression and DecisionTree models.



```
KNN with BOW accuracy = 76.66958296879557%
[[1016   25   27   20    5   65]
 [  85  925   19   35   30   51]
 [  85   50  889   46   25   35]
 [  63   75   39  881   30   43]
 [ 115   74   77   83  729   67]
 [  79   50   76   79   47  818]]
              precision    recall  f1-score   support

           0       0.70      0.88      0.78      1158
           1       0.77      0.81      0.79      1145
           2       0.79      0.79      0.79      1130
           3       0.77      0.78      0.77      1131
           4       0.84      0.64      0.73      1145
           5       0.76      0.71      0.73      1149

    accuracy                           0.77      6858
   macro avg       0.77      0.77      0.77      6858
weighted avg       0.77      0.77      0.77      6858
```

```
MLP with BOW accuracy = 77.17993584135317%
[[ 909   18    0    9   29  193]
 [   7  836    0   16    3  283]
 [  31   37  848   71  105   38]
 [  78   48  251  655   47   52]
 [  26    0   42   17  966   94]
 [  22    8   12   28    0 1079]]
              precision    recall  f1-score   support

           0       0.85      0.78      0.81      1158
           1       0.88      0.73      0.80      1145
           2       0.74      0.75      0.74      1130
           3       0.82      0.58      0.68      1131
           4       0.84      0.84      0.84      1145
           5       0.62      0.94      0.75      1149

    accuracy                           0.77      6858
   macro avg       0.79      0.77      0.77      6858
weighted avg       0.79      0.77      0.77      6858
```

Fig. 7 KNN & MLP accuracy



```
LR with BOW accuracy = 67.65820939049286%
[[790  69  62  88  74  75]
 [ 58 795  57  87  73  75]
 [ 71  60 781  59  70  89]
 [ 75  75  54 786  56  85]
 [ 65  75 101  97 715  92]
 [ 76  54  83  97  66 773]]
              precision    recall  f1-score   support

           0       0.70      0.68      0.69      1158
           1       0.70      0.69      0.70      1145
           2       0.69      0.69      0.69      1130
           3       0.65      0.69      0.67      1131
           4       0.68      0.62      0.65      1145
           5       0.65      0.67      0.66      1149

    accuracy                           0.68      6858
   macro avg       0.68      0.68      0.68      6858
weighted avg       0.68      0.68      0.68      6858
```

```
Deciontree with BOW accuracy = 64.69816272965879%
[[754  85 104  82  61  72]
 [ 78 733  62  90 107  75]
 [ 78  76 746  97  67  66]
 [102  71  68 757  68  65]
 [ 66  88  62 100 765  64]
 [ 85  81  96 106  99 682]]
              precision    recall  f1-score   support

           0       0.65      0.65      0.65      1158
           1       0.65      0.64      0.64      1145
           2       0.66      0.66      0.66      1130
           3       0.61      0.67      0.64      1131
           4       0.66      0.67      0.66      1145
           5       0.67      0.59      0.63      1149

    accuracy                           0.65      6858
   macro avg       0.65      0.65      0.65      6858
weighted avg       0.65      0.65      0.65      6858
```

Fig. 8 Logistic regression & Decision tree accuracy

Figures 7 and 8 can be seen that both models have achieved low accuracy compared to the KNN and MLP models; however, Logistic regression with alpha (0.01) has scored 68 percentage accuracy compared with DecisionTree, which has achieved 65 percentage. The panelty of logistic regression was L2, and the best optimal of alpha was 0.01.

Overall, we have implemented four models: logistic regression, Decision tree, MLP classifier, and K-nearest neighbor classifier applied with Bag-of-words; the upcoming table will sum up the accuracy and F1 score that each model has achieved.

Table 2. Results

Model	Word root	Accuracy (%)	F1-Scores
KNN-BoW	Steaming &Lem	76.66	77
MLP-BoW	Steaming &Lem	77	77
Logistic Regression-BoW	Steaming &Lem	67	68
Decision Tree-BoW	Steaming &Lem	65	65

The MLP model and Knn-Bow yielded the highest accuracy when compared to the Logistic Regression and Decision Tree models. Among the models tested, using

Minkowski as the distance metric along with lemmatizing resulted in an accuracy score of 76.66 and an F1 score of 77, making it a good option. However, a lower accuracy was obtained when applying Knn-Bow with the distance metric as cosine. Furthermore, it should be noted that the Decision Tree model took a long time to run due to its computation requirements, but it has the potential to improve its performance if looped.

4.3. Comparative Analysis

Our research found that the use of Word Count (WC) was more effective than Name-LDA or Name-WC in classifying performance outcomes. This led to high F1 scores for modules B, C, E, and F. To compare our findings with a previous study, we referred to a research paper [1], which also used WC and LDA to achieve scores ranging from 0.3 to 0.88.

However, our model outperformed the WC performance in the previous study, achieving an accuracy of 76.66% and an F1 score of 77%. This suggests that text analysis was successful in accurately categorizing unique modules.

5. Conclusion

To automate the directory categorization process, this study recommends utilizing machine learning, which could significantly aid in test case classification, a task that typically consumes a lot of time for testers to ensure that apps or features are performing correctly.

The approach that was utilized in this study performed the classification and utilized one distinct vector representation: bag-of-words. Four machine learning algorithms have applied this model, and a bag of words was used to represent a vectorizer of the test case description. As it required overfitting and underfitting to analyse, it's notable that test case descriptions needed to be categorized as the label contributed multiclass. KNN and MLP have scored good results and outperformed Logistic regression and decision trees.

Additionally, this study investigated how much an Android smartphone vendor would need to invest in deploying such a system and how much labour developers would need to put in to create a functional system.

References

[1] Junji Shimagaki et al., "Automatic Topic Classification of Test Cases using Text Mining at an Android Smartphone Vendor," *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 1-10, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[2] Lijun Shan, and Hong Zhu, "Generating Structurally Complex Test Cases by Data Mutation: A Case Study of Testing an Automated Modelling Tool," *The Computer Journal*, vol. 52, no. 5, pp. 571-588, 2009. [CrossRef] [Google Scholar] [Publisher Link]

[3] Stephen H. Edwards, "Using Software Testing to Move Students from Trial-and-Error to Reflection-in-Action," *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, pp. 26-30, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[4] Xiao-Yi Zhang, Zheng Zheng, and Kai-Yuan Cai, "Exploring the Usefulness of Unlabelled Test Cases in Software Fault Localization," *Journal of Systems and Software,* vol. 136, pp. 278-290, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[5] Saswat Anand et al., "An Orchestrated Survey of Methodologies for Automated Software Test Case Generation," *Journal of Systems and Software,* vol. 86, no. 8, pp. 1978-2001, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[6] P.A. Stocks, and D.A. Carrington, "Test Templates: A Specification-Based Testing Framework," *Proceedings of 1993 15th International Conference on Software Engineering, IEEE,* pp. 405-414, 1993. [CrossRef] [Google Scholar] [Publisher Link]

[7] P. Ammann, and J. Offutt, "Using Formal Methods to Derive Test Frames in Category-Partition Testing," *Proceedings of COMPASS'94-1994 IEEE 9th Annual Conference on Computer Assurance,* pp. 69-79, 1994. [CrossRef] [Google Scholar] [Publisher Link]

[8] A. Hartman, and K. Nagin., "The AGEDIS Tools for Model-Based Testing," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 4, pp. 129-132, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[9] Dathar A. Hasan et al., "The Impact of Test Case Generation Methods on the Software Performance: A Review," *International Journal of Science and Business,* vol. 5, no. 6, pp. 33-44, 2021. [Google Scholar] [Publisher Link]

[10] Yanjie Zhao et al., "Towards Automatically Repairing Compatibility Issues in Published Android Apps," *Proceedings of the 44th International Conference on Software Engineering*, pp. 2142-2153, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[11] He Ye et al., "Automated Classification of Overfitting Patches with Statically Extracted Code Features," *IEEE Transactions on Software Engineering,* vol. 48, no. 8, pp. 2920-2938, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Muhammad Khatibsyarbini et al., "Test Case Prioritization Approaches in Regression Testing: A Systematic Literature Review," *Information and Software Technology,* vol. 93, pp. 74-93, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[13] Karina Curcio et al., "Requirements Engineering: A Systematic Mapping Study in Agile Software Development," *Journal of Systems and Software,* vol. 139, pp. 32-50, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[14] Sillitti et al., "Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data," *Proceedings 29th Euromicro Conference,* vol. 3, pp. 336-342, 2003. [CrossRef] [Google Scholar] [Publisher Link]

[15] Raymond P. L. Buse, and Thomas Zimmermann, "Information Needs for Software Development Analytics," *34th International Conference on Software Engineering, IEEE,* pp. 987-996, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[16] Dan Han et al., “Understanding Android Fragmentation with Topic Analysis of Vendor-Specific Bugs,” *19th Working Conference on Reverse Engineering, IEEE,* pp. 83-92, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[17] Christof Ebert, and Hassan Soubra, “Functional Size Estimation Technologies for Software Maintenance,” *IEEE Software*, vol. 31, no. 6, pp. 24-29, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[18] Abram Hindle et al., “Relating Requirements to Implementation via Topic Analysis: Do Topics Extracted from Requirements Make Sense to Managers and Developers?,” *28th IEEE International Conference on Software Maintenance*, pp. 243-252, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[19] Annibale Panichella et al., “How to Effectively Use Topic Models for Software Engineering Tasks? An Approach based on Genetic Algorithms,” *35th International Conference on Software Engineering, IEEE*, pp. 522-531, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[20] Ed Keenan et al., “Tracelab: An Experimental Workbench for Equipping Researchers to Innovate, Synthesize, and Comparatively Evaluate Traceability Solutions,” *34th International Conference on Software Engineering, IEEE*, pp. 1375-1378, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Thomas H. Davenport, Jeanne G. Harris, and Robert Morison, *Analytics at Work: Smarter Decisions, Better Results,* Harvard Business Press, 2010. [Google Scholar] [Publisher Link]

[22] Stuart McIlroy et al., “Analyzing and Automatically Labelling the Types of User Issues that are Raised in Mobile App Reviews,” *Empirical Software Engineering*, vol. 21, no. 3, pp. 1067-1106, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[23] Haibing Li et al., "Financial Fraud Detection: Multi-Objective Genetic Programming with Grammars and Statistical Selection Learning," *SSRG International Journal of Computer Science and Engineering*, vol. 7, no. 2, pp. 1-18, 2020. [CrossRef] [Publisher Link]

[24] P.M. Johnson et al., “Improving Software Development Management through Software Project Telemetry,” *IEEE Software*, vol. 22, no. 4, pp. 76-85, 2005. [CrossRef] [Google Scholar] [Publisher Link]

[25] Dongmei Zhang et al., “ Software Analytics as a Learning Case in Practice: Approaches and Experiences,” *Proceedings of the International Workshop on Machine Learning Technologies in Software Engineering*, pp. 55-58, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[26] Akanksha Pandey, and L.S. Maurya, "Career Prediction Classifiers based on Academic Performance and Skills using Machine Learning," *SSRG International Journal of Computer Science and Engineering*, vol. 9, no. 3, pp. 5-20, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[27] Barbara A. Kitchenham, and Shari L. Pfleeger, “Personal Opinion Surveys,” *Guide to Advanced Empirical Software Engineering, Springer*, pp. 63-92, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[28] Anh Tuan Nguyen et al., “Duplicate Bug Report Detection with a Combination of Information Retrieval and Topic Modeling,” *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering,* pp. 70-79, 2012. [CrossRef] [Google Scholar] [Publisher Link]