International Journal of System Dynamics Applications
Volume 11 • Issue 2 • July-December 2022

Models for Efficient Utilization of Resources for Upgrading Android Mobile Technology

Abha Jain, Shaheed Rajguru College of Applied Sciences for Women, India
Ankita Bansal, Netaji Subhas University of Technology, India

ABSTRACT

The need of the customers to be connected to the network at all times has led to the evolution of mobile technology. Operating systems play a vitol role when we talk of technology. Nowadays, Android is one of the popularly used operating system in mobile phones. The authors have analysed three stable versions of Android, 6.0, 7.0, and 8.0. Incorporating a change in the version after it is released requires a lot of rework and thus huge amount of costs are incurred. In this paper, the aim is to reduce this rework by identifying certain parts of a version during early phase of development which need careful attention. Machine learning prediction models are developed to identify the parts which are more prone to changes. The accuracy of such models should be high as the developers heavily rely on them. The high dimensionality of the dataset may hamper the accuracy of the models. Thus, the authors explore four dimensionality reduction techniques, which are unexplored in the field of network and communication. The results concluded that the accuracy improves after reducing the features.

KEYWORDS

Android Operating System, Dimensionality Reduction, Feature Selection, Machine Learning, Mobile Technology, Model Prediction, Software Changes

INTRODUCTION

In today's rapidly growing industry, it is very essential to build an effective and reliable connectivity to establish an efficient communication among the clients and employees of any business organization. Communication involves sharing of critical information between the user and the organization where security plays a key role in terms of privacy (Auxilia et al., 2020). The customers want to be connected and be able to communicate with any business organization at any time and from anywhere. The demand of being connected 24/7 is only possible due to the evolution of mobile and Internet technology. The availability of newer operating systems in the competing market plays a significant role in improving the mobile technology. Among many other characteristics and applications; one of the distinguishing characteristics offered by mobile operating systems is that the users can get connected to the internet using the wireless service provider of their smart phones which is cost effective as the mobile system is completely wireless leading to benefits like saving of money and space as compared to wired sensor network (Elfouly et al., 2017). Various types of operating systems are available in the market, the most popular being Android. Android has seen various versions starting from version 1.0, then1.5, 1.6, 2.1, 2.2, 2.3, 3.0and so on. Each version has the improvements over the previous one and thus, it is always advisable to go with the latest version. The upgradation of a

DOI: 10.4018/IJSDA.20220701.oa2

version to a higher version may be due to a number of reasons, such as identification of some bug in the previous version, change in the customers' demands, change in technology in the market, etc. We can also see that change is the need of the hour and we as humans must adapt to those changes. However, seeing from the end of developer's site, we should understand that incorporating a change due to any of the above listed reason resulting in a new version is not at all an easy task. It comes with lots of difficulty and requires huge amount of resources in terms of time, money and manpower. To elaborate on this, let us understand that the development of a software goes through certain stages before it can be deployed. Incorporating a change in any part of the software (due to any of the reasons stated above) may need widespread changes in different parts of the software and thus lots of rework is required (Sharma et al. 2014). Cost and effort of this rework significantly increases with the stages in software development lifecycle (Boehm and Basili 2001).

The authors in this paper aim to work in the direction of reducing the rework and thus, saving of cost and other resources. For this, the three stable versions of Android, viz Android 6.0 (Marshmallow), Android 7.0 (Nougat) and Android 8.0 (Oreo) are analysed. Each version is fairly large in size consisting of a large number of classes. Due to the availability of limited resources, the developers fail to pay equal attention to all the classes, leading to poor quality software. Thus, the main idea revolves around identification of those classes which are more vulnerable to changes in the next software update. The authors have constructed the prediction models which can be used by the developers/designers in the early phases of software development to identify the classes which need focussed attention. Next, we discuss about the correctness or accuracy of the constructed prediction model. Since the developers are relying on these models for identifying the classes, it is very important that the model should be as accurate as possible. The high dimensionality of the dataset is one of the hindrance which may hamper the accuracy of the models. In addition to this, the high index of features (high dimensionality) makes the computation of data an expensive and tedious task (Rattanawadee and Srivihok, 2015). Dimensionality Reduction refers to the process of reducing the number of dimensions of a given data set. This leads to a reduction in the number of variables and utilization of a group of prime variables.

In this paper, the authors have used feature selection algorithms for selecting a subset of relevant features from a given set of features such that they would yield the most optimum results while building an effective and efficient predictive model (Padmaja and Vishnuvardhan, 2016).The literature shows the wide use of traditional statistical method known as regression analysis to extract the useful features. In this study, univariate Logistic Regression (LR) is used to find the effect of each independent variable with the dependent variable. Thereafter, multivariate LR is also used for constructing the model. In addition to the regression analysis which is a statistical approach, there are much newer and popularly used feature selection techniques broadly classified under the three approaches: filter approach, wrapper approach and embedded approach (Bachu and Anuradha 2019). While filter approach focuses on the data instead of the algorithm used for mining it and gathering the relevant information by analysing the nature of the data, the wrapper approach focuses on the applicability or pertinence of each feature and the optimality of the solution thus obtained (Bolón-Canedo et al. 2014). The embedded approach, on the other hand, is more focused and aims towards optimizing the model for a particular training algorithm. In this research, the authors have analysed four sequential search techniques viz. Sequential Forward Selection (SFS), Sequential Backward Selection (SBS), Sequential Forward Floating Selection (SFFS) and Sequential Backward Floating Selection (SBFS) that are employed by wrapper methods for feature selection. The authors found that the usage of these sequential search techniques in different domains such as gene selection, big data classification, pattern recognition and image recognition (Pudil et al. 1993 and Peralta et al. 2015) have produced promising results. However, these techniques are unexplored in the field of networks and communication. Thus, this motivated the authors to explore these search techniques to reduce the features of popular mobile technology, Android.

To determine the efficiency of these sequential search techniques, the prediction models are constructed using three popularly used machine learning classifiers viz. K-Nearest Neighbor (KNN),

Decision Tree (DT) and Random Forest (RF). In other words, after selecting useful features with each of the four sequential search techniques, the dataset is trained and tested using KNN, DT and RF. The authors also observed the relation between the number of features selected and the performance of the searching technique.

The results showed that the maximum accuracy was observed in the mid region where the number of features selected was neither too high nor too low. It was observed that the accuracy increases as the number of features increases from 1 upto the mid - range. There is a slight fluctuation in the accuracy for the mid values and it eventually decreases as the number of features reaches the maximum value. In addition to this, when comparing and analysing the accuracy of the models obtained using multivariate LR and the machine learning models, the authors observed that all the machine learning models outperformed the statistical model.

In this paper, the authors aim to find answers to the following Research Questions (RQs):

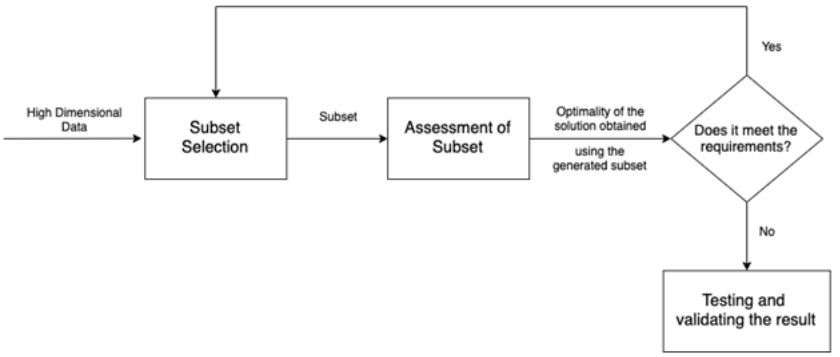
RQ1: How did the sequential search techniques used to reduce the number of features perform in the field of communication, i.e. on the datasets of Android operating system?
RQ2: How did the performance of machine learning classifiers improve when the number of features are reduced?
RQ3: What trend can be observed in the values of accuracy at different number of features selected? Is the highest accuracy achieved at the lowest possible value of number of features?
RQ4: How did multivariate LR model compare with the machine learning models in predicting the change prone classes of Android operating system?

This paper is organized as follows. Following this section, the related work is given which discusses the work done related to feature selection methods. Next section explains the concept and process of feature selection used to select the optimal set of features. Further, it highlights the different kinds of feature selection strategies and various searching techniques which can be used to implement it. After this, the background of the research which includes the details about the empirical data collection and the variables used are explained It also summarizes the empirical data used to validate the results. Next section explains the framework and design of the research. This is followed by the discussion of results. Application of the work and threats to validity are discussed thereafter. Finally, the work is concluded providing important insights.

RELATED WORK

Research shows that the majority of existing work on feature selection does not focus on change prediction. The authors chose to implement various wrapper selection methods for feature reduction to predict change prone classes of open source software as a large number of studies have found that wrapper methods perform best (Kohavi& John, 1997), especially while dealing with lower dimensionality. Wrapper methods like genetic search and sequential forward selection carry out a search over the set of all viable subsets of features, continually calling the induction algorithm as a procedure to assess multiple feature subsets. Work based on feature selection carried out in the past for textual problems turned out to be of great assistance in providing motivation and guidance for this study, which highlights a more substantial variety of metrics. For instance, the authors (Yang & Pedersen, 1997) have examined five feature selection metrics on the common Reuters dataset and OHSUMED. In the papers (Guyon & Elisseeff, 2003 and Liu & Yu, 2005),a thorough study for feature (or variable) selection have been done in the domain of machine learning and statistics. In the paper by Saeys et al. (2007), authors have worked in the domain of bioinformatics by applying feature selection techniques. The work was done in the field of wrapper and filter methods being studied. In the paper by Ma & Huang (2008), the selection of features is done on the basis of sparse regularization. The authors Yang et al. (2013) have proposed an ensemble-based wrapper approach for feature selection

Figure 1. Process of feature selection



from data with highly imbalanced class distribution by creating multiple balanced datasets from the original imbalanced dataset using sampling and then evaluating feature subsets using an ensemble of base classifiers each trained on a balanced dataset. The authors Zhou et al. (2008) presented a Genetic Algorithm (GA) based wrapper method which solves optimization problems using the methods of evolution and is based on survival of the fittest. Their work is based on the classification of hyper spectral data using Support Vector Machine (SVM). The authors (Maldonado and Weber, 2009) had introduced a novel wrapper algorithm for feature selection using SVM with kernel functions. Similar work was also done by the authors Rodriguez-Galiano et al. (2018) and Hu et al. (2015).

FEATURE SELECTION

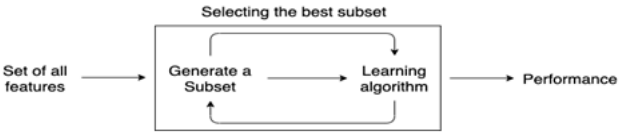
Feature selection is a technique of dimensionality reduction which being very popular, has been a matter of research in the recent times. It aims at reducing the number of features and providing a subset of the most optimal ones. The process of feature selection process has been highlighted in figure 1. It can be summarized as constituting of the following major steps; creating a subset, assessing the generated subset, checking for the stopping condition and finally testing and validating the result. Following are the advantages of feature selection for dimensionality reduction (Ladha and Deepa, 2011):-

1. Removes the un-important and useless data from the data set.
2. Makes the algorithm more efficient, thereby reducing the time taken by the algorithm to infer results.
3. Overcomes the curse of dimensionality. Curse of dimensionality refers to the difficulties (mainly time and resource constraints) faced in analyzing high dimensional data.
4. Improves the accuracy and performance of the models.

There are three main feature selection methods, namely; filter methods, wrapper methods and embedded methods (Ladha and Deepa 2011). Filter methods are usually implemented in the preconditioning stage. In these methods, the subset of features is picked on the grounds of the correlation value of the features with the output. There are four types of filter methods viz. Linear Discriminant Analysis, Pearson's Correlation, Chi-square and Analysis of Variance. Figure 2 highlights the process followed by filter methods to obtain a feature subset. Wrapper methods train a prediction model by using different subsets of features (Ang et. al, 2015). Based on the performance of the machine learning algorithm, we select the best subset of features from the given dataset. We may choose to include or exclude a feature based on the basis of the performance of a subset. Wrapper

Figure 2. Process followed by filter methods in order to obtain feature subset.

Figure 3. Process followed by wrapper methods to obtain the feature subset.



methods can be classified into two categories, namely; forward selection and backward elimination. In Forward Selection, we start with a void set of features, and keep adding new features in the set with each iteration. On the other hand, in backward elimination, we start a full set of features and keep on eliminating the most useless feature in the entire set with each iteration. Figure 3 shows the process followed by wrapper methods to obtain a feature subset. Embedded methods amalgamate the good attributes of wrapper and filter methods. However, they are complex in implementation. The differences between filter and wrapper methods have been summarized in table 1.Selecting the right feature selection method is a crucial step in determining the efficiency of the algorithm used along with the optimal subset of features to be used. Feature selection speeds up the training of the algorithm used. It also lowers the dimensionality of the dataset, thereby reducing the computational cost and time.

Wrapper methods can be implemented by using various search techniques (Kumari et. al, 2012) which can be categorized as exponential search, random search and sequential search. Exponential search, which may be referred to as total search or complete search, is a comprehensive approach. Although the results that it provides are superlative, this approach takes exponential time to execute. Thus, it cannot be used for large datasets or medium size datasets. An example of the same is Brute-Force search which looks for the solution in each subset. On the other hand, random search, as the name suggests, commences by picking features in a random fashion and then progresses with either of the following search techniques. The first one is a two-way searching approach, like simulated annealing and random hill-climbing. The second one uses techniques which have no uniform variation e.g., Genetic Algorithm (GA) and Tabu search. Finally, sequential search is a greedy optimization technique that finds the local optimum solution. Sequential search can be categorized into four types, namely; Sequential Forward Selection (SFS), Sequential Backward Selection (SBS), Sequential Forward Floating Selection (SFFS) and Sequential Backward Floating Selection (SBFS) which have been explained later in the paper.

Table 1. Difference between wrapper and filter methods

PARAMETER	FILTER METHODS	WRAPPER METHODS
Criteria of selection	Correlation between output variables and features	Utility of a subset of features
Time Complexity	These are faster than wrapper methods because they do not require training of mode with each subset.	These are slower than filter methods.
Evaluation Technique	Statistical methods	Cross Validation
Correctness	Might not be able to find the best solution available	Best Subset of features

Table 2. Comparison between different searching techniques

PARAMETER	EXPONENTIAL	RANDOMISED	SEQUENTIAL
Complexity	Exponential	Polynomial	Polynomial
Type of Algorithm	Brute Force	Greedy	Greedy
Implementation	Difficult to implement	Easier than exponential but more difficult than sequential	Easy to implement
Target dataset size	Small datasets	Small/Medium, but might produce incorrect results	Large datasets

Table 2 illustrates the comparison between different searching techniques used by the wrapper methods. It can be seen from table 2 that the sequential search techniques can be implemented in polynomial time as contrast to exponential search techniques. Although the complexity of randomized search techniques is also polynomial, but it is not suitable for large datasets and it is difficult to be implemented as compared to the sequential search techniques. Since both the datasets used in this paper are large in size, the authors choose to work on sequential search techniques. The authors mainly focus on sequential search techniques (SFS, SBS, SFFS, SBFS) used by wrapper methods and explore their working, optimality and performance. It was observed that the above mentioned techniques have been widely used in different domains such as gene selection, big data classification, pattern recognition and image recognition (Pudil et al. 1993 and Peralta et al. 2015) and have produced promising results. This provides a motivation to the authors to explore these techniques in the field of communication and network.

RESEARCH BACKGROUND

This section focuses on the dataset used highlighting the details of the same. Also, the section elaborates on the independent and dependent variables used in the study.

Empirical Data Collection

For empirical data collection, we have used Android operating system. Android is currently one of the most popular operating system being used in mobile phones and tablets. There are number of versions of Android released in the market till date. The first stable version released was Android 2.3, known by the name of Gingerbread and the latest version is Android 10.0. In this study, we have analyzed the three recent stable versions, Android 6.0, known as Marshmallow, Android 7.0, known as Nougat and Android 8.0, known as Oreo. Since Android is an open source dataset, we downloaded the source code of all these versions from https://source.android.com/source/initializing.html. The details of each version which includes their common name, the total number of classes, the number of classes changed (change - prone classes) and the release date are shown in table 3. The total number of classes is the common classes between the two successive versions. For example, table 3 shows that the total number of classes of Android 6.0 is 10,068. This implies that the common classes between Android 6.0 and Android 7.0 are 10,068.

Independent and Dependent Variables

The dataset consists of 44 independent variables and 1 dependent variable. The independent variables are various Object Oriented (OO) metrics used to determine OO relationships like coupling, cohesion, inheritance etc. The independent variables are provided in table 5. The independent variables are discrete in nature. The dependent variable on the other hand is binary in nature and is used to determine whether the class changes in the next version or not. Its value is 0 for a class if it has not changed

Table 3. Details of the dataset used

Version	Commonly known as	Total Number of Classes	Number of Classes Changed	Release Date
Android 6.0	Marshmallow	10,068	2918	October 5, 2015
Android 7.0	Nougat	11,428	3696	August 22, 2016
Android 8.0	Oreo	-	-	August 21, 2017

in the next version and the value is 1 if it has changed in the next version in terms of number of lines added, deleted and modified. This data consisting of independent and the dependent variable is collected with the help of a tool known as Change Report Generator (CRG) developed by one of the authors (Malhotra et al. 2016).

RESEARCH DESIGN

This section presents the research design used for empirical analysis of change prediction. As depicted in figure 4, the work in this study is conducted in three main phases viz. data filtration using regression analysis and wrapper methods in order to achieve optimal subset of features, thereafter model prediction by applying suitable statistical and machine learning algorithms and finally assessing the performance of the predicted models using suitable performance evaluation measure. Each of the phases has been explained in the subsequent sub-sections along with the methods/techniques followed in that phase.

Data Filtration Using Regression Analysis

In this study, the authors have used a statistical approach known as Logistic Regression (LR) to identify the useful features and for model building. LR is used to predict the dependent variable from a set of independent variables (Hosmer and Lameshow 1989). The authors have used LR as the outcome variable is binary or dichotomous (0 or 1). Both univariate and multivariate regression have been used in this study. Univariate logistic regression is used to find the relationship between the dependent variable and each independent variable. It finds whether there is any significant association between them. Multivariate logistic regression analyses which metrics are useful when they are used in combination. It is used to construct a prediction model used for identifying the change prone classes. To construct the multivariate model, metrics can be fed into the model using two stepwise selection methods, which are forward selection and backward elimination (Hosmer and Lameshow 1989). Forward selection examines the variables that are selected one at a time for entry at each step. The backward elimination method includes all the independent variables in the model and the variables are deleted one at a time from the model until the stopping criteria is fulfilled. However, the results of the model obtained using forward selection were poorer (i.e. the values of $R2$ and log-likelihood statistic defined below were low) than the model obtained from the backward elimination procedure. The authors therefore used backward elimination method in this study.

The general multivariate logistic regression formula is as follows:

Prob (X1, X2, …, Xn) =

where g(x) = B0 + B1*X1 + B2* X2 + … + Bn* Xn

'prob' is the probability of a class being change prone

Xi, (1£ i £ n) are independent variables

Data Filtration Using Wrapper Methods

In this sub-section, we discuss about the working of different wrapper methods which we have applied for the process of feature selection. The reduced feature set that is obtained as a result of

application of these wrapper methods is used for developing the prediction models. According to S´anchez-Maro˜no et al. (2007) and Setiono & Liu (1996), wrapper methods explore all possible combinations of the features and find the one that generates the best results for any given machine learning algorithm. Wrapper methods employ greedy search algorithms. As already mentioned, there are four main types of sequential search techniques (SFS, SBS, SFFS, SBFS) used to implement wrapper methods. Description of each of these search techniques has been presented below. The section also discusses the implementation of these techniques in terms of their pseudo code and how the code was implemented in python. The authors have written the programs in python. The classifiers have been implemented using scikit-learn library. This library has in-built classifiers which can be called using functions. The functions take in various parameters for the classifier to be built. The feature selection algorithm has been made using another python library, namely mlxtend. This is an open source python library which provides machine learning and data science utilities to python's computing stack. They have used the functional sequential feature selector implemented in this library. The function takes in parameters through which we determine if the algorithm is floating or not, and whether the algorithm is forward or not. Furthermore, it takes as an input the classifier which we implemented using the scikit-learn library of python.

In order to understand the working of these techniques, let us assume that there are D features in a dataset but we want only K features.

Sequential Forward Selection (SFS)

According to Marcano-Cedeño et al. (2010), begin with a void set of features and add features to the set one by one, measuring accuracy of the machine learning model at each step. The feature giving the maximum accuracy is retained. The process is repeated K times, in order to store K elements in the set. The final set thus obtained gives the maximum possible accuracy with the desired number of features.

Pseudo code of SFS

Process starts with F1 (a set containing no elements) and the method takes in a number of parameters viz. the classifier to be used, the number of features to select (k), the scoring function and the cross validation. Let Xj be a random variable for feature j and Y be the variable that determines the class label (e.g., healthy vs. unhealthy).
Steps:-

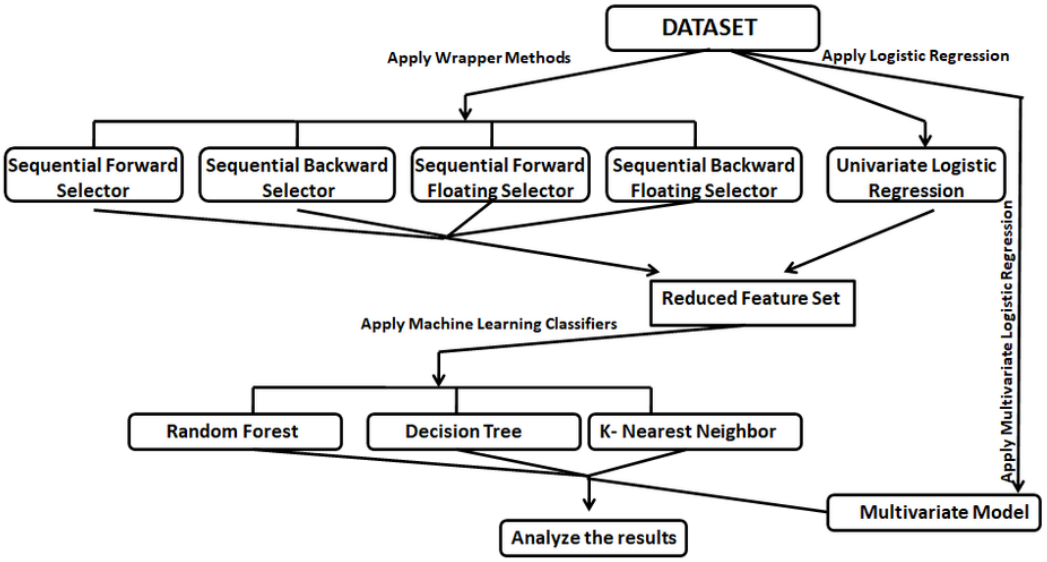
Firstly, a feature X_j is chosen which maximizes the objective function J that takes in the arguments X_j, Y, and F1.

1. The feature Xj that maximizes the objective function is added to F1 and removed from F. This process is repeated until the [REMOVED HYPERLINK FIELD]cardinality of F1 is k.

Sequential Backward Selection (SBS)

Begin with a set containing all the D features. For each of the (D - K) iterations, each feature is removed one by one and the corresponding change in accuracy is noted. The feature with the highest reduction in accuracy is removed. In the end, K features are left which give us the maximum accuracy.

Figure 4. Framework of the study



Pseudo code of SBS

The process starts with F1 (Set of all features), and the method takes in a number of parameters, viz. the classifier to be used, the number of features to select (k), the scoring function and the cross validation. Let Xj be a random variable for feature j and Y be the variable that determines the class label (e.g., healthy vs. unhealthy)
Steps:-

Firstly, feature X_j is chosen which minimizes the objective function J that takes in the arguments X_j, Y, and F1.

1. The feature X_j that minimizes the objective function is removed from F1. This process is repeated until the [REMOVED HYPERLINK FIELD]cardinality of F1 is k.

Sequential Forward Floating Selection (SFFS)

According to Solberg et al (1997), for each step of SFFS, perform the corresponding step of SFS and then find the worst feature in the feature set. Then remove that feature from the feature set if removing it increases the accuracy. Continue removing the worst features as long as the accuracy after removing that feature is greater than the one provided by the SFS set. Then move to the next iteration.

Pseudo code of SFFS

Process starts with F1 (a set containing no elements)
Steps:

Perform SFS.

Find the least significant feature in F1. If it is the feature just added, then keep it and return to step 1. Otherwise, exclude the feature k.

Keep repeating step 2 until we don't have a feature in F1 removing which improves the accuracy.

Sequential Backward Floating Selection (SBFS)

According to Dash et al. (1997) and Wu et al. (2013), for each step of SBFS, perform the corresponding step of SBS and then find the best feature in the feature set. Insert that feature into the feature set if

adding it increases the accuracy. Continue inserting the best features as long as the accuracy after adding that feature is greater than the one provided by the SBS set. Then move to the next iteration.

Pseudo code of SBFS

Steps:-

First perform SBS.

Find the most significant feature in remaining features. If it is the feature just removed, then let it be removed and return to step 1.

Keep repeating step 2 until we don't have a feature in remaining adding which improves the accuracy.

Model Development using Machine Learning Classifiers

Machine learning involves predicting and classifying data and to do so, various machine learning models are used in literature. There are a number of machine learning classifiers available in the literature which are widely used in the classification problems pertaining to diverse fields like software defect prediction and fault severity (Panda, 2019, Hussein et al. 2017), breast cancer classification (Majhi, 2018), health insurance claim prediction (Bhardwaj, 2020) etc. Machine learning classifiers have gained huge popularity in the recent years due to their capability in capturing complex nonlinear relationships among variables. Thus, the authors are motivated to explore the three most popularly used machine learning classifiers viz. K Nearest Neighbour, Random Forest and Decision Tree as their usage is minimal in the field of network for predicting the change prone classes of Android operating system. The overview of these machine learning classifiers is presented in table 4 (Singh et al. 2017, Jain & Vailaya 1996 and Corbane et al. 2009).

Machine learning models have certain parameters (also known as hyperparameters) which can be arbitrarily set by the user before the training process according to a given problem. However, it is challenging to know what values to use for the hyperparameters of a given algorithm on a given dataset. Moreover, there are many hyperparameters associated with each machine learning model and more the hyperparameters we need to tune, the slower the tuning process becomes. Not all model hyperparameters are equally important and some hyperparameters have an outsized effect on the behavior, and in turn, the performance of a machine learning algorithm. Therefore, it is desirable to select a minimum subset of model hyperparameters to search or tune.

In this paper, the authors have used the *Random Search* strategy (Bergstraand and Bengio, 2012) for hyperparameter optimization. In this study, the criteria for selecting the right set of hyperparameters is governed by the following two requirements: (1) minimum execution time and (2) maximum accuracy of the model.

In *Random Search*, a grid of hyperparameters is created which consists of some random values of these hyperparameters. The results of performance would be highly optimistic if training and testing are done the same dataset. Thus, in this study, we have used k-cross validation wherein a single dataset is divided into k parts, out of which one part is used for testing and the remaining other parts are used for training the model (Stone 1974). This process is repeated k number of times, so that each of the k part is used for testing once. In this study, the value of k is taken as 10. In addition to obtaining unbiased results using cross validation, it also has an another important advantage when implementing hyperparameter optimization. Since cross validation allows the dataset to be partitioned into training and testing sets, the authors have avoided using the hyperparameters which worked good on training data but not so good with the test data.

The authors implemented *Random Search* using a utility provided by Scikit-learn (Phython) known as *RandomSearchCV*. Using the scikit-learn *best-estimator attribute*, the authors retrieved the set of hyperparameters which gave the best accuracy of the model in minimum execution time.

Table 4. Description of machine learning techniques along with the values of parameters

Machine learning technique	Description
K Nearest Neighbour (KNN)	KNN is a classifying technique which does not make any assumptions on the underlying data distribution. In KNN, the testing data is classified on the basis of its Euclidean distance from the classes. We take the value of K as 4. The number of parallel workers is set to 10 so as to reduce the total computational time.
Random Forest (RF)	In this method, there are multiple individual decision trees that operate as an ensemble, each of which gives a class prediction and the class with the highest number of votes is returned as the model's prediction. The number of decision trees taken in the random forest is 100. The maximum depth of the tree is set to 2. The seed for generating random numbers is set to 58 which is the best possible value so far. The number of parallel workers is set to 10 so as to reduce the total computational time.
Decision Tree (DT)	It classifies the testing data by forming simple *if-else* statements which are deduced from the features of the training data set. The maximum depth of the tree is set to 2. The seed for generating random numbers is set to 58 which is the best possible value so far. The number of parallel workers is set to 10 so as to reduce the total computational time.

These hyperparameter values is provided in table 4. Once the model is trained, testing is performed using these values of hyperparameters obtained during the training process.

EXPERIMENTAL RESULTS

In this section, we explain the results of regression analysis and the model evaluation results obtained after applying ML classifiers on the features obtained from wrapper methods.

Evaluation of results using regression analysis

In this section, the results of univariate and multivariate LR are discussed (due to space constraint, the results are shown for Android 6.0 only). Similar observations were observed for Android 7.0. Table 5 represents the results of univariate analysis in terms of the coefficient (B) and statistical significance(sig.) for each metric. The parameter "sig" tells whether each of the metric is a significant in predicting the dependent variable (change proneness). If the "sig" value of a metric is below or at the significance threshold of 0.01 or 0.05, then the metric is said to be significant in predicting the change prone classes. In this study, the threshold value is considered as 0.05 (significant values are shown in bold in table 5). The coefficient "B" shows the strength of the independent variable. The higher the value, the higher the impact of the independent variable is. The sign of the coefficient tells whether the impact is positive or negative. Table 5 shows that only the metrics ALB, NOC, CDIM, WMC, NPM and RCC are found to be not significant, whereas all the other metrics are found to be significant predictors of change proneness.

Once the impact of each independent variable on the dependent variable is found, multivariate LR is used to determine the combined effect of independent variables on the dependent variable. All metrics are allowed to enter the model. The variables included in the model are shown in table 6. Table 6 shows that 23 metrics are included in the multivariate model. It can be observed from the table that the 'sig.' value of all the variables included in the model is less than 0.05. The sign of the coefficient of AL, ALB, CCB, CDCM, CDIM,CLB,CLCE,CLC,CS,ME,DIT,SCmetrics is negative, though it was positive in the univariate analysis. This is due to suppressor relationships among independent variables commonly observed in multivariate logistic regression analysis (Briand et al. 2000).

Table 7 shows the confusion matrix for this multivariate model which can be used to evaluate the performance of the model. To evaluate the performance of the models (LR and machine learning),

Table 5. Result analysis of univariate regression

S.No.	Metric	B	Sig.	S.No.	Metric	B	Sig.
1	AC (Average Cyclomatic)	.211	**0.000**	23	CL (Count Line)	.001	**0.000**
2	ACM (Average Cyclomatic Modified)	.291	**0.000**	24	CLB (Count Line Blank)	.002	**0.000**
3	ACS (Average Cyclomatic Strict)	.204	**0.000**	25	LOC (Lines of Code)	.001	**0.000**
4	AE (Average Essential)	.181	**0.000**	26	CLCD (Count Line Code Declared)	.004	**0.000**
5	AL (Average Line)	.042	**0.000**	27	CLCE (Count Line Code Executed)	.004	**0.000**
6	ALB (Average Line Blank)	.026	0.099	28	CLC (Count Line Comment)	.006	**0.000**
7	ALC (Average Line Code)	.050	**0.000**	29	(CS) Count Semicolon	.004	**0.000**
8	ALCo (Average Line Comment)	.078	**0.000**	30	CS (Count Statement)	.002	**0.000**
9	CCB (Count Class Base)	.195	**0.000**	31	CSD (Count Statement Declared)	.003	**0.000**
10	CBO (Coupling Between Objects)	.110	**0.000**	32	CSE (Count Statement Executed)	.004	**0.000**
11	NOC (Number Of Children)	.000	0.697	33	MC (Maximum Cyclomatic)	.112	**0.000**
12	CDCM (Count Declared Class Method)	.060	**0.000**	34	MCM (Maximum Cyclomatic Modified)	.145	**0.000**
13	CDCV (Count Declared Class Variable)	.045	**0.000**	35	MCS (Maximum Cyclomatic Strict)	.097	**0.000**
14	CDIM (Count Declared Instance Method)	.000	0.754	36	ME (Maximum Essential)	.177	**0.000**
15	CDF (Count Declared Function)	-.896	**0.000**	37	DIT (Depth of Inheritance Tree)	-.108	**0.000**
16	CDIV (Count Declared Instance Variable)	.129	**0.000**	38	MN (Maximum Nesting)	.544	**0.000**
17	WMC (Weighted Methods per Class)	.042	0.327	39	LCOM (Lack of Cohesion of Methods)	.019	**0.000**
18	RFC (Response For a Class)	.000	**0.007**	40	RCC (Ratio Comment to Code)	.009	0.422
19	CDMD (Count Declared Method Default)	.207	**0.000**	41	SC (Sum Cyclomatic)	.002	**0.000**
20	NPRM (Number of Private Methods)	.171	**0.000**	42	SCM (Sum Cyclomatic Modified)	.002	**0.000**
21	NPROM (Number of Protected Methods)	.146	**0.000**	43	SCS (Sum Cyclomatic Strict)	.002	**0.000**
22	NPM (Number of Public Methods)	.000	0.114	44	SE (Sum Essential)	.001	**0.000**

Table 6. Metrics included in the multivariate regression model

S.No.	Metrics included in the model	B	Sig.	S.No.	Metrics included in the model	B	Sig.
1	SE	.025	.001	12	CL	.011	.000
2	AL	-.026	.010	13	CLB	-.027	.000
3	ALB	-.160	.000	14	CLCE	-.007	.000
4	ALC	.052	.000	15	CLC	-.008	.000
5	CCB	-.099	.032	16	CS	-.006	.001
6	CBO	.081	.000	17	CSD	.007	.004
7	CDCM	-.026	.001	18	MC	.033	.000
8	CDIM	-.020	.003	19	ME	-.056	.000
9	CDIV	.024	.000	20	DIT	-.231	.000
10	RFC	.001	.000	21	LCOM	.004	.000
11	CDMD	.065	.000	22	SC	-.028	.001
				23	SCM	.016	.028

Table 7. Confusion matrix for multivariate logistic regression

		Predicted	
		0 (Not change prone)	1 (Change prone)
Actual	0 (Not change prone)	6730	420
	1 (Change prone)	1824	1094

a popularly used performance measure known as accuracy is employed. Accuracy is the fraction of predictions the model got right. In other words, it is defined as the ratio of correct predictions to the total number of predictions (both correct and incorrect). It can be seen from table 7 that the out of a total of 10068 classes, 7824 classes are correctly predicted. In other words, 6730 classes are correctly predicted to be change prone and 1094 classes are correctly predicted to be not change prone. Thus, the accuracy of the LR model is 77.77% which is good but much less as compared to the accuracy of the models achieved using machine learning classifiers (discussed in coming section).

Evaluation of results using wrapper methods

This section depicts the performance of three classifiers used in the study viz. KNN, RF and DT in terms of accuracy when the features have been reduced using four sequential search techniques of wrapper methods viz. SFS, SBS, SFFS and SBFS.

The authors depict the results with the values obtained for k=1,9,18,27,36,44 where k is the number of features obtained after feature selection. The authors aim to analyze the trend which can be observed in the accuracy obtained on the basis of the number of features selected and thereby studying that in order to get the best possible accuracy, should the number of features selected be high, low or average. Table 8 and 9 show the performance of classifiers when empirical validation has been conducted on Android 6.0 and Android 7.0 dataset respectively. The accuracy results are

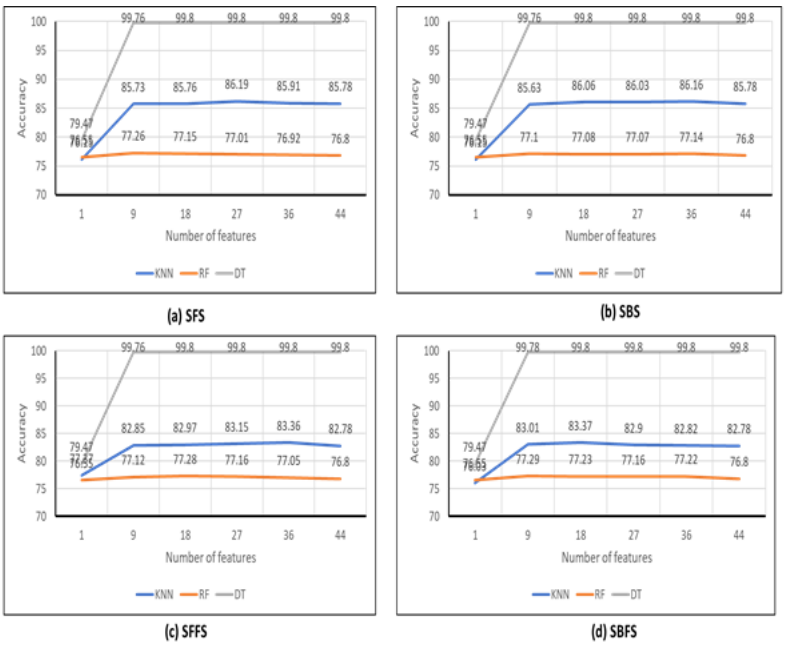
Table 8. Performance of classifiers with respect to different search techniques applied on Android 6.0dataset

SFS				SBS			
No. of features	**KNN**	**RF**	**DT**	**No. of features**	**KNN**	**RF**	**DT**
1	76.15	76.55	79.47	1	76.15	76.55	79.47
9	85.73	**77.26**	99.76	9	85.63	77.10	99.76
18	85.76	77.15	**99.80**	18	**86.06**	**77.08**	**99.80**
27	**86.19**	77.01	99.80	27	86.03	77.07	99.80
36	85.91	76.92	99.80	36	86.16	77.14	99.80
44	85.78	76.80	99.80	44	85.78	76.80	99.80
SFFS				**SBFS**			
No. of features	**KNN**	**RF**	**DT**	**No. of features**	**KNN**	**RF**	**DT**
1	77.37	76.55	79.47	1	76.03	76.55	79.47
9	82.85	77.12	99.76	9	83.01	**77.29**	99.78
18	82.97	**77.28**	**99.80**	18	**83.37**	77.23	**99.80**
27	83.15	77.16	99.80	27	82.90	77.16	99.80
36	**83.36**	77.05	99.80	36	82.82	77.22	99.80
44	82.78	76.80	99.80	44	82.78	76.80	99.80

Table 9. Performance of classifiers with respect to different search techniques applied on Android 7.0 dataset

SFS				SBS			
No. of features	**KNN**	**RF**	**DT**	**No. of features**	**KNN**	**RF**	**DT**
1	74.69	74.54	77.11	1	72.82	74.54	77.11
9	81.51	**75.28**	99.73	9	84.99	**75.08**	99.73
18	81.53	75.14	**99.78**	18	85.30	74.84	**99.78**
27	**81.70**	75.21	99.78	27	**85.53**	74.88	99.78
36	81.89	74.99	99.78	36	85.57	75.04	99.78
44	81.50	73.96	99.78	44	81.50	73.96	99.78
SFFS				**SBFS**			
No. of features	**KNN**	**RF**	**DT**	**No. of features**	**KNN**	**RF**	**DT**
1	74.69	74.54	77.11	1	74.69	74.54	77.11
9	81.51	**75.34**	99.73	9	81.51	**75.19**	99.73
18	81.53	75.25	**99.78**	18	81.53	75.17	**99.78**
27	81.95	75.03	99.78	27	**81.95**	75.11	99.78
36	**82.14**	75.03	99.78	36	81.76	75.05	99.78
44	81.50	73.96	99.78	44	81.50	73.96	99.78

Figure 5. Accuracy results of classifiers when feature selection of Android 6.0 dataset has been done using (a) SFS, (b) SBS, (c) SFFS, (d) SBFS

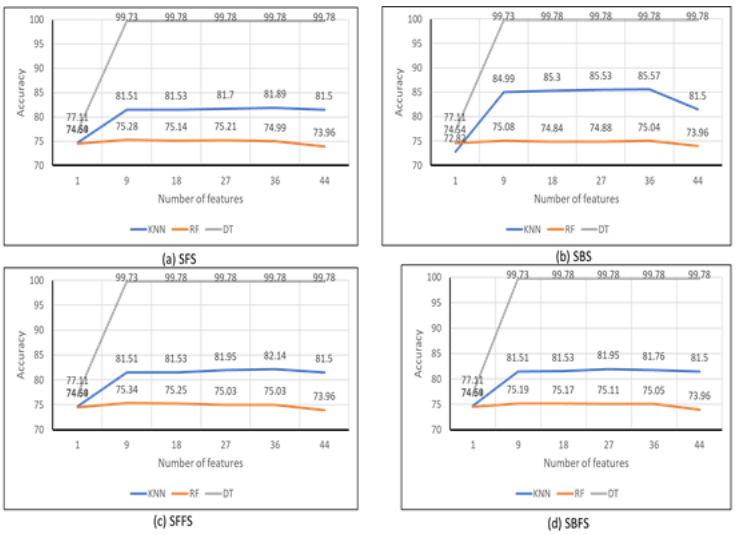


also graphically represented in figures5(a-d) and 6(a-d) depicting the performance of classifiers when SFS, SBS, SFFS and SBFS techniques have been applied on Android 6.0 and Android 7.0 dataset respectively.

Let us first discuss the accuracy achieved by each of the classifiers when SFS technique has been used for feature selection on Android 6.0 dataset. As it can be observed from table 8, DT achieves the highest accuracy values for all the values of k, while RF achieves the lowest values of accuracy. The accuracy in the case of each of the classifier increases as the value of k increases for lower values of k, however, it decreases when k approaches the maximum value. The highest accuracy value for each classifier is shown in bold. Analyzing the values of k at these highest accuracies show that in none of the cases the highest accuracy is attained at k=1 or k=44. This shows that when features are reduced, the accuracy increases but we have to carefully decide the minimum number of features. The accuracy value corresponding to DT begins at 79.47% for k=1, keeps increasing till k=18, attaining the value of 99.8% and then remains constant. The accuracy value for KNN begins at 76.15% for k=1 and increases until k=27, reaching the value 86.19% and then starts decreasing. The accuracy value observed for KNN is 85.78%when k=44, lower than the accuracy value corresponding to DT but higher than that of RF. The accuracy value with respect to RF begins at 76.55% for k=1, a value that is lower than the corresponding value of accuracy for DT but higher than KNN. The accuracy increases till k=9, reaching as high as 77.26% and then decreases continuously to stop at 76.80% for k=44. It can be observed that the most optimal results in general are obtained when the value of k= 18 with DT chosen as the classifier. This observation is consistent with other feature selection techniques viz. SBS, SFFS and SBFS too.

For Android 7.0, similar observations can be inferred. The highest accuracies are obtained at k=9 for RF and k=18 for DT using all the feature selection techniques. When KNN is used, highest accuracies are obtained at k=27 for all the feature selection techniques except SFFS which shows highest accuracy at k=36. DT shows the highest accuracy of 99.78% at k=18 amongst all the

Figure 6. Accuracy results of classifiers when feature selection of Android 7.0 dataset has been done using (a) SFS, (b) SBS, (c) SFFS, (d) SBFS



classifier with all the feature reduction techniques. As discussed above, same conclusion is drawn from the results on Android 6.0. Thus, the authors recommend the use of DT for model prediction after reducing the number of features.

APPLICATION OF THE WORK

The usage of software in every field of Network and Communication has become an integral part of everyone's lives. Due to multiple reasons such as ever changing demands of the customers, change in technology in the market, identification of some bug etc. lead to upgradation of a software from one version to the next. When software developers work on upgrading the version, lots of effort is required leading to utilization of large amount of resources. The results of this work will be of interest to researchers as well as practitioners from industry in reducing this effort in terms of time, money and manpower required in development of software, thus leading to the delivery of software with better quality. This is done by identifying those classes amongst other classes which may change during later phases of development and thus, need careful attention by the developers, designers and testers. Such classes ae termed as 'change - prone' classes. The authors in this paper have developed various machine learning models which can be used to predict change-prone classes. Timely identification of such classes would be of great benefit to the software developers from industry as these classes play a critical role in design and architecture of the system. The architecture of the system once built acts as the scaffolding in which the functionality of the system is delivered, thus ensuring that system delivered meets the customer's functional expectations and needs. Once the change - prone classes are identified during the early phases, the architecture of the system can be altered easily. For example, if a class 'A' is predicted to be change - prone, using the software metrics we can check its coupling, cohesion, inheritance etc. Accordingly we can modify the architecture in the design phases such that the values of these software metrics fall within the required range. In other words, certain corrective measures can be taken to ensure the software delivered is error-free which is actually significant when different versions of software are released frequently. The measurements derived from design can be used as benchmark in organizations, to assess the quality of software products. Releasing software

of good quality leads to happy and satisfied customer. This leads in increasing the reputation and status of the software organization (in which the software is developed) in the market. Thus, customer satisfaction which is of utmost importance in today's scenario is met. Also, design of the classes of future release can be re-checked and better designs can be suggested.

THREATS TO VALIDITY

The empirical validation in this work has certain limitations which may adversely affect the validity of the results. These limitations are discussed in terms of four threats to validity, viz. construct validity, internal validity, external validity and conclusion validity.

1. Construct Validity

This type of validity is one of the most important threats to validity. It is defined as the extent to which the variables (independent and dependent variables) and the performance parameters precisely measure the concept they intend to measure (Dean and Voss 1999, Zhou et al. 2009). This threat can be due to the improper collection of the dependent variable and the independent variables. There have been studies in research which have determined the accuracy of some of the OO metrics employed in this study(Briand et al. 1998, 1999, 2000). As a result, this threat is reduced in the study. The dependent and the independent variables in this study are collected using the tool known as CRG (Malhotra et al. 2016) developed by the authors themselves. Due to this, the exact steps or procedure to collect the dependent and the independent variables is known, which is important to provide for an accurate assessment of their construct validity. Hence, the threat due to improper collection of the variables is also reduced in this study. However, the authors have not taken into consideration the type of change (corrective, adaptive, perfective or preventive) a class may go through which may pose a threat on the evaluation of the results. In the future work, this threat can be reduced by considering the type of change as well.

2. Internal Validity

This validity is defined as the degree to which conclusions can be drawn about the causal effect of independent variable on the dependent variable" (Zhou et al. 2009). In this work, independent variables used are a set of OO metrics measuring different concepts of OO paradigm. These independent variables are not related to each other in any way. All these metrics together determine the value of the dependent variable. It is not possible to determine the causal effect of each independent variable on the dependent variable. In other words, goal of this study is to develop prediction models that will identify change - prone classes rather than to discover the cause-effect relationships. Thus, the threat to internal validity does not exist in the study.

3. External Validity

This validity deals with the generalization of the results obtained by the study. In other words, it concerns itself with finding out whether the results produced by the study are applicable in different domains or can be replicated in different scenarios for which the results are not evaluated (Harrison et al. 2000). For this, all such information which is required for replicating the study should be transparently available in the study. For example, the information about the availability of the dataset, nature of the dataset, details of the approaches used in the paper and their default or tuned parameters etc. should be clearly stated in the study. Since, in the present work, Android dataset used is an open source dataset, the source codes of all the versions can be downloaded from https://source.android.

com/source/initializing.html. In addition to this, the authors have provided complete details of the three versions that are analysed in this study in table 3. Moreover, for empirical validation, the authors have written the programs in python. The classifiers have been implemented using scikit-learn library. This library has in-built classifiers which can be called using functions. The functions take in various parameters for the classifier to be built. The feature selection algorithm has been made using another python library, namely mlxtend. The pseudo code of each of the feature selection algorithm is provided and the parameters used for the machine learning classifiers are also mentioned in table 4. Using all this information, the results can be replicated and generalized across different datasets. In this study, the authors have worked on different releases of software that large in size with the aim to draw meaningful conclusions. However, we cannot consider it as a complete generalization wherein the results could be applied universally. This is so because we are dealing with multiple versions of a single software as of now. In the future, we will be comparing the results of different datasets across different projects having diverse characteristics.

4. Conclusion Validity

This threat includes all those threats that may affect the conclusion of the study. In other words, all the threats which may lead to improper results or conclusions of the study are called as conclusion validity threats (Malhotra 2016). The authors in this study have not performed the statistical evaluation of the results using statistical tests. Thus, this leads to a conclusion validity threat.

CONCLUSION AND FUTURE WORK

The upcoming technologies in mobile phones have evolved rapidly over the last some years. Nowadays, one of the essential needs of human beings is that of mobile phones. It helps everyone to be connected to each other all the time and all the places. Amongst many other factors playing important role in improving mobile's technology, operating system is one of the important factor too. Android, Symbian, iOS, Palm OS etc. are some of the commonly used operating systems of mobile integrated with multiple user-friendly features. The most popularly and widely used mobile operating system is Android. Android has seen multiple versions, the most recent one being Android 10.0 (officially known as Android 10).Advancing from one version to a newer version requires a lot of effort in terms of resources utilized. We know that the resources such as time, money and manpower are very limited and thus, judicious use of such resources is very essential. In this paper, the broad objective is to reduce this effort, thus leading to saving of resources. For doing this, machine learning models are constructed which can be used during the initial phases of software development for identifying certain parts of software which have a higher tendency to change in the later stages. Once such parts (classes) are known, careful attention is required to be given to them in order to avoid wastage of essential resources. The accuracy of such models should be high and high dimensionality play a very important role in the accuracy of the models. The authors in this paper are working on feature selection methods to reduce the dimensionality of the data.

In other words, feature selection is an important method that can be used for dimensionality reduction of multi-dimensional data that we often encounter in real life situations. Feature selection can be achieved via wrapper methods, filter methods or embedded methods but the authors choose wrapper methods as they analyze the utility of a subset of features and return the best possible subset. Feature selection can also be viewed in terms of searching techniques like sequential search, exponential search and random search. However, the authors implemented sequential search techniques as they are greedy optimization techniques that find the local optimum solution in polynomial time complexity and are easy to implement, even with large datasets. Authors focused on the implementation of the four major sequential feature selection techniques which are Sequential Forward Selection (SFS), Sequential

Backward Selection (SBS), Sequential Forward Floating Selection (SFFS) and Sequential Backward Floating Selection (SBFS). In addition to the wrapper methods for selecting useful features, the authors have also used statistical approach of regression analysis to extract the useful features. Univariate and multivariate Logistic Regression have been used to extract useful features and constructing the prediction model respectively. Empirical validation was conducted against three stable versions of Android operating system, Android 6.0, Android 7.0 and Android 8.0. The dataset comprised of 44 independent variables. The aim is to reduce the number of features and select the most relevant ones from this set. In order to achieve the above objective, prediction models are constructed using three machine learning classifiers viz. K-Nearest Neighbor (KNN), Decision Tree (DT) and Random Forest (RF). These models are used to identify those change prone classes of the software in early phases of software development, leading to optimum utilization of limited resources.

Following are the important insights gathered from the results:

- DT classifier outperformed the remaining two classifiers (KNN and RF) depicting high values of accuracy for each of the sequential feature selector used.
- With respect to KNN classifier, it was observed that SBS is the best sequential selector. On the other hand, with respect to RF classifier, SFFS sequential classifier is the best.
- Highest value of accuracy was achieved when the number of features selected was 9 or 18 (in general). Therefore, it can be inferred that either too high(44) or too low(1) number of features selected leads to decrease in accuracy. So, we must choose the number of features which is neither too high nor too low.
- The machine learning models (built using KNN, DT and RF) outperformed the statistical model (built using multivariate Logistic Regression) by depicting higher values of accuracy. Thus, the authors suggest the use of machine learning models for predicting the change prone classes of Android operating system.

As future work, the authors plan to work on feature selection methods based on other searching techniques as well, i.e. exponential search and random search so that a comparative analysis can be done. For the purpose of empirical validation, the study can be extended on different datasets belong to diverse domains. In addition to this, more machine learning techniques could be used so that a fair and exhaustive evaluation is possible.

REFERENCES

Ang, J. C., Mirzal, A., Haron, H., & Hamed, H. N. A. (2016). Supervised, Unsupervised and Semisupervised Feature Selection: A Review on Gene Selection. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *13*(5), 971–989. doi:10.1109/TCBB.2015.2478454 PMID:26390495

Auxilia, M., Raja, K., & Kannan, K. (2020). Cloud-Based Access Control Framework for Effective Role Provisioning in Business Application. *International Journal of System Dynamics Applications*, *9*(1), 1–18. doi:10.4018/IJSDA.2020010104

Bachu, V., & Anuradha, J. (2019). A Review of Feature Selection and Its Methods. *Cybernetics and Information Technologies*, *19*(1), 3–26. doi:10.2478/cait-2019-0001

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, *13*, 281–305.

Bhardwaj, A. (2020). Health Insurance Claim Prediction Using Artificial Neural Networks. *International Journal of System Dynamics Applications*, *9*(3), 1–18.

Boehm, B., & Basili, V. R. (2001). Software defect reduction top 10 list. *Computer*, *34*(1), 135–137. doi:10.1109/2.962984

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2014). Data classification using an ensemble of filters. *Neurocomputing*, *135*, 13–20. doi:10.1016/j.neucom.2013.03.067

Briand, L., Daly, J., & Wust, J. (1998). A unified framework for cohesion measurement in object-oriented systems. *Empirical Software Engineering*, *3*(1), 65–117. doi:10.1023/A:1009783721306

Briand, L., Daly, J., & Wust, J. (1999). A unified framework for coupling measurement in object-oriented systems. *IEEE Transactions on Software Engineering*, *25*(1), 91–121. doi:10.1109/32.748920

Briand, L., Wust, J., Daly, J. W., & Victor Porter, D. (2000). Exploring the relationship between design measures and software quality in object-oriented Systems. *Journal of Systems and Software*, *51*(3), 245–273. doi:10.1016/S0164-1212(99)00102-8

Corbane, C., Baghdadi, N., Descombes, X., Wilson, G. J., Villeneuve, N., & Petit, M. (2009). Comparative Study on the Performance of Multiparameter SAR Data for Operational Urban Areas Extraction Using Textural Features. *Geoscience and Remote Sensing Letters, IEEE*, *6*(11), 728–732. doi:10.1109/LGRS.2009.2024225

Dash, M., Liu, H., & Yao, J. (1997). Dimensionality Reduction of Unsupervised Data. *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*. doi:10.1109/TAI.1997.632300

Dean, A., & Voss, D. (1999). *Design and analysis of experiments*. Springer. doi:10.1007/b97673

Elfouly, F. H., Ramadan, R. A., Mahmoud, M. I., & Dessouky, M. I. (2017). Efficient Data Reporting in a Multi-Object Tracking Using WSNs. *International Journal of System Dynamics Applications*, *6*(1), 1–20. doi:10.4018/IJSDA.2017010103

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hall, M. A., & Smith, L. A. (1999). Feature Selection for Machine Learning: Comparing a Correlation-based Filter Approach to the Wrapper. *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, 235-239.

Harrison, R., Counsell, S., & Nithi, R. (2000). Experimental Assessment of the Effect of Inheritance on the Maintainability of Object-Oriented Systems. *Journal of Systems and Software*, *52*(2-3), 173–179. doi:10.1016/S0164-1212(99)00144-2

Hosmer, D., & Lemeshow, S. (1989). *Applied logistic regression*. Wiley.

Hu, Z., Bao, Y., Xiong, T., & Chiong, R. (2015). Hybrid filter–wrapper feature selection for short-term load forecasting. *Engineering Applications of Artificial Intelligence, 40,* 17–27.

Hussein, H. A. T., Ammar, M. E., & Moustafa Hassan, M. A. (2017). Three Phase Induction Motor's Stator Turns Fault Analysis Based on Artificial Intelligence. *International Journal of System Dynamics Applications*, *6*(1), 1–19. doi:10.4018/IJSDA.2017070101

Jain, K., & Vailaya, A. (1996). Image Retrieval Using Color and Shape. *Pattern Recognition*, *29*(8), 1233–1244. doi:10.1016/0031-3203(95)00160-3

Jain, A., & Zonkar, D. (1997). Feature Selection: Application, Evaluation and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(2), 153–158. doi:10.1109/34.574797

Kagdi, H., & Maletic, J.I. (2016). Software-Change Prediction: Estimated+Actual. *International Journal of Computer Applications in Technology, 54*(4), 240-256.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*(1-2), 273–324. doi:10.1016/S0004-3702(97)00043-X

Kumar, V., & Minz, S. (2015). Feature Selection: A literature Review. *Proceedings of the Third International Symposium on Women in Computing and Informatics, 31-37.*

Kumari, A., Tripathi, R., Pal, M., & Chakraborty, S. (2012). Linear Search versus Binary Search: A statistical comparison for binomial inputs. *International Journal of Computer Science Engineering and Applications*, *2*(2), 29–39.

Ladha, L., & Deepa, T. (2011). Feature selection methods and algorithms. *International Journal on Computer Science and Engineering*, *3*(5), 1787–1797.

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classiðcation and clustering. *IEEE Transactions on Knowledge and Data Engineering*, *17*(4), 491–502. doi:10.1109/TKDE.2005.66

Ma, S., & Huang, J. (2008). Penalized feature selection and classiðcation in bioinformatics. *Briefings in Bioinformatics*, *9*(5), 392–403. doi:10.1093/bib/bbn027 PMID:18562478

Majhi, S. K. (2018). An Efficient Feed Foreword Network Model with Sine Cosine Algorithm for Breast Cancer Classification. *International Journal of System Dynamics Applications*, *7*(2), 1–14. doi:10.4018/IJSDA.2018040101

Malhotra, R. (2016). *Empirical research in software engineering: concepts, analysis, and applications*. CRC Press, Taylor & Francis. doi:10.1201/b19292

Malhotra, R., Bansal, A., & Jajoria, S. (2016). An Automated Tool for Generating Change Report from Open-Source Software. *Int. Conference on Advances in Computing, Communications and Informatics (ICACCI)*. doi:10.1109/ICACCI.2016.7732273

Maldonado, S., & Weber, R. (2009). A wrapper method for feature selection using Support Vector Machines. *Information Sciences, 179,* 2208–2217.

Marcano-Cedeño, A., Quintanilla-Domínguez, J., Cortina-Januchs, M. G., & Andina, D. (2010). Feature Selection Using Sequential Forward Selection and classification applying Artificial Metaplasticity Neural Network. *36th Annual Conference on IEEE Industrial Electronics Society (IECON).* doi:10.1109/IECON.2010.5675075

Padmaja, D. L., & Vishnuvardhan, B. (2016). Comparative Study of Feature Subset Selection Methods for Dimensionality Reduction on Scientific Data. *IEEE 6th International Conference on Advanced Computing (IACC).*

Panda, M. (2019). Software Defect Prediction Using Hybrid Distribution Base Balance Instance Selection and Radial Basis Function Classifier. *International Journal of System Dynamics Applications*, *8*(3), 53–75. doi:10.4018/IJSDA.2019070103

Peralta, D., Rios, S., Ramirez-Gallegos, S., Triguero, I., Benitez, J. M., & Herrera, F. (2015). Evolutionary Feature Selection for Big Data Classification: A MapReduce Approach. *Mathematical Problems in Engineering*, *2015*, 1–11. doi:10.1155/2015/246139

Pudil, P., Novovicova, J., & Kittler, J. (1993). Floating search methods in feature selection. *Pattern Recognition Letters*, *15*(11), 1119–1125. doi:10.1016/0167-8655(94)90127-9

Rattanawadee, P., & Srivihok, A. (2015). Wrapper Feature Subset Selection for Dimension Reduction Based on Ensemble Learning Algorithm. *Procedia Computer Science*, *72*, 162–169. doi:10.1016/j.procs.2015.12.117

Rodriguez-Galiano, V.F., Luque-Espinar, J.A., Chica-Olmo, M., & Mendes, M.P. (2018). Feature selection approaches for predictive modelling of groundwater nitrate pollution: An evaluation of filters, embedded and wrapper methods. *Science of the Total Environment, 624,* 661–672.

Saeys, Y., Inza, I., & Larraaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics (Oxford, England)*, *23*(19), 2507–2517. doi:10.1093/bioinformatics/btm344 PMID:17720704

S'anchez-Maro˜no, N., Alonso-Betanzos, A., & Tombilla-Sanrom'an, M. (2007). Filter Methods for Feature Selection- A Comparative Study. In H. Yin, P. Tino, E. Corchado, W. Byrne, & X. Yao (Eds.), Lecture Notes in Computer Science: Vol. 4881. *Intelligent Data Engineering and Automated Learning - IDEAL 2007* (pp. 178–187). Springer. doi:10.1007/978-3-540-77226-2_19

Setiono, R., & Liu, H. (1996). A probabilistic approach to feature selection - a filter solution. *ICML'96 Proceedings of the Thirteenth International Conference on International Conference on Machine Learning Pages,* 319-327.

Sharma, V. S., Ramnani, R. R., & Sengupta, S. (2014). A framework for identifying and analyzing non-functional requirements from text. *Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture*. doi:10.1145/2593861.2593862

Singh, A., Halgamuge, M. N., & Lakshmiganthan, R. (2017). Impact of Different Data Types on Classifier Performance of Random Forest, Naïve Bayes, and K-Nearest Neighbors Algorithms. *International Journal of Advanced Computer Science and Applications*, *8*(12). Advance online publication. doi:10.14569/IJACSA.2017.081201

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Society of Arts*, *36*(2), 111–114. doi:10.1111/j.2517-6161.1974.tb00994.x

Yang, P., Liu, W., Zhou, B. B., Chawla, S., & Zomaya, A. Y. (2013). Ensemble-Based Wrapper Methods for Feature Selection and Class Imbalance Learning. In J. Pei, V. S. Tseng, L. Cao, H. Motoda, & G. Xu (Eds.), Lecture Notes in Computer Science: Vol. 7818. *Advances in Knowledge Discovery and Data Mining. PAKDD 2013*. Springer. doi:10.1007/978-3-642-37453-1_45

Yang, Y., & Pedersen, J. O. (1997). A Comparative Study on Feature Selection in Text Categorization. *ICML '97 Proceedings of the Fourteenth International Conference on Machine Learning Pages,* 412-420.

Wu, S., Hu, Y., Wang, W., Feng, X., & Shu, W. (2013). Application of global optimization methods for feature selection and machine learning. *Mathematical Problems in Engineering*, ●●●, 1–9.

Zhou, Y., Leung, H., & Xu, B. (2009). Examining the Potentially Confounding Effect of Class Size on the Associations between Object-Oriented Metrics and Change-Proneness. *IEEE Transactions on Software Engineering*, *35*(5), 607–623. doi:10.1109/TSE.2009.32

Zhuo, L., Zheng, J., Li, X., Wang, F., Ai, B., & Qian, J. (2008). A genetic algorithm based wrapper feature selection method for classification of hyperspectral images using support vector machine. *Proc. SPIE 7147, Geoinformatics and Joint Conference on GIS and Built Environment: Classification of Remote Sensing Images,* 71471. doi:10.1117/12.813256

J. A. Jain is an Assistant Professor at the Department of Computer Science Engineering, Shaheed Rajguru College of Applied Sciences for Women, Delhi University, India. Prior to joining the college, she worked as full-time research scholar and received a doctoral research fellowship from Delhi Technological University (formerly Delhi College of Engineering). She received her master's and doctorate degree in software engineering from Delhi Technological University. Her research interests are IoT, networks and communication, data mining, software quality, and statistical and machine learning models. She has published papers in International journals and conferences.

A. Bansal is an Assistant Professor at the Department of Information Technology, Netaji Subhas University of Technology (formerly known as Netaji Subhas Institute of Technology), Delhi, India. Prior to joining the university, Ankita Bansal worked as full-time research scholar at Delhi Technological University (formerly Delhi College of Engineering), Delhi, India. She received her master's and doctoral degree in computer science from Delhi College of Engineering. Her research interests are network models, communication technology, IoT, software quality, soft computing, database management, machine learning and meta heuristic models. She has published papers in international journals and conferences.