Software Defect Prediction in Large Space Systems

Aman Kumar Sinha - Jay Gupta - Pritish Vilin Zunke

ABSTRACT

Machine learning algorithms, especially classification algorithms have been used to classify various types of bugs, faults, inconsistencies in various systems. Classification hence becomes crucial to prevent and curb such inconsistencies in a system. Extensive research is already underway to classify such faults. The data has been taken from NASA's data set for space systems and we have used 9 datasets to classify defects. The main objective of this paper was to identify an optimal set of features using methods like ***Hybrid Feature Selection (HFS)*** *and verify results using* ***Mathews correlation coefficient (MCC)*** *and* ***Accuracy*** *of the model. The results obtained after HFS was applied are compared to results after the* ***Entire Feature Set (EFS)*** *was taken. It was observed that HFS reduced the feature size by one third of the original data set, and Accuracy of classification had almost remained the same and in some cases, it was even better.*

INTRODUCTION

Software Defects are the deviations of the actual values from the expected result of a software application or system. These defects are caused by the developers in the development phases. When a developer or software programmer makes some mistake during the development of application software, it turns out to be bugs which are called defects. These defects are very undesirable but it's not possible to completely omit these defects during the development stage. There are different types of defects like Arithmetic Defects, Logical Defects, Syntax Defects, Interface Defects, Performance Defects, etc., all of which cause the reduced performance and operational efficiency of the software.

In case of small systems, these defects might not be having too much impact, but in case of Large Space Systems, which are hard real-time systems, cannot afford to have these defects as it results in reduced operational efficiency as well as cause safety issues.

This is why Software defect prediction has been one of the very important fields in software engineering.Software defect prediction makes use of the machine learning and artificial intelligence algorithms to predict the defects in the software during the development stage, which will make it possible to fix it before it might cause problems. Machine learning techniques, viz, feature selection and classification have proven to be effective methods in predicting biological defects and irregularities in data. Feature Selection is the method for selecting optimal important features for building machine learning models. This method is used to reduce the training time of prediction models as well as improving the models' accuracies. Classification is one of the supervised machine learning algorithms which is used for classification of data into different classes.

The aim of the paper is to identify a minimum feature subset which can be used in the classification model for predicting the defects in the software with better accuracies and reducing the training time of the model. The performance metrics used in the study were Matthew's Correlation Coefficient, Accuracy, Sensitivity and Specificity. Classification of defects is a tedious task especially when the number of features taken into consideration is very high. To solve this issue we implement a hybrid feature selection algorithm that intends to reduce the set of features and only the optimal number of features remain.

DATASET USED

The data sets used in this paper are taken from NASA's database for space systems, the datasets worked on in this paper include CM1, KC3, MW1, JM1, PC1, PC2, PC3, PC4, PC5.

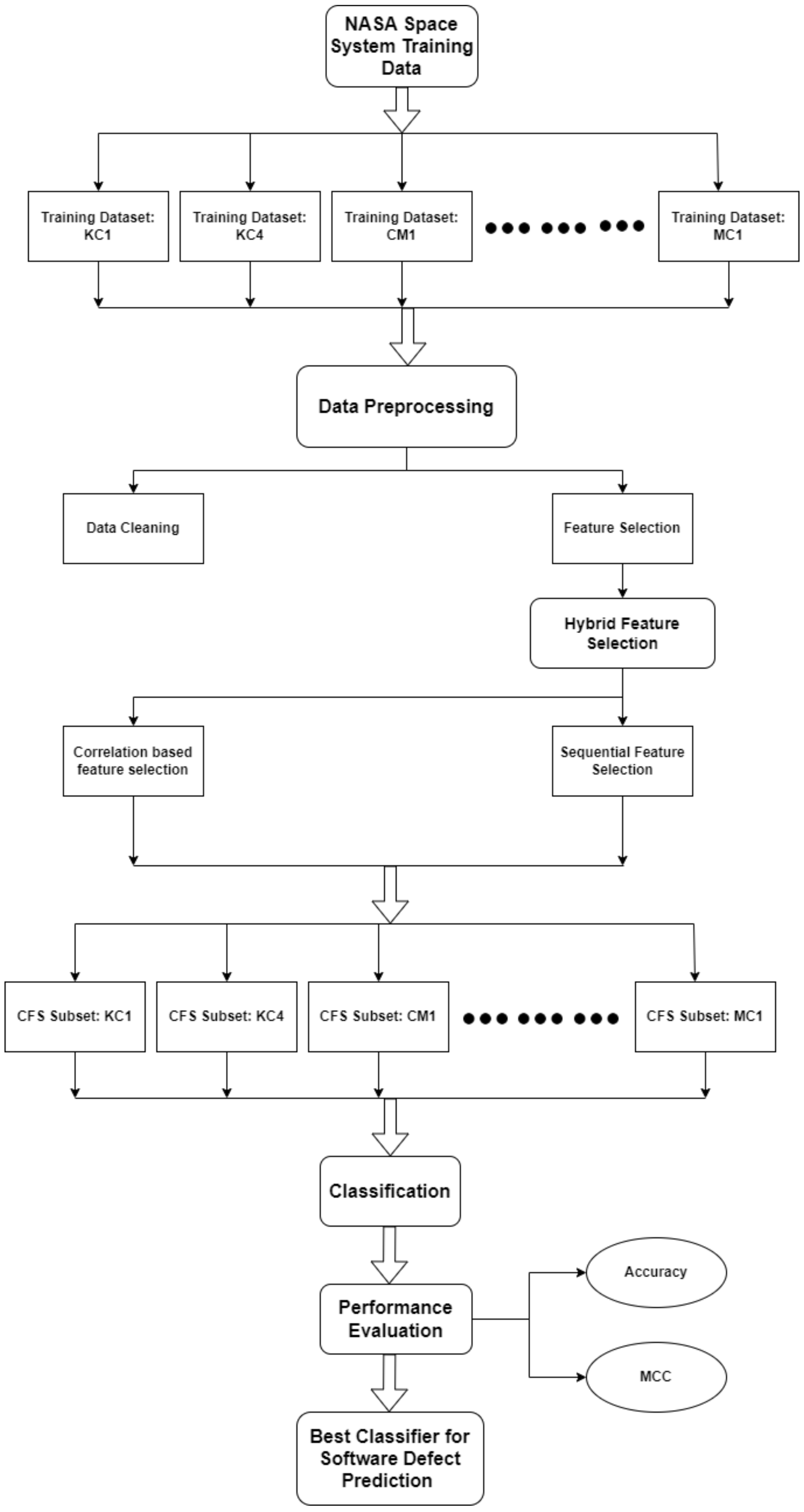
Dataset	Features	Observations	Description
CM1	38	327	NASA spacecraft instrument
KC3	40	194	Satellite-image data
MW1	38	253	Zero-gravity experiment related to combustion
JM1	22	7782	Real time predictive ground system
PC1	38	705	Flight software for earth orbiting satellite
PC2	37	745	Dynamic simulator for altitude control systems
PC3	38	1077	Flight software for earth orbiting satellite
PC4	38	1287	Flight software for earth orbiting satellite
PC5	39	1711	Flight software for earth orbiting satellite

FEATURE SELECTION

Classification of defects is a tedious task especially when the number of features taken into consideration is very high, to solve this issue we implement a hybrid feature selection algorithm that intends to reduce the set of feature and only optimal number of features remain to do this we apply the following:

- **Correlation based feature selection:**

In correlation based feature selection, correlation matrix is found for the given data set and then those features who have correlation greater than 0.95 with respect to other features are removed from the data set. This is done if two features have very similar characteristics, it is unnecessary to give both features to the model for training as it results in increase of the computational and time complexity of the model.


NASA Space System Training Data
Training Dataset: KC1
Training Dataset: KC4
Training Dataset: CM1
Training Dataset: MC1
Data Preprocessing
Data Cleaning
Feature Selection
Hybrid Feature Selection
Correlation based feature selection
Sequential Feature Selection
CFS Subset: KC1
CFS Subset: KC4
CFS Subset: CM1
CFS Subset: MC1
Classification
Performance Evaluation
Accuracy
MCC
Best Classifier for Software Defect Prediction

- **Sequential based feature selection:**

After the original set of features are passed to correlation based feature selector we take the obtained features and then pass it to the sequential feature selector.
In sequential based feature selection the Sequential Feature Selector appends or removes features to form a feature set in a greedy fashion. After which, this estimator chooses the most relevant feature to add or remove based on the cross-validation score of an estimator. We have used the forward sequential selection to find the set of optimal feature selection.
The set of features given after processing the feature set depends on the type of model passed in the sequential feature selector, i.e. the set of features obtained from KNN model will differ when compared to that of features that are obtained when the Decision Tree model is used.

Dataset	CM1	KC3	MW1	JM1	PC1	PC2	PC3	PC4	PC5
EFS	38	40	38	22	38	37	38	38	39
HFS	11	10	12	6	11	10	12	13	12

MODELS USED

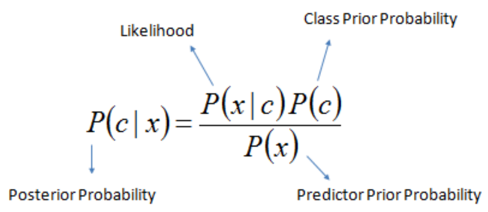
- **k-Nearest Neighbours (KNN)**
  KNN approach uses all classes that have been fed to the model, the class is classified according to the closest k nearest neighbors according to the distance .
  The distance measures used can be Euclidien, Manhattan, Minkowski.

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$ Euclidean

$$\sum_{i=1}^{k}|x_i - y_i|$$ Manhattan

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$ Minkowski

- **Naive Bayes (NB)**
  NB classifier classifies classes on probabilistic classifiers which are based on Bayes's theorem , this is one of the most simple Bayesian network models, but when used with kernel density estimation it would yield higher accuracy values.



$$P(c\mid x)=\frac{P(x\mid c)P(c)}{P(x)}$$

$$P(c\mid X) = P(x_1\mid c)\times P(x_2\mid c)\times\cdots\times P(x_n\mid c)\times P(c)$$

- **Decision Tree (DT)**
  DT is a supervised learning algorithm, which can be used for both classification or regression, the model classifies the data according to simple decision rules obtained from previous training data.
  The classification begins at the root node and form there we compare it with the record attributes, through this comparison we follow the branch and move to the next node, this process is continued until the leaf node is reached i.e. the model has classified the data.

  $$Gini = 1 - \sum_{i=1}^{n} p^2(c_i)$$

  $$Entropy = \sum_{i=1}^{n} -p(c_i)log_2(p(c_i))$$

  where $p(c_i)$ is the probability/percentage of class c_i in a node.

- **Random Forest (RF)**
  Random Forest classifier is an ensemble classifier, which is based on the Decision Tree. It fits multiple numbers of decision trees on different subsets of the dataset and then uses averaging to improve the accuracy and control the overfitting. Each decision tree in the random forest predicts the class and the class with highest votes is selected as the classifiers' output.

- **AdaBoost (AD)**
  The AdaBoost (Adaptive Boosting) Classifier works by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but the weights of incorrectly classified instances are adjusted such that the additional classifiers focus more on the difficult cases. It uses an algorithm known as AdaBoost-SAMME. One of the very important features of the AdaBoost classifier is that it is less affected by the overfitting problem.

METRICS USED

- **Accuracy Score**
  Finds the score base on correctly predicted and falsely predicted values. The values range from 0 to 1 where 0 means that the model produces results that are completely opposite to the actual results and 1 means that the model produces results that completely match the actual results.

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

- **Matthews Correlation Coefficient (MCC)**
  MCC is a correlation coefficient that find correlation between given pair of sets, the MCC value ranges from -1 to +1 where -1 indicated that the predicted values are completely opposite to the actual values, +1 indicated that the predicted values are the same as the actual values and 0 represents that the prediction is as good as a random prediction.

  $$N = TN + TP + FN + FP$$
  $$S = \frac{TP + FN}{N}$$
  $$P = \frac{TP + FP}{N}$$
  $$\text{MCC} = \frac{TP/N - S \times P}{\sqrt{PS(1-S)(1-P)}}$$

  TN is true negatives, TP is true positives, FN is false negatives and FP is false positives.

EXPERIMENTAL RESULTS

Accuracy for EFS

DATASET	KNN	NB	DT	RF	AD
CM1	0.899	0.859	0.869	0.879	0.828
KC3	0.763	0.729	0.797	0.729	0.746
MW1	0.934	0.776	0.789	0.894	0.881
PC1	0.919	0.863	0.896	0.915	0.873
PC2	0.987	0.933	0.969	0.987	0.964
JM1	0.785	0.789	0.695	0.797	0.791
PC3	0.873	0.614	0.842	0.867	0.867
PC4	0.899	0.881	0.863	0.906	0.883
PC5	0.719	0.727	0.739	0.774	0.752

MCC for EFS

DATASET	KNN	NB	DT	RF	AD
CM1	0.00	0.399	0.169	-0.048	0.011
KC3	0.00	0.076	0.385	0.008	0.165
MW1	0.00	0.266	0.020	0.143	0.119
PC1	0.00	0.184	0.257	0.322	0.113
PC2	0.00	0.130	0.016	0.00	0.018
JM1	0.035	0.192	0.115	0.245	0.171
PC3	0.00	0.247	0.249	0.146	0.231
PC4	0.00	0.344	0.388	0.438	0.459
PC5	0.133	0.191	0.351	0.385	0.313

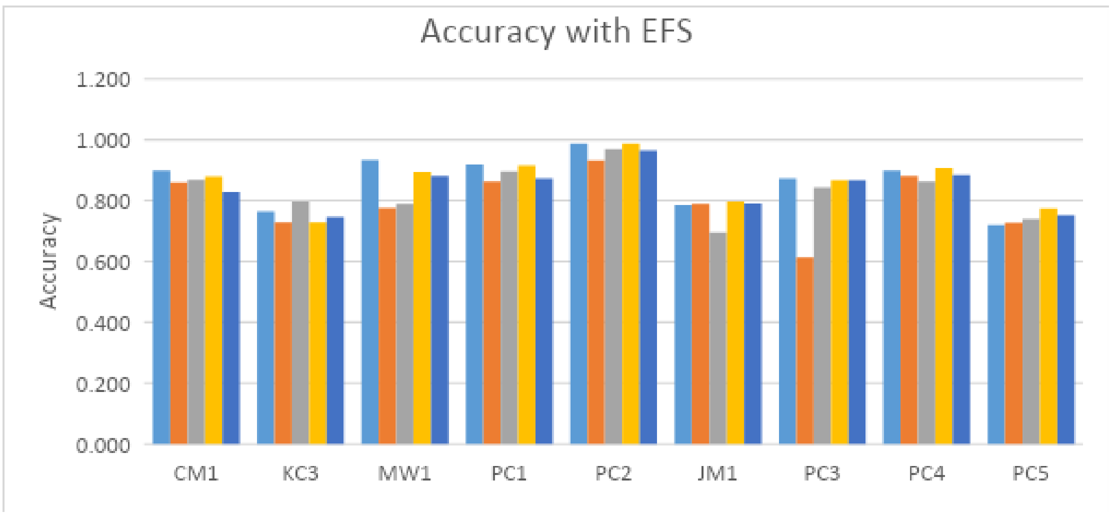
Accuracy for HFS

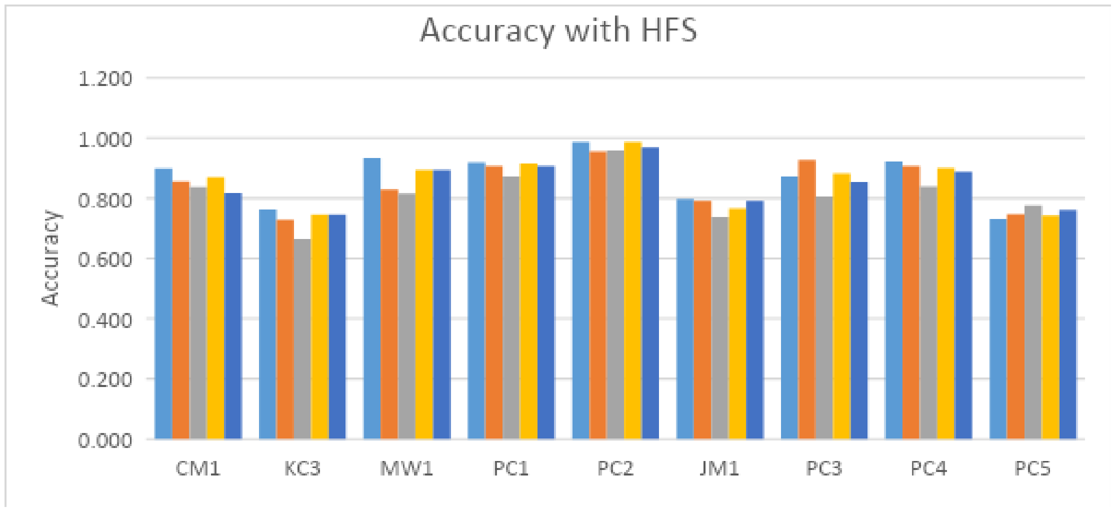
DATASET	KNN	NB	DT	RF	AD
CM1	0.899	0.856	0.838	0.869	0.818
KC3	0.763	0.729	0.664	0.746	0.746
MW1	0.934	0.829	0.816	0.895	0.895
PC1	0.919	0.906	0.873	0.915	0.906
PC2	0.987	0.955	0.959	0.988	0.969
JM1	0.797	0.791	0.737	0.766	0.791
PC3	0.873	0.927	0.805	0.882	0.854
PC4	0.922	0.906	0.839	0.901	0.888
PC5	0.731	0.747	0.776	0.743	0.761

MCC for HFS

DATASET	KNN	NB	DT	RF	AD
CM1	0.00	0.147	0.110	-0.059	-0.099
KC3	0.00	0.076	0.127	0.052	0.116
MW1	0.00	0.285	0.161	-0.053	0.114
PC1	0.00	0.286	0.313	0.282	0.286
PC2	0.00	0.021	0.187	0.00	0.016
JM1	0.203	0.208	0.166	0.155	0.159
PC3	0.00	0.234	0.164	0.262	0.192
PC4	0.466	0.400	0.302	0.433	0.350
PC5	0.205	0.208	0.329	0.398	0.268

GRAPHS





Models suitable for given datasets :

Dataset	CM1	KC3	JM1	MW1	PC1	PC2	PC3	PC4	PC5
Model	**KNN**	**KNN**	**KNN**	**KNN**	**KNN**	**RF**	**NB**	**KNN**	**DT**
Accuracy (%)	**89**	**76**	**91**	**79**	**91**	**98**	**92**	**92**	**74**

Hybrid feature selection algorithm reduced the set features to about one-third of the original feature set size. Even after the set of features were reduced the accuracy obtained was either increased or comparable to the accuracy obtained when the entire feature set was used.
For most of the datasets KNN was found to be the best performing model amongst the 5 models used, only the datasets PC2, PC3 and PC5 had other suitable models, RF, NB and DT respectively. It should be noted that for PC2, PC3 and PC5 KNN model accuracy values were comparable with respect to that of the best performing models.

CONCLUSION

This paper focused on the betterment of the software development process by identifying the best features and suitable models for detecting defects in large space systems. Nine datasets have been used, which were taken from NASA's database and identified best classification algorithms for the corresponding datasets. k-Nearest Neighbours was the best performing model, followed by viz., Decision Tree, Random Forest, Naive Bayes. To optimize the time and computational complexity, a hybrid feature selection algorithm was used, reducing the sets of features to one-third of their original size. This paper could be used to efficiently use the time and resources of software developers and minimize the length of the beta stage for large space systems.