Software Fault Prediction using Ensemble Techniques

Anil Kumar Pandey, Manjari Gupta
DST-CIMS, Banaras Hindu University, Lanka, Varanasi-221005
Uttar Pradesh, India
akpandey@bhu.ac.in

ABSTRACT: *In recent years, software fault classification became an important research area for the development of reliable and high quality software products. The fault classification helps in identification of fault software modules and allows the developers to concentrate on that module. It helps the developer to save the time and control financial losses to industry. Therefore, in this paper, a novel fault classification method based on feature ranking algorithms and ensemble techniques is proposed. The number of features available in the metrics are selected to represent the fault using feature ranking algorithms and operated on ensemble techniques to check the performance. Also, various hyper parameters are tuned for the ensemble techniques to identify the best model. The experimental result demonstrates the good results for bagging with K-nearest neighbor and random forest in comparison with other methods.*

Keyword: Ensemble Technique, Bagging, Nearest- Neighbour, Feature Ranking, Software Fault Prediction (SFP), Cognitive Computing

Received: 1 September 2019, Revised 28 November 2019, Accepted 6 December 2019

DOI: 10.6025/jnt/2020/11/1/25-41

1. Introduction

With the increasing role that is played by multiple software systems in our lives, the increase in their complexity is simply evident. This ever increasing complexity in software systems makes heavier vetting efforts necessary to maintain their quality. That is why a significant amount of focus has been laid on prioritization of software quality assurance efforts and software testing in recent relevant researches. At a high level, software testing is needed in order to detect the bugs in the software and to evaluate a software item to detect difference between given input and expected output. There are many benefits that can be achieved by using tools to support testing but at the same time there are some problems that are associated with tool such as huge consumptions of time and resources. We find that the defects are not distributed evenly among individual sections of software; that is why applying the same testing method on all the modules and sections of software cannot be considered an optimal solution [1].

Software defect prediction is a thriving research area with a very high amount of activity in software engineering today. The results from defect prediction modules provide with the list of source code artifacts that are prone to defects so that the teams looking over quality assurance can optimize and effectively allocate resources for validating software products by putting targeted effort on the particular defect-prone source codes. With a continuous and rather an exponential increase in the size of software projects, defect prediction techniques will gain importance in their role to support developers of software's as well as to speed up the time period taken to provide the market with reliable and defect-optimized software products. [2] Software defect prediction studies mandate historical data of the software which certainly cannot be applied on the first release of software or not on the software whose company doesn't store historical data. Therefore, it is better to build a software defect prediction model that is free from the requirement of historical data.

Software project clustering in order to detect software defect allows finding bug duplicates, to automate testing, to predict the testing workload and to improve the defect management practices. A preliminary study was conducted by Marian Jureczko and Lech Madeyski[3], where three clusters – proprietary projects, open-source projects and academic projects are investigated for their existence and the defect prediction model for open-source cluster was found statistically better and it was proved that this cluster exist.

This paper aims to propose different classification methods that might be helpful in predicting software defect. The ability to predict accurately defect-prone software modules can better streamline test effort, reduce costs incurred, and improve the quality of software. Use of machine learning classification algorithm is quite a common approach for predicting software defect. In this approach the software code attributes collected from previous development projects are categorized as defective or not defective. Classification algorithm is also able to predict which modules and components of the software show a tendency to be defect-prone, supports more streamlined targeted testing resources and as a result, improved efficiency.

2. Related Work

Defect prediction is a useful tool that is fast gaining importance in the field software engineering, it empowers software engineers so that they can pay more targeted attention to defect-prone code with software metrics, and this improves the software quality making better use of limited resources. Several studies have been made to help developers and testers of software's for identifying defects where Classification techniques are used on existing software metrics, as well as to help improve the quality of software which in turn will lead to minimization of maintenance cost. Different classification techniques have been surveyed in order to get the best result on different datasets.

Yuan Chen, et.al [4] had surveyed and applied the different classification techniques for software defect prediction. Their defect prediction model was based on Bayesian network and PRM. Hassan Najadat and IzzatAlsmadi [5] use Ridor algorithm to predict fault in modules in their new proposed model. They took the data sets that were provided by the American space agency NASA and tested the different classification techniques on them. The results clearly show that in accuracy and extraction of number of rules Ridor algorithm seems superior to any other existing technique. Ahmet Okutan, O lcayTanerYýldýz [6] came up with two new metrics apart from the metrics applied on PROMISE dataset repository – NOD (for the number of developers) and LOCQ (for source code quality) and using Classification technique they showed which metrics are more effective on dataset. Using Bayesian network classifier experiment proved that NOC & DIT are not trustworthy metrics. LOCQ is comparatively effective like CBO & WMC. NOD metric showed that there exists a positive correlation between the number of developers of software and the extent of defect. LOC has been proven to be one of the best metric for quick defect prediction. LCOM3 & LCOM have been less effective when compared to other metrics like LOC, CBO, RFC, and LOCQ & WMC.

Thair Nu Phyu [7] surveyed various techniques of classification such as Bayesian networks, case-based reasoning, decision tree induction, fuzzy logic techniques, genetic algorithm and k-nearest neighbour classifier, The results show that deciding which classification technique is more effective and provide best result is not easily predictable. Arvinder Kaur and Inderpreet Kaur [8] have tried identifying defects in classes as a means to estimate the quality of the software. For this purpose they have used six different classifiers such as Naive base, Logistic regression, Instance based (Nearest- Neighbour), Bagging, J48, Decision Tree and Random Forest. They applied this model on five different open source software to gauge the defects of 5885classes that were based on object oriented metrics and the result showed that only Bagging and J48 were the best. Wen Zhang et.al [9] has proposed Bayesian Regression Expectation Maximize algorithm for software effort prediction along with a couple of strategies that are embedded to handle missing data. In this predictive model the missing data was ignored in an iterative manner as a method. They have used datasets such as ISBSG and CSBSG. In case of no missing data, CR, BR, and SVR& M5 are clearly outperformed by BREM. If there are missing data, BREM with MDT and MDI embedded has a better performance than imputation technique which includes MI, BMI, CMI, and Mini and M5. In this model BREM is used for defect prediction in software and MDI is used for the purpose of finding missing data and values that are embedded with BREM.

K. Sankar et.al [10] has proposed a system for overcoming the problem of insufficiency in accuracy and also when a large number of features are used. To predict faults in software code they proposed the use of Feature Selection technique. This also measures the software code along with the performance of Naïve based and SVM classifier. The accuracy here is measured by the use of F-mean metric. Jureczko and Madeyski [11] have tried to use clustering techniques for division of various projects into different groups. They have hypothesized that a group is a set of projects with similar characteristics (such as predefined defects) and a defect prediction model should work well for all projects that belong to a same group. Jureczko's research points out a possible way for prediction of cross-project defects. If such groups can be successfully identified, we can do away with historical data and defect predivtion models from other projects which belong to the same group can be used instead.

A comparative research analysis on defect prediction by Turhan et al. [12], [13] based on project data of within company and cross-company. For the analysis, 10 projects (07 (NASA) and 03 (software company belonging to Turkey)) were selected and predictor was build using static code features. The conclusion was that the probability of defect detection (pd) increases with cross-company data at the cost of increasing false positive rate (pfr). Further to counter this problem the result from their experiment also shows that with cross-company data nearest neighbour filtering might come in handy to reduce pfr. However, to build a good predictor uses the static code from the same company. Nagappan et al. [14] studied different prediction models for their ability of generalization a for post-release defects. They investigated five various systems from Microsoft and found that single set of metrics incapable of providing the solution to all projects and formulated predictors tend to be more accurate when they are studied and taken from the same or similar projects. The requirement best static code attributes of defect prediction varies with different data sets. It shows that different data sets have different characterstics. This Difference in data characteristics, along with the different contexts that different projects have (process, developers, programming languages etc.), make cross-project defect prediction a big challenge [15].

Tang et al. [16] empirically studied three industrial real time systems, their study validated the CK object oriented metric suite. Their conclusion was: only WMC and RFC viably predict defect classes. They went on to propose a new set of metrics as practicable indicators of object oriented defect prone classes. Most of the earlier empirical studies seem to have ignored the confounding effect that the class size has while validating the metrics size. Fenton and O'Neil [17] have made a critical review of related literature and come up with a Bayesian networks based theoretical framework that could solve identified problems. They were against making the complexity metrics as the only predictor of defects, according to them proper attention should be paid to the data quality through statistical methodologies as well as the evaluation method. They also stressed that identification of the inherent relationships between faults and failures is also extremely important.

3. Materials and Methods

3.1 Metrics

Software metrics are measures used to improve the process of software development as well as its quality. Over the years there have been a continuous proposition and analysis of various software metrics. Among these, a few number of size and complexity metrics have been proven to be actually useful in defect prediction models. The Metrics suite suggested by Chidamber & Kemerer [1]:

3.2 Feature Ranking Algorithm

It is one of the feature ranking methods, this embedded method combines the qualities of a filter as well as wrapper methods. Algorithms having their own built-in feature selection methods are used by it in its implementation. LASSO and RIDGE regression are two of Embedded Feature Ranking Methods which have inbuilt penalization functions to reduce over fitting [18]. The ridge coefficient is given by

$$\beta'_{ridge} = \underset{\beta}{argmin} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2$$

subject to $\sum_{j=1}^{p} \beta_j^2 \leq t$

Similarly, lasso coefficient is given by

WMC - Weighted Methods for Class	$WMC = \sum_{i=1}^{n} c_i$
Depth of Inheritance	Depth of Inheritance = Depth of the class in the inheritance tree
Number of children (*NOC*)	Number of Children = Number of immediate descendants of the class
Coupling between object classes (*CBO*)	*CBO* = No. of Links / No. of classes
*LCOM*3	$LCOM3 = (M - sum(mA)/A) / (M-1)$
Number of Public Methods or Class Interface Size	*NPM* / *CIS* = Σ Public methods of a class
Data Access Metric (*DAM*):	$DAM = \frac{\text{No. of private or protected attribute of a class}}{\text{Total No. of attributes of class}}$
Measure of Aggregation (*MOA*)	MOA = Count of the data declarations whose types constitute the user defined classes.
Measure of Functional Abstraction (*MFA*):	$MFA = \frac{\text{No. of methods inherited by class}}{\text{No. of owned methods of class + No. of Inherited methods of class}}$
Cohesion among methods of class (*CAM*)	*CAM* = Relatedness or the amount of cohesion among methods of a class based upon the parameter list
Inheritance Coupling (*IC*)	*IC* = No of parent classes to which a given class is coupled
Coupling Between Methods (*CBM*)	*CBM* = Total number of new/redefined methods to which all the inherited methods are coupled.
Average Method Complexity (*AMC*)	*AMC* = Avg. (size of method for each class)
EFFERENT COUPLING (*CE*)	Efferent = Outgoing
AFFERENT COUPLING (*CA*)	Afferent = Incoming
Cyclomatic Complexity	$V(G) = E - N + 2$
Max (*CC*)	Greatest value of *CC* among methods of the investigated class
Avg (*CC*)	Arithmetic mean of the *CC* value in the investigated class.
Lines of Code	Count of the number of lines

Table 1. Software metrics

$$\beta'_{lasso} = \underset{\beta}{argmin} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2$$

subject to $\Sigma_{j=1}^{p} |\beta_j| \leq t$

3.3 Classifiers

In the proposed work, bagging with Decision Tree (B_DT), bagging with K-Nearest Neighbor (B_KNN), Adaboost with Support Vector Machine (A_SVM), extremely randomized trees (ET), and Random Forest (RF) classifiers are considered.

In bagging, replicate training sets are by sampling and then replaced from the training instances. These training sets are used to train the multiple classifiers to form a composite of classifiers. Each classifier votes to enhance the classifiers accuracy of classification [19]. In the proposed method, bagging is used along with decision tree and K-Nearest Neighbor. The algorithm 1-3 represents the algorithm for the B_DT and B_KNN [19-20].

Algorithm 1 Bagging

```
1:   $T$ = Number of bootstrap samples
2:   for $i$ = 1 to $T$ do
3:        Create training set of size $N$, $D_i$
4:        Train $C_i$ (base classifier) on $D_i$
5:   end for
6:   $C^*(x_{test}) = \arg\max_{y} \Sigma_i \delta(C_i(x_{test}) = y)$
```

Algorithm 2 Decision Tree

```
1:  $maxGain$ = 0 and SplitA = Null
2:  $e$ = Calculate Entropy of attributes
3:  for $all\ attributes\ a$ do
4:  $gain$ = Calculate the information gain using $a$ and $e$
5:  if $gain > max\ Gain$ then
6:  $maxGain = Gain$
7:  $splitA = a$
8:  end if
9:  end for
10: Create partition using $splitA$
11: Repeat the step 1 to 10 until all partitions are processed.
```

Algorithm 3 K-Nearest Neighbor

```
1: $k$ = Numbers of nearest neighbor, $X_{test}$ = Test sample set and $X_{train}$ = Training sample set
2: for $\forall x \in X_{train}$ do
3:    Calculate the distance between ($x$, $X_{test}$)
4: end for
5:Return the set indicating the $k$ smallest distance between ($x$, $X_{test}$)
```

Boosting is another method of repetition. In this, weights are maintained on training set to finds the importance. This method creates specific hypothesis by joining the a variety of weak hypotheses. However, each of week hypotheses is reasonably accurate [19-21]. Firstly, AdaBoost is a supervised iterative learning method and introduced by Freund and Schapire in [21] as a practical boosting. Algorithm 4 represents the AdaBoost. The weak classifier used in this method is support vector machine. The objective function of SVM is given by

$$\min_{w,b} \frac{\|w\|^2}{2}$$

With linear constraints

$$y_i(w^T x_i + b) \geq 1, \forall x_i \in X$$

Algorithm 4 AdaBoost

```
1:    $(X, Y)$ = Sample set
2:    Initialize the weights $D_1$
3:    Train the weak learner using distribution $D_t$
4:    Get the weak hypothesis $h_t$: X $\rightarrow \mathbb{R}$
5:    Choose $\alpha_t \in \mathbb{R}$ to update $D_{t+1}$
6:    Update $D_{t+1} = \frac{D_t - \alpha_t Y h_t(X)}{Z_t}$ where $Z_t$ is a normalization factor
7:    Repeat the steps 3 to 6 $T$ times
```

Further, Breiman [22] introduces the RF by utilizing random subset selection and bagging method. It provides the variety to RF by helping the weak learners. The algorithm for RF is shown in Figure 5. The RF improves the accuracy by reducing the variance in the datasets. Similar to RF, ET used for the reduction of variance by cut points and uses the complete dataset for building the tree [22-23]. The algorithm for ET is shown in Figure 6.

Algorithm 5 Random Forest

```
1: $T$ = numbers of trees and $mtry$ = value of feature set split
2: Resample the dataset $(X,Y)$ and create $T$ different subset
3: Apply decision tree on each subset
4: Classify based on each subset output
```

Algorithm 6 Extremely randomized trees

```
1: $T$ = numbers of trees and $mtry$ = value of feature set split
2: Resample the dataset $(X,Y)$ and create $T$ different subset
3: Apply decision tree on each subset with uniform random cut
4: Classify based on each subset output
```

4. Method

Based on the above theories, the proposed method is demonstrated in Figure 1. The following steps explain for fault classification.

1. The data set from the open source directory is analyzed.
2. Various metrics are analyzed and various feature ranking algorithms are used for ranking the features.
3. The feature set in created by adding the feature in input feature set.
4. The data is divided into two parts, 60% data used as training set and 40% data used as testing set.
5. Train classifiers with different combination estimators and feature sets.
6. Feed Testing data sets to analyze classification efficiency.

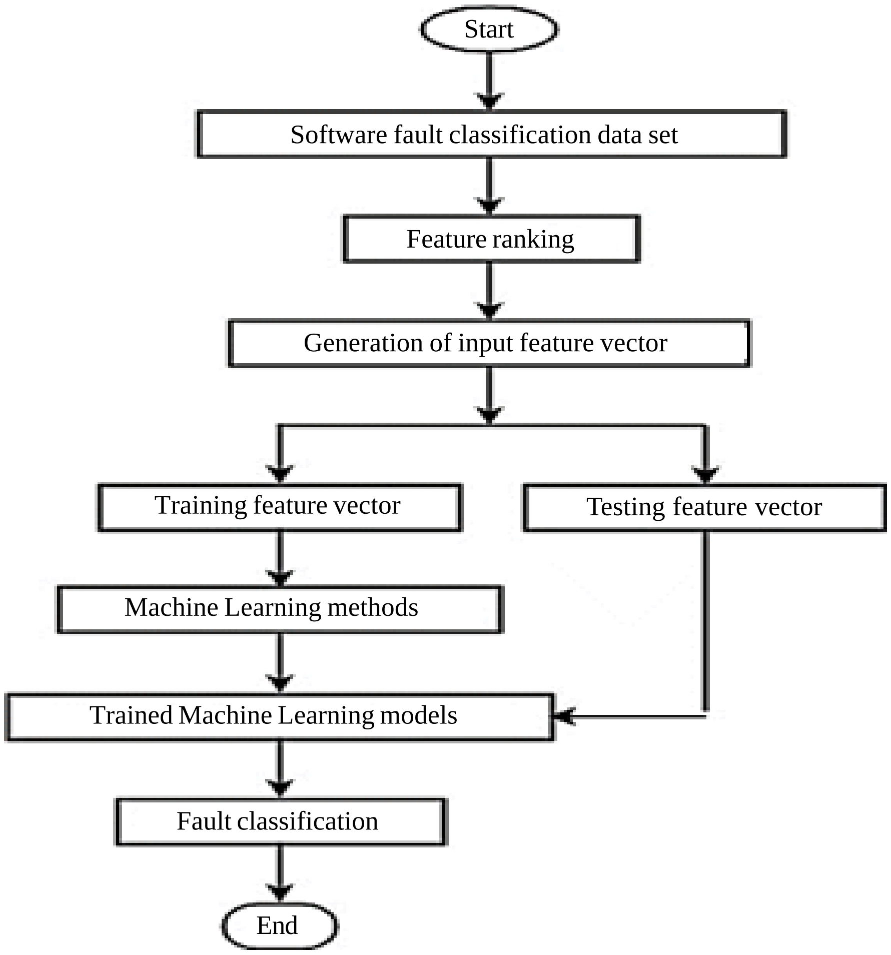


Figure 1. The proposed method for software fault classification

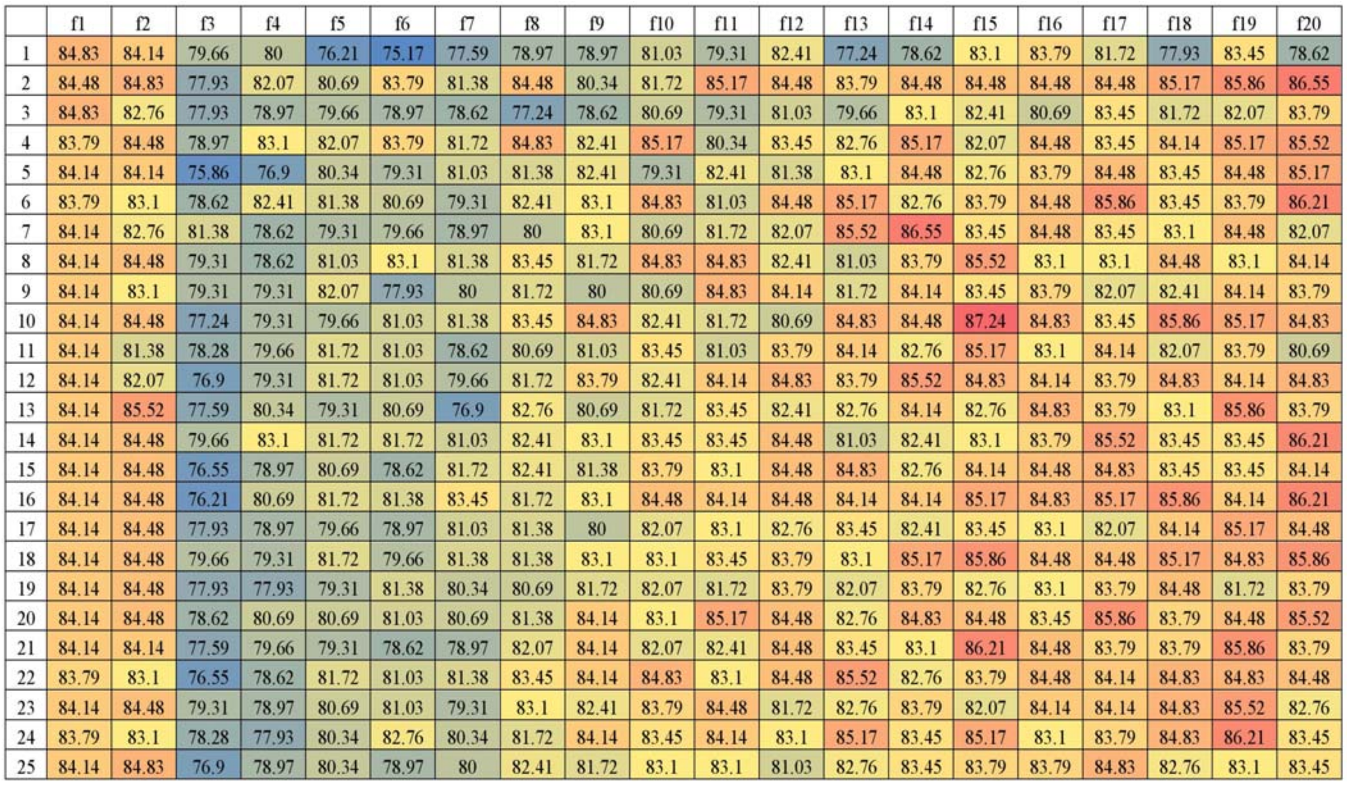
5. Results and Discussion

The experimental study is carried out on i5 core processor with RAM 4 GB and Ubuntu OS. 20 iteration of each experiment is carried out, and average values are taken for the demonstration of results. 10 cross-validation method is used for experimental purpose. Open source Xalan project were investigated. Only two classes are considered i.e. 0 and 1. Other classes are not considered due to less availability of number of samples. 20 features are considered for the classification. Lasso (Least Absolute Shrinkage and Selection Operator) and Ridge ranking algorithms are used for feature ranking. Lasso and Ridge performs L1 and L2 regularization respectively. Table 1 shows the order of feature by Lasso and Ridge feature ranking algorithms.

Firstly, feature ranking carried out by Lasso. Fig. 2-6 demonstrates the effect of Lasso with different classifiers like Bagging with Decision Tree (B_DT), Bagging with K-Nearest Neighbor (B_KNN), Adaboost with Support Vector Machine (A_SVM), Extremely randomized trees (ET), and Random Forest (RF). This ranking allows selecting topmost feature in feature set. In each iteration, the feature set is increased by one. It enables to test the importance of feature and its effect on the overall accuracy of the system. These features are presented as an input to the classifier. Also, the impact of an estimator is tested by increasing the estimator by 1. This process allows finding the optimum combination of estimator with a feature set for the classifier.

Rank	**f1**	**f2**	**f3**	**f4**	**f5**	**f6**	**f7**	**f8**	**f9**	**f10**	**f11**	**f12**	**f13**	**f14**	**f15**	**f16**	**f17**	**f18**	**f19**	**f20**
Lasso	moa	cbm	wmc	npm	mfa	ca	cbo	max_cc	rfc	amc	noc	loc	lcom3	lcom	ic	dit	dam	ce	cam	avg_cc
Ridge	mfa	cam	ic	moa	dam	dit	cbm	wmc	avg_cc	npm	cbo	ca	max_cc	ce	rfc	noc	lcom3	amc	loc	lcom

Table 2. Order of feature by Lasso and Ridge feature ranking algorithms



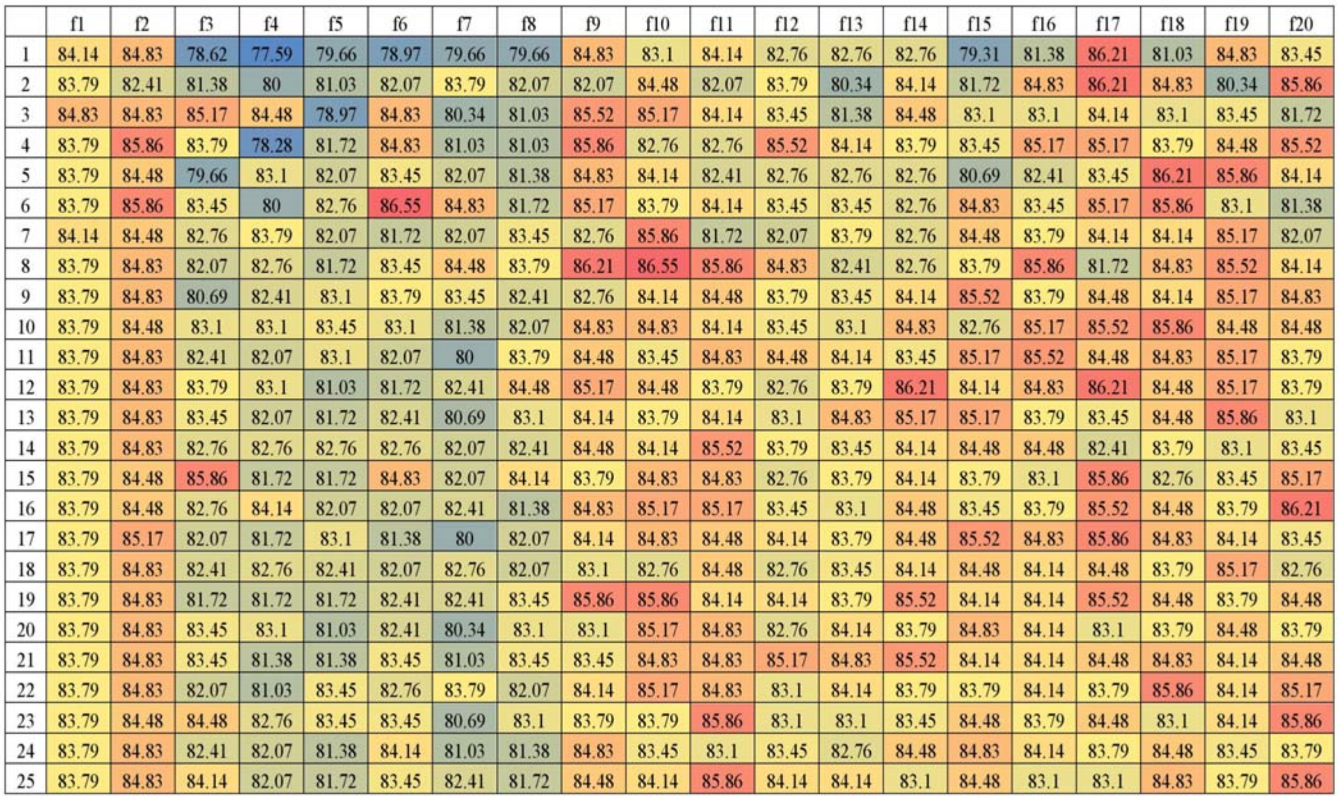
	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	84.83	84.14	79.66	80	76.21	75.17	77.59	78.97	78.97	81.03	79.31	82.41	77.24	78.62	83.1	83.79	81.72	77.93	83.45	78.62
2	84.48	84.83	77.93	82.07	80.69	83.79	81.38	84.48	80.34	81.72	85.17	84.48	83.79	84.48	84.48	84.48	84.48	85.17	85.86	86.55
3	84.83	82.76	77.93	78.97	79.66	78.97	78.62	77.24	78.62	80.69	79.31	81.03	79.66	83.1	82.41	80.69	83.45	81.72	82.07	83.79
4	83.79	84.48	78.97	83.1	82.07	83.79	81.72	84.83	82.41	85.17	80.34	83.45	82.76	85.17	82.07	84.48	83.45	84.14	85.17	85.52
5	84.14	84.14	75.86	76.9	80.34	79.31	81.03	81.38	82.41	79.31	82.41	81.38	83.1	84.48	82.76	83.79	84.48	83.45	84.48	85.17
6	83.79	83.1	78.62	82.41	81.38	80.69	79.31	82.41	83.1	84.83	81.03	84.48	85.17	82.76	83.79	84.48	85.86	83.45	83.79	86.21
7	84.14	82.76	81.38	78.62	79.31	79.66	78.97	80	83.1	80.69	81.72	82.07	85.52	86.55	83.45	84.48	83.45	83.1	84.48	82.07
8	84.14	84.48	79.31	78.62	81.03	83.1	81.38	83.45	81.72	84.83	84.83	82.41	81.03	83.79	85.52	83.1	83.1	84.48	83.1	84.14
9	84.14	83.1	79.31	79.31	82.07	77.93	80	81.72	80	80.69	84.83	84.14	81.72	84.14	83.45	83.79	82.07	82.41	84.14	83.79
10	84.14	84.48	77.24	79.31	79.66	81.03	81.38	83.45	84.83	82.41	81.72	80.69	84.83	84.48	87.24	84.83	83.45	85.86	85.17	84.83
11	84.14	81.38	78.28	79.66	81.72	81.03	78.62	80.69	81.03	83.45	81.03	83.79	84.14	82.76	85.17	83.1	84.14	82.07	83.79	80.69
12	84.14	82.07	76.9	79.31	81.72	81.03	79.66	81.72	83.79	82.41	84.14	84.83	83.79	85.52	84.83	84.14	83.79	84.83	84.14	84.83
13	84.14	85.52	77.59	80.34	79.31	80.69	76.9	82.76	80.69	81.72	83.45	82.41	82.76	84.14	82.76	84.83	83.79	83.1	85.86	83.79
14	84.14	84.48	79.66	83.1	81.72	81.72	81.03	82.41	83.1	83.45	83.45	84.48	81.03	82.41	83.1	83.79	85.52	83.45	83.45	86.21
15	84.14	84.48	76.55	78.97	80.69	78.62	81.72	82.41	81.38	83.79	83.1	84.48	84.83	82.76	84.14	84.48	84.83	83.45	83.45	84.14
16	84.14	84.48	76.21	80.69	81.72	81.38	83.45	81.72	83.1	84.48	84.14	84.48	84.14	84.14	85.17	84.83	85.17	85.86	84.14	86.21
17	84.14	84.48	77.93	78.97	79.66	78.97	81.03	81.38	80	82.07	83.1	82.76	83.45	82.41	83.45	83.1	82.07	84.14	85.17	84.48
18	84.14	84.48	79.66	79.31	81.72	79.66	81.38	81.38	83.1	83.1	83.45	83.79	83.1	85.17	85.86	84.48	84.48	85.17	84.83	85.86
19	84.14	84.48	77.93	77.93	79.31	81.38	80.34	80.69	81.72	82.07	81.72	83.79	82.07	83.79	82.76	83.1	83.79	84.48	81.72	83.79
20	84.14	84.48	78.62	80.69	80.69	81.03	80.69	81.38	84.14	83.1	85.17	84.48	82.76	84.83	84.48	83.45	85.86	83.79	84.48	85.52
21	84.14	84.14	77.59	79.66	79.31	78.62	78.97	82.07	84.14	82.07	82.41	84.48	83.45	83.1	86.21	84.48	83.79	83.79	85.86	83.79
22	83.79	83.1	76.55	78.62	81.72	81.03	81.38	83.45	84.14	84.83	83.1	84.48	85.52	82.76	83.79	84.48	84.14	84.83	84.83	84.48
23	84.14	84.48	79.31	78.97	80.69	81.03	79.31	83.1	82.41	83.79	84.48	81.72	82.76	83.79	82.07	84.14	84.14	84.83	85.52	82.76
24	83.79	83.1	78.28	77.93	80.34	82.76	80.34	81.72	84.14	83.45	84.14	83.1	85.17	83.45	85.17	83.1	83.79	84.83	86.21	83.45
25	84.14	84.83	76.9	78.97	80.34	78.97	80	82.41	81.72	83.1	83.1	81.03	82.76	83.45	83.79	83.79	84.83	82.76	83.1	83.45

Figure 2. Heat map of bagging with decision tree classifier using lasso feature ranking algorithm

Figure 2 demonstrates the results of B_DT classifier. It has been observed that single topmost feature is unable to provide the efficient solution. Also, overall accuracy remains almost constant with increase in the estimator for first two features. At feature set 3, the overall accuracy significantly decreased. After that with an increase in feature set and estimator, overall accuracy significantly increased. It is also noticed that the after feature 10 for all estimators, there is no significant change in the accuracy and almost remain constant. With feature set 15 and estimator 10, accuracy of achieved is > 87%.

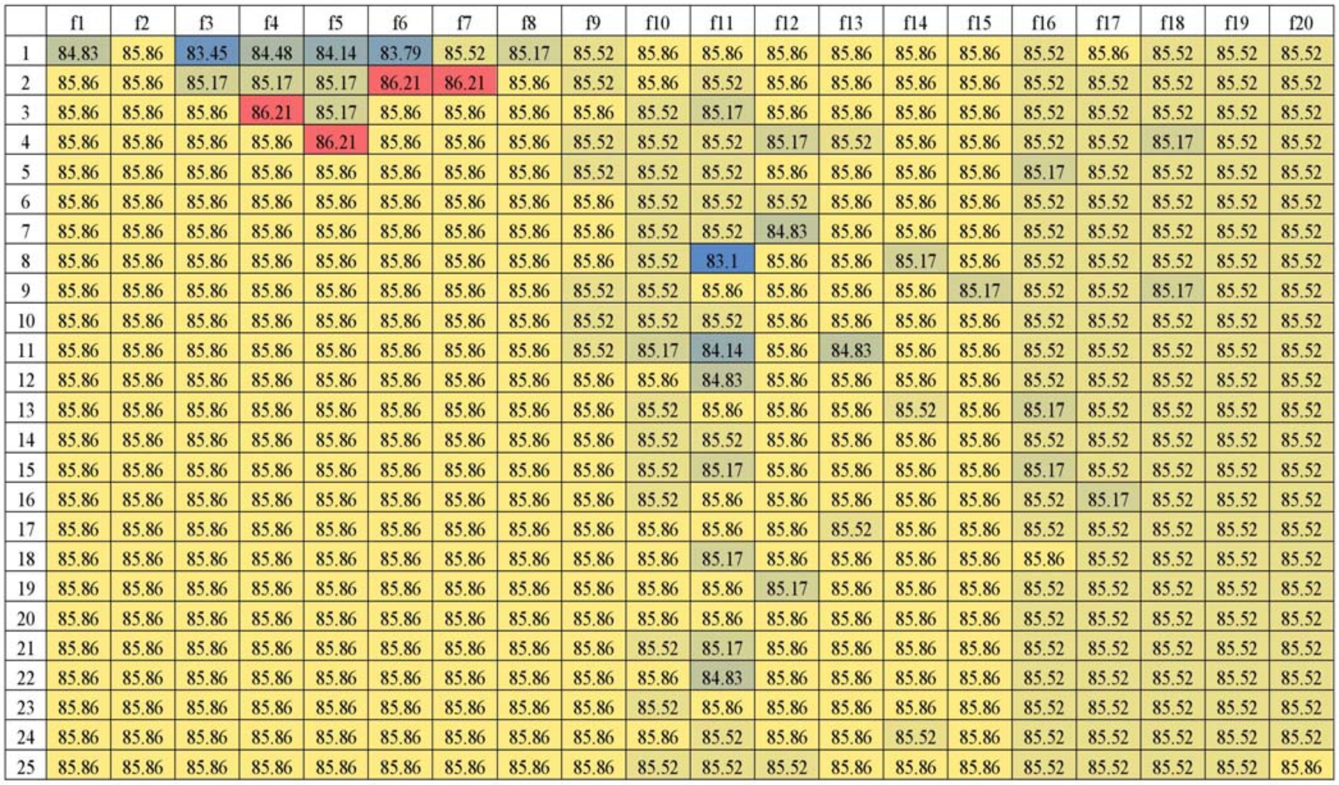
Figure 3 demonstrates the results of B_KNN classifier. In this case, also, the single topmost feature is unable to provide the efficient solution. Also, overall accuracy remains almost constant with increase in the estimator. For different combination of feature set and estimator, accuracy remain > 80%. However, for feature set 6, and estimator 6 provides accuracy 86.55%. It is also noticed that there different combination of feature set and estimator are available which provides accuracy 86.55%.

Figure 4 shows the results of A_SVM classifier. In this case, also, the single topmost feature is able to provide the efficient



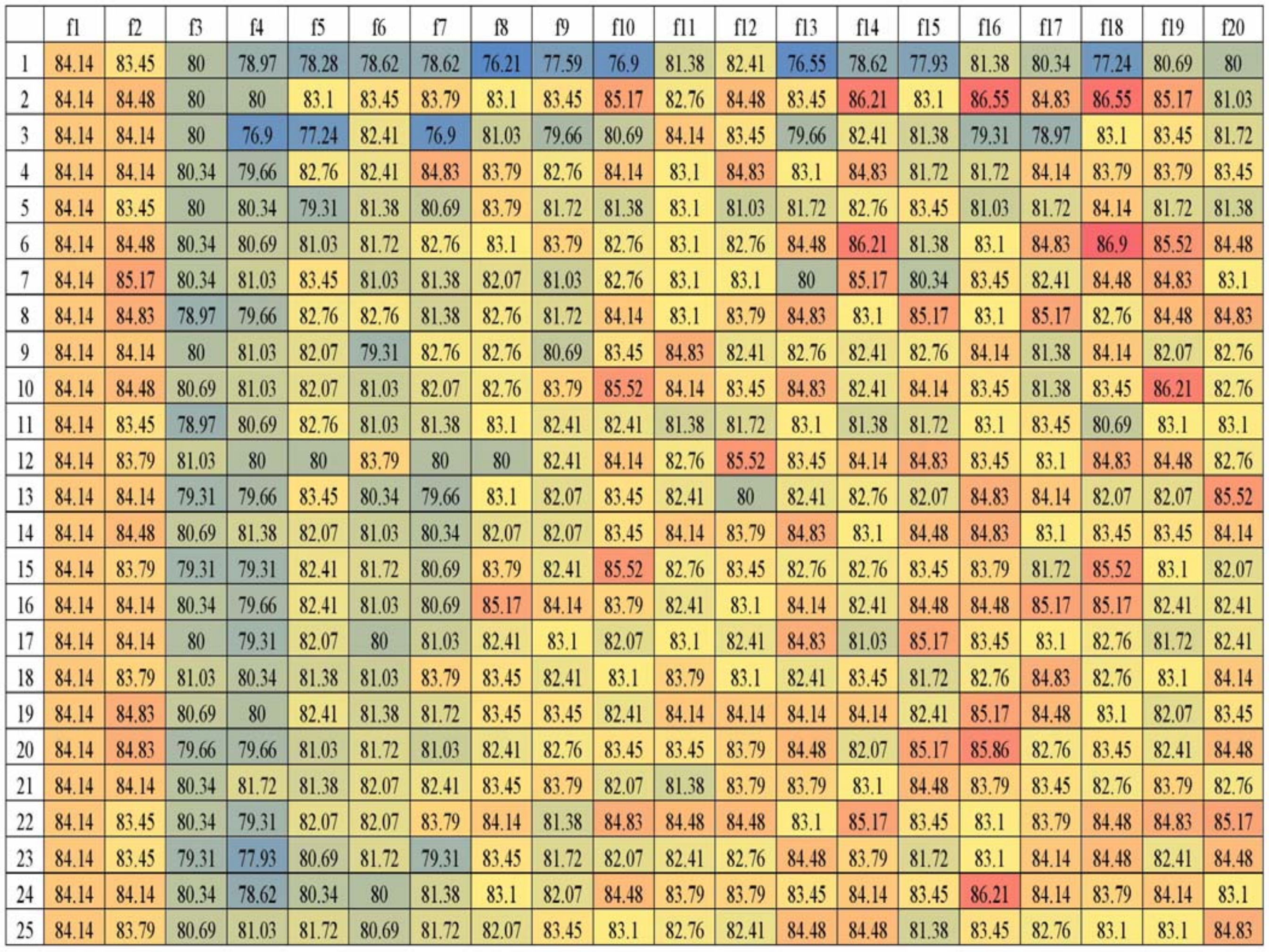
	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	84.14	84.83	78.62	77.59	79.66	78.97	79.66	79.66	84.83	83.1	84.14	82.76	82.76	82.76	79.31	81.38	86.21	81.03	84.83	83.45
2	83.79	82.41	81.38	80	81.03	82.07	83.79	82.07	82.07	84.48	82.07	83.79	80.34	84.14	81.72	84.83	86.21	84.83	80.34	85.86
3	84.83	84.83	85.17	84.48	78.97	84.83	80.34	81.03	85.52	85.17	84.14	83.45	81.38	84.48	83.1	83.1	84.14	83.1	83.45	81.72
4	83.79	85.86	83.79	78.28	81.72	84.83	81.03	81.03	85.86	82.76	82.76	85.52	84.14	83.79	83.45	85.17	85.17	83.79	84.48	85.52
5	83.79	84.48	79.66	83.1	82.07	83.45	82.07	81.38	84.83	84.14	82.41	82.76	82.76	82.76	80.69	82.41	83.45	86.21	85.86	84.14
6	83.79	85.86	83.45	80	82.76	86.55	84.83	81.72	85.17	83.79	84.14	83.45	83.45	82.76	84.83	83.45	85.17	85.86	83.1	81.38
7	84.14	84.48	82.76	83.79	82.07	81.72	82.07	83.45	82.76	85.86	81.72	82.07	83.79	82.76	84.48	83.79	84.14	84.14	85.17	82.07
8	83.79	84.83	82.07	82.76	81.72	83.45	84.48	83.79	86.21	86.55	85.86	84.83	82.41	82.76	83.79	85.86	81.72	84.83	85.52	84.14
9	83.79	84.83	80.69	82.41	83.1	83.79	83.45	82.41	82.76	84.14	84.48	83.79	83.45	84.14	85.52	83.79	84.48	84.14	85.17	84.83
10	83.79	84.48	83.1	83.1	83.45	83.1	81.38	82.07	84.83	84.83	84.14	83.45	83.1	84.83	82.76	85.17	85.52	85.86	84.48	84.48
11	83.79	84.83	82.41	82.07	83.1	82.07	80	83.79	84.48	83.45	84.83	84.48	84.14	83.45	85.17	85.52	84.48	84.83	85.17	83.79
12	83.79	84.83	83.79	83.1	81.03	81.72	82.41	84.48	85.17	84.48	83.79	82.76	83.79	86.21	84.14	84.83	86.21	84.48	85.17	83.79
13	83.79	84.83	83.45	82.07	81.72	82.41	80.69	83.1	84.14	83.79	84.14	83.1	84.83	85.17	85.17	83.79	83.45	84.48	85.86	83.1
14	83.79	84.83	82.76	82.76	82.76	82.76	82.07	82.41	84.48	84.14	85.52	83.79	83.45	84.14	84.48	84.48	82.41	83.79	83.1	83.45
15	83.79	84.48	85.86	81.72	81.72	84.83	82.07	84.14	83.79	84.83	84.83	82.76	83.79	84.14	83.79	83.1	85.86	82.76	83.45	85.17
16	83.79	84.48	82.76	84.14	82.07	82.07	82.41	81.38	84.83	85.17	85.17	83.45	83.1	84.48	83.45	83.79	85.52	84.48	83.79	86.21
17	83.79	85.17	82.07	81.72	83.1	81.38	80	82.07	84.14	84.83	84.48	84.14	83.79	84.48	85.52	84.83	85.86	84.83	84.14	83.45
18	83.79	84.83	82.41	82.76	82.41	82.07	82.76	82.07	83.1	82.76	84.48	82.76	83.45	84.14	84.48	84.14	84.48	83.79	85.17	82.76
19	83.79	84.83	81.72	81.72	81.72	82.41	82.41	83.45	85.86	85.86	84.14	84.14	83.79	85.52	84.14	84.14	85.52	84.48	83.79	84.48
20	83.79	84.83	83.45	83.1	81.03	82.41	80.34	83.1	83.1	85.17	84.83	82.76	84.14	83.79	84.83	84.14	83.1	83.79	84.48	83.79
21	83.79	84.83	83.45	81.38	81.38	83.45	81.03	83.45	83.45	84.83	84.83	85.17	84.83	85.52	84.14	84.14	84.48	84.83	84.14	84.48
22	83.79	84.83	82.07	81.03	83.45	82.76	83.79	82.07	84.14	85.17	84.83	83.1	84.14	83.79	83.79	84.14	83.79	85.86	84.14	85.17
23	83.79	84.48	84.48	82.76	83.45	83.45	80.69	83.1	83.79	83.79	85.86	83.1	83.1	83.45	84.48	83.79	84.48	83.1	84.14	85.86
24	83.79	84.83	82.41	82.07	81.38	84.14	81.03	81.38	84.83	83.45	83.1	83.45	82.76	84.48	84.83	84.14	83.79	84.48	83.45	83.79
25	83.79	84.83	84.14	82.07	81.72	83.45	82.41	81.72	84.48	84.14	85.86	84.14	84.14	83.1	84.48	83.1	83.1	84.83	83.79	85.86

Figure 3. Heat map of bagging with K-nearest neighbor classifier using lasso feature ranking algorithm



	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	84.83	85.86	83.45	84.48	84.14	83.79	85.52	85.17	85.52	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.52	85.52	85.52
2	85.86	85.86	85.17	85.17	85.17	86.21	86.21	85.86	85.52	85.86	85.52	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
3	85.86	85.86	85.86	86.21	85.17	85.86	85.86	85.86	85.86	85.52	85.17	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
4	85.86	85.86	85.86	85.86	86.21	85.86	85.86	85.86	85.52	85.52	85.52	85.17	85.52	85.86	85.86	85.52	85.52	85.17	85.52	85.52
5	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.86	85.86	85.86	85.86	85.17	85.52	85.52	85.52	85.52
6	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
7	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.52	84.83	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
8	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	83.1	85.86	85.86	85.17	85.86	85.52	85.52	85.52	85.52	85.52
9	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.86	85.86	85.86	85.86	85.17	85.52	85.52	85.17	85.52	85.52
10	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
11	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.17	84.14	85.86	84.83	85.86	85.86	85.52	85.52	85.52	85.52	85.52
12	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	84.83	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
13	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.86	85.52	85.86	85.17	85.52	85.52	85.52	85.52
14	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
15	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.17	85.86	85.86	85.86	85.86	85.17	85.52	85.52	85.52	85.52
16	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.86	85.86	85.86	85.52	85.17	85.52	85.52	85.52
17	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.52	85.52	85.52	85.52	85.52
18	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.17	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52
19	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.17	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
20	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
21	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.17	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
22	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	84.83	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
23	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.52
24	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.52	85.86	85.52	85.52	85.52	85.52	85.52
25	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.86

Figure 4. Heat map of adaboost with support vector machine classifier using lasso feature ranking algorithm



	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	84.14	83.45	80	78.97	78.28	78.62	78.62	76.21	77.59	76.9	81.38	82.41	76.55	78.62	77.93	81.38	80.34	77.24	80.69	80
2	84.14	84.48	80	80	83.1	83.45	83.79	83.1	83.45	85.17	82.76	84.48	83.45	86.21	83.1	86.55	84.83	86.55	85.17	81.03
3	84.14	84.14	80	76.9	77.24	82.41	76.9	81.03	79.66	80.69	84.14	83.45	79.66	82.41	81.38	79.31	78.97	83.1	83.45	81.72
4	84.14	84.14	80.34	79.66	82.76	82.41	84.83	83.79	82.76	84.14	83.1	84.83	83.1	84.83	81.72	81.72	84.14	83.79	83.79	83.45
5	84.14	83.45	80	80.34	79.31	81.38	80.69	83.79	81.72	81.38	83.1	81.03	81.72	82.76	83.45	81.03	81.72	84.14	81.72	81.38
6	84.14	84.48	80.34	80.69	81.03	81.72	82.76	83.1	83.79	82.76	83.1	82.76	84.48	86.21	81.38	83.1	84.83	86.9	85.52	84.48
7	84.14	85.17	80.34	81.03	83.45	81.03	81.38	82.07	81.03	82.76	83.1	83.1	80	85.17	80.34	83.45	82.41	84.48	84.83	83.1
8	84.14	84.83	78.97	79.66	82.76	82.76	81.38	82.76	81.72	84.14	83.1	83.79	84.83	83.1	85.17	83.1	85.17	82.76	84.48	84.83
9	84.14	84.14	80	81.03	82.07	79.31	82.76	82.76	80.69	83.45	84.83	82.41	82.76	82.41	82.76	84.14	81.38	84.14	82.07	82.76
10	84.14	84.48	80.69	81.03	82.07	81.03	82.07	82.76	83.79	85.52	84.14	83.45	84.83	82.41	84.14	83.45	81.38	83.45	86.21	82.76
11	84.14	83.45	78.97	80.69	82.76	81.03	81.38	83.1	82.41	82.41	81.38	81.72	83.1	81.38	81.72	83.1	83.45	80.69	83.1	83.1
12	84.14	83.79	81.03	80	80	83.79	80	80	82.41	84.14	82.76	85.52	83.45	84.14	84.83	83.45	83.1	84.83	84.48	82.76
13	84.14	84.14	79.31	79.66	83.45	80.34	79.66	83.1	82.07	83.45	82.41	80	82.41	82.76	82.07	84.83	84.14	82.07	82.07	85.52
14	84.14	84.48	80.69	81.38	82.07	81.03	80.34	82.07	82.07	83.45	84.14	83.79	84.83	83.1	84.48	84.83	83.1	83.45	83.45	84.14
15	84.14	83.79	79.31	79.31	82.41	81.72	80.69	83.79	82.41	85.52	82.76	83.45	82.76	82.76	83.45	83.79	81.72	85.52	83.1	82.07
16	84.14	84.14	80.34	79.66	82.41	81.03	80.69	85.17	84.14	83.79	82.41	83.1	84.14	82.41	84.48	84.48	85.17	85.17	82.41	82.41
17	84.14	84.14	80	79.31	82.07	80	81.03	82.41	83.1	82.07	83.1	82.41	84.83	81.03	85.17	83.45	83.1	82.76	81.72	82.41
18	84.14	83.79	81.03	80.34	81.38	81.03	83.79	83.45	82.41	83.1	83.79	83.1	82.41	83.45	81.72	82.76	84.83	82.76	83.1	84.14
19	84.14	84.83	80.69	80	82.41	81.38	81.72	83.45	83.45	82.41	84.14	84.14	84.14	84.14	82.41	85.17	84.48	83.1	82.07	83.45
20	84.14	84.83	79.66	79.66	81.03	81.72	81.03	82.41	82.76	83.45	83.45	83.79	84.48	82.07	85.17	85.86	82.76	83.45	82.41	84.48
21	84.14	84.14	80.34	81.72	81.38	82.07	82.41	83.45	83.79	82.07	81.38	83.79	83.79	83.1	84.48	83.79	83.45	82.76	83.79	82.76
22	84.14	83.45	80.34	79.31	82.07	82.07	83.79	84.14	81.38	84.83	84.48	84.48	83.1	85.17	83.45	83.1	83.79	84.48	84.83	85.17
23	84.14	83.45	79.31	77.93	80.69	81.72	79.31	83.45	81.72	82.07	82.41	82.76	84.48	83.79	81.72	83.1	84.14	84.48	82.41	84.48
24	84.14	84.14	80.34	78.62	80.34	80	81.38	83.1	82.07	84.48	83.79	83.79	83.45	84.14	83.45	86.21	84.14	83.79	84.14	83.1
25	84.14	83.79	80.69	81.03	81.72	80.69	81.72	82.07	83.45	83.1	82.76	82.41	84.48	84.48	81.38	83.45	82.76	83.1	83.1	84.83

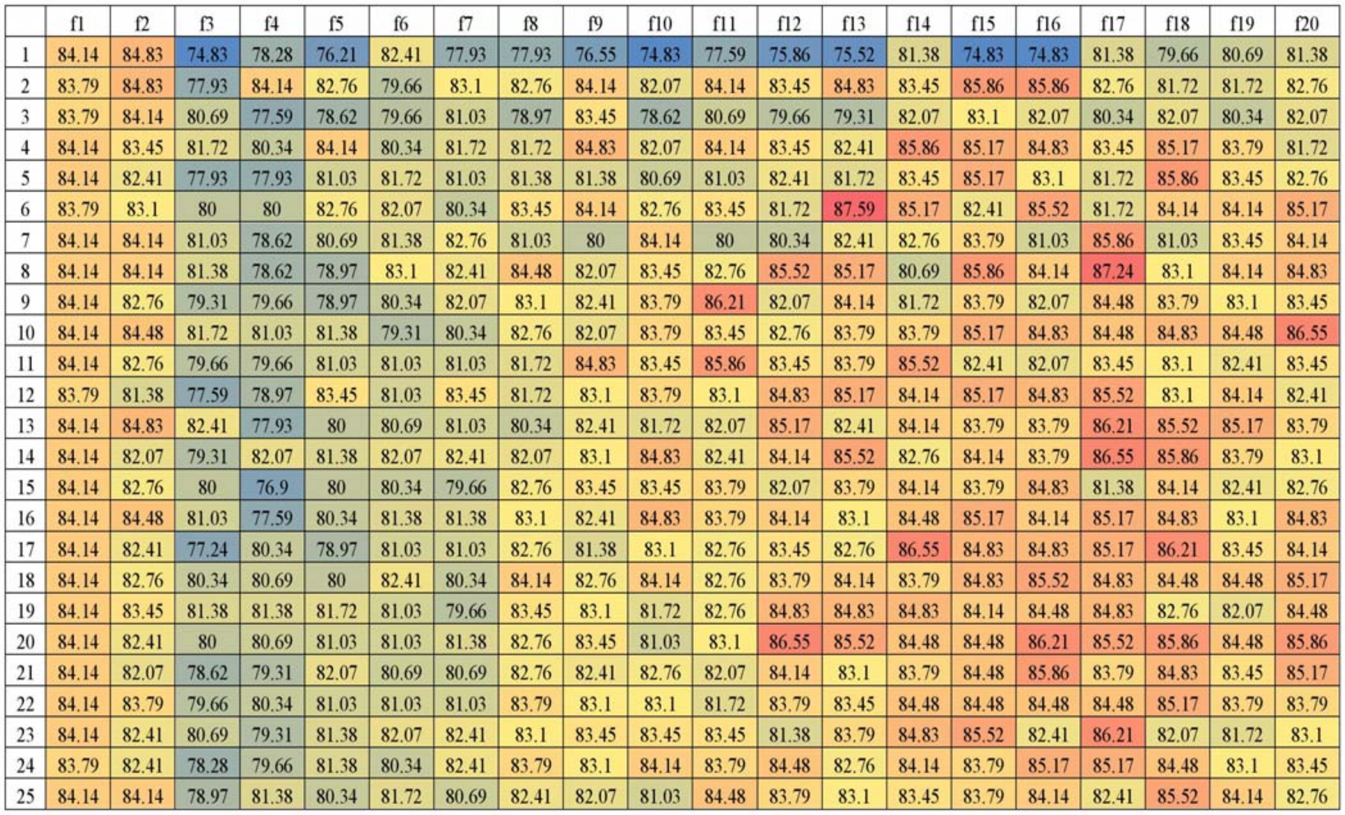
Figure 5. Heat map of extremely randomized trees classifier using lasso feature ranking algorithm

solution. Also, overall accuracy remains almost constant with increase in the estimator. For different combination of feature set and estimator, accuracy remain > 83%. However, for feature set 4, and estimator 3 onwards provides accuracy 86.21%.

Figure 5 and 6 illustrates the results of ET and RF classifier. In this case, also, the single topmost feature is unable to provide the efficient solution. It shows the similar behavior that of B_DT. For ET classifier, with feature set 18 and estimator 6, accuracy of achieved is 86.9%. It is also noticed that there different combination of feature set and estimator are available which provides accuracy > 86%. For RF classifier, with feature set 13 and estimator 6, accuracy of achieved > 87.5%. It is also noticed that there different combination of feature set and estimator are available which provides accuracy > 86%.

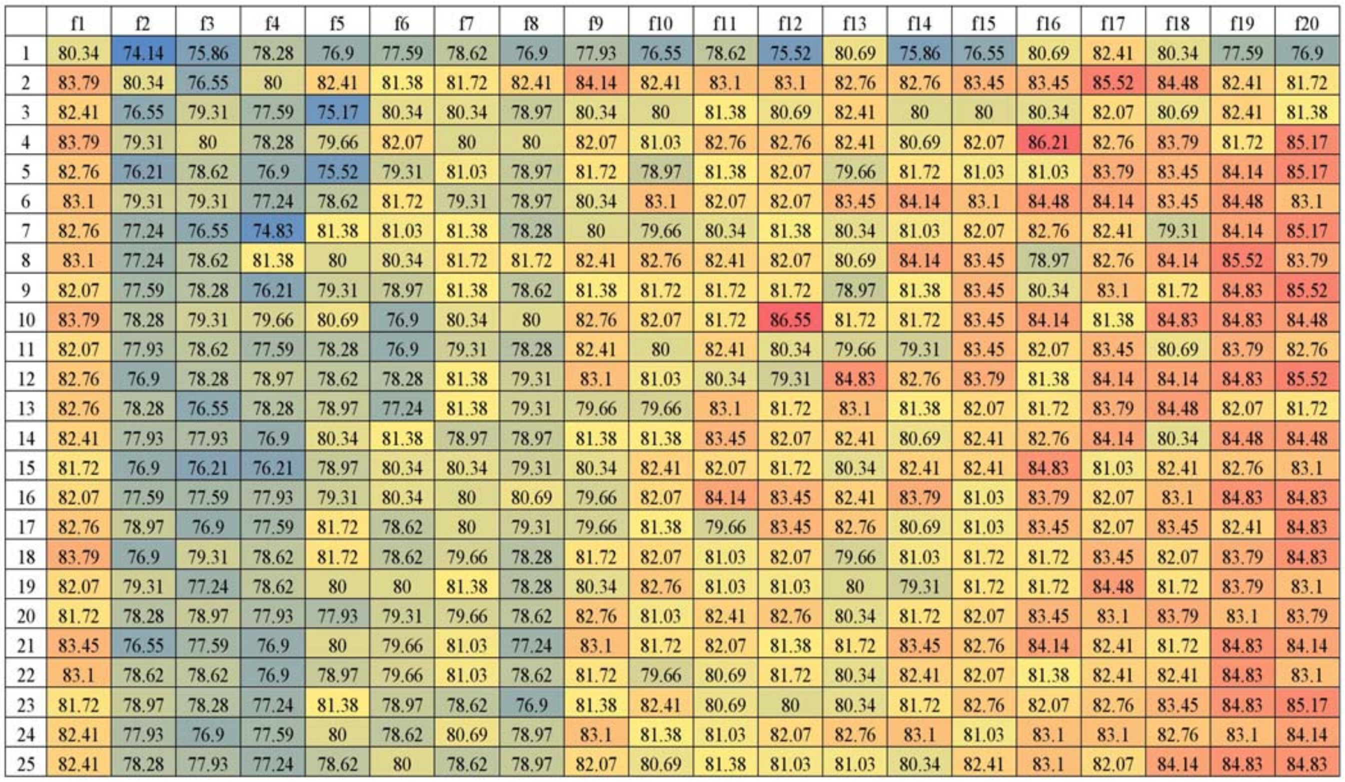
Figure 7-11 demonstrates the effect of Ridge feature ranking algorithm with different classifiers like B_DT, B_KNN, A_SVM, ET, and RF. Fig. 7 illustrates the results of B_DT. Its accuracy is less than B_DT with Lasso for feature set 1 for all estimators. Also, its accuracy decreased from feature set 2 onwards up till feature set 4. The highest accuracy of 86.55% is achieved for the combination of features set 12 and estimator 10. This accuracy is slightly lower than B_DT with Lasso. From feature set 9 onwards accuracy is constantly increasing with different combinations of feature sets and estimators.

Figure 8 shows the results of B_KNN. Its accuracy is better than B_KNN with Lasso for feature set 1 for all estimators. Also, its accuracy remains around 85% for feature set 1 with different estimators. The highest accuracy of 87.24% is achieved for the combination of features set 16 and estimator 2. This accuracy is slightly higher than B_KNN with Lasso. For different combination of feature sets and estimator, maximum time accuracy remains between 82-84%.



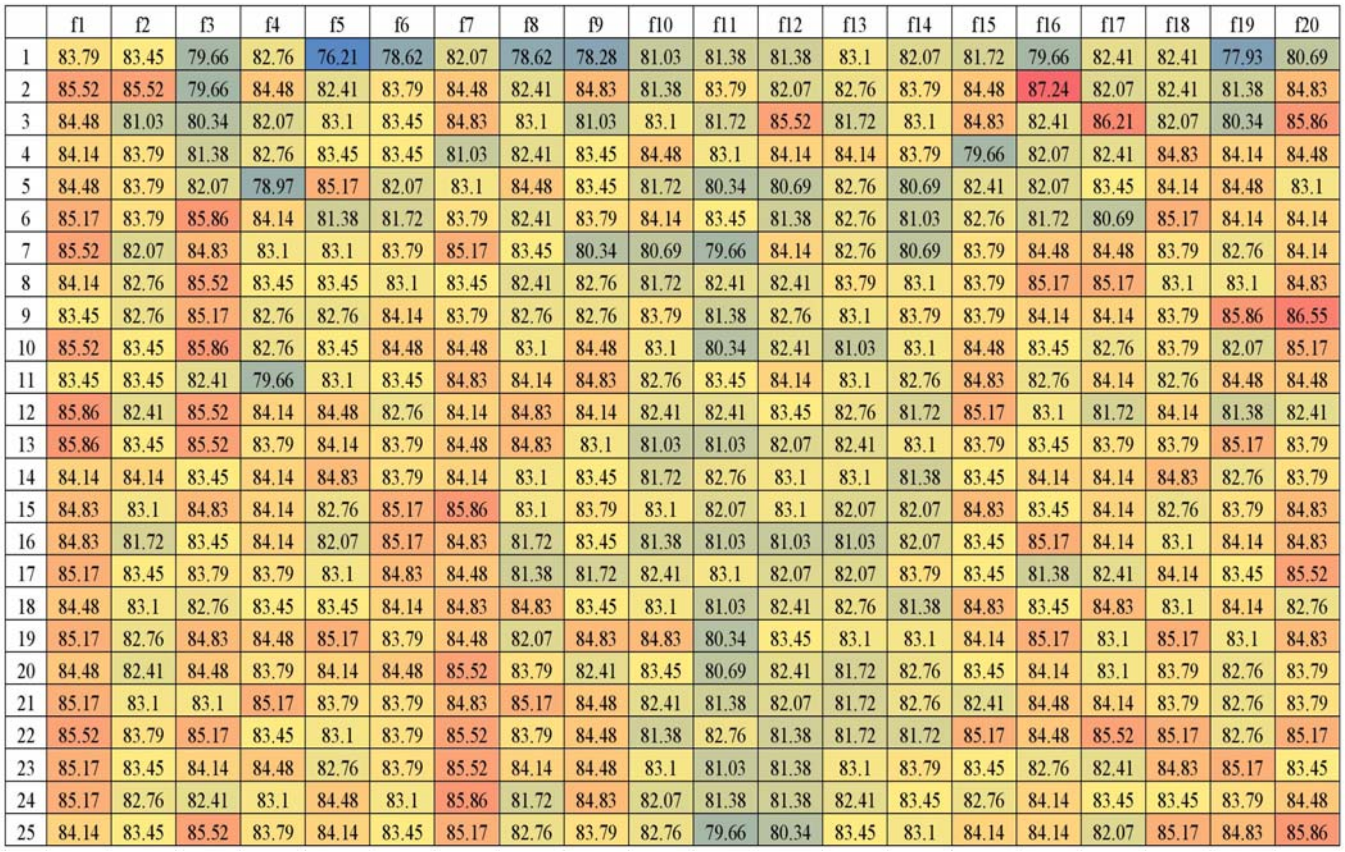
	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	84.14	84.83	74.83	78.28	76.21	82.41	77.93	77.93	76.55	74.83	77.59	75.86	75.52	81.38	74.83	74.83	81.38	79.66	80.69	81.38
2	83.79	84.83	77.93	84.14	82.76	79.66	83.1	82.76	84.14	82.07	84.14	83.45	84.83	83.45	85.86	85.86	82.76	81.72	81.72	82.76
3	83.79	84.14	80.69	77.59	78.62	79.66	81.03	78.97	83.45	78.62	80.69	79.66	79.31	82.07	83.1	82.07	80.34	82.07	80.34	82.07
4	84.14	83.45	81.72	80.34	84.14	80.34	81.72	81.72	84.83	82.07	84.14	83.45	82.41	85.86	85.17	84.83	83.45	85.17	83.79	81.72
5	84.14	82.41	77.93	77.93	81.03	81.72	81.03	81.38	81.38	80.69	81.03	82.41	81.72	83.45	85.17	83.1	81.72	85.86	83.45	82.76
6	83.79	83.1	80	80	82.76	82.07	80.34	83.45	84.14	82.76	83.45	81.72	87.59	85.17	82.41	85.52	81.72	84.14	84.14	85.17
7	84.14	84.14	81.03	78.62	80.69	81.38	82.76	81.03	80	84.14	80	80.34	82.41	82.76	83.79	81.03	85.86	81.03	83.45	84.14
8	84.14	84.14	81.38	78.62	78.97	83.1	82.41	84.48	82.07	83.45	82.76	85.52	85.17	80.69	85.86	84.14	87.24	83.1	84.14	84.83
9	84.14	82.76	79.31	79.66	78.97	80.34	82.07	83.1	82.41	83.79	86.21	82.07	84.14	81.72	83.79	82.07	84.48	83.79	83.1	83.45
10	84.14	84.48	81.72	81.03	81.38	79.31	80.34	82.76	82.07	83.79	83.45	82.76	83.79	83.79	85.17	84.83	84.48	84.83	84.48	86.55
11	84.14	82.76	79.66	79.66	81.03	81.03	81.03	81.72	84.83	83.45	85.86	83.45	83.79	85.52	82.41	82.07	83.45	83.1	82.41	83.45
12	83.79	81.38	77.59	78.97	83.45	81.03	83.45	81.72	83.1	83.79	83.1	84.83	85.17	84.14	85.17	84.83	85.52	83.1	84.14	82.41
13	84.14	84.83	82.41	77.93	80	80.69	81.03	80.34	82.41	81.72	82.07	85.17	82.41	84.14	83.79	83.79	86.21	85.52	85.17	83.79
14	84.14	82.07	79.31	82.07	81.38	82.07	82.41	82.07	83.1	84.83	82.41	84.14	85.52	82.76	84.14	83.79	86.55	85.86	83.79	83.1
15	84.14	82.76	80	76.9	80	80.34	79.66	82.76	83.45	83.45	83.79	82.07	83.79	84.14	83.79	84.83	81.38	84.14	82.41	82.76
16	84.14	84.48	81.03	77.59	80.34	81.38	81.38	83.1	82.41	84.83	83.79	84.14	83.1	84.48	85.17	84.14	85.17	84.83	83.1	84.83
17	84.14	82.41	77.24	80.34	78.97	81.03	81.03	82.76	81.38	83.1	82.76	83.45	82.76	86.55	84.83	84.83	85.17	86.21	83.45	84.14
18	84.14	82.76	80.34	80.69	80	82.41	80.34	84.14	82.76	84.14	82.76	83.79	84.14	83.79	84.83	85.52	84.83	84.48	84.48	85.17
19	84.14	83.45	81.38	81.38	81.72	81.03	79.66	83.45	83.1	81.72	82.76	84.83	84.83	84.83	84.14	84.48	84.83	82.76	82.07	84.48
20	84.14	82.41	80	80.69	81.03	81.03	81.38	82.76	83.45	81.03	83.1	86.55	85.52	84.48	84.48	86.21	85.52	85.86	84.48	85.86
21	84.14	82.07	78.62	79.31	82.07	80.69	80.69	82.76	82.41	82.76	82.07	84.14	83.1	83.79	84.48	85.86	83.79	84.83	83.45	85.17
22	84.14	83.79	79.66	80.34	81.03	81.03	81.03	83.79	83.1	83.1	81.72	83.79	83.45	84.48	84.48	84.48	84.48	85.17	83.79	83.79
23	84.14	82.41	80.69	79.31	81.38	82.07	82.41	83.1	83.45	83.45	83.45	81.38	83.79	84.83	85.52	82.41	86.21	82.07	81.72	83.1
24	83.79	82.41	78.28	79.66	81.38	80.34	82.41	83.79	83.1	84.14	83.79	84.48	82.76	84.14	83.79	85.17	85.17	84.48	83.1	83.45
25	84.14	84.14	78.97	81.38	80.34	81.72	80.69	82.41	82.07	81.03	84.48	83.79	83.1	83.45	83.79	84.14	82.41	85.52	84.14	82.76

Figure 6. Heat map of random forest classifier using lasso feature ranking algorithm



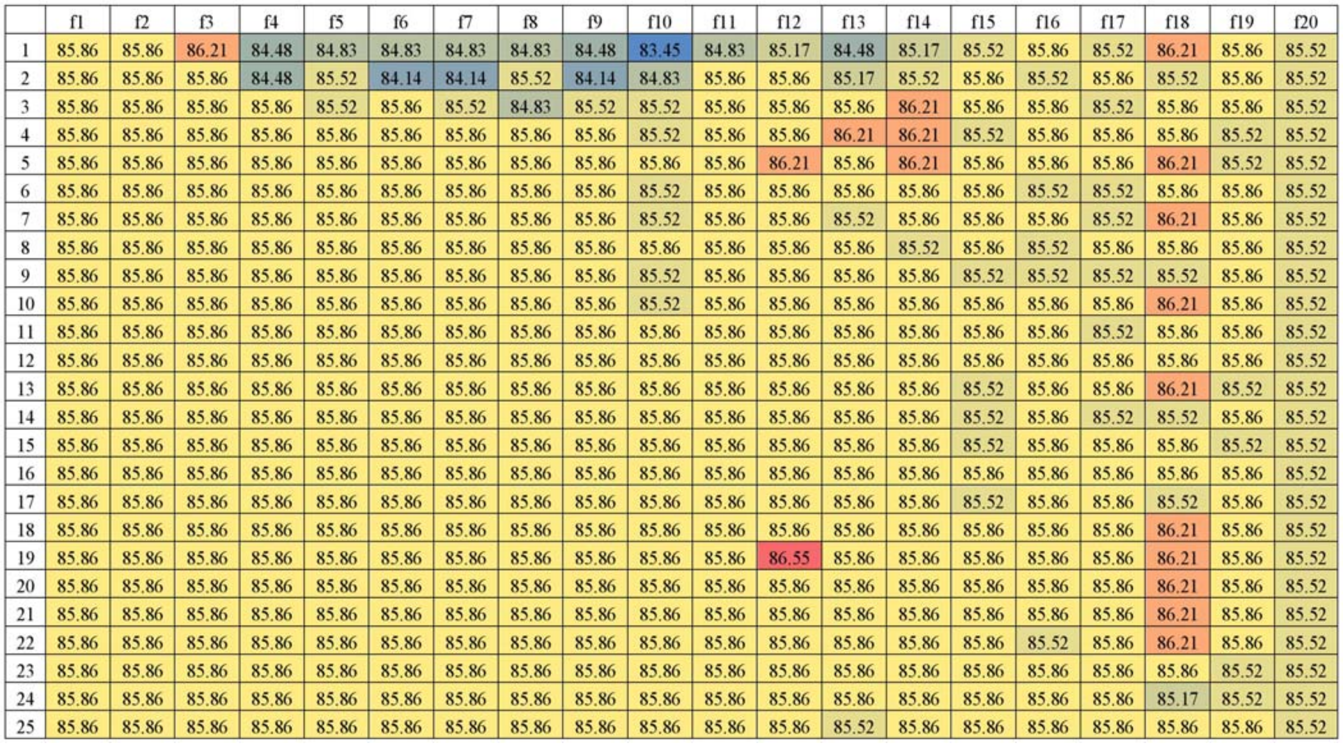
	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	80.34	74.14	75.86	78.28	76.9	77.59	78.62	76.9	77.93	76.55	78.62	75.52	80.69	75.86	76.55	80.69	82.41	80.34	77.59	76.9
2	83.79	80.34	76.55	80	82.41	81.38	81.72	82.41	84.14	82.41	83.1	83.1	82.76	82.76	83.45	83.45	85.52	84.48	82.41	81.72
3	82.41	76.55	79.31	77.59	75.17	80.34	80.34	78.97	80.34	80	81.38	80.69	82.41	80	80	80.34	82.07	80.69	82.41	81.38
4	83.79	79.31	80	78.28	79.66	82.07	80	80	82.07	81.03	82.76	82.76	82.41	80.69	82.07	86.21	82.76	83.79	81.72	85.17
5	82.76	76.21	78.62	76.9	75.52	79.31	81.03	78.97	81.72	78.97	81.38	82.07	79.66	81.72	81.03	81.03	83.79	83.45	84.14	85.17
6	83.1	79.31	79.31	77.24	78.62	81.72	79.31	78.97	80.34	83.1	82.07	82.07	83.45	84.14	83.1	84.48	84.14	83.45	84.48	83.1
7	82.76	77.24	76.55	74.83	81.38	81.03	81.38	78.28	80	79.66	80.34	81.38	80.34	81.03	82.07	82.76	82.41	79.31	84.14	85.17
8	83.1	77.24	78.62	81.38	80	80.34	81.72	81.72	82.41	82.76	82.41	82.07	80.69	84.14	83.45	78.97	82.76	84.14	85.52	83.79
9	82.07	77.59	78.28	76.21	79.31	78.97	81.38	78.62	81.38	81.72	81.72	81.72	78.97	81.38	83.45	80.34	83.1	81.72	84.83	85.52
10	83.79	78.28	79.31	79.66	80.69	76.9	80.34	80	82.76	82.07	81.72	86.55	81.72	81.72	83.45	84.14	81.38	84.83	84.83	84.48
11	82.07	77.93	78.62	77.59	78.28	76.9	79.31	78.28	82.41	80	82.41	80.34	79.66	79.31	83.45	82.07	83.45	80.69	83.79	82.76
12	82.76	76.9	78.28	78.97	78.62	78.28	81.38	79.31	83.1	81.03	80.34	79.31	84.83	82.76	83.79	81.38	84.14	84.14	84.83	85.52
13	82.76	78.28	76.55	78.28	78.97	77.24	81.38	79.31	79.66	79.66	83.1	81.72	83.1	81.38	82.07	81.72	83.79	84.48	82.07	81.72
14	82.41	77.93	77.93	76.9	80.34	81.38	78.97	78.97	81.38	81.38	83.45	82.07	82.41	80.69	82.41	82.76	84.14	80.34	84.48	84.48
15	81.72	76.9	76.21	76.21	78.97	80.34	80.34	79.31	80.34	82.41	82.07	81.72	80.34	82.41	82.41	84.83	81.03	82.41	82.76	83.1
16	82.07	77.59	77.59	77.93	79.31	80.34	80	80.69	79.66	82.07	84.14	83.45	82.41	83.79	81.03	83.79	82.07	83.1	84.83	84.83
17	82.76	78.97	76.9	77.59	81.72	78.62	80	79.31	79.66	81.38	79.66	83.45	82.76	80.69	81.03	83.45	82.07	83.45	82.41	84.83
18	83.79	76.9	79.31	78.62	81.72	78.62	79.66	78.28	81.72	82.07	81.03	82.07	79.66	81.03	81.72	81.72	83.45	82.07	83.79	84.83
19	82.07	79.31	77.24	78.62	80	80	81.38	78.28	80.34	82.76	81.03	81.03	80	79.31	81.72	81.72	84.48	81.72	83.79	83.1
20	81.72	78.28	78.97	77.93	77.93	79.31	79.66	78.62	82.76	81.03	82.41	82.76	80.34	81.72	82.07	83.45	83.1	83.79	83.1	83.79
21	83.45	76.55	77.59	76.9	80	79.66	81.03	77.24	83.1	81.72	82.07	81.38	81.72	83.45	82.76	84.14	82.41	81.72	84.83	84.14
22	83.1	78.62	78.62	76.9	78.97	79.66	81.03	78.62	81.72	79.66	80.69	81.72	80.34	82.41	82.07	81.38	82.41	82.41	84.83	83.1
23	81.72	78.97	78.28	77.24	81.38	78.97	78.62	76.9	81.38	82.41	80.69	80	80.34	81.72	82.76	82.07	82.76	83.45	84.83	85.17
24	82.41	77.93	76.9	77.59	80	78.62	80.69	78.97	83.1	81.38	81.03	82.07	82.76	83.1	81.03	83.1	83.1	82.76	83.1	84.14
25	82.41	78.28	77.93	77.24	78.62	80	78.62	78.97	82.07	80.69	81.38	81.03	81.03	80.34	82.41	83.1	82.07	84.14	84.83	84.83

Figure 7. Heat map of bagging with decision tree classifier using ridge feature ranking algorithm



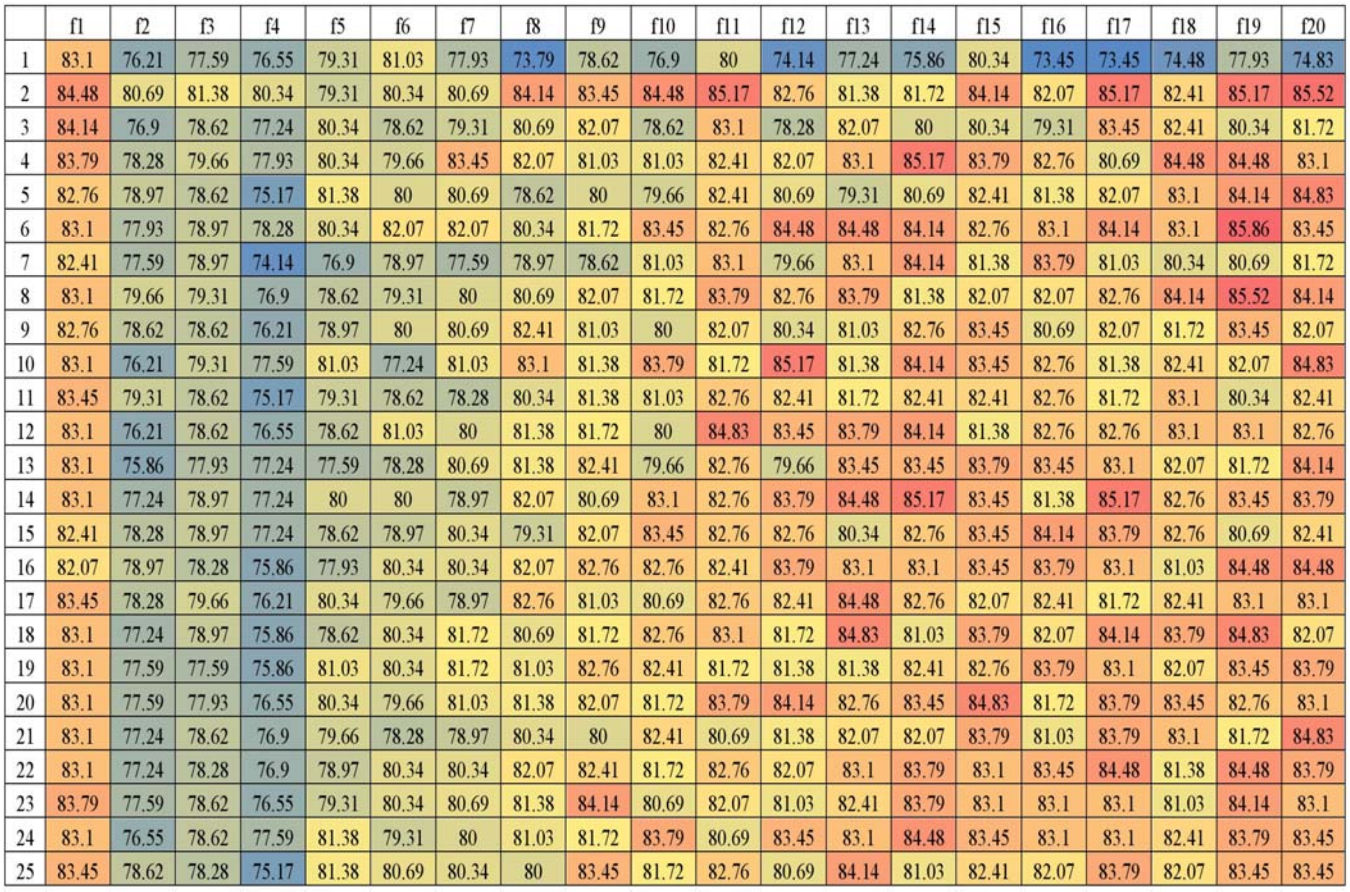
	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	83.79	83.45	79.66	82.76	76.21	78.62	82.07	78.62	78.28	81.03	81.38	81.38	83.1	82.07	81.72	79.66	82.41	82.41	77.93	80.69
2	85.52	85.52	79.66	84.48	82.41	83.79	84.48	82.41	84.83	81.38	83.79	82.07	82.76	83.79	84.48	87.24	82.07	82.41	81.38	84.83
3	84.48	81.03	80.34	82.07	83.1	83.45	84.83	83.1	81.03	83.1	81.72	85.52	81.72	83.1	84.83	82.41	86.21	82.07	80.34	85.86
4	84.14	83.79	81.38	82.76	83.45	83.45	81.03	82.41	83.45	84.48	83.1	84.14	84.14	83.79	79.66	82.07	82.41	84.83	84.14	84.48
5	84.48	83.79	82.07	78.97	85.17	82.07	83.1	84.48	83.45	81.72	80.34	80.69	82.76	80.69	82.41	82.07	83.45	84.14	84.48	83.1
6	85.17	83.79	85.86	84.14	81.38	81.72	83.79	82.41	83.79	84.14	83.45	81.38	82.76	81.03	82.76	81.72	80.69	85.17	84.14	84.14
7	85.52	82.07	84.83	83.1	83.1	83.79	85.17	83.45	80.34	80.69	79.66	84.14	82.76	80.69	83.79	84.48	84.48	83.79	82.76	84.14
8	84.14	82.76	85.52	83.45	83.45	83.1	83.45	82.41	82.76	81.72	82.41	82.41	83.79	83.1	83.79	85.17	85.17	83.1	83.1	84.83
9	83.45	82.76	85.17	82.76	82.76	84.14	83.79	82.76	82.76	83.79	81.38	82.76	83.1	83.79	83.79	84.14	84.14	83.79	85.86	86.55
10	85.52	83.45	85.86	82.76	83.45	84.48	84.48	83.1	84.48	83.1	80.34	82.41	81.03	83.1	84.48	83.45	82.76	83.79	82.07	85.17
11	83.45	83.45	82.41	79.66	83.1	83.45	84.83	84.14	84.83	82.76	83.45	84.14	83.1	82.76	84.83	82.76	84.14	82.76	84.48	84.48
12	85.86	82.41	85.52	84.14	84.48	82.76	84.14	84.83	84.14	82.41	82.41	83.45	82.76	81.72	85.17	83.1	81.72	84.14	81.38	82.41
13	85.86	83.45	85.52	83.79	84.14	83.79	84.48	84.83	83.1	81.03	81.03	82.07	82.41	83.1	83.79	83.45	83.79	83.79	85.17	83.79
14	84.14	84.14	83.45	84.14	84.83	83.79	84.14	83.1	83.45	81.72	82.76	83.1	83.1	81.38	83.45	84.14	84.14	84.83	82.76	83.79
15	84.83	83.1	84.83	84.14	82.76	85.17	85.86	83.1	83.79	83.1	82.07	83.1	82.07	82.07	84.83	83.45	84.14	82.76	83.79	84.83
16	84.83	81.72	83.45	84.14	82.07	85.17	84.83	81.72	83.45	81.38	81.03	81.03	81.03	82.07	83.45	85.17	84.14	83.1	84.14	84.83
17	85.17	83.45	83.79	83.79	83.1	84.83	84.48	81.38	81.72	82.41	83.1	82.07	82.07	83.79	83.45	81.38	82.41	84.14	83.45	85.52
18	84.48	83.1	82.76	83.45	83.45	84.14	84.83	84.83	83.45	83.1	81.03	82.41	82.76	81.38	84.83	83.45	84.83	83.1	84.14	82.76
19	85.17	82.76	84.83	84.48	85.17	83.79	84.48	82.07	84.83	84.83	80.34	83.45	83.1	83.1	84.14	85.17	83.1	85.17	83.1	84.83
20	84.48	82.41	84.48	83.79	84.14	84.48	85.52	83.79	82.41	83.45	80.69	82.41	81.72	82.76	83.45	84.14	83.1	83.79	82.76	83.79
21	85.17	83.1	83.1	85.17	83.79	83.79	84.83	85.17	84.48	82.41	81.38	82.07	81.72	82.76	82.41	84.48	84.14	83.79	82.76	83.79
22	85.52	83.79	85.17	83.45	83.1	83.79	85.52	83.79	84.48	81.38	82.76	81.38	81.72	81.72	85.17	84.48	85.52	85.17	82.76	85.17
23	85.17	83.45	84.14	84.48	82.76	83.79	85.52	84.14	84.48	83.1	81.03	81.38	83.1	83.79	83.45	82.76	82.41	84.83	85.17	83.45
24	85.17	82.76	82.41	83.1	84.48	83.1	85.86	81.72	84.83	82.07	81.38	81.38	82.41	83.45	82.76	84.14	83.45	83.45	83.79	84.48
25	84.14	83.45	85.52	83.79	84.14	83.45	85.17	82.76	83.79	82.76	79.66	80.34	83.45	83.1	84.14	84.14	82.07	85.17	84.83	85.86

Figure 8. Heat map of bagging with K-nearest neighbor classifier using ridge feature ranking algorithm



	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	85.86	85.86	86.21	84.48	84.83	84.83	84.83	84.83	84.48	83.45	84.83	85.17	84.48	85.17	85.52	85.86	85.52	86.21	85.86	85.52
2	85.86	85.86	85.86	84.48	85.52	84.14	84.14	85.52	84.14	84.83	85.86	85.86	85.17	85.52	85.86	85.52	85.86	85.52	85.86	85.52
3	85.86	85.86	85.86	85.86	85.52	85.86	85.52	84.83	85.52	85.52	85.86	85.86	85.86	86.21	85.86	85.86	85.52	85.86	85.86	85.52
4	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	86.21	86.21	85.52	85.86	85.86	85.86	85.52	85.52
5	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	86.21	85.86	86.21	85.86	85.86	85.86	86.21	85.52	85.52
6	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.86	85.86	85.86	85.52	85.52	85.86	85.86	85.52
7	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.52	85.86	85.86	85.86	85.52	86.21	85.86	85.52
8	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.52	85.86	85.86	85.86	85.52
9	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.86	85.86	85.52	85.52	85.52	85.52	85.86	85.52
10	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.86	85.86	85.86	85.86	85.86	86.21	85.86	85.52
11	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.52
12	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52
13	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	86.21	85.52	85.52
14	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.52	85.52	85.86	85.52
15	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.86	85.52	85.52
16	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52
17	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.52	85.86	85.52
18	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	86.21	85.86	85.52
19	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	86.55	85.86	85.86	85.86	85.86	85.86	86.21	85.86	85.52
20	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	86.21	85.86	85.52
21	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	86.21	85.86	85.52
22	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	86.21	85.86	85.52
23	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.52
24	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.17	85.52	85.52
25	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.86	85.52	85.86	85.86	85.86	85.86	85.86	85.86	85.52

Figure 9. Heat map of adaboost with support vector machine classifier using ridge feature ranking algorithm



	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	83.1	76.21	77.59	76.55	79.31	81.03	77.93	73.79	78.62	76.9	80	74.14	77.24	75.86	80.34	73.45	73.45	74.48	77.93	74.83
2	84.48	80.69	81.38	80.34	79.31	80.34	80.69	84.14	83.45	84.48	85.17	82.76	81.38	81.72	84.14	82.07	85.17	82.41	85.17	85.52
3	84.14	76.9	78.62	77.24	80.34	78.62	79.31	80.69	82.07	78.62	83.1	78.28	82.07	80	80.34	79.31	83.45	82.41	80.34	81.72
4	83.79	78.28	79.66	77.93	80.34	79.66	83.45	82.07	81.03	81.03	82.41	82.07	83.1	85.17	83.79	82.76	80.69	84.48	84.48	83.1
5	82.76	78.97	78.62	75.17	81.38	80	80.69	78.62	80	79.66	82.41	80.69	79.31	80.69	82.41	81.38	82.07	83.1	84.14	84.83
6	83.1	77.93	78.97	78.28	80.34	82.07	82.07	80.34	81.72	83.45	82.76	84.48	84.48	84.14	82.76	83.1	84.14	83.1	85.86	83.45
7	82.41	77.59	78.97	74.14	76.9	78.97	77.59	78.97	78.62	81.03	83.1	79.66	83.1	84.14	81.38	83.79	81.03	80.34	80.69	81.72
8	83.1	79.66	79.31	76.9	78.62	79.31	80	80.69	82.07	81.72	83.79	82.76	83.79	81.38	82.07	82.07	82.76	84.14	85.52	84.14
9	82.76	78.62	78.62	76.21	78.97	80	80.69	82.41	81.03	80	82.07	80.34	81.03	82.76	83.45	80.69	82.07	81.72	83.45	82.07
10	83.1	76.21	79.31	77.59	81.03	77.24	81.03	83.1	81.38	83.79	81.72	85.17	81.38	84.14	83.45	82.76	81.38	82.41	82.07	84.83
11	83.45	79.31	78.62	75.17	79.31	78.62	78.28	80.34	81.38	81.03	82.76	82.41	81.72	82.41	82.41	82.76	81.72	83.1	80.34	82.41
12	83.1	76.21	78.62	76.55	78.62	81.03	80	81.38	81.72	80	84.83	83.45	83.79	84.14	81.38	82.76	82.76	83.1	83.1	82.76
13	83.1	75.86	77.93	77.24	77.59	78.28	80.69	81.38	82.41	79.66	82.76	79.66	83.45	83.45	83.79	83.45	83.1	82.07	81.72	84.14
14	83.1	77.24	78.97	77.24	80	80	78.97	82.07	80.69	83.1	82.76	83.79	84.48	85.17	83.45	81.38	85.17	82.76	83.45	83.79
15	82.41	78.28	78.97	77.24	78.62	78.97	80.34	79.31	82.07	83.45	82.76	82.76	80.34	82.76	83.45	84.14	83.79	82.76	80.69	82.41
16	82.07	78.97	78.28	75.86	77.93	80.34	80.34	82.07	82.76	82.76	82.41	83.79	83.1	83.1	83.45	83.79	83.1	81.03	84.48	84.48
17	83.45	78.28	79.66	76.21	80.34	79.66	78.97	82.76	81.03	80.69	82.76	82.41	84.48	82.76	82.07	82.41	81.72	82.41	83.1	83.1
18	83.1	77.24	78.97	75.86	78.62	80.34	81.72	80.69	81.72	82.76	83.1	81.72	84.83	81.03	83.79	82.07	84.14	83.79	84.83	82.07
19	83.1	77.59	77.59	75.86	81.03	80.34	81.72	81.03	82.76	82.41	81.72	81.38	81.38	82.41	82.76	83.79	83.1	82.07	83.45	83.79
20	83.1	77.59	77.93	76.55	80.34	79.66	81.03	81.38	82.07	81.72	83.79	84.14	82.76	83.45	84.83	81.72	83.79	83.45	82.76	83.1
21	83.1	77.24	78.62	76.9	79.66	78.28	78.97	80.34	80	82.41	80.69	81.38	82.07	82.07	83.79	81.03	83.79	83.1	81.72	84.83
22	83.1	77.24	78.28	76.9	78.97	80.34	80.34	82.07	82.41	81.72	82.76	82.07	83.1	83.79	83.1	83.45	84.48	81.38	84.48	83.79
23	83.79	77.59	78.62	76.55	79.31	80.34	80.69	81.38	84.14	80.69	82.07	81.03	82.41	83.79	83.1	83.1	83.1	81.03	84.14	83.1
24	83.1	76.55	78.62	77.59	81.38	79.31	80	81.03	81.72	83.79	80.69	83.45	83.1	84.48	83.45	83.1	83.1	82.41	83.79	83.45
25	83.45	78.62	78.28	75.17	81.38	80.69	80.34	80	83.45	81.72	82.76	80.69	84.14	81.03	82.41	82.07	83.79	82.07	83.45	83.45

Figure 10. Heat map of extremely randomized trees classifier using ridge feature ranking algorithm

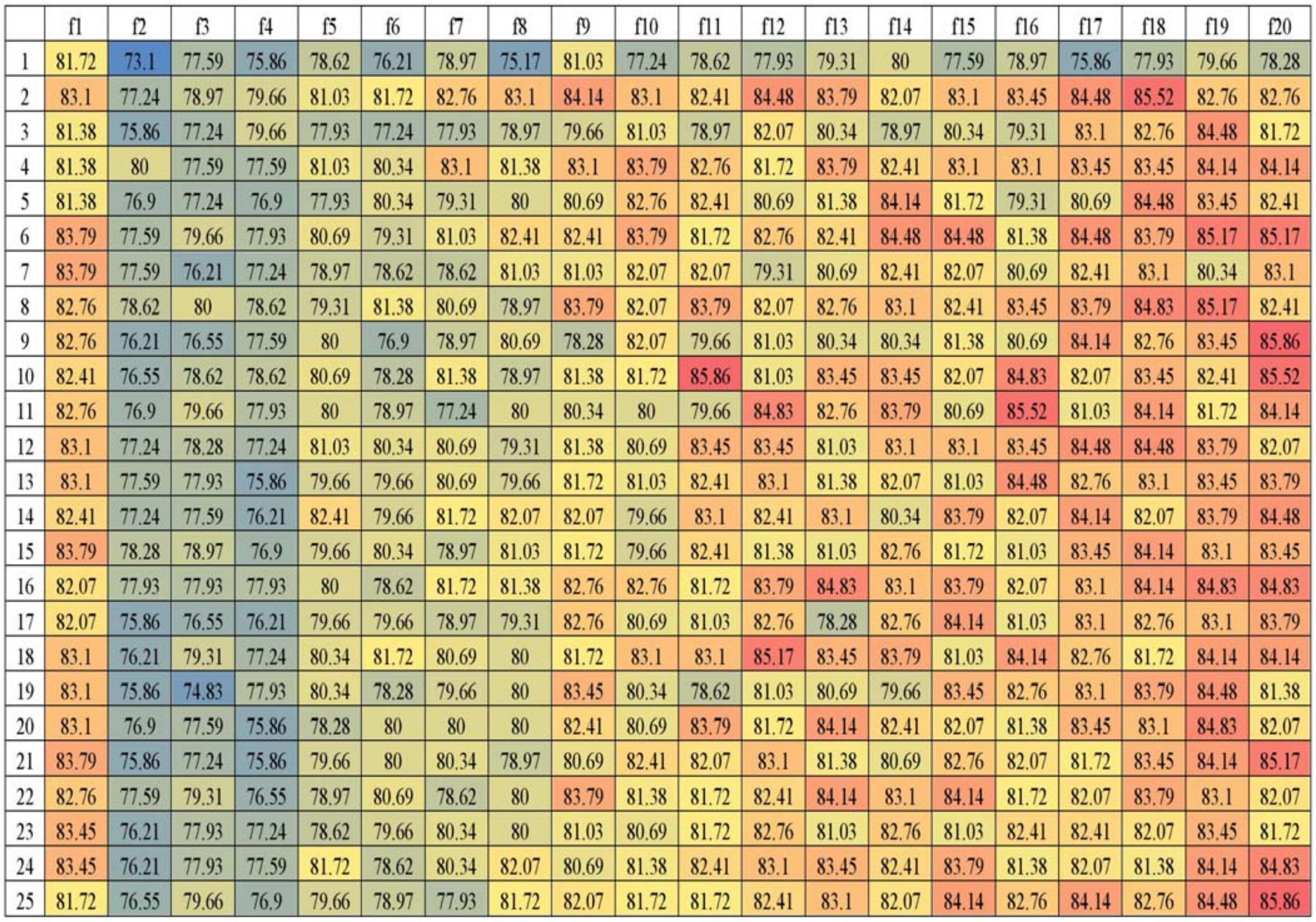
Figure 9 shows the results of A_SVM classifier. In this case, also, the single topmost feature is able to provide the efficient solution. Also, overall accuracy remains almost constant with increase in the estimator. For different combination of feature set and estimator, accuracy remain > 84%. However, for feature set 1, and estimator 1 provides accuracy 85.86%. Also, for different combination of feature sets and estimator, maximum time accuracy remains between 85.86%. A_SVM with Ridge shows almost same behavior that of A_SVM with Lasso. Maximum accuracy achieved is 86.55% for feature set 12 and estimator 19.

Figure 10 and 11 illustrates the results of ET and RF classifier. In this case, also, the single topmost feature is unable to provide the efficient solution. It shows the similar behavior that of B_DT. For ET classifier, with feature set 19 and estimator 6, accuracy of achieved is 85.86%. ET with Lasso show better accuracy than ET with Ridge. For RF classifier, with feature set 11 and estimator 10, accuracy of achieved 85.86%. RF with Lasso show better accuracy than RF with Ridge.

To do the further analysis of these ranking algorithms and classifiers, their standard deviation values are calculated to compare the results. In [11] shows the best accuracy of 96.4% but his classification accuracy varies from 71.4% to 96.4%. In proposed method, accuracy for all the classifier is more than 85%. It is less than 96.4% but the less variation in the result as shown in Figure 14. Therefore, proposed method is more robust.

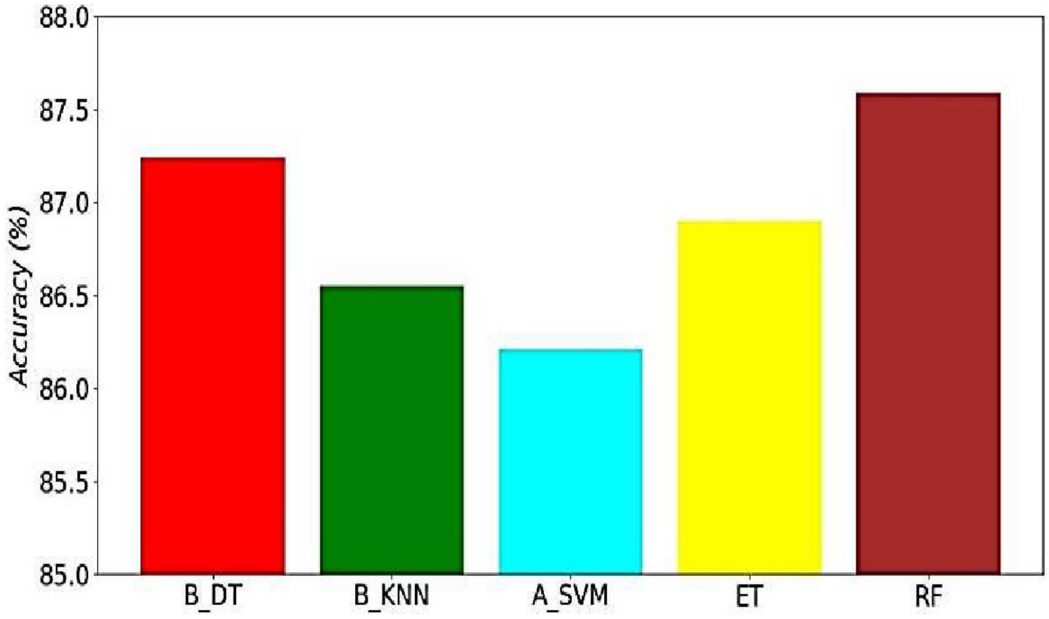
Also, evaluation time is compared for different classifier. For lasso feature ranking algorithm, RF performs well as it requires less time for execution and accuracy is also high as compared to other classifiers as shown in Figure 10 and 11. Also for ridge feature ranking algorithm, B_KNN performs well as its accuracy higher than other classifier and execution time is high.

In both ranking algorithm, A_SVM performance is below average as compared to other algorithms but also requires more execution time for classification as shown in Figure 12 and 13. For both the ranking algorithms, B_DT performance is above the average performance and required execution time is almost equal to the best classifier of both the ranking algorithms.

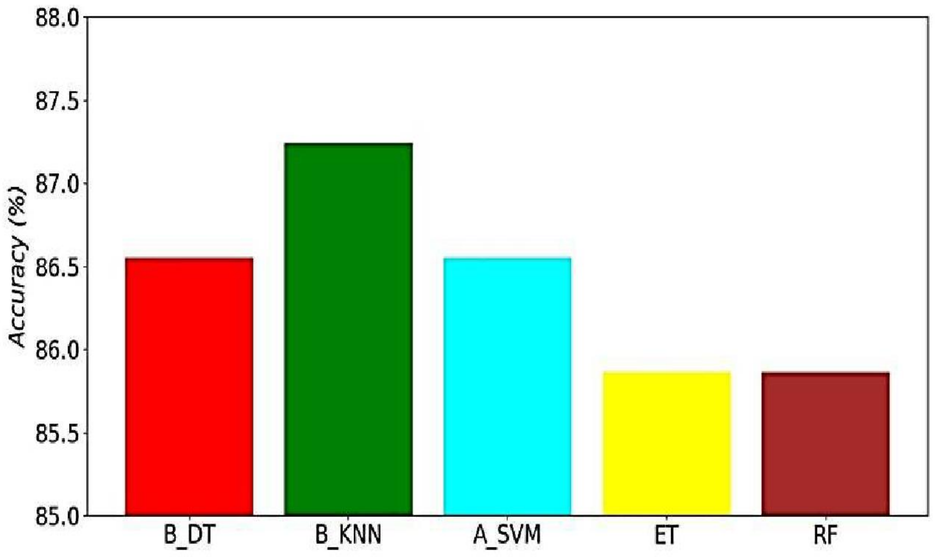


	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17	f18	f19	f20
1	81.72	73.1	77.59	75.86	78.62	76.21	78.97	75.17	81.03	77.24	78.62	77.93	79.31	80	77.59	78.97	75.86	77.93	79.66	78.28
2	83.1	77.24	78.97	79.66	81.03	81.72	82.76	83.1	84.14	83.1	82.41	84.48	83.79	82.07	83.1	83.45	84.48	85.52	82.76	82.76
3	81.38	75.86	77.24	79.66	77.93	77.24	77.93	78.97	79.66	81.03	78.97	82.07	80.34	78.97	80.34	79.31	83.1	82.76	84.48	81.72
4	81.38	80	77.59	77.59	81.03	80.34	83.1	81.38	83.1	83.79	82.76	81.72	83.79	82.41	83.1	83.1	83.45	83.45	84.14	84.14
5	81.38	76.9	77.24	76.9	77.93	80.34	79.31	80	80.69	82.76	82.41	80.69	81.38	84.14	81.72	79.31	80.69	84.48	83.45	82.41
6	83.79	77.59	79.66	77.93	80.69	79.31	81.03	82.41	82.41	83.79	81.72	82.76	82.41	84.48	84.48	81.38	84.48	83.79	85.17	85.17
7	83.79	77.59	76.21	77.24	78.97	78.62	78.62	81.03	81.03	82.07	82.07	79.31	80.69	82.41	82.07	80.69	82.41	83.1	80.34	83.1
8	82.76	78.62	80	78.62	79.31	81.38	80.69	78.97	83.79	82.07	83.79	82.07	82.76	83.1	82.41	83.45	83.79	84.83	85.17	82.41
9	82.76	76.21	76.55	77.59	80	76.9	78.97	80.69	78.28	82.07	79.66	81.03	80.34	80.34	81.38	80.69	84.14	82.76	83.45	85.86
10	82.41	76.55	78.62	78.62	80.69	78.28	81.38	78.97	81.38	81.72	85.86	81.03	83.45	83.45	82.07	84.83	82.07	83.45	82.41	85.52
11	82.76	76.9	79.66	77.93	80	78.97	77.24	80	80.34	80	79.66	84.83	82.76	83.79	80.69	85.52	81.03	84.14	81.72	84.14
12	83.1	77.24	78.28	77.24	81.03	80.34	80.69	79.31	81.38	80.69	83.45	83.45	81.03	83.1	83.1	83.45	84.48	84.48	83.79	82.07
13	83.1	77.59	77.93	75.86	79.66	79.66	80.69	79.66	81.72	81.03	82.41	83.1	81.38	82.07	81.03	84.48	82.76	83.1	83.45	83.79
14	82.41	77.24	77.59	76.21	82.41	79.66	81.72	82.07	82.07	79.66	83.1	82.41	83.1	80.34	83.79	82.07	84.14	82.07	83.79	84.48
15	83.79	78.28	78.97	76.9	79.66	80.34	78.97	81.03	81.72	79.66	82.41	81.38	81.03	82.76	81.72	81.03	83.45	84.14	83.1	83.45
16	82.07	77.93	77.93	77.93	80	78.62	81.72	81.38	82.76	82.76	81.72	83.79	84.83	83.1	83.79	82.07	83.1	84.14	84.83	84.83
17	82.07	75.86	76.55	76.21	79.66	79.66	78.97	79.31	82.76	80.69	81.03	82.76	78.28	82.76	84.14	81.03	83.1	82.76	83.1	83.79
18	83.1	76.21	79.31	77.24	80.34	81.72	80.69	80	81.72	83.1	83.1	85.17	83.45	83.79	81.03	84.14	82.76	81.72	84.14	84.14
19	83.1	75.86	74.83	77.93	80.34	78.28	79.66	80	83.45	80.34	78.62	81.03	80.69	79.66	83.45	82.76	83.1	83.79	84.48	81.38
20	83.1	76.9	77.59	75.86	78.28	80	80	80	82.41	80.69	83.79	81.72	84.14	82.41	82.07	81.38	83.45	83.1	84.83	82.07
21	83.79	75.86	77.24	75.86	79.66	80	80.34	78.97	80.69	82.41	82.07	83.1	81.38	80.69	82.76	82.07	81.72	83.45	84.14	85.17
22	82.76	77.59	79.31	76.55	78.97	80.69	78.62	80	83.79	81.38	81.72	82.41	84.14	83.1	84.14	81.72	82.07	83.79	83.1	82.07
23	83.45	76.21	77.93	77.24	78.62	79.66	80.34	80	81.03	80.69	81.72	82.76	81.03	82.76	81.03	82.41	82.41	82.07	83.45	81.72
24	83.45	76.21	77.93	77.59	81.72	78.62	80.34	82.07	80.69	81.38	82.41	83.1	83.45	82.41	83.79	81.38	82.07	81.38	84.14	84.83
25	81.72	76.55	79.66	76.9	79.66	78.97	77.93	81.72	82.07	81.72	81.72	82.41	83.1	82.07	84.14	82.76	84.14	82.76	84.48	85.86

Figure 11. Heat map of random forest classifier using ridge feature ranking algorithm

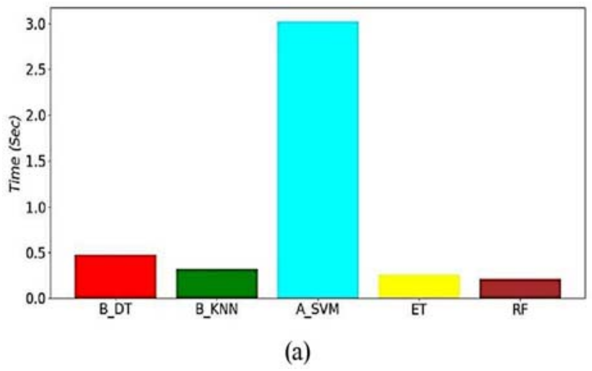


(a)



(b)

Figure 12. Accuracy comparison of different classifiers for (a) Lasso and (b) Ridge feature ranking algorithm



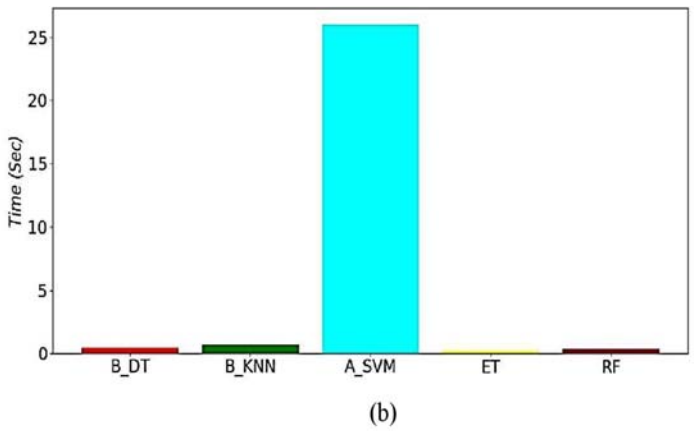
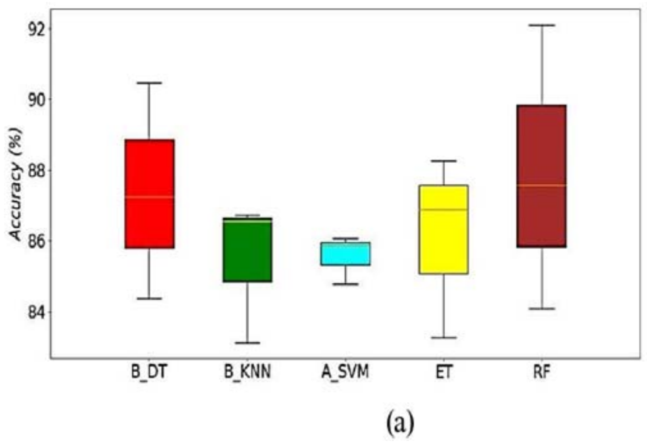


Figure 13. Execution time comparison of different classifiers for (a) Lasso and (b) Ridge feature ranking algorithm



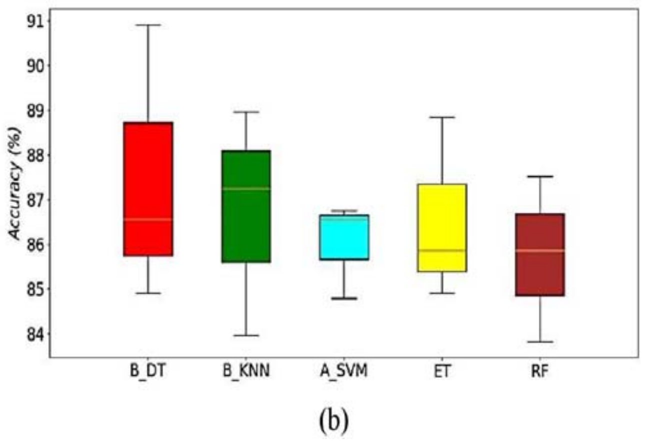


Figure 14. Accuracy deviation comparison of different classifiers for (a) Lasso and (b) Ridge feature ranking algorithm

6. Conclusion

Software fault classification is crucial to provide the good quality and reliable software. It helps to improve the software development process and tries to reduce the faults in each version. The software metrics provides the good information regarding the software faults by calculating various values. However, to efficiently represent fault specific features needs to be identified from metrics. Hence, the Ridge and Lasso feature ranking algorithms are used to penalize the feature in the proposed work. Also, ensemble techniques are used to handle the penalize features. The results demonstrate the B_KNN and RF perform better for fault classification.

References

[1] Chidamber, S. R., Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20 (6) 476-493.

[2] Turhan, B., Bener, A. (2009). Analysis of Naive Bayes' assumptions on software fault data: An empirical study. *Data & Knowledge Engineering*, 68 (2) 278-290.

[3] Jureczko, M., Madeyski, L. (2010). Prediction of defects based on software metrics - identification of project classes. *Proceedings of the National Conference on Software Engineering (KKIO 2010). PWNT*, p. 185-192.

[4] Chen, Y., Shen, X. H., Du, P., Ge, B. (2010). February. Research on software defect prediction based on data mining. *In*: Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on (1, p. 563-567). IEEE.

[5] Najadat, H., Alsmadi, I. (2012). Enhance rule based detection for software fault prone modules. *International Journal of Software Engineering and Its Applications*, 6 (1) 75-86.

[6] Okutan, A., Yýldýz, O. T. (2014). Software defect prediction using Bayesian networks. *Empirical Software Engineering*, 19 (1) 154-181.

[7] Phyu, T. N. (2009 March). Survey of classification techniques in data mining. *In*: Proceedings of the International Multi Conference of Engineers and Computer Scientists (1, p. 18-20).

[8] Kaur, A., Kaur, I. (2014). Empirical evaluation of machine learning algorithms for fault prediction. *Lecture Notes on Software Engineering*, 2 (2) 176.

[9] Zhang, W., Yang, Y., Wang, Q. (2015). Using Bayesian regression and EM algorithm with missing handling for software effort prediction. *Information and Software Technology*, 58, 58-70.

[10] Sankar, K., Kannan, S., Jennifer, P. (2014). Prediction of code fault using Naive Bayes and SVM classifiers. *Middle-East Journal of Scientific Research*, 20 (1) 108-113.

[11] Jureczko, M., Madeyski, L. (2010 September). Towards identifying software project clusters with regard to defect prediction. *In*: *Proceedings of the 6th International Conference on Predictive Models in Software Engineering* (p. 9). ACM.

[12] Turhan, B., Menzies, T., Bener, A. B., Stefano, Di., J. (2009). On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, 14 (5) 540-578.

[13] Turhan, B., Bener, A., Menzies, T. (2010 June). Regularities in learning defect predictors. *In*: International Conference on Product Focused Software Process Improvement (p. 116-130). Springer, Berlin, Heidelberg.

[14] Nagappan, N., Ball, T., Zeller, A. (2006, May). Mining metrics to predict component failures. *In*: Proceedings of the 28th international conference on Software engineering (p. 452-461). ACM.

[15] Menzies, T., Greenwald, J., Frank, A. (2007). Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, (1) 2-13.

[16] Tang, M. H., Kao, M. H., Chen, M. H. (1999). An empirical study on object-oriented metrics. *In*: Software Metrics Symposium, 1999. Proceedings. Sixth International (p. 242-249). IEEE.

[17] Fenton, N. E., Neil, M. (1999). A critique of software defect prediction models. *IEEE Transactions on software engineering*, 25(5) 675-689.

[18] Friedman, J., Hastie, T., Tibshirani, R. (2001). *The elements of statistical learning*, 1(10). New York, NY, USA: Springer series

in statistics.

[19] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2) 123-140.

[20] Breiman, L. (2017). *Classification and regression trees*. Routledge.

[21] Rätsch, G., Onoda, T., Müller, K. R. (2001). Soft margins for AdaBoost. *Machine learning*, 42(3), 287-320.

[22] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.

[23] Merentitis, A., Debes, C. (2015). Many hands make light work-on ensemble learning techniques for data fusion in remote sensing. *IEEE Geoscience and Remote Sensing Magazine*, 3(3) 86-99.