Chapter 4

An Unsupervised SBP Technique Using Threshold Derivation

This chapter focuses on the second contribution of this thesis detail in Section 1.7.2. We provide an introduction, motivation, and key feature for the problem to develop classification-based unsupervised SBP in Section 4.1. Section 4.2 demonstrates the proposed work to develop the classification-based unsupervised SBP. The experimental setup is given in Section 4.3. Section 4.4 covers the results obtained using the proposed technique. Section 4.5 reports the explainability and interoperability, and Section 4.6 reports the threat of validity associated with the chapter. Section 4.7 focuses on the conclusion and future scope of this chapter. Detailed related work is provided in Section 2.2.1.

4.1 Introduction

Software Fault Prediction (SFP) is a critical area in software engineering, attracting significant research attention due to its potential to focus testing efforts

on error-prone artifacts, thereby saving resources and time [14–18]. It plays a crucial role in aiding the Software Quality Assurance (SQA) team, enabling the application of stringent quality measures and analyzing software quality before final release [19]. SFP models have found active adoption in the software industry for software reliability prediction [20–22]. Researchers have proposed various software metrics and SFP models [19, 28, 108, 110]. While many SFP models are based on supervised learning, their application is hindered by the challenge of obtaining labeled datasets, making them less adaptable for new software projects or projects with limited historical data [24, 57, 113]. To address this, researchers have explored Cross-Project SFP (CPFP) models, but limitations arise when the metric sets of different projects don't have same metric distribution. To mitigate these issues, heterogeneous defect prediction (HDP) models have been introduced to normalize metric distributions [61, 62, 64]. However, these approaches may not be practical in all cases, and expert-based or threshold-based methods still require human intervention [115]. To overcome these limitations, software metric threshold-based unsupervised learning techniques like CLA/CLAMI [116], ACL [140], CLAMI+ [117] models have emerged, eliminating the need for human intervention and providing an automatic path forward in SFP. Further, we extended these SFP approaches and proposed a novel metric threshold-based unsupervised approach, TCL and TCLP, to design an automated SFP system on unlabelled DSs. Here, TCL and TCLP are abbreviated for Threshold Clustering Labelling/Threshold Clustering Labelling Plus. Here, Plus stands for metric selection, instance selection, learning model, and prediction on original DSs. Both TCL and TCLP are based on the concept of fault proneness tendency (Higher complexity in source code causes more fault proneness).

Fig. 4.1 shows the steps involved in the proposed approach TCL/TCLP. Fig. 4.1, contains 10 white boxes and 5 gray colored boxes. White boxes represent the TCL steps used to label the DS automatically. Further, the next 5 gray boxes show the last 5 steps of TCLP applied on TCL labeled DS.

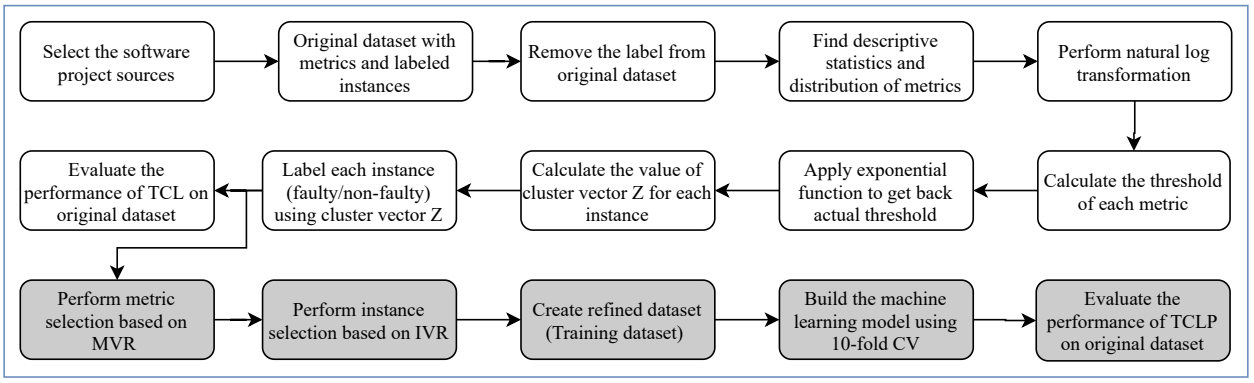


FIGURE 4.1: Block diagram of the proposed approach showing the steps involved in TCL/TCLP

4.2 Proposed Approach: TCL/TCLP

The proposed novel approach TCL/TCLP, is helpful to predict the fault in unlabelled DSs and does not require prior knowledge of the distribution of the DSs. This approach is applied to 28 DSs collected from five different repositories. TCL/TCLP is an extended method of CLA/CLAMI. TCL has three steps: **(1) T**hreshold calculation of each metric, **(2) C**lustering instances, and **(3) L**abelling the instances in two clusters. These three steps help to label or predict all instances as faulty or non-faulty in unlabelled DSs. Further, TCLP stands for **TCL P**lus. TCLP has three additional steps to prepare a refined training DS. These three additional steps are: **(4)** Metric selection, **(5)** Instance selection, and **(6)** Learning and prediction.

After combining the related steps of TCL and TCLP, we can describe our approach as a 4-stage process (**A.** Threshold derivation, **B.** Clustering and Labelling, **C.** Metric Selection and Instance Selection, and **D.** Learning and Prediction (applying a machine learning algorithm on the refined DS to build a SFP model and predict on original unlabelled DS). These four different stages have been described in the following subsections.

4.2.1 Threshold Derivation

Threshold calculation is central to the presented heuristic. Threshold derivation plays a principal role for clustering and labelling the data points in our approach. Hence, the first stage is to derive the threshold of the software metrics. It is a well-known fact that software metrics (Chidamber and Kemerer (CK) software metrics suite) [162] define the internal quality of the software code. This internal measurement of software quality helps both the software developer and the tester to improve those quality aspects for which the values of the measurements are not satisfactory. So, software metrics can be used as the software quality evaluation factor. Also, software code with high complexity is not desirable because it is said to be more fault-prone [24, 28]. Threshold derivation steps are shown in the following subsections.

4.2.1.1 Descriptive Statistics and Distribution

It has been found in the study conducted by Barkmann et al. [206] that the distribution of the CK software metrics suite [162] is skewed towards the positive. Other earlier researches [107, 207, 208] have also concluded that all CK software metrics are skewed towards the positive. Hence, software metrics are not always characterized by descriptive statistics. So, the implementation and usage of the software prediction system based on software metrics are also affected by the distribution of metrics. Therefore, it is required to first make the data distribution normal by bringing the skewness index value near zero [209]. From the literature survey, we have found that the natural logarithmic transformation method is more suitable to make the metric distribution normal [210]. There is limitation of log function $ln(x)$ that it can transform only those values that are greater than zero. To deal with zero values, a constant is added. Such as $Log(x) = ln(x + 1)$. Where x is the value of a software metric [210]. Practically, natural log transformation reduces the relative distances between data points, and so it helps to reduce the skewness. The skewness index is the parameter to measure the asymmetry in data distribution [209]. As we know,

any symmetric data should have skewness near zero, and the normal distribution has a skewness value of 0.

4.2.1.2 Software Metrics Threshold Calculation

After making the data closer to normal distribution or closer to symmetric, the distribution parameters, i.e., mean (μ_m) and standard deviation (Ω_m), are calculated using (4.1) and (4.2) respectively for each metric m to find the value of metric threshold ($T_m^{'}$). Equation (4.3) is used to calculate $T_m^{'}$.

$$\mu_m = \frac{\sum_{i=1}^{n} x_i}{n} \tag{4.1}$$

$$\Omega_m = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu_m)^2}{n-1}} \tag{4.2}$$

$$T_m^{'} = \mu_m + \Omega_m \tag{4.3}$$

$T_m^{'}$ is the symbolic threshold for transformed data. To calculate the representative threshold for the original data, T_m can be calculated using (4.4).

$$T_m = exp(T_m^{'}) \tag{4.4}$$

The value of T_m is calculated for all the metrics m of each DS. After the calculation of the threshold, clustering and labeling is performed.

4.2.2 Clustering and Labelling

The proposed method is more of a heuristic to perform grouping. This heuristic doesn't fit under conventional clustering methods. In this method, clustering/-grouping completely depends on the derived threshold of the software metrics. The threshold derivation and comparison of metric values with the corresponding threshold to label the DS is our core idea (Algorithm 3). The input for clustering/grouping is an unlabelled DS $DS[1:n, 1:m]$ and a threshold vector $DT[1:m]$ (calculated

TABLE 4.1: An example for an unlabeled DS to labeled based on metric threshold

Instance	M1	M2	M3	Label
I1	2	3	**2**	?
I2	**3**	2	1	?
I3	1	1	**2**	?
I4	**3**	3	**4**	?
I5	**4**	**9**	**7**	?
I6	1	0	1	?
I7	**6**	2	**8**	?
I8	2	3	1	?
Threshold	2.3	3.6	1.2	

using (4.3) and (4.4)). Where m is the number of metrics and n is the number of instances. Based on $DS[1:n, 1:m]$, and $DT[1:m]$, we can create a cluster vector $Z[1:n]$, where $Z[i]$ is defined as the total number of metrics greater than their corresponding threshold in each instance i. An example is demonstrated as follows:

Clustering: Let us consider an example of a multiclass classification of instances, as shown in Table 4.1. In this table, only three metrics, M1 to M3, and eight instances I1 to I8, are given as an unlabelled DS $DS[1:n, 1:m]$, and a threshold vector $DT[1:m]$ (calculated using (4.3) and (4.4)), here $n = 8$ and $m = 3$. Now the problem is to label these 8 instances marked as (?) in "Label" column of Table 4.1. So to label the instances, firstly we find the values of cluster vector $Z[1:n]$.

Fault proneness tendency: Higher complexity in source code causes more fault proneness [24, 28, 112]. The final labeling of instances is done based on the fact that higher metric values cause the instances to be fault-prone [211–213].

Solution: Let us assume the threshold values of metrics M1, M2, and M3 as calculated using (4.3) and (4.4)) are 2.3, 3.6, and 1.2, respectively, shown in the last row of Table 4.1. Firstly, we find those instances that have metrics values greater than the threshold. E.g., I2, I4, I5, and I7 have their M1 metric value (3, 3, 4, and 6) higher than the threshold value 2.3, shown in bold in Table 4.1. Similarly, we identify those instances having M2 and M3 metrics values higher than their corresponding

thresholds. Higher metrics values are shown in bold font in Table 4.1. After that, we count the metrics values that surpass the threshold value in each instance. E.g., in instance I4, the number of metrics values that surpass the threshold value is 2 (3 in M1 and 4 in M3). In this way, we find the Z value for each instance (shown in Table 4.2). Based on Z, the data points are divided into four clusters (0,1,2,3). E.g. cluster 2 has two data points I4 and I7. After getting these four clusters, we assign a label to each instance on the basis of its Z value. This process is described below:

Labeling: Cluster vector $Z[1:n]$ thus obtained is fed as input for labeling. We divide these cluster vector values into two classes to label each instance as faulty or non-faulty. For each instance i, if $(Z[i] > \lfloor max(Z)/2 \rfloor)$, then mark the instance i as faulty (YES) otherwise, mark it as non-faulty (NO) (shown in Table 4.3). E.g. in Table 4.3, instances 1, 2, 3, 6, and 8 are labeled as non-faulty (Label=NO), and instances 4, 5, 7 are labeled as fault-prone (Label=YES) after Labeling.

The threshold derivation, clustering, and labeling have successfully been completed on the unlabelled DSs. This proposed technique is known as TCL (Threshold, Clustering, and Labeling) approach. The Algorithm 3 is used to implement the TCL approach. We have evaluated the performance of TCL on 28 DSs, and results are shown in Table 4.7, Table 4.8, Table 4.9.

4.2.3 Metric Selection and Instance Selection

Metric selection: After completing the TCL method, we have a DS $DS[1:n, 1:m]$, a threshold vector $DT[1:m]$, and a label vector $DL[1:n]$. After labeling the instances using TCL, we found many metric-threshold violations (MTV) in DS. Since the quality of the SFP model depends on the metrics, many researchers widely chose to select the most informative metrics to develop the prediction model [26, 214–216]. Choosing informative metrics means removing those metrics that can deteriorate the performance of the prediction model [24, 28, 112]. So, to enhance

Algorithm 3: To implement TCL (Threshold Clustering Labeling) approach

Input: Dataset $DS[1:n, 1:m]$ //DS having n instances and m metrics
Output: Vector $DL[1:n]$ // Data point Label
Performance of TCL as Accuracy, FM and MCC
Algorithmic Steps:
Step 1: Initialize input DS as $DS[X]_{n*m}$
Step 2: $DS \leftarrow DS + 1$ // Add 1 to the DS (Subsection 4.2.1.1)
Step 3: $DS \leftarrow ln(DS)$ // Taking natural log
Step 4: Calculate the threshold of each metrics i
for $i \leftarrow 1$ **to** m **do**
  $\mu_m[i] \leftarrow mean(DS(:, i))$ // mean is calculated using (4.1)
  $\Omega_m[i] \leftarrow SD(DS(:, i))$ // SD is standard deviation calculated using (4.2)
  $T_m^{'}[i] \leftarrow \mu_m[i] + \Omega_m[i]$ // $T_m^{'}$ logarithmic threshold calculated using (4.3)
end
Step 5: Perform exponential transformation to get back original DS and threshold
$DS \leftarrow exp(DS)$ // exp is exponential function
$T_m \leftarrow exp(T_m^{'})$ // T_m is the original threshold calculated using (4.4)
Step 6: Calculate the cluster vector Z[1:n], Z[i] is number of metrics values greater than the corresponding threshold in an instance i
for $i \leftarrow 1$ **to** n **do**
  $t \leftarrow 0$
  for $j \leftarrow 1$ **to** m **do**
    if $DS[i, j] > T_m[j]$ **then**
      $t = t + 1$
  end
  $Z[i] \leftarrow t$
end
Step 7: Label the DS based on the values of Z[1:n]
for $i \leftarrow 1$ **to** n **do**
  if $Z[i] > \lfloor (max(Z)/2) \rfloor$ **then**
    $label\_fault[i] \leftarrow' YES'$
  else
    $label\_fault[i] \leftarrow' NO'$
end
Step 8: Calculate the performance (Accuracy, FM, and MCC) of TCL

TABLE 4.2: Finding the value of Z for each instance of unsupervised DS

Instance	M1	M2	M3	Label	Z
I1	2	3	**2**	?	1
I2	**3**	2	1	?	1
I3	1	1	**2**	?	1
I4	**3**	3	**4**	?	2
I5	**4**	**9**	**7**	?	3
I6	1	0	1	?	0
I7	**6**	2	**8**	?	2
I8	2	3	1	?	0
Threshold	2.3	3.6	1.2		

TABLE 4.3: Example for an unlabelled dataset to labeled dataset

Instance	M1	M2	M3	Label	Z	Label
I1	2	3	**2**	?	1	No
I2	**3**	2	1	?	1	No
I3	1	1	**2**	?	1	No
I4	**3**	3	**4**	?	2	Yes
I5	**4**	**9**	**7**	?	3	Yes
I6	1	0	1	?	0	No
I7	**6**	2	**8**	?	2	Yes
I8	2	3	1	?	0	No
Threshold	2.3	3.6	1.2			

the accuracy of the prediction model, we apply metric selection in the proposed approach TCLP (**T**hreshold **C**lustering **L**abelling **P**lus).

A metric-threshold violation (MTV) is when a software metric value exceeds the threshold and indicates a fault-prone instance, but the class has been labeled as non–faulty and vice-versa. The metric value 3 in M1 of the instance I2 (Table 4.3) is greater than the threshold (2.3), but I2 has been labeled as non–faulty, so such a case is known as MTV. The metric value 2 in M2 of instance I7 (labeled as faulty) is less than the threshold 3.6, so it is another case of MTV. So, for dataset $DS[1:n, 1:m]$, the metric violation ratio (MVR_i) is defined as the ratio of the metric violation score (MVS_i) and the total number of instances (n). MVS_i is

Algorithm 4: To calculate metric violation ratio (MVR)

Input: Dataset $DS[1:n, 1:m]$
Vector $DL[1:n]$
Vector $DT[1:m]$ // Metric threshold obtained from Section (4.2.1)
Output: A vector $MVR[1:m]$
Algorithmic Steps:
Step 1: Initialize the input DS as $DS[X]_{n*m}$
Step 2: Calculate MVR of each metrics

```
for i ← 1 to m do
    cy ← 0 //No. of violated metric values for 'YES'
    cn ← 0 // No. of violated metric values for 'NO'
    for j ← 1 to n do
        if (DL[j] ==' YES') then
            if (DS[j,i] <= DT[i]) then
                r=1
            else
                r=0
            cy=cy+r
        if (DL[j] ==' NO') then
            if (DS[j,i] > DT[i]) then
                s=1
            else
                s=0
            cn=cn+s
    end
    MVS[i] ← cy + cn
    MVR[i] ← MVS[i]/n
end
```

Table 4.4: Dataset after labeling and metric violation ratio

Instance	M1	M2	M3	Label
I1	2	3	**2**	No
I2	**3**	2	1	No
I3	1	1	**2**	No
I4	3	**3**	4	Yes
I5	4	9	7	Yes
I6	1	0	1	No
I7	6	**2**	8	Yes
I8	2	3	1	No
Threshold	2.3	3.6	1.2	
MVR	1/8	2/8	2/8	

TABLE 4.5: Dataset after selected metrics and instance violation ratio

Instance	M1	Label	IVR
I1	2	No	0/1
I2	**3**	No	1/1
I3	1	No	0/1
I4	3	Yes	0/1
I5	4	Yes	0/1
I6	1	No	0/1
I7	6	Yes	0/1
I8	2	No	0/1
Threshold	2.3		
MVR	1/8		

Algorithm 5: To calculate instance violation ratio (IVR)

Input: Dataset $DS[1:n, 1:p]$ // A DS after Algorithm 4
Vector $DL[1:n]$
Vector $DT[1:s]$
Output: A vector $IVR[1:n]$
Algorithmic Steps:
Step 1: Initialize input dataset as $DS[X]_{n*p}$
Step 2: Calculate IVR of each instances

```
for i ← 1 to n do
    iy ← 0 //No. of violated instance values for 'YES'
    in ← 0 //No. of violated instance values for 'NO'
    for j ← 1 to p do
        if (DL[i] ==' YES') then
            if (DS[i, j] <= DT[j]) then
                r = 1
            else
                r = 0
            iy = iy + r
        if (DL[i] ==' NO') then
            if (DS[i, j] > DT[j]) then
                s = 1
            else
                s = 0
            in = in + s
    end
    IVS[i] ← iy + in
    IVR[i] ← IVS[i]/p
end
```

TABLE 4.6: Dataset after selected metrics and instances (Training DS)

Instance	M1	Label
I1	2	No
I3	1	No
I4	3	Yes
I5	4	Yes
I6	1	No
I7	6	Yes
I8	2	No

defined as the total no. of metric values violated in metric i. So, the metric violation ratio (MVR_i) is calculated using (4.5), and the details are given in Algorithm 4.

$$MVR_i = MVS_i/n \quad \forall \quad i = 1, 2, 3 \tag{4.5}$$

In Table 4.4, bold font represents the metric violation values. Table 4.4 shows the MVR of each metric in the last row. For example, the MVR of M1 (MVR_1) will be 1/8 (=0.125), where MVS_i is 1 and n is 8. Using (4.5), we can calculate the MVR of all the other metrics. Then select the metrics that have minimum MVR. In Table 4.4, only the M1 metric has minimum MVR, so select the metric M1. Hence, after selecting the metric M1, the dataset $DS[1:n, 1:m]$ is reduced and becomes $DS[1:n, 1:p]$ where p is the total number of selected metrics. For the next step of instance selection, reduced dataset $DS[1:n, 1:p]$ becomes the input DS.

Instance selection: Many researchers have widely used the instance selection method to develop the prediction model [217–219]. After selecting the metrics, we still found some instances-threshold violations (ITV) existing in the DS. E.g., in Table 4.5, the M1 value 3 of I2 is greater than threshold 2.3; it should be label as 'YES', but I2 is labeled as 'NO'. So, it is a case of ITV. Hence, instances also violate the fault proneness tendency. So, choosing the informative instances means minimizing the ITV of the DS by removing irrelevant data points. So, the instance violation ratio (IVR_j) is defined as the ratio of the instance violation score (IVS_j) and the total number of metrics (p). IVS_j is defined as the total no. of metric

values violated in instance j. So, the instance violation ratio (IVR_j) is calculated using (4.6) and the details are given in Algorithm 5.

$$IVR_j = IVS_j/p \quad \forall \quad j = 1, 2, 3, 4, 5, 6, 7, 8 \tag{4.6}$$

For example, IVR of I2 (IVR_2) is 1/1 (=1), where IVS_j is 1 and p is also 1. From Table 4.5, only instance I2 is violated. So, we select all the instances with the minimum IVR (0), i.e., I1, I3 to I8. After selecting the instances, dataset $DS[1:n, 1:p]$ is reduced and becomes $DS[1:q, 1:p]$ where q is the total number of instances selected and p is the total number of metrics selected. Table 4.6 shows the final training dataset.

This selection process of selecting metrics and instances using minimum MVR and IVR may cause problems such as selecting only a single metric or data points that belong just to a single class. But, we need a balanced DS in order to build a prediction model that is general enough to be used on other DSs and gives high accuracy. In this situation, we have included more metrics by selecting the next minimum metric violation ratio (MVR) and, accordingly, IVR. In the given example, the condition is same as selected metrics is one. Hence, now select the metrics M2 and M3 with next minimum MVR.

In this experimental implementation of TCLP, we have performed rigorous experiments and have found that TCLP is performing superior on selecting minimum $\lceil log_2 m \rceil$ metrics [220] with minimum MVR and minimum 70% of instances with minimum IVR with the condition that at least two faulty instances should be selected. After selecting minimum $\lceil log_2 m \rceil$ metrics and minimum 70% instances, if we get zero or only one faulty instance, then we select 80% of the instances with minimum IVR. This process is repeated till 99% of the instances are selected. The metric selection and instance selection are helpful to minimize MTV and ITV. After metrics selection and instances selection, we obtain reduced and clean dataset

$DS[1:q, 1:p]$. Now, this DS is used as the input DS (training DS) to the machine learning algorithm.

4.2.4 Learning and Prediction

Initially, we have an unlabeled dataset $DS[1:n, 1:m]$ that is labeled using the proposed method TCL (Algorithm 3). After labeling the data points using TCL, the DS is refined by removing MTV and ITV on the basis of MVR and IVR. Now, we have refined and labeled dataset $DS[1:q, 1:p]$, which closely follows the fault proneness tendency concept. But, this refined dataset $DS[1:q, 1:p]$ is highly imbalanced. So, to balance this DS, ROSE (Random oversampling example) techniques are used to balance the DS. We have chosen this method because ROSE outperforms SMOTE in case of extreme levels of imbalance and small sample sizes [54]. To leverage the use of a machine learning algorithm, we have built a SFP model TCLP on refined and balanced DS (Algorithm 6). When we provide this refined training DS (most representative metrics and instances) to the machine learning algorithm, it naturally understands the training better.

We have used the Random forest algorithm with 10-fold cross-validation to train the model. In the random forest algorithm, the main tuning parameter is mtry (no. of variables/predictors randomly sampled as candidates at each split). We have got the optimal mtry using a graph between accuracy (cross-validation) and randomly selected predictors. Other parameters are default (e.g.,ntree (number of tree)=500) [205]. When we increase mtry, the performance of random forest algorithm decreases. Based on recall value (almost equal for both the classes) of the model, we have found that the model is not overfitted. Random forest has consistently lower generalization error, accuracy is as good as Adaboost, is robust to outliers and noise, and is faster than bagging or boosting [221]. Random forest algorithm involves a lot of randomization and, therefore, produces different results on each run. To obtain reproducible results, one needs to initialize the random number generator with a fixed number (set.seed(1)).

Algorithm 6: To implement TCLP (Threshold Clustering Labeling Plus)

Input: A dataset $DS[1:n, 1:m]$
A vector $DT[1:m]$
A vector $DL[1:n]$
Output: Predicted labels of DS and performance measures of TCLP
Algorithmic Steps:
Step 1: Calculate MVR using Algorithm 4 and Select all the metrics with the minimum MVR (Section 4.2.3)
Step 2: Calculate IVR using Algorithm 5 and Select all the instances with minimum IVR (Section 4.2.3)
Step 3: Initialize: f =no. of faulty instances in $RD[1:q, 1:p]$
while *($f < 2$)* **do**
  if *(selected_instances $< q$)* **then**
    Add more instances in $RD[1:q, 1:p]$ with next minimum IVR
    Again calculate f in RD
  else
    Exit
end
if *($f < 2$)* **then**
  Go to Step 1 and add more metrics with next minimum MVR
  Repeat Step 2 and Step 3
else
  Go to Step 4
Step 4: Train a ML model with random forest algorithm on RD with label DL
$Train\_Model \leftarrow train_{rf}(Data = (RD, DL))$
$Predict\_label \leftarrow predict(Train\_Model, DS[1:n, 1:m])$
Step 5: In Step 4, RF yields the performance on actual DS $DS[1:n, 1:m]$

After getting the trained model, we passed the original unlabelled DS to the model and predicted the label for each data point. Finally, we have evaluated our proposed approach to TCLP using three performance parameters, i.e., accuracy, FM, and MCC. The main goal of proposing TCLP with TCL is to take advantage of machine learning techniques. TCL can label all the instances, but there is a chance of wrong labeling due to MTV and ITV. We have applied metric selection and instance selection to minimize MTV and ITV by removing the violating metrics and instances. Further, we trained the machine learning model on refined DSs (consisting of the most effective metrics and instances) and predicted the performance of the TCLP (machine learning model) model on original DSs. The pseudo-code of the proposed approach TCL/TCLP is written as Algorithm 6.

4.3 Experimental Design

The core contribution in the proposed approach TCL/TCLP is about 'automated labeling'. This gives rise to an important question that the labels provided by our approach are how much closer to those that are provided by human experts. To answer this question, we considered 28 projects from 5 repositories [1] where the DSs are labeled by human experts. To effectively compare the labeling of our approach with that of human experts, we designed research questions that are described in the following subsections.

4.3.1 Research Questions

RQ1: Is the performance of TCL/TCLP approach comparable to the performance obtained using the standard supervised learning approaches?
To answer this question, we have compared the performance of the TCL/TCLP with the five supervised learning models (Naive Bayes (NB), Support vector machine with RBF kernel (SVM), K nearest neighbor (KNN), Random forest (RF) and C5.0 (C50)). All these algorithms are implemented using caret package of R library [205]. If TCL/TCLP is comparable to these supervised algorithms, then it can be used in industry for labeled software projects.

RQ2: Is the performance of TCL/TCLP approach comparable to the performance obtained using the standard unsupervised learning approaches?
We have compared the TCL/TCLP approach with the well-known four unsupervised standard algorithms (K-means (KMS) , Neural gas clustering (NGC), Model based clustering (MBC), Hierarchical clustering (HC)) for unlabelled DSs [115]. These algorithms are implemented in RStudio. If TCL/TCLP is comparable to these unsupervised algorithms, then it can be widely used in practice for SFP in unlabelled DSs.

[1] `https://github.com/klainfo/DefectData`

RQ3: Is the performance of TCL/TCLP approach comparable to the existing threshold-based state-of-the-art techniques?
To answer this question, we have compared the TCL/TCLP approach with the three threshold-based state-of-the-art approaches (CLA/CLAMI, CLAMI+, Average Clustering Label (ACL)) for unlabelled DSs. If we get better results than these state-of-the-art techniques, then TCL/TCLP approach will prove to be more beneficial for the software industry.

RQ4: Is the performance of TCLP approach without sampling technique comparable to that of TCLP with sampling technique (SMOTE)?
To answer this question, we have implemented the TCLP approach both with and without sampling technique (SMOTE) applied to the original DS and have compared the obtained results.

4.3.2 Datasets (DSs)

Table 2.3 shows the 28 benchmark DSs collected from the five repositories AEEEM [25, 28], SOFTLAB [27], NASA [178], MORPH [180], and ReLink [179]. We have selected only five NASA DSs that have common software metrics as features [181]. These benchmark DSs are used to validate the proposed approach TCL/TCLP and perform its comparative analysis with the existing techniques.

4.3.3 Experimental Baselines

To compare the performance of our proposed approach TCL/TCLP and answer our three research questions, we have implemented five supervised algorithms, four unsupervised algorithms, and three threshold-based methods.

- **Benchmark 1:** As benchmark technique 1, we have built five supervised learning models on labeled DSs with a 2-fold cross-validation method. Typical SFP models developed for single software projects use the supervised learning techniques [19, 24, 28, 30, 59, 112, 113]. Fig. 4.1 shows a step-by-step process to

build the TCL/TCLP model. We have compared the prediction performance of the proposed model TCL/TCLP with the typical supervised learning algorithms. The results of this comparison are used to answer RQ1.

- **Benchmark 2:** As benchmark technique 2, we have developed four SFP models using unsupervised algorithms. We have compared the results of TCL/TCLP with these four unsupervised algorithms. This comparison answers RQ2.
- **Benchmark 3:** As benchmark technique 3, we have developed three SFP models using three threshold-based techniques. These are the state-of-the-art threshold-based techniques to predict the fault on unlabelled DSs. So, we have compared the performance of TCL/TCLP with these unsupervised techniques on the same DS with unlabelled instances. This comparison answers RQ-3.
- **Benchmark 4:** As the considered DSs have class imbalance problems. So, we have implemented a sampling technique SMOTE to balance the DSs. Then, we developed a SFP model TCLP on the balanced DS. We have compared the results of TCLP with SMOTE to that of TCLP without SMOTE to answer RQ4.

4.3.4 Performance Measures

To compare the performance of the proposed approach TCL/TCLP and other benchmark techniques, we have computed accuracy using (2.2), FM using (2.4), and Mathew's correlation coefficients (MCC) using (2.5) as performance parameters. The detailed description is provided in Section 2.4.3.1. Two-way statistical evaluation has been performed for strong comparison. Firstly, we applied Wilcoxon signed-rank test to compare the different approaches statistically. Secondly, to find the overall statistically significant differences in performance across the six different approaches, we have employed the Friedman test [193] followed by the Nemenyi test [192] suggested by Demsar [194].

4.4 Experimental Results

This section will provide the results of our experiments using three performance measures (details in Section 4.3.4). Table 4.7, Table 4.8, and Table 4.9 show the performance of TCL/TCLP model as compared to the existing approaches in terms of accuracy, FM, and MCC over 28 DSs. In Table 4.7, Table 4.8, Table 4.9, the last rows of each group DSs provide the median values of the different approaches. The median value of the proposed approach is shown in bold wherever it is greater than or equal to the other approaches. The performance value of supervised, unsupervised, and threshold-based methods are evaluated using complete DSs. Additionally, the proposed algorithm TCLP is heuristically minimizing the MVR and IVR for each metric and each instance individually. The metrics selected using MVR work as input features for IVR. So, if we select a larger number of metrics, then both MVR and IVR are likely to increase. Due to this, there will be more metrics and instances removal, which will lead to enhanced data imbalance. After rigorous experiments, we found that TCLP performance is superior when trained on selecting $\lceil log_2 m \rceil$ metrics and removing maximum 30% instances as per minimum MVR and IVR [220]. The result of TCL/TCLP shown in Table 4.7, Table 4.8, and Table 4.9 are based on prediction results on original DS (means all the data points of each DS are predicted using TCLP model) (Algorithm 6). The graphical comparison using boxplot over all DSs is shown in Fig. 4.2 in terms of accuracy, FM, and MCC.

In terms of accuracy and FM, the proposed method is comparable to the supervised algorithms (Naive Bayes (NB), Support vector machine (SVM), K-nearest neighbor (KNN), Random forest (RF) and C5.0 (C50), unsupervised algorithm (HC) and completely superior to other threshold-based methods (CLA/CLAMI, CLAMI+ and ACL). The best performance is given by RF and Hierarchical clustering (HC) among the considered supervised and unsupervised algorithms, respectively. TCL/TCLP are performing better than other threshold-based methods. In terms of MCC, the Supervised algorithm is superior to all other algorithms used.

TABLE 4.7: Performance of different SFP techniques in terms of accuracy

Groups	Projects	Accuracy %														
		Supervised Methods					Unsupervised Methods				Threshold Based Methods					
		NB	SVM	KNN	RF	C50	KMS	NGC	MBC	HC	CLA	CLAMI	CLAMI+	ACL	TCL	TCLP
AEEEM	Equinox	73.15	72.24	70.39	75.12	76.24	63.89	63.89	37.35	63.58	68.83	68.83	70.68	71.60	67.28	68.21
	JDT	82.35	83.11	84.85	86.22	84.51	79.84	79.84	67.80	79.54	48.85	55.37	57.07	68.71	83.65	81.64
	Lucene	85.82	90.81	90.63	90.82	90.60	91.17	90.59	76.12	90.88	36.03	38.93	52.53	60.20	85.96	87.99
	Mylyn	84.37	87.07	86.06	86.94	86.81	86.73	86.79	73.15	86.73	37.65	38.83	51.07	60.69	84.69	87.16
	PDE	83.97	86.20	85.16	86.24	85.56	84.70	84.90	72.28	86.11	42.55	46.96	54.11	59.99	81.70	84.64
	MED	83.97	86.20	85.16	86.24	85.56	84.70	84.90	72.28	86.11	42.55	46.96	54.11	60.69	83.65	84.64
SOFTLAB	AR1	84.30	92.01	92.13	90.38	91.67	92.56	90.08	75.21	92.56	53.72	39.67	57.02	53.72	85.95	84.30
	AR3	87.30	86.29	87.24	91.57	87.98	88.89	87.30	68.25	92.06	60.32	57.14	60.32	61.90	88.89	88.89
	AR4	85.05	83.02	83.91	84.75	79.45	85.05	85.05	73.83	82.24	61.68	58.88	64.49	57.94	87.85	87.85
	AR5	86.11	78.00	79.17	84.00	84.63	77.78	80.56	86.11	77.78	75.00	75.00	75.00	75.00	86.11	86.11
	AR6	87.13	84.20	85.05	88.30	87.21	78.22	78.22	72.28	83.17	52.48	37.62	53.47	57.43	80.20	75.25
	MED	86.11	84.20	85.05	88.30	87.21	85.05	85.05	73.83	83.17	60.32	57.14	60.32	57.94	86.11	86.11
NASA	CM1	82.57	86.37	86.24	85.55	85.83	86.54	86.85	71.25	86.54	48.93	36.39	55.35	47.40	80.73	80.73
	MW1	79.45	89.58	89.12	87.88	88.09	87.75	81.03	64.03	88.93	42.69	51.38	54.94	41.11	85.38	83.00
	PC1	88.37	91.37	91.12	92.29	91.51	91.35	91.49	70.78	91.49	47.52	51.06	53.05	49.93	83.97	80.00
	PC3	79.02	87.33	86.87	87.53	86.54	87.47	87.37	76.79	87.47	52.00	53.30	56.64	53.30	81.24	85.79
	PC4	87.96	90.04	85.24	89.87	89.13	85.31	84.93	64.80	86.32	51.75	54.08	56.95	55.48	79.10	76.69
	MED	82.57	89.58	86.87	87.88	88.09	87.47	86.85	70.78	87.47	48.93	51.38	55.35	49.93	81.24	80.73
MORPH	Ant_1.3	82.40	82.81	82.46	78.23	82.12	84.00	81.60	81.60	84.00	39.20	57.60	60.00	50.40	84.00	82.40
	Arc	87.18	87.29	87.95	86.30	87.10	87.61	87.18	74.36	88.03	21.37	21.79	49.57	52.99	85.90	85.47
	Camel_1.0	92.63	95.71	96.12	95.40	95.94	94.69	94.69	75.22	95.87	21.83	27.43	47.20	41.30	89.38	89.38
	Poi_1.5	74.68	73.35	69.92	75.14	72.46	41.35	41.77	49.79	41.35	66.24	63.71	65.82	67.09	46.84	52.74
	Redaktor	77.27	89.75	85.37	89.74	88.86	76.70	76.70	62.50	83.52	21.02	29.55	47.16	39.77	81.25	79.55
	Skarbonka	91.11	80.90	79.20	65.70	76.03	73.33	71.11	44.44	73.33	26.67	53.33	64.44	68.89	68.89	71.11
	Tomcat	86.13	90.07	90.02	91.42	90.79	91.14	90.91	73.89	91.14	18.07	18.76	51.40	59.91	88.69	90.91
	Velocity_1.4	90.31	86.62	75.95	85.57	83.26	23.98	23.98	41.33	23.98	73.47	68.88	52.55	51.53	29.59	33.67
	Xalan_2.4	81.60	85.06	85.04	85.25	83.84	85.20	85.20	71.92	84.51	45.92	67.50	58.64	57.12	82.85	82.85
	Xerces_1.2	79.77	83.83	82.78	84.24	83.60	83.64	81.59	54.09	83.64	16.36	16.36	40.91	51.36	82.73	82.73
	MED	84.27	85.84	83.91	85.41	83.72	83.82	81.60	67.21	83.82	24.25	41.44	51.97	52.26	82.79	82.56
ReLink	Apache	77.32	70.73	72.80	78.61	76.09	50.00	50.00	66.49	50.00	70.62	69.07	70.62	71.65	57.73	58.76
	Safe	82.14	66.13	71.61	68.71	68.47	73.21	80.36	33.93	66.07	75.00	73.21	73.21	75.00	78.57	78.57
	Zxing	68.67	71.79	70.75	70.71	69.91	69.17	69.17	59.65	70.18	56.64	62.41	58.65	58.40	69.42	68.42
	MED	77.32	70.73	71.61	70.71	69.91	69.17	69.17	59.65	66.07	70.62	69.07	70.62	71.65	69.42	68.42

The performance of TCL/TCLP exceeds that of unsupervised algorithms, CLA/-CLAMI, and is almost equivalent to CLAMI+ and ACL.

The Wilcoxon signed-rank test (a non-parametric statistical test) has been applied to compare the approaches. Mean differences among the different approaches and p-value with statistical significance (Yes/No) (analyzed based on p-value<0.05) are shown in Table 4.10. In Table 4.10, the Mean difference is the difference between the average performance of the algorithms over all the DSs. The left bottom triangle in Table 4.10 shows the difference between column and row algorithms. The p-value calculated using Wilcoxon signed-rank test is used to represent the statistically significant difference between the two approaches. The right upper triangle shows the

TABLE 4.8: Performance of different SFP techniques in terms of FM

Groups	Projects	FM														
		Supervised Methods					Unsupervised Methods				Threshold Based Methods					
		NB	SVM	KNN	RF	C50	KMS	NGC	MBC	HC	CLA	CLAMI	CLAMI+	ACL	TCL	TCLP
AEEEM	Equinox	0.80	0.77	0.76	0.74	0.77	0.77	0.17	0.54	0.77	0.67	0.72	0.73	0.76	0.78	0.77
	JDT	0.90	0.90	0.91	0.92	0.91	0.89	0.05	0.77	0.89	0.55	0.63	0.65	0.78	0.90	0.88
	Lucene	0.92	0.95	0.95	0.95	0.95	0.95	0.08	0.86	0.95	0.47	0.51	0.66	0.73	0.92	0.94
	Mylyn	0.91	0.93	0.92	0.93	0.93	0.93	0.93	0.83	0.92	0.46	0.47	0.63	0.73	0.91	0.93
	PDE	0.91	0.92	0.92	0.92	0.92	0.92	0.17	0.30	0.93	0.52	0.57	0.65	0.72	0.90	0.92
	MED	0.91	0.92	0.92	0.92	0.92	0.92	0.17	0.77	0.92	0.52	0.57	0.65	0.73	0.90	**0.92**
SOFTLAB	AR1	0.91	0.96	0.96	0.95	0.96	0.18	0.95	0.21	0.96	0.67	0.52	0.70	0.67	0.92	0.91
	AR3	0.92	0.92	0.93	0.95	0.93	0.53	0.50	0.38	0.96	0.71	0.68	0.71	0.73	0.93	0.93
	AR4	0.91	0.90	0.91	0.91	0.87	0.91	0.38	0.83	0.90	0.70	0.67	0.73	0.66	0.93	0.93
	AR5	0.91	0.85	0.86	0.88	0.89	0.87	0.46	0.74	0.87	0.81	0.81	0.81	0.81	0.91	0.91
	AR6	0.93	0.91	0.92	0.92	0.92	0.87	0.27	0.83	0.91	0.63	0.43	0.65	0.69	0.88	0.85
	MED	0.91	0.91	0.92	0.92	0.92	0.87	0.46	0.74	0.91	0.70	0.67	0.71	0.69	**0.92**	0.91
NASA	CM1	0.90	0.93	0.93	0.92	0.92	0.93	0.04	0.82	0.93	0.61	0.43	0.67	0.58	0.89	0.89
	MW1	0.88	0.94	0.94	0.93	0.94	0.11	0.89	0.76	0.94	0.55	0.64	0.68	0.52	0.92	0.90
	PC1	0.94	0.95	0.95	0.96	0.95	0.03	0.06	0.82	0.96	0.61	0.64	0.66	0.63	0.91	0.88
	PC3	0.87	0.93	0.93	0.93	0.93	0.93	0.93	0.86	0.93	0.63	0.65	0.68	0.65	0.89	0.92
	PC4	0.93	0.94	0.92	0.94	0.94	0.92	0.92	0.76	0.93	0.63	0.66	0.69	0.67	0.88	0.86
	MED	0.90	0.94	0.93	0.93	0.94	0.92	0.89	0.82	0.93	0.61	0.64	0.68	0.63	0.89	0.89
MORPH	Ant_1.3	0.89	0.90	0.90	0.87	0.90	0.09	0.49	0.51	0.91	0.44	0.67	0.70	0.59	0.91	0.89
	Arc	0.93	0.93	0.93	0.92	0.93	0.36	0.93	0.85	0.94	0.20	0.21	0.61	0.66	0.92	0.92
	Camel_1.0	0.96	0.98	0.98	0.98	0.98	0.97	NA	0.86	0.98	0.32	0.39	0.62	0.57	0.94	0.94
	Poi_1.5	0.71	0.63	0.61	0.69	0.64	0.58	0.58	0.56	0.58	0.57	0.61	0.58	0.55	0.59	0.60
	Redaktor	0.85	0.94	0.92	0.94	0.94	0.13	0.87	0.76	0.91	0.13	0.29	0.59	0.47	0.89	0.88
	Skarbonka	0.94	0.89	0.88	0.82	0.86	0.25	0.32	0.07	0.84	0.15	0.59	0.71	0.76	0.81	0.82
	Tomcat	0.92	0.95	0.95	0.95	0.95	0.03	0.95	0.84	0.95	0.19	0.20	0.64	0.72	0.94	0.95
	Velocity_1.4	0.77	0.69	0.38	0.65	0.61	0.39	0.39	0.48	0.39	0.21	0.21	0.30	0.32	0.38	0.40
	Xalan_2.4	0.89	0.92	0.92	0.92	0.91	0.92	0.92	0.82	0.92	0.54	0.77	0.68	0.67	0.90	0.90
	Xerces_1.2	0.88	0.91	0.90	0.91	0.91	0.91	0.13	0.66	0.91	0.04	0.04	0.53	0.65	0.90	0.90
	MED	0.89	0.91	0.91	0.91	0.91	0.37	0.58	0.71	0.91	0.21	0.34	0.62	0.62	0.90	0.90
ReLink	Apache	0.78	0.69	0.71	0.80	0.72	0.66	0.04	0.61	0.66	0.70	0.64	0.70	0.71	0.69	0.69
	Safe	0.86	0.74	0.77	0.72	0.79	0.48	0.67	0.45	0.78	0.77	0.75	0.76	0.77	0.85	0.85
	Zxing	0.77	0.83	0.81	0.81	0.80	0.80	0.80	0.44	0.82	0.61	0.71	0.65	0.65	0.81	0.79
	MED	0.78	0.74	0.77	0.80	0.79	0.66	0.67	0.45	0.78	0.70	0.71	0.70	0.71	**0.81**	0.79

p-value between two algorithms. If the p-value is greater than 0.05, it shows no significant difference between the models. If the p-value is less than 0.05, it shows that the pair of models are significantly different [191].

The critical diagram of the post hoc Nemenyi test is shown in Fig. 4.3. This diagram contains Friedman's p-value (with a hypothesis (Ha) as significantly different), critical distance value (4.053 and 4.128), and mean score of each approach. The approach name followed by the mean score is shown on X-axis in increasing order. This means the top-performing method is shown on the right-most side of the diagram. The test is simple. The stick length (in Fig. 4.3) of an approach

TABLE 4.9: Performance of different SFP techniques in terms of MCC

Groups	Projects	MCC														
		Supervised Methods					Unsupervised Methods				Threshold Based Methods					
		NB	SVM	KNN	RF	C50	KMS	NGC	MBC	HC	CLA	CLAMI	CLAMI+	ACL	TCL	TCLP
AEEEM	Equinox	0.43	0.44	0.42	0.38	0.46	0.24	0.24	-0.44	0.23	0.47	0.38	0.44	0.42	0.31	0.31
	JDT	0.37	0.44	0.51	0.55	0.56	0.14	0.14	0.27	0.09	0.22	0.25	0.25	0.32	0.44	0.45
	Lucene	0.17	0.19	0.00	0.27	0.28	0.21	0.12	0.21	0.12	0.11	0.11	0.14	0.16	0.18	0.02
	Mylyn	0.25	0.21	0.16	0.26	0.27	0.08	0.13	0.18	-0.01	0.14	0.14	0.13	0.18	0.24	0.19
	PDE	0.20	0.19	0.05	0.25	0.16	0.13	0.14	0.16	0.06	0.14	0.17	0.19	0.20	0.18	0.15
	MED	0.25	0.21	0.16	0.27	0.28	0.14	0.14	0.18	0.09	0.14	0.17	0.19	0.20	0.24	0.19
SOFTLAB	AR1	0.24	0.00	0.00	0.04	0.00	0.21	0.35	0.14	0.21	0.21	0.20	0.23	0.21	0.26	0.24
	AR3	0.54	0.29	0.36	0.62	0.54	0.47	0.43	0.29	0.59	0.29	0.27	0.29	0.30	0.58	0.58
	AR4	0.55	0.41	0.44	0.55	0.54	0.40	0.40	0.34	0.20	0.38	0.36	0.38	0.32	0.56	0.56
	AR5	0.66	0.36	0.66	0.52	0.68	0.16	0.36	0.66	0.16	0.57	0.57	0.57	0.57	0.62	0.66
	AR6	0.39	0.09	0.00	0.25	0.32	0.14	0.14	0.19	0.13	0.20	0.18	0.17	0.20	0.30	0.22
	MED	0.54	0.29	0.36	0.52	0.54	0.21	0.36	0.29	0.20	0.29	0.27	0.29	0.30	0.56	**0.56**
NASA	CM1	0.24	-0.02	0.00	-0.04	0.31	-0.03	0.06	0.19	-0.03	0.14	0.18	0.18	0.17	0.20	0.24
	MW1	0.30	0.35	0.00	0.21	0.31	0.08	0.09	0.21	-0.02	0.08	0.16	0.17	0.12	0.36	0.31
	PC1	0.29	0.26	0.00	0.44	0.44	0.08	0.13	0.18	0.13	0.16	0.20	0.17	0.19	0.20	0.15
	PC3	0.34	0.17	-0.02	0.32	0.23	-0.01	-0.02	0.22	-0.01	0.22	0.23	0.24	0.23	0.13	-0.03
	PC4	0.42	0.54	0.01	0.56	0.55	0.05	0.04	0.19	0.07	0.18	0.17	0.19	0.22	0.10	0.13
	MED	0.30	0.26	0.00	0.32	0.31	0.05	0.06	0.19	-0.01	0.16	0.18	0.18	0.19	0.20	0.15
MORPH	Ant_1.3	0.40	0.07	0.35	0.39	0.32	0.12	0.38	0.41	0.12	0.20	0.31	0.33	0.25	0.33	0.49
	Arc	0.35	0.08	0.28	0.07	0.24	0.30	0.28	0.05	0.13	0.12	0.12	0.17	0.13	0.30	0.33
	Camel_1.0	0.35	0.30	0.26	0.32	0.29	-0.02	-0.02	0.13	-0.01	0.09	0.11	0.15	0.04	0.05	0.05
	Poi_1.5	0.49	0.47	0.40	0.53	0.45	0.04	0.06	0.10	0.04	0.29	0.29	0.29	0.30	0.13	0.20
	Redaktor	0.45	0.54	0.35	0.63	0.55	0.00	0.00	-0.11	-0.05	0.10	0.17	0.03	0.13	0.07	0.04
	Skarbonka	0.72	0.30	0.00	0.22	0.00	0.09	0.13	-0.29	0.09	0.13	0.35	0.45	0.49	-0.06	0.06
	Tomcat	0.28	0.33	0.02	0.36	0.31	0.11	0.08	0.30	0.11	0.06	0.07	0.23	0.25	0.33	0.16
	Velocity_1.4	0.73	0.65	0.32	0.61	0.58	-0.18	-0.18	-0.06	-0.18	0.12	0.03	-0.02	-0.01	-0.03	0.06
	Xalan_2.4	0.26	0.25	0.32	0.31	0.37	0.26	0.26	0.25	0.05	0.24	0.35	0.32	0.27	0.23	0.23
	Xerces_1.2	0.14	0.14	-0.02	0.28	0.34	0.09	0.07	0.09	0.09	-0.14	-0.14	-0.09	-0.06	0.20	0.19
	MED	0.37	0.30	0.30	0.34	0.33	0.09	0.08	0.10	0.07	0.12	0.15	0.20	0.19	0.17	0.18
ReLink	Apache	0.55	0.43	0.50	0.54	0.52	0.04	0.04	0.35	0.04	0.41	0.39	0.41	0.43	0.24	0.25
	Safe	0.62	0.39	0.51	0.55	0.54	0.47	0.61	-0.37	0.30	0.51	0.48	0.47	0.51	0.58	0.58
	Zxing	0.26	0.21	0.35	0.33	0.30	0.13	0.13	0.14	-0.03	0.24	0.18	0.22	0.21	0.14	0.17
	MED	0.55	0.39	0.50	0.54	0.52	0.13	0.13	0.14	0.04	0.41	0.39	0.41	0.43	0.24	0.25

is within plus/minus the critical distance (CD) from the mean score. If the stick length of an approach is nearly parallel to the other approach stick length (mean score plus/minus CD), then there is no evidence of differences. The average score of different approaches is shown in two colors in the critical diagram. A blue stick with a red dot represents worse performance (shaded background) than a blue stick with a black dot.

TABLE 4.10: Wilcoxon signed-rank test analysis (p<0.05): Mean difference and p-value among the performance of different approaches

(a) Accuracy (%)

Mean difference (Column-Row)(Left bottom triangle), and p-value (Right upper triangle)

	NB	SVM	KNN	RF	C50	KMS	NGC	MBC	HC	CLA	CLAMI	CLAMI+	ACL	TCL	TCLP
NB	0.00	0.21	0.52	0.02	0.09	0.64	0.57	0.00	0.97	0.00	0.00	0.00	0.00	0.17	0.15
SVM	-0.84	0.00	0.24	0.26	0.54	0.05	0.01	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00
KNN	-0.18	0.66	0.00	0.05	0.33	0.13	0.02	0.00	0.13	0.00	0.00	0.00	0.00	0.00	0.00
RF	-1.23	-0.39	-1.05	0.00	0.08	0.03	0.01	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00
C50	-0.93	-0.09	-0.75	0.29	0.00	0.13	0.01	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00
KMS	4.53	5.37	4.71	5.76	5.46	0.00	0.21	0.00	0.21	0.00	0.00	0.00	0.00	0.63	0.64
NGC	4.82	5.66	5.00	6.05	5.75	0.29	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.83	1.00
MBC	17.46	18.30	17.64	18.69	18.39	12.93	12.64	0.00	0.00	0.00	0.00	0.02	0.03	0.00	0.00
HC	4.18	5.02	4.36	5.41	5.11	-0.35	-0.64	-13.28	0.00	0.00	0.00	0.00	0.00	0.38	0.41
CLA	35.56	36.40	35.74	36.79	36.49	31.03	30.74	18.10	31.38	0.00	0.22	0.00	0.00	0.00	0.00
CLAMI	33.40	34.24	33.58	34.62	34.33	28.87	28.58	15.94	29.21	-2.17	0.00	0.00	0.01	0.00	0.00
CLAMI+	25.55	26.39	25.72	26.77	26.48	21.01	20.72	8.09	21.36	-10.02	-7.85	0.00	0.75	0.00	0.00
ACL	25.30	26.14	25.48	26.52	26.23	20.77	20.48	7.84	21.11	-10.26	-8.10	-0.25	0.00	0.00	0.00
TCL	4.99	5.83	5.16	6.21	5.92	0.45	0.16	-12.48	0.80	-30.58	-28.41	-20.56	-20.31	0.00	0.59
TCLP	4.76	5.60	4.94	5.99	5.69	0.23	**-0.06**	**-12.70**	0.58	**-30.80**	**-28.63**	**-20.78**	**-20.54**	**-0.22**	0.00

(b) FM

Mean difference (Column-Row)(Left bottom triangle), and p-value (Right upper triangle)

	NB	SVM	KNN	RF	C50	KMS	NGC	MBC	HC	CLA	CLAMI	CLAMI+	ACL	TCL	TCLP
NB	0.00	0.75	0.69	0.25	0.64	0.00	0.00	0.00	0.63	0.00	0.00	0.00	0.00	0.35	0.28
SVM	0.00	0.00	0.46	0.86	1.00	0.00	0.00	0.00	0.42	0.00	0.00	0.00	0.00	0.01	0.01
KNN	0.01	0.01	0.00	0.40	0.19	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.02	0.02
RF	0.00	0.00	-0.01	0.00	0.63	0.00	0.00	0.00	0.85	0.00	0.00	0.00	0.00	0.02	0.01
C50	0.00	0.00	-0.01	0.00	0.00	0.00	0.00	0.00	0.31	0.00	0.00	0.00	0.00	0.01	0.00
KMS	0.26	0.26	0.25	0.27	0.26	0.00	0.41	0.36	0.00	0.09	0.28	0.74	0.67	0.04	0.02
NGC	0.36	0.36	0.35	0.37	0.36	0.09	0.00	0.27	0.00	0.86	0.61	0.04	0.06	0.00	0.00
MBC	0.23	0.23	0.22	0.23	0.23	-0.03	-0.13	0.00	0.00	0.03	0.10	0.82	0.70	0.00	0.00
HC	0.01	0.01	0.00	0.01	0.01	-0.25	-0.35	-0.22	0.00	0.00	0.00	0.00	0.00	0.03	0.04
CLA	0.38	0.38	0.37	0.38	0.38	0.12	0.00	0.15	0.37	0.00	0.07	0.00	0.00	0.00	0.00
CLAMI	0.34	0.34	0.33	0.34	0.34	0.08	-0.03	0.11	0.33	-0.04	0.00	0.00	0.00	0.00	0.00
CLAMI+	0.23	0.23	0.21	0.23	0.23	-0.04	-0.14	-0.01	0.21	-0.15	-0.12	0.00	0.77	0.00	0.00
ACL	0.22	0.22	0.21	0.23	0.22	-0.04	-0.15	-0.01	0.21	-0.15	-0.12	0.00	0.00	0.00	0.00
TCL	0.02	0.03	0.01	0.03	0.02	-0.24	-0.34	-0.21	0.01	-0.35	-0.32	-0.20	-0.20	0.00	0.44
TCLP	0.03	0.03	0.02	0.03	0.03	**-0.24**	**-0.34**	**-0.20**	0.01	**-0.35**	**-0.32**	**-0.20**	**-0.20**	0.00	0.00

(c) MCC

Mean difference (Column-Row)(Left bottom triangle), and p-value (Right upper triangle)

	NB	SVM	KNN	RF	C50	KMS	NGC	MBC	HC	CLA	CLAMI	CLAMI+	ACL	TCL	TCLP
NB	0.00	0.00	0.00	0.82	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SVM	0.10	0.00	0.13	0.00	0.00	0.00	0.03	0.03	0.00	0.07	0.22	0.43	0.36	0.71	0.52
KNN	0.17	0.07	0.00	0.00	0.00	0.18	0.48	0.34	0.01	0.68	0.91	0.55	0.57	0.37	0.38
RF	0.02	-0.08	-0.15	0.00	0.68	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.01
C50	0.02	-0.09	-0.15	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
KMS	0.26	0.15	0.09	0.23	0.24	0.00	0.09	0.29	0.01	0.02	0.01	0.00	0.00	0.00	0.00
NGC	0.23	0.12	0.06	0.20	0.21	-0.03	0.00	0.37	0.00	0.16	0.04	0.01	0.02	0.00	0.00
MBC	0.25	0.15	0.08	0.23	0.23	-0.01	0.02	0.00	0.07	0.68	0.41	0.14	0.19	0.01	0.03
HC	0.30	0.20	0.13	0.28	0.28	0.04	0.07	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00
CLA	0.18	0.08	0.01	0.16	0.16	-0.07	-0.05	-0.07	-0.12	0.00	0.31	0.05	0.04	0.16	0.28
CLAMI	0.17	0.06	0.00	0.14	0.15	-0.09	-0.06	-0.08	-0.13	-0.01	0.00	0.04	0.26	0.33	0.43
CLAMI+	0.15	0.05	-0.02	0.13	0.13	-0.10	-0.08	-0.10	-0.15	-0.03	-0.02	0.00	0.62	0.32	0.59
ACL	0.15	0.05	-0.02	0.13	0.13	-0.11	-0.08	-0.10	-0.15	-0.03	-0.02	0.00	0.00	0.41	0.79
TCL	0.14	0.03	-0.03	0.11	0.12	-0.12	-0.09	-0.11	-0.16	-0.05	-0.03	-0.02	-0.02	0.00	0.72
TCLP	0.14	0.04	**-0.03**	0.12	0.12	**-0.11**	**-0.09**	**-0.11**	**-0.16**	**-0.04**	**-0.03**	**-0.01**	**-0.01**	0.01	0.00

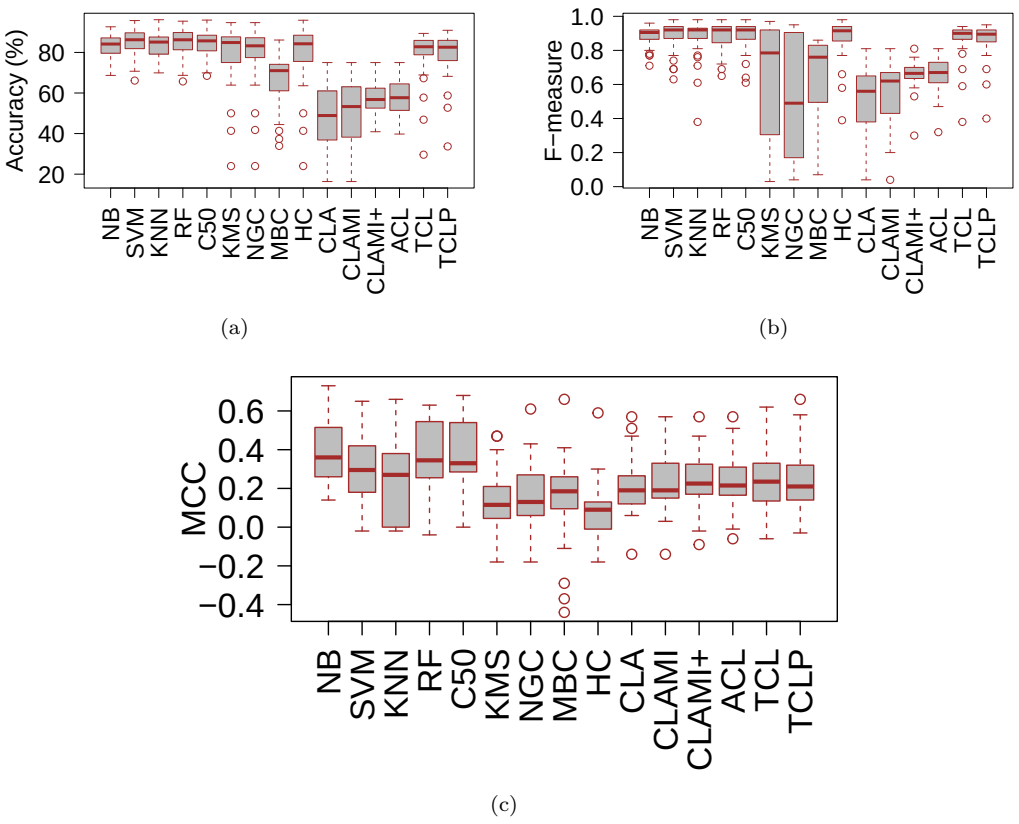
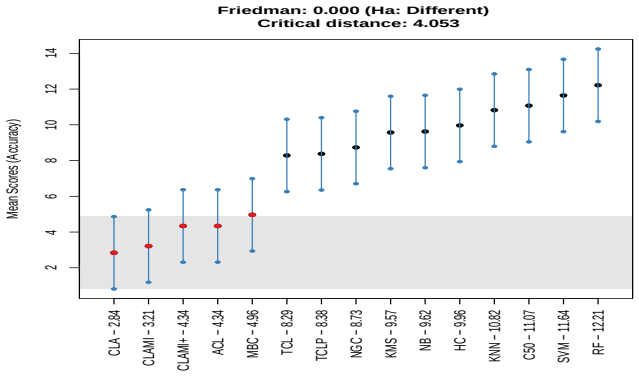


FIGURE 4.2: Boxplot performance of different approaches in term of (a) Accuracy, (b) FM, and (c) MCC

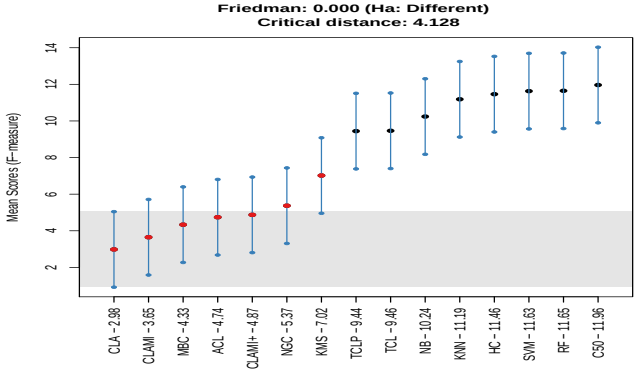
4.4.1 Results in Terms of Accuracy

The accuracy performance shown in Table 4.7. The accuracy of TCL/TCLP is superior in 4 groups (except ReLink) out of 5 group DSs compared to all threshold-based methods. TCL/TCLP performed better in one group (SOFTLAB) compared to all unsupervised algorithms. TCL/TCLP is better than NB, SVM, and KNN, at least in one group of DSs. The ensemble algorithm RF and C50 are superior to TCL/TCLP in all group DSs.

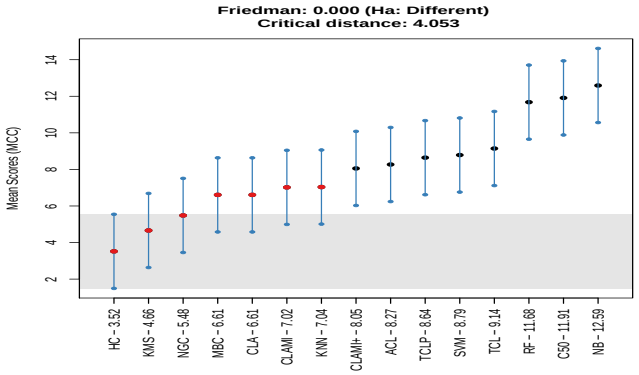
From the results shown in Table 4.10, we conclude that the performance of TCLP is not significantly different from NB, KMS, and HC, where it is performing less. E.g., TCLP is 0.23% less than KMS, but a p-value of 0.64 (>0.05) shows that the two techniques are not significantly different. Also, mean value of TCLP is greater than the mean value of NGC, MBC, CLA, CLAMI, CLAMI+, ACL, and



(a)



(b)



(c)

FIGURE 4.3: Performance analysis of proposed approach and other approaches using Post Hoc Nemenyi Test conducted on 28 DSs in terms of (a) Accuracy, (b) FM, and (c) MCC

TCL, as shown in Table 4.10 (greater values shown in bold). From Fig. 4.2a, we conclude that TCL/TCLP outperforms the MBC, CLA/CLAMI, CLAMI+, and ACL approach with a big difference and looks somewhat equivalent to the NB, KNN, KMS and NGC. From Fig. 4.3a, based on the mean score of CLA/CLAMI (2.84/3.21), CLAMI+ /ACL(4.34) are marked as low performing group (shaded background), and TCL/TCLP, along with NGC, KMS, NB, HC, KNN, C50, SVM, and RF are belong to high performing group. The mean score of TCLP (8.38) is less than mean score of other algorithms of the group, but these are coming in the same group and deriving that the performance is significantly the same. The MBC, with a mean score of 4.96, is on the borderline of the high-performing and low-performing groups, so it isn't easy to deduce its performance.

4.4.2 Results in Terms of FM

The FM performance is shown in Table 4.8. The FM of TCL/TCLP is better or equal to other approaches in 3 out of 5 groups of DSs. TCL/TCLP FM is superior to all threshold-based methods in all groups of DSs. Table 4.10 concludes that TCL/TCLP performs worse than NB, but it is not significantly different because the p-value is 0.28. However, TCLP performs better than KMS, NGC, MBC, CLA/-CLAMI, CLAMI+, and ACL (shown in bold) with significant differences. As shown in Fig. 4.2b, TCL/TCLP outperforms KMS, NGC, MBC, CLA/CLAMI, CLAMI+, and ACL. Based on the boxplot, TCL/TCLP performance looks equivalent to NB, SVM, KNN, RF, C50, and HC. From Fig. 4.3b, we conclude that based on the mean score, CLA/CLAMI (3.65) belongs to a low-performing group with MBC, ACL, and CLAMI+. While TCL/TCLP (9.44) belongs to a high-performing group with NB, KNN, HC, SVM, RF, and C50. From the critical diagram, it is clear that TCL/T-CLP is equivalent to the NB, KNN, HC, SVM, RF, and C50 significantly. CLAMI+ and NGC are on the borderline of the high-performing and low-performing groups, so it isn't easy to deduce their performance.

4.4.3 Results in Terms of MCC

The MCC performance shown in Table 4.9. The MCC values of TCL/TCLP are superior to all other approaches on only 1 out of 5 groups of DSs. TCL/TCLP outperforms all unsupervised algorithms in all groups of DSs. From Table 4.10, we conclude that TCL/TCLP performs worse than SVM but does not show a significant difference due to the p-value 0.52. However, TCL/TCLP performance surpasses KNN, KMS, NGC, MBC, HC, CLA/CLAMI, CLAMI+, and ACL, shown in bold font. As shown in Fig. 4.2c, the performance of TCL/TCLP is finer than KMS, NGC, MBC, HC, CLA/CLAMI, CLAMI+, and ACL but worse than NB, SVM, KNN, RF, and C50. In Fig. 4.3c, we see that KMS and HC belong to low-performing groups, while TCL/TCLP and NB belong to high-performing groups. Also, NGC, MBC, CLA/CLAMI, and KNN are on the borderline of the high-performing and low-performing groups, so it isn't easy to deduce their performance. However it can be clearly said that TCL/TCLP performs better than all unsupervised algorithms and threshold-based algorithms (except CLAMI+ and ACL).

4.4.4 Results After Applying Sampling Technique

To consider the DS imbalance problem, we have implemented three sampling techniques: random undersampling, random oversampling, and SMOTE [134]. We have found that TCLP model performs better with SMOTE technique as compared to other sampling techniques. Firstly, we balanced the DS with SMOTE technique (42.86% - 57.14% faulty classes). Then, we developed the TCLP model using balanced DSs. Thereafter, we tested the TCLP model on the original DS. The performance of TCLP model without sampling technique and with sampling technique is shown in Table 4.11. The last row of Table 4.11 presents the median value of the TCLP performance over 28 DSs. Based on the median value, it is concluded that after applying the sampling technique, median accuracy of TCLP increases from 82.56% to 83.09% (0.53% increment). However, there is no effect of sampling on the median values of FM (0.90) and MCC (0.21). So, we can conclude that there is

no significant effect of the sampling technique on the performance of TCLP on all the DSs. These results justify that the proposed approach TCLP is not significantly affected by the class imbalance problem.

4.4.5 Answer of Research Questions

Answer of RQ1: Is the performance of TCL/TCLP approach comparable to the performance obtained using the standard supervised learning approaches?

From the results shown in Table 4.10 and Fig. 4.2 and 4.3 and also from the discussion stated above, we can conclude that the performance of TCL/TCLP is comparable to standard supervised learning algorithms. The proposed threshold-based unsupervised learning approach TCL/TCLP performs significantly equivalent to the supervised algorithms in terms of the three performance measures i.e., accuracy, FM, and MCC.

Answer of RQ2: Is the performance of TCL/TCLP approach comparable to the performance obtained using the standard unsupervised learning approaches?

From Table 4.10, Fig. 4.2 and 4.3, and aforesaid discussion, we can conclude that the performance of TCL/TCLP is comparable to standard unsupervised learning algorithms. In general, it can be justified that TCL/TCLP performs significantly equivalent or superior in terms of accuracy and FM while it performs considerably finer in terms of the most robust performance measure, i.e., MCC.

Answer of RQ3: Is the performance of TCL/TCLP approach comparable to the existing threshold-based state-of-the-art technique?

From Table 4.10, Fig. 4.2 and 4.3, and aforesaid discussion, we can conclude that the performance of TCL/TCLP is comparable to CLA/CLAMI. In general, it can be justified that TCL/TCLP outperforms the threshold-based methods entirely in terms of accuracy, FM, and MCC.

Answer of RQ4: Is the performance of TCLP approach without sampling technique comparable to that of TCLP with sampling technique (SMOTE)?

From Table 4.11, we have concluded that the median accuracy of TCLP with SMOTE technique is increased by 0.53% as compared to TCLP without sampling technique. We also found that the values of median FM and median MCC of TCLP model do not change even after applying the sampling technique. These results justify that the proposed approach TCLP does not get significantly affected by the class imbalance problem.

Overall, the prediction performance of TCL and TCLP does not significantly differ, as shown in Fig. 4.2 and Fig. 4.3. The mean difference between the performance of TCL and TCLP is quite low, e.g., 0.22% (accuracy), 0.001 (FM), and 0.005 (MCC). The difference in the mean score of TCL and TCLP is also marginal, e.g., 8.29 vs. 8.38 (accuracy,) 9.46 vs. 9.44 (FM), and 9.14 vs. 8.64 (MCC). The mean score of TCL is slightly better than TCLP but significantly equivalent. Hence, TCL does not require any machine learning algorithms. TCL is a more straightforward approach as compared to TCLP. In the case of DS group MORPH and ReLink, TCLP finer by 0.01 than TCL in term of MCC, as shown in Table 4.9. Therefore, it is interesting to explore the contexts/situations in which it would be better to use TCL with unlabeled datasets and TCLP with labeled datasets.

Overall, in general, TCL and TCLP perform significantly equivalent or sometimes even better than supervised (RQ1), unsupervised (RQ2), and threshold-based methods (RQ3) in terms of accuracy, FM, and MCC. It is to be noted that TCL/TCLP does not require a labeled DS and manual interference. Still, it obtains comparably significant prediction performance compared to the most state-of-the-art approaches in terms of accuracy, FM, and MCC. TCL/TCLP method is an interpretable and explainable one (as described in section 4.5).

TABLE 4.11: Performance of TCLP without sampling and with sampling (SMOTE) technique

		TCLP without sampling			TCLP with SMOTE		
Group	Project ID	Accuracy %	FM	MCC	Accuracy %	FM	MCC
AEEEM	SP1	68.21	0.77	0.31	67.90	0.78	0.32
	SP2	81.64	0.88	0.45	82.85	0.89	0.45
	SP3	87.99	0.94	0.02	86.11	0.92	0.15
	SP4	87.16	0.93	0.19	87.27	0.93	0.19
	SP5	84.64	0.92	0.15	84.70	0.91	0.19
SOFTLAB	SP6	84.30	0.91	0.24	87.60	0.93	0.29
	SP7	88.89	0.93	0.58	87.30	0.93	0.36
	SP8	87.85	0.93	0.56	86.92	0.92	0.50
	SP9	86.11	0.91	0.66	83.33	0.90	0.45
	SP10	75.25	0.85	0.22	81.19	0.89	0.28
NASA	SP11	80.73	0.89	0.24	81.04	0.89	0.19
	SP12	83.00	0.90	0.31	84.98	0.91	0.33
	SP13	80.00	0.88	0.15	83.97	0.91	0.20
	SP14	85.79	0.92	-0.03	78.18	0.87	0.14
	SP15	76.69	0.86	0.13	74.20	0.84	0.14
MORPH	SP16	82.40	0.89	0.49	82.40	0.90	0.32
	SP17	85.47	0.92	0.33	88.03	0.93	0.33
	SP18	89.38	0.94	0.05	89.68	0.95	0.11
	SP19	52.74	0.60	0.20	43.88	0.58	0.10
	SP20	79.55	0.88	0.04	77.84	0.87	0.01
	SP21	71.11	0.82	0.06	73.33	0.84	0.09
	SP22	90.91	0.95	0.16	85.66	0.92	0.29
	SP23	33.67	0.40	0.06	28.57	0.39	-0.03
	SP24	82.85	0.90	0.23	85.06	0.92	0.23
	SP25	82.73	0.90	0.19	83.86	0.91	0.24
ReLink	SP26	58.76	0.69	0.25	56.19	0.68	0.20
	SP27	78.57	0.85	0.58	78.57	0.85	0.58
	SP28	68.42	0.79	0.17	70.68	0.82	0.10
MEDIAN		82.56	0.90	0.21	**83.09**	0.90	0.21

4.5 Explainability and Interpretability of TCL/T-CLP

The proposed TCL/TCLP method consists of two major steps. First is TCL, which automatically performs labeling of the DS based on metrics threshold derived. The second is TCLP, which consists of three additional steps: metrics selection, instances selection and training (machine learning model), and prediction on the original DS. The metrics selection and instances selection is a heuristic approach. So, analyzing the different combinations of metrics selection and instances selection are discussed in the following subsections.

4.5.1 Explanation on Number of Metrics and Instances Selected for Training

TCL/TCLP comprises the derivation of the metrics threshold and labeling the instances by comparing the metrics value with the threshold value. TCLP algorithm heuristically minimizes the number of metric selections and instance selections based on MVR and IVR of each metric and instance individually. After much experimentation, we found that TCLP is performing superior on selecting minimum $\lceil log_2 m \rceil$ metrics with minimum MVR and minimum 70% instances with minimum IVR with the condition that at least two faulty instances should be selected in the refined DS. After selecting minimum $\lceil log_2 m \rceil$ metrics and minimum 70% instances, if we get zero or one faulty instance, then we select 80% of instances with minimum IVR. This process is done until 99% of instances are selected. The metrics and number of instances selected from each project after filtration are shown in Table 4.12. This table information is used as input for learning and prediction steps (Section 4.2.4). The most common metrics across the group are shown as bold in Table 4.12

A group-wise list of original metrics in different DSs is shown in Table 2.2. In Table 4.12, it can be seen that the few selected metrics from the DS across the

group are common. As shown in Table 4.12, metrics A9, A17, A16 are the common metrics in all DSs of AEEEM group; S12, S9 are common in four DSs, S11, S8, S23 are common in two DSs of SOFTLAB group; N13, N37 are common in 4 DSs, N29 is common in two DSs of NASA group; M11 is common in eight DSs, M1 is common in seven DSs of MORPH group; and R18, R13, R9, R26 are common in 2 DSs of ReLink group. With this, we have found the essential metrics that are more relevant and useful to predict fault proneness. It also helps to reduce the complexity of the model. It can be stated that these common metrics across the group are more effective metrics for the threshold-based SFP approach.

4.5.2 Number of Data Points Correctly Labeled by Proposed Algorithm (TCLP) Differ from the Rest of the Methods within Each Group

The number of data points correctly labeled by the proposed algorithm TCLP differ from the rest of the 13 methods within each group is shown in Table 4.13. In Table 4.13, the difference is calculated by subtracting the number of data points correctly labeled by TCLP from the other 13 methods. The higher negative value shows, the greater number of data points correctly labeled by TCLP than other methods. It can be seen that TCLP is correctly labeling more number of data points compared to all three threshold-based methods. The maximum difference is 2063 data points, which means TCLP correctly labeled 2063 more data points than the CLAMI method in the AEEEM group. We can also analyze that TCLP correctly labels more data points than unsupervised methods KMS, NGC, MBC in at least 2 groups. TCLP ultimately beat the MBC algorithm in all groups. TCLP correctly labels 54 more data points than NB in the AEEEM group. Compared to TCL and TCLP, TCLP labels more data points in the AEEEM and MORPH groups. Overall, this table shows the effectiveness of the TCLP compared to other methods across the group in terms of difference in number of data points correctly classified.

TABLE 4.12: Selected metrics and number of instances from each project after filtering

Groups	Projects	Selected metrics	#instances selected
AEEEM	Equinox	**A9**, **A16**, **A17**, A10, A13	227
	JDT	**A9**, **A17**, **A16**, A6, A7	698
	Lucene	**A9**, A1, A6, **A16**, **A17**	622
	Mylyn	**A9**, **A17**, **A16**, A10, A5	1676
	PDE	A10, **A9**, **A16**, A5, **A17**	1048
SOFTLAB	AR1	S12, S7, S23, S18, S19	85
	AR3	**S9**, S12, S15, S17, S1	44
	AR4	**S9**, S11, S12, S8, S7	75
	AR5	**S9**, S11, S12, S8, S23	25
	AR6	S10, S6, **S9**, S19, S7	71
NASA	CM1	N37, N13, N16, N20, N24, N29	229
	MW1	**N29**, N13, N16, N1, N7, N37	177
	PC1	N2, N37, **N29**, N4, N10, N5	494
	PC3	N13, **N29**, N35, N37, N13, N18	754
	PC4	N25, N22, N35, N34, N13, N11	901
MORPH	Ant_1.3	**M1**, M11, M4, M9 ,M5	88
	Arc	M4, **M1**, M8, M13, M11	164
	Camel_1.0	M11, M5, M10, M12, M14	336
	Poi_1.5	M5, M11, **M1**, M4, M8	166
	Redaktor	M11, **M1**, M5, M10, M12	123
	Skarbonka	**M1**, M8, M11, M13, M2	32
	Tomcat	**M1**, M5, M9, M16, M17	773
	Velocity_1.4	**M1**, M5, M9, M8, M11	137
	Xalan_2.4	M5, M11, M7, M8, M16	716
	Xerces_1.2	M13, M4, M10, M12, M14	436
ReLink	Apache	R18, R13, R9, R26, R10	136
	Safe	R13, R15, R11, R16, R18	39
	Zxing	R23, R24, R25, R26, R9	280

TABLE 4.13: Difference in correctly labeled data points between the proposed algorithm (TCLP) and rest of the 13 methods within each group (Other method-TCLP)

Groups	Total Instances	Supervised Methods					Unsupervised Methods				Threshold Based Methods			
		NB	SVM	KNN	RF	C50	KMS	NGC	MBC	HC	CLAMI	CLAMI+	ACL	TCL
AEEEM	5371	-54	69	44	107	80	-17	-17	-766	-2	-2063	-1611	-1172	-87
SOFTLAB	428	8	9	12	18	11	7	4	-42	11	-147	-102	-108	7
NASA	3649	128	304	233	303	279	240	219	-394	257	-1070	-912	-1050	16
MORPH	3373	120	215	175	216	198	-2	-17	-460	10	-1413	-945	-871	-36
ReLink	649	39	30	33	42	34	-17	-13	-45	-17	-7	-19	-17	2

4.5.3 Automated Labeling Versus Expert Labeling

The proposed automated labeling is based on the statistical parameters of software metrics (i.e. mean and standard deviation) that are completely uniform for all software metrics and software projects. Whereas expert labeling completely depends upon the experience/knowledge of software expert team, that cannot be uniform for all software metrics and software projects. Expert labeling is subjective, while automated labeling is objective and machine learning dependent. Automated labeling does not require experienced software experts, so it is cheaper and faster because labeling is performed by a defined formula/equation. However, our proposed approach reaches up to 83.09% median accuracy compared to label available in the original DS, which is fairly better than other unsupervised approaches.

An expert-based SFP approach is proposed by Catal et al. [115]. This approach has a two-stage process, firstly performed clustering using a machine learning algorithm and the statistical information (min, max, mean, median, cluster size, 75 percentiles, and 90 percentile) of the cluster given to an expert to inspect and decide the label of cluster(s). They reported K-means and Neural gas clustering algorithm with an expert on a NASA DS provided 81.73% and 83.46% accuracy, respectively. TCL/TCLP also provided an average accuracy of 81.42% on NASA DS that is near to expert based SFP approach [115]. Based on the above discussion, TCL/TCLP method based on thresholding, clustering, MVR, IVR achieves labels that are at par with human annotators.

4.5.4 Significance of Variation in Number of Metrics Selected Across Each DS Group Using MVR

We have implemented the TCLP methods with the variation of different metrics and fixed the number of selected instances (100%). The results are shown in Fig. 4.4a,b,c, in terms of accuracy, FM, and MCC. This Fig. 4.4 are drawn based on the median value of the performance over all DSs across group. E.g., in Fig. 4.4a, the maximum value is 86% on minimum selected metric 5, which means the median accuracy of TCLP with selected metrics five on all DS of group SOFTLAB is 86%. From Fig. 4.4a, we conclude that TCLP model is performing better in terms of accuracy at minimum selected metrics 3(AEEEM), 2(SOFTLAB), 5(NASA), 4(MORPH), and 4(ReLink). From Fig. 4.4b, we conclude that TCLP model is performing better in terms of FM at minimum selected metrics 2-3(AEEEM), 2(SOFTLAB), 5(NASA), 4(MORPH), and 2-3(ReLink). From Fig. 4.4c, we conclude that TCLP model is performing better in terms of MCC at minimum selected metrics 3(AEEEM), 5(SOFTLAB), 2(NASA), 2-3(MORPH), and 2-3(ReLink).

Somewhere, on selecting more metrics, we got an increment in the performance, but it is a marginal difference. E.g., in Fig. 4.4c, the performance of ReLink is slightly greater on selecting minimum metrics 20 than 2 or 3 metrics. But, it increases the complexity of the model. Overall, it signifies that when the number of selected instances is fixed, metric selection with minimum MVR is effective. In which we can get better results by selecting a minimum 2 to maximum 5 metrics from the five groups of DSs.

4.5.5 Significance of Variation in Percentage of Instances Selected Across Each DS Group Using IVR

We have implemented the TCLP methods by selecting varying percentages of instances across all DSs and fixed the chosen number of metrics to $\lceil log_2 m \rceil$. The results are shown in Fig. 4.5a,b,c, in terms of accuracy, FM, and MCC. These Fig.

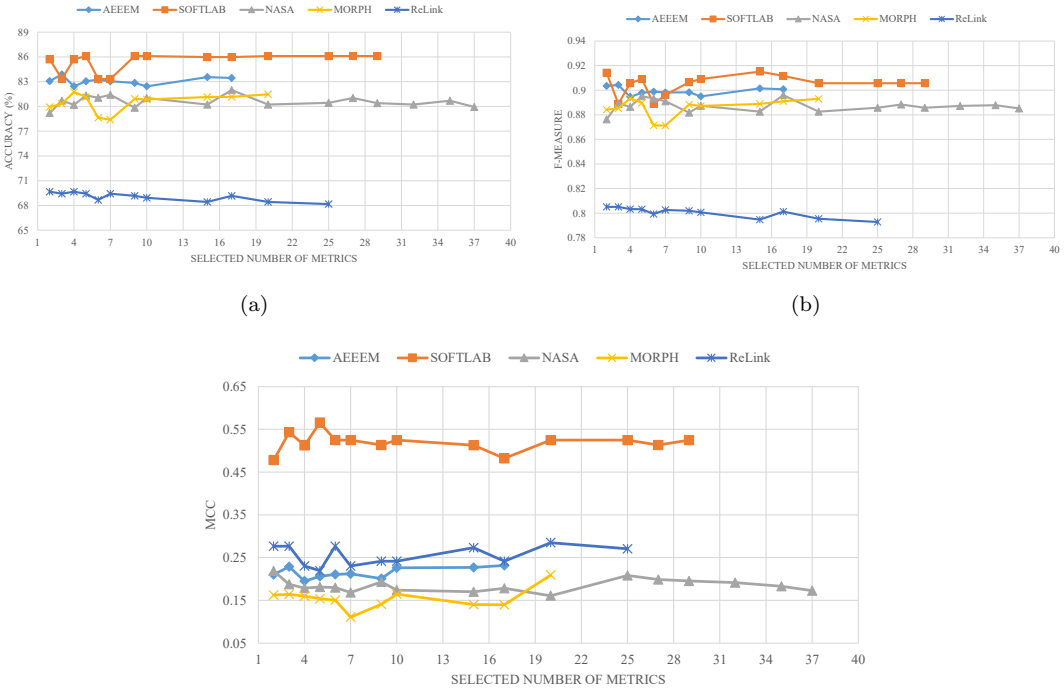


FIGURE 4.4: Performance of TCLP approach with variation in the number of metrics selected in terms of (a) Accuracy, (b) FM, and (c) MCC

4.5 are drawn based on median values of the performance of all DSs across the group. E.g., in Fig. 4.5a, the maximum value is 70% (ReLink) on minimum selected metric 65% of instances, which means the median accuracy of TCLP with selected instances 65% from all DSs of group ReLink is 70%. We have chosen a minimum of 65% and a maximum of 100% of instances from each group with fixed number of metrics. When we select a minimum percentage of instances below 65%. It produces the problem of getting either one or zero faulty data point after instance selection (extremely imbalanced DS).

From Fig. 4.5a, we conclude that TCLP model is performing maximum in terms of accuracy at minimum selected % of instances 65% across group (SOFTLAB), (NASA), (ReLink). It is also seen that AEEEM and MORPH group's no data producing performance on selecting a percentage of instances below 85%. The median

performance of TCLP across AEEEM and MORPH is maximum at selecting a minimum of 95% of instances from the DS of the group. From Fig. 4.5b, we conclude that TCLP model performance in terms of FM is almost the same over selecting varying percentages of instances (only SOFTLAB, NASA, ReLink). Whereas TCLP is producing results for AEEEM and MORPH group DSs on selecting a minimum of 85% of instances. From Fig. 4.5c, we conclude that TCLP model performance in terms of MCC at minimum selected percentage of instances are maximum and vary after increasing percentage of instance selection. The significant changes are seen in ReLink, NASA, and AEEEM group of DSs.

Somewhere, on selecting more percentage of instances, we got an increment in the performance, but it is a marginal difference. E.g., in Fig. 4.5c, the performance of SOFTLAB is slightly decreasing and increasing on selecting more percentage of instances. Overall, it signifies that instance selection (minimum 65% of instances) with minimum IVR is effective in terms of MCC only, not much fruitful in terms of accuracy and FM. For a fixed number of selected metrics, instance selection can help to get better results on selecting a minimum of 85% of instances across all groups.

4.5.6 Execution Time of Different Models

The execution time in seconds of each model is shown in Table 4.14. Average time complexity shown group wise and total average (T_AVG) shown in last row of the Table. We observe that TCLP is faster compared to SVM, RF, C50, MBC, and CLAMI.

We have conducted Cohen's D test to calculate the effect size between TCLP and other SBP models and results shown in Table B.7.

4.6 Threats to Validity

The proposed approach has considered some existing risks to validate the complete SFP approach. This approach may suffer from the following risks:

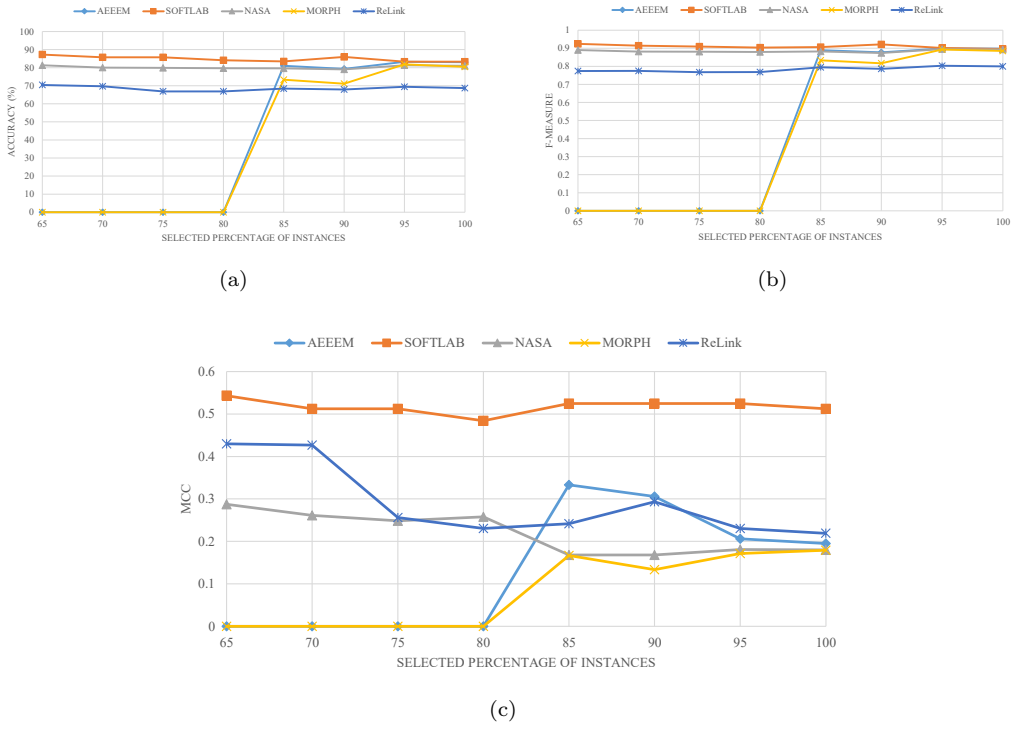


FIGURE 4.5: Performance of TCLP approach with variation in percentage of instances selected in terms of (a) Accuracy, (b) FM, and (c) MCC

Internal validity The proposed approach was implemented carefully on 28 DSs. We have chosen publicly available software projects to identify the faults. The public DSs were carefully collected from the software project groups such as AEEEM, SOFTLAB, NASA, MORPH, and ReLink. So, the first threat is the consistency of the software project SFP DS. However, it was collected consistently to the best of our knowledge, but we cannot claim that the DS is 100% accurate.

Construct validity In this work, the developed approach TCL/TCLP predicts only whether the specific module is faulty or non-faulty but does not predict the number of faults within a particular module. So, it is a potential threat to validate the proposed approach. Also, the other method CLA/CLAMI, CLAMI+, ACL, with which the performance of the proposed approach has been compared, was implemented based on the information provided in the paper only. The supervised and unsupervised algorithms was implemented using the standard library with the

TABLE 4.14: Execution time of different SBP models in seconds

Groups	Projects	Execution time in seconds													
		Supervised Methods					Unsupervised Methods				Threshold Based Methods				
		NB	SVM	KNN	RF	C50	KMS	NGC	MBC	HC	CLAMI	CLAMI+	ACL	TCL	TCLP
AEEEM	Equinox	0.59	2.41	0.65	9.41	5.86	0.02	0.02	0.13	0.01	1.62	0.61	0.10	0.11	1.94
	JDT	1.37	17.25	2.09	38.77	24.03	0.09	0.19	23.56	0.07	12.00	1.95	0.31	0.33	5.79
	Lucene	1.34	7.84	0.80	18.33	6.76	0.06	0.09	24.14	0.04	13.92	1.75	0.24	0.19	6.63
	Mylyn	0.86	38.37	1.60	74.54	8.81	0.31	0.23	24.02	0.20	36.75	2.20	1.06	0.58	27.40
	PDE	0.50	11.67	0.84	34.06	7.89	0.08	0.06	7.80	0.08	11.02	0.89	0.27	0.31	9.22
	AVG	0.93	15.51	1.20	35.02	10.67	0.11	0.12	15.93	0.08	15.06	1.48	0.40	0.30	10.20
SOFTLAB	AR1	0.55	1.01	0.45	2.64	2.73	0.01	0.02	0.02	0.01	1.00	0.39	0.05	0.08	1.13
	AR3	0.52	0.95	0.45	1.39	2.27	0.01	0.01	0.01	0.01	0.70	0.38	0.03	0.05	0.78
	AR4	0.53	0.98	0.45	1.84	2.50	0.01	0.01	0.01	0.01	0.05	0.35	0.04	0.06	0.91
	AR5	0.54	0.86	0.46	0.98	2.28	0.01	0.01	0.03	0.01	0.56	0.34	0.11	0.07	0.70
	AR6	0.95	2.92	0.82	4.08	5.01	0.01	0.01	0.06	0.01	1.18	0.64	0.06	0.09	1.28
	AVG	0.62	1.34	0.53	2.19	2.96	0.01	0.01	0.03	0.01	0.70	0.42	0.06	0.07	0.96
NASA	CM1	0.86	3.36	0.72	14.10	11.01	0.03	0.05	0.20	0.01	4.37	0.82	0.20	0.22	4.61
	MW1	0.83	2.16	0.67	9.82	6.80	0.02	0.02	0.12	0.01	3.21	0.74	0.14	0.19	2.86
	PC1	0.86	6.61	1.25	26.88	18.26	0.06	0.04	0.50	0.03	8.53	1.05	0.37	0.45	9.08
	PC3	2.31	22.33	1.15	87.11	71.25	0.22	0.20	2.65	0.17	25.31	3.20	0.60	0.69	14.56
	PC4	1.19	14.96	1.92	73.89	54.82	0.15	0.13	1.81	0.09	16.92	1.70	0.68	0.83	15.89
	AVG	1.21	9.88	1.14	42.36	32.43	0.10	0.09	1.06	0.06	11.67	1.50	0.40	0.48	9.40
MORPH	Ant_1.3	0.63	1.39	0.66	3.44	4.11	0.01	0.02	48.08	0.01	1.59	0.56	0.05	0.07	1.64
	Arc	0.85	1.85	0.69	6.42	4.22	0.02	0.02	0.08	0.02	1.70	0.63	0.08	0.12	2.13
	Camel_1.0	0.87	2.56	0.98	8.37	5.71	0.03	0.03	38.65	0.01	4.53	0.93	0.11	0.19	3.86
	Poi_1.5	1.03	2.82	0.98	8.35	5.28	0.01	0.02	0.10	0.02	2.44	0.82	0.11	0.11	2.39
	Redaktor	0.88	2.19	0.89	6.51	4.84	0.01	0.03	0.06	0.01	1.75	0.79	0.08	0.10	2.17
	Skarbonka	0.59	1.46	0.70	1.56	3.21	0.01	0.02	0.01	0.02	0.91	0.58	0.03	0.05	1.00
	Tomcat	0.78	7.69	0.90	23.27	13.50	0.05	0.06	0.51	0.05	7.56	1.11	0.50	0.33	9.18
	Velocity_1.4	0.64	1.62	0.66	5.34	4.86	0.02	0.04	86.03	0.01	2.04	0.67	0.07	0.09	2.19
	Xalan_2.4	0.82	5.48	0.86	21.64	12.65	0.03	0.07	43.13	0.03	7.46	0.93	0.22	0.27	7.09
	Xerces_1.2	0.64	2.90	0.75	12.36	6.02	0.01	0.03	36.86	0.02	3.94	0.72	0.15	0.18	4.24
	AVG	0.77	3.00	0.81	9.73	6.44	0.02	0.03	25.35	0.02	3.39	0.77	0.14	0.15	3.59
ReLink	Apache	0.75	1.91	0.68	7.02	3.85	0.01	0.02	0.06	0.01	1.70	0.67	0.07	0.11	1.93
	Safe	0.65	1.15	0.52	2.50	1.65	0.01	0.02	0.10	0.01	1.10	0.43	0.03	0.06	1.20
	Zxing	0.72	3.23	0.75	16.00	5.22	0.01	0.02	0.17	0.02	3.22	0.75	0.16	0.20	3.22
	AVG	0.71	2.10	0.65	8.51	3.57	0.01	0.02	0.11	0.01	2.01	0.62	0.09	0.12	2.12
	T_AVG	0.85	6.37	0.86	19.56	11.21	0.05	0.05	8.49	0.04	6.57	0.96	0.22	0.22	5.25

default parameters.

External validity In this research work, the SFP has been applied to three types of granularity (class, file, and function); however, the developed approach is likely to be valid for other remaining programming paradigms as well. Number of software metrics depends on software projects that are considered to design the TCL/TCLP approach. Some of the other software metrics may also be useful in developing the SFP approach.

4.7 Conclusion and Future Work

Developing a robust and general SFP approach for new software projects or software projects with limited historical information is challenging. To mitigate the limitations of various existing approaches, we have proposed TCL/TCLP approach that can be used to develop automated and high-performing SFP models on unlabelled DSs. This empirical study validated the proposed approach on 28 software projects (having a different number of software metrics of various types) collected from five software groups. TCL/TCLP approach performs significantly better or equivalent to the existing state-of-the-art techniques such as supervised learning, unsupervised learning, and state-of-the-art threshold-based approach in terms of accuracy, FM, and MCC. The produced tabular, boxplot, and statistical results conclude that practically TCL/TCLP has a strong potential for SFP on unlabelled DSs without any human effort. Our experimental results justify that TCL/TCLP performs better than CLA/CLAMI with 28.63% higher accuracy, 0.31 higher FM, and 0.032 higher MCC. Results from Table 4.11 justify that the proposed approach TCLP does not get significantly affected with the class imbalance problem. To validate the proposed approach in the industry, we have planned to implement this approach on some private DSs in the future.