International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 01, 2020
ISSN: 1475-7192

GKS Algorithmic Technique for Early Defect Prediction (GKS: A Genetic Feature K-means Clustering with Support Vector Machines)

P. Patchaiammal and R. Thirumalaiselvi

Abstract--- *Post production defects are one of the reasons behind the rework and also the failure of software. To reduce the defects, we have to find features in earlier post production. Many experts will be developing intelligent decision support systems related to software to get better ability in detection of defect. The defect identification and discovery using machine learning techniques provide the reasonable result with accuracy. In order to stimulate the accuracy level, we combine supervised and unsupervised learning techniques. This helps to design an intelligent decision support system for early defect prediction. In our paper, clustering and classification algorithms are combined with genetic feature set to form GKS (A Genetic feature K-means clustering with Support Vector Machines) technique. This proposed algorithmic technique works in three parts, first a predictive analysis is carried out on Bugzilla eclipse Dataset and features are collected by using genetic algorithm and followed by the second part in which clustering set is formed by unsupervised clustering technique known as K-Means clustering and finally the performance parameters like precision, recall, f1 – score and accuracy level are found using supervised classification technique known as SVM (Support Vector Machine).The results are mapped into the roc – auc curve. At last, the clustered labels are mapped with the defect taxonomy list to categorize the defect features in appropriate defect occurring phase. In this paper, the feature classification is improved by using K-Means centroid algorithm with the help of SVM technique. This paper also provides a model to implement the GKS algorithmic technique. The focus of our model is used to classify the feature according to post production defect list so as to have better defect taxonomy.*

Keywords--- *Machine Learning (ML), GKS (A Genetic Feature K-means Clustering with Support Vector Machines), Feature Engineering, MAE (Mean Absolute Error), AUC (Area Under the Curve), ROC (Receiver Operating Characteristic curve).*

I. INTRODUCTION

1.1 Machine Learning

Machines can learn the things how we can teach it. Machine learning is the computer learning study and predicts the things, which improve the learning automatically through experience. In general, traditional programming we have typical data input for program to produce the output, but in ML the output is the training set of learning the system. Along with the training set and testing data includes the input not the corresponding expected output.

Historical data are used as the prediction Program results. We have same training data, testing historical data, and test data in machine learning. Datasets consist of a number of examples in which each of it is known as

P. Patchaiammal, Research Scholar, Bharath University, Chennai. E-mail: sarandsk1@gmail.com
R. Thirumalaiselvi, Assistant Professor, Dept. of Computer Science, Govt. Arts College (Men), Chennai.

DOI: 10.37200/IJPR/V24I1/PR200282
Received: 22 Nov 2019 | Revised: 26 Dec 2019 | Accepted: 10 Jan 2020

attributes and labels. The first phase is known as “training,” which is a model generation phase in which the model attempts to generalize how attributes are related to the label. The second phase is known as “testing,” which is the model applying phase in which the model is applied to the previously-unseen data set with unknown labels to produce a classification. This second phase uses ranking. At last it is used as the validation for the prediction.

Generally, we have three types of execution in software development. First is Pre-execution, which involves code review with test case study. Second one is Execution automated running and defect analysis. Final one is Post-execution, which includes Debugging, Regression, and Updating. Defect analysis and prediction is a critical defect in software development process. Machine learning techniques are one of the best techniques used in SDLC phase for finding and minimizing the post production defect by prediction. Automated testing and defect analysis are good at execution level. Even though lots of automation available in Pre-execution and Post-execution defect analysis, which has some limitations to it, so there is a need of new prediction model to give best result. To achieve this, there is need of the combination of supervised learning algorithm and unsupervised learning is necessary. Software repositories are very big knowledge set. It has a lot of details about software defects. Software defect classification is used for classifying the software defect into different categories. The proposed defect classification is provided for the evaluation of the intended technique.

1.2 Defect in software

The realized software should be reliable. The reliability is achieved by performance of the intended function in a certain period of time for a certain condition. To achieve this software used will be create fewer defects. The defect in software may cause failure. Proneness of defect is nothing but a likelihood of the defect in the software. Defect prediction in the software is used to minimize rework and cost in the post production. The software metrics and the process maturity are mainly used for the defect prediction criteria. The defect of eclipse in Bugzilla data in the form of csv file is considered as the input. The defects at the end of the development that is requirement, design and coding are the output evaluated in the prediction model.

Defect prediction taxonomy set is formed by grouping is done by clustering. The clustering technique used is K-Means technique. After clustering SVM technique is applied for classification. This is the one of the best classifiers for non-linear data set. The Roc curve is used to show the accuracy level of the hybrid K-Means clustering with SVM classification.

II. METHODOLOGY

The main objective of the proposed work is to create the classification of the software defect taxonomy in the basis of the clustering. This work aims to provide the effective defect taxonomy classified group for the dataset in faster manner. Before clustering, the features chosen are checked by using the genetic fitness function. So, the software analyst, designer and also tester use this defect taxonomy group for organizing the task to reduce rework and cost. The overall methodology of software defect prediction set formation by clustering and classification is shown in diagram 1.

DOI: 10.37200/IJPR/V24I1/PR200282
Received: 22 Nov 2019 | Revised: 26 Dec 2019 | Accepted: 10 Jan 2020

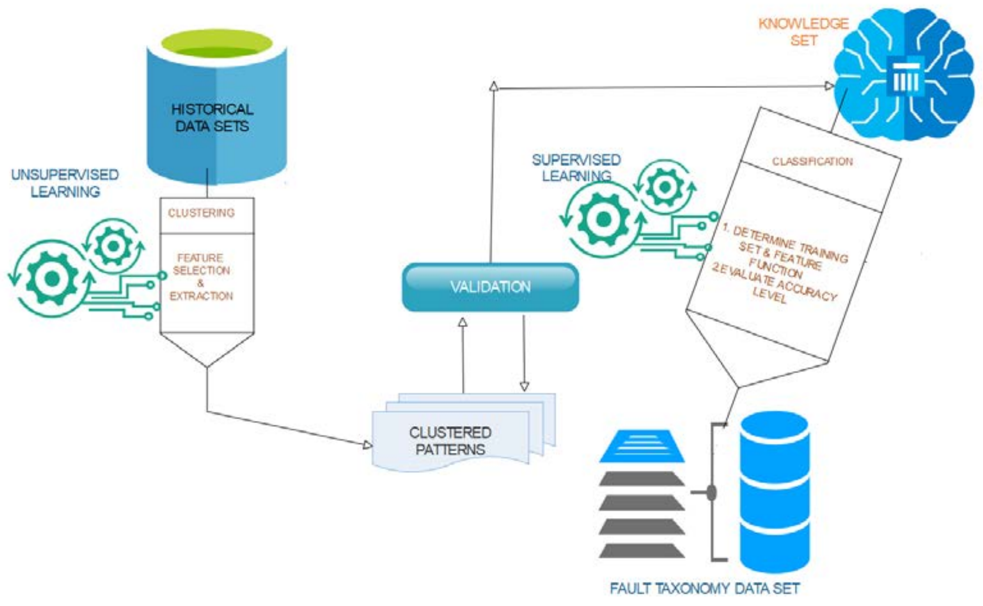


Diagram 1: GKM (A Genetic feature K-means clustering with SVM Algorithmic techniques) formation Model

According to the above diagram, defect prediction is parted into the following steps: The first step includes analyzing the software defects from the histories of software repositories and some local organizational defect database. Defect analyzing is taken to form the defect feature set. This is done by feature selection and extraction [3]. In the next step, the clustering is performed to form the defect prediction set using K-Means algorithm. Clustering is an unsupervised learning technique, which is used to discover the feature patterns in an effective manner. The proposed defect data is used to from the feature set and is fed into the clustering model in which the defects are created by using the similarity of the feature attributes in each phase. By using the defect cluster model, the regular labels are generated by using the occurrence of the feature term which is habitually available in the individual cluster. The next step is to plot the terms present in the cluster feature labels into the defect set for the classification of the defect in the development phases. The final step is to generate the confusion matrix for the classified defect using SVM techniques for the calculations of precision, recall, f - score and accuracy average.

III. DEFECT PREDICTION TAXONOMY FORMATION

The defect prediction taxonomy needs right machine learning technique. This paper will help to combine both unsupervised and supervised learning technique to form the best taxonomy. To achieve the best taxonomy clustering in unsupervised learning and classification in supervised learning are combined in this paper. This section first describes about the clustering in grouping. Next it describes about classification in performance evaluation.

3.1 Grouping by clustering

Clustering is the technique used to identify the input and output variables according to the subjective knowledge set. From the domain of historical data, the knowledge base set is formed. This is used to cluster the inputs to outputs. This produces the output as the number of defects at each phase.

Clustering is the unsupervised method, which is used to identify the similar group of features in a dataset. Clustering is used to isolate data task from the group of data. In our previous paper, we formed the feature set with the help of hypothesis testing and in our paper, we used fitness function to confirm those features. Clustering is used in defect data set to identify what type of development phase defect fit into the group. In data sciences, there are four types of clustering used. These are used for grouping the data set as points in a way that is characterized in accordance with other. Mainly creation of the clustering is used to find out how our data is related to one another. The basic types of the clustering are categorized as centroid, density, distribution, and connectivity clustering. In our paper, we used centroid clustering.

Centroid clustering is applied for classification-based problems. According to the algorithm, centers are selected to group the data points into specified clusters. In our defect dataset, the clusters are chosen and a line is then separated in the data set points. The centroid points are repositioned in accordance to all the points in each cluster. The iteration process started. The centroids and the points in clusters are adjusted to get the optimized clusters. In the defect data set, the cluster analysis is used and subdivision done for the two clusters one for defective = ‘Y’ and another for defective=’N’. In our paper, we use K-means clustering concept for identify the data to form group. We used this because this algorithm uses only the scenarios to understand the population to take necessary features. The ‘k’ was able to find the clusters corresponding to the “true” class correctly. If ‘k’ is given then k-means clustering performed well.

3.1.1 K-Means clustering

This is an unsupervised learning technique used for clustering. This algorithm is the popular algorithm, which follows the centroid clustering model category. In this algorithm ‘K’ is the number of clusters used to identify the knowledge of the data set. It is used to understand the population and find the sample from it to form the feature set. At the beginning stage the clusters are unknown. In the training phase of this algorithm, K observations are arbitrarily selected to find centroids. The data points in the dataset are assigned to the nearest Euclidean distance. Once the clusters are formed the centroids updated to form the new centroids. Iterate this process until the centroids become the mean of the clusters. The predictions are done according with the nearest centroids.

Always feature engineering plays a vital role in the cluster learning. The meaningful feature chosen for clustering is the essential for the grouping and categorizing of data. This helps in working of the algorithm. This also helps in the data regulation. The necessary feature chosen needs dimensionality reduction. We apply this reduction in the basis of fitness function found and we chose only four features for our prediction model with the k value as 3(k=3). We form three group of clusters for the features DECISION_DENSITY, DESIGN_COMPLEXITY and DESIGN_DENSITY. The formulaic function used for the fitness dimensionality reduction by using python sklearn is

$$kmeans = kMeans(n_{clusters} \ = kvalue).fit(variable) \rightarrow (1)$$

In equation (1) K-Means() is a function used to set the target cluster of the K-Means Clustering algorithm. Generally, the k-value is used to determine the accurate estimate. K is the distance of mean between the data features and their centroid cluster. The increase in the ‘K’ will decrease the result chosen.

DOI: 10.37200/IJPR/V24I1/PR200282
Received: 22 Nov 2019 | Revised: 26 Dec 2019 | Accepted: 10 Jan 2020

3.2 Performance evaluation by Classification

This is next step of clustering. Generally, classification is about predicting a feature label. The classification problem mostly takes discrete label of features. The target variable is definite and that is known as classification learning problem. The supervised learning techniques are used to form the problem of developing a model using the historical data for making the prediction. Predictive modelling can be defined approximating the mapping function with input to output feature variables. In this type of problems, the output features are either labels or categories. For the observations considered the mapping function is used to predict the relevant class.

For classification type of prediction, the probability to be predicted can be inferred as the chance or confidence of a given example belongs to each of the class. In general, the highest value in the set is considered as the probability to a class of the feature label to be selected. Even though there are many ways for selecting the prediction the most common method used is accuracy calculation. Choose the best from the classified set will helps to get the proper prediction.SVM is a support vector machine is a supervised machine learning algorithm. This algorithm is best for classification or regression challenges. It is best for classification problems in real world regularization of known parameter features.

3.2.1 SVM Classification

SVM (Support Vector Machine) is one of the best choices for the classification type problems. For the ‘n’ number of features in Support Vector Machine we have n-dimensional space with the observed feature values represented as the coordinates. The working of SVM is based on the finding of hyperplane, which is used for differentiate the classes considered. In this paper, we have two classes known as defective and non-defective. There are three states of hyperplane. First one is by selecting the plane which isolate two classes. This type of selection will help to gain proper performance classification. The second one is to choose the plane, which is nearer to the data class that is marginal value selection. This type of selection will help to avoid the wrong classification. The third stage works for more feature set. In this type, SVM kernel is added to take the input spaces. This type of classification is used in the mostly useful non-linear problem in order to handle complex data set. We implemented SVM in python with the help of pandas and sklearn library.The formulaic function used for the fitness dimensionality reduction by using python sklearn is described in the following equations (2), (3) and (4).

$$svm\_classifier = svm.SVC() \qquad \rightarrow (2)$$

$$svm\_classifier.fit(X_{train}, Y_{train}) \qquad \rightarrow (3)$$

$$predictions = svm\_classifier.predict(X_{test}) \qquad (4)$$

In equation (2) SVM_classifier formula with svc() classifier is given. The dataset is separated as training set and testing set. The two training sets X_{train} and Y_{train} are given as the input for the fitness function of SVM classification.

The function is$svm_classifier.fit()$. It is described in equation (3). The testing dataset (X_{test}) is passed as the input argument of the SVM classification prediction. The function is $svm_{classifier}\ .predict()$. It is described in equation (4).

3.2.2 Various mode of performance

- **<u>True Positive</u>** (T_p)

The case which are predicted 'yes' and those data which have the defect result.

- **<u>True Negative</u>** (T_n)

The cases which are predicted 'no' and those data which don't have defect result.

- **<u>False Positive</u>** (F_p)

It is called type I error. The cases which are predicted 'yes', but actually those data don't have defect result.

- **<u>False Negative</u>** (F_n)

It is a type II error. The cases which are predicted 'no', but those do have defect result.

3.2.2.1. Accuracy level

It is used to describe how the classifier is accurate to measure the result. It is the percentage of the value predicted with the correct value. It is obtained by using the formula

$$Accuracy\ (A) = (T_p + F_n)/(T_p + T_n + F_p + F_n) \rightarrow (5)$$

Where (T_p) the number of true positives, (F_n) is the number of false negatives, (T_n) is the number of true negatives and (F_p) is the number of false positives. P is the ability of the classifier which is not to label as the positive a sample that is negative. The corresponding python code of accuracy is

$$accuracy = (accuracy - score(Y_{test}, predictions) \rightarrow (5(a))$$

3.2.2.2. Precision

It is the percentage of sample values of the classifier which acquired right out of the total values that is predicted for a data set.

$$Precision\ (P) = (T_p)/(T_p + F_p) \rightarrow (6)$$

Where $(T_p)$the number of is true positives and (F_p) is the number of false positives. P is the ability of the classifier which is not to label as the positive a sample that is negative.

3.2.2.3. Recall

It is the percentage of the sample values of the classifier predicted for a given value of the total number of examples that it should have predicted for the given data set.

$$Recall\ (R) = (T_p)/(T_p + F_n) \quad \rightarrow (7)$$

Where $(T_p)$the number of is true positives and (F_n) is the number of false negatives. R is the ability of the classifier which finds all the positive samples.

3.2.2.4. F1- score

It is nothing but the harmonic mean of the precision and the recall value of the given dataset. It has two values as 1 and 0. If the score reaches 1 then it is the best value and reaches 0 means it is the worst case.

DOI: 10.37200/IJPR/V24I1/PR200282
Received: 22 Nov 2019 | Revised: 26 Dec 2019 | Accepted: 10 Jan 2020

$$F1 - score = 2 * \frac{(Precision\ )*(Recall\ )}{(Precision\ )+(recall\ )} \rightarrow (8)$$

3.2.2.5 Confusion Matrix

The following table 1 is used to define the confusion matrix format. It is used to measure the performance of supervised machine learning.

Table 1: Confusion matrix format

		Classifier Prediction	
Actual Value		**Positive**	**Negative**
	Positive	**True Positive**	**False Negative**
	Negative	**False Positive**	**True Negative**

The python code used for confusion matrix is described in the following equation (9).

$$confusion_{matrix} = (confusion\_matrix(Y_{test}, predictions) \rightarrow (9)$$

3.2.2.6 Classification report

Mapping a function from inputs to output by approximating the function generally forms the predictive modelling. Historical data is used for making the model. The classification report includes precision, recall and f1-score. The format of classification report is shown in the table 2.

Table 2: Classification report format

Precision		Recall	F1-score	Support
class 0	…	…	…	…
class 1	…	…	…	…
…	…	…	…	…
class n	…	…	…	…
avg/total	…	…	…	…

The python code used for classification report is described in the following equation (10).

$$classification_{report} = (classification\_report(Y_{test}, predictions) \rightarrow (10)$$

In the supervised machine learning approach, classification plays an important role. It is used to design the model for describing the performance of the classifier. In this paper,SVM classifier is used to measure the performance measure of the K-Means cluster used to identify the feature of defect dataset.

3.2.2.7 ROC Curve

This graph represents the score of the best defect from each generation. In some generations of the defects, the graph is low and for some of the defect the graph is very high to gain the higher overall fitness value. ROC curve means Receiver Operating Characteristic curve. This curve is used to create a complete sensitivity or specificity report for test of evaluation. According to the ROC curve, the true positive rate that is sensitivity is plotted in the function of the false positive rate that is specificity for different cut-off points of the parameter. Each point of parameters in ROC curve represents a decision threshold.

3.2.2.8 Mean Absolute Error

It is also known as MAE. It is the average mean of all absolute error. It means the difference between the absolute error and the predicted error. The predicted error is the difference between the actual value and the predicted value.

$$MAE = \frac{\sum_{i=1}^{n}|Y_i - X_i|}{n} \qquad \rightarrow (11)$$

where Y_i is the actual value,X_i is the predicted value and 'n' is the total number of observations.

IV. GKM (A Genetic Feature K-means Clustering With SVM Algorithmic Techniques)

This section consists of a working of the GKM techniques presented. This technique is a way of describing the performance of the defect prediction model creation. GKM is divided into six steps. The algorithm is started by data selection to identify the cluster ‘k’. This is done by selecting the feature regarding the one BUG_ID taken at a time from the data set Bugzilla. The second step is Data Extraction, which is used to extract the most commonly features $F_1, F_2, and\ so\ on$. The next step is applying clustering here K-Means Clustering. Generation is formed by finding the similarities between the centroids chosen. Then the performances are measured using SVM techniques by calculating accuracy, precision, recall and f-score.

4.1 GKM Algorithm

Input: *Selected features*. Here$F_1, F_2, F_3 and\ F_4$. known as DECISION_ COUNT, DECISION_ DENSITY, DESIGN_ COMPLEXITY and DESIGN_ DENSITY.

Output: 1. Cluster Accuracy Level.
2. Accuracy, Precision, Recall and F-score.
3. tpr ,fpr and AUC in ROC curve.

Step 1: (*Data Selection*)
- ✓ Find the fitness value of the feature using genetics to confirm the feature of the feature set to be formed.
- ✓ Identify 'K' from the individuals selected from the population feature set done by dimensionality reduction.

Step 2: (*Data preprocessing and Extraction*)
- ✓ Preprocess and Extract to Select the most relevant feature attribute say F_1 , $F_2, and\ so\ on$ by eliminating the unfilled blank data and irrelevant data. It is used to form training set and testing set.

Step 3: (*K-Means clustering*)
- ✓ Clusters are formed with the selected point to be near to the Euclidean distance. C_1 ,$C_2$and so on. Always lesser k-value used to get the accurate result.

Step 4: (*Generic Generation*)
- ✓ For each defect list calculate the centroid using similarities summary report.
- ✓ Feature selection is done by finding means.
- ✓ Update the centroids in each stage to check and extract the feature.
- ✓ Iterate the above three steps until all the points in the centroid feature relocated in the mean of clusters.
- ✓ Find the accuracy values by preprocessing until there is decrease in the accuracy level.
- ✓ Finalize the last increase value as the accuracy result.

Step 5: (*SVM - classification to Performance Evaluation)*
- ✓ Use the dataset obtains in step 1 and 2 for dataset selection and preprocessing.
- ✓ Compute Accuracy, Precision, Recall and F-score using tpr and fpr of SVM.

Step 6: (*ROC-AUC curve)*
- ✓ Measure the level of accuracy using ROC curve in true positive rate and false positive rate.

V. DATA COLLECTION

We choose the selected features from the Bugzilla report database, which is submitted by any reporter with details about the defect was observed. In this the detail status of a defect also mentioned. In addition, the severity and priority issue of a defect with range also mentioned. The various status of defect in Bugzilla database are new, assigned, and resolved, closed, fixed, unconfirmed and invalid. The dataset is downloaded in the local machine and stored in the form of .csv format file. The csv file is read by using pandas library as "*dataset = pandas. read_csv("path")*".

The selected feature set of defects using genetics in our previous paper in the basis of development phases is listed in the table 3. The defect data classification done into four categories like design phase, implementation phase, testing phase and maintenance phase.

Table 3: Selected Feature List in the Development Phases using Genetics

DESIGN PHASE	IMPLEMENTATION PHASE	TESTING PHASE	MAINTENANCE PHASE
DECISION_COUNT	LOC_BLANK	CYCLOMATIC_COMPLEXITY	MAINTENANCE_SEVERITY
DECISION_DENSITY	CALL_PAIRS	CYCLOMATIC_DENSITY	
DESIGN_COMPLEXITY	LOC_CODE_AND_COMMENT	NORMALIZED_CYLOMATIC_COMPLEXITY	
DESIGN_DENSITY	LOC_COMMENTS	HALSTEAD_CONTENT	
EDGE_COUNT	LOC_EXECUTABLE	HALSTEAD_DIFFICULTY	
ESSENTIAL_COMPLEXITY	PERCENT_COMMENTS	HALSTEAD_EFFORT	
ESSENTIAL_DENSITY	LOC_TOTAL	HALSTEAD_ERROR_EST	
	BRANCH_COUNT	HALSTEAD_LENGTH	
	PARAMETER_COUNT	HALSTEAD_LEVEL	
	CONDITION_COUNT	HALSTEAD_PROG_TIME	
	MODIFIED_CONDITION_COUNT	HALSTEAD_VOLUME	
	MULTIPLE_CONDITION_COUNT		
	NODE_COUNT		
	NUM_OPERANDS		
	NUM_OPERATORS		
	NUM_UNIQUE_OPERANDS		
	NUM_UNIQUE_OPERATORS		
	NUMBER_OF_LINES		

VI. GKS IMPLEMENTATION

6.1 Data selection

Using step 1 of GKS algorithm the above table is formed with 29 feature set. We chose only four main features class level metrics known as DECISION_COUNT, DECISION_DENSITY, DESIGN_COMPLEXITY and DESIGN_DENSITY as the cluster learning technique fields.

DOI: 10.37200/IJPR/V24I1/PR200282
Received: 22 Nov 2019 | Revised: 26 Dec 2019 | Accepted: 10 Jan 2020

6.2 Data preprocessing

By using algorithm step 2 the two set of input is formed one is training set and another is testing set. In machine learning we must have both the testing and training data to be identical and independent distribution. Then only the measurements are meaningful. After the feature formed from the data set random seeding is used for dataset separation. Our dataset has total number of rows as 344. A new vector variable is created by using random splitting ratio as 0.5. So, the dataset of 344 rows is separated as 50% of training set and 50% of testing set as 174 rows in training set and 170 in testing set. It is displayed in the following table 4 and 5.

Table 4: Training data set of defect feature

S. No	defect feature	data type	total defects
1	DECISION_COUNT	non-null float64	174
2	DECISION_DENSITY	non-null float64	174
3	DESIGN_COMPLEXITY	non-null float64	174
4	DESIGN_DENSITY	non-null float64	174

The above table 4 describes the set of input and output examples provided for the experiment. The training set contains several records about the defects of software. It has four features DECISION_DENSITY, DESIGN_COMPLEXITY and DESIGN_DENSITY. The data set is labeled as defective or non-defective. By dropping defective column, the dataset is turned into unlabeled so as to fit for clustering. The objective is to cluster the records as defective or not.

Table 5: Testing data set of defect feature

S.No	**Defect Feature**	**Data Type**	**total defects**
1	DECISION_COUNT	non-null float64	170
2	DECISION_DENSITY	non-null float64	170
3	DESIGN_COMPLEXITY	non-null float64	170
4	DESIGN_DENSITY	non-null float64	170

The above table 5 is used for generalizing the performance of the model used for producing the correct output values for the input. This testing set is used to get the objective measurement of learning performance. The testing set contains several records about the defects of software. It has fourfeatures like DECISION_ COUNT, DECISION_DENSITY, DESIGN_COMPLEXITY and DESIGN_DENSITY.

6.3 K-Means Clustering

By step 3 the clustering implementation is done by using python coding. Major packages like pandas, numpy, scikit-learn, seaborn and matplotlib are used. The quick plotting graph is used to display significance level of features chosen. All these are classified as defective and non-defective. The separation of the four selected features in the basis of defective and non-defective is shown in the following graph using python seaborn histogram in diagram 2, 3, 4, and 5.

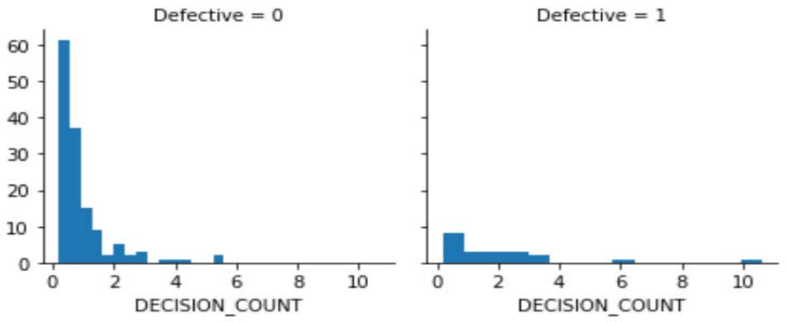


Diagram 2: Histogram of DECISION_COUNT in the basis of Defective and Non_Defective

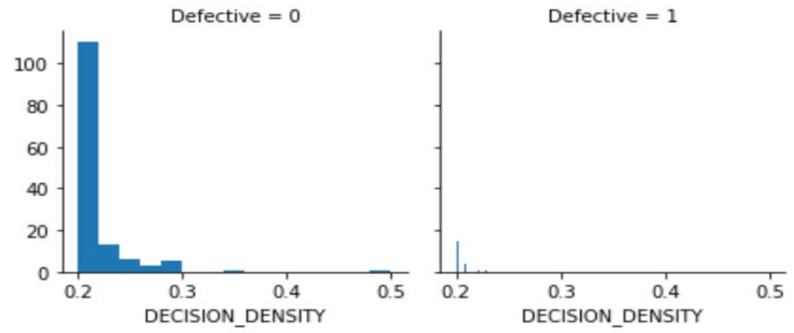


Diagram 3: Histogram of DECISION_DENSITY in the basis of Defective and Non_Defective

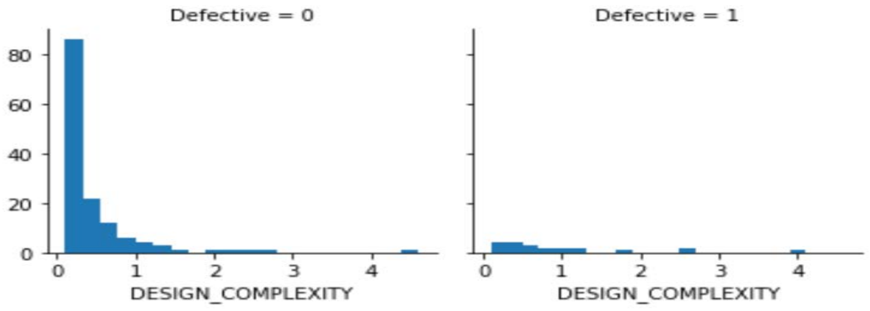


Diagram 4: Histogram of DESIGN_COMPLEXITY in the basis of Defective and Non_Defective

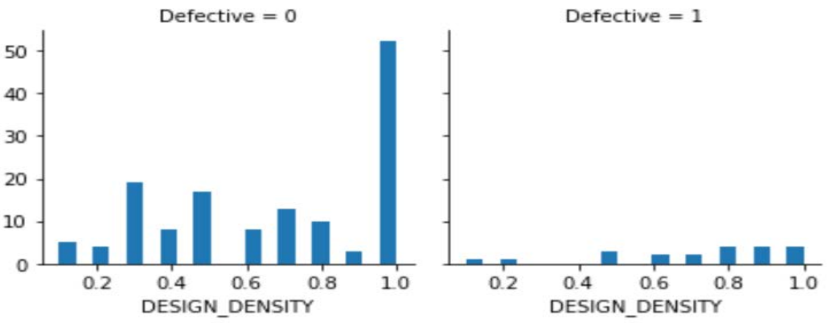


Diagram 5: Histogram of DESIGN_DENSITY in the basis of Defective and Non_Defective

The training and testing set using in the K-Means clustering in python programming of Bugzilla dataset is listed in the following table 6 and table 7. The memory usage of training set is7.6 KB and testing set is 8.8 KB.

Table 6: Training data result in K-Means clustering

cluster fields	**DECISION_COUNT**	**DECISION_DENSITY**	**DESIGN_COMPLEXITY**	**DESIGN_DENSITY**	**Defective**
count	174	174	174	174	174
mean	1.025287	0.211897	0.503448	0.69483	0.12069
std	1.241034	0.032388	0.642961	0.28514	0.326706
min	0.2	0.2	0.1	0.1	0
25%	0.4	0.2	0.2	0.5	0
50%	0.6	0.2	0.3	0.7	0
75%	1.2	0.2	0.575	1	0
max	10.6	0.5	4.6	1	1

Table 7: Testing data result in K-Means clustering

cluster fields	**DECISION_ COUNT**	**DECISION_ DENSITY**	**DESIGN_ COMPLEXITY**	**DESIGN_ DENSITY**	**Defective**
count	170	170	170	170	170
mean	0.876471	0.211353	0.451765	0.73059	0.123529
std	1.255346	0.030902	0.606398	0.25583	0.330016
min	0.2	0.2	0.1	0.1	0
25%	0.2	0.2	0.125	0.5	0
50%	0.4	0.2	0.3	0.8	0
75%	1	0.21	0.575	1	0
max	11.8	0.4	6.3	1	1

Following diagram 6 represents the scatterplot of the defect Features DECISION_COUNT, DECISION_DENSITY,DESIGN_COMPLEXITY and DESIGN_DENSITY with three set of scatters. Diagram 7 have small generalized variance of clusters moved using the centroids. It is performed using the K-Means Clustering algorithm's k-value. The cluster implementation of defective and non-defective set is visually described in diagram 6 and 7. The diagram 6 and 7 shows two different clusters known as defective and non-defective. It shows the centroids in two clusters defective and non-defective. Pink color dot in the circle is the defective set centroid (diagram 6) and purple color dot is the non-defective set centroid (diagram 7).

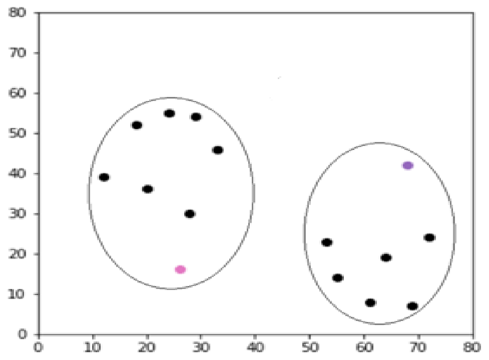


Diagram 6: Euclidean for centroid selection of cluster in defective /non-defective data in K-Means clustering (k=2)

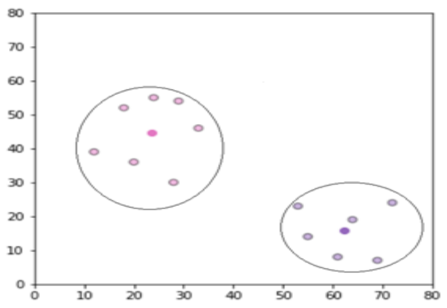
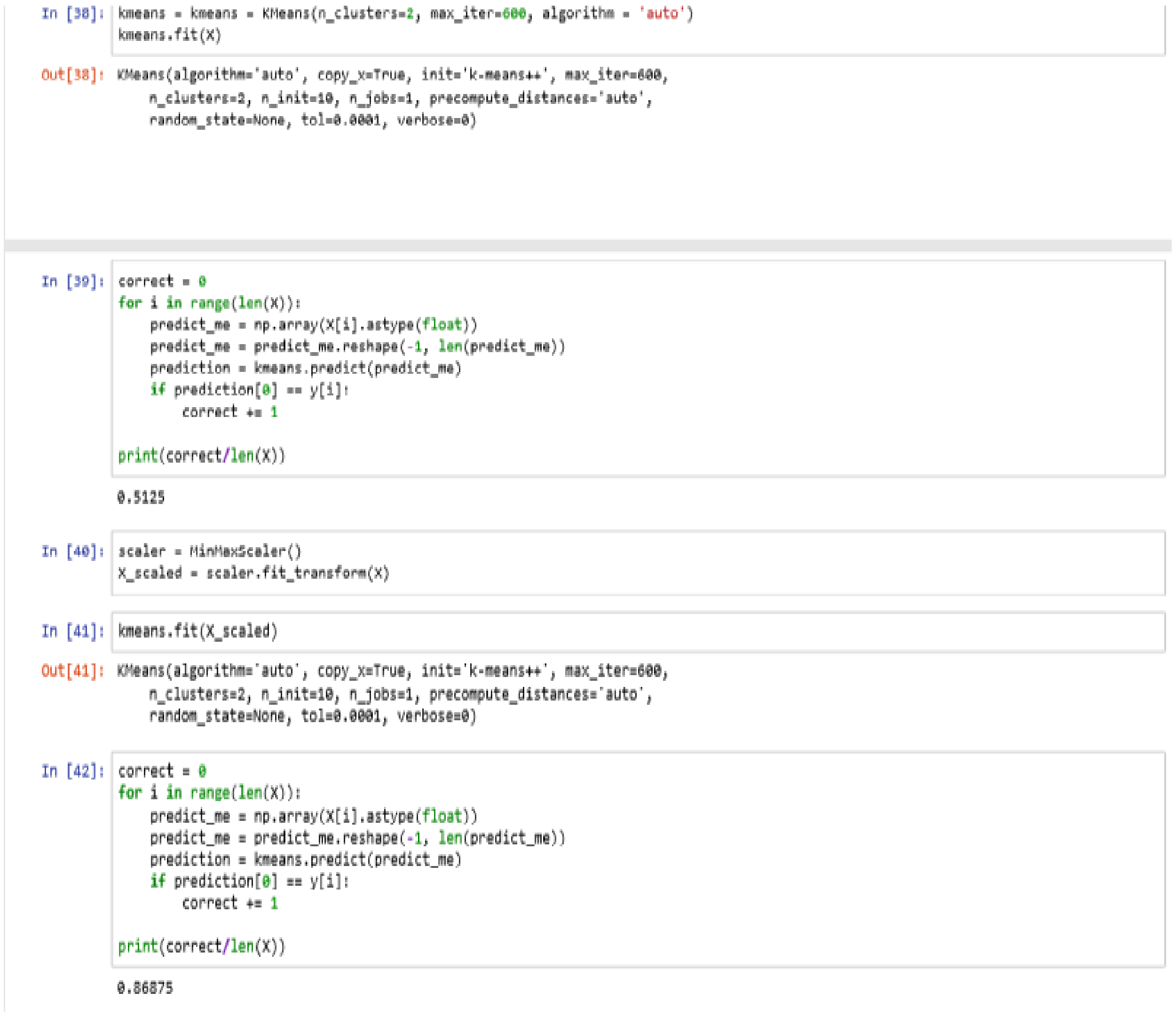


Diagram 7: Centroid moved to the mean cluster in defective and non-defective data in K-Means clustering (k=2)

The data points in the dataset are assigned to the nearest Euclidean distance. Once the clusters are formed the centroids updated to form the new centroids. The main objective of clustering is iterating this process until the centroids become the mean of the clusters. The predictions are done according with the nearest centroids. The python running snapshot is shown in the below Diagram 8.



```
In [38]: kmeans = kmeans = KMeans(n_clusters=2, max_iter=600, algorithm = 'auto')
         kmeans.fit(X)

Out[38]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=600,
             n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
             random_state=None, tol=0.0001, verbose=0)

In [39]: correct = 0
         for i in range(len(X)):
             predict_me = np.array(X[i].astype(float))
             predict_me = predict_me.reshape(-1, len(predict_me))
             prediction = kmeans.predict(predict_me)
             if prediction[0] == y[i]:
                 correct += 1

         print(correct/len(X))

         0.5125

In [40]: scaler = MinMaxScaler()
         X_scaled = scaler.fit_transform(X)

In [41]: kmeans.fit(X_scaled)

Out[41]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=600,
             n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
             random_state=None, tol=0.0001, verbose=0)

In [42]: correct = 0
         for i in range(len(X)):
             predict_me = np.array(X[i].astype(float))
             predict_me = predict_me.reshape(-1, len(predict_me))
             prediction = kmeans.predict(predict_me)
             if prediction[0] == y[i]:
                 correct += 1

         print(correct/len(X))

         0.86875
```

Diagram 8: Python snapshot of K-Means clustering of defect dataset

Only four features are chosen from the 29 feature values because these are better to train the model and will have significant impact in the training of K-Means clustering. It is done by feature engineering by using best fit value. The k-means cluster is built by using equation (1) in python coding. The first value is 0.5125 which is 50% of the accuracy level. Again, preprocessing done and we obtain the result as 0.86875 which is 86% of accuracy. Again, scale we obtain the result as 0.69672 which shows 17% decrease in the accuracy. So, we stop scaling at 86% of accuracy. It is the final accuracy level of k-Means clustering of the defective dataset.

6.4 SVM classification

6.4.1 Dataset selection

It is done by dataset reading and feature vector forming using sklearn. Then training set and testing set are used to form the mode of performances. The various performances are accuracy, confusion matrix, precision, recall and f-score. Finally, the result is plotted using ROC curve. The SVM.svc classifier is used in our dataset.

6.4.2 Dataset preprocessing

The training and the testing feature set are used for measuring the mode of performances. The range values of defect Features DECISION_COUNT, DECISION_DENSITY, DESIGN_COMPLEXITY and DESIGN_DENSITY in relation with defective used in the SVM classifier is displayed in the following box and whisker plots diagram 9.

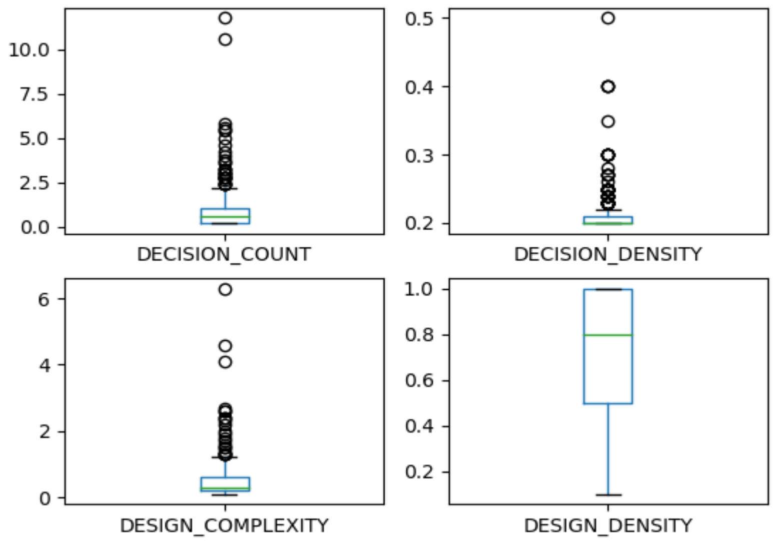


Diagram 9: Whisker Plot of selected features in relation with defective

SVM learning techniques helps to analyze the accuracy level of the clustered features (here K-Means clustering). This is done by developing the approach to creating the training and the test dataset for the problem domain to analyze the implementation’s runtime options. The following table 5 is used to describe the mean and standard deviation used in the clustering algorithm. The table also lists the total data set count of the four features selected DECISION_COUNT, DECISION_DENSITY,DESIGN_COMPLEXITY and DESIGN_DENSITY in relation with defective as 344. The minimum and maximum values of the features also listed.

Table 8: Performance values of SVM Classification

Classification_ field	DECISION_ COUNT	DECISION_ DENSITY	DESIGN_ COMPLEXITY	DESIGN_ DENSITY
count	344	344	344	344
mean	0.951744	0.211628	0.477907	0.7125
std	1.248531	0.031618	0.624785	0.271247
min	0.2	0.2	0.1	0.1
25%	0.2	0.2	0.2	0.5
50%	0.6	0.2	0.3	0.8
75%	1	0.21	0.6	1
max	11.8	0.5	6.3	1

6.4.3 Accuracy, Precision, Recall and F1-score

Accuracy value is calculated by using the function accuracy_ score (). The calculated value for our defect dataset is 0.9855. Precision value is calculated by using the function precision_ score (). The calculated value for our defect dataset is 1.0. Recall value is calculated by using the function recall_ score (). The calculated value for our defect dataset is 0.9. Here F1 score is a helpful metric for comparing the classifiers defective and non-defective. F1- score value is calculated by using the function f1_ score (). The calculated value for our defect dataset is 0.9473.

6.4.4 Confusion matrix

This matrix is used to measure the performance of a SVM classifier with a fixed threshold. The confusion matrix used to show the particular misclassification in the calculation which is desired. The confusion matrix obtained for the defective dataset is

```
confusion matrix - by SVM_ classifier
*************************************
[[59 0]
[ 1 9]]
```

6.4.5 Classification report

It is the visual display of the precision, recall, f1-score and support score for the model used. It gives the deeper intuition of the classifier (here, SVM) behavior over accuracy. It is also used to report to select as model or not for the dataset considered. The classification report obtained for the defective dataset is

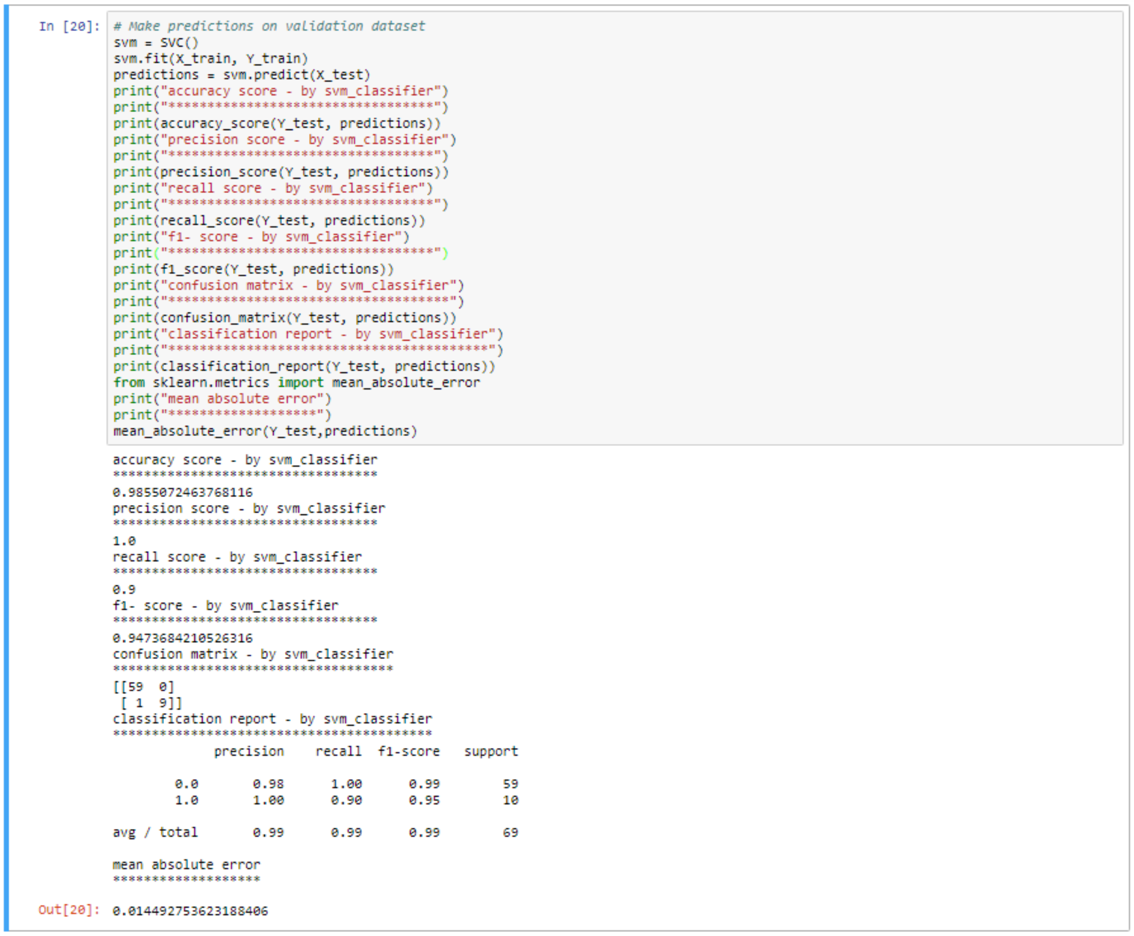
```
classification report - by SVM_ classifier
******************************************
precision recall f1-score support


 0.0 0.98 1.00 0.99 59
 1.0 1.00 0.90 0.95 10


avg / total 0.99 0.99 0.99 69
```

6.4.6 Mean Absolute Error

Mean absolute error (MAE) is a measure of the difference between the predicted and observed results. Here the MAE value for the defective dataset is 0.0144.The python running snapshot of the SVM classification is shown in the Diagram 10.



```
In [20]: # Make predictions on validation dataset
         svm = SVC()
         svm.fit(X_train, Y_train)
         predictions = svm.predict(X_test)
         print("accuracy score - by svm_classifier")
         print("**********************************")
         print(accuracy_score(Y_test, predictions))
         print("precision score - by svm_classifier")
         print("**********************************")
         print(precision_score(Y_test, predictions))
         print("recall score - by svm_classifier")
         print("**********************************")
         print(recall_score(Y_test, predictions))
         print("f1- score - by svm_classifier")
         print("**********************************")
         print(f1_score(Y_test, predictions))
         print("confusion matrix - by svm_classifier")
         print("************************************")
         print(confusion_matrix(Y_test, predictions))
         print("classification report - by svm_classifier")
         print("*****************************************")
         print(classification_report(Y_test, predictions))
         from sklearn.metrics import mean_absolute_error
         print("mean absolute error")
         print("*******************")
         mean_absolute_error(Y_test,predictions)

         accuracy score - by svm_classifier
         **********************************
         0.9855072463768116
         precision score - by svm_classifier
         **********************************
         1.0
         recall score - by svm_classifier
         **********************************
         0.9
         f1- score - by svm_classifier
         **********************************
         0.9473684210526316
         confusion matrix - by svm_classifier
         ************************************
         [[59  0]
          [ 1  9]]
         classification report - by svm_classifier
         *****************************************
                      precision    recall  f1-score   support

                 0.0       0.98      1.00      0.99        59
                 1.0       1.00      0.90      0.95        10

         avg / total       0.99      0.99      0.99        69

         mean absolute error
         *******************
Out[20]: 0.014492753623188406
```

Diagram 10: Python snapshot of K-Means clustering of defect dataset

6.4.7 AUC and ROC curve

AUC measures the two-dimensional area under the entire ROC curve between (0,0) and (1,1). The value is approximated by using the function auc (). The calculated value of the AUC for the defective dataset is 0.950. The range of AUC is from 0 to 1. The AUC value is 0 means the prediction was fully wrong and if the value is 1.0 means the value is fully correct. If the value is close to 1 means classifier is better for the dataset. The ROC curve is a graph used to show the performance of a classification model at all thresholds. The ROC curve is a measure of discriminates between the two groups of datasets defective or non-defective. To activate the accuracy of test's results, ROC is used. ROC curve along with confusion matrix defect map has been the key in making the analysis of defect classification. Our ROC analysis finds the T_p (Y-axis) is the true positive rate which means Precision Rate and F_p (X-axis) is the rate of Recall Rate respectively. These two are the mathematical quantities used for the deciding structure. The ROC AUC curve is diagrammatic represented in Diagram 11.

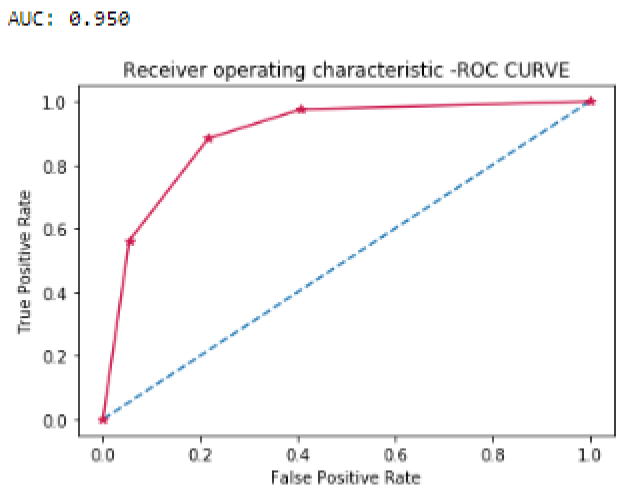


Diagram 11: ROC AUC curve for performance analysis of SVM

The above diagram 11 is the ROC curve obtained to verify the performance of the K-Means clustering is verified using SVM the supervised learning.

VII. Conclusion

Population of software growing in exponential way. Software failure in post-production is also increased day by day. The only alternative to control this is to predict the defect in software and prevent it before it occurs. In this paper, clustering and classification are used for making software defect taxonomy. Our hybrid approach is the combination of unsupervised clustering (K-Means Clustering) and supervised classification [Support Vector Machine (SVM)] machine learning and forms a newly developed algorithmic technique known as GKS. It first performs feature selection and pre-processing and then the necessary labels are generated for each cluster by K-Means technique with 86% of accuracy. The cluster labels are mapped to form defect taxonomy list. The performance is evaluated by calculating Precision – 1.0, Recall – 0.9, f1 score - 0.9473 and accuracy level – 0.9855 using SVM Technique. The result is plotted in to ROC – AUC curve. This curve gives the accuracy rate as 0.950 which is higher accuracy rate of Prediction. Therefore, it shows the GKS technique is best for Defect Prediction Taxonomy formation. As software are developed every year for all domains, huge software dataset is available. Researches are applying machine learning techniques on this data to analyze defect in software. By using Defect taxonomy for identifying the early defects in software will reduce the rework in development. A good Defect Taxonomy helps to avoid misunderstanding and it also improves the development process. The Performance reports shown in this paper for machine learning algorithms K-Means clustering and SVM proved to be best to classify the defect knowledge from large amount of available dataset.

References

[1] Application of Genetic Algorithms in Machine learning, Harsh Bhasin, Surbhi Bhatia, *(IJCSIT) International Journal of Computer Science and Information Technologies,* Vol. 2 (5) , 2011, 2412-2415

[2] Software Defect Prediction Models for Quality Improvement: A Literature Study, Mrinal Singh Rawat1, Sanjay Kumar Dubey, *IJCSI International Journal of Computer Science Issues,* Vol. 9, Issue 5, No 2, September 2012

[3] Software Fault Prediction Exploration Using Machine Learning Techniques, P. Patchaiammal, R. Thirumalaiselvi, *International Journal of Recent Technology and Engineering (IJRTE),* ISSN: 2277-3878, Volume-7 Issue-6S3 April, 2019

[4] Contemporary Trends in Defect Prevention: A Survey Report, Muhammad Faizan, Muhammad Naeem Ahmed Khan, Sami Ulhaq, I.J. *Modern Education and Computer Science,* 2012, 3, 14-20

[5] Analytical Survey on Bug Tracking System, Dr. Sanjay Kumar Dubey, Shivani, *International Journal of Computer and Communication System Engineering (IJCCSE)* Vol. 1 No.02 August 2014

[6] Ada95 Object-Oriented and Real-Time Support for Development of Software Fault Tolerance Reusable Components, Eltefaat H. Shokri Kam S. Tso.

[7] High-Coverage Fault Tolerance in Real-Time Systems Based on Point-to-Point Communication, K. H. (Kane) Kim, Chittur Subbaraman, Eltefaat Shokri

[8] ReSoFT: A Reusable Testbed for Development and Evaluation of Software Fault-Tolerant Systems, Kam S. Tso, Eltefaat H. Shokri, Roger J. Dziegiel, Jr.

[9] Engineering Oriented Dependability Evaluation: MEADEP and Its Applications, Dong Tang, Myron Hecht, Jeffrey Agron, Jeffrey Miller, Herbert Hecht, 1997 *Pacific Rim International Symposium on Fault-Tolerant Systems,* Taipei, Taiwan, Dec. 15-16, 1997, pp. 85-90

[10] Protection against Software Failures: Low Failure Rate and Fault Tolerance, Herbert Hecht

[11] A Software Flaw Taxonomy: Aiming Tools at Security, Sam Weber Paul A. Karger Amit Paradkar, *Software Engineering for Secure Systems – Building Trustworthy Applications, (SESS'05) St. Louis, Missouri, USA.*

[12] A Survey on Software Testing Techniques using Genetic Algorithm, Chayanika Sharma, Sangeeta Sabharwal, RituSibal, *IJCSI International Journal of Computer Science Issues,* Vol. 10, Issue 1, No 1, January 2013.

[13] Finding Defective Modules from Highly Unbalanced Datasets, J C Riquelme, R Ruiz, D Rodrıguez, J Moreno, Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, Vol. 2, No. 1, 2008

[14] Re-using Generators of Complex Test Data, Simon Poulding, Robert Feldt, Software Engineering Research Lab (SERL Sweden), Blekinge Institute of Technology, Sweden.

[15] A Taxonomy for Requirements Engineering and Software Test Alignment, M. Unterkalmsteiner, R. Feldt, T. Gorschek, *ACM Transactions on Software Engineering and Methodology,* Vol. V, No. N, Article A

[16] A survey on software fault detection based on different prediction approaches, GolnoushAbaei, Ali Selamat, *Vietnam J Comput Sci* (2014) 1:79–95.

[17] Analogy Based Defect Prediction Model, Elham Paikari.

[18] Defects4J: A Database of Existing Faults to Enable Controlled Testing Studies for Java Programs, René Just, Darioush Jalali, Michael D. Ernst ISSTA '14, July 21–25, 2014, San Jose, CA, USA

[19] Automatic Extraction of Bug Localization Benchmarks from History, Valentin Dallmeier, Thomas Zimmermann.

[20] https://en.wikipedia.org/wiki/Precision_and_recall

[21] https://bugs.eclipse.org/bugs/buglist.cgi?quicksearch=functional

[22] https://www.bugzilla.org.