CSE471 - Statistical Methods in AI

Project Report

Project Title

Techniques for Intrusion Detection

Team 7

Murtuza Bohra (20172104),
Rashmi KethiReddy (20172044),
Abhay Rawat (20172082),
Nitin Ramrakhiyani (20172091)
Mentor TA: Hemanth Kumar Veeranki

Introduction

The project aims at exploring and implementing techniques for Intrusion Detection in Networks. The basic task and data are as per the KDD Cup 99 contest of detecting intrusions. Details of the contest can be obtained from: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Data

The data consists of feature representations of various network connections made in a empirically modelled pseudo - US Air force network. Each data point describes a connection with 41 features of which 34 are continuous and 7 are categorical features.

The data set consists of about 50 lakh such connection instances. A smaller dataset of size 10% is also provided which has about 5 lakh instances. The 10% dataset forms the basis of most of the experimentation carried out in this project.

The features and are described in Table 1.

duration: continuous.	urgent: continuous.		is_host_login: symbolic.	dst_host_count: continuous.
protocol_type: symbolic.	hot: continuous.		is_guest_login: symbolic.	dst_host_srv_count : continuous.
service: symbolic.	num_failed_logins: continuous.		count: continuous.	dst_host_same_srv _rate: continuous.
flag: symbolic.	logged_in: symbolic.		srv_count: continuous.	dst_host_diff_srv_r ate: continuous.
src_bytes: continuous.	num_compromised : continuous.		serror_rate: continuous.	dst_host_same_src _port_rate: continuous.
dst_bytes: continuous.	root_shell:		srv_serror_rate:	dst_host_srv_diff_h

	continuous.		continuous.	ost_rate: continuous.
land: symbolic.	num_root: continuous.		rerror_rate: continuous.	dst_host_serror_rate: continuous.
wrong_fragment: continuous.	num_file_creations: continuous.		srv_rerror_rate: continuous.	dst_host_srv_serror_rate: continuous.
su_attempted: continuous.	num_shells: continuous.		same_srv_rate: continuous.	dst_host_rerror_rate: continuous.
num_outbound_cmds: continuous.	srv_diff_host_rate: continuous.		diff_srv_rate: continuous.	dst_host_srv_rerror_rate: continuous.
num_access_files: continuous.				

Table 1: Features for each connection instances

Each instance is classified into 5 major classes - normal, DoS, probe, r2l, u2r. Except the normal class the rest of the four classes represent the attack class. The class distribution is shown in Table 2.

Class	Count
normal	97278
attack ● DoS ● probe ● r2l ● u2r	396743 391458 4107 1126 52

Table 2: Class distribution in the dataset

Fast Feature Reduction

In this section we introduce some of most successful similarity measures. These successful similarity measures includes: Correlation Coefficient, Least Square Regression Error and Maximal Information Compression Index.

Correlation Coefficient

One of the most popular and known measures of similarity between two random variable is correlation coefficient. It also called cross-correlation coefficient. We can use (1) to measure how much random variable x is similar to random variable y

$$Relation(x,y) = \frac{cov(x,y)}{\sqrt{var(x) \times var(y)}}$$

where var() corresponds to variance of variable x and cov() correspond to covariance of random variable x and y .Covariance of two variables shows strength of relation between those variables. If x grow when y rises with exact linear relation, covariance of these two variable will be 1 and if x shrink when y rises covariance will be -1. If two variables are unrelated at all covariance will be 0. We can use this as a similarity measure. For reducing features we will look as dissimilarity instead of similarity and hence our final relation or similarity metric would look like this:

1 - Relation(x,y)

Least Square Regression Error:
Another similarity measure that is used regularly is least square regression error or residual variance. It is the error of predicting y from y=bx+a. Where a and b are the regression coefficients which can be calculated by minimizing e(x,y)2 in:

$$e(x,y)^2 = \frac{1}{n}\sum e(x,y)_i^2$$

Where e(x,y)i can be calculated with

$$e(x,y)^2 = y_i - a - bx_i$$

Which gives us the following values for a and b:

$$a = \overline{y}$$

$$b = \frac{cov(x,y)}{var(x)}$$

The final relation comes out to be as follows. The closer this value is to 0 the more linearly related the features are.

$$e(x,y) = var(y) \times (1 - Relation(x,y)^2)$$

Maximal Information Compression Index
This measure fixed some defects of two previous described measures. We refer to it by MICI and use □𝔩 for simplicity. MICI can be calculated using:
Where ∑ correspond to covariance of x and y and egiv is a vector of eigenvalues of that covariance matrix.

$$\lambda_2(x,y) = \min(eigv(\sum))$$

𝔩 will equal to 0 when the feature have linear relation and as much as the relation fade away, the value of 𝔩 increases. As the formula represent 𝔩 is only an eigenvalue for a direction where data have the most elongation. In the other word it is the main idea behind Principle Component Analysis (PCA). PCA is based on the fact that if a data be projected along its principal component direction, it yields maximum information compaction. If we reconstruct the data, the amount of lost information is related to eigenvectors that is not considered in PCA calculation. In

PCA we always use k eigenvectors that have maximum corresponding eigenvalues. So the amount of lost information will be minimal.

Feature Selection Method

The task of feature selection involves two steps, namely, partitioning the original feature set into a number of homogenous subsets(clusters) and selecting a representative feature from each cluster. Partitioning of the features is done based on the k-NN principle using the similarity measures discussed above. In doing so, we first compute the k nearest neighbours of each feature. Among the the feature having the most compact subset is selected, and its k neighbouring features are discarded. The process is repeated for the remaining features until all of them are either selected or rejected.

While determining the k nearest neighbours of the features, we assign a constant error threshold (ϵ) which is set equal to the distance of the k-th nearest neighbour of the feature selected in the first iteration. In subsequent iterations , we check the similarity measure value, corresponding to a subset of features, whether it is greater than ϵ or not. If yes, then we decrease the value of k. Therefore the value of k is varying over iterations.

Proposed Method - Fast Feature Reduction

Let X={xi}1N be our intrusion detection dataset, containing network connection samples containing all attack and normal connections. Each is a vector with D dimension. The idea of feature selection is to take the features with highest variance across the classes, so that the classification objective is achieved.

Algorithm:

1. Partition the data into different classes.

2. Calculate the mean feature vector for each class.

3. Now calculate the global mean of the class means.

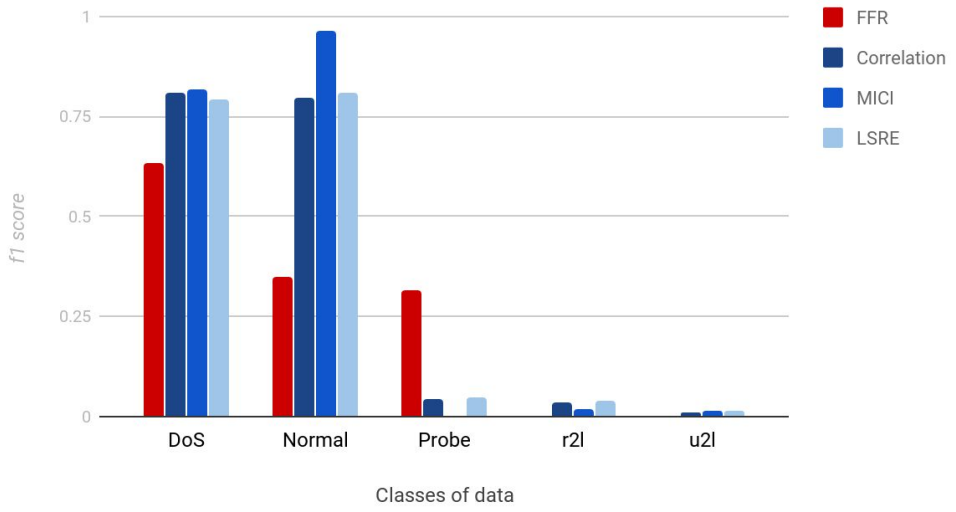
4. Compute variance about the global mean.

5. Take the top t features with highest variance across the different classes.

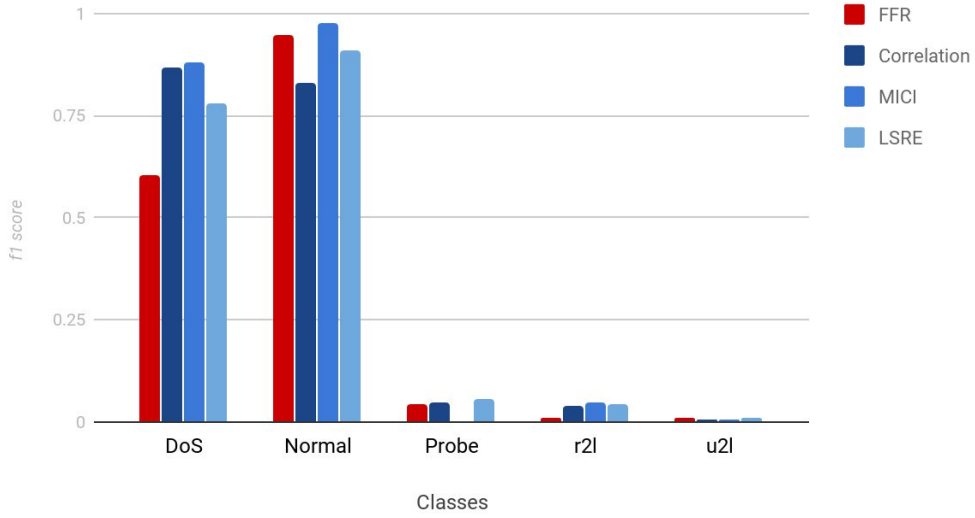
Challenges:

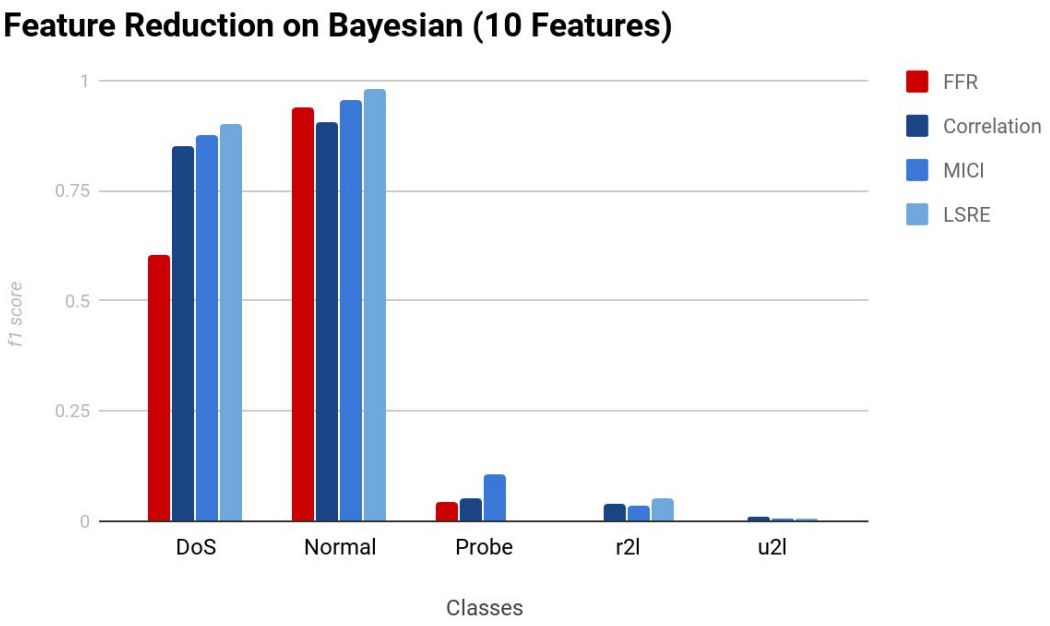
1. Normalization of data: Because the scale of each feature can be different from the other feature, the variance needs to be normalised on a scale of 1.
2. Categorical features: There is no direct method to compute the variance of categorical features, so we tried 2-3 different approaches -
    a. Integer representation of categorical values
    b. One hot encoding of categorical feature by splitting it into binary feature of it's all different values.
    c. Doing feature reduction on other feature keeping all 7 categorical feature as it i\

Feature Reduction on Bayesian (30 Features)



Feature Reduction on Bayesian (20 Features)





Classification using SVM and Neural Networks

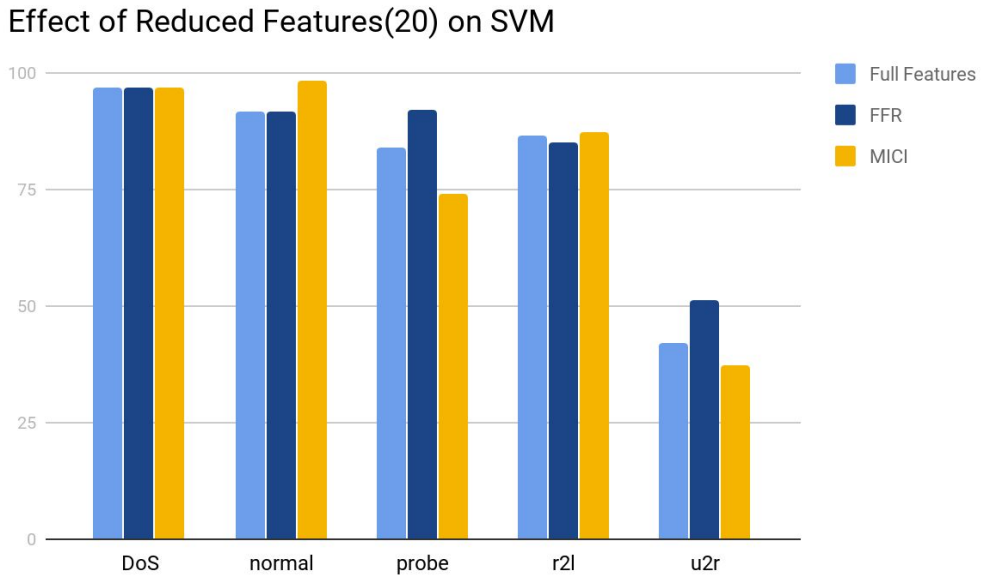
The problem can be solved by posing it as a classification problem and using Support vector machines (SVM) and Neural Networks as classifiers. The paper - 'Intrusion Detection Using Neural Networks and Support Vector Machines' by Mukkamala et al. uses this approach to solve the problem. Some details about their approach are as follows:

- They treat the problem as a binary classification problem with two classes - attack and normal. They ignore the finer classification of the attack class into DoS, probe, r2l and u2r.
- For both the approaches, they use a subset of 14292 instances and split the dataset into 7312 as training and the rest as testing.
- For the SVM they use the RBF kernel and report accuracy of 99.5%.
- For the neural network they use a multi-layer feed-forward network with the nodes arranged as
  Input:41~~~Hidden1 (Sigmoid):40~~~Hidden2 (Sigmoid):40~~~Dense(Sigmoid):1
  They report an accuracy of 99.25% using the neural network.

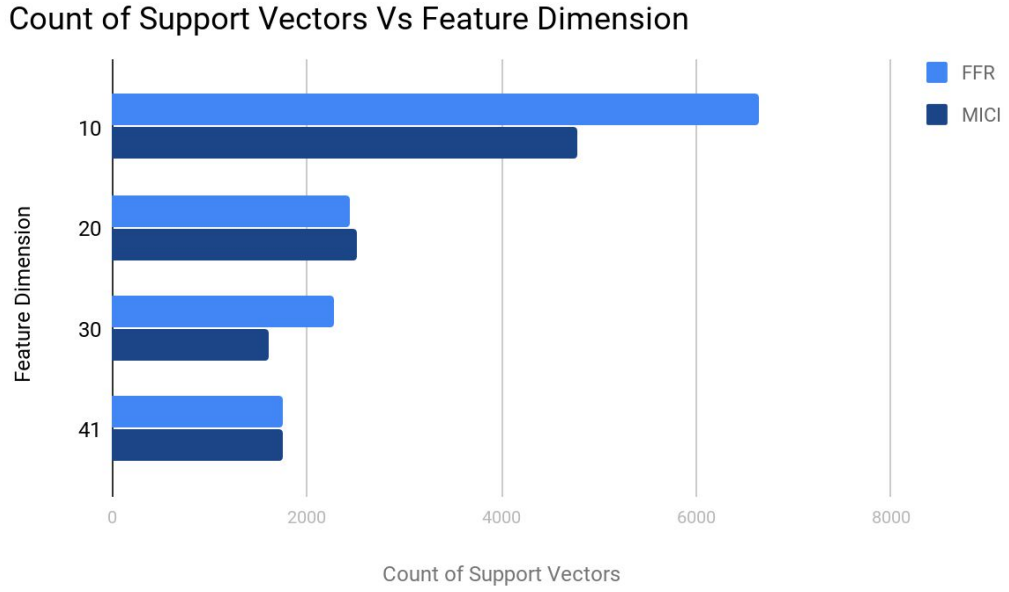
Our Contributions

- We also setup the same experiment as the paper, but instead of using a smaller set of 15000 instances we employ the techniques on the full KDD 10% dataset containing 5 lakh examples.
- We pose the problem also as a multi-class classification problem and solve it using modified a neural network and a one-vs-one SVM. The neural network's output layer is modified to a 5 node layer with softmax activation i.e:
  Input:41~~~Hidden1 (Sigmoid):40~~~Hidden2 (Sigmoid):40~~~Dense(Softmax):5
- Additionally, we try the experiments with two reduced sets of features obtained from the techniques discussed in Paper 1.

Following graphs present a representative set of results we obtain.
Graph1: The effect of reduced features on the performance of the SVM is shown through this F1 score vs classes graph

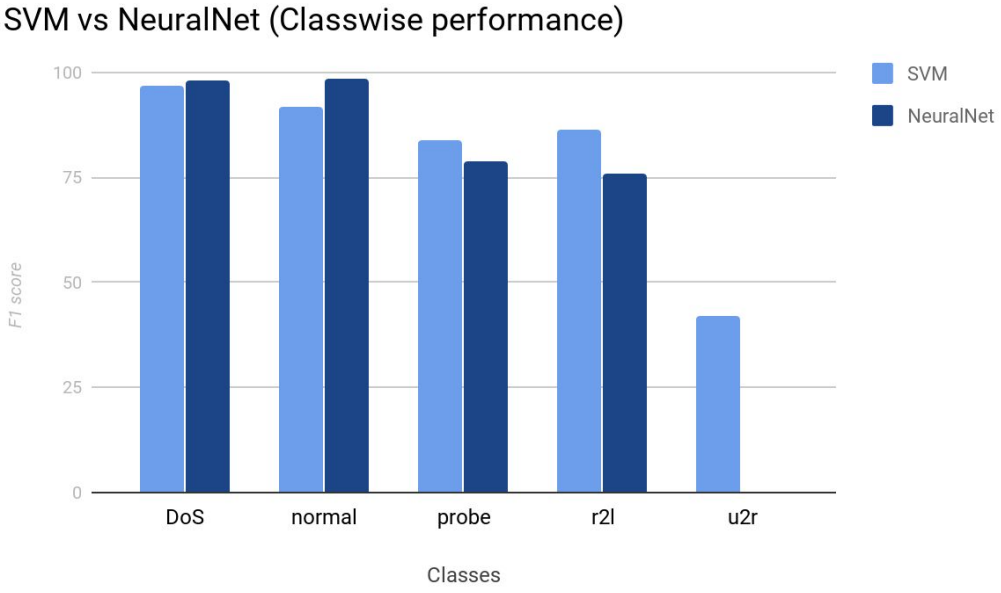


Graph2: The effect of reduced features on the support vectors generated in the SVM is shown through this number of features vs support vectors graph. It is important to observe that as the number of features decrease the data representation is made compact and the classifier requires mode support vectors to classify the data.



Graph 3: The following graph shows the comparative performance of the SVM and Neural network for the classes in an all feature setting. It is important to observe that the SVM manages

to classify little for the scantily represented class u2r but the neural network fails to do so.



Unsupervised Unlabeled Clustering

In real time scenario the data collected from the network would be more of normal activity than the attack activity and often new intrusion activities come up which are not in the training data but come up during testing. In both the above scenarios supervised learning will train on the labeled data which has only known intrusions and equal distribution of normal and attack activities.

In this section a clustering algorithm which will help to detect the attacks from the data collected from network without having any human interventions for labelling the data. Unsupervised unlabeled clustering data can work on the data collected from the network and train on the data with intrusions induced in it. There are 2 assumptions for this algorithm : Normal data is more dense than the intrusion data and Normal instances vary qualitatively from intrusion

Preprocessing and sampling:

The similarity measure used for clustering the instances is euclidean distance.Features are in different scale.To reduce the bias towards few features over others data should be normalized. In case of categorical data a constant c is added for different values.

Normalized_inst = (inst - avg_inst) / std_vect

KDD data provided has equal distribution of normal and attack data. So, the data is sampled such that it should contain 98% of normal instances and 2% of attack instances.

Training:

- Initialize the clusters with one cluster with a centroid at first instance
- For every cluster find d distance from every cluster

- If the minimum d < W then associate the data instance with the cluster otherwise new cluster is defined with the data instance as cluster centroid

Labelling:

- The first N % highly populated clusters are labelled as normal and other clusters are attack.

Testing:

- For each instance find the closest cluster
- Assign the label of closest cluster

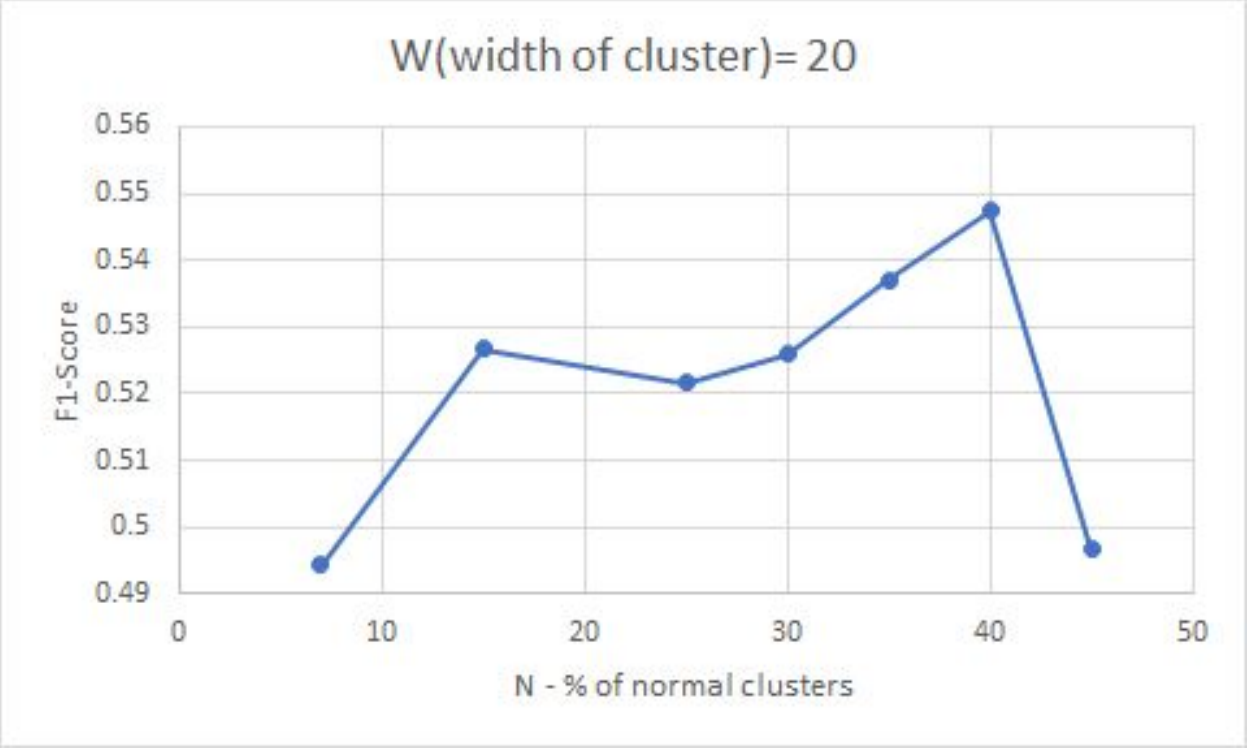
Hyperparameters :

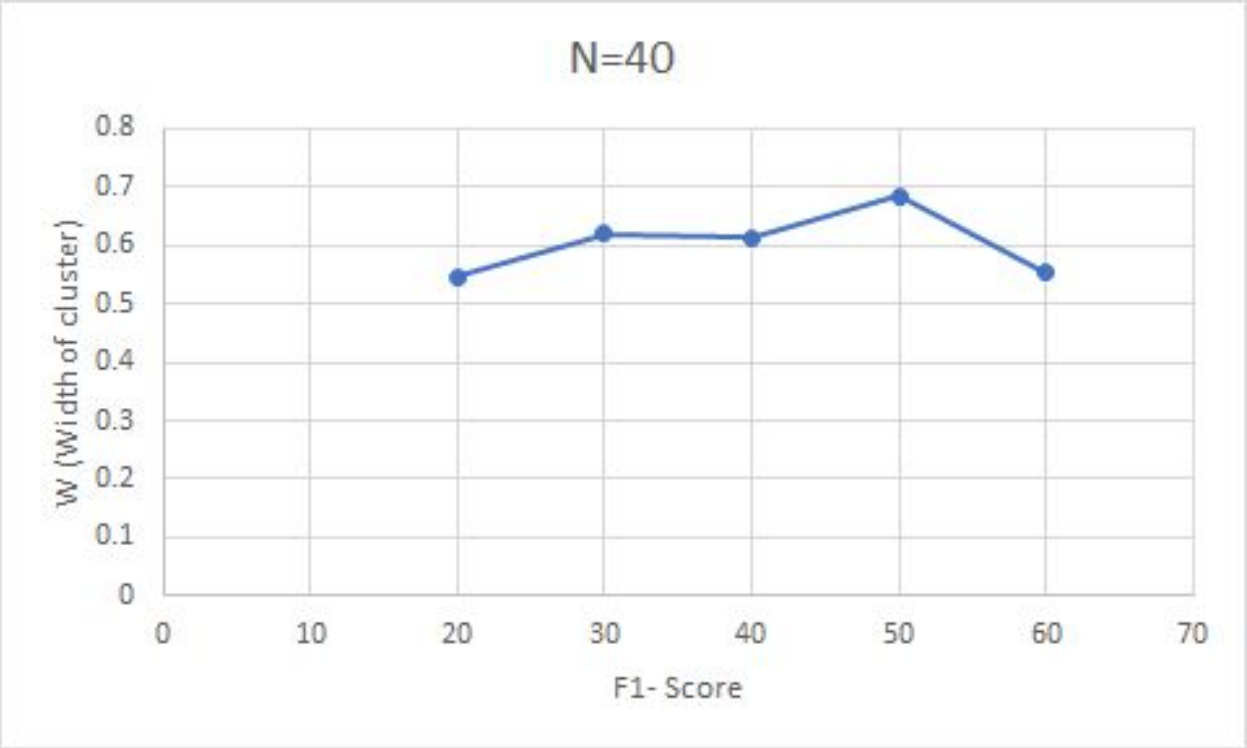
W : width assigned to each cluster

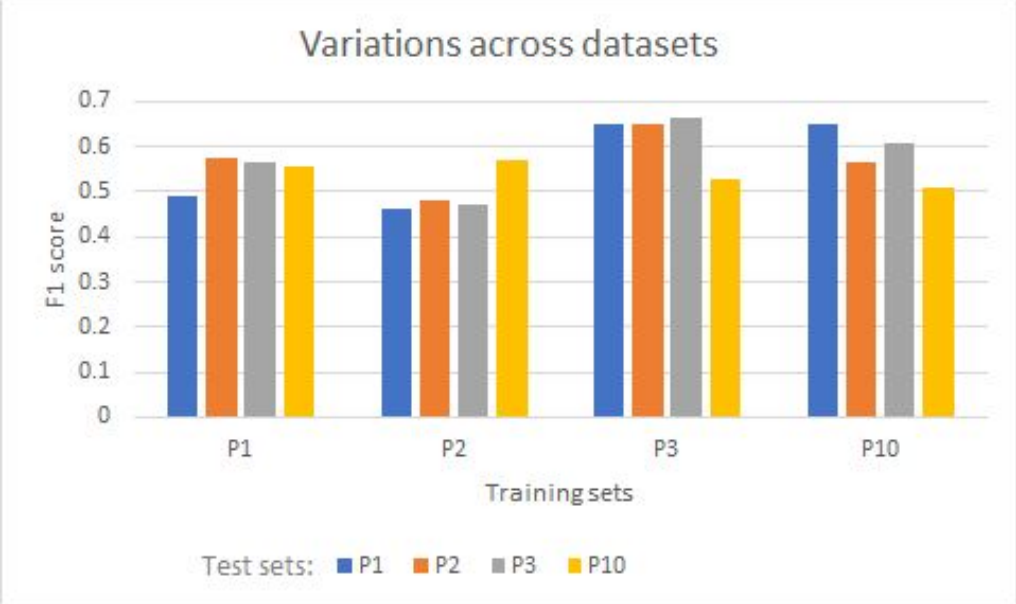
N : As per the algorithms assumption the first N highly populated clusters are assigned as normal

Parameter estimation:

Hyperparameters are estimated by training the data on 10% KDD(best representation of entire KDD) by fixing W find the best N and by fixing N find the best W. Here, from our observations N = 40 and W = 50 gave best results but W = 50 is greater than the number of features so, the next best between 30 and 40 is chosen W for further experiments.

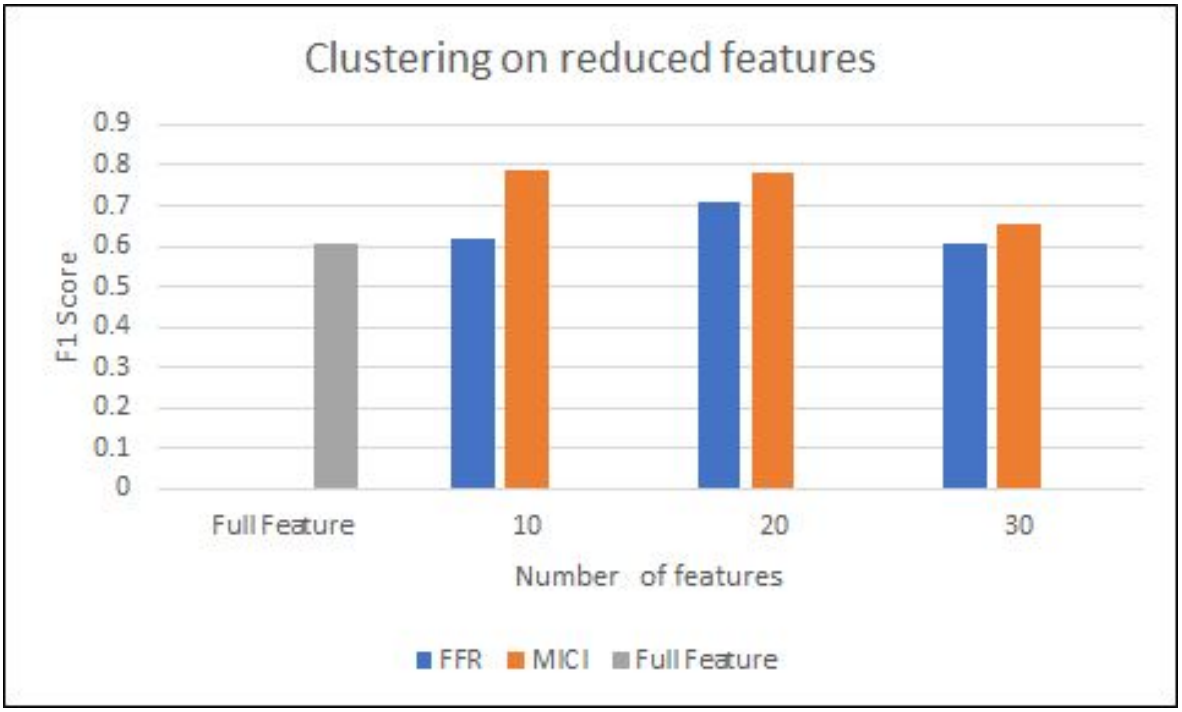






Variations across dataset: In the above figure with W= 35 and N = 40 trained and tested on different variations of data. These cross validation sets are collected from the entire KDD cup data. P4,P5,P6,P7,P8,P9 cross validations consists of only attack data. The results are dependent of the training set used. These training sets are variated by types of attack they have.Training and testing on same datasets gave equal f1-score as

training and testing on different datasets. This shows the algorithm is not overfitting and detecting even the new types of attacks which are not found in training.



Clustering on reduced features: Above graph shows that clustering worked better on the reduced feature sets. This is due to the removal of features which are correlated across the classes.