Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection

In this paper author is evaluating performance of various data mining algorithms such as SVM (Support Vector Machine), Random Forest and Extreme Machine Learning (ELM) using NSL KDD intrusion detection dataset. Details of all this algorithms can be read from paper. Now-a-days all network activities are under threat of attacks and this attacks can be of various types such DOS (denial of service) or DDOS (distributed denial of service) or making server to crash by sending huge amount of requests.

To avoid such attacks IDS systems has developed which process each incoming request to detect such attacks and if request is coming from genuine users then only it will forward to server for processing, if request contains attack signatures then IDS will drop that request and log such request data into dataset for future detection purpose.

To detect such attacks IDS will be prior train with all possible attacks signatures coming from malicious user’s request and then generate a training model. Upon receiving new request IDS will apply that request on that train model to predict it class whether request belongs to normal class or attack class. To train such models and prediction various data mining classification or prediction algorithms will be used.

In this paper author is evaluating performance of SVM, random forest and elm algorithms, from this algorithms ELM performance is best compare to SVM and random forest.

**Extension Concept**

**In this algorithms author has not applied any features selection algorithm which can further help algorithm to improve prediction performance in the form of ACCURACY, Precision and Recall. So in extension i am applying PCA (Principal Component Analysis) feature selection concept on ELM algorithm to further improve it accuracy.**

To conduct experiment author has used NSL KDD Dataset and below is some example records of that dataset which contains request signatures. I have also used same dataset and this dataset is available inside ‘dataset’ folder.

Dataset example

**duration,protocol\_type,service,flag,src\_bytes,dst\_bytes,land,wrong\_fragment,urgent,hot,num\_failed\_logins,logged\_in,num\_compromised,root\_shell,su\_attempted,num\_root,num\_file\_creations,num\_shells,num\_access\_files,num\_outbound\_cmds,is\_host\_login,is\_guest\_login,count,srv\_count,serror\_rate,srv\_serror\_rate,rerror\_rate,srv\_rerror\_rate,same\_srv\_rate,diff\_srv\_rate,srv\_diff\_host\_rate,dst\_host\_count,dst\_host\_srv\_count,dst\_host\_same\_srv\_rate,dst\_host\_diff\_srv\_rate,dst\_host\_same\_src\_port\_rate,dst\_host\_srv\_diff\_host\_rate,dst\_host\_serror\_rate,dst\_host\_srv\_serror\_rate,dst\_host\_rerror\_rate,dst\_host\_srv\_rerror\_rate,label**

All above comma separated names in bold format are the names of request signature

0,tcp,ftp\_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0,0,0,0,1,0,0,150,25,0.17,0.03,0.17,0,0,0,0.05,0,normal

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1,1,0,0,0.05,0.06,0,255,9,0.04,0.05,0,0,1,1,0,0,Neptune

Above two records are the signature values and last value contains class label such as normal request signature or attack signature. In second record ‘Neptune’ is a name of attack. Similarly in dataset you can find nearly 30 different names of attacks.

In above dataset records we can see some values are in string format such as tcp, ftp\_data and these values are not important for prediction and these values will be remove out by applying PREPROCESSING Concept. All attack names will not be identified by algorithm if it’s given in string format so we need to assign numeric value for each attack. All this will be done in PREPROCESS steps and then new file will be generated called ‘clean.txt’ which will use to generate training model.

In below line i am assigning numeric id to each attack

"normal":0,"neptune":1,"warezclient":2,"ipsweep":3,"portsweep":4,"teardrop":5,"nmap":6,"satan":7,"smurf":8,"pod":9,"back":10,"guess\_passwd":11,"ftp\_write":12,"multihop":13,"rootkit":14,"buffer\_overflow":15,"imap":16,"warezmaster":17,"phf":18,"land":19,"loadmodule":20,"spy":21,"perl":22,"saint":23,"mscan":24,"apache2":25,"snmpgetattack":26,"processtable":27,"httptunnel":28,"ps":29,"snmpguess":30,"mailbomb":31,"named":32,"sendmail":33,"xterm":34,"worm":35,"xlock":36,"xsnoop":37,"sqlattack":38,"udpstorm":39

In above lines we can see normal is having id 0 and Neptune 1 and goes on for all attacks.

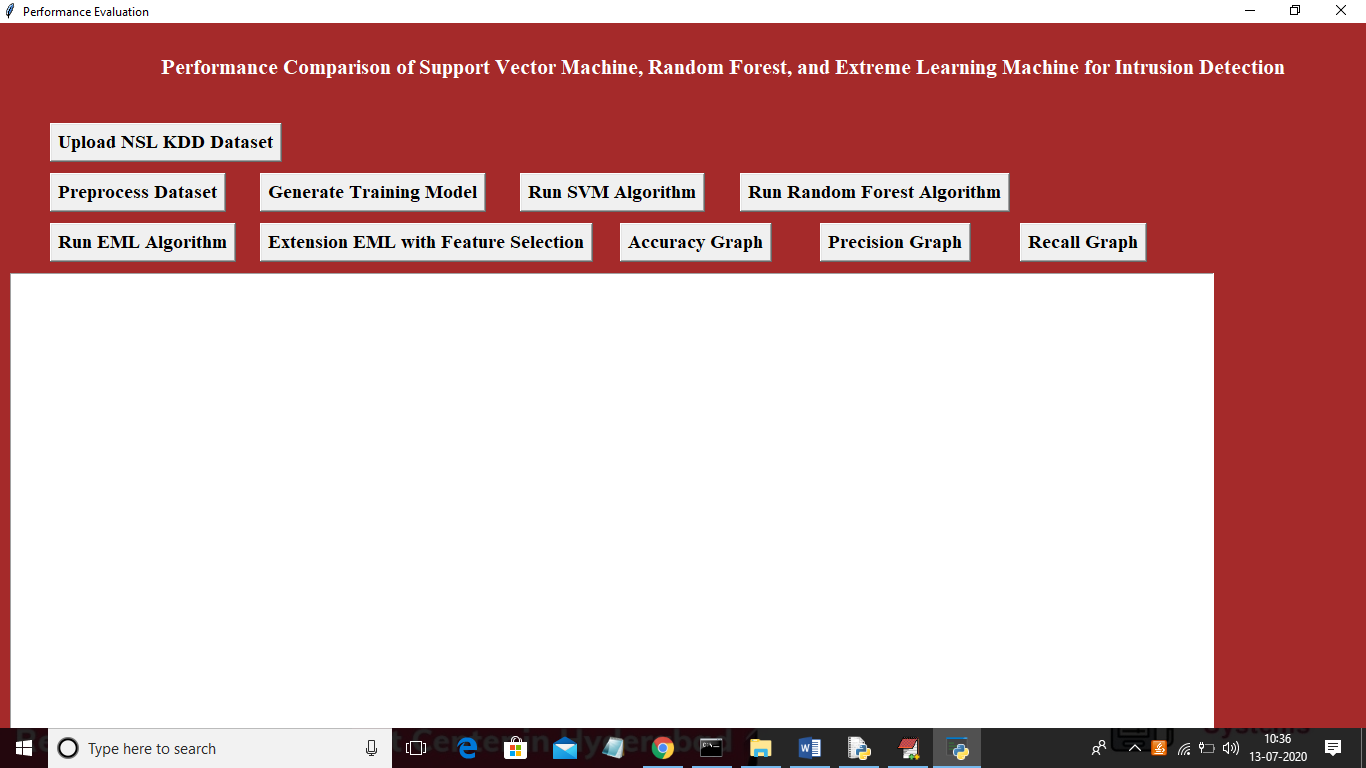
Before running code execute below two commands

pip install elm

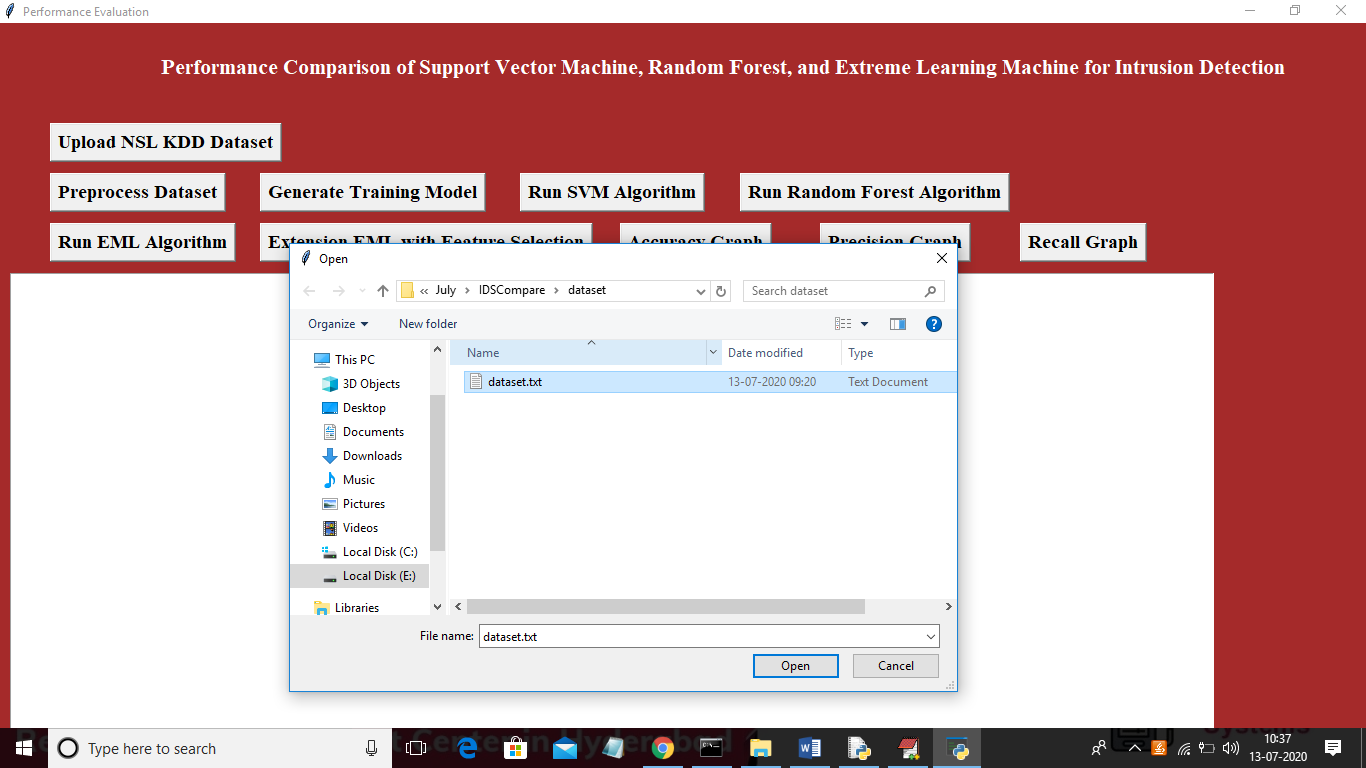
pip install sklearn-extensions

Screen shots

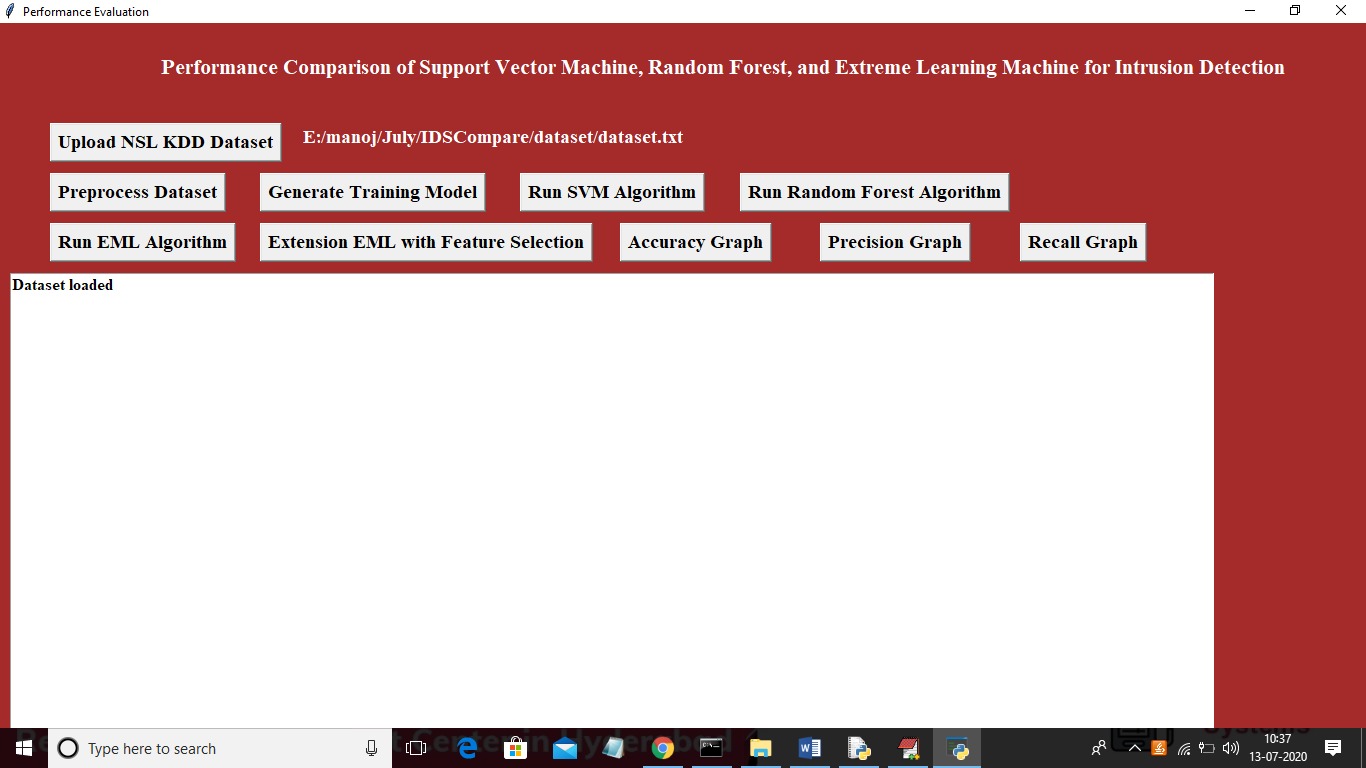
Double click on ‘run.bat’ file to get below screen



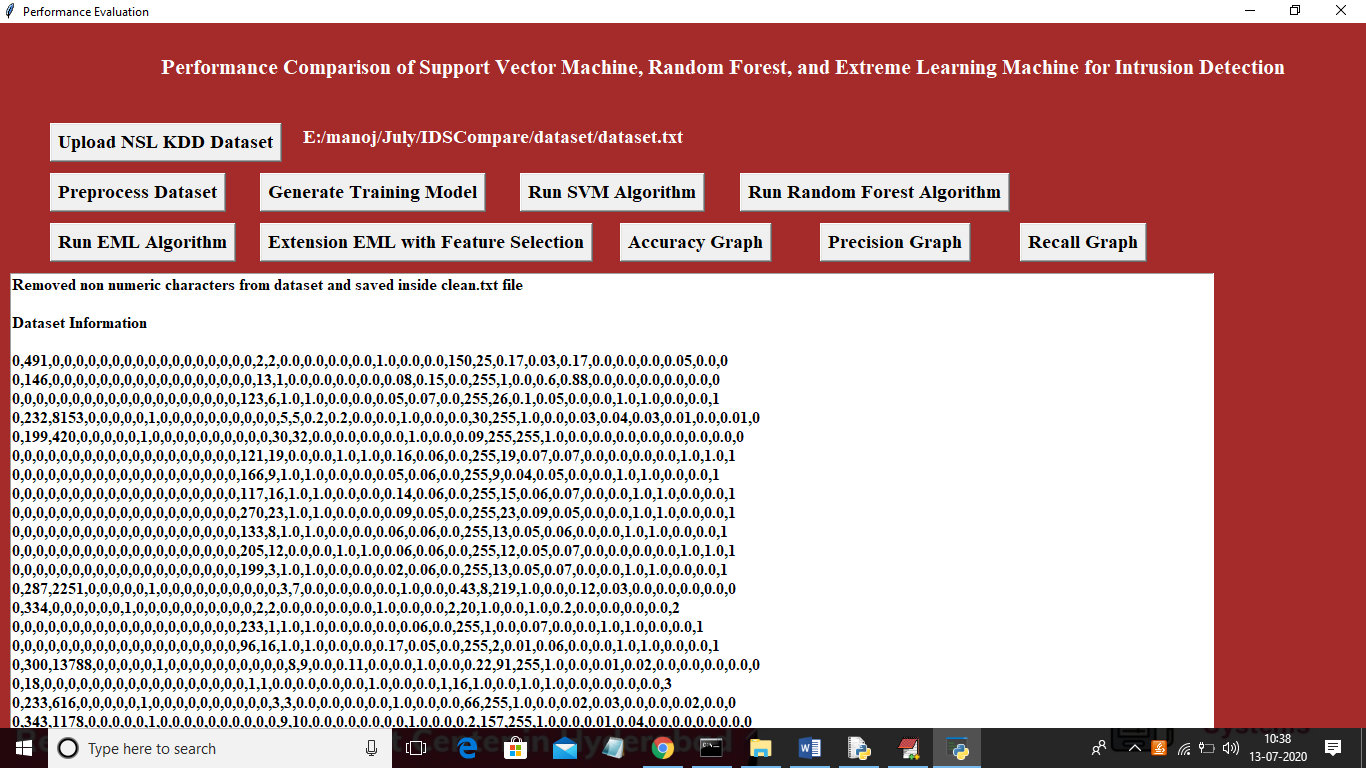
Click on ‘Upload NSL KDD Dataset’ button to upload dataset



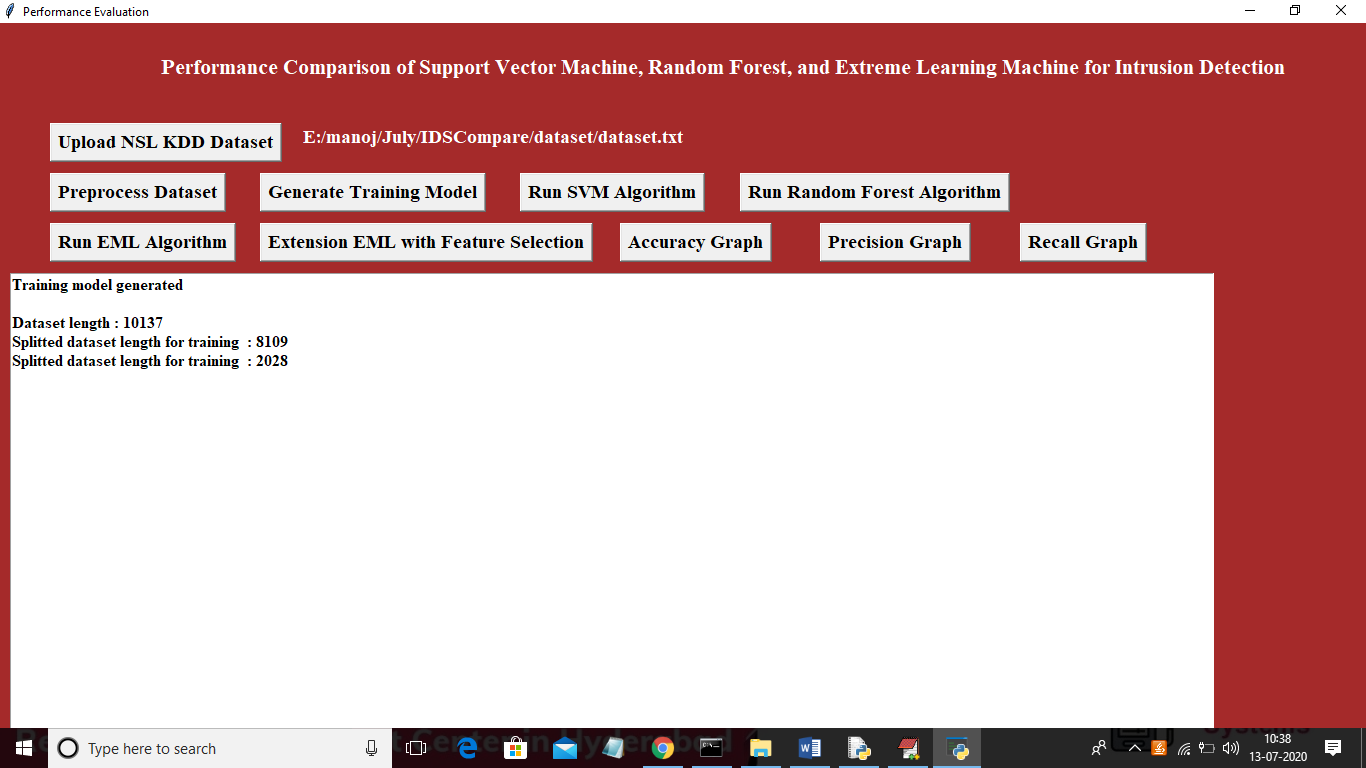
After uploading will get below screen



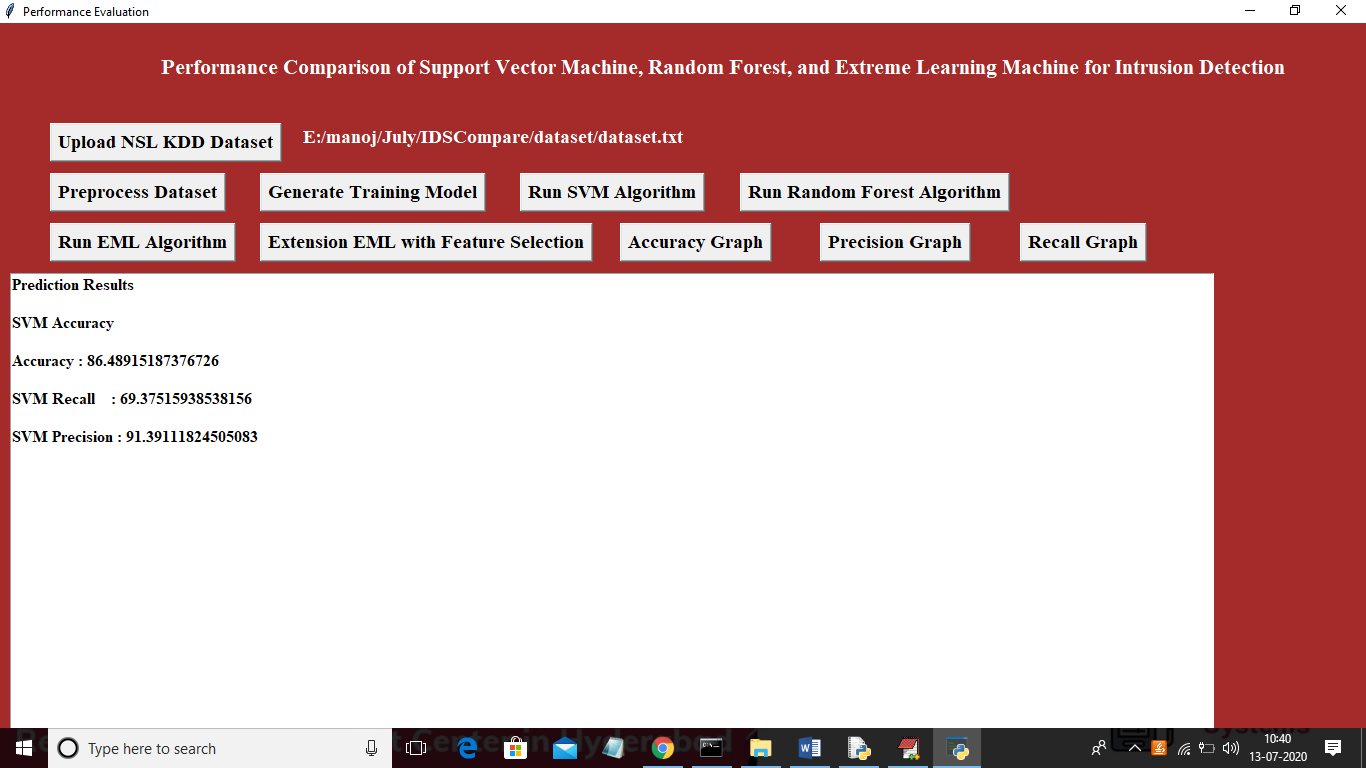
Now click on ‘Preprocess Dataset’ to apply PREPROCESSING technique and this will generate ‘clean.txt’ file which will used by application for further processing. Note: this option may take much time for pre processing and if need then only u can run this option otherwise after uploading dataset continue direct running from ‘Generate Training Model’. After pre process will get below screen



In above screen we can see using pre process we remove all characters from dataset and all attack names are replaced with their ids. Now click on ‘Generate Training Model’ to train model.

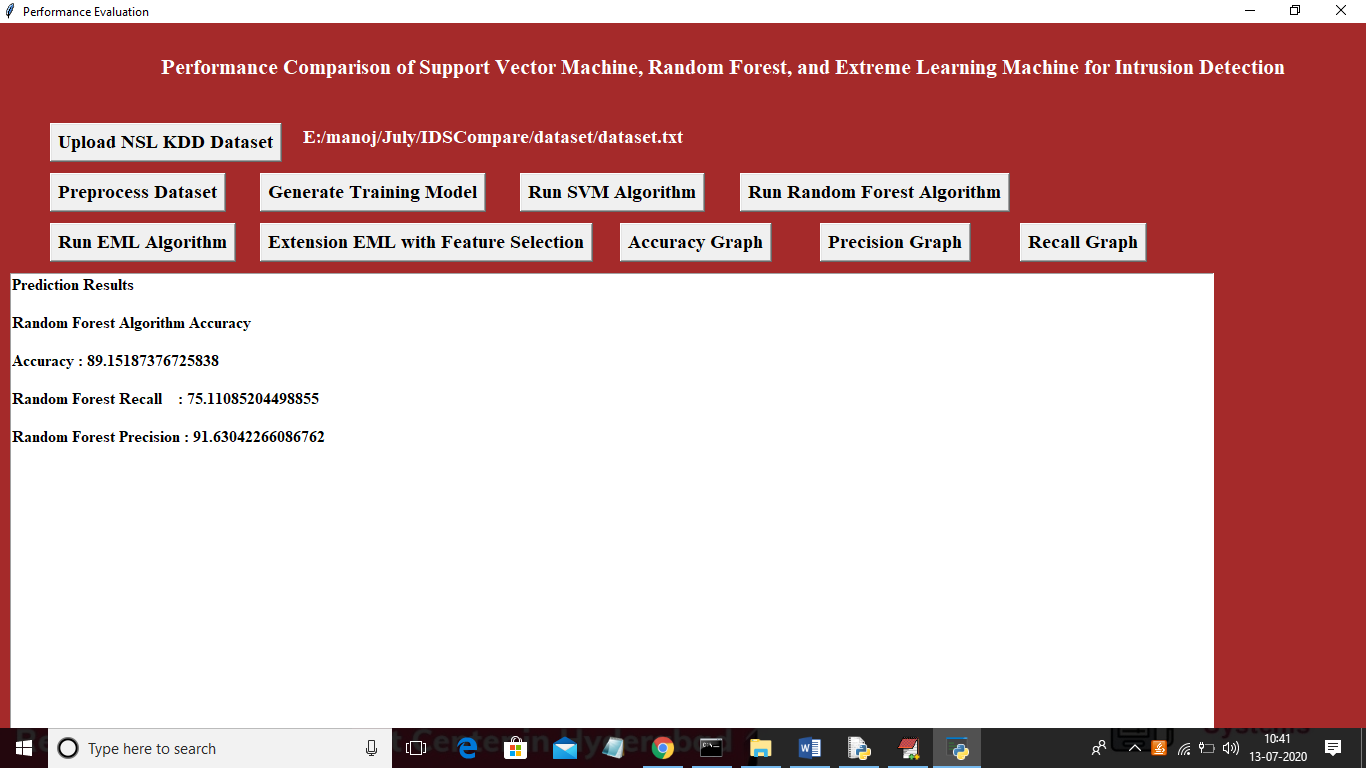


In above screen we can see training model generated and dataset contains total 10137 records and application split that dataset and using 8109 records for training and 2028 for testing. Now click on ‘Run SVM Algorithm’ to perform prediction with SVM. In paper from complete dataset author has used 20% dataset for test and 80% dataset for train. Application will choose 20% random records from dataset as test records. Due to this random selection accuracy may vary. See below screen form SVM accuracy

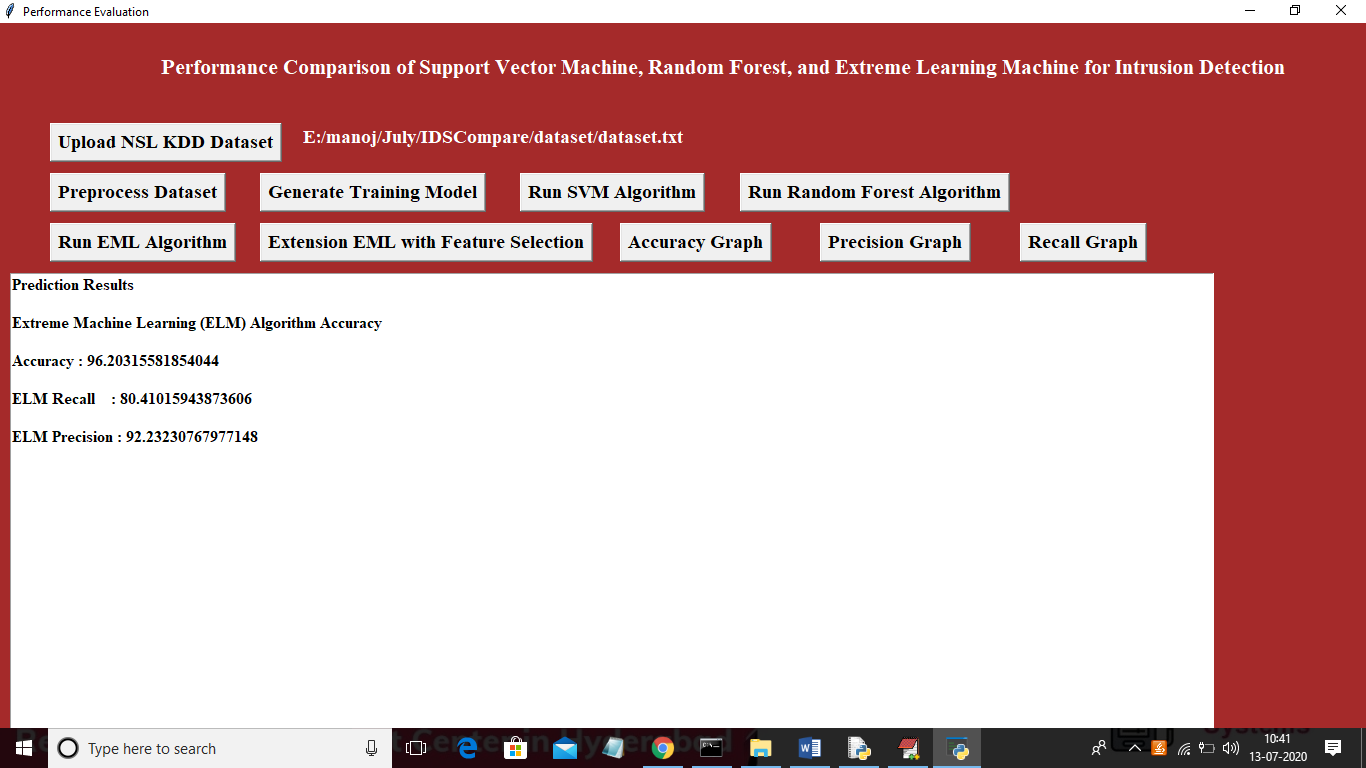


In above screen we can see SVM accuracy is 86% and we can see precision and recall values also.

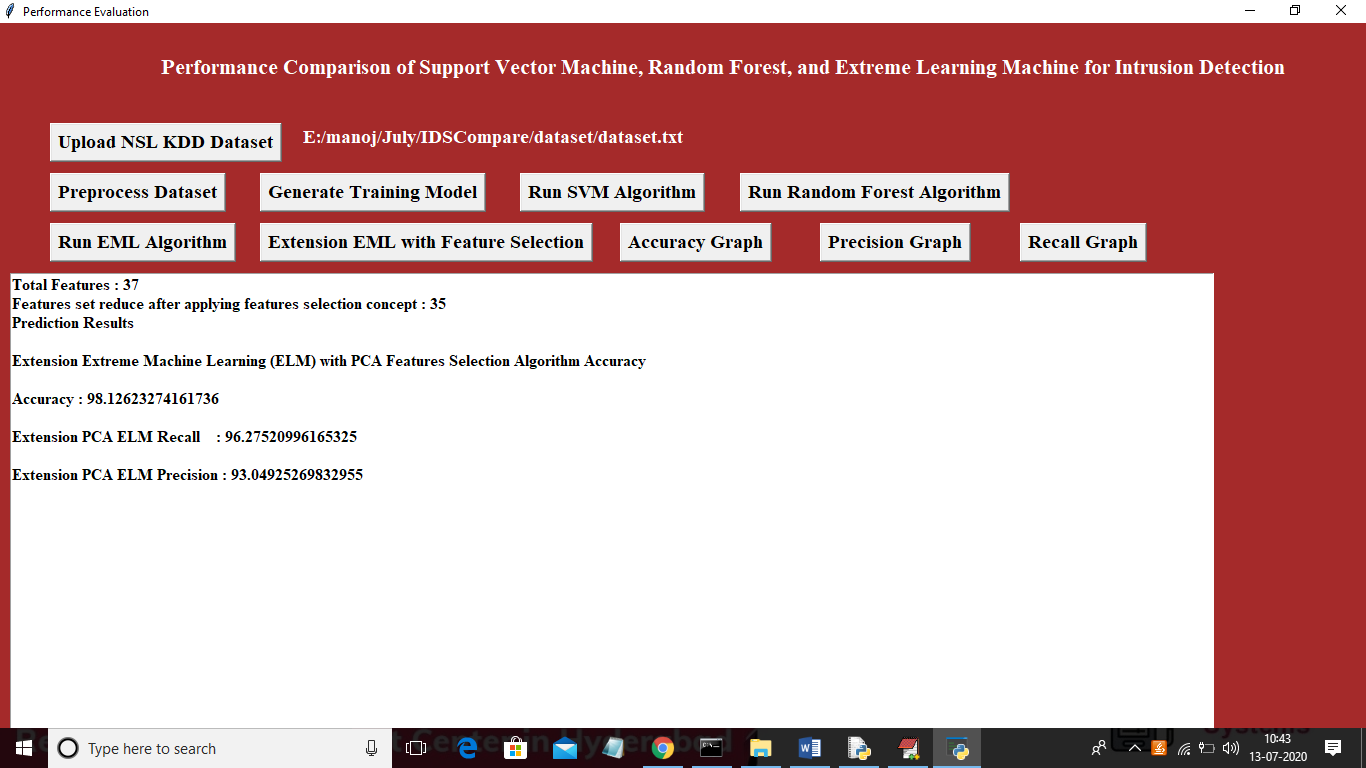
Now run ‘Random Forest’ to get below screen



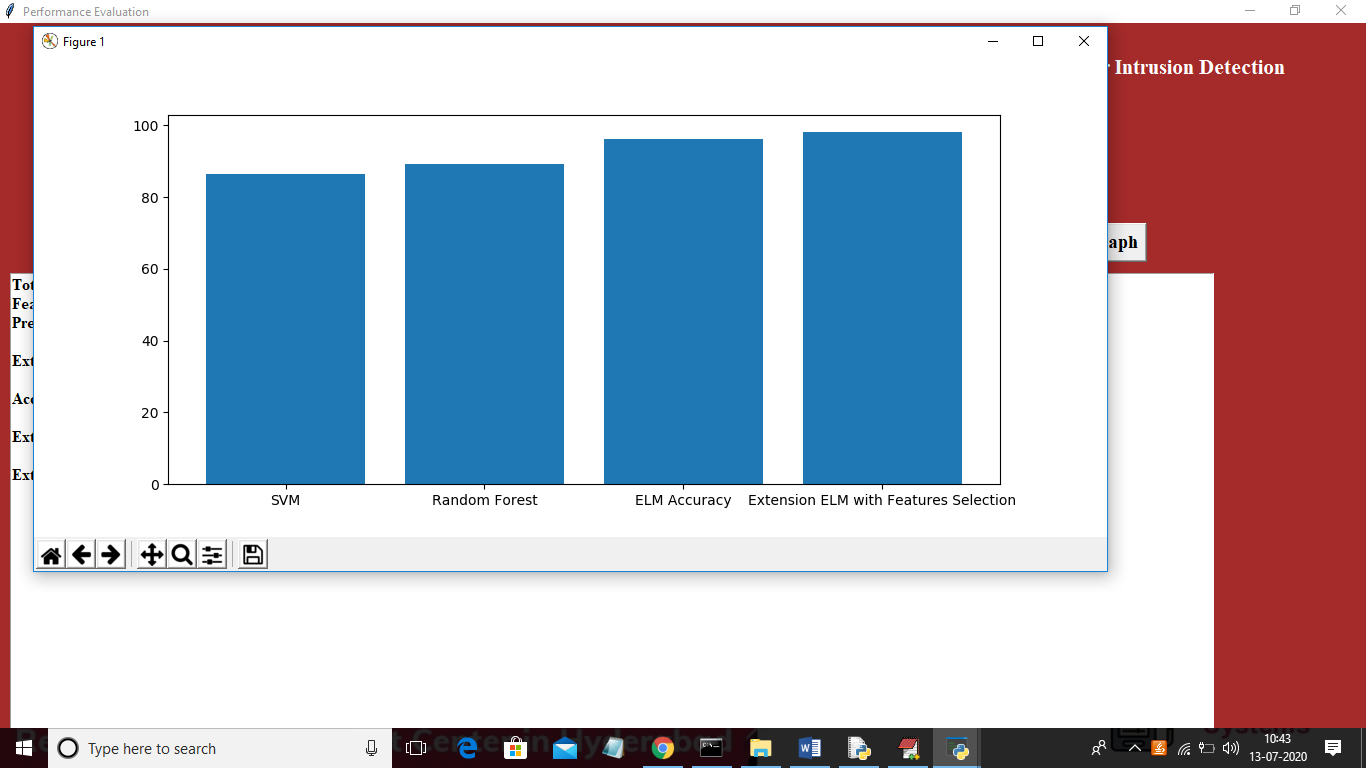
In above screen random forest accuracy is 89%, now run EML algorithm



In above screen ‘EML accuracy is 96% greater than SVM and random forest. Now run Extension EML with Feature Selection



In above screen Extension EML accuracy is 98% greater than SVM, random forest and normal EML. Now click on ‘Accuracy Graph’ button to get below graph



In above graph x-axis represents algorithm name and y-axis represents accuracy value. From all 4 algorithms we can see Extension EML with features selection and normal EML is performing better. Now click on ‘Precision Graph’ button to get below graph



Now click on ‘Recall Graph’ button to get below graph

