MACHINE LEARNING FOR NETWORK INTRUSION DETECTION

Project Report
March 2016

Project Summary

Project Duration:
4 weeks

Project Workload:
40 hour/man

Team Workers in Group 16:
040110078 Nurefşan Sertbaş
040090508 Betül Kantepe

Project Leader:
Nurefşan Sertbaş
sertbasn@itu.edu.tr

CONTENTS

List of Figures

List of Tables

1. Introduction to Spark MLLib

Apache Spark is an engine which has capability of processing large scale data. MLLib is a library that contains different machine learning algorithms is built on Spark. In this Project we will be using Spark MLLib tool in order to make implementation of decision tress. Spark has a huge power over such data in terms of efficiency when compare it with Weka or other machine learning tools.

1.1 Setting up environment for Spark MLLib

In project we prefer to use Scala as a programming language. For this aim we have done related installations for using Spark MLLib with the help of IntelliJ IDEA 14.1. Detailed information for setup steps are given in 'Software Documentation Report'.

1.2 Testing the Environment with Sample Dataset

We have tested the environment by using given heart disease dataset. For this aim, we have done following steps respectively.

- Dataset is given in csv format so that we have used Matlab functions to read the file and we have transform data to libSVMformat in order to use it with Spark MLLib functions [4].

- Libsvm has a format such as

  Label index1:value1 index2:value2 index3:value3...

  For transforming libsvm format we have used a Library for Support Vector Machines (LIBSVM) [4].

➢ Also we examine the extracted features for given data as following

	Feature Name	*Feature Explanation*	*Range*
1	#3 (age)	age in years	
2	#4 (sex)	1 = male; 0 = female	0-1
3	#9 (cp)	chest pain type -- Value 1: typical angina -- Value 2: atypical angina -- Value 3: non-anginal pain -- Value 4: asymptomatic	
4	#10 (trestbps)	resting blood pressure (in mm Hg on admission to the hospital)	
5	#12 (chol)	serum cholesterol in mg/dl	
6	#16 (fbs)	fasting blood sugar > 120 mg/dl (1 = true; 0 = false)	0-1
7	#19 (restecg)	resting electrocardiographic results -- Value 0: normal -- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV) -- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria	0-2
8	#32 (thalach)	maximum heart rate achieved	
9	#38 (exang)	exercise induced angina (1 = yes; 0 = no)	0-1
10	#40 (oldpeak)	ST depression induced by exercise relative to rest	
11	#41 (slope)	the slope of the peak exercise ST segment -- Value 1: upsloping -- Value 2: flat -- Value 3: downsloping	
12	#44 (ca)	number of major vessels (0-3) colored by flourosopy	
13	#51 (thal)	3 = normal; 6 = fixed defect; 7 = reversable defect	

Table 1: Extracted features for heart disease dataset

➢ After preprocessing data we have gave it to Spark decision tree by using Scala programming language. We have set *maxDept* parameter as 5 and *maxBins* parameter as 32.

➢ We have compiled our code and run.

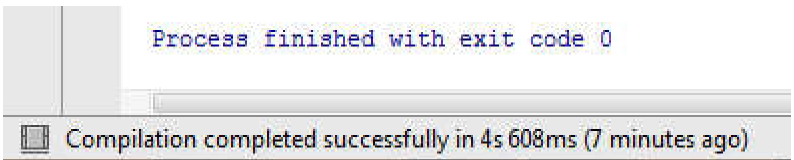

Figure 1: Compilation termination

➢ Also, we have printed the built model as can be seen in Figure 2.

```
Test Error = 0.3821656050955414
Learned classification tree model:
DecisionTreeModel classifier of depth 5 with 29 nodes
  If (feature 8 <= 0.0)
   If (feature 0 <= 59.0)
    If (feature 6 <= -9.0)
     Predict: 3.0
    Else (feature 6 > -9.0)
     If (feature 9 <= 1.5)
      If (feature 4 <= 392.0)
       Predict: 0.0
      Else (feature 4 > 392.0)
       Predict: 0.0
     Else (feature 9 > 1.5)
      If (feature 3 <= 108.0)
       Predict: 0.0
      Else (feature 3 > 108.0)
       Predict: 1.0
   Else (feature 0 > 59.0)
    Predict: 1.0
  Else (feature 8 > 0.0)
   If (feature 7 <= 143.0)
    If (feature 9 <= 1.5)
     If (feature 7 <= 130.0)
      If (feature 4 <= 268.0)
       Predict: 2.0
      Else (feature 4 > 268.0)
       Predict: 0.0
     Else (feature 7 > 130.0)
      Predict: 0.0
    Else (feature 9 > 1.5)
     If (feature 4 <= 237.0)
      If (feature 9 <= 2.0)
       Predict: 2.0
      Else (feature 9 > 2.0)
       Predict: 1.0
     Else (feature 4 > 237.0)
      If (feature 7 <= 124.0)
       Predict: 3.0
      Else (feature 7 > 124.0)
       Predict: 4.0
   Else (feature 7 > 143.0)
    If (feature 7 <= 153.0)
     Predict: 1.0
    Else (feature 7 > 153.0)
     Predict: 0.0
```

Figure 2: Built model for heart disease dataset

- We have measured test error as 38% of the model that built by 29 nodes. Accuracy rate is not sufficiently good but it may derived from the small number of training data. We will analyze effect of different numbers of test and training sets in future sections.

- Also, we have tested the model with the train data in order to be assure we have built model properly. In this case we were expecting 0 testing error because test data is already found in training set it means that this is not the first time our tree encounters with the data. As we expect we have got 0 testing error that means that our tree will perform properly its operation and there is no extraordinary situation in our environment.

2. Machine Learning for Network Intrusion Detection

2.1 Introduction to KDD99 Network Security Dataset

In this project, we aimed to develop an attack detector in order to adjust bad connections from the normal ones. This detection will work over a sequence of TCP packets and each of the packets are well-defined in terms of source-destination and other connection parameters. This information will be used into feature extraction.

For this aim, we have got the KDD99 Network Security (10 percent) data from *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.* Our dataset contains some data about TCP connection. Each line of data has a class which describe its characteristics. The dataset contains 23 classes as follows:

back,buffer_overflow,ftp_write,guess_passwd,imap,ipsweep,land,loadmodule,multihop,neptune,nmap,normal,perl,phf,pod,portsweep,rootkit,satan,smurf,spy,teardrop,warezclient,warezmaster

As mentioned in [2] we can categorized these attacks with four main group as:

<u>1: DOS (denial of service attacks)</u>

Attacker aims to make the system cannot response its normal services. One of the most common dos attack is done by creating huge volume of data traffic or huge number of connection requests to network so that the network cannot give response[3].

Teardrop, pod, land, back, neptune, smurf.

<u>2: R2L (Root to Local Attack)</u>

Attacker does not have an account to access the machine but she/he tries to find vulnerable point for the system by sending packets.

ftp_write, phf, guess_passwd, imap, warezclient, warezmaster, spy, multihop

<u>3: U2R (User to Root Attack)</u>

In these type of attacks, attacker has an account as a normal user and she/he tries to gain root privileges for the system.

Perl, loadmodule, rootkit, buffer overflow

4: Probing

Attacker gather some information about target system in order to initiate an attack.

ipsweep, satan, nmap, portsweep

2.2 Introduction to Feature Extraction Process

As we mentioned above, we will try to distinguish these 23 classes from each other. For this aim, we will build a model from the training data then we will test the model with the new data. In order to build a model extracted features will be used as a training data.

These features can be divided into three categories as,

a) Basic features of TCP connections

b) Content features within a connection suggested by domain knowledge

c) Traffic features computed using a two-second time window

The details of the categories and their features are given below as follows [5]:

Category-a features: Basic features

They can be easily extracted from the packet headers.

	feature name	*description*	*type*
1	duration	length (number of seconds) of the connection	continuous
2	protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
3	service	network service on the destination, e.g., http, telnet, etc.	discrete
4	src_bytes	number of data bytes from source to destination	continuous
5	dst_bytes	number of data bytes from destination to source	continuous
6	flag	normal or error status of the connection	discrete
7	land	1 if connection is from/to the same host/port; 0 otherwise	discrete
8	wrong_fragment	number of ``wrong'' fragments	continuous
9	urgent	number of urgent packets	continuous

Table 2: Basic features for KDD99 dataset

Category-b features: Content Features

They represents distinguishable behavior in data packets. U2L and R2L attacks are embedded into data packets and they don’t have any pattern [1]. Also, they make many connections in a short time so that there may some cases such as increasing number of failed logins. As a result, these features are important to classify them properly.

	feature name	*description*	*type*
10	hot	number of ``hot'' indicators	continuous
11	num_failed_logins	number of failed login attempts	continuous
12	logged_in	1 if successfully logged in; 0 otherwise	discrete
13	num_compromised	number of ``compromised'' conditions	continuous
14	root_shell	1 if root shell is obtained; 0 otherwise	discrete
15	su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
16	num_root	number of ``root'' accesses	continuous
17	num_file_creations	number of file creation operations	continuous
18	num_shells	number of shell prompts	continuous
19	num_access_files	number of operations on access control files	continuous
20	num_outbound_cmds	number of outbound commands in an ftp session	continuous
21	is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
22	is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete

Table 3: Content features for KDD99 dataset

Category-c features: Traffic features

These features are aims to extract properties for 2 second window.

	feature name	*description*	*type*
23	count	number of connections to the same host as the current connection in the past two seconds	continuous
		Note: The following features refer to these same-host connections.	
24	serror_rate	% of connections that have ``SYN'' errors	continuous
25	rerror_rate	% of connections that have ``REJ'' errors	continuous
26	same_srv_rate	% of connections to the same service	continuous
27	diff_srv_rate	% of connections to different services	continuous
28	srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
		Note: The following features refer to these same-service connections.	
29	srv_serror_rate	% of connections that have ``SYN'' errors	continuous
30	srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
31	srv_diff_host_rate	% of connections to different hosts	continuous

Table 4: Traffic features for KDD99 dataset

As can be seen in above tables we have several different features extracted for 23 labeled data.

2.3 Preparing Datasets

2.3.1 Transforming Dataset to Suitable Format for Spark

There is a critical point that libsvm gives support only numerical data. It means that our 2nd, 3rd, 4th and 42nd columns are recognizable for Spark MLLib so we need to find a solution for this problem. In order to solve the problem, we have changed non numerical data (string) to numerical by using simple enumeration. The detailed implementation is given in below tables.

For this enumeration we have read main dataset and extract all possible values for related columns. For instance in column 2 we have only 3 possible values so that we have gave appropriate numbers related with its value. The values and their numeric provisions are given in table 5, 6 and 7.

Variables - allvalues_at_coloumn2

allvalues_at_coloumn2

3x2 cell

	1	2	3
1	'icmp'	1	
2	'tcp'	2	
3	'udp'	3	
4			
5			

Variables - allvalues_at_coloumn4

allvalues_at_coloumn4

11x2 cell

	1	2	3
1	'OTH'	1	
2	'REJ'	2	
3	'RSTO'	3	
4	'RSTOS0'	4	
5	'RSTR'	5	
6	'S0'	6	
7	'S1'	7	
8	'S2'	8	
9	'S3'	9	
10	'SF'	10	
11	'SH'	11	
12			

Table 5: Possible values for column 2 and 4.

allvalues_at_coloumn3

66x2 cell

	1	2
1	'IRC'	1
2	'X11'	2
3	'Z39_50'	3
4	'auth'	4
5	'bgp'	5
6	'courier'	6
7	'csnet_ns'	7
8	'ctf'	8
9	'daytime'	9
10	'discard'	10
11	'domain'	11
12	'domain_u'	12
13	'echo'	13
14	'eco_i'	14
15	'ecr_i'	15
16	'efs'	16
17	'exec'	17
18	'finger'	18
19	'ftp'	19
20	'ftp_data'	20
21	'gopher'	21
22	'hostnames'	22
23	'http'	23
24	'http_443'	24
25	'imap4'	25
26	'iso_tsap'	26
27	'klogin'	27
28	'kshell'	28
29	'ldap'	29
30	'link'	30
31	'login'	31
32	'mtp'	32
33	'name'	33
34	'netbios_dg...	34
35	'netbios_ns'	35
36	'netbios_ssn'	36
37	'netstat'	37
38	'nnsp'	38
39	'nntp'	39
40	'ntp_u'	40
41	'other'	41
42	'pm_dump'	42
43	'pop_2'	43
44	'pop_3'	44
45	'printer'	45
46	'private'	46
47	'red_i'	47
48	'remote_job'	48
49	'rje'	49
50	'shell'	50
51	'smtp'	51
52	'sql_net'	52
53	'ssh'	53
54	'sunrpc'	54
55	'supdup'	55
56	'systat'	56
57	'telnet'	57
58	'tftp_u'	58
59	'tim_i'	59
60	'time'	60
61	'urh_i'	61
62	'urp_i'	62
63	'uucp'	63
64	'uucp_path'	64
65	'vmnet'	65
66	'whois'	66

Table 6: Possible values for column3.

At 42nd column, we have labels for each line such as ‘smurf’, ‘neptune’, … Because of these values are not numeric we have to replace the label names with numeric values as given in Table 7.

Representation in decision tree	Class Label (D23 and U23)
1	back
2	buffer_overflow
3	ftp_write
4	guess_passwd
5	imap
6	ipsweep
7	land
8	loadmodule
9	multihop
10	neptune
11	nmap
12	normal
13	perl
14	phf
15	pod
16	portsweep
17	rootkit
18	satan
19	smurf
20	spy
21	teardrop
22	warezclient
23	warezmaster

Representation in decision tree	Class Label (D5 and U5)
1	dos
2	probe
3	U2r
4	R2l
5	normal

Representation in decision tree	Class Label (D2 and U2)
1	normal
2	attack

Table 7: Possible values for column 42.

After this procedure, our data is ready to partite into different datasets.

2.3.2 Test-Train Datasets Partition

We have partite our given dataset into two as test (validation) and training. During this partition, we have read each line from the file with its label and write it to file named with the related labels' name.

Content of the file for each class is given in Table 8.

Class	Number of Samples	Class	Number of Samples
back	2203	**perl**	3
buffer_overflow	30	**phf**	4
ftp_write	8	**pod**	264
guess_passwd	53	**portsweep**	1040
imap	12	**rootkit**	10
ipsweep	1247	**satan**	1589
land	21	**smurf**	280790
loadmodule	9	**spy**	2
multihop	7	**teardrop**	979
neptune	107201	**warezclient**	1020
nmap	231	**warezmaster**	20
normal	97278		
			TOTAL: 494021

Table 8: Number of samples in the dataset grouped by class

Suppose we read below line from the file

```
0,tcp,http,SF,…………………………………………………..9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,normal.
```

We wrote it to 'normal.txt' file. We have done this process for each of the 23 classes. Then, we take first 90% is for training set and remaining for the test set.

2.3.3 D23-D5-D2 Datasets Generation

- D23: 23 classes (22 attacks and normal)

D23 dataset contains 23 classes as it already is so we do not need to do any transformation. Number of test and training samples are given in Table 9.

	Class	Number of Train Samples (90 %)	Number of Test Samples (10%)
1	back	1982	221
2	buffer_overflow	27	3
3	ftp_write	7	1
4	guess_passwd	48	5
5	imap	11	1
6	ipsweep	1122	125
7	land	19	2
8	loadmodule	8	1
9	multihop	6	1
10	neptune	96481	10720
11	nmap	208	23
12	normal	87551	9727
13	perl	3	0
14	phf	4	0
15	pod	238	26
16	portsweep	936	104
17	rootkit	9	1
18	satan	1430	159
19	smurf	252711	28079
20	spy	2	0
21	teardrop	881	98
22	warezclient	918	102
23	warezmaster	18	2
	TOTAL	444620	49401

Table 9: Number of test and training samples for D23 dataset

Note:

We don't know which classes are occurred in test and train sets. There may be a case that the class is found at test data but not in the train data so that detector encounter with unknown class. It may have important role on accuracy rate. This is not included in our scope, it is one of the main problem of active learning. The reverse of the case is also true. There may be a case that class is found in train set but not in test set.

- D5 : 5 classes (4 attacks: probe, dos, u2r, url(r2l), and normal)

In order to build such a dataset we need to change some labels and group them more general as given in Table 10. For example we have renamed lines with ipsweep, Satan, Nmap and Portsweep labels as 'probe'.

Class		Train Samples	Test Samples
Dos	teardrop	352312	39146
	Pod		
	Land		
	Back		
	Neptune		
	Smurf		
Probe	ipsweep	3696	411
	Satan		
	Nmap		
	Portsweep		
U2r	Perl	47	5
	Loadmodule		
	Rootkit		
	buffer overflow		
R2l	ftp_write	1013	112
	phf		
	guess_passwd		
	Imap		
	Warezclient		
	Warezmaster		
	Spy		
	Multihop		
Normal	normal	87551	9727
	TOTAL	**444620**	**49401**

Table 10: Number of test and training samples for D5 dataset

- D2: 2 classes (attack and normal)

We have done the similar labeling for constructing D2 dataset as given below table.

Class		Train	Test
Attack	Dos	357069	39674
	Probe		
	U2r		
	R2l		
Normal	Normal	87551	9727
	TOTAL	**444620**	**49401**

Table 11: Number of test and training samples for D2 dataset

2.3.4 U23-U5-U2 Datasets Generation

In generation of the U type unbalanced datasets we have used D type datasets (U2 derived from D2). Also in order to make them unbalanced we have applied simple rule over them as

```
If instance is normal
        Keep it
Else
        Keep the first one pass following 9 samples
```

By implementing the basic rule we have created unbalanced datasets as given below.

- U2: 2 classes (attack with 10% probability and normal)

Class		Train	Test
Attack	Dos	35707	3968
	Probe		
	U2r		
	R2l		
Normal	Normal	87551	9727
	TOTAL		

Table 12: Number of test and training samples for U2 dataset

➢ U5 : 5 classes (4 attacks: probe, dos, u2r, url(r2l), and normal)

Class		**Train**	**Test**
Dos	teardrop	35231	3914
	Pod		
	Land		
	Back		
	Neptune		
	Smurf		
Probe	ipsweep	370	41
	Satan		
	Nmap		
	Portsweep		
U2r	Perl	4	1
	Loadmodule		
	Rootkit		
	buffer overflow		
R2l	ftp_write	102	12
	phf		
	guess_passwd		
	Imap		
	Warezclient		
	Warezmaster		
	Spy		
	Multihop		
Normal	normal	87551	9727
	TOTAL		

Table 13: Number of test and training samples for U5 dataset

➢ U23: 23 classes (22 attacks and normal)

	Class	**Train**	**Test**
1	back	199	23
2	buffer_overflow	2	0
3	ftp_write	1	0
4	guess_passwd	5	0
5	imap	1	1
6	ipsweep	112	12
7	land	2	0
8	loadmodule	1	0
9	multihop	0	0
10	neptune	9649	1072
11	nmap	20	3
12	normal	87551	9727
13	perl	1	0
14	phf	0	0
15	pod	24	2
16	portsweep	93	11
17	rootkit	1	0
18	satan	143	16
19	smurf	25271	2808
20	spy	1	0
21	teardrop	88	9
22	warezclient	92	11
23	warezmaster	1	0
	TOTAL	**123258**	**49401**

Table 14: Number of test and training samples for D23 dataset

Now, we have 6 different datasets as D2, D5, D23, U2, U5, U23 and we are ready to test and train the datasets. For this purpose, we will give them as an input to Spark MLLib decision tree with some parameters so that we will create a model based on training samples. Then, we will be able to test the model with our test set. As a result we will extract the accuracy rates, testing and training times and confusion matrices for each dataset. Then, we will analyze them what does the results mean in machine learning concepts.

2.4 Spark MLLib Decision Tree Performance

As we mentioned before first we will built a model then test it. There are some parameters that affects the performance of the classifier. We will first give these parameters and their explanations then we give tree metrics for different values of the given parameters.

During the building model phase, there are two parameters exist to be changed

- **MaxDepth:** It represents maximum depth of the tree. Increasing maxDepth parameter is likely to increase accuracy rate and decrease the test error. However; it increases the complexity of the training process.

- **MaxBins:** It is the parameter that can be used in making continuous signals discrete. Increasing MaxBins parameters may result better split. But it increases complexity.

We will introduce you performance metrics for each of the datasets in below tables. After this step, we will decide which parameters should be set to which values in order to increase efficiency without increasing complexity.

It is desired to get high accuracy rate with low complexity. In this content, complexity can be measured in terms of time metrics and tree characteristics. For instance, number of nodes and depth of the tree should be small as much as possible. However; increasing number of nodes may represent the classes better.

Note that in below tables we have gave compilation time but it depends on the state of the computer mostly so that it is not too important as a metric for the performance comparison.

2.4.1 Performance Measurements

➢ U23 Dataset Performance Measurements

	COMPILATION TIME	MODEL BUILDING TIME	TESTING TIME	CLASSIFICATION TEST ERROR	MAXDEPTH PARAMETER	MAXBINS PARAMETER	NUMBER OF NODES
1	6488ms	10021ms	8,3ms	0,4673		64	23
2	4584ms	14405ms	1,8ms	0,7886	4	32	23
3	5554ms	5497ms	7ms	1,7597		16	23
4	5052ms	9151ms	12,4ms	0,3577		64	37
5	5824ms	7478ms	2,6ms	0,5841	5	32	37
6	4450ms	9651ms	5,54ms	1,5991		16	35
7	6344ms	6736ms	8,2ms	0,6206		64	41
8	5468ms	6413ms	5,9ms	0,8908	6	32	49
9	6186ms	6509ms	5,3ms	1,6721		16	49

Table 15: U23 Dataset Performance Measurements

In addition to tests that are given in above table we did some simple tests to see its effect on the same data. During these tests maxBins parameter is set to 32 and we tried to find the effect of the depth of tree parameter over U23 dataset as given in below table.

CLASSIFICATION TEST ERROR	MAXDEPTH PARAMETER	NUMBER OF NODES
8,4775	1	3
1,2194	2	7
1,37	3	13
0,7886	4	25
0,5841	5	37
0,8908	6	49

Table 16: Effect of maxDepth parameter on classifier performance

As can be seen in above table, increasing depth of the parameter from 1 to 6 decrease the test error 10 times.

Also when we set depth of the tree as 1 it is surprisingly gives high accuracy although it uses only 3 nodes in tree. We thought that there may be some error because it should give higher error so that we analyze confusion matrix and built model.

```
Learned classification tree model:
DecisionTreeModel classifier of depth 1 with 3 nodes
  If (feature 22 <= 106.0)
   Predict: 12.0
  Else (feature 22 > 106.0)
   Predict: 19.0
```

Figure 3: Built model for U23 dataset

Here it is important to notice that

Class 12 has 87551 sample in training set and 9727 in test

Class 19 has 25271 sample in training set and 2808 in test

Total U23 data contain 123258 sample in training set and 49401 samples in test set. It means that just covering these two classes it covers most of the data in the set so that it is not surprise to give high accuracy.

➢ U5 Dataset Performance Measurements

	COMPILATION TIME	MODEL BUILDING TIME	TESTING TIME	CLASSIFICATION TEST ERROR	MAXDEPTH PARAMETER	MAXBINS PARAMETER	NUMBER OF NODES
1	5664ms	7237ms	8,4ms	0,4016		64	23
2	5692ms	16876ms	11,5ms	1,4165	4	32	23
3	4745ms	6715ms	5,3ms	0,387		16	21
4	5777ms	11993ms	5,8ms	0,8324		64	29
5	6937ms	8772ms	8ms	0,6425	5	32	31
6	5886ms	9581ms	12ms	0,3797		16	29
7	5285ms	8994ms	5,3ms	0,4454		64	37
8	5882ms	8466ms	10,1ms	0,5768	6	32	35
9	4956ms	18532ms	10,5ms	0,365		16	41

Table 17: U5 Dataset Performance Measurements

➢ U2 Dataset Performance Measurements

	COMPILATION TIME	MODEL BUILDING TIME	TESTING TIME	CLASSIFICATION TEST ERROR	MAXDEPTH PARAMETER	MAXBINS PARAMETER	NUMBER OF NODES
1	5439ms	14321ms	8ms	0,4819		64	19
2	6221ms	5598ms	8ms	0,8178	4	32	19
3	5473ms	8493ms	10ms	0,3650		16	21
4	5832ms	5953ms	7ms	0,4454		64	25
5	7175ms	7122ms	5ms	0,7520	5	32	25
6	6805ms	15353ms	21ms	0,3650		16	29
7	5323ms	7969ms	6ms	0,3577		64	39
8	6673ms	6192ms	6,1ms	0,7228	6	32	29
9	3610ms	6913ms	5,8ms	0,3431		16	41

Table 18: U2 Dataset Performance Measurements

➢ D23 Dataset Performance Measurements

	COMPILATION TIME	MODEL BUILDING TIME	TESTING TIME	CLASSIFICATION TEST ERROR	MAXDEPTH PARAMETER	MAXBINS PARAMETER	NUMBER OF NODES
1	5086ms	14911ms	5,8ms	1,2003		64	19
2	6362ms	8891ms	6ms	1,4351	4	32	19
3	6803ms	7521ms	6,4ms	0,6963		16	31
4	6224ms	9547ms	11,6ms	0,9352		64	35
5	7113ms	8676ms	5,8ms	0,9230	5	32	39
6	7414ms	13233ms	5,1ms	0,6518		16	55
7	5221ms	8974ms	8ms	0,7874		64	65
8	4661ms	8706ms	8,52ms	0,8117	6	32	69
9	6525ms	9192ms	8,4ms	0,5485		16	81

Table 19: D23 Dataset Performance Measurements

➢ D5 Dataset Performance Measurements

	COMPILATION TIME	MODEL BUILDING TIME	TESTING TIME	CLASSIFICATION TEST ERROR	MAXDEPTH PARAMETER	MAXBINS PARAMETER	NUMBER OF NODES
1	5934ms	18194ms	10,7ms	1,4392		64	25
2	4902ms	18799ms	14,3ms	1,5789	4	32	25
3	5348ms	7827ms	5,1ms	0,9898		16	29
4	8043ms	8726ms	6,9ms	0,8542		64	47
5	5147ms	9640ms	9,2ms	1,1214	5	32	41
6	5803ms	9326ms	4,9ms	1,6214		16	39
7	5637s	8513ms	6,6ms	1,2752		64	61
8	7859ms	12174ms	6,3ms	1,1659	6	32	51
9	5585ms	18266ms	10,5ms	0,4716		16	75

Table 20: D5 Dataset Performance Measurements

➢ D2 Dataset Performance Measurements

	COMPILATION TIME	MODEL BUILDING TIME	TESTING TIME	CLASSIFICATION TEST ERROR	MAXDEPTH PARAMETER	MAXBINS PARAMETER	NUMBER OF NODES
1	4983ms	7620ms	6ms	0,8562		64	29
2	5126	8620ms	8ms	1,7185	4	32	35
3	4412ms	8563ms	8ms	0,4514		16	49
4	5328ms	9270ms	6,8ms	1,4837		64	39
5	5787ms	15880ms	5,1ms	1,8663	5	32	31
6	5366ms	7975ms	5,4ms	0,3785		16	55
7	5547ms	9044ms	6,5ms	0,7408		64	69
8	6260ms	8668ms	8,3ms	1,4817	6	32	47
9	5511ms	8575ms	6ms	0,1457		16	83

Table 21: D2 Dataset Performance Measurements

2.4.2 Classification results

After all of the tests we decide maxDepth parameter as 2 because its test error is acceptably small and in this case there are less number of nodes exist. MaxBins parameter is set to 16 as a result of above trials. In this case, we will analyze our datasets with these parameters.

	DATASET	MODEL BUILDING TIME	TESTING TIME	CLASSIFICATION TEST ERROR	NUMBER OF NODES
1	D23	16394ms	10,1ms	2,7266	7
2	D5	6658ms	5,2ms	1,2853	7
3	D2	6553ms	5,1ms	0,9736	7
4	U23	6008ms	36,2ms	4,4760	7
5	U5	6069ms	5,2ms	0,5257	7
6	U2	15343ms	1,4ns	0,4381	7

Table 22: Performance Measurements for best parameters

Classification results for D23 dataset

```
Learned classification tree model:
DecisionTreeModel classifier of depth 2 with 7 nodes
  If (feature 1 <= 1.0)
   If (feature 22 <= 20.0)
    Predict: 12.0
   Else (feature 22 > 20.0)
    Predict: 19.0
  Else (feature 1 > 1.0)
   If (feature 29 <= 0.03)
    Predict: 12.0
   Else (feature 29 > 0.03)
    Predict: 10.0
```

Class 10, class 12 and Class 19 are classified with a high accuracy as a result of built model

Figure 4: Built model for D23 dataset

As can be seen from above figure Class 10 (Neptune) with 96481 samples, Class 12(normal) with 87551 samples and Class 19(smurf) with 252711 samples are used in building model phase because total number of these samples are 436743 out of 444620. Because of this reason classifier learns these classes successfully.

*We cannot give confusion matrix here for D23 and U23 datasets because it is too large to represent within the screen.

Classification results for D5 dataset

```
Learned classification tree model:
DecisionTreeModel classifier of depth 2 with 7 nodes
  If (feature 22 <= 19.0)
   If (feature 12 <= 0.0)
    Predict: 5.0
   Else (feature 12 > 0.0)
    Predict: 1.0
  Else (feature 22 > 19.0)
   If (feature 5 <= 0.0)
    Predict: 1.0
   Else (feature 5 > 0.0)
    Predict: 5.0
```

Class 1(dos) and class 5(normal) are classified with a high accuracy as a result of built model

Figure 5: Built model for D5 dataset

Confusion matrix is given below

Dos	Probe	u2r	r2l	Normal	
39061.0	0.0	0.0	0.0	85.0	Dos
134.0	0.0	0.0	0.0	277.0	Probe
3.0	0.0	0.0	0.0	2.0	u2r
1.0	0.0	0.0	0.0	111.0	r2l
22.0	0.0	0.0	0.0	9705.0	Normal

Figure 6: Confusion matrix for D5 dataset

As can be seen in above confusion matrix of D5,

→39061 of 39146 samples are classified accurately as dos and 85 of them classified wrong as normal
→None of the 411 samples are classified accurately as probe and 134 of them are classified as dos and 277 of them are classified as normal
→ None of the 5 samples are classified accurately as u2r and 3 of them are classified as dos and 2 of them are classified as normal
→ None of the 112 samples are classified accurately as probe and 1 of them are classified as dos and 111 of them are classified as normal
→ 9705 of 9727 samples are classified accurately as normal and only 22 of them classified wrong as dos

Class	*Accuracy for each class*	
Dos	39061/39146	99,7%
Probe	0/411	0
u2r	0/5	0
r2l	0/112	0
Normal	9705/9727	99,77%

Table 23: Accuracy rates for each class in D5

According to above statements we can say that,

- There are huge number of dos and normal class samples in training set so that classifier can learn these two class sufficiently good.
- Probe, u2r and r2l classes cannot classified properly.
- Due to classifier has learned dos and normal classes, it thought probe, u2r and r2l as dos or probe.

Classification results for D2 dataset

```
Learned classification tree model:
DecisionTreeModel classifier of depth 2 with 7 nodes
  If (feature 5 <= 0.0)
   If (feature 22 <= 16.0)
    Predict: 1.0
   Else (feature 22 > 16.0)
    Predict: 2.0
  Else (feature 5 > 0.0)
   If (feature 9 <= 0.0)
    Predict: 1.0
   Else (feature 9 > 0.0)
    Predict: 2.0
```

Class 1(attack) and class 2(normal) are classified with a high accuracy as a result of built model

Figure 7: Built model for D2 dataset

Confusion matrix is given below

Normal	Attack	
9669.0	58.0	**Normal**
423.0	39251.0	**Attack**

Figure 8: Confusion matrix for D2 dataset

→ 9669 of the 9727 samples are classified accurately as normal and 58 of them are classified as attack
→ 39251 of 39674 samples are classified accurately as attack and 423 of them classified wrong as normal

Class	***Accuracy for each class***	
Attack	39251/ 39674	98,93%
Normal	9669/9727	99,4%

Table 24: Accuracy rates for each class in D2

Classification results for U23 dataset

```
Learned classification tree model:
DecisionTreeModel classifier of depth 2 with 7 nodes
  If (feature 22 <= 204.0)
   If (feature 38 <= 0.75)
    Predict: 12.0
   Else (feature 38 > 0.75)
    Predict: 10.0
  Else (feature 22 > 204.0)
   If (feature 1 <= 1.0)
    Predict: 19.0
   Else (feature 1 > 1.0)
    Predict: 10.0
```

Class 10, class 12 and Class 19 are classified with a high accuracy as a result of built model

Figure 9: Built model for U23 dataset

As can be seen from above figure Class 10 (neptune) with 9649 samples, Class 12(normal) with 87551 samples and Class 19(smurf) with 25271 samples are used in building model phase because total number of these samples are 122471 out of 123258. Because of this reason classifier learns these classes successfully.

- When we compare D23 and U23 dataset classification results we can say that from the table xx, unexpectedly test error is increased from 2.72 in D23 to 4,5 in U23. It is easy to understand from their confusion matrixes.

Classification results for U5 dataset

```
Learned classification tree model:
DecisionTreeModel classifier of depth 2 with 7 nodes
  If (feature 22 <= 30.0)
   If (feature 12 <= 0.0)
    Predict: 5.0
   Else (feature 12 > 0.0)
    Predict: 1.0
  Else (feature 22 > 30.0)
   If (feature 5 <= 0.0)
    Predict: 1.0
   Else (feature 5 > 0.0)
    Predict: 5.0
```

Class 1(dos) and class 5(normal) are classified with a high accuracy as a result of built model

Figure 10: Built model for U5 dataset

Confusion matrix is given below

	Dos	Probe	u2r	r2l	Normal
Dos	3903.0	0.0	0.0	0.0	11.0
Probe	12.0	0.0	0.0	0.0	29.0
u2r	0.0	0.0	0.0	0.0	1.0
r2l	0.0	0.0	0.0	0.0	12.0
Normal	7.0	0.0	0.0	0.0	972.0

Figure 11: Confusion matrix for U5 dataset

As can be seen in above confusion matrix of U5,

→3903 of 3914 samples are classified accurately as dos and 11 of them classified wrong as normal
→None of the 41 samples are classified accurately as probe and 12 of them are classified as dos and 29 of them are classified as normal
→None of the 1 samples are classified accurately as u2r and 1 of them are classified as normal
→None of the 12 samples are classified accurately as r2l and 12 of them are classified as normal
→972 of 979 samples are classified accurately as normal and 7 of them classified wrong as dos

Class	***Accuracy for each class***	
Dos	3903/3914	99,71%
Probe	0/41	0
u2r	0/1	0
r2l	0/12	0
Normal	972/979	99,28%

Table 25: Accuracy rates for each class in U5

When we compare D5 and U5 dataset classification results we can say that from the table xx, test error is decreased from in U5 as we expected.

Classification results for U2 dataset

```
Learned classification tree model:
DecisionTreeModel classifier of depth 2 with 7 nodes
  If (feature 22 <= 30.0)
   If (feature 12 <= 0.0)
    Predict: 1.0
   Else (feature 12 > 0.0)
    Predict: 2.0
  Else (feature 22 > 30.0)
   If (feature 5 <= 0.0)
    Predict: 2.0
   Else (feature 5 > 0.0)
    Predict: 1.0
```

Class 1(attack) and class 2(normal) are classified with a high accuracy as a result of built model

Figure 12: Built model for U2 dataset

Confusion matrix is given below

Normal	Attack	
9720.0	7.0	Normal
53.0	3915.0	Attack

Figure 13: Confusion matrix forU2 dataset

→ 9720 of the 9727 samples are classified accurately as normal and 7 of them are classified as attack
→ 3915 of 3968 samples are classified accurately as attack and 53 of them classified wrong as normal

Class	*Accuracy for each class*	
Attack	9720/9727	99,92%
Normal	3915/3968	98,66%

Table 26: Accuracy rates for each class in U2

- In D2 there are 58 mistakes in normal class however in this case number of mistakes only 7. Same performance increment occurs for attack class from 423 mistakes to only 53 mistakes.

2.5 Spark MLLib Decision Tree Visualization

In this part of the project, we try to visualize our built models in previous section (2.4) for each data set. Our aim is comparing testing and training performances of the decision trees by using produced trees that transformed from training models.

D23 and U23 Decision Trees

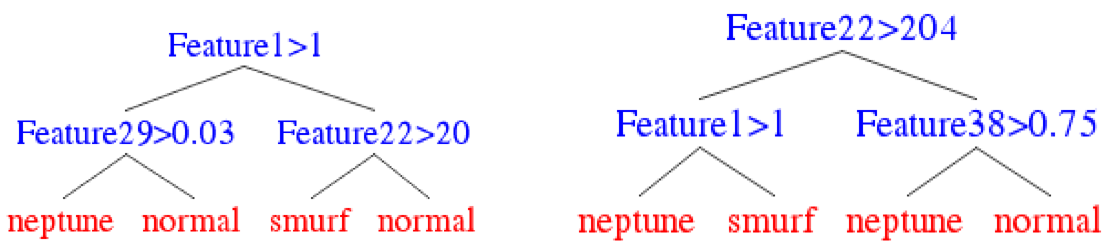

Figure 14: Decision tree visualization for D23-U23 datasets

D5 and U5 Decision Trees

It is obvious from the trees that U23 and D23 datasets approximately have the same model so that we can say that selected features like feature 5, 12 and 22 are very decisive for 5 class data.

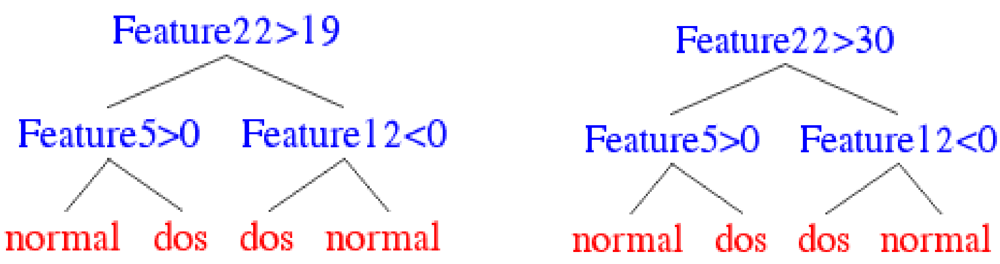

Figure 15: Decision tree visualization for D5-U5 datasets

D2 and U2 Decision Trees

As can be seen in below models, there is significant point that models are not so similar. It may derived from the contents of the training sets because feature 12 may be decisive for classes in U2 but this may not valid for D2.

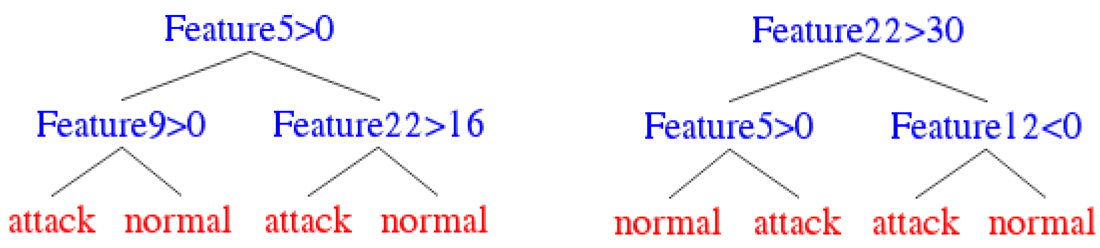

Figure 16: Decision tree visualization for D2-U2 datasets

3. References

[1] Abbass, H. A. (Ed.). (2005). Applications of Information Systems to Homeland Security and Defense. IGI Global.

[2] Dehuri, S. (Ed.). (2012). Intelligent Techniques in Recommendation Systems: Contextual Advancements and New Methods: Contextual Advancements and New Methods. IGI Global.

[3]Denial of Service Attacks (Published 1997). (n.d.). Retrieved March 04, 2016, from http://www.cert.org/information-for/denial_of_service.cfm

[4] LIBSVM -- A Library for Support Vector Machines. (n.d.). Retrieved March 09, 2016, from https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[5] (n.d.). Retrieved March 22, 2016, from http://kdd.ics.uci.edu/

[6] Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., & Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: Results from the JAM project. In DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings (Vol. 2, pp. 130-144). IEEE.