

Project Three: KDD Cup '99 Data Analysis

Andrew Bauman, Mark Miller

Introduction

Intrusion Detection Systems (IDSs) have become an important tools in preventing unauthorized access to networks by monitoring network data and classifying it as either normal or abnormal. For this project, we were tasked with working with the KDD Cup dataset from 1999 to replicate the results of an experiment from “The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 (The Fifth International Conference on Knowledge Discovery and Data Mining).” Accordingly, we have modeled the data and explored the effectiveness of varying predictive models.

Context: The Data *copied from the assignment instructions*

The following text, adapted from the 1999 paper [Cost-based Modeling and Evaluation for Data Mining with Application to Fraud and Intrusion Detection: Results from the JAM Project](#) by Salvatore J. Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K. Chan, was given to competitors to understand the task at hand and the data set with which they were confronted.

Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between “bad” connections, called intrusions or attacks, and “good” normal connections. The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. The 1999 KDD intrusion detection contest uses a version of this dataset. Lincoln Labs set up an environment to acquire

nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks.

The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes

Attacks fall into four main categories:

- **DOS:** denial-of-service, e.g. syn flood;
- **R2L:** unauthorized access from a remote machine, e.g. guessing password;
- **U2R:** unauthorized access to local superuser (root) privileges, e.g., various "buffer overflow" attacks;
- **probing:** surveillance and other probing, e.g., port scanning.

It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types not in the training data. This makes the task more realistic. Some intrusion experts believe that most novel attacks are variants of known attacks and the "signature" of known attacks can be sufficient to catch novel variants. The datasets contain a total of 24 training attack types, with an additional 14 types in the test data only. Table 1 displays the names of attacks and the category they belong to.

Attack Name	Attack Type
back	dos
buffer_overflow	u2r
ftp_write	r2l

Attack Name	Attack Type
perl	u2r
phf	r2l
pod	dos

guess_passwd	r2l	portsweep	probe
imap	r2l	rootkit	u2r
ipsweep	probe	satan	probe
land	dos	smurf	dos
loadmodule	u2r	spy	r2l
multihop	r2l	teardrop	dos
neptune	dos	warezclient	r2l
nmap	probe	warezmaster	r2l

Table 1. Attack names and types

Stolfo et al. defined higher-level features that help in distinguishing normal connections from attacks. There are several categories of derived features. The "same host" features examine only the connections in the past two seconds that have the same destination host as the current connection, and calculate statistics related to protocol behavior, service, etc. The similar "same service" features examine only the connections in the past two seconds that have the same service as the current connection. "Same host" and "same service" features are together called time-based traffic features of the connection records. Some probing attacks scan the hosts (or ports) using a much larger time interval than two seconds, for example once per minute. Therefore, connection records were also sorted by destination host, and features were constructed using a window of 100 connections to the same host instead of a time window. This yields a set of so-called host-based traffic features. Unlike most of the DOS and probing attacks, there appear to be no sequential patterns that are frequent in records of R2L and U2R attacks. This is because the DOS and probing attacks involve many connections to some host(s) in a very short period of time, but the R2L and U2R attacks are embedded in the data portions of packets, and normally involve only a single connection. Useful algorithms for mining the unstructured data portions of packets automatically are an open research question. Stolfo et al. used domain knowledge to add features that look for suspicious behavior in the data portions, such as the number of failed login attempts. These features are called "content" features. A complete listing of the set of features defined for the connection records is given Tables 2,3,4 below. The data schema of the contest dataset is summarized in Table 5.

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

Table 2. Basic features of individual TCP connections

feature name	description	type
hot	number of ``hot" indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised" conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if ``su root" command attempted; 0 otherwise	discrete
num_root	number of ``root" accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the ``hot" list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest" login; 0 otherwise	discrete

Table 3. Content features within a connection suggested by domain knowledge

feature name	description	type
count	number of connections to the same host as the current connection in the past two seconds	continuous

	<i>Note: The following features refer to these same-host connections.</i>	
serror_rate	% of connections that have ``SYN'' errors	continuous
error_rate	% of connections that have ``REJ'' errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
	<i>Note: The following features refer to these same-service connections.</i>	
srv_serror_rate	% of connections that have ``SYN'' errors	continuous
srv_error_rate	% of connections that have ``REJ'' errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Table 4. Traffic features computed using a two-second time window

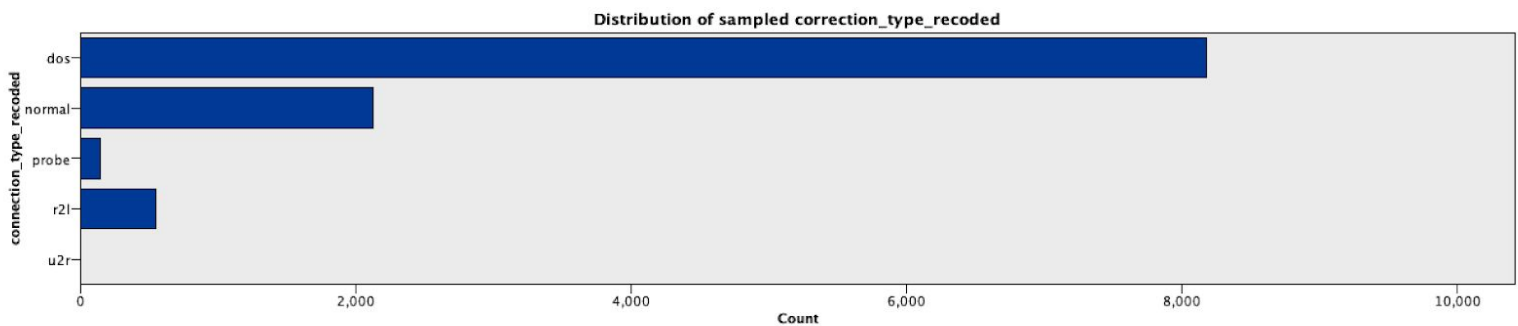
1. duration: continuous	22.is_guest_login: categorical
2. protocol_type: categorical	23.count: continuous
3. service_category: categorical	24.srv_count: continuous
4. flag : categorical	25.serror_rate: continuous
5. src_bytes: continuous	26.srv_serror_rate: continuous
6. dst_bytes: continuous	27.error_rate: continuous
7. land : categorical	28.srv_error_rate: continuous
8. wrong_fragment: continuous	29.same_srv_rate: continuous
9. urgent: continuous	30.diff_srv_rate: continuous
10.hot: continuous	31.srv_diff_host_rate: continuous
11.num_failed_logins: continuous	32.dst_host_count: continuous
12.logged_in: categorical	33.dst_host_srv_count: continuous
13.num_compromised: continuous	34.dst_host_same_srv_rate: continuous
14.root_shell: continuous	35.dst_host_diff_srv_rate: continuous
15.su_attempted: continuous	36.dst_host_same_src_port_rate: continuous
16.num_root: continuous	37.dst_host_srv_diff_host_rate: continuous
17.num_file_creations: continuous	38.dst_host_serror_rate: continuous

18.num_shells: continuous	39.dst_host_srv_error_rate: continuous
19.num_access_files: continuous	40.dst_host_error_rate: continuous
20.num_outbound_cmds:continuous	41.dst_host_srv_error_rate: continuous
21.is_host_login: categorical	42.connection_type: categorical (class attribute)
connection_type: apache2., back., buffer_overflow., guess_passwd., httpunnel., ipsweep., mailbomb., mscan., multihop., neptune., nmap., normal., pod., portsweep., processtable., rootkit., saint., satan., smurf., snmpgetattack., snmpguess., sqlattack., warezmaster., xlock. Note: normal. means a "normal" connection (no attack)	

Table 5. Data File Format

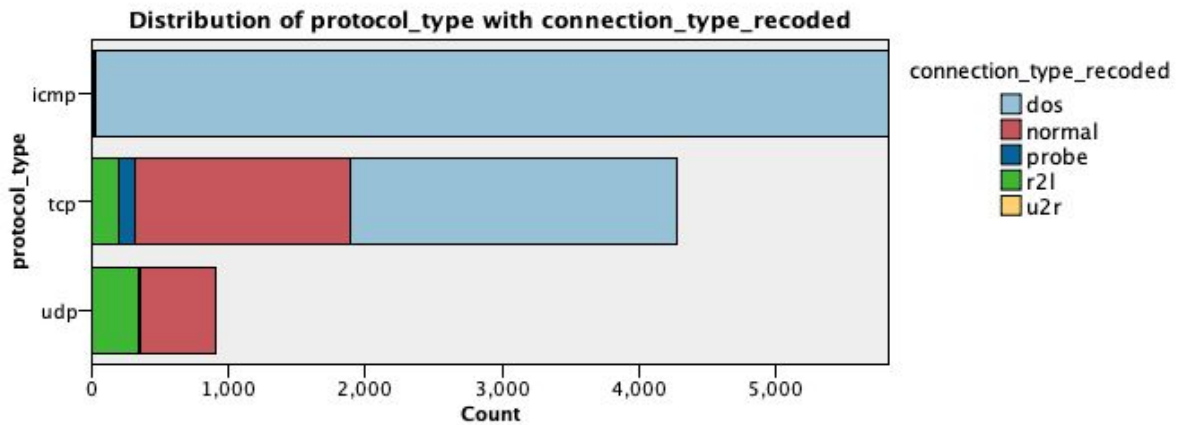
Our process

To begin, we sampled the data set, taking 11000 records, approximately 10000 for training and 1000 for testing. Following this, we recoded connection_type as connection_type_recoded in order to match the four main attack types while also maintaining the records initial data values. Here, you can see the distribution of this sampled and partitioned data:

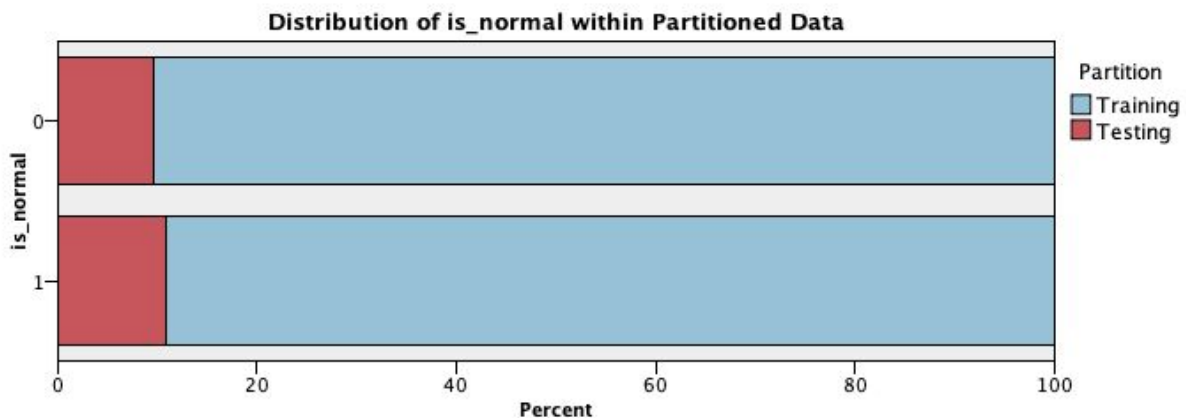


Value	%	Count
1_Training	90.09	9910
2_Testing	9.91	1090

After performing exploratory data analysis on the partitioned data, we can see the distribution of records by protocol_type with connection_type_recoded:

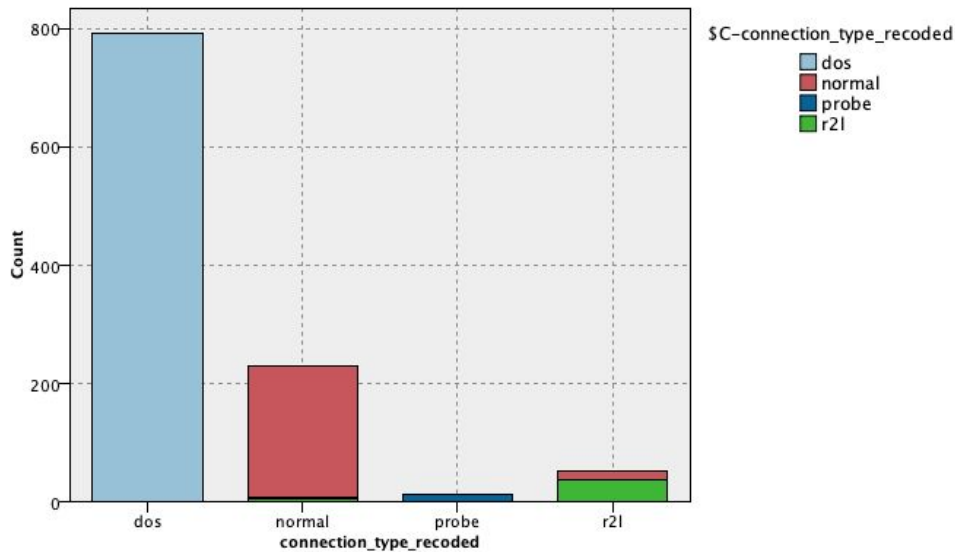


We can simplify our findings even further, showing the distribution of the partitioned data by is_normal, a derived value denoting whether or not a record describes normal or malicious network activity:



Modeling the Data







Once prepared, the data was modeled using an Auto Classify node in order to efficiently determine the best predictive model, maintaining connection_type_recoded as our target value. After its completion, we were able to achieve a 97% overall accuracy using the C5 Tree Node:



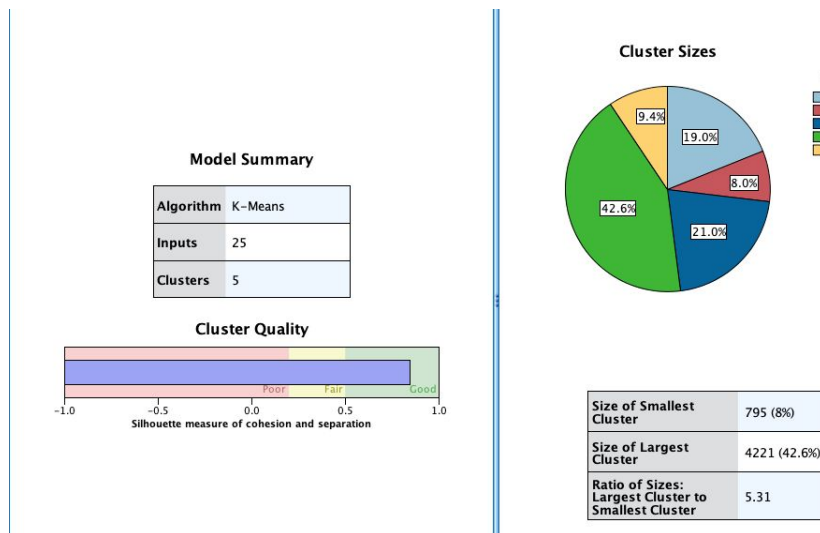
Additionally, we were able to reach a 93% accuracy using a C&R Tree to characterize the four types of attacks. It's decision rules are as follows:

```
count <= 89.500 [ Mode: normal ]
├── service_category in [ "eco_i" "ecr_i" "finger" "ftp" "ftp_data" "imap4" "iso_tsap" "other" "pop_3" "private" "smtp" "sunrpc" "telnet" ] [ Mode: normal ]
├── src_bytes <= 2581.500 [ Mode: normal ]
│   ├── src_bytes <= 54.500 [ Mode: r2l ]
│   │   ├── src_bytes <= 25 [ Mode: probe ] ⇒ probe
│   │   └── src_bytes > 25 [ Mode: r2l ] ⇒ r2l
│   └── src_bytes > 54.500 [ Mode: normal ] ⇒ normal
├── src_bytes > 2581.500 [ Mode: dos ] ⇒ dos
├── service_category in [ "auth" "domain_u" "http" "ntp_u" "time" ] [ Mode: normal ] ⇒ normal
└── count > 89.500 [ Mode: dos ] ⇒ dos
```


Following this, we recoded connection_type_recoded as a flag variable, is_normal, with the goal of running classifiers on it to create a model capable of predicting normal or malicious behavior. Again employing an Auto Classifier node, we were only able to achieve an overall 20.92% Overall Accuracy value across all three finalized models.

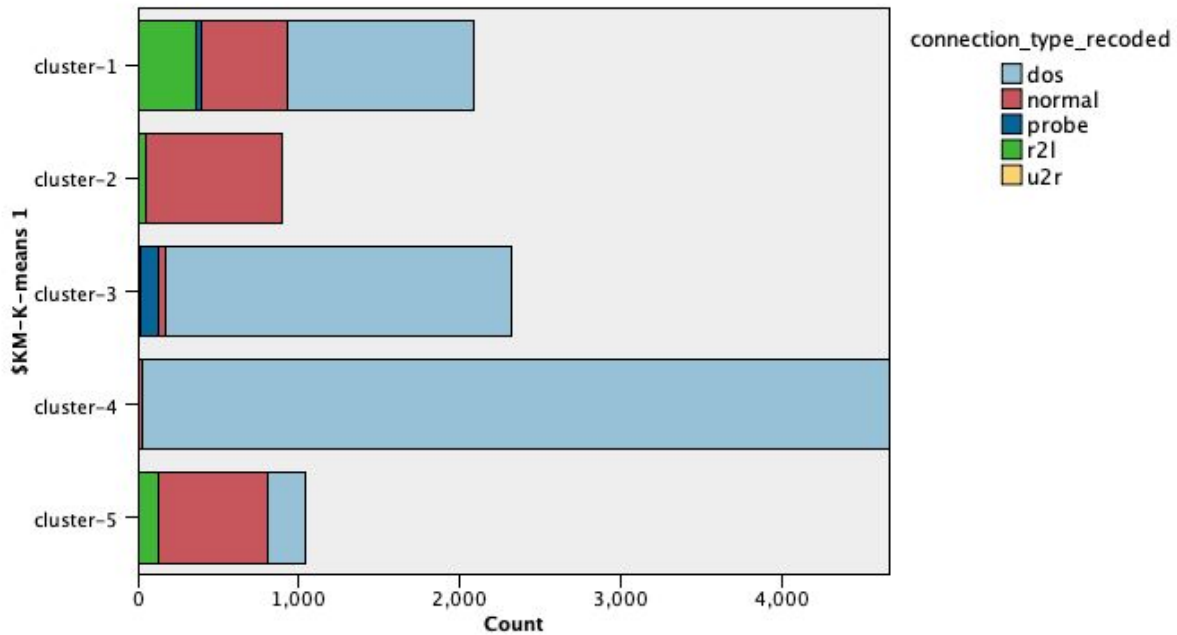
Use?	Graph	Model	Build Time (mins)	Max Profit	Max Profit Occurs in (%)	Lift (Top 30%)	Overall Accuracy (%)	No. Fields Used	Area Under Curve
<input checked="" type="checkbox"/>		Bayesian Network...	< 1	-50.0	0		20.917	25	0.0
<input checked="" type="checkbox"/>		LSVM 1	< 1	-50.0	0		20.917	25	0.0
<input checked="" type="checkbox"/>		Random Trees 1	< 1	-50.0	0		20.917	25	0.0

Finally, we investigated how helpful clustering the data through an unsupervised approach, clustering the unlabeled data records according to their common characteristics. Employing a K-Means node, we were able to achieve a Model with relatively good cluster quality with a total of 5 clusters.



While this does, at first, appear to be a feasible option, we do see that each cluster is made up of varying data and, therefore, it would not be helpful in labeling the data and especially not for predicting values of new records.

Here, we can see the distribution of the five clusters produced as a result of the K-Means node. It is apparent that, although the records have been grouped with others which have similar values, this method would not be helpful in definitively labeling the data.



In the end, although our results were not what we would have hoped they would be, we are content with the near 97% accuracy of the models to predict connection_type_recoded. If a new data set were to be run through this model, we could then derive the flag value 'is_normal' again in order to maintain this accuracy rather than trying to skip a step and predict 'is_normal' itself. Regardless, we are confident these data models would be helpful in establishing a more permanent solution to the process of Network Intrusion Detection.