Data Cleaning and Feature Engineering

JANUARY 2024

Tim Shimeall
Clarence Worrell
CERT Program

Carnegie Mellon University
Software Engineering Institute

Document Markings

Carnegie Mellon University Software Engineering Institute

DM23-2389

Agenda

- Introduction
- Data Cleaning
- Feature Engineering
- Machine Learning
- Practicum
- Conclusion

Course Objectives

Upon completion, participants will be able to:

- **State the purpose of data cleaning and feature engineering**
- **Explain basic steps in these processes**
- **Use open-source tools to accomplish these processes**
- **Adapt data cleaning and feature engineering to suit analysis needs**

I’ve queried data – now what?

Analyze it!

Avoid

- Unwanted data included in query
- Distracting groups unrelated to goal
- Duplicate data that confuses volume measurements
- Corrupted or incomplete data

Ensure

- Data is regular
- Formatted, scoped, coded to make analysis easier

Flow Records

This course uses network flow data for examples.

Network flow data is aggregated network packet headers (IP and transport-layer protocols): source & destination addresses and ports, transport protocol, total bytes, packets, TCP flags, start time, duration.

Augmented with collection information: sensor ID, router information, direction of traffic, collection attributes, application code.

Collected via export from routers or from dedicated sensor software

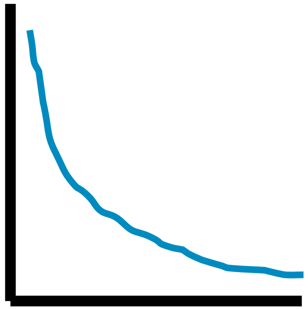
Analyzed via a variety of tool suites, although this course uses SiLK.

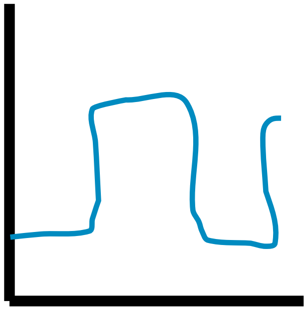
Data cleaning and Feature Engineering

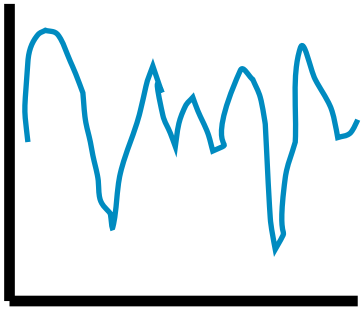
“More data beats clever algorithms, but better data beats more data.” - Peter Norvig (Google/Stanford)

More data is often easier – gather from more devices, longer timeframe

But analyzing data without cleaning and engineering may lead to misleading results.







Goal

Carnegie Mellon University Software Engineering Institute

For a given flow, we want to:

- **predict the application label,**
- **based on its flow attributes (ports, addresses, bytes, durations, etc.).**

The ongoing example

Basic scenario: To establish a baseline for normal network behavior, an analyst needs to identify the application-level protocols used in network activity. YAF does application labeling, using a set of rules, but there are issues with just using its labels.

Analyst wants more reliable results – cleaning data and engineering features

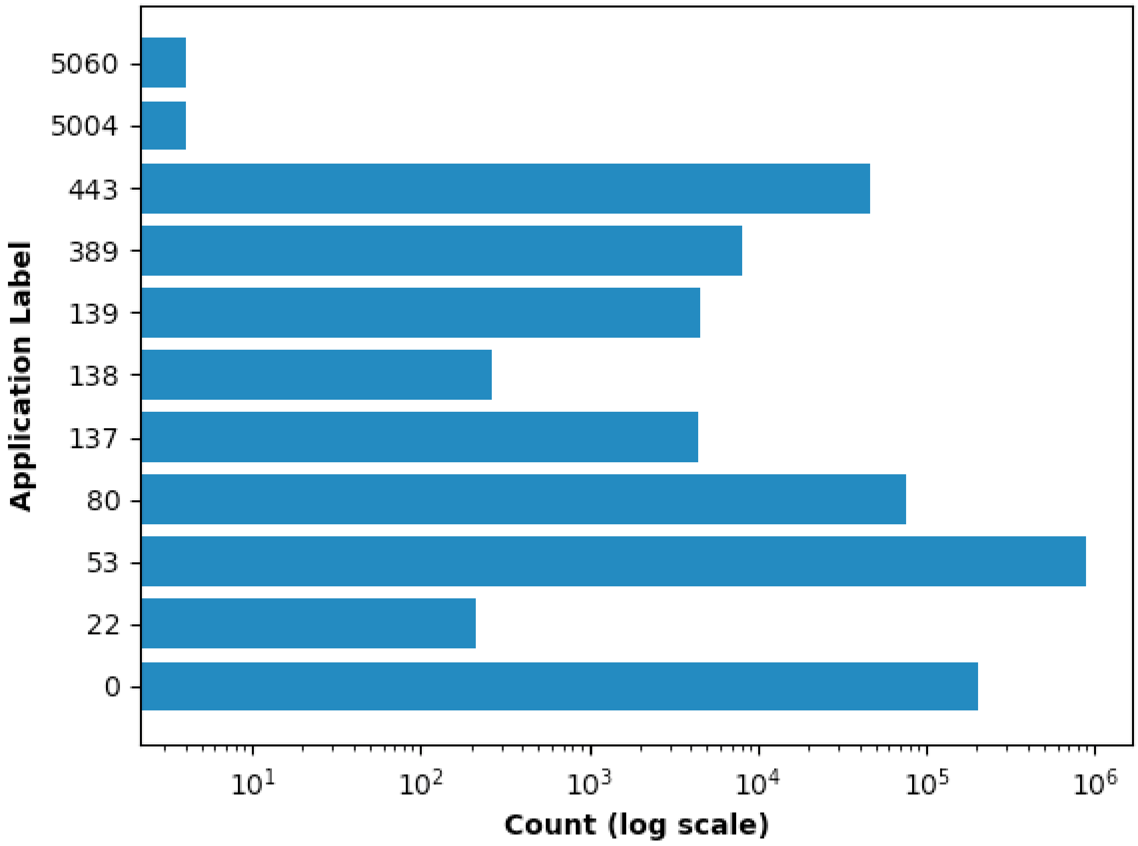
YAF – Yet Another Flowmeter – a tool that processes packets into flows

Initial data query

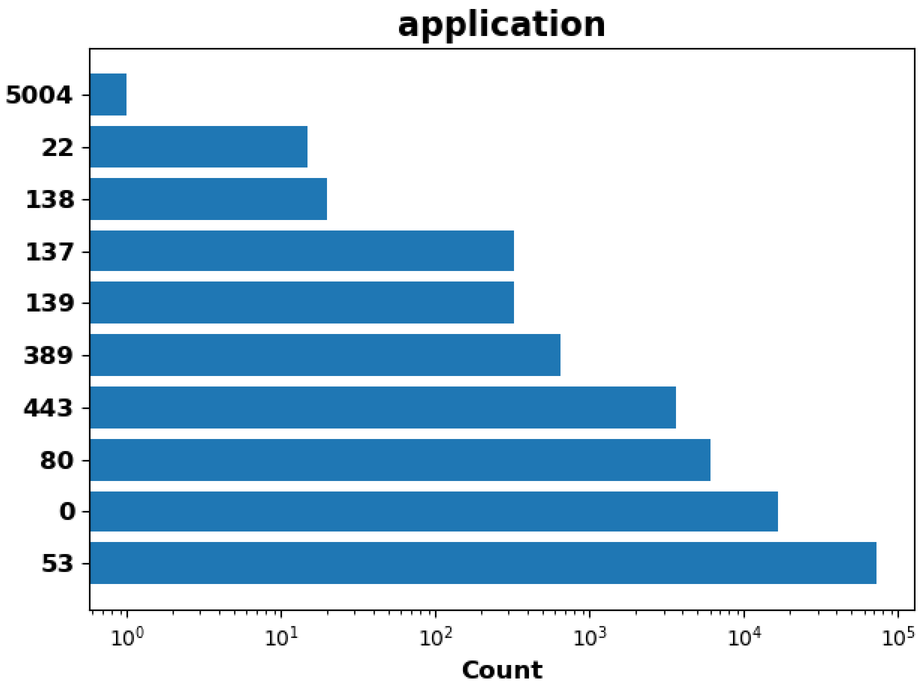
```
$  rwfilter --start=2015/06/14 --end=2015/06/18 \
--type=in,out,inweb,outweb,int2int --proto=0- --pass=stdout \
| rwstats --fields=application --values=bytes,flows --count=50
INPUT: 38227994 Records for 12 Bins and 66016107477 Total Bytes
OUTPUT: Top 50 Bins by Bytes
appli|               Bytes|   Records|    %Bytes|   cumul_%|
    0|         60325769483|  36590255| 91.380379| 91.380379|
   80|          4827534293|    152080|  7.312661| 98.693041|
  443|           345711966|    121072|  0.523678| 99.216719|
   53|           193345037|   1263273|  0.292876| 99.509594|
  139|           155090685|      9434|  0.234929| 99.744523|
  137|            61377837|     71560|  0.092974| 99.837497|
 5004|            50989842|         8|  0.077238| 99.914735|
  389|            38211066|     15525|  0.057881| 99.972617|
   22|            16859154|       474|  0.025538| 99.998155|
  138|             1186822|      4111|  0.001798| 99.999953|
 5060|               21492|         6|  0.000033| 99.999985|
   69|                9800|       196|  0.000015|100.000000|
```

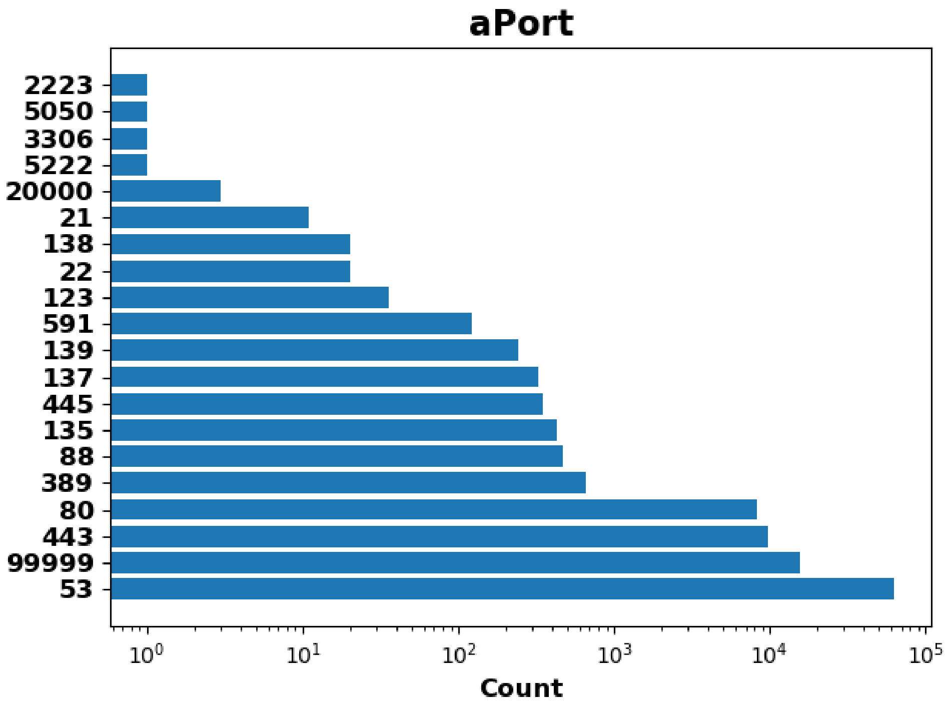
App Labels in the Training Data

App Label	Description
0	'unrecognized'
22	SSH
53	DNS
80	HTTP
137	MS NETBIOS
138	MS NETBIOS datagram service
139	MS SMB
389	Active directory
443	HTTPS
5004	RTP
5060	SIP



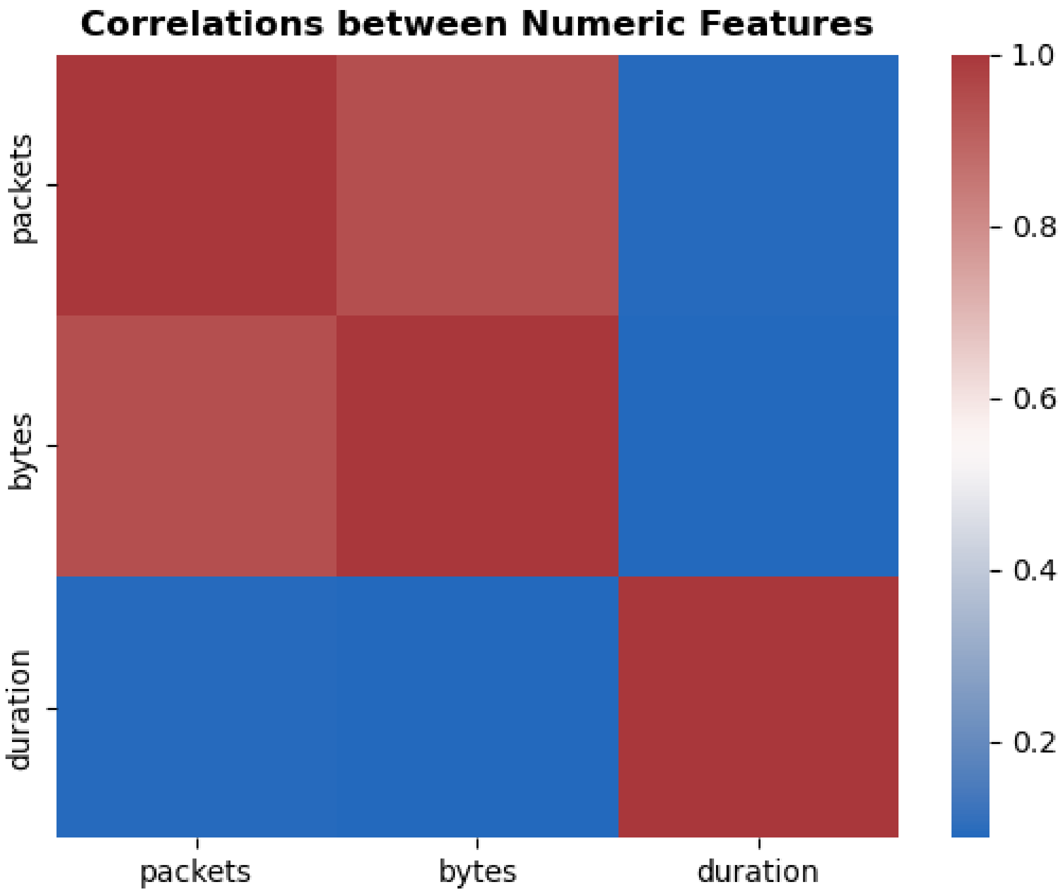
App Label & Destination Port





`application` "correlated" to `aPort`?

Correlations



Data Cleaning

Carnegie Mellon University Software Engineering Institute

Purpose and process of data cleaning

Handle missing data

Manage outliers

Fix errors

Remove unwanted observations

Data cleaning is the process of identifying and correcting or removing errors, inconsistencies, and inaccuracies in the data to improve its quality and usability[1]

Cleaning steps may not all be needed and may be done in various orders – and cycle.

[1] https://www.geeksforgeeks.org/data-cleansing-introduction/

Removing unwanted observations

Unwanted – unrelated or distracting from data under observation

- Irrelevant protocol
- Useless source or destination
- Bytes, packets, durations too large or too small for activity of interest

Query – filter model

```
rwfilter --start=2015/06/17 --type=in --protocol=6,17 \
   --pass=data.rw
rwfilter data.rw --proto=6 --bytes-per=60- --pass=data60.rw
rwfilter data.rw --proto=17 --bytes-per=40- --pass=data40.rw
rwcat data60.rw data40.rw \
| rwfilter stdin --not-anyset=ignore.set --pass=datasd.rw
```

Fix errors

Carnegie Mellon University Software Engineering Institute

Errors – structural or value artifacts deriving from the data collection process rather than from the phenomena being analyzed

- Out-of-order observations – rwsort
- Duplicate observations – rwdedup
- Split observations – rwcombine
- Unconnected observation – rwgroup & rwmatch
- Value errors – rwcut & rwtuc

rwsort

Merge and sort flow record files based on specified series of fields.

Places records in known order, rather than order from rwfilter

Enables or makes efficient further analyses

```
rwsort --fields=start,1-5 datasd.rw --out=datasd-sort.rw
```

rwdedupe

Omits duplicate records, based on parameters

Deals with same flow being detected by multiple sensors

Deals with overlapping collections of records

```
rwdedupe --ignore-fields=sensor --bytes-delta=50 \
  --stime-delta=100 --duration-delta=50 \
  < datasd-sort.rw > datasrt-ded.rw
```

rwcombine

Unifies records split by active timeout (flow attributes of T and C)
 - preserves semantics of split flows

Leaves other records unchanged

Only works for flows aggregated by sensor (not exported from router)

```
rwcombine datasrt-ded.rw --output=datasrt-com.rw
```

rwgroup and rwmatch

Carnegie Mellon University Software Engineering Institute

rwgroup – link, label (optionally summarize) groups of flow records

Group is defined by field values, requires sorting

```
rwsort datasrt-ded.rw --fields=1-5,sensor \
| rwgroup --id-fields=1-5,sensor --summarize >datagrp.rw
```

rwmatch – label collections of flows that relate to each other (suppress others)

Relating is defined by pairs of field values

```
rwfilter data.rw --type=out,outweb --pass=stdout \
| rwsort --fields=1-5,stime >data-qry.rw
rwfilter data.rw --type=in,inweb --pass=stdout \
| rwsort --fields=2,1,4,3,5,stime > data-rsp.rw
rwmatch --relate=1,2 --relate=2,1 --relate=3,4 --relate=4,3 \
   --relate=5,5 data-qry.rw data-rsp.rw data-mat.rw
```

Rwcut and rwtuc

Carnegie Mellon University Software Engineering Institute

rwcut – eliminate uninteresting features, format data

```
rwcut --fields=1-5,packets,stime data-mat.rw \
  --delim=, > data-mat.csv
```

rwtuc -- convert formatted data into binary data

- Useful if other tools have further cleaned the data (trim off milliseconds)

```
sed –E ‘s/T([^.]*)[^,]*/T\1/’ <data-mat.csv \
    >data-matsec.csv
rwtuc --column-sep=, data.matsec.csv \
    --output-path=data-matsec.rw
```

Manage outliers

Outliers – values requiring special handling in analysis

rwuniq – generate contingency tables

```
rwuniq data-matsec.rw --values=packets,bytes \
    --fields=sport,dport \
    --threshold=packets=10 --output=data-trim.txt --sort
```

Exclude cases where aggregate packets are too small (noise)

Can also threshold on bytes, flows, sip-distinct, dip-distinct, distinct:field

Can also threshold with min-max range

Handle missing data

Missing data –

- Inconsistent observations (collection not complete)
- Missing observations (gap in collection or lacking sensor)

```
rwfilter data-matsec.rw --proto=6 --flags=/SARFPU \
   --fail=data-clean.rw
```

Data Cleaning in Example

Not all of these cleaning steps are needed

Over cleaning also causes issues (Ozone hole example)

Data Cleaning and Feature Engineering

Feature Engineering

Carnegie Mellon University Software Engineering Institute

Process of Feature Engineering

Carnegie Mellon University Software Engineering Institute

Data Cleaning → **Transformation** → **Feature Selection**

Data Cleaning

- Missing values
- Outliers

Transformation

- Log transform
- Normalizing
- Encoding

Feature Selection

- Select minimal feature set required to achieve goal

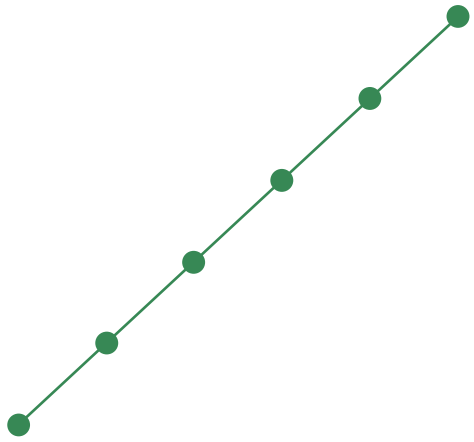
Data Types

```
packets          int64
bytes            int64
duration       float64
type            object
sIP             object
dIP             object
sPort           object
dPort           object
protocol        object
flags           object
application     object
```

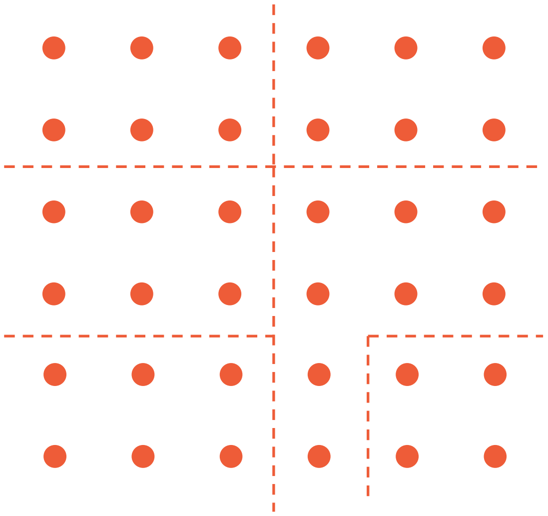
Numeric ☺

Categorical ☹

Numerical vs. Categorical



Underlying function



Underlying function?

Other Challenging Features

- 20 features (columns): **20-dimensional space**

- Some features have **many levels** (e.g., IPs)

Feature Engineering Steps

- **Queried from SiLK:**
  - sIP, dIP, sPort, dPort, protocol, packets, bytes, flags, sTime, duration, eTime sensor, in, out, nhIP, initialFlags, sessionFlags, attributes, application, class, type, sTime+msec, eTime+msec, dur+msec, iType, iCode

- **Dropped:**
  - in, out, nhIP, initialFlags, class, iType, iCode (no variation)
  - attributes (in training data but not test data)
  - dur+msec (redundant to 'duration')
  - sTime, eTime, sTime+msec, eTime+msec (not relevant)

- **Added:**
  - aPort

- **Transformed:**
  - sPort, dPort (grouped ephemeral ports together)
  - sIP, dIP (mapped to "internal" and "external)

Feature Engineering Steps

Carnegie Mellon University Software Engineering Institute

- **Final set of features:**
  - type, sIP, dIP, sPort, dPort, aPort, protocol, packets, bytes, duration, sessionFlags

Data Cleaning and Feature Engineering

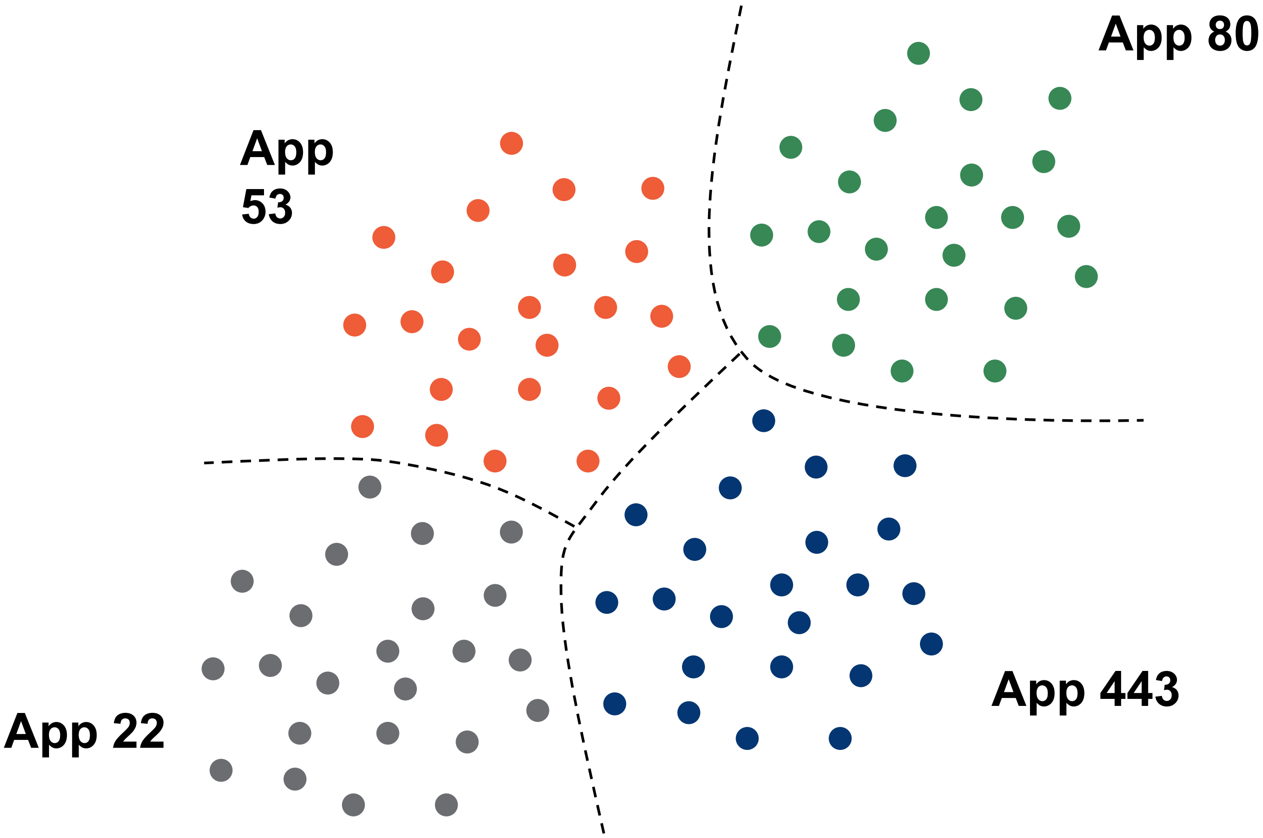
Machine Learning

How can we predict App Label?

Let’s try:

- Unsupervised clustering
- Supervised classification

Do logs clusters by app label?



k-Modes Clustering

- **"Similarity" between two logs:** number of equivalently valued fields

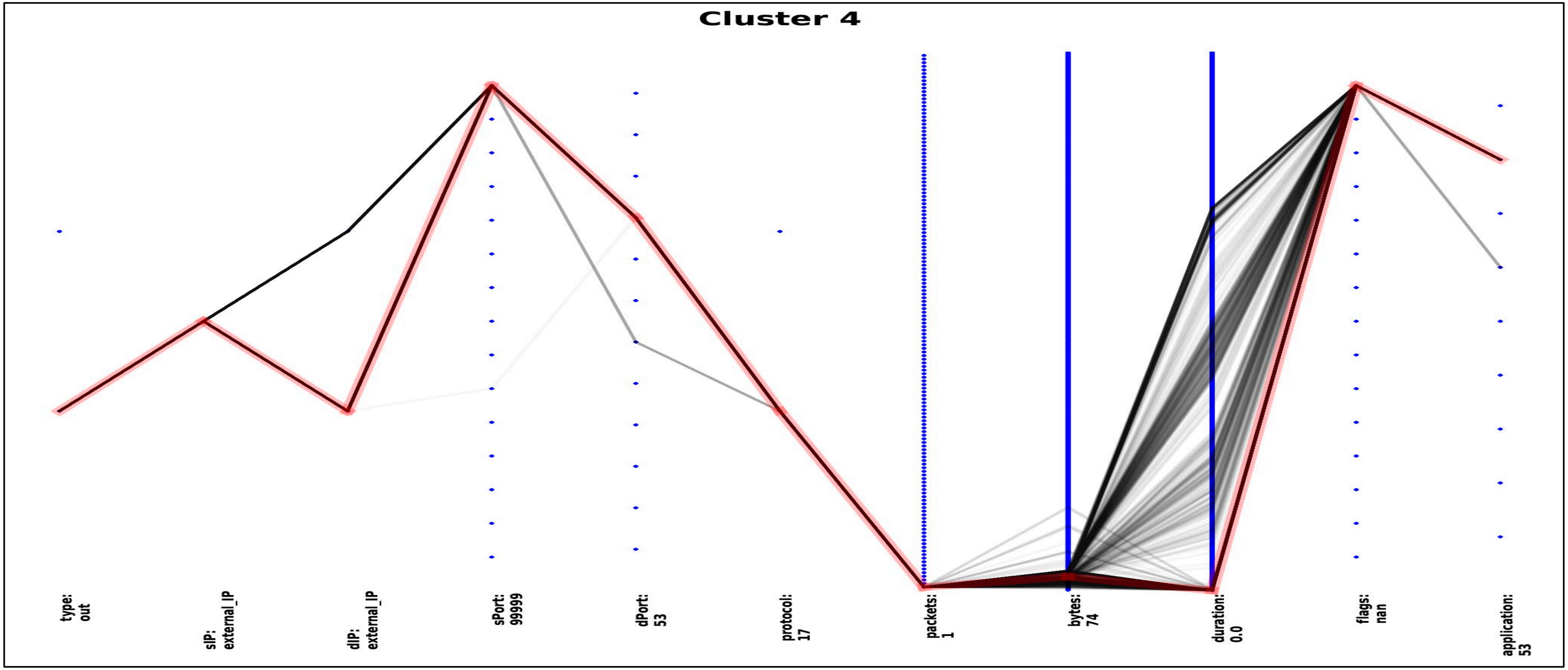
sIP	dIP	sPort	dPort	protocol	packets	bytes	flags	duration	application	type
192.168.111.94	192.168.40.20	51125	53	17	2	148		7.639	53	out
192.168.122.141	192.168.40.20	58081	53	17	5	370		19.417	53	out

- **Minimize total dissimilarity** between all logs and their assigned clusters

$$\min L_0 = \sum_{i=1}^{n} \sum_{j=1}^{m} \left| x_{ij} - \sum_{l=1}^{k} w_{il} z_{lj} \right|^0$$

- NP-hard

Interpreting the clustered logs

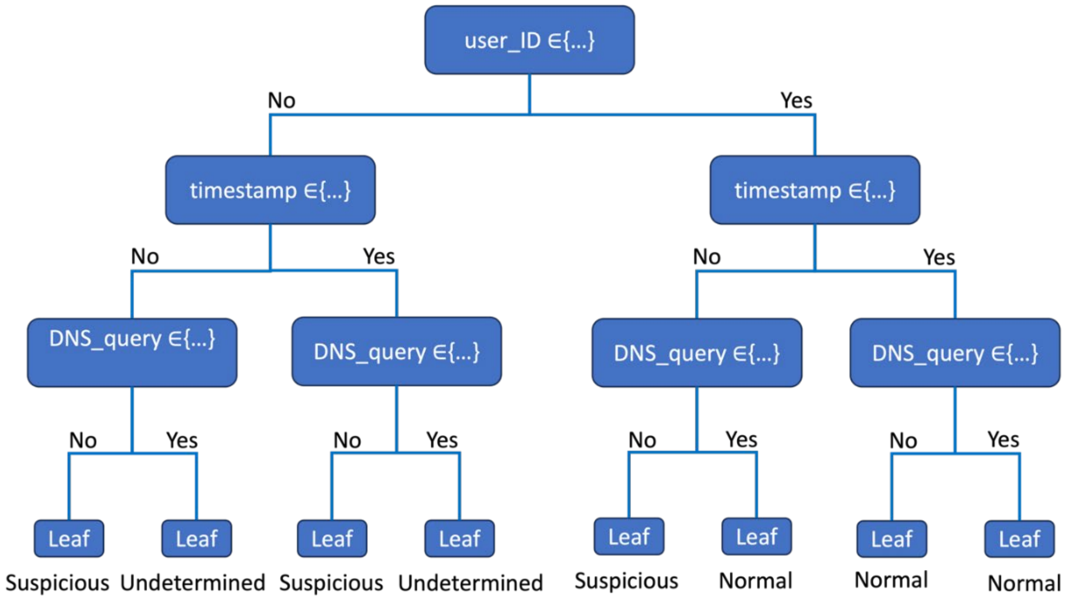


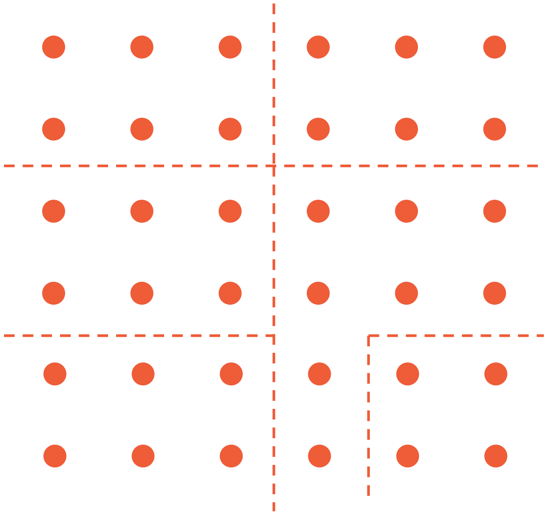
Single Packet, Short Duration, External-to-External DNS

Clustering Accuracy as an App Label Predictor

App Label	Description	Accuracy
53	DNS	**98%**
0	'unrecognized'	**89%**
80	HTTP	**57%**
22	SSH	**0%**
137	MS NETBIOS	**0%**
138	MS NETBIOS datagram service	**0%**
139	MS SMB	**0%**
389	Active directory	**0%**
443	HTTPS	**0%**
5004	RTP	**0%**
5060	SIP	**0%**

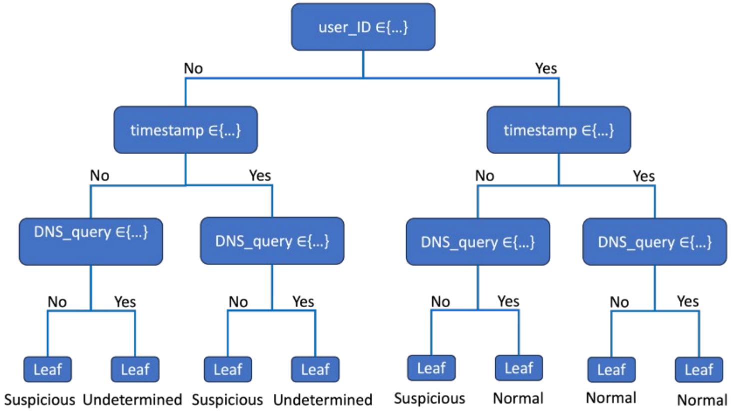
Decision Tree





Decision Trees
***Partition* the Data**

Human-Readable Rules




```
if user ID is within {employee 1, employee 2,...},
and if timestamp is between 8:00 AM and 5:00 PM,
and if DNS query is within {amazonaws.com, cisa.gov,...},
then suspicious = FALSE.
```

Trained Model Accuracy

Features	Accuracy
aPort	85%
aPort, duration	95%
aPort, duration, bytes	97%
aPort, protocol, duration, bytes	99.7%
type, sIP, dIP, sPort, dPort, aPort, protocol, packets, bytes, duration, sessionFlags	99.8%

Trained Model Logic (DNS)

```
*********   DECISION TREE FOR APPLICATION LABEL = 443   *********
If ( aPort is in [ 443 ]) {
        If ( duration >= 10.546646 ) {
                If ( bytes >= 419.0 or bytes is NaN ) {
                        Predicted value: 0.8999998
                } else {
                        Predicted value: -0.1
                }
        } else {
                If ( bytes >= 671.5 ) {
                        If ( bytes >= 3704.5 ) {
                                If ( bytes >= 6995.5 ) {
                                        Predicted value: 0.8599998
                                } else {
                                        Predicted value: 0.64999986
                                }
                        } else {
                                If ( duration >= 0.174625 or duration is NaN ) {
                                        Predicted value: 0.8999998
                                } else {
                                        Predicted value: 0.89452034
                                }
                        }
                } else {
                        If ( bytes >= 622.5 ) {
                                Predicted value: 0.09999999
                        } else {
                                Predicted value: -0.1
                        }
                }
        }
} else if ( aPort is in [ 21 22 53 80 88 123 135 137 138 139 389 445 591 2223 3306 5050 20000 99999 ]) {
        If ( aPort is in [ 99999 ]) {
                If ( duration >= 7.031098 ) {
                        If ( duration >= 21.038363 ) {
                                If ( duration >= 104.42536 ) {
                                        Predicted value: -0.09386503
                                } else {
                                        Predicted value: -0.059016395
                                }
                        } else {
                                If ( duration >= 9.375059 ) {
                                        Predicted value: 0.37902093
                                } else {
                                        Predicted value: -0.04029851
                                }
                        }
                } else {
                        If ( duration >= 4.998671 ) {
                                If ( duration >= 5.300795 ) {
                                        Predicted value: 0.39999995
                                } else {
                                        Predicted value: 0.8642855
                                }
                        } else {
                                If ( duration >= 2.7727003 ) {
                                        Predicted value: -0.07058824
                                } else {
                                        Predicted value: -0.09991479
                                }
                        }
                }
        } else if ( aPort is in [ 21 22 53 80 88 123 135 137 138 139 389 445 591 2223 3306 5050 20000 ]) {
                Predicted value: -0.1
        }
}
```


```
**********   DECISION TREE FOR APPLICATION LABEL = 443   **********
If ( aPort is in [ 443 ]) {

     If ( duration >= 10.546646 ) {

          If ( bytes >= 419.0 or bytes is NaN ) {

               Predicted value: 0.8999998

          } else {

               Predicted value: -0.1

          }
     } else {
```


Resulting rules can be implemented into cybersecurity systems without having to use machine learning.

Carnegie Mellon University Software Engineering Institute

Data Cleaning and Feature Engineering

Practicum

Carnegie Mellon University Software Engineering Institute

How would you proceed?

1. Certain web sites are larger consumers of bandwidth than others. What features can we use to predict the total bytes consumed by traffic involving a given web server IP Address? Why would this distinction be of importance?
2. What features can we use to predict the regularity with which certain DNS resolvers are consulted in this traffic? Is the traffic most of interest likely to be with periodic resolvers or aperiodic ones?

Summary

Carnegie Mellon University Software Engineering Institute

- ✓ State the purpose of data cleaning and feature engineering
- ✓ Explain basic steps in these processes
- ✓ Suggest open-source tools to accomplish these processes
- ✓ Adapt data cleaning and feature engineering to suit analysis needs

Contact Info

Tim Shimeall
Principal Engineer

Clarence Worrell
Senior Data Scientist

SEI Contact Info
Netsa-help@cert.org
+1 412-268-5800