Congestion Detection in Software-Defined Networks using Machine Learning

Master Thesis Final Presentation

Ali Murad Talpur

Supervised by:
Prof. Dr. Anna Förster
Dr.-Ing. Andreas Könsgen

13th Feb, 2017

Sustainable Communication Network
University of Bremen

Universität Bremen

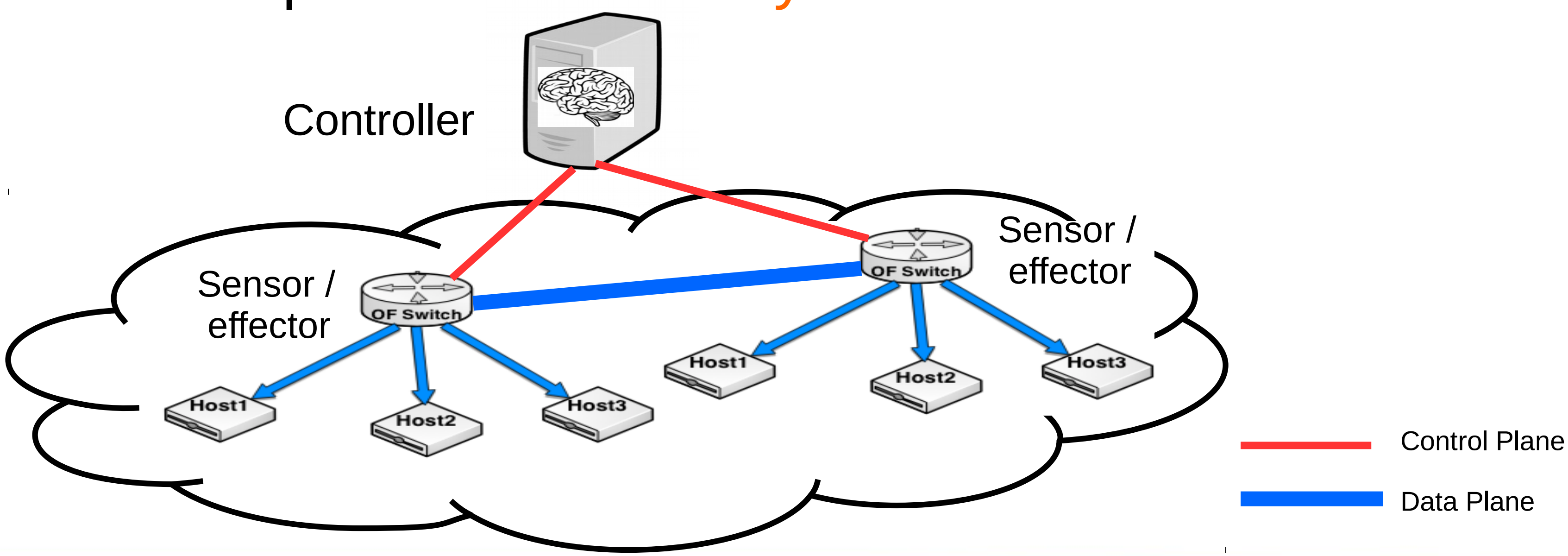
comnets

Motivation

- Todays' Networks scenario:
  - Hard to manage & configure – complex & heterogeneous
  - Do innovations – limited to vendors only
  - Perform real world experiments
- Consequences:
  - Slow rate of innovation – Ossification
  - Protocols are designed in isolation
- Need of
  - Easier Network Management
  - Vendor agnostic & open interfaces

Motivation

- Goals of Software-Defined Network (SDN)
  - Easier network management through abstraction
  - Programmability + Open Interfaces → New innovations
- Autonomic Network Management (ANM)
  - Self-managing network → Machine Learning
- ANM Control loop: Monitor → Analyze → Plan → Execute



Contents

Main Thesis Tasks

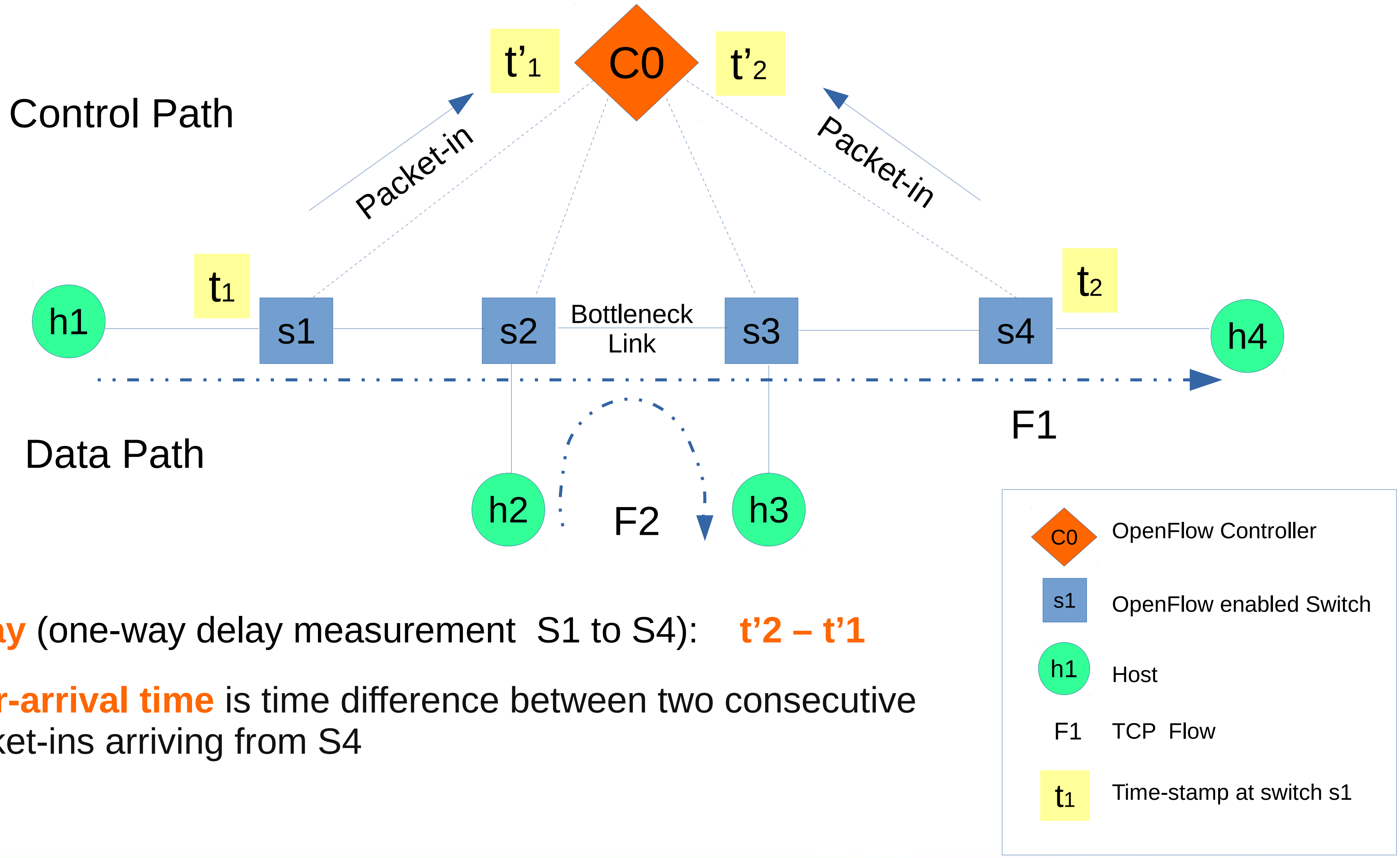
- Implementation of network with
  - OpenFlow (OF) protocol in Mininet emulator
- Selection of Suitable Features
- Run different experiments & Collect data from switches
- Apply Machine Learning algorithms to detect
  - Congestion in the network
  - Multiple flows in a bottleneck link
- Evaluate performance of classifier

Features Selection for Congestion Detection

- Features used in literature:
  - Round Trip Time (RTT) [4][5][8]
  - No. of Packets Lost, [8]
  - Congestion Window, [8][9]
  - One-way delay, [6][7][9]
  - Inter-arrival times [9][10]

- In the following experiments using OpenFlow (OF) protocol:
  - **One-way delay and Inter-arrival times** measured from OF switches
  - both are orthogonal as suggested in [6]

Method of Congestion Detection

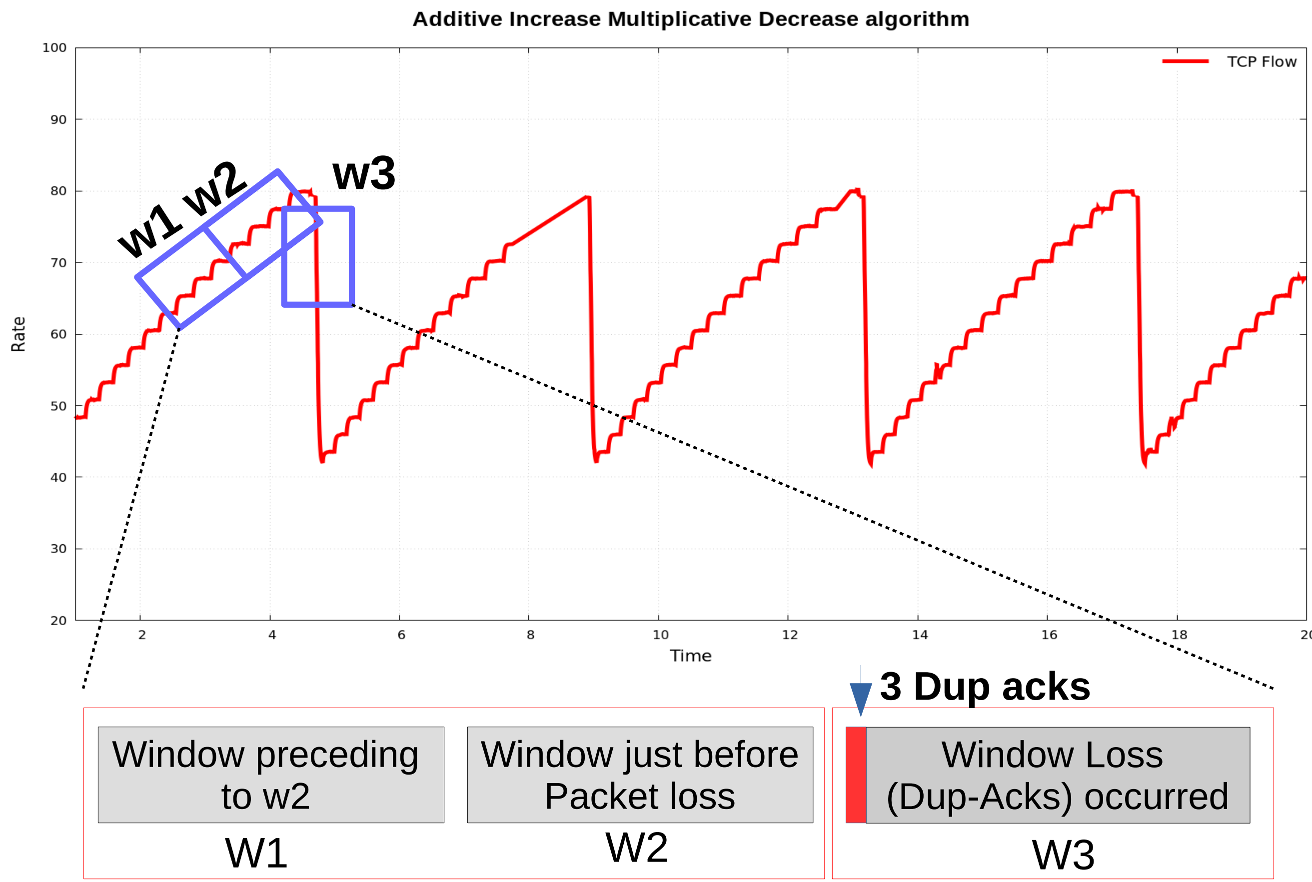


Delay (one-way delay measurement S1 to S4): **t'2 – t'1**

Inter-arrival time is time difference between two consecutive packet-ins arriving from S4

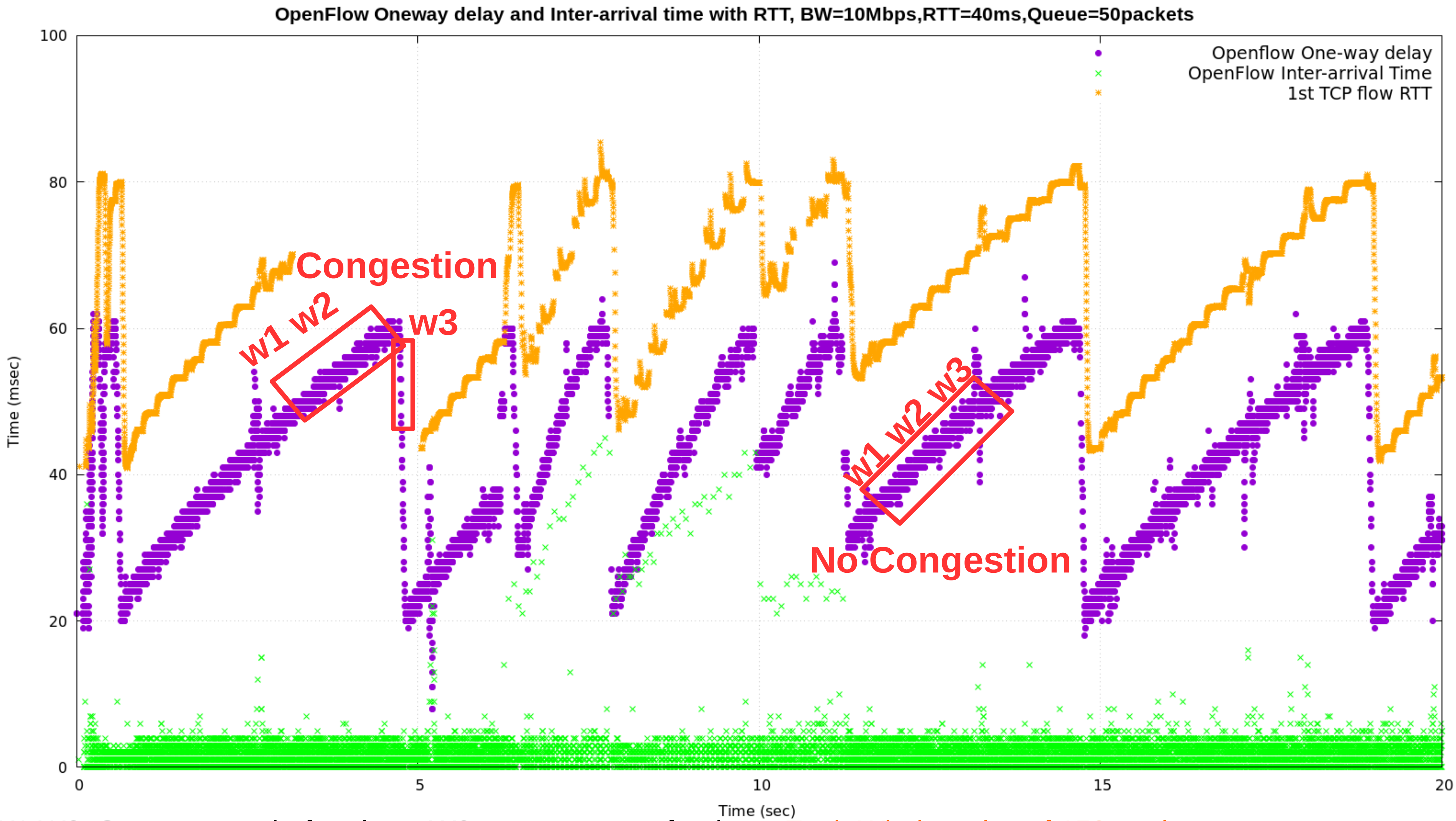
Method of Congestion Detection

- Important to know **SYSTEM STATE BEFORE AND AFTER LOSS**
- Take window (list) of packets just before and after the loss



System state **before loss** System state **after Loss**

RTT, One-way delay and IAT Graph

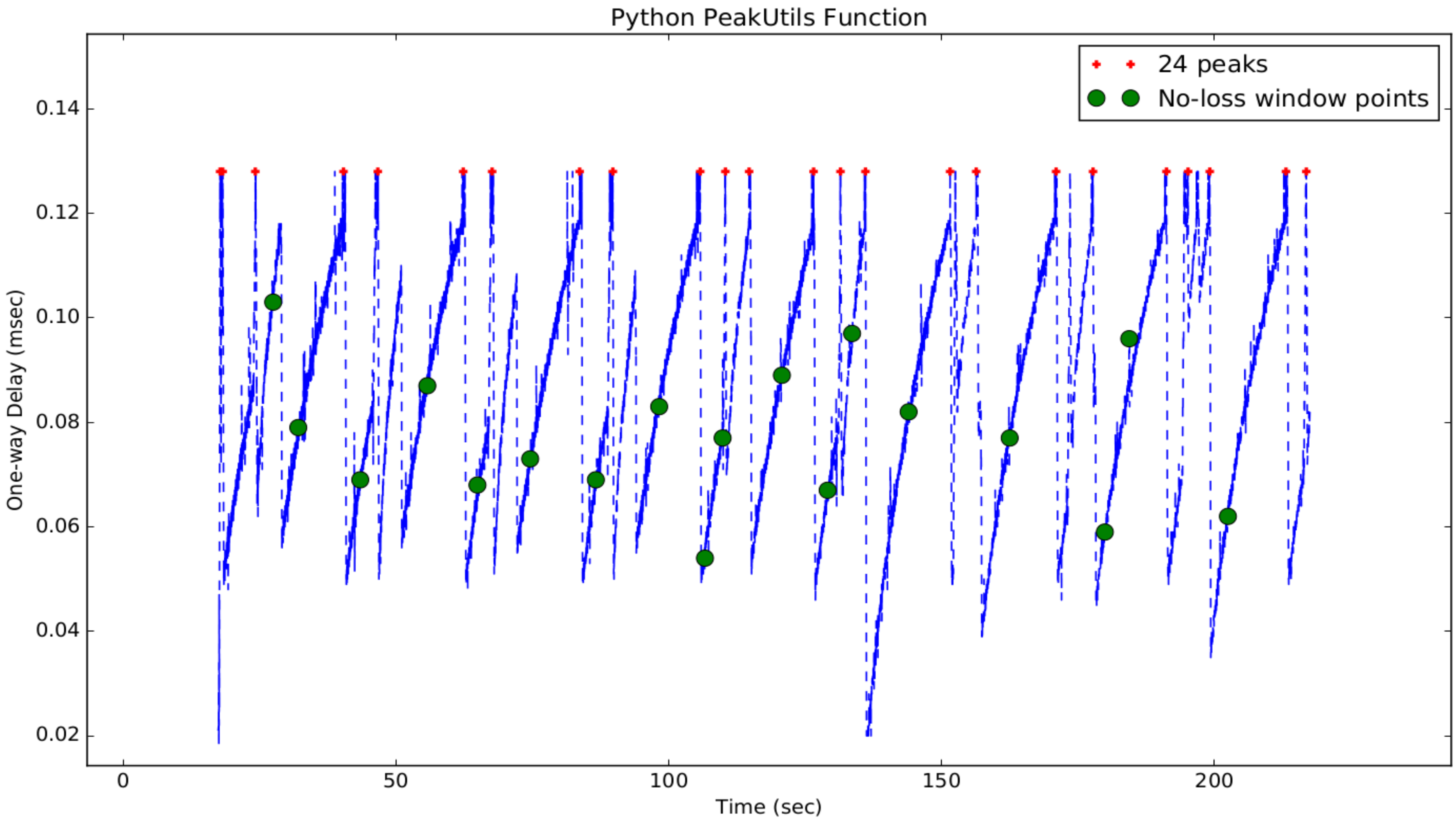


W1,W2: System state before loss, W3 system state after loss, Each Window size of 150 packets

RTT is measure from kernel print-k statements. Inter-arrival and One-way delay are measured from OF statistics

Congestion and No-congestion Points

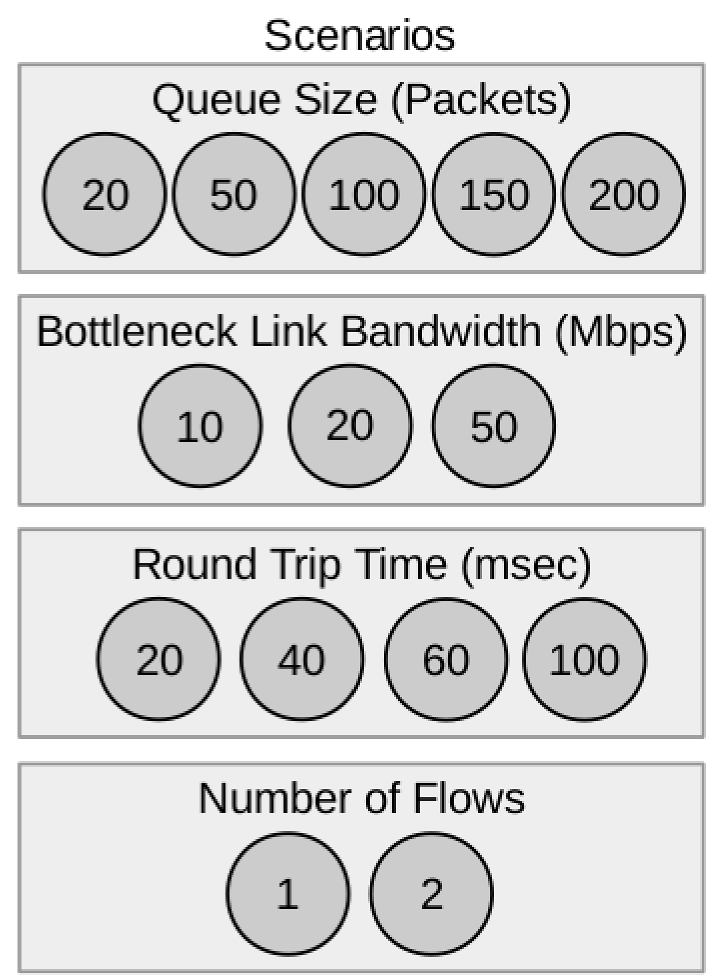
- Python **PeakUtils** Function: peakutils(data-vector, Threshold, Min-distance)



Scenarios and Attributes

- Scenarios
- 2 Features, 63 Attributes
  - Mean, Max, Min, Std deviation
  - and Ratio of Mean, Max, Min, SD of windows
- Example of an attribute:
  - maxw3delay_div_minw1delay is ratio of one-way delay of Max value in window3 by Min value in window1

$$\frac{\text{Max of one way delay in Window 3}}{\text{Min of one way delay in Window 1}}$$

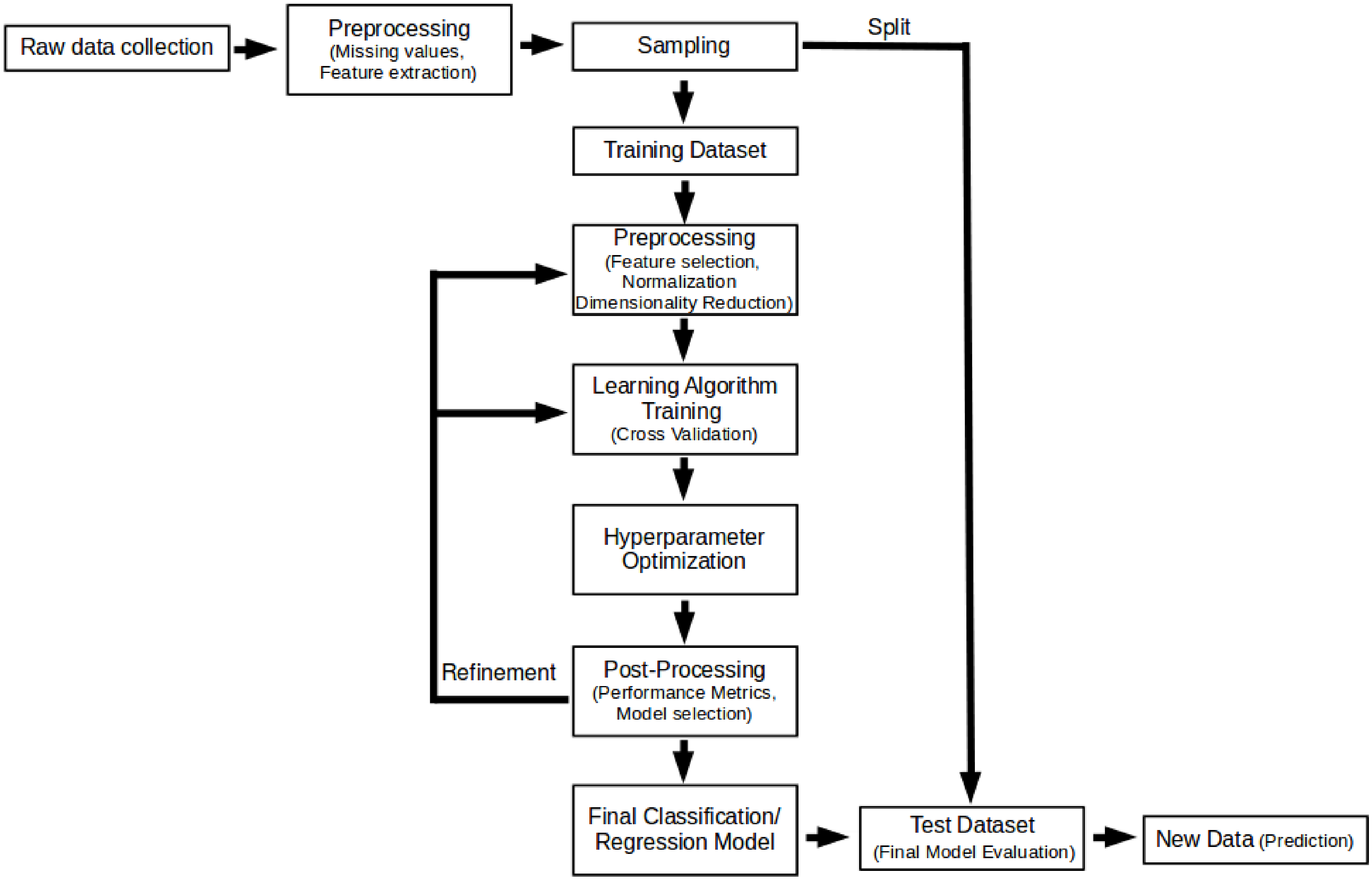


Datasets

- Balanced dataset: Total **1296** data instances
  - Congestion 621 (48%) and no-congestion 675 (52%) instances
- WEKA Machine Learning tool, Univ of Waikato NZ
- From single dataset → Shuffle/Randomized → 3 different datasets
- **60%-20%-20%** split into **Training, Validation & Test datasets**

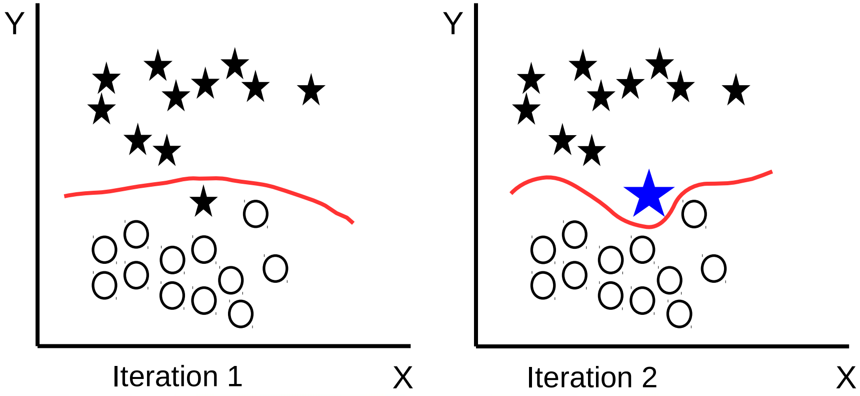
Dataset	
Total instances	1296
Training+Validation instances	777+260
Test instances	259
Congestion instances	621
No congestion instances	675

Supervised Machine Learning



Supervised Learning Algorithms

- Decision Tree: Robust, Fast & Comprehensible
- **C4.5 Decision Tree classifier**
  - Most popular, Works on Gain ratio (information gain and split information)
  - Avoids over fitting by pruning the tree
- Ensembles: Use different models together, to make predictions more reliable
- **Bagging** (Bootstrap Aggregating) classifier
  - Same sized, several Training data using sampling (with replacement)
  - Build model with Equal weights, Combine by voting
- **Boosting** (AdaBoost M1) classifier
  - Models are build in sequence (iteratively)
  - Non-equal weights extra weight for misclassified instances
  - Combine by voting



Results: C4.5 algorithm

- C4.5 Decision tree algorithm results

1

Confusion Matrix		
Actual	Classified as	
	C	N
C	110	11
N	17	122

2

Confusion Matrix		
Actual	Classified as	
	C	N
C	109	21
N	12	118

3

Confusion Matrix		
Actual	Classified as	
	C	N
C	115	16
N	18	111

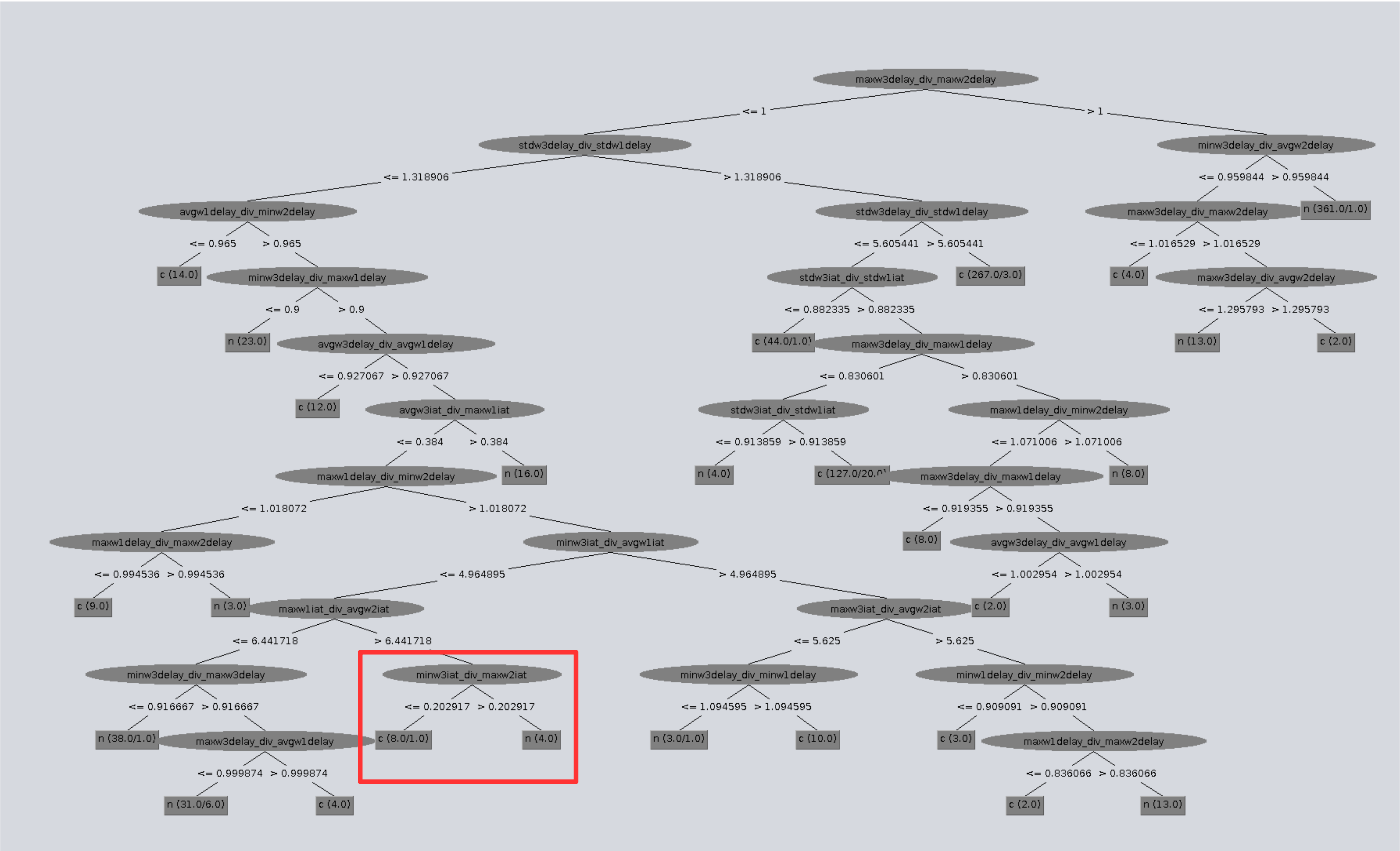
C4.5		1	2	3	Mean
	Validation	90.3475 %	92.278 %	90.7336 %	91.1197 %
	Test	89.2308 %	87.3077 %	86.9231 %	87.82053 %

C = Congestion
N= No-Congestion

True Negative (TN): actual negative, predicted negative
True Positive (TP): actual positive, predicted positive
False Negative (FN): actual positive, predicted negative
False Positive (FP): actual negative, predicted positive

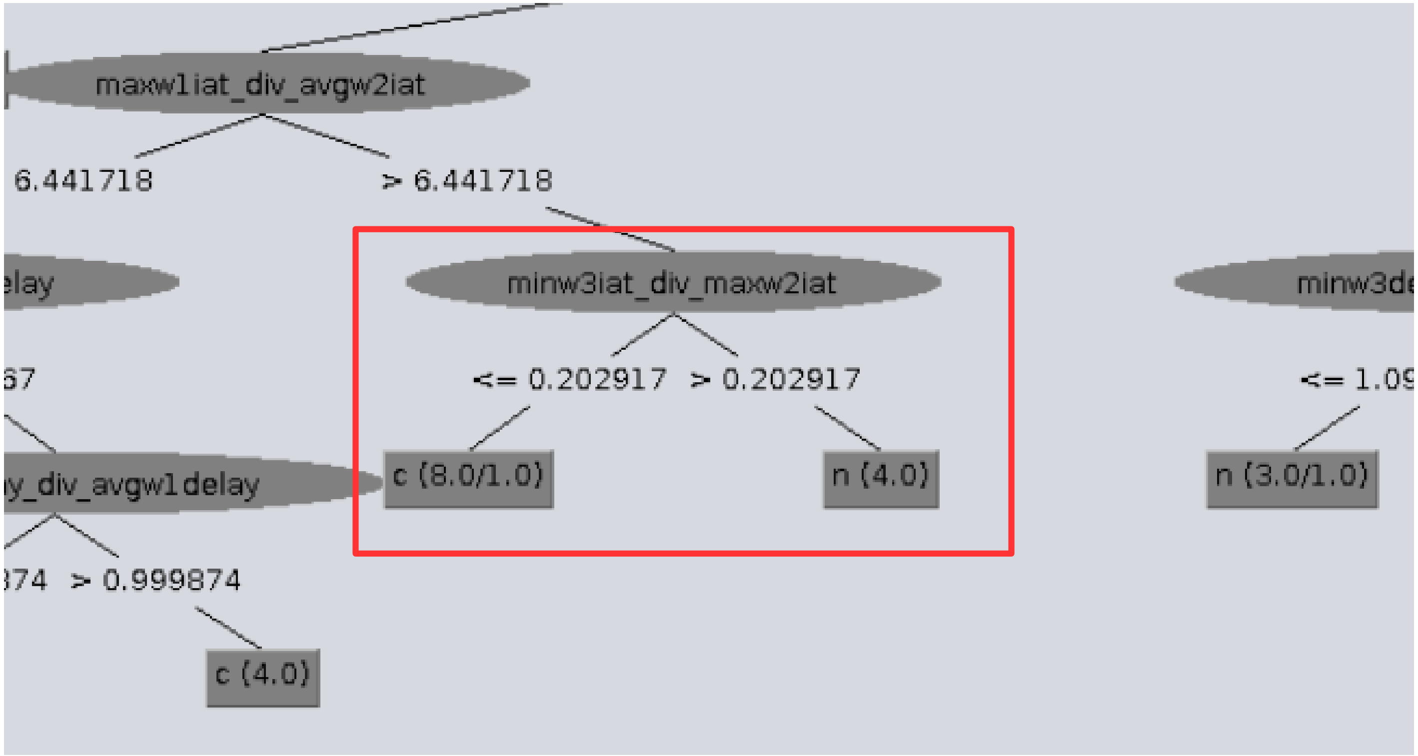
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Results: C4.5 Decision Tree



Results: C4.5 algorithm

- Decision Tree structure (pruned)
  - Number of Leaves : 28
  - Size of the tree: 55



Results: AdaBoost and Boosting

- AdaBoost algorithm results

Confusion Matrix		
Actual	Classified as	
	C	N
C	109	12
N	22	117

Confusion Matrix		
Actual	Classified as	
	C	N
C	119	17
N	13	111

Confusion Matrix		
Actual	Classified as	
	C	N
C	121	8
N	26	105

AdaBoost		1	2	3	Mean
	Validation	89.1892 %	89.5753 %	88.417 %	89.0605 %
	Test	86.9231 %	88.4615 %	86.9231 %	87.4359 %

- Bagging algorithm results

Confusion Matrix		
Actual	Classified as	
	C	N
C	117	12
N	13	118

Confusion Matrix		
Actual	Classified as	
	C	N
C	121	9
N	10	120

Confusion Matrix		
Actual	Classified as	
	C	N
C	126	10
N	9	115

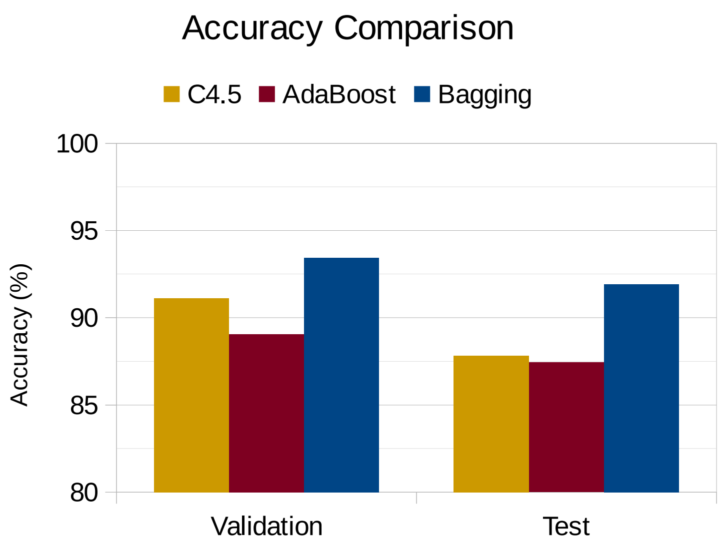
Bagging		1	2	3	Mean
	Validation	93.8224 %	93.4363 %	93.0502 %	93.4363 %
	Test	90.3846 %	92.6923 %	92.6923 %	91.923 %

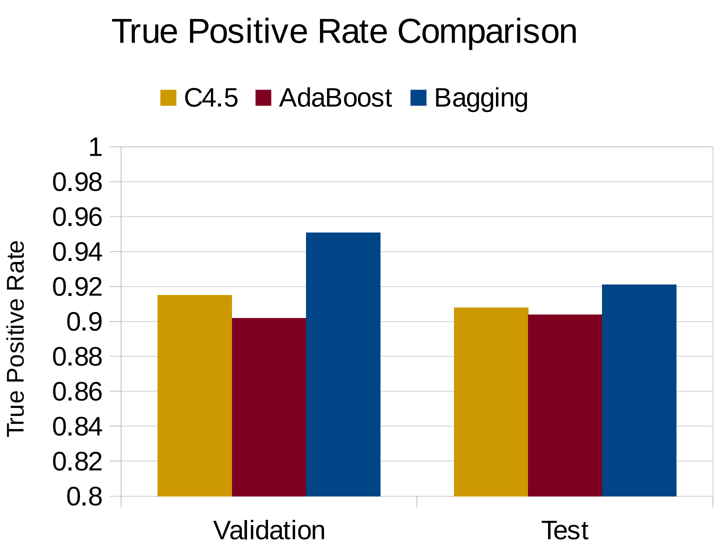
Overall Results

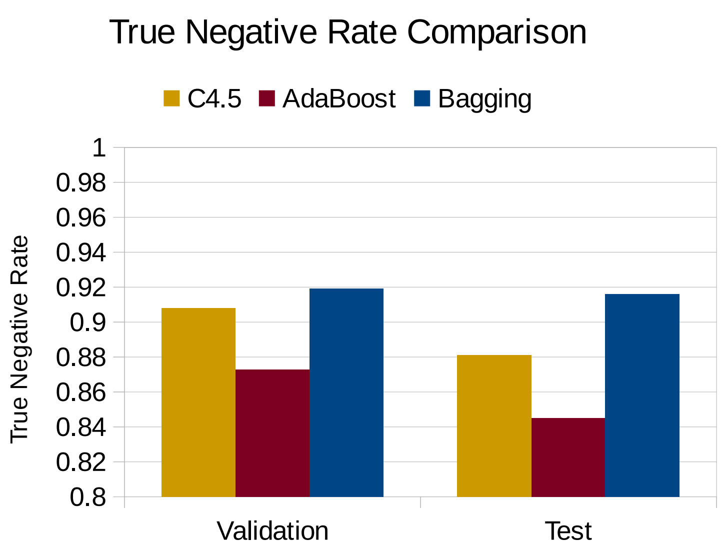
$$Accuracy(A) = \frac{TP + TN}{N}$$

$$TPR = \frac{TP}{TP + FN}$$

$$TNR = \frac{TN}{TN + FP}$$







Overall Results

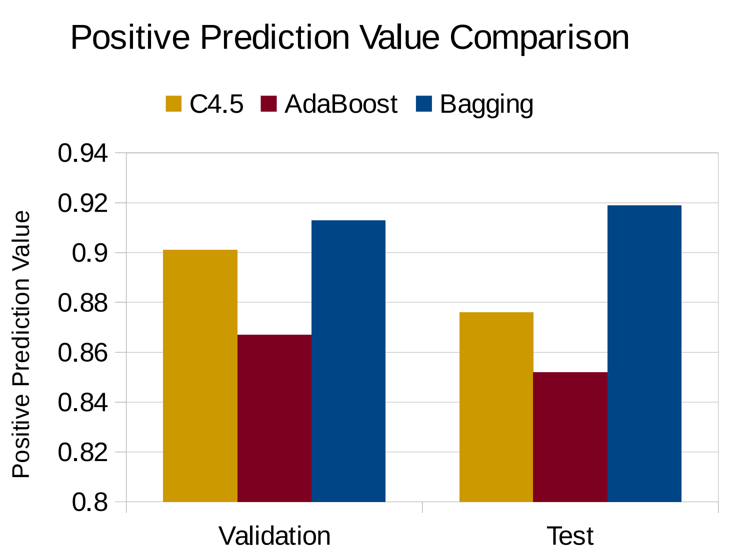
- PPV

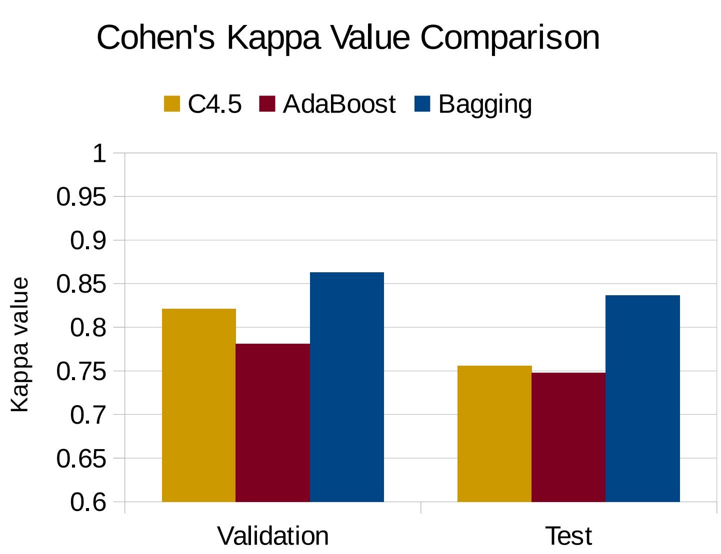
$$PPV = \frac{TP}{TP + FP}$$

- Cohen's Kappa

$$\kappa = \frac{A - AC}{1 - AC}$$

$$AC = \frac{TP + FP}{N} \cdot \frac{TP + FN}{N} + \frac{TN + FN}{N} \cdot \frac{TN + FP}{N}$$





Overall Results

- Comparing three algorithms
- **Bagging Classifier** performs best in Validation and Test results

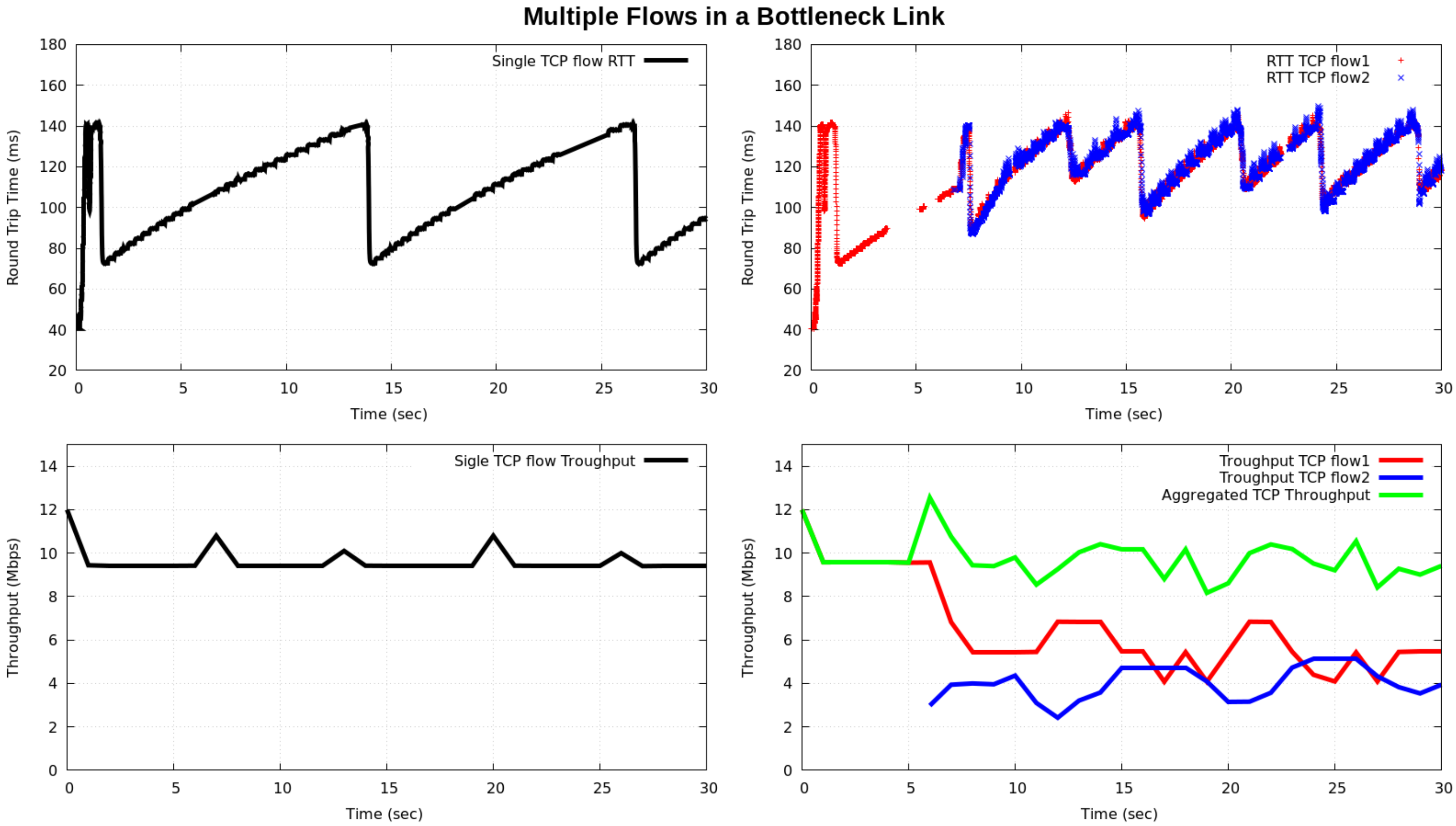
C4.5		1	2	3	Mean
	Validation	90.3475 %	92.278 %	90.7336 %	91.1197 %
	Test	89.2308 %	87.3077 %	86.9231 %	87.82053 %

AdaBoost		1	2	3	Mean
	Validation	89.1892 %	89.5753 %	88.417 %	89.0605 %
	Test	86.9231 %	88.4615 %	86.9231 %	87.4359 %

Bagging		1	2	3	Mean
	Validation	93.8224 %	93.4363 %	93.0502 %	**93.4363 %**
	Test	90.3846 %	92.6923 %	92.6923 %	**91.923 %**

Multiple Flows Detection

- Detecting number of flow sharing a bottleneck link: Helps to avoid congestion
- High number of flows leads to
  - higher losses and potentially lower throughput per flow

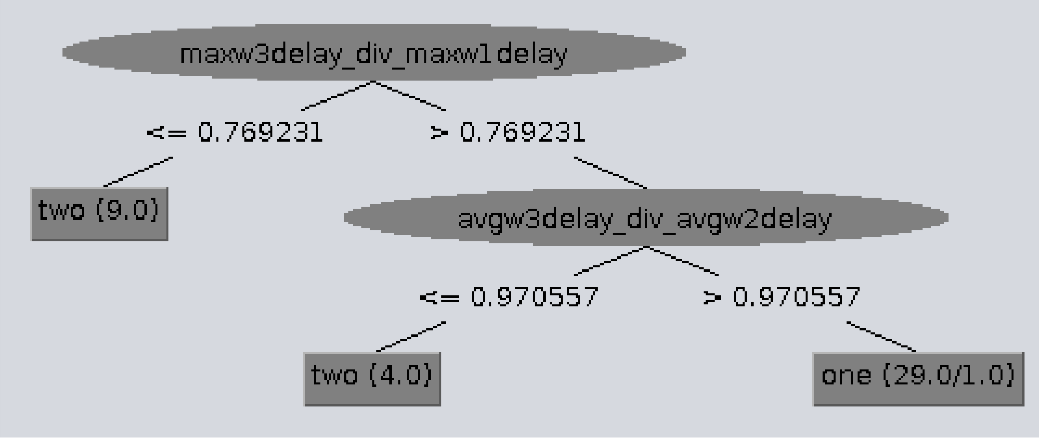


Universität Bremen

comnets

Number of Flows Detection

- Detection of One vs. Two flows
- Feature: One-way delay, attribute: 32
- Number of samples: 50
- Cross validation accuracy: **92.8571 %**
- A good topic for next Thesis/ Project (identifying new features etc.)



Conclusion

- Implementation: OF/SDN on Mininet
- Congestion Detection
  - Feature selection, Datasets preparation
  - Comparison of C4.5, AdaBoost, Bagging
  - Bagging performed best: 91.1% accuracy
- Proposed method of Congestion Detection works with acceptable level of Accuracy
- Number of Flows detection
- Future Work
  - New OF protocol version – New Features
  - Based on the classifier results to re-route the network traffic

References

[1] OpenFlow: https://www.opennetworking.org/

[2] Customizable ANM, Intergrating ANM and SDN http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7047267&tag=1

[3]Vision of Autonomic Computing – Jeffrey O. Kephart, IBM Thomas J. Watson Research Center

[4]J. Liu, I. Matta, and M. Crovella, "End-to-end inference of loss nature in a hybrid wired/wireless environment," Boston University Computer Science Department,Tech. Rep., 2002.

[5]D. Barman and I. Matta, "Effectiveness of loss labeling in improving tcpperformancein wired/wireless networks," in Network Protocols, 2002. Proceedings. 10th IEEEInternational Conference on. IEEE, 2002, pp. 2–11.

[6]Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, "Achieving moderatefairness for udp flows by path-status classification," in Local Computer Networks,2000. LCN 2000. Proceedings. 25th Annual IEEE Conference on. IEEE, 2000, pp.252–261.

[7]S. Cen,Cosman, and Voelker, "End-to-end differentiation of congestion & wireless losses,

[8]"Adaptive Control TCP," https://homepages.staff.os3.nl/~delaat/news/2013-03-05/tcppred.pdf, accessed:2017-01-15.

[9]I. El Khayat, P. Geurts, and G. Leduc, "Enhancement of tcp over wired/wireless networks with packet loss classifiers inferred by supervised learning,"

[10]S. Biaz and N. H. Vaidya, "Discriminating congestion losses from wireless lossesusing inter-arrival times at the receiver," in Application-Specific Systems and SoftwareEngineering and Technology, 1999.

[11]Peakutils: http://pythonhosted.org/PeakUtils/tutorial_a.html

[12]Weka ML tool: http://www.cs.waikato.ac.nz/ml/weka/

[13]https://github.com/mininet/mininet

The End
&
Thank you!

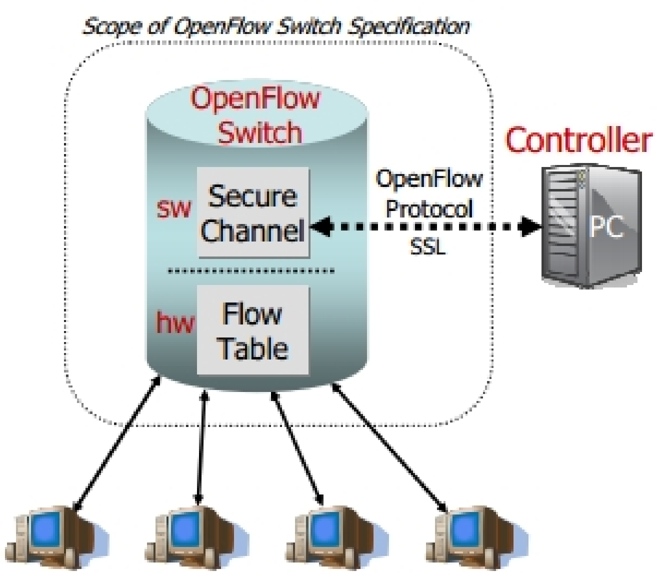
Backup Slides

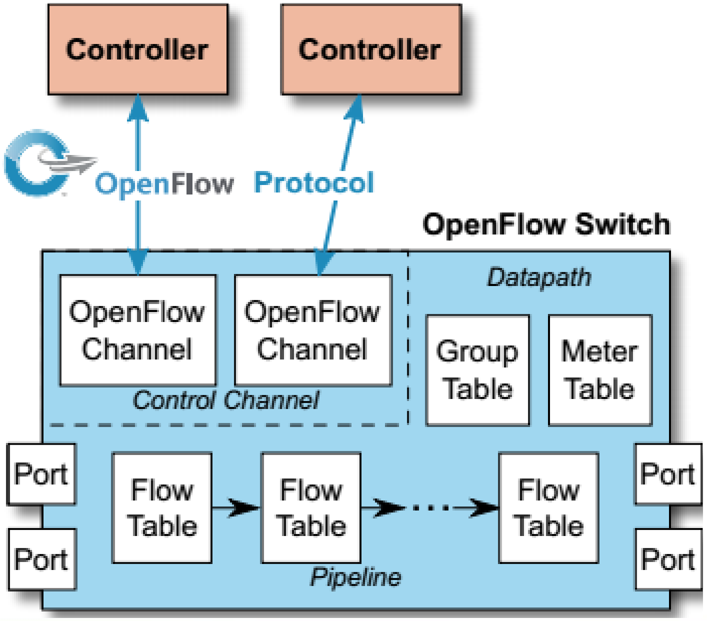
Software Defined Networking – SDN

- SDN architecture:
  - Separates Control & Data planes
  - Global network view – centralized control
  - Abstraction of network infrastructure
- SDN brings:
  - Easier network management
  - Programmability & Cost effective research – faster growth
  - Innovation & Customization – network development
  - Integration of new software application

OpenFlow – OF [1]

- Protocol for Control & Data plane communication
- Components: Controller, Switch, Secure Channel (TLS)
- Runs on Commodity switches: $ 5 R-pi can be an OF switch



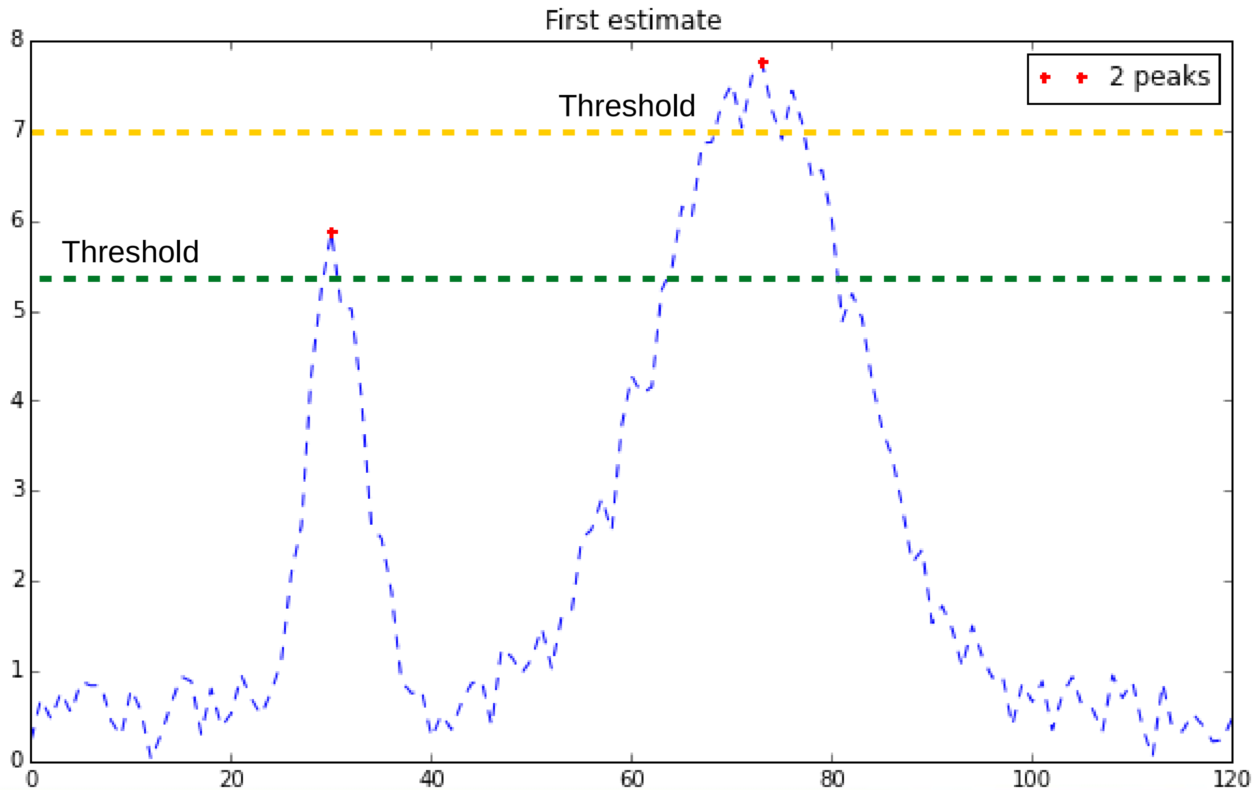


Mininet Emulator [13]

- Virtualized environment
  - Virtual network emulator
  - Hosts, links, switches, controller etc.
  - Hosts with Linux kernel & network stack
- Supports for research & development
- Support for SDN / OF
  - Kernel & User space OF switches
  - OF based Controllers

Packet window: *peakutils* function

- PeakUtils: *peakutils(data_vector, threshold, min_distance)*
- *peakutils(data_vector, 0.9, 10)*
- *peakutils(data_vector, 0.75, 10)*



Machine Learning Techniques

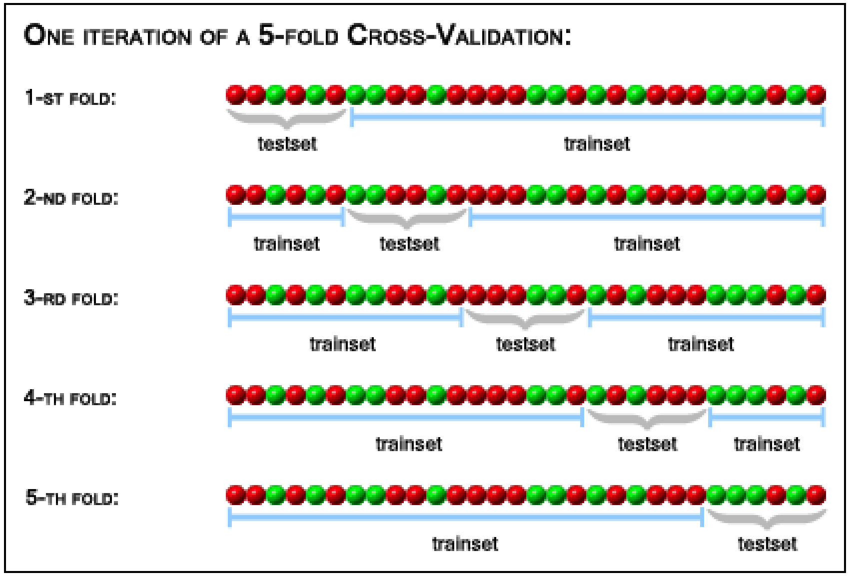
- "*Based on past experience, automatically learn to make accurate predictions*"
- Supervised, Unsupervised & Reinforcement learning
- Applications: Traffic classification, Congestion, Fault detection etc
- Fault detection or pattern detection: Classification problem
- Classifiers: Fault or no fault
  - Input: A vector, Output: Class, e.g. Email spam filter
- Machine Learning Approaches:
  - Neural Networks (NN), Decision Trees (DT), Support vector machine (SVM), Bayesian Networks (BN) etc
- **Selection of technique? No single criteria! (No free lunch theorem)**

ML basics

-Training set: used for learner to learn the relationship of input to output

-Test set is independent of training set and is used to asses the quality of learner

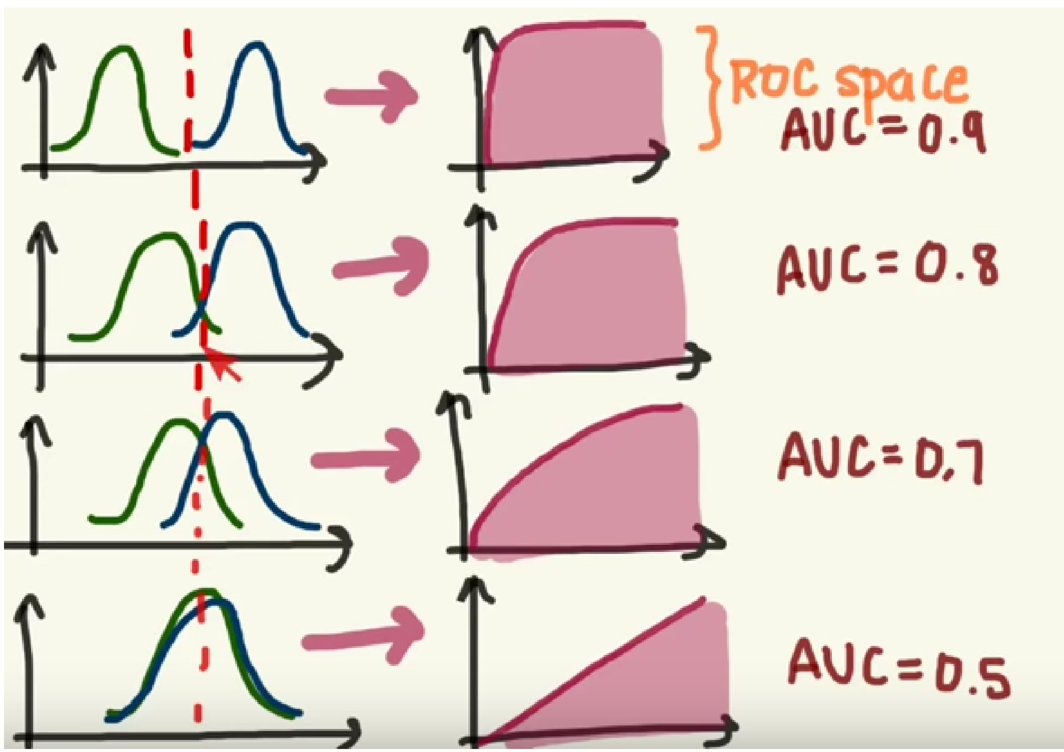
-cross-validation can be used to estimate the test error associated with a given statistical learning method in order to evaluate its performance, or to select the appropriate level of flexibility.

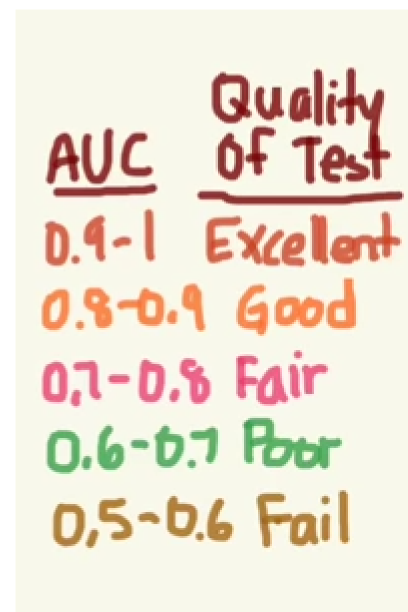


http://genome.tugraz.at/proclassify/help/pages/XV.html

ML basics

- ROC is a graphical plot of (TPR) / (FPR) for a binary classifier as its discrimination threshold is varied.
- AUC is the area under the ROC curve. As the TPR and the FPR range from 0 to 1, the maximal possible value of AUC is 1 which means a perfect classification. In practice, AUC is smaller than 1. The closer it is to 1, the better.





https://www.youtube.com/watch?v=21Igj5Pr6u4&index=2&list=PL41ckbAGB5S2PavLIXUETzAmi5reIod23

More Results

- C4.5 Results

	Validation dataset			Test dataset		
Seed value	2	545	2225	2	545	2225
Accuracy (A)	90.347%	92.278%	90.7336%	89.230%	87.307%	86.923%
Error (E)	9.652%	7.722%	9.266%	10.769%	12.692%	13.076%
True Positive Rate (TPR)	0.901	0.949	0.895	0.909	0.838	0.878
False Positive Rate (FPR)	0.094	0.099	0.081	0.122	0.092	0.140
True Negative Rate (TNR)	0.906	0.901	0.918	0.877	0.907	0.860
False Negative Rate (FNR)	0.099	0.051	0.104	0.090	0.161	0.122
Positive Prediction Value (PPV)	0.907	0.888	0.909	0.866	0.900	0.864
Negative Prediction Value (NPV)	0.899	0.955	0.905	0.917	0.848	0.874
F1 Score	0.904	0.917	0.902	0.887	0.869	0.871
Cohen's Kappa	0.806	0.845	0.814	0.784	0.746	0.738
ROC area	0.916	0.935	0.897	0.908	0.855	0.842

More Results

- AdaBoost Results

	Validation dataset			Test dataset		
Seed value	2	1023	1600	2	1023	1600
Accuracy (A)	89.189%	89.575%	88.417%	86.923%	88.461%	86.923%
Error (E)	10.810%	10.424%	11.583%	13.076%	11.538%	13.076%
True Positive Rate (TPR)	0.913	0.838	0.956	0.901	0.875	0.938
False Positive Rate (FPR)	0.121	0.092	0.172	0.158	0.105	0.198
True Negative Rate (TNR)	0.914	0.878	0.827	0.841	0.895	0.801
False Negative Rate (FNR)	0.129	0.086	0.043	0.099	0.125	0.062
Positive Prediction Value (PPV)	0.912	0.878	0.813	0.832	0.901	0.823
Negative Prediction Value (NPV)	0.873	0.913	0.960	0.906	0.867	0.929
F1 Score	0.891	0.896	0.879	0.865	0.888	0.877
Cohen's Kappa	0.783	0.791	0.769	0.738	0.769	0.738
ROC area	0.947	0.966	0.967	0.942	0.961	0.914

More Results

- Bagging Results

	Validation dataset			Test dataset		
Seed value	312	545	1023	312	545	1023
Accuracy (A)	93.822%	93.436%	93.050%	90.384%	92.692%	92.692%
Error (E)	6.177%	6.563%	6.949%	9.615%	7.307%	7.307%
True Positive Rate (TPR)	0.944	0.966	0.945	0.907	0.931	0.926
False Positive Rate (FPR)	0.067	0.092	0.083	0.099	0.077	0.073
True Negative Rate (TNR)	0.934	0.908	0.916	0.900	0.923	0.927
False Negative Rate (FNR)	0.056	0.112	0.086	0.093	0.069	0.073
Positive Prediction Value (PPV)	0.928	0.896	0.916	0.900	0.923	0.934
Negative Prediction Value (NPV)	0.947	0.969	0.945	0.907	0.930	0. 920
F1 Score	0.936	0.930	0.930	0.903	0.927	0.930
Cohen's Kappa	0.876	0.868	0.861	0.807	0.853	0.853
ROC area	0.981	0.981	0.981	0.964	0.979	0.983

More Results

- Bagging ROC curve for test dataset (seed value 312)

Weka ML tool [12]

- Open source ML tool from Univ. of Waikato, NZ
- Weka contains
  - Data pre-processing and filtering tools,
  - Classifiers using Bayes, Decision Trees, Rules, functions, meta
  - Regression
  - Clustering
  - Visualization