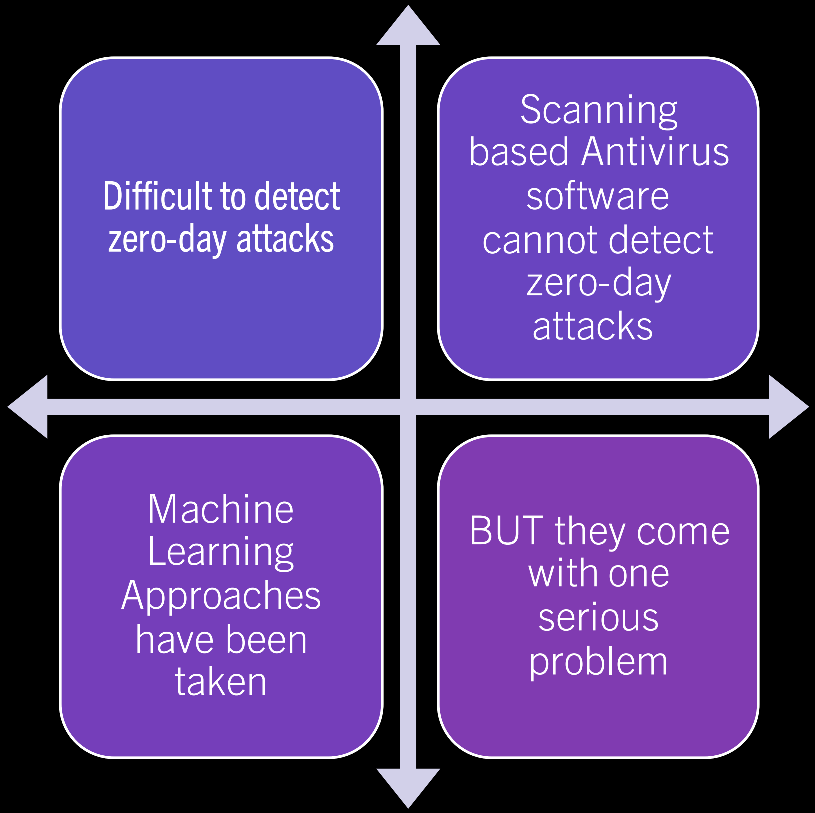
HOW A MODEL THINKS? HOW MULTI-LAYER PERCEPTRON DISTINGUISHES DATA POINTS?

Ashim Dahal

THE PROBLEM



Difficult to detect zero-day attacks

Scanning based Antivirus software cannot detect zero-day attacks

Machine Learning Approaches have been taken

BUT they come with one serious problem

HOW CAN MACHINE LEARNING FAIL?

- BECAUSE OF HIGH ACCURACY
- RESEARCHERS FOCUS ON GETTING THE BEST ACCURACY IN THE KDD99 DATASET
- BUT IN CASES LIKE THESE, ACCURACY AS A SOLE METRIC DOESN'T SUFFICE
- THIS RESEARCH FOCUSES ON REDUCED BIAS AND TRY TO EXPLAIN WHY SUCH BIAS EXISTED IN THE MODEL IRRESPECTIVE OF OUTPUT

DATASET AND LITERATURE REVIEW

KDD99: 4.8 Million samples of 23 attack types, 2.8 Million belong to Smurf and 1 Million belong to Neptune

Out of the 23 classes in the dataset, the sum of number of samples for bottom 20 is less than 50,000.

99.98% accuracy = 20 unnoticed classes

Machine Learning learns from the data and these data make model biased

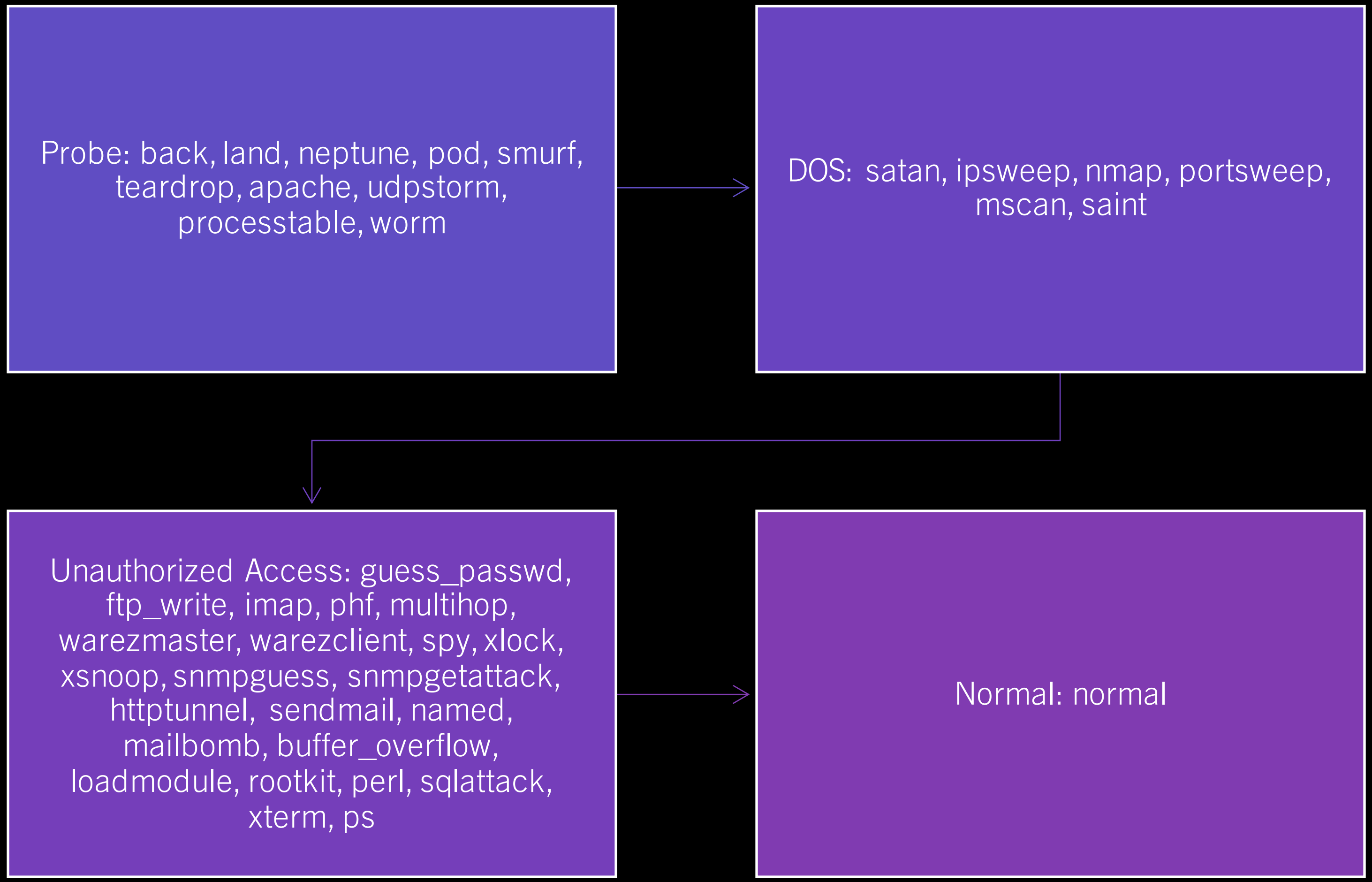
THE THREE STEP SOLUTION

STEP 1: MAKE THE DATASET LESS BIASED IN ITSELF

STEP 2: BUILD A ROBUST ML MODEL THAT ACKNOWLEDGES THE DISPARITY ON THE DATA DISTRIBUTION IN THE DATASET

STEP 3: ANALYZE THE BLACKBOX APPROACH USING EXPLAINABLE AI (XAI)

STEP 1: DEBIASING THE DATASET



Probe: back, land, neptune, pod, smurf, teardrop, apache, udpstorm, processtable, worm

DOS: satan, ipsweep, nmap, portsweep, mscan, saint

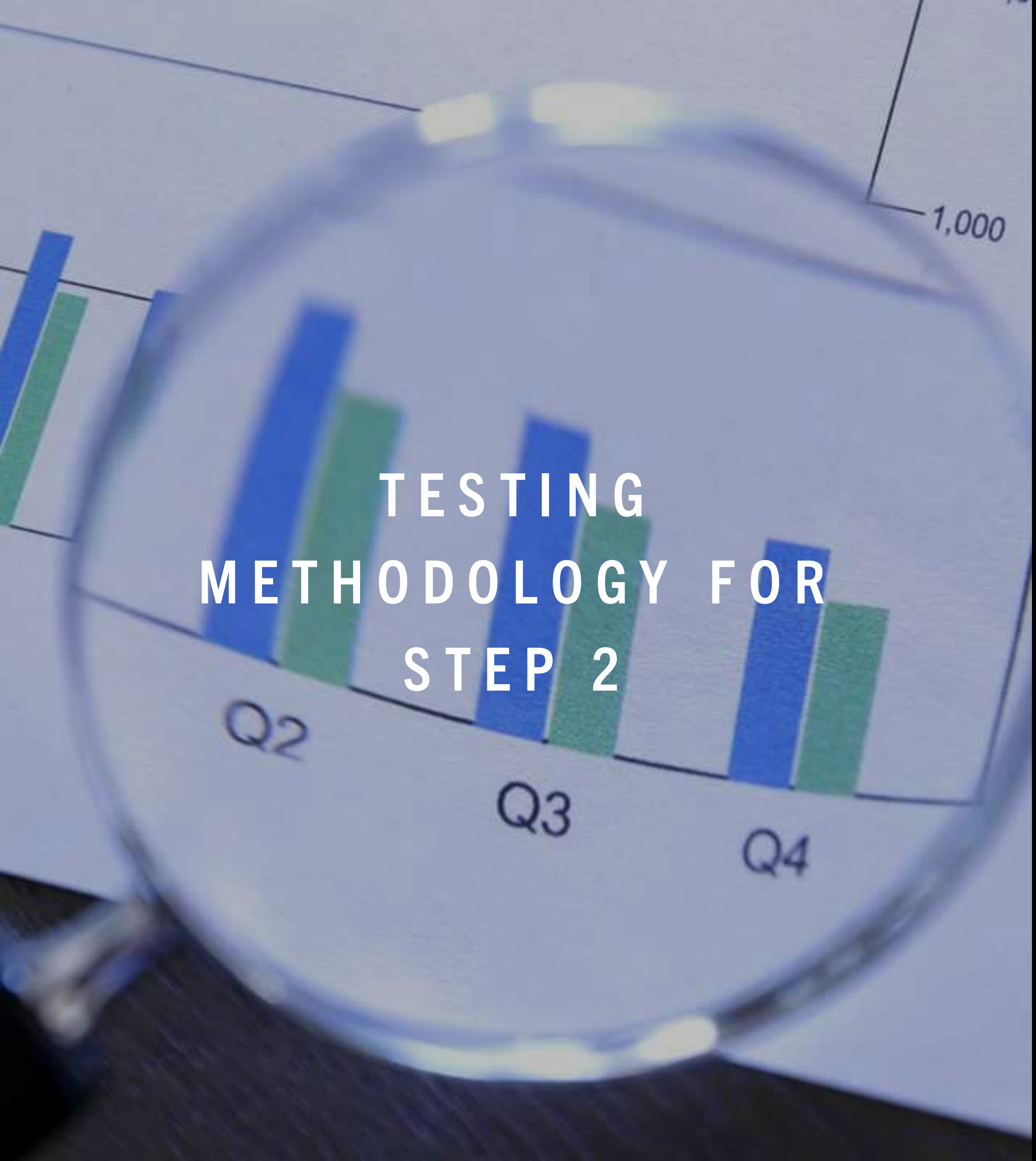
Unauthorized Access: guess_passwd, ftp_write, imap, phf, multihop, warezmaster, warezclient, spy, xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named, mailbomb, buffer_overflow, loadmodule, rootkit, perl, sqlattack, xterm, ps

Normal: normal

STEP 2: MACHINE LEARNING WITHOUT BIAS

- Used special technique to change the way the model was evaluated
- Weights β were calculated such that the model would have relatively higher value of loss for classes with lower number of samples and vice versa

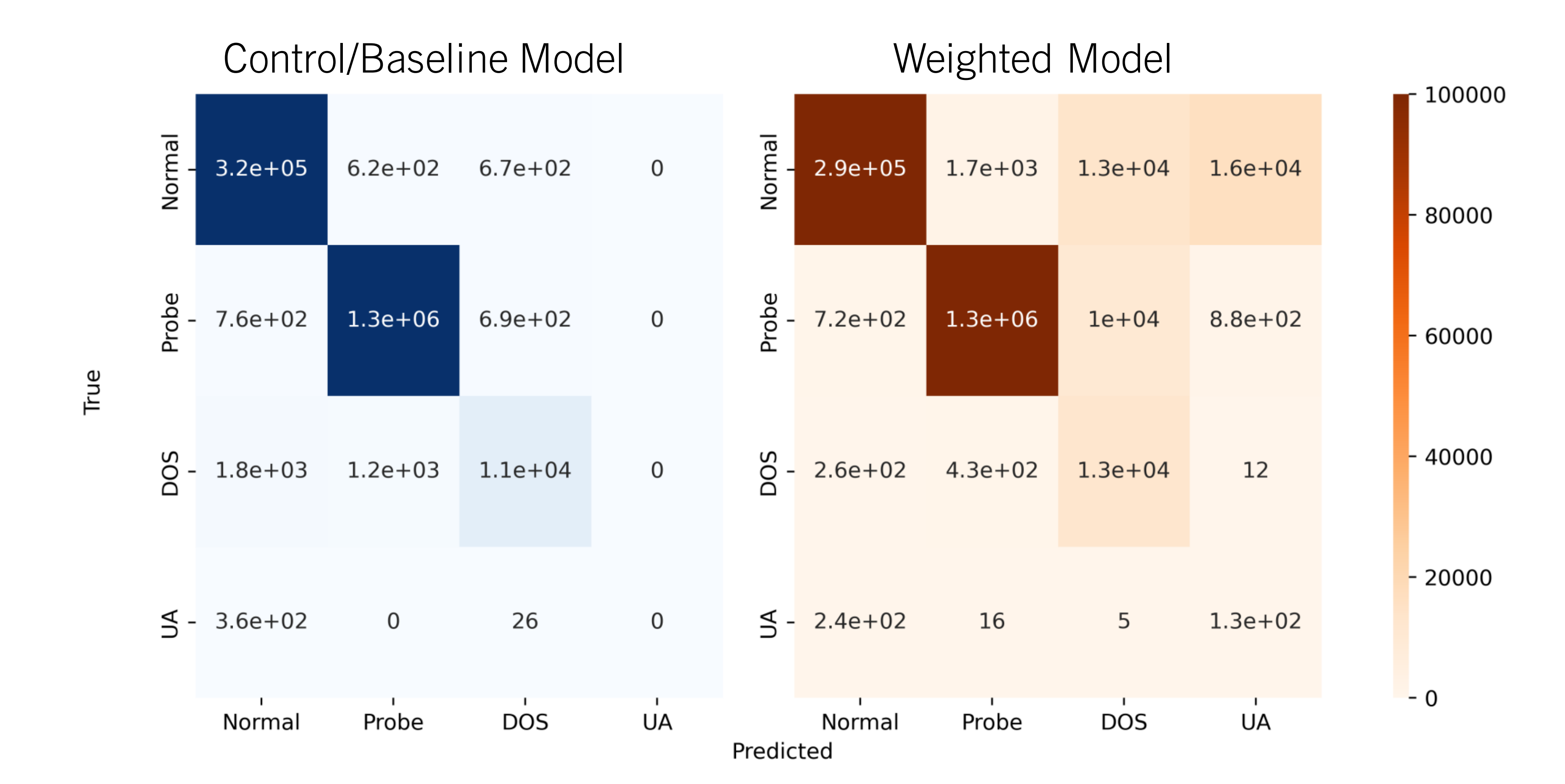
$$H(t,p) \; = \; -\frac{1}{N}\sum_{i=1}^{n} \beta t_i log(p_i) + (1 - t_i)(1 - \beta)\, log(1 - p_i) \tag{2}$$



TESTING METHODOLOGY FOR STEP 2

- Two Machine Learning models were trained for the grouped dataset
- Control model didn't use weighted loss and experimental model used
- Confusion matrix and Classification report show interesting results in next pages

RESULTS


Control/Baseline Model
Weighted Model
True
Predicted
Normal
Probe
DOS
UA
3.2e+05
6.2e+02
6.7e+02
0
7.6e+02
1.3e+06
6.9e+02
0
1.8e+03
1.2e+03
1.1e+04
0
3.6e+02
0
26
0
2.9e+05
1.7e+03
1.3e+04
1.6e+04
7.2e+02
1.3e+06
1e+04
8.8e+02
2.6e+02
4.3e+02
1.3e+04
12
2.4e+02
16
5
1.3e+02
100000
80000
60000
40000
20000
0

METRICS EVALUATION

Class	Control Model			Weighted Model			support
	precision	recall	f1-score	precision	recall	f1-score	
Normal	0.9908	0.996	0.9934	0.9958	0.9023	0.9468	321018
probe	0.9986	0.9989	0.9987	0.9983	0.9907	0.9945	1281513
DOS	0.8842	0.7773	0.8273	0.3507	0.9482	0.512	13563
Unauthorized Access	1	0	0	0.0076	0.3368	0.0149	389
accuracy	0.9962			0.9726			0.9726
macro avg	0.9684	0.693	0.7048	0.5881	0.7945	0.617	1616483
weighted avg	0.9961	0.9962	0.996	0.9921	0.9726	0.9807	1616483

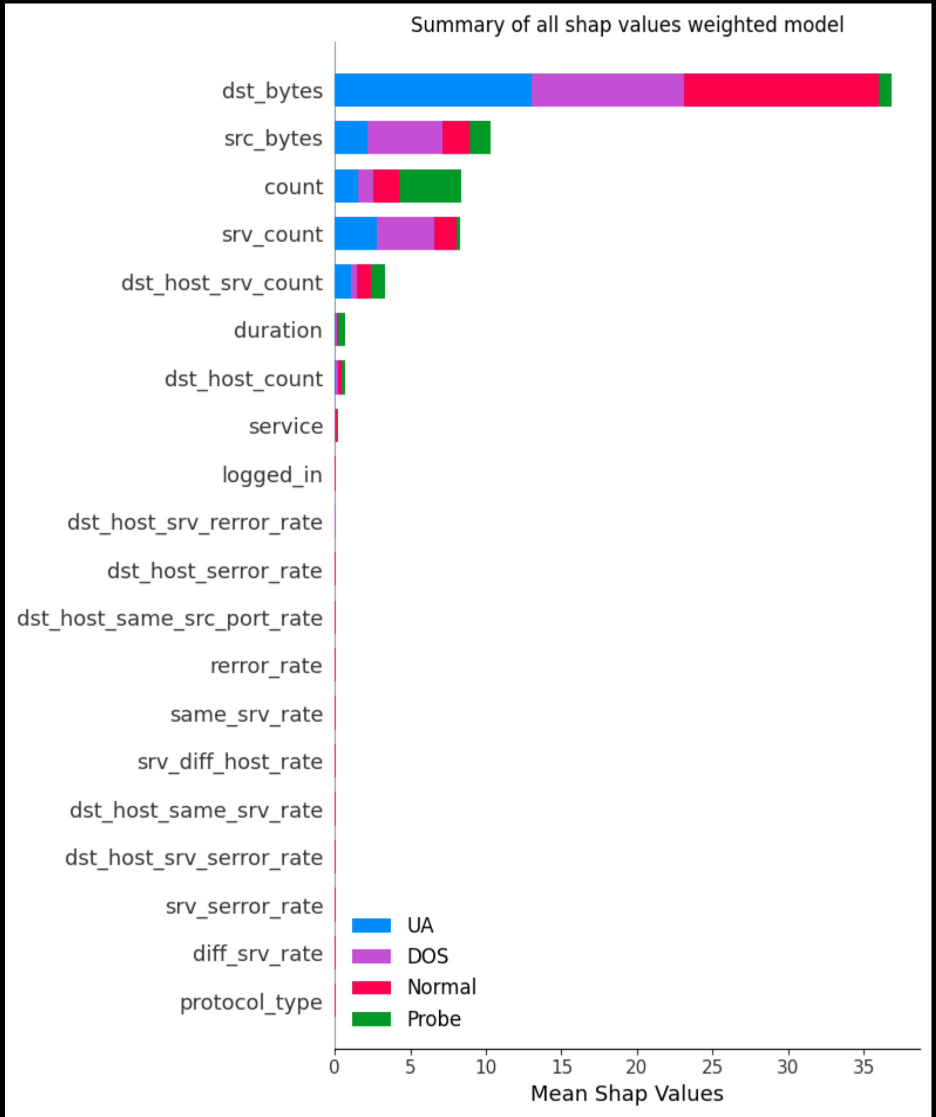
HOW EXACTLY DOES THE MODEL DECIDE?

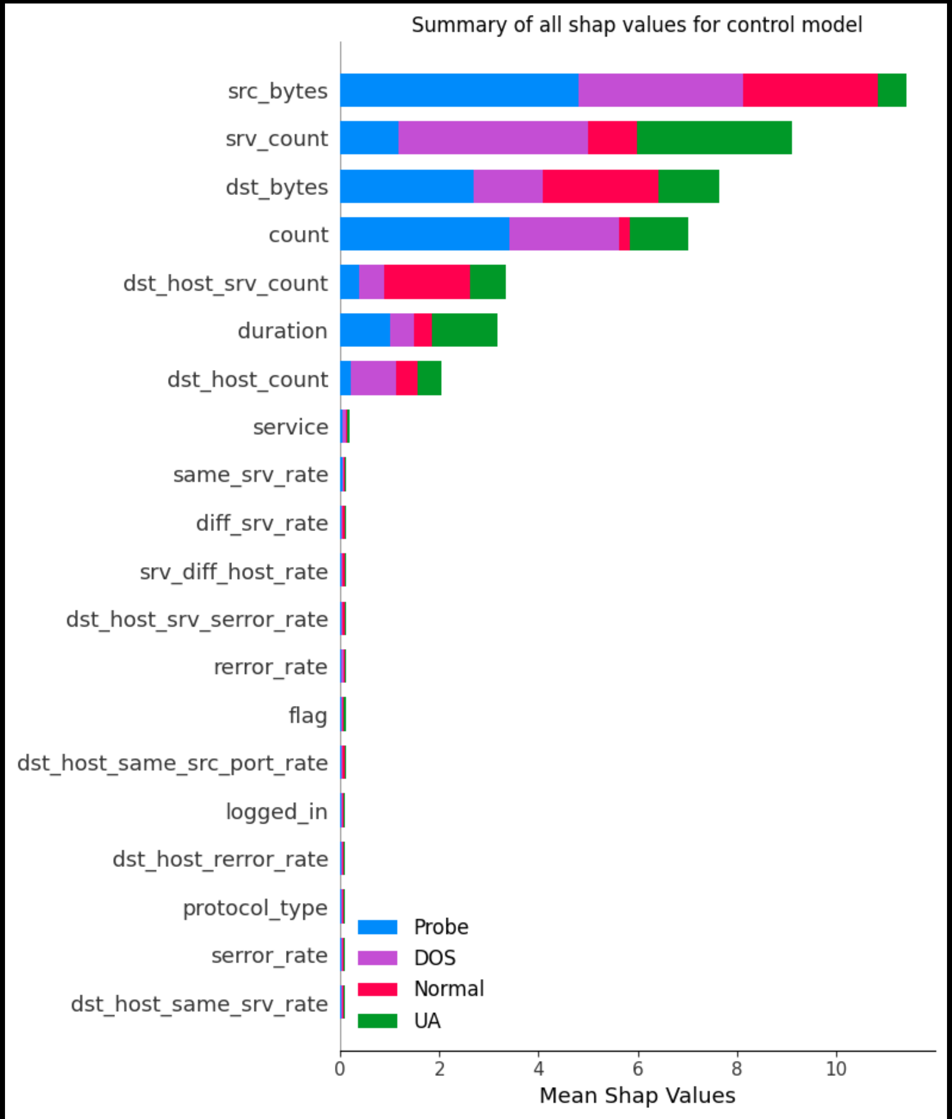
SHAP (SHapley Additive exPlanations) is a game theoretic approach to explain the output of any machine learning model.

Random 50 samples from the output were choose from the validation dataset to be sent to a SHAP explainer

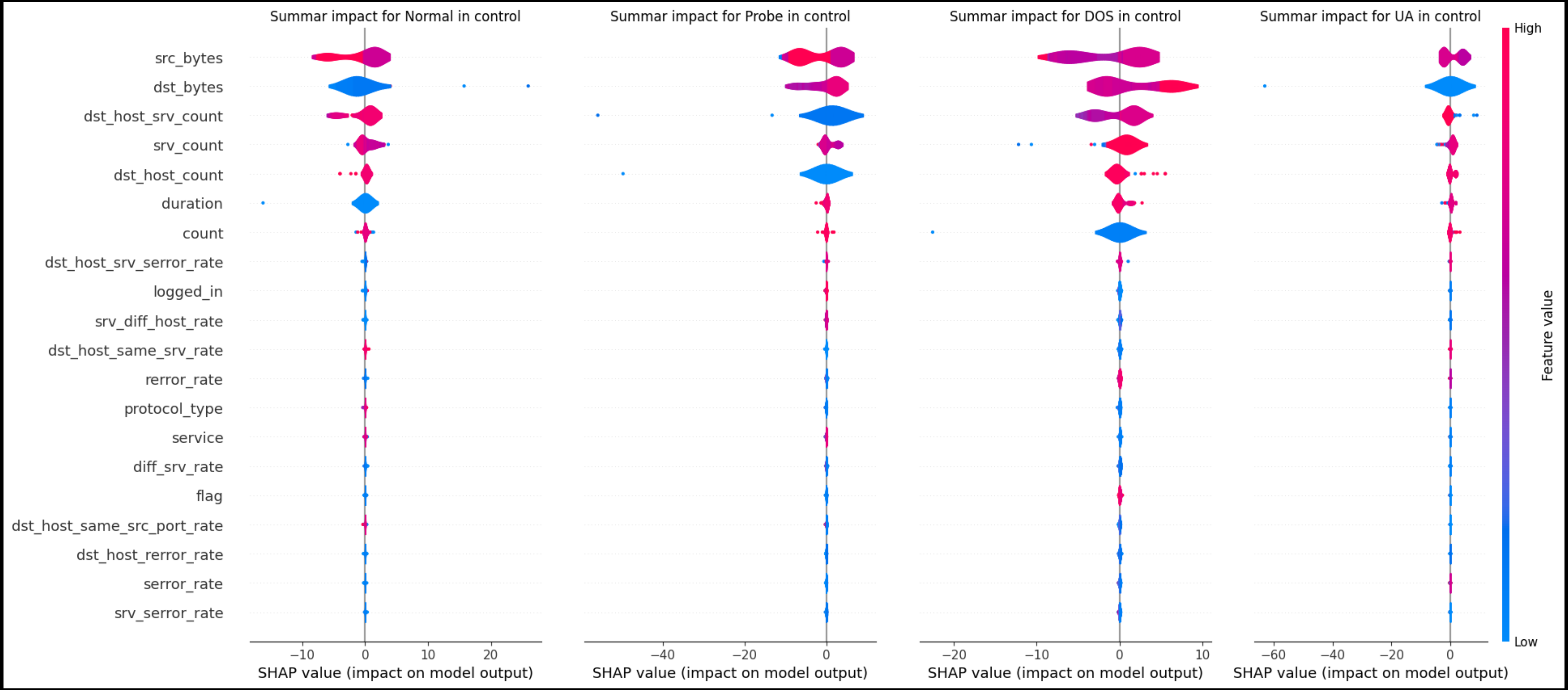
The Shap Explainer gave insight into how and why did the model made its decisions discussed in the next paragraphs

SHAP COMPARISIONS

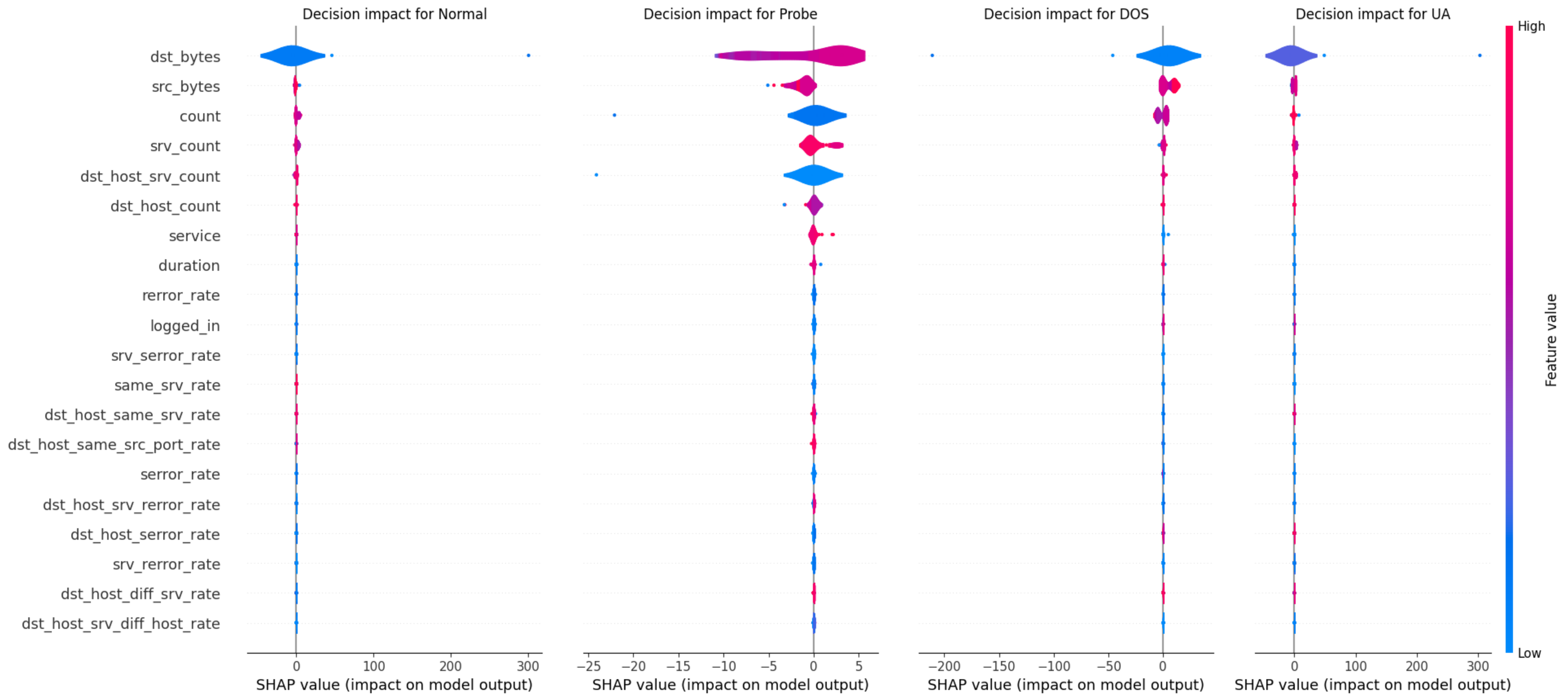




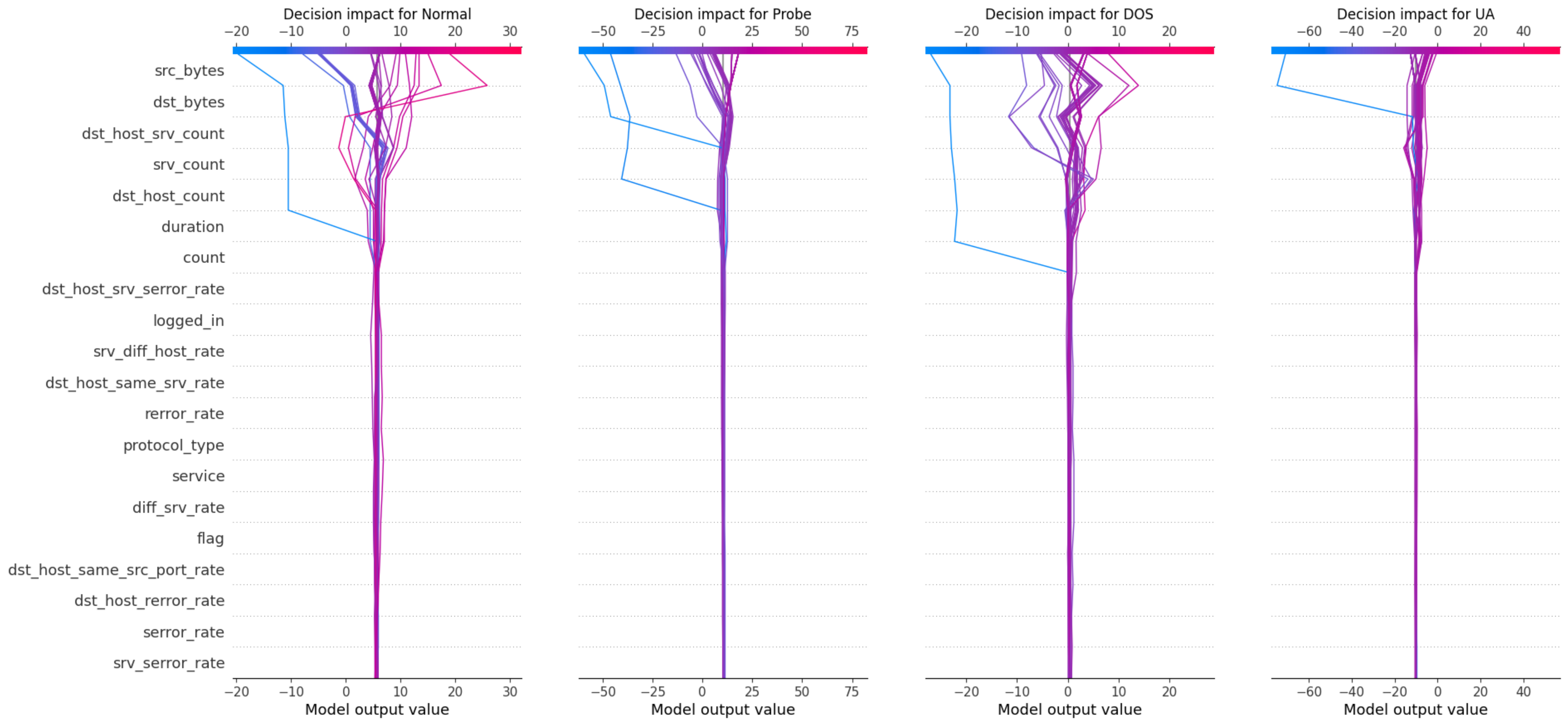
SHAP COMPARISIONS: SUMMARY IMPACT CONTROL



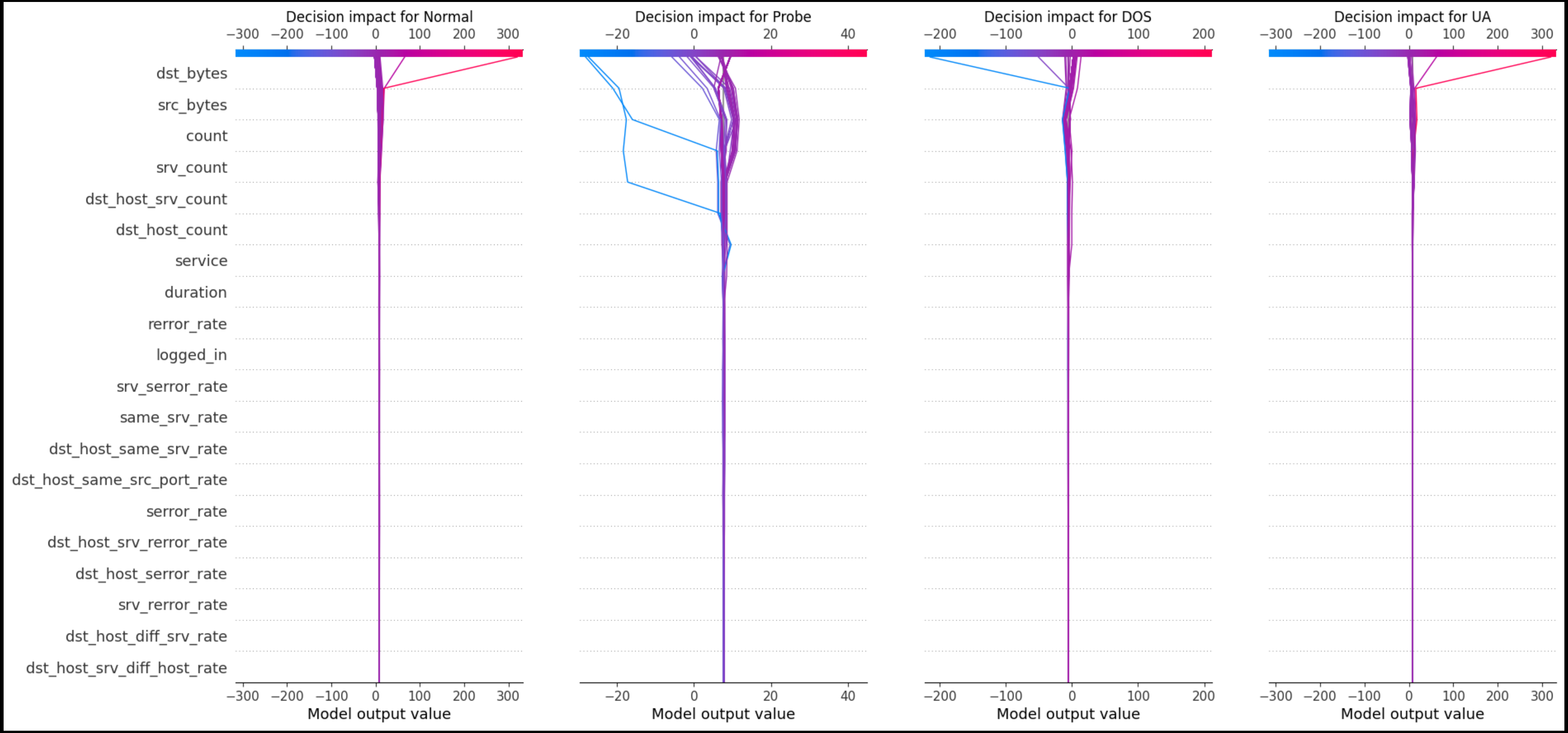
SHAP COMPARISIONS: SUMMARY IMPACT WEIGHTED



SHAP COMPARISION: DECISION IMPACT CONTROL



SHAP COMPARISION: DECISION IMPACT CONTROL



CONCLUSIONS AND FUTURE WORK

Accuracy can be deceiving

Weighted loss can be a strong method to tackle a biased dataset

An entire classification report should be preferred above score reports in ML model evaluation

Learn a meta model to analyze the result from both models to produce even stronger Intrusion Detection Systems

THANK YOU

- Code on : github.com/ashimdahal/

- Project part of Cyber Innovations Lab's continuation work

- In parts compiled by Prabin Bajgai