

Detecting Malicious Attacks Through The Network Traffic

SDAIA BootCamp T5

Presented by:

Hatim Alshehri & Fahad Alnafisa



Methodology

- Problem Statement
- Data Analysis
- Data Pre-processing and feature engineering
- Classification models
- Conclusion





Problem Statement

With the rise of Internet usage, it is very important to protect Networks. The most common risk to a network's security is an intrusion such as brute force, denial of service or even an infiltration from within a network.

Problem Statement



Detecting Intrusion



network



Problem Statement

Data set

- UNSW-NB15
- +170k rows
- UNSW Sydney
- 45 columns

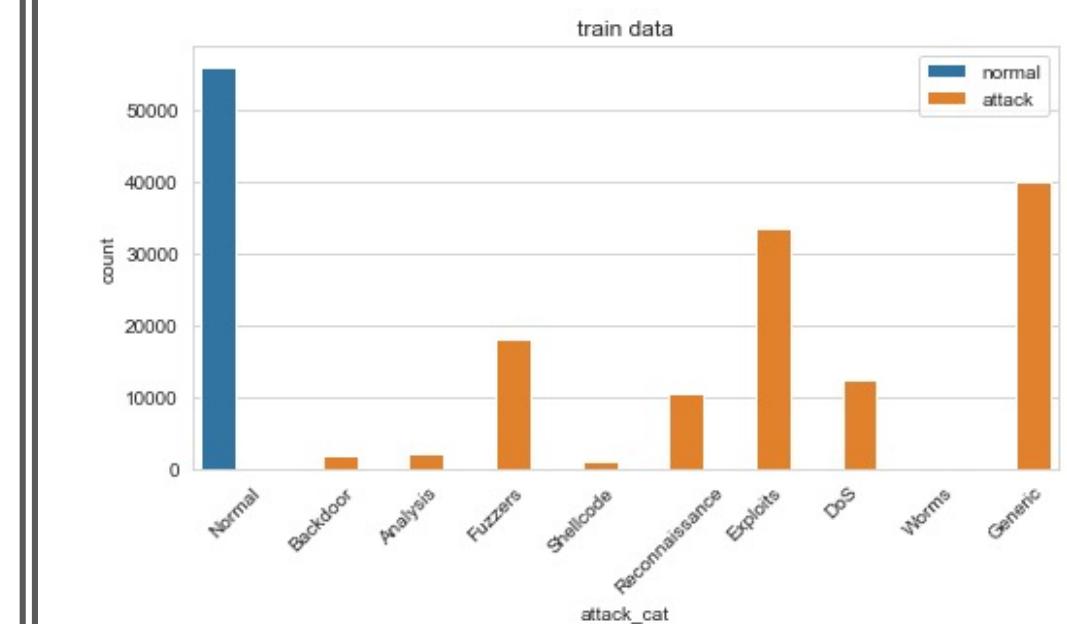
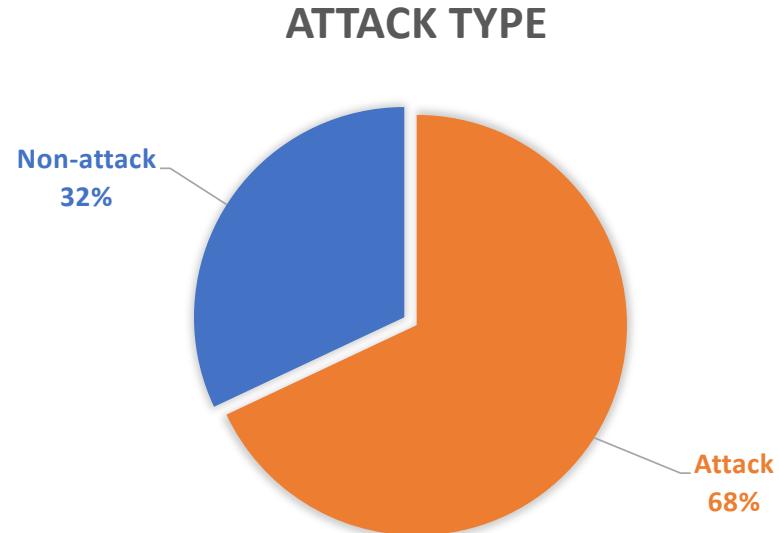
Tools

- Pandas
- Sklearn
- Numpy
- XGBoost
- Matplotlib
- Pickle
- Seaborn
- Prettytable



The background image shows a server room with rows of server racks. Overlaid on the scene is a glowing blue network graph. The graph consists of numerous white circles of varying sizes connected by thin white lines, forming a complex web of connections. Some nodes are brightly lit, while others are smaller dots. The overall effect is one of data flow and connectivity within a technological environment.

Data analysis



Data Analysis

Data Pre-processing and feature engineering

```
mirror_mod = modifier_obj.modifiers[0]
# Set mirror object to mirror
mirror_mod.mirror_object = context.object

operation = "MIRROR_X":
    mirror_mod.use_x = True
    mirror_mod.use_y = False
    mirror_mod.use_z = False
operation == "MIRROR_Y":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

#selection at the end -add
modifier_obj.select= 1
modifier.select=1
context.scene.objects.active = modifier
print("Selected" + str(modifier))
modifier.select = 0
bpy.context.selected_objects.append(modifier)
data.objects[one.name].select = 1
print("please select exactly one object")

- OPERATOR CLASSES -

```

```
types.Operator:
    bl_idname = "MIRROR_OT_MIRROR"
    bl_label = "X mirror to the selected object.mirror_mirror_x"
    bl_options = {'REGISTER', 'UNDO'}
```

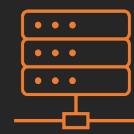
```
def execute(self, context):
    if context.object is None:
        print("context.active_object is not set")
```

Data Pre-processing and feature engineering

Cleaning



Remove
nan-values



Convenient
data type



Data
reformation



Data Pre-processing and feature engineering

Feature Engineering

✗
High correlation

⊕
New Features

Scaling Encoding
①

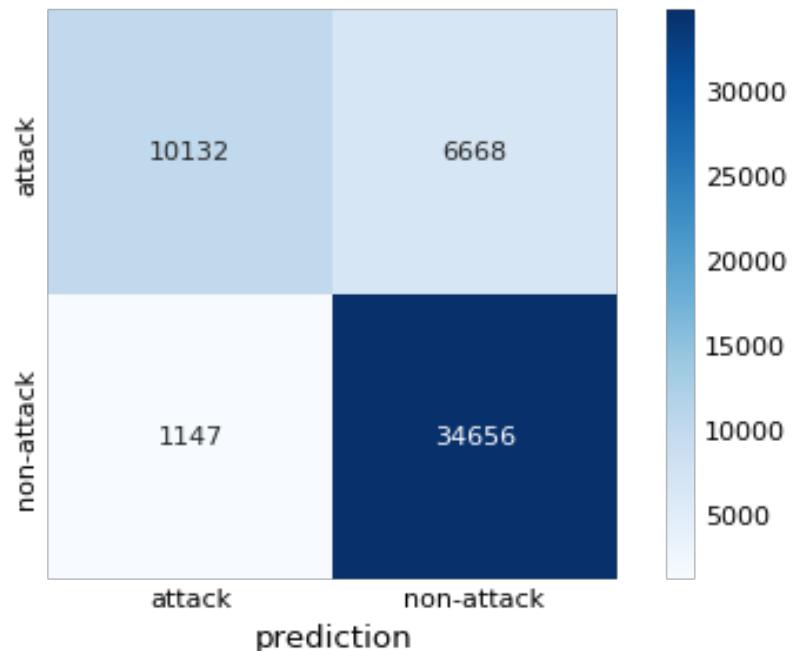


MACHINE LEARNING

Binary Classification Models

Baseline Model

Logistic Regression



Accuracy		Precision		Recall	
Train	Validation	Train	Validation	Train	Validation
0.7865	0.7855	0.8395	0.8386	0.9669	0.9679

MODELS RESULT

MODEL	ACCURACY		F1-SCORE		FALSE ALARM RATE	
	TRAIN	VALIDATION	TRAIN	VALIDATION	TRAIN	VALIDATION
LOGISTIC REGRESSION	0.9052	0.9021	0.9544	0.9534	0.0947	0.0978
K-NEAREST NEIGHBORS	0.9173	0.9100	0.9591	0.9554	0.0826	0.0899
DECISION TREE	0.8926	0.8882	0.9517	0.9500	0.1073	0.1117
RANDOM FOREST	0.9143	0.9028	0.9612	0.9560	0.0856	0.0971
XGBOOST	0.9669	0.9391	0.9819	0.9665	0.0330	0.0608

ENSEMBLING

MODELS

- LOGISTIC REGRESSION
- RANDOM FOREST
- K-NEAREST NEIGHBORS
- EXTRA TREE

MODEL	ACCURACY		F1-SCORE		FALSE ALARM RATE	
	TRAIN	VALIDATION	TRAIN	VALIDATION	TRAIN	VALIDATION
VOTING CLASSIFIER (HARD)	0.9990	0.9500	0.9992	0.9641	0.0008	0.0656
VOTING CLASSIFIER (SOFT)	0.9990	0.9489	0.9992	0.9637	0.0014	0.0729
STACKING	0.9990	0.9502	0.9992	0.9644	0.0008	0.0664

BEST CLASSIFICATION MODEL

VOTING CLASSIFIER (HARD)

ACCURACY

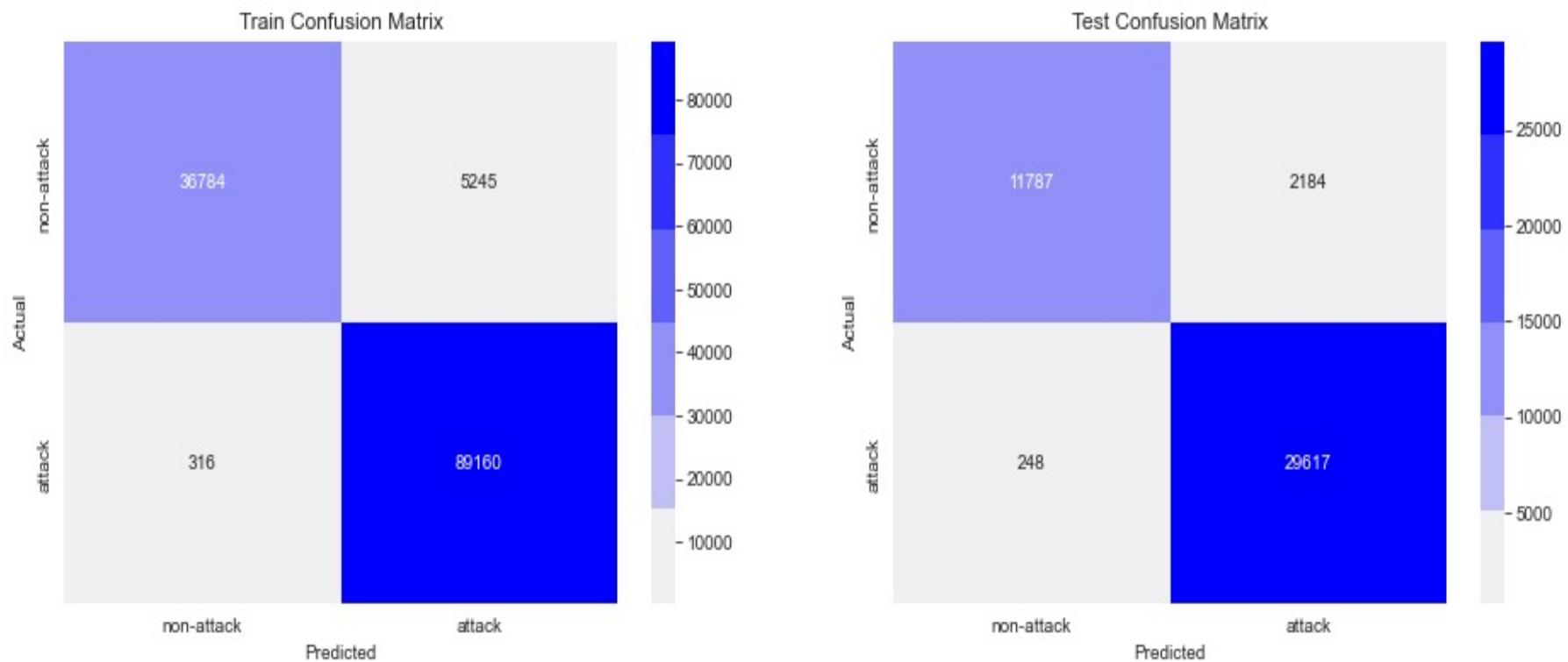
F1

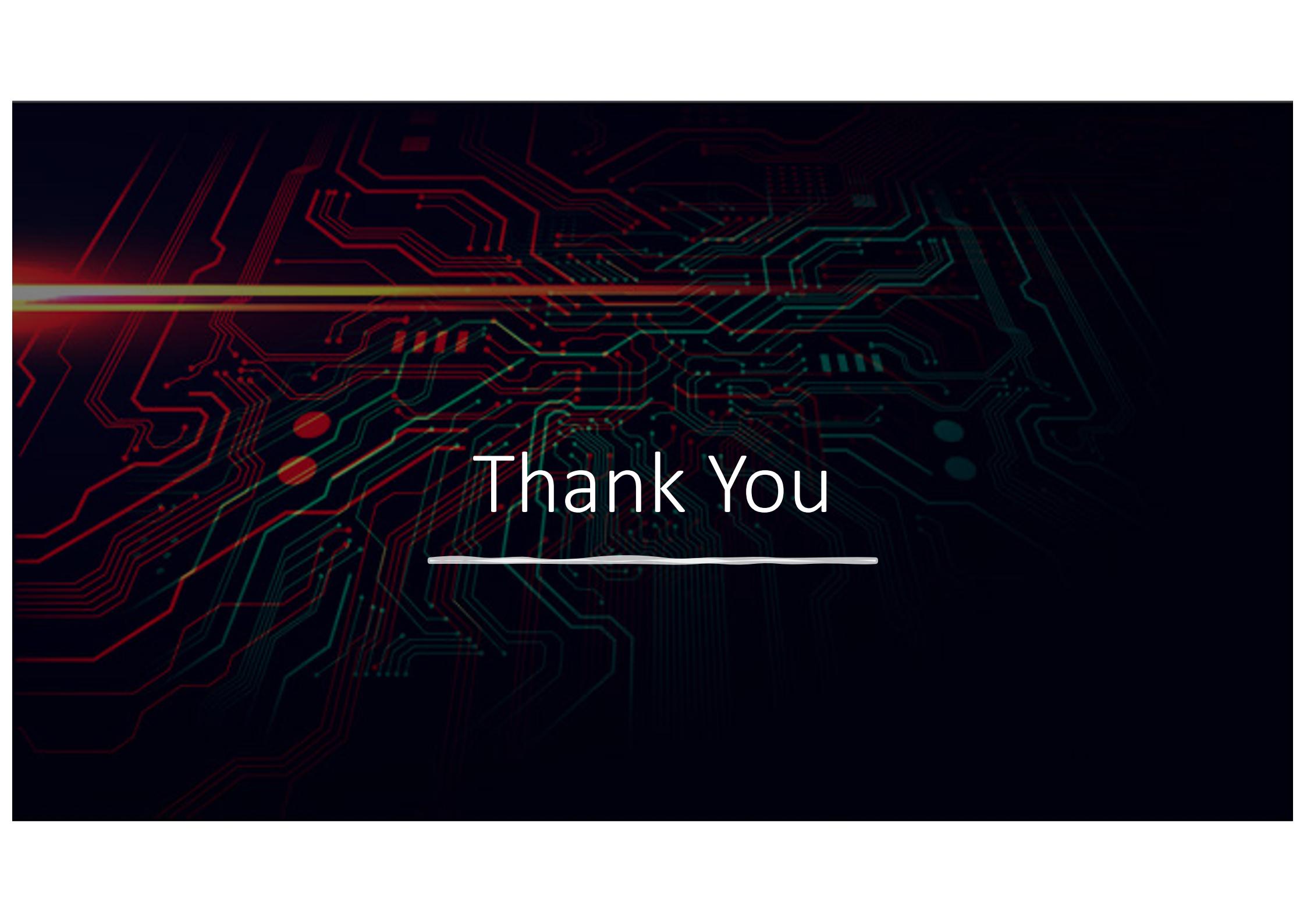
PRECISION

RECALL

TRAIN	TEST	TRAIN	TEST	TRAIN	TEST	TRAIN	TEST
0.9559	0.9431	0.9685	0.9596	0.9420	0.9291	0.9965	0.9921

VOTING CLASSIFIER (HARD)





A detailed circuit board pattern is visible in the background, featuring a dense network of red and green conductive traces on a dark blue-green background. The pattern is highly symmetrical and organic in shape, resembling a brain or a complex network. In the center of this pattern, the words "Thank You" are written in a clean, white, sans-serif font. A thin horizontal white line extends from the bottom of the "Thank You" text towards the left, ending in a small white circle.

Thank You