Final Year Project 2

COLLEGE OF COMPUTING AND INFORMATICS

Network Intrusion Detection Model Based on Real and Encrypted Synthetic Attack Traffic using Decision Tree Algorithm

Name : MIKHAIL AMZAR BIN KAMARUDDIN
Student ID : CS0107477
Supervisor : MD NABIL BIN AHMAD ZAMAWI, TS

Contents

1. Introduction

Network Intrusion Detection Model Based on Real and Encrypted Synthetic Attack Traffic using Decision Tree Algorithm

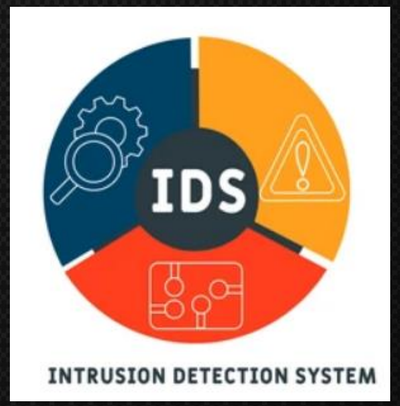
Problem Statement/Motivation:

Develop a network intrusion detection model that analyze attack traffic using datasets of real and encrypted synthetic attack traffic. The model will use a decision tree algorithm and will provide user with a dashboard containing relevant data from the model.

Objectives

1. To design a network intrusion detection model that applies Decision Tree algorithm.

2. To produce a machine learning model and dashboard for network intrusion detection model.



Scope

User Scope

Users using the model and dashboard to observe metrics of the model and its related details.

System Scope

- Implementing decision tree machine learning algorithm.
- The system will only perform intrusion detection.
- Dataset that will be used for the training of the algorithm is a publicly available dataset – ALLFLOWMETER_HIKARI2021.csv.

Expected Outcomes

1

A machine learning model and dashboard must be produced for the network intrusion detection model.

2

The network intrusion detection model should apply Decision Tree algorithm.

2. Literature Review

1

Network Intrusion Detection System

- Network-Based IDS
- Host Based IDS

Detection methods:

- Signature-based Detection
- Anomaly-based Detection

2

Machine Learning

Subfield of Artificial Intelligence

Supervised learning
Unsupervised learning

<u>Machine Learning Modelling Cycle</u>

- Manage data
- Train Model
- Evaluate Model
- Deploy Model

Decision Tree Algorithm

3

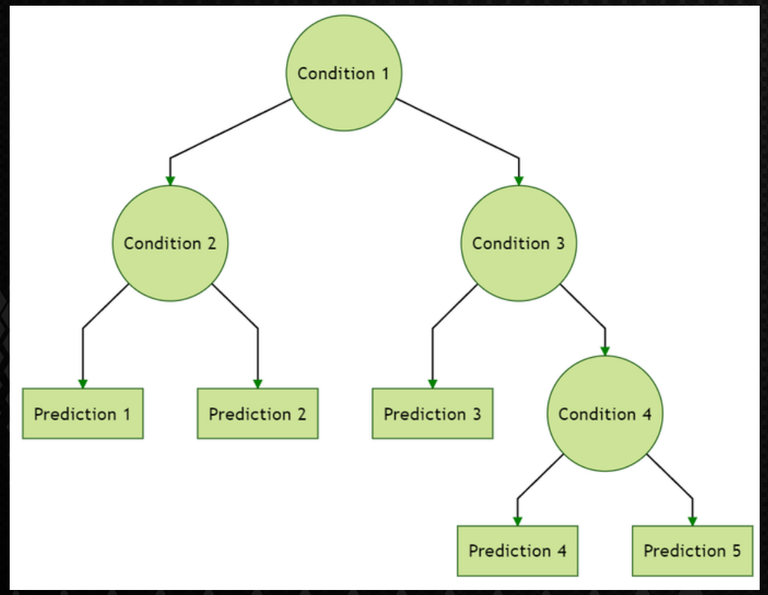
Software Development Methodology

- Waterfall methodology
- Agile methodology

Decision Tree Algorithm

Machine Learning Algorithm for :

- Classification task
- Regression task



Formed in a tree-like structure with branches.

A type of supervised learning. Uses labelled input and output datasets.

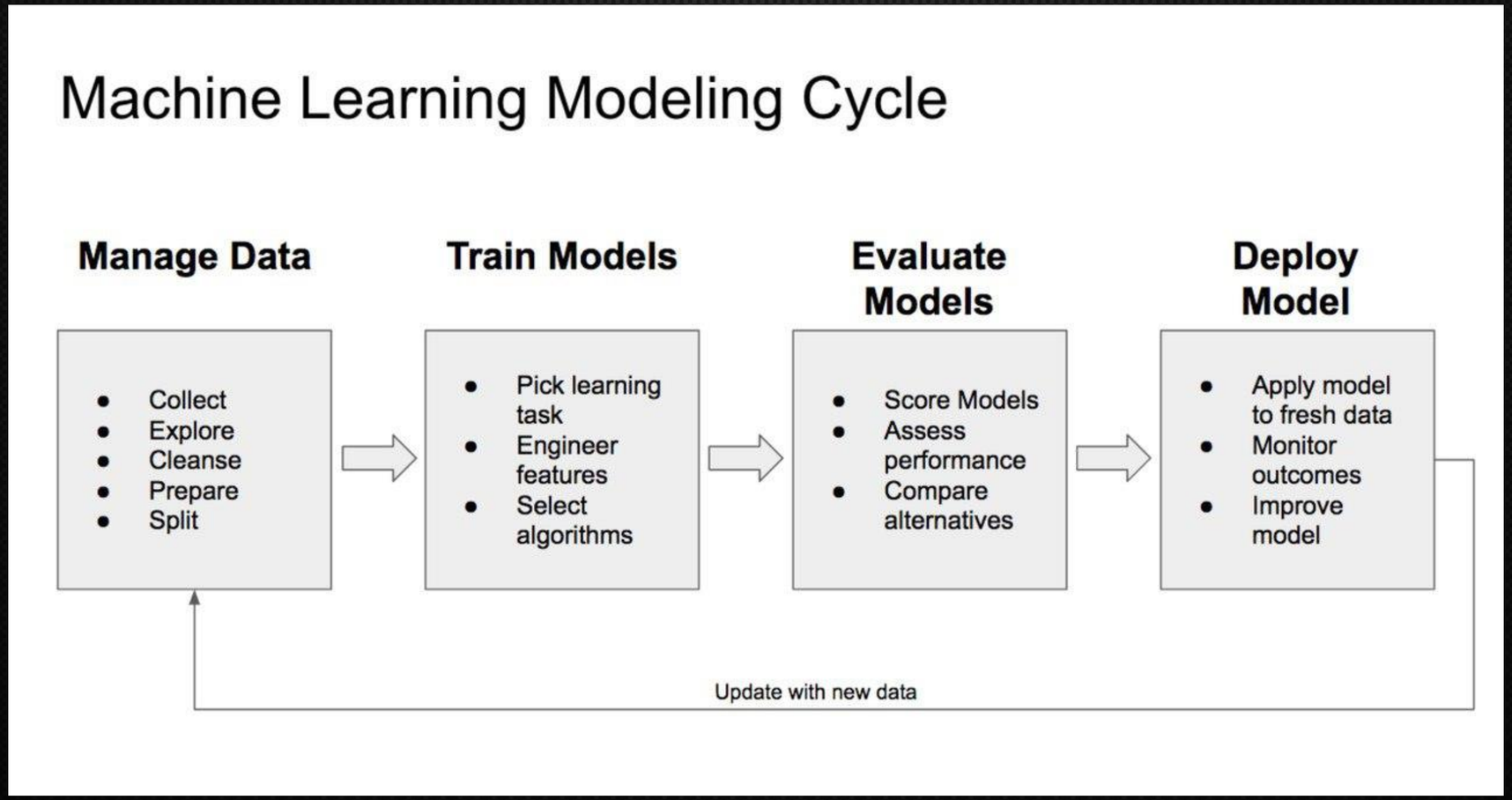
Mainly used to solve **classification problem.**

- Categorise or classify an object.

Classification tree	Regression Tree
The model is trained to classify whether data is a part of a known object class. Models are trained to assign class labels to processed data.	Models are designed to forecast or predict a continuous value, such as predicting house prices or stock price changes.

Machine Learning

Modelling Cycle



Machine learning tools

Weka

Anaconda

Google Colab

Dashboard tools

Grafana

.NET

neptune.ai

3. Methodology

Agile

Flexible, iterative approach to software development that emphasizes :

- rapid prototyping
- frequent releases
- continuous improvement.

Machine learning tool

Google Colab

- Easy setup
- Access to hardware via Google Cloud
- Interactive document

System Features

- Providing Decision Tree Model able to identify malicious traffic
- Logging evaluation metrics using a dashboard for observation.

Dashboard tool

Neptune.ai

- Quick integration
- Can store many runs and model logs

Dataset

HIKARI-2021

- Recent dataset
- Real and Encrypted Synthetic Attack Traffic

3. Methodology

Dataset HIKARI-2021

Contains a variety of traffic types.

- Background
- Benign
- Bruteforce
- Bruteforce-XML
- Probing
- XMRGICC CryptoMiner

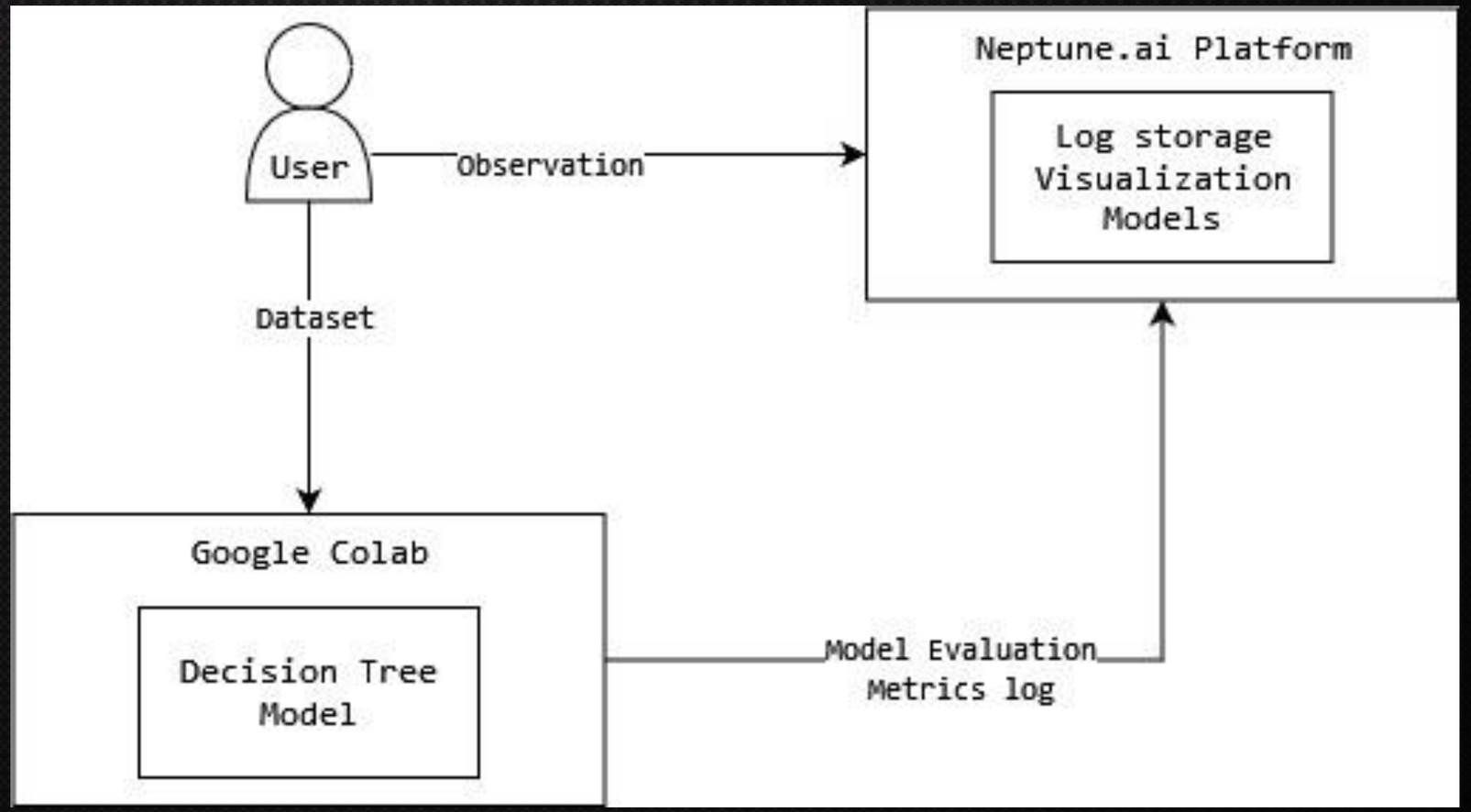
Contains ground truth data from real production network (Background)

Traffic completely labelled with target columns traffic_category and Label

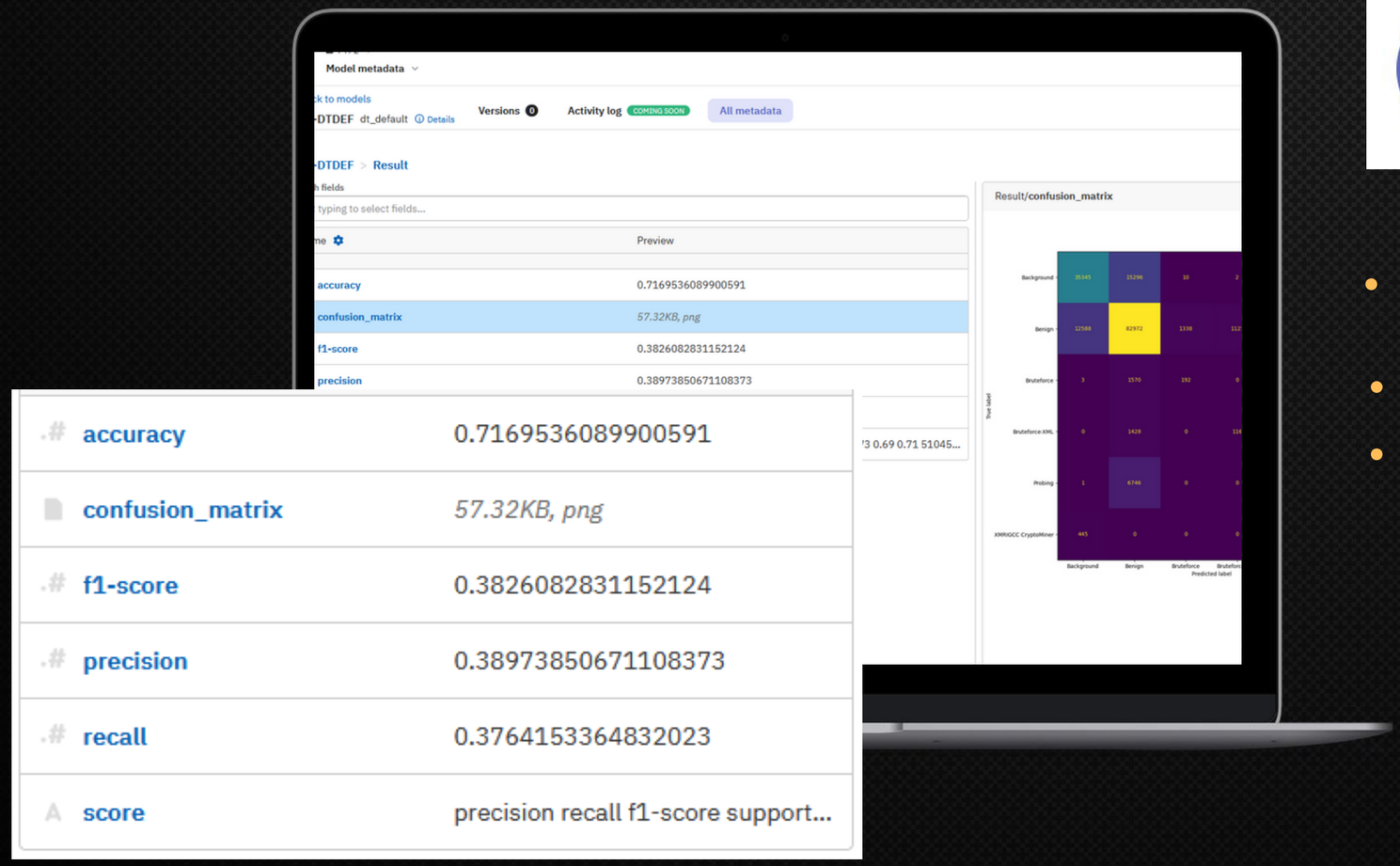
4. Design

System Architecture

- User interaction with machine learning model through Google Colab.
- User interaction with Dashboard.



Dashboard Design



Neptune.ai

- Fast to integrate
- Data logging features.
- Collaborate with other users in a team working on a project.

5. Implementation

Data preprocessing in Google Colab

- Import libraries
  - sklearn
  - matplotlib
  - pandas

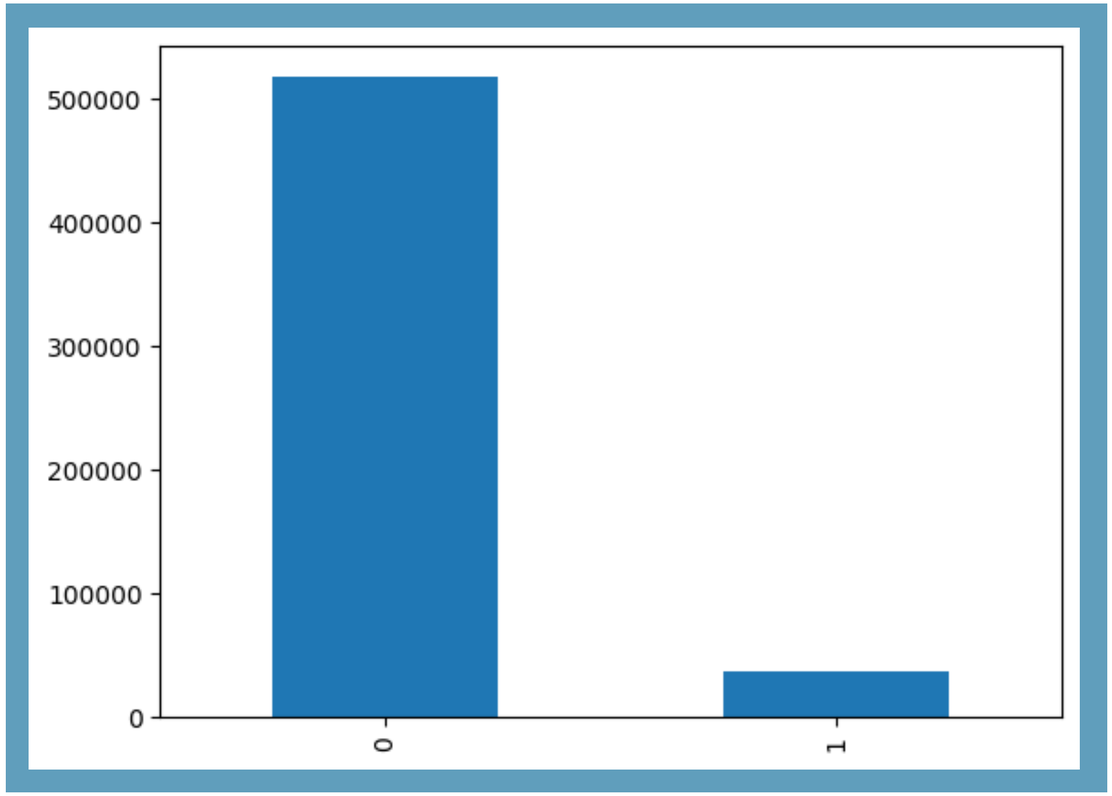
2. Import dataset into Colab & Prepare dataset
  - Visualizing the data
  - Checking for null values

Importing important libraries and Preparing the data

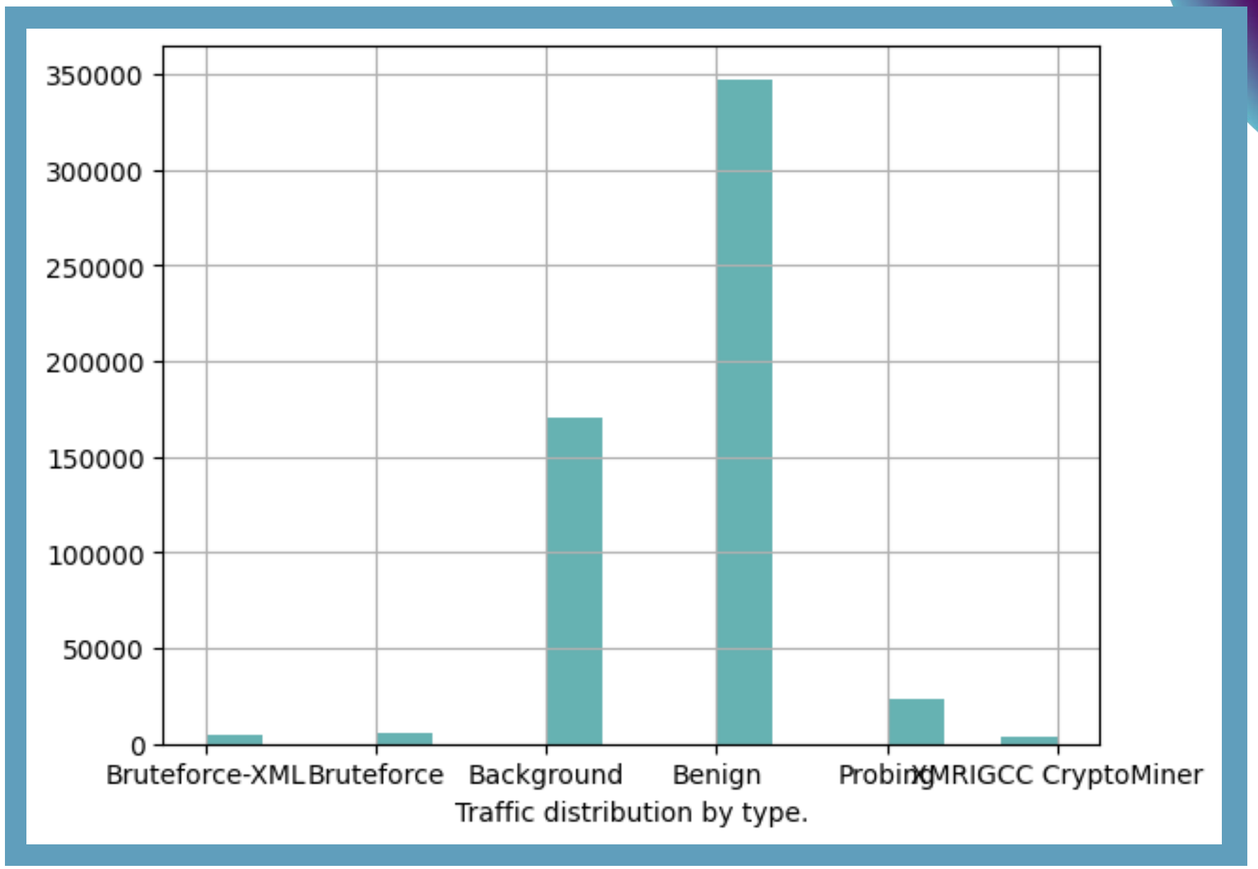
- Importing dataset file from Google Drive.
- Preprocess the dataset.

```
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from io import StringIO


from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
pd.options.display.max_columns = None
pd.options.display.max_rows = None


# Import dataset
from google.colab import drive
drive.mount('/content/drive/')
df = pd.read_csv("/content/drive/MyDrive/ALLFLOWMETER_HIKARI2021.csv")
df.head(1)
```

Data preprocessing



Malicious (1) vs Non-Malicious (0)



Traffic distribution by Type

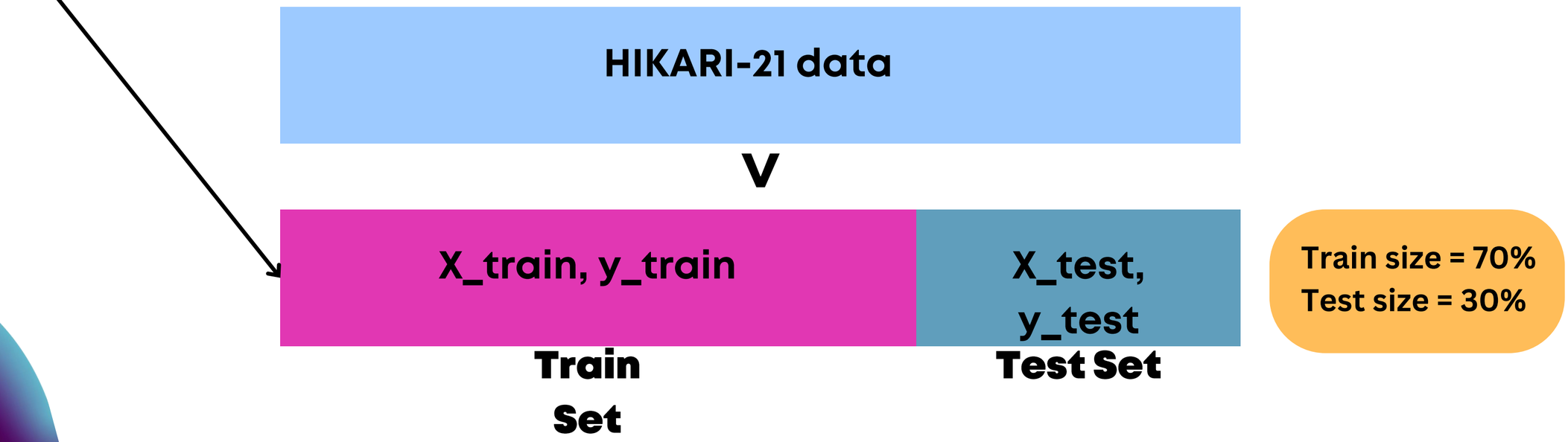
Split train & test

```
# Assigning features and target to an object

X = df.drop(['Label','traffic_category'], axis=1)
y = df.traffic_category
```

Isolating the features columns (X) and target (y).

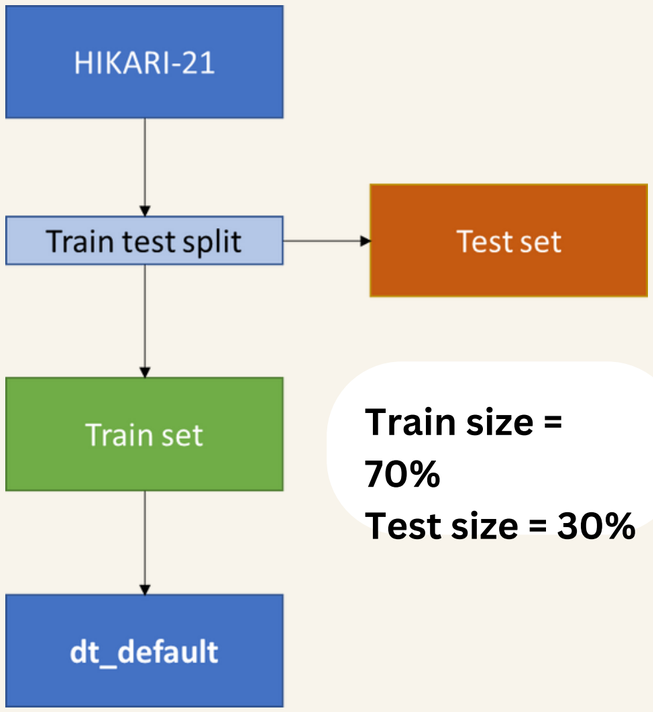
- X is the features columns.
- y is the target class column.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,stratify=y)
```



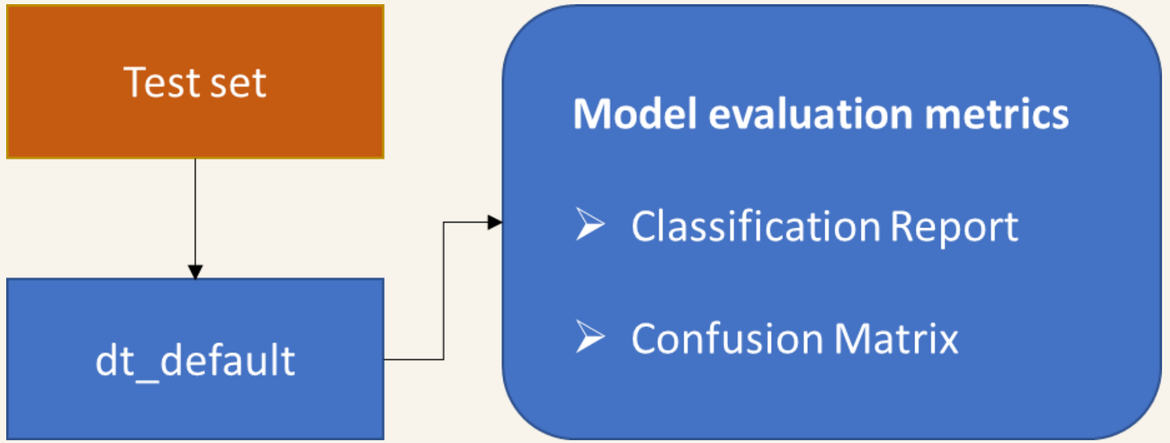
Workflow

Training flow



Train size = 70%
Test size = 30%

Testing flow



Model evaluation metrics transmit to Neptune.ai for logging.

Classification Report

dt_default Decision tree model classification.

Accuracy : 0.72
Precision : 0.39
Recall : 0.38
F1 - score : 0.38

HIKARI-21 is an Imbalanced dataset. Accuracy will not be used to measure the model's performance.

```
from sklearn.metrics import classification_report
report = classification_report(y_test, y_pred)
print(report)
```

	precision	recall	f1-score	support
Background	0.73	0.69	0.71	51045
Benign	0.77	0.80	0.78	104230
Bruteforce	0.12	0.11	0.12	1765
Bruteforce-XML	0.09	0.08	0.08	1544
Probing	0.04	0.04	0.04	7016
XMRIGCC CryptoMiner	0.58	0.55	0.56	984
accuracy			0.72	166584
macro avg	0.39	0.38	0.38	166584
weighted avg	0.71	0.72	0.71	166584

Confusion matrix : dt_default

TARGET / OUTPUT	Background	Benign	Bruteforce	Bruteforce-XML	Probing	XMRIGCC CryptoMiner	SUM
Background	35345 21.22%	15296 9.18%	10 0.01%	2 0.00%	2 0.00%	390 0.23%	51045 69.24% 30.76%
Benign	12588 7.56%	82972 49.81%	1338 0.80%	1125 0.68%	6207 3.73%	0 0.00%	104230 79.60% 20.40%
Bruteforce	3 0.00%	1570 0.94%	192 0.12%	0 0.00%	0 0.00%	0 0.00%	1765 10.88% 89.12%
Bruteforce-XML	0 0.00%	1428 0.86%	0 0.00%	116 0.07%	0 0.00%	0 0.00%	1544 7.51% 92.49%
Probing	1 0.00%	6746 4.05%	0 0.00%	0 0.00%	269 0.16%	0 0.00%	7016 3.83% 96.17%
RIGCC CryptoMi	445 0.27%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	539 0.32%	984 54.78% 45.22%
SUM	48382 73.05% 26.95%	108012 76.82% 23.18%	1540 12.47% 87.53%	1243 9.33% 90.67%	6478 4.15% 95.85%	929 58.02% 41.98%	119433 / 166584 71.70% 28.30%

Evaluation

Multiclass-classification score comparison.

HIKARI-21 authors conducted basic performance analysis using other ML algorithms such as:

- K- Nearest Neighbour
- Multilayer Perceptron
- Support Vector Machine
- Random Machine

Algorithm	Accuracy	Precision	Recall	F1
KNN	0.98	0.86	0.90	0.88
MLP	0.99	0.99	0.99	0.99
SVM	0.99	0.99	0.98	0.99
RF	0.99	0.99	0.99	0.99
dt_default	0.72	0.38	0.38	0.38

Observation :
Decision tree model do not produce good scores in multiclass-classification problem.

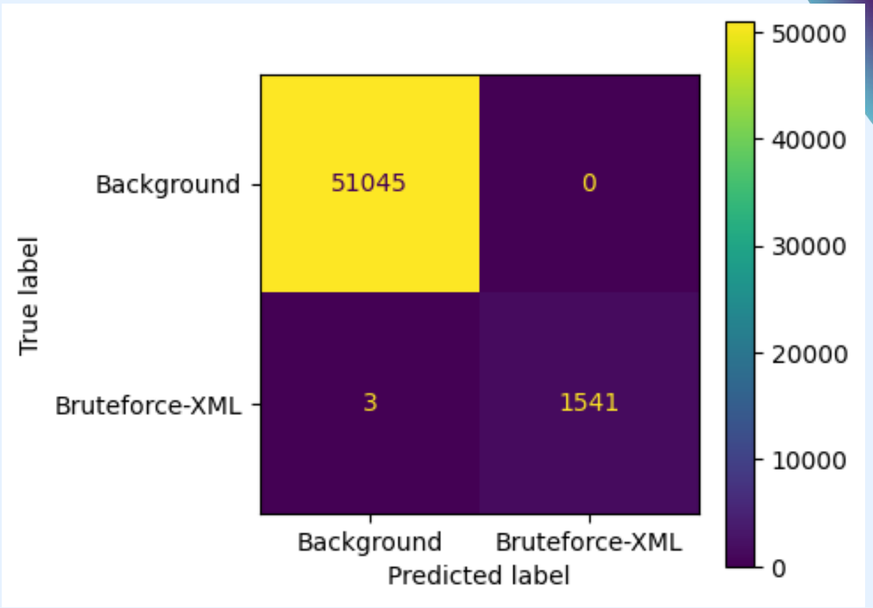
Classification Report

Binary Classification Decision tree model

```
Background        170151
Bruteforce-XML      5145
Name: traffic_category, dtype: int64
```

Only 2 target classes to predict.

Accuracy : 1.00
Precision : 1.00
Recall : 1.00
F1 - score : 1.00




```
from sklearn.metrics import classification_report
report = (classification_report(y_test, y_pred))
print(report)
```

	precision	recall	f1-score	support
Background	1.00	1.00	1.00	51045
Bruteforce-XML	1.00	1.00	1.00	1544
accuracy			1.00	52589
macro avg	1.00	1.00	1.00	52589
weighted avg	1.00	1.00	1.00	52589

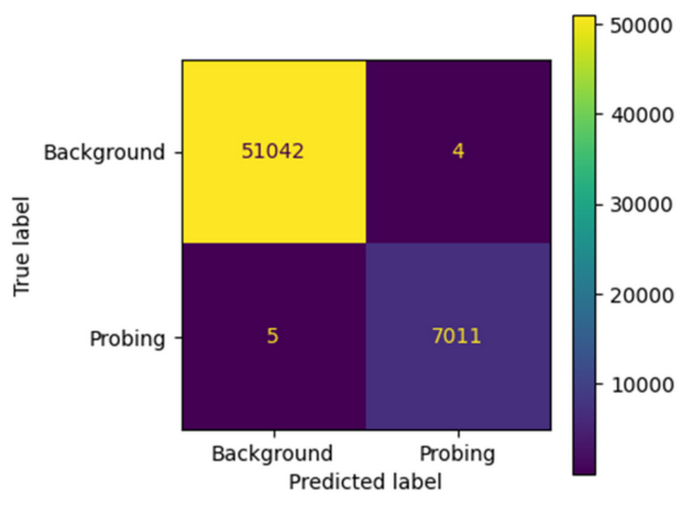
Classification Report

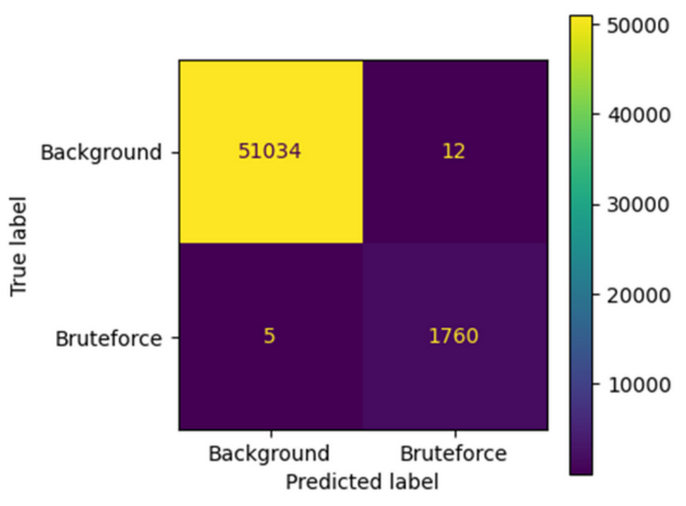
Binary Classification Decision tree model

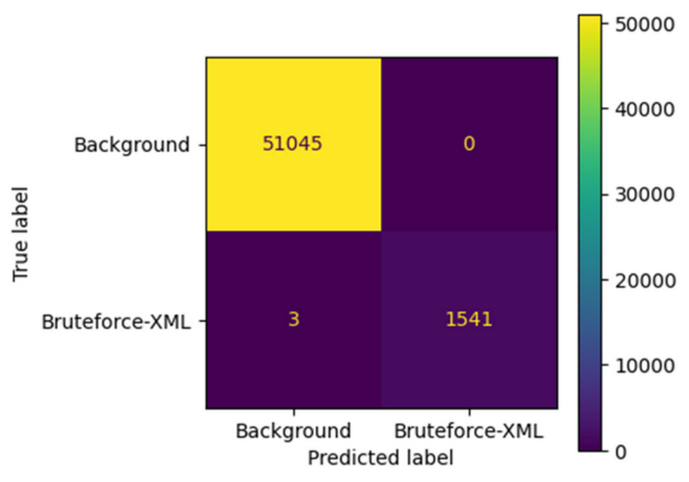
Model	**Accuracy**	**Precision**	Recall	**F1**
Background & Probing	1.00	1.00	1.00	1.00
Background & Bruteforce	1.00	1.00	1.00	1.00
Background & Bruteforce-XML	1.00	1.00	1.00	1.00
Background & XMRIGCC CryptoMiner	0.98	0.75	0.77	0.76

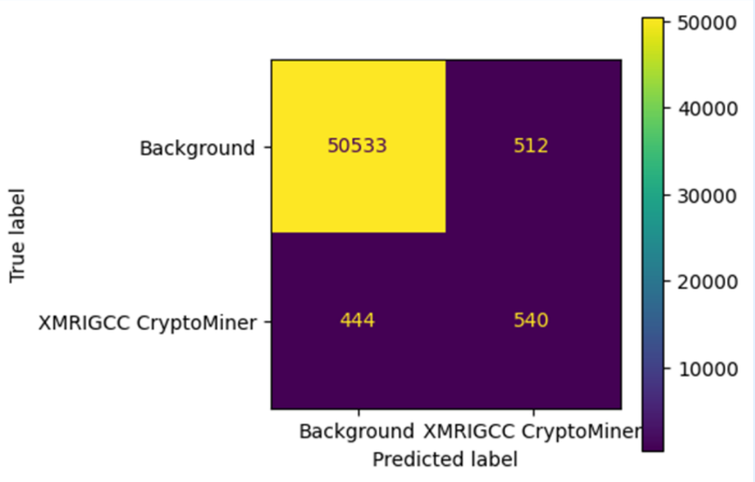
Confusion matrix

Binary Classification Decision tree model









Dashboard: Neptune.ai

Neptune run

```
[9]  from sklearn.metrics import accuracy_score
     from sklearn.metrics import f1_score
     from sklearn.metrics import precision_score
     from sklearn.metrics import recall_score

     acc = accuracy_score(y_test,dt_pred)
     f1 = f1_score(y_test,dt_pred, average='macro')
     pre = precision_score(y_test,dt_pred, average='macro')
     rec = recall_score(y_test,dt_pred, average='macro')
```

```
[10] run["train_dataset"].track_files("/content/drive/MyDrive/ALLFLOWMETER_HIKARI2021.csv")

     PARAMS = {"criterion":"gini","Resample":"None","Test size":0.3}

     run["my_params"] = PARAMS

     # You can also specify parameters one by one

     # Update lr value
     run["Result/score"] = report
     run["Result/accuracy"] = acc
     run["Result/f1-score"] = f1
     run["Result/precision"] = pre
     run["Result/recall"] = rec
     run["Result/confusion_matrix"] = fig
```

```
     model = neptune.init_model(
         name="dt_default",
         key="DTDEF",
         project="mikhailamzar/FYP2",
         api_token=my_api_token, # your credentials
     )

     model["Dataset"].track_files("/content/drive/MyDrive/ALLFLOWMETER_HIKARI2021.csv")
     model["Result/score"] = report
     model["Result/accuracy"] = acc
     model["Result/f1-score"] = f1
     model["Result/precision"] = pre
     model["Result/recall"] = rec
     model["Result/confusion_matrix"] = fig
```

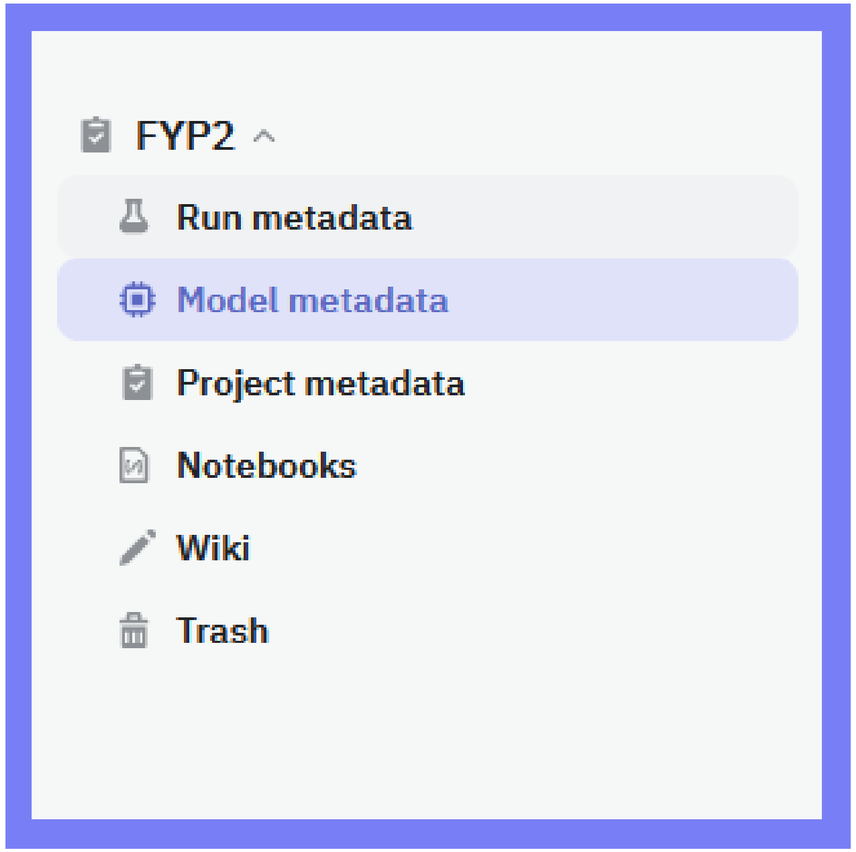
https://app.neptune.ai/mikhailamzar/FYP2/m/FYP-DTDEF

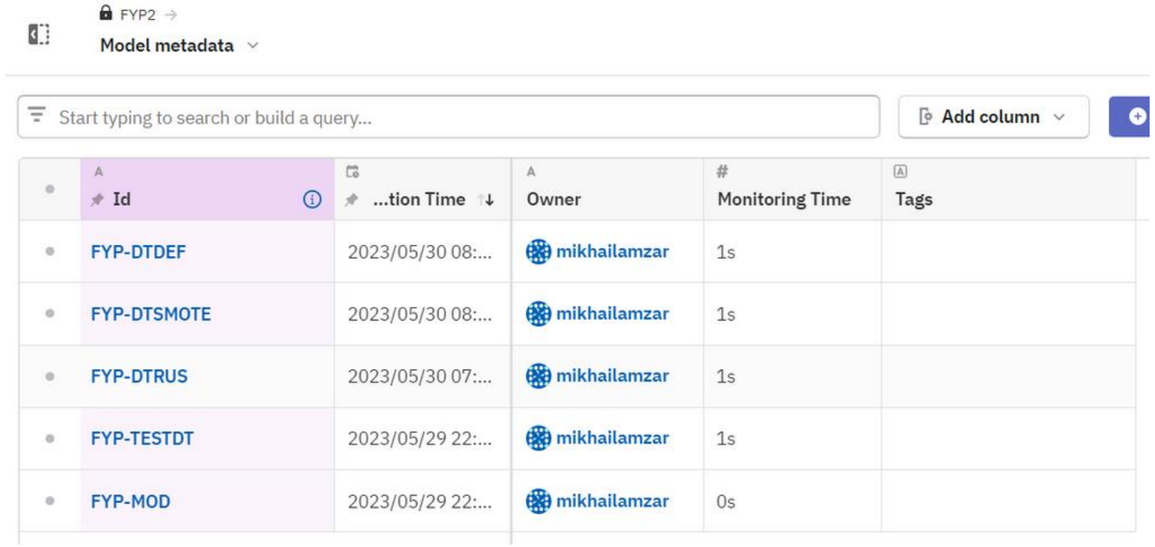
Model scores are sent to the Neptune.ai platform and logged permanently.

More metrics can be sent by extracting more attributes or charts & Integrating the code inside the Colab runtime.

Depends on the machine learning algorithm.

Dashboard: Neptune.ai





Logging by models and different runtimes

Dashboard: Neptune.ai

FYP-13

All metadata | Charts | Images | Monitoring | Source code | Artifacts | + New dashboard

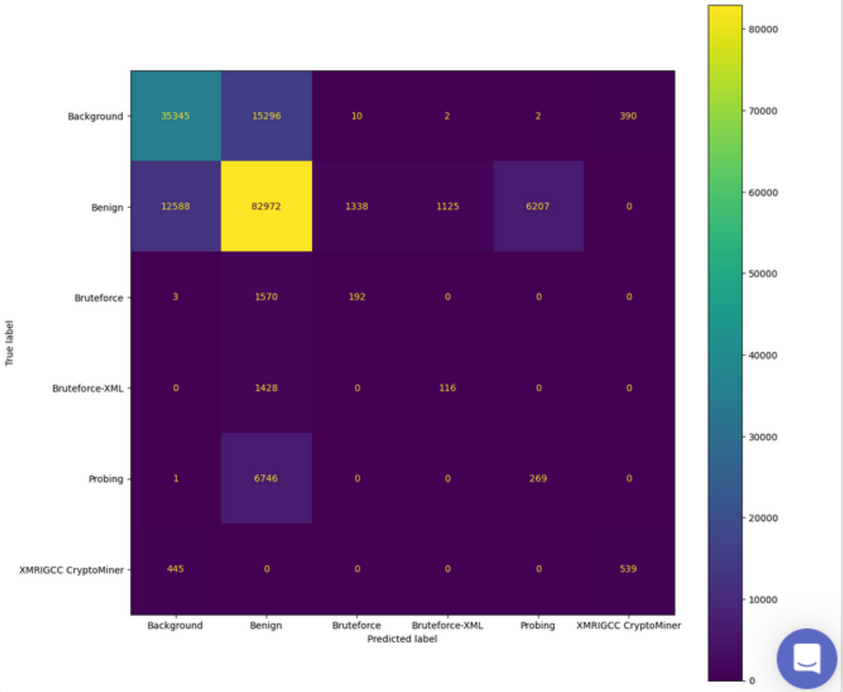
FYP-13 > Result

Search fields

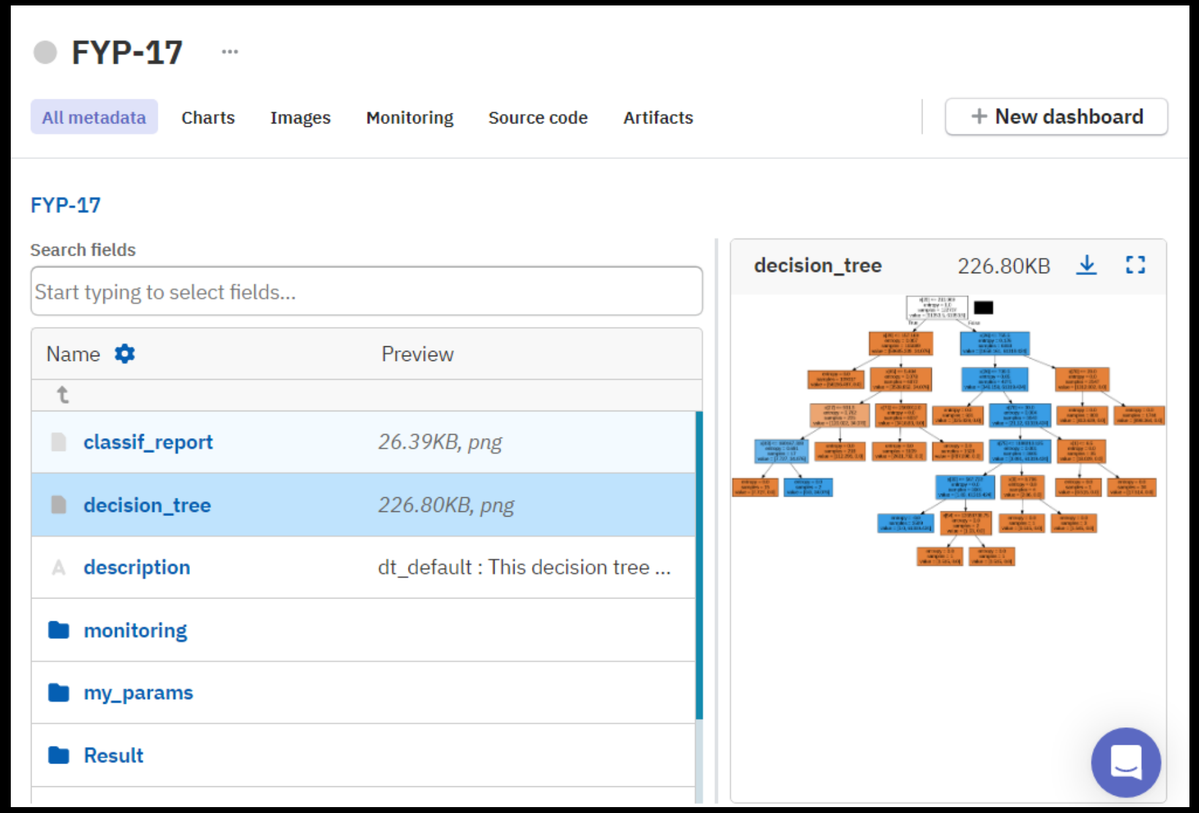
Start typing to select fields...

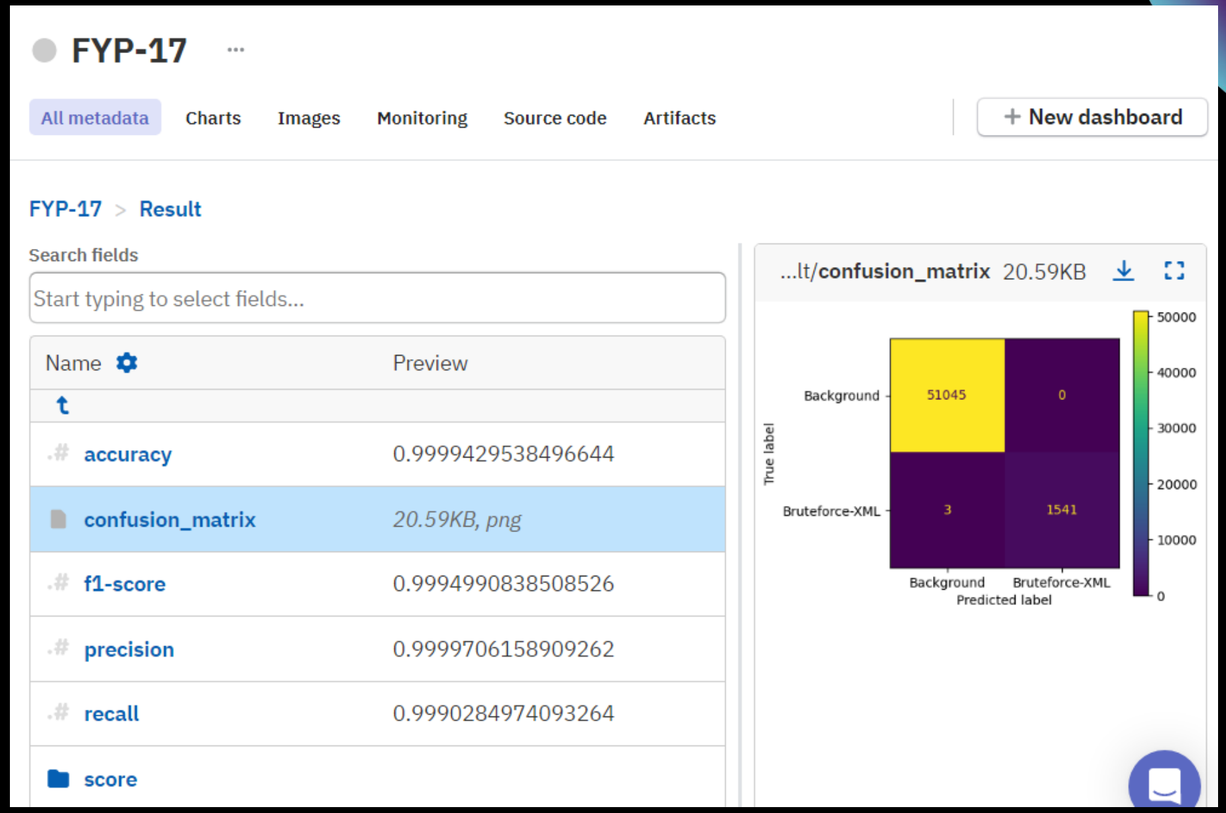
Name	Preview
accuracy	0.7169536089900591
confusion_matrix	*57.32KB, png*
f1-score	0.3826082831152124
precision	0.38973850671108373
recall	0.3764153364832023
score	precision recall f1-score support...

Result/**confusion_matrix** 57.32KB



Dashboard: Neptune.ai





Binary Classification Decision tree model dashboard in Neptune.ai

6. Conclusion

Decision Tree

- A multiclass-classification Decision Tree for HIKARI-21 produce worse scores than KNN, MLP, SVM, and Random Forest. It is not a viable Network Intrusion Detection Model for multi-class task with HIKARI-21.
- Binary Classification Decision Tree for HIKARI-21 produce better scores. It is better suited for an IDS that employs binary classification on specific traffics.

Dashboard

- Neptune.ai dashboard integration is implemented.
  - Allows to track different machine learning models.
  - Keep evaluation metrics and images.
  - Allow for easier workflow for recording machine learning results.

Thank You

UNIVERSITI TENAGA NASIONAL