NETWORK INTRUSION DETECTION MODEL BASED ON

Real and Encrypted Synthetic Attack Traffic using Decision Tree Algorithm

MIKHAIL AMZAR BIN KAMARUDDIN (CS0107477)
Bachelor of Computer Science (Cyber Security) Hons.
Supervised by Md Nabil Bin Ahmad Zawawi, TS

OVERVIEW

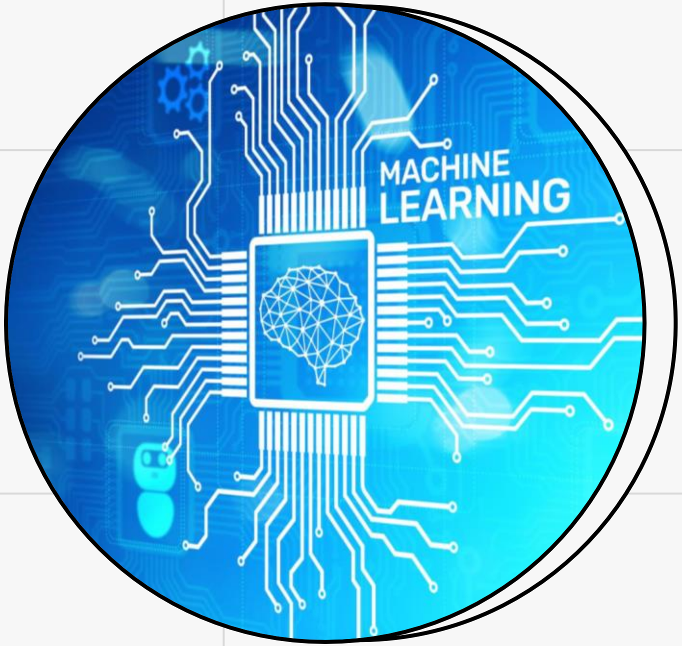
01

INTRODUCTION

INTRODUCTION

The application of machine learning to cybersecurity and network threat detection is a rapidly growing field of study, with numerous new methods and software designed to enhance the effectiveness of these systems.

Various aspects of cybersecurity and threat detection, such as intrusion detection can be approached with machine learning algorithms and methodologies.



MOTIVATION

Develop a network intrusion detection model using Decision Tree based on dataset of real and encrypted synthetic attack traffic.

Evaluate whether a Decision Tree model is a viable algorithm for an Intrusion Detection System when using HIKARI-2021

OBJECTIVES

01

To use Decision Tree Algorithm

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

02

Dashboard

To produce a dashboard for logging and storing the
evaluation metrics details

Evaluate Decision Tree Model

Decision Tree performance using the dataset,
HIKARI-21

PROJECT SCOPE

01

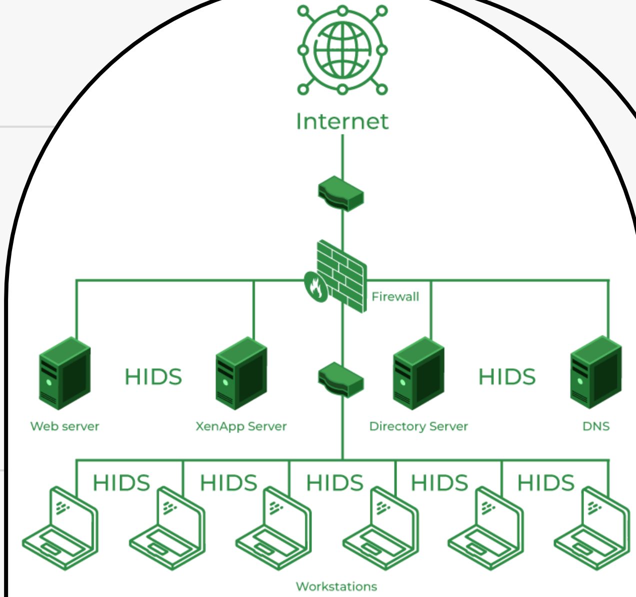
Develop decision tree model

02

Using Dataset: HIKARI-21

Real and Encrypted Synthetic Attack Traffic

02 LITERATURE REVIEW



Intrusion Detection System

Device or software that monitors a network or system for malicious activity.

Types

- **Network-based IDS** : Placed in strategic points within the network, typically data chokepoints.
- **Host-based IDS** : Runs on the host system it is placed in.

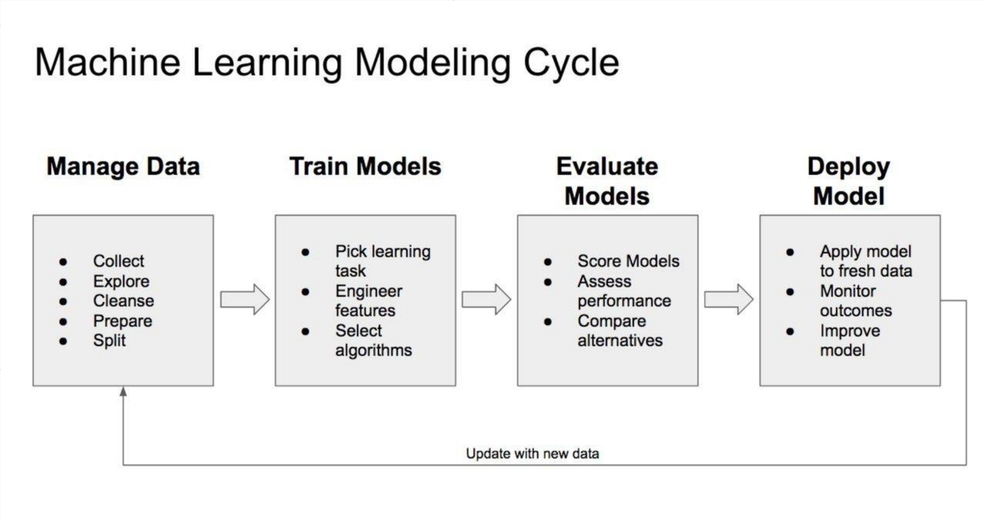
MACHINE LEARNING

Algorithms and statistical models that enable computers to learn
from data and make judgements based on the data it is trained with.

- **Supervised Learning**
- **Unsupervised Learning**

Algorithms are trained to make classifications or predictions.

- Decision Tree model trained to make predictions on network traffic.



DECISION TREE

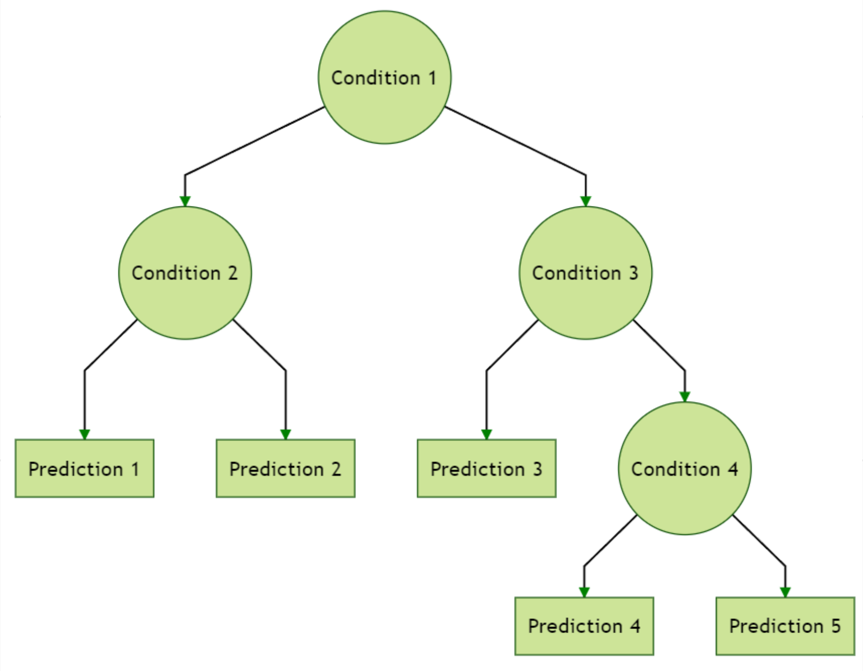
Decision tree structure resembles a tree.
It is a type of supervised machine learning algorithm used to categorize or make predictions based on the set of data fitted.

Types of decision trees.

- Categorical
- Regression

Classification problems, which involve categorizing or classifying an object or input.

Decision trees are also utilized in **regression problems** where it is used in predictive analytics to forecast continuous values.



DATASET : HIKARI-21

(Real and Encrypted Synthetic Attack Traffic)

6 Traffic class

- Background
- Benign
- Bruteforce
- Bruteforce-XML
- Probing
- XMRIGCC Cryptominer

Labelled rows

Traffic completely labelled with target columns *traffic_category* and *Label*

uid	originh	originp	responh	responp	flow_duration	fwd_pkts_tot	bwd_pkts_tot	fwd_data_pkts_tot	bwd_dat
Cg61Jch3vdz9DBptj	103.255.15.23	13316	128.199.242.104	443	2.207588	15	14	6	
CdRllqLWdj35Y9vW9	103.255.15.23	13318	128.199.242.104	443	15.624266	15	14	6	
CLzp9Khd0Y09Qkgrg	103.255.15.23	13320	128.199.242.104	443	12.203357	14	13	6	
Cnf1YA4iLB4CSNWB88	103.255.15.23	13322	128.199.242.104	443	9.992448	14	13	6	
C4ZKvv3fpO72EAOsJ6	103.255.15.23	13324	128.199.242.104	443	7.780611	14	14	6	
CyC8D5X7lIG7U95l4	103.255.15.23	13326	128.199.242.104	443	4.571433	14	13	6	
CEXyM013OxRuUddrS2	103.255.15.23	13328	128.199.242.104	443	2.192640	14	13	6	
CVFc4q26WLSGblwO2c	103.255.15.23	13330	128.199.242.104	443	16.082514	14	14	6	
CCvZhO2f7ztLs9Hopc	103.255.15.23	13332	128.199.242.104	443	13.873240	15	14	6	
ClPZU1mfhrkqqix49	103.255.15.23	13334	128.199.242.104	443	11.331464	14	13	6	
CBTv463lWatXHzgmf3	103.255.15.23	13336	128.199.242.104	443	9.117416	14	13	6	
C87B4VRML4MlzqiFc	103.255.15.23	13338	128.199.242.104	443	6.907568	14	13	6	
CDqmaw1RdkdRYvaDMk	103.255.15.23	13340	128.199.242.104	443	4.692540	15	14	6	
CKPhym3QVbT8Al3aw7	103.255.15.23	13342	128.199.242.104	443	2.198671	14	14	6	
CeAyf115VvqK6kdv6k	103.255.15.23	13344	128.199.242.104	443	16.387078	14	13	6	

03 IMPLEMENTATION

TOOLS USED

Google Colab

Online tool for the development of machine learning and data science related projects

- Execute codes in python notebook documents using the browser.

Neptune.ai

A metadata store that offers experiment tracking and model registry for machine learning researchers and engineers.

- Log, query, manage, display, and compare all our model metadata in a single place.

DATA PREPROCESSING & LIBRARIES

```python
import pandas as pd
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns

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 required libraries.

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

Mounted at /content/drive/

	uid	originh	originp	responh	responp	flow_duration
0	Cg61Jch3vdz9DBptj	103.255.15.23	13316	128.199.242.104	443	2.207588

Import dataset
- Stored in Drive

Traffic Class Distribution in HIKARI-21

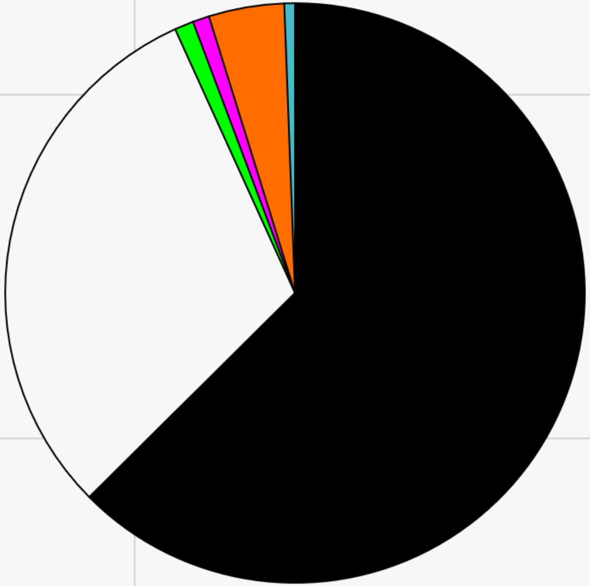
```
df.traffic_category.value_counts()

Benign                  347431
Background              170151
Probing                  23388
Bruteforce                5884
Bruteforce-XML            5145
XMRIGCC CryptoMiner       3279
Name: traffic_category, dtype: int64
```

```
#check for column with missing values

df.isnull().sum()
```

- 62.6% Benign
- 30.6% Background
- 4.2% Probing
- 1.1% Bruteforce
- 0.9% Bruteforce-XML
- 0.6% XMRIGCC Cryptominer



Multi-class Model

1. SPLIT TRAIN & TEST

```
# Assigning features and target column

X = df.drop(['Label','traffic_category'], axis=1)
y = df.traffic_category
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,stratify=y)
```

Using train_test_split to split into training and testing set of data

Whole dataset → **Training set** | **Test set**

2. FIT TRAINING DATA INTO MODEL

```
dt = DecisionTreeClassifier(random_state=1)
dt = dt.fit(X_train,y_train)
```

3. FIT TEST DATA INTO MODEL

```
y_pred = dt.predict(X_test)
```

Where the trained model is tested with new data.

Binary-class Model

```
df = df[ (df['traffic_category'] != "XMRIGCC CryptoMiner") & (df['traffic_category'] != "Bruteforce-XML") & (df['traffic_category'] != "Benign") & (df['traffic_category'] != "Bruteforce")]
df.traffic_category.value_counts()
```

```
Background    170151
Probing        23388
Name: traffic_category, dtype: int64
```

Only choosing 2 traffic types for classification.

1. SPLIT TRAIN & TEST

```
# Assigning features and target column

X = df.drop(['Label','traffic_category'], axis=1)
y = df.traffic_category
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1,stratify=y)
```

Using train_test_split to split into training and testing set of data

Whole dataset → **Training set** | **Test set**

2. FIT TRAINING DATA INTO MODEL

```
dt = DecisionTreeClassifier(random_state=1)
dt = dt.fit(X_train,y_train)
```

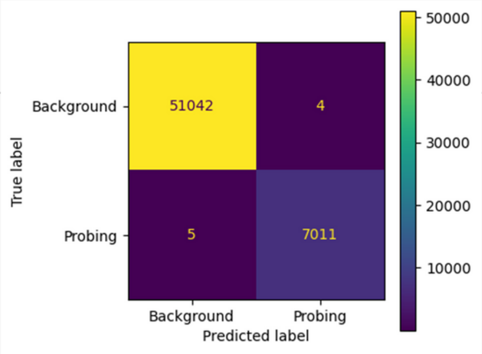
3. FIT TEST DATA INTO MODEL

```
y_pred = dt.predict(X_test)
```

Where the trained model is tested with new data.

04 RESULTS & FINDINGS

Evaluation metrics



Confusion Matrix

Performance metric in the form of a table for classification problems in machine learning. It is a summary of prediction results on a classification problem.

Accuracy

Can be defined as the proportion of correct predictions to the total number of predictions in the model.

Precision

The proportion of accurately predicted positive labels out of all positive predictions made

Recall

Represents the model's ability to accurately predict true positives. In simpler terms, from the total actual positive label, how many are correctly predicted as positive.

Multi-class Model

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%

Model Scores

Accuracy : 0.72
Precision : 0.39
Recall : 0.38

Confusion Matrix

Poor results and predictions
From the multi-class model.

Multi-class Model

Comparison of scores against other algorithms.

Provided by the creators of HIKARI-21, where they perform performance analysis using other algorithms with HIKARI-21.

Algorithm	Accuracy	Precision	Recall
KNN	0.98	0.86	0.90
MLP	0.99	0.99	0.99
SVM	0.99	0.99	0.98
RF	0.99	0.99	0.99
Decision Tree	0.72	0.38	0.38

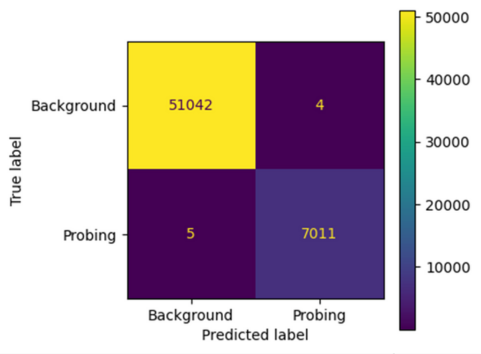
Binary-class Model

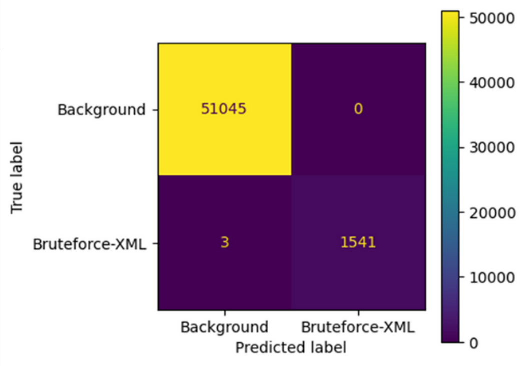
Model Scores

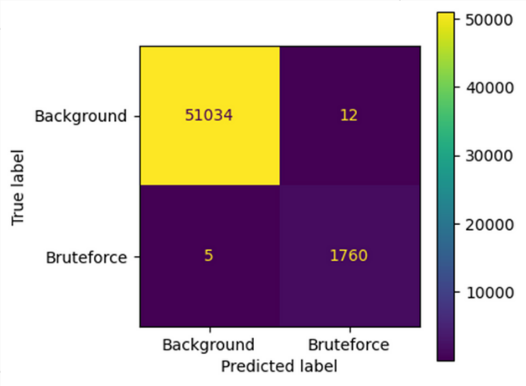
Model	Accuracy	Precision	Recall
Background & Probing	1.00	1.00	1.00
Background & Bruteforce	1.00	1.00	1.00
Background & Bruteforce-XML	1.00	1.00	1.00
Background & XMRIGCC CryptoMiner	0.98	0.75	0.77

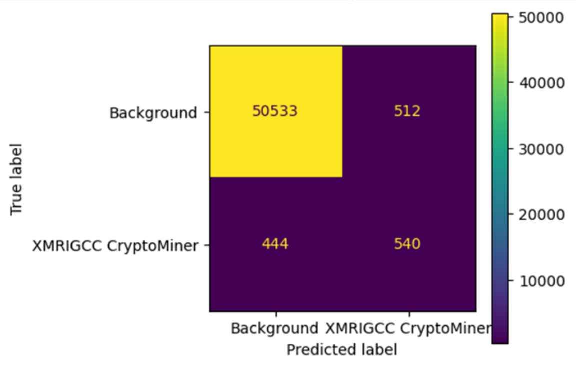
Binary-class Model

Confusion Matrix









Binary-class Model

Balancing the distribution

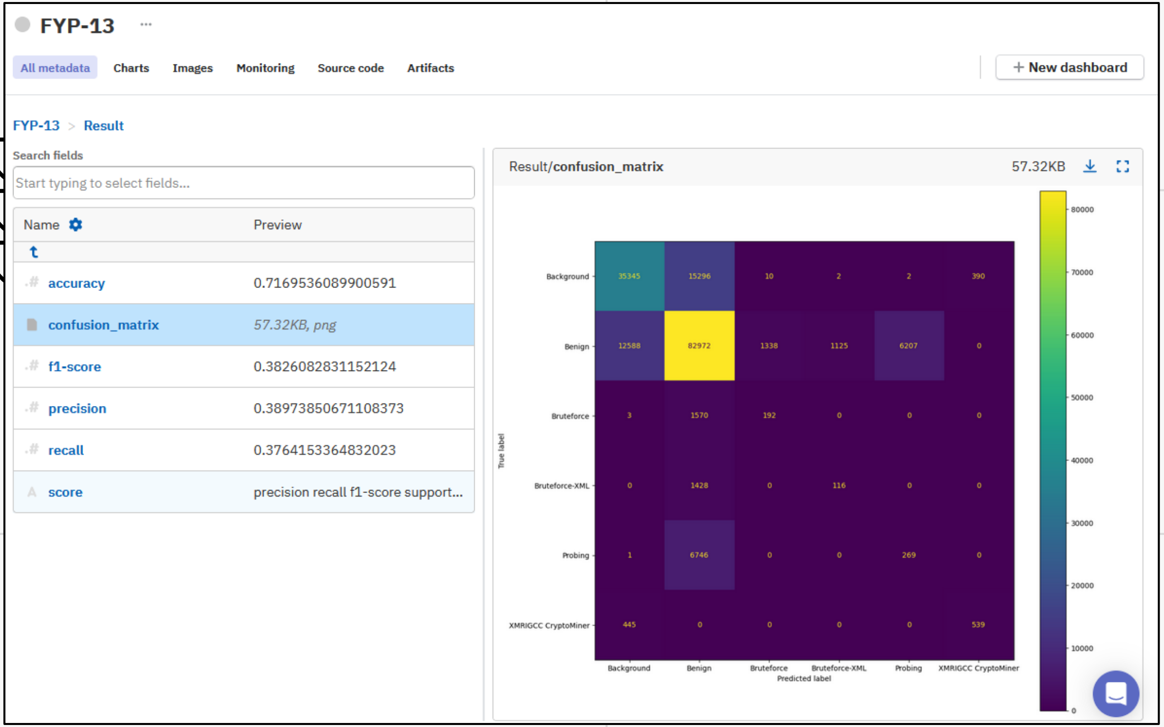
```
Background             170151
XMRIGCC CryptoMiner      3279
Name: traffic_category, dtype: int64
```

```
df_sub = df.drop(df[df['traffic_category'] == 'Background'].sample(frac=.86).index)

df_sub.traffic_category.value_counts()
```

```
Background             23821
XMRIGCC CryptoMiner     3279
Name: traffic_category, dtype: int64
```

Background & XMRIGCC CryptoMiner	0.98	0.75	0.77
Background & XMRIGCC CryptoMiner (After reducing Background)	0.98	0.94	0.94



Dashboard

The evaluation metrics of finished model are stored in Neptune.ai

05 CONCLUSION

CONCLUSIONS

01

Multi-class Decision Tree model

Poor result could be due to imbalanced class distribution, leads to learning bias in the model. Other algorithms are more suitable.

02

Binary-class Decision Tree Model

Perform prediction better and produce high model scores.

03

Balancing the dataset class distribution

Can improve performance and reduce learning bias in the model.

FUTURE WORKS

01

Improving Multi-class Model

Balancing the class distribution, using resampling techniques (Oversampling, undersampling)

THANK YOU