National College of Ireland

Configuration Manual

MSc Research Practicum Part 2

MSc. in Data Analytics

Sahana Lakshmikanthaiah

Student ID: x24172812

School of Computing

National College of Ireland

Supervisor:Prof. Anderson Simiscuka

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Sahana Lakshmikanthaiah

Student ID: x24172812

Programme: Msc Data Analytics **Year:** 2025-2026

Module: MSc Practicum Part 2

Lecturer: Prof. Anderson Simiscuka

Submission Due Date: 27/01/2026

Project Title: Intrusion-and-Impact: Forecasting UAV Communication Attacks and their short-horizon operation degradation

……772……

Word Count: **PageCount:**……12……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature :……Sahana Lakshmikanthaiah……

Date: ……27/01/2026……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sahana Lakshmikanthaiah
Student ID: x24172812

1 Introduction

This report will provide us with a step-by-step guide on how to set up and implement the project.

2 Environment Setup

2.1 System Configuration

2.1.1 Software Requirements

Requirement	Specification
Programming Language	Python 3.10
Notebook Environment	Jupyter Notebook
Libraries	NumPy, Pandas, Scikit-learn, LightGBM, XGBoost, CatBoost, Matplotlib, Seaborn
ML Frameworks	Gradient Boosting Libraries (LGBM/XGB/CAT)
Probability Calibration	sklearn.calibration
OS Used	macOS 14.x (Sonoma)
Browsers / Tools	Chrome, Terminal, Conda

2.1.2 Hardware Requirements

Requirement	Specification
Device	Apple MacBook (M3)
CPU	Apple M3
GPU	Apple M3 GPU via Metal Performance Shaders (MPS)
RAM	16 GB
Storage	256 GB

2.2 Python Environment Setup

2.2.1 Create and Activate Virtual Environment

```
python3 -m venv uav_env
source uav_env/bin/activate      # macOS / Linux
uav_env\Scripts\activate         # Windows
```

2.2.2 Install Required Libraries

```
pip install numpy pandas matplotlib seaborn scikit-learn
pip install lightgbm xgboost catboost
pip install shap
```

3 Dataset

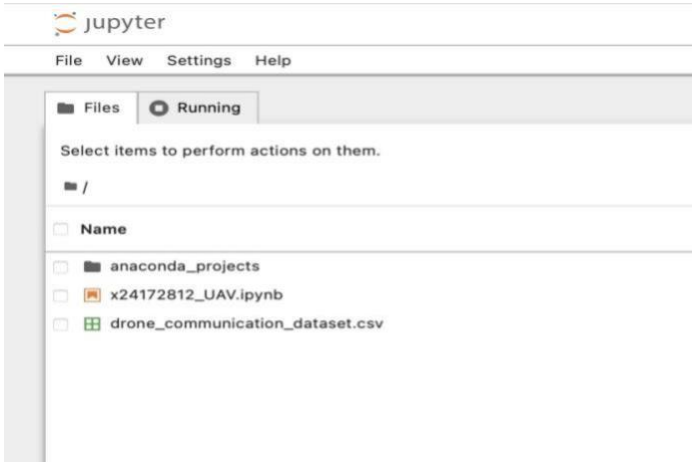
Source: https://www.kaggle.com/datasets/datasetengineer/tokyo-drone-communication-and-security-dataset?select=drone_communication_dataset.csv

4 Running the Project

4.1.1 Step 1: Launch the Notebook

Open Jupyter : jupyter notebook

Then open **x24172812_UAV.ipynb**



4.1.2 Data is downloaded.

```
# Purpose: load CSV, normalize timestamps, derive IntrusionBinary from Target Label (attack/intrusion=1).
# Assumption: Target Label strings include 'attack' or 'intrusion' for positives; 'Normal' = 0.

#Load + Path to your CSV
DATA_PATH = "drone_communication_dataset.csv"

# Load
df = pd.read_csv(DATA_PATH)
print(df.shape)
df.head()
```

```
(52585, 29)
```

	Timestamp	Source Drone ID	Destination ID	Packet Size	Transmission Rate	Signal Strength	Error Rate	Encryption Status	Protocol Type	Response Time	...	Signal-to-Noise Ratio	Data Throughput	Port Number	Commu
0	2018-01-01 00:00:00	D1	D1	2000	1	-48.674464	0.05	1	UDP	100	...	25.649452	43.669663	443	
1	2018-01-01 01:00:00	D3	D2	500	5	-51.596394	0.01	1	UDP	10	...	23.916592	61.761303	443	
2	2018-01-01 02:00:00	D2	BaseStation	500	1	-44.884766	0.01	1	UDP	1000	...	26.361278	17.959860	80	
3	2018-01-01 03:00:00	D1	D3	2000	1	-58.707466	0.01	1	TCP	10	...	15.833704	82.039414	8080	
4	2018-01-01 04:00:00	D3	D2	1000	1	-44.503895	0.01	1	TCP	100	...	20.536170	41.238151	80	

5 rows × 29 columns

4.1.3 **Data Preparation Blocks**

What happens in these cells:

- Load and clean telemetry
- Normalize timestamps
- Enforce strict chronological ordering
- Build 3-minute intrusion labels
- Build QoS degradation targets (ΔRTT, ΔJitter, ΔThroughput)
- Create SLA violation labels (binary)

```
#3-minute horizon alignment (CRISP-DM: Data Understanding)
assert 'Timestamp' in df.columns, "Timestamp column is required."
assert 'Target Label' in df.columns, "Target Label column is required."

# Parse & sort time
df['Timestamp'] = pd.to_datetime(df['Timestamp'], errors='coerce')
df = df.dropna(subset=['Timestamp']).sort_values('Timestamp').reset_index(drop=True)

# Create IntrusionBinary from Target Label (attack/intrusion => 1, else 0)
lab = df['Target Label'].astype(str).str.lower()
df['IntrusionBinary'] = np.where(lab.str.contains('attack') | lab.str.contains('intrusion'), 1, 0)

print("Original rows:", len(df), "Intrusion rate:", df['IntrusionBinary'].mean())
```

```
Original rows: 52585 Intrusion rate: 0.14914899686222308
```

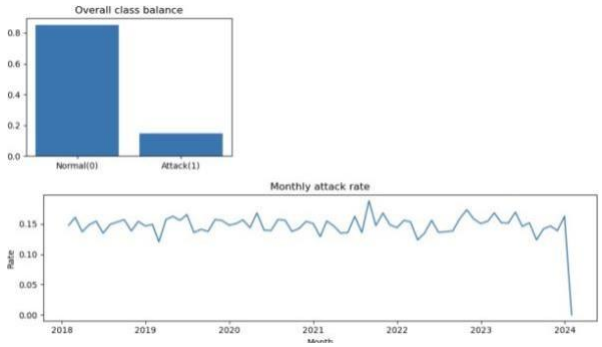
```
# Output: df (aligned), numeric_features/categorical_features, H_STEP>=1.

# Fix horizon to 3 minutes
H_MINUTES = 3
TARGET_FREQ = f'{H_MINUTES}T'  # '3T' = 3 minutes

# Feature lists from YOUR dataset
NUM_COLS = [
    'Packet Size','Transmission Rate','Signal Strength','Error Rate',
    'Response Time','Average Latency','Packet Loss Rate','Connection Duration',
    'Round Trip Time','Hop Count','Jitter','Drone Velocity',
    'Signal-to-Noise Ratio','Data Throughput',
    'Communication Interval','Control Command Frequency','Drone Altitude',
    'CPU Usage','Memory Utilization','Distance to Base Station'
]
CAT_COLS = ['Protocol Type','Source Drone ID','Destination ID','Encryption Status','Port Number']

NUM_COLS = [c for c in NUM_COLS if c in df.columns]
CAT_COLS = [c for c in CAT_COLS if c in df.columns]

# Coerce numerics
for c in NUM_COLS:
    df[c] = pd.to_numeric(df[c], errors='coerce')

# Check current step (seconds)
step_sec = None
if len(df) >= 2:
    step_sec = float(df['Timestamp'].diff().dropna().dt.total_seconds().median())

# Resample to 3-minute grid if needed
df_res = df.set_index('Timestamp').sort_index()

need_resample = True
if step_sec is not None:
    need_resample = step_sec != (H_MINUTES * 60)

# === SAFEGUARD: only keep columns that exist
keep_cols = [c for c in (NUM_COLS + CAT_COLS + ['Target Label','IntrusionBinary']) if c in df_res.columns]
df_res = df_res[keep_cols]

if need_resample:
    df_res = df_res.resample(TARGET_FREQ).asfreq()
    if NUM_COLS:
        df_res[NUM_COLS] = df_res[NUM_COLS].interpolate(method='time', limit_direction='both')
    ffill_cols = [c for c in (CAT_COLS + ['Target Label','IntrusionBinary']) if c in df_res.columns]
    if ffill_cols:
        df_res[ffill_cols] = df_res[ffill_cols].ffill()

df = df_res.reset_index().rename(columns={'index':'Timestamp'})

# Rebuild time features on the 3-minute grid
df['Year'] = df['Timestamp'].dt.year
df['Month'] = df['Timestamp'].dt.month
df['Day'] = df['Timestamp'].dt.day
df['DayOfWeek'] = df['Timestamp'].dt.dayofweek
```

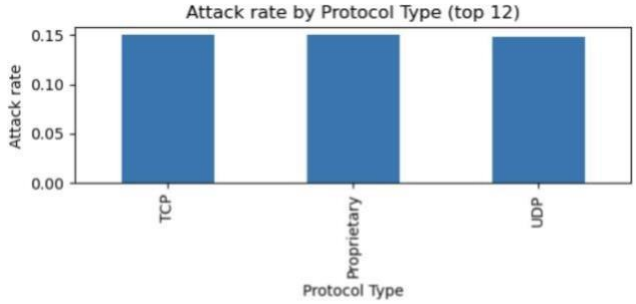
4.1.4 Exploratory Data Analysis

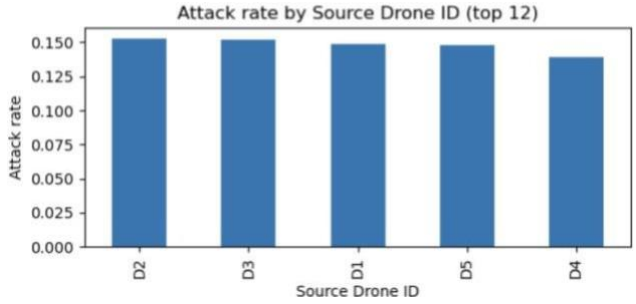
The notebook includes visualizations for:

- Missing values
- Class imbalance
- KPI distributions
- Attack rate by category

This validates data quality and informs later modeling decisions.







4.1.5 Feature Engineering

The system automatically generates:

- Short lags (t-1, t-2)
- Rolling statistics (3-min windows)
- Ratios and normalized KPIs
- Compact, leakage-free features
- Train-only fit for scalers / encoders

```
# Impact targets (1 over next 3 minutes) for your KPIs
kpi_series = {}
if 'Data Throughput' in df.columns:
    kpi_series['Data Throughput'] = future_min_delta(df['Data Throughput'], H_STEPS)
if 'Round Trip Time' in df.columns:
    kpi_series['Round Trip Time'] = future_max_delta(df['Round Trip Time'], H_STEPS)
if 'Jitter' in df.columns:
    kpi_series['Jitter'] = future_max_delta(df['Jitter'], H_STEPS)
if 'Packet Loss Rate' in df.columns:
    kpi_series['Packet Loss Rate'] = future_max_delta(df['Packet Loss Rate'], H_STEPS)

for k in list(kpi_series.keys()):
    kpi_series[k] = kpi_series[k].loc[valid_idx]

print("Targets ready. Positives(attack in next 3m):", int(y_attack_h.loc[valid_idx].sum()))
```

```
Targets ready. Positives(attack in next 3m): 156860
```

```
# === NEW: KPI SLA Violation Labels within (t, t+H] ===
KPI_THRESHOLDS = {
    'Round Trip Time': 150.0,
    'Jitter': 20.0,
    'Packet Loss Rate': 0.05,
    'Data Throughput': 2.0
}

def future_window_extrema(series, H_STEPS, mode='max'):
    vals = series.to_numpy()
    out = np.full(len(vals), np.nan, float)
    for t in range(len(vals)-H_STEPS):
        window = vals[t+1:t+H_STEPS+1]
        if len(window)==0 or np.any(np.isnan(window)):
            continue
        out[t] = np.max(window) if mode=='max' else np.min(window)
    return pd.Series(out, index=series.index)

y_viol = {}
if 'Round Trip Time' in df.columns:
    fv = future_window_extrema(df['Round Trip Time'], H_STEPS, mode='max')
    y_viol['Round Trip Time'] = (fv.loc[valid_idx] > KPI_THRESHOLDS['Round Trip Time']).astype(int)
if 'Jitter' in df.columns:
    fv = future_window_extrema(df['Jitter'], H_STEPS, mode='max')
    y_viol['Jitter'] = (fv.loc[valid_idx] > KPI_THRESHOLDS['Jitter']).astype(int)
if 'Packet Loss Rate' in df.columns:
    fv = future_window_extrema(df['Packet Loss Rate'], H_STEPS, mode='max')
    y_viol['Packet Loss Rate'] = (fv.loc[valid_idx] > KPI_THRESHOLDS['Packet Loss Rate']).astype(int)
if 'Data Throughput' in df.columns:
    fv = future_window_extrema(df['Data Throughput'], H_STEPS, mode='min')
    y_viol['Data Throughput'] = (fv.loc[valid_idx] < KPI_THRESHOLDS['Data Throughput']).astype(int)

print({k:int(v.sum()) for k,v in y_viol.items()})
```

```
{'Round Trip Time': 0, 'Jitter': 0, 'Packet Loss Rate': 56109, 'Data Throughput': 851}
```

4.1.6 Intrusion Prediction Models

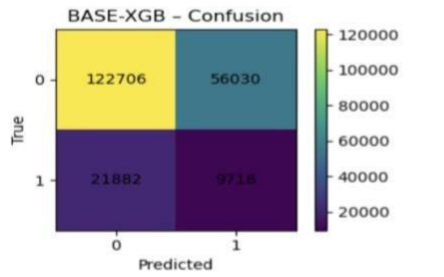
Three models are trained:

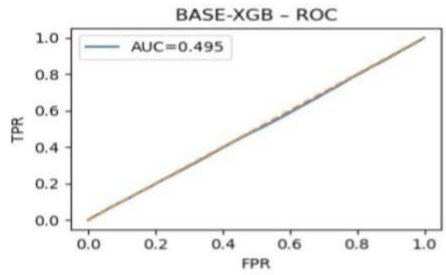
- **LightGBM**
- **XGBoost**
- **CatBoost**

Outputs include:

- ROC-AUC
- Precision–Recall curves
- Confusion matrices
- Brier score
- Calibration curves

```
BASE-XGB  Acc=0.630  F1=0.200  ROC_AUC=0.495  PR_AUC=0.150  Brier=0.228
              precision    recall  f1-score   support

           0      0.849     0.687     0.759    178736
           1      0.148     0.308     0.200     31600

    accuracy                          0.630    210336
   macro avg      0.498     0.497     0.479    210336
weighted avg      0.743     0.630     0.675    210336
```





4.1.7 QoS Impact Forecasting Models

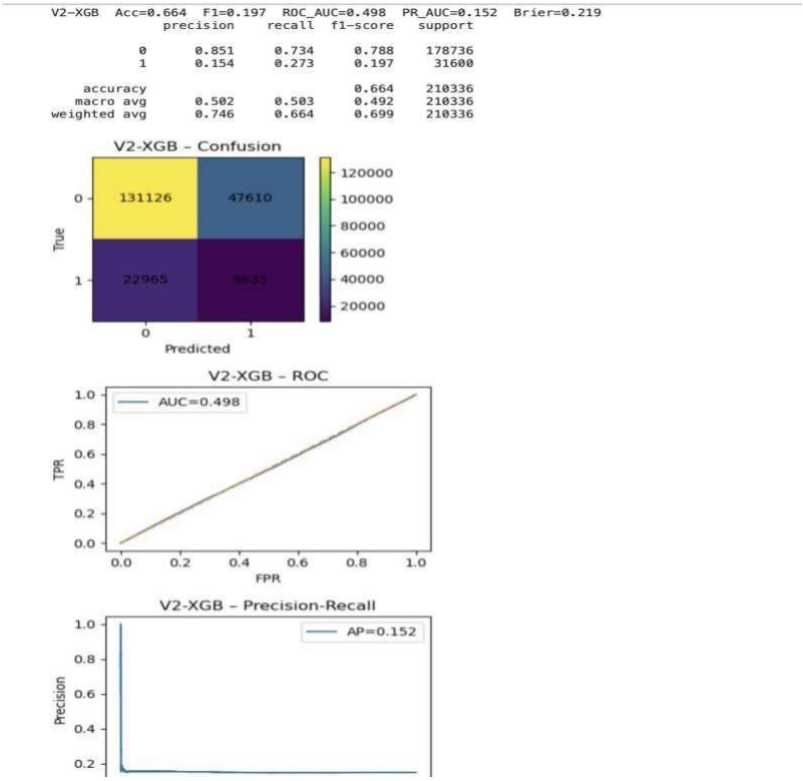
For each KPI (RTT, Jitter, Throughput), three regressors are trained:

- LightGBM

- XGBoost
- CatBoost

Reports include:

- MAE
- RMSE
- R^2



4.1.8 SLA Violation Models

Binary classifiers predict:

- RTT violation
- Jitter violation
- Throughput violation

Outputs:

- Probability of violation within 3 minutes
- Accuracy metrics
- Brier score
- ROC-AUC

Name	Accuracy	F1	ROC_AUC	PR_AUC	Brier
BASE-LGBM	0.732000	0.165000	0.507000	0.153000	0.203700
BASE-CAT	0.601000	0.211000	0.498000	0.152000	0.234500
BASE-XGB	0.630000	0.200000	0.495000	0.150000	0.228400

Name	Accuracy	F1	ROC_AUC	PR_AUC	Brier
V2-CAT	0.623000	0.206000	0.504000	0.151000	0.229700
V2-XGB	0.664000	0.197000	0.498000	0.152000	0.218500
V2-LGBM	0.750000	0.148000	0.498000	0.151000	0.195200

Label	MAE	RMSE	R2
Data Throughput – CAT	0.243800	0.539300	0.853700
Data Throughput – LGBM	0.179400	0.361800	0.934200
Data Throughput – XGB	0.193700	0.391200	0.923000
Jitter – CAT	0.024700	0.055000	0.847400
Jitter – LGBM	0.017500	0.036100	0.934300
Jitter – XGB	0.019300	0.039700	0.920300
Packet Loss Rate – CAT	0.000100	0.000500	0.934800
Packet Loss Rate – LGBM	0.000100	0.000300	0.970700
Packet Loss Rate – XGB	0.000100	0.000300	0.970000
Round Trip Time – CAT	0.149700	0.512900	0.901800
Round Trip Time – LGBM	0.062800	0.276800	0.971400
Round Trip Time – XGB	0.066200	0.281100	0.970500

4.1.9 Unified Risk Scoring

The final block computes: Risk Score = $P(\text{Intrusion}) \times \Sigma(\text{weight}_k \times |\Delta KPI_k|)$

Weights used:

- RTT: 1.0
- Throughput: 0.9
- Jitter: 0.7
- Packet-loss: 0.8

A second variant uses **violation probabilities** instead of magnitude deltas.

The output is:

- A ranked list of high-risk windows
- A timeline showing spikes that align with intrusion periods

```
Top 10 risky 3-minute windows:
948914    2.765646
948918    2.744070
948919    2.603439
977513    2.587890
977516    2.585247
977514    2.583538
977512    2.553700
977515    2.483674
977510    2.461179
977509    2.436062
Name: RiskScore, dtype: float64
```

```
Top 10 risky windows (Violation-based RiskScore):
917819     0.436682
886061     0.434665
886062     0.433476
944279     0.328030
1009839    0.314335
1023818    0.303384
843879     0.301396
904539     0.301396
914759     0.301394
914758     0.301394
Name: RiskScore_Violation, dtype: float64
```

5 Configuration Knobs (User-Adjustable Settings)

Component	Configurable Parameter	Effect
Forecast Horizon	H = 3 minutes	Adjusts future window size
Lag Depth	lag_steps	Controls temporal sensitivity
Model Choice	LGBM/XGB/CAT	Can prefer speed vs accuracy
Calibration Method	isotonic / Platt	Improves probability reliability
KPI Weights	RTT=1.0, Jitter=0.7, etc.	Prioritizes mission-critical KPIs
Train/Val/Test Split	Date boundaries	Controls time-honest segmentation
Thresholds	SLA-based or quantile-based	Affects violation detection

6 Troubleshooting

6.1.1 Dataset not loading

- Ensure file exists under /data/
- Check CSV encoding
- Validate timestamp formatting

6.1.2 Calibration Errors

- Ensure classifier is already fitted before calibration
- Check class imbalance; consider adjusting class weights

6.1.3 LGBM/XGB not installing on Mac

Use: pip install lightgbm --config-settings cmake.define.USE_METAL=ON