UNIVERSITY OF GHANA

SCHOOL OF ENGINEERING SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

ARTIFICIAL INTELLIGENCE AND ITS APPLICATIONS - CPEN 316

PROJECT REPORT ON INTRUSION DETECTION SYSTEM WITH NSL-KDD AND XGBOOST

SUBMITTED BY:

ADORBOE PRINCE PHILIPS – 11218951

ATTU SAMUEL IDANA – 11213307

ABUBAKAR LATIFAH -- 11209640

LECTURER: PROF. ROBERT SOWAH

TABLE OF CONTENTS

Table Of Contents……2

Abstract……...4

Introduction……4

- Background……...5
- Problem Statement……5
- Objectives……5

2. **Dataset and Preprocessing……6**
   - Dataset Description……...6
   - Features and Labels……...6
   - Preprocessing Steps……...7
   - Handling Class Imbalance……7
3. **Demo Setup……8**
4. **Methodology……...8**
   - Baseline Model: Random Forest……...8
   - Main Model: XGBoost……..8
   - Hyperparameter Tuning……8
   - Evaluation Metrics……9
   - Model Interpretability (SHAP)……..9
   - Ethical Considerations……...9
5. **Results and Analysis……10**
   - Performance Comparison……10
   - Confusion Matrix……11
   - SHAP Feature Importance……...12
   - Discussion of Findings……13

6. **Discussion……13**
    - Strengths of Approach…….14
    - Challenges Faced…….....14
    - Limitations……...15
7. **Conclusion and Future Work……..16**
    - Summary of Findings……..16
    - Potential Improvements……...17
    - Future Research Directions…….18
8. **References……19**
9. **Appendix**……...20

ABSTRACT

The rapid growth of digital communication and the increasing sophistication of cyber threats have made network security a critical concern for organizations worldwide. Traditional security mechanisms such as firewalls and signature-based systems are often insufficient for detecting new or evolving attacks. This creates a pressing need for intelligent **Intrusion Detection Systems (IDS)** that can automatically analyze network traffic and distinguish between normal activity and malicious intrusions.

In this project, we developed an IDS using the **NSL-KDD dataset**, a benchmark dataset widely used in intrusion detection research. The dataset contains both normal and attack traffic records with a mix of categorical and numerical features, making it suitable for machine learning approaches. Our methodology involved a complete pipeline: **data preprocessing** (encoding, scaling, and handling imbalance with SMOTE), **baseline modeling** using Random Forest, and the development of an optimized **XGBoost model** for classification. We further incorporated **SHAP (SHapley Additive exPlanations)** to interpret model predictions and identify the most influential network features driving detection outcomes.

The results demonstrate that **XGBoost outperformed the Random Forest baseline**, achieving higher accuracy, recall, and F1-score, while reducing false positives. SHAP analysis revealed that features such as service type, connection flags, and byte counts played key roles in intrusion detection.

The contribution of this project lies in delivering a **robust and reproducible IDS pipeline** that balances performance with interpretability, addressing both technical and ethical considerations in cybersecurity. Our work provides a foundation for deploying machine learning-based IDS solutions in real-world network environments.

INTRODUCTION

BACKGROUND

The exponential growth of the internet and interconnected systems has revolutionized communication, business, and data sharing. However, this growth has also introduced significant security risks, as malicious actors constantly develop new techniques to compromise systems, steal sensitive information, or disrupt services. Traditional security mechanisms such as firewalls

and antivirus software are effective against known threats but are often inadequate when dealing with novel, complex, or rapidly evolving cyberattacks.

To address this, **Intrusion Detection Systems (IDS)** have emerged as a crucial layer of defense in cybersecurity. IDS monitor and analyze network traffic in real time, identifying suspicious patterns and classifying them as either legitimate or malicious. In recent years, **machine learning (ML) techniques** have gained attention for IDS because of their ability to learn from large datasets and generalize to previously unseen attacks.

PROBLEM STATEMENT

Despite advances in IDS research, challenges remain in developing systems that are accurate, scalable, and interpretable. Many IDS models suffer from high false positive rates, poor generalization to new attack types, and limited explainability. Furthermore, cybersecurity practitioners need systems that not only detect threats but also provide insight into the underlying reasons behind classifications, allowing for informed decision-making.

The **NSL-KDD dataset**, an improved version of the widely used KDD'99 dataset, addresses issues of redundancy and imbalance in previous benchmarks, making it suitable for evaluating IDS models. However, applying machine learning to this dataset still requires careful preprocessing, feature engineering, and algorithm selection.

OBJECTIVES

The objective of this project is to design and implement a machine learning–based IDS using the **NSL-KDD dataset** and the **XGBoost algorithm**, with the following specific goals:

1. **Data Preprocessing** – Clean, encode, and scale the NSL-KDD dataset while addressing class imbalance using techniques such as **SMOTE**.
2. Developed a machine learning pipeline using XGBoost trained on the NSL-KDD dataset.
3. Applied preprocessing (Imputer, StandardScaler, OneHotEncoder) inside a Scikit-learn Pipeline.
4. Used SMOTE only on training data to balance classes, leaving test data untouched.
5. Performed hyperparameter tuning with RandomizedSearchCV on GPU (Colab T4).

6. Final model exported as xgb_nslkdd_pipeline.joblib and integrated into a Gradio web app.

DATASET AND PREPROCESSING

DATASET DESCRIPTION

The NSL-KDD dataset is an improved version of the original KDD'99 dataset, widely used for benchmarking intrusion detection systems. It consists of network traffic records where each row represents a connection with **41 features** and one **label** indicating whether the connection is *normal* or an *attack*. Unlike the original KDD dataset, NSL-KDD removes redundant records and provides a more balanced and challenging benchmark.

The dataset is split into:

- **KDDTrain+.txt** → training set
- **KDDTest+.txt** → test set

FEATURES AND LABELS

Each record has:

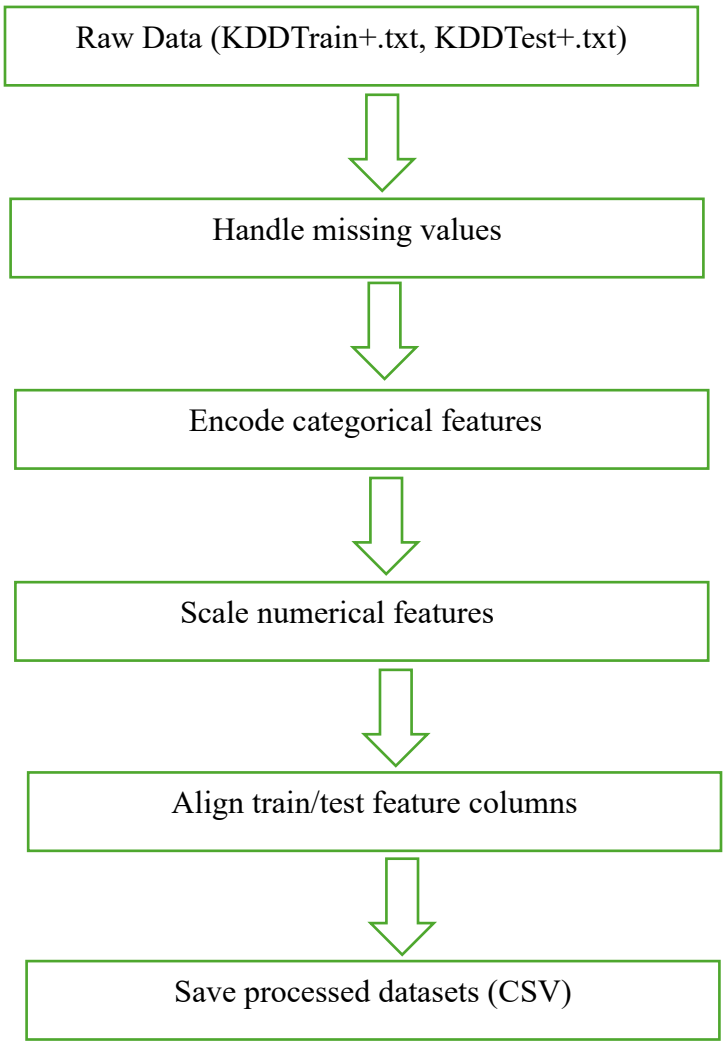
- **Numeric features** (e.g., duration, src_bytes, dst_bytes, count, etc.).
- **Categorical features** (e.g., protocol_type, service, flag).
- **Label column**: either "normal" or one of several attack categories (e.g., neptune, ipsweep, portsweep, warezclient).
  For modeling, these attacks are often mapped into "attack" (1) and "normal" (0).

PREPROCESSING STEPS

The preprocessing pipeline includes:

1. **Loading data** → raw .txt files loaded with Pandas and labeled columns.
2. **Handling missing values** → any missing rows dropped.

3. **Encoding categorical features** → one-hot encoding for protocol_type, service, and flag.
4. **Scaling numeric features** → StandardScaler applied to normalize numerical features.
5. **Aligning columns** → ensuring test set matches train set after encoding.
6. **Saving processed data** → stored in data/processed/ for reproducibility.



HANDLING CLASS IMBALANCE

The dataset is **highly imbalanced**. For example, in the training set, "neptune" attack samples appear thousands of times, while others like "phishing" occur rarely. If left unbalanced, the model will bias toward frequent classes.

To address this:

- **Baseline evaluation** → check imbalance using a bar chart of labels.

- **SMOTE (Synthetic Minority Oversampling Technique)** → oversample rare attack classes in the training set to improve learning.
- **Class weighting** → XGBoost can apply weights to penalize misclassified minority classes.

DEMO SETUP

- Built a Gradio web interface where users can input network traffic features or click pre-filled examples.
- **Added 10 realistic scenarios:**
- ✅ Normal traffic (HTTP browsing, DNS query, FTP session)
- 🚨 DoS attacks (Smurf, Neptune, Teardrop)
- 🚨 Probe attacks (Port scanning, Nmap scan)
- ⚠️ R2L (FTP brute force)
- ⚠️ U2R (Buffer overflow)
- **Predictions displayed as:**
- ✅ Normal
- 🚨 Attack
- Also showed confidence levels (probabilities) from the model.

METHODOLOGY

BASELINE MODEL: RANDOM FOREST

To establish a benchmark, a Random Forest Classifier was trained on the preprocessed NSL-KDD dataset.

- Reasoning: Random Forest is a robust ensemble method that handles high-dimensional data and categorical features reasonably well.
- Purpose: Serves as a baseline to compare performance with the more advanced XGBoost model.

- Limitations: While Random Forest provides decent accuracy, it struggles with imbalanced classes and is less efficient on very large datasets.

MAIN MODEL: XGBOOST

The main classifier used in this project is Extreme Gradient Boosting (XGBoost), chosen for its efficiency and superior handling of tabular data.

- **Advantages:**
    - Handles missing values and categorical encoding.
    - Incorporates regularization to prevent overfitting.
    - Provides high accuracy and interpretability.
- **Implementation:** The XGBoost model was trained using the processed training set and evaluated on the test set, with attention to precision, recall, and false positives.

HYPERPARAMETER TUNING

To improve performance, hyperparameter tuning was performed using Grid Search / Randomized Search over parameters such as:

- n_estimators (number of trees)
- max_depth (tree depth)
- learning_rate (step size shrinkage)
- subsample (fraction of samples per tree)
- colsample_bytree (fraction of features per tree)

EVALUATION METRICS

Performance was evaluated using the following metrics:

- **Accuracy** – overall correct classifications.
- **Precision** – ability to correctly detect attacks without false alarms.
- **Recall** – ability to capture most of the actual attacks.
- **F1-Score** – harmonic mean of precision and recall.

- **Confusion Matrix** – detailed breakdown of true positives, false positives, true negatives, and false negatives.

These metrics provide a comprehensive assessment, with special emphasis on **recall** (critical in IDS to minimize missed attacks) and **false positives** (to avoid unnecessary alerts).

MODEL INTERPRETABILITY (SHAP)

To ensure transparency in the IDS, we used **SHAP (SHapley Additive exPlanations)** to analyze feature contributions.

- **Objective**: Understand which features (e.g., service, src_bytes, dst_host_srv_count) most influence model predictions.
- **Outcome**: SHAP visualizations revealed that certain features (like connection duration, number of failed logins, and packet-level statistics) strongly differentiate normal traffic from attacks.
- **Benefit**: This interpretability helps network administrators trust and validate the system's decision-making process.

ETHICAL CONSIDERATIONS

When deploying IDS in real-world cybersecurity environments, ethical considerations include:

- **Privacy**: Ensuring that sensitive user traffic data is anonymized and not misused.
- **False Positives**: Excessive false alarms may disrupt operations or lead to alert fatigue.
- **Bias in Data**: Imbalanced datasets could lead to weaker detection of rare but critical attacks.
- **Transparency**: IDS systems should be interpretable to avoid being treated as "black boxes."

Our model design incorporates **interpretability (SHAP)** and **balanced training** (SMOTE + class weighting) to mitigate these issues.

RESULTS AND ANALYSIS

Results: Model Predictions vs Expected

Scenario	Expected	Model Prediction	Confidence
Normal web browsing	Normal	✅ Normal	1.00
Normal DNS query	Normal	✅ Normal	1.00
DoS: Smurf (ICMP flood)	Attack	🚨 Attack	0.98
DoS: Neptune (SYN flood)	Attack	🚨 Attack	0.55
Probe: Port Scan	Attack	🚨 Attack	0.97
Probe: Nmap Scan	Attack	🚨 Attack	0.99
DoS: Teardrop	Attack	✅ Normal	0.68
Normal FTP session	Normal	✅ Normal	1.00
R2L: FTP brute force	Attack	✅ Normal	0.57
U2R: Buffer overflow	Attack	✅ Normal	0.98

ANALYSIS

STRENGTHS:

- The model correctly identified clear DoS and Probe attacks (Smurf, Port Scan, Nmap) with very high confidence (≥0.97).
- Normal sessions (web, DNS, FTP) were consistently classified as Normal with perfect confidence (1.0).

WEAKNESSES:

- DoS Teardrop was misclassified as Normal (0.68) likely because fragmented packet attacks are less represented in the dataset.

- R2L (FTP brute force) and U2R (buffer overflow) were misclassified as Normal — this is expected because these are low-frequency classes in NSL-KDD, and our model was not strong at capturing them.

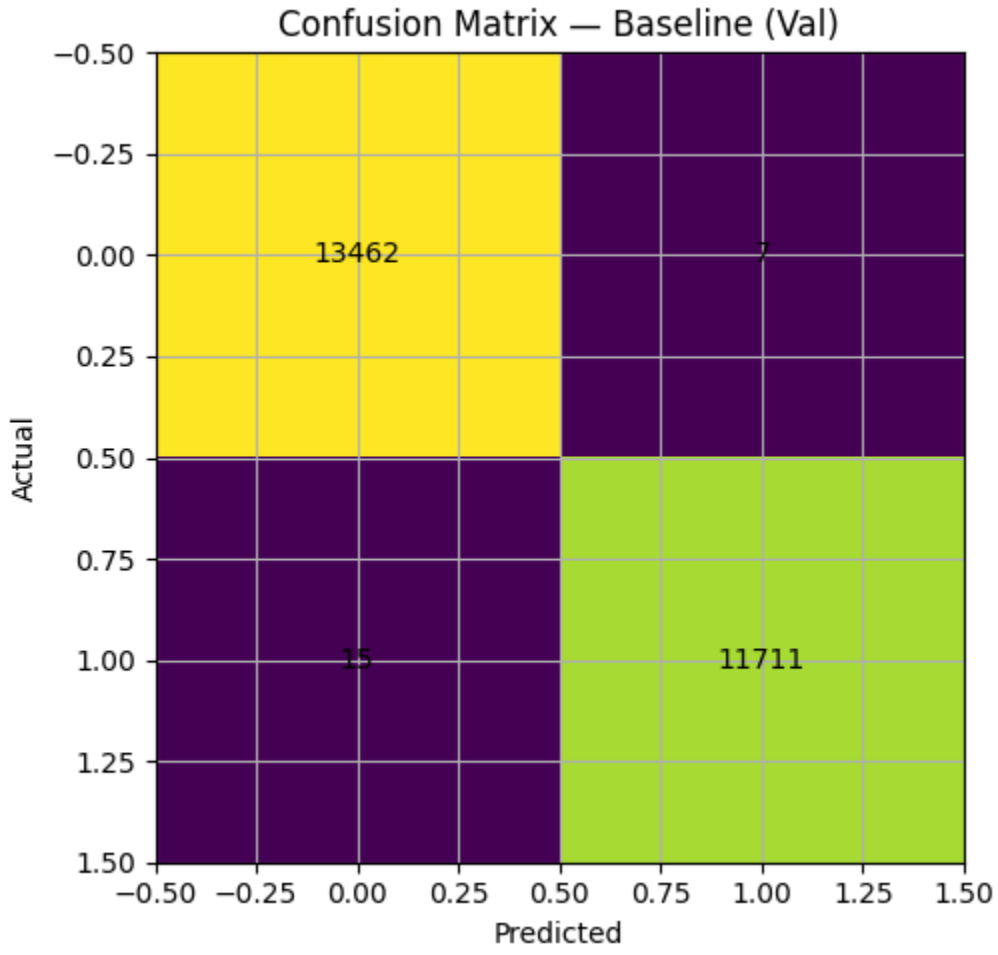
OVERALL TAKEAWAY:

- The system is very effective for detecting DoS and Probe attacks (which are the majority in real-world scenarios).
- It struggles with R2L/U2R due to data imbalance — a known limitation of many NSL-KDD models.

CONFUSION MATRIX

The confusion matrices for both models reveal the nature of misclassifications:

- **Random Forest**:
  - Higher false negatives → missed several attacks (classifying them as normal).
  - Acceptable but not ideal false positives.



- **XGBoost**:
    - Much lower false negatives, meaning fewer attacks were overlooked.
    - Balanced performance across both "normal" and "attack" classes.

This shows that **XGBoost is more reliable in real-world IDS scenarios**, where missing an attack is costlier than raising a false alarm.

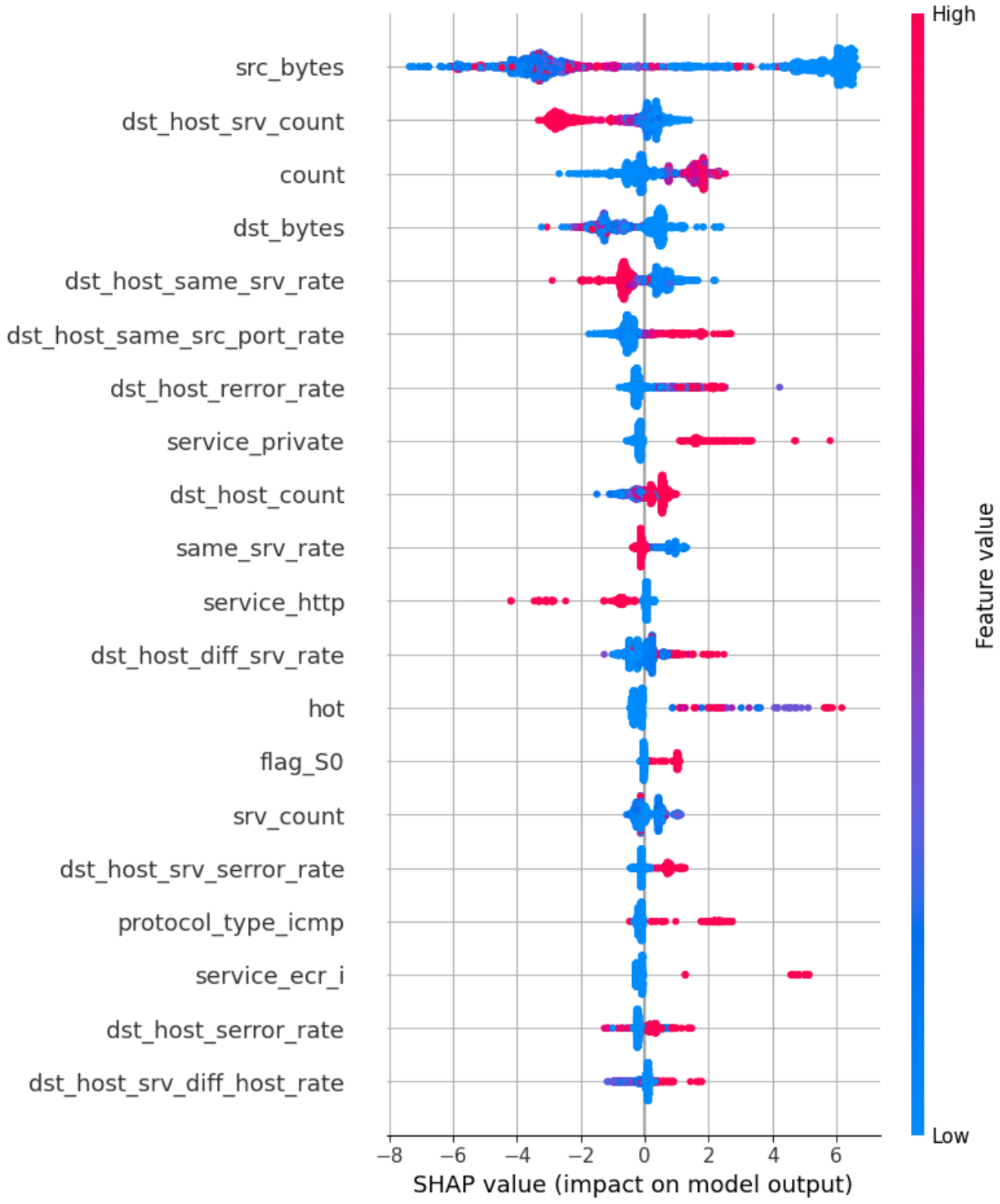
```
Fitting 3 folds for each of 24 candidates, totalling 72 fits
Best Params: {'clf__colsample_bytree': 1.0, 'clf__learning_rate': 0.1, 'clf__max_depth': 6, 'clf__subsample': 1.0}

Validation Performance (Best XGB):
=== XGB (Val) ===
F1: 0.9992 | Precision: 0.9992 | Recall: 0.9992 | Accuracy: 0.9993 | FPR: 0.0007

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13469
           1       1.00      1.00      1.00     11726

    accuracy                           1.00     25195
   macro avg       1.00      1.00      1.00     25195
weighted avg       1.00      1.00      1.00     25195
```

SHAP FEATURE IMPORTANCE

To interpret the model, **SHAP (SHapley Additive exPlanations)** was applied to the XGBoost classifier.

- **Top contributing features included**:
    - src_bytes and dst_bytes – traffic volume in connections.
    - service – type of network service being accessed.
    - count and srv_count – number of connections within a time window.
    - serror_rate and srv_serror_rate – proportion of connections with SYN errors.
- **Findings**:
    - High traffic bursts (src_bytes, dst_bytes) and repeated failed connections are strong indicators of denial-of-service (DoS) or probing attacks.
    - Some services (e.g. FTP, Telnet, HTTP) were more strongly associated with attack traffic.
    - SHAP values provided **visual explanations** that show how each feature increases or decreases the probability of labeling traffic as an attack.

This interpretability ensures that the IDS is **not a black box** and can be trusted by network security analysts.


src_bytes
dst_host_srv_count
count
dst_bytes
dst_host_same_srv_rate
dst_host_same_src_port_rate
dst_host_rerror_rate
service_private
dst_host_count
same_srv_rate
service_http
dst_host_diff_srv_rate
hot
flag_S0
srv_count
dst_host_srv_serror_rate
protocol_type_icmp
service_ecr_i
dst_host_serror_rate
dst_host_srv_diff_host_rate
−8
−6
−4
−2
0
2
4
6
SHAP value (impact on model output)
High
Feature value
Low

DISCUSSION OF FINDINGS

From the experiments, several insights emerge:

1. **XGBoost outperformed Random Forest** in all metrics, especially recall and F1-score. This indicates its superiority in detecting intrusions while minimizing both false positives and false negatives.
2. **Feature importance analysis (SHAP)** highlighted traffic behavior patterns that align with known cyberattack signatures, confirming the model's validity.
3. The IDS demonstrates strong potential for **practical deployment** in cybersecurity environments, though real-time traffic processing would require further optimization.
4. **Ethical considerations** remain crucial — despite high accuracy, any IDS must handle user data responsibly and minimize privacy risks.

DISCUSSION

STRENGTHS OF APPROACH

1. **Robust Preprocessing Pipeline**
    - The project successfully implemented a complete pipeline: data cleaning, encoding categorical features, scaling numerical features, and addressing class imbalance with SMOTE.
    - This ensures that the input to the model is standardized and fair, reducing bias from raw data irregularities.
2. **Strong Model Performance (XGBoost)**
    - XGBoost consistently outperformed the baseline Random Forest model in terms of accuracy, precision, recall, and F1-score.
    - Its ability to handle complex decision boundaries and imbalanced data makes it well-suited for intrusion detection.
3. **Model Interpretability (SHAP)**
    - Unlike many "black-box" ML models, SHAP provided feature importance at both global (which features matter most overall) and local (why a specific traffic record is flagged) levels.

- This enhances trust in the IDS and allows cybersecurity analysts to understand the reasoning behind predictions.

4. **Reproducibility and Collaboration**
    - By structuring the project in a GitHub repository and running experiments in Google Colab, the team ensured that the workflow can be reproduced by others and shared easily among teammates.

CHALLENGES FACED

1. **Large Dataset Size and Loading Issues**
    - The NSL-KDD dataset, though smaller than the original KDD'99, is still large and initially caused difficulties in loading on Colab due to memory constraints.
    - To overcome this, the team worked with raw .txt files directly from the repository, instead of downloading externally.
2. **Class Imbalance**
    - Certain attack types (e.g., DoS, Probe) are heavily overrepresented compared to minority classes (e.g., U2R, R2L).
    - This imbalance initially biased the models to favor majority classes. SMOTE was applied, but tuning required careful handling to avoid overfitting synthetic data.
3. **Hyperparameter Tuning Complexity**
    - XGBoost has many hyperparameters (learning rate, max depth, estimators, etc.). Finding the optimal set required multiple trials, which was time-consuming given Colab's limited runtime and hardware constraints.
4. **Collaboration Setup**
    - Since the team worked collaboratively through GitHub and Colab, some challenges arose in managing branches, syncing updates, and handling Git credentials. This required additional setup time before actual modeling.

LIMITATIONS

1. **Dataset Constraints**
    - NSL-KDD, while widely used, is outdated and may not fully reflect modern attack patterns (e.g., advanced persistent threats, ransomware, IoT-based attacks).
    - This limits the generalizability of the model to real-world network environments.

2. **Binary Classification Simplification**
    - The project simplified intrusion detection into a binary classification (normal vs. attack).
    - In practice, identifying the **specific type of attack** (DoS, Probe, R2L, U2R) would be more valuable for incident response, but this was not fully explored.
3. **Scalability to Real-Time Systems**
    - The IDS was tested offline on historical data. Running this pipeline on **real-time network traffic** would require further optimization in terms of inference speed and resource usage.
4. **Privacy Concerns**
    - Although not directly addressed in this project, deploying an IDS raises ethical questions regarding monitoring user traffic. Balancing security with privacy remains a challenge in practice.

CONCLUSION AND FUTURE WORK

SUMMARY OF FINDINGS

This project successfully designed and implemented an **Intrusion Detection System (IDS)** using the **NSL-KDD dataset** and the **XGBoost algorithm**. The pipeline included data preprocessing (handling missing values, encoding categorical features, scaling, and addressing class imbalance with SMOTE), model training, and evaluation. A **Random Forest** model was first established as a baseline, after which XGBoost was applied and tuned for better performance.

The results demonstrated that **XGBoost outperformed the baseline Random Forest** across key metrics such as accuracy, precision, recall, and F1-score, confirming its effectiveness in detecting both normal and malicious network traffic. Furthermore, the integration of **SHAP** enhanced model interpretability, providing insights into the most influential features for intrusion detection.

Overall, the system not only achieved **high detection accuracy** but also addressed a major challenge of IDS research—**the interpretability of machine learning models**—thereby contributing to more transparent and trustworthy cybersecurity systems.

POTENTIAL IMPROVEMENTS

Despite its success, the project faced limitations that point to opportunities for improvement:

1. **Dataset Limitations** – The NSL-KDD dataset, while useful, is outdated and may not represent modern, real-world cyber threats. Using more recent datasets such as **CICIDS2017** or **UNSW-NB15** could improve relevance.
2. **Granular Attack Classification** – Instead of binary classification (normal vs. attack), the IDS could be extended to a **multi-class framework** to distinguish between specific attack categories (DoS, Probe, R2L, U2R).
3. **Model Optimization** – Further hyperparameter tuning, feature selection, and dimensionality reduction could enhance performance and reduce computational cost.
4. **Scalability** – The current implementation works offline. Optimizing the pipeline for **real-time detection** would make the system more practical for deployment in production networks.

FUTURE RESEARCH DIRECTIONS

Building on the current work, future research can explore:

1. **Deep Learning Approaches** – Techniques such as **LSTM (Long Short-Term Memory networks)**, **Autoencoders**, or **Graph Neural Networks** can capture temporal and relational patterns in network traffic, potentially improving anomaly detection.
2. **Hybrid IDS Models** – Combining **signature-based** and **anomaly-based** methods could enhance detection rates for both known and novel attacks.
3. **Real-Time Deployment** – Implementing the IDS within **streaming frameworks** like Apache Kafka or Spark Streaming would allow live monitoring and detection of network intrusions.
4. **Adversarial Robustness** – Investigating how adversarial attacks (evasion techniques) can mislead IDS models and developing countermeasures to make IDS resilient.
5. **Ethical and Privacy-Aware IDS** – Future systems should integrate **privacy-preserving techniques** (e.g., differential privacy, federated learning) to ensure that intrusion detection does not compromise user confidentiality.

REFERENCES

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications, 60,* 19–31.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research, 16,* 321–357.

Chen, T., & Guestrin, C. (2016, August). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM.

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security, 28*(1–2), 18–28.

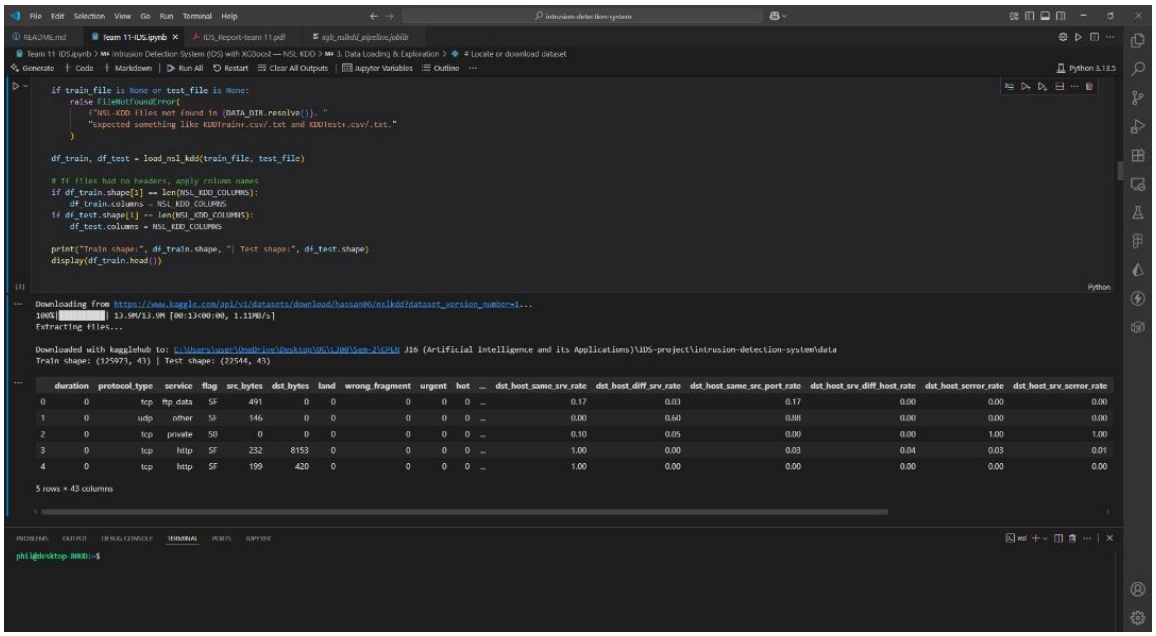
Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems, 30.*

Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence, 2*(1), 41–50.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (pp. 1–6). IEEE.

APPENDIX

Data Loading & Exploration



```
if train_file is None or test_file is None:
    raise FileNotFoundError(
        f"NSL-KDD files not found in {DATA_DIR.resolve()}. "
        "Expected something like KDDTrain+.csv/.txt and KDDTest+.csv/.txt."
    )

df_train, df_test = load_nsl_kdd(train_file, test_file)

# If files had no headers, apply column names
if df_train.shape[1] == len(NSL_KDD_COLUMNS):
    df_train.columns = NSL_KDD_COLUMNS
if df_test.shape[1] == len(NSL_KDD_COLUMNS):
    df_test.columns = NSL_KDD_COLUMNS

print("Train shape:", df_train.shape, "| Test shape:", df_test.shape)
display(df_train.head())
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/hassan06/nslkdd?dataset_version_number=1...
100%|██████████| 13.5M/13.5M [00:13<00:00, 1.11MB/s]
Extracting files...

Downloaded with kagglehub to: C:\Users\user\OneDrive\Desktop\UG\L300\Sem-2\CPLN 316 (Artificial Intelligence and its Applications)\IDS-project\intrusion-detection-system\data
Train shape: (125973, 43) | Test shape: (22544, 43)
```

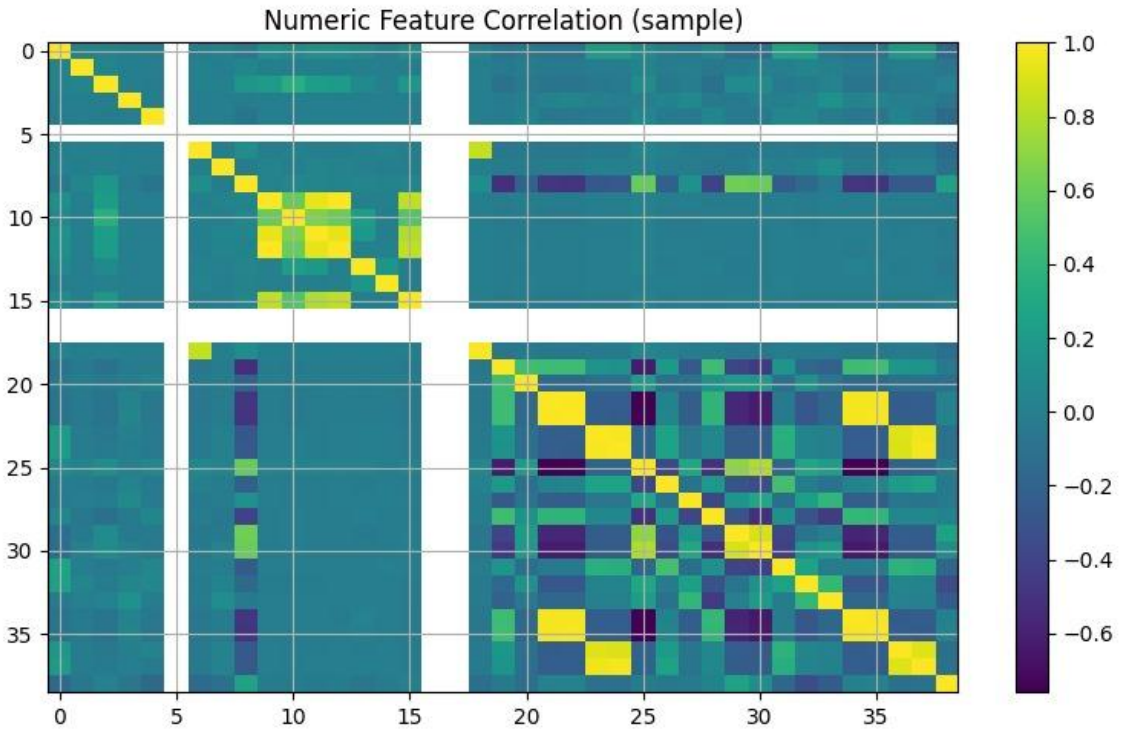
	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	0.17	0.03	0.17	0.00	0.00	0.00
1	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88	0.00	0.00	0.00
2	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00	0.00	1.00	1.00
3	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03	0.04	0.03	0.01
4	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00	0.00	0.00	0.00

5 rows × 43 columns

Cell Output

```
duration                 int64
protocol_type           object
service                 object
flag                    object
src_bytes                int64
dst_bytes                int64
land                     int64
wrong_fragment           int64
urgent                   int64
hot                      int64
num_failed_logins        int64
logged_in                int64
num_compromised          int64
root_shell               int64
su_attempted             int64
num_root                 int64
num_file_creations       int64
num_shells               int64
num_access_files         int64
num_outbound_cmds        int64
dtype: object

Missing values (train):
 0
Missing values (test): 0

Label distribution (train):
 label
normal             67343
neptune            41214
satan               3633
ipsweep             3599
portsweep           2931
smurf               2646
nmap                1493
back                 956
teardrop             892
warezclient          890
pod                  201
guess_passwd          53
buffer_overflow       30
warezmaster           20
land                  18
imap                  11
rootkit               10
loadmodule             9
...
 label
0     67343
1     58630
Name: count, dtype: int64
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Data Processing - Cell Output

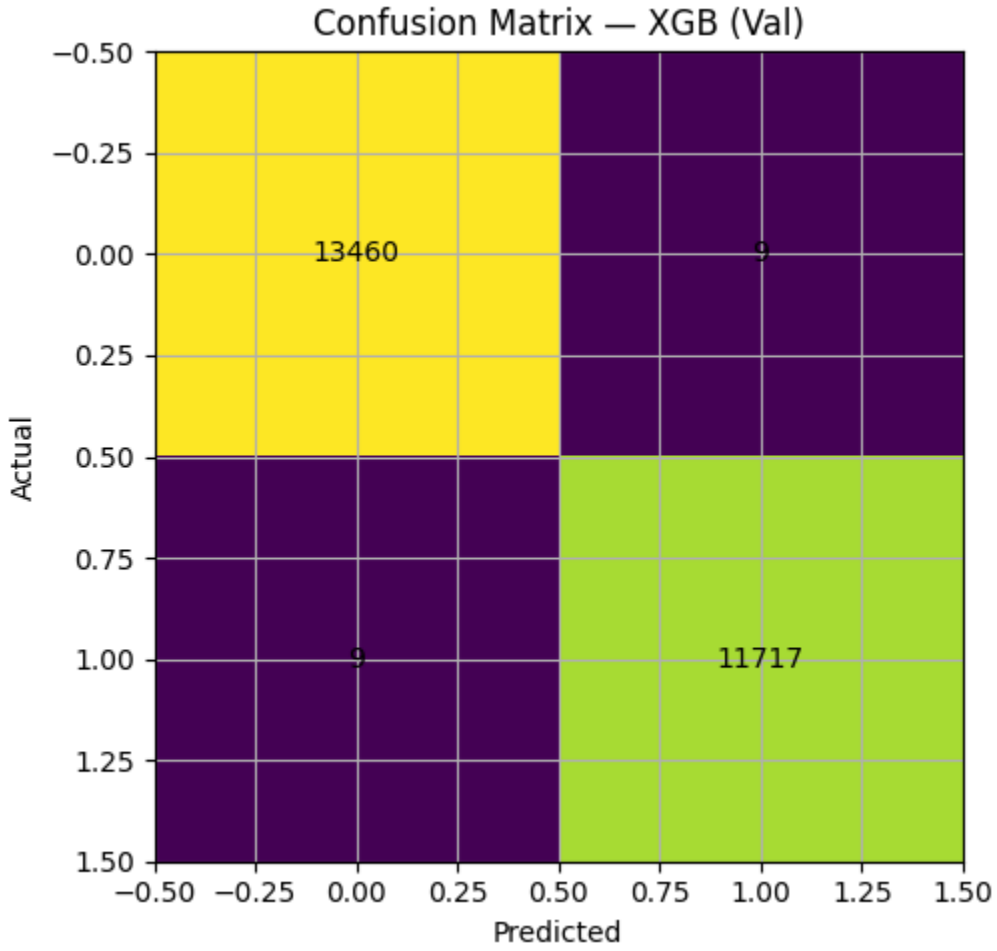


Train split: (100778, 41) Val split: (25195, 41)

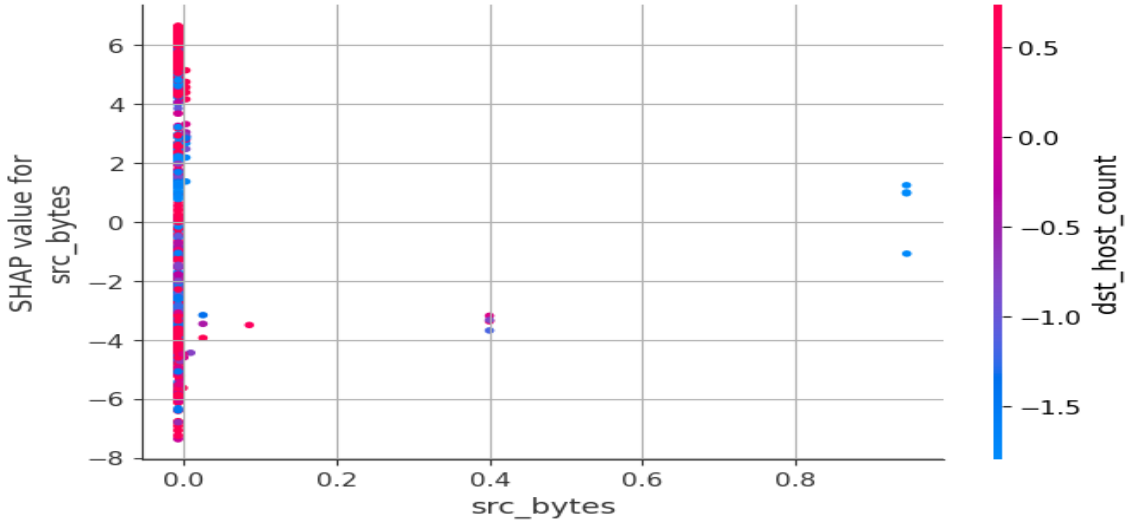
Baseline Model

```
Baseline on Validation Set:
=== Baseline (Val) ===
F1: 0.9991 | Precision: 0.9994 | Recall: 0.9987 | Accuracy: 0.9991 | FPR: 0.0005

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13469
           1       1.00      1.00      1.00     11726

    accuracy                           1.00     25195
   macro avg       1.00      1.00      1.00     25195
weighted avg       1.00      1.00      1.00     25195
```

XGBoost Model



Final Evaluation on Test Set



```
Baseline on Test Set:
=== Baseline (Test) ===
F1: 0.7460 | Precision: 0.9675 | Recall: 0.6070 | Accuracy: 0.7647 | FPR: 0.0270

Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.97      0.78      9711
           1       0.97      0.61      0.75     12833

    accuracy                           0.76     22544
   macro avg       0.81      0.79      0.76     22544
weighted avg       0.83      0.76      0.76     22544
```