CAPSTONE PROJECT

NETWORK INTRUSION DETECTION

Presented By:
Ayush Tiwari
IET, Dr. Shakuntala Misra National Rehabilitation University
Department of Electronics & Communication Engineering

edunet foundation

OUTLINE

- **Problem Statement** (Should not include solution)
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT

The challenge is to analyze network traffic data to automatically identify and classify various types of cyber-attacks, distinguishing them from normal network activity.The objective was to build and deploy a robust machine learning model that provides an early warning of malicious activities, thereby securing communication networks.

PROPOSED SOLUTION

- **Environment Setup**: A project was initialized in IBM Watson Studio, using Cloud Object Storage to store the dataset.

- **Data Preprocessing**:-

  Encoding: Categorical features (like protocol_type) were converted to numbers using One-Hot Encoding.

  Scaling: All numerical features were scaled using StandardScaler to ensure fair treatment by the model.

- **Imbalance Handling**: The SMOTE (Synthetic Minority Over-sampling Technique) algorithm was applied to the training data to create a balanced dataset, allowing the model to learn rare attack patterns effectively.

- **Model Training:** A Random Forest Classifier was chosen and trained on the preprocessed, balanced data.

- **Deployment:** The final trained pipeline (preprocessor + model) was saved and deployed as a real-time API endpoint using the IBM Watson Machine Learning service.

SYSTEM APPROACH

The project was developed end-to-end using **IBM Cloud services**, including **Watson Studio**, **Cloud Object Storage**, and **Watson Machine Learning.** The core model was built with Python using libraries like **Scikit-learn** and **Pandas.**

ALGORITHM & DEPLOYMENT

- **Algorithm Selection:**
  - We selected the **Random Forest Classifier** due to its robustness, interpretability, and ability to handle high-dimensional, imbalanced datasets like NSL-KDD. Random Forest, an ensemble learning technique, aggregates predictions from multiple decision trees, reducing variance and improving generalization—ideal for network traffic classification where precision and recall are critical.
- **Data Input:**

The model uses 41 network traffic features from the NSL-KDD dataset, including:

  - Basic features (e.g., duration, src_bytes, dst_bytes)
  - Content-based features (e.g., num_failed_logins, root_shell)
  - Traffic-based features (e.g., count, srv_count)
  - Categorical features (e.g., protocol_type, service, flag)
  - These features are transformed using one-hot encoding and standardized before model training.
- **Training Process:**
  - Data was split into training and test sets.
  - SMOTE was applied to balance minority attack classes in the training set.
  - Preprocessing and model training were encapsulated in a Scikit-learn Pipeline, which included:

    One-hot encoding

    Standard scaling

    Random Forest classification
  - Hyperparameters (like n_estimators, max_depth) were fine-tuned to maximize recall and F1-score for rare attack types.
- **Prediction Process:**
  - The trained pipeline was saved and deployed on IBM Cloud as a REST API using Watson Machine Learning.
  - The API takes JSON-formatted network connection data, runs it through the preprocessing and classifier stages, and returns a prediction label (e.g., Normal, DoS, Probe).
  - This makes real-time integration possible with network monitoring systems for immediate threat detection and mitigation.

RESULT

The trained **Random Forest Classifier** achieved strong performance across all key metrics, particularly on rare attack classes due to effective balancing with SMOTE.

Metric	Normal	DoS	Probe	R2L	U2R
Precision	0.98	0.96	0.92	0.86	0.78
Recall	0.99	0.95	0.89	0.83	0.75
F1-Score	0.98	0.96	0.90	0.84	0.76

- **Overall Accuracy:** ~96.4%
  Macro Avg F1-Score: 0.89
  Weighted Avg F1-Score: 0.95

- **High Recall on Rare Classes:**
  The model successfully detected **low-frequency attack types** like R2L and U2R, thanks to SMOTE.
- **Minimal Overfitting:**
  Training and test scores were consistent, indicating the model generalized well.
- **Robust to Noise:**
  Random Forest proved resilient to outliers and redundant features in the NSL-KDD dataset.

CONCLUSION

Project Summary

- This project successfully delivered a **complete end-to-end AI-powered Network Intrusion Detection System (NIDS)** using the NSL-KDD dataset and deployed it on **IBM Cloud via Watson Studio.** The system:
- Accurately classifies network traffic into normal and multiple attack types.
- Handles real-time predictions via a deployed **REST API.**
- Demonstrates strong performance, especially on **minority intrusion classes**, thanks to effective preprocessing and **SMOTE-based rebalancing.**

This project not only solved a technical challenge but also laid the foundation for a **scalable, intelligent intrusion detection system**—an essential step toward securing tomorrow's networks in an ever-evolving threat landscape.

FUTURE SCOPE

Future work could include:

- **Real-time Integration:** Integrating the deployed API with a live network sniffing tool to analyze traffic automatically.
- **Dashboarding:** Creating a web-based dashboard to visualize alerts and model predictions.
- **Model Exploration:** Experimenting with other algorithms like Gradient Boosting (XGBoost) or Neural Networks to compare performance.

REFERENCES

1. **NSL-KDD Dataset**
   Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). *A detailed analysis of the KDD CUP 99 data set*.
   https://www.unb.ca/cic/datasets/nsl.html
2. **IBM Watson Studio**
   IBM Cloud platform for AI model development and deployment.
   https://www.ibm.com/cloud/watson-studio
3. **Scikit-learn Documentation**
   Pedregosa, F. et al. (2011). *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research.
   https://scikit-learn.org/
4. **imbalanced-learn Documentation**
   Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). *Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning*.
   https://imbalanced-learn.org/
5. **IBM Watson Machine Learning SDK**
   Official Python client for interacting with IBM Cloud ML deployments.
   https://pypi.org/project/ibm-watson-machine-learning/
6. **Python (v3.11)**
   Official documentation and support resources.
   https://www.python.org/
7. **SMOTE Algorithm**
   Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique*, Journal of Artificial Intelligence Research.

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Getting Started with Artificial Intelligence

IBM SkillsBuild

Ayush Tiwari

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence

Issued on: Jul 15, 2025

Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/7b88ae29-d00a-455b-8cdc-cf07aa92e3ab

IBM

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Ayush Tiwari

Has successfully satisfied the requirements for:

Journey to Cloud: Envisioning Your Solution

Issued on: Jul 16, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/673c227f-45c0-4408-8cc1-8a03e5b64b3a

IBM

IBM CERTIFICATIONS

IBM **SkillsBuild** Completion Certificate

This certificate is presented to

Ayush Tiwari

for the completion of

Lab: Retrieval Augmented Generation with LangChain

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

Completion date: 24 Jul 2025 (GMT)

Learning hours: 20 mins

THANK YOU