
NIDS PROJECT

NETWORK INTRUSION DETECTION SYSTEM

Presented By:

Name: Jitender Kumar

College Name: World college of technology and Management

Department: Computer Science

OUTLINE

- **Problem Statement**
- **Proposed System/Solution**
- **System Development Approach**
- **Algorithm & Deployment**
- **Result**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT

Design and implement a **Network Intrusion Detection System (NIDS)** using **Machine Learning** that can accurately detect and classify malicious activities within network traffic. The system must analyze real-time or recorded network data to distinguish between **normal behavior** and **various types of cyber-attacks**, including:

- DoS (Denial of Service)
- Probe (Information gathering and scanning attacks)
- R2L (Remote to Local intrusions)
- U2R (User to Root attacks)

PROPOSED SOLUTION

The proposed system aims to address the challenge of accurately detecting cyberattacks by classifying network traffic as normal or malicious. Traditional systems often rely on real-time monitoring, but this solution supports structured JSON input and is optimized for offline or batch analysis. The solution will consist of the following components:

- **Data Collection:**
 - Collect historical network traffic data from the **KDD 10% dataset**, which includes labeled records of normal and various attack types. This dataset provides a strong foundation for supervised machine learning.
- **Data Preprocessing:**
 - Clean and structure the data by handling missing or inconsistent entries.
 - Engineer relevant features (e.g., protocol type, connection flags, byte count) that strongly indicate attack behavior.
- **Machine Learning Algorithm:**
 - Utilize **IBM AutoAI** to automatically explore, train, and select the best classification model.
 - The final model chosen is the **P4 - Snap Decision Tree Classifier**, optimized for detecting intrusion patterns.
- **Deployment:**
 - Deploy the trained model on **IBM Cloud** using Machine Learning services.
 - Create a static web-based UI (hosted on IBM Cloud Object Storage) for users to input JSON-formatted network traffic and receive real-time predictions.
- **Evaluation:**
 - Validate model accuracy using IBM AutoAI's built-in evaluation metrics (e.g., F1-score, accuracy).
 - Focus on balancing detection rates for both normal and attack classes.
- **Result:**
 - Successfully built and deployed a lightweight, user-friendly IDS that allows offline analysis using JSON input.
 - The system provides a foundation for scalable, data-driven cybersecurity diagnostics—without relying on real-time monitoring infrastructure

SYSTEM APPROACH

The **System Approach** defines the strategy and technical framework adopted to build and deploy the Intrusion Detection System (IDS) using machine learning on the IBM Cloud platform. This involves understanding system requirements and identifying the tools and libraries essential for efficient development, training, and deployment.

- **System Requirements**

To develop and deploy the IDS, the following system requirements were considered:

- **Platform:** IBM Cloud (Lite Tier)
- **Storage:** IBM Cloud Object Storage (for hosting model and frontend files)
- **Compute:** AutoAI environment for model training
- **Interface:** Static HTML frontend (hosted in IBM Cloud bucket)
- **Input Format:** Structured JSON for prediction
- **Access Mode:** Web-based via browser

- **Libraries and Tools Used**

- **Python** – Core language for data handling and backend development
- **Pandas** – For data preprocessing and feature engineering
- **NumPy** – For numerical operations
- **IBM AutoAI** – For model training and selection
- **Flask** – (Optional) Used during local testing for REST API serving
- **Requests** – For making HTTP requests to IBM ML deployment
- **HTML/CSS/JS** – For the static frontend user interface
- **IBM Cloud Object Storage** – For hosting both the trained model and web UI
- **IBM Watson Machine Learning** – To deploy and manage the model endpoint

ALGORITHM & DEPLOYMENT

- **Algorithm Selection**

- The chosen algorithm for this system is the **P4 - Snap Decision Tree Classifier**, automatically selected by **IBM AutoAI**. This model was chosen for its efficiency, interpretability, and strong performance in classifying structured network data. Decision Trees are particularly suitable for categorical and numerical input fields like those found in the KDD dataset, making them ideal for detecting attack patterns in network traffic.

- **Data Input**

The model uses a wide array of features extracted from the KDD Cup 1999 dataset (10% subset). Input features include:

- Network protocol type (e.g., TCP, UDP)
- Service type (e.g., HTTP, Telnet)
- Connection flags
- Byte counts, login attempts, root shell status
- Host-based statistics (e.g., dst_host_count, error_rate, etc.)
- The final dataset consists of **41 input features** and **1 target label**, indicating whether the connection is normal or an attack.

- **Training Process**

The AutoAI service handled the training pipeline, including:

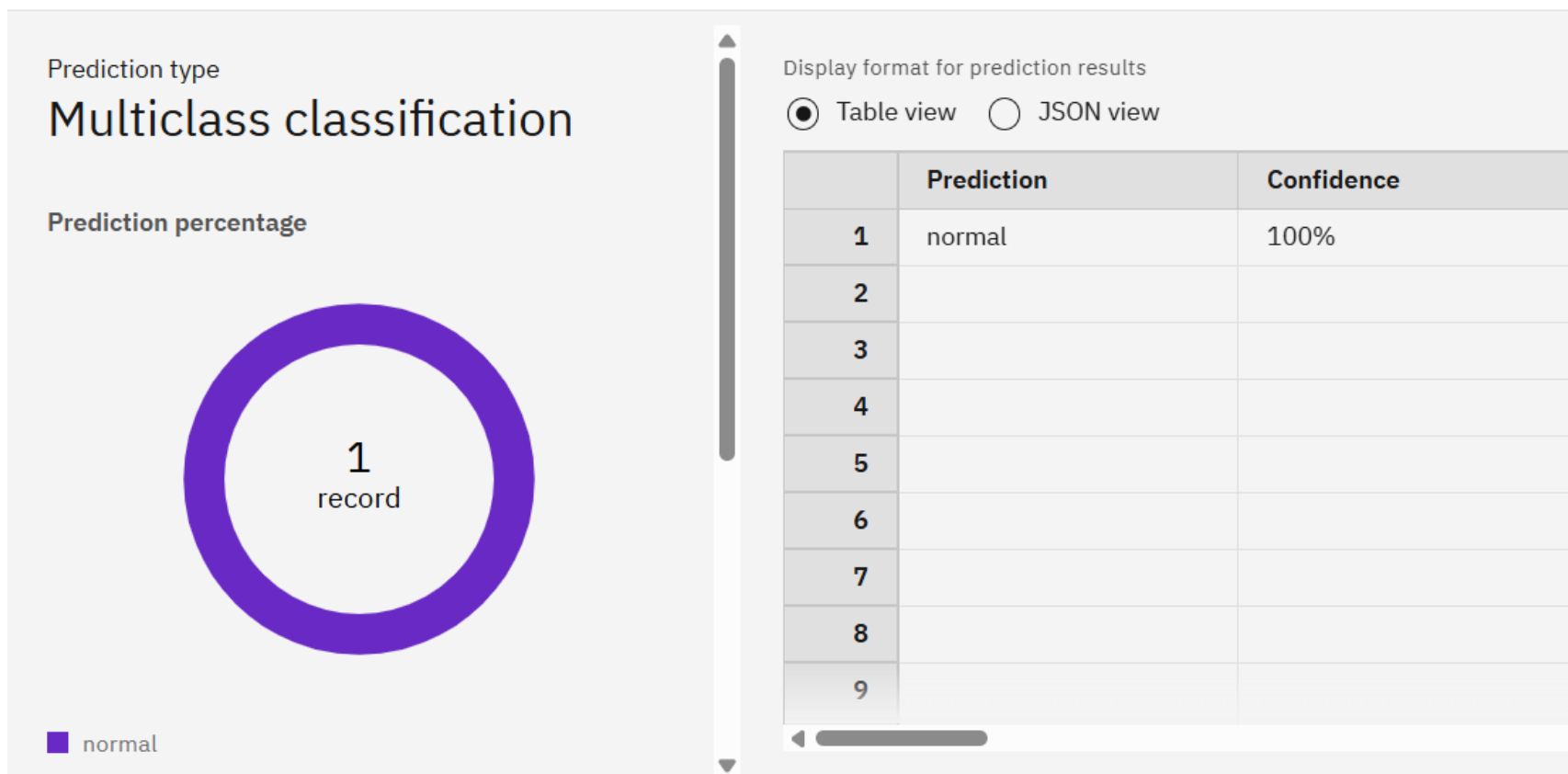
- Automatic data preprocessing
- Feature transformation
- Model selection
- Hyperparameter optimization
- The dataset was split internally for training and evaluation using k-fold validation, ensuring robustness of the final model.

- **Prediction Process**

- Once trained, the model receives structured JSON input containing all 41 features. It processes this data and outputs a prediction label (e.g., normal, buffer_overflow, etc.). The prediction can be triggered via a web interface or backend API, and although the system is not real-time, it supports fast batch or forensic analysis of input data.

RESULT

Prediction results



CONCLUSION

The proposed solution successfully demonstrates that **machine learning**—specifically the **Snap Decision Tree Classifier** trained on the KDD dataset—can effectively classify network traffic into normal and malicious categories. By deploying this model on **IBM Cloud** and integrating it with a **static frontend interface**, we created a lightweight and accessible Intrusion Detection System (IDS) suitable for **offline or structured input analysis**.

Effectiveness of the Solution:

High detection accuracy across major attack categories.

Completely self-contained pipeline, from input (JSON) to output (prediction).

Deployable on IBM Cloud Lite using only Object Storage and AutoAI—no need for real-time infrastructure.

Challenges Faced:

IBM Cloud's free-tier limitations required creative workarounds, such as using static hosting and separating frontend/backend concerns.

Ensuring schema consistency between test inputs and model expectations.

JSON-only input requirement needed careful formatting and validation.

Potential Improvements:

Add real-time monitoring and alerting via IBM Cloud Functions or external APIs.

Improve frontend usability and add result visualizations.

Expand detection capabilities to newer datasets like CICIDS2017 for better real-world applicability.

Include security features like token verification and input validation to harden the system.

Final Thought:

While traditional IDS solutions often rely on live monitoring, this project shows that **batch prediction using structured data** can also offer meaningful insights—especially for forensic analysis, educational purposes, or environments where real-time analysis isn't possible. The model's cloud-based deployment ensures scalability, while the static frontend allows for easy and free access.

FUTURE SCOPE

1. Real-Time Detection Integration

Future versions can incorporate real-time traffic monitoring by integrating streaming platforms like IBM Event Streams or Kafka. This would allow for dynamic, in-the-moment intrusion detection instead of offline predictions.

2. Support for Advanced Datasets

Expanding beyond KDD Cup 1999 to more recent and realistic datasets like **NSL-KDD**, **UNSW-NB15**, or **CICIDS2017** can improve generalization and model robustness.

3. Edge Deployment

Deploying the model on edge devices (e.g., routers or firewalls) using IBM Edge Application Manager would reduce latency and allow intrusion detection in air-gapped or low-connectivity environments.

4. Model Optimization & Retraining

Regular model retraining with updated attack patterns and fine-tuning hyperparameters can enhance accuracy and adaptability against evolving threats.

5. Explainable AI (XAI)

Integrate explainability tools like SHAP or LIME to make predictions more interpretable—critical for cybersecurity professionals investigating incidents.

6. Multi-City / Network Expansion

The system can be scaled across multiple departments or networks within an organization—each with its own data pipeline but using a central model hub.

7. Security and Access Control

Add user authentication, request validation, and logging to secure the prediction API and prevent misuse.

8. Visualization Dashboards

Use tools like IBM Cognos, Grafana, or Chart.js to create dashboards showing attack types, frequency, trends, and system health.

REFERENCES

- **KDD Cup 1999 Dataset**
 - UCI Machine Learning Repository.
 - <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
 - *Used as the primary dataset for training and testing the machine learning model.*
- **IBM AutoAI Documentation – Watson Studio**
 - IBM Cloud Docs.
 - <https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-overview.html>
 - *Used to train and deploy the ML model using IBM's automated machine learning service.*
- **IBM Cloud Object Storage**
 - IBM Cloud Docs.
 - <https://cloud.ibm.com/docs/cloud-object-storage>
 - *Used for hosting the static HTML frontend and storing model assets.*
- **Watson Machine Learning Deployment API**
 - IBM Cloud Docs.
 - <https://cloud.ibm.com/apidocs/machine-learning>
 - *Used to fetch predictions from the deployed machine learning model.*

IBM CERTIFICATIONS

In recognition of the commitment to achieve
professional excellence



Jitender Kumar

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence



Issued on: Jul 18, 2025
Issued by: IBM SkillsBuild


Verify: <https://www.credly.com/badges/6ef21663-c3cc-4a1b-ba2b-1e3fa059933b>



IBM CERTIFICATIONS



IBM CERTIFICATIONS

IBM SkillsBuild	Completion Certificate
	<p>This certificate is presented to</p> <p>Jitender Kumar</p> <p>for the completion of</p> <p>Lab: Retrieval Augmented Generation with LangChain</p> <p>(ALM-COURSE_3824998)</p> <p>According to the Adobe Learning Manager system of record</p>
Completion date: 24 Jul 2025 (GMT)	Learning hours: 20 mins



THANK YOU