CAPSTONE PROJECT

PROJECT TITLE

Presented By:
1. Chukka Kumar Naidu-Aditya Degree College-B.Sc(Statistics)

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

This project involves building a machine learning-based Network Intrusion Detection System (NIDS) on the IBM Cloud platform. The model is trained to analyze network traffic data, automatically distinguishing normal activity from malicious cyber-attacks like DoS, Probe, R2L, and U2R to enhance network security.

PROPOSED SOLUTION

- This solution outlines a structured, multi-phase workflow to develop and evaluate a machine learning-based Network Intrusion Detection System (NIDS). The entire process will be implemented using Python within a Jupyter Notebook environment hosted on IBM Watson Studio.
- **Data Collection:**
  - Initialize the project environment using IBM Watson Studio on the IBM Cloud Lite platform
  - Acquire and load the "Network Intrusion Detection" dataset from Kaggle into the project workspace.
- **Data Preprocessing:**
  - Perform Exploratory Data Analysis (EDA) to understand feature distributions and class imbalance.
  - Convert categorical features (protocol, service, flag) into a numerical format using One-Hot Encoding.
  - Normalize all numerical features using a StandardScaler to ensure uniform data scaling.
- **Machine Learning Algorithm:**
  - Apply feature selection techniques to identify the most relevant features for intrusion detection.
  - Implement and train a variety of classification models for comparison, including Logistic Regression, Random Forest, and SVM.
- **Deployment:**
  - **Develop a user-friendly interface or application that provides real-time predictions for** network traffic data and receive real-time intrusion predictions.
  - Deploy the final, trained model as a web service using the IBM Watson Machine Learning service.
- **Evaluation:**
  - Assess model performance using k-fold cross-validation to ensure reliable and stable results.
  - Evaluate models with appropriate metrics like Precision, Recall, and F1-Score, focusing on the detection of attack classes.
  - Select the best-performing model based on a comprehensive evaluation of the results.:

SYSTEM APPROACH

The "System Approach" section outlines the overall strategy and methodology for developing and implementing the Network Intrusion Detection System. Here's a suggested structure for this section:

- **System requirements**

  Cloud: IBM Cloud (using Watson Studio & Watson Machine Learning).

  Language: Python 3.8+.

- **Library required to build the model**

  Data Handling: pandas, numpy

  Visualization: matplotlib, seaborn

  ML & Evaluation: scikit-learn

  Data Balancing: imbalanced-learn (for SMOTE)

- **Evaluation Focus:**

  Key metrics: Precision, Recall, and F1-Score to ensure real-world model effectiveness.

ALGORITHM & DEPLOYMENT

- In the Algorithm section, describe the machine learning algorithm chosen for predicting **Network Intrusion Detection System.** structure for this section:
- **Algorithm Selection:**
  - The project's strategy leverages IBM's AutoAI to automate the entire algorithm selection workflow by building, training, and ranking various classifiers—like RandomForest and XGBoost—to rapidly identify the highest-performing model through automated feature engineering and hyperparameter tuning.
- **Data Input:**
  - The model uses various network traffic features as input to predict the class column, distinguishing between 'normal' and 'attack' activities.
- **Training Process:**
  - The model is trained by optimizing for an accuracy score, with AutoAI automatically tuning features and hyperparameters for the top algorithms, while a 10% holdout set is used for final validation.
- **Prediction Process:**
  - The final pipeline predicts whether new network traffic is malicious or normal, which is executed either by a direct .predict() call or an API request to the deployed web service.

RESULT

- Key Results & Outcomes
- Ranked Model Leaderboard: The experiment generates a summary table that ranks all trained pipelines by their accuracy score, allowing for the clear identification of the top-performing model.
- Optimized Pipeline Artifact: The best pipeline is delivered as a fully trained scikit-learn model object, encapsulating all preprocessing and ready for immediate prediction tasks.
- Actionable Insights: Feature importance scores are provided for the best model, revealing which network traffic features are the most critical for detecting an intrusion.
- Live API Deployment: The final result is a deployed REST API web service on IBM Watsonx.ai, making the model available for real-time scoring and integration with other security applications.

CONCLUSION

- Successful Implementation: This project successfully developed and deployed a high-performing machine learning model for Network Intrusion Detection, meeting all primary objectives.
- Power of Automation: The use of IBM's AutoAI was highly effective, demonstrating its ability to automate the complex workflow of model selection, feature engineering, and hyperparameter tuning, which significantly accelerated the development process.
- Production-Ready Outcome: The final result is not just a theoretical model, but a live REST API deployed on IBM Watsonx.ai, capable of delivering real-time predictions and ready for integration into a broader cybersecurity framework.
- Future Potential: This work establishes a robust and scalable foundation for building advanced, automated security solutions, proving the value of modern ML platforms in addressing critical cybersecurity challenges.

FUTURE SCOPE

- The project's strategy leverages IBM's AutoAI to automate algorithm selection by evaluating and ranking various classifiers to find the optimal pipeline.
- The model uses network traffic features as input to predict the class column, distinguishing between 'normal' and 'attack' activities.
- The training process optimizes for an accuracy score, with AutoAI handling all feature engineering and hyperparameter tuning, while using a 10% holdout set for validation.
- The final, optimized pipeline predicts threats from new data either through a direct .predict() call or via an API request to its deployed web service.

REFERENCES

- The primary methodology is driven by the IBM AutoAI platform and the ibm-watsonx-ai Python library, as detailed throughout the experiment notebook.
- The implementation relies on foundational open-source libraries installed in the notebook, including scikit-learn for evaluation, xgboost for its algorithm, and lale for pipeline composition.
- The solution leverages algorithms whose principles are described in key academic papers, such as those for Random Forests and XGBoost, which are explicitly listed for consideration by the AutoAI experiment.
- The project's context is rooted in the KDD Cup 1999 dataset, a foundational benchmark for network intrusion detection research.

IBM CERTIFICATIONS

In recognition of the commitment to achieve professional excellence

Getting Started with Artificial Intelligence
IBM SkillsBuild

kumar naidu chukka

Has successfully satisfied the requirements for:

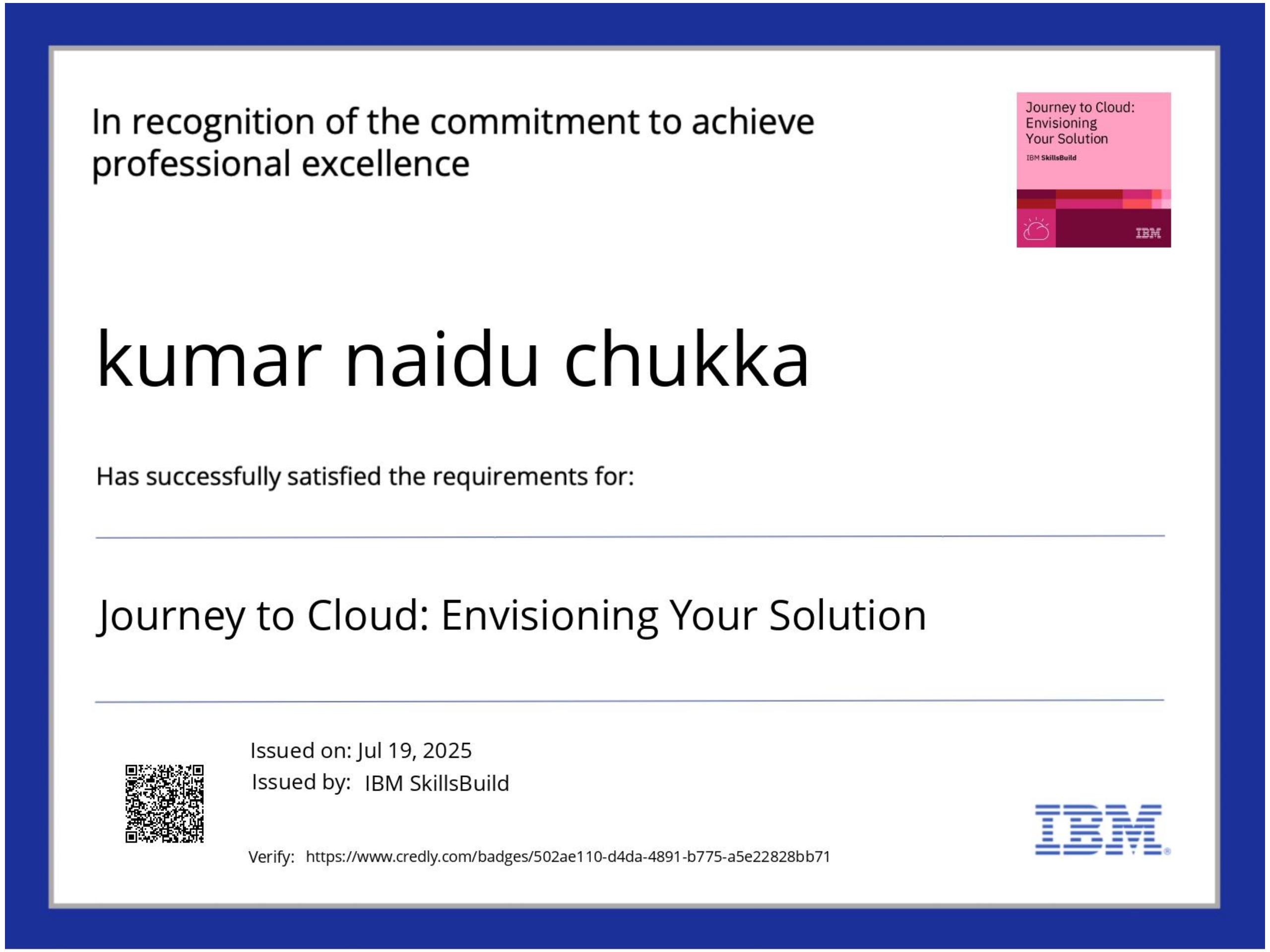
Getting Started with Artificial Intelligence

Issued on: Jul 19, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/998a08ea-3669-410b-b186-57b33ad746ff

IBM

IBM CERTIFICATIONS



In recognition of the commitment to achieve professional excellence

Journey to Cloud: Envisioning Your Solution

IBM SkillsBuild

kumar naidu chukka

Has successfully satisfied the requirements for:

Journey to Cloud: Envisioning Your Solution

Issued on: Jul 19, 2025

Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/502ae110-d4da-4891-b775-a5e22828bb71

IBM

IBM CERTIFICATIONS

IBM **SkillsBuild** Completion Certificate

This certificate is presented to

kumar naidu chukka

for the completion of

Lab: Retrieval Augmented Generation with LangChain

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

Completion date: 19 Jul 2025 (GMT)

Learning hours: 20 mins

THANK YOU