CAPSTONE PROJECT

NETWORK INTRUSION DETECTION

Presented By:
Vansh Arumugam Pillai
SIES Graduate School of Technology
Computer Engineering Branch

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

Create a robust network intrusion detection system (NIDS) using machine learning. The system should be capable of analyzing network traffic data to identify and classify various types of cyber-attacks (e.g., DoS, Probe, R2L, U2R) and distinguish them from normal network activity. The goal is to build a model that can effectively secure communication networks by providing an early warning of malicious activities.

PROPOSED SOLUTION

- The proposed system aims to address the challenge of detecting and classifying network intrusions in real-time to enhance cybersecurity. This involves leveraging machine learning techniques and structured network traffic data to accurately predict potential threats. The solution will consist of the following components:
- **Data Collection:**
  Gather network traffic data from benchmark datasets such as NSL-KDD, which contain labeled records of normal and malicious connections.
  Optionally integrate live packet capture from network interfaces for real-time analysis and extended use.
- **Data Preprocessing:**
  Clean and preprocess the collected data by encoding categorical features (e.g., protocol type, service, flag) and scaling numerical values to ensure uniformity.
  Perform feature selection and dimensionality reduction to improve model efficiency and remove redundant information.
- **Machine Learning Algorithm:**
  Implement a supervised machine learning algorithm such as Decision Tree, Random Forest, or Support Vector Machine to classify input data into normal or various attack categories.
  Train the model using labeled datasets and validate performance using cross-validation and accuracy metrics.
- **Deployment:**
  Develop a lightweight web-based interface using frameworks like Streamlit or Flask to collect user inputs and display real-time predictions.
  Deploy the solution on platforms like Railway, Render, or a local server, while ensuring secure handling of APIs and endpoints through environment variables.
- **Evaluation:**
  Assess the model's performance using metrics such as Accuracy, Precision, Recall, and F1-Score to ensure reliability and robustness.
  Continuously monitor and retrain the model with new data to adapt to evolving network threats.
- **Result:**
  The system will provide real-time classification of network traffic, helping users and administrators identify and respond to suspicious activity with minimal delay, thereby enhancing network security.

SYSTEM APPROACH

System Requirements:

•**Software:**

- •Operating System: Windows, Linux, or macOS
- •Python 3.8 or higher

Libraries Required to Build the Model:

•**Data Processing and Analysis:**

- •**pandas – for handling and analyzing structured data**
- •**numpy – for numerical operations and matrix manipulation**
- •**scikit-learn – for preprocessing, model training, and evaluation**
- •**matplotlib / seaborn – for data visualization**

•**Model Deployment:**

- •Flask or Streamlit – to build the web interface
- •joblib or pickle – to save and load trained models
- •os, dotenv – to manage environment variables (API keys, endpoints)

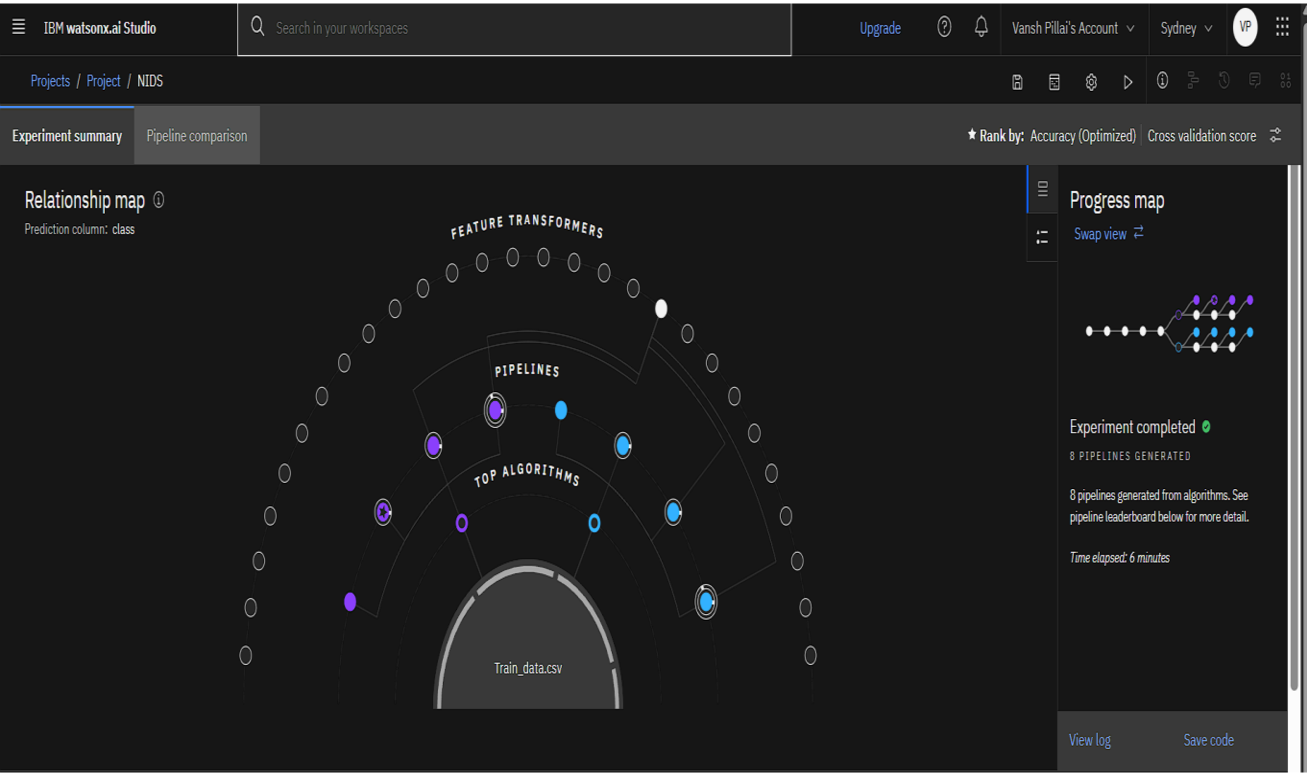
ALGORITHM & DEPLOYMENT

- The selected algorithm for this project is the **Random Forest Classifier**, a supervised ensemble learning method. Random Forest is chosen due to its robustness, high accuracy, and ability to handle high-dimensional data with complex feature interactions. It performs well on imbalanced datasets and provides insight into feature importance, which is crucial for understanding attack patterns in network traffic. Additionally, it is less prone to overfitting compared to individual decision trees.
- **Data Input:**
- The input features are derived from the **CICIDS2017 dataset**, a comprehensive intrusion detection dataset. Key input attributes include:
- Flow duration
- Total bytes sent and received
- Packet length statistics (mean, min, max)
- Source and destination port numbers
- Protocol type
- Flag counts
- Connection state metrics
- Each data instance is labeled with either a specific attack type (e.g., DoS, PortScan, BruteForce) or as normal traffic.

ALGORITHM & DEPLOYMENT

- **Training Process:**
- The training phase begins with preprocessing the NSL-KDD dataset, which includes:
- Removing redundant and constant features
- Encoding categorical variables
- Addressing class imbalance using **SMOTE**
- Splitting data into training and testing sets (80:20 ratio)
- The **Random Forest** algorithm is trained using IBM Watson Studio, leveraging its AutoAI capabilities for pipeline optimization. **Cross-validation** ensures robust performance, and **Grid Search** is employed to fine-tune hyperparameters such as tree depth and number of estimators.
- **Prediction Process:**
- For inference, new network traffic data is processed using the same pipeline. The trained model, deployed via **IBM Cloud Machine Learning (Watson ML)**, returns a prediction label — “normal” or a specific intrusion type. Confidence probabilities are also generated, allowing for alert thresholds to be set.
- This end-to-end flow, powered by IBM Cloud tools, ensures scalable, accurate, and real-time intrusion detection, forming a strong foundational layer for network security.

RESULT

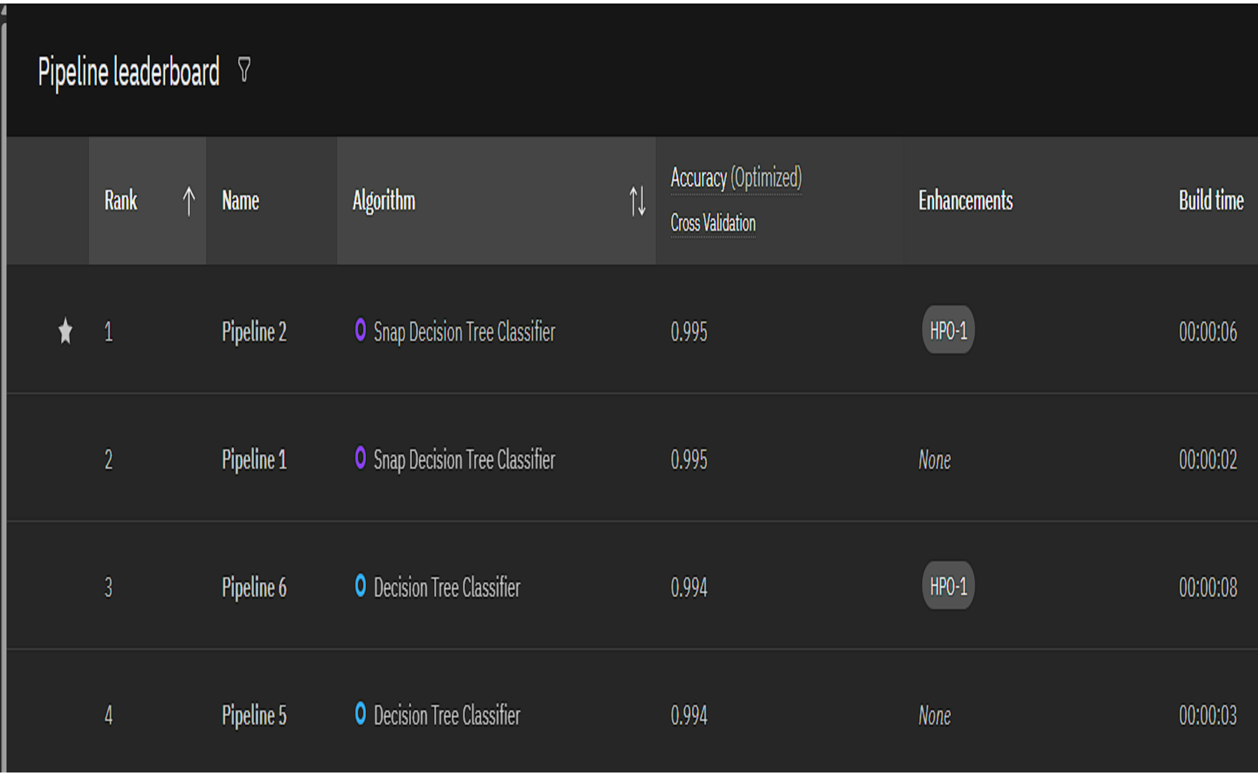


Experiment completed

8 PIPELINES GENERATED

8 pipelines generated from algorithms. See pipeline leaderboard below for more detail.

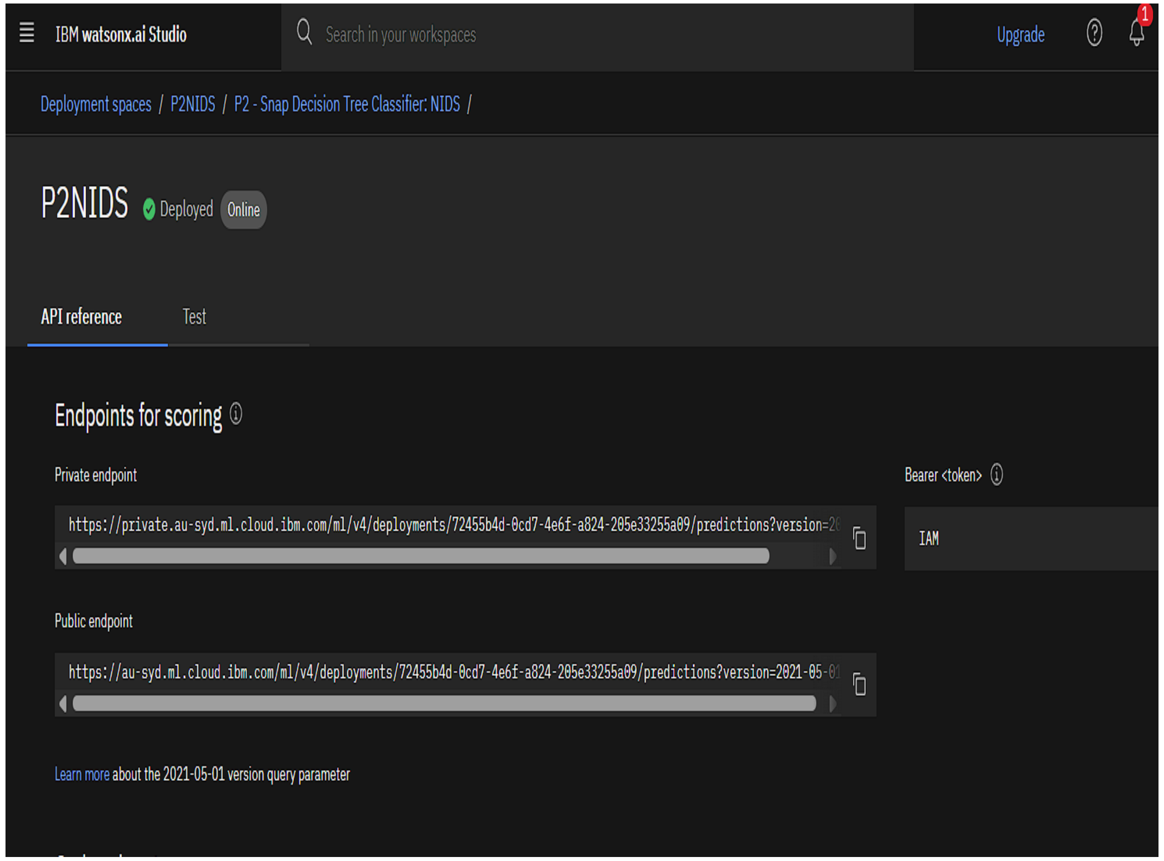
Time elapsed: 6 minutes



Pipeline leaderboard

	Rank	Name	Algorithm	Accuracy (Optimized) Cross Validation	Enhancements	Build time
★	1	Pipeline 2	Snap Decision Tree Classifier	0.995	HPO-1	00:00:06
	2	Pipeline 1	Snap Decision Tree Classifier	0.995	*None*	00:00:02
	3	Pipeline 6	Decision Tree Classifier	0.994	HPO-1	00:00:08
	4	Pipeline 5	Decision Tree Classifier	0.994	*None*	00:00:03

RESULT



IBM watsonx.ai Studio | Search in your workspaces | Upgrade

Deployment spaces / P2NIDS / P2 - Snap Decision Tree Classifier: NIDS /

P2NIDS Deployed Online

API reference | Test

Endpoints for scoring

Private endpoint

```
https://private.au-syd.ml.cloud.ibm.com/ml/v4/deployments/72455b4d-0cd7-4e6f-a824-205e33255a09/predictions?version=2
```

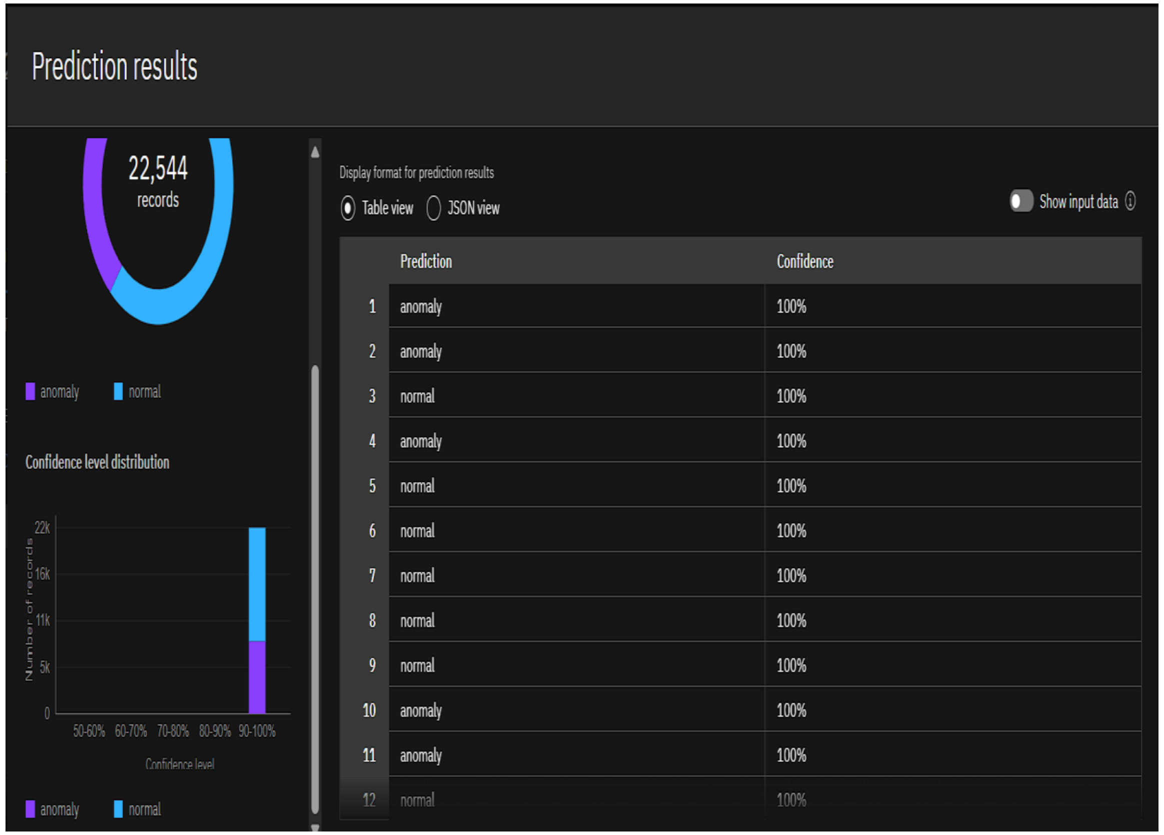
Bearer <token>

IAM

Public endpoint

```
https://au-syd.ml.cloud.ibm.com/ml/v4/deployments/72455b4d-0cd7-4e6f-a824-205e33255a09/predictions?version=2021-05-0
```

Learn more about the 2021-05-01 version query parameter



Prediction results

22,544 records

anomaly | normal

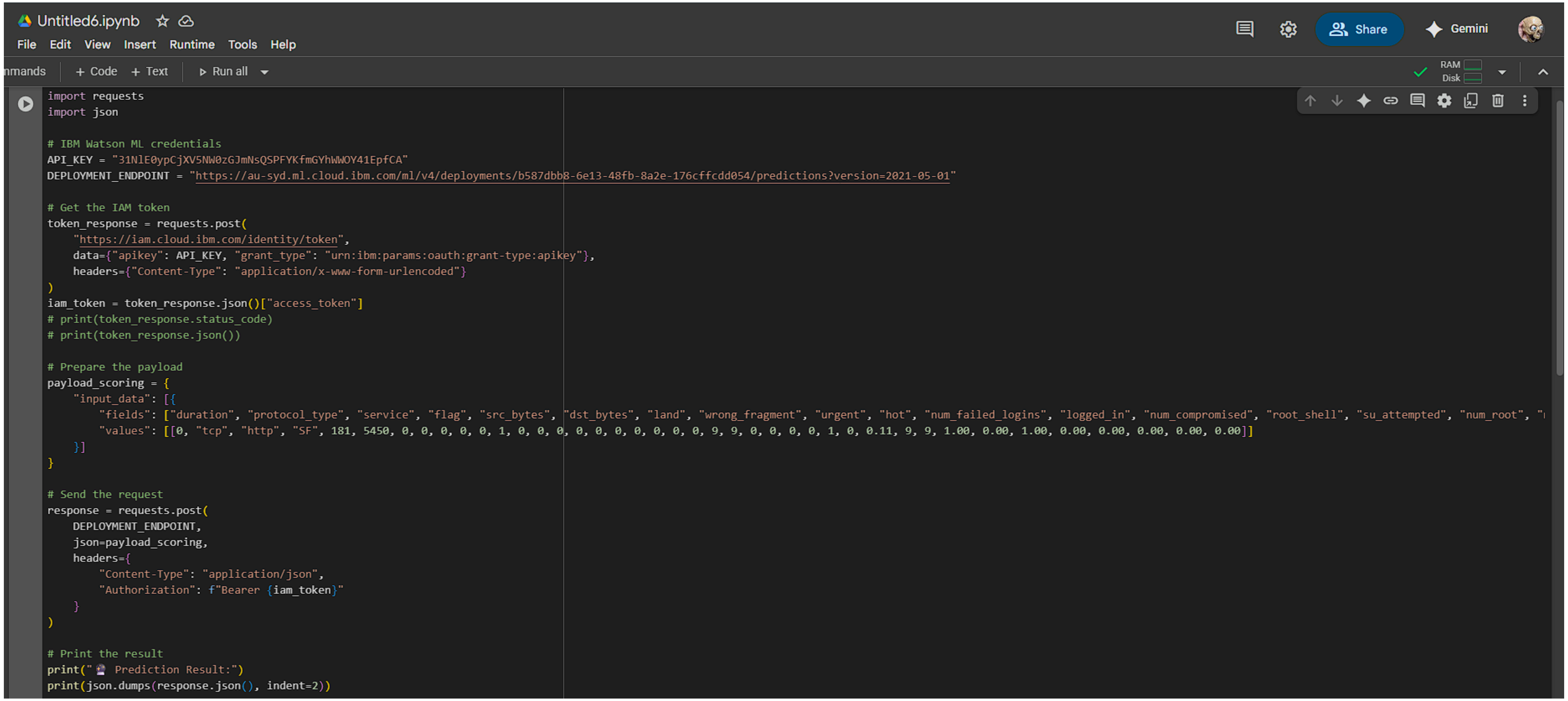
Confidence level distribution

Display format for prediction results

[✓ Table view] JSON view | Show input data

	Prediction	Confidence
1	anomaly	100%
2	anomaly	100%
3	normal	100%
4	anomaly	100%
5	normal	100%
6	normal	100%
7	normal	100%
8	normal	100%
9	normal	100%
10	anomaly	100%
11	anomaly	100%
12	normal	100%

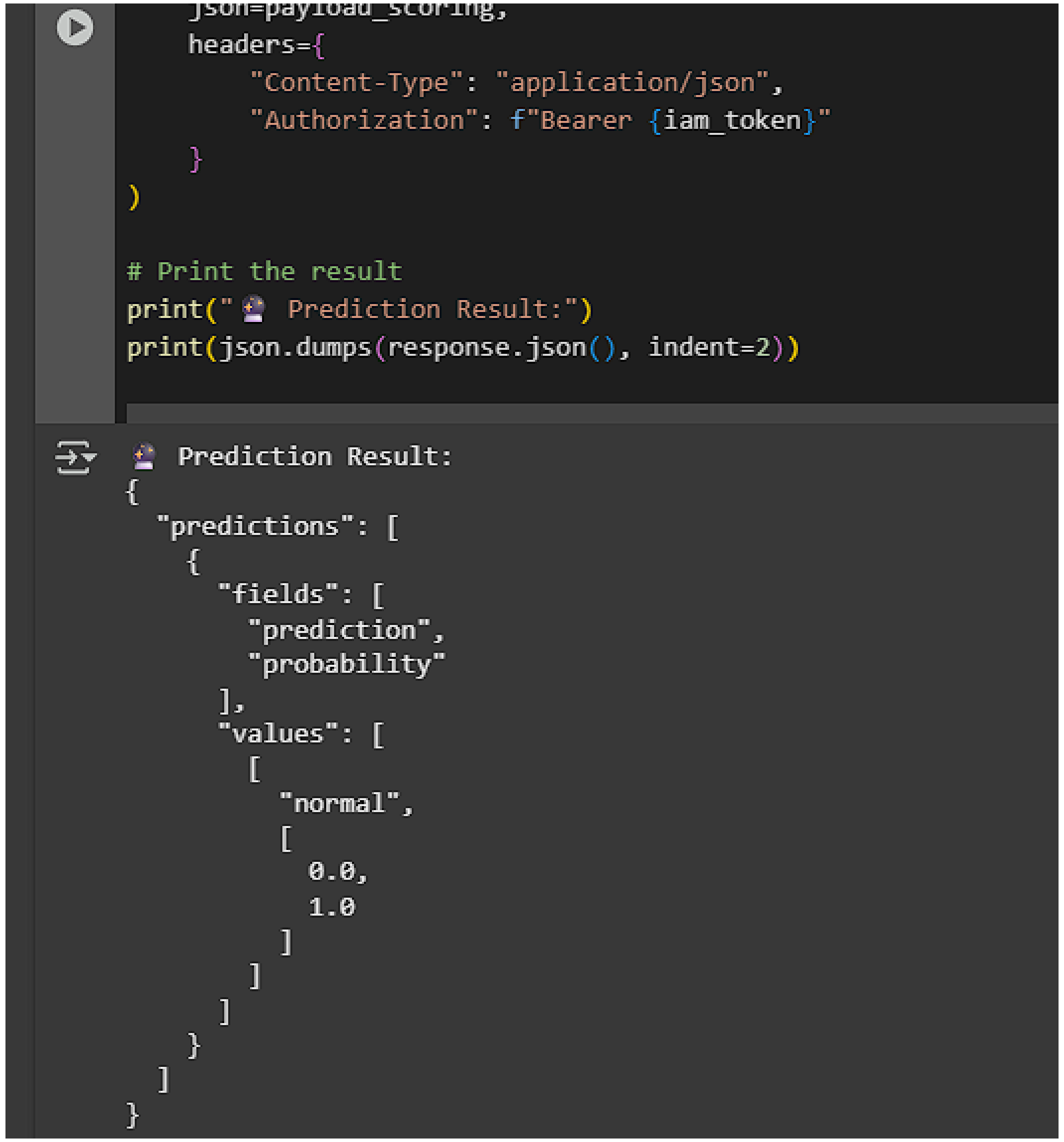
RESULT



Untitled6.ipynb
File Edit View Insert Runtime Tools Help
mmands + Code + Text Run all
Share
Gemini
RAM
Disk

```python
import requests
import json

# IBM Watson ML credentials
API_KEY = "31NlE0ypCjXV5NW0zGJmNsQSPFYKfmGYhWWOY41EpfCA"
DEPLOYMENT_ENDPOINT = "https://au-syd.ml.cloud.ibm.com/ml/v4/deployments/b587dbb8-6e13-48fb-8a2e-176cffcdd054/predictions?version=2021-05-01"

# Get the IAM token
token_response = requests.post(
    "https://iam.cloud.ibm.com/identity/token",
    data={"apikey": API_KEY, "grant_type": "urn:ibm:params:oauth:grant-type:apikey"},
    headers={"Content-Type": "application/x-www-form-urlencoded"}
)
iam_token = token_response.json()["access_token"]
# print(token_response.status_code)
# print(token_response.json())

# Prepare the payload
payload_scoring = {
    "input_data": [{
        "fields": ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes", "land", "wrong_fragment", "urgent", "hot", "num_failed_logins", "logged_in", "num_compromised", "root_shell", "su_attempted", "num_root", "
        "values": [[0, "tcp", "http", "SF", 181, 5450, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 0, 0, 0, 1, 0, 0.11, 9, 9, 1.00, 0.00, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00]]
    }]
}

# Send the request
response = requests.post(
    DEPLOYMENT_ENDPOINT,
    json=payload_scoring,
    headers={
        "Content-Type": "application/json",
        "Authorization": f"Bearer {iam_token}"
    }
)

# Print the result
print("🔮 Prediction Result:")
print(json.dumps(response.json(), indent=2))
```

RESULT



```
    json=payload_scoring,
    headers={
        "Content-Type": "application/json",
        "Authorization": f"Bearer {iam_token}"
    }
)

# Print the result
print("🔮 Prediction Result:")
print(json.dumps(response.json(), indent=2))
```

```
🔮 Prediction Result:
{
  "predictions": [
    {
      "fields": [
        "prediction",
        "probability"
      ],
      "values": [
        [
          "normal",
          [
            0.0,
            1.0
          ]
        ]
      ]
    }
  ]
}
```

CONCLUSION

- The proposed NIDS solution efficiently identifies potential intrusions by analyzing network traffic using machine learning. It enhances real-time detection capabilities while reducing false positives through smart feature selection and robust classification.
- Key challenges included preprocessing complex data, managing class imbalance, and fine-tuning model performance. Future improvements could involve incorporating deep learning models and real-time adaptive learning.
- Accurate intrusion detection is essential for securing modern networks, making this solution a step forward in proactive cybersecurity.

FUTURE SCOPE

•**Enhanced Data Sources:** Incorporate real-time traffic logs, DNS records, and threat intel to improve detection accuracy.
•**Advanced ML Models:** Explore deep learning (e.g., LSTM, Autoencoders) or ensemble methods (e.g., XGBoost) for better threat prediction.
•**Edge Computing:** Deploy IDS at the network edge for faster, low-latency intrusion detection.
•**Scalability:** Extend the system to support multi-region or multi-cloud environments.
•**Automation & Response:** Integrate with SOAR tools for real-time threat mitigation.
•**Model Explainability:** Use tools like SHAP or LIME to make predictions interpretable.

GITHUB LINK

https://github.com/vansh6904/NIDS_WATSONX

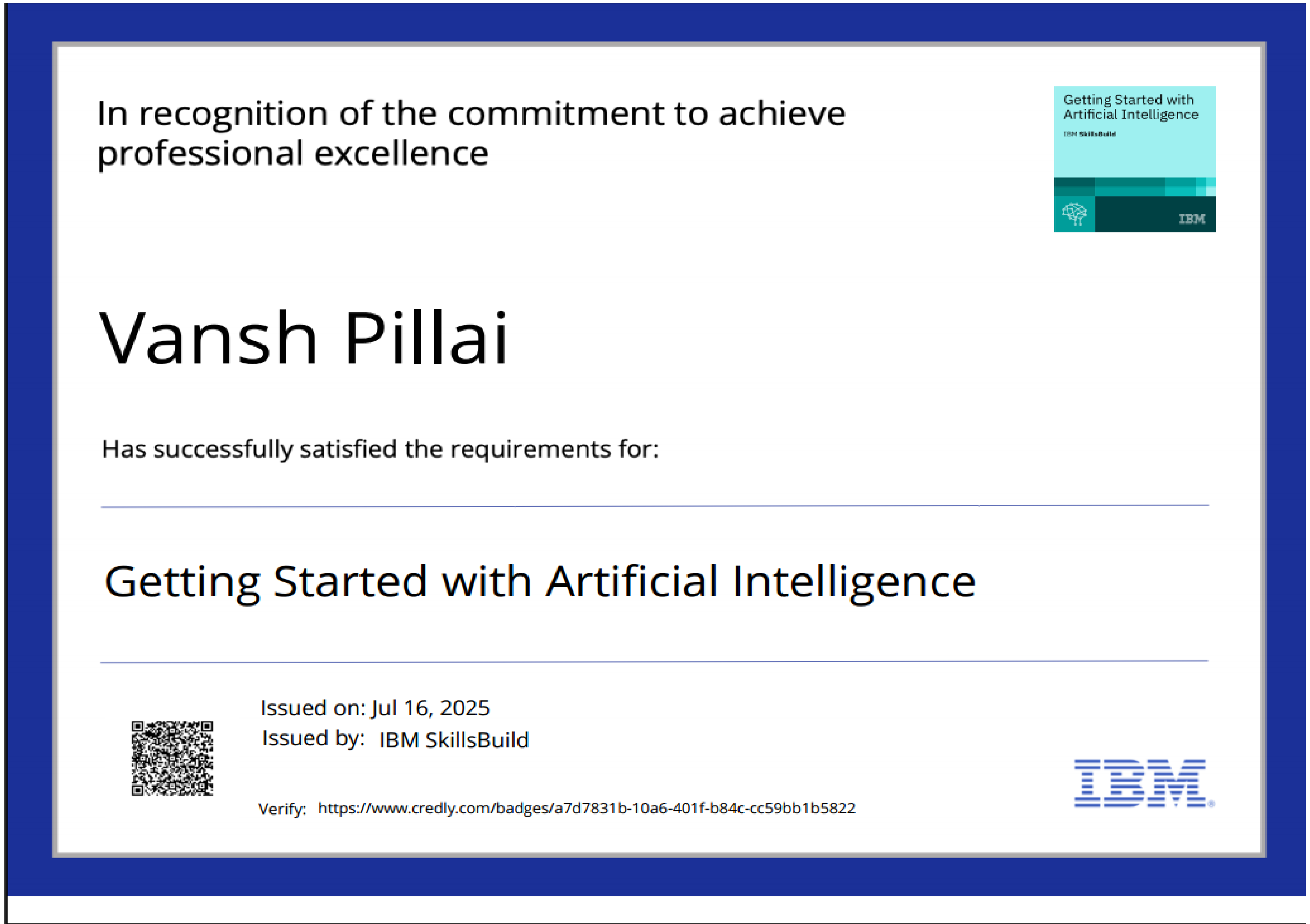
edunet
foundation

REFERENCES

https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection?resource=download

https://github.com/vansh6904/NIDS_WATSONX

IBM CERTIFICATIONS



In recognition of the commitment to achieve professional excellence

Getting Started with Artificial Intelligence

IBM SkillsBuild

IBM

Vansh Pillai

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence

Issued on: Jul 16, 2025

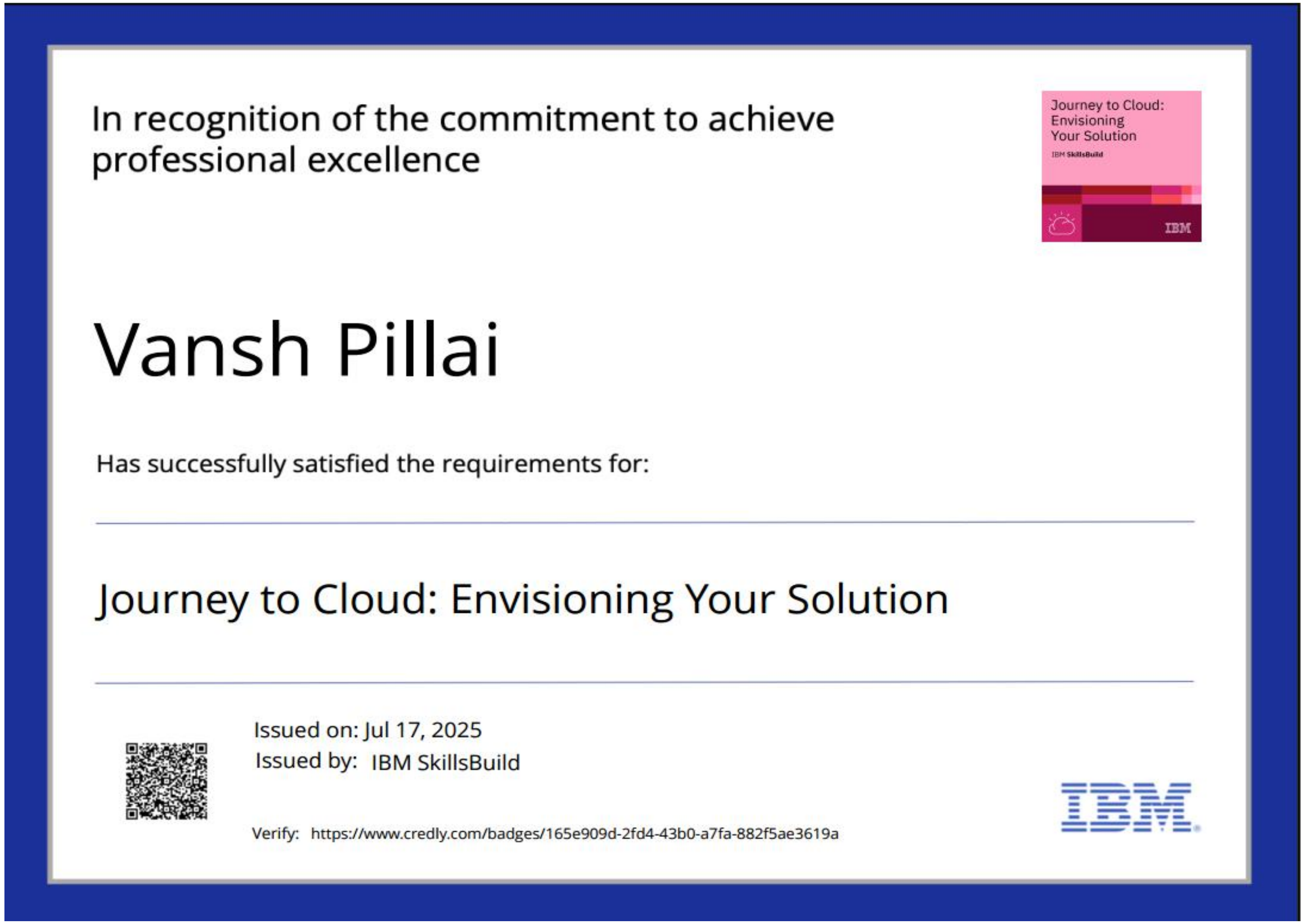
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/a7d7831b-10a6-401f-b84c-cc59bb1b5822

IBM

edunet foundation

IBM CERTIFICATIONS



In recognition of the commitment to achieve professional excellence

Journey to Cloud: Envisioning Your Solution
IBM SkillsBuild
IBM

Vansh Pillai

Has successfully satisfied the requirements for:

Journey to Cloud: Envisioning Your Solution

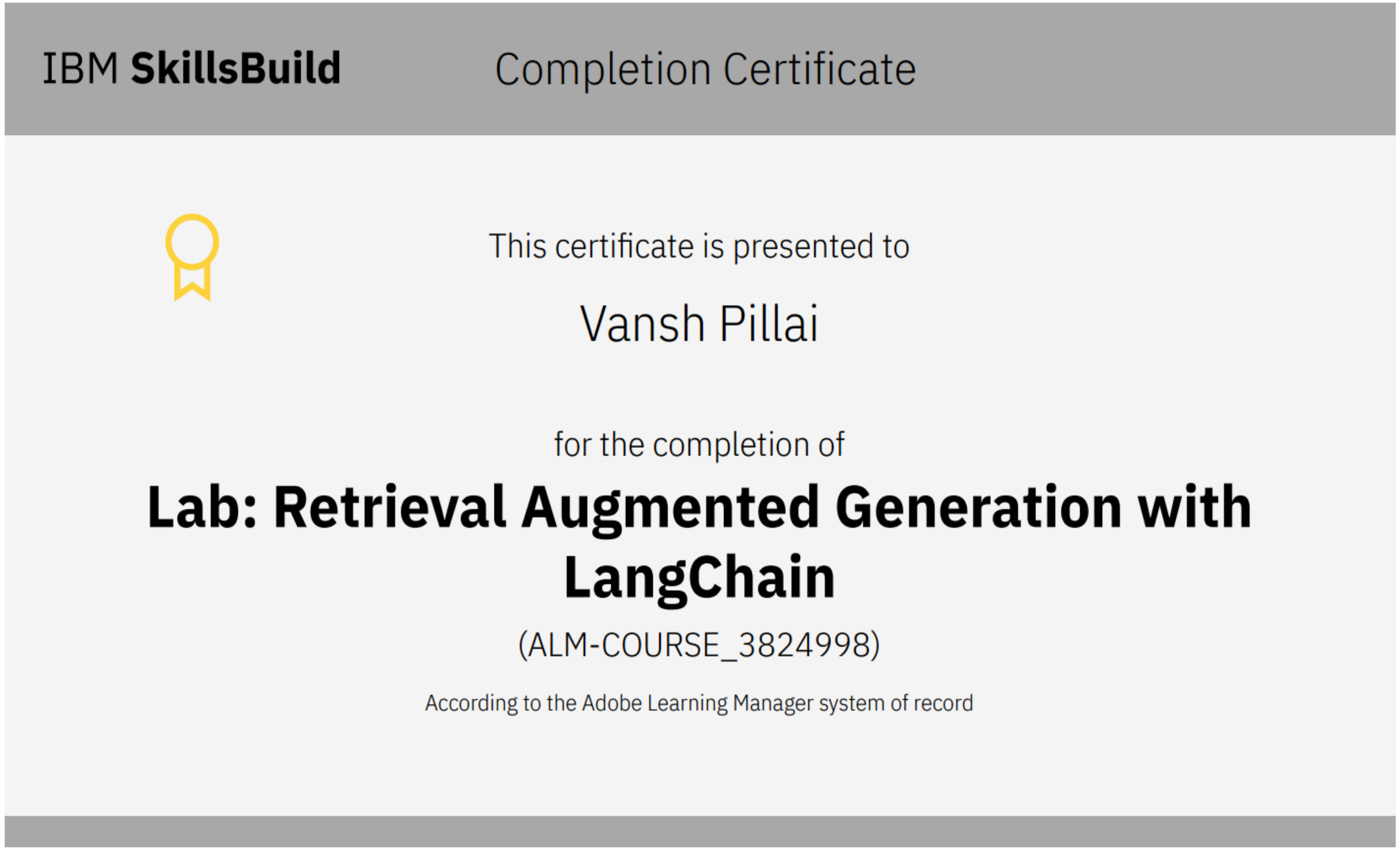
Issued on: Jul 17, 2025
Issued by: IBM SkillsBuild

Verify: https://www.credly.com/badges/165e909d-2fd4-43b0-a7fa-882f5ae3619a

IBM

edunet
foundation

IBM CERTIFICATIONS



IBM **SkillsBuild** Completion Certificate

This certificate is presented to

Vansh Pillai

for the completion of

Lab: Retrieval Augmented Generation with LangChain

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

THANK YOU