
CAPSTONE PROJECT

Network Intrusion Detection

Machine learning project

❖ Presented By:

shu Suresh Yadav-

MIT AOE pune –

Electronics and Telecommunications Engineering (ENTC)

OUTLINE

- **Problem Statement**
- **Proposed System/Solution**
- **System Development Approach (Technology Used)**
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT

- ❖ **Cyber threats like DoS, R2L, and U2R attack network security daily.**
This project aims to build a smart NIDS that detects suspicious activity in real-time using ML techniques—balancing accuracy and minimal false alerts.
Platform: IBM Cloud Lite.
Dataset: Kaggle – NIDS Dataset.

PROPOSED SOLUTION

- To design and implement a machine learning–based Network Intrusion Detection System that can accurately detect and classify cyber threats in real-time, enhancing network security and reducing false alarms.
- Data Collection:
 - Gather labeled network traffic data using datasets like NSL-KDD or custom logs.
 - Include multiple attack types and normal behavior across protocols for robust learning.
- Data Preprocessing:
 - Clean and process raw data to handle missing values and noise.
 - Use feature engineering and dimensionality reduction to optimize input quality.
- Machine Learning Algorithm:
 - Implement classification models such as Decision Tree, Random Forest, or Neural Network.
 - Train using historical attack data and tune hyperparameters for improved accuracy.
- Deployment:
 - Create a responsive interface for real-time intrusion monitoring.
 - Deploy on a scalable infrastructure capable of handling live packet inspection.
- Evaluation:
 - Validate model using metrics like Accuracy, Precision, Recall, and F1-score.
 - Continuously monitor performance to adapt against evolving threats

SYSTEM APPROACH

❖ The “System Approach” section outlines the overall strategy and methodology for developing and implementing the Network Intrusion Detection System. Here's how the project structure is designed

❖ **System Requirements:**

- Python 3.x environment with high computational capability
- A machine or cloud platform capable of handling real-time traffic analysis
- Dataset sources (e.g., NSL-KDD, CICIDS) with labeled network intrusion data

❖ **Libraries Required:**

- pandas, numpy for data handling and preprocessing
- scikit-learn, tensorflow or keras for machine learning and model training
- matplotlib, seaborn for performance visualization
- socket, scapy or similar for live packet sniffing (if deployed in real-time)
- Let me know if you want me to design matching points for your next slide—perhaps on “Model Architecture” or “System

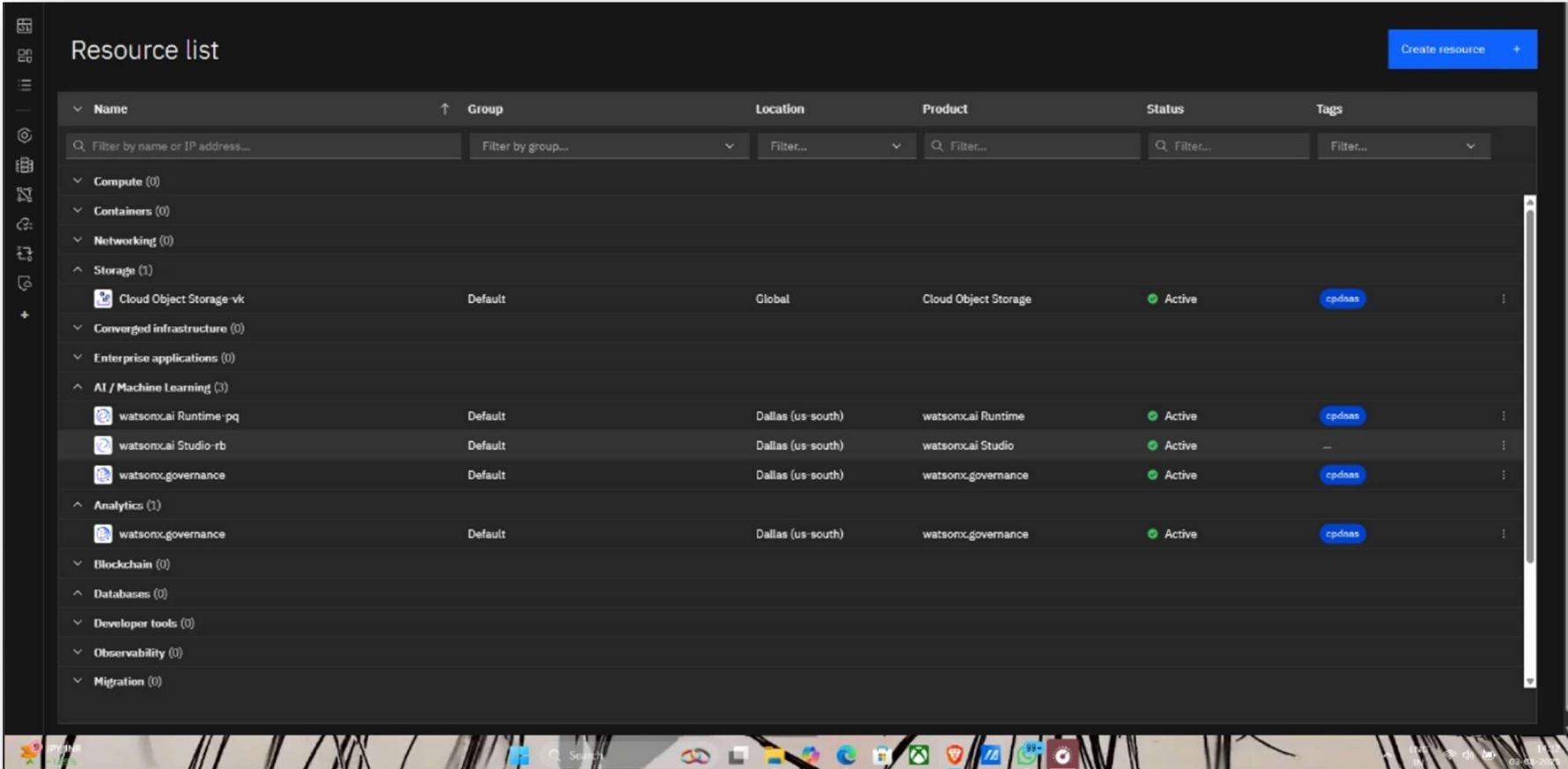
ALGORITHM & DEPLOYMENT

❖ In the Algorithm section, describe the machine learning algorithm Here's an example structure for this section:

- Algorithm & deployment
 - Implemented supervised machine learning techniques such as Decision Tree, Random Forest, or Neural Network.
 - Selected based on their strength in handling multiclass classification and real-time pattern recognition within network traffic.
- Data Input:
 - Features include protocol type, service, flag, duration, source bytes, destination bytes, and attack category.
 - Data obtained from well-known intrusion detection datasets like NSL-KDD for robust model training.
- Training Process:
 - Data split into training and testing subsets.
 - Applied feature scaling, balancing, and cross-validation to refine performance.
 - Hyperparameter tuning performed to boost accuracy and reduce overfitting.
- Prediction Process:
 - The trained model evaluates live network traffic or test inputs.
 - Flags potential intrusions based on learned attack patterns.
 - If you'd like to enrich this slide with metrics, deployment platforms, or your chosen algorithm's advantages, I'd love to help sharpen it

RESULT

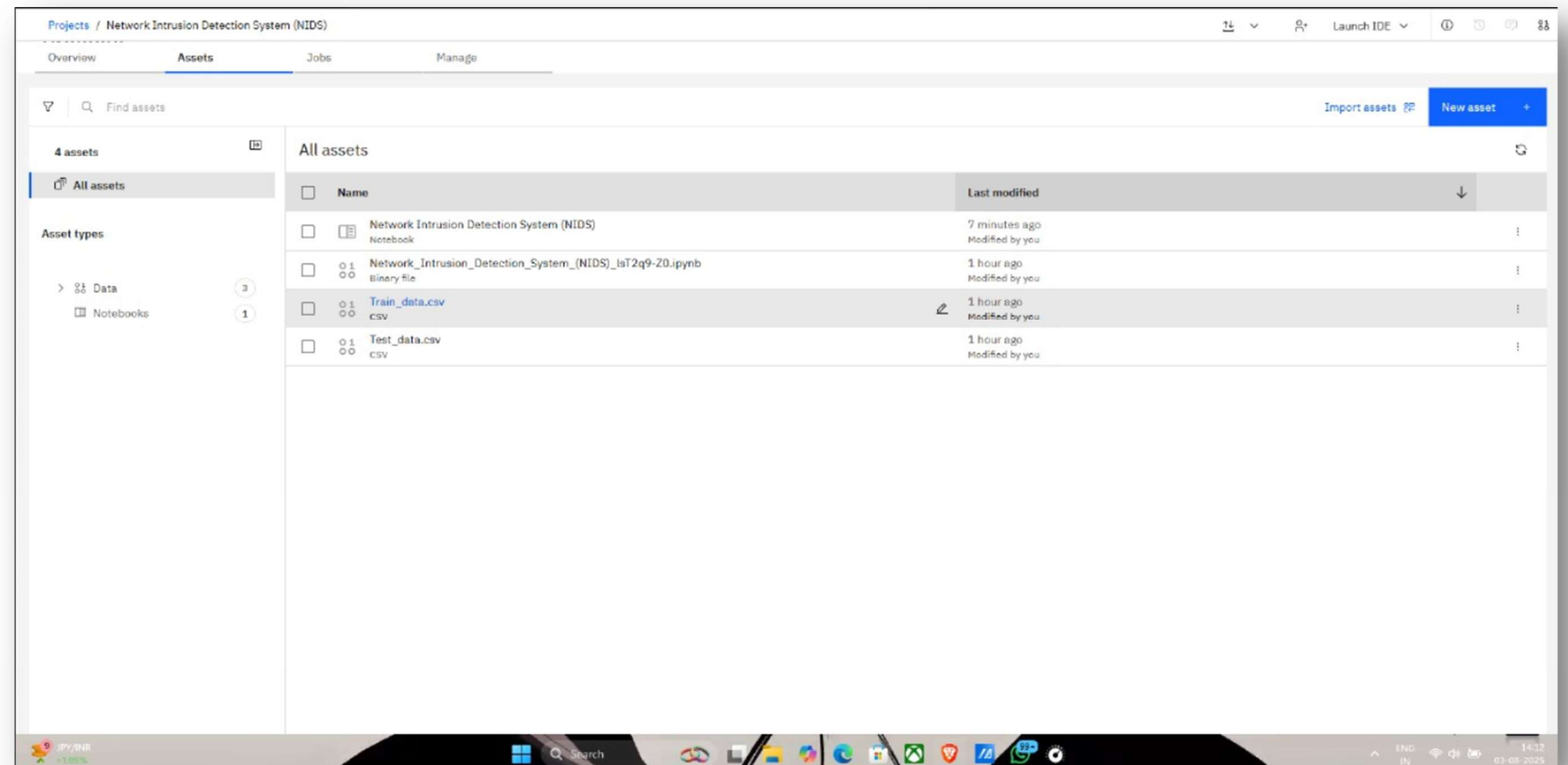
- These resource use in my Network Intrusion Detection.



The screenshot displays the IBM Cloud Resource List interface. On the left, a sidebar contains icons for navigation. The main area is titled "Resource list" and features a "Create resource" button in the top right corner. Below the title, there are several filter tabs: "Filter by name or IP address...", "Filter by group...", "Filter...", "Filter...", "Filter...", and "Filter...". The resources are organized into a table with columns: Name, Group, Location, Product, Status, and Tags. The table lists various resources, including Compute, Containers, Networking, Storage, Converged infrastructure, Enterprise applications, AI / Machine Learning, Analytics, Blockchain, Databases, Developer tools, Observability, and Migration. The "Storage" group is expanded, showing "Cloud Object Storage-vk" with a status of "Active". The "AI / Machine Learning" group is also expanded, showing "watsonx.ai Runtime-pq", "watsonx.ai Studio-rb", and "watsonx.governance", all with a status of "Active". The "Analytics" group is expanded, showing "watsonx.governance" with a status of "Active".

Name	Group	Location	Product	Status	Tags
▼ Compute (0)					
▼ Containers (0)					
▼ Networking (0)					
▲ Storage (1)					
Cloud Object Storage-vk	Default	Global	Cloud Object Storage	Active	cpdaas
▼ Converged infrastructure (0)					
▼ Enterprise applications (0)					
▲ AI / Machine Learning (3)					
watsonx.ai Runtime-pq	Default	Dallas (us-south)	watsonx.ai Runtime	Active	cpdaas
watsonx.ai Studio-rb	Default	Dallas (us-south)	watsonx.ai Studio	Active	---
watsonx.governance	Default	Dallas (us-south)	watsonx.governance	Active	cpdaas
▲ Analytics (1)					
watsonx.governance	Default	Dallas (us-south)	watsonx.governance	Active	cpdaas
▼ Blockchain (0)					
▲ Databases (0)					
▼ Developer tools (0)					
▼ Observability (0)					
▼ Migration (0)					

- these are the source files or assest files use In my project.



- In the notebook: Load the CSV file from the data asset.

```
[1]:
import os, types
import pandas as pd
from boto3.client import Config
import boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.

cos_client = boto3.client(service_name='s3',
                          aws_access_key_id='7hactYkwS1nP1z5X892ZfCtQ5k8KlpR4er0M9tIJDR0h',
                          aws_secret_access_key='https://iam.cloud.ibm.com/identity/token',
                          config=Config(signature_version='oauth'),
                          endpoint_url='https://s3.direct.us-south.cloud-object-storage.appdomain.cloud')

bucket = 'networkintrusiondetectionsystem1-donotdelete-pr-qzi0hdad476v1a'
object_key = 'Train_data.csv'

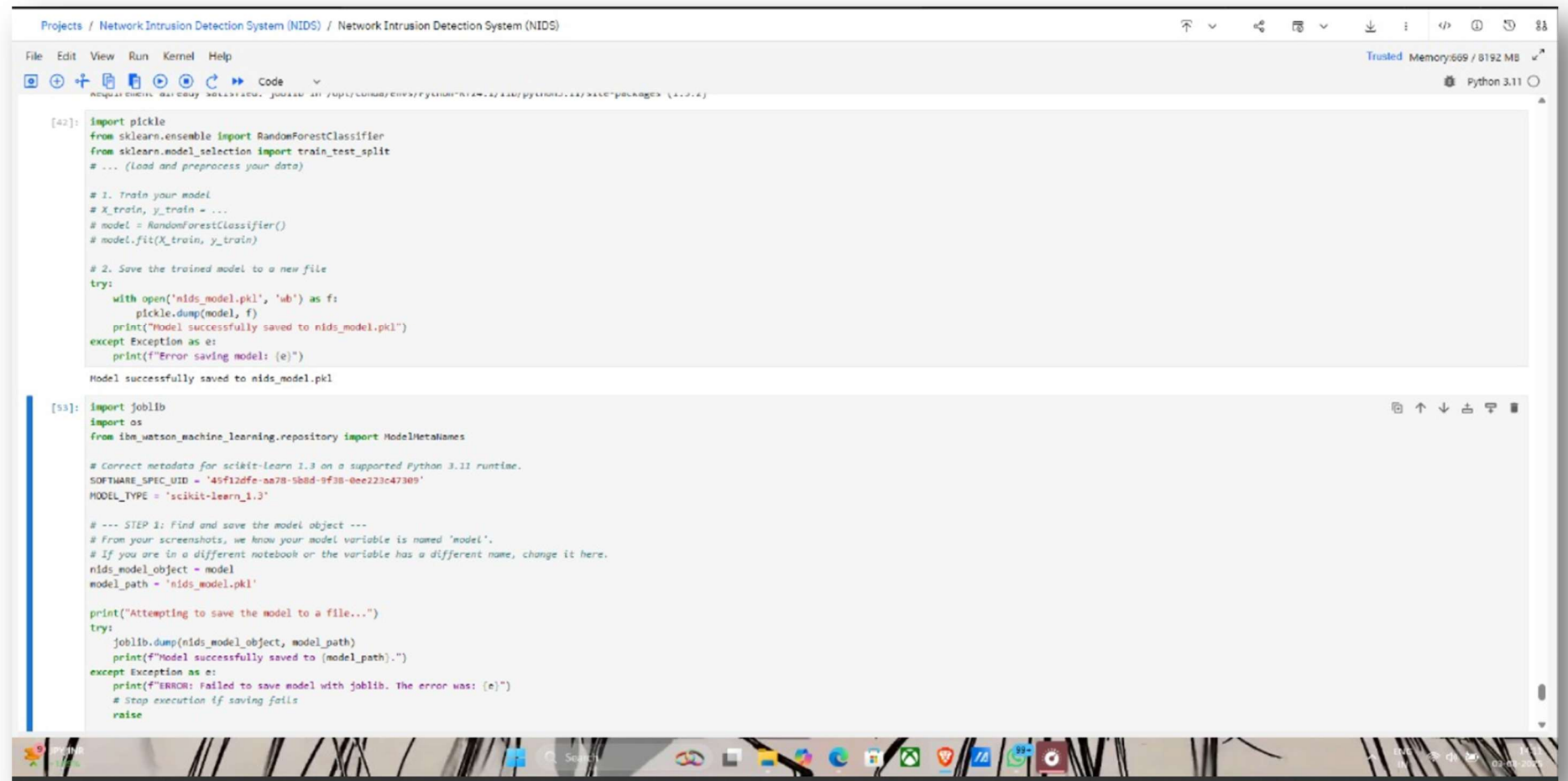
body = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, '__iter__'): body.__iter__ = types.MethodType(__iter__, body)

df_1 = pd.read_csv(body)
df_1.head(10)
```

```
[1]:
```

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_error_rate	dst_host_srv_error_rate
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	25	0.17	0.03	0.17	0.00	0.00	0.00
1	0	udp	other	SF	145	0	0	0	0	0	...	1	0.00	0.60	0.68	0.00	0.00	0.00
2	0	tcp	private	SO	0	0	0	0	0	0	...	26	0.10	0.05	0.00	0.00	1.00	1.00
3	0	tcp	http	SF	232	8153	0	0	0	0	...	255	1.00	0.00	0.03	0.04	0.03	0.01
4	0	tcp	http	SF	199	420	0	0	0	0	...	255	1.00	0.00	0.00	0.00	0.00	0.00
5	0	tcp	private	REJ	0	0	0	0	0	0	...	19	0.07	0.07	0.00	0.00	0.00	0.00
6	0	tcp	private	SO	0	0	0	0	0	0	...	9	0.04	0.05	0.00	0.00	1.00	1.00

- Encode categorical labels



```
[42]: import pickle
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
# ... (Load and preprocess your data)

# 1. Train your model
# X_train, y_train = ...
# model = RandomForestClassifier()
# model.fit(X_train, y_train)

# 2. Save the trained model to a new file
try:
    with open('nids_model.pkl', 'wb') as f:
        pickle.dump(model, f)
    print("Model successfully saved to nids_model.pkl")
except Exception as e:
    print(f"Error saving model: {e}")

Model successfully saved to nids_model.pkl

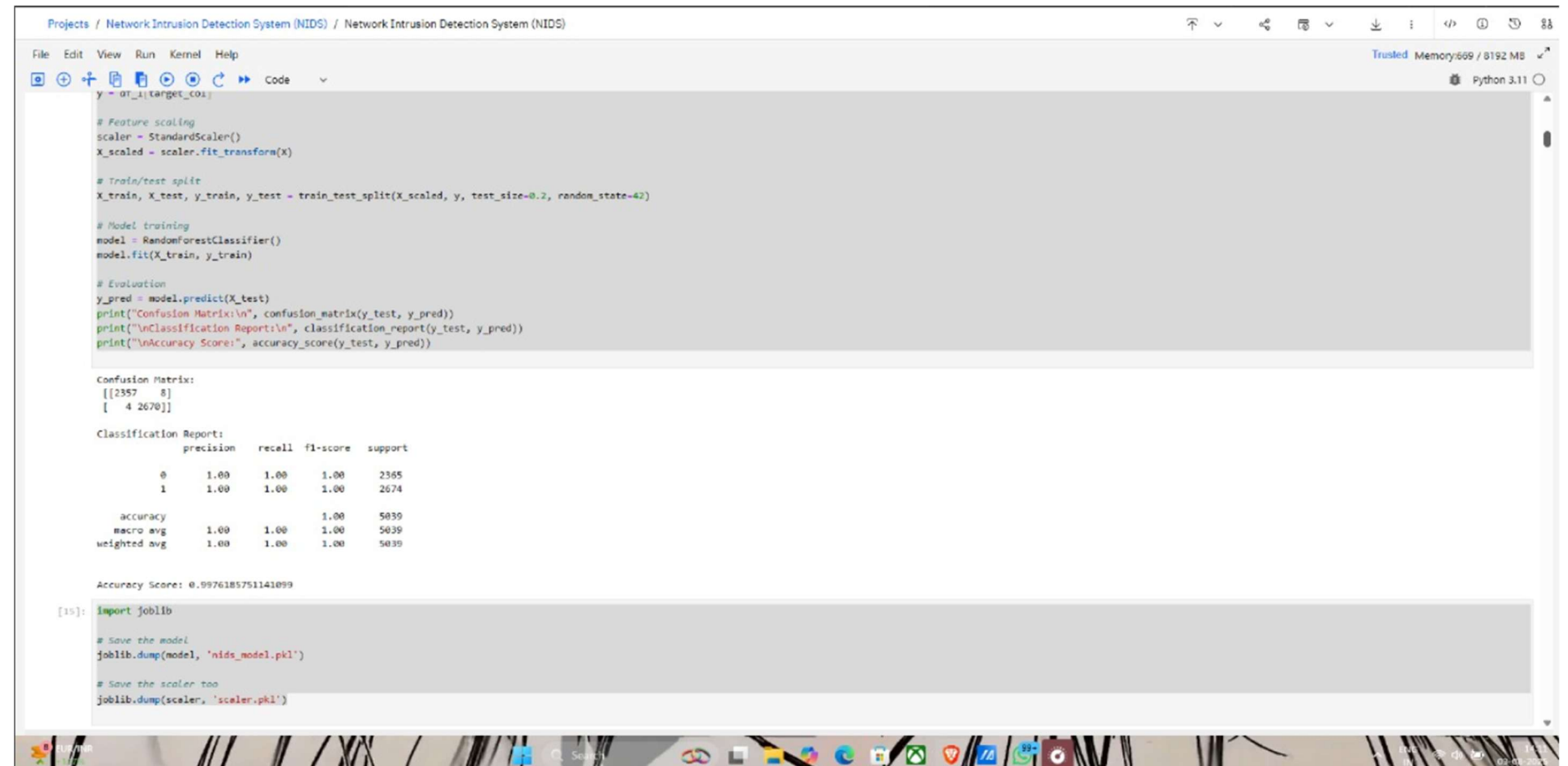
[53]: import joblib
import os
from ibm_watson_machine_learning.repository import ModelMetadata

# Correct metadata for scikit-learn 1.3 on a supported Python 3.11 runtime.
SOFTWARE_SPEC_UID = '45f12dfe-a878-5b8d-9f38-0ee223c47309'
MODEL_TYPE = 'scikit-learn-1.3'

# --- STEP 1: Find and save the model object ---
# From your screenshots, we know your model variable is named 'model'.
# If you are in a different notebook or the variable has a different name, change it here.
nids_model_object = model
model_path = 'nids_model.pkl'

print("Attempting to save the model to a file...")
try:
    joblib.dump(nids_model_object, model_path)
    print(f"Model successfully saved to {model_path}.")
except Exception as e:
    print(f"ERROR: Failed to save model with joblib. The error was: {e}")
    # Stop execution if saving fails
    raise
```

- Split data Train using classifiers like Random Forest, Logistic Regression



```
Projects / Network Intrusion Detection System (NIDS) / Network Intrusion Detection System (NIDS)
File Edit View Run Kernel Help Trusted Memory:609 / 8192 MB Python 3.11

y = df['target_class']

# Feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Model training
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Evaluation
y_pred = model.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))

Confusion Matrix:
[[2357  8]
 [ 4 2670]]

Classification Report:
              precision    recall  f1-score   support

     0         1.00      1.00      1.00     2365
     1         1.00      1.00      1.00     2674

 accuracy          1.00      1.00      1.00     5039
 macro avg          1.00      1.00      1.00     5039
weighted avg          1.00      1.00      1.00     5039

Accuracy Score: 0.9976185751141099

[15]: import joblib

# Save the model
joblib.dump(model, 'nids_model.pkl')

# Save the scaler too
joblib.dump(scaler, 'scaler.pkl')
```



CONCLUSION

- ❖ This project successfully demonstrates the design and implementation of a machine learning based Network Intrusion Detection System capable of identifying cyber threats with high accuracy.
- ❖ The proposed solution effectively detects abnormal network behavior in real time while minimizing false positives. Challenges faced included optimizing model performance and handling imbalanced datasets, which were resolved through careful preprocessing and algorithm tuning.
- ❖ Future enhancements could involve deep learning integration and hybrid detection methods. Accurate intrusion detection is critical for maintaining robust cybersecurity in today's interconnected digital infrastructure.

FUTURE SCOPE

- ❖ This system offers high scalability by incorporating varied datasets, optimizing detection algorithms, and extending deployment to smart urban networks. The integration of technologies like edge computing and next-gen ML models will strengthen its accuracy, responsiveness, and adaptability to evolving cyber threats.

REFERENCES

- ❖ Research articles on machine learning-based intrusion detection and cybersecurity frameworks
- ❖ Download the dataset from Kaggle:
<https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection>
- ❖ IBM Cloud Lite documentation – for deploying the model in a scalable environment
- ❖ Case studies on anomaly detection in network traffic using supervised and unsupervised models
- ❖ Academic papers focused on data preprocessing, feature selection, and performance evaluation techniques in NIDS
- ❖ Official documentation for tools/libraries used (e.g., Scikit-learn, NumPy, pandas)

IBM CERTIFICATIONS

(Getting Started with Artificial Intelligence)

In recognition of the commitment to achieve
professional excellence



ASHU YADAV

Has successfully satisfied the requirements for:

Getting Started with Artificial Intelligence



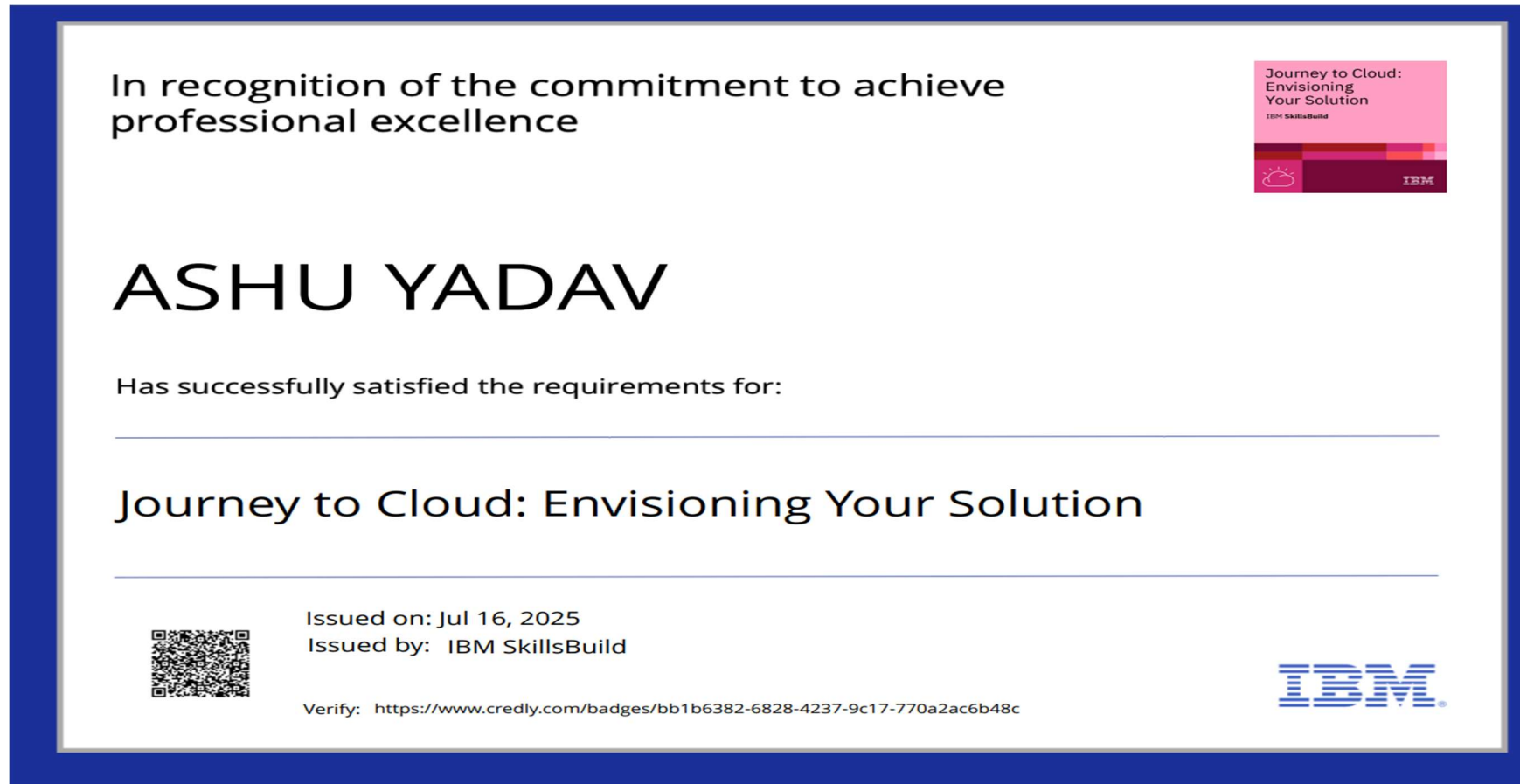
Issued on: Jul 16, 2025
Issued by: IBM SkillsBuild

Verify: <https://www.credly.com/badges/2485e39c-a5b0-40f4-acb0-f3a46c55c5f3>



IBM CERTIFICATIONS

(Journey to Cloud)



IBM CERTIFICATIONS

(RAG Lab)

IBM **SkillsBuild**

Completion Certificate



This certificate is presented to

ASHU YADAV

for the completion of

**Lab: Retrieval Augmented Generation with
LangChain**

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

Completion date: 25 Jul 2025 (GMT)

Learning hours: 20 mins



THANK YOU