
CAPSTONE PROJECT

NETWORK INTRUSION DETECTION SYSTEM

Presented By:

Guruaravindh K – Anna University Regional Campus, Madurai.(ECE)

OUTLINE

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

PROBLEM STATEMENT

Cyber-attacks such as DoS, Probe, R2L, and U2R are increasingly threatening network infrastructure. Existing systems struggle to distinguish malicious activity from normal traffic in real-time. Early detection of network intrusions is critical to maintaining security in communication networks, that system should be capable of analyzing network traffic data to identify and classify various types of cyber-attacks.

PROPOSED SOLUTION

- The proposed system aims to address the challenge of predicting the required bike count at each hour to ensure a stable supply of rental bikes. This involves leveraging data analytics and machine learning techniques to forecast demand patterns accurately. The solution will consist of the following components:
- Data Collection:
 - Gathers the information on the types of attacks and how they occurs , how frequent they occurs.
 - Utilize real-time data sources, such as no of failed logins, no of current active user, duration ,protocol type, service, flag.
- Data Preprocessing:
 - Clean and preprocess the collected data to handle missing values, outliers, and inconsistencies.
 - Feature engineering to extract relevant features from the data that might impact the detection of the anomaly.
- Machine Learning Algorithm:
 - Implement a machine learning algorithm, such as decision tree model, snap decision tree model and used 8 pipelines to get the best outcome.
 - Consider incorporating other factors like error rate, serror rate, srv error rate and destination host error cunt to improve prediction accuracy.
- Deployment:
 - Develop a user-friendly interface or application that provides real-time predictions anomaly detection.
 - Deploy the solution on a scalable and reliable platform, considering factors like server infrastructure, response time, and user accessibility.
- Evaluation:
 - Assess the model's performance using appropriate metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or other relevant metrics.
 - Fine-tune the model based on feedback and continuous monitoring of prediction accuracy.

SYSTEM APPROACH

- System Requirements:

- IBM Cloud Lite Account
- IBM Watson Studio & AutoAI
- Python (for model testing)

- Dataset Source:

- Kaggle: Network Intrusion Detection Dataset

- Creating the model:

- Open IBM cloud and initiate watsonx.ai service.
- Create a ml project model and associate a runtime service which will act as hardware for our model and add storage for data.
- Upload the dataset and select which parameter we need to predict ,then run the experiment.
- After completion select the best model and save code to promote it to space and create a deployment space and deploy model.
- Now we can test the model with actual input and get output, also we can integrate on external UI and use API to use the model from internet.

ALGORITHM & DEPLOYMENT

- **Algorithm Selection:**

- The model was built using AutoAI on IBM Watson Studio, which tested multiple machine learning algorithms such as Decision Trees, Random Forests, Gradient Boosting, and Logistic Regression.
- The best-performing pipeline was an ensemble-based classification model (likely Gradient Boosting or Extra Trees), automatically selected based on accuracy and F1-score.

- **Data Input:**

- The dataset has 41 diverse features, including categorical (e.g., protocol_type) and numerical (e.g., src_bytes) was taken from Kaggle and included labeled network traffic data..
- Ensemble models like Gradient Boosting and Random Forests handle mixed data types well. So the model can take both input data and analyse the dataset.

- **Training Process:**

- There are totally 41 input are given to the model consisting both numerical and categorical which are obtained from the previous real world activities of the users storing in normal and abnormal activities.
- IBM AutoAI automatically explored different machine learning pipelines, including preprocessing, feature engineering, and model selection. It tested models like Logistic Regression, Decision Trees, Random Forests, and Gradient Boosting.

- **Prediction Process:**

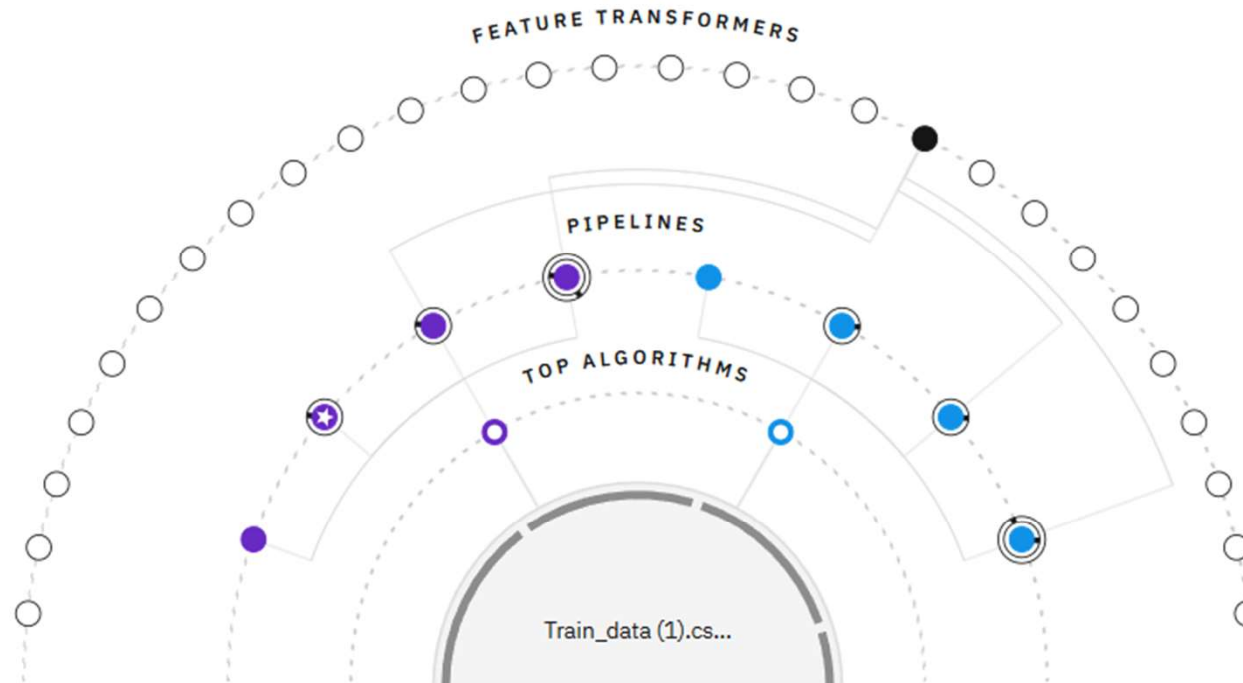
- Each pipeline was evaluated using cross-validation on the training data. Metrics such as accuracy, F1-score, and ROC AUC were used to rank pipelines.
- The pipeline with the best performance (highest F1-score) was selected as the final model. AutoAI trained the model by testing multiple pipelines, choosing the best one automatically based on performance.

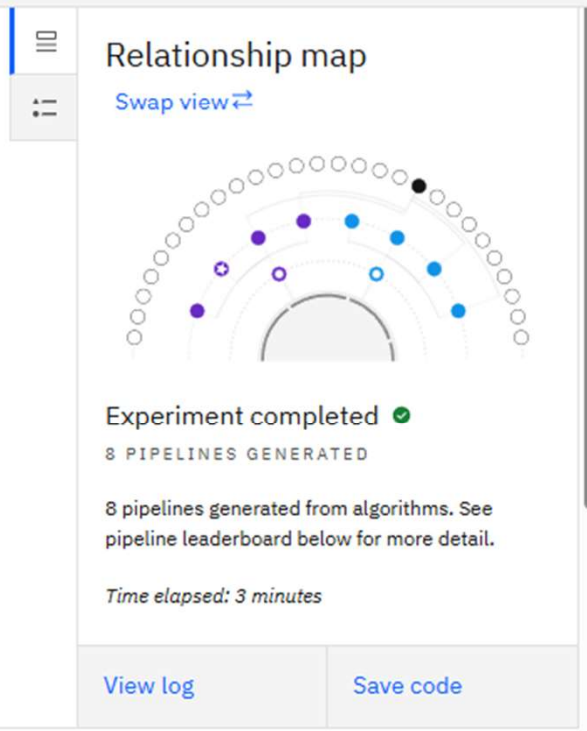
RESULT

The model was successfully trained using IBM Watson AutoAI, which evaluated multiple pipelines and selected the best-performing one based on metrics like F1-score and accuracy. After deployment, the model was tested using structured JSON input containing 41 features from the dataset. It correctly classified inputs into “normal” and “anomaly,” confirming that the model learned to distinguish malicious traffic patterns effectively. The scoring responses were accurate and reflected the predicted class with associated probability values. This validated the end-to-end functionality of the system, from training to deployment and prediction via API.











Relationship map ⓘ

Prediction column: class





Pipeline leaderboard

	Rank 	Name	Algorithm	Accuracy (Optimized) Cross Validation	Enhancements 	Build time	
★	1	Pipeline 2	 Snap Decision Tree Classifier	0.995	HPO-1	00:00:07	
	2	Pipeline 1	 Snap Decision Tree Classifier	0.995	None	00:00:03	
	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	
	5	Pipeline 4	 Snap Decision Tree Classifier	0.994	HPO-1 FE HPO-2	00:00:42	
	6	Pipeline 3	 Snap Decision Tree Classifier	0.994	HPO-1 FE	00:00:37	
	7	Pipeline 8	 Decision Tree Classifier	0.993	HPO-1 FE HPO-2	00:00:55	Save as
	8	Pipeline 7	 Decision Tree Classifier	0.993	HPO-1 FE	00:00:49	

NIDS_DEPLOYMENT

Deployed Online

API reference

Test

Enter input data

Text

JSON

Enter data manually or use a CSV file to populate the spreadsheet. Max file size is 50 MB.

Download CSV template

Browse local files

Search in space

Clear all

	duration (double)	protocol_type (other)	service (other)	flag (other)	src_bytes (double)	dst_bytes (double)	land (double)	wrong_fragment (double)	urgent (double)	hot (double)	num_failed_logins (double)	logged_in (double)	
1	0	tcp	private	S0	0	0	0	0	0	0	0	0	
2	0	udp	private	REJ	0	0	0	0	0	0	0	0	
3	0	tcp	private	S0	0	0	0	0	0	0	0	0	
4	0	tcp	private	S0	0	0	0	0	0	0	0	0	
5	0	udp	private	REJ	0	0	0	0	0	0	0	0	
6	0	tcp	private	S0	0	0	0	0	0	0	0	0	
7	0	udp	private	REJ	0	0	0	0	0	0	0	0	
8	0	tcp	private	S0	0	0	0	0	0	0	0	0	
9	0	udp	private	REJ	0	0	0	0	0	0	0	0	
10	0	udp	private	REJ	0	0	0	0	0	0	0	0	
11	0	tcp	http	SF	181	5450	0	0	0	0	0	1	

Prediction results

Prediction type

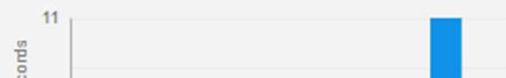
Binary classification

Prediction percentage



■ anomaly ■ normal

Confidence level distribution



Display format for prediction results

☒ Table view ☐ JSON view

☐ Show input data ⓘ

	Prediction	Confidence
1	anomaly	100%
2	normal	100%
3	anomaly	100%
4	anomaly	100%
5	normal	100%
6	anomaly	100%
7	normal	100%
8	anomaly	100%
9	normal	100%
10	normal	100%
11	normal	100%
12		
13		

CONCLUSION

This project demonstrated the development of a robust Machine Learning-based Network Intrusion Detection System (NIDS) capable of detecting various types of cyber-attacks such as DoS, Probe, R2L, and U2R. By using IBM Watson AutoAI, the process of data preprocessing, model training, and selection was automated, significantly reducing development time and complexity. The deployed model could be accessed using a REST API, making it suitable for integration into larger network security systems. Despite limitations like dataset imbalance and limited tuning control within AutoAI, the project successfully met its objective of classifying and detecting intrusions. It provides a scalable, cloud-based solution for early detection of malicious activity in communication networks.

FUTURE SCOPE

Looking ahead, the system can be improved by retraining it with up-to-date, real-world network traffic datasets to enhance its accuracy and adaptability. It can be extended to support multi-class classification to differentiate between various attack types rather than just normal vs anomaly. A web-based interface could be developed for user-friendly monitoring and input submission. Additionally, integrating the model with real-time monitoring and SIEM tools could enable automatic alerting and response mechanisms. For higher performance, future versions could explore deep learning approaches, such as LSTM or CNN, to capture temporal patterns in network data and improve detection of sophisticated threats.

REFERENCES

- IBM Watson Studio & AutoAI Documentation
- Kaggle: Network Intrusion Detection Dataset
- Machine Learning Techniques for Intrusion Detection: A Comparative Study
Tavallae, M., Bagheri, E., Lu, W., & Ghorbani, A. A
- A Review of Machine Learning Algorithms for Network Intrusion Detection Systems
(2020)
S. Dhanabal and S. P. Shantharajah

IBM CERTIFICATIONS



IBM CERTIFICATIONS



IBM CERTIFICATIONS

IBM SkillsBuild

Completion Certificate



This certificate is presented to

GURUARAVINDH K

for the completion of

**Lab: Retrieval Augmented Generation with
LangChain**

(ALM-COURSE_3824998)

According to the Adobe Learning Manager system of record

Completion date: 23 Jul 2025 (GMT)

Learning hours: 20 mins



THANK YOU