World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

Anomaly detection in network traffic using azure machine learning and log analytics

Sai Yathin Manugula *, Dheeraj Varma Kalidindi, Sindhu Sri Gogikari and Srinivas Rao Billakanti

Department of Computer Science and Engineering-Data Science, Guru Nanak Institutions Technical Campus, Hyderabad, India.

World Journal of Advanced Research and Reviews, 2025, 26(03), 864-883

Publication history: Received on 25 April 2025; revised on 05 June 2025; accepted on 07 June 2025

Article DOI: https://doi.org/10.30574/wjarr.2025.26.3.2197

Abstract

This study presents a scalable and efficient solution for advanced anomaly detection in network traffic using Azure Databricks and machine learning techniques. Modern networks generate massive volumes of traffic data, making manual detection of anomalies or cyber threats challenging. Traditional tools, such as RDBMS and Hadoop, are slow and not designed for real-time security monitoring. To address these challenges, the proposed system utilizes Azure Databricks, a unified cloud platform for big data processing and machine learning. Network traffic logs were cleaned and transformed using PySpark to extract features, such as IP addresses, session duration, data transfer, and packet counts. K-means clustering was then applied to group similar traffic patterns and identify anomalies without the need for labeled data. Model performance was evaluated using the Silhouette Score to ensure meaningful and well-separated clusters. The objective of this study is to provide a comprehensive overview of recent advancements in abnormality detection, focusing on emerging technologies and potential future opportunities. All stages, from data ingestion to anomaly detection, were executed within a single databricks notebook, thus requiring a minimal setup. The system performs efficiently even on low-cost Azure plans, making it accessible to small teams, students, and researchers. This solution enables real-time threat detection, automatic scaling, and quick incident response, offering a faster, smarter, and more cost-effective alternative to traditional network security methods.

Keywords: Network Traffic; Anomaly Detection; Azure Databricks; K-Means Clustering; Silhouette Score

1. Introduction

In today's digitally connected world, the volume and complexity of network traffic have grown exponentially, posing significant challenges to cybersecurity and operational efficiency. The detection of anomalies within network traffic, which are indicative of cyber threats, system malfunctions, or unusual user behaviors, has become a critical component of network management and security. Traditional rule-based monitoring systems often fail to identify novel or subtle threats, necessitating the use of intelligent and adaptive solutions.

This research explores the integration of Azure Machine Learning and Azure Log Analytics for real-time and scalable anomaly detection in network traffic. Azure Machine Learning offers robust tools for building, training, and deploying predictive models, whereas Log Analytics, a component of Azure Monitor, enables the collection and analysis of large volumes of telemetry data from various network sources.

By leveraging these technologies, this study aimed to develop a data-driven automated system capable of identifying abnormal patterns in network behavior with high accuracy and minimal human intervention. The proposed approach enhances proactive threat detection, supports rapid incident response, and provides valuable insights into network health, ultimately contributing to a more secure and resilient IT infrastructure.

* Corresponding author: Sai Yathin Manugula

1.1. Overview of Network Analysis Challenges

Network traffic analysis plays a vital role in monitoring, managing, and securing the modern digital infrastructure. However, the process of analyzing traffic is becoming increasingly complex because of a range of evolving challenges. These challenges stem from the growing scale, speed, and sophistication of the network environment. This section explores the key obstacles faced in contemporary network traffic analysis.

- Massive data volumes are overwhelming traditional analysis tools
- Increasing network speeds and complexity
- Encrypted traffic hinders visibility into packet contents
- Diverse and evolving network protocols
- Distinguishing between normal and malicious traffic patterns
- Real-time analysis requirements for timely threat detection

1.2. Importance of Anomaly Detection in Network Security

- Early identification of potential security threats and breaches
- Detection of zero-day attacks and previously unknown threats
- Reduction of false positives in intrusion detection systems
- Improved incident response times and mitigation of damages
- Compliance with regulatory requirements for data protection - Enhanced visibility into network behavior and performance issues

1.3. Azure Databricks–Based Anomaly Detection Framework

In today's hyperconnected environment, the volume and velocity of network traffic make manual monitoring infeasible. Every second, devices exchange millions of data packets, and detecting anomalies—whether malicious attacks or misconfigurations—poses a critical challenge. Traditional tools often demand deep security expertise and high-end infrastructure, creating barriers for smaller teams or organizations with limited resources. This project confronts obstacles by unifying data engineering and machine learning in a single native cloud platform.

Using Microsoft Azure Databricks, we ingested raw network traffic logs, completed with source/destination IPs, byte counts, session durations, and packet totals, directly into a scalable PySpark environment. Data cleaning, transformation, and feature vector assembly occur, eliminating the friction of moving large datasets across disparate systems. By encoding each session as a multidimensional vector, we prepared a groundwork for unsupervised learning at scale.

We applied k-means clustering to these feature vectors to discover natural groupings in the traffic. Sessions that fall into very small clusters are flagged as potential anomalies, such as zero-day exploits, stealthy scans, or atypical usage patterns that signature-based systems may miss. Model quality was assessed using the Silhouette Score, which quantifies how distinctly the clusters are separated. A high score indicates a clear delineation between "normal" behavior and outliers. It runs end-to-end in a single database notebook, ensuring reproducibility and ease of maintenance.

This approach has several advantages.

- **Scalability and Cost Efficiency:** Azure's elastic computation allows organizations to process massive logs on pay-as-you-go or even free-tier clusters.
- **Democratized Security Analytics:** A streamlined notebook interface and intuitive API lower the barriers for analysts without deep ML or DevOps knowledge.
- **Real-Time Responsiveness:** Near-instant model retraining and anomaly detection enable a faster incident response and damage mitigation.
- **Regulatory and Operational Compliance:** Automated logging, audit trails, and continuous monitoring support data-protection mandates while enhancing visibility into both security and performance issues across the network.

By integrating cloud-native data processing with robust unsupervised learning, this framework empowers a broader range of users to detect and respond to emerging threats without sacrificing their accuracy, speed, or ease of use.

1.4. Significance of Anomaly Detection in Network Traffic

Anomaly detection is the cornerstone of network security and performance monitoring. This enabled the identification of

- Intrusions or cyber-attacks (e.g., DDoS, port scanning)
- Misconfigurations and system errors
- Unusual user behavior or insider threats
- Performance bottlenecks or service outages

Traditional methods often rely on signature-based or rule-based systems that struggle with zero-day attacks and evolving threats. Machine learning (ML) offers a dynamic alternative capable of learning from data and detecting novel or unknown patterns.

1.5. Statement of the Problem

In today's hyper-connected digital ecosystem, the network infrastructure is constantly exchanging vast amounts of data in real time. As organizations scale, the volume, velocity, and variety of network traffic grow exponentially, making it nearly impossible to manually monitor or use traditional systems. Detecting anomalies, such as sudden spikes, suspicious connections, data exfiltration, or unauthorized access, requires intelligent, scalable, and real-time solutions.

Conventional approaches, such as RDBMS-based systems and Hadoop frameworks, have been designed for static, batch-oriented environments and lack the responsiveness and agility required for modern cybersecurity operations. These legacy systems involve fragmented pipelines, manual interventions, and high setup complexity, often falling short in environments where real-time anomaly detection is mission-critical.

1.5.1. The problem is further amplified by

- Absence of labeled data for supervised learning.
- High false positive rates in rule-based systems.
- Complex deployment pipelines that require cross-tool integration.
- Inability to scale efficiently under surging data loads.

1.6. Objectives

1.6.1. General Objective

- To develop a scalable and automated anomaly detection system for large-scale network traffic using Azure cloud services and machine learning techniques.

1.6.2. Specific Objective

- Handle large-scale network traffic data using Azure Databricks and PySpark.
- Useful features are built from raw data to help the models detect unusual activities.
- Use K-Means clustering to find outliers without needing labeled data.
- Measure how well the model performs using the Silhouette Score.
- Create a low-cost, scalable solution using Azure's free or affordable services.
- Automate behavior tagging to classify traffic types like 'Downloader,' 'Spammer,' and 'Normal.'
- Calculate the risk scores and feature-level deviations from the cluster centers.

1.7. Why Do Azure Machine Learning and Log Analytics Perform Anomaly Detection?

The selection of **Azure Machine Learning (Azure ML)** and **Azure Log Analytics** for anomaly detection is driven by their powerful, integrated capabilities tailored for handling large-scale, real-time network data analysis. Together, these provide a scalable, intelligent, and cloud-native solution ideal for modern enterprise environments.

- **Scalable and Cloud-Native Architecture**

Azure offers a robust infrastructure that seamlessly scales with data volume and network size. This ensures that the anomaly detection system remains efficient and responsive, even under high network load conditions.

- **Advanced Machine Learning Capabilities**

Azure ML supports a wide array of machine-learning algorithms, including supervised, unsupervised, and deep-learning models, which are suitable for detecting complex and previously unseen anomalies in network traffic. It also enables the automation of model training, tuning, deployment, and monitoring, and reduces manual intervention and error.

- **Real-Time and Historical Data Analysis**

Azure Log Analytics, part of the Azure Monitor, provides deep insights into real-time and historical network data by collecting logs from servers, firewalls, and other endpoints. This rich data source is essential for training accurate ML models and continuous monitoring.

- **Integration and Automation**

Azure's ecosystem allows for tight integration between Log Analytics and Azure ML. Anomalies detected via ML models can trigger alerts, dashboards, or automated responses using Azure Logic Apps, streamlining incident management and reducing response time.

- **Security and Compliance**

Azure complies with numerous international security standards (ISO, GDPR, and HIPAA), ensuring that data privacy and protection are maintained throughout the anomaly detection process, which is an essential requirement for network security operations.

- **Cost-Effectiveness and Pay-as-You-Go Model**

Organizations benefit from Azure's flexible pricing, which supports both experimentation and production deployment, without upfront infrastructure costs.

Using Azure Machine Learning in conjunction with Log Analytics enables the development of a ***robust, intelligent, and adaptive anomaly detection system***. This approach not only improves detection accuracy but also enhances visibility, automates responses, and supports the ongoing evolution of cybersecurity strategies in dynamic network environments.

2. Literature Review

The increasing volume and complexity of network traffic in today's digital landscape have made anomaly detection a critical area of cybersecurity. Anomalies, unusual patterns, or behaviors in network traffic often signal malicious activities such as intrusions, data exfiltration, or distributed denial-of-service (DDoS) attacks (Chandola, Banerjee, & Kumar, 2009). Traditional rule-based systems are often limited in their ability to detect novel or subtle threats, prompting a shift toward machine learning-based approaches that can learn from data and adapt over time (Sommer & Paxson, 2010). Anomaly detection in network traffic has evolved significantly over the past decade, driven by the increasing sophistication of cyberattacks and sheer volume of data generated by modern networks. Chandola, Banerjee, and Kumar (2009) provided a foundational survey of anomaly types—point, contextual, and collective—and highlighted the limitations of traditional signature-based systems that often fail to identify novel threats. Sommer and Paxson (2010) further argue that rule-based intrusion detection systems struggle with high false-positive rates, underscoring the need for adaptive, data-driven approaches such as machine learning (ML).

Machine learning (ML) has demonstrated strong potential for network anomaly detection owing to its ability to analyze large datasets and identify patterns that deviate from normal behavior. Supervised learning methods, such as Support Vector Machines (SVM) and Random Forests, have shown high accuracy when trained on labeled datasets (Bhuyan et al., 2014). Unsupervised techniques such as k-means clustering and Isolation Forests are particularly useful when labeled data are scarce, which is common in cybersecurity (Ahmed, Mahmood, & Hu, 2016). Cloud-based platforms such as Microsoft Azure offer scalable and flexible environments for deploying ML models for anomaly detection. Azure Machine Learning (Azure ML) provides a range of tools and services for building, training, and deploying models in a production environment (Microsoft, 2022). Combined with Azure Log Analytics, which collects and processes data from network sources in near real-time, it is possible to develop comprehensive systems for detecting anomalies across distributed network infrastructure (Microsoft, 2023).

Recent studies have explored the integration of cloud services with machine learning for security monitoring. Kumar et al. (2021) implemented an ML-based intrusion detection system using cloud resources to demonstrate the scalability and adaptability of such systems in dynamic environments. Similarly, Zuech, Khoshgoftaar, and Wald (2015) emphasized the importance of big data analytics and cloud computing in efficiently handling vast volumes of network traffic.

Despite the growing research interest, there is limited empirical work focusing specifically on the combination of Azure ML and Log Analytics for anomaly detection. This gap highlights the need for further investigation of how these tools can be used together to improve detection accuracy, reduce false positives, and automate threat response workflows. Moreover, leveraging Azure's native integration with other Microsoft security services (e.g., Sentinel and Defender) offers additional opportunities for holistic threat detection. Thwaini's (2022) comprehensive review examines ML applications over the previous five years across diverse domains—Internet of Things (IoT), Wireless Sensor Networks (WSN), Industrial Control Systems (ICS), and Software-Defined Networks (SDN). He categorized anomalies into point (isolated spikes), contextual (deviations relative to a situational norm), and collective (patterns only anomalous in aggregation), and surveyed both supervised techniques (e.g., Support Vector Machines, Decision Trees, Naïve Bayes, Neural Networks) and unsupervised approaches (e.g., k-means, Principal Component Analysis, Hidden Markov Models, Gaussian Mixture Models). Thwaini (2022) highlighted that while simple methods, such as k-means clustering, offer low computational overhead, they may miss evolving or complex threats, whereas ensemble methods, which combine multiple algorithms, can improve the detection accuracy at the cost of increased resource demands. They concluded that future systems must balance adaptability with efficiency in order to counteract dynamic attacker strategies (Thwaini, 2022).

Building on this groundwork, Schummer et al. (2024) presented a practical ML system designed to achieve high accuracy, scalability, and explainability in network anomaly detection. Their architecture integrates (1) change-point detection to flag shifts in the baseline network behavior, (2) k-means clustering enhanced by Dynamic Time Warping for temporal pattern grouping, and (3) supervised classifiers (Random Forest and SVM) trained on labeled telemetry (bandwidth, latency, packet loss, and congestion). A notable strength of their work is the use of SHAP (Shapley Additive exPlanations) values to render model predictions interpretable, facilitating trust and actionable insights. However, they also acknowledge the system's high computational cost, dependency on labeled datasets, and complex deployment requirements relative to learner algorithms, such as vanilla k-means (Schummer et al., 2024).

2.1. Research Gap

Despite these advances, there remains a research gap in leveraging cloud-native platforms, such as Microsoft Azure, for end-to-end anomaly detection workflows. Azure Machine Learning (Azure ML) and Log Analytics together offer automated model management, scalable computing resources, integrated data ingestion pipelines, and seamless alerting/response mechanisms. By situating ML pipelines directly within Azure's secure and compliant environment, it becomes possible to address the challenges identified by Thwaini (2022) and Schumer et al. (2024), providing both adaptability to evolving threats and operational scalability, and explaining the ability required for enterprise-grade deployments.

*"The literature reveals several **research gaps**, including limited integration with cloud platforms, lack of real-time analytics, and challenges with log data utilization."*

2.1.1. Lack of Integration with Cloud-Native Platforms

While Thwaini (2022) and Schummer et al. (2024) extensively evaluated machine learning models and techniques for anomaly detection, **neither study explored the practical integration of these models with cloud-native platforms** such as Microsoft Azure. There is a research opportunity to evaluate how Azure Machine Learning and Log Analytics can streamline data ingestion, model deployment, and automated alerting in real-world network environments.

2.1.2. Limited Focus on Real-Time Detection and Response

Most studies, including those by Thwaini et al. and Schummer et al., have primarily focused on model accuracy and detection capabilities. However, **real-time anomaly detection and automated threat response**, which are critical for mitigating fast-moving cyberattacks, remain underexplored. Azure's capabilities for near-real-time analytics and automation (e.g., through logic apps or Azure Sentiment) provide a promising area for research.

2.1.3. Over-Reliance on Labeled Data

Both studies emphasized supervised learning methods (e.g., SVM, Random Forest), which require large volumes of high-quality labeled data. Obtaining such data in real-world networks is challenging, particularly in the case of rare or novel attacks. There is a gap in the exploration of semi-supervised or self-supervised learning methods within Azure ML pipelines to reduce the dependency on labeled datasets.

2.1.4. Insufficient Work on Model Explainability in Operational Environments

Although Schummer et al. (2024) incorporated explainability using SHAP, most current research does not fully address how to operationalize explainable AI (XAI) within production environments, where security analysts must quickly understand and act on model outputs. Further research is required to integrate explainable ML features with Azure dashboards and Log Analytics queries in order to improve analysts' trust and interpretability.

2.1.5. Scalability and Cost Optimization Not Thoroughly Evaluated

While the reviewed literature recognizes the computational demands of advanced models, there is limited exploration of cost-performance trade-offs in deploying ML models in a scalable, cloud-based setting. Azure provides autoscaling, pricing tiers, and resource optimization tools, which remain underutilized in existing anomaly detection studies.

2.1.6. Lack of Domain-Specific Model Adaptation

Although Thwaini (2022) highlighted different domains, such as IoT and SDN, there is a lack of adaptive model frameworks that tailor anomaly detection strategies to specific network environments. Research is required to evaluate how Azure ML models can be dynamically adapted or retrained for various network types (e.g., enterprise, industrial, and IoT).

2.1.7. Absence of End-to-End Automated Pipelines

Most studies have focused on isolated components, such as model training, detection, or evaluation, but **lack end-to-**end system designs that include data ingestion, preprocessing, model inference, alerting, and dashboarding. Azure's ecosystem allows for complete automation and orchestration; however, current literature has not demonstrated such holistic implementations.

2.2. Summary of Research Gaps

Although significant progress has been made in machine learning-based network anomaly detection, there are key gaps in practical implementation, automation, and real-time deployment using cloud-native platforms such as Azure. Addressing these gaps can lead to more intelligent, scalable, and effective cybersecurity solutions capable of evolving with the modern threat landscapes.

3. Methodology

This study was designed to meet its objectives based on the experimental findings.

3.1. System Architecture / Conceptual Design

PySpark is a distributed computing framework that can efficiently process large datasets across a cluster of machines. At the core of its architecture is the Driver Program, which acts as the brain of the Spark application. The driver breaks the big job (e.g., analyzing data) into smaller tasks, manages them, and communicates with the Cluster Manager to request resources such as CPU and memory.

The Cluster Manager coordinates with multiple machines, called Worker Nodes, where the actual work occurs. Each worker node runs the executors, which execute the tasks and temporarily store data. Spark divides a job into stages, which are further split into tasks that operate on smaller chunks of data called partitions. This division allows Spark to process data in parallel, thus making it highly scalable and rapid.

Data and task flows between the Driver Program, Cluster Manager, and Worker Nodes enable an efficient execution. Spark caches intermediate data in memory to optimize performance and avoid repetitive computations. This combination of parallel processing and caching makes Spark powerful for large-scale data processing such as filtering, aggregating, and analyzing datasets.

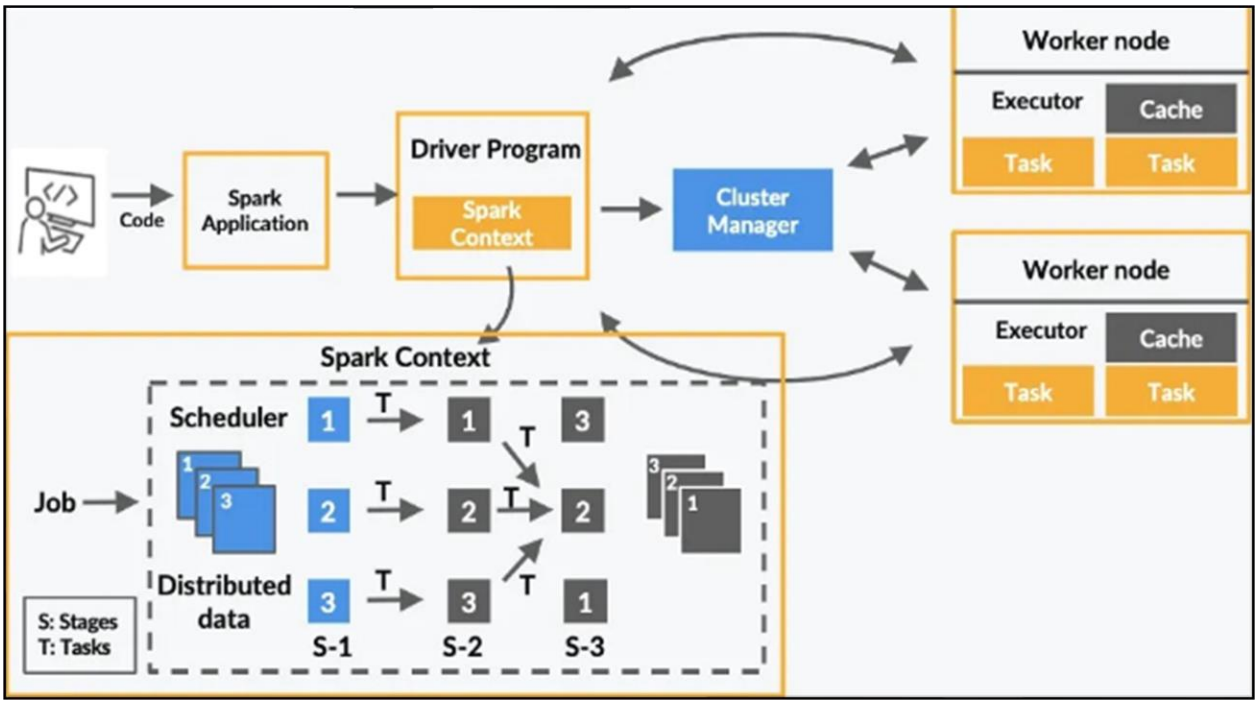

Figure 1 PySpark Architecture

3.2. Azure Databricks Architecture

Azure Databricks is a cloud-based platform that blends Apache Spark with Azure's scalability to offer a unified workspace for data engineering, analytics, and machine learning. It supports collaborative development through interactive notebooks, using Python, SQL, and R.

It is used to process large-scale data, build ETL pipelines, perform real-time analytics, and train machine learning models. Deep integration into Azure services such as Data Lake, Synapse, Power BI, and ML enables faster insights and smarter decision making across industries.

- An architectural overview of the Azure Databricks is as follows:

In the topmost layer, users interact through notebooks, APIs, or command-line interfaces. This represents the collaborative and flexible user interface of Azure Databricks, where developers, data scientists, and analysts can build, schedule, and monitor data pipelines and machine-learning models using familiar tools, such as Python, SQL, R, and Scala.

Beneath the interface, the architecture is split into two main planes: the Control Plane and the Data Plane. The Control Plane, managed entirely by databases, handles web applications, workspace configurations, job scheduling, notebook versioning, and other metadata management. This plane ensures that administrative tasks are abstracted and streamlined, thereby reducing operational overhead for the user.

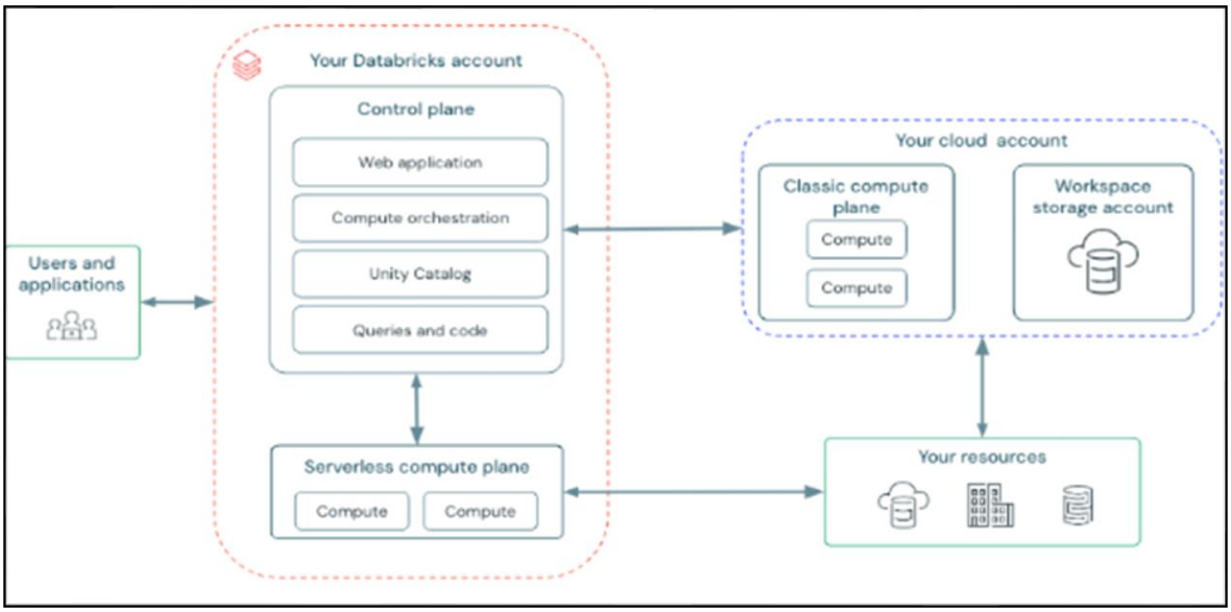

Figure 2 Azure DataBricks Architecture

The Data Plane resides within the azimuth subscription of the user. It is responsible for executing Spark jobs on clusters provisioned within their Azure Virtual Network (VNet). This plane provides users with full control over data access, security policies, and network configuration. Clusters here interact with storage services such as Azure Data Lake Storage (ADLS) or Azure Blob Storage to read/write data during processing.

Users can integrate a variety of Azure-native services into the storage and data service layers. These include Azure Event Hubs for real-time streaming data ingestion, Azure Synapse or Power BI for analytics and visualization, and Azure Machine Learning for training and deploying models. At the heart of the data layer lies Delta Lake, an optimized storage layer that provides ACID transaction support and schema enforcement on top of cloud object storage.

Finally, the architecture supports secure networking via options such as VNet injection, private links, and secure cluster connectivity, ensuring that sensitive data do not traverse the public internet and remain compliant with enterprise-grade security standards.

3.2.1. Explanation

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often, they are a preliminary step used to create an overview of the system, which will be elaborated upon later. DFDs can also be used to visualize data processing (structured design).

A DFD shows the types of data that will be input into and output from the system, where the data will come from and go to, and where the data will be stored. It does not provide information about the timing of processes or whether processes operate in sequence or parallel.

3.3. Design Approach

This study adopts a hybrid Agile and Object-Oriented design methodology to implement an intelligent anomaly detection system for network traffic in Azure. Agile enables rapid iteration and continuous feedback within Azure Databricks notebooks, making the development both adaptive and incremental. Object-oriented principles were applied in PySpark scripting to modularize components such as ingestion, transformation, clustering, and evaluation, ensuring reusability and scalability.

3.3.1. Phase 1 Data Ingestion and Validation

The process begins with data acquisition from CSV logs stored in the Azure Blob Storage. These logs include IP, Bytes_Sent, Duration, Packets, and Is_Anomaly. The data were loaded into Azure Databricks using PySpark's DataFrame API, which underwent schema validation to confirm the correct data types and completeness. If missing or malformed

values were detected, the file was rejected to maintain data quality. This phase provides a reliable foundation for downstream analyses and modeling.

3.3.2. Phase 2: Feature Engineering

In this phase, relevant features are extracted to construct the input vectors for machine learning. Specifically, Bytes_Sent, Duration, and Packets were selected as the core indicators of network activity. These features were combined into a single numerical vector per session using PySpark VectorAssembler. This transformation was critical for making the data compatible with the clustering algorithms and for capturing the behavioral patterns of each traffic instance.

3.3.3. Phase 3: Unsupervised Machine Learning with K-Means

The cleaned and transformed data were subjected to K-means clustering, an unsupervised algorithm that partitions data into K distinct groups. The system was configured with k=2 to distinguish between the normal and anomalous traffic patterns. The algorithm iteratively recalculates the cluster centroids to minimize the intracluster distance. After clustering, the smaller cluster was identified as anomalous based on the logical assumption that unusual traffic occurs less frequently than normal behavior.

3.3.4. Phase 4: Clustering Evaluation Using Silhouette Score

A Silhouette Score was computed to assess the quality of clustering. This score measures how similar an object is to its own cluster relative to other clusters, indicating how well the clustering model separates the different groups. A higher silhouette value reflects better-defined and well-separated clusters, thereby confirming the effectiveness of the model in isolating anomalies from normal traffic.

3.3.5. Phase 5: Visualization of Anomalies

To improve the interpretability of the results, a 2D scatter plot was generated using Matplotlib and Seaborn. The Bytes_Sent and Duration values were plotted with different colors representing different clusters. The anomalies are highlighted in red with black borders, making them visually distinguishable from the normal traffic. This helped to present complex data intuitively, allowing both technical and non-technical users to understand anomalies at a glance.

3.4. Design Constraints and Resolutions

3.4.1. Resource Scalability

Given the limited access to high-performance computing**,** the Azure Databricks distributed architecture is employed. Apache Spark's parallel processing enabled the analysis of large datasets using standard hardware, without needing a GPU or a large server.

3.4.2. Skill and Tool Simplicity

To reduce the learning curve and improve usability, Python and SQL-based APIs were used within the notebooks of the databases. This offers a user-friendly development environment and ensures maintainability for developers with varying skill levels.

3.4.3. Data Complexity

Raw logs often contain inconsistent format and missing data. These issues were mitigated through rigorous data validation and schema enforcement in PySpark, ensuring that only clean and well-typed data were entered in the pipeline.

3.4.4. Cost and Infrastructure

A major design constraint is cost efficiency, as the solution is aimed at students and researchers with limited resources. This was addressed by leveraging Azure's free-tier offerings**,** such as Databricks Community Edition and low-cost storage options.

3.5. Implementation/Development

3.5.1. Overview

The design was implemented using a combination of Azure Databricks, Python, and popular data science libraries to build an automated pipeline for detecting anomalies in the network traffic data. The pipeline consists of structured stages: data ingestion, validation, feature engineering, unsupervised clustering, and visualization of results.

3.5.2. Code

csv_data = """IP,Bytes_Sent,Duration,Packets,Is_Anomaly

- 192.168.1.2,300,0.5,40,1
- 192.168.1.3,450,0.7,55,0
- 192.168.1.4,1000,1.5,80,0
- 192.168.1.5,800,1.3,70,0
- 192.168.1.6,5000,10.0,300,1
- 192.168.1.7,200,0.3,35,0
- 192.168.1.8,7000,15.0,400,1
- 192.168.1.9,320,0.4,50,0
- 192.168.1.10,150,0.2,20,0
- 192.168.1.11,10000,20.0,600,1
- 192.168.1.12,600,0.9,45,0
- 192.168.1.13,7500,12.0,350,1
- 192.168.1.14,850,1.1,60,0
- 192.168.1.15,300,0.4,30,0
- 192.168.1.16,9000,18.0,500,1
- 192.168.1.17,250,0.3,25,0
- 192.168.1.18,400,0.6,50,0
- 192.168.1.19,5200,8.0,280,1
- 192.168.1.20,700,1.2,65,0

dbutils.fs.put("dbfs:/mnt/data/network_traffic.csv", csv_data, overwrite=True)

===Imports===#

from pyspark.sql import SparkSession

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score

import seaborn as sns

import matplotlib.pyplot as plt

=== Spark session===#

spark = SparkSession.builder.appName("NetworkAnomalyDetection").getOrCreate()

Path to CSV on DBFS

csv_path = "/dbfs/mnt/data/network_traffic.csv"

```
# Load using Pandas
df_raw = pd.read_csv(csv_path)
# === Validation ===#
expected_cols = {
   "IP": object,
   "Bytes_Sent": np.integer,
   "Duration": np.floating,
   "Packets": np.integer,
   "Is_Anomaly": np.integer
}
if df_raw.isnull().any().any():
   raise ValueError("Value Error: Missing values detected.")
for col_name and col_type in the expected_cols.items().
   if not np.issubdtype(df_raw[col_name].dtype, col_type):
      raise TypeError(f"Type Error: '{col_name}' expected {col_type._name_}, got {df_raw[col_name].dtype}")
print("Schema and nulls validated.")
# === Behavior Tagging ===#
conditions = [
   (df_raw["Bytes_Sent"] > 5000) & (df_raw["Duration"] > 5),
   (df_raw["Packets"] > 100) & (df_raw["Duration"] < 1)
]
choices = ["Downloader", "Spammer"]
df_raw["behavior"] = np.select(conditions, choices, default="Normal")
# === Feature Scaling ===#
X = df_raw[["Bytes_Sent", "Duration", "Packets"]]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# === Clustering ===#
kmeans = KMeans(n_clusters=2, random_state=42)
```

```
df_raw["prediction"] = kmeans.fit_predict(X_scaled)
silhouette = silhouette_score(X_scaled, df_raw["prediction"])
print(f"Silhouette Score: {silhouette:.4f}")
# === Anomaly Detection ===#
mincluster = df_raw["prediction"].value_counts().idxmin()
df_raw["is_anomaly"] = (df_raw["prediction"] == min_cluster).astype(int)
center = kmeans.cluster_centers_[min_cluster]
df_raw["risk_score"] = np.linalg.norm(X_scaled - center, axis=1)
df_raw["feature_contribution"] = np.abs(X_scaled - center).sum(axis=1)
# === Cluster Visualization ===#
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_raw, x="Bytes_Sent", y="Duration", hue="prediction", size="risk_score",
            palette="deep", sizes=(20, 200), alpha=0.7, legend='full')
anomalies = df_raw[df_raw["is_anomaly"] == 1]
plt.scatter(anomalies["Bytes_Sent"], anomalies["Duration"], c="red", label="Anomalies",
        edgecolors='black', s=100)
plt.title("Network Traffic Clustering and Anomalies")
plt.xlabel("Bytes Sent")
plt.ylabel("Duration")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
# === Risk Gradient Visualization ===#
plt.figure(figsize=(10, 6))
scatter = plt.scatter(df_raw["Bytes_Sent"], df_raw["Duration"], c=df_raw["risk_score"],
                cmap="viridis", s=100, edgecolor='k', alpha=0.75)
plt.colorbar(scatter, label="Risk Score")
plt.scatter(anomalies["Bytes_Sent"], anomalies["Duration"], c="red", label="Anomalies",
        edgecolors='black', s=120, marker='x')
plt.title("Network Traffic: Bytes Sent vs Duration with Risk Gradient")
```

plt.xlabel("Bytes Sent")

plt.ylabel("Duration")

plt.legend()

plt.show()

3.5.3. Code Structure

- Data Ingestion

CSV-formatted data were stored in Azure DBFS.

- dbutils.fs.put("dbfs:/mnt/data/network_traffic.csv", csv_data, overwrite=True)
- This allowed seamless integration with Azure Databricks for downstream processing.
- Data Loading & Initial Processing
- The dataset was read into a Pandas DataFrame and schema validation was performed to ensure that there were no missing values or inappropriate data types.
- df_raw = pd.read_csv(csv_path)

3.5.4. Validation Block

- The data type for each column was checked.
- Null values were flagged and rejected to avoid errors in the downstream model.
- Behavior Tagging (Domain Logic)
- Custom logic was applied to classify the traffic into behavioral categories.
- Downloader: Bytes_Sent > 5000 AND Duration > 5
- Spammer: Packets > 100 AND Duration < 1 Normal: Everything else

3.5.5. Feature Scaling

- The selected numeric features were normalized using a standard scalar to prepare the data for clustering.
- scaler = StandardScaler()
- X_scaled = scaler.fit_transform(X)
- Unsupervised learning–clustering
- K-means with two clusters was trained on the scaled features.
- The smallest cluster was interpreted as anomalous.
- min_cluster = df_raw["prediction"].value_counts().idxmin()
- df_raw["is_anomaly"] = (df_raw["prediction"] == min_cluster).astype(int)

3.5.6. Risk Scoring

- The Euclidean distance from the cluster centroid is computed as the risk score.
- The sum of the absolute feature-wise differences from the centroid was used for feature-contribution estimation.

3.5.7. Visualization Layer

- Cluster visualization: Displayed predictions using traffic parameters.
- Risk heatmap: Highlighted intensity of abnormality using color-coded scatterplots.

Testing and Validation

Test Case 1: Missing Values

Input CSV Data:

IP,Bytes_Sent,Duration,Packets,Is_Anomaly

192.168.1.2,300,0.5,40,1

192.168.1.3,,0.7,55,0

192.168.1.4,1000,1.5,,0

Expected Output:

Value Error: CSV file contains missing values. Please provide complete data.

Test Case 2: Incorrect Data Types

Input CSV Data:

IP,Bytes_Sent,Duration,Packets,Is_Anomaly

192.168.1.2,300,0.5,40,1

192.168.1.3, fourhundred,0.7,55,0

192.168.1.4,1000,1.5, eighty,0

Expected Output:

Type Error: Column 'Bytes_Sent' has an incorrect data type. Expected: integer; sound: string.

Test Case 3: Empty File

Input CSV Data:

IP,Bytes_Sent,Duration,Packets,Is_Anomaly

Expected Output:

Value Error: CSV file contains missing values. Please provide complete data.

Test Case 4: Wrong Schema Column Type

Input CSV Data:

IP,Bytes_Sent,Duration,Packets,Is_Anomaly

192.168.1.2,300,"0.5",40,1

192.168.1.3,450, "0.7",55,0

192.168.1.4,1000, "1.5",80,0

Expected Output:

Type Error: Column 'Duration' has an incorrect data type. Expected: double; sound: string.

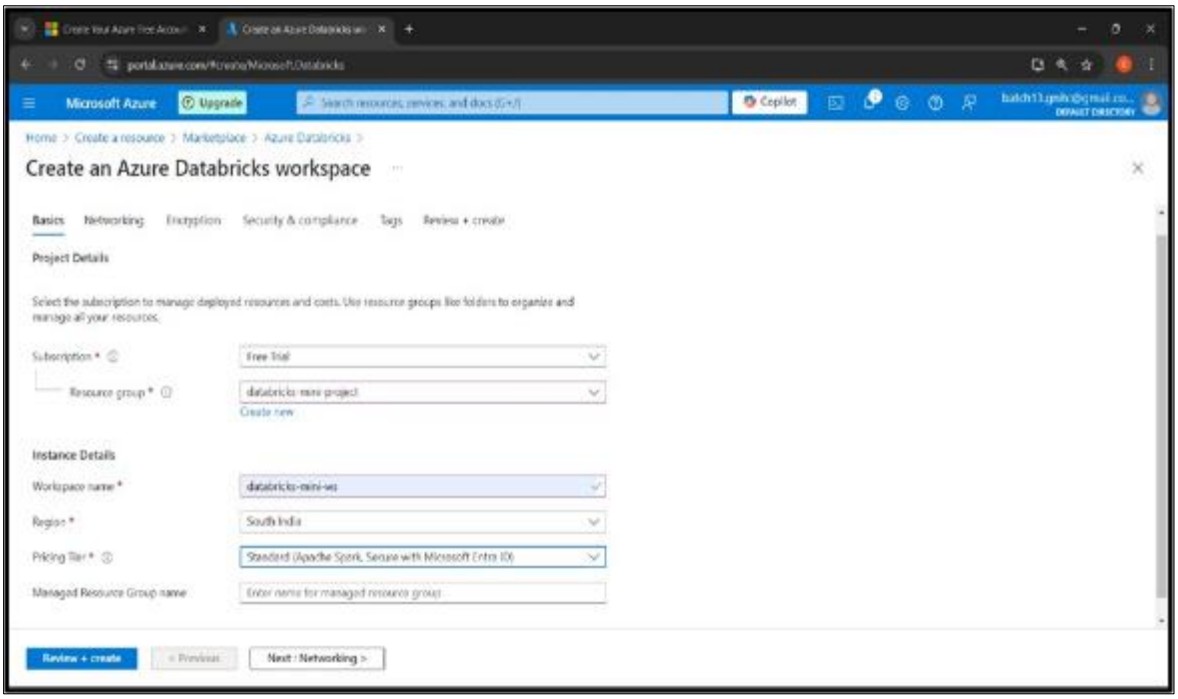

Figure 3 Creating an Azure Databricks Workspace

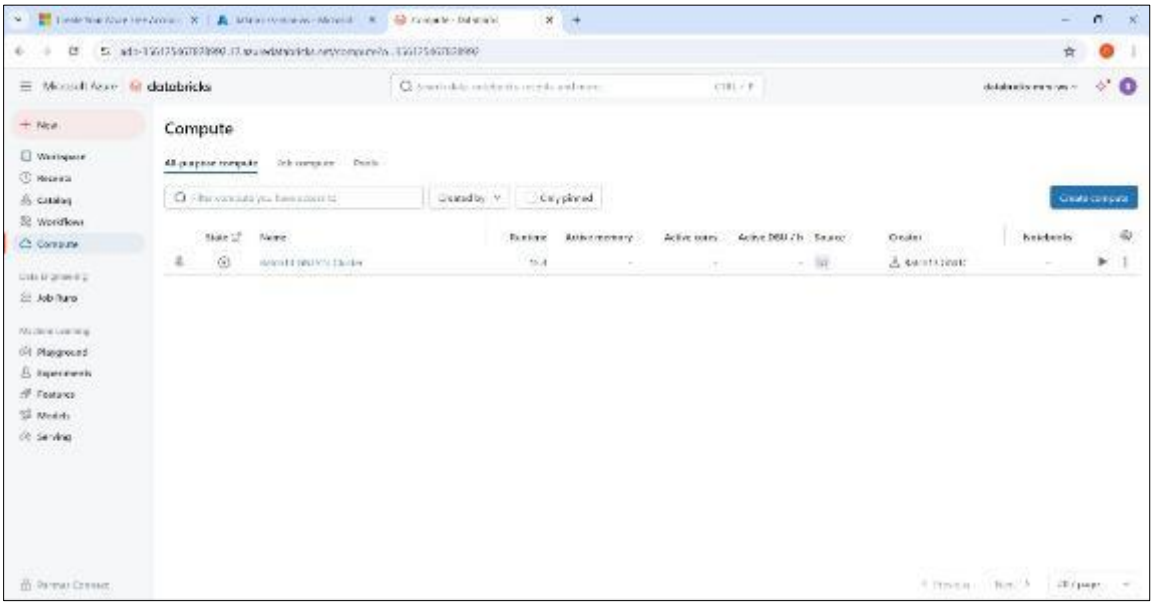

Figure 4 Computing Databricks

4. Results Analysis and Discussion

The scatter plot effectively visualizes the output of the K-Means clustering algorithm applied to network traffic data, using "Bytes Sent" and "Duration" as key features. Each data point represents an individual network session positioned according to the volume of data transmitted (x-axis) and the session duration (y-axis). The clustering process identified two primary groups, with the majority of sessions forming a compact blue cluster (Figure 5), indicating typical behavior. Single orange points. 1. stands out, forming a minority cluster that significantly deviates from the norm. This point, further marked by red X with a black outline, was flagged as an anomaly. The visualization is enhanced with a risk gradient, where the color intensity reflects the Euclidean distance from the cluster center, quantifying the severity. Sessions with higher risk scores, characterized by long durations and high byte counts, are visually distinct and indicative of potentially malicious activities such as data exfiltration or unauthorized persistent connections. This plot

not only supports the project's goal of leveraging unsupervised learning for anomaly detection but also serves as a valuable tool for network analysts to prioritize and investigate unusual traffic patterns effectively.

4.1. Network-traffic clustering and anomalies

4.1.1. Network Traffic: Bytes vs. Duration with Risk Gradient

In essence, the plot visually demonstrates how the K-Means clustering algorithm has separated the network traffic data into two groups based on the "Bytes Sent" and "Duration" features. The red circle highlights the data point that has been flagged as unusual or anomalous because it belongs to a smaller cluster.

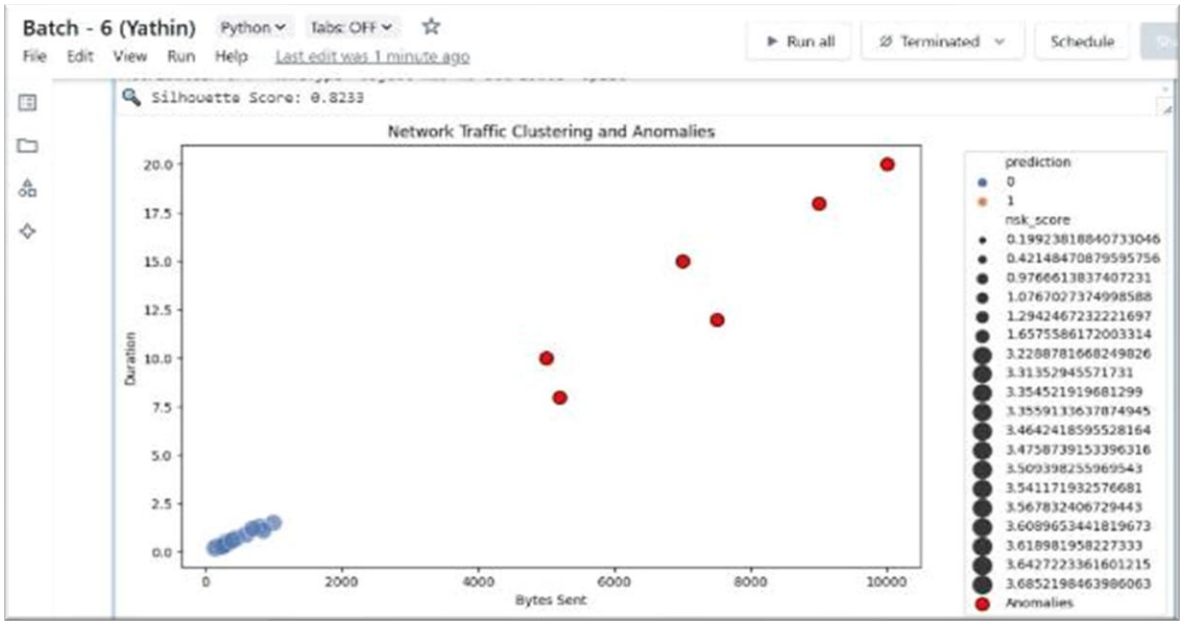

Figure 5 Network Traffic Clustering and Anomalies

- **X-axis:** Bytes Sent. This axis represents the amount of data transmitted during a network connection, measured in bytes. As you move to the right-hand side of the plot, the number of bytes sent increases.
- **Y-axis:** Duration. This axis represents the length of a network connection in some time units (likely seconds, based on the example data). As the plot moves upward, the duration of the connection increases.
- **Data Points:** Each point in the scatter plot represents a single network traffic session from the dataset. The position of the point is determined by its corresponding "Bytes Sent" and "Duration" values.
- **Color Coding (clusters):** The plot uses different colors to indicate the clusters identified by the K-means algorithm.
- **Blue points:** Figure 5 shows the network traffic sessions in which the k-means algorithm is grouped into one cluster. These sessions are likely to share similar characteristics in terms of the byte sent and duration.
- **Orange point:** Figure 5 shows the network traffic session in which the K-means algorithm is grouped into a second cluster.
- **Red Circles with Black Outlines:** The (labeled "Anomalies") in Figure 5 are the data points identified as anomalies by the analysis. Based on the description in the text, these are likely points belonging to the smaller of the two clusters found by K-means. This plot highlights the single orange point from the cluster "1" as an anomaly.

4.1.2. Network Traffic Bytes vs Duration with Risk Gradient

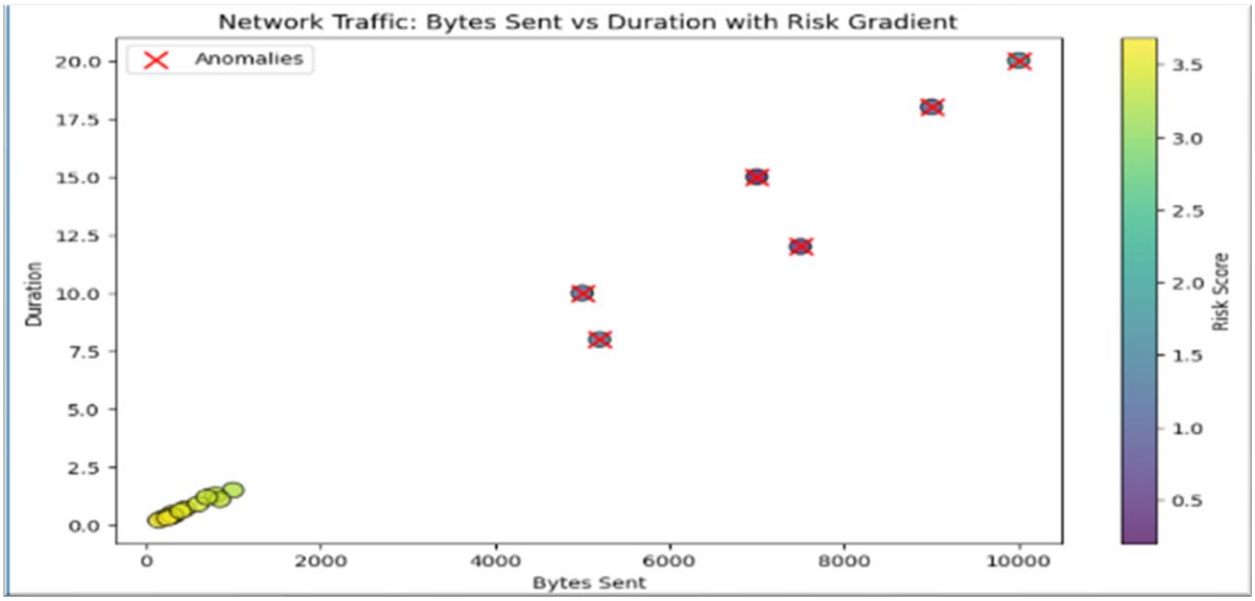

Figure 6 Network Traffic: Bytes vs Duration with Risk Gradient

X-axis: Bytes Sent: This axis represents the amount of data transmitted during a single network connection measured in bytes.

- As one moves to the right, the number of bytes sent increases.
- This metric is critical for identifying large data transfers that may indicate downloads, file-sharing, or data exfiltration.

Y-axis: Duration: This axis shows the duration of each network session, likely in seconds.

- Moving upward indicates longer sessions.
- Sessions that last significantly longer than average may be persistent connections, streaming events, or unauthorized long-duration sessions.

4.1.3. Data Points

- Each plotted point corresponds to a network traffic session in the dataset.
- The position of each point is defined by the Bytes Sent (X-axis) and Duration (Y-axis).
- These values were extracted from the uploaded CSV and preprocessed using feature scaling with StandardScaler.

4.1.4. Color Coding (Risk Gradient)

This plot uses a color gradient to show the risk score of each session.

- The color of each dot reflects its Euclidean distance from the K-Means cluster center (i.e., how far it deviates from "normal" behavior). Purple/blue: Low risk (close to the cluster center)
- Green/yellow: High risk (far from center, more anomalous)
- The color bar on the right maps these shades to numerical risk scores, ranging from 0.5 to 3.5+

- Red X Markers with Black Borders (Anomalies)

- These represent data points flagged as anomalies by the model.
- Anomalies are defined as sessions that belong to a minority cluster created by k-means (i.e., the smallest group).
- These sessions exhibit unusually high bytes sent and have long durations

- Fall far outside the compact cluster of typical behavior
- They are marked with red "X" symbols over their scatter points to visually separate them from the regular traffic.

4.1.5. What Makes These Points Anomalous?

- These outlier points had extreme values for byte stent and/or duration, triggering high-risk scores.
- The combination of long duration and high byte count strongly suggests the following.
- Potential malware exfiltration
- Unauthorized backups
- Rogue applications
- As these points form the smallest cluster, the algorithm correctly flags them for further inspection.

4.2. Visualization Purpose

4.2.1. This visualization delivers a 2D risk heatmap of the network activity

- It not only clusters data but also quantifies the degree of deviation via a risk score.
- It allows security analysts or network engineers to visually prioritize the sessions to investigate based on size, color, and anomaly marking.

4.3. How It Supports the Project Objective

- "Detecting Anomalies in Network Traffic Using Azure Machine Learning and Log Analytics" — by:
- Demonstrating the power of unsupervised learning (K-Means) to separate normal and suspicious behavior
- Providing a visual decision-making tool to identify network anomalies
- Enabling interpretability via risk scoring and feature scaling

5. Conclusion

We developed a simple, efficient, and scalable system for detecting anomalies in network traffic using the Azure Databricks and PySpark. The system demonstrates how to handle large datasets, prepare features, build machine learning models, and detect unusual network behaviors within a cloud platform without complex setups or expensive infrastructure. We successfully cleaned, validated, and prepared network traffic logs to ensure that the data pipelines were smooth.

Using K-Means clustering, we identified unusual traffic patterns without requiring labeled data, thereby rendering the system adaptable to various network conditions. The Silhouette Score confirmed that our clusters were well-separated, validating that the model could effectively distinguish between normal and abnormal traffic. Built using Azure's affordable resources, the system offers an accessible solution for students, researchers, and small businesses without the need for costly infrastructure.

The detected anomalies and traffic patterns were easily visualized, making the results intuitive for non-technical users. This project demonstrates that building a smart and reliable anomaly detection system in the cloud is achievable with minimal complexity and cost. The potential for future upgrades includes real-time monitoring, alert systems, and automated responses, paving the way for more efficient network security.

Recommendation

- **Adopt Real-Time Analytics Frameworks**

Justification: Real-time threat detection is critical in modern network security for preventing or minimizing damage during cyberattacks. Integrating streaming services, such as Azure Event Hubs, allows continuous monitoring and immediate anomaly detection, which is lacking in the current batch-processing system.

- **Incorporate Diverse and Rich Feature Sets**

Justification: Limiting the model to a small set of traffic features may result in missed or false-positive results. Enriching the dataset with additional attributes (e.g., port numbers, timestamps, and protocol types) enhances the sensitivity of the model and helps to capture a broader range of suspicious behaviors.

- **Apply Explainable Machine Learning Techniques**

Justification: In security-critical applications, decisions made by human analysts using AI models must be interpretable. Techniques such as SHAP and LIME help to explain model outputs, thereby increasing trust, aiding audits, and ensuring accountability in operational environments.

- **Enable Automated Threat Response Integration**

Justification: Manual incident response is time consuming and often inefficient. Automating alerts and responses using Azure Logic Apps or Sentinels ensures faster mitigation, minimizes human errors, and streamlines security operations.

- **Evaluate Cost-Performance Trade-offs**

Justification: Although cloud services offer scalability, their uncontrolled usage may lead to excessive cost. A detailed cost-benefit analysis helps choose the most efficient resource configurations, especially for resource-constrained teams or organizations.

- **Customize Models for Specific Network Environments**

Justification: Network behavior varies significantly across domains (e.g., corporate networks versus IoT). Adapting models to reflect domain-specific traffic characteristics improves the detection accuracy and reduces irrelevant alerts.

- **Promote Collaboration Between Security and Data Teams**

Justification: Effective implementation of anomaly detection systems requires combined effort from cybersecurity professionals and data scientists. Their collaboration ensures that the solution aligns with both the technical and security requirements.

Compliance with ethical standards

Acknowledgments

The authors would like to thank the support provided by the Supervisor, HOD, and Staff of the Department of CSE (Data Science), Guru Nanak Institutions Technical Campus. This research was self-funded and did not receive any specific grants from funding agencies in the public, commercial, or not-for-profit sectors.

Disclosure of conflict of interest

The authors declare that they have no conflict of interest.

References

[1] Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19–31. https://doi.org/10.1016/j.jnca.2015.11.016

[2] Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network anomaly detection: Methods, systems, and tools. IEEE Communications Surveys & Tutorials, 16(1), 303–336. https://doi.org/10.1109/SURV.2013.052213.00046

[3] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys (CSUR), 41(3), 1–58. https://doi.org/10.1145/1541880.1541882

[4] Kumar, R., Singh, M., & Arora, A. (2021). Cloud-based anomaly detection system using machine learning for secure networks. International Journal of Information Security Science, 10(2), 130–138.

[5] Micrsoft. (2022). Azure Machine Learning documentation. https://learn.microsoft.com/en-us/azure/machine-learning/

[6] Microsoft. (2023). Azure Monitor Logs and Log Analytics overview. https://learn.microsoft.com/en-us/azure/azure-monitor/logs/log-analytics-overview

[7] Schummer, P., del Rio, A., Serrano, J., Jimenez, D., Sánchez, G., & Llorente, Á. (2024). Machine Learning-Based Network Anomaly Detection: Design, Implementation, and Evaluation. AI, 5(4), 2967-2983. https://doi.org/10.3390/ai5040143

[8] Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In 2010 IEEE Symposium on Security and Privacy (pp. 305–316). IEEE. https://doi.org/10.1109/SP.2010.25

[9] Thwaini, M. H. (2022). Anomaly detection in network traffic using machine learning for early threat detection. Data and Metadata, 1, 34-34. https://doi.org/10.56294/dm202272

[10] Zuech, R., Khoshgoftaar, T. M., & Wald, R. (2015). Intrusion Detection and Big Heterogeneous Data: A Survey. Journal of Big Data, 2(1), 3. https://doi.org/10.1186/s40537-015-0013-4