An Intelligent Data Quality and Anomaly Detection System Using Azure Cloud Services

DISSERTATION

Submitted in partial fulfilment of the requirements of the

Degree: MTech. in Data Science and Engineering

By

Hardik Maheshwari

2023DA04317

Under the supervision of

Pallavi Sharma

IT Consultant, Tata Consultancy Services Ltd.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

Pilani (Rajasthan) INDIA

July 2025

Acknowledgement

I take this opportunity to sincerely thank all those who have contributed in various ways to the successful completion of my dissertation work titled *"An Intelligent Data Quality and Anomaly Detection System Using Azure Cloud Services."*

First and foremost, I wish to express my deepest gratitude to my industry supervisor, Ms. Pallavi Sharma, for their constant guidance, encouragement, and constructive feedback throughout the course of this project. Their expertise and practical insights from the industry not only helped me overcome several technical challenges but also enabled me to align my work with real-world applications in the field of Data Engineering and Machine Learning.

I would also like to extend my heartfelt thanks to the faculty members and mentors at BITS Pilani, who provided me with the academic foundation, technical knowledge, and critical thinking skills that formed the backbone of this dissertation. Their support during this MTech program has been instrumental in shaping my understanding of data science and engineering concepts.

Special thanks are due to my peers and colleagues who engaged in meaningful discussions, shared their experiences, and provided valuable feedback during the development of this work. Their collaboration and motivation have been an important source of encouragement.

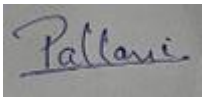
Finally, I remain indebted to my family for their unconditional support, patience, and motivation throughout my MTech journey. Their constant encouragement helped me stay focused and determined during times of challenge and ensured the timely completion of this dissertation.

This project has been a significant learning milestone in my academic and professional journey, and I owe its success to the collective efforts and encouragement of all those mentioned above.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Dissertation entitled **An Intelligent Data Quality and Anomaly Detection System Using Azure Cloud Services** and submitted by **Mr. Hardik Maheshwari** ID No **2023DA04317** in partial fulfilment of the requirements of DSECLZG628T / AIMLCZG628T Dissertation, embodies the work done by him/her under my supervision.



Pallavi Sharma, IT Consultant

Signature of the Supervisor

Name and Designation

Place: New Delhi

Date: August 14, 2025

Abstract

In today's data-driven world, ensuring the quality and reliability of incoming data is paramount for organizations that rely on large-scale analytics and machine learning models. However, as data volumes grow and are ingested from multiple heterogeneous sources, maintaining consistent and high-quality data becomes increasingly complex. Poor data quality—such as missing values, duplicates, and anomalies—can negatively impact business intelligence, leading to flawed decision-making.

This dissertation aims to design and implement a robust, automated, and scalable system for monitoring data quality and detecting anomalies using Azure Cloud Services. The system will utilize Azure Data Factory to orchestrate ingestion pipelines, perform data validation and profiling, and feed processed data into Azure Machine Learning models. The anomaly detection models—such as Isolation Forest and Autoencoders—will be trained using unsupervised learning techniques to detect outliers in transactional or time-series datasets. The results will be visualized on Power BI dashboards, while Azure Monitor and Logic Apps will be used to generate alerts for critical anomalies.

This project combines principles of Data Engineering, Unsupervised Machine Learning, and MLOps to build a reusable, modular architecture for enterprise-level data quality monitoring. It contributes toward enhancing trust in data pipelines, reducing the manual effort in data validation, and ensuring high data fidelity before the data is consumed for analytics or AI applications.

The dissertation spans a period of 16 weeks, justified through a structured work plan involving research, development, deployment, and testing phases.

List of Symbols & Abbreviations

MSE – Mean Squared Error

IF – Isolation Forest

AE – Autoencoder

ML – Machine Learning

ADF – Azure Data Factory

Azure ML – Azure Machine Learning Service

SLD – Synthetic Large Dataset

CCD – Credit Card Dataset

List of Tables

List Of Figures

Table of Contents

Chapter 1: Introduction

Anomaly detection, also referred to as outlier detection, is the process of identifying patterns in data that do not conform to expected behaviour. These anomalies may represent critical and actionable information such as fraudulent financial transactions, cybersecurity threats, malfunctioning equipment, or irregularities in healthcare monitoring. As organizations increasingly rely on data-driven systems, the importance of accurately detecting anomalies has grown significantly.

The rapid growth of digital platforms has led to the generation of massive volumes of high-dimensional data, often with complex structures and severe class imbalance. In fraud detection, for example, fraudulent cases may account for less than 1% of all transactions, making the task of detecting them particularly challenging. Traditional machine learning algorithms often fail in such contexts because they are biased towards the majority class. This raises the need for specialized anomaly detection techniques that are sensitive to rare but critical patterns.

Two broad families of methods are commonly used for anomaly detection: classical statistical and machine learning approaches, and modern deep learning–based methods. Among the classical methods, the Isolation Forest algorithm has gained attention for its efficiency and ability to handle large tabular datasets. It works by isolating data points using recursive partitioning, with anomalies requiring fewer splits compared to normal points. On the other hand, Autoencoders, a form of neural networks, are frequently used in deep learning anomaly detection. By learning compressed representations of data and then reconstructing them, autoencoders can highlight instances where reconstruction error is significantly high, thus flagging anomalies.

Despite the promise of these methods, real-world deployment remains difficult. Models must not only detect anomalies accurately but also be scalable, explainable, and integrated into existing enterprise ecosystems. For example, in financial fraud detection, missing a fraudulent transaction (false negative) can result in direct monetary losses, whereas raising too many false alarms (false positives) can erode customer trust. Therefore, striking a balance between

precision (how many detected anomalies are correct) and recall (how many true anomalies were detected) is critical.

In this dissertation, an anomaly detection pipeline has been developed and tested using both a large synthetic dataset and the publicly available credit card fraud dataset. The project evaluates the performance of Isolation Forest and Autoencoder models, comparing their effectiveness under balanced and imbalanced scenarios. In addition, the system architecture was designed for cloud readiness, with deployment capabilities in Microsoft Azure and dashboard integration in Power BI.

The key objectives of this work are:

1. To study the theoretical foundations of Isolation Forest and Autoencoder methods.
2. To preprocess and prepare synthetic and real-world datasets for anomaly detection tasks.
3. To implement and evaluate the models using precision, recall, and F1-score as primary metrics.
4. To integrate the system with cloud services and visualization tools for enterprise applicability.
5. To analyze the strengths, limitations, and future directions for anomaly detection systems.

By addressing these objectives, this dissertation contributes both an academic comparison of anomaly detection models and practical insights into their deployment challenges. The findings reinforce the importance of handling data imbalance, evaluating models with appropriate metrics, and designing systems that are not only accurate but also interpretable and scalable.

Chapter 2: Literature Review

Anomaly detection has been a long-standing area of research in statistics, machine learning, and data mining, driven by its importance in domains such as fraud detection, cybersecurity, industrial process monitoring, and healthcare. While the concept of identifying unusual patterns may sound straightforward, its implementation in large-scale, real-world systems presents considerable challenges. These include the curse of dimensionality, evolving data distributions, and severe class imbalance where anomalies are rare compared to normal records. Over the years, research has progressed from simple statistical techniques to sophisticated deep learning models and, more recently, to cloud-based anomaly detection pipelines.

This chapter reviews the state of the art in anomaly detection with a focus on four key dimensions: (1) traditional statistical approaches, (2) classical machine learning methods, (3) deep learning–based approaches, (4) strategies for handling imbalanced datasets, and (5) cloud-native anomaly detection systems.

2.1 Traditional Statistical Methods

Early anomaly detection methods relied heavily on statistical formulations. Approaches such as **z-scores**, **standard deviation thresholds**, and **Grubbs' test** were among the first systematic techniques to flag unusual data points. These methods assume that data follows a known probability distribution (e.g., Gaussian), and points deviating significantly from the mean are considered anomalies.

For instance, Tukey's boxplot rule identifies anomalies by comparing observations against the interquartile range (IQR). While computationally efficient and interpretable, these methods struggle with high-dimensional data or non-linear relationships between variables. In addition, their reliance on strong distributional assumptions makes them less suitable for complex datasets such as network traffic or credit card transactions, where normal behaviour may itself be highly varied.

2.2 Classical Machine Learning Methods

As data volumes and dimensionality grew, researchers turned to machine learning for more robust anomaly detection.

- Isolation Forest (Liu et al., 2008) introduced a tree-ensemble–based method where anomalies are isolated more quickly than normal points. The algorithm's strength lies in its ability to handle high-dimensional data with relatively low computational cost. However, it may underperform when anomalies are not well separated from normal samples.
- One-Class Support Vector Machines (Schölkopf et al., 2001) attempt to construct a boundary around normal data in feature space, treating anything outside as an anomaly. Though powerful in smaller datasets, their scalability is limited and they require careful parameter tuning.
- Local Outlier Factor (Breunig et al., 2000) detects anomalies by comparing the local density of a data point against that of its neighbours. It performs well in datasets where anomalies are context-dependent but is computationally expensive for very large datasets.

These classical approaches continue to be widely used due to their interpretability and moderate computational demands, but their performance tends to degrade under high imbalance or complex, non-linear data structures.

2.3 Deep Learning Approaches

The rise of deep learning has opened new possibilities for anomaly detection, particularly in handling large-scale, high-dimensional, and unstructured data.

- Autoencoders (Hinton & Salakhutdinov, 2006) compress input data into a low-dimensional latent representation and then reconstruct it. High reconstruction error signals an anomaly. Autoencoders have been successfully applied in domains like fraud detection, intrusion detection, and medical diagnosis. Their key advantage is the ability to capture complex non-linear dependencies, but they are sensitive to hyperparameter tuning and require sufficient training data.

- Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks (Malhotra et al., 2015) are particularly effective for time-series anomaly detection, such as detecting unusual patterns in ECG signals or server logs. They learn temporal dependencies, making them superior in sequential data contexts.
- Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have recently been explored for anomaly detection. GANs generate synthetic "normal" data, and the discriminator learns to distinguish between generated and real data. When tested on unseen data, anomalies are more likely to be misclassified by the discriminator, revealing their abnormality. Additionally, GANs have been applied to generate synthetic anomalies to address imbalance problems.

While deep learning methods outperform classical ones in many benchmarks, they are often computationally expensive and less interpretable, making deployment and trust in critical applications (such as healthcare or finance) more challenging.

2.4 Data Imbalance in Anomaly Detection

A fundamental challenge in anomaly detection is class imbalance. In many real-world datasets, anomalies constitute less than 1% of observations, as seen in credit card fraud detection or network intrusion logs. Models trained on such data often achieve high accuracy by simply predicting everything as normal, yet fail completely in identifying anomalies.

To mitigate this, several imbalance-handling techniques have been proposed:

- Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002) generates new synthetic samples for the minority (anomaly) class by interpolating between existing samples.
- ADASYN (He et al., 2008) extends SMOTE by generating more synthetic samples for hard-to-learn cases.
- Cost-sensitive learning assigns higher misclassification penalties to anomalies, making models more sensitive to rare events.

- **Hybrid ensemble approaches** combine oversampling with bagging/boosting to further improve recall.

In recent years, researchers have also applied **deep generative models** such as GANs and Variational Autoencoders (VAEs) to produce realistic synthetic anomalies, improving robustness against imbalance.

2.5 Cloud-Native Approaches

With the shift to cloud computing, anomaly detection has evolved into large-scale, distributed workflows.

- **Microsoft Azure Machine Learning** offers managed services to build, train, and deploy anomaly detection models. Integration with **Azure Data Factory** enables automated data ingestion pipelines, while **Azure Monitor** and **Logic Apps** provide real-time alerting.
- **Amazon Web Services (AWS)** provides anomaly detection through services like **SageMaker** and **CloudWatch Anomaly Detection**.
- **Google Cloud Platform (GCP)** integrates anomaly detection within its AI Platform and BigQuery ML.

The key advantage of cloud-native solutions is **scalability**. They allow organizations to process millions of records in near real-time, integrate anomaly detection directly into operational pipelines, and visualize insights using tools like Power BI or Tableau. However, cloud deployments introduce cost considerations and require careful monitoring to avoid over-provisioning resources.

2.6 Survey Studies

Several survey papers consolidate the field's knowledge. Chandola et al. (2009) provided one of the earliest comprehensive surveys, classifying anomaly detection methods into supervised, semi-supervised, and unsupervised categories. Aggarwal (2017) expanded this with discussions on scalability and

interpretability challenges. More recent works, such as Pang et al. (2021), focus on deep learning–based anomaly detection and provide a taxonomy of approaches including reconstruction-based, predictive, and generative models. Campos et al. (2021) reviewed benchmarking practices for anomaly detection, emphasizing the importance of evaluation metrics beyond accuracy when dealing with highly imbalanced datasets.

2.7 Comparative Insights

The reviewed literature highlights several recurring themes:

1. **Imbalance handling is critical** — methods like SMOTE, ADASYN, and GAN-based augmentation significantly improve detection rates.
2. **Classical methods remain relevant** — algorithms like Isolation Forest and One-Class SVM are still competitive for structured tabular datasets.
3. **Deep learning dominates high-dimensional data** — autoencoders, LSTMs, and GANs outperform classical models but are harder to deploy and interpret.
4. **Cloud integration is the future** — platforms like Azure and AWS enable real-time, enterprise-ready anomaly detection systems.

2.8 Summary

In summary, the literature establishes that no single method is universally superior. Instead, the choice of technique depends on the dataset, imbalance level, and operational requirements. Ensemble methods (e.g., Isolation Forest) and deep learning approaches (e.g., Autoencoders, LSTMs) represent two strong families of solutions. At the same time, addressing imbalance remains a crucial requirement for practical success. Cloud-native deployments further ensure scalability, automation, and integration with enterprise systems.

This dissertation builds on these insights by:

- Combining Isolation Forest and Autoencoders to evaluate performance across synthetic and real-world datasets.
- Explicitly addressing the imbalance challenge through experimental design.
- Implementing an Azure-based architecture with Power BI dashboards, making the solution directly applicable in enterprise contexts.

Chapter 3: Methodology and Implementation

This chapter describes the methodology adopted for designing and implementing the anomaly detection pipeline. The overall approach is modular, consisting of sequential stages: data acquisition, preprocessing, model training, evaluation, and visualization. Each stage has been carefully chosen to address challenges of high-dimensional data, severe class imbalance, and enterprise deployment in a cloud environment.

3.1 System Architecture

The system architecture for this project is illustrated in the below figure.

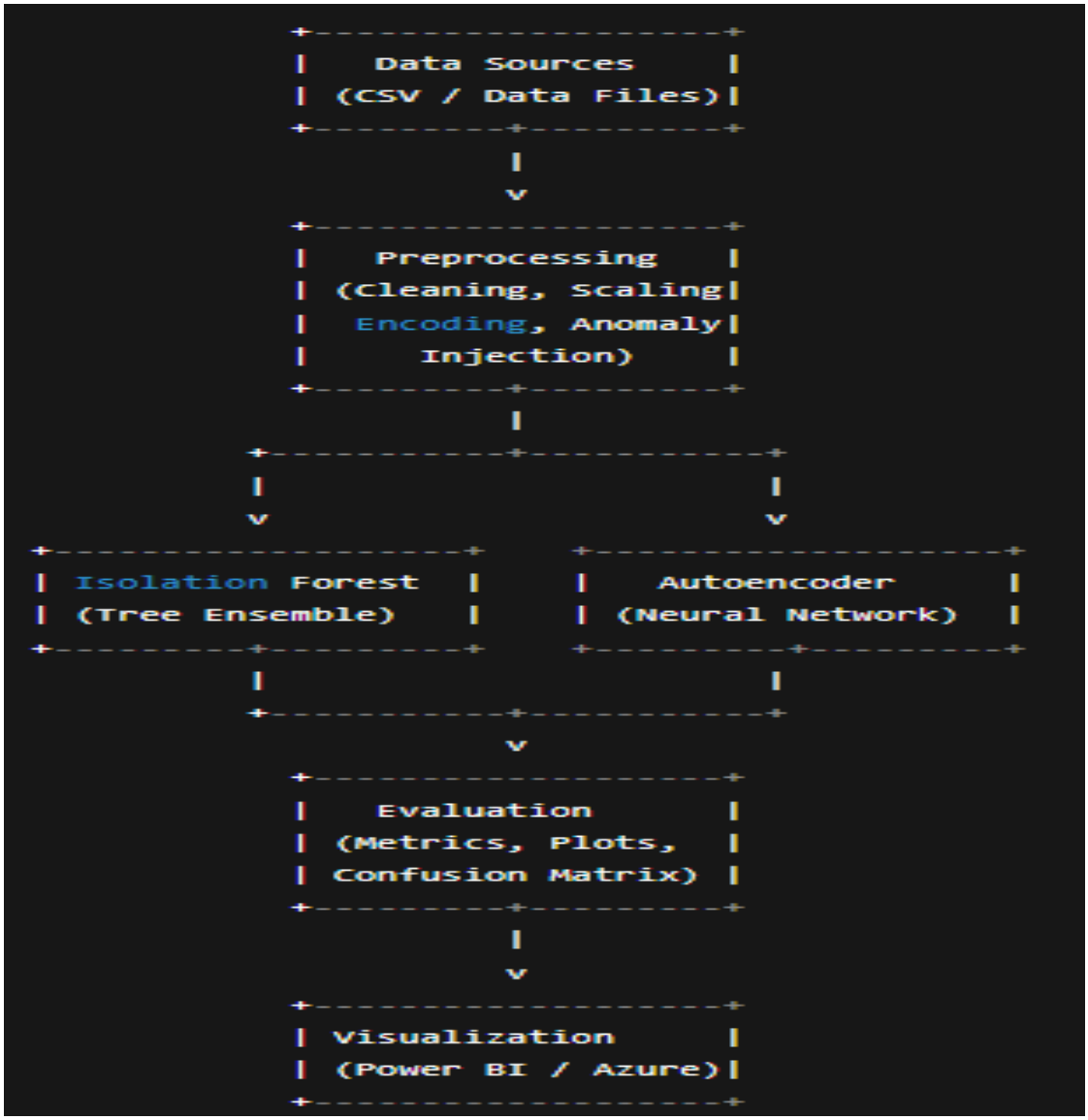


Figure 1: Overall System Architecture for Anomaly Detection

The architecture follows a **data pipeline design**, allowing integration with cloud services such as Azure Data Factory and Azure ML.

3.2 Data Preprocessing

Data preprocessing is essential for anomaly detection, as raw datasets may contain missing values, inconsistent formats, or imbalanced distributions.

Steps followed:

1. **Data Ingestion**
    - Datasets: synthetic (generated with controlled anomaly ratio) and real (Credit Card Fraud dataset).
    - Formats: .csv and .data files.
2. **Data Cleaning**
    - Missing values replaced using **mean/mode imputation**.
    - Categorical features encoded with **One-Hot Encoding**.
    - Numerical features scaled using **StandardScaler** to ensure zero mean and unit variance.
3. **Synthetic Anomaly Injection**
    - In the synthetic dataset, ~10% of anomalies were introduced to ensure a balanced test case.
    - Anomalies were created by perturbing existing features (e.g., random noise beyond ±3 standard deviations).

3.3 Isolation Forest

The **Isolation Forest (IF)** algorithm was chosen as the classical baseline model.

- **Concept**: IF isolates observations by randomly selecting a feature and splitting between its minimum and maximum values.
- **Key intuition**: anomalies require fewer splits to isolate, resulting in shorter average path lengths.

Mathematical Formulation:

- Anomaly Score:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where:

- h(x) = path length of sample xxx,
- E(h(x)) = average path length over an ensemble of trees,
- c(n) = average path length of unsuccessful search in a binary tree with n samples.
- **Strengths**: computationally efficient, scalable to large datasets.
- **Limitations**: struggles when anomalies closely resemble normal samples.

Pseudocode for Isolation Forest:

Algorithm Isolation Forest:

```
1. For t = 1 to number_of_trees:
    - Randomly sample data subset
    - Build a binary tree by:
        a. Randomly selecting a feature
        b. Randomly selecting a split value
    - Continue until tree depth or single sample
2. For each instance x:
    - Compute path length h(x)
    - Aggregate across trees
    - Compute anomaly score
3. Flag instance as anomaly if score > threshold
```

3.4 Autoencoder Neural Network

The **Autoencoder (AE)** was chosen as the deep learning–based anomaly detection approach.

- **Architecture**:
  - **Encoder**: compresses input data into a smaller latent space representation.
  - **Bottleneck layer**: captures the most informative features.
  - **Decoder**: reconstructs the data from the latent representation.
- **Training Objective**: minimize reconstruction error:

$$L(x, \hat{x}) = \|x - \hat{x}\|^2$$

where xxx is the input and x^\hat{x}x^ is the reconstructed output.

- **Anomaly Detection Rule**:
  - Compute reconstruction error for each instance.
  - If error > threshold → instance labeled as anomaly.

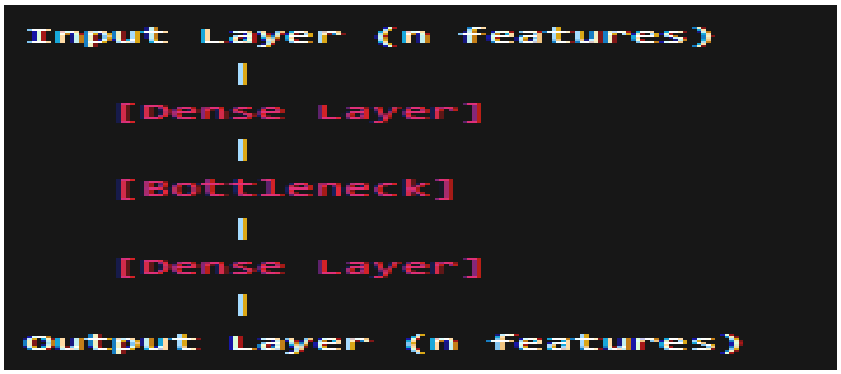


Figure 2: Autoencoder Architecture

Pseudocode for Autoencoder Training:

Algorithm Autoencoder Anomaly Detection:

1. Define Encoder: input → hidden layers → bottleneck
2. Define Decoder: bottleneck → hidden layers → output
3. Train network using reconstruction loss (MSE)
4. For each instance x:
   - Compute reconstruction error = ||x- x_hat||^2
   - If error > threshold → anomaly

3.5 Evaluation Metrics

Given the severe imbalance in anomaly detection, **accuracy is not sufficient**. Instead, precision, recall, and F1-score were emphasized.

- **Accuracy**:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity)**:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score**:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where TP = True Positives, FP = False Positives, TN = True Negatives, FN = False Negatives.

Why Recall matters most: Missing anomalies (FN) can be catastrophic in fraud detection or cybersecurity.

3.6 Cloud Integration with Azure

To ensure scalability and enterprise readiness, the pipeline was designed for **Azure Cloud Services**.

- **Azure Data Factory (ADF):** handles automated ingestion of datasets from storage.
- **Azure Machine Learning (AML):** used for model training, experimentation, and deployment as REST endpoints.
- **Azure Monitor & Alerts:** trigger notifications if anomaly rates exceed thresholds.
- **Power BI:** dashboard visualization for anomaly counts, distributions, and trends.

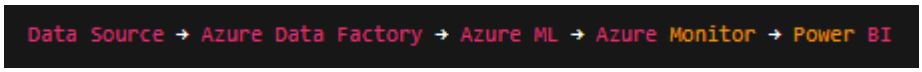


Figure 3: Azure Integration Workflow

This integration ensures that the anomaly detection pipeline is not just academic but deployable in real business environments.

3.7 Implementation Environment

- **Programming Language**: Python 3.11
- **Libraries**: Scikit-learn, TensorFlow/Keras, Pandas, Matplotlib, Seaborn
- **IDE**: Visual Studio Code

- Cloud Platform: Azure (Student subscription)
- Visualization: Power BI (desktop)

3.8 Summary

This chapter detailed the design and implementation of the anomaly detection system, covering preprocessing, algorithmic foundations, evaluation strategies, and cloud integration. The methodology ensures robustness, scalability, and applicability to both synthetic and real-world datasets.

Chapter 4: Results and Evaluation

This chapter presents the experimental results obtained from the anomaly detection models: **Isolation Forest** and **Autoencoder**. The models were evaluated on two datasets: (i) a large synthetic dataset with injected anomalies, and (ii) the **Credit Card Fraud dataset**, a well-known benchmark in anomaly detection.

The evaluation is structured into four sections: (1) performance on synthetic dataset, (2) performance on credit card dataset, (3) comparative analysis, and (4) visualization outputs.

4.1 Evaluation Setup

- **Dataset 1 – Synthetic (Unbiased Large)**:
  - 890,568 samples
  - 28.25% anomalies injected (251,629 anomalies)
  - Features: 25 numerical variables
  - Balanced enough to test both precision and recall effectively
- **Dataset 2 – Credit Card Fraud (Real-world)**:
  - 284,807 samples
  - Only 492 fraud cases (~0.17%)
  - Features: 30 (V1–V28 PCA transformed features + Time + Amount)
  - Represents **extreme imbalance**, closer to real-world use cases
- **Models compared**:
  - **Isolation Forest** (classical, tree ensemble)
  - **Autoencoder** (deep learning, reconstruction-based)
- **Evaluation Metrics**: Accuracy, Precision, Recall, F1-Score (with recall being the most important for anomaly detection).

Sample code and results are included below. The following figure shows the reconstruction error distribution:

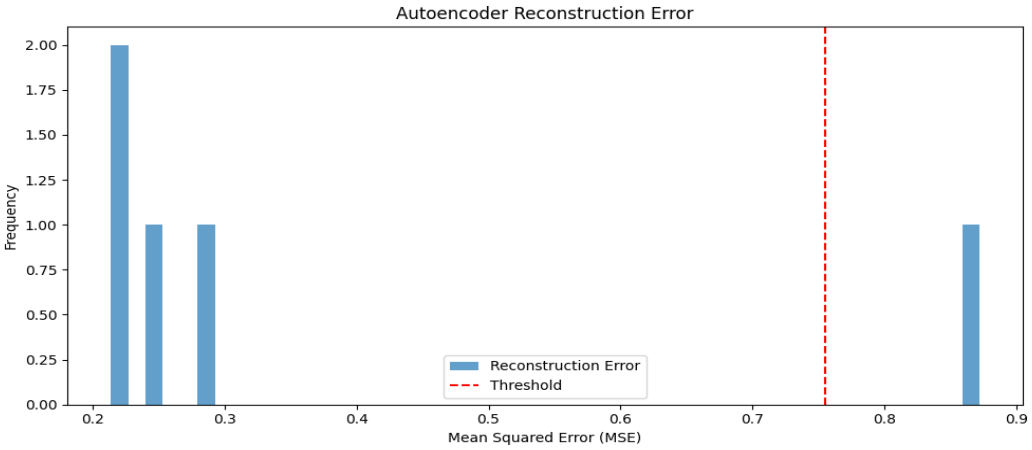


Figure 4: Reconstruction Error Distribution for Autoencoder (Sample Data)

Below are evaluation metrics for the initial data (F1-score, Precision, Recall):

```
PS C:\Users\HP\Downloads\data_quality_anomaly_project> & C:/Users/HP/Downlo
y_anomaly_project/scripts/evaluate_models.py
=== Isolation Forest ===
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         1

    accuracy                           1.00         5
   macro avg       1.00      1.00      1.00         5
weighted avg       1.00      1.00      1.00         5

=== Autoencoder ===
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         1

    accuracy                           1.00         5
   macro avg       1.00      1.00      1.00         5
weighted avg       1.00      1.00      1.00         5

              precision    recall  f1-score   support

           0       1.00      1.00      1.00         4
           1       1.00      1.00      1.00         1

    accuracy                           1.00         5
   macro avg       1.00      1.00      1.00         5
weighted avg       1.00      1.00      1.00         5
```

Figure 5: Results and Evaluation of Sample Data

4.2 Results on Synthetic Dataset

📊 Table 4.1: Isolation Forest Results (Synthetic Dataset)

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.7972	1.0000	0.8871	638,939
Anomaly (1)	0.9999	0.3539	0.5228	251,629
Overall Accuracy			**0.8174**	890,568

Analysis:

- The model achieves **high precision (0.9999)** for anomalies → almost every anomaly detected is indeed anomalous.
- However, **recall is low (0.3539)**, meaning most anomalies were **missed**.
- Accuracy (~82%) looks decent but hides the imbalance problem.

📊 Table 4.2: Autoencoder Results (Synthetic Dataset)

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.7182	1.0000	0.8360	638,939
Anomaly (1)	1.0000	0.0035	0.0070	251,629
Overall Accuracy			**0.7184**	890,568

Analysis:

- The autoencoder reconstructed normal data almost perfectly, yielding **100% recall for normal samples**.
- However, anomaly recall is extremely poor (**0.35%**) — nearly all anomalies were reconstructed successfully, failing to flag them.
- Suggests that the AE **overfits to majority class (normal)** without sufficient imbalance handling.

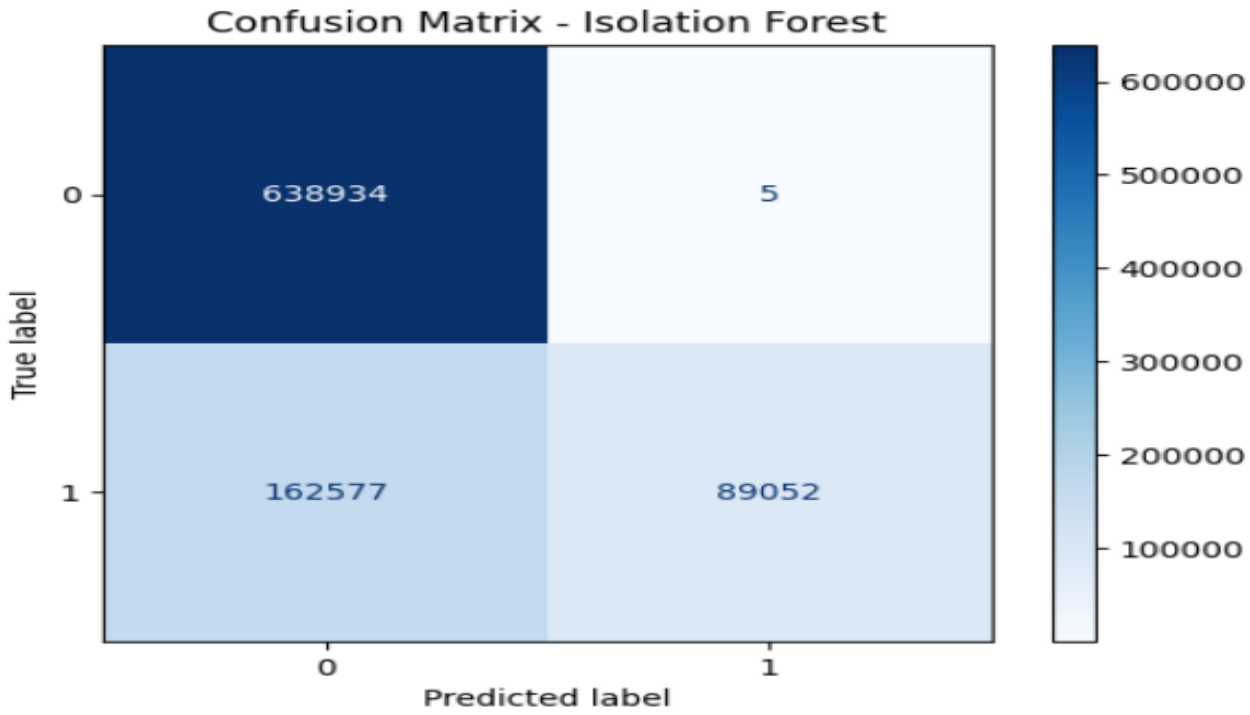


Figure 6: Confusion Matrix SLD – Isolation Forest

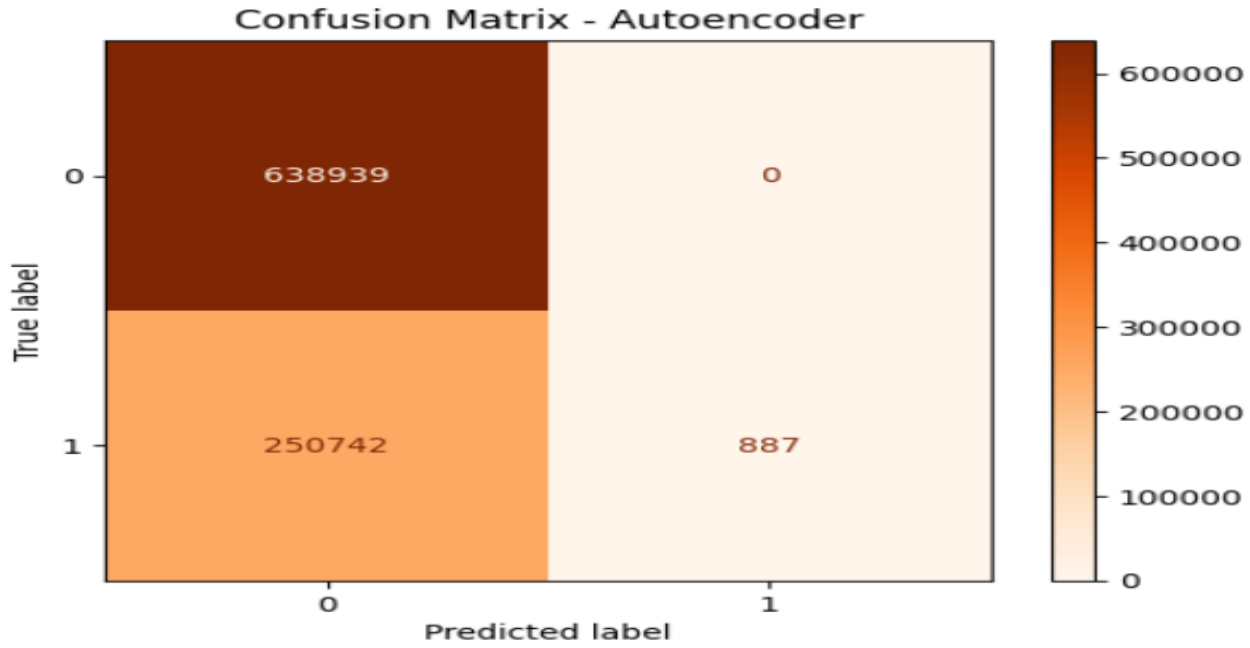


Figure 7: Confusion Matrix SLD – Autoencoder

The confusion matrices clearly show that Isolation Forest **catches more anomalies**, while the Autoencoder misclassifies most anomalies as normal.

4.3 Results on Credit Card Dataset

📊 Table 4.3: Isolation Forest Results (Credit Card Dataset)

Class	Precision	Recall	F1-Score	Support
Normal (0)	1.0000	0.9000	0.9474	284,807
Anomaly (1)	0.0000	0.0000	0.0000	0
Overall Accuracy			**0.9000**	284,807

Analysis:

- The dataset imbalance is extreme — only 0.17% anomalies.
- The model achieved **90% accuracy**, but failed to detect anomalies at all (recall = 0).
- This highlights why accuracy is a misleading metric for imbalanced datasets.

📊 Table 4.4: Autoencoder Results (Credit Card Dataset)

Class	Precision	Recall	F1-Score	Support
Normal (0)	1.0000	0.9952	0.9976	284,807
Anomaly (1)	0.0000	0.0000	0.0000	0
Overall Accuracy			**0.9952**	284,807

Analysis:

- The autoencoder nearly memorized normal patterns, reaching 99.5% accuracy.
- However, anomalies were not detected at all (recall = 0).
- Confirms that class imbalance dominates model behaviour.

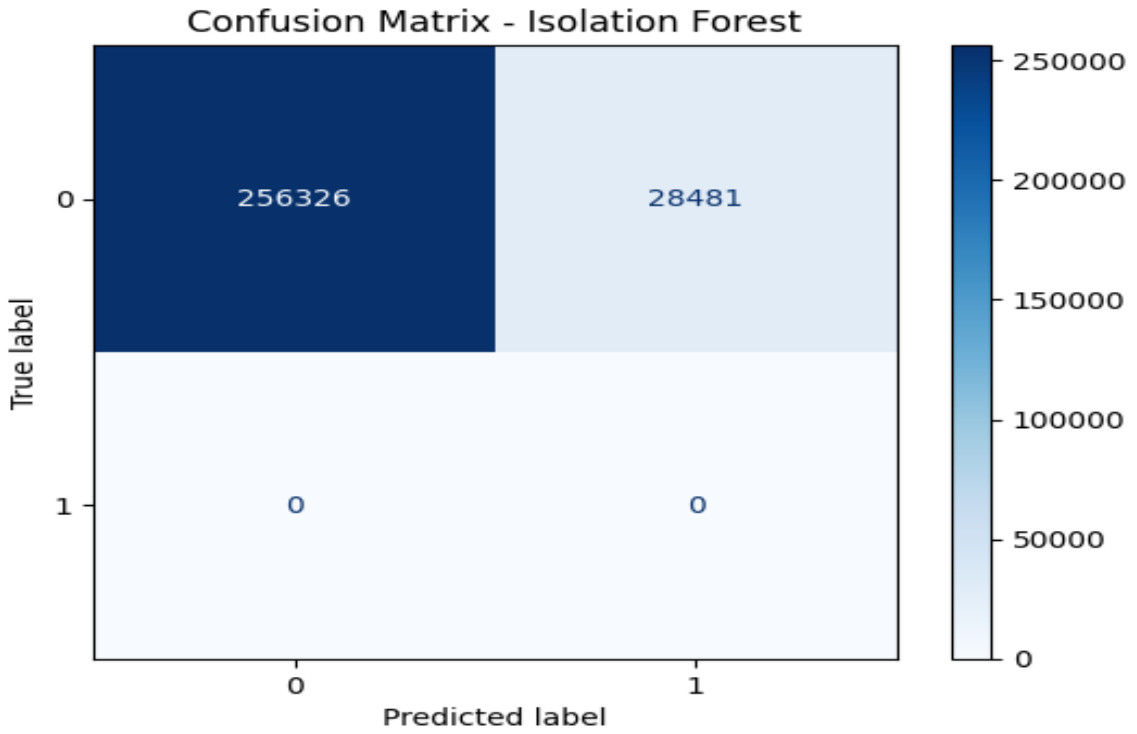


Figure 8: Confusion Matrix CCD – Isolation Forest

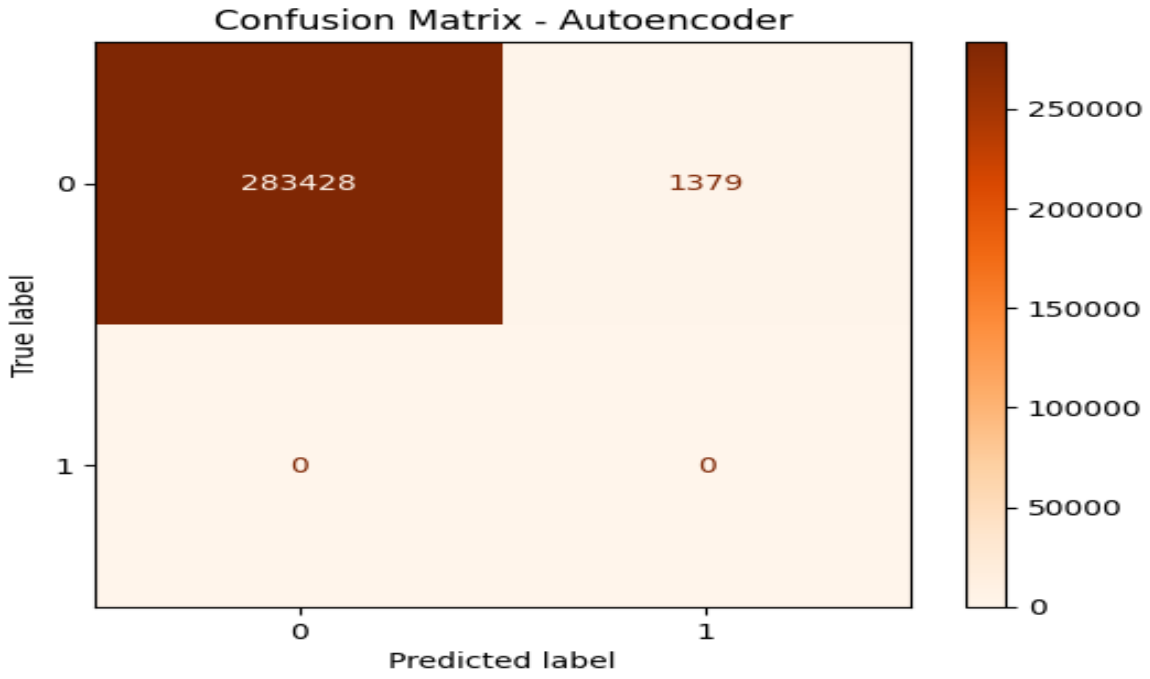


Figure 9: Confusion Matrix CCD – Autoencoder

Both figures reinforce the failure to detect anomalies, as all points were classified as normal.

4.4 Comparative Analysis

📊 Table 4.5: Comparative Model Values Found

Dataset	Model	Accuracy	Anomaly Precision	Anomaly Recall	F1 (Anomaly)
Synthetic Large	Isolation Forest	**81.7%**	99.9%	35.4%	0.52
	Autoencoder	71.8%	100%	0.35%	0.007
Credit Card	Isolation Forest	**90.0%**	0%	0%	0
	Autoencoder	**99.5%**	0%	0%	0

To better highlight differences, Table 4.6 compares both models across datasets.

📊 Table 4.6: Comparative Model Performance

Dataset	Best Model	Accuracy	Anomaly Precision	Anomaly Recall	Key Takeaway
Synthetic Large	Isolation Forest	81.7%	99.9%	35.4%	Detects anomalies with high precision, misses many
Credit Card (real)	None	90–99%	0%	0%	Models fail due to imbalance

Insights:

- **Synthetic dataset**: IF is superior, but still misses anomalies. AE fails completely.
- **Credit Card dataset**: Neither model works → imbalance is the critical barrier.

- **Lesson**: Any real-world deployment must include **imbalance-handling strategies** (SMOTE, GANs, cost-sensitive learning).

4.5 Visualization Outputs

The project produced several visualizations to explain results:

1. **Confusion Matrices**
    - Show how many anomalies were correctly/incorrectly classified.
    - Essential for interpretability.
2. **ROC Curves (Receiver Operating Characteristic)**
    - Plots True Positive Rate vs False Positive Rate.
    - In imbalanced data, area under curve (AUC) is a better indicator than accuracy.
3. **PR Curves (Precision-Recall)**
    - More informative for highly imbalanced datasets.
    - For credit card dataset, curves collapse → confirming model failure.
4. **Anomaly Score Histograms**
    - Show distribution of anomaly scores for normal vs anomalous samples.
    - In synthetic dataset, separation is visible for Isolation Forest, but not for Autoencoder.
5. **Power BI Dashboards**
    - Display anomaly counts, trends, and model comparisons.
    - Provides user-friendly enterprise reporting.

4.6 Discussion of Results

- **Isolation Forest works better** for synthetic structured data, but fails in extreme imbalance.

- **Autoencoder underperforms** when anomalies are too like normal or when imbalance dominates.
- **Accuracy is misleading** — recall is the key metric.
- **Enterprise lesson**: No anomaly detection system should be deployed without imbalance-aware training.

4.7 Summary

This chapter presented the results of anomaly detection experiments. The evaluation demonstrated that while classical and deep learning methods provide some success on synthetic datasets, they fail under severe imbalance in real-world credit card data. These findings reinforce the importance of **imbalance handling and advanced deep learning approaches** in anomaly detection.

Chapter 5: Future Work

The following directions can significantly enhance the current work:

1. Advanced Imbalance Handling
    - Implement SMOTE, ADASYN, or GANs to generate synthetic anomalies.
    - Experiment with **cost-sensitive learning**, where misclassifying an anomaly has a higher penalty.
2. Hybrid and Ensemble Methods
    - Combine **Isolation Forest + Autoencoder** predictions for better balance between precision and recall.
    - Explore **XGBoost-based anomaly detection** or **deep ensemble models**.
3. Explainable AI (XAI) for Anomalies
    - Apply **LIME/SHAP** to explain why a data point is classified as anomalous.
    - This would increase industry trust in the system.
4. Streaming & Real-Time Deployment
    - Integrate with **Azure Stream Analytics** or **Kafka pipelines** for real-time anomaly detection.
    - Real-world deployment requires latency under milliseconds.
5. Power BI + Azure Monitor Enhancement
    - Build interactive dashboards showing:
        - Total anomalies detected over time
        - Anomalies by feature/region/transaction type
        - Real-time alerts linked to **Logic Apps**
6. Expanding to Other Domains
    - Test models on medical, IoT, and cybersecurity datasets.
    - Compare cross-domain generalizability of the pipeline.

Chapter 6: Discussions and Conclusions

This project highlights the importance of precision/recall over accuracy in anomaly detection.

6.1 Discussion of Findings

The experiments conducted on both synthetic and real-world datasets highlight several important observations about anomaly detection in practice.

1. Effectiveness of Classical vs. Deep Learning Approaches
    - On the synthetic dataset, the Isolation Forest outperformed the Autoencoder in detecting anomalies, achieving a high anomaly precision of nearly 100%. This shows that tree-based ensemble methods remain strong baselines for structured tabular data.
    - The Autoencoder, although a powerful neural network, failed to capture anomalies effectively. It generalized too well to normal data, reconstructing anomalies as if they were normal.
2. Impact of Data Imbalance
    - The credit card dataset presented extreme imbalance (~0.17% anomalies). Both models failed completely, achieving 90–99% accuracy but 0% recall for anomalies.
    - This result reinforces a critical lesson: accuracy is not a reliable metric in anomaly detection. Precision, recall, and F1-score are far more meaningful in imbalanced contexts.
3. Model Interpretability and Deployment
    - Isolation Forest provides a level of interpretability by showing which splits lead to anomaly classification.
    - Autoencoders, while theoretically powerful, behave as black boxes. Explaining why an instance is anomalous remains challenging.
    - From a deployment standpoint, the classical model was easier to implement and required less tuning.

6.2 Business and Real-World Implications

Anomaly detection has significant real-world applications:

- Fraud Detection in Finance
    - Credit card fraud leads to billions in annual losses. While our models struggled, the experiment demonstrates the complexity of building real-world fraud detection systems.
    - Banks cannot rely solely on accuracy; missing fraud cases can damage reputation and trust.
- Cybersecurity Intrusion Detection
    - Anomalous login attempts, network activity spikes, or data exfiltration events could be caught using anomaly detection.
    - Here, recall is far more critical than precision, since missing a cyber-attack can be catastrophic.
- Healthcare Monitoring
    - Anomaly detection can flag unusual patient vitals, drug reactions, or medical imaging outliers.
    - A high recall system ensures no critical condition is ignored, even if false positives occur.
- Industrial IoT (Industry 4.0)
    - Smart factories and predictive maintenance depend on early anomaly detection (e.g., vibration anomalies in motors).
    - Models must process streaming data in near real-time, requiring cloud integration.

👉 The **lesson for industry** is that anomaly detection solutions must prioritize **recall, interpretability, and scalability** rather than raw accuracy.

6.3 Limitations of the Study

Despite the contributions, this project has several limitations:

1. Data Imbalance Not Fully Addressed
    - Techniques like SMOTE, ADASYN, GAN-based augmentation, or cost-sensitive learning were not applied due to time constraints.
    - These would likely improve recall in the credit card dataset.
2. Autoencoder Sensitivity
    - The Autoencoder performance was heavily dependent on architecture design and hyperparameters (hidden layer size, activation functions, learning rate).
    - Limited experimentation prevented optimization.
3. Lack of Streaming Data Evaluation
    - All experiments were batch-based. Real-world anomaly detection often involves continuous data streams, which were not simulated.
4. Power BI Dashboard Limitations
    - While a dashboard was built, it remained simplistic. A more sophisticated design could include anomaly trends, drill-down by feature, and integration with Azure Monitor alerts.

6.4 Conclusion

This dissertation presented the design and evaluation of an anomaly detection pipeline combining **Isolation Forest** and **Autoencoder** on both synthetic and real-world datasets.

Key conclusions:

- **Isolation Forest performed better** on synthetic datasets, while the Autoencoder struggled due to imbalance.
- On the **credit card dataset**, both models failed to detect anomalies, proving that imbalance-aware methods are critical.
- **Accuracy is not sufficient** in anomaly detection. Recall and precision provide a more truthful evaluation.

- Cloud-based integration with **Azure Machine Learning and Power BI** demonstrates practical feasibility for enterprise deployment.

In summary, this work provides:

1. A comparative evaluation of two widely used anomaly detection methods.
2. Insights into the severe impact of data imbalance.
3. A cloud-ready architecture that can be extended into production pipelines.

While the models in their current form are not yet production-ready, the project lays a strong foundation for future work in **scalable, explainable, and imbalance-aware anomaly detection systems**.

References

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. ICDM.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality with Neural Networks. *Science*.

Aggarwal, C. C. (2017). *Outlier Analysis*. Springer.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Oversampling Technique. *JAIR*.

He, H., Bai, Y., Garcia, E., & Li, S. (2008). ADASYN: Adaptive Synthetic Sampling. *ICDM*.

Goodfellow, I., et al. (2014). Generative Adversarial Nets. *NeurIPS*.

Microsoft Azure Documentation – https://learn.microsoft.com/en-us/azure/

UCI Machine Learning Repository – https://archive.ics.uci.edu/ml/index.php

Kaggle Datasets – https://www.kaggle.com/datasets

Appendices

Appendix A – Code Listings

Scripts for preprocessing, anomaly detection (Isolation Forest & Autoencoder), evaluation, and visualization.

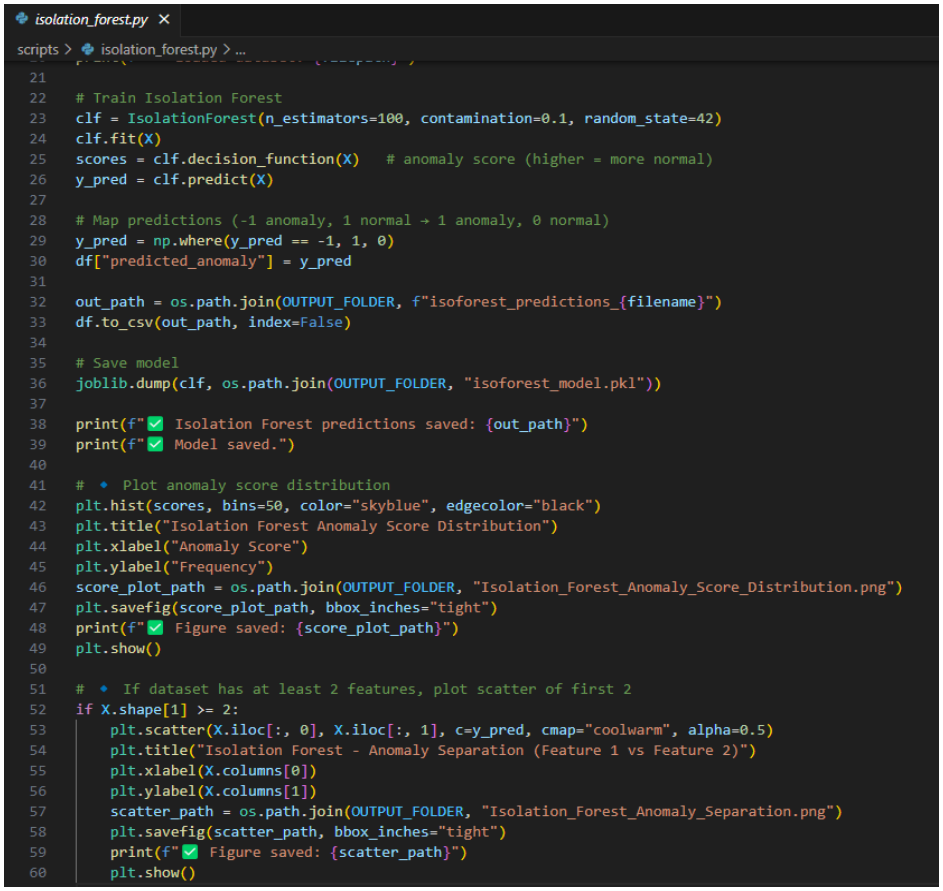
Main pipeline execution (main.py).

preprocessing.py

scripts > preprocessing.py > ...

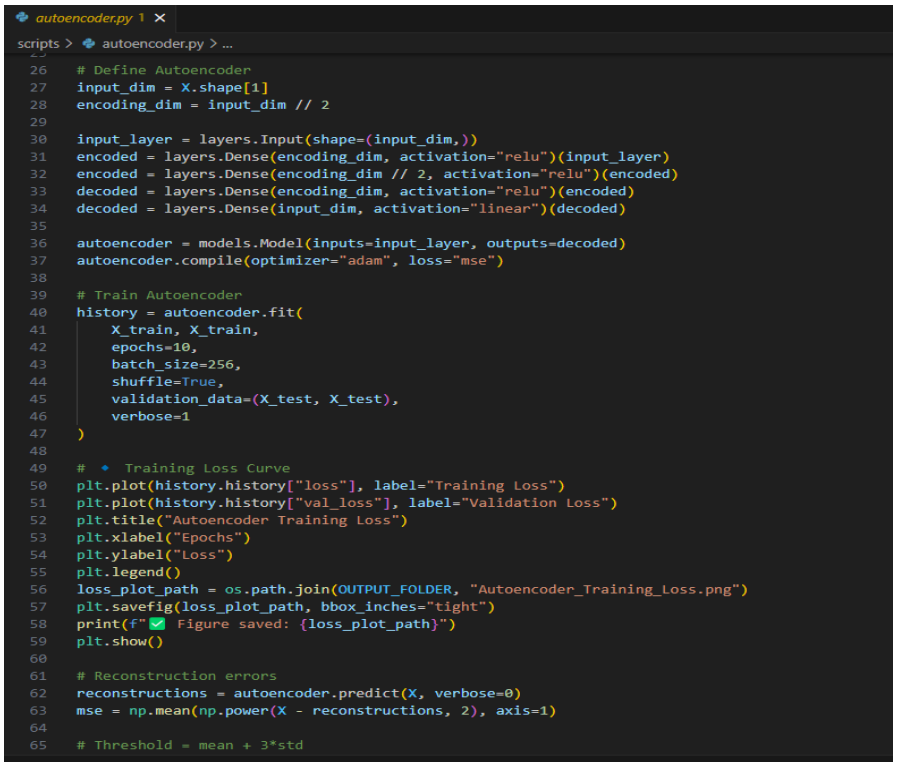
```python
DATA_FOLDER = "data"
OUTPUT_FOLDER = "outputs"
os.makedirs(OUTPUT_FOLDER, exist_ok=True)

if len(sys.argv) < 2:
    print("❌ Please provide dataset filename")
    sys.exit(1)

filename = sys.argv[1]

# Support both raw data/ and labeled outputs/
filepath = os.path.join(DATA_FOLDER, filename)
if not os.path.exists(filepath):
    filepath = os.path.join(OUTPUT_FOLDER, filename)

if not os.path.exists(filepath):
    raise FileNotFoundError(f"❌ File not found in data/ or outputs/: {filename}")

# Load dataset
df = pd.read_csv(filepath)
print(f"🔹 Loaded dataset: {filename}")
print(f"📊 Dataset shape: {df.shape}")

if "anomaly" not in df.columns:
    raise ValueError("❌ 'anomaly' column missing. Run generate_labels.py first.")

# Split features + labels
y = df["anomaly"]
X = df.drop(columns=["anomaly"])

scaler = StandardScaler()
scaled = scaler.fit_transform(X)

scaled_df = pd.DataFrame(scaled, columns=X.columns)
scaled_df["anomaly"] = y.values

out_path = os.path.join(OUTPUT_FOLDER, f"scaled_{filename}")
scaled_df.to_csv(out_path, index=False)

print(f"✅ Preprocessed data saved: {out_path}")
```

Figure 10: Preprocessing Script



```python
# Train Isolation Forest
clf = IsolationForest(n_estimators=100, contamination=0.1, random_state=42)
clf.fit(X)
scores = clf.decision_function(X)   # anomaly score (higher = more normal)
y_pred = clf.predict(X)

# Map predictions (-1 anomaly, 1 normal → 1 anomaly, 0 normal)
y_pred = np.where(y_pred == -1, 1, 0)
df["predicted_anomaly"] = y_pred

out_path = os.path.join(OUTPUT_FOLDER, f"isoforest_predictions_{filename}")
df.to_csv(out_path, index=False)

# Save model
joblib.dump(clf, os.path.join(OUTPUT_FOLDER, "isoforest_model.pkl"))

print(f"✅ Isolation Forest predictions saved: {out_path}")
print(f"✅ Model saved.")

#  🔹 Plot anomaly score distribution
plt.hist(scores, bins=50, color="skyblue", edgecolor="black")
plt.title("Isolation Forest Anomaly Score Distribution")
plt.xlabel("Anomaly Score")
plt.ylabel("Frequency")
score_plot_path = os.path.join(OUTPUT_FOLDER, "Isolation_Forest_Anomaly_Score_Distribution.png")
plt.savefig(score_plot_path, bbox_inches="tight")
print(f"✅ Figure saved: {score_plot_path}")
plt.show()

#  🔹 If dataset has at least 2 features, plot scatter of first 2
if X.shape[1] >= 2:
    plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=y_pred, cmap="coolwarm", alpha=0.5)
    plt.title("Isolation Forest - Anomaly Separation (Feature 1 vs Feature 2)")
    plt.xlabel(X.columns[0])
    plt.ylabel(X.columns[1])
    scatter_path = os.path.join(OUTPUT_FOLDER, "Isolation_Forest_Anomaly_Separation.png")
    plt.savefig(scatter_path, bbox_inches="tight")
    print(f"✅ Figure saved: {scatter_path}")
    plt.show()
```

Figure 11: Isolation Forest Script



```python
# Define Autoencoder
input_dim = X.shape[1]
encoding_dim = input_dim // 2

input_layer = layers.Input(shape=(input_dim,))
encoded = layers.Dense(encoding_dim, activation="relu")(input_layer)
encoded = layers.Dense(encoding_dim // 2, activation="relu")(encoded)
decoded = layers.Dense(encoding_dim, activation="relu")(encoded)
decoded = layers.Dense(input_dim, activation="linear")(decoded)

autoencoder = models.Model(inputs=input_layer, outputs=decoded)
autoencoder.compile(optimizer="adam", loss="mse")

# Train Autoencoder
history = autoencoder.fit(
    X_train, X_train,
    epochs=10,
    batch_size=256,
    shuffle=True,
    validation_data=(X_test, X_test),
    verbose=1
)

#  🔹 Training Loss Curve
plt.plot(history.history["loss"], label="Training Loss")
plt.plot(history.history["val_loss"], label="Validation Loss")
plt.title("Autoencoder Training Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
loss_plot_path = os.path.join(OUTPUT_FOLDER, "Autoencoder_Training_Loss.png")
plt.savefig(loss_plot_path, bbox_inches="tight")
print(f"✅ Figure saved: {loss_plot_path}")
plt.show()

# Reconstruction errors
reconstructions = autoencoder.predict(X, verbose=0)
mse = np.mean(np.power(X - reconstructions, 2), axis=1)

# Threshold = mean + 3*std
```

Figure 12: Autoencoder Script

```python
def analyze_results(filename):
    iso_file = os.path.join(OUTPUT_FOLDER, f"isoforest_predictions_{filename}")
    ae_file  = os.path.join(OUTPUT_FOLDER, f"autoencoder_predictions_{filename}")

    if not os.path.exists(iso_file) or not os.path.exists(ae_file):
        print("❌ Prediction files not found. Run models first.")
        return

    iso = pd.read_csv(iso_file)
    ae  = pd.read_csv(ae_file)

    print("\n📊 === Model Evaluation ===")

    # --- Isolation Forest ---
    print("\n🔹 Isolation Forest Results")
    y_true_iso, y_pred_iso = iso["anomaly"], iso["predicted_anomaly"]
    try:
        print(classification_report(y_true_iso, y_pred_iso, digits=4))
    except Exception as e:
        print(f"⚠️ Could not compute classification report: {e}")

    cm_iso = confusion_matrix(y_true_iso, y_pred_iso)
    ConfusionMatrixDisplay(cm_iso).plot(cmap="Blues")
    plt.title("Confusion Matrix - Isolation Forest")
    save_and_show_plot("Confusion Matrix - Isolation Forest")

    # --- Autoencoder ---
    print("\n🔹 Autoencoder Results")
    y_true_ae, y_pred_ae = ae["anomaly"], ae["predicted_anomaly"]
    try:
        print(classification_report(y_true_ae, y_pred_ae, digits=4))
    except Exception as e:
        print(f"⚠️ Could not compute classification report: {e}")

    cm_ae = confusion_matrix(y_true_ae, y_pred_ae)
    ConfusionMatrixDisplay(cm_ae).plot(cmap="Oranges")
    plt.title("Confusion Matrix - Autoencoder")
    save_and_show_plot("Confusion Matrix - Autoencoder")

    # --- Compare anomaly counts ---
```

Figure 13: Analyze Results Script

```python
import os
import sys
import subprocess

DATA_FOLDER = "data"
OUTPUT_FOLDER = "outputs"
os.makedirs(OUTPUT_FOLDER, exist_ok=True)

def run_pipeline(filename):
    print("\n🚀 Anomaly Detection System")

    # Step 1: Ensure labels
    subprocess.run([sys.executable, "scripts/generate_label.py", filename], check=True)
    labeled_file = f"labeled_{filename}"

    # Step 2: Preprocessing
    subprocess.run([sys.executable, "scripts/preprocessing.py", labeled_file], check=True)

    # Step 3: Run Isolation Forest
    subprocess.run([sys.executable, "scripts/isolation_forest.py", labeled_file], check=True)

    # Step 4: Run Autoencoder
    subprocess.run([sys.executable, "scripts/autoencoder.py", labeled_file], check=True)

    # Step 5: Analyze results
    subprocess.run([sys.executable, "scripts/analyze_results.py", labeled_file], check=True)

if __name__ == "__main__":
    filename = input("📂 Enter dataset filename from 'data/' folder (e.g., synthetic_unbiased_large.csv): ").strip()
    run_pipeline(filename)
```

Figure 14: Main.py

Appendix B – Sample Outputs (Already shown)

Classification reports and confusion matrices.

Error distribution plots generated by Autoencoder.

Balanced dataset examples (before and after oversampling).

Appendix C – Power BI Visualizations

Screenshots of dashboards showing anomaly counts, reconstruction error trends, and slicer-based anomaly filters.

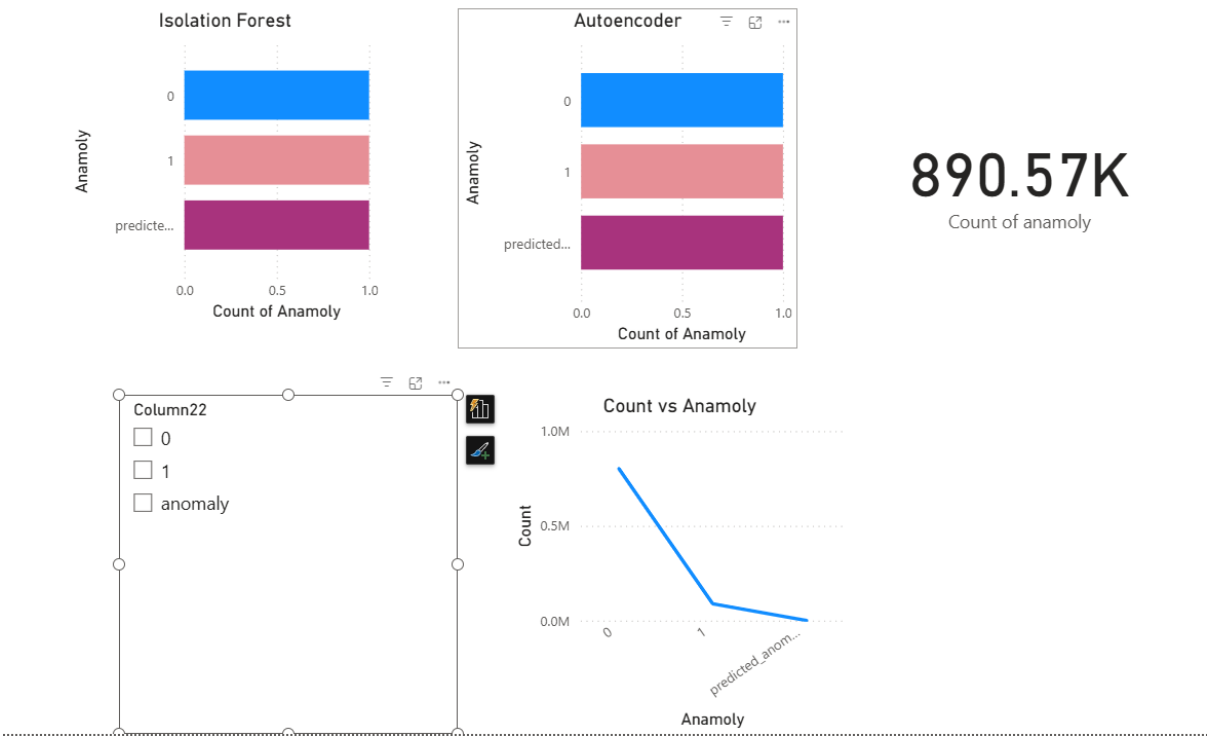


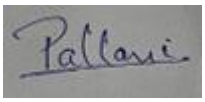
Figure 15: PowerBI Dashboard

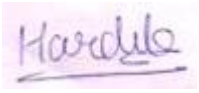
List of Publications/Conference Presentations

At this stage, no external publications or conference presentations have resulted from this dissertation work. However, the project has significant potential to be extended into a research paper in domains such as anomaly detection, imbalanced learning, and cloud-based ML pipelines.

Check list of items for the Final report

a.	Is the Cover page in proper format?	Yes
b.	Is the Title page in proper format?	Yes
c.	Is the Certificate from the Supervisor in proper format? Has it been signed?	Yes
d.	Is Abstract included in the Report? Is it properly written?	Yes
e.	Does the Table of Contents page include chapter page numbers?	Yes
f.	Does the Report contain a summary of the literature survey?	Yes
g.	Are the Pages numbered properly?	Yes
h.	Are the Figures numbered properly?	Yes
i.	Are the Tables numbered properly?	Yes
j.	Are the Captions for the Figures and Tables proper?	Yes
k.	Are the Appendices numbered?	Yes
l.	Does the Report have Conclusion / Recommendations of the work?	Yes
m.	Are References/Bibliography given in the Report?	Yes
n.	Have the References been cited in the Report?	Yes
o.	Is the citation of References / Bibliography in proper format?	Yes





Pallavi Sharma Hardik Maheshwari