Anomaly Detection with Isolation Forest for Network Traffic

A Course End Project Submitted in Partial Fulfillment of the Requirements
for the Course of

INFORMATION SECURITY

In

Department of Computer Science and Engineering

By

ROLL NUMBER	NAME OF THE STUDENT
21881A05D7	Aakash Reddy
21881A05D9	Vaishnavi Reddy
21881A05F8	Krishna Varshita Reddy

VARDHAMAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Affiliated to **JNTUH**, Approved by **AICTE**, Accredited by **NAAC** with **A++** Grade, **ISO 9001:2015** Certified
Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

September 2024

VARDHAMAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

Affiliated to **JNTUH**, Approved by **AICTE**, Accredited by **NAAC** with **A++** Grade, **ISO 9001:2015** Certified
Kacharam, Shamshabad, Hyderabad – 501218, Telangana, India

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that this is a bonafide record of the course end project work entitled, " **Anomaly Detection with Isolation Forest for Network Traffic** ", *done by Aakash Reddy [21881A05D7], Vaishnavi Reddy[21881A05D9], Krishna Varshita Reddy[21881A05F8]* submitted to the faculty of **Computer Science and Engineering**, in partial fulfillment of the requirements for the course of **Information Security** during the year 2023-2024 (VI Semester).

Course Instructor:	**Head of the Department:**
Ms. Farhana Begum	**Dr. Ramesh Karnati**
ASSOCIATE PROFESSOR,	HOD,
Dept of IT	Dept of CSE
	Vardhaman College of Engineering

Semester End Examination held on……………………………………………

SIGNATURE OF THE EXAMINER(S)

1.

2.

Abstract

The ever-evolving threat landscape necessitates robust network security solutions. Anomaly detection, the identification of deviations from established traffic patterns, plays a critical role in this domain. This work investigates the application of Isolation Forest, an unsupervised machine learning algorithm, for anomaly detection in network traffic data.

Traditional signature-based intrusion detection systems (IDS) rely on predefined attack signatures, rendering them ineffective against novel threats (zero-day attacks). Isolation Forest addresses this limitation by learning patterns of normal traffic behaviour from unlabelled data. This unsupervised approach eliminates the need for pre-classified network traffic data, often a laborious and resource-intensive task.

The provided Python code serves as a practical illustration of implementing an Isolation Forest model. It walks through the essential steps of data preparation, model training, and anomaly prediction. By analysing network traffic features like packet size, source/destination IP, and protocol information, the model can effectively isolate anomalous traffic patterns that might indicate malicious activity.

However, it's important to acknowledge the inherent trade-offs associated with unsupervised anomaly detection. While Isolation Forest offers the advantage of not requiring labelled data, it might not achieve the same level of accuracy as supervised methods for well-defined attack patterns. Additionally, tuning hyperparameters like contamination rate is crucial for optimal performance.

In conclusion, this work highlights the potential of Isolation Forest as a valuable tool for network anomaly detection. Future exploration should delve into the comparative analysis of different anomaly detection algorithms and their suitability for specific network security scenarios. Integrating domain knowledge for feature selection and implementing the model on real-world network traffic data can further enhance its effectiveness in securing critical infrastructure.

List of Contents

CHAPTER-1

Introduction

What is Anomaly Detection?

Anomalies, also known as outliners, are data points that deviate significantly from the expected behaviour or norm within a data set. They are crucial to identify because they can signal potential problems, fraudulent activities, or interesting discoveries. Anomaly detection plays a vital role in various fields, including data analysis, machine learning, and network security.

Types of Anomalies

1. **Point Anomalies (Global Anomalies)**:
   - Point anomalies are individual data points that are statistically different from the majority of the data. They stand out as singular instances of unusual behavior within the dataset. For example, in financial transactions, a credit card purchase with an unusually high amount compared to typical transactions would be considered a point anomaly. Detection of point anomalies involves comparing each data point against a global statistical threshold.
2. **Contextual Anomalies (Conditional Anomalies)**:
   - Contextual anomalies depend on specific conditions or contexts. They are often observed in time-series data where patterns can change over time. A contextual anomaly may appear normal under one set of circumstances but abnormal under another. For instance, a sudden spike in temperature during winter months would be considered a contextual anomaly in weather data. Detecting contextual anomalies requires analyzing data in relation to its contextual factors or conditions.
3. **Collective Anomalies**:
   - Collective anomalies involve groups of related data points that together exhibit abnormal behavior, even though individual data points may appear normal when considered independently. These anomalies disrupt the overall data distribution or expected patterns. Detecting collective anomalies typically involves using complex pattern-based algorithms, especially in dynamic environments such as network traffic data. For example, a distributed denial-of-service (DDoS) attack involving multiple coordinated requests from different sources would be classified as a collective anomaly

.

CHAPTER-2

PROJECT DESCRIPTION

Isolation Forest is an unsupervised machine learning algorithm designed for anomaly detection. Unlike traditional clustering-based approaches, Isolation Forest operates on the principle of isolating anomalies rather than profiling normal points. This unique approach makes Isolation Forest computationally efficient and effective for high-dimensional data.

The fundamental concept behind Isolation Forest is that anomalies are rare and distinct, making them easier to isolate from the rest of the data. Anomalies deviate significantly from normal data points, which means they can be identified through fewer splits in a decision tree structure, leading to shorter path lengths in isolation trees.

How Isolation Forest Works

1. **Building Isolation Trees**:
   - The algorithm starts by creating a set of isolation trees, typically hundreds or even thousands of them. These trees are similar to traditional decision trees but are not built to classify data points into specific categories. Instead, isolation trees aim to isolate individual data points by repeatedly splitting the data based on randomly chosen features and split values. The construction of multiple trees helps to capture different aspects of the data, enhancing the robustness of anomaly detection.
2. **Splitting on Random Features**:
   - Isolation trees introduce randomness at each node. At each step, a random feature from the dataset is selected, and then a random split value is chosen within the range of that particular feature’s values. This randomness ensures that anomalies, which tend to be distinct from the majority of data points, are not hidden within specific branches of the tree. By introducing randomness, the algorithm avoids any bias towards specific patterns, making it more effective in detecting anomalies that are scattered across different dimensions of the dataset.

3. **Isolating Data Points**:
    - Data points are directed down the branches of the isolation tree based on their feature values. The process of splitting continues until each data point is isolated in a leaf node or a predefined maximum depth is reached. The essence of isolation lies in the idea that anomalies will be isolated much faster (i.e., with fewer splits) compared to normal points.
4. **Anomaly Score**:
    - The key concept behind Isolation Forests lies in the path length of a data point through an isolation tree. The path length is defined as the number of edges traversed from the root node to a leaf node. Since anomalies are easier to isolate, they tend to have shorter path lengths. Conversely, normal points require more splits to be isolated, resulting in longer path lengths.
5. **Anomaly Score Calculation**:
    - Each data point is evaluated through all the isolation trees in the forest. The anomaly score is calculated based on the average path length from all the trees. A normalization factor is applied to ensure that the scores are comparable across different datasets and tree structures.
6. **Identifying Anomalies**:
    - Data points with shorter average path lengths are considered more likely to be anomalies. This is because they were easier to isolate, suggesting they deviate significantly from the bulk of the data. A threshold is set to define the anomaly score that separates normal data points from anomalies. By adjusting this threshold, the sensitivity of the anomaly detection can be controlled to balance between false positives and false negatives.

Key Steps in the Project

1. **Data Collection**:
    - Acquire a comprehensive dataset of network traffic. This could include logs from network devices, packet captures, or synthetic datasets that simulate network traffic patterns. The dataset should cover a wide range of network activities, including both normal operations and known anomalies.
2. **Data Preprocessing**:

   - Clean and preprocess the data to ensure it is in a suitable format for analysis. This may involve handling missing values, normalizing data, and feature extraction. Preprocessing is crucial to ensure that the data fed into the Isolation Forest algorithm is of high quality and relevant to the detection task.

3. **Feature Engineering**:
   - Extract meaningful features from raw network traffic data. Examples include packet size, duration, source and destination IP addresses, protocols used, and time intervals between packets. Feature engineering transforms raw data into a structured format that the algorithm can process effectively, capturing the essential characteristics of network traffic.
4. **Implementation of Isolation Forest**:
   - Apply the Isolation Forest algorithm to the preprocessed data. This involves training the model on normal traffic patterns to establish a baseline and then using it to identify anomalies. The implementation will leverage existing machine learning libraries such as scikit-learn to build and train the isolation forest model.

5. **Evaluation**:
   - Evaluate the performance of the Isolation Forest model using metrics such as precision, recall, F1-score, and ROC-AUC. Validate the model's effectiveness in detecting true anomalies while minimizing false positives. Evaluation involves comparing the detected anomalies against known ground truth or labeled data, if available.
6. **Visualization**:
   - Create visualizations to interpret the results. This could include plots of the anomaly scores, distributions of network features, and visual representations of detected anomalies. Visualizations help in understanding the model's behavior and the nature of the detected anomalies, making it easier to communicate findings to stakeholders.
7. **Optimization and Tuning**:
   - Fine-tune the model parameters and preprocessing techniques to improve detection accuracy. This step may involve cross-validation and grid search methods. Optimization ensures that the model is performing at its best, striking a balance between sensitivity and specificity.

8. **Deployment**:
    - Develop a pipeline for real-time anomaly detection, where the trained model can analyze live network traffic and raise alerts for detected anomalies. Deployment involves integrating the anomaly detection system into the network infrastructure, ensuring it can operate in real-time and provide timely alerts to network administrators.

Dataset Description

The Iris dataset consists of 150 samples from three different species of iris flowers (Iris setosa, Iris versicolor, and Iris virginica). Each sample has four features: sepal length, sepal width, petal length, and petal width. For this anomaly detection task, the labels (species) are not used in the model training, adhering to the unsupervised nature of Isolation Forest.

Steps and Explanation

1. **Loading the Dataset**:
    - The Iris dataset is loaded, and the feature data and target labels are extracted. This dataset provides a well-structured example for testing anomaly detection methods.
2. **Splitting the Data**:
    - The dataset is split into training and testing sets to ensure that the model can be trained on one portion of the data and tested on another to evaluate its performance. This helps in assessing the generalization ability of the model.
3. **Training the Isolation Forest Model**:
    - An Isolation Forest model is initialized with a contamination parameter set to 0.1, indicating that approximately 10% of the data is expected to be anomalous. The model is then trained on the training

data. This parameter helps the model to understand the expected proportion of anomalies in the dataset.

4. **Predicting Anomalies**:
    ○ The model is used to predict anomalies in both the training and testing sets. The predictions label data points as either normal (1) or anomalous (-1). This step involves evaluating the model's ability to detect unusual patterns in unseen data.
5. **Printing Predictions**:
    ○ The predicted anomaly labels for both the training and testing sets are printed to the console to review the model's output. This provides an initial understanding of how many data points were classified as anomalies.
6. **Creating Scatter Plots**:
    ○ A function is defined to create scatter plots of the first two features of the dataset, highlighting normal and anomalous data points in different colors. This visualization helps to understand the distribution of anomalies detected by the model and provides a visual check on the model's performance.
7. **Visualizing the Results**:
    ○ The scatter plot function is called with the training and testing data, along with their respective anomaly predictions, to visualize the results. These visualizations help in interpreting the model's behavior and the nature of the detected anomalies, making it easier to communicate findings.

Anomaly detection using Isolation Forest: Implementation

Step 1-Load Python Libraries

```
from sklearn.ensemble import IsolationForest
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
```

Step 2-loading and Splitting the data set

```
iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

Step 3-Fitting the model

```
# initialize and fit the model
clf = IsolationForest(contamination=0.1)
clf.fit(X_train)
```

Step 4-Predictions

```
# predict the anomalies in the data
y_pred_train = clf.predict(X_train)
y_pred_test = clf.predict(X_test)
print(y_pred_train)
print(y_pred_test)
```

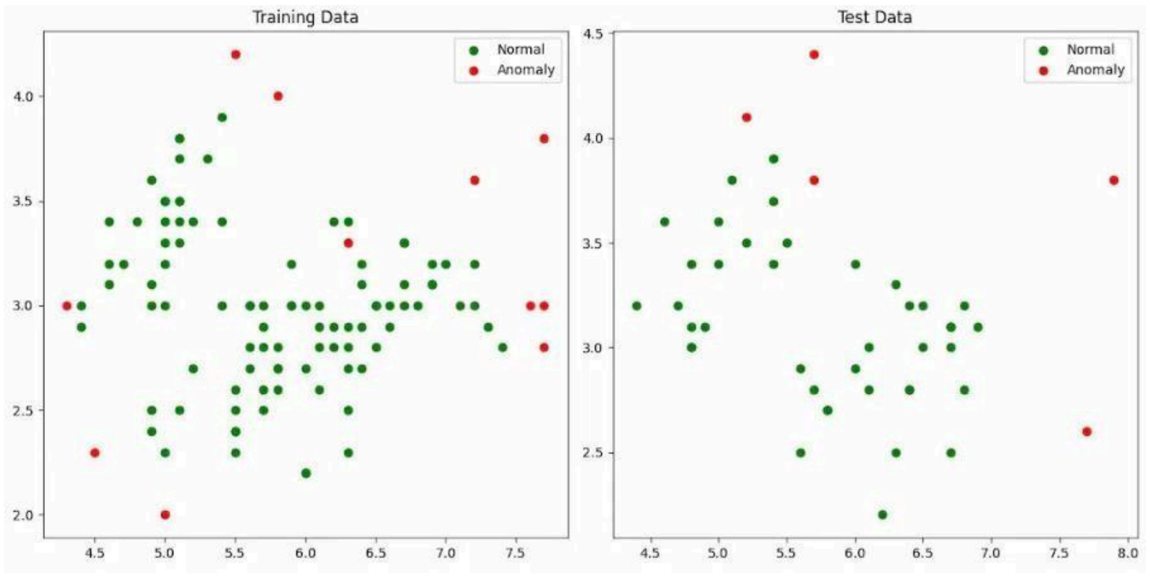
Output-

```
[ 1  1  1  1 -1  1 -1  1  1 -1  1  1  1  1 -1  1  1  1  1  1  1  1
-1  1
  1  1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1
  1 -1  1  1 -1  1  1  1  1  1  1  1  1  1 -1  1  1  1  1  1  1  1
1  1
  1  1  1  1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1
  1  1  1  1  1  1 -1  1  1]
[ 1 -1 -1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1
 -1  1  1  1  1  1  1 -1  1  1  1  1  1  1  1 -1  1  1  1 -1  1]
```

Step 5- Visualization

```
def create_scatter_plots(X1, y1, title1, X2, y2, title2):
    fig, axes = plt.subplots(1, 2, figsize=(12, 6))

    # Scatter plot for the first set of data
    axes[0].scatter(X1[y1==1, 0], X1[y1==1, 1], color='green',
label='Normal')
    axes[0].scatter(X1[y1==-1, 0], X1[y1==-1, 1], color='red',
label='Anomaly')
    axes[0].set_title(title1)
    axes[0].legend()

    # Scatter plot for the second set of data
    axes[1].scatter(X2[y2==1, 0], X2[y2==1, 1], color='green',
label='Normal')
    axes[1].scatter(X2[y2==-1, 0], X2[y2==-1, 1], color='red',
label='Anomaly')
    axes[1].set_title(title2)
    axes[1].legend()

    plt.tight_layout()
    plt.show()

# scatter plots
create_scatter_plots(X_train, y_pred_train, 'Training Data', X_test,
y_pred_test, 'Test Data')
```

Final Output



CHAPTER – 3

Advantages and Applications

Advantages

Effective for Unlabelled Data: Isolation Forests do not require labeled data (normal vs. anomaly) for training, making them suitable for scenarios where labelled data is scarce.

Efficient for High-Dimensional Data: The algorithm scales well with high-dimensional data sets, which can be challenging for other anomaly detection methods.

Robust to Noise: Isolation Forests are relatively insensitive to noise and outliers within the data, making them reliable for real-world data sets.

Versatility: Isolation Forest can be applied to various types of network traffic data, including raw packet data, NetFlow data, or application-level protocol data. It's adaptable to different network environments and
configurations.

Applications

Intrusion Detection: Detecting unauthorized access attempts, malware activity, or other security breaches in a network.

Network Monitoring: Identifying unusual patterns or spikes in network traffic that could indicate performance issues, network congestion, or infrastructure problems.

Forensic Analysis: Investigating past incidents by analysing historical network traffic data for anomalous behaviour.

Baseline Establishment: Building a baseline of normal network behaviour to compare against, helping to identify deviations and potential threats

CHAPTER– 4

Conclusions

Leveraging Isolation Forest for anomaly detection in network traffic presents a compelling solution with several advantages and diverse applications. Its efficiency, scalability, and ability to handle high-dimensional data make it particularly well-suited for the complex and dynamic nature of network environments. The algorithm's computational efficiency enables it to process large volumes of network traffic data quickly, essential for real-time anomaly detection where timely responses are critical. Isolation Forest's scalability ensures it can handle increasing network data complexity, being parallelizable and deployable across distributed computing environments. A standout feature is its unsupervised nature, meaning it does not require labeled data for training, which is especially beneficial in network security where labeled anomalies are rare and expensive to obtain. By detecting rare events and anomalies without labeled data, Isolation Forest adapts to new patterns, providing insights into potential security threats, performance issues, and abnormal behavior in real-time.

Its ability to handle high-dimensional data effectively makes it powerful for various applications. In intrusion detection, it identifies unusual network traffic patterns indicating threats like malware infections, unauthorized access attempts, and DDoS attacks. Its real-time detection capabilities allow prompt responses, minimizing attack impacts. In network monitoring, Isolation Forest detects performance anomalies such as latency issues and bandwidth bottlenecks, maintaining optimal network performance. In forensic analysis, it traces the root cause of anomalies, providing insights into security breaches or system malfunctions. By enabling organizations to proactively identify and mitigate risks, establish baseline behavior, and enhance security posture, Isolation Forest plays a crucial role in network security and performance. As network threats evolve, it remains a versatile tool for addressing emerging challenges, ensuring robust and efficient anomaly detection against a wide range of network issues.

References

1. **Isolation Forest Algorithm**:Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining* (pp. 413-422). IEEE.
2. **Scikit-Learn Documentation**:Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.
3. **Iris Dataset**:Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7*(2), 179-188.
4. **Anomaly Detection in Network Traffic**:Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Computing Surveys (CSUR), 41*(3), 15.
5. **Machine Learning for Network Security**:Sommer, R., & Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *IEEE Symposium on Security and Privacy*, 305-316.
6. **Visualization Techniques**:Ware, C. (2012). Information Visualization: Perception for Design. *Elsevier*.
7. **Python for Data Analysis**:McKinney, W. (2012). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. *O'Reilly Media, Inc.*