Anomaly Detection using Isolation Forest (Group Number : 7)

Name: Aakash Borse

Scholar No : 22U03028 (IT)

Organisation: Indian Institute of Information Technology, Bhopal.

Place: Bhopal, India. Email:

[borseaakash082004@gmail.com](mailto:borseaakash082004@gmail.com)

Name: Rahul Raj

Scholar No :22U03027 (IT)

Organisation: Indian Institute of Information Technology, Bhopal.

Place: Bhopal, India. Email:

[rjrahul02@gmail.c](mailto:rjrahul02@gmail.c) om

Name: Rishab Mavi

Scholar No : 22U02012 (IT)

Organisation: Indian Institute of Information Technology, Bhopal.

Place: Bhopal, India.

Email: [mavirishab01@gmail.com](mailto:mavirishab01@gmail.com)

I. Introduction

The approach uses machine learning algorithms to detect anomalies in social media platforms based on user behavior patterns such as the frequency of posts, sentiment, and engagement metrics. The process can be broken down into the following steps:

1. **Data Preprocessing**: Collect and clean the data by preparing a dataset with relevant features such as the frequency of posts, sentiment, and user engagement metrics (likes, comments, followers).
2. **Feature Engineering**: Extract key features from the data, including sentiment analysis, posting patterns, and engagement metrics, which are critical for detecting unusual behavior.
3. **Model Selection**: Choose a suitable unsupervised machine learning model like **Isolation Forest** or **DBSCAN** for anomaly detection, which are effective at

5.

**Evaluation**: Evaluate the model’s performance using accuracy metrics like precision and recall to ensure the model effectively detects anomalies.

# Example Code for Anomaly Detection using Isolation Forest

We will implement anomaly detection using the **Isolation Forest** algorithm, a common technique for detecting anomalies in high-dimensional datasets.

Step 1: Import Libraries

Step 2: Create a Simulated Social Media Dataset Step 3: Visualize the Data

Step 4: Apply Isolation Forest for Anomaly Detection

Step 5: Visualize the Anomalies

identifying outliers in data without

requiring labeled examples.

1. **Model Training**: Train the model on the prepared dataset, and the algorithm will learn to distinguish between normal and

abnormal behavior based on the input features.

Meaning of Each Graph

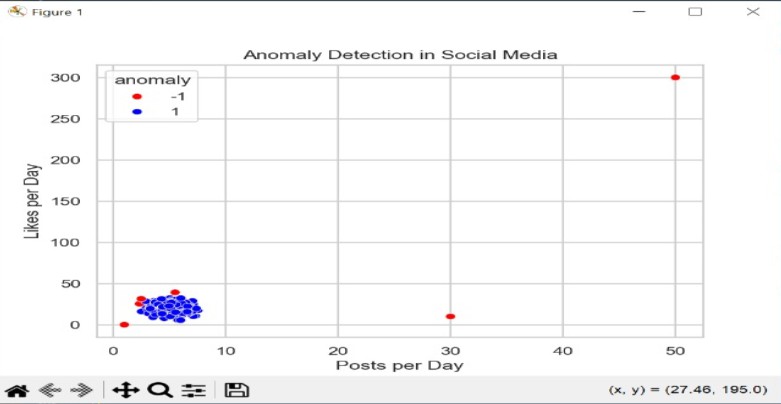
* 1. Pairwise Feature Visualization (Pairplot):

Purpose: Shows relationships between pairs of features and allows us to identify distributions and potential outliers.

Interpretation:

Normal users tend to cluster, while anomalies (e.g., bots or fake accounts) are scattered.

For example, the "Posts per day" vs "Likes per day" scatter plot may show normal behavior as a consistent relationship, with anomalies deviating.



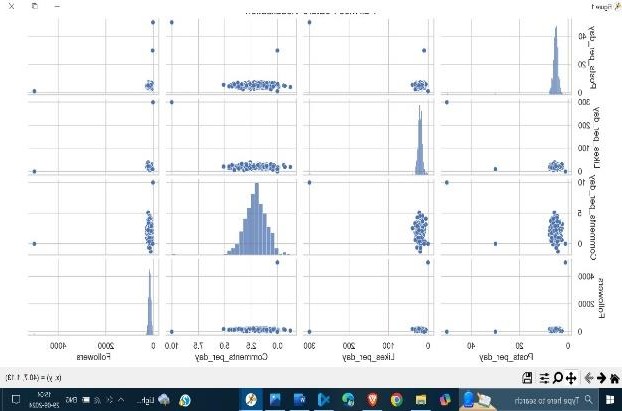
* 1. Scatter Plots for Different Feature Pairs**:**

Purpose: Helps in visually distinguishing normal behavior from anomalous behavior based on two features.

Interpretation:

Posts vs Likes: Anomalies like bots or fake accounts may show high likes but low posts or vice versa.

Comments vs Followers: Anomalies may show few followers with many comments, or a very large number of followers with no engagement.



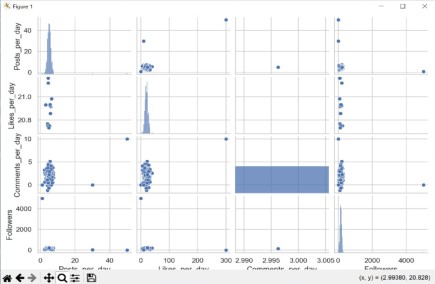
* 1. **Feature Distribution Histograms:**

**Purpose: Shows the distribution of each feature across the dataset.**

**Interpretation:**

Anomalies often fall outside the normal distribution of posts, likes, comments, and followers.

For instance, most users might have 3-7 posts per day, but anomalies might have outlier values like 50 posts per day.



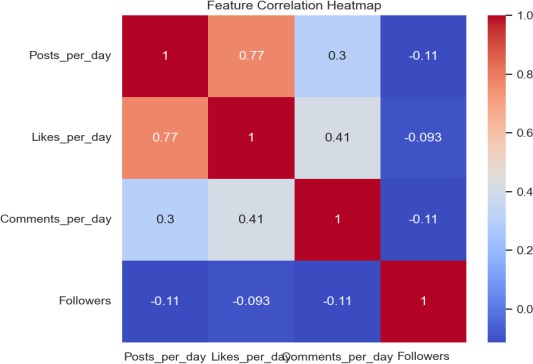
# Correlation Heatmap:

**Purpose: Shows the correlation between features.**

Interpretation:

Positive correlations (near 1) show that features tend to increase together (e.g., more posts lead to more likes).

Negative or low correlations may indicate abnormal behavior, such as many posts with few likes or comments.



Approach to Anomaly Detection

1. Dataset Creation:

Simulated normal behavior is generated with features such as Posts\_per\_day, Likes\_per\_day, Comments\_per\_day, and Followers.

Anomalous data points representing suspicious activity (e.g., bots, fake accounts) are injected to simulate abnormal user behavior.

1. Model Selection:

The Isolation Forest algorithm is selected for its efficiency in detecting outliers in high-dimensional datasets. It works by isolating points that differ significantly from the majority of the data.

1. Visualization:

Visualizations like pair plots, scatter plots, and histograms are used to confirm the effectiveness of the model in distinguishing between normal and anomalous behavior.

The correlation heatmap helps understand how features relate to each other and how anomalies disrupt normal patterns.

Algorithm:

Isolation Forest (iForest) is based on the concept of "isolation" rather than fitting a profile or distribution of normal data. This makes it distinct from other anomaly detection techniques like One- Class SVM or density-based methods, which often assume a structure or distribution of normal data and identify deviations. The isolation-based approach works especially well for high- dimensional data where calculating distances or densities can become computationally challenging and unreliable.

# Working Principle

Isolation Forest constructs multiple trees by recursively dividing data points across random splits. Each tree partitions the data until it isolates individual points. Since anomalies are generally sparse and lie farther from clusters, they tend to be isolated faster—resulting in shorter path lengths within the trees. For normal points, which are in dense regions, more splits are required to reach isolation, leading to longer path lengths.

# Scoring Anomalies

For each data point, Isolation Forest averages the path lengths across all trees. These path lengths are then normalized to produce an anomaly score. A shorter average path length (closer to the root of the tree) indicates a likely anomaly, while a longer path (deeper in the tree) suggests a normal data point.

The scores are typically in a range from 0 to 1, with values close to 1 indicating strong anomalies.

# Advantages

**Scalability**: The algorithm is highly scalable, with linear time complexity, and can handle large datasets due to its reliance on simple partitioning rather than distance or density calculations.

**High-dimensional Suitability**: Isolation Forest does not suffer from the curse of dimensionality as it does not compute distances explicitly. Instead, it uses random splits, making it robust in high- dimensional spaces.

**Versatility**: Isolation Forest can be applied across varied types of data distributions and is model- agnostic, so it does not assume any particular distribution of normal data.

# Use Cases

Isolation Forest has been applied successfully in applications like fraud detection, network security, intrusion detection, and quality assurance. Its efficiency and ability to handle both high- dimensional and large datasets make it suitable for real-time and large-scale applications, where rapid identification of anomalies is critical.

In summary, Isolation Forest is a unique, efficient algorithm that leverages the randomness of isolation processes to detect anomalies effectively. By taking advantage of random splits to isolate points, it excels in environments where other methods struggle, especially with high-dimensional or large data.

# Path Length Calculation

For a given point xxx, the average path length h(x) across all trees is calculated. This path length refers to the number of splits required to isolate xxx in each tree. Since anomalies are isolated faster, they have shorter path lengths.

To compute the anomaly score, Isolation Forest normalizes h(x) by comparing it to the average path length for an isolation tree with n samples. This normalization step is crucial, as it adjusts the score to account for variations in data size.

# Code:

*# Importing Libraries* import numpy as np import pandas as pd

from sklearn.ensemble import IsolationForest import matplotlib.pyplot as plt

import seaborn as sns

*# Set style for seaborn plots*

sns.set(style="whitegrid")

*# Step 1: Create a Simulated Social Media Dataset*

*# Simulate normal user behavior data*

np.random.seed(42) n\_samples = 300

*# Features: [Posts per day, Likes per day, Comments per day, Followers]*

normal\_data = np.random.normal(loc=[5, 20, 2,

150], scale=[1, 5, 1, 50], size=(n\_samples, 4))

*# Inject anomalous data (representing bots or fake accounts)*

anomalous\_data = np.array([[50, 300, 10, 10], *#*

*Bot-like behavior*

[1, 0, 0, 5000], *# Fake*

*influencer account*

*# Step 2: Visualize the Data (Pairplot)*

sns.pairplot(df) plt.show()

*# Step 3: Apply Isolation Forest for Anomaly Detection*

*# Define the Isolation Forest model*

model = IsolationForest(n\_estimators=100, contamination=0.02, random\_state=42)

*# Fit the model to data*

model.fit(df)

*# Predict anomalies (-1 for anomaly, 1 for normal)*

df['anomaly'] = model.predict(df)

*# Extract anomalies*

anomalies = df[df['anomaly'] == -1]

*# Display detected anomalies* print("Detected Anomalies:") print(anomalies)

*# Step 4: Visualize the Anomalies*

sns.scatterplot(data=df, x='Posts\_per\_day', y='Likes\_per\_day', hue='anomaly', palette={1: 'blue', -1: 'red'})

*activity*

[30, 10, 0, 15]]) *# Suspicious*

plt.title("Anomaly Detection in Social Media") plt.xlabel("Posts per Day")

*# Combine normal and anomalous data*

X = np.vstack([normal\_data, anomalous\_data])

*# Convert to DataFrame for visualization*

df = pd.DataFrame(X, columns=['Posts\_per\_day', 'Likes\_per\_day', 'Comments\_per\_day', 'Followers'])

plt.ylabel("Likes per Day") plt.show()

*# Additional Analysis: Correlation Heatmap*

plt.figure(figsize=(8, 6))

sns.heatmap(df.drop(columns='anomaly').corr(), annot=True, cmap='coolwarm')

plt.title("Feature Correlation Heatmap") plt.show()

**Conclusion**

The Isolation Forest model is effective at identifying users who exhibit outlier behavior, such as bots or fake accounts, by analyzing their activity patterns across multiple features.

Visualization tools like scatter plots,histograms, and correlation heatmaps aid in understanding the patterns in the data and verifying the results of the model.

This approach can be extended to larger datasets and more complex behavior patterns to improve the detection of anomalies across various domains such as social media platforms, fraud detection systems, and more.