Federated K-Nearest Neighbors (KNN) Search System

Hala Medhat 49-4186

June 6, 2024

1 Project Description

This project involves developing a federated K-Nearest Neighbors (KNN) search system, with a strong focus on privacy, security, and performance in a distributed setting. The system is designed to explore various result aggregation methods, apply the STRIDE methodology for threat modeling, and evaluate the impact of different distance metrics on the system's performance. Additionally, the project aims to simulate attacks identified during the threat modeling process and propose effective mitigation strategies. We have taken inspiration in our project from these papers [1] and [2]

2 Objectives

The primary objective of this project is to develop a decentralized KNN system where each node, using its local dataset, contributes to classifying new input samples. This decentralized approach ensures that data remains localized, enhancing privacy and security. Another key objective is to investigate various result aggregation techniques to effectively combine the results from all nodes. This involves implementing multiple aggregation methods such as majority voting, minimum distance, weighted voting, and distance-based aggregation, and analyzing their performance.

The project also involves conducting a comprehensive STRIDE analysis to identify potential security threats within the federated system. This analysis helps in understanding and mitigating risks related to Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. Furthermore, the project evaluates the impact of different distance metrics on the performance and accuracy of the KNN algorithm, providing insights into how the choice of distance metric affects the overall system performance.

3 System Architecture

The federated KNN system consists of multiple nodes, each with its local dataset. These nodes collaboratively classify new input samples by sending their predictions to a central coordinator, which aggregates the results to produce a final prediction

3.1 Components

3.1.1 Nodes

Each KNN node handles local KNN training and prediction, listens for classification requests, processes them, and sends back the results

3.1.2 Aggregator

The aggregator manages sending requests to nodes, collecting responses, and applying various aggregation techniques.

3.1.3 Redis Server

Facilitates communication between nodes and the coordinator

A key challenge addressed in the implementation is the handling of multiple simultaneous requests. The aggregator uses unique request IDs to track responses and ensures that responses are aggregated correctly.

4 Implementation

4.1 Node Implementation

Each node uses scikit-learn to implement the KNN algorithm using different distance metric and Redis for communication. The node script subscribes to a request channel of name knn-name of distance metric used-request-channel-node number, processes the sample using its local KNN model, and publishes the top k predictions and distances to the response channel.

4.1.1 Distance Meterics

Euclidean Distance

Euclidean distance calculates the straight-line distance between two points, considering differences in both magnitude and direction. It's often used in scenarios where such differences are significant, like image recognition or clustering.

Manhattan Distance

Known as the city block or taxicab distance, Manhattan distance measures the distance between points along orthogonal axes. It's less affected by outliers compared to Euclidean distance and finds applications in route planning or network analysis.

Cosine Similarity

Unlike the previous metrics, cosine similarity doesn't consider magnitude but focuses solely on the direction of vectors. It's particularly useful in text mining, document clustering, or recommendation systems, where the magnitude of vectors is less relevant than their orientation in the feature space.

4.2 Coordinator Implementation

The coordinator broadcast a query to all of the nodes and listens to any published messages. When predictions arrived they are collected and aggregated using various techniques.

4.2.1 Result Aggregation Techniques

Majority Voting

In majority voting, each node's prediction is collected, and the prediction that appears most frequently is chosen as the final prediction.

Minimun Distance

In minimum distance, each node's ditances and predictions are collected, and the prediction that has least distance is chosen.

Weighted Voting

In weighted voting, each node's prediction is weighted based on its reliability or confidence level indicated by aggregator. The final prediction is determined by the weighted sum of predictions

Distance Based Aggrgation

In distance-based aggregation, predictions are weighted by the inverse of the sum of distances from the KNN model. Predictions with smaller distances (indicating closer neighbors) are given more weight

5 STRIDE Threat Modeling

5.1 STRIDE Analysis

In our analysis using the STRIDE methodology, we identified several potential security threats to the federated KNN search system, along with corresponding mitigation strategies

5.1.1 Spoofing

Spoofing refers to the possibility of nodes impersonating other legitimate nodes within the system and sending responses to the aggregator when a message is published To mitigate this threat, robust node authentication mechanisms are essential. By implementing strong authentication protocols, the system can verify the identity of each participating node and prevent unauthorized access.

5.1.2 Tampering

The threat of data tampering arises when malicious entities attempt to alter or modify data during transmission between nodes. It can also happen when a malicious node join network and modify its local data before fitting it to KNN model by label flipping for example. To counter this threat, data encryption techniques can be employed to secure data in transit. By encrypting data using strong cryptographic algorithms, the system ensures that any tampering attempts are detected and prevented, thus maintaining the integrity and confidentiality of the information exchanged.

5.1.3 Repudiation

Nodes may attempt to deny their involvement in sending certain data, leading to repudiation issues. To address this threat, robust logging mechanisms and non-repudiation mechanisms are crucial. By maintaining detailed logs of all communication and interaction activities, the system can establish an audit trail that holds nodes accountable for their actions. Non-repudiation mechanisms, such as digital signatures or timestamps, further strengthen the system's ability to prove the origin and authenticity of transmitted data, preventing nodes from disowning their actions.

5.1.4 Information Disclosure

The risk of information disclosure arises when sensitive data is exposed to unauthorized parties. To safeguard against this threat, the system employs data encryption and access control measures. By encrypting sensitive information before transmission and implementing strict access controls based on user roles and permissions, the system ensures that only authorized entities can access and manipulate data, thereby mitigating the risk of information leakage.

5.1.5 Denial of Service (DoS)

Denial of Service (DoS) attacks can disrupt the normal operation of the system by overwhelming nodes with excessive requests. To mitigate this threat, the system implements rate limiting and load balancing mechanisms. By imposing limits on the number of requests a node can handle within a given timeframe and distributing incoming traffic across multiple nodes using load balancers, the system ensures equitable resource allocation and prevents any single node from becoming a bottleneck susceptible to DoS attacks.

5.1.6 Elevation of Privilege

The threat of elevation of privilege occurs when unauthorized users gain higher privileges within the system, enabling them to access sensitive resources or execute privileged operations. To counter this threat, the system enforces strict access control and authentication mechanisms. By defining granular access policies based on user roles and implementing multi-factor authentication, the system verifies the identity and permissions of users before granting access to sensitive functionalities, thereby preventing unauthorized elevation of privilege attempts.

6 Attack Simulations and Mitigation

6.1 Attack Simulations

6.1.1 Spoofing

A spoofed node was implemented which subscribe to the channel of node 1. Whenever aggregator publish a query it capture the message and it also send false data back to the aggregator

6.1.2 Tampering

One of the nodes was simulated as a malicious node, this nodes flipped the labels of some data, where digits 2 it label become 1 and digit 1 its label become 2.

6.1.3 DoS

A DoS attack was simulated where an attacker kept sending 10000 requests to the aggregator to keep it busy. During this attack, it took any query around 19 minutes to get a response back.

6.2 Mitigations Implemented

6.2.1 Spoofing

For this type of attack we implemented authentication to countermeasure this attack, first we authenticated the Redis server so that not any one can join the network if they don't provide the secret password configured within Redis.Second, we implemented an encryption algorithm to encrypt data sent by aggregator to any of nodes so that it can't decrypt and know its contents.

6.2.2 Tampering

For Tampering we implemented 2 types of mitigation, First to mitigate against data tampering while in transit, we used a message integrity's checks. Where every message send to aggregator was signed using HMAC using a secret key, and when published back to aggregator, it will not be processed if it is not valid. The second mitigation technique countermeasure the fact of flipping the labels in dataset. So we implemented an anomaly detection algorithm using ML model which is Isolation Forest. So all the distances and predictions were concatenated and the model was fitted to give the nodes that are outliers. The outliers were given given a repudiation score, which is the weight used in Weight Voting aggregation technique indicating a decrease in confidence level. Whenever an anomaly is detected a flag will be raised to take the measure needed.

6.2.3 DoS Attack

For DoS attack we implemented a rate limiting mitigation of 10 requests per second. After implementing this the query was executed immediately.

7 Results

We test our implementation on a test dataset of size 360

7.1 Without any malicious nodes

In the absence of any malicious nodes, we evaluated the performance of different distance metrics. Cosine distance achieved the highest accuracy at 98.8%, followed by Euclidean distance with an accuracy of 98.3%, and Manhattan distance with an accuracy of 97.7%. It is noteworthy that all voting methods yielded the same accuracy, indicating that the aggregation technique did not significantly impact the performance in a benign environment

7.2 With label flipping tampering

7.2.1 Without mitigation

When simulating label flipping tampering without applying any mitigation strategies, we observed varying accuracies across different distance metrics and voting methods. The highest accuracies were achieved using cosine distance with majority voting at 98.3%, Manhattan distance with weighted voting at 97.5%, and Euclidean distance with majority voting at 98.1%. The minimum distance voting method consistently resulted in the lowest accuracies for all distance metrics, highlighting its vulnerability to tampering. This analysis suggests that majority voting is the most effective method for mitigating the impact of label flipping tampering, as it consistently outperformed other voting techniques in maintaining higher accuracy levels.

7.2.2 With Isolation Forest Mitigation

When applying the Isolation Forest mitigation technique, we observed some interesting changes in accuracy. The highest accuracy was achieved using the cosine distance metric with majority voting at 98.1%. However, for both distance-based and majority voting methods, the accuracies across all distance metrics were slightly lower compared to the scenarios without mitigation. This indicates that while the Isolation Forest technique may inadvertently remove some correct data, it still benefits the overall accuracy by filtering out potentially harmful data.

Notably, the accuracy for the minimum distance voting method improved drastically, from around 91% to approximately 97%. This suggests that the Isolation Forest was particularly effective in identifying and removing invalid data introduced by the attacker, which had previously led to misleading model predictions. Thus, while there might be a minor trade-off in accuracy for majority and distance-based voting due to the removal of some beneficial data, the overall improvement in the minimum distance voting highlights the effectiveness of Isolation Forest in enhancing the robustness of the system against tampering attacks.

8 Conclusion

In this project, we successfully developed a decentralized K-Nearest Neighbors (KNN) system, where each node contributes to classifying new input samples using its local dataset. We explored various result aggregation techniques and conducted a STRIDE analysis to identify potential security threats within the federated system and tried to implement several techniques for mitigaton. Our implementation demonstrated high accuracy across different distance metrics, with cosine distance consistently yielding the best performance.

Our findings highlight the importance of selecting appropriate distance metrics and aggregation methods to maintain high accuracy and robustness in federated learning environments. This project underscores the critical role of combining advanced machine learning techniques with thorough security analysis to develop resilient and accurate distributed systems.

References

[1] Z. Liu, L. Wang, and K. Chen, “Secure efficient federated knn for recommendation systems,” in *The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, pp. 1808–1819, Springer, 2020.

[2] X. Zhang, Q. Wang, C. Xu, Y. Peng, and J. Xu, “Fedknn: Secure federated k-nearest neighbor search,” *Proceedings of the ACM on Management of Data*, vol. 2, no. 1, pp. 1–26, 2024.