DNS cache poisoning attack

By:

Ziad Mahmoud Abdeljawad 2205021

Abdallah Mohammed Hegazy 2205228

Introduction:

DNS cache poisoning attack is an attack that attacker try to spoof a DNS resolver's cache and inject malicious DNS information that redirecting the user to a fraudulant website. The project involves detecting DNS anomalies using machine learning algorithms like: Random forest, Naive Bayes, and logistic regression. DNS anomalies like cache poisoning are detected using statistical features extracted from synthetic DNS log dataset.

Explore the Data:

synthetic DNS log dataset consists of 10000 enrties, 15 features, and a label column. The precentage of normal packets to malicious packets is 98% to 2%.

Data preprocessing:

First we drop the useless features like: DNS proxy name, unused data, log source, and empty field. Then, we determine the most important features that directly affected on the model accuracy by calculate the wight of features. Those features are: query volume, transaction ID, average TTL. Then, we encode some features to numeric features like: query type and response code. Finally, we split the data to 30% test data and 70% train data. We split the data before handling the imbalance to make the test set representative of real-world data.

Handling imbalance data:

To handle imbalance data we used smote function. Smote function focus on the minority and pick a random sample from it as a seed sample. Then, it uses K-NN algorithm to find the k closest sample from the seed sample. Then, it generates a Synthetic Sample by selecting a random neighbor and generate a new sample along the line segment between the seed and the neighbor.

Synthetic Sample =

Seed Sample+Random Fraction×(Neighbor Sample−Seed Sample)

It repeats that until balanced is achieved.

Data Modeling:

In that project we applied 3 models and determine the best one depending on their statistics.

- **Random forest:** A tree-based ensemble learning algorithm that builds multiple decision trees during training and combines their outputs to improve classification performance.

It is particularly useful for feature importance analysis, providing insights into which features contribute most to anomaly detection.

- **Naive Bayes:** A probabilistic classification model based on Bayes' Theorem, assuming independence between features.

It works well with smaller datasets and handles imbalanced data efficiently by calculating probabilities directly.

- **logistic regression:** A linear model that predicts probabilities for binary classification problems.

It is well-suited for datasets where the relationship between features and the target is approximately linear.

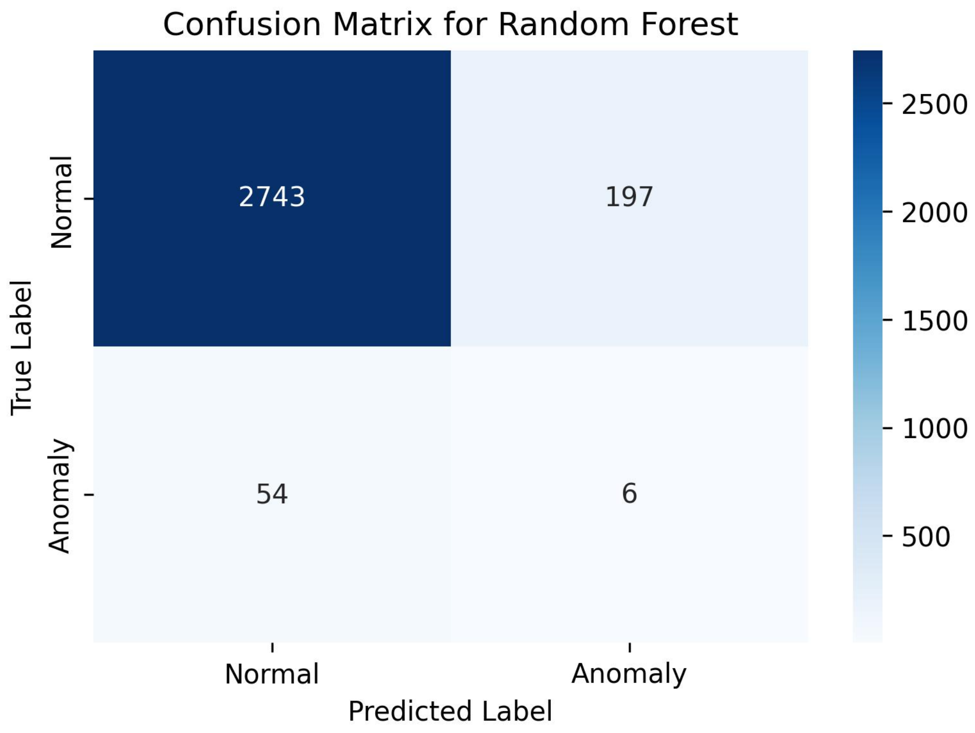
The models are evaluated using the following metrics:

1- **Accuracy**: Measures the percentage of correctly classified samples.
2- **Precision**: Focuses on the proportion of true positive predictions among all positive predictions, indicating how many flagged anomalies are actually anomalies.
3- **Recall**: Measures the proportion of actual anomalies that are correctly identified.
4- **F1-Score**: The harmonic mean of precision and recall, balancing the trade-off between the two.
5- **Confusion Matrix**: A visualization tool to assess the distribution of true positive, true negative, false positive, and false negative predictions.

Modeling Results:

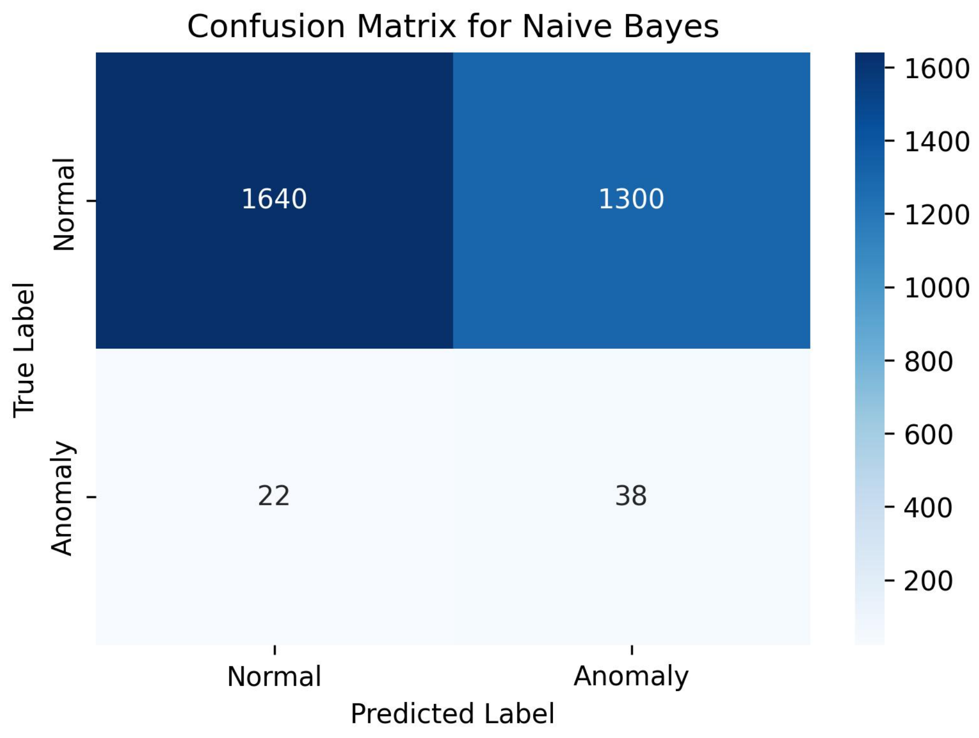
1- Random forest:

Random forest represented accuracy 91.63% and f1-score 0.0456



2- Naive Bayes:

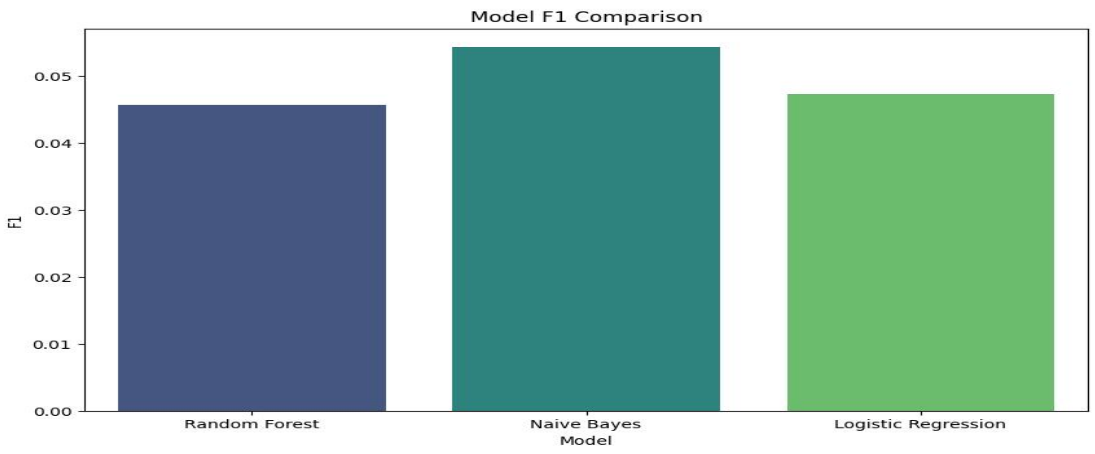
Naive Bayes model represented accuracy 55.93% and f1-score 0.0544.



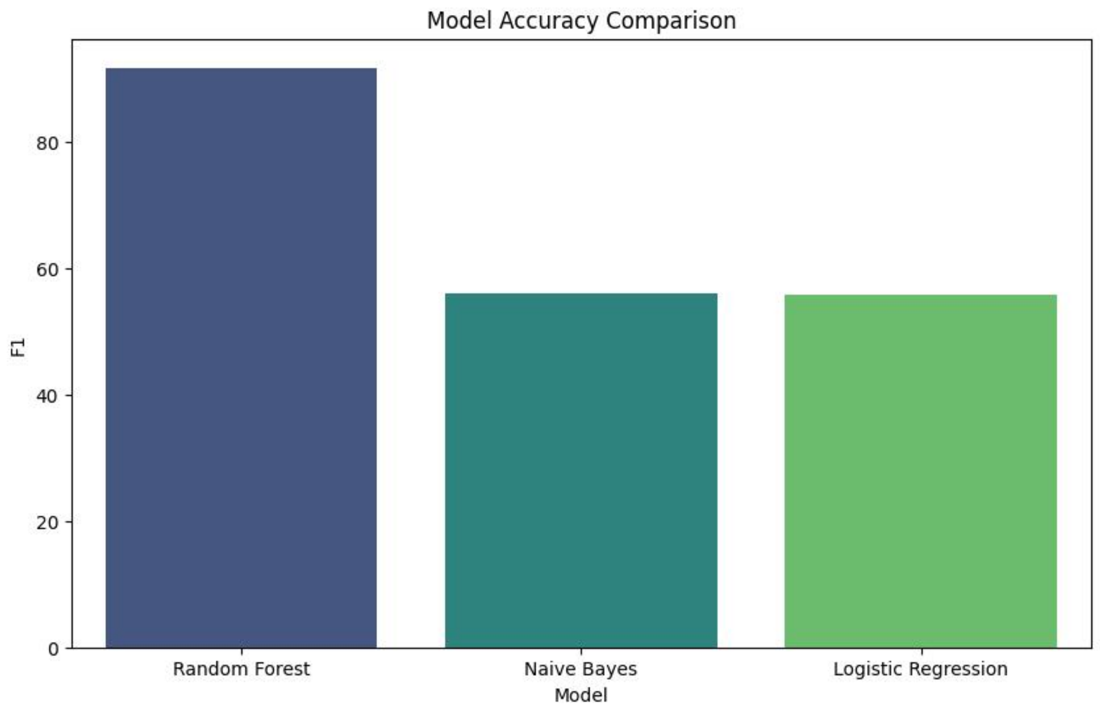
3- Logistic Regression:

Logistic Regression model represented accuracy 55.73% and f1-score 0.0473.

So, depending on F1-score the best model is Naive Bayes model with score 0.0544.



But, depending on the accuracy the best model is Random Forest model with accuracy 91.63%


Model Accuracy Comparison
F1
Random Forest
Naive Bayes
Logistic Regression
Model