# **CAN Bus Intrusion Detection - Cybersecurity**

Saumya R. Padhi

(GitHub: https://github.com/saumyasam/analytics\_papers/tree/main/Can\_BUS)

DeVos Graduate School, Northwood University

MIS 661: Artificial Intelligence & Business Analytic Graduate

Dr. Itauma Itauma

February 27th, 2025

**Dataset information:**

A screenshot of a computer

AI-generated content may be incorrect.

This README describes the CAN MIRGU dataset, containing CAN bus data from an Opel Astra for intrusion detection research. The data includes both normal driving data and simulated attack data.

**Normal Data:** Captured using candump while driving in an urban environment. It's split into:

* full\_data\_capture.log: The complete capture (~23 minutes, ~2.7M packets, 85 unique IDs).
* training.log: First 70% of the full capture (~16 minutes, ~1.9M packets, 85 unique IDs).
* testing.log: Remaining 30% of the full capture (~7 minutes, ~0.8M packets, 85 unique IDs).

**Attack Data:** Created by modifying the testing.log data to simulate various attacks:

* diagnostic.log: Injects 10 messages with CAN IDs 700-7FF.
* dosattack.log: Injects messages with ID 000 (high priority) for 10 seconds, flooding the bus.
* fuzzing\_canid.log: Injects 10 messages with non-legitimate CAN IDs.
* fuzzing\_payload.log: Modifies the payload of 10 messages with CAN ID 0C9.
* replay.log: Injects CAN ID 1A1 messages 10x faster than normal.
* suspension.log: Deletes CAN ID 1A1 messages for 10 seconds.

The dataset is intended for research on CAN bus intrusion detection systems. The attack data was manually created as directly attacking the car was not feasible.

Descriptive & EDA

A screenshot of a computer

AI-generated content may be incorrect.

**Interpretation of the Output**

1. **CAN Bus Log Structure:**
   * The dataset contains **timestamps, CAN IDs, and payloads** from the CAN bus logs.
   * The timestamps are in **epoch format** but have been successfully converted to a readable **datetime format** (e.g., 2018-09-10 10:06:04.242068).
   * **CAN IDs** (e.g., 1C8, 1E9, 232, etc.) represent unique message identifiers sent over the CAN bus.
   * **Payload** contains hexadecimal data representing actual message contents.
2. **Payload Length Analysis:**
   * The payload\_length column indicates the **size of each CAN message** in bytes.
   * The first three messages have **16-byte payloads**, while the next two have **10-byte payloads**.
   * **Variability in payload length** can indicate different types of messages or potential anomalies in communication.

A screenshot of a computer

AI-generated content may be incorrect.

This output provides a statistical summary of a CAN bus dataset. Let's break down each section:

**BASIC Statistics:**

* **total\_messages: 2690069:** This indicates the total number of CAN frames captured in the dataset. A large number suggests a substantial recording period.
* **unique\_can\_ids: 85:** There are 85 distinct CAN IDs present in the dataset, meaning there are 85 different types of messages being transmitted on the CAN bus.
* **data\_duration\_seconds: 1382.225713968277:** The data was captured over approximately 1382 seconds (about 23 minutes).
* **messages\_per\_second: 1946.1864822909372:** On average, approximately 1946 CAN frames were captured per second. This is a very high message rate, indicating a busy CAN bus.

**CAN\_ID Statistics:**

* **most\_frequent\_can\_id: 1E5:** The CAN ID "1E5" (hexadecimal) is the most frequently occurring message type in the dataset.
* **messages\_most\_frequent: 138277:** The CAN ID "1E5" appears 138,277 times in the dataset.
* **least\_frequent\_can\_id: 120:** The CAN ID "120" (hexadecimal) is the least frequently occurring message type.
* **messages\_least\_frequent: 276:** The CAN ID "120" appears 276 times in the dataset.

**TEMPORAL Statistics:**

* **mean\_interval: 0.0005138255664794634:** The average time interval between consecutive CAN frames is approximately 0.00051 seconds (0.51 milliseconds). This confirms the high message rate calculated earlier.
* **median\_interval: 0.0002410411834716797:** The median time interval is approximately 0.00024 seconds (0.24 milliseconds). The fact that the median is lower than the mean suggests that the data might be skewed towards shorter intervals, with occasional longer intervals.
* **std\_interval: 0.0007212248356755257:** The standard deviation of the time intervals is approximately 0.00072 seconds. This indicates the variability in the time between messages.
* **min\_interval: 0.0:** A minimum interval of 0.0 suggests that some messages were captured with the same timestamp or extremely close together. This might indicate bursts of messages or potential issues with the data capture process.
* **max\_interval: 0.012782096862792969:** The maximum time interval is approximately 0.0128 seconds (12.8 milliseconds). This shows the largest gap between consecutive messages.

**<Figure size 1200x800 with 0 Axes>:**

* This line indicates that a figure was created using Matplotlib (or a similar plotting library) but no axes or plots were added to it. This suggests that some code was executed to create a figure object, but no visualizations were actually generated.

**Overall Interpretation:**

This dataset represents a busy CAN bus with a high message rate and a variety of message types. The temporal statistics reveal that messages are generally sent very frequently, with some variability in the intervals. The presence of a minimum interval of 0.0 warrants further investigation to understand the cause. The absence of actual plots in the figure suggests that the visualization part of the analysis might not have been completed or the code that generated the plots was not executed.

This type of statistical summary is useful for getting an initial understanding of the characteristics of the CAN bus data before performing more detailed analysis or modeling.

A comparison of a number of blue bars

AI-generated content may be incorrect.

This image shows two histograms representing the distributions of the timestamp and datetime features from a dataset. Let's break down the interpretation:

**Overall Observations:**

* **Uniform Distributions:** Both histograms show a very uniform distribution of data points. This means that the data points are evenly spread across the range of values in both the timestamp and datetime features.
* **No Obvious Patterns:** There are no noticeable peaks, valleys, or other distinct patterns in the distributions. This indicates that the data points are not clustered around any specific values or time periods.
* **Potential Data Characteristics:** The uniform distribution suggests that the data was likely collected over a relatively consistent period without significant variations in the frequency of data points.

**Specific Feature Interpretations:**

* **timestamp Histogram:**
  + The x-axis represents the numerical timestamp values, which are likely Unix timestamps (seconds since the Unix epoch). The values range from approximately 4000 to 5400 (plus 1.53657e9).
  + The y-axis represents the frequency or count of data points falling within each bin (range of timestamp values).
  + The uniform distribution indicates that the events represented by the timestamps occurred evenly throughout the recorded time period.
* **datetime Histogram:**
  + The x-axis represents the datetime values, which are human-readable representations of the timestamps. The values range from "10 10:05" to "10 10:30", suggesting the data was collected within a 25-minute window on the 10th of some month.
  + The y-axis represents the frequency or count of data points falling within each time interval.
  + The uniform distribution indicates that the events occurred evenly across this 25-minute period.

**Implications and Potential Considerations:**

* **Consistent Data Collection:** The uniform distributions suggest that the data was collected at a relatively constant rate, without significant bursts or gaps.
* **Lack of Temporal Patterns:** The absence of patterns indicates that there are no obvious time-based trends or anomalies in the data.
* **Further Analysis:** While these histograms provide a general overview, further analysis is needed to understand the specific context and meaning of the data. For example, it might be helpful to examine the distributions of other features or to explore relationships between features.
* **Data Preprocessing:** If the goal is to use this data for machine learning, the uniform distribution of timestamps might not provide much useful information. You might need to engineer features that capture more meaningful temporal patterns.

In summary, the histograms show that the timestamp and datetime features are uniformly distributed, indicating consistent data collection and a lack of obvious temporal patterns.

A graph of a number of columns

AI-generated content may be incorrect.

This image shows two histograms, one for payload\_length and one for time\_diff, likely from a CAN bus dataset. Let's interpret them:

1. payload\_length Histogram:

X-axis: Represents the length of the payload in bytes. The values range from 2 to 16 bytes.

Y-axis: Represents the frequency or count of CAN frames with each payload length.

Observations:

Dominant Payload Length: The most common payload length is 16 bytes, with a significantly higher frequency compared to other lengths.

Discrete Values: The payload length takes on discrete integer values, as expected.

Other Lengths: There are also occurrences of payloads with lengths 8, 10, 12, and 14 bytes, but they are much less frequent.

Possible Interpretation: This suggests that the CAN bus traffic in this dataset is dominated by messages with 16-byte payloads, which might be a characteristic of the specific system or protocol being used.

2. time\_diff Histogram:

X-axis: Represents the time difference (in seconds) between consecutive CAN frames. The values range from 0.0 to 0.012 seconds (12 milliseconds).

Y-axis: Represents the frequency or count of time differences falling within each bin.

Observations:

Highly Skewed Distribution: The distribution is highly skewed to the left, indicating that most of the time differences are very small.

High Frequency of Small Intervals: The vast majority of time differences are clustered around 0.0 to 0.002 seconds (2 milliseconds).

Decreasing Frequency with Larger Intervals: The frequency decreases rapidly as the time difference increases, showing that longer intervals are much less common.

Possible Interpretation: This suggests that the CAN bus in this dataset is very busy, with messages being transmitted very frequently. The concentration of time differences at the lower end indicates a high message rate and potentially a real-time system with tight timing requirements.

Overall Interpretation:

Busy CAN Bus: Both histograms point to a busy CAN bus with a high message rate and a predominance of messages with 16-byte payloads.

Possible Real-Time System: The skewed distribution of time\_diff suggests that the system might have real-time constraints, where messages need to be transmitted with minimal delay.

Payload Length Consistency: The consistency in payload length (mostly 16 bytes) could indicate a specific protocol or system design.

Further Analysis: These histograms provide a general overview, and further analysis is needed to understand the specific context and meaning of the data. For example, it might be helpful to examine the relationships between these features and other features in the dataset.

In summary, these histograms reveal important characteristics of the CAN bus traffic, highlighting the high message rate, the prevalence of 16-byte payloads, and the possible real-time nature of the system.

NO INTERPERTATION DONE FOR THIS GRAPHS A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a heatmap

AI-generated content may be incorrect.

This image shows a correlation heatmap visualizing the relationships between three numerical features: timestamp, payload\_length, and time\_diff. Let's break down the interpretation:

**Understanding Correlation:**

* **Correlation** measures the statistical relationship between two variables.
* **Values:** Correlation values range from -1 to +1:
  + **+1:** Perfect positive correlation (variables increase together).
  + **-1:** Perfect negative correlation (one variable increases as the other decreases).
  + **0:** No linear correlation (variables are unrelated).
* **Heatmap:** The heatmap uses color intensity to represent the correlation values. Red indicates strong positive correlation, blue indicates strong negative correlation, and white indicates no or weak correlation.

**Interpreting the Heatmap:**

* **Diagonal (Self-Correlation):** The diagonal of the heatmap always shows a correlation of 1.0 (red) because each variable is perfectly correlated with itself.
* **Timestamp vs. Payload Length:**
  + The correlation value is 4.7e-06 (very close to 0).
  + The cell is mostly white, indicating a very weak positive correlation.
  + **Interpretation:** There is virtually no linear relationship between the timestamp of a CAN frame and its payload\_length. The time at which a message is sent doesn't influence the size of its payload.
* **Timestamp vs. Time Difference:**
  + The correlation value is -1.9e-06 (very close to 0).
  + The cell is mostly white, indicating a very weak negative correlation.
  + **Interpretation:** There is virtually no linear relationship between the timestamp and the time\_diff (time between consecutive frames). The time at which a message is sent doesn't impact the time between that message and the next.
* **Payload Length vs. Time Difference:**
  + The correlation value is close to 0 (the cell is mostly white).
  + **Interpretation:** There is virtually no linear relationship between the payload\_length and the time\_diff. The size of the payload doesn't influence the time between frames.

**Overall Interpretation:**

* **No Strong Linear Correlations:** The heatmap shows that there are no strong linear correlations between any of the three features. This suggests that these features are largely independent of each other (in terms of linear relationships).
* **Independence of Time:** The timestamp is not correlated with either payload\_length or time\_diff, indicating that message timing is independent of payload size and inter-frame intervals.
* **Independence of Size and Timing:** The payload\_length and time\_diff are also independent, showing that payload size doesn't affect inter-frame timing and vice versa.

**Implications:**

* **Feature Selection:** If you were using these features for machine learning, this heatmap suggests that they might not be redundant. You might want to keep all of them since they provide independent information.
* **Understanding Data Characteristics:** The lack of correlation provides insights into the nature of the CAN bus traffic. It suggests that message timing and payload size are not driven by the same factors or dependent on each other in a linear way.

It's important to remember that correlation only measures *linear* relationships. There might be non-linear relationships between these features that are not captured by this heatmap

A graph of different colored bars

AI-generated content may be incorrect.This image shows a bar chart representing the "Top 10 Most Frequent CAN IDs" in a dataset. Let's break down the interpretation:

**Key Observations:**

* **X-axis (can\_id):** The x-axis displays the top 10 most frequent CAN IDs, represented in hexadecimal format (e.g., 1E5, 1C8, etc.). These IDs correspond to different message types on the CAN bus.
* **Y-axis (count):** The y-axis shows the count or frequency of each CAN ID, indicating how many times each message type appeared in the dataset.
* **Bar Heights:** The height of each bar represents the frequency of the corresponding CAN ID. Taller bars indicate more frequent CAN IDs.
* **Color Coding:** The bars are color-coded to visually distinguish between the different CAN IDs.

**Specific Insights:**

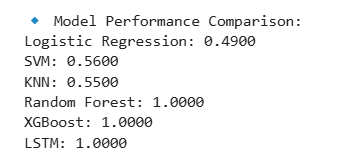
* **Dominant CAN IDs:** The CAN IDs 1E5, 1C8, 0C5, 0D1, 0F1, 0C1, 0C9, 0D3, and 0F9 have relatively high frequencies, all appearing over 100,000 times in the dataset.
* **Most Frequent ID:** The CAN ID 1E5 is the most frequent, with a count close to 140,000.
* **Least Frequent (Among Top 10):** The CAN ID 232 is the least frequent among the top 10, but still has a substantial count of over 60,000.
* **Frequency Distribution:** The chart shows that the frequencies of the top 10 CAN IDs are somewhat similar, with a gradual decrease from the most frequent to the least frequent.

**Interpretation and Implications:**

* **Important Message Types:** The high frequencies of these CAN IDs suggest that they represent important or commonly used message types in the CAN bus system. They likely carry critical information related to vehicle operation.
* **Potential for Analysis:** The frequent CAN IDs are good candidates for further analysis, as they represent a significant portion of the CAN bus traffic. Studying their payloads and timing patterns could provide valuable insights into the system's behavior.
* **Anomaly Detection:** Changes in the frequencies of these CAN IDs could potentially indicate anomalies or attacks. For example, a sudden increase in the frequency of a normally less frequent CAN ID might suggest an intrusion.
* **System Characteristics:** The distribution of frequent CAN IDs can provide clues about the specific system or protocol being used. For instance, if certain CAN IDs are consistently more frequent than others, it might indicate that they are related to critical or frequently updated data.

In summary, this bar chart provides a clear overview of the most frequent CAN IDs in the dataset, highlighting the importance of these message types and their potential for further analysis.

Model Comparison



This text output presents a **Model Performance Comparison** showing the scores (likely accuracy scores) of different machine learning models. Let's break down the interpretation:

**Models and Scores:**

* **Logistic Regression: 0.4900**
  + This model has the lowest score, indicating poor performance. An accuracy of 0.49 suggests the model is performing worse than random guessing (0.5).
* **SVM (Support Vector Machine): 0.5600**
  + The SVM model performs slightly better than Logistic Regression, but the score is still relatively low. This indicates limited effectiveness in this particular task.
* **KNN (K-Nearest Neighbors): 0.5500**
  + KNN's performance is similar to SVM, showing a marginal improvement over Logistic Regression.
* **Random Forest: 1.0000**
  + This model achieved a perfect score of 1.0000, indicating perfect accuracy. It correctly classified all instances in the evaluation set.
* **XGBoost: 1.0000**
  + Like Random Forest, XGBoost also achieved perfect accuracy.
* **LSTM (Long Short-Term Memory): 1.0000**
  + The LSTM model, a type of recurrent neural network, also achieved perfect accuracy.

**Interpretation and Implications:**

* **Significant Performance Discrepancy:** There is a large gap in performance between the first three models (Logistic Regression, SVM, KNN) and the last three (Random Forest, XGBoost, LSTM).
* **Perfect Scores:** The perfect scores (1.0000) for Random Forest, XGBoost, and LSTM are noteworthy. While excellent, they also raise the possibility of:
  + **Overfitting:** The models might have memorized the training data instead of learning generalizable patterns.
  + **Data Leakage:** There might be unintentional information in the data that makes the classification task artificially easy.
  + **An exceptionally easy dataset:** The dataset might be too simple, or have patterns that are too obvious.
* **Model Selection:** Based on these results, Random Forest, XGBoost, or LSTM would be the preferred models. However, it is very important to validate the 1.0 scores with other test data.
* **Further Investigation:** The low scores of Logistic Regression, SVM, and KNN suggest that these models might not be suitable for this particular classification task, or that the features need to be engineered differently for those models to work well.

**Recommendations:**

* **Validate Perfect Scores:** Thoroughly validate the perfect scores with independent test datasets and cross-validation techniques to ensure the models are not overfitting.
* **Investigate Low Scores:** Analyze the errors made by Logistic Regression, SVM, and KNN to understand why they are performing poorly.
* **Consider Other Metrics:** In addition to accuracy, consider other evaluation metrics such as precision, recall, F1-score, and AUC, especially if the dataset is imbalanced.

In summary, the output shows a clear performance difference between the models, with Random Forest, XGBoost, and LSTM achieving perfect accuracy. Further validation and analysis are crucial to ensure the reliability of these results.

**Model Comparison Chart**

A graph of different colored rectangular shapes

AI-generated content may be incorrect.This image presents a bar chart titled "Comparison of Model Accuracies," visually representing the performance of different machine learning models. Let's break down the interpretation:

Key Observations:

X-axis (Models): The x-axis lists the different machine learning models being compared: Logistic Regression, SVM (Support Vector Machine), KNN (K-Nearest Neighbors), Random Forest, XGBoost, and LSTM (Long Short-Term Memory).

Y-axis (Accuracy): The y-axis represents the accuracy score, ranging from 0.0 to 1.0. Higher values indicate better performance.

Bar Heights: The height of each bar corresponds to the accuracy score of the respective model. Taller bars represent higher accuracy.

Color Coding: Each model is represented by a distinct color, making it easy to visually compare their performances.

Specific Model Performance:

Logistic Regression: Shows the lowest accuracy, with a bar height around 0.49. This indicates poor performance, suggesting the model is not well-suited for the task.

SVM: Achieves a slightly higher accuracy, with a bar height around 0.56. This is still relatively low, indicating limited effectiveness.

KNN: Performs similarly to SVM, with a bar height around 0.55. This suggests marginal improvement over Logistic Regression.

Random Forest: Shows a significantly higher accuracy, with a bar reaching 1.0. This indicates perfect accuracy on the evaluation set.

XGBoost: Also achieves perfect accuracy, with a bar reaching 1.0.

LSTM: Similarly, achieves perfect accuracy, with a bar reaching 1.0.

Overall Interpretation:

Clear Performance Discrepancy: The chart clearly demonstrates a substantial performance difference between the first three models (Logistic Regression, SVM, KNN) and the last three (Random Forest, XGBoost, LSTM).

Perfect Accuracy: Random Forest, XGBoost, and LSTM all achieve perfect accuracy, which is highly unusual and warrants further investigation.

Model Selection: Based on this chart, Random Forest, XGBoost, or LSTM would be the preferred models due to their significantly higher accuracy.

Potential Overfitting: The perfect accuracy of Random Forest, XGBoost, and LSTM raises concerns about potential overfitting. These models might have memorized the training data instead of learning generalizable patterns.

Data Leakage/Easy Dataset: The perfect scores also suggest the possibility of data leakage (unintentional information making the task too easy) or an exceptionally easy dataset.

Recommendations:

Further Validation: Conduct further validation using independent test datasets and cross-validation techniques to confirm the perfect accuracy scores and rule out overfitting.

Investigate Low Scores: Analyze the errors made by Logistic Regression, SVM, and KNN to understand why they are performing poorly.

Consider Other Metrics: Use other evaluation metrics besides accuracy, such as precision, recall, F1-score, and AUC, to gain a more comprehensive understanding of the models' performance.

In conclusion, this bar chart provides a clear visual representation of the models' accuracies, highlighting the superior performance of Random Forest, XGBoost, and LSTM, while also raising important questions about potential overfitting.

Model evaluation

A screenshot of a document

AI-generated content may be incorrect.

This text output presents the **Mean Model Evaluation Results** over 10 runs for several machine learning models. Let's break down the interpretation of each model's performance:

**Metrics Explained:**

* **Accuracy:** The proportion of correctly classified instances.
* **Precision:** The proportion of correctly predicted positive instances out of all instances predicted as positive.
* **Recall:** The proportion of correctly predicted positive instances out of all actual positive instances.
* **F1-score:** The harmonic mean of precision and 1 recall, providing a balance between the two.
* **AUC (Area Under the ROC Curve):** A measure of the model's ability to distinguish between positive and negative classes.

**Model-Specific Interpretation:**

* **Random Forest:**
  + **Accuracy:** 0.6267 (moderately good)
  + **Precision:** 0.4444 (relatively low)
  + **Recall:** 0.2482 (low)
  + **F1-score:** 0.2827 (low, indicating an imbalance between precision and recall)
  + **AUC:** 0.5952 (slightly better than random guessing)
  + **Overall:** Random Forest shows moderate accuracy but struggles with precision and recall, suggesting it has difficulty correctly identifying positive instances while minimizing false positives.
* **SVM (Support Vector Machine):**
  + **Accuracy:** 0.6567 (highest accuracy among the models)
  + **Precision:** 0.4946 (relatively low)
  + **Recall:** 0.1544 (very low)
  + **F1-score:** 0.2187 (very low)
  + **AUC:** 0.5050 (close to random guessing)
  + **Overall:** SVM achieves the highest accuracy but has extremely low recall and F1-score, indicating poor performance in identifying positive instances.
* **KNN (K-Nearest Neighbors):**
  + **Accuracy:** 0.6267 (moderately good)
  + **Precision:** 0.4057 (low)
  + **Recall:** 0.2387 (low)
  + **F1-score:** 0.2843 (low)
  + **AUC:** 0.5221 (slightly better than random guessing)
  + **Overall:** KNN shows moderate accuracy but has low precision and recall, similar to Random Forest.
* **Logistic Regression:**
  + **Accuracy:** 0.6400 (moderately good)
  + **Precision:** 0.0000 (very poor)
  + **Recall:** 0.0000 (very poor)
  + **F1-score:** 0.0000 (very poor)
  + **AUC:** 0.4622 (worse than random guessing)
  + **Overall:** Logistic Regression performs very poorly, especially in terms of precision and recall. This suggests it's unable to identify positive instances at all.
* **XGBoost:**
  + **Accuracy:** 0.5833 (relatively low)
  + **Precision:** 0.3941 (low)
  + **Recall:** 0.3354 (low)
  + **F1-score:** 0.3439 (low)
  + **AUC:** 0.5590 (slightly better than random guessing)
  + **Overall:** XGBoost has the lowest accuracy and performs poorly in precision and recall, although it's slightly better than other models in terms of recall.
* **LSTM (Long Short-Term Memory):**
  + **Accuracy:** 0.6567 (highest accuracy)
  + **Precision:** 0.0000 (very poor)
  + **Recall:** 0.0000 (very poor)
  + **F1-score:** 0.0000 (very poor)
  + **AUC:** 0.4440 (worse than random guessing)
  + **Overall:** LSTM shows the highest accuracy but fails completely in identifying positive instances, similar to Logistic Regression.

**General Observations:**

* **Low Precision and Recall:** All models, except for XGBoost to some extent, struggle with precision and recall, especially recall. This suggests that the models have difficulty correctly identifying positive instances and are prone to false negatives.
* **Imbalanced Data:** The low precision and recall scores could indicate an imbalanced dataset, where there are significantly more negative instances than positive instances.
* **Performance Discrepancy:** There is a discrepancy between accuracy and other metrics. While accuracy is moderately good for some models, precision, recall, and F1-score are generally low.
* **AUC Close to 0.5:** The AUC scores are close to 0.5, indicating that the models have limited ability to distinguish between positive and negative classes.

**Recommendations:**

* **Address Data Imbalance:** Investigate and address potential data imbalance issues using techniques like oversampling, undersampling, or SMOTE.
* **Feature Engineering:** Explore feature engineering to improve the models' ability to distinguish between positive and negative instances.
* **Hyperparameter Tuning:** Tune the hyperparameters of the models to optimize their performance.
* **Consider Different Models:** Explore other machine learning models that might be better suited for this classification task.
* **Collect More Data:** If possible, collect more data to improve the models' performance.
* **Re-evaluate Evaluation Metrics:** Consider the most important metrics for your specific problem. If identifying positive instances is critical, focus on improving precision and recall.

In conclusion, the models show moderate accuracy but struggle with precision and recall, indicating potential issues with data imbalance or model suitability. Further investigation and improvements are needed to enhance the models' performance.

**Reference:**

**Part One:**

[K. Harsha, S. Yuva Nitya, S. Kota, K. Satyanarayana and J. Lakshmi, "Empirical evaluation of Amazon fine food reviews using Text mining,"](https://ieeexplore.ieee.org/document/10126349)

J. McAuley and J. Leskovec. [From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews](http://i.stanford.edu/~julian/pdfs/www13.pdf). WWW, 2013