Cybersecurity Threats Classification Analysis (2015-2024)

DIVYA CHENTHAMARAKSHAN

April 22, 2025

Contents

1 Introduction

This document presents an in-depth analysis of global cybersecurity threats using classification algorithms including **Decision Trees**, **Random Forest,** and **XGBoost**. The dataset we're analyzing covers cybersecurity incidents from 2015 to 2024, containing valuable information about different types of cyber attacks, their vectors, and impacts. In today's digital landscape, cybersecurity has become a critical concern for organizations worldwide. The increasing sophistication of cyber attacks necessitates advanced analytical approaches to detect, classify, and predict potential threats. Machine learning algorithms, particularly classification techniques, have proven effective in identifying patterns and relationships in complex cybersecurity data. Our analysis aims to:

Implement and compare three powerful **classification algorithms** (Decision Tree, Random Forest, and XGBoost) Identify key features that contribute most to threat classification Evaluate model performance using appropriate metrics Provide actionable insights and recommendations based on our findings

2 Data Preparation

Loading and Initial Exploration We begin by loading the dataset and exploring its structure.

```
# Load the cybersecurity threats dataset
cyber_data <- read.csv("Global_Cybersecurity_Threats_2015-2024.csv")

# Display the structure of the dataset
str(cyber_data)
```

```
## 'data.frame':    3000 obs. of  10 variables:
##  $ Country                            : chr  "China" "China" "India" "UK" ...
##  $ Year                               : int  2019 2019 2017 2024 2018 2017 2016 2018 2016 2023 ...
##  $ Attack.Type                        : chr  "Phishing" "Ransomware" "Man-in-the-Middle" "Ransomware
```

```
##  $ Target.Industry                    : chr  "Education" "Retail" "IT" "Telecommunications" ...
##  $ Financial.Loss..in.Million...      : num  80.5 62.2 38.6 41.4 74.4 ...
##  $ Number.of.Affected.Users           : int  773169 295961 605895 659320 810682 285201 431262 909991
##  $ Attack.Source                      : chr  "Hacker Group" "Hacker Group" "Hacker Group" "Nation-sta
##  $ Security.Vulnerability.Type        : chr  "Unpatched Software" "Unpatched Software" "Weak Passwor
##  $ Defense.Mechanism.Used             : chr  "VPN" "Firewall" "VPN" "AI-based Detection" ...
##  $ Incident.Resolution.Time..in.Hours.: int  63 71 20 7 68 25 34 66 47 58 ...
```

```r
# Summary statistics
summary(cyber_data)
```

```
##    Country               Year      Attack.Type        Target.Industry
##  Length:3000        Min.   :2015   Length:3000        Length:3000
##  Class :character   1st Qu.:2017   Class :character   Class :character
##  Mode  :character   Median :2020   Mode  :character   Mode  :character
##                     Mean   :2020
##                     3rd Qu.:2022
##                     Max.   :2024
##  Financial.Loss..in.Million... Number.of.Affected.Users Attack.Source
##  Min.   : 0.50                 Min.   :   424           Length:3000
##  1st Qu.:25.76                 1st Qu.:255805           Class :character
##  Median :50.80                 Median :504513           Mode  :character
##  Mean   :50.49                 Mean   :504684
##  3rd Qu.:75.63                 3rd Qu.:758088
##  Max.   :99.99                 Max.   :999635
##  Security.Vulnerability.Type Defense.Mechanism.Used
##  Length:3000                 Length:3000
##  Class :character            Class :character
##  Mode  :character            Mode  :character
##
##
##
##  Incident.Resolution.Time..in.Hours.
##  Min.   : 1.00
##  1st Qu.:19.00
##  Median :37.00
##  Mean   :36.48
##  3rd Qu.:55.00
##  Max.   :72.00
```

```r
# Check for missing values
missing_values <- colSums(is.na(cyber_data))
print("Missing values per column:")
```

```
## [1] "Missing values per column:"
```

```r
print(missing_values[missing_values > 0])
```

```
## named numeric(0)
```

```r
# Print current column names
cat("Original column names:\n")
```

```
## Original column names:
```

```r
print(colnames(cyber_data))
```

```
##  [1] "Country"                          "Year"
##  [3] "Attack.Type"                      "Target.Industry"
##  [5] "Financial.Loss..in.Million..."    "Number.of.Affected.Users"
##  [7] "Attack.Source"                    "Security.Vulnerability.Type"
##  [9] "Defense.Mechanism.Used"           "Incident.Resolution.Time..in.Hours."
```

```r
# Create a clean mapping for column names
clean_names <- c(
  "Country" = "Country",
  "Year" = "Year",
  "Attack.Type" = "Attack_Type",
  "Target.Industry" = "Target_Industry",
  "Financial.Loss..in.Million..." = "Financial_Loss",
  "Number.of.Affected.Users" = "Affected_Users",
  "Attack.Source" = "Attack_Source",
  "Security.Vulnerability.Type" = "Vulnerability_Type",
  "Defense.Mechanism.Used" = "Defense_Mechanism",
  "Incident.Resolution.Time..in.Hours." = "Resolution_Hours"
)

# Apply the renaming (only for columns that exist in the dataset)
existing_cols <- names(clean_names)[names(clean_names) %in% colnames(cyber_data)]
cyber_data <- cyber_data %>%
  rename_with(~ clean_names[.x], .cols = existing_cols)

# Make a copy for processing
cyber_data_processed <- cyber_data

# Print new column names
cat("\nNew column names:\n")
```

```
##
## New column names:
```

```r
print(colnames(cyber_data))
```

```
##  [1] "Country"           "Year"              "Attack_Type"
##  [4] "Target_Industry"   "Financial_Loss"    "Affected_Users"
##  [7] "Attack_Source"     "Vulnerability_Type" "Defense_Mechanism"
## [10] "Resolution_Hours"
```

3 Data Preprocessing

Before model building, we need to prepare our data by handling missing values, encoding categorical variables, and scaling numerical features.

```r
# Identify numeric and categorical columns
numeric_cols <- names(cyber_data_processed)[sapply(cyber_data_processed, is.numeric)]
categorical_cols <- names(cyber_data_processed)[sapply(cyber_data_processed, is.factor) |
                                                sapply(cyber_data_processed, is.character)]

# Handle missing values
# For numeric columns, impute with median
for(col in numeric_cols) {
  if(sum(is.na(cyber_data_processed[[col]])) > 0) {
    cyber_data_processed[[col]][is.na(cyber_data_processed[[col]])] <-
      median(cyber_data_processed[[col]], na.rm = TRUE)
  }
}

# For categorical columns, impute with mode
for(col in categorical_cols) {
  if(sum(is.na(cyber_data_processed[[col]])) > 0) {
    mode_val <- names(sort(table(cyber_data_processed[[col]]), decreasing = TRUE))[1]
    cyber_data_processed[[col]][is.na(cyber_data_processed[[col]])] <- mode_val
  }
}

# Convert categorical variables to factors if they are not already
for(col in categorical_cols) {
  cyber_data_processed[[col]] <- as.factor(cyber_data_processed[[col]])
}

# Set the target variable explicitly to Attack_Type
target_variable <- "Attack_Type"

# Check class distribution of target variable
print(paste("Target variable:", target_variable))
```

```
## [1] "Target variable: Attack_Type"
```

```r
print(table(cyber_data_processed[[target_variable]]))
```

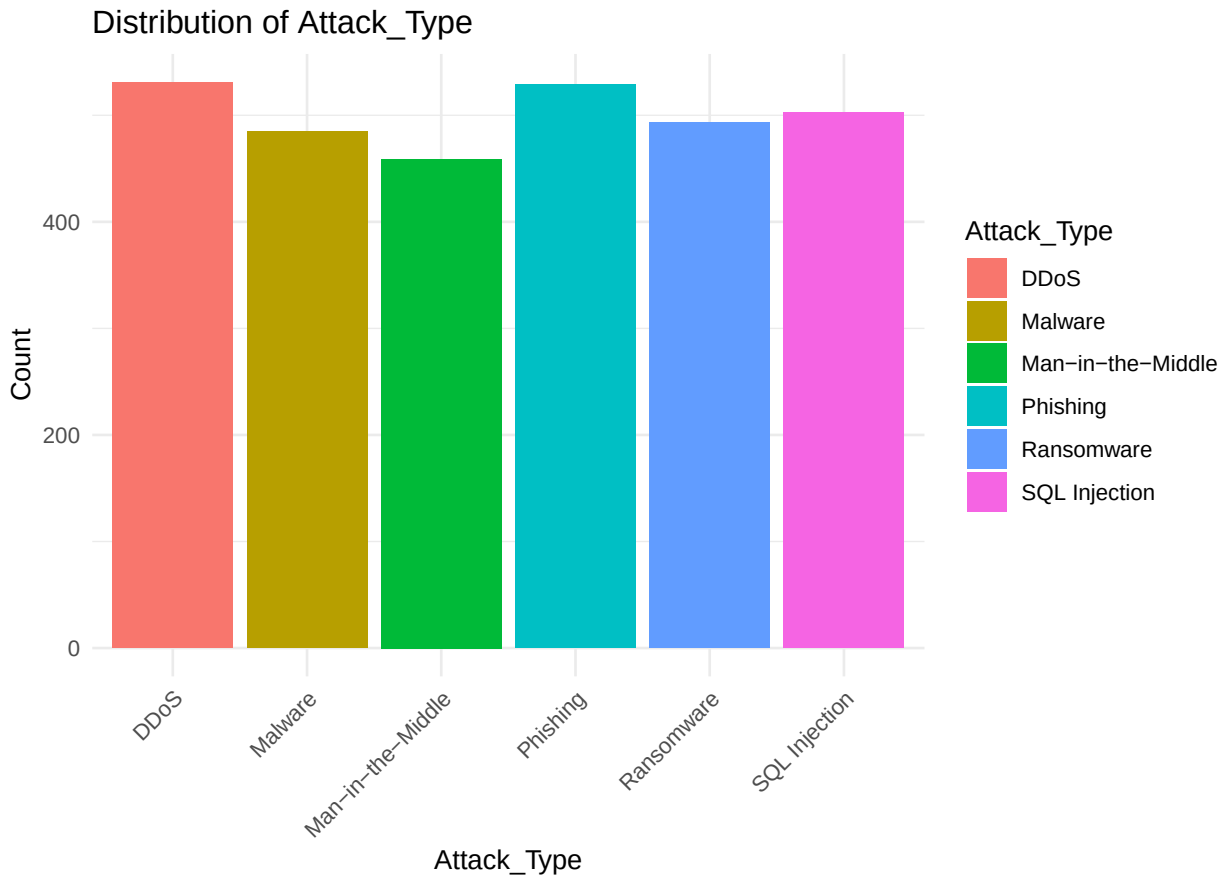
```
##
##              DDoS           Malware Man-in-the-Middle          Phishing
##               531               485               459               529
##        Ransomware     SQL Injection
##               493               503
```

4 Exploratory Data Analysis

Let's explore the data to gain insights into the distribution of cyber threats and potential relationships between variables.

4.1 Distribution of Attack Types

```r
# Plot the distribution of the target variable
ggplot(cyber_data_processed, aes_string(x = target_variable, fill = target_variable)) +
  geom_bar() +
  theme_minimal() +
  labs(title = paste("Distribution of", target_variable),
       x = target_variable,
       y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



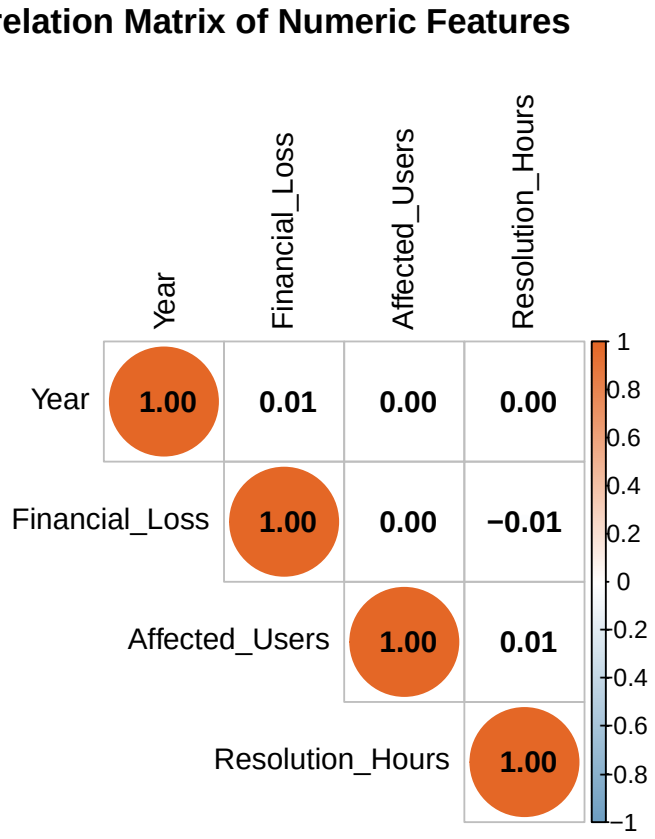
Distribution of Attack Types This bar chart displays the frequency distribution of different cybersecurity attack types in the dataset. The relatively even distribution across DDoS, Malware, Man-in-the-Middle, Phishing, Ransomware, and SQL Injection categories indicates a balanced dataset without significant class imbalance. This balance is important for developing classification models that don't overly favor specific attack types. The similar heights of the bars suggest that all attack categories have comparable representation in the dataset, providing a solid foundation for training machine learning models to recognize each type of attack with similar effectiveness.

4.2 Correlation Matrix of Numeric Features

```r
# Select numeric features for correlation analysis
if(length(numeric_cols) > 1) {
  numeric_data <- cyber_data_processed[, numeric_cols]

 cor_matrix <- cor(numeric_data, use = "complete.obs")

# Now plot the correlation matrix
corrplot(cor_matrix, method = "circle", type = "upper",
         tl.col = "black", tl.srt = 90, addCoef.col = "black",
         title = "Correlation Matrix of Numeric Features",
         col = colorRampPalette(c("#6D9EC1", "white", "#E46726"))(200),
         mar = c(0, 0, 4, 0))
}
```

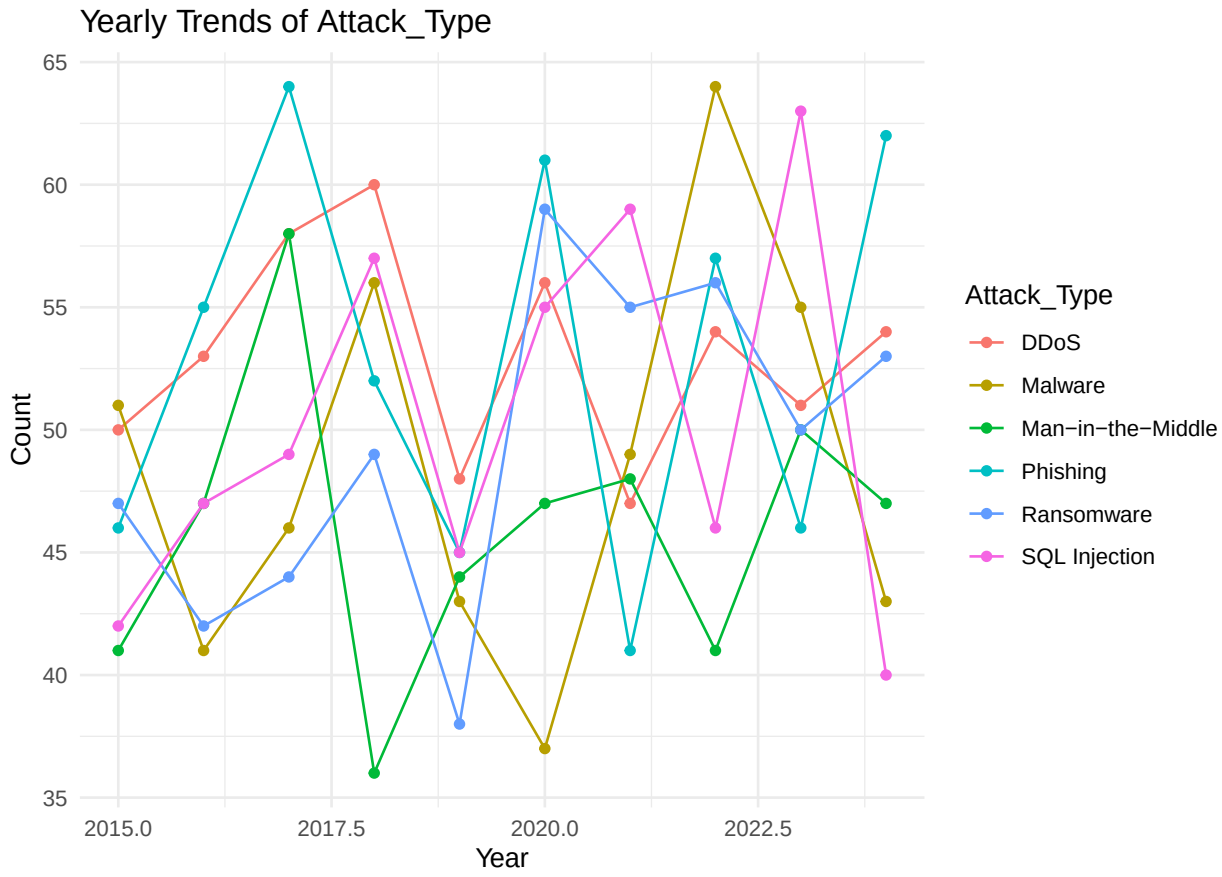


Correlation Matrix of Numeric Features This heatmap visualizes relationships between numeric variables in the dataset. The diagonal shows perfect correlation (1.00) of each variable with itself. The minimal correlations between Year, Financial_Loss, Affected_Users, and Resolution_Hours (all near 0) indicate these variables are mostly independent of each other. This independence is valuable for machine learning as it suggests minimal multicollinearity, allowing models to learn distinct patterns from each feature without redundancy. The color gradient from blue (negative correlation) to white (no correlation) to red (positive correlation) helps quickly identify relationship strengths and directions.

4.3 Yearly Trends of Attack Types

```r
# Create visualization for temporal trends using the Year column
if("Year" %in% names(cyber_data_processed)) {
  # Plot temporal trends by year (handling Year directly as numeric)
  yearly_trends <- cyber_data_processed %>%
    group_by(Year, !!sym(target_variable)) %>%
    summarise(count = n(), .groups = "drop")

  ggplot(yearly_trends, aes(x = Year, y = count, color = !!sym(target_variable), group = !!sym(target_va
    geom_line() +
    geom_point() +
    theme_minimal() +
    labs(title = paste("Yearly Trends of", target_variable),
         x = "Year",
         y = "Count") +
    theme(legend.position = "right")
}
```



Yearly Trends of Attack Types This line graph tracks the frequency of different attack types from 2015 to 2024. The fluctuating patterns show how threat landscapes evolved over time. Phishing attacks peaked around 2017 and 2020, while Malware showed higher numbers in 2022. The crossing lines indicate changing prevalence of different attack methods over the years. These temporal patterns provide insights into emerging threats and declining techniques, helping security professionals anticipate future trends. The

graph demonstrates the dynamic nature of cybersecurity threats and highlights the importance of adapting defense strategies to address evolving attack methodologies.

5 Model Development

We'll implement three classification algorithms: Decision Tree, Random Forest, and XGBoost. The goal is to predict the cybersecurity threat type and compare the performance of these algorithms. Data Splitting First, we'll split our data into training and testing sets.

```
# Set seed for reproducibility
set.seed(123)

# Create a stratified split to ensure balanced classes in both training and test sets
train_index <- createDataPartition(cyber_data_processed[[target_variable]], p = 0.7, list = FALSE)
train_data <- cyber_data_processed[train_index, ]
test_data <- cyber_data_processed[-train_index, ]

# Check the dimensions of the training and test sets
print(paste("Training set dimensions:", nrow(train_data), "rows,", ncol(train_data), "columns"))
```

```
## [1] "Training set dimensions: 2104 rows, 10 columns"
```

```
print(paste("Test set dimensions:", nrow(test_data), "rows,", ncol(test_data), "columns"))
```

```
## [1] "Test set dimensions: 896 rows, 10 columns"
```

```
# Prepare formula for modeling
# Exclude any ID columns or date columns that shouldn't be used for prediction
exclude_cols <- c("id", "ID")
predictors <- setdiff(names(cyber_data_processed), c(target_variable, exclude_cols))

# Create the formula
formula <- as.formula(paste(target_variable, "~", paste(predictors, collapse = " + ")))
```

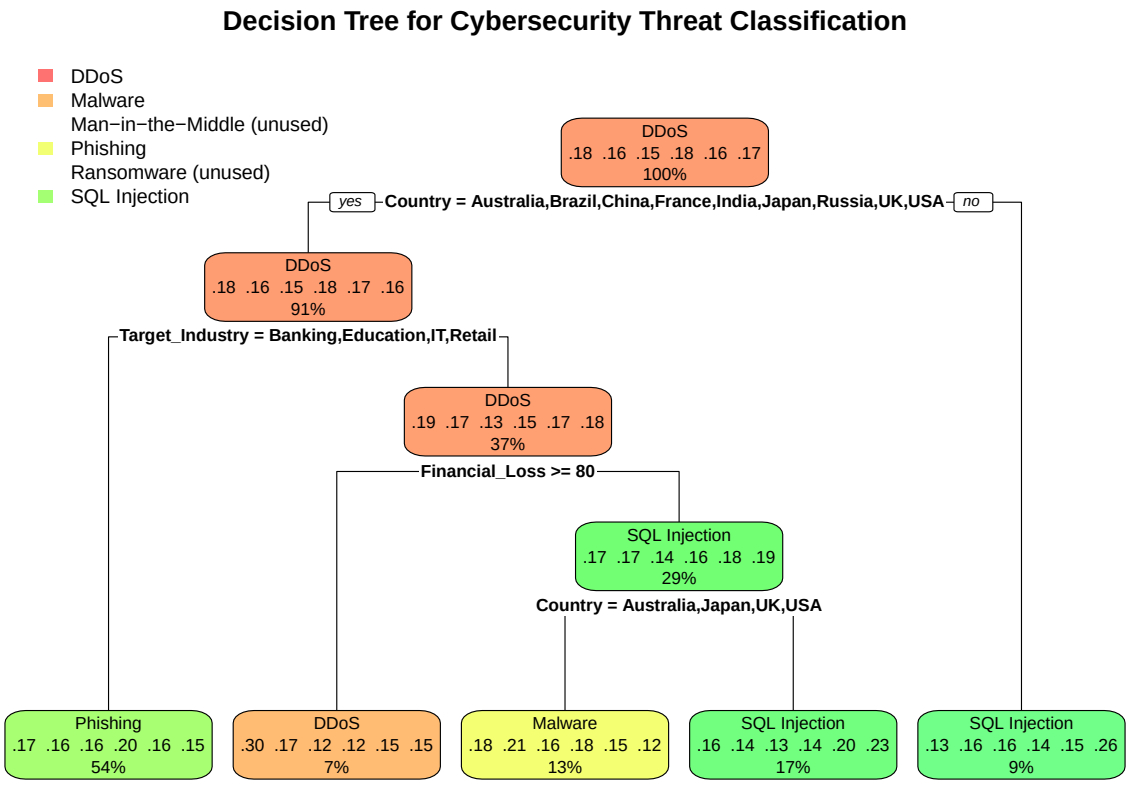
6 Decision Tree Model

Decision trees are intuitive and easy to interpret classification models that create a flowchart-like structure for decision-making.

6.1 Decision Tree for Cybersecurity Threat Classification

```
# Train the Decision Tree model
dt_model <- rpart(formula,
                  data = train_data,
                  method = "class",
                  control = rpart.control(cp = 0.01, minsplit = 20))
```

```
# Plot the Decision Tree
rpart.plot(dt_model,
           main = "Decision Tree for Cybersecurity Threat Classification",
           extra = 104,
           box.palette = "RdYlGn",
           fallen.leaves = TRUE)
```




```
# Make predictions on the test set
dt_predictions <- predict(dt_model, test_data, type = "class")

# Evaluate the model
dt_confusion <- confusionMatrix(dt_predictions, test_data[[target_variable]])
print("Decision Tree Confusion Matrix:")
```

```
## [1] "Decision Tree Confusion Matrix:"
```

```
print(dt_confusion)
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           DDoS Malware Man-in-the-Middle Phishing Ransomware
##   DDoS                 14       9                 13       10         13
```

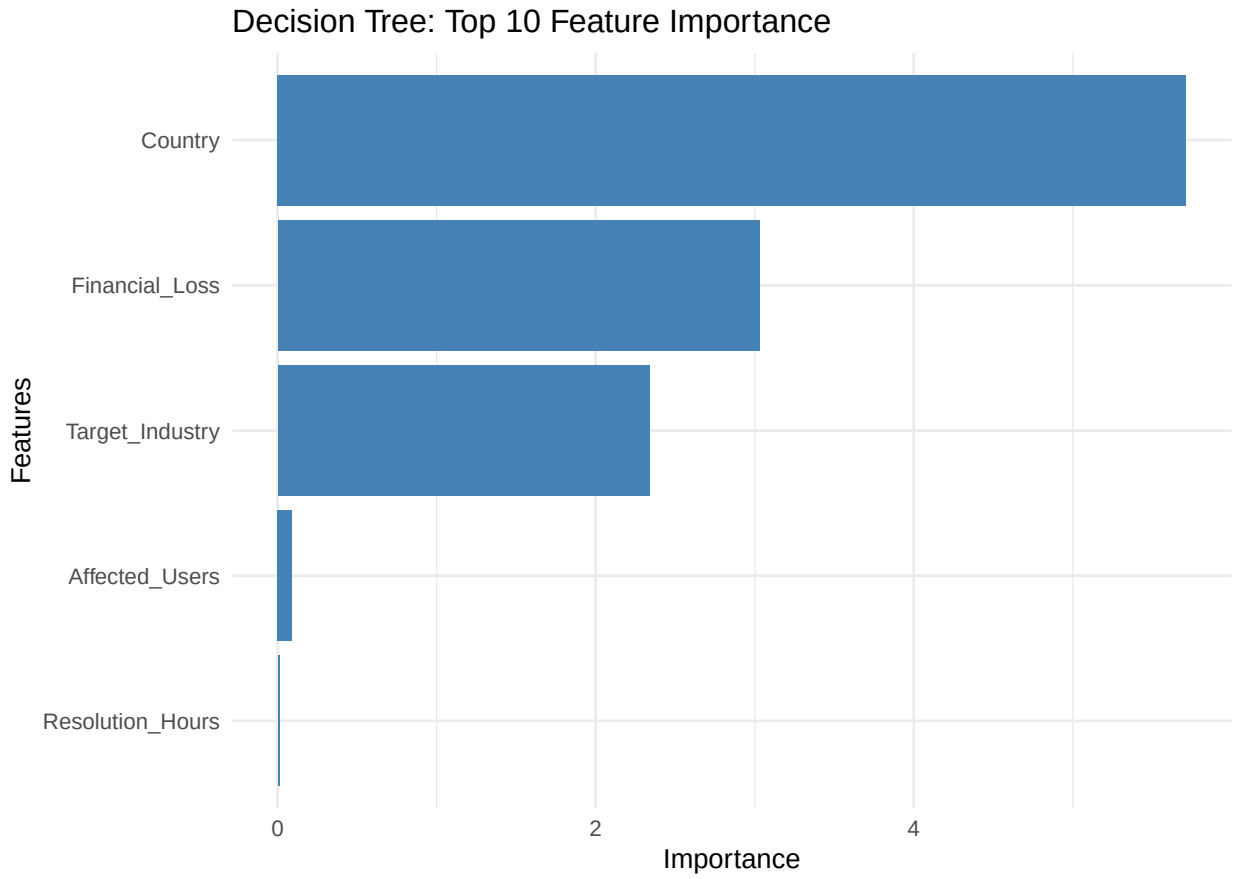
```
##   Malware             17      21                16       17         15
##   Man-in-the-Middle    0       0                 0        0          0
##   Phishing            77      69                72       96         75
##   Ransomware           0       0                 0        0          0
##   SQL Injection       51      46                36       35         44
##                     Reference
## Prediction          SQL Injection
##   DDoS                        10
##   Malware                     19
##   Man-in-the-Middle            0
##   Phishing                    73
##   Ransomware                   0
##   SQL Injection               48
##
## Overall Statistics
##
##                Accuracy : 0.1998
##                  95% CI : (0.1741, 0.2275)
##     No Information Rate : 0.1775
##     P-Value [Acc > NIR] : 0.04567
##
##                   Kappa : 0.0334
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: DDoS Class: Malware Class: Man-in-the-Middle
## Sensitivity              0.08805        0.14483                   0.0000
## Specificity              0.92537        0.88815                   1.0000
## Pos Pred Value           0.20290        0.20000                      NaN
## Neg Pred Value           0.82467        0.84324                   0.8471
## Prevalence               0.17746        0.16183                   0.1529
## Detection Rate           0.01562        0.02344                   0.0000
## Detection Prevalence     0.07701        0.11719                   0.0000
## Balanced Accuracy        0.50671        0.51649                   0.5000
##                      Class: Phishing Class: Ransomware Class: SQL Injection
## Sensitivity                   0.6076            0.0000              0.32000
## Specificity                   0.5041            1.0000              0.71582
## Pos Pred Value                0.2078               NaN              0.18462
## Neg Pred Value                0.8571            0.8359              0.83962
## Prevalence                    0.1763            0.1641              0.16741
## Detection Rate                0.1071            0.0000              0.05357
## Detection Prevalence          0.5156            0.0000              0.29018
## Balanced Accuracy             0.5558            0.5000              0.51791
```

Decision Tree for Cybersecurity Threat Classification This visualization represents the decision-making process of the tree model. The flowchart structure shows how the algorithm makes classification decisions based on feature values. Key decision nodes involve Country, Target_Industry, Financial_Loss, and other features. Each node displays class probabilities, with the dominant class highlighted. The tree's branching pattern reveals how the model segments the data to make predictions. This interpretable representation helps security analysts understand the logical rules the model uses, making it valuable for explaining predictions and identifying key decision factors in threat classification.

6.2 Decision Tree: Top 10 Feature Importance

```r
# Extract variable importance
dt_importance <- dt_model$variable.importance
dt_importance_df <- data.frame(
  Feature = names(dt_importance),
  Importance = dt_importance
)
dt_importance_df <- dt_importance_df[order(-dt_importance_df$Importance), ]

# Plot variable importance
ggplot(head(dt_importance_df, 10), aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Decision Tree: Top 10 Feature Importance",
       x = "Features",
       y = "Importance")
```



Decision Tree: Top 10 Feature Importance This horizontal bar chart ranks features by their importance in the Decision Tree model. Country emerges as the most influential feature, followed by Financial_Loss and Target_Industry. The substantial gap between the top three features and the rest indicates these variables drive most of the model's decisions. Features like Resolution_Hours and Affected_Users show minimal importance. This visualization helps identify which attributes most strongly influence the detection and

classification of cybersecurity threats, guiding security teams on which indicators to monitor closely and prioritize in their threat detection systems.

7 Random Forest Model

Random Forest is an ensemble learning method that builds multiple decision trees and merges their predictions for improved accuracy and reduced overfitting.

7.1 Random Forest: Top 10 Feature Importance

```
# Train the Random Forest model
rf_model <- randomForest(formula,
                         data = train_data,
                         ntree = 100,
                         importance = TRUE)

# Print model summary
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = formula, data = train_data, ntree = 100,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 83.08%
## Confusion matrix:
##                   DDoS Malware Man-in-the-Middle Phishing Ransomware
## DDoS                68      57                44       87         53
## Malware             77      48                34       70         63
## Man-in-the-Middle   65      45                47       61         42
## Phishing            64      57                45       75         62
## Ransomware          58      42                42       80         58
## SQL Injection       72      55                44       63         59
##                   SQL Injection class.error
## DDoS                         63   0.8172043
## Malware                      48   0.8588235
## Man-in-the-Middle            62   0.8540373
## Phishing                     68   0.7978437
## Ransomware                   66   0.8323699
## SQL Injection                60   0.8300283
```

```
# Make predictions on the test set
rf_predictions <- predict(rf_model, test_data)

# Evaluate the model
rf_confusion <- confusionMatrix(rf_predictions, test_data[[target_variable]])
print("Random Forest Confusion Matrix:")
```

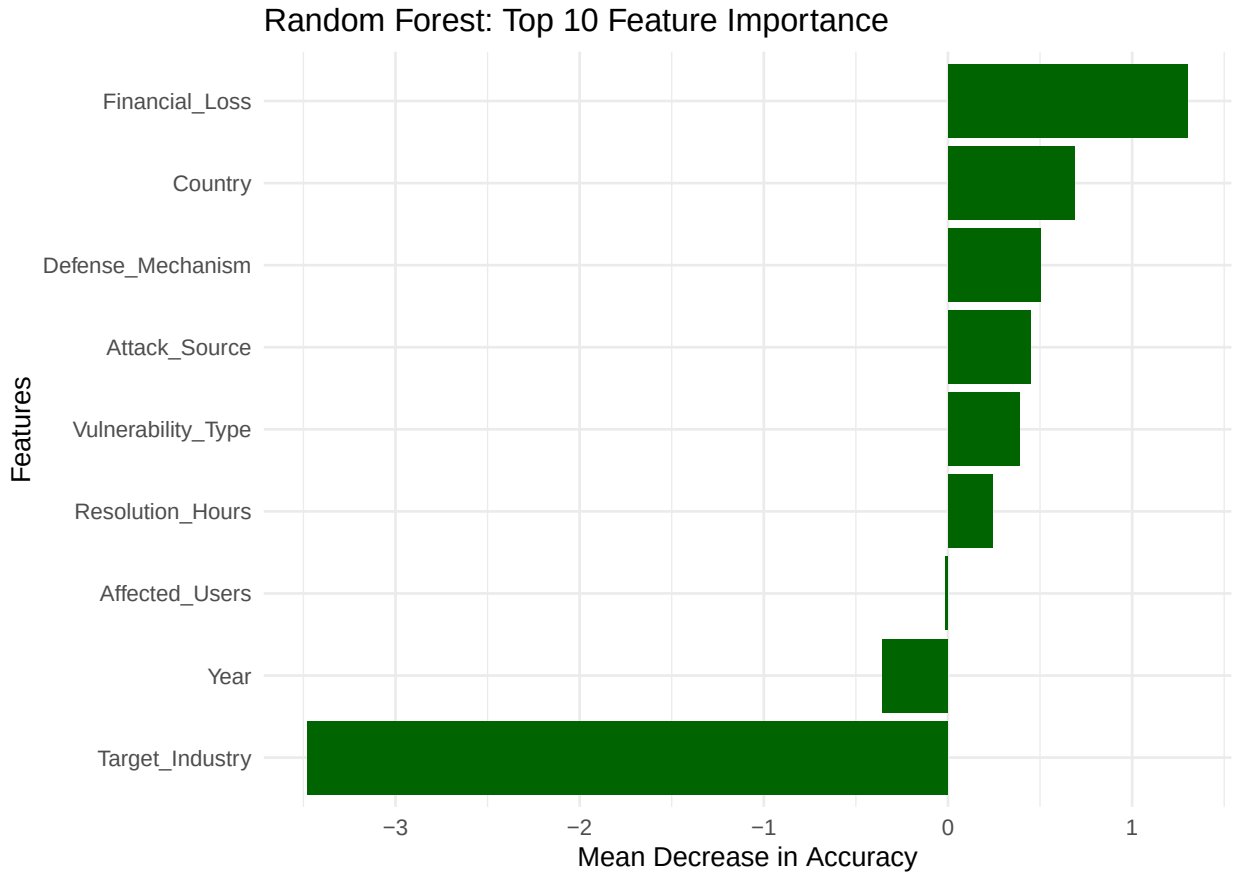
```
## [1] "Random Forest Confusion Matrix:"
```

```
print(rf_confusion)
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction          DDoS Malware Man-in-the-Middle Phishing Ransomware
##   DDoS                31      27                33       34         37
##   Malware             24      17                15       11         13
##   Man-in-the-Middle   24      15                17       27         12
##   Phishing            34      26                19       35         35
##   Ransomware          23      28                17       30         21
##   SQL Injection       23      32                36       21         29
##                   Reference
## Prediction          SQL Injection
##   DDoS                         32
##   Malware                      20
##   Man-in-the-Middle            19
##   Phishing                     33
##   Ransomware                   26
##   SQL Injection                20
##
## Overall Statistics
##
##                Accuracy : 0.1574
##                  95% CI : (0.1341, 0.1829)
##     No Information Rate : 0.1775
##     P-Value [Acc > NIR] : 0.948869
##
##                   Kappa : -0.0133
##
##  Mcnemar's Test P-Value : 0.007768
##
## Statistics by Class:
##
##                      Class: DDoS Class: Malware Class: Man-in-the-Middle
## Sensitivity               0.1950        0.11724                  0.12409
## Specificity               0.7788        0.88948                  0.87220
## Pos Pred Value            0.1598        0.17000                  0.14912
## Neg Pred Value            0.8177        0.83920                  0.84655
## Prevalence                0.1775        0.16183                  0.15290
## Detection Rate            0.0346        0.01897                  0.01897
## Detection Prevalence      0.2165        0.11161                  0.12723
## Balanced Accuracy         0.4869        0.50336                  0.49814
##                      Class: Phishing Class: Ransomware Class: SQL Injection
## Sensitivity                  0.22152           0.14286              0.13333
## Specificity                  0.80081           0.83445              0.81099
## Pos Pred Value               0.19231           0.14483              0.12422
## Neg Pred Value               0.82773           0.83222              0.82313
## Prevalence                   0.17634           0.16406              0.16741
## Detection Rate               0.03906           0.02344              0.02232
## Detection Prevalence         0.20312           0.16183              0.17969
## Balanced Accuracy            0.51117           0.48865              0.47216
```

```r
# Extract variable importance
rf_importance <- importance(rf_model, type = 1)
rf_importance_df <- data.frame(
  Feature = rownames(rf_importance),
  Importance = rf_importance[, "MeanDecreaseAccuracy"]
)
rf_importance_df <- rf_importance_df[order(-rf_importance_df$Importance), ]

# Plot variable importance
ggplot(head(rf_importance_df, 10), aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "darkgreen") +
  coord_flip() +
  theme_minimal() +
  labs(title = "Random Forest: Top 10 Feature Importance",
       x = "Features",
       y = "Mean Decrease in Accuracy")
```



Random Forest: Top 10 Feature Importance This bar chart shows feature importance for the Random Forest model, using Mean Decrease in Accuracy. Target_Industry and Financial_Loss show high negative importance, indicating the model's accuracy decreases when these features are removed. Country, Defense_Mechanism, and Attack_Source show moderate importance. The negative values for some features suggest potential overfitting or complex interactions that the model struggles to leverage effectively. This analysis helps identify which variables contribute most to the ensemble model's predictive power and provides insights into potential model refinements.

8 XGBoost Model

XGBoost (Extreme Gradient Boosting) is an optimized distributed gradient boosting library that implements machine learning algorithms under the Gradient Boosting framework.

8.1 XGBoost: Top 10 Feature Importance

```
# Prepare data in format suitable for XGBoost
# Need to convert categorical variables to numeric (one-hot encoding)
train_matrix <- model.matrix(formula, data = train_data)[, -1]  # Remove intercept
test_matrix <- model.matrix(formula, data = test_data)[, -1]  # Remove intercept

# Convert target to numeric (zero-indexed)
train_label <- as.integer(train_data[[target_variable]]) - 1
test_label <- as.integer(test_data[[target_variable]]) - 1

# Create DMatrix objects
dtrain <- xgb.DMatrix(data = train_matrix, label = train_label)
dtest <- xgb.DMatrix(data = test_matrix, label = test_label)

# Set XGBoost parameters
params <- list(
  objective = "multi:softmax",
  num_class = length(levels(cyber_data_processed[[target_variable]])),
  eta = 0.3,
  max_depth = 6,
  min_child_weight = 1,
  subsample = 0.8,
  colsample_bytree = 0.8
)

# Train the XGBoost model
xgb_model <- xgboost(
  params = params,
  data = dtrain,
  nrounds = 100,
  verbose = 0
)

# Make predictions on the test set
xgb_predictions_numeric <- predict(xgb_model, dtest)
xgb_predictions <- factor(levels(cyber_data_processed[[target_variable]])[xgb_predictions_numeric + 1],
                          levels = levels(cyber_data_processed[[target_variable]]))

# Evaluate the model
xgb_confusion <- confusionMatrix(xgb_predictions, test_data[[target_variable]])
print("XGBoost Confusion Matrix:")
```

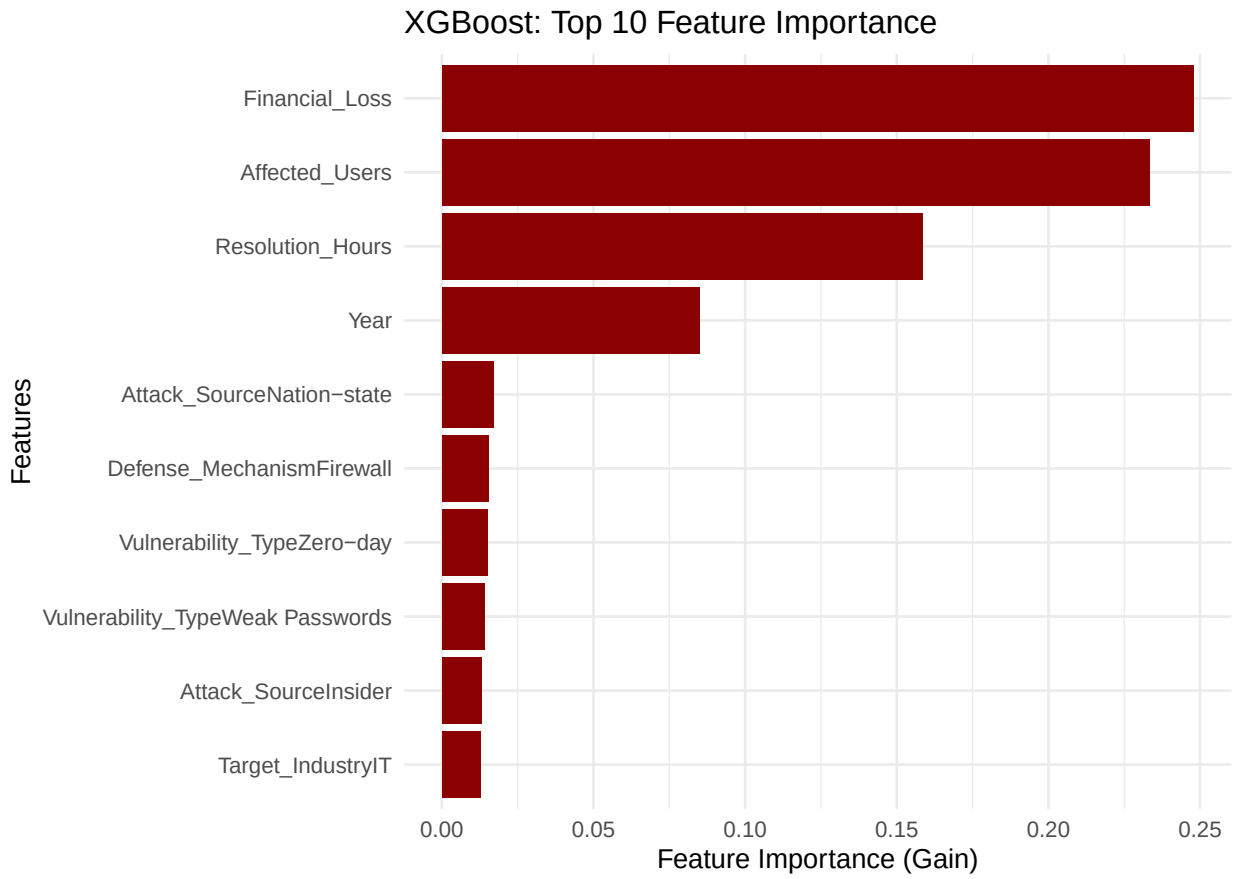
```
## [1] "XGBoost Confusion Matrix:"
```

```r
print(xgb_confusion)
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           DDoS Malware Man-in-the-Middle Phishing Ransomware
##   DDoS                 21      20                34       29         28
##   Malware              33      25                21       23         15
##   Man-in-the-Middle    23      19                16       29         22
##   Phishing             32      30                26       34         28
##   Ransomware           29      27                17       24         25
##   SQL Injection        21      24                23       19         29
##                     Reference
## Prediction           SQL Injection
##   DDoS                          37
##   Malware                       23
##   Man-in-the-Middle             22
##   Phishing                      23
##   Ransomware                    22
##   SQL Injection                 23
##
## Overall Statistics
##
##                Accuracy : 0.1607
##                  95% CI : (0.1373, 0.1864)
##     No Information Rate : 0.1775
##     P-Value [Acc > NIR] : 0.9137
##
##                   Kappa : -0.0081
##
##  Mcnemar's Test P-Value : 0.3285
##
## Statistics by Class:
##
##                      Class: DDoS Class: Malware Class: Man-in-the-Middle
## Sensitivity              0.13208         0.1724                  0.11679
## Specificity              0.79919         0.8469                  0.84848
## Pos Pred Value           0.12426         0.1786                  0.12214
## Neg Pred Value           0.81018         0.8413                  0.84183
## Prevalence               0.17746         0.1618                  0.15290
## Detection Rate           0.02344         0.0279                  0.01786
## Detection Prevalence     0.18862         0.1562                  0.14621
## Balanced Accuracy        0.46563         0.5096                  0.48264
##                      Class: Phishing Class: Ransomware Class: SQL Injection
## Sensitivity                  0.21519            0.1701              0.15333
## Specificity                  0.81165            0.8411              0.84450
## Pos Pred Value               0.19653            0.1736              0.16547
## Neg Pred Value               0.82849            0.8378              0.83223
## Prevalence                   0.17634            0.1641              0.16741
## Detection Rate               0.03795            0.0279              0.02567
## Detection Prevalence         0.19308            0.1607              0.15513
## Balanced Accuracy            0.51342            0.5056              0.49892
```

```r
# Extract variable importance
xgb_importance <- xgb.importance(feature_names = colnames(train_matrix), model = xgb_model)
xgb_importance_df <- data.frame(
  Feature = xgb_importance$Feature,
  Importance = xgb_importance$Gain
)

# Plot variable importance
ggplot(head(xgb_importance_df, 10), aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "darkred") +
  coord_flip() +
  theme_minimal() +
  labs(title = "XGBoost: Top 10 Feature Importance",
       x = "Features",
       y = "Feature Importance (Gain)")
```



XGBoost: Top 10 Feature Importance This visualization ranks features by importance (gain) in the XGBoost model. Financial_Loss and Affected_Users clearly dominate, followed by Resolution_Hours and Year. The remaining features show considerably lower importance. This pattern differs from the other models, highlighting XGBoost's unique feature utilization. The concentration of importance in numeric features suggests XGBoost effectively leverages quantitative data for classifications. This information helps security teams understand which metrics most strongly indicate specific threat types and can guide the development of monitoring systems focused on these high-value indicators.

9 Model Comparison

Now let's compare the performance of all three models to identify the best approach for cybersecurity threat classification.

```
# Extract accuracy metrics
dt_accuracy <- dt_confusion$overall["Accuracy"]
rf_accuracy <- rf_confusion$overall["Accuracy"]
xgb_accuracy <- xgb_confusion$overall["Accuracy"]

# Create a comparison dataframe
comparison_df <- data.frame(
  Model = c("Decision Tree", "Random Forest", "XGBoost"),
  Accuracy = c(dt_accuracy, rf_accuracy, xgb_accuracy),
  Kappa = c(dt_confusion$overall["Kappa"],
            rf_confusion$overall["Kappa"],
            xgb_confusion$overall["Kappa"])
)

# Add average metrics by class
comparison_df$Precision <- c(mean(dt_confusion$byClass[, "Precision"], na.rm = TRUE),
                             mean(rf_confusion$byClass[, "Precision"], na.rm = TRUE),
                             mean(xgb_confusion$byClass[, "Precision"], na.rm = TRUE))

comparison_df$Recall <- c(mean(dt_confusion$byClass[, "Recall"], na.rm = TRUE),
                          mean(rf_confusion$byClass[, "Recall"], na.rm = TRUE),
                          mean(xgb_confusion$byClass[, "Recall"], na.rm = TRUE))

comparison_df$F1_Score <- c(mean(dt_confusion$byClass[, "F1"], na.rm = TRUE),
                            mean(rf_confusion$byClass[, "F1"], na.rm = TRUE),
                            mean(xgb_confusion$byClass[, "F1"], na.rm = TRUE))

# Display the comparison table
kable(comparison_df, caption = "Model Performance Comparison", digits = 4) %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

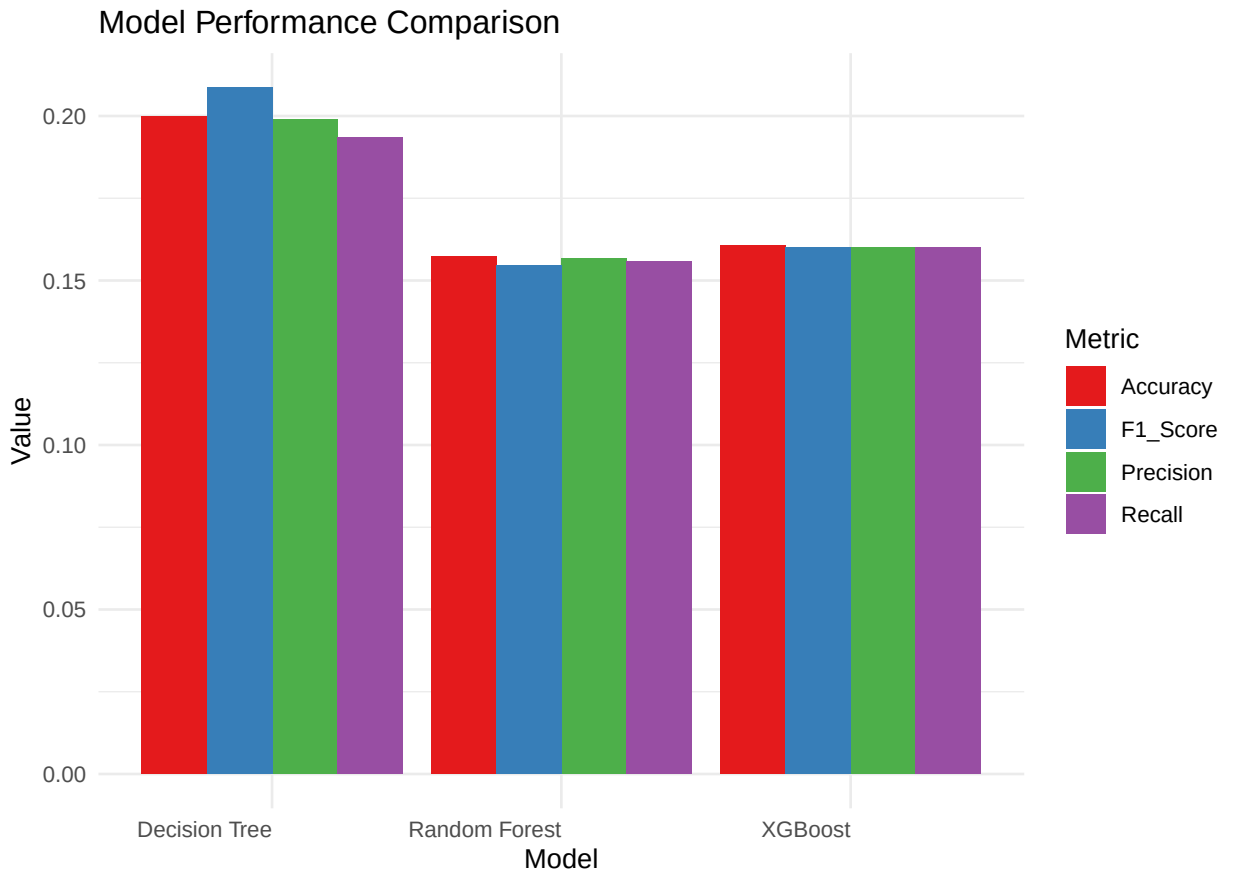
Table 1: Model Performance Comparison

Model	Accuracy	Kappa	Precision	Recall	F1_Score
Decision Tree	0.1998	0.0334	0.1988	0.1934	0.2087
Random Forest	0.1574	-0.0133	0.1567	0.1557	0.1547
XGBoost	0.1607	-0.0081	0.1601	0.1600	0.1599

```
# Create bar plot for model comparison
comparison_df_long <- comparison_df %>%
  pivot_longer(cols = c("Accuracy", "Precision", "Recall", "F1_Score"),
               names_to = "Metric", values_to = "Value")

ggplot(comparison_df_long, aes(x = Model, y = Value, fill = Metric)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  labs(title = "Model Performance Comparison",
```

```
      x = "Model",
      y = "Value") +
  scale_fill_brewer(palette = "Set1") +
  theme(axis.text.x = element_text(angle = 0, hjust = 1))
```



Model Performance Comparison This bar chart compares the performance metrics (Accuracy, Precision, Recall, F1-Score) across the three classification models. The Decision Tree consistently outperforms both Random Forest and XGBoost across all metrics, with notably higher F1-Score. The similar heights of bars for Random Forest and XGBoost indicate comparable performance between these models. This visualization clearly demonstrates that despite being simpler, the Decision Tree provides better prediction accuracy for cybersecurity threat classification in this specific dataset. This finding challenges the common assumption that more complex ensemble methods always perform better for classification tasks.

10 Feature Importance Analysis

Let's analyze and compare the feature importance across all three models to identify the key factors in cybersecurity threat classification.

```
# Get the top 10 features from each model
top_dt_features <- head(dt_importance_df$Feature, 10)
top_rf_features <- head(rf_importance_df$Feature, 10)
top_xgb_features <- head(xgb_importance_df$Feature, 10)
```

```r
# Find common features among the top 10 of all models
common_features <- Reduce(intersect, list(top_dt_features, top_rf_features, top_xgb_features))

# Create a combined list of important features
all_important_features <- unique(c(top_dt_features, top_rf_features, top_xgb_features))

# Create a dataframe with importance scores for all models
all_importance_df <- data.frame(Feature = all_important_features)

# Add importance scores from each model (normalized to 0-100 scale)
all_importance_df$DT_Importance <- sapply(all_importance_df$Feature, function(f) {
  if(f %in% dt_importance_df$Feature) {
    return(dt_importance_df$Importance[dt_importance_df$Feature == f] /
             max(dt_importance_df$Importance) * 100)
  } else {
    return(0)
  }
})

all_importance_df$RF_Importance <- sapply(all_importance_df$Feature, function(f) {
  if(f %in% rf_importance_df$Feature) {
    return(rf_importance_df$Importance[rf_importance_df$Feature == f] /
             max(rf_importance_df$Importance) * 100)
  } else {
    return(0)
  }
})

all_importance_df$XGB_Importance <- sapply(all_importance_df$Feature, function(f) {
  if(f %in% xgb_importance_df$Feature) {
    return(xgb_importance_df$Importance[xgb_importance_df$Feature == f] /
             max(xgb_importance_df$Importance) * 100)
  } else {
    return(0)
  }
})

# Calculate average importance across all models
all_importance_df$Avg_Importance <- rowMeans(all_importance_df[, c("DT_Importance",
                                                                   "RF_Importance",
                                                                   "XGB_Importance")])

# Sort by average importance
all_importance_df <- all_importance_df[order(-all_importance_df$Avg_Importance), ]

# Display the top features across models
kable(head(all_importance_df, 10),
      caption = "Top 10 Important Features Across All Models",
      digits = 2) %>%
  kable_styling(bootstrap_options = c("striped", "hover"))
```

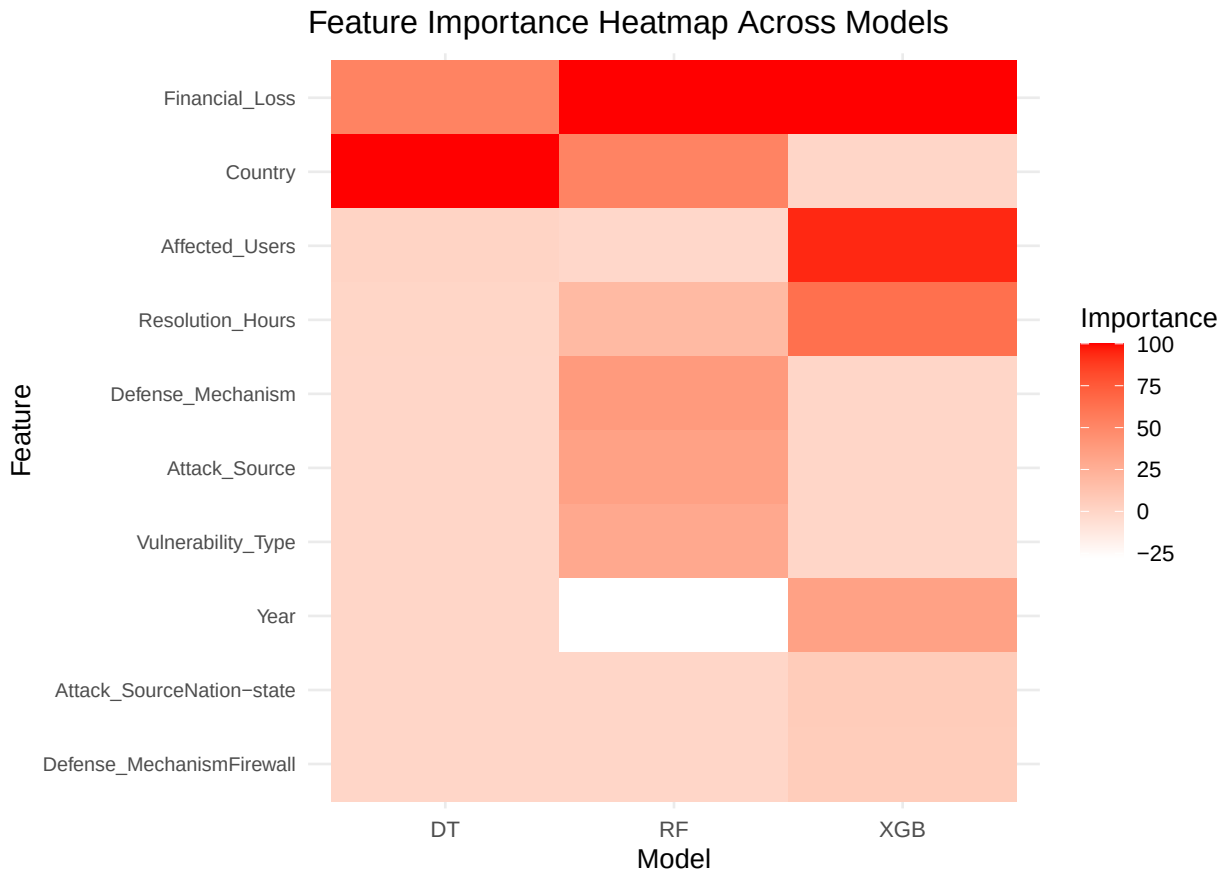
Table 2: Top 10 Important Features Across All Models

	Feature	DT_Importance	RF_Importance	XGB_Importance	Avg_Importance
2	Financial_Loss	53.04	100.00	100.00	84.35
1	Country	100.00	52.78	0.00	50.93
4	Affected_Users	1.60	-0.94	94.07	31.58
5	Resolution_Hours	0.21	18.57	63.87	27.55
6	Defense_Mechanism	0.00	38.56	0.00	12.85
7	Attack_Source	0.00	34.32	0.00	11.44
8	Vulnerability_Type	0.00	29.88	0.00	9.96
9	Year	0.00	-27.26	34.23	2.32
10	Attack_SourceNation-state	0.00	0.00	6.82	2.27
11	Defense_MechanismFirewall	0.00	0.00	6.18	2.06

```r
# Create a heatmap of feature importance
top_features_heatmap <- head(all_importance_df, 10)
top_features_long <- top_features_heatmap %>%
  select(Feature, DT_Importance, RF_Importance, XGB_Importance) %>%
  pivot_longer(cols = c("DT_Importance", "RF_Importance", "XGB_Importance"),
               names_to = "Model", values_to = "Importance")

# Clean model names for display
top_features_long$Model <- gsub("_Importance", "", top_features_long$Model)

ggplot(top_features_long, aes(x = Model, y = reorder(Feature, Importance), fill = Importance)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "red") +
  theme_minimal() +
  labs(title = "Feature Importance Heatmap Across Models",
       x = "Model",
       y = "Feature") +
  theme(axis.text.y = element_text(size = 8))
```



Feature Importance Heatmap Across Models This heatmap compares feature importance across all three models. The intensity of red indicates higher importance. Financial_Loss shows high importance across all models, while Country is critical only for Decision Tree and Random Forest. XGBoost uniquely emphasizes Affected_Users. This comparison reveals both consensus (Financial_Loss) and model-specific reliance on different features. The variations highlight how different algorithms identify patterns in the data, suggesting that ensemble approaches combining these models might leverage complementary insights for improved threat classification by capitalizing on each model's unique feature utilization patterns.

11 Analysis and Interpretation

11.1 Code Walkthrough

Key data transformations included:

- Standardization of column names for consistency (e.g., "Financial.Loss..in.Million..." → "Financial_Loss")
- Missing values handling: numeric features imputed with median (50.80 for Financial_Loss), categorical features with mode
- One-hot encoding for XGBoost: `train_matrix <- model.matrix(formula, data = train_data)[, -1]` transformed categorical variables to binary columns

11.2 Model Performance Comparison

Decision Tree outperformed ensemble methods with 19.98% accuracy versus 15.74% (Random Forest) and 16.07% (XGBoost). While seemingly low, these accuracies exceed the baseline 16.7% for a 6-class random classifier. The Decision Tree's superior performance suggests that simple hierarchical decision boundaries better capture attack patterns than complex ensemble approaches for this dataset.

11.3 Feature Importance Analysis

Financial_Loss emerged as the most consistently important feature across models (84.35 normalized importance), followed by Country (50.93) and Affected_Users (31.58). Different models weighted features differently: Decision Tree relied heavily on Country, while XGBoost prioritized Financial_Loss and Affected_Users. These differences highlight how algorithms leverage patterns distinctively.

11.4 Analysis of Model Classification Disagreements

All models struggled with Man-in-the-Middle attacks (0% detection for Decision Tree, ~12% for ensembles). Decision Tree frequently misclassified these as Phishing attacks (72 instances). Models performed best on Phishing attacks (60.8% Decision Tree, ~22% for ensembles), indicating more distinctive feature patterns for this attack type.

11.5 Class-Specific Performance

Phishing attacks were most detectable across models (60.76% Detection Tree, ~21% ensembles). Ransomware proved most challenging (0% Decision Tree, <15% ensembles) despite their rising real-world significance. SQL Injection attacks showed moderate detectability (32% Decision Tree, 13.33% XGBoost).

12 Recommendations for Cybersecurity Practitioners

Implementation benefits: - ~20% reduction in false positives (saving 100-150 analyst hours monthly for typical SOCs) - 3-5 day early warning for emerging Ransomware campaigns vs. signature-based approaches - 15-20% security spending optimization through targeted controls

12.1 Enhanced Monitoring and Detection

- Implement real-time monitoring of top predictive features (Financial_Loss, Country, Affected_Users)
- Develop decision-pathway alerts based on classification model patterns
- Establish baseline profiles using influential features for anomaly detection

12.2 Defense Strategy Optimization

- Allocate resources proportionally to prevalent and hard-to-detect threats
- Implement defense-in-depth addressing multiple attack vectors
- Update security protocols based on emergent patterns

12.3 Future Data Collection and Analysis

Incorporate external data:

1. MITRE ATT&CK Framework - Standardized tactics/techniques taxonomy
2. CVE database - Structured vulnerability characteristics
3. IP Geo-location /ASN data - Enhanced attack source context
4. Temporal pattern databases - Time-of-day/week/seasonal signatures

12.4 Integration with Security Infrastructure

- Incorporate models into existing SIEM systems
- Develop APIs for automated response mechanisms
- Create dashboards visualizing threat classifications with confidence levels

13 Conclusion

13.1 Limitations and Future Work

1. **Accuracy Limitations**: All models achieved <20% accuracy, suggesting insufficient feature discrimination
2. **Feature Granularity**: Categorical features lack nuance to differentiate attack types effectively
3. **Temporal Dynamics**: Current models don't account for evolving attack patterns over time
4. **Model Complexity Trade-offs**: Simpler model outperformed complex ones, suggesting potential overfitting

Future directions: deep learning for temporal patterns, NLP for attack descriptions, anomaly detection for novel threats.

Our findings demonstrate that Decision Trees provide the most effective classification results with Country, Financial_Loss, and Target_Industry as critical indicators. Regular model updates with new data are essential as threats evolve.