# Random Forest

Anju Sambasivan

2024-10-05

# Introduction

## Random Forest Theory and How It Works

Random Forest is a popular ensemble learning algorithm, primarily used for classification and regression tasks. It builds upon the concept of Decision Trees by creating a collection (or "forest") of trees, each built on random subsets of the data. The final prediction is determined by aggregating the predictions of all individual trees in the forest.

### Key Concepts:

- Ensemble Learning: Random Forest is an ensemble method, meaning it combines the results of multiple models (in this case, decision trees) to make a prediction. The idea is that a group of weak learners (trees) can come together to form a strong learner, making more accurate predictions.
- Decision Trees: A decision tree splits the data based on feature values to classify or predict an outcome. However, decision trees alone can overfit (fit too closely to the training data) and perform poorly on unseen data. Random Forest mitigates this issue by using multiple trees.
- Randomness in Random Forest: Random Forest introduces two types of randomness to build a diverse set of trees:
- Bootstrap Sampling: Each tree is trained on a random subset of the original data, selected with replacement. This is called bagging (Bootstrap Aggregating).
- Random Feature Selection: At each split in the tree, only a random subset of the features is considered. This prevents the trees from all looking too similar.
- Majority Voting (for classification) or Averaging (for regression): In classification problems (like victimization data), each tree in the forest votes for a class. The class with the majority of votes is selected as the final prediction. In regression problems, the average prediction from all trees is taken.

## How Random Forest Works Step by Step:

1. Data Preparation: The dataset is divided into features (independent variables) and the target (dependent variable). The dataset is randomly sampled multiple times (with replacement) to create different training subsets for each tree.
2. Building Trees: For each subset of data:

- A decision tree is trained, but at each split, only a random subset of the features is used to determine the best split.
- The decision tree is grown to a maximum depth or until it meets certain stopping criteria (e.g., minimum number of samples in a leaf node).

3. Prediction (for classification):

- Each tree in the forest predicts a class for the input data.
- The final prediction is determined by majority vote: the class that receives the most votes from the trees is selected as the model's output.

4. Out-of-Bag Error: During the training process, each tree is built using a subset of the data. The samples that are not used to train a particular tree are called "out-of-bag" (OOB) samples. The OOB samples can be used to estimate the error of the model without needing a separate validation dataset. This provides an unbiased estimate of the model's performance.

### Why Random Forest is Effective:

1. Reduces Overfitting: Since each tree is built on a different random subset of data and considers only a subset of features, the model becomes more robust and less likely to overfit.
2. Handles High Dimensional Data: Random Forest works well even when there are many features, making it useful for complex datasets.
3. Handles Missing Data: Random Forest can handle missing data by making decisions at each node without requiring all data points.
4. Variable Importance: Random Forests can provide insights into which features are most important for prediction. This is done by looking at how much the accuracy decreases when a particular feature is removed.

## Implementation

- Date: 1/01/2024 to 1/05/2024
- Total data (data_clean): 49,332 observations (rows).
- Training data (train_data): 34,535 observations (approximately 70% of the total data).
- Testing data (test_data): 14,797 observations (approximately 30% of the total data).

```
library(dplyr)
library(randomForest)
library(caret)
```

- dplyr: For data manipulation tasks such as filtering, selecting, and modifying data.
- randomForest: To implement the Random Forest algorithm.
- caret: For data partitioning and model evaluation (including confusion matrix creation).

```
data <- read.csv("data2.csv")
```

- read.csv(): This function used to read CSV files

```
data$AgeGroup <- as.factor(data$AgeGroup)
data$LocationType <- as.factor(data$LocationType)
data$ROVDivision <- as.factor(data$ROVDivision)
data$TerritorialAuthority <- as.factor(data$TerritorialAuthority)
data$PersonOrOrganization <- as.factor(data$PersonOrOrganization)
data$AnzsocDivision <- as.factor(data$AnzsocDivision)
```

- as.factor(): Converts columns with categorical data (e.g., AgeGroup, LocationType) into factor data types. This is necessary because the Random Forest algorithm requires categorical variables to be treated as factors.

```
str(data)
```

- str(data): Prints the structure of the dataset, displaying each column's data type and the first few values. It is useful for checking if the data types are correctly set.

```
data$Date <- as.Date(data$Date)
```

- as.Date(): Converts the Date column into a proper date format. This is important if the Date column is stored as a string or numeric format and you want to perform operations based on date.

```
# Remove missing values
data_clean <- na.omit(data)
```

- na.omit: Removes rows from the dataset that contain missing (NA) values. This step is essential to prevent errors during model training.

```
# Split the data into training and testing sets
set.seed(123)  # For reproducibility
trainIndex <- createDataPartition(data_clean$AnzsocDivision, p = 0.7, list = FALSE)
train_data <- data_clean[trainIndex, ]
test_data <- data_clean[-trainIndex, ]
```

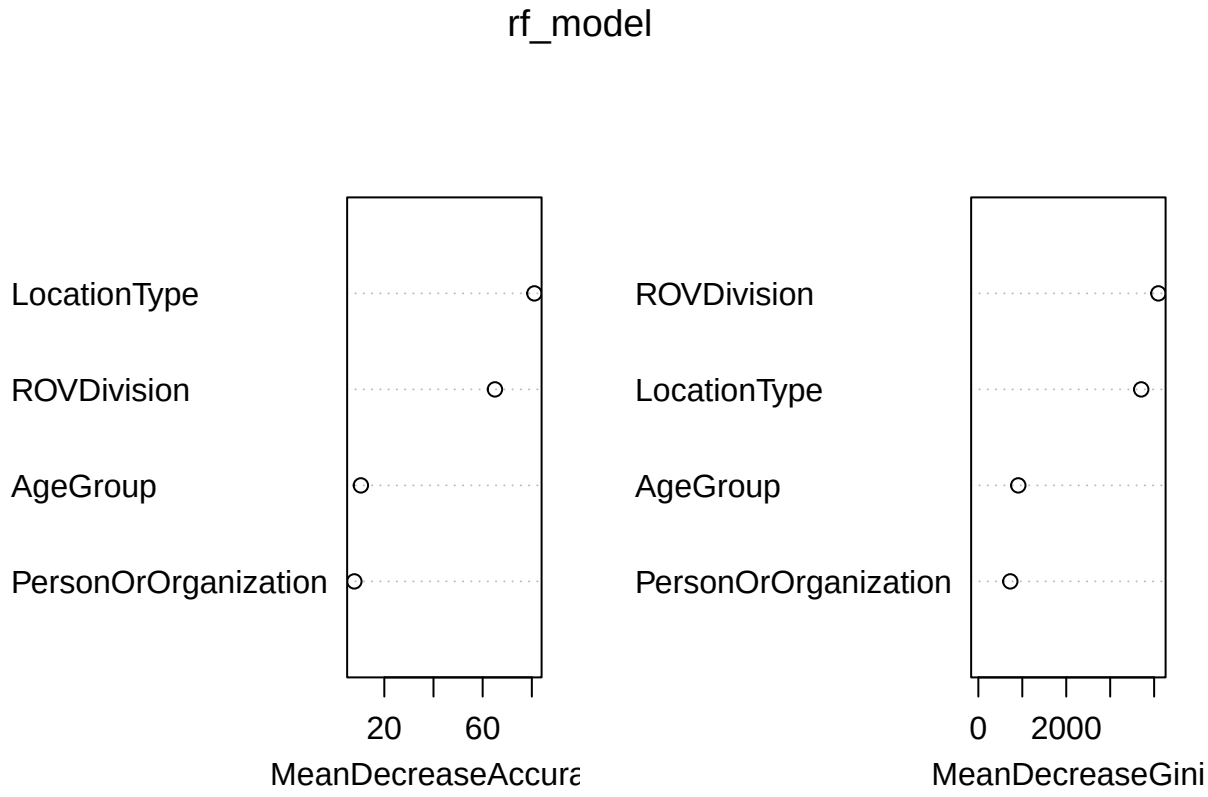
- set.seed(123): Ensures reproducibility. Using the same seed guarantees that the random operations produce the same result each time the code is run.
- createDataPartition: Splits the dataset into a training set (70%) and a testing set (30%) based on the - AnzsocDivision column (target variable). The result is stored in trainIndex.
- train_data <- data_clean[trainIndex, ]: Selects 70% of the data for training.
- test_data <- data_clean[-trainIndex, ]: Selects the remaining 30% for testing.

```
# Train a Random Forest model
rf_model <- randomForest(AnzsocDivision ~ AgeGroup + LocationType + ROVDivision + PersonOrOrganization,
                         data = train_data,
                         ntree = 100,
                         mtry = 3,
                         importance = TRUE)
# ntree = 100,  # Number of trees
# mtry = 3,  Number of variables randomly sampled as candidates at each split
```

- randomForest: Trains a Random Forest model using the specified features (AgeGroup, LocationType, ROVDivision, and PersonOrOrganization) to predict the target variable (AnzsocDivision).
- ntree = 100: The number of decision trees to grow in the forest.
- mtry = 3: The number of features randomly selected at each split.
- importance = TRUE: Enables calculation of feature importance, which will show which variables contributed most to the predictions.

```
# Print the random forest model summary
print(rf_model)
```

```
# Variable importance plot
varImpPlot(rf_model)
```




- varImpPlot: Plots the importance of each feature in the Random Forest model. This helps to identify which features contribute the most to the predictions.
- In the Mean Decrease Gini plot in Random Forest analysis, a higher mean decrease in Gini indicates that a variable contributes more to making nodes purer, which makes it more important for classifying the data correctly.

Mean Decrease in Accuracy (left plot): This measures how much the model accuracy decreases if a particular feature is removed. The higher the value, the more important the feature is in maintaining the overall model accuracy.

Mean Decrease in Gini (right plot): This measures how much each feature contributes to the model by decreasing the Gini impurity. The higher the value, the more important the feature is in splitting the data and reducing the classification error.

- This plot shows the variable importance from the Random Forest model based on two metrics:

1. Mean Decrease in Accuracy (Left Plot):

- This plot shows how much the accuracy of the model decreases when the values of a specific variable are permuted (randomized).
- Higher values indicate that the variable is important in improving model accuracy.
- For example, the variable LocationType has the highest mean decrease in accuracy, meaning it's the most influential feature in predicting the target variable (AnzsocDivision).
- Other variables like ROVDivision and AgeGroup also contribute to the model's accuracy, but to a lesser extent.

2. Mean Decrease in Gini (Right Plot):

- This plot indicates how much each variable contributes to the purity of the nodes in the trees in the random forest.
- A higher Gini decrease means the variable helps make more accurate splits in the decision trees.

ROVDivision has the highest mean decrease in Gini, making it the most important variable for node splits in this dataset.

- Other features such as LocationType and AgeGroup also show some contribution, but PersonOrOrganization has less importance compared to the other variables.

**Interpretation:**

1. ROVDivision appears to be the most critical feature in the dataset, playing a key role in predicting the type of crime (AnzsocDivision).
2. LocationType and AgeGroup are moderately important, while PersonOrOrganization has less impact on both accuracy and Gini impurity reduction.

This information helps understand which features are most influential in predicting the crime category, and it can guide further analysis or feature selection if needed.

```
# Predict on the test set
rf_predictions <- predict(rf_model, newdata = test_data)
```

- predict: Makes predictions on the test set using the trained Random Forest model.

```
# Evaluate the model using a confusion matrix
conf_matrix_rf <- confusionMatrix(rf_predictions, test_data$AnzsocDivision)
print(conf_matrix_rf)
```

- confusionMatrix: Computes a confusion matrix to evaluate the model's performance. It compares the predicted values (rf_predictions) with the actual values (test_data$AnzsocDivision), showing how well the model performs on each class.

```
# Convert confusion matrix to a matrix if not already
conf_matrix_mat <- as.matrix(conf_matrix_rf$table)
```

- as.matrix(conf_matrix_rf$table): Converts the confusion matrix from a table format to a matrix, which may be useful for further analysis or visualization.

```
print(conf_matrix_mat)
```

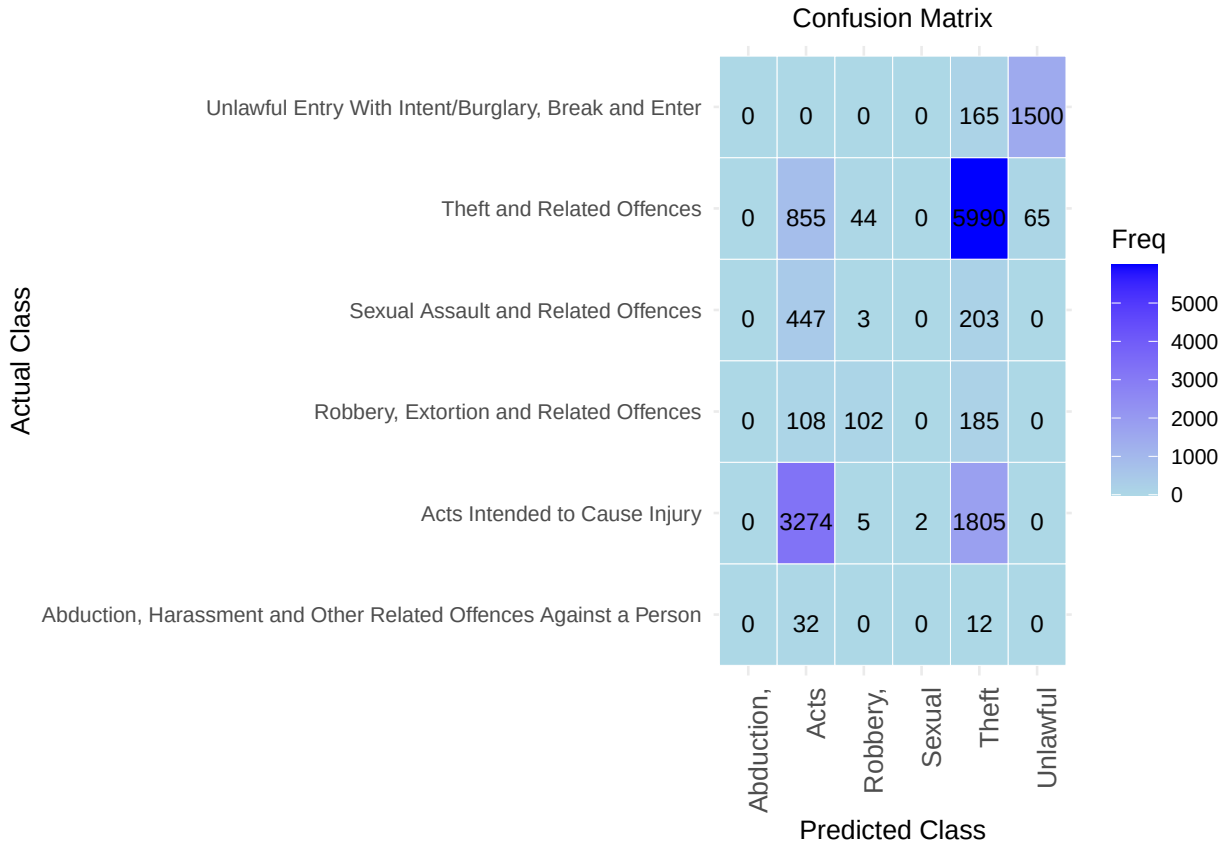
```
# Confusion matrix object is created
conf_matrix_rf <- confusionMatrix(rf_predictions, test_data$AnzsocDivision)

# Convert the confusion matrix into a data frame for ggplot
conf_matrix_df <- as.data.frame(conf_matrix_rf$table)  # Create conf_matrix_df here

# Extract the first word from the Prediction labels
conf_matrix_df$Prediction <- sapply(strsplit(as.character(conf_matrix_df$Prediction), " "), `[`, 1)

# Plot the confusion matrix using ggplot
ggplot(conf_matrix_df, aes(Prediction, Reference)) +
  geom_tile(aes(fill = Freq), colour = "white") +
  scale_fill_gradient(low = "lightblue", high = "blue") +
  labs(x = "Predicted Class", y = "Actual Class", title = "Confusion Matrix") +
  theme_minimal(base_size = 10) +  # Increase the overall base size of the plot
  geom_text(aes(label = Freq), vjust = 1, size = 3) +  # Increase size of text in cells
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 10),  # Increase and rotate x-axis lab
```

```
        axis.text.y = element_text(size = 8),  # Reduce y-axis label size
        plot.title = element_text(size = 10, hjust = 0.5))  # Increase title size and center it
```




## Interpretation of the Confusion Matrix:

### Correct Classifications (Diagonal Elements):

The diagonal elements represent cases where the actual class and the predicted class match, meaning correct predictions.

While the model performs well for some categories, significant misclassifications between some categories suggest that the model could benefit from additional tuning, better feature selection, or more training data to improve its ability to differentiate between these similar categories.

### Misclassifications (Off-Diagonal Elements):

The off-diagonal elements represent cases where the model predicted the wrong class.

```
# Calculate accuracy
accuracy <- conf_matrix_rf$overall['Accuracy']
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.734338041494898"
```

- accuracy <- conf_matrix_rf$overall[‘Accuracy’]: Extracts the overall accuracy of the model from the confusion matrix. print(paste(“Accuracy:”, accuracy)): Prints the accuracy of the model.

This model is moderate in performance, showing reasonable accuracy, but it may need further tuning or feature engineering to improve performance, particularly in classifying certain categories more accurately.

Exploring more advanced techniques, tuning hyperparameters, or rebalancing the dataset could improve the results.

**Real World Scenario**

In a real-world application, the Random Forest model using on victimization data is designed to predict the type of crime (AnzsocDivision) based on several characteristics such as:

- AgeGroup: The age group of the victim or person involved in the crime.
- LocationType: The type of location where the crime occurred, such as a public place or a private residence.
- ROVDivision: Information about the relationship between the offender and the victim (e.g., whether the crime was committed by a stranger or someone known to the victim).
- PersonOrOrganization: Whether the victim was a person or an organization.

**What is the Model Predicting?**

The Random Forest model is trained to classify crimes into specific categories (like "Theft", "Assault", etc.) based on these features. After training, it will predict what type of crime occurred given similar data in new cases.

**Example:**

**Input Features:**

- AgeGroup: 20+
- LocationType: Street/Footpath
- ROVDivision: Stranger
- PersonOrOrganization: Person

Output Prediction: Based on these details, the model might predict that the crime was "Acts Intended to Cause Injury" or "Theft and Related Offenses."

**How Could This Model Be Used in Practice?**

1. Predict crime types based on available data from police reports: If a report includes details about the victim's age, location, and whether the offender is known, the model can help predict the most likely crime category.
2. Identify crime trends: Law enforcement agencies could analyze victimization patterns based on locations, victim ages, and other features to target crime prevention efforts.
3. Assist investigations: When there's uncertainty about the crime type, the model can provide predictions to guide investigators in the right direction.

**Real-World Limitations:**

1. Accuracy depends on data quality: The model's predictions rely on the quality and completeness of the input data. If the data is inaccurate or biased, the model's predictions may be unreliable.
2. Human judgment is still essential: The model's predictions should be treated as supporting information. Final decisions, especially in sensitive areas like law enforcement, still require expert judgment and legal oversight.

This application could help police or crime analysts in predicting crime types and identifying trends for better resource allocation or prevention strategies.

**Notes**

- Gini Impurity is a measure used to evaluate how well a dataset is split when building decision trees (which are part of the random forest model). It tells us how "mixed" the data is in terms of categories.
- Low Gini (closer to 0) means the data in that split is mostly of one type, meaning it's "pure." High Gini (closer to 1) means the data is very mixed and not well-separated into different categories.

Consider a basket of fruits. We have apples and oranges, and you want to separate them into different groups:

- If one group has only apples and the other has only oranges, the groups are very pure, and the Gini score would be close to 0.
- But if both groups have a mix of apples and oranges, the groups are less pure, and the Gini score would be closer to 1.
- Gini Impurity helps the model decide where to split the data. The goal is to reduce the Gini score at each step, meaning the data gets better separated into categories.
- The Mean Decrease in Gini plot (in our random forest results) shows which variables (like AgeGroup or LocationType) help make these pure splits. The higher the value, the more important that variable is in separating the data.

1. Mean Decrease Accuracy (Left Plot):

- This plot shows how much accuracy is lost when a particular variable is removed from the model.
- The x-axis values represent the importance of each variable in predicting the outcome (crime type).
- A higher value means that the variable is more important for the model's accuracy.
- Example: ROVDivision has the highest value on the x-axis, meaning it is the most important variable in improving the accuracy of the model.

2. Mean Decrease Gini (Right Plot):

- This plot shows the importance of each variable in splitting the data and reducing uncertainty (impurity) during the construction of decision trees in the Random Forest.
- The x-axis represents the Gini score, which measures how well a variable separates the data into categories.
- A higher Gini value means the variable is important in making pure (clear) splits.
- Again, ROVDivision has the highest Gini value, indicating it helps the model make better splits for classifying crimes.