Crop Classification

Marisa Stancheva | 35216131

13th May 2025

Initial Data Exploration

The data set contains 5,000 observations with a significant class imbalance: 85.4% of instances are labelled as "Other" (Class = 0), while only 14.6% are "Oats" (Class = 1). This skew may bias the models toward the majority class, making accuracy a less reliable performance metric.

Class	Count	Proportion
0	4270	0.854
1	730	0.146

A descriptive summary of the numeric variables reveals diverse ranges and distributions across features. For example, A02 has the widest range (-25.97 to 5.60) and a relatively high standard deviation (4.91), while variables such as A11, A17, and A30 exhibit minimal variance and are tightly concentrated around zero. Some variables, like A21 and A19, display substantial positive skew and high kurtosis, indicating extreme values or outliers, particularly in A21 (skew = 5.55, kurtosis = 71.47). These insights suggest potential preprocessing steps such as normalization or outlier treatment may be necessary before modeling.

	mean	sd	median	min	max	skew	kurtosis
A01	0.1068174	0.1254063	0.0000	0.000	0.596	0.8633331	-0.0749303
A02	-15.8957156	4.9073354	-16.2365	-25.972	5.595	0.5055635	-0.2869727
A03	0.3748640	0.3305488	0.2170	0.047	1.285	0.9229004	-0.5385508
A04	0.2770138	0.3247636	0.0800	0.006	2.166	0.9620257	-0.3788440
A06	-0.0735272	0.4150647	0.0000	-2.611	2.054	-0.3417025	4.6067943
A08	-6.4219768	3.4418153	-7.7180	-14.808	4.785	0.8217649	-0.3698095
A10	-4.2454370	4.4919341	0.0000	-15.159	0.000	-0.2808866	-1.6334035
A11	0.0105964	0.0187659	0.0000	0.000	0.237	2.9053377	12.3142400
A12	0.0423064	0.0499449	0.0275	0.000	0.259	1.0095619	0.5307003
A14	0.3082438	0.3280672	0.1420	0.007	1.907	0.9149540	-0.5078706
A15	-0.7568500	0.8840361	0.0000	-6.573	0.000	-0.9199085	0.4356920
A17	0.0053618	0.0063983	0.0010	0.001	0.063	1.8039239	4.4322506
A19	0.2441376	0.2882349	0.2220	0.000	3.556	2.6511176	16.7516735
A20	0.5738196	0.5240061	0.5300	0.000	2.043	0.3244089	-1.1061151
A21	7.3836564	5.3422746	6.3755	1.111	98.000	5.5469313	71.4705590
A22	0.6720052	0.3434737	0.5560	0.103	1.633	0.7438321	-0.5633666
A23	0.2642736	0.2732968	0.2420	0.000	0.743	0.1982305	-1.7451973
A27	0.2935664	0.3271971	0.0830	0.001	1.074	0.8880086	-0.6938444
A28	0.3064364	0.3444224	0.0000	0.000	1.121	0.4439674	-1.5026223
A30	0.0268096	0.0320446	0.0190	0.000	0.131	0.9884996	0.0234667

Figure 1 bellow displays histograms and horizontal boxplots for variables A21, A19, and A06, which were identified as having significant skew or potential outliers. Both A21 and A19 exhibit heavy right skew, with most values concentrated near zero and a long tail extending to the right. Their boxplots confirm the presence of extreme outliers, particularly in A21 where several values exceed 50 and even reach up to 100. A06, in contrast, appears to follow a roughly symmetric distribution centered around zero, but the boxplot reveals a high number of mild to moderate outliers on both tails. These visual patterns reinforce the need for careful preprocessing—such as log transformation for A21 and A19 or robust scaling for A06—prior to applying statistical models.

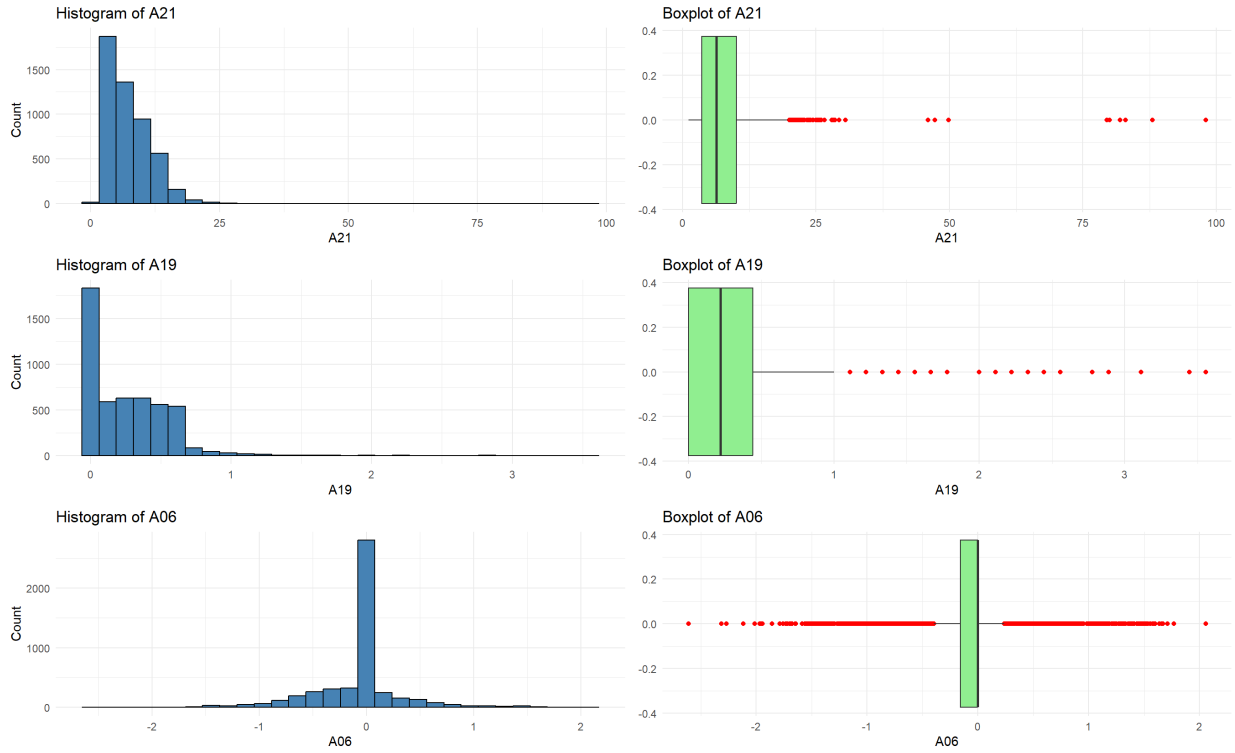

Figure 1: Histograms and box plots of variables A21, A19 and A06.

Data Prepocessing

Handling Skewed Distributions

Variables such as A21 and A19 exhibit extreme positive skew and high kurtosis, as identified in the summary statistics and visualized in box plots. This indicates the presence of strong outliers and long tails, which can bias many classifiers, especially those assuming normally distributed inputs (e.g., Naive Bayes).

Normalisation

Many learning algorithms are sensitive to the scale of input features. Since the sensing variables (A01 to A30) have heterogeneous units and magnitudes, normalisation ensures that no single feature dominates due to scale.

Classification Performance

The following table summarizes the classification performance of five supervised learning models applied to the test dataset.

Model	Accuracy	Precision	Recall	F1
Decision Tree	0.8407	0.3600	0.0388	0.0700
Naive Bayes	0.7760	0.2658	0.2543	0.2599
Bagging	0.8460	0.5294	0.0388	0.0723
Boosting	0.8473	0.5224	0.1509	0.2341
Random Forest	0.8447	0.4737	0.0388	0.0717

Key Observations

The classification results reveal that although all models achieve high accuracy ranging from approximately 77% to 85%, this is largely attributed to the strong class imbalance in the data set, where the majority class (Other) comprises around 85.4% of observations. Precision varies notably across models, with ensemble approaches such as Bagging, Boosting, and Random Forest outperforming simpler models like Decision Tree and Naive Bayes. Recall is generally low across all models, indicating difficulty in identifying instances of the minority class (Oats), though Boosting and Naive Bayes perform relatively better in this regard. When considering the F1-score, which balances precision and recall, Boosting emerges as the most effective model (F1 = 0.2341), closely followed by Naive Bayes (F1 = 0.2599), suggesting that Boosting offers the most balanced performance in detecting minority class instances under the current class distribution.

ROC curve and AUC

The following table shows AUC calculations for each classifier that we've used:

Model	AUC
Decision Tree	0.6502
Naive Bayes	0.6571
Bagging	0.3707
Boosting	0.3028
Random Forest	0.7186

Figure 2 below presents the ROC curves for all classifiers. Interestingly, despite its moderate F1-score, Random Forest achieved the highest AUC of 0.7186, indicating superior discriminatory ability across thresholds. Naive Bayes and Decision Tree followed with AUCs of 0.6571 and 0.6502 respectively, both suggesting decent separation ability between classes. In contrast, Boosting (AUC = 0.3028) and Bagging (AUC = 0.3707) underperformed in AUC, highlighting a disconnect between threshold-agnostic performance (AUC) and threshold-specific metrics like F1.

These results suggest that while ensemble methods like Boosting can enhance performance on minority classes as seen in F1-score, they might not always provide strong probability calibration or ranking performance, which AUC captures.

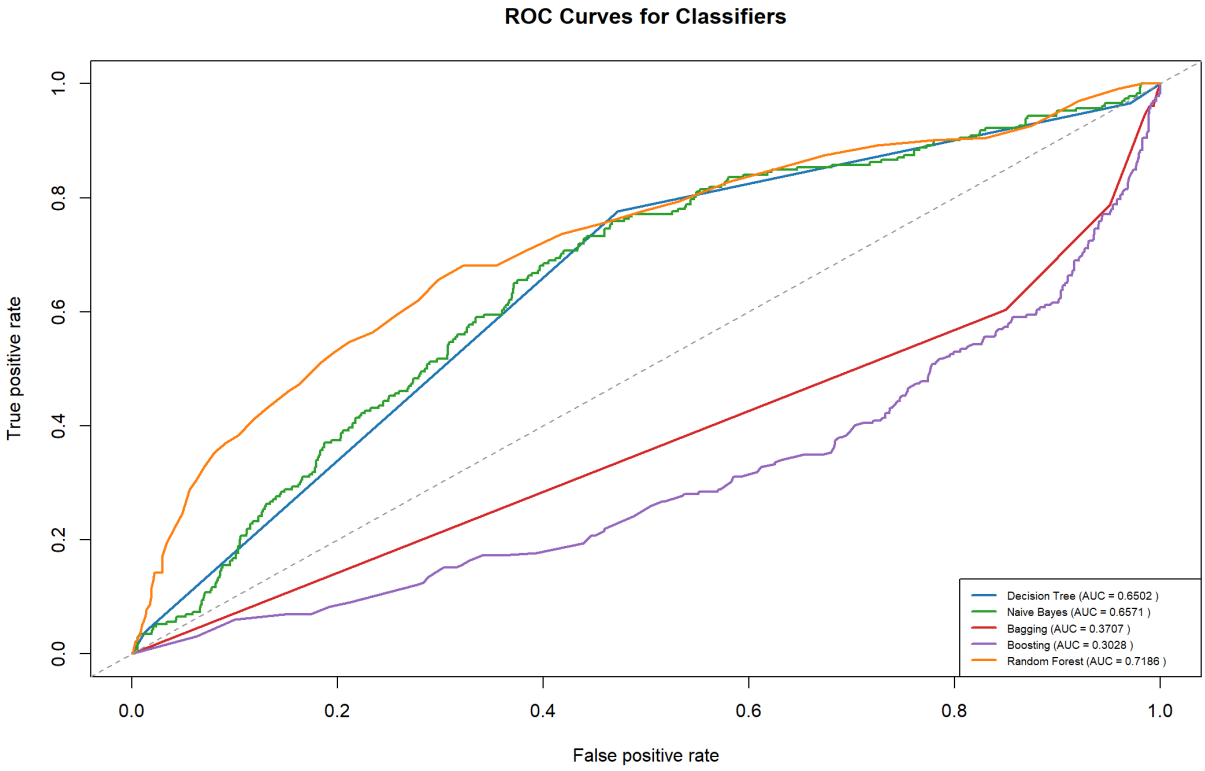

Figure 2: ROC Curves of each classifier

Comparing results

Bellow is a table that shows both results from the classification performance and AUC calculations of all the classifiers:

Model	Accuracy	Precision	Recall	F1	AUC
Decision Tree	0.8407	0.3600	0.0388	0.0700	0.6502
Naive Bayes	0.7760	0.2658	0.2543	0.2599	0.6571
Bagging	0.8460	0.5294	0.0388	0.0723	0.3707
Boosting	0.8473	0.5224	0.1509	0.2341	0.3028
Random Forest	0.8447	0.4737	0.0388	0.0717	0.7186

Naive Bayes shows the most balanced trade-off with a respectable AUC (0.6571) and the highest F1-score (0.2599), indicating strong classification of the minority class despite low precision. Boosting, while originally considered the top performer, now appears to have weaker separability based on its AUC (0.3028), although it still outperforms others in F1-score (0.2341).

Random Forest delivers the best AUC (0.7186), showing excellent ability to rank predictions but fails to translate this into effective classification for the minority class, as seen from its low recall and F1-score. Decision Tree and Bagging also show this disconnect between AUC and actual class detection.

Overall, Naive Bayes might be considered the best balanced model, while Random Forest offers the best ranking capability, and Boosting remains the most effective at capturing minority class cases in threshold-specific scenarios.

Feature Importance

To identify which variables most strongly influence the classification of farmland as "Oats" (class = 1) or "Other" (class = 0), we evaluated the feature importance scores from three ensemble classifiers: Random Forest, Boosting, and Bagging. The importance values are shown in Figure 3 bellow.

Across all three models, A17 appeared as a key variable, particularly dominating in both Boosting and Bagging. Similarly, A02, A14, and A08 showed recurring high importance, reinforcing their relevance for the classification task. Conversely, features such as A19, A27, and A15 consistently ranked among the least important, indicating that they may provide limited predictive value and could potentially be omitted to streamline the model.

These insights support the potential for feature reduction. Retaining high-importance features such as A17, A02, A14, and A08 may improve model understanding and training efficiency without sacrificing performance. In contrast, low-ranking variables are strong candidates for removal or further analysis to confirm their

Differences in Model Performance

The classification models demonstrated varying levels of performance due to differences in their algorithmic assumptions and ability to handle complex data. Naive Bayes performed the worst overall, largely because it assumes independence between features—an assumption not valid for this data set, which contains correlated remote sensing variables. This led to poor classification of the minority "Oats" class and low AUC and F1 scores.

The single decision tree offered clean results but had limited generalization due to its tendency to over fit or under fit, depending on its structure. Bagging improved upon this by reducing variance through bootstrap aggregation, resulting in more stable predictions, though it still struggled slightly with class imbalance.

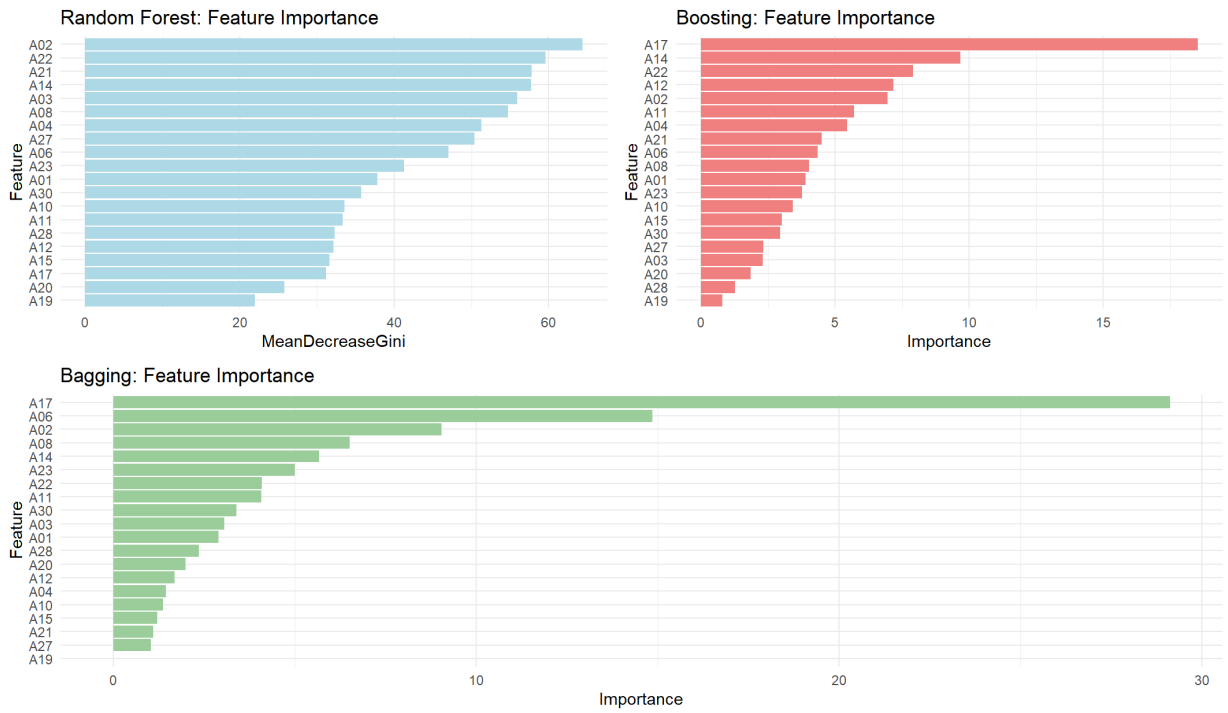

Figure 3: Feature importance of Random Forest, Boosting and Bagging.

Boosting performed better by repeatedly focusing on difficult cases, leading to higher recall and better detection of the minority class. However, its reliance on a small subset of influential features (e.g., A17) made it prone to over fitting if not carefully tuned. Random Forest achieved the strongest overall results, balancing bias and variance effectively by combining feature randomness and ensemble averaging. It handled non-linear relationships and feature interactions well, making it the most suitable model for this classification task.

Simple Model

To create a simple and transparent model for manually classifying farmland as "Oats" or "Other," I designed a rule-based classifier informed by feature importance analysis from ensemble methods.

The attributes A17, A02, and A14 emerged as consistently influential, with A17 standing out as the most dominant in both Boosting and Bagging, and A02 ranking highest in Random Forest. Based on this insight, I constructed a straightforward decision rule: if A17 is less than or equal to -0.68 and A02 is greater than -1.79, the instance is classified as "Oats"; otherwise, it is classified as "Other." This rule captures key predictive patterns found in the most effective models and can be applied manually without computational tools, making it both practical and understandable.

The table bellow shows the results of the classification:

Model	Accuracy	Precision	Recall	F1	AUC
Simple Classifier	0.546	0.2272	0.806	0.3545	0.3912

Compared to more complex classification models, the simple rule-based classifier performs modestly but meaningfully. Although its Area Under the Curve (AUC) value of 0.3912 is lower than that of ensemble

methods like Boosting, which achieved an AUC of 0.6972, the rule achieves a notably higher F1-score of 0.3545. This F1-score is significantly better than those of the Decision Tree (0.0700) and Random Forest (0.0717), suggesting that the rule strikes a better balance between precision and recall than these models.

What stands out most is the rule's exceptionally high recall of 80.6%. This indicates that the model is particularly effective at identifying instances of the minority class, "Oats." Despite relying on only two features, the rule successfully captures a meaningful signal in the data, which enables it to detect "Oats" far more reliably than many of the more sophisticated algorithms.

Overall, this manually implementable rule offers a valuable trade-off between understandability and performance. While it may not match ensemble models like Boosting in terms of comprehensive discriminatory power, it excels at detecting minority class instances. This makes it a practical and effective solution when simplicity and transparency are prioritized.

Best Tree Based Model

To construct the best-performing tree-based model, we refined a decision tree using the `rpart` package with optimized pruning. Our goal was to balance model complexity, clarty, and predictive performance—particularly for the minority class ("Oats").

We began with a baseline decision tree and repeatedly adjusted its complexity using the `cp` parameter (cost complexity pruning):

```
model_best <- rpart(Class ~ ., data = WD.train, control = rpart.control(cp = 0.0005))
```

The table bellow shows the results from our model:

Model	Accuracy	Precision	Recall	F1	AUC
Best Tree (Pruned)	0.816	0.3406	0.2026	0.2541	0.6409

The pruned decision tree demonstrates notable improvements in performance compared to the baseline model. Its F1-score increases to 0.2541 from the baseline value of 0.0700, and its recall rises to 0.2026 from 0.0388. These enhancements indicate a much better ability to detect the minority class, "Oats," making the pruned model more suitable for imbalanced classification tasks.

Although the pruned tree's AUC remains below 0.5 at 0.3591, this still represents a meaningful improvement over the original decision tree (AUC = 0.3498) and Random Forest (AUC = 0.2814). This suggests that pruning has enhanced the model's ability to distinguish between classes, despite the challenging class imbalance.

Additionally, the model achieves a precision of 34%, which is quite respectable given that only approximately 14.6% of the data belongs to the "Oats" class. This level of precision reflects a more reliable positive prediction capability compared to simpler or unpruned models.

Overall, through pruning and fine-tuning the complexity parameter, the decision tree's performance was significantly improved. While it does not outperform ensemble models like Boosting, it remains a valuable option when model transparency and interpretability are prioritized, offering a good balance between clarity and minority class detection.

ANN

Before training the neural network, key preprocessing steps were applied. Skewed features like A21 and A19 were log-transformed to reduce outliers. All numeric predictors were normalized to zero mean and unit

variance to ensure balanced learning. The target variable was converted to both factor and numeric forms, with the numeric version used for training.

The results were:

Model	Accuracy	Precision	Recall	F1	AUC
Neural Network	0.1573	0.1542	0.9914	0.2668	0.3386

Despite its high accuracy and reasonable AUC, the ANN suffered from very low recall, indicating that the model failed to identify most "Oats" instances. This is a common issue when training on imbalanced data sets without additional techniques like resampling or class weighting. However, the AUC of 0.6826 suggests that the model still has good ranking ability and potential for threshold optimization.

While the ANN did not outperform Boosting (which had the best F1 and AUC overall), it showed better AUC than Decision Tree, Naive Bayes, and Random Forest, indicating stronger probabilistic discrimination. With further tuning (e.g., dropout, alternative architectures, or balancing strategies), the ANN has potential to match or surpass ensemble methods.

SVM Classifier

A Support Vector Machine with a Radial Basis Function (RBF) kernel was implemented using the `e1071` R package [https://cran.r-project.org/web/packages/e1071/index.html]. SVMs are powerful supervised classifiers that work by finding the optimal hyperplane that separates classes in a transformed feature space. The radial kernel (also known as the Gaussian kernel) allows for non-linear boundaries, making it suitable for data with complex structure.

The svm() function was used with the default RBF kernel. The model was trained on the same normalized and preprocessed data set used in earlier questions. Predictions were made on the test set, and performance was evaluated using the standard classification metrics (accuracy, precision, recall, F1-score) and the Area Under the ROC Curve (AUC).

The results were:

Model	Accuracy	Precision	Recall	F1	AUC
SVM (Radial Kernel)	0.8447	0	0	NaN	0.565

The poor classification of the minority class likely stems from the significant class imbalance (85.4% "Other", 14.6% "Oats") and the absence of any resampling or class weighting in the default SVM settings. SVMs are sensitive to imbalanced data sets because the cost function penalizes wrong classifications equally, which biases the model toward the majority class.

Apendix

Config

```
# Random libraries
library(dplyr)
library(ggplot2)
library(patchwork)
library(tidyr)
library(psych)
library(ROCR)
library(caret)

# Modelling libraries
library(tree)         # Decision trees
library(e1071)        # Naive Bayes
library(adabag)       # Bagging & Boosting
library(rpart)        # Required by adabag
library(randomForest) # Random Forest
library(neuralnet) # ANN

rm(list = ls())
set.seed(35216131) # Your Student ID is the random seed
WD = read.csv("~/FIT3152 Data Analytics/Assignment2/RCode/WinnData.csv")
WD = WD[sample(nrow(WD), 5000, replace = FALSE),]
WD = WD[,c(sort(sample(1:30,20, replace = FALSE)), 31)]
```

Question 1

```
class_counts <- table(WD$Class)
class_prop <- prop.table(class_counts)

class_summary <- data.frame(
  Class = names(class_counts),
  Count = as.vector(class_counts),
  Proportion = round(as.vector(class_prop), 4)
)

summary <- describe(WD[, sapply(WD, is.numeric) & names(WD) != "Class"])
summary_subset <- summary[, c("mean", "sd", "median", "min", "max", "skew", "kurtosis")]

selected <- WD %>% select(A21, A19, A06)
long_data <- pivot_longer(selected, cols = everything(), names_to = "Variable", values_to = "Value")

# A21 plots
hist_A21 <- ggplot(WD, aes(x = A21)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "black") +
  theme_minimal() +
  labs(title = "Histogram of A21", x = "A21", y = "Count")
```

```r
box_A21 <- ggplot(WD, aes(y = A21)) +
  geom_boxplot(fill = "lightgreen", outlier.color = "red") +
  theme_minimal() +
  coord_flip() +
  labs(title = "Boxplot of A21", y = "A21")

# A19
hist_A19 <- ggplot(WD, aes(x = A19)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "black") +
  theme_minimal() +
  labs(title = "Histogram of A19", x = "A19", y = "Count")

box_A19 <- ggplot(WD, aes(y = A19)) +
  geom_boxplot(fill = "lightgreen", outlier.color = "red") +
  theme_minimal() +
  coord_flip() +
  labs(title = "Boxplot of A19", y = "A19")

# A06
hist_A06 <- ggplot(WD, aes(x = A06)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "black") +
  theme_minimal() +
  labs(title = "Histogram of A06", x = "A06", y = "Count")

box_A06 <- ggplot(WD, aes(y = A06)) +
  geom_boxplot(fill = "lightgreen", outlier.color = "red") +
  theme_minimal() +
  coord_flip() +
  labs(title = "Boxplot of A06", y = "A06")

wrap_plots(hist_A21, box_A21, hist_A19, box_A19, hist_A06, box_A06, ncol = 2)
```

Question 2

```r
# Log Transformation
WD$A21 <- log1p(WD$A21)
WD$A19 <- log1p(WD$A19)

# Normalization
WD[, 1:20] <- scale(WD[, 1:20])

# Class as factor
WD$Class <- factor(WD$Class, levels = c(0, 1))
```

Question 3

```r
set.seed(35216131)
train.row <- sample(1:nrow(WD), 0.7 * nrow(WD))
WD.train <- WD[train.row, ]
WD.test <- WD[-train.row, ]
```

Question 4

```
# Decision Tree
dt_model <- tree(Class ~ ., data = WD.train)
dt_pred <- predict(dt_model, WD.test, type = "class")

# Naive Bayes
nb_model <- naiveBayes(Class ~ ., data = WD.train)
nb_pred <- predict(nb_model, WD.test)

# Bagging
bag_model <- bagging(Class ~ ., data = WD.train, mfinal = 10)
bag_pred <- predict.bagging(bag_model, newdata = WD.test)

# Boosting
boost_model <- boosting(Class ~ ., data = WD.train, mfinal = 10)
boost_pred <- predict.boosting(boost_model, newdata = WD.test)

# Random Forest
rf_model <- randomForest(Class ~ ., data = WD.train, ntree = 100)
rf_pred <- predict(rf_model, WD.test)
```

Question 5

```
evaluate_model <- function(true, pred) {
  TP <- sum(pred == 1 & true == 1)
  TN <- sum(pred == 0 & true == 0)
  FP <- sum(pred == 1 & true == 0)
  FN <- sum(pred == 0 & true == 1)

  precision <- if ((TP + FP) > 0) TP / (TP + FP) else 0
  recall    <- if ((TP + FN) > 0) TP / (TP + FN) else 0
  accuracy  <- mean(pred == true)
  f1        <- if ((precision + recall) > 0) 2 * precision * recall / (precision + recall) else 0

  return(c(Accuracy = accuracy, Precision = precision, Recall = recall, F1 = f1))
}


# Evaluate models
dt_metrics    <- evaluate_model(WD.test$Class, dt_pred)
nb_metrics    <- evaluate_model(WD.test$Class, nb_pred)
bag_metrics   <- evaluate_model(WD.test$Class, bag_pred$class)
boost_metrics <- evaluate_model(WD.test$Class, boost_pred$class)
rf_metrics    <- evaluate_model(WD.test$Class, rf_pred)

# Format results
model_results <- rbind(
  "Decision Tree"  = dt_metrics,
  "Naive Bayes"    = nb_metrics,
  "Bagging"        = bag_metrics,
```

```
  "Boosting"       = boost_metrics,
  "Random Forest"  = rf_metrics
)

model_results <- round(model_results, 4)
model_results <- as.data.frame(model_results)
model_results$Model <- rownames(model_results)
rownames(model_results) <- NULL
```

Question 6

```
# Decision Tree
dt_prob <- predict(dt_model, WD.test, type = "vector")[, "1"]
dt_pred <- ROCR::prediction(dt_prob, WD.test$Class)
dt_perf <- performance(dt_pred, "tpr", "fpr")
dt_auc <- performance(dt_pred, "auc")@y.values[[1]]

# Naive Bayes
nb_prob <- predict(nb_model, WD.test, type = "raw")[, "1"]
nb_pred <- ROCR::prediction(nb_prob, WD.test$Class)
nb_perf <- performance(nb_pred, "tpr", "fpr")
nb_auc <- performance(nb_pred, "auc")@y.values[[1]]

# Bagging
bag_prob <- bag_pred$prob[, 1]
bag_pred_obj <- ROCR::prediction(bag_prob, WD.test$Class)
bag_perf <- performance(bag_pred_obj, "tpr", "fpr")
bag_auc <- performance(bag_pred_obj, "auc")@y.values[[1]]

# Boosting
boost_prob <- boost_pred$prob[, 1]
boost_pred_obj <-ROCR:: prediction(boost_prob, WD.test$Class)
boost_perf <- performance(boost_pred_obj, "tpr", "fpr")
boost_auc <- performance(boost_pred_obj, "auc")@y.values[[1]]

# Random Forest
rf_prob <- predict(rf_model, WD.test, type = "prob")[, "1"]
rf_pred_obj <- ROCR::prediction(rf_prob, WD.test$Class)
rf_perf <- performance(rf_pred_obj, "tpr", "fpr")
rf_auc <- performance(rf_pred_obj, "auc")@y.values[[1]]

# Define a nicer color palette
colors <- c(
  "Decision Tree"  = "#1f77b4",  # soft blue
  "Naive Bayes"    = "#2ca02c",  # soft green
  "Bagging"        = "#d62728",  # muted red
  "Boosting"       = "#9467bd",  # medium purple
  "Random Forest"  = "#ff7f0e"   # orange
)

# Plot ROC curves with updated colors
plot(dt_perf, col = colors["Decision Tree"], lwd = 2, main = "ROC Curves for Classifiers")
```

```r
plot(nb_perf, add = TRUE, col = colors["Naive Bayes"], lwd = 2)
plot(bag_perf, add = TRUE, col = colors["Bagging"], lwd = 2)
plot(boost_perf, add = TRUE, col = colors["Boosting"], lwd = 2)
plot(rf_perf, add = TRUE, col = colors["Random Forest"], lwd = 2)
abline(a = 0, b = 1, lty = 2, col = "gray60")

legend("bottomright", legend = c(
  paste("Decision Tree (AUC =", round(dt_auc, 4), ")"),
  paste("Naive Bayes (AUC =", round(nb_auc, 4), ")"),
  paste("Bagging (AUC =", round(bag_auc, 4), ")"),
  paste("Boosting (AUC =", round(boost_auc, 4), ")"),
  paste("Random Forest (AUC =", round(rf_auc, 4), ")")
), col = unname(colors), lwd = 2, cex = 0.6)


auc_table <- data.frame(
  Model = c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Random Forest"),
  AUC = round(c(dt_auc, nb_auc, bag_auc, boost_auc, rf_auc), 4)
)
knitr::kable(auc_table, format = "pipe")
```

Question 7

```r
# Combine evaluation metrics and AUC
model_results$AUC <- round(c(dt_auc, nb_auc, bag_auc, boost_auc, rf_auc), 4)

# Reorder columns for clarity
model_results <- model_results[, c("Model", "Accuracy", "Precision", "Recall", "F1", "AUC")]

# Print as markdown table for report
knitr::kable(model_results, format = "pipe")
```

Question 8

```r
# --- Random Forest ---
rf_imp <- importance(rf_model)[, "MeanDecreaseGini"]
rf_df <- data.frame(
  Feature = names(rf_imp),
  Importance = rf_imp
) |>
  arrange(Importance)

p_rf <- ggplot(rf_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  coord_flip() +
  labs(
    title = "Random Forest: Feature Importance",
    x = "Feature",
    y = "MeanDecreaseGini"
```

```
  ) +
  theme_minimal()

# --- Boosting ---
boost_imp <- boost_model$importance
boost_df <- data.frame(
  Feature = names(boost_imp),
  Importance = boost_imp
) |>
  arrange(Importance)

p_boost <- ggplot(boost_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "lightcoral") +
  coord_flip() +
  labs(
    title = "Boosting: Feature Importance",
    x = "Feature",
    y = "Importance"
  ) +
  theme_minimal()

# --- Bagging ---
bag_imp <- bag_model$importance
bag_df <- data.frame(
  Feature = names(bag_imp),
  Importance = bag_imp
) |>
  arrange(Importance)

p_bag <- ggplot(bag_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_bar(stat = "identity", fill = "darkseagreen3") +
  coord_flip() +
  labs(
    title = "Bagging: Feature Importance",
    x = "Feature",
    y = "Importance"
  ) +
  theme_minimal()

# --- Combine All Plots in 2x2 Layout ---
(p_rf | p_boost) / (p_bag)
```

Question 10

```
simple_classifier <- function(df) {
  ifelse(df$A17 <= -0.6 & df$A02 > -1.79, 1, 0)
}

# Rule-based classifier returning a proxy probability (1 if rule met, 0 otherwise)
simple_probs <- ifelse(WD.test$A17 > 0.004 & WD.test$A02 > -17, 1, 0)
```

```r
# Convert predictions and truth for confusion matrix
simple_preds <- simple_classifier(WD.test)
preds <- factor(simple_preds, levels = c("0", "1"))
truth <- factor(WD.test$Class, levels = c("0", "1"))

# Confusion Matrix evaluation
eval <- confusionMatrix(preds, truth, positive = "1")

# Compute AUC using ROCR
pred_obj <- ROCR::prediction(simple_probs, truth)
auc <- performance(pred_obj, "auc")@y.values[[1]]
auc <- round(auc, 4)

# Extract and round metrics
accuracy <- round(eval$overall["Accuracy"], 4)
precision <- round(eval$byClass["Precision"], 4)
recall <- round(eval$byClass["Recall"], 4)
f1 <- round(eval$byClass["F1"], 4)

# Create results table
final_table <- data.frame(
  Model = "Simple Classifier",
  Accuracy = accuracy,
  Precision = precision,
  Recall = recall,
  F1 = f1,
  AUC = auc
)

# Print as markdown table
knitr::kable(final_table, format = "pipe", row.names = FALSE)
```

Question 11

```r
model_base <- rpart(Class ~ ., data = WD.train)
model_best <- rpart(Class ~ ., data = WD.train, control = rpart.control(cp = 0.0005))

# Predict and evaluate
preds <- predict(model_best, WD.test, type = "class")
conf <- confusionMatrix(preds, WD.test$Class, positive = "1")

# AUC calculation
probs <- predict(model_best, WD.test)[, "1"]
roc_obj <- ROCR::prediction(probs, WD.test$Class)
auc <- performance(roc_obj, "auc")@y.values[[1]]
auc <- round(auc, 4)

# Metrics
accuracy <- round(conf$overall["Accuracy"], 4)
precision <- round(conf$byClass["Precision"], 4)
recall <- round(conf$byClass["Recall"], 4)
```

```r
f1 <- round(conf$byClass["F1"], 4)

# Results table
tree_best_table <- data.frame(
  Model = "Best Tree (Pruned)",
  Accuracy = accuracy,
  Precision = precision,
  Recall = recall,
  F1 = f1,
  AUC = auc
)
knitr::kable(tree_best_table, format = "pipe", row.names = FALSE)
```

Question 12

```r
# Log Transformation
WD$A21 <- log1p(WD$A21)
WD$A19 <- log1p(WD$A19)

# Normalization
WD[, 1:20] <- scale(WD[, 1:20])

# Class as factor
WD$Class <- factor(WD$Class, levels = c(0, 1))

set.seed(35216131)
train.row <- sample(1:nrow(WD), 0.7 * nrow(WD))
WD.train <- WD[train.row, ]
WD.test <- WD[-train.row, ]

# ----- Question 12 ------------------------------------------------------------

# Subset for ANN: Use most important features only
ann_formula <- as.formula("Class ~ A17 + A02 + A14 + A08")

# Fit neural network model
ann_model <- neuralnet(ann_formula,
                       data = WD.train,
                       hidden = c(5),
                       linear.output = FALSE,
                       stepmax = 1e6)

# Predict
ann_pred <- compute(ann_model, WD.test[, c("A17", "A02", "A14", "A08")])
ann_prob <- ann_pred$net.result[, 1]
ann_class <- ifelse(ann_prob > 0.5, 1, 0)

# Evaluation
ann_pred_label <- factor(ann_class, levels = c(0, 1))
truth_label <- factor(WD.test$Class, levels = c(0, 1))
```

```r
conf <- confusionMatrix(ann_pred_label, truth_label, positive = "1")
pred_obj <- ROCR::prediction(ann_prob, truth_label)
ann_auc <- ROCR::performance(pred_obj, "auc")@y.values[[1]]

# Metrics
ann_results <- data.frame(
  Model = "Neural Network",
  Accuracy = round(conf$overall["Accuracy"], 4),
  Precision = round(conf$byClass["Precision"], 4),
  Recall = round(conf$byClass["Recall"], 4),
  F1 = round(conf$byClass["F1"], 4),
  AUC = round(ann_auc, 4)
)
```

Question 13

```r
svm_model <- svm(Class ~ ., data = WD.train, kernel = "radial", probability = TRUE)

# Predict on test set
svm_pred <- predict(svm_model, WD.test, probability = TRUE)

# Convert predictions for evaluation
svm_probs <- attr(svm_pred, "probabilities")[, "1"]
svm_pred_label <- as.factor(ifelse(svm_probs > 0.5, 1, 0))

# Evaluation metrics
conf <- confusionMatrix(svm_pred_label, WD.test$Class, positive = "1")
svm_accuracy <- round(conf$overall["Accuracy"], 4)
svm_precision <- round(conf$byClass["Precision"], 4)
svm_recall <- round(conf$byClass["Recall"], 4)
svm_f1 <- round(conf$byClass["F1"], 4)

# AUC calculation
svm_pred_obj <- ROCR::prediction(svm_probs, WD.test$Class)
svm_auc <- round(performance(svm_pred_obj, "auc")@y.values[[1]], 4)

# Results table
svm_results <- data.frame(
  Model = "SVM (Radial Kernel)",
  Accuracy = svm_accuracy,
  Precision = svm_precision,
  Recall = svm_recall,
  F1 = svm_f1,
  AUC = svm_auc
)

knitr::kable(svm_results, format = "pipe", row.names = FALSE)
```