Learning to Plot ROC Curves with ggplot2: A Step-by-Step Guide

Authored by

Mohammed looti

November 6, 2025

The Role of Binary Classification in Predictive Analytics

In the vast landscape of data science and predictive analytics, models designed to forecast binary outcomes--such as whether a customer will churn, a loan applicant will default, or a patient has a specific disease--are fundamental. This domain, known as **binary classification**, relies on sophisticated statistical tools to estimate the probability of one of two possible states occurring. Among the most popular and foundational methods for tackling this challenge is Logistic Regression. Unlike standard linear models that predict continuous numeric values, logistic regression employs the logistic function (or sigmoid function) to transform the output into a probability score constrained strictly between 0 and 1, making it perfectly suited for modeling likelihoods.

Building a robust classification model is only the first step. The subsequent and equally critical phase involves **rigorous model assessment**. A model's success cannot be judged solely by its training accuracy; its true value lies in its ability to generalize to unseen data and, more specifically, how effectively it differentiates between the positive and negative classes. This differentiation capability is paramount in high-stakes environments, such as finance or healthcare, where misclassification errors carry significant costs. Consequently, data scientists must move beyond simple aggregate metrics like overall accuracy, which can often be misleading, especially when dealing with imbalanced datasets.

A comprehensive evaluation requires the use of specialized metrics that quantify the model's performance across various potential classification thresholds. These metrics help us understand the specific types of errors the model makes: **False Positives** (predicting an event will happen when it does not) and **False Negatives** (predicting an event will not happen when it actually does). To gain a nuanced view of this predictive power and ensure we select a model that aligns with the business or research objectives, we must leverage foundational concepts like Sensitivity and Specificity, which form the bedrock for advanced visualizations like the ROC curve.

Deconstructing Model Performance: Sensitivity and Specificity

To truly understand a model's discriminatory ability, we must break down the results into the four possible outcomes relative to the true class. These results are typically summarized in a **Confusion Matrix**, which defines: **True Positives (TP)**, **True Negatives (TN)**, **False Positives (FP)**, and **False Negatives (FN)**. From these counts, two primary metrics--Sensitivity and Specificity--are derived, providing a clearer, more actionable picture of performance than a single accuracy score.

Sensitivity (True Positive Rate - TPR): This metric measures the proportion of actual positive cases that were correctly identified by the model. Calculated as TP / (TP + FN), high sensitivity is crucial when the cost of a false negative is high. For example, in a medical screening test, a high

sensitivity ensures that very few actual disease cases are missed.

Specificity (True Negative Rate - TNR): This metric measures the proportion of actual negative cases that were correctly identified. Calculated as TN / (TN + FP), high specificity is essential when the cost of a false positive is high. For instance, in fraud detection, high specificity minimizes unnecessary investigations into legitimate transactions.

It is vital to recognize the inherent **inverse relationship** between these two metrics. The classification process requires setting a probability threshold (e.g., 0.5): probabilities above this threshold are classified as positive, and those below as negative. If we lower this threshold (making it easier to classify an observation as positive), we typically increase Sensitivity (capturing more true positives) but decrease Specificity (increasing the rate of false positives). Conversely, raising the threshold increases Specificity but decreases Sensitivity. This trade-off is central to model selection and threshold tuning, forcing the analyst to decide which type of error is more tolerable given the application context. The graphical representation of this fundamental trade-off is precisely what the ROC curve captures.

Visualizing the Trade-off: The Receiver Operating Characteristic (ROC) Curve

The challenge in optimizing a binary classification model often boils down to selecting the single best probability threshold. The Receiver Operating Characteristic (ROC) curve offers a comprehensive solution by visualizing the model's performance across **all possible classification thresholds simultaneously**. This curve is an essential tool for assessing the inherent discriminatory power of a predictive model, independent of any specific threshold choice.

The ROC curve is constructed by plotting the **True Positive Rate (Sensitivity)** on the Y-axis against the **False Positive Rate (FPR)**, which is simply 1 - Specificity, on the X-axis. Each point along the resulting curve corresponds to the sensitivity and specificity achieved at a particular threshold setting. A perfect classifier would pass through the top-left corner of the plot, achieving 100% Sensitivity (TPR = 1) and 0% False Positive Rate (FPR = 0). Therefore, a curve that bows significantly toward this upper-left corner signifies a superior model that can achieve high true positive rates while maintaining a low false positive rate.

While the visual shape of the curve offers immediate insight, a single quantitative metric is often used to summarize the model's overall quality: the **Area Under the Curve (AUC)**. The AUC represents the probability that the model will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Values for the AUC range from 0.5 (which indicates performance no better than random guessing, corresponding to the diagonal line) to 1.0 (indicating a perfect classifier). A higher AUC score suggests better overall separability between classes. Our goal in this tutorial is to leverage the power of the R programming language, specifically the ggplot2 package, to generate and customize this vital visualization.

Preparing the Data Environment in R: A Practical Walkthrough

To effectively demonstrate the process of plotting and interpreting an ROC curve, we must first establish a controlled environment within R. This involves loading the required data, partitioning it appropriately, and fitting the predictive model. We will use the well-known *Default* dataset, available within the ISLR package, which contains financial and demographic variables used to predict the probability of loan default. This practical setup ensures that our ROC analysis is based on realistic model output.

Data partitioning is a critical step in any robust machine learning workflow. By dividing the dataset into separate training (70%) and testing (30%) sets, we ensure that the model's performance metrics, including the ROC curve and AUC, are evaluated on observations that the model has never encountered before. This prevents optimistic bias and provides an unbiased assessment of the model's ability to **generalize** to new, unseen data points. The initial code chunk below loads the data, establishes the random split using `set.seed(1)` for reproducibility, and defines the training and testing subsets.

Following the data split, we fit a Logistic Regression model to the training set. We use the `glm()` function with the argument `family="binomial"` to model the probability of default based on key predictors like student status, account balance, and income. Crucially, we then use this fitted model to generate predicted probabilities on the **test set**. These predicted probabilities, paired with the actual outcomes from the test set, are the essential inputs required for calculating and plotting the ROC curve.

#load *Default* dataset from ISLR book
data <- ISLR::Default

#divide dataset into training and test set
set.seed(1)
sample <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE, prob=c(0.7,0.3))
train <- data
test <- data

#fit logistic regression model to training set
model <- glm(default~student+balance+income, family="binomial", data=train)

#use model to make predictions on test set
predicted <- predict(model, test, type="response")

Constructing the ROC Curve using the pROC and ggplot2 Packages

Once we have the true outcomes and the predicted probabilities from our test set, we turn to specialized R packages to handle the curve calculation and plotting. We rely on the `pROC` package, which is specifically designed for analyzing and representing ROC curves, and the highly versatile ggplot2 package for visualization. The synergistic use of these two libraries allows us to easily transform raw data into a professional-grade graphical output.

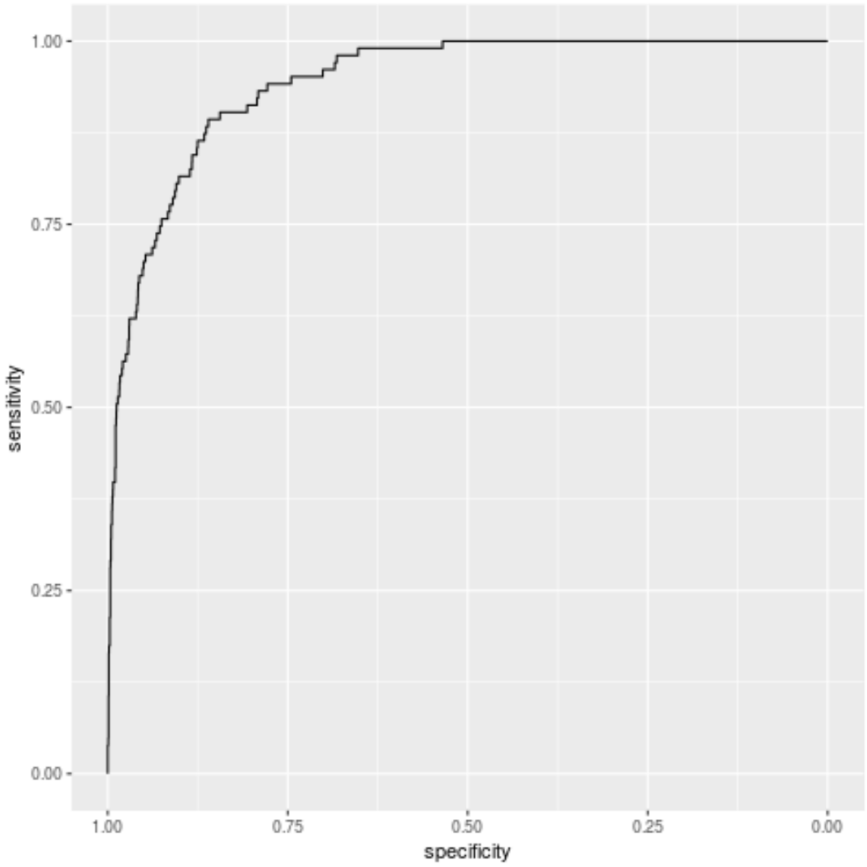
The process begins with the `roc()` function from the pROC package. This function takes the vector of true class labels (`test$default`) and the vector of predicted probabilities (`predicted`) and calculates all the necessary True Positive Rate and False Positive Rate pairs, storing them in an ROC object. This object encapsulates the complete performance profile of the model across all possible thresholds. Next, the `ggroc()` wrapper function is called, which recognizes the ROC object and automatically generates the base plot using ggplot2's infrastructure.

The result of this initial plotting sequence is a functional, unstyled visualization of the ROC curve. The visual assessment confirms that the curve deviates substantially from the diagonal line of chance (y=x), indicating that our Logistic Regression model possesses significant discriminatory power. Examination of the axes confirms that the Y-axis represents **Sensitivity** (TPR), and the X-axis represents the **False Positive Rate** (1 - Specificity).

```
#load necessary packages
library(ggplot2)
library(pROC)

#define object to plot
rocobj <- roc(test$default, predicted)

#create ROC plot
ggroc(rocobj)
```



Achieving Publication Quality: Incorporating AUC and Styling

While the basic ROC curve provides a valuable visual heuristic, professional data visualization standards dictate that the quantitative measure of model quality--the **Area Under the Curve (AUC)**--should be included directly on the graph. This practice immediately informs the viewer of the model's objective performance score, facilitating quicker interpretation and comparison. The `pROC` package simplifies this process greatly, providing a dedicated `auc()` function that takes the same inputs as the `roc()` function.

To enhance the visual appeal and clarity, we utilize ggplot2's layering system. We can modify the appearance of the curve itself by specifying aesthetic parameters such as `colour` and `size` within the `ggroc()` function. More importantly, we calculate the AUC score, round it to an appropriate decimal place, and dynamically embed this value into the plot title using the `ggtitle()` function in combination with R's string concatenation tool, `paste0()`. This technique ensures that the plot is self-contained and communicates both the visual shape and the quantitative magnitude of the model's success.

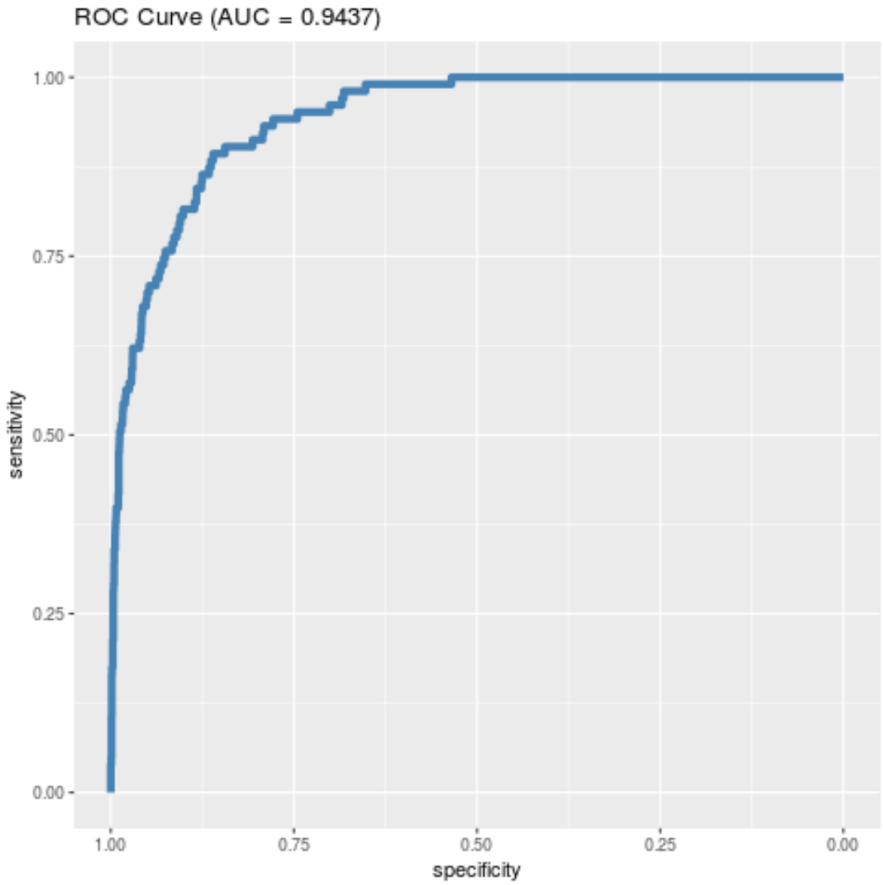
The following code snippet demonstrates the calculation of the AUC and its integration into a stylized plot, featuring a thick, contrasting line color for improved emphasis. This step transforms

the basic visualization into a polished graphical asset suitable for reports or presentations.

```
#load necessary packages
library(ggplot2)
library(pROC)

#define object to plot and calculate AUC
rocobj <- roc(test$default, predicted)
auc <- round(auc(test$default, predicted),4)

#create ROC plot
ggroc(rocobj, colour = 'steelblue', size = 2) +
ggtitle(paste0('ROC Curve ', '(AUC = ', auc, ')'))
```



Refining Visual Aesthetics with Themes for Clarity

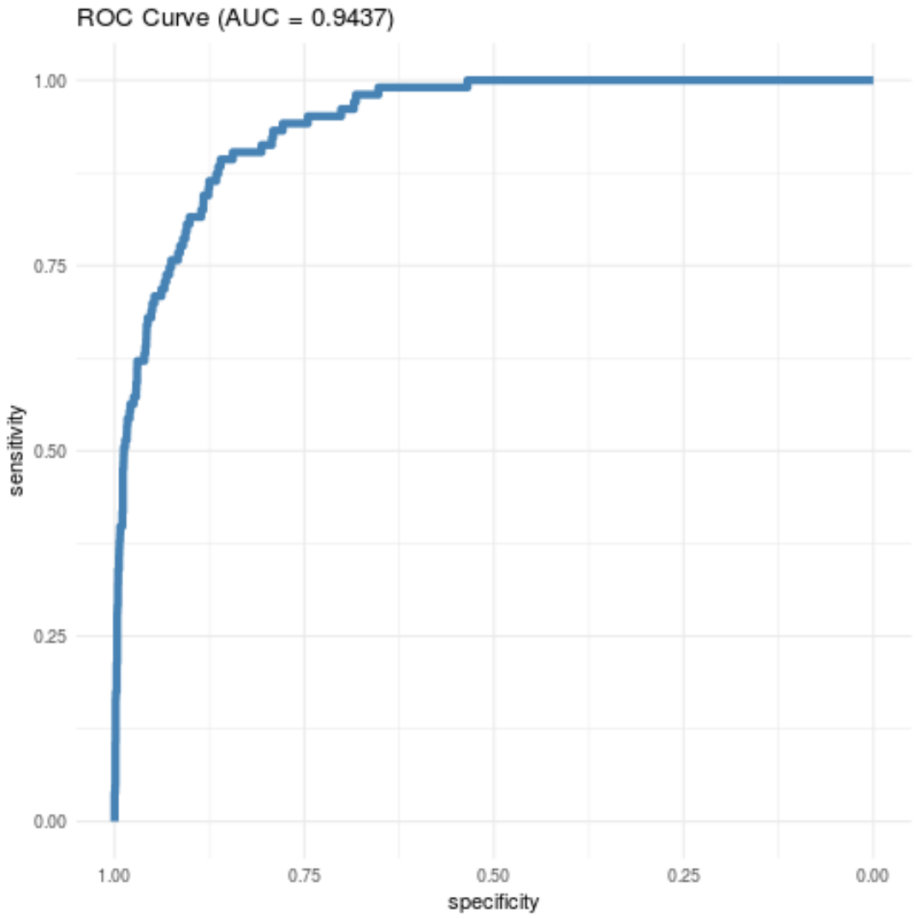
The final stage of creating a high-quality visualization involves refining the overall aesthetic presentation using themes. The ggplot2 library offers immense flexibility through its theme system,

allowing users to instantly modify non-data elements such as background colors, axis lines, and grid patterns. For academic or professional settings, eliminating unnecessary visual clutter is often crucial for maximizing impact and readability.

By applying the `theme_minimal()` layer to our existing plot code, we instantly strip away the default gray background and reduce the visual prominence of the grid lines. This transformation directs the viewer's attention entirely to the curve itself and the associated AUC metric, ensuring that the model performance is communicated clearly and elegantly. The ability to seamlessly switch themes is one of the most powerful features of the ggplot2 framework, allowing analysts to tailor visualizations to specific audiences and publication standards without tedious manual adjustments.

This refined visualization represents the culmination of the data science process: from model fitting and probability prediction to complex metric calculation using the pROC package, all presented through the clean, professional lens of a ggplot2 theme. Mastering this process is fundamental for anyone serious about evaluating and communicating the efficacy of their predictive models in a comprehensible manner.

```
#create ROC plot with minimal theme
ggroc(rocobj, colour = 'steelblue', size = 2) +
ggtitle(paste0('ROC Curve ', '(AUC = ', auc, ')')) +
theme_minimal()
```



In summary, the ROC curve remains the gold standard for visualizing and interpreting the performance of binary classification models. By leveraging the integrated power of the pROC package for statistical calculation and the ggplot2 package for aesthetic control, we can generate sophisticated, publication-quality graphics that clearly communicate a model's discriminatory ability across all possible operating points.