Drills with R: K-NN Models

Lisa Hall

University of the Cumberlands

Statistics for Data Science (MSDS-531-A01) - First Bi-term

Dr. Mark D. Mosley

February 14, 2025

Drills with R: K-NN Models

K-Nearest Neighbors (kNN) is an algorithm for classification and regression. It is a non-parametric, instance-based learning method (Thulin, 2024). kNN calculates the distance between data points and uses that distance to classify a new data point by assigning the most common class among the K nearest neighbors. For regression, KNN predicts a value by averaging the values of the nearest neighbors. This modeling process is simplified by using R.

**Evaluating kNN Using R**

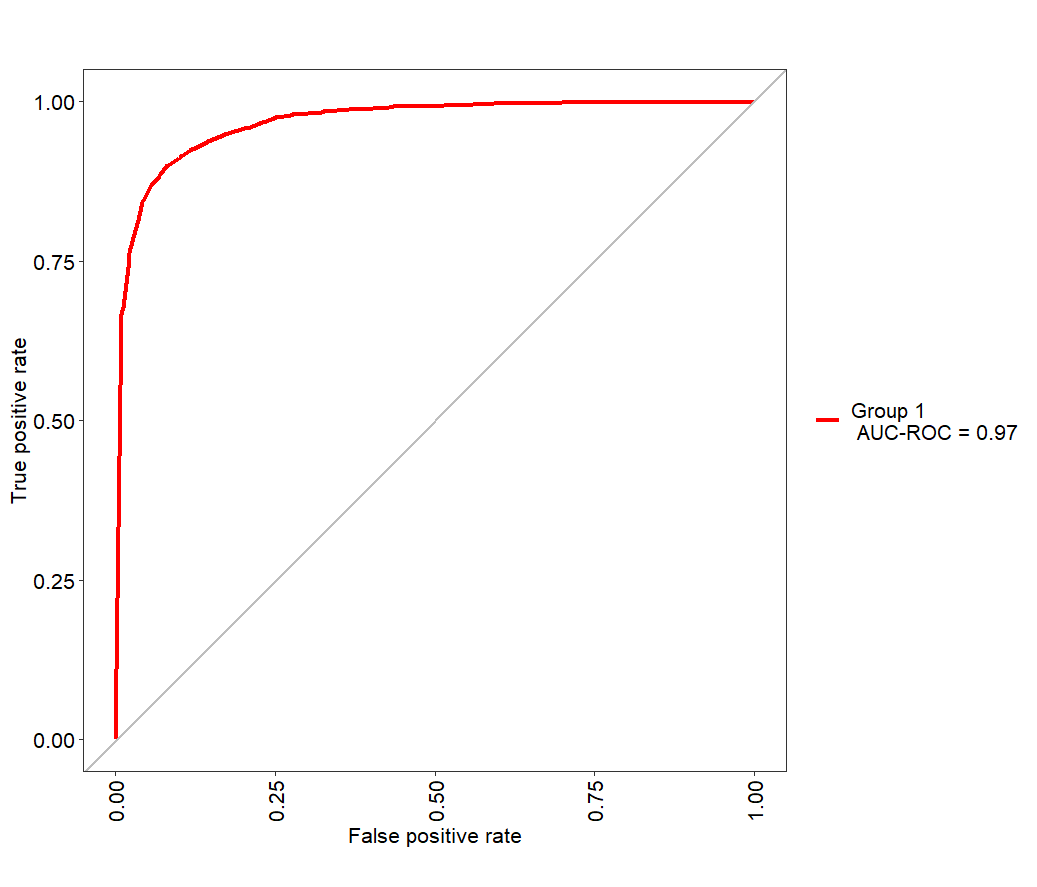
To evaluate the kNN modeling process, libraries such as caret (Kuhn, 2008), which provides the function train(); mleval () (John, 2020), which is used for evaluating models with ROC curves; pROC (Robin et al., 2011), which handles receiver operating characteristic curve analysis; and class (Venables & Ripley, 2002), which contains kNN classification functions will be utilized. For this example, we will fit a kNN classification model to wine data, using alcohol, fixed acidity, residual sugar, and pH as explanatory variables. We will evaluate its performance using a 10-fold cross-validation using AUC. In Appendix A, we can observe the R code utilized to install and load the necessary libraries. We can also observe the csv imported and the merging of the datasets.

The R code selects the column names used for the kNN classification. We can observe that the value “selected\_vars” is set to represent the values in Appendix A. There are four features selected: pH, alcohol, and fixed. acidity and residual sugar. With the value set, we can then set another value for the training control; this is used for model validation. “train\_control” value creates a cross-validation strategy; method = cv in the trainControl() function represents this, and number = 10 indicates it will be 10 fold. Further, in the trainControl() function, we can enable class probability estimates using classProbs = True, and further use receiver operating characteristics – area under curve metric by setting summaryFunction = twoClassSummary. We can then save the final predictions by indicating savePredictions = “final.” The ROC curve is a graphical tool used as a metric to evaluate a model (Bruce & Bruce, 2019).

Next, in Appendix A, we can use set.seed to ensure the reproducibility of the model. The value “knn\_fit” trains the kNN classifier; using the train(), it sets the target variable, uses the cross-validation set earlier, sets the metric as “ROC,” tuneLength = 10 tests 10 different values of k, “preprocess” normalizes the data. The value set as knn\_results uses the eval () function to compute performance metrics; the results are then plotted. The output for AUC-ROC indicates that the model has excellent performance; a perfect classifier would be 1.0, but ours is incredibly close at .97. We can also observe there is a class imbalance, with more white wines than red wines.

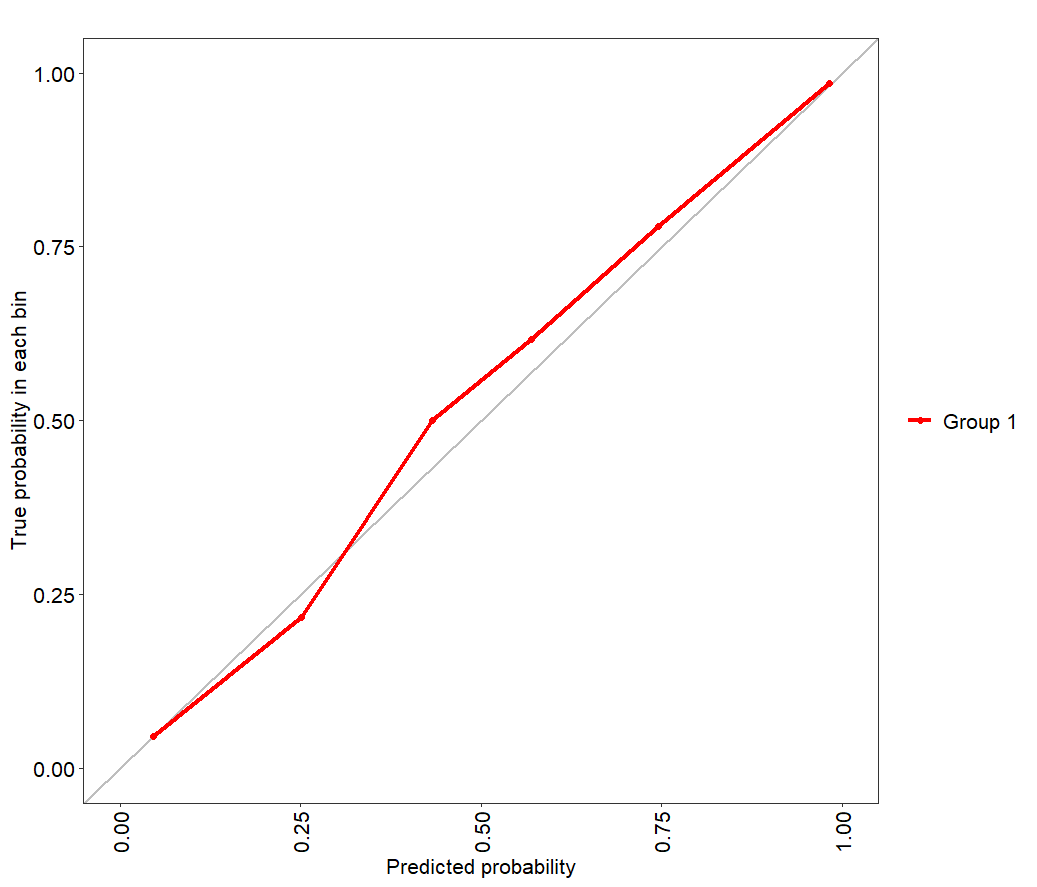
As observed in appendix A, we can also find the best k value. Setting value “best\_k” by using knn\_fit$bestTune$k retrieves the best k from the model. Using the plot() function, we can also plot the ROC curve. Figure 1 shows the ROC curve plot. The high AUC value (0.97) indicates that model has very good classification performance.

**Figure 1: ROC Cuver for kNN Model Wine Dataset**



We can evaluate one step further by creating a calibration plot. A calibration plot visualizes how the predicted probabilities from the kNN model align with the actual probabilities. Figure 2 demonstrates that the model is well calibrated; if the red line were to differentiate from the diagonal, it would either underestimate the true probability of it being above the line or overestimate if it were below the line. Since there is only some deviation, it represents good calibration.

Figure 2 Calibration Plot kNN



**Conclusion**

The kNN model demonstrates how wine can be classified using chemical composition. It is important to note that kNN can be expensive in time and space, so it does not always garner near-perfect results (Zumel & Mount, 2014). It is important to carefully select modeling types based on data size, time, and effectiveness. While it can be computationally expensive, kNN is still a great choice for classification models.

**References**

Bruce, P., & Bruce, A. (2019). *Practical statistics for data scientists: 50+ essential concepts using R and Python* (2nd ed.). O'Reilly Media.

John, C. R. (2020). *MLeval: Machine learning model evaluation* (Version 0.3) [R package]. https://CRAN.R-project.org/package=MLeval

Kuhn, M. (2008). Building predictive models in R using the caret package. *Journal of Statistical Software, 28*(5), 1–26. https://doi.org/10.18637/jss.v028.i05

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C., & Müller, M. (2011). pROC: An open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics, 12*, 77. https://doi.org/10.1186/1471-2105-12-77

Thulin, M. (2024). *Modern statistics with R: From wrangling and exploring data to inference and predictive modelling* (2nd ed.). Chapman and Hall/CRC.

Venables, W. N., & Ripley, B. D. (2002). *Modern applied statistics with S* (4th ed.). Springer. <https://www.stats.ox.ac.uk/pub/MASS4/>

Zumel, N., & Mount, J. (2014). *Practical data science with R*. Manning Publications.

**Appendix A**

**kNN Evaluation Using R**

install.packages("caret", dependencies = TRUE)

install.packages("MLeval", dependencies = TRUE)

install.packages("pROC", dependencies = TRUE)

install.packages("class", dependencies = TRUE)

library(caret)

library(MLeval)

library(pROC)

library(class)

white <- read.csv("https://tinyurl.com/winedata1", sep = ";")

red <- read.csv("https://tinyurl.com/winedata2", sep = ";")

white$type <- "white"

red$type <- "red"

wine <- rbind(white, red)

wine$type <- factor(wine$type, levels = c("red", "white"))

selected\_vars <- c("pH", "alcohol", "fixed.acidity", "residual.sugar")

train\_control <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction = twoClassSummary, savePredictions = "final")

set.seed(123)

knn\_fit = "ROC", tuneLength = 10, preProcess = c("center", "scale"))<- train(type ~ ., data = wine[, c(selected\_vars, "type")], method = "knn", trControl = train\_control, metric

knn\_results <- evalm(knn\_fit)

best\_k <- knn\_fit$bestTune$k

print(paste("Best k value:", best\_k))

plot(knn\_results$roc)

plots$cc