Machine Learning I

KNN y Naives Bayes

Dante Conti, Sergi Ramirez, (c) IDEAI

2025-10-09

Table of contents

0.1 Definición del proyecto

0.2 KNN

0.2.1 Distancias

0.2.2 KNN Classifier

0.2.2.1 R Base

```r
library(ISLR)
library(class)
library(caret)
```

```r
set.seed(42)
Default$student = as.numeric(Default$student) - 1
default_idx = sample(nrow(Default), 5000)
default_trn = Default[default_idx, ]
default_tst = Default[-default_idx, ]
```

```r
# training data
X_default_trn = default_trn[, -1]
y_default_trn = default_trn$default

# testing data
X_default_tst = default_tst[, -1]
y_default_tst = default_tst$default
```

```r
prediccion <- knn(train = X_default_trn, test  = X_default_tst,
                  cl = y_default_trn, k = 3)
head(prediccion)
```

```r
calc_class_err = function(actual, predicted) {
  mean(actual != predicted)
}
```

```r
calc_class_err(actual    = y_default_tst,
               predicted = knn(train = X_default_trn,
                               test  = X_default_tst,
                               cl    = y_default_trn,
                               k     = 5))
```

```r
calc_class_err(actual    = y_default_tst,
               predicted = knn(train = scale(X_default_trn),
                               test  = scale(X_default_tst),
                               cl    = y_default_trn,
                               k     = 5))
```

```r
set.seed(42)
k_to_try = 1:100
err_k = rep(x = 0, times = length(k_to_try))

for (i in seq_along(k_to_try)) {
  pred = knn(train = scale(X_default_trn),
             test  = scale(X_default_tst),
```

```r
             cl    = y_default_trn,
             k     = k_to_try[i],
             prob = T)
  err_k[i] = calc_class_err(y_default_tst, pred)
}
```

```r
# plot error vs choice of k
plot(err_k, type = "b", col = "dodgerblue", cex = 1, pch = 20,
     xlab = "k, number of neighbors", ylab = "classification error",
     main = "(Test) Error Rate vs Neighbors")
# add line for min error seen
abline(h = min(err_k), col = "darkorange", lty = 3)
# add line for minority prevalence in test set
abline(h = mean(y_default_tst == "Yes"), col = "grey", lty = 2)
```

```r
min(err_k)
```

```r
which(err_k == min(err_k))
```

```r
max(which(err_k == min(err_k)))
```

0.2.2.2 packages `Caret`

```r
set.seed(430)
default_idx = createDataPartition(Default$default, p = 0.75, list = FALSE)
default_trn = Default[default_idx, ]
default_tst = Default[-default_idx, ]
```

```r
modelLookup("knn")
```

```r
sim_knn_mod = train(
  default ~ .,
  data = default_trn,
  method = "knn",
  trControl = trainControl(method = "cv", number = 5),
  # preProcess = c("center", "scale"),
  tuneGrid = expand.grid(k = seq(1, 31, by = 2)))

sim_knn_mod
```

```
sim_knn_mod$modelType
```

```
get_best_result = function(caret_fit) {
  best = which(rownames(caret_fit$results) == rownames(caret_fit$bestTune))
  best_result = caret_fit$results[best, ]
  rownames(best_result) = NULL
  best_result
}
```

```
head(sim_knn_mod$results, 5)
```

```
get_best_result(sim_knn_mod)
```

```
plot(sim_knn_mod)
```

```
sim_knn_mod$finalModel
```

```
head(predict(sim_knn_mod, newdata = default_tst, type = "prob"))
```

```
caret::confusionMatrix(predict(sim_knn_mod), dp_entr_NUM$CLS_PRO_pro13)
```

0.2.3 KNN Regresor

```
library(FNN)
library(MASS)
data(Boston)
```

```
set.seed(42)
boston_idx = sample(1:nrow(Boston), size = 250)
trn_boston = Boston[boston_idx, ]
tst_boston  = Boston[-boston_idx, ]
```

```
X_trn_boston = trn_boston[-ncol(trn_boston)]
X_tst_boston = tst_boston[-ncol(trn_boston)]
y_trn_boston = trn_boston["medv"]
y_tst_boston = tst_boston["medv"]
```

```r
pred_001 = knn.reg(train = X_trn_boston, test = X_tst_boston, y = y_trn_boston, k = 1)
pred_005 = knn.reg(train = X_trn_boston, test = X_tst_boston, y = y_trn_boston, k = 5)
pred_010 = knn.reg(train = X_trn_boston, test = X_tst_boston, y = y_trn_boston, k = 10)
pred_050 = knn.reg(train = X_trn_boston, test = X_tst_boston, y = y_trn_boston, k = 50)
pred_100 = knn.reg(train = X_trn_boston, test = X_tst_boston, y = y_trn_boston, k = 100)
pred_250 = knn.reg(train = X_trn_boston, test = X_tst_boston, y = y_trn_boston, k = 250)
```

```r
rmse = function(actual, predicted) {
  sqrt(mean((actual - predicted) ^ 2))
}
```

```r
# define helper function for getting knn.reg predictions
# note: this function is highly specific to this situation and dataset
make_knn_pred = function(k = 1, training, predicting) {
  pred = FNN::knn.reg(train = training["lstat"],
                      test = predicting["lstat"],
                      y = training$medv, k = k)$pred
  act  = predicting$medv
  rmse(predicted = pred, actual = act)
}
```

```r
# define values of k to evaluate
k = c(1, 5, 10, 25, 50, 250)

# get requested train RMSEs
knn_trn_rmse = sapply(k, make_knn_pred,
                      training = trn_boston,
                      predicting = trn_boston)
```

```r
# get requested test RMSEs
knn_tst_rmse = sapply(k, make_knn_pred,
                      training = trn_boston,
                      predicting = tst_boston)
```

```r
# determine "best" k
best_k = k[which.min(knn_tst_rmse)]
```

```r
# find overfitting, underfitting, and "best"" k
fit_status = ifelse(k < best_k, "Over", ifelse(k == best_k, "Best", "Under"))
```

```r
# summarize results
knn_results = data.frame(
  k,
  round(knn_trn_rmse, 2),
  round(knn_tst_rmse, 2),
  fit_status
)
colnames(knn_results) = c("k", "Train RMSE", "Test RMSE", "Fit?")

# display results
knitr::kable(knn_results, escape = FALSE, booktabs = TRUE)
```

0.2.3.1 packages `Caret`

```r
caret::modelLookup("knn")
```

```r
library("CDR")
library("class")
library("caret")
library("reshape")
library("ggplot2")
```

```r
data(dp_entr_NUM)
head(dp_entr_NUM)
```

```r
# Definimos un método de remuestreo
cv <- trainControl(
  method = "repeatedcv",
  number = 10,
  repeats = 5,
  classProbs = TRUE,
  preProcOptions = list("center"),
  summaryFunction = twoClassSummary
)
```

```r
# Definimos la red de posibles valores del hiperparámetro
hyper_grid <- expand.grid(k = c(1:10,15,20,30,50,75,100))
```

```r
set.seed(101)
# Se entrena el modelo ajustando el hiperparámetro óptimo
model <- train(
  CLS_PRO_pro13 ~ .,
  data = dp_entr_NUM,
  method = "knn",
  trControl = cv,
  tuneGrid = hyper_grid,
  metric = "ROC"
)
```

```r
ggplot(model) +
  geom_vline(xintercept = unlist(model$bestTune),col="red",linetype="dashed") +
  theme_light()
```

```r
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
   geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)
```

```r
set.seed(101)
confusionMatrix(predict(model), dp_entr_NUM$CLS_PRO_pro13)
```

0.3 Naives Bayes

0.3.1 NB Classifier

```r
set.seed(430)
iris_obs = nrow(iris)
iris_idx = sample(iris_obs, size = trunc(0.50 * iris_obs))
# iris_index = sample(iris_obs, size = trunc(0.10 * iris_obs))
iris_trn = iris[iris_idx, ]
iris_tst = iris[-iris_idx, ]
```

```r
library(e1071)

iris_nb = naiveBayes(Species ~ ., data = iris_trn)
iris_nb
```

```r
head(predict(iris_nb, iris_trn))
head(predict(iris_nb, iris_trn, type = "class"))
head(predict(iris_nb, iris_trn, type = "raw"))
```

```r
iris_nb_trn_pred = predict(iris_nb, iris_trn)
iris_nb_tst_pred = predict(iris_nb, iris_tst)
```

```r
calc_class_err(predicted = iris_nb_trn_pred, actual = iris_trn$Species)
```

```r
calc_class_err(predicted = iris_nb_tst_pred, actual = iris_tst$Species)
```

```r
table(predicted = iris_nb_tst_pred, actual = iris_tst$Species)
```

0.3.2 Packages caret

```r
library("caret")
library("naivebayes")
library("reshape")
library("ggplot2")
library("CDR")

data("dp_entr")
```

```r
# se fija la semilla aleatoria
set.seed(101)

# se entrena el modelo
model <- train(CLS_PRO_pro13 ~ .,
            data=dp_entr,
            method="nb",
            metric="Accuracy",
            trControl=trainControl(classProbs = TRUE,
                                   method = "cv",
                                   number = 10))
```

```r
# se muestra la salida del modelo
model
```

```r
confusionMatrix(model)
```

```r
ggplot(melt(model$resample[,-4]), aes(x = variable, y = value, fill=variable)) +
  geom_boxplot(show.legend=FALSE) +
  xlab(NULL) + ylab(NULL)
```

0.4 Bibliografia

- https://daviddalpiaz.github.io/r
-