Cla&Clu: Breast Cancer

Tolga Tabanli

2025-10-22

Introduction

Cancer is a highly complex cellular physiological state caused by genetic events that mislead the cell cycle control. This physiology however can make itself apparent at a phenotypical level that can be recorded and evaluated, such as cell size and shape, mitoses, and adhesion characteristics etc. (more information in the .names file).

```r
library(tidyverse)
library(tidymodels)

data <- read_csv("data/breast-cancer-wisconsin.data",
                 na = "?",
                 col_names = c("id_number", "clump_thickness",
                               "uniformity_of_cell_size", "uniformity_of_cell_shape",
                               "marginal_adhesion", "single_epithelial_cell_size",
                               "bare_nuclei", "bland_chromatin", "normal_nucleoli",
                               "mitoses", "class"))
data$class <- as.factor(data$class)
```

Visualize data

Plot distribution of values stratified by the "class".
Hint: As before, the steps are `pivot_longer()`, `ggplot()` specifying `aes()` with `x` and `fill`, `geom_histogram()` followed by `facet_wrap()`.

```r
# TODO
```

Visualize missingness

Use `vis_miss()` from naniar package.

```r
library(naniar)

# TODO
```

Visualize correlations

Plot correlation plot using `corrplot()` from {corrplot}.

```r
# TODO
```

Clustering

Here, we will act as if we don't know the actual cancer status of the cells, and we will use the other variables to see whether the cells show any pattern in this variable space. We are going to apply PCA to reduce the dimensionality of the data to two principal axes that capture the most variance. We will use all the data for the clustering since we are not using the outcome to predict anything to test afterwards.

It is also possible to use `recipes` for this purpose, although we are not training a model per se. Your task is:

1. Create a recipe with formula `~ .`. This just means that we do not focus on any outcome variable yet and use all the variables.
2. We have an id column named **id_number** and we want to ignore the outcome. Update their roles accordingly.
3. Normalize all the predictors.
4. Impute missing values using kNN.
5. Apply PCA to get all the principal components (10).
6. *Prepare* the recipe and *bake* on the data. The components will be named as PC1, PC2, etc.
7. Plot the principal components in pairs using the outcome as the color of the points.
   Hint: We can plot principal components pairwise using `ggpairs()` from the GGally package.
   Example:

```r
library(GGally)

pc_axes <- paste0("PC", 1:4)

ggpairs(
  data = YOUR_DATA_WITH_PC,
  columns = pc_axes,
  mapping = aes(color = as.factor(TARGET))
)
```

8. Optional: Using factoextra package, plot the scree plot that shows the variance explained by the principal components, and the biplot to show the direction of the contribution of original variables.

   Example:

```r
fviz_eig(YOUR_PCA_RESULT, addlabels = TRUE)

# == Biplot ==
# biplot requires the coordinates of
# the data points in PC space in matrix form:
YOUR_PCA_RESULT$x <- YOUR_BAKED_DATA %>%
  select(starts_with("PC")) %>%
  as.matrix()

fviz_pca_biplot(YOUR_PCA_RESULT,
                col.ind = data$class,
                label = "var",
                repel = TRUE)
```

```r
library(GGally)
library(factoextra)

# Recipe
# TODO

# Prepare and bake the data
# TODO

## plot pairwise PCs using ggpairs()
# TODO

# extract the "res" object from third preprocessing step
# TODO

# Scree plot using fviz_eig()
# TODO

# == Biplot ==
# we need the original data points as matrix:
# TODO
```

Next, calculate the k-means clusters using the base `kmeans()`. Example:

```r
your_kmeans_object <- kmeans(YOUR_DATA[, c("col1", "col2")],
                             centers = HOW_MANY_CLUSTERS)
your_kmeans_clusters <- your_kmeans_object$cluster
```

and then draw two plots with PC1 and PC2 as axes next to each other, where points of the first are colored with class from the original data set and those of others with cluster from the kmeans clustering.
Hint: save the kmeans cluster result as a new column in the baked data frame. Then apply `pivot_longer()` on this and class column to get them under a "grouping" column with their values in a "class" column. **Colour** points according to the value under class and **facet** by grouping.

```r
# === k-means clustering and comparison with target cluster
# 1. calculate the k-means clusters
# TODO

# 2. Add kmeans_clusters as cluster
# TODO
```

Prediction model

Split the data into training (80 %) and test set (20 %).

```r
# Data split
# TODO
```

Use the same preprocessing as before but specify the formula with the target and predictors. Train a *logistic regression* model and a *random forest*. In the random forest, tune for all the three parameters using accuracy as the objective metric. Additionally, tune the number of components in PCA step in logistic regression, set it to 4 in random forest. After finalizing the models, report the accuracy and AUC for PR and ROC curves on the test set.

```
# recipe
# TODO

# models
# TODO

# workflows
# TODO

# hyperparameter opt
# TODO
```

```
# Finalize workflows
# TODO

# Perform last fit on the data split
# TODO

# Show the metrics
# TODO
```