University of Applied Sciences and Arts Northwestern Switzerland
School of Business

DS-04 Classification - Lab 1: kNN

Dr. Gwendolin Wilke

Task 1: Explore the dataset `Default{ISLR}`

- Explore the dataset `Default` from the `ISLR` library.
- The dataset holds historic records on credit card customers who have or have not defaulted on their credit card payments.
- Explore the dataset with the goal in mind that you want to train a classification model that predicts the value of the variable `default`.

Image source: https://cmci.colorado.edu/classes/INFO-4604/fa17/

What is a credit card default?

A credit card default happens when you're severely late on a credit payment, usually six months or longer. Defaults can cause serious damage to your credit score (in the US – I don't even know how it works in CH 🤔), and cost a lot of money in interest charges.

Default {ISLR}

Credit Card Default Data

Description

A simulated data set containing information on ten thousand customers. The aim here is to predict which customers will default on their credit card debt.

Usage

```
Default
```

Format

A data frame with 10000 observations on the following 4 variables.

`default`

A factor with levels `No` and `Yes` indicating whether the customer defaulted on their debt

`student`

A factor with levels `No` and `Yes` indicating whether the customer is a student

`balance`

The average balance that the customer has remaining on their credit card after making their monthly payment

`income`

Income of customer

Source

Simulated data

References

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013) *An Introduction to Statistical Learning with applications in R*, https://www.statlearning.com, Springer-Verlag, New York

A bit of data exploration

- Load the `ISLR` library, and inspect the included data set `Default` using the usual suspects and interpret them. What info do you get out of it? E.g., use `help()`, `head()`, `View()`, `str()`, `summary()`, ...
- Explore your data from different "angles", and interpret the results:
  - Apply the tools you already know , e.g.
    - `plot(Default)`
      What happens in the single cells? What can you read from it?
    - `hist.data.frame(Default)`
      Can you identify obvious outliers? What can you read from the distributions? Is the data balanced / unbalanced?
    - `sum(is.na())`
      What about missing values?
  - What predictors may have a strong influence on the target variable? How do the predictors relate to each other?
    - Can you use `corrplot()` to answer these questions?
    - What other tools would make sense instead? What graphs would you like to see to help you answer these questions?
    - Can `plot()`, `hist.data.frame()`, `summary()`, `boxplot()` or `table()` help you in this?

Task 2: Fit a kNN classifier to predict `default`

- Fit a kNN classifier to the training data for `k=1,3,10,20,30,40,50,100,200`.
- Evaluate and compare the results.
- Choose your best model.

Image source: https://cmci.colorado.edu/classes/INFO-4604/fa17/

Variable selection

We only have 3 potential input variables, `student, balance`, and `income` – so there is not much to select from anyways.

Yet, notice that kNN only works with numerical input variables. This means we cannot use `student` as a predictor. We have to kick it out.

Data preparation for kNN

1. *Standardizing the predictors*
   - Because the KNN classifier predicts the class of a given test observation by identifying the observations that are nearest to it, the scale of the variables matters.
   - Variables that are on a large scale will have a much larger effect on the distance between the observations, and hence on the KNN classifier, than variables that are on a small scale.
   - A good way to handle this problem is to standardize the data so that all variables are given a mean of zero and a standard deviation of one: `Default.stand.pred <- scale(Default[, -c(1,2)])`
   - Notice that we exclude the variables `student` and `default` from standardizing, since they are categorical.

2. *Converting the data to the knn()-specific format*
   - kNN only accepts a specific data format as input.
   - The predictors and the target variable ("class variable") must each be stored in a separate matrix or data frame.
   - This is easy to do, since we had to separate the predictors from the class variable already for standardizing.

Split the data in training and test data using a 20% - 80% split

```
set.seed(1)

indices <- sort(sample(1:dim(Default)[1], 2000)) # create 2000 randomly sampled indices -  we use a 80% - 20% split

test.data.pred <- Default.stand.pred[indices,] # select the corresponding observations for the test set
training.data.pred <- Default.stand.pred[-indices,] # select the remaining observations for the training set

test.data.class <- Default$default[indices] # store the class labels for the test set in a separate vector
training.data.class <- Default$default[-indices] # store the class labels for the training set in a separate vector
```

Fit a kNN model for k=1

```
set.seed (1)
knn.pred.1 <- knn(training.data.pred, test.data.pred, training.data.class, k=1)
```

- Why does kNN require us to provide the test data for model fitting? 🤔
- Inspect the output. What do you think it is?
- Evaluate the model using the test error rate `mean(knn.pred.1 != test.data$default)`. What does it tell you? Is it a good prediction accuracy?
- Evaluate the model using `table(knn.pred, test.data$default)`. What does it tell you? How does it relate to the test error rate above?

Fit a kNN model for k=3, 10, 20, 30, 40, 50, 100, 200.

- Evaluate all the models using the test error rate.
- Compare the evaluation results of all the models.
- Which k gives you the sweet spot between over- and underfitting?