Machine learning II

Master Data Science

Winter Semester 2022/23 **Prof. Tim Downie**

Workshop 1
Imputing missing values in R

As in Machine Learning 1 you can either use R and RStudio on the lab computers or your own computer. When using the lab computers your account from last semester should still work. If you don't have an account, please let the lecturer know.

Some weeks the worksheet will include written exercises. Usually these are intended for you to work through at home.

Download the file `ML2_Wk1_MissingData.R` from Moodle, and save it to a sensible directory.

This Workshop uses the following libraries, which you will probably need to install, if you are using your own computer. `mice`, `VIM`, and `NHANES`. Install these using

```
> install.packages("mice","VIM","NHANES")
```

Exercise 1: You will load the tropical atmosphere ocean data, do some simple data exploration and fit a linear regression on the non-missing observations.
Exercise 2: You will implement some of the imputation methods covered in the lecture using the `tao` data set (in the `VIM` library).
Exercise 3: You will use the Gibbs sampling method in the `mice` library for the `tao` data.
Exercise 4: You will use the Gibbs sampling method to impute the missing data frome the `Diabetes2` data set described in the lecture.

Exercise 1 The tropical atmosphere ocean data

The `tao` data set is a small subsample of the Tropical Atmosphere Ocean (TAO) project data, containing daily measurements at 5 locations, for two years; one El-Niño year and one La-Niña year. The variables are: `Year`, `Latitude`, `Longitude`, `Sea.Surface.Temp`, `Air.Temp`, `Humidity`, `UWind` (East-West daily average) and `VWind` (North-South daily average).

(a) After loading the data, there are some commands to inspect the number missing of missing values in each variable, and the pattern of the non missing values.

The graphics functions do strange things to the plot window margins and do not re-set them, which is bad practice for a publicly available R package. It is best to specify the margins yourself. I have

provided the code for you, but a quick explanation is:
The command `startMar<-par()$mar` saves the initial values of the plot window margins, margins so that you can return to the default settings later.
`par(mar=c(0,0,0,0)+0.1)` sets the appropriate margins for `md.pattern()` and `aggr()`.
`par(mar=startMar)` re-sets the original margin parameters once you have finished.

(b) How many observations have a missing value for *humidity*? How many observations have more than one missing value?

(c) The `marginplots` show information about the joint behaviour of two variables. In the centre is a scatter plot for the non-missing observations. On the left and bottom there are box plots for that variable, red for the observations where the other variable is missing and blue for observations where the other variable is known. Next to that is a dot plot equivalent to the red box plot. The number in the bottom left shows the number of casses missing for both variables.

There is quite a noticeable difference between the red and blue box plots implying that univariate imputation will not give good results.

(d) Fit a linear model to predict the `Sea.Surface.Temp` using the "non-missing data"; specifically those rows with no missing values.

Exercise 2 Univariate imputation

(a) Define a function which completes the missing data using the *mean replacement* method. Inspect the imputed values and compare the linear model results with the model obtained using the "known" data.

(b) Repeat using the *Mean/Variance Simulation* method

(c) Repeat using *Direct Random Sampling*.

Exercise 3 Multivariate imputation using Gibbs sampling

(a) Using multivariate imputation, we can use the information from `Year`, `Sea.Surface.Temp` etc. to get more realistic imputed values. The function `mice()` calls the Gibbs sampling routine. `maxit=50` and `m=5` has been specified. This means that 50 full iterations of the Gibbs sample are run and the last 5 will be used for the imputation. This means that we get 5 versions of the imputed data.

(b) Only the imputed values are stored in `GibbsData`. To get a full data set with known and imputed values use `complete()`.

(c) The `with()` function runs the lm() function 5 times, once for each of the imputed data sets.

N.B. `with()` is a generic function. Because GibbsData has class `"mids"` ('multiply imputed data set') the function `with.mids()` will be called. This runs the `lm()` function 5 times using each of the imputed data sets.

(d) To aggregate the results of several models, use the `pool()` function. Choose a final model.

Exercise 4 Imputing missing data for the Diabetes data set
In ML 1, Worksheet 7 you used the data set `Diabetes` which contained just 3 variables. The data were obtained from the NHANES data (American National Health and Nutrition Examination surveys) which is provided as a package in R.

The data set `Diabetes2` used in this week's lectures comes from the same source but uses 13 variables. The code to create this is given in the workshop's R code. In total there are n=10 000 observations, but there are only 6492 observations with no missing values. You will now use `mice()` to impute the missing values for these 13 variables.

(a) Load the data and store the 13 variables in the data frame `Diabetes`. Inspect the missing values.

(b) Run the Gibbs sampler on the `Diabetes2` data. Because this data set is much larger than in Exercise 3, the Gibbs sampler takes much longer to run. The total number of iterations is reduced to 10 which takes roughly 3 minutes to run. This is sufficient for a Workshop, in practice it is better to use a longer burn-in period.

(c) Fit a logistic regression model to the non-missing rows, and to the first of the imputed data sets.

(d) Fit an aggregated logistic regression model. Remove the non significant variables to obtain the final logistic regression model.

(e) This last section of code is an example of using the imputed data for binary classification of diabetes. It uses the Gibbs sampled imputed values to fit a tree classifier to the data.