ICT for Health
Laboratory # 2
Decision tree and chronic kidney disease

Monica Visintin

Politecnico di Torino

ICT4SS
Information and Communications Technologies for Smart Societies

2017/18

Table of Contents

Chronic kidney disease [1]

See the slides by Prof. Pagana for a description of the disease.

Table of Contents

Prepare the data [1]

- Download from `https://archive.ics.uci.edu/ml/datasets/chronic_kidney_disease` the Data Folder and Data Set Description.
- Use Pandas `read_csv` class to import the data frame, but note that:
  1. the initial rows of the file `chronic_kidney_disease.arff` contain the description of the features and should be skipped
  2. the file is a normal csv file that uses `,` as field separator
  3. some rows have an extra `,` so that 26 columns are read instead of 25
  4. many data are missing (identified with `?`)
  5. some columns contain numerical features, other columns contain categorical features
  6. there are "hidden" typing errors (a "yes" becomes a " yes" with an extra blank or a " yes", with an extra tab)

Prepare the data [2]

- You need to perform an initial **cleaning on the data**, which is a task that is almost always needed, takes time, and is frustrating. It is better you get used to this preprocessing phase!
- First option: you clean the data **manually**, by editing the original csv file (it works if you have very short files...)
- Second option: you **exploit the arguments of Pandas** `read_csv`.
  1. You can specify that the separator is `,` by writing `sep=','`
  2. You can skip the first 5 lines (for example) by writing `skiprows=5`
  3. You can specify that no header is present in the file (i.e. that there is no first row with the names of the features) by writing `header=None`
  4. You can specify that `?` is not a number by writing `na_values=['?','\t?']`
  5. You can specify the feature names by writing `names=feat_names` where `feat_names` is a list that contains exactly as many elements (typically strings) as the columns in the file (example feat_names=[“a”,“b”,“c”])

Classification tree in Python [1]

- Scikit Learn (library of Python) has the function that implements C4.5 (or a similar algorithm) for the hierarchical classification
- Even if the hierarchical classification based on mutual information works with **categorical data**, the **Scikit Learn implementation requires numerical data** (a clear limitation of Python). It is then necessary to map all the values of **categorical features into numbers**. You can do this exploiting the method `replace` of Pandas Dataframes
- The file has 24 features whereas the 25-th column is the class to be estimated (either “ckd” -chronic kidney disease- or “notckd”)

Classification tree in Python [2]

- The Scikit Learn class that implements hierarchical/decision trees is `tree.DecisionTreeClassifier(‘‘entropy’’)`
  You must first instantiate an object of that class, and then perform the training:
  ```
  clf = tree.DecisionTreeClassifier("entropy")
  clf = clf.fit(data, target)
  ```
  where data is the original Dataframe without the last column and target is the last column of the original Dataframe
- Two possibilities are available to manage **NaN values**:
  1. You remove the rows that contain NaN values, through method `dropna` of Pandas Dataframe
  2. You treat NaN simply as another possible value of the random variables/features, but you have to substitute it with a number that is not already present in the dataset.

  Use both methods in the lab and write your comments in the report.

Classification tree in Python [3]

- To **view the decision tree**,
  - use in the Python script:
    `dot_data = tree.export_graphviz(clf,out_file="Tree.dot", feature_names=feat_names,class_names=target_names,filled=True, rounded=True,special_characters=True)`
    where `feat_names` is the list with the names of the features (this produces a nicer tree) and the other arguments are used for aesthetic reasons
  - in the Windows/Linux shell give the command:
    `dot -Tpng Tree.dot -o Tree.png`
    which generates the file `Tree.png` with the tree
- Use the other methods of `DecisionTreeClassifier` (read the manual at `http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier`) to produce more information about the dataset (for example the "importance" of each feature, etc.)