**Software project 2 – Diabetes prediction with decision trees**

**1. Problem description**

Diabetes is a health condition that has a negative effect on the way the body decomposes food into glucose, which is characterized by the pancreas being unable to secrete sufficient amounts of insulin. As a consequence, the level of sugar in the bloodstream increases excessively, leading to problems like potential vision loss, kidney disease and more. Identifying diabetes in individuals is therefore crucial to ensure that they live with their disease under control, thus preventing any health complications.

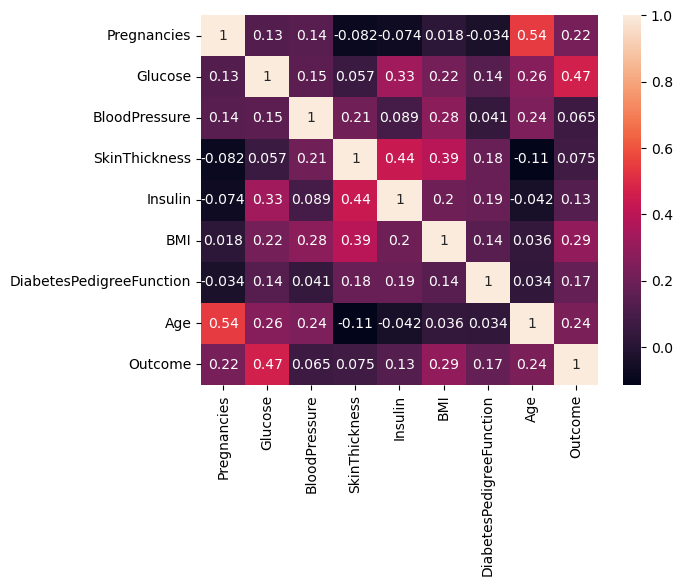
The dataset used is the popular Pima Indian Diabetes Dataset which contains several features pertaining to the health of a relatively small population of individuals, all of them females older than 21 of Pima Indian heritage. Therefore, the input data to the decision tree algorithm will be the CSV file containing the dataset, with the last column representing the verdict regarding the diabetes status of a patient, containing values of 0 for a negative prediction and 1 for a positive prediction. Knowing this, the output of the algorithm will contain either a 0 or a 1, with the same meaning conveyed by the values in the last column of the dataset, which means that the task being solved is a classification task, with two possible classes: “has no diabetes” or “has diabetes”.

Decision trees are built on the basis of a formula called the “Gini impurity”. This is a measurement which determines how the features of the dataset should split nodes to build the tree. We first have to decide which feature is going to be placed in the root node of the tree by choosing the feature with the lowest Gini impurity. Each feature will be placed in a node in the same manner, by considering the remaining features which haven’t been picked and selecting the one that yields the lowest Gini impurity. If a node is considered to be pure, the node will be a leaf of the tree. For nodes that contain numerical data instead of categorical data, the threshold will be chosen by trying each value of the feature and picking the one that yields the lowest Gini impurity.

**2. Exploratory data analysis**

As mentioned earlier, the data that will be used is the Pima Indians Diabetes dataset, a set of health measurements of 768 women. With the help of the info function from the Pandas module, we found out that there are no null values in the dataset. Two of the columns, namely BMI and DiabetesPedigreeFunction contain floating point numbers, while the rest have the integer data type.

The “Pregnancies” column contains the number of times a woman has had a pregnancy. “Glucose” represents the glucose concentration obtained from an oral glucose tolerance test. “BloodPressure” measures the diastolic blood pressure in mmHg. “SkinThickness” contains the thickness of the triceps skin fold in mm. “DiabetesPedigreeFunction” indicates the function which scores the likelihood of diabetes based on a patient’s family history. The rest of the columns, namely “Age”, “BMI” and “Insulin” are self-explanatory.



We notice that most correlations barely exceed the value of 0.5. An observation that can be deducted without the given data is that age and the number of pregnancies are positively correlated, with a value of 0.54. Another quite natural finding is that the level of blood glucose is positively correlated with the outcome of the experiment: the higher the blood sugar, the greater the chance for having diabetes. Finally, among the largest positive correlations in the heatmap is the one between the level of insulin and skin thickness, represented by a value of 0.44, and possibly related to the fact that skin thickness and the BMI of a patient are also correlated, with a value of 0.39. One interesting fact is that insulin and blood pressure are independent from each other, indicated by a correlation of 0.089. Also, the number of pregnancies doesn’t seem to affect the levels of insulin, as suggested by a negative correlation of –0.074.

As part of the solution, in order to pick the optimal amount of features necessary for learning we would have to apply a pruning algorithm to the decision tree. This means that the decision tree sometimes performs better if subtrees contained within it are trimmed. For this we will implement cost-complexity pruning. In essence, the alpha parameter of the algorithm will be varied A good idea would be to perform cross-validation, as we have a limited amount of data. Considering that we have 768 patients, we will have 8 folds, out of which 7 will be for training and 1 for validation.

**3. Design**

The implementation of the algorithm will be written in Java. There are 4 classes that should be included: a DecisionTree class that will contain the implementation of our decision tree, a Pruning class with an implementation of the cost-complexity pruning algorithm, a CrossValidation class which will have the necessary methods for splitting the dataset in 8 folds and a Main class which will glue together the previously mentioned classes.

