1. Implement ID3 algorithm by completing the following sub steps. The algorithm should take in data frame as a first argument and a column number that indicates, which of the columns is the target attribute. The target variable does not have to be a binary variable.

(a) Write a function that tabulates occurrences of the target variable in the data frame and computes frequencies. Use frequencies to compute entropy of the target variable. Hint you can use `table` function. (1p)

(b) Write a function that splits the data frame into sub-frames according to the values of a categorical variable (`factor`) and computes the corresponding information gain. (0.5p)

(c) Implement a function that splits the data frame into two sub-frames according to the values of continuous variable and computes the corresponding information gain. (0.5p)

(d) Implement a naive statistical test for determining relevance of a split. Do at least 100 time the following procedure (simulation). Randomly reorder the target variable (use `sample`). Compute the resulting information gains for the split. Do the split only if the original information gain is bigger than at least 95 information gains obtained in the simulations. (1p)

(e) Assemble all functions described above into a recursive function that implements ID3 decision tree learning. As an output, print out decision tree in tabbed format, e.g. all the same level decisions are with the same number of tabs. Test the algorithm on Iris dataset by trying to predict the class attribute. Discretize three out of four features. (3p)

(f) Modify the function defined above so that it takes in two data frames: one for training and the other for validation. Additionally, print out number of used training samples and information gains on test and training data for each node. In leaf nodes, print out number of used training samples and the error rate on test and training data. Interpret the outcome (1p).

2. Study the effect of pruning and complexity penalty on the prediction error. Use Breast Cancer Wisconsin dataset for benchmarking. Split the data randomly so that training set consists of 66% and test set of 34% samples. Use standard implementations like `rpart` or `tree` and study how corresponding `rpart.control` and `prune` functions work. Try at least one strategy that uses only training data for pruning and one that uses
only the behaviour on the test data. Report and interpret the results. (3p)

3. Study the effect of classification errors on the prediction error. Use Iris or Breast Cancer Wisconsin dataset in experiments or some other easily classifiable dataset like 3D checker board pattern. Split the data randomly so that training set consists of 66% and test set of 34% samples. On the training data randomly flip 1, 5, 10, 25, 50, 100% labels of the target value. If you consider more than two class labels then describe the flipping strategy, as well. Use two decision tree learning algorithms one with limited complexity (e.g. severely limited tree depth) and other without limitations. Do several experiments and present the average behaviour together with some variance estimate (e.g. \textit{boxplot}). Visualise and interpret results. (2p)

4. The main drawback of a decision tree are rectangular decision borders. For continuous variables, such borders do not make much sense. Hence, one could be more intelligent and use more complex borders. Use linear SVM-s (\texttt{e1071} package) as a black-box method for splitting the data into halves according to class labels. After that train a new SVM on each part separately until all elements are correctly classified or some other stopping criterion is met. The resulting decision tree should work much better on datasets with complex borders. Build such a decision tree and test it on 2D examples: 45° tilted checker board and ball patterns. Compare your algorithm with ID3 algorithm. Report and interpret results. (5p)