Grundlagen der Künstlichen Intelligenz I

Summer Term 2025

2 – Decision Trees and Regression Trees in Practice

Submissions are due via **eCampus** by **May 8, 2025**[1] until **12:00**!

Solutions will be discussed on **May 15, 2025**

Teaching Assistant(s)

Simon Kraft	s49skraf@uni-bonn.de
Felix Frederik Müller	s6fdmuel@uni-bonn.de
Denis Jan Schafranski	s6dsscha@uni-bonn.de

Predicting Trisomy in Mice

In the following tasks 1–5, you will apply decision tree models to a practical biological problem.

Biological Background

Gene expression is the process of transcribing a segment of DNA to an RNA and translating this RNA to a protein. Since proteins are the "workhorse" of cells, gene expressions are of central importance to living organisms. By *genes* we mean segments of DNA that may be transcribed to RNA, which, in turn, is translated into a protein.[2] An interesting property is that genes may be turned "on" or "off". We say that a gene is turned "on" (resp. "off") if the corresponding DNA-segment is transcribed and translated (resp. *not* transcribed and therefore the corresponding protein is not produced). Whether a gene is turned "on" or "off" is influenced by various factors, such as the presence of certain diseases, drugs, environmental factors, lifestyle etc. The above model of genes being turned "on" or "off" is extremely simplified. Since genes are not strictly turned "on" or "off", their states, called *gene expression levels*, are represented by continuous values in practice.

To study the genetic background of a particular disease, researchers often use *animal models*. For example, mice are surprisingly similar to humans on the level of molecular biology. The *goal* of this project is to learn a decision tree distinguishing between healthy and trisomic mice on the basis of gene expression levels. A mouse is *trisomic* if it has three instances of a particular chromosome, instead of the normal two. Such a decision tree could be used not only for predicting trisomy, but also to understand the relationship between trisomy and gene expressions.

Data

In the archive `gene-expression-datasets.zip` that you can download from eCampus there are several CSV files.[3] Please consider two of them:

- `gene_expression_training.csv` that contains the training data and
- `gene_expression_test.csv` that contains the test data.

In case of both files, each line corresponds to an instance of the dataset (except for the first line which is the header).

- The *header* (first line) shows the attribute names separated by commas. Each attribute name corresponds to a gene, except for the last one, which is the class attribute indicating if the mouse is healthy ("0") or trisomic ("1").

[1] There will be no lecture or tutorial on May 1, 2025.

[2] Note that we present the biological background in a very simplified way.

[3] All data used in this project are based on the *Mouse Protein Expression Data* from the UCI Repository (`http://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression`). Note, however, that the data has been preprocessed (genes and instances with missing values have been removed). Furthermore, we split it into train and test subsets, therefore, we ask you to work with the files provided on eCampus.

- For each *labeled example* (i.e., row of the table), we give the expression levels of the genes as well as the value of the class attribute. The values are separated by commas.

1 Learning a Decision Tree Model

Programming **10 points**

We start by learning a basic decision tree classifier for trisomy in mice.

1.1 Learn a decision tree model for the *training* data ("gene_expression_training"). The maximum depth of the tree should be 3 (without counting the root node as a separate level). Use the entropy-based information gain as a heuristic.

1.2 Visualize the decision tree you obtained in the previous task. There are several options for plotting the tree with its branching conditions and leaf nodes, e.g., using `matplotlib` as demonstrated in the last tutorial. Just make sure that the visualization is clear and that

- the inner nodes include their split condition and the number of positive and negative examples in the particular branch and
- the leaf nodes show the assigned class and the number of positive and negative examples.

1.3 Classify the instances of the test data using your decision tree (i.e., predict the class of each test instance) and report the accuracy (i.e., the proportion of correctly classified instances) of your predictions.

1.4 Based on your decision tree, which genes are differently expressed in case of trisomic and healthy mice? For each gene that is used in your decision tree, search for "trisomy" together with the name of the gene in GoogleScholar[4], and take the number of scientific papers returned. For which gene name is this number maximal? Report your findings in your Jupyter notebook.

2 Examine Overfitting

Programming **9 points**

For this task, recall the following definition of overfitting from the Lecture 2025-04-24:

Definition (overfitting)**.** *A hypothesis $h \in \mathcal{H}$* overfits *the training data E if there exists an alternative hypothesis $h' \in \mathcal{H}$ such that*

$$\text{error}_E(h) < \text{error}_E(h') \ ,$$

but

$$\text{error}_{\mathcal{D}}(h) > \text{error}_{\mathcal{D}}(h') \ .$$

2.1 Measure the accuracy of your previously constructed decision tree on the *training data* as well (i.e., repeat task **1.3**, but use the training data instead of the test data).

2.2 Construct two other decision trees with (maximally) 5 and 10 levels respectively, and measure their performance both on the training data and test data.

2.3 Do you observe overfitting?

3 The Effect of Measurement Noise

Programming **8 points**

3.1 Generate a "noisy" version of the training data by adding random numbers to each measurement in the training data (i.e., to each gene expression level of each instance). The random numbers should be sampled from a Gaussian distribution with

1. $\mu = 0, \sigma = 0.1$, and
2. $\mu = 0, \sigma = 0.25$, respectively.

(That is, you are asked to consider two different cases: one with relatively low level of noise, and another one with relatively high level of noise.)

[4] http://scholar.google.com

3.2 Construct a decision tree using the "noisy" training data created in task **3.1**.

3.3 Measure the accuracy of the decision tree on the test data.

4 The Effect of Label Noise

Programming 8 points

4.1 "Flip" each class label of the *training data* with a probability of

1. 0.1, and
2. 0.25, respectively.

(With "flipping", we mean that the label indicating trisomy, is replaced by the label indicating a healthy mouse, and vica versa.) This way, you generate a "noisy" version of the labels of the training data.

4.2 Construct a decision tree using the training data with "noisy" labels.

4.3 Measure the accuracy of the decision tree on the test data. (Please note that you should *not* flip the labels of the test data!)

5 Handling Missing Values

Programming 5 points

5.1 Some of the genes were excluded from the homework, because their expression levels contained missing values. Please consider the full dataset

`gene_expression_with_missing_values.csv`

that contains missing values, and use this data in the subsequent analysis. (In case of a missing value, the delimiter (comma) is repeated.)

5.2 Replace the missing values by

1. a random value uniformly chosen from the interval $[0, 1]$,
2. the average of the expressions levels of the respective gene, and
3. the median of the expression levels of the respective gene.

5.3 Construct decision trees using the data produced in the above cases and compare these trees: did the decision trees identify the same gene as being the most important as before?

Regression Trees

Notice that tree-based models can also be used for regression tasks. In fact, the learning algorithm for so-called *regression trees* works in a very similar fashion as that for decision trees.

6 Learn About Regression Trees

8 points

Search for regression trees online and learn how they work and what commonly-used algorithms there are for fitting regression tree models.

6.1 Explain the algorithm and demonstrate how to use it using, e.g., the off-the-shelf implementation in `sklearn`.

6.2 Compare regression trees and decision trees. What are similarities? What is different?

7 Regression Trees in Practice

Programming 12 points

In this exercise, we will use two datasets from `sklearn` library: the California housing dataset and the diabetes dataset.

```
from sklearn.datasets import fetch_california_housing, load_diabetes
```

7.1 Split the *California housing* data into train/test subsets using an 80/20 split ratio.

7.2 Learn a *regression tree* model for the training data. Use `GridSearchCV` to find the best *max_depth* value in the range $[2, 20]$.

7.3 Evaluate the two fitted models using Mean Squared Error (MSE).

7.4 Repeat the model fitting and evaluating steps for the *diabetes* dataset. Compare the performance of the two models on both datasets.