First you have to form a team. Each team should consist of 3 students that attend the course. Choose a name for your team. Be creative! Organize the infrastructure that allows all the team members working on the project/source code. Send an email to *tatiana.gossen@ovgu.de* until the 4th of May 2015 with information about your team, i.e. a list of members and the team name. Send both the source code as well as documentation to *tatiana.gossen@ovgu.de* at the latest on 8am on Monday, 15th of June 2015. The results have to be presented (5-10min, slides) at one of the two exercise classes, either on 22th of June or 29th of June 2015.

Assignment 1.1

Use the Diabetes 130-US hospitals for years 1999-2008 Data Set[1]. This dataset contains records with 55 features for more than 100000 patients. 55 features include information about the diabetic encounters, including demographics, diagnoses, diabetic medications, number of visits in the year preceding the encounter, and payer information. Note that even though some attributes are codified using numeric values, they are nominal (not numeric) attributes. Browse additional information about the attributes in the paper cited on the dataset website[2].
The goal of this assignment is to find suitable methods in the area of machine learning to determine the readmission attribute of a patient and to estimate the quality of selected approaches. In order to achieve this goal, the following tasks have to be completed:

- analyse the data set and its various attributes
- clean the data (e.g. missing values)
- select an appropriate subset of the attributes and explain your choice
- use different suitable machine learning algorithms (either implement them, or use existing libraries, e.g. Weka)
- determine the quality of your model (e.g. through cross-validation, log loss[3], confusion matrix)
- compare your results between different algorithms

Specifically, different suitable machine learning algorithms should include:

- at least two classification algorithms, one of which, SVM, you learn during the course
- at least one algorithm for semi-supervised classification. Use a part of the dataset as unlabeled data for learning (omit the label).

[1] https://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008

[2] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore. Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records. BioMed Research International, 2014.

[3] https://www.kaggle.com/wiki/LogarithmicLoss

The documentation of your solution should contain details on experiments you performed. Furthermore, document how to use your program. Provide technical details about your solution, details about classes, libraries you used (and what for), algorithms.

Instructions for Programming Assignments

Correctness and Completeness of the Solution

The solution of a programming assignment has to be correct and complete. *Eclipse* should be used as a development environment. It will be used to run your solutions. *Java* programming language has to be used for assignments. You are allowed to use Open source Java code libraries only.

Understandability of the Solution

The concept and approach of the solution must be sketched in a **separate document**, so that it can be understood without difficulties. This does not need to be a very long text, but it must become clear how the task was solved.

Style of Submission

The program must **run without any corrections or modifications** of the runtime environment (Java 8) or source code (e.g. use relative instead of absolute paths). A short **note on how to execute the code** has to be placed in a separate file (e.g. a README.txt file containing the java command, a make file, ant-script, etc.). Everything necessary to run the program must be handed in. The archive may not include any other files that are irrelevant for this task.

Programming Style

The code must be understandable without difficulties. This can be mainly achieved through **well structured source code** (e.g. splitting code into multiple methods or classes; following naming conventions), and an **appropriate documentation**. This can be done in the source code itself using e.g. the javadoc syntax. Undocumented code that is difficult to read and understand may result in a significant reduction of the score, since the correctness and completeness of the solution can not be verified anymore.

Strongly recommended naming conventions in Java:
http://en.wikipedia.org/wiki/Naming_convention_%28programming%29#Java