Examination:
Programming for Data Science (ID2214)

Course code: ID2214
Course name: Programming for data science
Literature and tools:
The following rules apply to both parts of the examination:
Literature, other documents, including lecture slides, notes, etc. are not allowed.
Computers, tablets, phones, etc. are not allowed.
Date and time: April 17, 2019, 14:00-18:00
Examiner: Henrik Boström

15 points are required to pass.
On part I, keep the text short and to the point.
The answers (including blank ones) should be numbered and ordered.
Unreadable answers will be ignored.
Good luck!

Part I (Theory, 10 points)

1a. Methodology, 2 points

Assume that we have a dataset that we would like to use for training a model as well as evaluating its performance. If we decide to randomly split the data into a training and a test set only once, what tradeoff needs to be considered when choosing the relative size of the two subsets?

1b. Data preparation, 2 points

Give one reason for why equal-sized binning may be preferred to equal-width binning, and conversely, one reason for why equal-width binning may be preferred to equal-sized binning.

1c. Performance metrics, 2 points

Assume that we have generated a regression model for predicting the outdoor temperature using a large training set. However, we then find out that the mean-squared error of the model is significantly higher than of a default model, which just predicts the average outdoor temperature in the training set. Could our model still be more useful for prediction tasks than the default model? Explain your reasoning.

1d. Decision trees, 2 points

Assume that we attempt to normalize numerical features using min-max normalization prior to learning a decision tree. What effect will this preprocessing step have on the outcome of the decision tree learning algorithm? Consider both the case when numeric features are discretized before tree generation and during tree generation.

1e. Association rule mining, 2 points

Is an association rule $A \rightarrow C$ always preferable to an association rule $B \rightarrow C$, if the confidence of the former is higher than the latter? Explain your reasoning.

Part II (Programming, 20 points)

2a. Data preparation, 10 points

Your task is to define the following Python function for feature selection:

```
select_features(df,no_features)
```

which given a dataframe `df`, where the columns correspond to features (except for a column named `CLASS`) and the rows to instances, and an integer `no_features` (which defaults to 5), returns a dataframe which includes the column named `CLASS` and the `no_features` other columns that have received the highest score, according to a (pre-defined) function

```
evaluate(feature_values,class_labels)
```

which returns a numeric score, given an array of values for some feature, as well as an array of class labels (which is the same for all features). In case of any ties, i.e., several features receiving the same score, these may be handled arbitrarily.

Hint: you may use `result.sort(reverse=True)` to sort a list of items `result` in a descending order (in case these items are tuples, they are sorted element-wise in order).

2b. Combining models, 10 points

Assume that you have been provided with a definition of the following Python function:

```
regression_tree(df,depth)
```

which given a pandas dataframe `df`, where the columns correspond to features, except for a column named `REGRESSION`, which contains the target values, and the rows correspond to instances, and an integer `depth` ≥ 0 (which defaults to 3), returns a regression tree of depth `depth`.

Also assume that you have been provided with the following:

```
predict(df,regression_tree)
```

which given a pandas dataframe `df` and a regression tree `regression_tree` (see above), returns an array of predictions, i.e., one predicted numerical value for each instance in the dataframe.

Your task is to define the following function:

```
gbm(df,depth,no_trees)
```

which given a pandas dataframe `df` and an integer `depth` (see above) returns a gradient boosting machine, i.e., a list of regression models $M_0, \ldots, M_{no\_trees}$, where M_o is the average regression value in the dataframe, and the regression trees (of depth `depth`) $M_1, \ldots, M_{no\_trees}$ are formed in sequence from the same dataframe, except for that when generating a regression tree M_i the regression value for each instance should consist of the residual, i.e., the original regression value for the instance minus the sum of the predictions of the previous models $(M_0, \ldots, M_{i-1})$ for the instance.

Hint: You may keep track of the predictions of the previous models in the sequence in an array `predictions`, which is of the same length as the number of regression values (found in the dataframe column named `REGRESSION`).