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 for searching for answers or communicating with anyone except for the examiner. A text editor/word processor (on a computer, tablet or phone) may be used for writing answers to the questions, and on part II, any tool, such as an editor, Jupyter notebook or IDE may be used for developing the programs.
Date and time: April 14, 2020, 08:00-12:00
Examiner: Henrik Boström

Requirements to pass: 5 points on part I and 10 points on part II.

On part I, keep the text short and to the point.
On part II, only the Python standard library, the NumPy and pandas libraries may be assumed, in addition to functions explicitly stated in the tasks.
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 want to develop a model and estimate its performance on independent data. We have therefore have decided to randomly split an available dataset into a training and test set, using the former to train the model and the latter to estimate its performance. However, since the learning algorithm that we would like to use cannot directly deal with missing values, we have decided to employ some imputation technique prior to applying the algorithm. The question is now whether there may be any potential risk in applying the imputation technique using the whole dataset, before randomly splitting the data into training and testing. Explain your answer.

1b. Data preparation, 2 points

Assume that we want to discretize numerical features prior to applying the decision-tree learning algorithm. Will it have any effect on the resulting tree if we employ equal-width or equal-sized binning? Explain your reasoning.

1c. Performance metrics, 2 points

Assume that we have a binary classification model that, given a test instance, outputs an estimate of the probability for the positive class. By choosing some other threshold than 0.5 for whether to assign a positive or negative label to the test instances, the predictive performance may be affected. Should we increase or decrease the threshold to increase a) precision and b) recall, of the positive class? Explain your answer.

1d. Combining models, 2 points

The predictive performance of an ensemble of classifiers, for which the predictions are formed by averaging predictions of the individual members, is dependent on the diversity of the members. Describe how an ensemble of naïve Bayes classifiers would be trained, if similar techniques that are used to form random forests would be employed.

1e. Association rules, 2 points

Assume that we have generated a set of association rules with a specified support and confidence, from a dataset with a set of binary features and binary class labels, encoded as itemsets. Assume that we have selected a subset of the rules, for which the heads (consequents) contain only a class label. If we want to use this subset of rules to classify a novel test instance, i.e., to assign one of the two class labels, what are the potential problems we may encounter? Explain your reasoning.

Part II (Programming, 20 points)

2a. Data preparation, 10 points

Your task is to define the following Python function that aggregates multiple rows with the same identifier into a single row:

```
aggregate(df)
```

which given a pandas dataframe `df`, where the columns correspond to numerical features, except for a column named `CLASS`, which contains (categorical) class values, and a column named `ID`, which contains identifiers for the instances. In case one or more instances share the same identifier, these should in the new dataframe be represented by a single row, where the identifier should appear in the `ID` column, and the value for each numerical feature of the instance should be the mean of the values for instances sharing the identifier, and the value for the class label of the instance should be the mode of the values for instances sharing the identifier (in case the mode is not unique, one of the mode values

may be chosen arbitrarily).

For example, given a dataframe `df`:

```
   ID V1 V2 CLASS
0  1  1  1  A
1  1  2  0  A
2  2  3  1  B
3  2  4  0  C
4  3  5  1  C
```

then `aggregate(df)` should return the following dataframe:

```
   ID V1   V2   CLASS
0  1  1.5  0.5  A
1  2  3.5  0.5  B
2  3  5.0  1.0  C
```

Hint: You may obtain unique values from a pandas data series `values` by `values.unique()`, the mean by `values.mean()` and the first mode value by `values.mode()[0]`.

2b. Combining models, 10 points

Your task is to define the following Python function:

```
stacking(level0_df,level1_df,base_learners,learner)
```

which given two pandas dataframes `level0_df` and `level1_df`, with the same set of columns, which correspond to features, except for a column named `CLASS`, which contains the class values, and where the rows correspond to instances, a list of learning algorithms `base_learners` (see below), and a learning algorithm `learner` (see below), will return a list of base models and a stacking model, where each base model is trained on `level0_df` using the corresponding base learning algorithm, and the stacking model is trained using the class labels of `level1_df` and the predictions of the base models on `level1_df` as features.

Each learning algorithm `alg`, i.e., as specified by `learner` and the elements of `base_learners` above, can be used to generate a model from a dataframe by:

```
model = alg.fit(df)
```

Each resulting model can be used to make predictions (produce a list of class labels) for a dataframe (in which the `CLASS` column is ignored) by:

```
predictions = model.predict(df)
```

For example, given two dataframes `df0` and `df1`, with the same columns, and the learning algorithms `alg1,alg2,alg3,alg4`, then:

```
stacking(df0,df1,[alg1,alg2,alg3],alg4)
```

should return a list of three (base) models trained using `alg1,alg2,alg3` on `df0`, and a (stacking) model trained using `alg4` on a dataframe with four columns; one column for each base model with the predictions for `df1`, and one column containing the class labels of `df1`.