Lab 3: Supervised Learning: Classification Algorithms

Due dates:
Part 1: Tuesday, October 19, 10:00pm
Part 2: Tuesday, October 26, 10:00pm

Lab Assignment

This is a pair programming lab. If you want to, you can keep your partner from Labs 1 and/or 2. After this lab, we will force a change of partners. A team of three people will incur additional incremental tasks as part of this lab.

The lab consists of two parts, each with its own set of deliverables and tasks to complete. Both parts of the lab are assigned at the same time to allow each team additional flexibility in working on the assignments, and to give you all a road map of all classification works you are to complete for this course.

For Part 1 of the Lab you have to implement the version of C4.5 decision tree induction algorithm that works only with categorical independent variables. You shall

1. implement C4.5 in your language of choice (Python or Java are suggested languages but you can pick any language)
2. output the decision tree in a ”proprietary” format
3. implement the classification algorithm that uses the output of your C4.5 implementation
4. implement cross-validation
5. run your implementation on three datasets, report results.

For Part 2 of the Lab you shall

1. extend your implementation of C4.5 to allow your program to handle numeric attributes
2. implement a Random Forest classification algorithm that uses your C4.5 implementation to build individual decision trees.
3. implement the K Nearest Neighbors classifier
4. test and run your implementations on several datasets
5. hypertune your Random Forest classifier to produce highest accuracy results on each dataset.
6. report results of your investigations

Datasets

We will be using a number of datasets from the UCI (Univeristy of California at Irvine) Machine Learning Dataset Repository. Each of the datasets has a long history of being used for classification. For Part 2 of the lab you are also using some of the datasets uploaded to Kaggle (kaggle.com).

Below is a list of the datasets provided with brief description of each. Please note, for most datasets, a file with additional information is also made available. In the interest of saving space in this document, I will not repeat the information found in those files - you should consult them in order to determine the variables provided to you and their domains.

All datasets can be downloaded from the Lab 3 data page:

http://users.csc.calpoly.edu/∼dekhtyar/466-Fall2019/labs/lab03.html

We also provide links to the UCI Repository or Kaggle pages for each dataset. Please note, that in some cases, we have changed file names, and, modified the files themselves to make parsing easier (You can, however, use any files as input). You are also allowed to modify the format of input files to your liking (adding or removing header lines, etc...)

Data Format

To give you a little bit of help when parsing data, the datasets are all provided in a modified CSV format. The data itself is comma-separated rows, like standard CSV format. However, the first three lines of the file have to be parsed separately.

The **first line** contains the names of all columns (a standard practice when creating CSV data files).

The **second line** provides information about the domain of each variable. The line contains a comma-separated list of integer numbers. The meaning of the numbers is as follows:

-1	The column is a `rowId`. It should be ignored by classificaiton algorithms.
0	The column is a numeric variable.
$n > 0$	number of possible values of the variable.

The **third line** contains the name of the *class variable*. For all problems studied in this lab, the class variable is unique for each dataset.

For example, the following header of your data file:

```
No,Name,Income,Sex,Education,Occupation,Rating
-1,-1,0,2,5,7,3
Rating
```

describes a data file that contains seven columns, of which the first two (`No` and `Name`) are essentially row Ids (or otherwise contain meta data not necessary for classification), the next four columns (`Income`, `Sex`, `Education` and `Occupation`) represent independent variables, with `Income` being a numeric attribute, while the remaining three attributes having two, five, and seven different levels (values) respectively[1]. The last column, `Rating` is a class variable, and it has three different levels (values).

Part 2 Datasets

In Part 2 of the Lab, you will run your Random Forest implementations on Mushroom and Nursery datasets (you can also run KNN classifiers that use distance metrics for categorical attributes). In addition, you will also use the following datasets, which contain numeric indepenent variables.

Iris Dataset

The Iris dataset is one of the most popular Machine Learning datasets. It is a simple dataset containing a few hundred records. Each record depicts physical dimensions of a specific iris flower from one of three different subspecies of iris: *Iris Setosa*, *Iris Versicolor*, or *Iris Virginica*. There are four physical parameters measured for each flower, listed below in the order in which they occur in the data file:

- Sepal length in cm
- Sepal width in cm
- Petal length in cm
- Petal width in cm

[1]Historically, government and businesses used binary attributes to record sex and/or gender. As most available datasets come from such sources, many legacy datasets use binary identifiers for sex and/or gender and also occasionally use "sex" and "gender" as synonyms.

The dataset is available from the UCI Machine Learning repository. Lab 2 data page,

http://users.csc.calpoly.edu/∼dekhtyar/466-Spring2018/labs/lab02.html

contains the links to the data page and the data file.

Letter Recognition Dataset

The Letter Recognition dataset[2] contains 20,000 records each describing 16 features extracted from an image of a letter. The class variable is the letter itself (26 letter of latin alphabet). The data file, `letter-recognition.data.csv` contains 17 columns. The first column is the letter, the remaining columns are the 16 extracted features (described in the `letter-recognition.names.txt` files). All features are numeric (integer).

Wine Quality Dataset

The Wine Quality dataset[3] contains information about the chemical properties of around 4900 red and white varieties of Portuguese Vinho Verde wine. Each wine is described by 11 numeric features (such as acidity, citric acid, residual sugar, etc..), followed by the class variable describing the quality of the wine. The class variable takes values 0,1,. . . , 10. The goal is to predict from the chemistry of the wine its quality.

Note that there are two separate datasets, `winequality-red-fixed.csv` and `winequality-white-fixed.csv` corresponding to red and white wine collections. Because red and white wines have very different chemical profiles, treat these two datasets as independent and separate, i.e., build separate classifiers for red and white wine.

Credit Approval Dataset

The Credit Approval dataset [4] contains information about credit approval decisions of 690 individuals. Due to the sensitive nature of the data, not only are the identities of the individuals hidden, but the 15 features describing the credit applicants and their applications are obscured. The features are named A1 through A15, feature A16 is the class variable taking two values: `"+"` for approved application and `"-"` for rejected application. The independent variables (features) are a mix of numeric (continuous) variables and categorical variables. The domains of all variables are also abstracted, and are specified in the `crx.names.txt` file.

Your goal is to predict whether the credit application is approved or rejected.

Heart Disease Dataset

The Heart Disease dataset [5] is a small (920 data points) dataset of information about a group of patients who sought medical care for chest pain symptoms. Some of the patients were diagnosed with a heart disease, while others were not. The HeartDisease binary variable is the category variable in this dataset, your goal is to predict whether the patient had heart disease or not. There are 11 total features (see Kaggle web page for the full list of attributes and possible values) that are a mix of numeric (age, blood pressure, cholesterol levels, etc.) and categorical (sex, cardiogram results, etc.).

Part 2 Tasks

Task 1: Complete C4.5 implementation

Your first task is to complete your `C4.5` implementation by adding the functionality to classify on numeric attributes. A number of datasets presented to you in the lab contain numeric attributes, making it impossible for you to complete the rest of the lab assignment unless your `C4.5` algorithm implementation handles numeric attributes properly.

Note on data format: When splitting on a numeric attribute, your output JSON format needs to undergo a slight update. A node that is split on the value x of a numeric attribute A will have the following structure

[2]https://archive.ics.uci.edu/ml/datasets/Letter+Recognition.
[3]https://archive.ics.uci.edu/ml/datasets/Wine+Quality
[4]https://archive.ics.uci.edu/ml/datasets/Credit+Approval
[5]https://www.kaggle.com/fedesoriano/heart-failure-prediction

```
"node": { "var": <attributeName>,
         "edges": [ {"edge": {"value": <x>,
                              "direction": "le",
                              ...
                              },
                    },
                    {"edge": {"value": <x>,
                               "direction": "gt",
                                ...
                               }
                    }
                  ]
          }
```

Here, "direction" is the new JSON attribute used only in the edges representing split on numeric features, and its values are "le" for ”less than or equal to” (the path representing the dataset where the values of the feature are less than or equal to the split value) and "gt" for ”greater than” (for the path representing the dataset where the values of the feature are greater than the split value).

Task 2: Random Forset implementation

Implement the Random Forest classifier. Your implementation shall behave as follows.

Input Parameters. Your implementation shall take as input the following parameters:

- m or NumAttributes: this parameter controls how many attributes each decision tree built by the Random Forest classifier shall contain.
- k or NumDataPoints: the number of data points selected randomly with replacement to form a dataset for each decision tree.
- N or NumTrees: the number of the decision trees to build.

Dataset Selection. Write a method/function that given a full dataset D and the parameters NumAttributes and NumDataPoints selects the appropriate number of data points, and randomly selects NumAttributes and returns the constructed set back. This functionality will be used by your Random Forest classifier.

Behavior. Your Random Forest implementation shall take as input a training set D, and the three input parameters described above. It shall produce the requisite number of small decision trees, as guided by the input parameters. Each decision tree shall be produced by an appropriate call to your C4.5 implementation.

Evaluation. You will use 10-fold cross-validation to compute the accuracy of the Random Forest classifier on the datasets you select. (Strictly speaking, cross-validation is not necessary when evaluating Random Forests, but it is more complicated to evaluate Random Forests in other ways, and you already should have cross-validation implemented). Essentially, for each fold, you construct a Random Forest out of the remaining nine folds, and then classify the data points from the holdout fold. Each fold winds up with its own Random Forest built according to the input parameters of your process.

The program. Your Random Forest implementation shall be named randomForest.py or randomForest.java (or a similar file name for a programming language of your choice). It shall take as input the dataset filename, and the three input parameters described above. It shall produce, as output, a results.txt or results.csv file that produces predictions for each individual row in the dataset based on the 10-fold cross-validation evaluation. Separately, it shall also output the confusion matrix and the accuracy of prediction.

Task 3: K Nearest Neighbors

Your final task is to implement the KNN classifier. This classifier shall take one parameter, K - the number of neighbors to consider. Please note, that because KNN is a lazy classifier, there is no training step. Because of this, your implementation shall consist of a single program (`knn.py` or `knn.java`) which takes as input a dataset file, the parameter K, and any additional arguments necessary for your program to properly parse and evaluate the dataset.

The program shall produce the prediction for each input data point, and format the output in the same way as the outputs of your C4.5 and Random Forest classification programs.

For numeric attributes, implement standard distance measures (Manhattan, Eucledian distance). If you do not like their performance, you can look into cosine similarity measure.

For categorical attributes, you can choose either of the solutions we discussed. You can implement any of the similarity measures for categorical attributes from the handout, or you can convert categorical variables into numeric using one hot encoding or other dummification procedures. If implementing multiple distance/similarity functions, make sure that they can be triggered by an input parameter to your program.

Evaluation and Report

With the three implemented methods you shall conduct an evaluation study. This study has two goals:

1. Find the best hyperparameter values for each of the three classifiers for each of the chosen datasets.
2. Compare the accuracy of the three classifiers to each other for each of the chosen datasets.

Dataset use. You shall conduct your study using all dataset presented to you in Part 2 of the Lab. *Do not use any Part 1 datasets.*

Report. Write and submit a report documenting your findings. In your report, describe the details of your implementation of C4.5, Random Forest and KNN classifiers, provide the description of your evaluation procedures for each of the methods, and describe the comparative study of your implementations and the results. Discuss your findings.

More specifically, for your classifier evaluation procedures include the following information:

- For each dataset, specify which hyper-parameters you were tuning, and what values of the hyper-parameters you were considering.
- For each dataset, provide the observed accuracy results for your study in tabular (confusion matrix), and (if you can) graphical form.
- For each dataset, include the discussion of what how the tuning went, and how the best model performed.

For your comparative study,

- Specify exactly which hyper-parameter settings for each method you used in your comparison.
- Describe the cross-validation procedure you used.
- Provide results in tabular and (if you can) graphical form.
- Provide the analysis of the accuracy of your methods on each of the datasets.
- Provide overall analysis of your methods - which tended to perform better on all datasets, and which tended to perform worse?

Submission Instructions

The following is an **updated** list of deliverables. New deliverables are *in italics.*

- `README`. Shall contain the names and email addresses of all students in the team. Also, put any specific instructions and notes in this file. (e.g., if you choose a different implementation language, include translation/running instructions). *Include any instructions on how to run your classifers.*
- All your programs implementing C4.5, Random Forest and KNN classifiers.
- Outputs of your classifiers on the datasets you chose for the best values of hyper-parameters (include the list of output files in your README file with appropriate hyperparameter values)
- Your written report submitted as a PDF document named `Lab3-report.pdf`.

Submit the `README` file and your report as separate files. Submit the rest of your files in a single archive named either `lab03.zip` or `lab03.tar.gz`.

To submit use the following commands:

Section 01:

```
$ handin dekhtyar lab03-2-466-01 <files>
```

Section 03:

```
$ handin dekhtyar lab03-2-466-03 <files>
```