Assignment 2: Classification

Deadlines

Submission: 5pm, Friday 18th May, 2018 (week 10)

This assignment is worth 20% of your final mark.

Task description

In this assignment you will implement the Naïve Bayes and Decision Tree algorithms and evaluate them on a real dataset using the stratified cross validation method. You will also evaluate the performance of other classifiers on the same dataset using Weka. Finally, you will investigate the effect of feature selection, in particular the Correlation-based Feature Selection method (CFS) from Weka.

Late submissions policy

No late submissions are allowed.

Programming languages

Your implementation can be written in Python, Java, C, C++ or MATLAB. The assignment will be tested on the University machines, so your code must be compatible with the language version installed on those machines. You are not allowed to use any of the built-in classification libraries for the purposes of this assignment.

Submission and pair work

Your assignment can be completed individually or in pairs. See the submission details section for more information about how to submit.

This assignment will be submitted using the submission system PASTA (https://comp3308.it.usyd.edu.au/PASTA/). In order to connect to the website, you'll need to be connected to the university VPN. You can read this page to find out how to connect to the VPN. PASTA will allow you to make as many submissions as you wish, and each submission will provide you with feedback on each of the components of the assignment. Your last submission before the assignment deadline will be marked, and the mark displayed on PASTA will be the final mark for your code (12 marks).

1. Data

The dataset for this assignment is the Pima Indian Diabetes dataset. It contains 768 instances described by 8 numeric attributes. There are two classes - **yes** and **no**. Each entry in the dataset corresponds to a patient's record; the attributes are personal characteristics and test measurements; the class shows if the person shows signs of diabetes or not. The patients are from Pima Indian heritage, hence the name of the dataset.

A copy of the dataset can be downloaded from Canvas. There are 3 files associated with the dataset. The first file, **`*.names`**, describes the data, including the number and the type of the attributes and classes, as well as their meaning. The second file, **`*.data`**, contains the numerical data described in the `*.names` file. The last file, `*.discrete`, is the same data after having been discretised. Your task is to predict the class, where the class can be **yes** or **no**.

Note: The original dataset can be sourced from UCI Machine Learning Repository. However, you need to use the dataset available on Canvas as it has been modified for consistency.

2. Data preprocessing

Read the `pima-indians-diabetes.names` file and learn more about the meaning of the attributes and the classes. Use Weka's in-built normalisation filter to normalise the values of each attribute to make sure they are in the range [0,1]. The normalisation should be done along each column (attribute), not each row (entry). The class attribute is not normalised – it should remain unchanged. Save the preprocessed file as `pima.csv`.

Warning: In order to ensure that Weka can process the data, you will need to add headers to the data file and save it as a .csv file. You can do this in any text editor. The headers should be removed after preprocessing.

3. Classification algorithms

Naïve Bayes

The Naïve Bayes should be implemented for numeric attributes, using a probability density function. Assume a normal distribution, i.e. use the probability density function for a normal distribution. If there is ever a tie between the two classes, choose class **yes**.

Decision Tree

The Decision Tree classifier should be implemented for nominal attributes, and will be evaluated using the nominal data provided to you. Use Information Gain to choose which attribute to split on at each stage of the tree, and do not implement any pruning. When building the Decision Tree, use the definitions of Information Gain and stopping conditions as provided to you in the lectures. As before, if there is ever a tie between the two classes, choose class **yes**.

Note: Carefully read section 6 to find out how your program will be expected to receive input and give output.

4. 10-fold stratified cross-validation

In order to evaluate the performance of the classifiers, you will have to implement 10-fold stratified cross-validation. Your program should be able to show the algorithm's average accuracy over the 10 folds. This information will be required to complete the report.

Your implementation of 10-fold stratified cross-validation will be tested based on your `pima-folds.csv` file. The information about the folds should be stored in `pima-folds.csv` in the following format for each fold:

- Name of the fold, fold1 to fold10.
- Contents of the fold, with each entry on a new line.
- A single blank line to separate the folds from each other.

An example of the **pima-folds.csv** file would look as follows (made up data):

```
fold1
0.588,0.628,0.574,0.263,0.136,0.463,0.054,0.333,yes
0.243,0.274,0.224,0.894,0.113,0.168,0.735,0.321,no

fold2
0.588,0.628,0.574,0.263,0.136,0.463,0.054,0.333,yes
0.243,0.274,0.224,0.894,0.113,0.168,0.735,0.321,no

...
fold10
0.588,0.628,0.574,0.263,0.136,0.463,0.054,0.333,yes
0.243,0.274,0.224,0.894,0.113,0.168,0.735,0.321,no
```

Note: The number of instances per fold should not vary by more than one. If the total number of instances is not divisible by ten, the remaining items should be distributed amongst the folds rather than being placed in one fold.

5. Feature selection

Correlation-based feature selection (CFS) is a method for selecting a subset of the original features (attributes). It searches for the best subset of features, where best is defined by a heuristic which considers how good the individual features are at predicting the class and how much they correlate with the other features. Good subsets of features contain features that are highly correlated with the class and uncorrelated with each other.

Load the **pima.csv** file in Weka, and apply CFS to reduce the number of features. It is available from the "Select attributes" tab in Weka. Use "Best-First Search" as the search method. Save the CSV file with the reduced number of attributes (this can be done in Weka) and name it **pima-CFS.csv**. Repeat the same for the discretised data and save the result as **pima-discretised-CFS.csv**.

Warning: As before, in order to ensure Weka can understand the data, you'll need to add headers. Once you are done processing, remove the headers

6. Input and output

Input

Your program will need to be named **MyClassifier**, however may be written in any of the languages mentioned in the "Programming languages" section.

Your program should take 3 command line arguments. The first argument is the path to the training data file, the second is the path to the testing data file, and the third is the name of the algorithm to be executed (**NB** for Naïve Bayes and **DT** for the Decision Tree).

For example, if you were to make a submission in Java, your main class would be **MyClassifier.java**, and the following are possible inputs to the program:

```
$ java MyClassifier pima.csv examples.csv NB
$ java MyClassifier pima-CFS.csv examples.csv DT
```

The input testing data file will consist of several new examples to test your data on. The file will not have headers, will have one example per line, and each line will consist of a value for each of the non-class attributes separated by commas. The values will either be normalised and numeric if the required classifier is **NB** or nominal if the required classifier is **DT**.

An input file might look like this for NB:

```
0.58,0.62,0.57,0.26,0.13,0.46,0.05,0.33
0.24,0.27,0.22,0.89,0.11,0.16,0.73,0.32
0.73,0.29,0.92,0.11,0.69,0.66,0.48,0.52
```

Or like this for DT:

```
high,low,high,high,high,low,high,low
low,high,low,high,low,high,low,high
high,very high,high,high,high,low,high,low
```

The following examples show how the program would be run for each of the submission languages, assuming we want to run the NB classifier, the training data is in a file called training.txt, and the testing data is in a file called testing.txt.

Python (version 3.5.3):

```
python MyClassifier.py training.txt testing.txt NB
```

Java (version 1.8):

```
javac MyClassifier.java
java MyClassifier training.txt testing.txt NB
```

C (gcc version 6.3.0):

```
gcc –lm -w -std=c99 –o MyClassifier MyClassifier.c *.c
./MyClassifier training.txt testing.txt NB
```

C++ (gcc version 6.3.0):

```
g++ –c MyClassifier.cpp *.cpp *.h
gcc –lstdc++ –o MyClassifier *.o
./MyClassifier training.txt testing.txt NB
```

MATLAB (R2017b):

```
mcc -m -o MyClassifier -R -nodisplay -R -nojvm MyClassifier
./run_MyClassifier.sh <MATLAB_directory> training.txt testing.txt NB
```

Note: MATLAB must be run this way (compiled first) to speed up MATLAB running submissions. The arguments are passed to your MyClassifier function as strings. For example, the example above will be executed as a function call like this:

MyClassifier('training.txt', 'testing.txt', 'NB')

Output

Your program will output to standard output (a.k.a. "the console"). The output should be one class value (**yes** or **no**) per line – each line representing your program's classification of the corresponding line in the input file. An example output should look as follows:

```
yes
no
yes
```

Note: These outputs are in no way related to the sample inputs given above. If you have any questions or need any clarifications about program input or output, ask a question on Piazza or ask your tutor. Since your program will be automatically tested by PASTA, it is important that you follow the instructions exactly.

Your program should also be able to output the Decision Tree that is build when running the Decision Tree classifier. This output will NOT be tested automatically, however should only be outputted if the user requests it so (for example with a certain argument flag). DO NOT make your program output this diagram every time, as it will interfere with the testing system's ability to process your output.

7. Weka evaluation

In Weka select 10-fold cross validation (it is actually 10-fold stratified cross validation) and run the following algorithms: ZeroR, 1R, k-Nearest Neighbor (k-NN; IBk in Weka), Naïve Bayes (NB), Decision Tree (DT; J48 in Weka), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM; SMO in Weka) and Random Forest (RF). Select the nominal data when evaluating the DT classifier.

Compare their performance of the Weka's classifiers with your Naïve Bayes and Decision Tree classifiers (again using nominal data for the DT and numeric for NB). Do this without feature selection (using **`pima.csv`** or relevant nominal data) and with CFS feature selection (using **`pima-CFS.csv`** or **`pima-discretised-CFS.csv`**).

8. Report

You will have to describe your analysis and findings in a report similar to a research paper. Your report should include 5 sections. There is no minimum or maximum length for the report – you will be marked on the quality of the content that you provide.

Aim

This section should briefly state the aim of your study and include a paragraph about why this study is important.

Data

This section should describe the dataset, mentioning the number of attributes and classes. It should also briefly describe the CFS method and list the attributes selected by the CFS.

Results and discussion

The accuracy results should be presented (in percentage, using 10-fold cross validation) in the following two tables where MyNB and MyDT are your implementations of the NB and DT algorithms, evaluated using your stratified 10-fold cross validation. For the DT classifier in Weka, present two results – for an unpruned tree and for a pruned tree (using the default pruning option).

Classifier accuracy

Numeric Data	ZeroR	1R	1NN	5NN	NB	MLP	SVM	RF	MyNB
No feature selection									
CFS									

Nominal Data	DT unpruned	DT pruned	MyDT
No feature selection			
CFS			

DT diagrams

Build DTs using the whole training data and include a DT diagram for each of the three DT classifiers (MyDT and the two Weka versions, pruned and unpruned). To do this in Weka, select “Test option: Use training set”.

As the diagram for unpruned trees can be very large, leaving the diagram as a text-based diagram (like the ones produced in Weka) is acceptable.

Discussion

In the discussion, compare the performance of the classifiers, with and without feature selection. Compare your implementations of NB and DT with Weka’s.

Discuss the effect of the feature selection – did CFS select a subset of the original features, and if so, did the selected subset make intuitive sense to you? Was the feature selection beneficial, i.e. did it improve accuracy, or have any other advantages? Why do you think this is the case?

How does your DT classifier compare with the unpruned and pruned DT generated by Weka? Discuss the role and effect of pruning.

Include anything else that you consider important.

Conclusion

Summarise your main findings and, if possible, suggest future work.

Reflection

Write one or two paragraphs describing the most important thing that you have learned throughout this assignment.

9. Submission Details

This assignment is to be submitted electronically via the PASTA submission system.

Individual submissions setup

The first thing you must do is create an individual group on PASTA. This is due to a limitation of PASTA. To create a group, follow the instructions below:

1. Click on the "Group Management" button (3 people icon), next to the submit button.
2. Click on the plus button in the bottom right to add a new group.
3. Scroll to the bottom of the list of groups and click on "Join Group" next to the group you just created.
4. Click on "Lock Group" to lock the group and stop others from joining the group (optional).

Pair submissions setup

The first thing you must do is create/join a group on PASTA. Follow the instructions below:

1. Click on the "Group Management" button (3 people icon), next to the submit button.
2. If your pair has not yet formed a group on PASTA, click on the plus button in the bottom right to add a new group, otherwise go to step 3.
3. Click on "Join Group" next to your group in the "Other Existing Groups" section.
4. If you wish to stop anyone from joining your group, click on "Lock Group".

All submissions

Your submission should be zipped together in a single .zip file and include the following:

- The report in PDF format.
- The source code with a main program called `MyClassifier`. Valid extensions are `.java`, `.py`, `.c`, `.cpp`, `.cc`, and `.m`.
- Four data files: **`pima.csv`**, **`pima-CFS.csv`**, **`pima-folds.csv`** and **`pima-discretised-CFS.csv`**.

A valid submission might look like this:

```
submission.zip
|- pima.csv
|- pima-folds.csv
|- pima-CSF.csv
|- pima-discretised-CFS
|- report/
|  +- report.pdf
|- MyClassifier.java
+- extrapackage/
   |- MyClass.java
   +- OtherClass.java
```

Upload your submission on PASTA under **Assignment 2 – Classification (Advanced)**. Make sure you tick the box saying that you're submitting on behalf of your group (even if you're working individually). The submission won't work if you don't.

10. Marking criteria

[12 marks] Code – based on the tests in PASTA; automatic marking

[8 marks] Report:

[0.5 marks] Introduction

- What is the aim of the study?
- Why is this study (the problem) important?

[0.5 marks] Data – well explained

- Dataset – brief description of the dataset
- Attribute selection – brief summary of CFS and a list of the selected attributes

[4 marks] Results and discussion

- All results presented
- Correct and deep discussion of the results
- Effect of the feature selection – beneficial or not (accuracy, other advantages)
- Comparison between the classifiers (accuracy, other advantages)
- Comparison between the DT classifiers and discussion of pruning

[1.5 marks] Conclusions and future work

- Meaningful conclusions based on the results
- Meaningful future work suggested

[0.5 marks] Reflection (meaningful and relevant personal reflection)

[1 mark] English and presentation

- Academic style, grammatical sentences, no spelling mistakes
- Good structure and layout; consistent formatting