The University of Melbourne
School of Computing and Information Systems
COMP30027 Machine Learning, 2020 Semester 1

Project 1: Discrete and Continuous Naïve Bayes

Due:	7pm, 20 Apr 2019
Submission:	Source code (in Python) and written responses
Marks:	The Project will be marked out of 20, and will contribute 20% of your total mark. This will be equally weighted between implementation and responses to the questions.
Groups:	You may choose to form a group of 1 or 2. Groups of 2 will respond to more questions, and commensurately produce more implementation.

Overview

In this Project, you will implement a supervised naïve Bayes learner and evaluate it with respect to various supervised datasets. You will then use your observations to respond to some conceptual questions about naïve Bayes.

Naive Bayes classifiers

There are some suggestions for implementing your learner in the "Naïve Bayes" and "Discrete & Continuous" lectures, but ultimately, the specifics of your implementation are up to you. Your implementation must be able to perform the following functions:

- `preprocess()` the data by reading it from a file and converting it into a useful format for training and testing
- `train()` by calculating prior probabilities and likelihoods from the training data and using these to build a naive Bayes model
- `predict()` classes for new items in a test dataset (for the purposes of this assignment, you can re-use the training data as a test set)
- `evaluate()` the prediction performance by comparing your model's class outputs to ground truth labels

Your implementation should be able to handle both nominal and numeric attribute types in the same dataset. You can assume numeric attributes are Gaussian-distributed. When handling discrete attributes, you should implement some type of smoothing to ensure the likelihoods are greater than zero. Your implementation should actually compute the priors, likelihoods, and posterior probabilities for the naïve Bayes model and **may not simply call an existing implementation such as `GaussianNB` from `scikit-learn`**. A Jupyter Notebook template has been provided on the LMS that you can use as a starting point for your implementation.

so do bayes without discretizing, then q1 is about discretizing

Data

For this assignment, we have adapted some of the classification datasets available from the UCI machine learning repository (`https://archive.ics.uci.edu/ml/index.html`). In all of these datasets, the task is classifcation, but the attribute types vary:

Datasets with nominal attributes only:

- `breast-cancer-wisconsin`
- `mushroom`
- `lymphography`

Datasets with numeric attributes only:

- `wdbc`
- `wine`

Datasets with ordinal attributes only:

- `car`
- `nursery`
- `somerville`

Datasets with a mix of attribute types:

- `adult`
- `bank`
- `university`

These datasets vary in terms of number of instances and number of classes, in addition to the number and type of attributes. More information is provided in the README file included with the datasets. You are not required to use all of these datasets in your submission, however it is strongly recommended that you use multiple datasets to answer the questions below. Different datasets will produce different results, so if you only test your algorithm on one or two datasets, you may arrive at an incorrect conclusion due to a small sample space.

Questions

The following problems are designed to pique your curiosity when running your classifier(s) over the given data sets:

1. Try discretising the numeric attributes in these datasets and treating them as discrete variables in the naïve Bayes classifier. You can use a discretisation method of your choice and group the numeric values into any number of levels (but around 3 to 5 levels would probably be a good starting point). Does discretizing the variables improve classification performance, compared to the Gaussian naïve Bayes approach? Why or why not?

2. Implement a baseline model (e.g., random or 0R) and compare the performance of the naïve Bayes classifier to this baseline on multiple datasets. Discuss why the baseline performance varies across datasets, and to what extent the naïve Bayes classifier improves on the baseline performance.

3. Since it's difficult to model the probabilities of ordinal data, ordinal attributes are often treated as either nominal variables or numeric variables. Compare these strategies on the ordinal datasets provided. Deterimine which approach gives higher classification accuracy and discuss why.

4. Evaluating the model on the same data that we use to train the model is considered to be a major mistake in Machine Learning. Implement a hold–out or cross–validation evaluation strategy (you should implement this yourself and do not simply call existing implementations from `scikit-learn`). How does your estimate of effectiveness change, compared to testing on the training data? Explain why. (The result might surprise you!)

5. Implement one of the advanced smoothing regimes (add-k, Good-Turing). Does changing the smoothing regime (or indeed, not smoothing at all) affect the effectiveness of the naïve Bayes classifier? Explain why, or why not.

6. The Gaussian naïve Bayes classifier assumes that numeric attributes come from a Gaussian distribution. Is this assumption always true for the numeric attributes in these datasets? Identify some cases where the Gaussian assumption is violated and describe any evidence (or lack thereof) that this has some effect on the NB classifier's predictions.

If you are in a group of 1, you will respond to question (1), and one other of your choosing (two responses in total). If you are in a group of 2, you will respond to question (1) and question (2), and two others of your choosing (four responses in total). A response to a question should take about 150–250 words, and make reference to the data wherever possible. Note that not all questions are equally difficult. Also note that not all questions are equally interesting. `(-:`

Submission

Submission will be made via the LMS. Please submit your code and written report separately:

- Your code submission should be a .zip or .tar.gz file which includes your code, results files, and any additional files we would need to run your code and replicate your results. (You don't need to include the datasets that we provided, but you should include any custom datasets you created yourself, or code to recreate your train/test splits.) Please also include a README file that tells us how to run your code and recreate your results.

- Your written report should be uploaded separately as a .pdf, using the Turnitin submission link.

If you worked in a group, please include both group members' names on the written report and in your code file (in the README file or a `group.txt` file).

Please note that **the deadlines on the LMS submission page may be after the assignment deadline**. We set these deadlines late to accomodate late submissions. The true deadline is the one listed at the top of this assignment specification.

Late submission

The submission mechanism will stay open for one week after the submission deadline. Late submissions will be penalised at 10% per 24-hour period after the original deadline. Submissions will be closed 7 days (168 hours) after the published assignment deadline, and no further submissions will be accepted after this point.

Assessment

10 of the marks available for this assignment will be based on the implementation of the naïve Bayes classifier, specifically the five Python functions specified above. Any other functions you've implemented will not be directly assessed, unless they are required to make these five functions work correctly.

10 of the marks will be assigned to accurate and insightful responses to the questions, divided evenly among the questions that you are required to attempt. We will be looking for evidence that you have an implementation that allows you to explore the problem, but also that you have thought deeply about the data and the behaviour of the relevant classifier(s).

Updates to the assignment specifications

If any changes or clarifications are made to the project specification, these will be posted on the LMS.

Academic misconduct

You are welcome — indeed encouraged — to collaborate with your peers in terms of the conceptualisation and framing of the problem. For example, we encourage you to discuss what the assignment specification is asking you to do, or what you would need to implement to be able to respond to a question.

However, sharing materials beyond your group — for example, plagiarising code or colluding in writing responses to questions — will be considered cheating. We will invoke University's Academic Misconduct policy (`http://academichonesty.unimelb.edu.au/policy.html`) where inappropriate levels of plagiarism or collusion are deemed to have taken place.

Data references

Census income dataset (`adult`) is thanks to:

Ronny Kohavi and Barry Becker
Data Mining and Visualization
Silicon Graphics
`http://archive.ics.uci.edu/ml/datasets/Adult`

Bank marketing dataset (`bank`) is thanks to:

S. Moro, P. Cortez and P. Rita
A Data-Driven Approach to Predict the Success of Bank Telemarketing
Decision Support Systems, Elsevier, 62:22-31, June 2014
`http://archive.ics.uci.edu/ml/datasets/Bank+Marketing`

Wisconsin breast cancer dataset (`breast-cancer-wisconsin, wdbc`) is thanks to:

Dr. William H. Wolberg, General Surgery Dept.
W. Nick Street, Computer Sciences Dept.
Olvi L. Mangasarian, Computer Sciences Dept.
University of Wisconsin
`http://archive.ics.uci.edu/ml/machine-learning-databases/`
`breast-cancer-wisconsin/`

Car evaluation dataset (`car`) is thanks to:

Marko Bohanec (creator, donor)
Blaz Zupan (donor)
`http://archive.ics.uci.edu/ml/datasets/Car+Evaluation`

Nursery dataset (`nursery`) is thanks to:

Vladislav Rajkovic et al. (creator)
Marko Bohanec (donor)
Blaz Zupan (donor)
`http://archive.ics.uci.edu/ml/datasets/Nursery`

Lymphography dataset (`lymphography`) is thanks to:

Igor Kononenko, University E.Kardelj (donor)
Bojan Cestnik, Jozef Stefan Institute (donor)
`https://archive.ics.uci.edu/ml/datasets/Lymphography`

Mushroom dataset (`mushroom`) is thanks to:

Jeff Schlimmer (donor)
`https://archive.ics.uci.edu/ml/datasets/Mushroom`

Somerville Happiness Survey dataset (`somerville`) is thanks to:

Waldemar W. Koczkodaj
`http://archive.ics.uci.edu/ml/datasets/Somerville+Happiness+`
`Survey`

University dataset (`university`) is thanks to:

Steve Souders (donor)
`http://archive.ics.uci.edu/ml/datasets/University`

Wine dataset (`wine`) is thanks to:

Forina, M. et al, PARVUS (creator)
An Extendible Package for Data Exploration, Classification and Correlation
Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy
Stefan Aeberhard (donor)
`http://archive.ics.uci.edu/ml/datasets/Wine`