The University of Melbourne
School of Computing and Information Systems
COMP30027 Machine Learning, 2022 Semester 1

Naïve Bayes Leaner for `Adult` Database

Due:	7 pm, 8 April 2022 (week 6, Fri)
Submission:	Source code (in Python)
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.
Marks:	The project will be marked out of 16 points (individual project) or 24 points (group project). In either case, this project will contribute 20% of your total mark.
Main contact:	Ni Ding (email: ni.ding@unimelb.edu.au)

1 Overview

In the UCI machine learning repository (Asuncion and Newman, 2007), the `Adult` database was extracted from the census bureau database found at `http://www.census.gov/ftp/pub/DES/www/welcome.html`. In `adult.csv`, we have extracted the records and attributes for assignment 1.[1] The file `adult.csv` contains in total 1,000 samples/instances each of them has 11 attributes below and a class label to indicate whether the income is over 50K or not.

```
age    work class    education    education num    marital status
   occupation    relationship    race    sex    hours per week
                          native country
```

The question mark "`?`" indicates missing value.

In this project, you will build a **Naïve Bayes learner/classifier** to classify the income. You will train, test, and evaluate your classifier on `adult.csv`. You will then answer some conceptual questions by exploring and interpreting the results and extending the basic model in certain ways.

2 Naïve Bayes classifier

For m being the number of attributes. The Naïve Bayes classifier is

$$\hat{c} = \arg\max_{c_j} P(c_j) \prod_{i=1}^{m} P(X_i|c_j) \tag{1}$$

where X_i denotes attribute i and c_j denotes the value of class label. There are only two values of class label c_j for $j \in \{1, 2\}$: c_1 ="`<=50K`" and c_2 ="`>50K`". Your implementation of the Naïve Bayes classifier must be able to perform the following functions:

[1] Do not download the original adult dataset from the UCI machine learning repository. Just use the `adult.csv` provided on LMS/Assignment and dataset overview in this assignment spec.

- `preprocess()` the data by reading it from `adult.csv` and converting it into a useful format for training and testing
  - Implement 90-10 splitting: use the first 90% records for training and the remaining 10% records for testing. Do not shuffle the records before data-splitting.
- `train()` by calculating probabilities in (1).
  - For *nominal attributes*, treat the missing value as a new category; For *numeric attributes*, you should fit a Gaussian distribution to the conditional probability $P(X_i|c_j)$. Your implementation should actually compute the prior and conditional probabilities for the naïve Bayes model and not simply call an existing implementation such as `GaussianNB` from `scikit-learn`. You should have only one `train` function dealing with both nominal and numeric attributes.
- `predict()` classes for new items in the tesing data.
- `evaluate()` the prediction performance by comparing your model's class outputs to ground truth labels. This function should return and print
  - the accuracy, a 2×2 confusion matrix in the form of[2]

		Predicted	
		Positive	Negative
True	Positive	TP	FN
	Negative	FP	TN

  and the F_1 score. Choose `Positive` for class `<=50K` and `Negative` for class `>50K`.

You will be given an iPython notebook template `COMP3027ASS1.ipynb`. You should use this template to implement the four functions above.

3 Questions

The following problems are designed to pique your curiosity when running your classifier(s) over the given dataset and suggest methods for improving or extending the basic model.

- ***Individual***: if you are in a group of 1, you should respond to Q1 and Q2;
- ***Group***: if you are in group of 2, you should respond to Q1, Q2, Q3 and Q4.

You should write your answers in `COMP3027ASS1.ipynb`. A response to a question should take about 150–250 words, and make reference to the data wherever possible. We strongly recommend including figures or tables to support your responses.

Q1 Sensitivity and specificity are two model evaluation metrics. A good model should have both sensitivity and specificity high. Use the 2×2 confusion matrix returned by `evaluate()` to calculate the sensitivity and specificity. Do you see a difference between them? If so, what causes this difference? Provide suggestions to improve the model performance.

[2] You do not need to plot the index or column name of this confusion matrix. But, we will assume the row and column indices refer to the true and predicted labels, respectively.

Q2 You can adopt different methods for training and/or testing, which will produce different results in model evaluation.

(a) Instead of Gaussian, implement KDE for $P(X_i|c_j)$ for numeric attributes X_i. Compare the evaluation results with Gaussian. Which one do you think is more suitable to model $P(X_i|c_j)$, Gaussian or KDE? Observe all numeric attributes and justify your answer.

You can choose an arbitrary value for kernel bandwidth σ for KDE, but a value between 3 and 15 is recommended. You should write code to implement KDE, not call an existing function/method such as `KernelDensity` from `scikit-learn`.

(b) Implement 10-fold and 2-fold cross-validations. Observe the evaluation results in each fold and the average accuracy, recall and specificity over all folds. Comment on what is the effect by changing the values of m in m-fold cross-validation.[3]

Q3 In `train()` in Section 2, you are asked to treat the missing value of nominal attributes as a new category. There is another option (as suggested in Thu lecture in week 2): ignoring the missing values. Compare the two methods in both large and small datasets. Comment and explain your observations. You can extract the first 50 records to construct a small dataset.[4]

Q4 In week 4, we have learned how to obtain information gain (IG) and gain ratio (GR) to choose an attribute to split a node in A decision tree. We will see how to apply them in the Naïve Bayes classification.[5]

(a) Compute the GR of each attribute X_i, relative to the class distribution.[6] In the Naïve Bayes classifier, remove attributes in the ascending order of GR: first, remove $P(X_i|c_j)$ such that X_i has the least GR; second, remove $P(X_{i'}|c_j)$ such that $X_{i'}$ has the second least GR,....., until there is only one X_{i*} with the largest GR remaining in the maximand $P(c_j)P(X_{i^*}|c_j)$. Observe the change of the accuracy for both Gaussian and KDE.[7] Describe and explain your observations.

(b) Compute the IG between each pair of attributes. Describe and explain your observations.Choose an attribute and implement an estimator to predict the value of `education num`. Explain why you choose this attribute. Enumerate two other examples that an attribute can be used to estimate the other and explain the reason.

4 Implementation tips

In the training phase of your algorithm, you will need to set up data structures to hold the prior probabilities $P(c_j)$ for all classes c_j's and the likelihoods/conditional probability $P(X_i|c_j)$ for each attribute X_i in each class c_j. For $P(X_i|c_j)$ being a KDE, you need to store all sample values of attribute X_i with class label c_j; For $P(X_i|c_j)$ being modeled by Gaussian distribution, it suffices to store two parameters: a *mean* and a *standard deviation* for each attribute X_i and class c_j. In both

[3] You can choose either Gaussian or KDE Naïve Bayes for cross-validation

[4] Use Gaussian Naïve Bayes for Q3.

[5] You do not need to answer this question, but is worth thinking: why you are asked to compute GR for Q4(a), but just IG for Q4(b)?

[6] `adult.csv` contains integer numeric attributes only. To get GR, apply the same method as shown in Tue lecture in week 4, i.e., counting the occurences.

[7] Choose bandwidth $\sigma = 10$ for KDE. You do not need to implement cross-validation for Q4(a).

cases, a 2D array may be a convenient data structure to store these parameters. But, you are free to choose other data structures.

Multiplying many probabilities in the range $(0, 1]$ can result in very low values and lead to *underflow* (numbers smaller than the computer can represent), e.g, you could have the value of the maximand $P(c_j)\prod_{i=1}^{m} P(X_i|c_j)$ in (1) in the order of 10^{-19}. When implementing a Naïve Bayes model, it is strongly recommended to apply $\arg\max$ to the logarithm of the maximand: by doing so, it is clear that (1) is equivalent to

$$\hat{c} = \arg\max_{c_j} \left(\log P(c_j) + \sum_{i=1}^{m} \log P(X_i|c_j) \right). \tag{2}$$

5 Submission

Complete and submit `COMP3027ASS1.ipynb` via LMS. If you are working in a group, please include both group members' student id numbers in `COMP3027ASS1.ipynb`.

- The submitted `COMP3027ASS1.ipynb` must be executable in the Jupiter Notebook environment. Otherwise, we will not be able to mark your assignment. Please use kernel Python 3. The markers are not able to use different kernels of IDEs to run your code. Please note that it is your responsibility to make your submission executable.
- You should not submit another file. All figures and tables supporting the question answers, e.g., inserted in the plain text in the Markdown cells, must be repreducable by your code. Please make your code clean and readable: you should add comments and specify how to load the file `adult.csv`.

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.

6 Assessment

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

Each question is worth 4 marks. 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).

Because the number of questions depends on the group size, individual projects can receive a total of 16 marks and group projects can receive a total of 24 marks. In both cases, the project will contribute 20% of the final mark in this subject. In group projects, both members of the group will receive the same mark.

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.

References

Asuncion, A. and Newman, D. (2007). UCI machine learning repository https://archive.ics.uci.edu/ml/index.php.