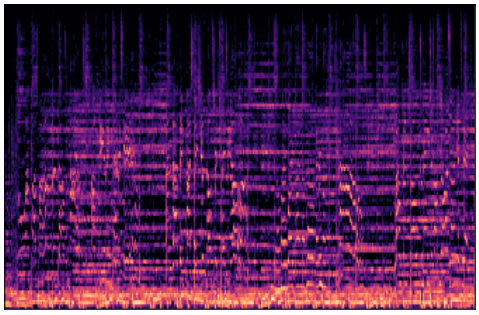
The University of Melbourne
School of Computing and Information Systems
COMP30027 Machine Learning, 2023 Semester 1

Project 1: Music genre classification with naïve Bayes

Due:	7 pm, 7 April 2021
Submission:	Source code (in Python) and written responses
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.

Overview

State-of-the-art AI research is focused on developing computer systems that can recognize and understand text, images, and audio in the ways that humans do. A classic problem in audio AI is the problem of music genre classification, which is useful for applications like music recommendation systems. Given a piece of music, how do we interpret what "type" of music it is (e.g., pop, classical, hip-hop, or jazz)? This task is challenging for computers because the artists, styles, and features of music within a genre can be quite varied, and songs from different genres may share some features.



A visualisation (mel spectrogram) of a music clip from the GZTAN dataset [3].

In this project, you will implement a supervised naïve Bayes learner to classify the genre of a music clip from high-level acoustic features. You will train, test, and evaluate your classifier on a provided dataset, and then you will have a choice of either extending this basic model in various ways, or using it to answer some conceptual questions about naïve Bayes.

Data

The data for this assignment is drawn from the GTZAN music genre dataset [1], a dataset for music genre classification. It consists of 1000 30-second mp3 audio clips from 10 different classes (100 samples per class). The classes are blues, classical, country, disco, hip hop, jazz, metal, pop, reggae, and rock. For this assignment, we'll use a processed version of the dataset from Kaggle [2], which provides 57 high-level acoustic features [3] extracted from the music clips. You do not need the original audio files for this assignment, though if you are interested, you can download them through Kaggle.

Separate training and test datasets are provided. Please use the provided train/test splits for this assignment, unless a question asks you to create your own splits. Each row in the dataset is a music

clip with the class label given in the `label` column.

Naive Bayes classifier [4 marks]

There are some suggestions for implementing your learner in the "Naïve Bayes" and "Discrete & Continuous Data" 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 likelihood parameters from the training data and using these to build a naive Bayes model
- `predict()` classes for new items in a test dataset
- `evaluate()` the prediction performance by comparing your model's class outputs to ground truth labels

Your implementation should be able to handle numeric attributes and it should assume that numeric attributes are Gaussian-distributed. Your model will not be expected to handle nominal attributes.

Your implementation should actually compute the priors, likelihoods, and posterior probabilities for the naïve Bayes model. You may use built-in functions to read data and compute Gaussian probabilities. However, you must implement the naïve Bayes algorithm yourself and not simply call an existing implementation such as `GaussianNB` from `scikit-learn`.

Task 1. Pop vs. classical music classification [8 marks]

Use the `pop_vs_classical_train.csv` dataset to train your naïve Bayes model and then evaluate it on the `pop_vs_classical_test.csv` dataset. Answer questions 1-2 below in a short write-up (no more than 250 words total).

1. Compute and report the accuracy, precision, and recall of your model (treat "classical" as the "positive" class). **[3 marks]**
2. For each of the features X below, plot the probability density functions $P(X|Class = pop)$ and $P(X|Class = classical)$. If you had to classify pop vs. classical music using just one of these three features, which feature would you use and why? Refer to your plots to support your answer. **[5 marks]**
    - `spectral_centroid_mean`
    - `harmony_mean`
    - `tempo`

Task 2. 10-way music genre classification [4 marks (individual) or 12 marks (group of 2)]

Use the `gztan_train.csv` dataset to train your naïve Bayes model and then evaluate it on the `gztan_test.csv` dataset. If you are working in a **group of 1, answer 1 of the questions** below for 4 marks. If you are working in a **group of 2, answer 3 of the questions** below for 12 marks. Each question response should be no more than 250 words and include figures and/or tables as appropriate. When reporting the performance of a model, you should include a breakdown of the performance over categories in addition to reporting the overall accuracy.

3. Compare the performance of the full model to a 0R baseline and a one-attribute baseline. The one-attribute baseline should be the best possible naïve Bayes model which uses only a prior and a single attribute. In your write-up, explain how you implemented the 0R and one-attribute baselines. **[4 marks]**

4. Train and test your model with a range of training set sizes by setting up your own train/test splits. With each split, use cross-fold validation so you can report the performance on the entire dataset (1000 items). You may use built-in functions to set up cross-validation splits. In your write-up, evaluate how model performance changes with training set size. **[4 marks]**

5. Implement a kernel density estimate (KDE) naïve Bayes model and compare its performance to your Gaussian naïve Bayes model. You may use built-in functions and automatic ("rule of thumb") bandwidth selectors to compute the KDE probabilities, but you should implement the naïve Bayes logic yourself. You should give the parameters of the KDE implementation (namely, what bandwidth(s) you used and how they were chosen) in your write-up. **[4 marks]**

6. Modify your naïve Bayes model to handle missing attributes in the test data. Recall from lecture that you can handle missing attributes at test by skipping the missing attributes and computing the posterior probability from the non-missing attributes. Randomly delete some attributes from the provided test set to test how robust your model is to missing data. In your write-up, evaluate how your model's performance changes as the amount of missing data increases. **[4 marks]**

Implementation tips

In the training phase of your algorithm, you will need to set up data structures to hold the prior probabilities for each class, and the parameters needed to compute likelihoods $P(x_i|c_j)$ for each attribute x_i in each class c_j. Recall that you will need two parameters (mean and standard deviation) to define the Gaussian distribution for each attribute $\times$ class. A 2D array may be a convenient data structure to store these parameters.

You are allowed to use built-in functions to compute the Gaussian pdf, but these functions tend to be slow, so you may prefer to write your own function to do this. The probability density function for a Gaussian with mean of μ and standard deviation of σ is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

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). When implementing a naïve Bayes model, it is strongly recommended to take the $log()$ of each probability and sum them instead of multiplying. E.g., instead of computing:

$$P(c_j) \prod_i P(x_i|c_j) \tag{2}$$

compute:

$$log(P(c_j)) + \sum_i log(P(x_i|c_j)) \tag{3}$$

Submission

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

- Your code submission should use the provided .ipynb notebook template. Your submission must include comments or a README section that explain how to run your code so we can reproduce your results.
- Your written report should be uploaded separately as a .pdf, using the Turnitin submission link.

If you are working in a group, please include both group members' student id numbers on the written report and in your code file (in the README file or a group.txt file).

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

4 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.

The questions should be answered in a written .pdf report, and will be marked as indicated above. 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

[1] George Tzanetakis and P Cook. Gtzan genre collection. Web resource, 2001. `http://marsyas.info/downloads/datasets.html`

[2] Andrada Olteanu, James Wiltshire, Lauren O'Hare and Minyu Lei. GTZAN Dataset - Music Genre Classification. Web resource, 2020. `https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification`

[3] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing, 10(5):293–302, 2002.