CSCM35, CSLM35 Big Data and Data Mining

by Dr. Jingjing Deng

Released on 17th Feb 2022

The assignment consists of multiple tasks that are designed **to be completed during the lab sessions and signed off by either module instructor or teaching assistant**. If you are not able to complete the tasks during the lab sessions, then you should do them at home and have them ready to be marked off by the deadline. **All lab tasks must be uploaded to Canvas before the deadline stated on each assignment sheet. Source codes must be written in Jupyter Notebook and formatted neatly with sufficient and clear comments.** Plagiarism will not be tolerated. Zip all your files with the following naming convention for submission:

- [Student Number]-[Last Name][First Initial]-[Assignment][Number].zip
- For example: *123456-DengJ-Assignment2.zip*

CSCM35/CSLM35 Assignment 2 Complete by 4th/Mar/2022

This assignment is about getting familiar with an important supervised classification algorithm, Naïve Bayes classifier that was taught in the module.

Iris Flower Dataset

The Iris flower data set or Fisher's Iris dataset is a multivariate dataset introduced by the British statistician and biologist Ronald Fisher in his 1936 paper. The dataset contains a set of 150 records under five attributes - petal length, petal width, sepal length, sepal width and species. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor), and four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. A scatter plot of Fisher's Iris dataset is illustrated in Fig 1. The dataset is provided with *Python Scikit-Learn* package. More details on the data structure of the dataset and how to load using *Python* can be found from `https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html`. In order to perform accuracy evaluation, you should split the dataset into two subsets for training (40 samples per category) and testing (10 samples per category). These two subsets should not have any overlap.

□ Task – Classification with Naïve Bayes (10 marks)

Naïve Bayes classifiers are a family of simple"probabilistic classifier" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. In our case, we assume that the length and the width of the sepals and petals are independence random variables. The details of Naïve Bayes algorithm can be found at Section 8.3 of our recommended textbook (*Data Mining Concepts and Techniques, 3rd Edition*).

- **Cross Validation**: Split the dataset into two subsets, one for training (40 samples per category) and one for testing (10 samples per category). **(1/10 mark)**

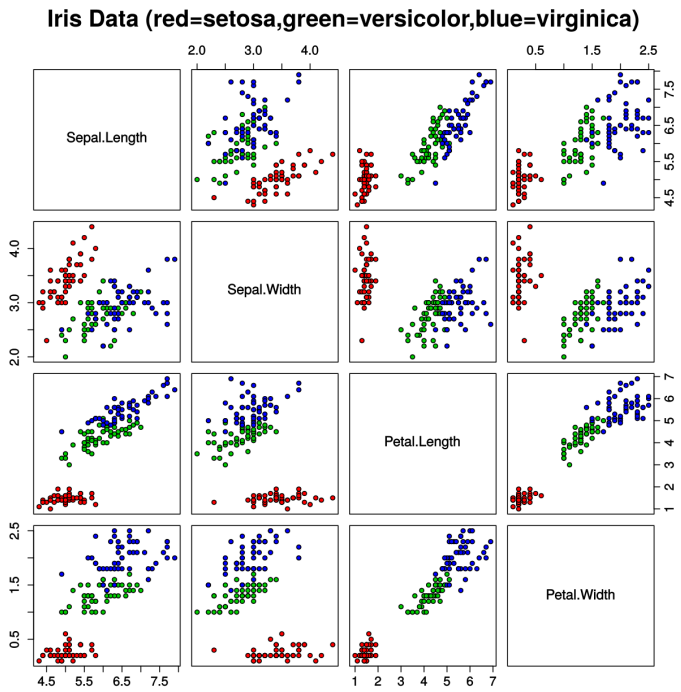

Figure 1: Scatterplot of Fisher's Iris dataset.

- **Likelihood** $P(X_i|C_j)$: For individual feature of each category, construct a probabilistic model using Gaussian distribution, where you need to evaluate the mean and standard deviation of each feature across the samples which belong to the category. X_i represents a feature and C_j represents a category. **(2/10 marks)**
- **Priori** $P(C_j)$: Priori probabilities of individual categories can be evaluated using the sample population. **(1/10 mark)**
- **Posterior** $P(C_j|X)$: Given a testing sample X, you can calculate the posterior probability $P(C_j|X)$ of a certain category C_j using Bayes' theorem and independence assumption of random variables (features). The category with highest posterior probability is voted as the prediction for given testing sample X. **(3/10 marks)**
- **Performance Evaluation**: Calculate the prediction accuracy. **(1/10 mark)**
- **Challenging Task**: Generate some Iris samples based on the priori and likelihood that are estimated above. **(2/10 marks)**

Note that the prediction accuracy is **NOT** important and it will **NOT** be considered for evaluating your work. You should **avoid using 3rd-party Naïve Bayes Classifier directly**. You may use some standard distribution estimation or density calculation functions, i.e., scipy.stats.norm. Submit your *Python Jupyter* code containing above calculations and describe your steps concisely using either markdown or comment whenever necessary.