CS 145 Project 3: Car Quality Prediction using NBC & Decision Tree

(A) Goal:

In this problem, you will implement a Naive Bayesian classifier and a Decision Tree classifier on the real-world *Car Evaluation* dataset from UCI's machine learning data repository, http://archive.ics.uci.edu/ml/datasets/Car+Evaluation.

(B) Data:

Please use the link mentioned above to download the dataset. The detailed description of the data is also available at the aforementioned site. A brief summary of the dataset is provided below for reference:
Each record in the dataset has 7 columns (6 attributes + 1 class).
All the attributes are categorical.
There are no missing values in this dataset.

Column 1 : *Buying* – Indicates buying price.
Column 2 : *Maintenance* – Indicates price for maintenance.
Column 3 : *Doors* – Number of doors in the car.
Column 4 : *Persons* – Number of persons that can be accomodated.
Column 5 : *Luggage Boot* – Size of luggage boot.
Column 6 : *Safety* – Safety standard of the car.
Column 7 : *Quality* (class value) – Designated quality of the car based on the above 6 attributes.

(C) Project Description:

Now, carefully read and perform the following steps:
(1). Download the dataset and split it randomly into two parts –
a) *training* dataset consisting of roughly 80% of the records, and
b) *test* dataset consisting of around 20% of the records.
Note: The split should be non-deterministic i.e. every time you partition the data, you should get different training and test datasets (but the ratio of their sizes should always be roughly 4:1).
(2). Use the *training* dataset to build a *Naive Bayesian* classifier that will predict the *quality* of the car based on its attribute values. Determine the accuracy of this classifier on the *training* data itself.
Note: You will be dealing with very small numbers (likelihood probabilities <<< 1). Multiplying these small numbers may not be a good idea. Instead try to use log and transform these series of multiplications into additions.
(3). Now, determine the accuracy of the classifier, built in the previous step, on the *test* data.
Note: Some <feature value, class> combination may exist in the *test* data, which was absent in the *training* data. In such cases, use a small *default* probability (say 10^{-3}) to facilitate your computation. Alternatively, you can use **add-one** smoothing.
Reference: https://en.wikipedia.org/wiki/Additive_smoothing

(4). Build a *confusion matrix* from the results obtained in the last step.
What can you infer from the *confusion matrix*?
Reference: https://en.wikipedia.org/wiki/Confusion_matrix

(5a). Build a decision tree on the *training* data (used in step 2) using *Gini Index*. Would the decision tree be any different, had you used *Information Gain*?

(5b). Visualize/draw the decision tree in both the cases. Please indicate if either decision tree has any *pure* leaves (i.e. leaves which has only one kind of class).

(6). Determine the accuracy of the decision tree classifier (built on *Gini Index*) on the *training* and *test* dataset.

(7). Now, sometimes, we need to convert a categorical feature with k categories into k new binary features, where the i^{th} binary feature indicates whether the original feature belongs to the i^{th} category or not. This strategy is called "**one-hot encoding**". Perform "one-hot encoding" on all the features of both *training* and *test* data to generate *transformed training* and *test* datasets.

(8). Repeat steps (2) – (6) with these *transformed training* and *test* datasets.
Does the accuracy increase or decrease after "one-hot encoding"?
Note: Please ***skip*** drawing the decision trees, as stated in **(5b)**, as it has too many features now.

(D) Submission Details:

For every team, **only** one person needs to submit the following on the CCLE website by due date:

(i) Assignment report
(ii) Source code and the *training* and *test* datasets, all compressed in zip format

(E) Requirements:

Source code **-**

(i) You should attempt to write a modular code.
(ii) You can use any programming language. However, you **cannot** use any built-in function that directly builds decision trees or Naïve Bayesian classifiers.
E.g. if you are using Matlab or Octave, you **should not** use `NaiveBayes.fit` or `fitctree`.

Assignment report **-**

(i) First page of your report should consist of the full names of all your team members along with their respective UCLA IDs.
(ii) In your assignment report, discuss elaborately, what you did in each of the steps **(1) - (8)** and what was the outcome.
(iii) You must mention also which subroutine/module you are calling to perform each of the steps **(1) – (8)** so that we can execute them and verify.
(iv) If you faced or conceived of any additional challenges, please discuss them separately.
E.g. sometimes records can have missing values (both for features and classes).
Note: There is no restriction on minimum/maximum number of pages for the report, as long as, you adequately fulfill the above criteria.