Algorithms For Massive Dataset Exam Project
Tree Predictors for Binary Classification

Riccardo Raffini - 45999A

June 2024

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

Contents

1. Introduction

2. Dataset

2.1 Original dataset

The project described in this report is carried out considering the **Secondary Mushroom** dataset, whose files and description are available at this page. The considered version of the dataset is the one available at June 2024, however copies of the employed files are also included in the project repository.

The dataset is made up of two main *CSV* files, namely *primary* and *secondary*, but only the latter is analyzed; this file contains a total of 61069 examples that were *synthesized* (artificially generated) from the real ones inside the primary file, additional information about the process are available in the original article [1].

Each example represents an hypothetical mushroom and it is characterized by **20 features**, belonging to **continuous** and **categorical** domains, plus a **categorical label** indicating whether the mushroom is *edible* or *poisonous*, with an approximate distribution of the two categories close to 55% and 45% respectively.
However either by reading the description on the linked page or by directly loading the CSV file containing the data, it is clear that this dataset is incomplete, meaning that there are **missing feature** values among the examples. More precisely, a summary of the percentage of missing features is shown in table 2.1.

Feature name	Type	Missing values (%)
class	categorical	0.00
cap-diameter	continuous	0.00
cap-shape	categorical	0.00
cap-surface	categorical	0.23
cap-color	categorical	0.00
does-bruise-or-bleed	categorical	0.00
gill-attachment	categorical	0.16
gill-spacing	categorical	0.41
gill-color	categorical	0.00
stem-height	continuous	0.00
stem-width	continuous	0.00
stem-root	categorical	0.84
stem-surface	categorical	0.62
stem-color	categorical	0.00
veil-type	categorical	0.95
veil-color	categorical	0.88
has-ring	categorical	0.00
ring-type	categorical	0.04
spore-print-color	categorical	0.90
habitat	categorical	0.00
season	categorical	0.00

Table 2.1: Features, types and percentages of missing values with respect to the total number of examples.

For this reason it is necessary to carry out transformations and cleaning processes on the data before they can be used for the project purpose of classification using binary tree predictors.

2.2 Data pre-processing

Further analysis of the dataset shows that the whole list of examples is affected by at least one missing value, so the idea of simply discarding the affected examples is not applicable in this situation, as this would leave an empty dataset, thus more suitable processing techniques should be employed.

Missing data is a common challenge in machine learning applications, as it can affect the quality and the performance of models, some of the main approaches that can be used for handling missing values in a dataset are *deletion*, *imputation* and *learning*.
Although a learning strategy could be interesting, it may be too complex and inadequate for an unknown generated data domain.

So a strategy that mix the effects of both the **deletion** and **imputation** is chosen as alternative. Rather than removing affected examples, it might be better to remove features, but considering also the fact that some of the affected ones do not have very high missing percentages, deleting all them indiscriminately could be a too extreme decision. Thus features are deleted observing whether or not each of them has a *significant* quantity of missing values, where the importance can be formalized by a threshold value that implies the dropping of the feature if and only if its quantity is above the chosen value.
After this first process, it is possible that some examples still contain missing components, but if they have passed the previous phase, it means that the quantities related to such components are not so high to determine the deletion of the columns. Hence to complete the cleaning process, imputation is used to fill these remaining values with estimated or substituted ones. There are many ways of filling features, so the most simple and custom strategy is the possibility of providing a generic value computed following any approach.

In this way, the advantage is that no example is removed from the dataset, but deleting and adding data have some drawbacks that can influence the generalization and prediction capabilities of models, because they introduce distortion in the original data distribution.

This cleaning process translates into the following functions named `delete_dataset_features()` and `fill_dataset_samples()` defined inside the module `datasets.methods`:

[code implementation]

In the specific case of the considered dataset, the two function are applied with 0.40 as missing feature threshold and, since all remaining features are categorical (and nominal), a new attribute `'u'` (standing for *unknown*) is used as filling value.

To conclude the data pre-processing section, the examples originally loaded as a `pandas` *DataFrame* are turned into more easily manipulable `numpy` arrays and at the same time, the `class` feature (label) can be separated from the other example's features.
The following `extract_samples_labels()` allows to do so and eventually apply a mapping to the labels, for instance this could be used to turn categorical labels into numerical ones.

[code implementation]

Since the examples and labels will be used by predictors belonging to the decision tree class, there is no need of further processing the data, also because as written in the dataset description, all the remaining categorical feature are nominal (there is no ordinal feature), additional transformation would make no sense to the end of classification purpose, but this has to be taken in account for the predictor definition and implementation.

Bibliography

[1] Heider D. Wagner, Dennis and Georges Hattab. Secondary Mushroom. UCI Machine Learning Repository, 2023. DOI: https://doi.org/10.24432/C5FP5Q.