Mushrooms project

Alchieri Leonardo Badalotti Davide Bonardi Pietro Ravazzi Lucia

June 2020 - Università degli Studi di Milano Bicocca

ABSTRACT: In the present work, our main goal has been to introduce and show how machine learning algorithms can provide an efficient, fast and reliable way to determine whether a mushroom is edible or not.
In particular, we firstly compared the performance of various algorithms, according to multiple parameters, including accuracy, F-measure, Cohen's Kappa etc.
Then, we made use of a cost matrix to supply the classifiers analyzed with additional information about the data, highlighting the perks and disadvantages of this choice on the classification performance.
In addition, we applied a feature selection procedure to show that the use of a smaller set of features is feasible for this problem, without substantial drops in performance.
Lastly, we deployed the same models on another feature, `stalk-root`, a multi-label classification problem. We showed with this question how machine learning techniques can be applied to supplement missing information.
The results obtained in our research, other than being useful for field experts, highlights the power and capabilities of machine learning frameworks, together with the differences, perks and disadvantages of the various methods for classification and evaluation.

Summary

1. Introduction

Mushrooms are the fruiting bodies of highly sophisticated organisms called mycelia, whose presence all over the globe has made them a staple in many diets.[1] It has thus been fundamental for human populations, throughout history, to establish knowledge about this food source. While many mushrooms are edible and safe for human consumption, many more are not and distinguishing such trait is of vital importance. A few hundred species even are lethal to man, some just in small quantities, such as the death cap (*Amanita phalloides*).

The only way to truly recognize edibility is through common information, which has been built over thousands of years in many parts of Europe and China. With growing popularity all over the world, and especially in the United States, whose sorely cultivated mushrooms industry equates around 30 billion dollars in market sales, many efforts have been developed over the years to build a safe and comprehensive knowledge. Indeed, one such work was The Audubon Society Field Guide to North American Mushrooms, whose analysis in 1981 detailed more than 8000 mushrooms and their edibility.[2] Although a colossal work, as stated in the guide, there is no easy trick to determine such important trait.
In our work, we thus decided to help in such difficult task using the field of Machine Learning, specifically the use of classification techniques, developed through the years.[3] In particular, our focus is towards future applications in the field of mushrooms studies. For this reason, we juxtapose to the more traditional classification of edibility, two more practical approaches to this dataset: the first one is to show how feature selection might be applied to the classification of edibility, in order to reduce perspective burdens for researchers; the second one is to show how classification algorithms can be successfully deployed for those traits where missing values are abundant.
We will introduce in the next Section the dataset in more detail, and the characteristics it refers to. We then focus on the edibility classification, both general and weighting it to some cost; in this analysis, we introduce our *feature selection* that can be potentially interesting to researches in the field of mushrooms. And finally, we conclude our article describing classification algorithms

applied to a trait that presents a large amount of missing values as a *secondary task*, to show the possibility of using Machine Learning to supplement non-available information.

2. Dataset

This dataset was originally donated by Jeff Schlimmer to UCI Machine Learning repository on 27 April 1987. It includes description about samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Basically it describes mushrooms in terms of their physical characteristics: it has 8124 number of instances and 23 categorical attributes. The reason why it is of great interest is the fact that for each species it specifies poisonousness or edibility.[2]

2.1 Description

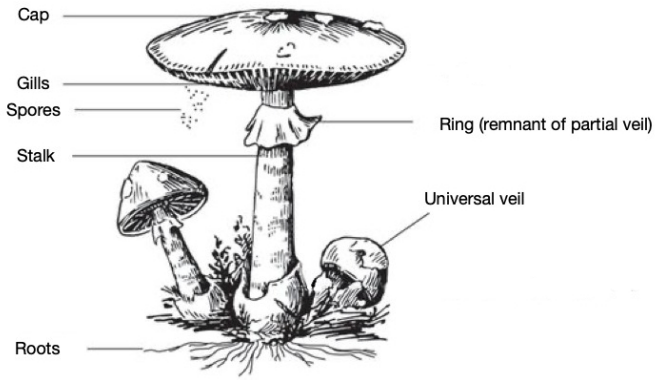

Figure 1: Simple drawing of a generic mushroom, with some highlighted parts. Inspired by an image found on the *Visual Dictionary Online*.

As stated before, mushrooms are the fruiting bodies of mycelia: we show some of the fundamental parts researchers have studies in Fig. 1. In order to understand what the Audubon scientists' work, it is fundamental to clarify some aspects of such organisms. From top to bottom:

Cap, differently shaped and colored upper part of the mushroom that protects the gills.

Associated attributes:

- cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
- cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s
- cap-color: brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y.

Gills, fertile spore-producing part of the mushroom, located under the cap.

Associated attributes:

- gill-attachment: attached=a, descending=d, free=f, notched=n
- gill-spacing: close=c, crowded=w, distant=d
- gill-size: broad=b, narrow=n
- gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y

Spores, microscopic seeds acting as reproductive agents; usually spread through the air.

Associated attributes:

- spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y

Ring, membrane located under the cap and circling the stem.

Associated attributes:

- ring-number: none=n, one=o, two=t
- ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z

Stalk, axis supporting the mushroom's cap.

Associated attributes:

- stalk-shape: enlarging=e, tapering=t
- stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
- stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s
- stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s
- stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
- stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y

Universal veil, a temporary structure of tissue used to isolate and protect the developing spore-producing surface.

Associated attributes:

- veil-type: partial=p, universal=u
- veil-color: brown=n, orange=o, white=w, yellow=y

Volva, remnant of a membrane that completely covered the immature mushroom and ruptured as the stem grew.

Roots, tangle of hyphae created through spore germination, from which the aboveground part of the mushroom develops.

Other attributes

- classes: edible=e, poisonous=p
- bruises: bruises=t,no=f
- odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
- population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
- habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

2.2 Exploration

Before diving into the heart of our work, i.e. the two main classifications, we decided to operate some exploratory analysis to gain insight with the data at hand. First of all, the data set is made of only categorical attributes. All classifications models can work with these type of data, except for one type of Neural Network described in the Section 3.

We analysed the distribution of the edibility class and the correlation between the data attributes. We then explored the presence of missing values and other non-trivial problems.

As a first observation, we show in Fig. 2 the distribution of the traits in the `class` attribute. As one can see, the two values are almost equally present. As we will show in Section 3, given this result, a specific sampling techinque is not needed.

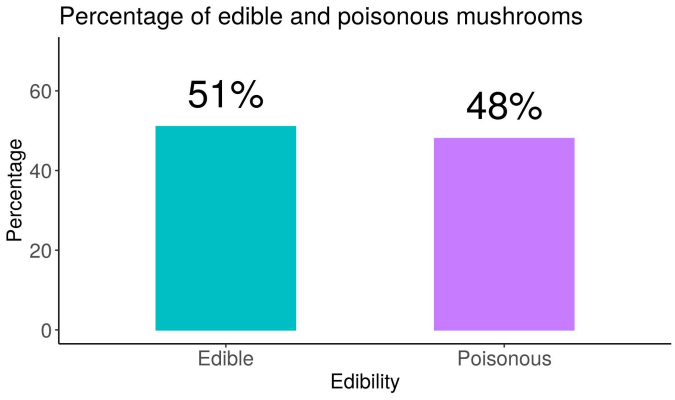

Figure 2: The two bars represent the distribution of the edibility and poisonousness of the mushrooms in the dataset.

Analysing the dataset also showed how one attribute, `veil-type`, is the same for each observation. We decided not to use it: it would not give any kind of insight to the problem; with its elimination, we also reduced the number of dimensions onto which train our models.

Our next focus was on the correlation between variables. Indeed, if some correlation would be present between the dataset attributes, we could theoretically avoid using some of them, for they would not give any additional information. In order to do this, we followed Mr. Zychlinski's work[4] in using a **Theil's U** correlation. Also called **uncertainty coefficient**, its role is to evaluate the asymmetric dependence of one variable x with respect to the variable y.[5] Mathematically, it is defined as:

$$U(y|x) = \frac{H(y) - H(y|x)}{H(y)}$$

where $H(y)$ is the entropy for the variable y and $H(y|x)$ is the conditional entropy. Due to its definition, $U(y|x) \neq U(x|y)$ is valid, thus making this coefficent interesting for studying non symmetrical correlations. Indeed, for our dataset, we considered the possibility of relations being asymmetrical, e.g. maybe a mushroom with a certain cap has always a specific root, but not the opposite. We then used this coefficient to extrapolate asymmetrical correlation between the dataset attributes, as we show in Fig. 3. As can be seen, there are some strong correlation

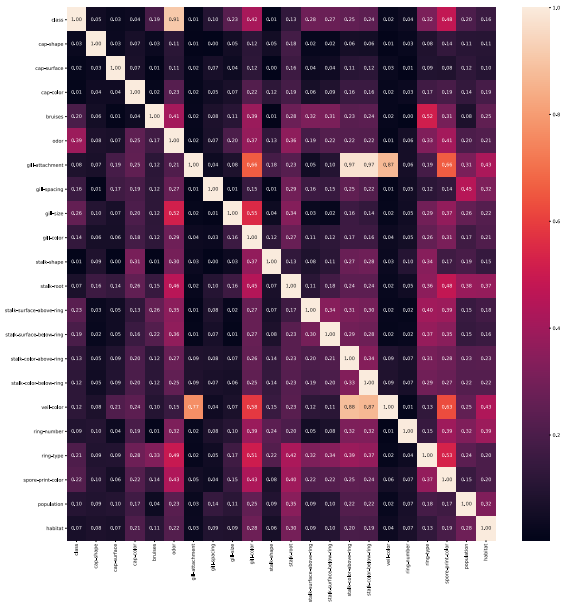

Figure 3: Theil's U calculated for all couplings of attributes.

between some attributes, e.g. `stalk-color-above-ring` with `gill-attachment`, but they do not result to be simmetrically valid, i.e. they are unilateral correlations. Given these facts, we decided not to exclude any column. We also noticed that there is a strong unilateral correlation between the `class` attribute and `odor`.

As stated previously, one of the attributes in the dataset, specifically **`stalk-root`**, gave some problems regarding its values: it has a quite significant (around 30% of the total) presence of missing values. The use of this col-

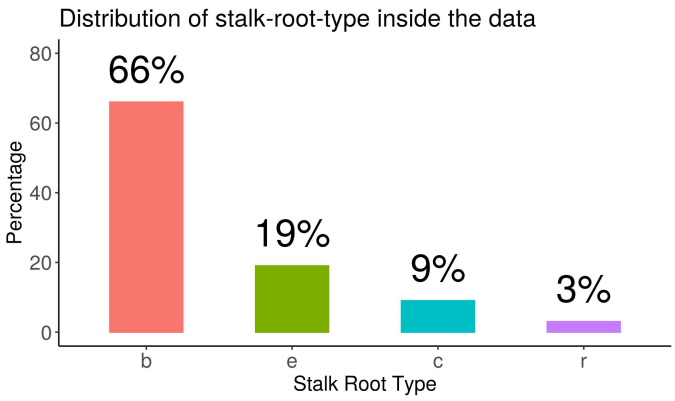

Figure 4: Percentage distribution of `stalk-root` values, excluding the missing ones.

umn in order to predict others, e.g. the `class` attribute, would pose some problems regarding the treatment of the missing information. Thus, we decided to exclude the `stalk-root` for `class` attribute classification. We do not believe the exclusion of this attribute, whose correlation with the other variables is not strong, to hinder our ability to predict.
Nevertheless, we wanted to show a meaningful way one could use to substitute those `stalk-root` values classified as missing. Indeed, as we will describe later in Section 5, this can be achieved using machine learning based methods. Preliminary, we explored the distribution of the recorded values for this variable, as we show in Figure 4. As one can see, they are not equally present, with some of them occupying less than 10% of the total. In such instances, sampling the whole set available might give results which are not fully satisfying; we will show later how the use of **equal sampling** to reduce imbalance could be interesting in this scenario.

3. Selected models

In order to develop both our tasks, we decided to deploy similar methods and practices. In particular, given the **classification** nature of our work, we deployed multiple classification algorithms, with the aim of finding the fittest for such data. All of the models were implemented using the **Knime** software, which allowed for fast and easy implementation of multiple classifications.

- **Heuristic models** such as *Decision Tree Learner*, *J48*, *J48Graft*, *IB1*, *Kstar* and *Random Forest*.
- **Regression models**, in particular *Logistic* and *Simple logistic* regressions.
- **Separation models**, specifically *Support Vector Machines* and *Neural Networks*. Regarding the first one, we implemented them with different kernel activations, such as *poly* or *puk*; we also implemented a variation of an SVM with Primal Estimated sub-GrAdient SOlver, called *Spegasos*.[6] As for *Neural Networks*, we worked with a *Multi-Layer Perceptron* (MLP) and a Neural Network with *Resilient Backpropagation* (Rprop).

  For this latter we decided to implement a *brute force search* in order to find the parameters which maximize the accuracy. Finally for performing the RProp, it was necessary to change the attribute type from categorical to numerical.
- **Probabilistic models**, i.e. models based on the Bayes theorem. The once used in this work are a simple *Naïve Bayes* algorithm and its tree implementation, called *NB Tree*.

The metrics used to confront the fitness of all of these models have been multiple. As for parametric based once, we thought the use of different paradigms would give us a more complete view of the models, particularly regarding confrontations. Indeed, as we will show later, the high value of performance of many models required us to have multiple metrics in order to distinguish the less performing ones. We describe briefly such evaluation metrics.

- **Accuracy**, defined as:

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

  where TP and TN represent, respectively, the*True Positives* and *True Negatives*, where FP and FN are the miss-classified ones, i.e. *False Positives* and *False Negatives*. We chose as the positive class the **edible value**. This parameter measures how many records were classified correctly, among all of the labelings.
- **Recall**, defined as:

$$rec = \frac{TP}{TP + FN}$$

  also called true positive rate.
- **Precision**:

$$prec = \frac{TP}{TP + FP}$$

  also called positive predictive value.
- F_1**-measure**, defined as the harmonic mean between *recall* and *precision*.

  When dealing with the multivariate classification, in Section 5, we decided to use averages of multiple F-measure as metrics. In particular, without going into detail, one can evaluate a “normal" bivariate F-measure between the i-th value of the class attribute and all of the other ones, and then average all of such values, with specific techniques. We decided to implement **macro F-measure**, **micro F-measure**;

and their weighted sum, **weighted F-measure**. The macro one is calculated averaging all of the different binary F measures, while the micro is evaluated using an average of precision and recall on the standard F measure definition.[7]

- **Specificity**:
$$spec = \frac{TN}{TN + FN}$$
also called *true negative rate*.

- **Cohen's Kappa**, a more robust measure than accuracy which accounts for random agreement.

With the metrics described above, we decided to utilize the three main techniques in Machine Learning to separate data for training and testing purposes: **holdout**, **iterated holdout** and **cross validation**. We used the first methodology to split the dataset each time before learning into a *training set* and a *testing set*, using as proportions, respectively, $\frac{2}{3}$ and $\frac{1}{3}$. In the second one, we split the data set as above, but repeated the process 10 times. The **cross validation** technique tries to overcome bias-related problems with the previous 2, dividing the dataset into k subsets and performing k times the learning algorithm, using each time $k-1$ sets together for training and the k-th one for testing. We applied 10-fold cross validation, in similarity with the previous method. Each one of these methodologies was deployed for every model we used, for both research questions.
In parallel to the analytical metrics just described, we utilized a very common graphical representation in Machine Learning: the **ROC curve**. This curve represents the **true positive rate** (y axis) to the **false positive rate** (x axis). Each point is drawn from different configurations of **thresholds**, i.e. the probability value which discriminates between positive and negative in the model classification. Its best configuration resembles a capital gamma Γ, while the worst one is the "random" 1st-3rd quadrant bisecant line. For obvious practicality reasons, we confronted ROC curves obtained from *holdout* only, even if it might be afflicted by some form of bias. Indeed, techniques for averaging ROC curves in *iterated holdout* and *cross validation* are non trivial and, to this day, controversial.
Another means to compare models, especially relative to their efficiency, is the **computational time**. For each model, we evaluated a training and testing time togheter on *holdout*, which we used for confrontation. In particular, we run this *benchmarks* multiple times, and averaged them. Due to the different nature of the various models we used, we decided the number of iterations for this calculation depending on necessity.
Given the nature of our first research question, we thought the use of a **cost matrix** might help distinguish between better performing models. Indeed, in our classification, where we distinguish the edibility and poisonousness of mushrooms, it's very important for a Machine Learning algorithm not to classify wrongly a poisonous one. If one might want to deploy such a technique in a real life scenario, a model which predicts safety where there is danger very few times is far worse than a model which does the opposite more times. The use of a **cost matrix** allows to take into consideration this factor: we applied a very high cost, of 100, to wrongly classified poisonous mushrooms, and a cost of 1 to the opposite. A standard cost of 0 was put for the correctly classified instances. These values are summarized in Table 1.

	P	N
P	0	1
N	100	0

Table 1: Each cell represent the cost linked to each type of prediction. P and N recall the positive and negative class. Columns consist of a inducer's predictions, while rows refer to the class value of records in the test set.

We chose the reported values as the fittest for our purposes, after empirical observations of different configurations. We will show in the next Section how this factor can be included or not into the learning algorithm itself.

4. Is this mushroom edible?

The aim of this section is to give an idea of which are the best algorithms to manage our task. To do this successfully we built a results structure in similar way for both non-cost and cost sensitive. In the former case, classification models are built without any type of influence from the cost matrix, while in the latter case, this aspect occurs. It's important to note that, after performing the non-cost sensitive task, we computed the total cost in this case as well, in order to select the classification models which minimize that quantity. Therefore, the *cost sensitive classification* is the only one where the cost matrix was used for the training itself, while on both methodologies it was applied a cost score after the test session.
This section touches four different aspects of our classifiers: performance measures, ROC curves, execution time and cost. We compute the first and the last quantities for holdout, iterated holdout and cross validation procedures in order to understand better these mechanisms. The obtained results are similar but, since the cross validation outcomes are affected by a lower bias, we present and analyze in this work only this latter methodology.
ROC curves are obtained on the holdout procedure. As mentioned in Section 3, it is not a trivial task to determine ROC curve in iterated procedures.
The execution time is computed with the holdout procedure iterated several times in order to collect different results for computing the mean and the standard error, as mentioned in the previous Section. Moreover, we as-

sume that the mean of the distribution of execution times for each algorithm is normal in order to use the quantile of the normal distribution into the confidence interval. In this case, we assume a confidence equal to 68% with a quantile equal to one. We don't decide to use a level of confidence equal to 95 % because algorithms with a few iterations have a large intervals and consequently, it may be possible to have some results that aren't statistically significant. So, we preferred to decrease the confidence in order to have the possibility of distinguishing some results.
Finally the cost is computed as the product between the cost matrix and the confusion matrix. Since we show here only the cross validation procedure, each iteration has its cost value.

4.1 Non-cost sensitive classification

For the non-cost sensitive classification the first comparison between algorithms performances are summarized in the Table 2.

Algorithm	-	Accuracy	+	Cohen's Kappa	Recall	Precision	Specifity	F-measure
Decision_Tree	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
IB1	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
J48	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
J48Graft	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
KStar	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
Logistic	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
MLP	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
Naïve Bayes	0,015	0,963	0,011	0,92	0,99	0,94	0,93	0,97
NB Tree	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
RProp	0,013	0,976	0,009	0,95	0,97	0,99	0,98	0,98
Random Forest	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
SMO(Poly)	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
SMO(Puk)	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
SPegasos	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00
Simple Logistic	0,005	1,00	0	1,00	1,00	1,00	1,00	1,00

Heuristic Regression Probabilistic Separation

Table 2: Cross Validation algorithms performances for non cost sensitive classification.

To make the table more readable we grouped algorithms by classification techniques. We start discussing the properties of the accuracy measure. The confidence interval of the empirical accuracy relies on the *Wilson score interval* which assumes that the accuracy is normally distributed. We use a confidence level equal to 95%. We note that for many algorithms the upper limit of the confidence interval touches the unit. The only two inducers which don't show this property are the Naïve Bayes and RProp. It's noteworthy to point out that the RProp parameters (number of hidden layers and the number of neurons for each one) are chosen to maximize the accuracy of the inducer. Although a better result can be reached considering other aspects, such as other loss functions, this approach is reasonable for our goals.
We note that if the upper limit of confidence interval is equal to the unit, all of the others metrics have values equal to one. We don't compute the confidence interval for these latters. It's important to underline that **the positive value of the class attribute is the edible one**. It's noteworthy to observe that those last classifiers are able to *detect* all mushrooms types. Indeed, the *recall* and *precision* are equal to one. It means that there aren't any false negatives, namely edible mushrooms which are classified as poisonous, and false positive, hence poisonous mushroom which are classified as edible. These algorithms don't know that it is worst to classify a poisonous mushroom as an edible one rather than the vice versa. Indeed, during a non cost sensitive training, all models weight False Negatives and False Positives in the same manner. However, even if this external knowledge isn't included, those algorithms are able to detect successfully all mushrooms. On the other hand, RProp and Naïve Bayes don't perform in this manner. Especially, the *precision* of Naive Bayes is lower than the *recall*: it classifies more FP than FN. The RProp shows the opposite behaviour. In conclusion, the Naïve Bayes should be avoided, while the others ones can be used.
The next outcome reported is the **ROC curve** for the holdout procedure, shown in Fig. 5. As one might expect from all metrics scores in Table 2, every algorithm except for Naïve Bayes achieves a perfect curve, i.e. a Γ-like shape. This means that, varying the threshold with which the class is predicted within the range $(0, 1)$, all of these algorithms never miss-classify a record. Indeed, from empirical observations on the predicted values, we saw that, in these cases, the inducers assigned the probability of being poisonous in $\{0, 1\}$, i.e. no values in between, and consequently, when the threshold is within the range $(0, 1)$, it is capable of recognizing always all mushrooms in a correct way. The only discernable difference between a quasi-perfect ROC curve is present for the *Naïve Bayes* one. Indeed, as mentioned previously, this model fares the worst under non-cost sensitive circumstances: thus, it is not shocking its curve has a lower rate of TP for some thresholds. Some of these results may also due to the intrinsic nature of the Bayes Theorem, whose underlying probabilistic structure can give rise to lower classification strength, e.g. a record may be classified as edible with probability 60%, and not a perfect 100% like most other algorithms presented.

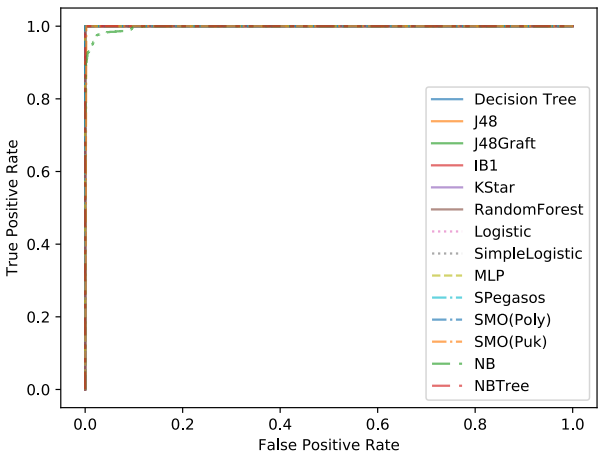

Figure 5: Non cost sensitive, ROC curve for holdout separation techniques. Due to implementation issues, we were not able to depict the ROC for the `RProp` classifier.

We know that classification models differ for **computation time**, so for this reason we show in Fig. 6 a benchmark. Especially, we have measured the time needed to train and test the classification model. However, the following results don't split these two times and they are present as unique value. For each classification model, the execution time is measured several times, as we have prevoiusly said. So, in order to summarize the variations occur during this process, execution times are presented with an error bar which are built from the mean and the associated standard error. Each bar of the Fig. 6 is therefore a confidence interval with a 68% of confidence level. It's important to underline that also the computer's performances impact on the computation of these values.

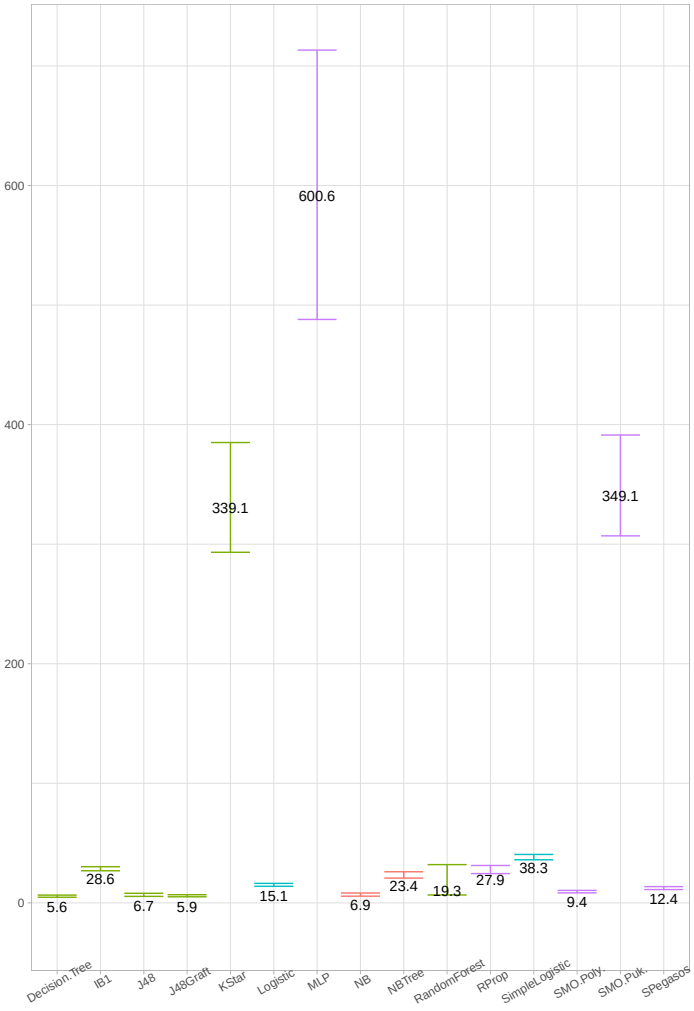

Figure 6: Algorithms execution times expressed in second for non cost sensitive classification.

Once again algorithms are grouped by classification techniques. The first thing to notice is that SMO(Puk), KStar and MLP are the slowest. In particular, KStar is very slow in testing phase while the others in the training phase. These statements rely on a qualitative analysis accomplished during the measurements of the speed. In order to explain the slowness of KStar, it's noteworthy to underline that it's an *istance based algorithm*, namely a classification problem in which each new instance is

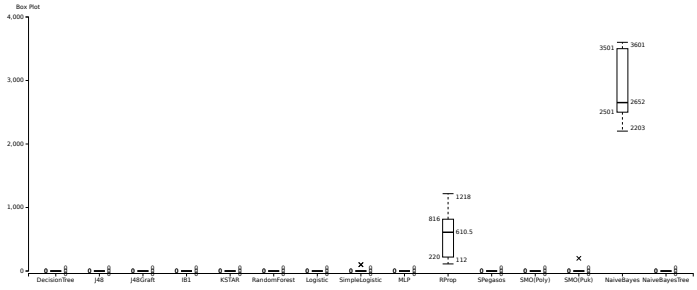

Figure 7: Costs Boxplots for non cost sensitive classification

compared with existing ones using a distance metric, and the closest existing instance is used to assign the class to the new one. So, when KStar must classify a new record of the test set, it computes a distance between the unseen record and those of the training set. It's clear that this procedure has a complexity of $\mathcal{O}(n)$, where n is the number of record into the training set. So, the total complexity of the testing phase is proportional to the cartesian product of the number of records of the test and training set. This reasoning explains reasonably the slowness of KStar algorithm.

The slowness of RProp and SMO (Puk) may rely on the procedures applied to build these classification models. Indeed, during the training phase, the learning algorithm of the neural networks has to select weights that minimize/maximize a specific loss function. This procedure requires always a lot of computational power. An analogous reasoning can be made for the support vector machines. Indeed, they seek the best hyperplane in k dimensional space, where k is the number of attributes, which maximize the margin and accordingly, this mechanism requires a lot of effort in the training part.

The speed of the remain algorithms are under the 40 seconds. Especially, the Decision Tree Learner algorithm is the fastest one. However, in general, the decision trees are the fastest between classifiers except for KStar.

The last result, namely the **cost**, is shown in Fig. 7. These values are obtained by the cross validation procedure. So, there is a cost value for each iteration for each algorithm. Rather than having only an aggregate value of the cost for each classifier, such as the mean, we opted to plot a boxplot. Indeed, it helps to visualize that for same algorithms, the classifier classifies correctly (zero cost) for several iterations but sometimes the cost is different from zero. This asymmetric behaviour is well shown in the boxplot. Indeed, the iterations in which the cost aren't zero are presented as outliers.

It's clear that many classification models have all value of the cost equal to zero. It means that they predict correctly each record of the test set for each iteration. SimpleLogistic and SOM (Puk) algorithms classify incorrectly only one and two times respectively. Since the cost of classifying a poisonous mushroom as an edible

one is equal to 100 and the outliers of the boxplots of the classifiers are multiple of that value, we state that these algorithms tend to classify in the most grave way. Indeed, they suggest us to eat a mushroom that is poisonous. This reasoning is supported by the values of the confusion matrixes, here not reported. Moreover, there are others algorithms which performs worst, namely the RProp and the Naive Bayes. We have already noted that they are the ones with the worst values of performance measures. Therefore, it's better to not use them.
After analyzing these aspects of our classification models, we try to suggest one of them if someone want to classify a mushroom. Maybe, Decision Tree Learner can be a good choice since it achieves almost perfect performance measures, it's the fastest one, all value of the cost are equal to zero and finally, its ROC curve is a Γ.

4.2 Cost sensitive classification

In this section we want to perform several classification models taking into account the cost matrix. Especially, in the training phase, training instances are re-weighted according to the total cost assigned to each class. Cost matrix is described in Table 1. Our inducers are designed to avoid the classification of a poisonous mushroom as an edible one. When it classifies incorrectly, it seeks to shift the error from the false positive to the false negative because the total cost will be lower. In particular, we have chosen a cost matrix in which the cost of a false positive is equal to 100. The idea behind our choice is that eating a poisonous mushroom can have way more dire consequences as opposed to discarding an edible one.
For showing and analyzing the result, we use the same schema of the previous section. First of all, the performance measures are described into Table 3.

Algorithm	-	Accuracy	+	Cohen's Kappa	Recall	Precision	Specifity	F-measure
IB1	0,014	0,971	0,010	0,94	0,94	1,00	1,00	0,97
J48	0,014	0,971	0,009	0,94	0,94	1,00	1,00	0,97
J48Graft	0,012	0,982	0,007	0,96	0,97	1,00	1,00	0,98
KStar	0,014	0,969	0,010	0,94	0,94	1,00	1,00	0,97
Logistic	0,018	0,942	0,014	0,88	0,89	1,00	1,00	0,94
MLP	0,005	1,000	0,000	1,00	1,00	1,00	1,00	1,00
NB Tree	0,005	1,000	0,000	1,00	1,00	1,00	1,00	1,00
Naïve Bayes	0,005	0,999	0,001	1,00	1,00	1,00	1,00	1,00
Random Forest	0,009	0,991	0,005	0,98	0,98	1,00	1,00	0,99
SMO(poly)	0,005	1,000	0,000	1,00	1,00	1,00	1,00	1,00
SimpleLogistic	0,005	1,000	0,000	1,00	1,00	1,00	1,00	1,00

Heuristic Regression Probabilistic Separation

Table 3: Cross Validation algorithms performance for cost sensitive classification

Here the differences from the non-cost sensitive are that in this case it was not possible to compute SPegasos, SMO(puk), RProp and Decision Tree Learner due to implementation issues.
The confidence intervals for accuracy are computed as the non-cost sensitive case. We note that classifiers which achieve with their upper limits the unit in the non-cost sensitive classification, such as IB1, J48, J48Graft, KStar, RandomForest and Logistic, don't reach the top in this situation. Only MLP, NBTree, SMO (poly) and SimpleLogistic show the same performances. Naive Bayes improve decisively. This shows that the influence of the cost during the training phase can make *the performances worst in order to avoid the classification of false positive*. Indeed, inducers are build with the aim to minimize the number of false positive that, in other words, it's equal to say that the *precision* must be maximized. So, it's perfectly reasonable that the accuracy, for some algorithm, don't reach the top result because we aren't maximizing that loss function with our cost matrix.
It is therefore remarkable that **the precision measures for all algorithms are equal to one**. This is a positive aspect: our classifiers never predict a poisonous mushroom as an edible one, i.e. the number of false positive is equal to zero. Moreover, the recall for many algorithms are lower than the respective non-cost sensitive case. Indeed, our cost sensitive models prefer to classify an edible mushroom as a poisonous one rather than the vice versa, due to the lower value of the associated cost. We can state that the best now are the probabilistic ans separation methods on the basis of the achieved values of performance measures.
We show in Fig. 8 all **ROC curves** for this cost sensitive classification. Most results are au pair with the previous Section, with the best performing algorithms under the Table 3 metrics achieving a perfect Γ-like shape. But there are notable differences from what presented in the previous non-cost sensitive analysis. The most discernible one is the IB1 model, which has a lower curve than all other algorithms. J48Graft and J48, which do have lower metrics scores, struggle to classify some TP at some threshold, probably due to lower strength in the classification. From all curves in the representation, no systematic miss-classification of FP is present, indicating that, if some poisonous mushrooms are indeed classified as edible, they are but a handful. This argument will be strengthened in the next cost depiction.

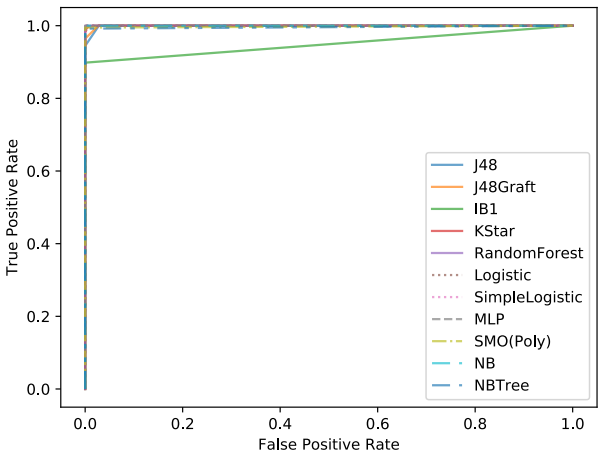

Figure 8: Cost sensitive classification, ROC curve for holdout.

The next result to test algorithms performance concerns the **execution time**. As in the non-cost sensitive methodology, it merges the value of training and test duration. The construction of the error bars is the same as in Section 3. In Fig. 9 is shown the result.

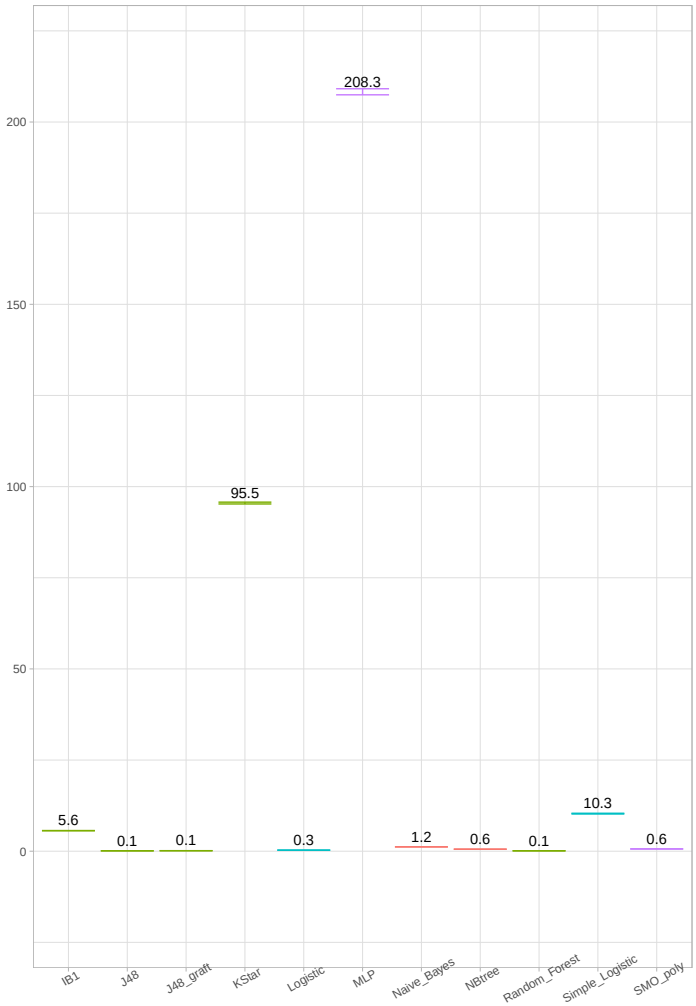

Figure 9: Algorithms execution time expressed in second for cost sensitive classification.

In general, the speed values are lower than the ones in the non-cost sensitive case. Also here the KStar is slower in testing phase while MLP in training phase. The explanation of those facts is the same of the non-cost sensitive section. The last algorithms have speed under the 11 seconds. The fastest ones are decision trees, especially the Random Forest, J48 and J48Graft.

The last results depicts in Fig. 10, shown the **cost**. It's computed as the previous section.

The first thing to note is that the order of magnitude on the y-axis is decreased twofold. It's clear that the effect of introducing the cost-sensitive algorithm influences heavily the final cost. Algorithms such as MLP, SMO (Poly) and NB Tree have cost, as in the above solution, equal to zero for all steps in the cross validation procedure. The SimpleLogistic presents now the previous property. It is noteworthy that some algorithms,

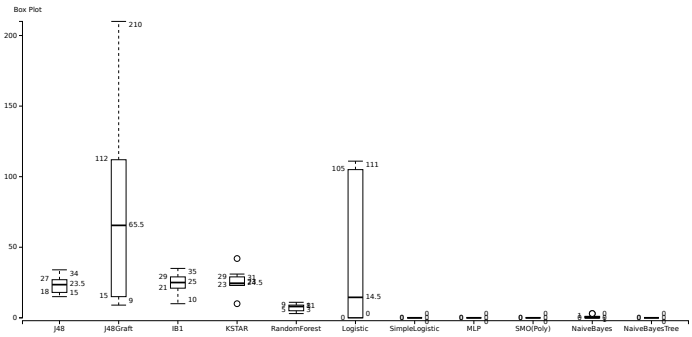

Figure 10: Cost Boxplots for the cost sensitive classification.

such as J48, J48Graft, IB1, KStar, RandomForest and Logistic have a median cost higher than the non-cost sensitive case. It can be puzzling this last aspect because maybe, it's natural to expect the opposite situation. A possible explanation relies on the choice of the cost of False Positive. Indeed, it's possible that the structure of some of these algorithms is built in order to minimize the incorrect classification of edible mushrooms, which maybe could entail more incorrect classification towards false negative. The median cost of those algorithms are lower than one hundred. This means that since these algorithms, except for J48Graft, sometimes make incorrect predictions, they prefer to predict false negative rather than false positive. Naïve Bayes has improved a lot.

In conclusion, if someone wants to classify mushrooms taking into account the cost, we suggest to use the Support Vector Machines with Poly option due to its good performances.

4.3 Reducing Features

Now that we established the best methods to classify the edibility of mushrooms, the objective is to see whether it is possible to reduce the number of features used for the classification, without visibly affecting the performance of the models (the scope of the term *visibly* will be specified later on). This operation could help us define a small set of features to look for in a mushroom in order to determine its edibility with sufficient confidence. In other words, rather than analyzing the mushroom as a whole, one could look for single specific features and have the same response, with obvious advantages.

In order to carry out a feature selection task, we have lots of methods and approaches. Since our final aim is to determine a set of features to handle the classification without substantially affecting the performance, we decided to use a wrapper, i.e. we rely on the classifier to find the optimal subset of available attributes. This choice has a main crucial advantage: it is, by construction, the method that better allows us to control the models performance. Filters, on the other hand, despite being less computationally intensive and classifier

independent, have a higher risk of leading us toward a sub-optimal set of features (that, for example, penalizes too much the performance or has a higher dimension).
By making these choices, we obtain results that are classifier dependent and thus less general, but more geared toward a higher performance. The overall approach is summarized below:

- We chose 2 different classifiers that had a fair performance and total execution and training times sufficiently low on the whole set of attributes: **J48 Tree Learner** and **Logistic Regression**.
- We applied a wrapper to these classifiers, using a **Greedy Stepwise, backward search**. That is, starting from the whole set of attributes, the classifier removes the least important attribute from the set. In this case, the removed attribute is the one that less affects the accuracy score.

  We decided to use this approach since it is more aligned with the overall logic of the task, that is to answer the question: *Which of the attributes is less important for the classification?*, or, in other words, *Inside the whole set of attributes, what are the ones that I can skip, without sensibly affecting the accuracy?* Hence the backward stepwise search.
- Lastly, once the attributes have been selected, the model is trained and after testing it, the following metrics are evaluated: accuracy, recall, precision, F-measure, specificity, sensitivity, Cohen's Kappa.

The overall logic is based on the train-validate-test procedure. In other words, the dataset is first split in training and test set.
The training set is used to determine the reduced set of attributes and to train the model, than, the test set is used for the performance evaluation, using a simple holdout approach.
Our first intention here was to apply a cross validation procedure, but due to the limitations of the software used, we were unable to. Indeed, using a k-fold cross-validation makes it impossible to retain the extracted attributes for any fold, except the last one, so we opted for a simple Holdout approach as described in Section 3.
We are aware of the fact that using a simple holdout approach has some drawbacks, since part of the instances are never used for the training process. Also, a single holdout procedure prevent us from calculating confidence intervals of the performance scores.
On the other hand, it seemed the only viable option to analyze effectively the subset of features remained after the selection.
The operation of extracting the optimal subset of features, on the other hand, is done via 5-folds cross validation.
In other words, the training set is split into 5 disjoint sets. Alternatively, 4 of these are used to form the training set, the remaining one is called *validation set.*
We can see this feature selection process with the same logic of an *hyperparameter tuning process*. That is, the training set is used to determine the optimal value of the parameters (or in this case, attributes), than, the validation set is used to provide an unbiased evaluation of the model fit during the selection process.
Once the attributes are selected, the validation and training set are joined, and the model is trained one last time, and then validated on the test set.
Using a 5-fold cross validation during the *tuning phase* of the model (the selection of the attributes) improves the stability of the evaluations, thus providing a more solid result.
Now that the procedure is clear, we show in Table 4 the subset of attributes remained after the selection for each classifier:

J48	**Logistic**
odor	odor
cap-color	cap-color
gill-size	gill-size
gill-spacing	gill-spacing
spore-print-color	spore-print-color

Table 4: Selected attributes for each classifier, after feature selection.

As we can see, both classifiers keep the same 5 attributes.

Score	**J48**	**Logistic**
Accuracy	1	1
Recall	1	1
Precision	1	1
F-Measure	1	1
Sensitivity	1	1
Specificity	1	1
Cohen's Kappa	1	1

Table 5: Performance scores for each classifier, after feature selection.

Regarding the performance of each model, we summarized the results inside Table 5. As we can see, all the values are equal to 1. We conclude that these classifiers are, from a performance point of view, virtually equivalent. Regarding the execution times, the Logistic Regression is sensibly slower to train and validate, with time differences in the order of minutes (not reported). We are not sure whether this time difference is entirely due to the classifier or to the instances inside the training set. On the other hand, if we compare the performance of these classifiers with their counterparts before the feature selection, reported in Section 4.1, we see fundamentally no change in values.

First of all, regarding the accuracy, we can see that all the values are compatible, before and after the feature selection process. For the other measures, despite the absence of a confidence interval, we can see that the differences are in the order of 0.01, or even 10 times less.
We can conclude that, for this specific classification process, we can rely on a set of only 5 features, with the classifiers *J48* and *Logistic*, which are able to complete the classification process without any visible decrease in performance whatsoever. The just described set of features is classifier dependent and the whole process of training and selection is slightly more demanding on the computational and time side, but it affect only the training part of the process.

5. Which is the mushroom stalk rook?

As a second research question, we have decided to focus on classifying one specific attribute in the dataset: the **stalk root**. As mentioned briefly previously, our choice is not random: indeed, this characteristic is the only one which presents *missing values*. Thus, we decided to focus on developing good classification models for the stalk root, using all of the other available attributes, as described in Section 3, to show an alternative possibility at substituting missing values.
We would like to point out how we never actually used the next models to substitute *missing values* in the original dataset. With our work, we want to show a robust way to predict this attribute, using all of the *non-missing* data for both training and testing. Indeed, while, after the classifications and the identification of the best performing model, we could predict a value for those mushrooms who did not have one. However, we don't implement this task in our work.
We still hope the following technique might be of interest for further analysis and could give insight to a better understanding of mushrooms.

5.1 Without equal sampling

In order to classify the **stalk root** class, we obviously had to discard those records which presented a *missing value*. Indeed, our objective was to identify a classification methodology to substitute them as correctly as possible.
As stated in Section 2.1, without considering the missing values, which by themselves account for about 30% of the records, the remaining distinct values a stalk root can take are four, b, e, c and r. This by itself presented a non trivial problem for the classification: we could not deploy traditional binary classifications. Nevertheless, due to the flexibility of all of the algorithms we have used throughout our project, we managed to use them all without any hassle. Also, as already mentioned in Section 3, we had to apply a slightly different metrics system, using only the **accuracy** and three types of averaged **F measures**.
The main difficulty we have faced was connected to the distribution of the values. As we have previously shown, in Fig. 4, this class presents a quite strong imbalance. For this reason, we have decided to take two approaches, and confront them, regarding the sampling method for selecting a *training* and *testing* set: **without equal sampling** and **with equal sampling**. In the first one, the set from which to select training and the testing ones is the complete available; in the latter, the pool from which to select has equally distributed values for the attribute, i.e. each value is present the number of times in this *equally sampled set* as the least present one in the total dataset.
In this Section we will describe the results regarding the first of the two sampling methodologies, leaving to the next Section a more thorough discussion about *equal sampling*.
The models used were all of those described in Section 3, except for the *RProp* Neural Network and Support Vector Machine *SPegasos*, due to implementation issues faced. We operated all three main paradigms, as described, i.e. *Holdout*, *Iterated Holdout* and *Cross Validation*. In all three scenarios, we found similar results, with small discrepancies related mostly to small biases in the first two methodologies. We thus report results for *Cross Validation only*, being the least affected by selection-related bias.
In Table 6 we show our results: all of the models achieve high metrics scores.

Algorithm	-	Accuracy	+	F_1 micro	F_1 macro	F_1 weighted
Decision Tree Learner	0,007	1,00	0	1,00	1,00	1,00
J48	0,007	1,00	0	1,00	1,00	1,00
J48 Graft	0,007	1,00	0	1,00	1,00	1,00
IB1	0,007	1,00	0	1,00	1,00	1,00
KStar	0,007	1,00	0	1,00	1,00	1,00
Random Forest	0,007	1,00	0	1,00	1,00	1,00
Logistic Regression	0,007	1,00	0	1,00	1,00	1,00
Simple Logistic Regression	0,007	1,00	0	1,00	1,00	1,00
Multi Layer Perceptron	0,007	1,00	0	1,00	1,00	1,00
SMO Poly	0,007	1,00	0	1,00	1,00	1,00
SMO Puk	0,008	0,999	0,001	1,00	1,00	1,00
Naïve Bayes (standard)	0,013	0,987	0,007	0,99	0,99	0,99
Naïve Bayes (Weka)	0,008	0,998	0,002	1,00	1,00	1,00
NB Tree	0,007	1,000	0	1,00	1,00	1,00

Heuristic Regression Probabilistic Separation

Table 6: Results for the cross validation in random sampling.

In particular, it can be noticed that all F_1 values are au pair with the accuracy, meaning that no specific bias happens during all classification processes. Some models do have almost perfect score, with deviations related to the confidence interval. In particular, Heuristic Models, Regression models, Naive Bayes Tree, the Neural Network and one of the SVM, are the best. With respect to the first of these, indeed one might expect Decision Trees and related models to handle this kind of nominal data the best. On the other hand, **Naïve Bayes** methods are those struggling the most. Nevertheless, all algorithms

are comparable.
While the best performing models manage to achieve a perfect accuracy even when selecting without regard of the class distribution, some models do struggle and have lower metrics values. We will compare, in the next Section, the results just shown with those taken using **equal sampling**, to see if some bias might be present with regard to this methodology.

5.2 Equal sampling

Due to the nature of the **stalk root** class, i.e. imbalance between values, we tried a specific approach regarding the selection of records for training and testing. In contrast with what was done in the previous Section, we decided to reduce the pool of records in order to have an equal number of values for the class attribute: **equal sampling**.
With this technique, confronting with the previous case, we want to see if discernible differences are present. Indeed, the use of *equal sampling* should allow us to overcome the bias related to having an unevenly distributed class attribute.
On the other hand, this method does pose some drawbacks, especially regarding the size of the dataset: in order to have an equal number of values, the lowest proportion must be selected. In our case, we went from having 5644 total records (and $^2/_3$ of this for training) to just 774. Having around 13% of the original dataset at hand certainly can have meaningful impact on model training and evaluation.
Given both theoretical benefits and downsides of the sampling methodology hitherto discussed, we will now show the results achieved.
As in the previous Section, we show the results for the *Cross Validation* only, which should eliminate any bias related to the selection of the training and testing sets. Indeed, this bias might be even more relevant in the presence of a smaller dataset than before. In Table 7 we depict the achieved metrics. As can be seen, lower values than *without equal sampling* are present, with larger confidence intervals for the accuracy. This is mostly due to the reduced size of the dataset: on the one hand, less training data means lower metrics results; on the other, a lower testing set means larger confidence intervals. Indeed, in this scenario all models are comparable.
As one can notice, some algorithms are less well performing when compared to what happened in Section 5.1. Indeed, some models are more definitely more influenced by a lower availability of data.
We conclude the discussion about the second research question with a few considerations regarding all of the results obtained and discuss the best models. We can successfully say that, using Machine Learning algorithms, it is possible to predict, with high accuracy, the **stalk root** attribute in mushrooms. Due to the high imbalance of the values of the class, we tested both without and with equal

Algorithm	-	Accuracy	+	F_1 micro	F_1 macro	F_1 weighted
Decision Tree Learner	0,053	0,997	0,003	1,00	1,00	1,00
J48	0,054	0,996	0,004	1,00	1,00	1,00
J48 Graft	0,055	0,994	0,005	0,99	0,99	0,99
IB1	0,052	0,999	0,001	1,00	1,00	1,00
KStar	0,052	0,999	0,001	1,00	1,00	1,00
Random Forest	0,052	0,999	0,001	1,00	1,00	1,00
Logistic Regression	0,055	0,994	0,005	0,99	0,99	0,99
Simple Logistic Regression	0,053	0,997	0,003	1,00	1,00	1,00
Multi Layer Perceptron	0,052	0,999	0,001	1,00	1,00	1,00
SMO Poly	0,052	0,999	0,001	1,00	1,00	1,00
SMO Puk	0,053	0,997	0,003	1,00	1,00	1,00
Naïve Bayes (standard)	0,059	0,989	0,009	0,99	0,99	0,99
Naïve Bayes (Weka)	0,058	0,990	0,008	0,99	0,99	0,99
NB Tree	0,054	0,996	0,004	1,00	1,00	1,00

Heuristic Regression Probabilistic Separation

Table 7: Results for accuracy and different types of F_1 *measures* for the *Cross Validation* procedure.

sampling, resulting in comparable results, even though the latter are way less accurate. With the first, some models were less performing, with **Decision Trees** and **Logistic Regression** being the best ones overall. Similar results can be obtained from the latter discussion, but with less strength in the argument, due to larger confidence intervals and lower training sets. Nevertheless, we consider the usage of *equal sampling* for this classification viable, if presented with enough records. Indeed, while in our case the large initial sample allowed for its applicability, future studies might not have this luxury. Even in the face of impossibility of using it, we showed that an approach without it can be effective, with some models achieving 100% accuracy even when tested in **Cross Validation**.
We hope our answer to this question might be of use to future studies, especially when confronted with difficulties in finding such attribute. Indeed, using Machine Learning models might be a cost effective and fast way to substitute missing values, because all models presented, and the best performing ones in particular, reach more than satisfying metrics for this task.

6. Conclusions

We conclude our research with a few final remarks. Regarding the edibility of mushrooms, we can safely say that many models presented achieve high metrics scores, for both non cost and cost sensitive techniques. Regarding the former, it is worthwhile to mention that the Naïve Bayes is the only algorithm which presents a high degree of *False Positives*, i.e. poisonous mushrooms classified as edibles. In the latter analysis, due to the introduction of the cost matrix in the learning process, all models achieve a perfect 1 regarding the **precision**, i.e. no poisonous mushroom is classified as not poisonous. Both methodologies suggest to use **Naïve Bayes Tree** due to its good performances.
We also showed that the implementation of **feature selection** can be achieved without visibly reducing any performance metrics. We hope this results could be of interest use for domain experts, since the use of only 5 features instead of 21 can be handy in many situa-

tions. We nevertheless recognize the limitations of using only 2 classification models for this procedure with an algorithm dependent selector.
Eventually, we showed that the use of machine learning algorithms is a viable option to predict the `stalk-root` class. In particular, we achieved the objective of showing a way to predict missing values in this class. We would like to point out how we did not actually perform the substitution operation, but just wanted to show a methodology.
As a final remark, we can safely assert that we reached all initial objectives. We hope this work might be of interest for future analysis in this field.

6. References

[1] Ian Robert Hall, Peter K Buchanan, Anthony LJ Cole, Wang Yun, and Steve Stephenson. *Edible and poisonous mushrooms of the world*, volume 103. Timber Press Portland, 2003.

[2] Gary H Lincoff. The audubon society field guide to north american mushrooms. Technical report, 1981.

[3] Zijian Zheng. *A benchmark for classifier learning*. Citeseer, 1993.

[4] Shaked Zychlinski. medium.com, Feb 2018. URL https://towardsdatascience.com/the-search-for-categorical-correlation-a1cf7f1888c9.

[5] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, 2007. ISBN 0521880688. URL http://www.amazon.com/Numerical-Recipes-3rd-Scientific-Computing/dp/0521880688/ref=sr_1_1?ie=UTF8&s=books&qid=1280322496&sr=8-1.

[6] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated subgradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[7] Vincent Van Asch. Macro-and micro-averaged evaluation measures [[basic draft]]. *Belgium: CLiPS*, 49, 2013.