DATA AND INFORMATION QUALITY PROJECT REPORT

PROJECT ID: 4

PROJECT NUMBER 1

ASSIGNED DATASETS: car.csv

STUDENT: MATTEO CACCAVALE 10693694

ASSIGNED TASK: Classification

1. SETUP CHOICES

a. Chosen Machine Learning algorithms

The Machine Learning algorithms I choose for the proposed classification task are the Decision Tree learning and the Logistic Regression. The former was chosen because it is a widely used algorithm and it performs better when the dataset is mainly composed of categorical attributes rather than numerical ones. This is the case of the dataset provided for the project ("car.csv") that contains only categorical attributes, so no discretization of numerical variables was needed. The other algorithm chosen, Logistic Regression, is a very common algorithm as well and admits a very simple extension to solve classification problems that have more than two classes. This is the case of the faced problem, in which the class column ("safety") has four different values therefore there is the same number of classes. The particoular classification problem considered is simple enough to allow the adoption of not so sofisticated algorithms.

b. Chosen Machine Learning performance evaluation metrics

A commonly used metric to evaluate the results of a classification algorithm is the accuracy of the prediction on unseen data during the training, which I also adopted in this project. The accuracy is computed as the number of samples correctly classified over the total evaluated during the test phase. Unfortunately, the dataset provided has very unbalanced classes, as can be seen from the *Figure 1*. In particoular, more than 70% of the samples in the dataset belong to the class "unacc" while the classes "good" and "vgood" have less than 4% of the total rows each. This has negative effects on training and high accuracy results are achieved just by classifing all the data as the most frequent class, therefore I also chose to adopt other metrics to evaluate the results that are more reliable in these cases, that are the precision, the recall and the F1 score. They are very common metrics used in classification tasks as well. I also chose to plot the

confusion matrix for each trained classifier in order to visually check if there are some unpredicted classes and the effects of having such an unbalanced dataset.

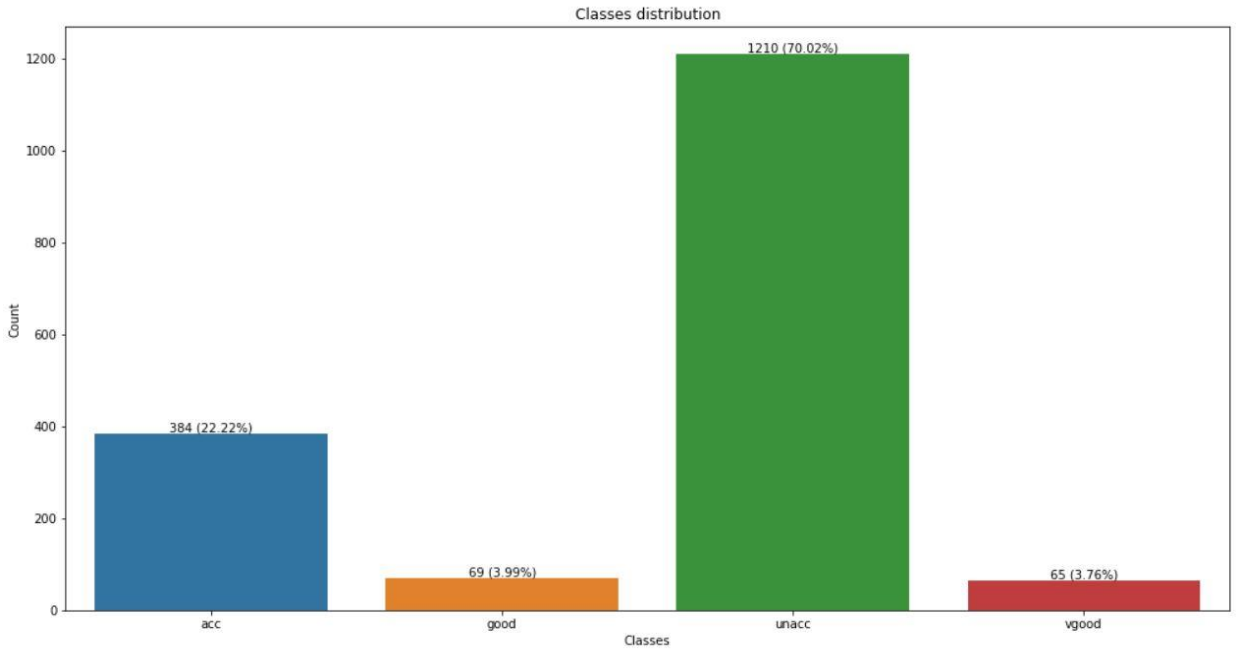

Figure 1.

c. Imputation tecniques selected

The standard imputation tecnique I selected for the project is the mode imputation. This choice was made mainly due to the fact that all the attributes of the dataset are categorical so imputation methods like the mean couldn't be used. Other tecniques that work well also with cathegorical varibles that are the forward/barckward fill were also considered but since the dataset has many consecutive equal values in every column these very simple tecniques allow to reach a reconstruction accuracy of around 90%. In order to have a fairer comparison with the chosen advanced tecnique and to avoid exploiting such particular regularity in data, the forward/barckward fill tecniques were abandoned.
The choice of the advanced imputation tecnique was not straightforward. I first tried to implement a KNN imputation method, but the imputer class provided by Sklearn uses as default distance metric the euclidean distance, which is not a good choice for data that have categorical attributes. Therefore, I chose to work with the Hamming distance that instead works well for categorical attributes. Unfortunately, the KNN imputer method provided by Sklearn requires the users to define their own distance metric that can work with the other methods, in particoular in handling the missing values, whose class type must be passed as an argument to the imputer. I tried to implement a Hamming distance function but many errors arised due to incompatibility with other methods and I was not able to figure out how to handle missing values, so I decided to change algorithm. I decided to exploit the fact that all the attributes to impute were ordinal and therefore I implemented a Matrix Factorization imputer. Matrix Factorization algorithms are very common in the recommender systems field to predict

the ratings of some users about some items, that can be seen as ordinal categories. Therefore, I thought that they a Matrix Factorization algorithm can also be used for the impuation problem considered provided that the values of the attributes are encoded as numbers.

2. PIPELINE IMPLEMENTATION

The first step of the the code implementation was the import of the dataset and its inspection as I first wanted to gather some information about data thus discovering that all attributes are categorical and in particoular ordinal, by printing their unique values, and that in the target column there are 4 different values so there are 4 classes. I also plotted a barplot to count how many samples there are in each class thus discovering that the dataset is very unbalanced. Then I used the function provided in the file "dirty_completeness.py" to inject randomly some missing values in the dataset in different percentages and I checked if the procedure was done correctly by evaluating the completeness of the dirty datasets.
The next step was the implementation of the imputation tecniques as function I could call multiple times. I started with the standard one, the mode, which uses the methods already provided by the Pandas library. The advanced imputation tecnique, the matrix Factorization, required the implementation of a function to encode the the values of the dataset, except the missing values, into integer numbers before giving it as an imput to the MF imputer, and another function that instead decodes the numbers of the imputed dataset so to obtain the original categorical values.
After, there is the implementation of the Machine Learning algorithms. In the code there are two functions that are used for both algorithms. The first one encodes the labels in the target column of the input dataset as integer numbers by using the same method implemented for the Matrix Factorization imputer and converts the values of the other columns in one-hot format. Moreover, the dataset is split into a training set and test set according to a split ratio I fixed equal to 20% (of total data in the test set) for every execution of the algorithms. The other function instead has the task of evaluating the results of a trained classifier by computing the metrics reported in the previous section and the confusion matrix as it recives as input the predictions and the true labels of the data in the test set. The implementation and training of the classifiers was done by using the methods of the library Sklearn.
All the methods described above are called into a for loop that iterates through all the five dirty datasets obtained after the missing values injecture procedure and one dataset at a time is used to perform the following operations in sequence:

- Completeness evaluation of the dirty dataset.
- Data imputation with standard tecnique (mode) and accuracy assessment of the imputed dataset.
- Data imputation with advanced tecnique (MF) and accuracy assessment of the imputed dataset.

- Training of a Decision Tree classifier on the dataset imputed with the standard tecnique.
- Training of a Decision Tree classifier on the dataset imputed with the advanced tecnique.
- Training of a Logistic Regression classifier on the dataset imputed with the standard tecnique.
- Training of a Logistic Regression classifier on the dataset imputed with the advanced tecnique.

The results of each operation at each iteration are stored into a list specific for the particoular operation. After the main loop, the final step was showing the results with some plots, as I will discuss in the next section.

3. RESULTS

In order to better visualize the results obtained I choose to plot a bar graph for each classifier performance evaluation metric (accuracy, precision, recall, …) that has along the x-axis the percentage of injected missing values while on the y-axis the metric values are reported. For each missing values percentage there are two bars in the same plot, one blue on the left that corresponds to the dataset imputed with standard tecnique (the mode) and the other orange on the right that corresponds to the dataset imputed with the advanced tecnique (Matrix Factorization). A bar graph build in the same way is also used to compare the accuracies of the imputed datasets with the two different tecniques, as can be seen in the *Figure 2*.

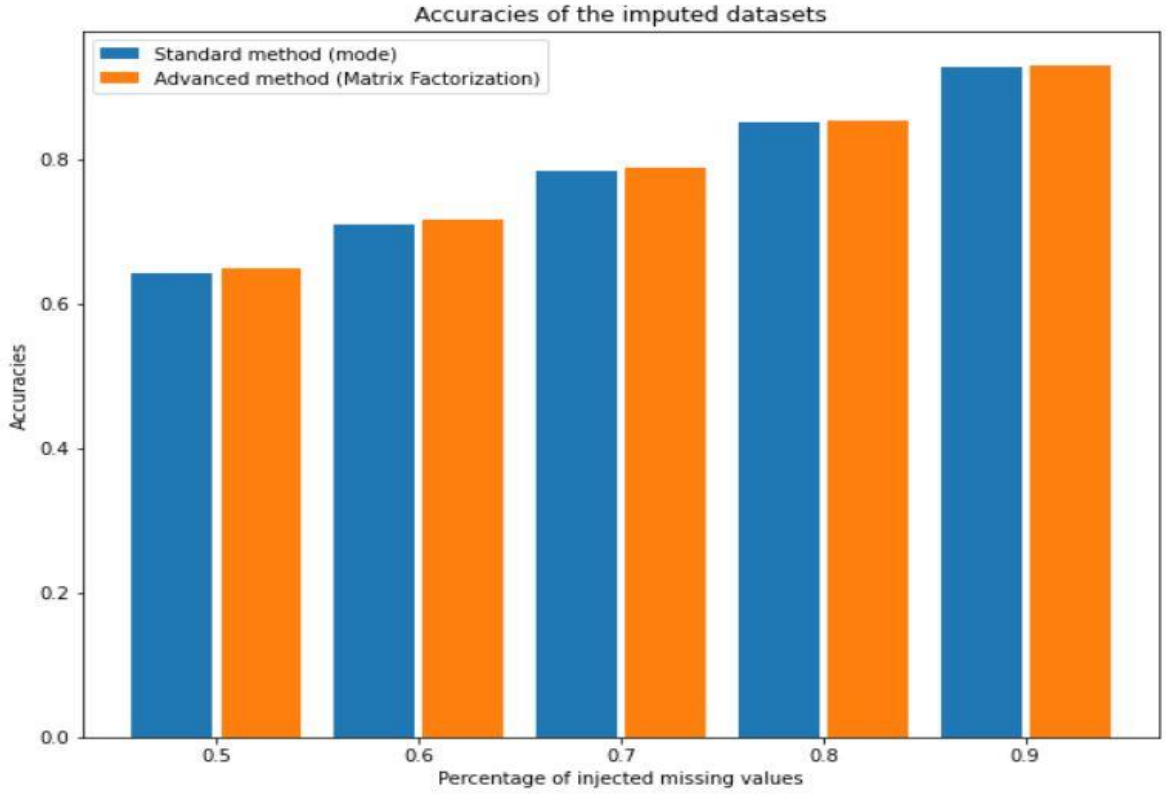

Figure 2.

The accuracies of the imputed datasets for each percentage differ only for values in the order of 10^{-2} between the two tecniques but those of the MF method are slightly better.

About the results of the Machine Learning algorithms, it can be observed that for high percentages of injected missing values (i.e. 50% and 60%) the classifiers performed better when trained on the dataset imputed with the mode tecnique while for the other percentages the results are in general better with the datasets imputed with the Matrix Factorization algorithm, in particular for the precision, recall and F1 score of the classifiers. Examples of this behavoiur are evident in the *Figure 3* and *Figure 4*. These results might be explained by considering that the Matrix Factorization tecnique has a training phase which of course gives better results when there are more data to use as trainig set while the test set is made of the data points with the missing values. In addition, the mode tecnique has as drawback the fact that reduces the variability of the dataset which in general affects negatively the training of a learner.

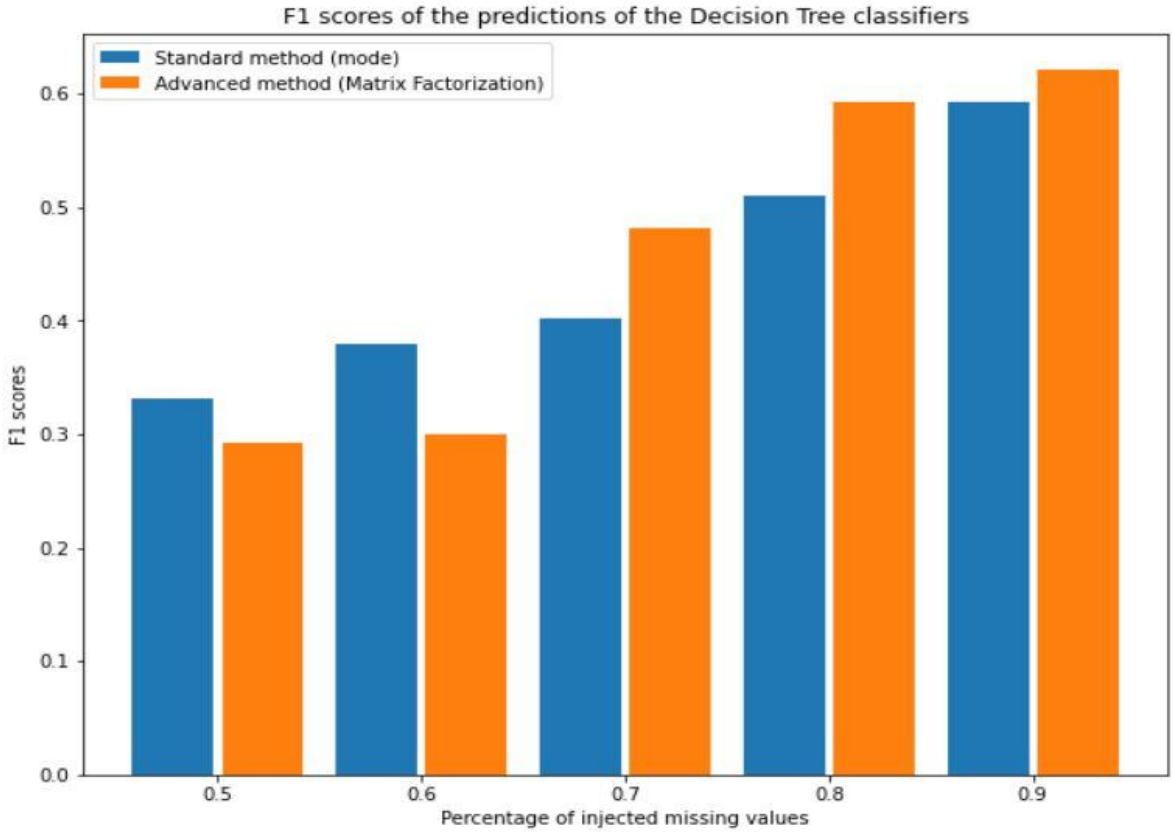

Figure 3.

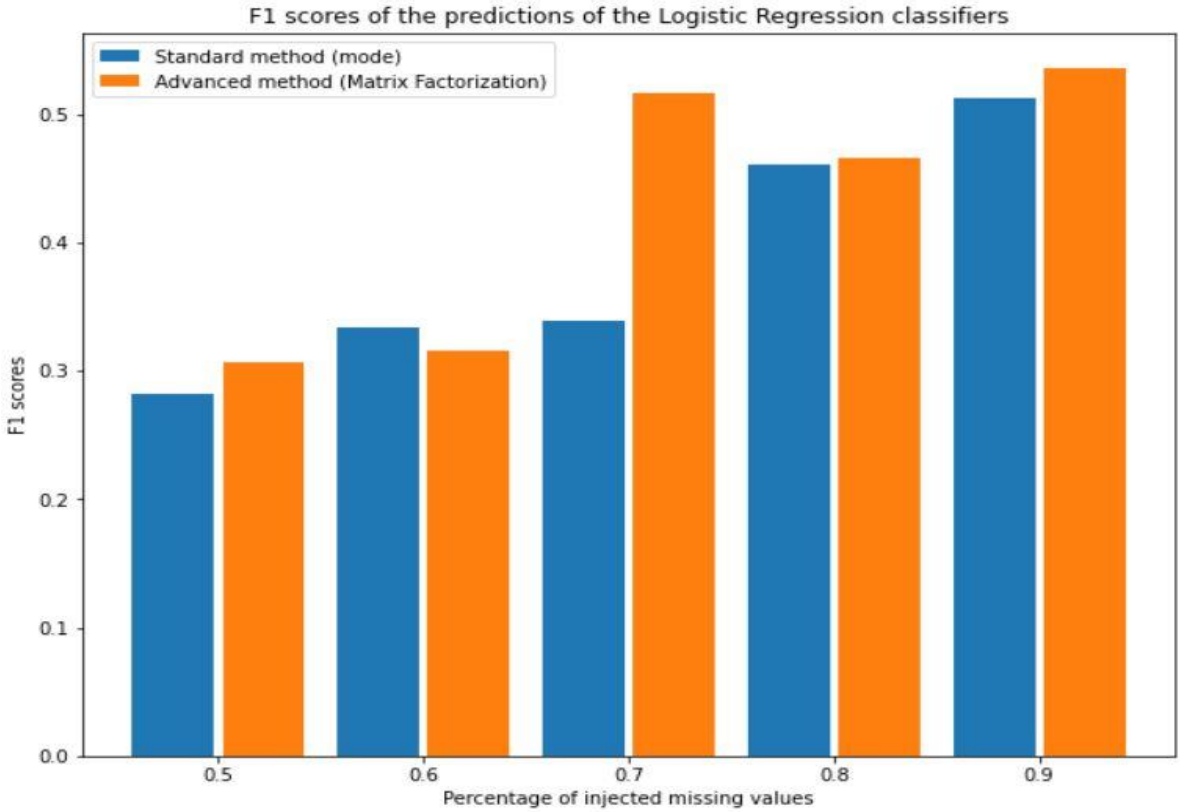

Figure 4.

Comparing the two algorithms, they both reached similar results in the accuracy of the predictions (*Figure 5* and *Figure 6*), maybe also due to the unbalance in the dataset, but the other metrics were in general higher for the Decision Tree classifiers, as can be seen in the plots of the F1 scores since they are computed from the precision scores and the recall scores.

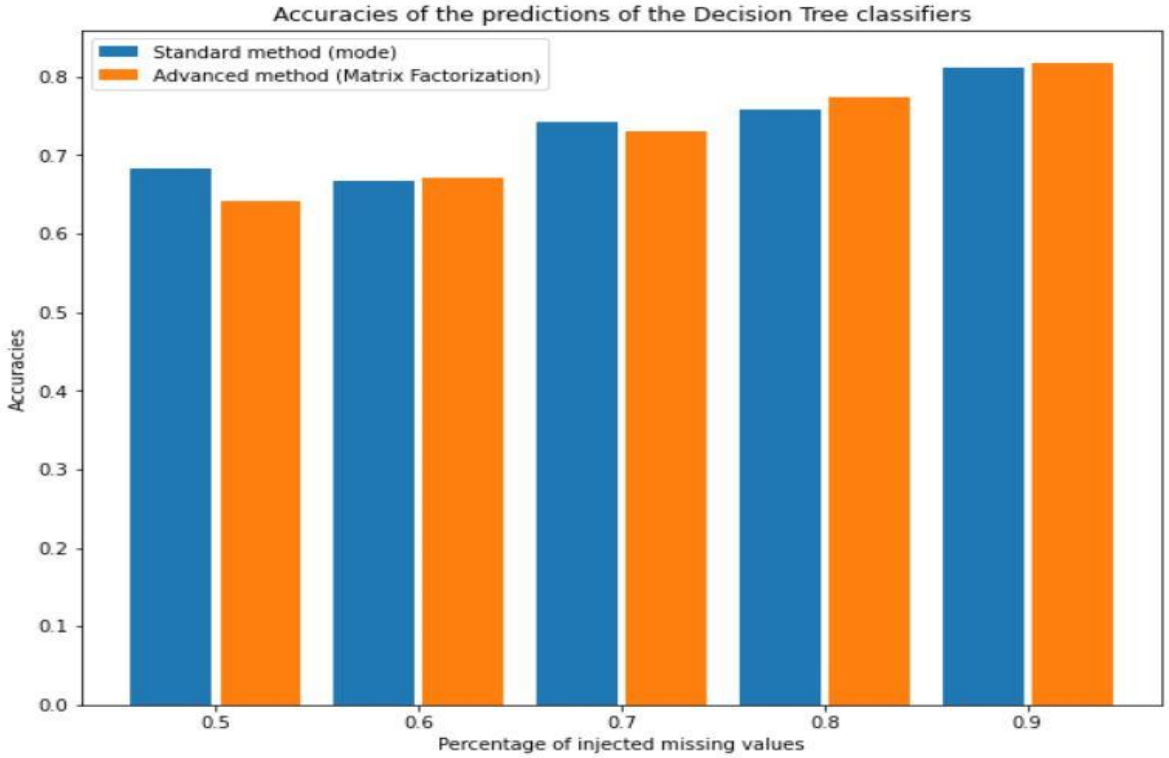

Figure 5.

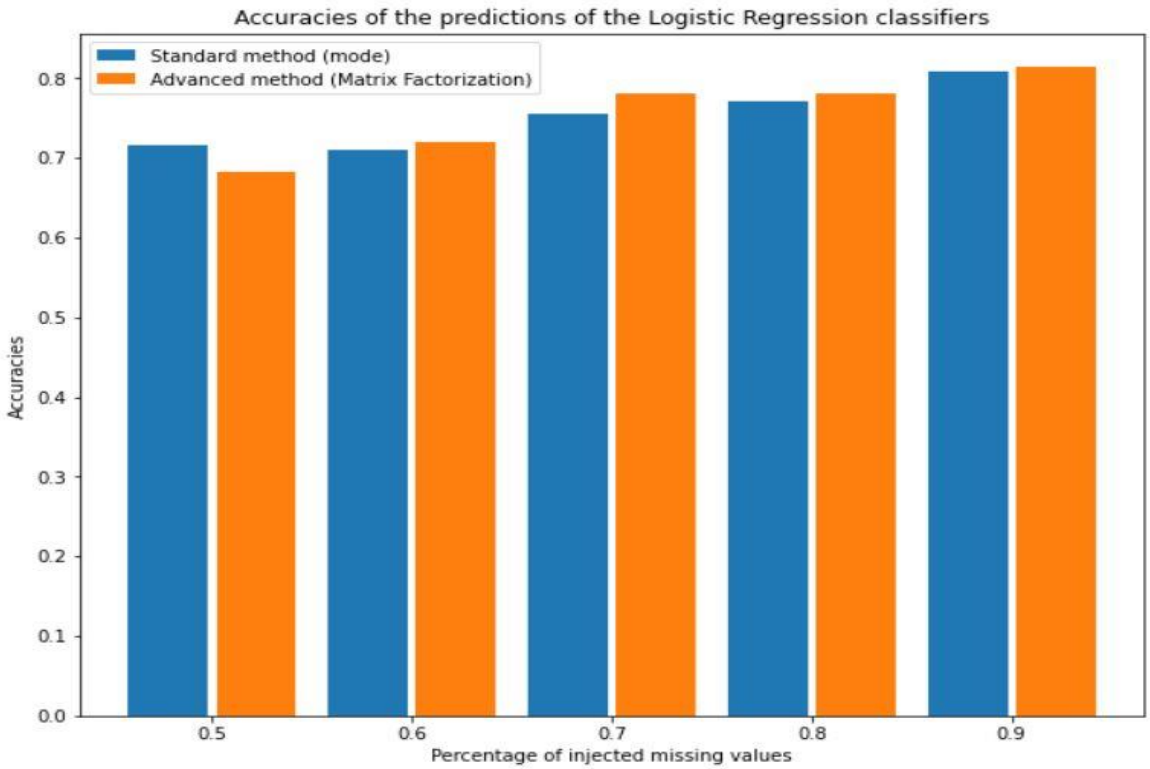

Figure 6.

In the end, a look at the confusion matrices (*Figure 7* and *Figure 8*) higlights that when the percentage of injected missing values is high the classifiers tend to predict many data points in the test set as “unacc”, that is the most frequent class, while the least frequent classes are almost unpredicted. Anyway, this bahaviour shows less for the other percentages.

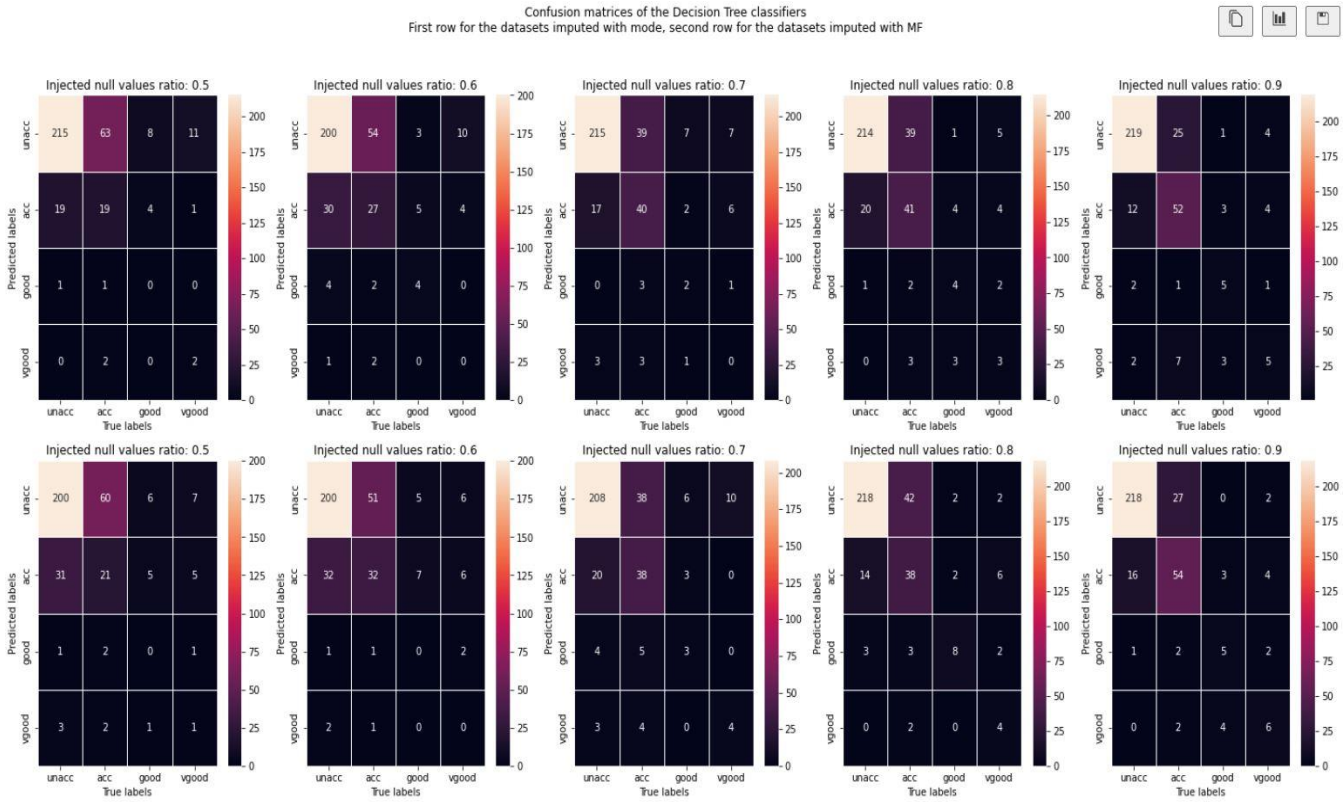

Figure 7.

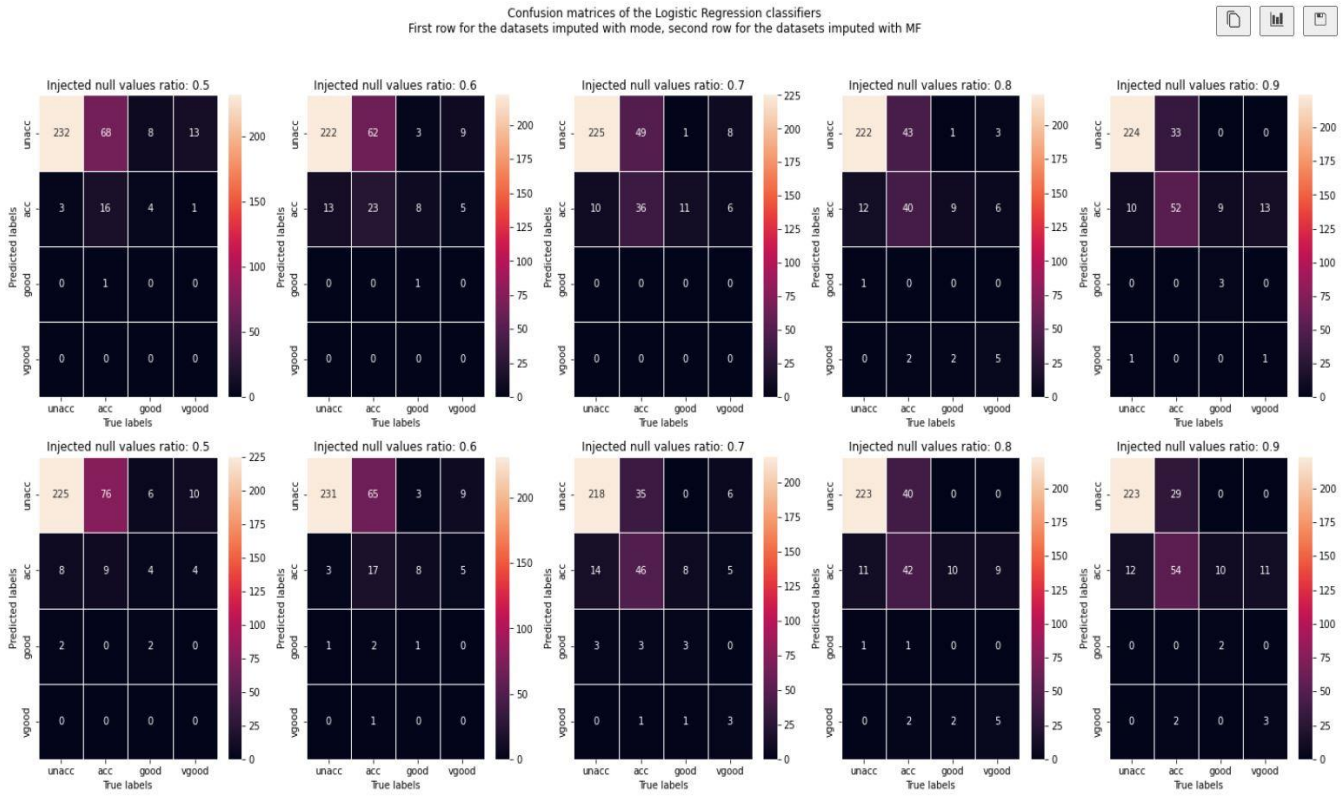

Figure 8.