Data and Information Quality

Project Report

Project ID: *32*
Project number: *2*
Assigned dataset: *german.csv*
Student: *Giuseppe Piccirillo 10568059*
Assigned task: *Classification*

1 Setup choices

1.1 Chosen ML algorithms

Reading the literature, I decided to choose the following two machine learning algorithms: Support Vector Machines and Decision Trees.
Both of them are algorithms used for classification task, but the first one is known to be robust to outliers, instead the second one is sensitive to outliers.

I wanted to verify if these characteristics are still valid on this dataset.

1.2 Chosen ML performance evaluation metrics

I decided to choose accuracy as metric for both of the algorithms used, because the goal of the classification task on this dataset is to evaluate the risk of a bank to give credit to a person, so being inaccurate can lead to a waste of money.

1.2.1 Notable observation

I tried to run several metrics to compare different performances of the machine learning algorithms.
Using SVM, the metric ”fowlkes_mallows_score” performed around 5% better than the accuracy most of the time, but it’s a metric most commonly used for clustering tasks.
Using DT, the metric ”precision_weighted” performed around 2% better than the accuracy most of the time.

However, to be coherent in the entire project, I decided to choose accuracy, which was always the first or the second best performing metric.

1.3 Outlier detection techniques selected

I decided to test one distance based outlier detection technique and one model based outlier technique: Local Outlier Factor and k-Nearest-Neighbours.
The first type of technique can work well choosing a good threshold, otherwise the risk is to identify right values as outliers.
The second type of technique is more effective and faster than the previous one, but being a supervised method it needs accurate labels.

1.3.1 Deletion/correction

I decided to show the effects of delete the detected outliers in every dataset with different percentage of outliers injected, and likewise for correction (where I decided to substitute the outliers with the average of the corresponding attribute's values in the dataset).

I wanted to verify if the more outliers there are in the dataset, the more is convenient to correct them, otherwise it would be a waste of too many tuples of the dataset.
Similarly, I expected to verify that if there are few outliers, deleting them can be a possible solution from a performance point of view.

2 Pipeline implementation

2.1 Analyze dataset with outliers

After importing the data and injecting the outliers with five different percentage, I created the profile and generated the corresponding report for every injected dataset.
I noticed that in all the datasets there are no duplicates or missing values, just outliers.
These outliers consists of modifying some numerical values with random positive or negative numbers (the categorical values didn't change).
In the original dataset there is a high correlation between the attributes "Credit amount" and "Duration", but the more we inject outliers, the more this correlation disappears.

2.2 Data preparation

It's always necessary to do some pre-processing of the data in order to run ML algorithms.
Both of the chosen algorithms need no missing values, which actually are not present in our data.
Decision Trees work better without outliers, instead Support Vector Machines are robust to them.

Given the fact the injected outliers are only in numerical attributes, in this phase I also created a smaller dataframe from the original one, dropping the categorical attributes.
Furthermore, I created a normalized version of this dataframe to show the differences on the improvements with a non-normalized dataset, and also because distance based outlier techniques should work better with a normalized dataset.

2.3 Compute the ML algorithms on the dataset with outliers

In this phase, I run the chosen algorithms on the datasets with different percentage of outliers, and I saved the performance results.

Fine tuning the train and test sets, I noticed that setting test_size=0.25 and n_splits=5 of the "ShuffleSplit" function is a good trade-off in both the algorithms.
Also, I set the default "rbf" kernel of the SVM classifier, because it performed slightly better than "poly", "linear" and "sigmoid".

2.4 Outlier detection and deletion/correction

Considering the percentage of injections is known, I choose every time a threshold for the LOF and KNN technique in order to find more or less the same percentage of outliers.

Found the probable outliers, I both deleted them and corrected them (substituting them with the average of the corresponding attribute's values) to show the differences in the performances, according to the percentage injected.

2.5 Compute the ML algorithms after the deletion/correction of outliers

I run again the chosen ML algorithms, but on the new datasets: without outliers or with the corrected outliers.
I saved all the values obtained in order to compare them through the graphs in the next section.

3 Results

I chose to show three different comparisons for each machine learning algorithm applied.
The first one is the comparison of the resulted performances using the LOF outlier detection technique, so comparing the performances of the algorithm with the deletion and the correction of outliers, and the normalization of the dataset(Figure 1 and Figure 5).
Similarly, the second comparison is about the use of the KNN outlier detection technique (Figure 2 and Figure 6).
Lastly, a global comparison between LOF and KNN techniques (Figure 3 and Figure 7).

In order to check the if the outlier detections gave a general improvements on the given dataset, I compared the performances obtained by each ML algorithm after the outlier detection and deletion/correction with the results obtained by the corresponding ML algorithm run on the datasets with outliers (Figure 4 and Figure 8).

Figure 1: Comparison of the LOF technique with SVM

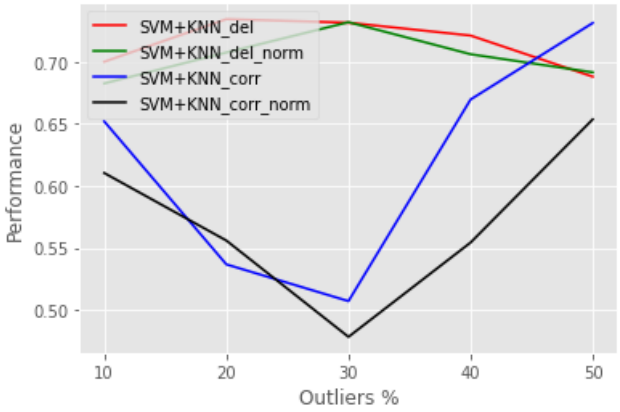

Figure 2: Comparison of the KNN technique with SVM

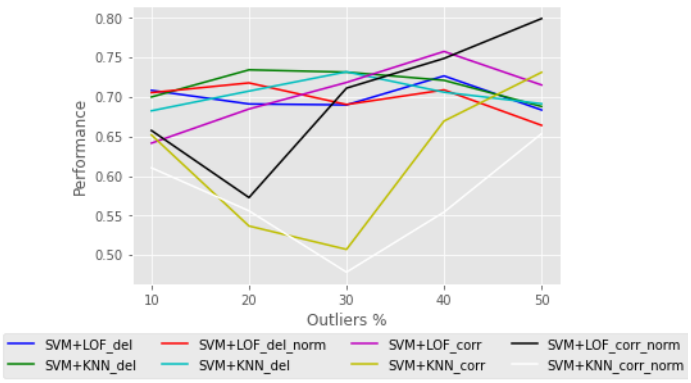

Figure 3: Comparison between LOF and KNN with SVM

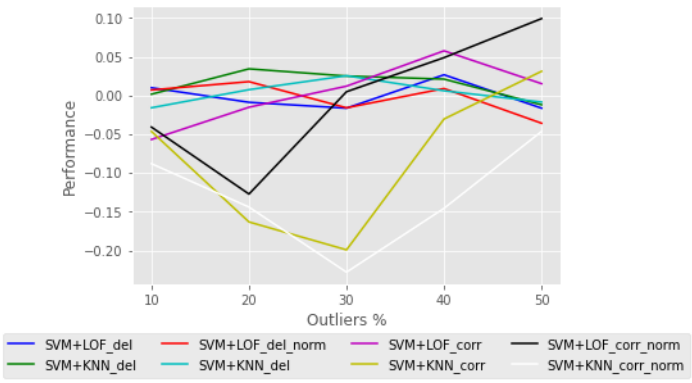

Figure 4: Improvements of the outliers detection with SVM

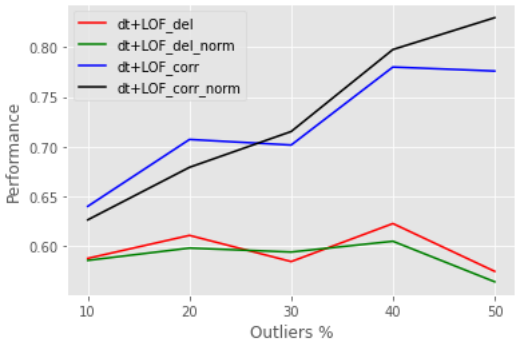

Figure 5: Comparison of the LOF technique with DT

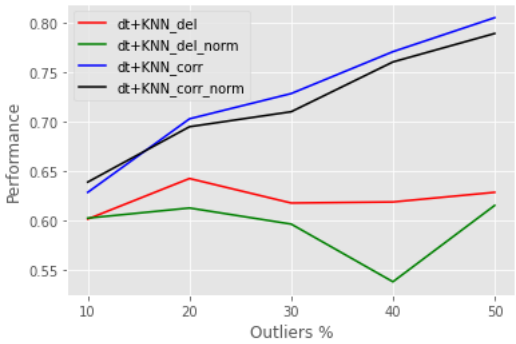

Figure 6: Comparison of the KNN technique with DT

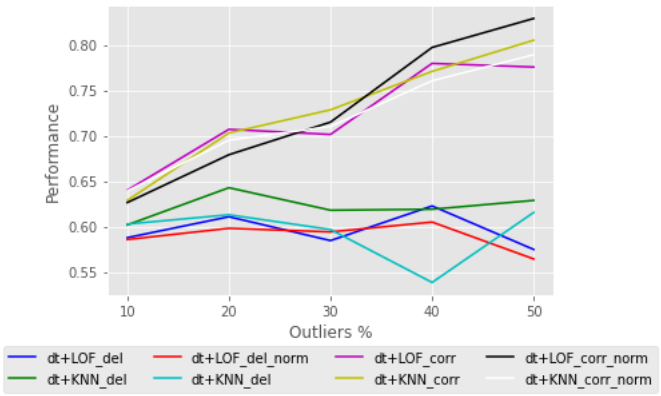

Figure 7: Comparison between LOF and KNN with DT

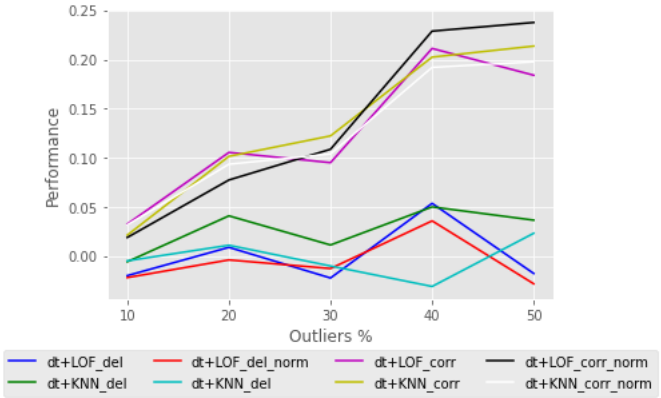

Figure 8: Improvements of the outliers detection with DT

3.1 Conclusions

Looking at the graphs it's possible to notice that on average SVM performs better using LOF than KNN, and the more the percentage of outliers increases the more there is a better performance (if the correction of outliers is applied).
Actually, looking at the graphs about DT, it's possible to notice more clearly the different trends of the accuracy working with a dataset with corrected outliers or not: in the first case, the more outliers we have, the more accuracy there is (as expected).
Indeed, deleting outliers slightly decreases the results of the datasets with 30% and above.

Normalize the datasets seems to lightly help the accuracy with a large number of outliers.

At the end, it's possible to notice that SVM is stable (Figure 4) even if the percentage of outliers increases, instead DT changes significantly (Figure 8).
Furthermore, it's possible to notice that the outlier detection techniques lead to an improvement in most of the cases.