DATA AND INFORMATION QUALITY PROJECT REPORT

PROJECT ID: 48

PROJECT NUMBER: 2

ASSIGNED DATASETS: adult.csv – frogs.csv

STUDENTS: Lara Ferro 10622035 – Stefano Fumagalli 10628587

ASSIGNED TASK: Classification

1. SETUP CHOICES

For the classification task on the provided datasets, we chose ML algorithms from two different scikit learn sublibraries:

- RidgeClassifier from the *sklearn.linear_model* library
- DecisionTreeClassifier from the *sklearn.tree* library

To evaluate the performance of the chosen ML algorithms we focused on four different metrics from *sklearn.metrics* scikit learn's library:

- Accuracy → Fraction of predictions the model got right ($\frac{TP+TN}{TP+TN+FP+FN}$)
- Recall → Proportion of actual positives that was identified correctly ($\frac{TP}{TP+FN}$)
- Precision → Proportion of positive identifications that was actually correct ($\frac{TP}{TP+FP}$)
- F1 score → Combination of precision and recall by taking their harmonic mean, used to compare the performance of two classifiers ($\frac{2(P*R)}{P+R}$)

As outlier detection techniques we selected:

- Z-Score as standard technique
- K-Nearest-Neighbors as advanced technique

2. PIPELINE IMPLEMENTATION

Import dataset

For both the datasets, we first imported the data by means of the read_csv method of the pandas library and got the dirty datasets injecting wrong values.

For each dirty dataset:

ML algorithms performance evaluation before outlier detection

We defined as test-set the data in the original dataset, transforming all the string attributes in integers through the LabelEncoder object and then setting as labels the column *income* for the adult dataset and *Species* for the frogs one.

After this step, we also split in *X, y* pairs the dirty dataset and used these values as training-set to fit the ML Classifiers.

Then, we evaluated the performance of the algorithms with the chosen metrics, using the test-set to make the predictions.

In parallel:

Outlier detection with standard technique

We examined each column at a time, applying the Z-Score technique to the numerical ones. To do this, we isolated the column we were interested in and passed it as parameter to the function, together with the threshold that we chose after several tuning attempts.

Once detected, we stored the position of the outliers in a vector and we converted these values in NaN, in order to correct them performing imputation.

With the aid of graphs, we studied the distributions of the values and thus we selected the technique we thought it could be the most appropriate to clean the wrong values: we mainly focused on filling them with the mode or the MICE technique, depending on the attribute analyzed.

Outlier detection with advanced technique

As for the detection with the standard technique, we examined each column at a time, applying the KNN technique to the numerical ones. Once again, starting from the dirty dataset, we isolated the column we were interested in, reshaped its values and, after having defined the number of neighbors, we fitted them and plotted the mean of k-distances of each observation.

Then, we detected the outliers comparing the mean distance with a threshold and, if it was greater than the chosen value, the position of the data was stored in a vector. We also plotted outliers and real value in a graph to observe them.

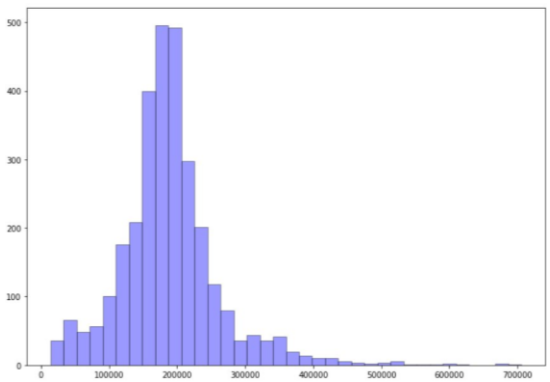
After that, we converted the outliers in NaN values, in order to then correct them performing imputation.

Always with the aid of graphs, we studied the distributions of the values and then we selected the technique we thought it could be the most appropriate to clean the wrong values: once again, we mainly focused on filling them with the mode or the MICE technique, depending on the attribute analyzed

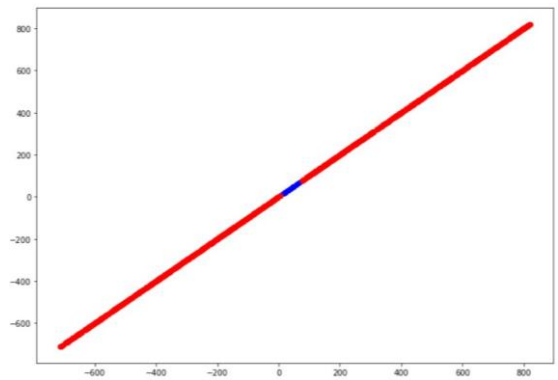
ML algorithms performance evaluation after outlier detection

We split again in *X, y* pairs the dataset after having performed outlier detection and corrected them, both with Z-Score and KNN and, as done before, we used these values as training-set to fit the ML Classifiers.

As last step, we evaluated the performance of the algorithms with the same evaluation metrics, using the original data as test-set to make the predictions.



Example of graph used with ZS



Example of graph used with KNN

3. RESULTS

After performing outlier detection and imputation we were able to analyze the results of our work by measuring the performance with the two machine learning algorithms.

As shown in the following tables, the accuracy, recall, precision and F1 score of the classifiers always increased when the cleaned datasets were used to learn instead of the dirty ones.

We found out that with Ridge Classifier the performances were the same for each level of accuracy while with Decision Tree Classifier the performances were linear with the accuracy of the dataset with reference to the original data.

Another interesting result was that after cleaning the dataset we were able to get a different behavior from the two classifiers: with Ridge Classifier the performances were very similar for each level of accuracy of the dirty datasets, while with Decision Tree Classifier we were able to obtain better performances starting from the dataset with less outliers and having a decrescent trend till the dataset with more outliers.

In general, we found out that Decision Tree Classifier worked better than Ridge Classifier with a considerable difference in the *adult* dataset and a less strong difference in the *frogs* dataset, as shown by the F1 score.

Finally, we noticed that the difference between Z-score and K-Nearest-Neighbors were almost negligible: KNN were able to obtain better performances in almost all the cases but with a small margin of improvement with respect to ZS.

Adults		Accuracy 90%			Accuracy 80%			Accuracy 70%			Accuracy 60%			Accuracy 50%		
		Before	*After ZS*	*After KNN*	*Before*	*After ZS*	*After KNN*	*Before*	*After ZS*	*After KNN*	*Before*	*After ZS*	*After KNN*	*Before*	*After ZS*	*After KNN*
RidgeClassifier	*Accuracy*	74,88%	80,88%	81,67%	74,88%	81,37%	82,13%	74,88%	82,57%	82,66%	74,88%	82,13%	82,10%	74,88%	82,43%	79,95%
	Recall	0,13%	42,14%	39,63%	0,13%	47,03%	41,35%	0,13%	47,16%	45,97%	0,13%	48,75%	46,90%	0,13%	85,97%	41,88%
	Precision	33,33%	69,65%	75,76%	33,33%	68,86%	76,72%	33,33%	73,91%	75,32%	33,33%	70,96%	72,01%	33,33%	74,20%	65,77%
	F1 Score	0,26%	52,51%	52,04%	0,26%	55,89%	53,73%	0,26%	57,58%	57,10%	0,26%	57,79%	56,80%	0,26%	56,77%	51,17%
DecisionTreeClassifier	*Accuracy*	93,17%	91,28%	96,39%	88,56%	91,71%	92,94%	85,48%	90,06%	90,06%	83,66%	87,24%	88,17%	81,47%	84,59%	86,08%
	Recall	85,87%	84,54%	93,13%	76,75%	86,13%	87,19%	71,47%	82,03%	82,43%	66,05%	77,01%	79,39%	58,12%	74,50%	79,26%
	Precision	86,78%	81,42%	92,52%	77,47%	81,81%	85,05%	70,90%	79,11%	78,89%	67,93%	73,43%	74,94%	64,52%	67,46%	69,52%
	F1 Score	86,32%	82,96%	92,82%	77,11%	83,91%	86,11%	71,18%	80,54%	80,62%	66,98%	75,18%	77,10%	61,15%	70,81%	74,07%

Performances of adult dataset

Frogs		Accuracy 90%			Accuracy 80%			Accuracy 70%			Accuracy 60%			Accuracy 50%		
		Before	*After ZS*	*After KNN*	*Before*	*After ZS*	*After KNN*	*Before*	*After ZS*	*After KNN*	*Before*	*After ZS*	*After KNN*	*Before*	*After ZS*	*After KNN*
RidgeClassifier	*Accuracy*	47,60%	88,91%	89,31%	47,60%	89,14%	89,11%	47,60%	88,22%	88,84%	47,60%	87,85%	88,08%	47,60%	86,59%	87,72%
	Recall	47,60%	88,91%	89,31%	47,60%	89,14%	89,11%	47,60%	88,22%	88,84%	47,60%	87,85%	88,08%	47,60%	86,59%	87,72%
	Precision	22,66%	87,75%	87,55%	22,66%	87,63%	87,29%	22,66%	85,72%	87,65%	22,66%	85,25%	85,33%	22,66%	84,29%	85,11%
	F1 Score	30,70%	86,33%	86,74%	30,70%	86,64%	86,67%	30,70%	85,28%	86,16%	30,70%	84,75%	84,99%	30,70%	83,43%	84,68%
DecisionTreeClassifier	*Accuracy*	97,85%	97,65%	98,44%	94,77%	96,56%	96,66%	91,56%	93,21%	92,88%	88,84%	91,69%	91,49%	84,67%	87,39%	87,95%
	Recall	97,85%	97,65%	98,44%	94,77%	96,56%	96,66%	91,56%	93,21%	92,88%	88,84%	91,69%	91,49%	84,67%	87,39%	87,95%
	Precision	97,86%	97,65%	98,45%	94,97%	96,55%	96,67%	91,71%	93,28%	93,04%	88,71%	91,70%	91,57%	84,91%	87,95%	88,26%
	F1 Score	97,83%	97,63%	98,44%	94,82%	96,54%	96,63%	91,48%	93,20%	92,90%	88,70%	91,65%	91,51%	84,14%	87,48%	88,02%

Performances of frogs dataset