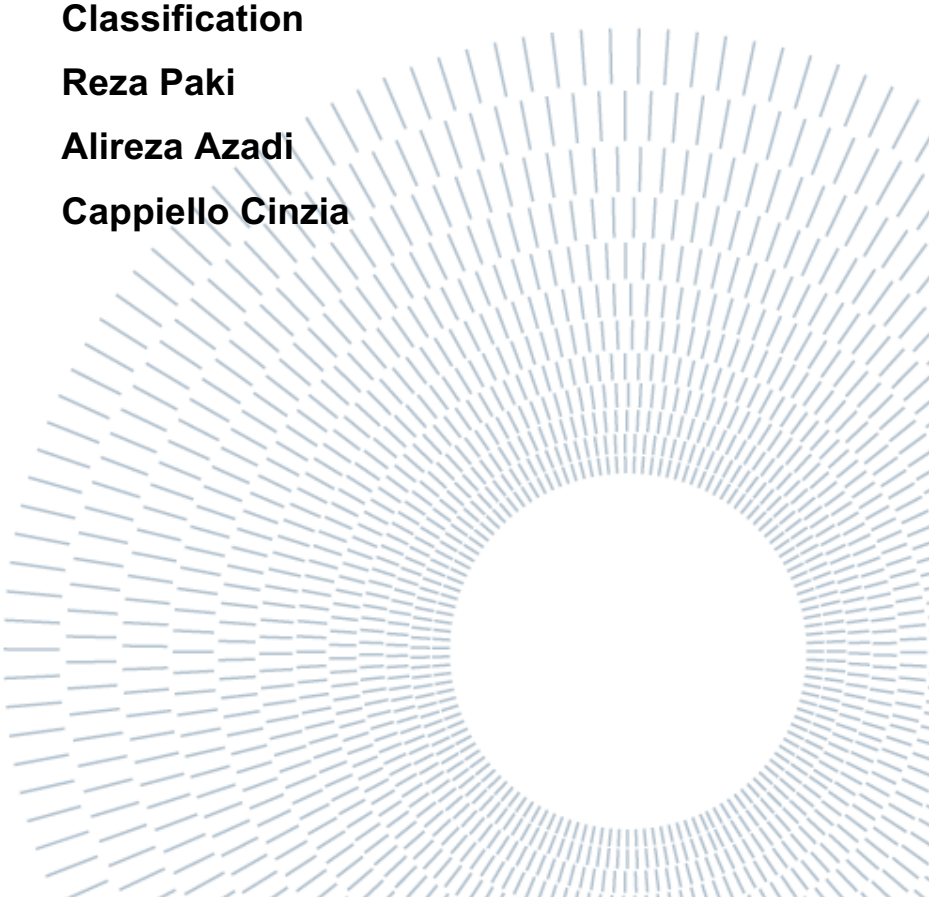
POLITECNICO
MILANO 1863

Department of Electronics, Information and Bioengineering
Doctoral Programme In Information Technology

DIQ project report

Duplication and variable type DQ issue handling

Project ID:	**40**
DQ issue:	**Duplication&Variable type**
ML task:	**Classification**
Student 1 Name and ID:	**Reza Paki**
Student 2 Name and ID:	**Alireza Azadi**
Professor Name:	**Cappiello Cinzia**



1. Setup choices:

In this project, we address two distinct Data Quality (DQ) issues: variable type issue and data duplication. Our evaluation involves testing six diverse classification methods—DecisionTree, LogisticRegression, KNN, RandomForest, AdaBoost, and MLP—on a dataset comprising 1000 samples to assess various scenarios. Given that our methods are classification-oriented, real targets for the generated data are essential. The collected results encompass model performance (f1-score), algorithm speed (corresponds to the running time, higher speed means higher execution time), and distance (difference of f1-score between train and test). This comprehensive analysis provides insights into the effectiveness and efficiency of the employed classification methods under different conditions.

1.1. Variable type:

In addressing the variable type data quality (DQ) issue, our implementation involves working with 15 features. Out of these, 5 features need to be converted into Boolean variables, and another 5 into categorical variables. Since the data generator function currently generates only numeric variables, we need to devise a method to transform these numeric variables into Boolean and categorical ones.

1.2. Duplication:

The first function that we created is data_generation function that we have created, here 1000 samples with 10 numerical features created. As far the generated data is in numpy.array format we renamed the column names from A to J. First, we check the existed duplication, and we announced that there is no duplication in none of the features.

2. Pipeline implementation:

2.1. Variable type:

The initial function implemented for categorization is 'categorize,' which takes the generated data as input. Since all variables are initially numeric, the function transforms 5 of them into categorical variables using various methods:

1) The first variable is converted into a categorical variable with 10 categories. This is achieved by considering the minimum and maximum values of the column, dividing it into 10 different ranges, and assigning a category randomly to each range.
2) Similar to the first variable, but with 5 categories the 4th variable.
3) Randomly assigns 5 categories to each record in the 5th variable.
4) Randomly assigns a category to each row, selecting from 10 categories, for the 8th column.
5) For the 12th variable utilizes a combined method by randomly assigning a category to half of the data, and for the other half, employs the approach used for the 1st variable, choosing from 10 different categories.

The second function, 'boolean_features', takes the data as input and transforms certain variables into boolean variables. It applies the following operations:

Converts the 2nd variable into a boolean variable using a threshold of 0. Values above 0 are considered '1', while others are '0'. Transforms the 6th variable into a boolean variable with a threshold of 1. Converts the 9th variable into a boolean variable with a threshold of -1. Randomly assigns boolean values to variables 10 and 13.

The last function, 'randomly_feature_selection', takes the input data and randomly selects between 5 to 15 features for analysis. The selected features can include numerical, boolean, and categorical variables.

2.2. Duplication:

The "duplication_creation" function is designed to generate duplicated data with specified similarity and duplication levels. Given input data(X,y), a similarity level, and a duplication level, the function creates duplicates with a similarity percentage, ensuring a proportion of columns remains unchanged while the rest undergo modifications. For instance, with a 40% similarity level, 40% of columns retain their original values, and the remaining 60% are altered by introducing

uniform noise within the respective column's range. The duplicated data is then randomly inserted into the dataset, by considering random labels to the duplicated samples.

The "duplication_detection" function takes input data (X and y) along with a specified similarity level. The function initiates a comprehensive record linkage process, creating linkages between records. For each linkage, the "compare_indexes" function is invoked to assess the similarity between selected rows based on the provided similarity level. Upon detection of duplicated data, it is systematically removed from the dataset. This function serves as an essential step in identifying and eliminating duplicate entries within the dataset, ensuring data quality and integrity.

The subsequent phase involves evaluating the models with respect to the Data Quality (DQ) issues under consideration. Our objective is to run the models for each similarity level, duplication level, and classification algorithm, capturing results at two distinct instances: first, with the inclusion of duplicated data, and second, after performing data cleaning and removing duplications. It is imperative to store and compare the results through visualizations such as plots and tabulated summaries to gain insights into the impact of data duplication on model performance and the effectiveness of data cleaning procedures.

3. Results:

3.1. Variable type

First of all we had several prilimary results on performance by decision tree algorithm on different set of features to see and have an intuition to the upper bounds for each kind of data type.

1. Evaluation model with all numeric features that we got 83.90,
2. Evaluation model with 5 numeric features 57.15,
3. Evaluate model with 10 numeric features and 5 categorical features71.19,
4. Evaluate model with 10 numeric features and 5 boolean features 56.66,
5. Evaluate model with 5 numeric features,
6. 5 boolean features and 5 categorical features 73.91.

Run number	Algorithm name/ Selected features					
	Decision Tree	Logistic Regression	KNN	Random Forest	AdaBoost	MLP
1	0-2-3-4-5-6-7-8-11-12-14	0-1-2-4-5-7-8-9-12-13-14	0-4-5-8-10-11	0-2-3-4-5-8-9-13	2-5-6-7-9-12	0-1-2-3-4-5-6-7-8-10-11-12-14
2	0-1-2-3-5-7-9-10-11-12-13-14	0-1-3-4-5-6-9-12-13-14	2-5-6-10-11-12-13	0-1-3-5-6-7-8-9-10-11-12-13-14	0-1-2-3-4-5-6-7-8-9-10-13-14	0-1-3-4-6-11-12-13
3	0-1-2-3-4-5-6-8-9-10-11-12-14	1-2-3-4-5-6-7-8-10-11-12-13-14	1-3-5-7-8-9-10-11-13-14	2-4-6-7-8-10-14	0-1-2-5-9-11-12	0-1-2-3-4-5-6-7-8-9-10-11-12-13-14
4	2-3-5-8-9-10-11-13-14	0-1-2-3-4-5-6-7-8-10-11-12-13-14	1-2-3-4-5-6-7-8-9-10-11-12-13-14	0-3-9-10-12-13-14	0-2-3-5-6-7-11-14	0-4-7-11-14
5	0-1-2-3-4-5-6-7-8-9-11-13-14	0-1-2-3-4-5-6-7-8-9-10-11-12-13-14	0-1-3-5-6-7-9-12-14	0-2-3-4-5-6-7-9-10-11-12-13-14	0-1-3-4-5-6-7-8-9-10-11-12-13-14	1-2-3-4-5-7-9-11-12-13-14
6	0-1-2-3-4-5-6-7-9-10-11-12-13-14	0-2-3-5-6-10-13	0-2-4-7-10-14	0-1-2-3-4-5-7-8-9-10-11-12-13-14	1-2-3-4-5-7-8-9-10-11-13-14	0-1-2-3-4-5-6-7-8-9-10-11-12-13-14
7	5-7-8-10-12-14	0-1-2-4-5-6-7-8-10-11-14	3-6-8-10-13-14	1-4-6-7-8-9-10-11-13-14	0-1-3-4-6-8-12-14	0-1-2-3-4-5-7-8-9-10-11-12-13-14

8	0-2-3-4-7-8-9-10-12-13-14	1-2-3-4-5-6-8-9-10-14	0-1-2-3-4-5-6-7-8-9-10-11-12-13-14	1-2-3-13-14	0-1-2-3-4-6-7-8-9-10-11-12-13-14	0-1-3-5-8-9-11-12-13-14
9	2-3-5-7-9-10-11-13-14	0-1-2-5-6-8-9-10-11-12-13-14	0-2-3-5-8-9-11	0-1-2-3-5-6-7-8-9-10-11-12-13-14	0-1-3-6-8-9	2-3-5-6-9-11-12-13-14
10	1-3-5-6-8-9-10-11-12-13-14	1-2-4-10-13	0-2-4-5-6-7-9-11-12-13-14	1-2-3-5-7-10-11	0-2-4-5-7-10-13-14	0-1-2-3-4-5-6-7-8-9-10-11-12-13-14

Table 1: Displaying the randomly selected variables for each run and algorithm.

1) Performance evaluation:

In table 2 the bolded cells represent the highest performance among different runs for that specific ML algorithm. The green colors indicate the second-best run, the yellow one shows the third-best one, and the red one represents the worst case.

The general notion is that, in most instances, having more features tends to work well because the ML model can effectively distinguish between samples. However, this principle doesn't hold true universally and is contingent on the nature of the ML algorithm. Generally, numerical values provide descriptive information about samples, making them more informative. ML models can extract useful patterns from numerical features, potentially enhancing the model's performance. Categorical values excel in distinguishing between samples, as comparisons between different categories are straightforward, aiding in sample classification. On the other hand, Boolean variables are less informative. While classifying samples based on Boolean values is easy, there are instances where multiple samples share the same Boolean values, leading to confusion in classification. These are broad ideas about different types of variables, and their applicability depends on the algorithms used.

For instance, in a decision tree, if a numeric value is present in each branch, the algorithm divides the numeric variable into different ranges and compares samples, repeating this process multiple times to ultimately classify all samples. In contrast, when dealing with categorical variables, the decision tree considers a branch based on whether it belongs to category X or not. This distinction underlines the importance of collecting the indices of selected features to compare, at the very least, the three best performances of each model with the worst-case scenario.

For the decision tree algorithm, the best run is the 3rd run with 13 selected features, comprising 5 categorical, 4 Boolean, and 4 numerical features. The second-best run is the 8th one with 11 features selected, including 3 categorical, 4 Boolean, and 4 numerical features. The third-best run involves 11 features, with 4 categorical, 2 Boolean, and 5 numerical features. The worst-case scenario encompasses 6 features, where 3 are categorical, 1 is Boolean, and 2 are numerical. Notably, our intuition aligns with the results, indicating that having more features led to better predictions. A comparison with the worst-case scenario suggests that having more numerical and categorical features contributed to improved predictions.

For the logistic regression method, the best run is the 8th one with a total of 10 features, including 4 categorical, 4 Boolean, and 2 numerical. The second-best run is run 1 with 5 categorical, 3 Boolean, and 3 numerical features, totaling 11 variables. The third one has a total of 13 features, containing 5 categorical, 4 Boolean, and 4 numerical features. The worst-case scenario is run 6, containing 1 categorical, 4 Boolean, and 2 numerical features out of 7 selected features. Here again by having more Boolean variables in worth case the performance decreased a lot and having more categorical and numerical variables helped to having better classification results.

In KNN, the best run is the 6th, where out of 6 features collected, 1 is categorical, 2 are boolean, and 3 are numeric. The second-best is the 10th run, with 11 features in total, consisting of 3 categorical, 4 boolean, and 4

numeric features. The third-best is the first run, incorporating 6 features in total, including 3 categorical, 1 boolean, and 2 numerical features. The worst-case scenario is observed in the 7th run, where 7 features are selected, comprising 2 categorical, 4 boolean, and 1 numerical feature. In this classification algorithm also in three best results we have more categorical and numerical variables and this cause to having high performance.

In the random forest algorithm, the best run is the 7th, involving 10 features, where 3 are categorical, 4 are boolean, and 3 are numeric. The second-best is the 5th run, comprising 13 features, with 3 categorical, 5 boolean, and 5 numeric. The third-best is the 6th run, utilizing 14 features, with 5 categorical, 4 boolean, and 5 numeric. The worst-case scenario is observed in the 10th run, where 6 features are selected, including 2 categorical, 2 boolean, and 2 numeric. Again, we observe that as the number of selected Boolean variables increases and the overall number of selected features decreases, the performance declines.
In the AdaBoost algorithm, the best run is the 8th, involving 14 features, where 4 are categorical, 5 are boolean, and 5 are numeric. The second-best is the 6th run, comprising 12 features, with 4 categorical, 4 boolean, and 4 numeric. The third-best is the 7th run, utilizing 8 features, with 4 categorical, 1 boolean, and 3 numeric. The worst-case scenario is observed in the 3rd run, where 7 features are selected, including 3 categorical, 2 boolean, and 2 numeric. We can figure out in the best case we selected 14 features and have higher performance but as much the selected features decreased having boolean features caused reduction in performance.

In the MLP algorithm, the best run is the 4th, utilizing 5 features, with 1 categorical, 0 boolean, and 4 numeric. The second-best is the 7th run, incorporating 14 features, where 5 are categorical, 4 are boolean, and 5 are numeric. The third-best is the 6th run, involving 15 features, with 5 categorical, 5 boolean, and 5 numeric. The worst-case scenario is observed in the 8th run, where 10 features are selected, including 4 categorical, 2 boolean, and 4 numeric. The best case is the combination of just numerical

and categorical variables and by adding more boolean variables even by increasing the number of slected features the performance decreased.
All of these intuitions are explained, but as we mentioned earlier, this is not always true. However, in this case, the results seem to support our idea, indicating that having both categorical and numerical variables aids the classifier in better classifying the samples.

Algorithm/Run number	1	2	3	4	5	6	7	8	9	10
DecisionTree	75	56.91	**76.61**	56.12	74.58	74.92	54.86	76.59	56.41	56.5
LogisticRegression	78.96	78.69	78.94	77.75	78.18	46.16	79.23	**80.06**	58.76	76.96
KNN	72.03	48.53	51.87	69.08	52.67	**78.47**	56.46	67.02	49.35	74.05
RandomForest	75.61	59.81	78.99	58.87	79.77	79.63	**80.18**	57.94	58.68	49.38
AdaBoost	48.64	77.06	47.03	59.41	76.28	77.32	77.09	**77.47**	48.03	75.99
MLP	76.97	75.32	76.95	**80**	76.55	77.03	77.98	54.97	58.34	76.89

Table 2: illustration of mean performance.

2) Distance

When comparing the distance between the training and test sets, similar conclusions can be drawn. The presence of informative features, such as categorical and numerical variables, contributes to a more effective learning process, ultimately resulting in a smaller distance. This observation holds true in the majority of cases, aligning with our initial hypothesis and reinforcing the notion that having informative features tends to yield a more favorable outcome in reducing the discrepancy between training and test datasets.

Algorithm/Run number	1	2	3	4	5	6	7	8	9	10
DecisionTree	0.248	0.419	0.247	0.413	0.264	0.27	0.447	**0.245**	0.42	0.43
LogisticRegression	0.049	0.028	0.039	0.045	0.048	0.072	0.042	0.036	0.059	**0.024**
KNN	0.114	0.187	0.201	0.12	0.167	**0.07**	0.161	0.117	0.224	0.101
RandomForest	0.247	0.389	0.219	0.403	0.214	0.217	**0.204**	0.412	0.409	0.518
AdaBoost	0.145	0.076	0.149	0.136	0.092	0.093	**0.069**	0.089	0.157	0.081
MLP	0.242	0.178	0.233	**0.041**	0.219	0.235	0.225	0.422	0.255	0.231

Table 3: illustration of distance between train and test set.

3) Speed:

When comparing the speed of algorithms, the approach is somewhat different. Some algorithms may terminate more quickly when there are more boolean variables, resulting in lower running times. Conversely, a higher number of numerical and categorical variables may lead to increased computational effort in classifying samples, consequently leading to longer running times. Generally, an increase in the number of selected features tends to correspond to an increase in running time. For instance, in decision tree algorithms, the best-performing runs in terms of performance are 3, 8, and 1, while the worst-performing run is 7. However, in terms of speed, the best runs are 7, 4, and 8, with the worst case being run 7. Similarly, in logistic regression, the top-performing runs are 8, 1, and 3, while the worst is run 6. Yet, in terms of speed, the best runs are 6, 2, and 1, and the worst case is run 3. These patterns in speed are distinct from those observed in performance and distance evaluations, reflecting the varying nature of columns and how they contribute to the informativeness and separability for classification.

Algorithm/Run number	1	2	3	4	5	6	7	8	9	10
DecisionTree	0.068	0.075	0.069	0.058	0.073	0.07	**0.055**	0.059	0.065	0.064
LogisticRegression	0.15	0.109	0.293	0.257	0.267	**0.107**	0.267	0.274	0.184	0.202
KNN	0.654	0.19	0.193	0.212	0.19	0.187	**0.184**	0.222	0.191	0.193
RandomForest	1.943	3.364	**1.917**	2.107	2.821	2.224	2.011	2.01	3.227	2.061
AdaBoost	**1.006**	1.182	1.554	1.5	1.301	1.311	1.165	1.473	1.514	1.103
MLP	13.201	17.539	12.974	12.491	13.455	12.258	14.053	**10.649**	13.026	13.982

Table 4: illustration of speed of each algorithm.

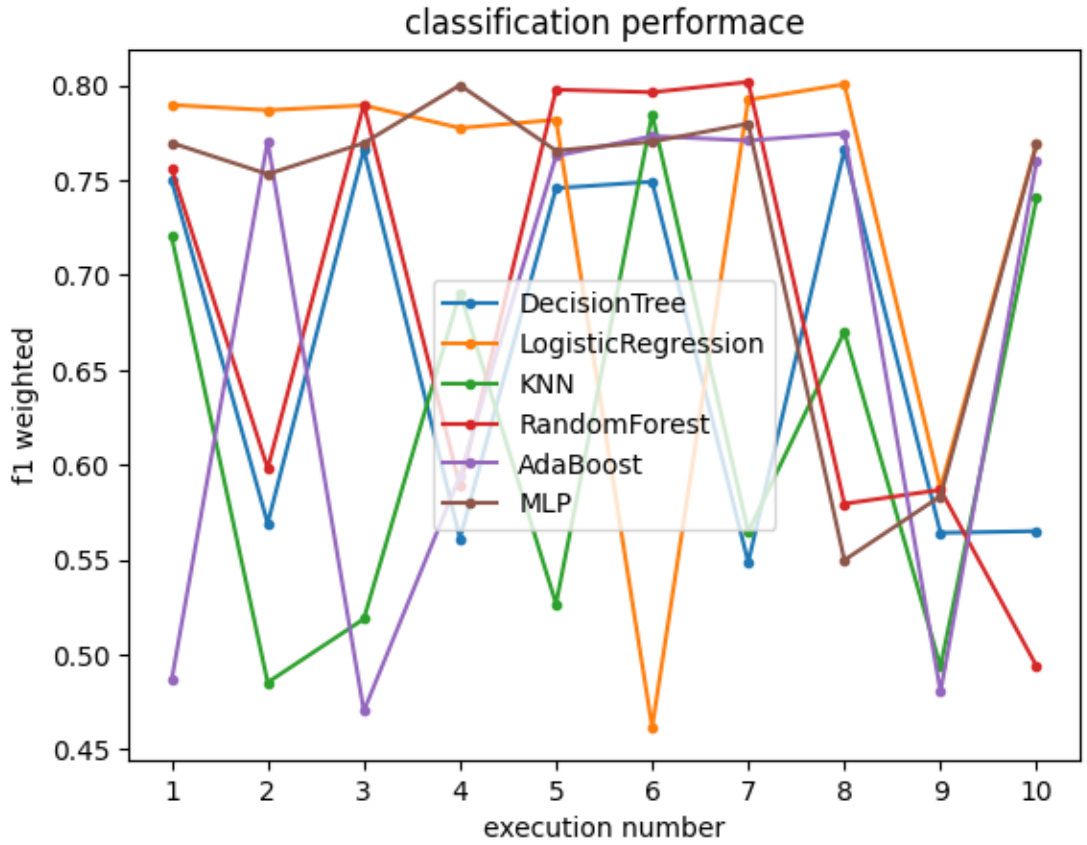


Image 1: illustration of mean performance for each different algorithm in different execution with different set of variables.

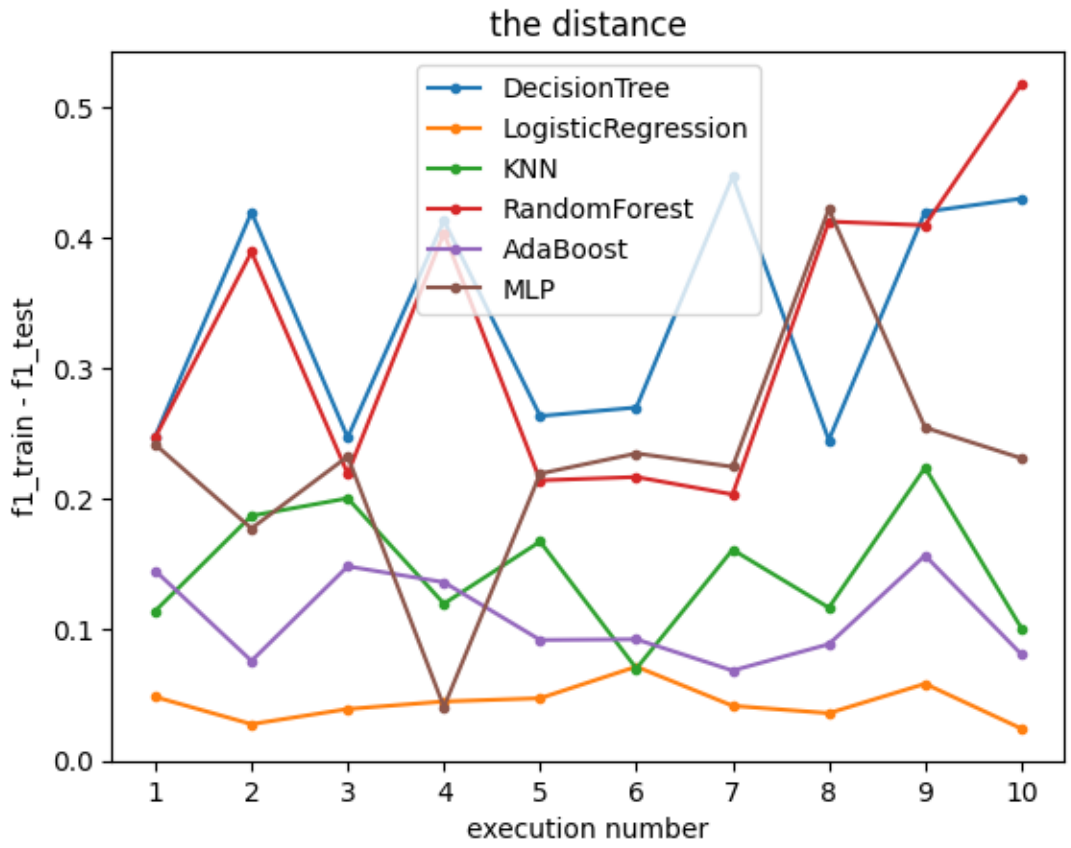


Image 2: illustration of distance between train and test for each different algorithm in different execution with different set of variables.

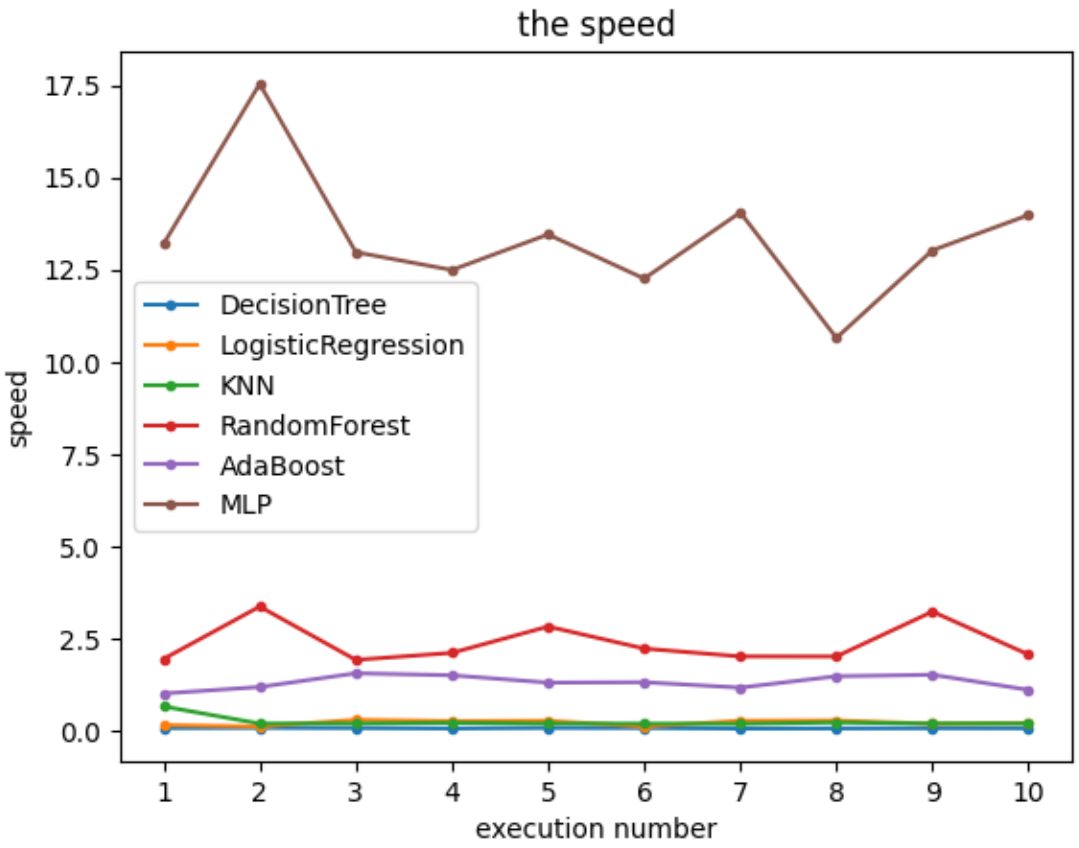


Image 3: illustration of speed for each different algorithm in different execution with different set of variables.

3.2. Duplication:

For duplication, we have collected a total of 120 experiences. For each similarity level, we create separate tables to compare the results in both cases – with and without duplication in different duplication percentages.

1) Performance evaluation:

In the domain of performance evaluation, a general expectation is that the accuracy of predictions should improve after removing duplications, particularly when the labels of duplicated data differ from the real data. This expectation aligns with our experimental setup, where the targets of duplicated data derived completely randomly.

in the presented results, showed in tables 5 and 6 it is obvious that performance improved in most of the cases after the removal of duplications.

This phenomenon can be attributed to the unique nature of our non-exact duplicated data. The ML model struggles to discover patterns when faced with diverse data that labeled different.

In our duplication setup, each duplicated dataset exhibits varying levels of similarity with the real data, 40% or 80%. This implies that, for instance, in a dataset with 10 columns, between duplicated data and real data, 6 or 2 columns have different values. This deliberate variation poses a challenge for the ML model, preventing its ability to robustly extract patterns and accurately classify all samples.

By systematically removing these duplicates, we provide the model with a more consistent and representative dataset. In essence, the model becomes liberated from the confounding effects introduced by non-exact duplications.

Other kind of conclusion is when we compare similarity levels, when similarity level is 40%, generally we can make better estimation than 80% before removing duplication. The rational behind this can be, as far duplicated data are different with real data in more columns and thus cause less confounding for the model because the model sees these samples as a new sample and the ML model tries to discover and learn patterns for these duplicated data as well. Finally, the model is able to make a good prediction on these duplicated samples as well. In tables 5 and 6 the yellow-highlighted cells indicate the comparison between 40% and 80%. In 36 out of 60 cells before cleaning, the performance at 40% similarity is higher than that at 80% similarity. But for the case of after removing duplication we can't figure out such an insight because the duplication removing algorithm removes records randomly and we don't know the real one removed or the duplicated one.

Other aspect is that when the duplication percentage increases, the performance falls. The main reason is that model sees more inaccurate data and can't model the data and make an accurate prediction on it.

algorithm	Data cleaning status	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%

DecisionTree	Before	79.99	77.97	74.92	73.46	68.84	71.67	68.57	66.57	65.3	63.44
	After	**82.7**	**80.62**	**79.12**	**77.86**	**76.96**	**74.13**	**72.34**	**73.24**	**67.93**	**69.02**
LogisticRegression	Before	83.4	83.34	81.41	80.89	78.06	76.56	75.86	75.26	74.64	73.78
	After	**84.15**	**83.67**	**81.45**	**81.57**	**80.95**	**79.25**	**77.4**	**76.75**	**77.25**	**76.88**
KNN	Before	**87.02**	83.8	81.81	81.71	79.29	78.75	76.39	73.75	73.5	71.47
	After	86.95	**86.82**	**84.42**	**84.85**	**81.26**	**79.32**	**80.2**	**77.91**	**77.4**	**74.9**
RandomForest	Before	**86.27**	84.79	82.15	83.48	80.56	79.29	76.9	73.16	75.48	72.98
	After	88.16	**86.22**	**87.06**	**85.92**	**84.12**	**83.15**	**81.21**	**81.73**	**80.02**	**77.83**
AdaBoost	Before	82.57	81.26	80.19	77.83	76.89	75.69	74.65	73.87	73.1	71.98
	After	**84.54**	**81.89**	**81.77**	**81.13**	**80.33**	**76.65**	**77.2**	**78.15**	**75.26**	**75.78**
MLP	Before	88.64	86.32	84.61	84.17	82.1	79.06	79.81	76.17	77.66	75.8
	After	**90.67**	**88.62**	**87.58**	**86.67**	**83.66**	**83.46**	**80.57**	**81.44**	**81.49**	**78.01**

Table 5: The mean performance at the 40% similarity level is represented in the table. The cells in bold highlight higher performance in the comparison between before and after cleaning, while the yellow-highlighted cells represent the comparison between 40% and 80% similarity levels.

algorithm	Data cleaning status	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
DecisionTree	Before	81.11	78.41	75.83	72.31	70.41	69.09	67.36	63.34	63.5	60.55
	After	**83.75**	**81.04**	**82.82**	**77.33**	**76.14**	**75.02**	**76.22**	**72.75**	**72.85**	**69.61**
LogisticRegression	Before	83.93	83.37	79.93	78.55	77.05	74.9	76.32	77.01	72.76	72.34
	After	**84.66**	**84.08**	**81.72**	**81.19**	**81.08**	**79.13**	**79.03**	**78.61**	**75.16**	**73.91**
KNN	Before	85.92	82.38	81.96	79.94	75.92	77.84	75.31	73.94	75.01	73.17
	After	**86.5**	**86.8**	**84.48**	**83.37**	**80.79**	**79.97**	**78.9**	**77.9**	**78.28**	**78.58**
RandomForest	Before	85.29	83.77	80.51	79.89	77.65	72.49	72.46	72.22	69.74	68.85

	After	**87.74**	**87.13**	**85.79**	**85.81**	**85.08**	**81.05**	**82.32**	**80.54**	**79.53**	**76.44**
AdaBoost	Before	**84.87**	81.36	78.9	78.74	77.62	74.85	75.09	74.23	70.61	72.6
	After	84.16	**82.21**	**82.58**	**82.87**	**80.49**	**79.46**	**74.77**	**75.9**	**76.35**	**74.31**
MLP	Before	88.1	86.85	84.7	83.14	82.56	80.86	79.12	76.38	76.28	75.44
	After	**91.37**	**89.16**	**88.78**	**87.28**	**84.75**	**83.83**	**82.21**	**81.53**	**79.12**	**78.12**

Table 6: The mean performance at the 80% similarity level is represented in the table. The cells in bold highlight higher performance in the comparison between before and after cleaning, while the yellow-highlighted cells represent the comparison between 40% and 80% similarity levels.

2) Distance:

The distance in F1-score between the train and test datasets serves as a measure of model generalization. It's evident that with cleaner data, this distance is expected to be smaller compared to situations involving duplicated data. The primary reason behind this observation lies in the training phase. When dealing with duplicated rows, the model struggles to fit entirely on the training data, resulting in a lower F1-score during inference. Consequently, the difference is more when compared to scenarios with clean data. Clean data leads to a more similar distribution between the train and test datasets, allowing the ML model to learn patterns more effectively and make accurate predictions during both training and inference. This observation is particularly evident when the similarity level is set to 80% in most cases. However, for the 40% similarity level, the situation is a bit different. In many instances, before data cleaning, there is less distance. The primary reason for this phenomenon could be that when the similarity is low, duplicated data can be perceived as new samples. The machine learning model then attempts to capture patterns for these samples and learns them effectively and has less confusion. Of course, the extent of success depends on the nature of the classification method being employed.

Another observation from the results is that, in most cases, the distance is smaller when the similarity level is 40%, before the data cleaning, compared to

the scenario with a similarity level of 80%. This phenomenon can be attributed to model confusion during the training phase. As the similarity between duplicated data and real data increases, accompanied by varying labels, the model encounters ambiguity, making accurate predictions challenging. This ambiguity persists even after removing duplications, as distinguishing between real and duplicated data becomes uncertain. Consequently, the model's performance is more affected when dealing with a higher similarity level of 80%. As in table 7 and 8 highlighted in yellow, in 42 cells out of 60 before the data cleaning in similarity level of 40% the distances are smaller, but in the case of the after data cleaning we can't evident such a thing as far cleaning is done completely randomly.

Other evidence is that as much the duplication percentage increases of course the distance increase and this because of seeing more inaccurate and duplicated samples by the model and problem of training properly on these samples.

algorithm	Data cleaning status	5%	10%	15%	20%	25%	30%	35%	40%	45%	50
DecisionTree	Before	0.189	0.215	0.233	0.269	0.307	0.278	0.316	0.332	0.352	0.3
	After	**0.166**	**0.191**	**0.207**	**0.205**	**0.238**	**0.266**	**0.266**	**0.302**	**0.308**	**0.3**
LogisticRegression	Before	0.012	**-0.002**	**0.005**	0.011	**-0.005**	**-0.002**	**0.016**	**0.012**	0.019	**0.0**
	After	**0.009**	0.003	0.02	**0.01**	0.022	0.008	0.041	0.016	**0.012**	0.0
KNN	Before	**0.04**	**0.046**	0.053	0.041	0.07	**0.052**	**0.053**	**0.062**	0.069	0.0
	After	0.046	0.048	**0.047**	**0.034**	**0.051**	0.061	0.064	0.078	**0.067**	**0.0**
RandomForest	Before	0.136	0.154	0.175	0.162	0.221	0.217	0.238	0.264	0.236	0.2
	After	**0.113**	**0.143**	**0.144**	**0.136**	**0.155**	**0.169**	**0.183**	**0.185**	**0.197**	**0.2**
AdaBoost	Before	**0.064**	**0.068**	**0.073**	**0.066**	**0.071**	**0.061**	**0.068**	**0.076**	**0.084**	**0.0**
	After	0.076	0.072	0.087	0.068	0.075	0.08	0.069	0.084	0.099	0.0
MLP	Before	**0.044**	0.053	**0.031**	**0.041**	**0.059**	**0.041**	**0.056**	**0.068**	**0.036**	**0.0**

	After	0.054	**0.043**	0.056	0.065	0.067	0.055	0.072	0.083	0.05	0.(

Table 7: Illustrates the distance between the performance of the train and test datasets for a 40% similarity level. The bolded cells indicate the comparison between before and after data cleaning. The yellow-highlighted cells represent the comparison between 40% and 80% similarity levels.

algorithm	Data cleaning status	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
DecisionTree	Before	0.191	0.22	0.25	0.266	0.278	0.321	0.325	0.355	0.375	0.397
	After	**0.021**	**-0.001**	**0.011**	**0.029**	**0.014**	**0.011**	**0.021**	**0.017**	**0.011**	**0.016**
LogisticRegression	Before	0.021	**-0.001**	**0.011**	0.029	**0.014**	0.011	0.021	0.017	**0.011**	0.016
	After	**0.007**	0.016	0.03	**0.002**	0.015	**0**	**0.002**	**0.002**	0.012	**0.015**
KNN	Before	**0.052**	**0.043**	**0.061**	0.066	**0.06**	**0.056**	**0.064**	0.074	0.062	0.078
	After	0.057	0.048	0.062	**0.048**	0.065	0.066	0.066	**0.064**	**0.06**	**0.062**
RandomForest	Before	0.146	0.158	0.193	0.192	0.223	0.267	0.272	0.269	0.292	0.319
	After	**0.119**	**0.126**	**0.147**	**0.136**	**0.156**	**0.194**	**0.18**	**0.186**	**0.202**	**0.23**
AdaBoost	Before	**0.072**	0.08	0.07	**0.05**	**0.061**	**0.058**	0.079	**0.081**	0.089	**0.067**
	After	0.087	**0.079**	**0.059**	0.089	0.079	0.11	**0.072**	0.087	**0.067**	0.089
MLP	Before	0.044	**0.054**	**0.031**	**0.045**	**0.055**	**0.062**	0.074	**0.058**	**0.036**	**0.063**
	After	**0.031**	0.057	0.052	0.057	0.059	0.07	**0.061**	0.078	0.062	0.092

Table 8: Illustrates the distance between the performance of the train and test datasets for a 80% similarity level. The bolded cells indicate the comparison between before and after data cleaning. The yellow-highlighted cells represent the comparison between 40% and 80% similarity levels.

3) Speed:

It is evident that having duplicated data provides us with a larger dataset for evaluation, allowing us to dedicate more time to assessing the entire dataset. Consequently, in most of the cases before data cleaning, the processing speed is lower than in the case after data cleaning. However, the outcome is generally depending on the characteristics of the model and its hyperparameters. Machine learning-based models necessitate convergence to an optimal

minimum, and, in some scenarios, they converge faster even before data cleaning, leading to the termination of the algorithm at an earlier stage.

Additionally, increasing the number of duplicated samples inherently extends the training phase, requiring more time for completion. Thus, as much the duplicated samples growth the model require to spend more time to model the input data. In the collected results, there is no strong evidence for comparing similarity levels, as the running time depends on various factors such as caching data by RAM, which speeds up the process, the status of the CPU at that moment, or the nature of the machine learning algorithm that sometimes gets stuck in a local minimum and terminates sooner. This type of measurement should be compared with distance and mean performance to find a better trade-off between the accuracy of the model and running time. Solely considering speed is not very informative for us.

algorithm	Data cleaning status	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
DecisionTree	Before	0.08	0.086	0.089	0.111	0.107	0.11	0.106	0.118	0.133	0.452
	After	**0.072**	**0.077**	**0.079**	**0.078**	**0.08**	**0.079**	**0.089**	**0.083**	**0.079**	**0.226**
LogisticRegression	Before	**0.071**	0.216	**0.04**	**0.041**	0.044	0.046	0.042	**0.041**	0.063	0.039
	After	0.163	**0.063**	0.046	0.085	**0.04**	**0.036**	**0.036**	0.043	**0.035**	**0.036**
KNN	Before	0.158	0.095	0.1	0.103	0.108	0.112	0.116	0.133	0.137	0.143
	After	**0.09**	**0.091**	**0.092**	**0.091**	**0.091**	**0.092**	**0.093**	**0.09**	**0.09**	**0.091**
RandomForest	Before	1.844	1.928	2.051	2.141	2.459	4.399	2.506	2.676	2.658	2.823
	After	**1.743**	**1.764**	**1.806**	**1.808**	**1.867**	**1.852**	**1.874**	**1.899**	**1.89**	**1.893**
AdaBoost	Before	0.915	0.911	0.921	0.958	0.982	0.99	1.017	1.046	1.074	1.081
	After	**0.88**	**0.88**	**0.875**	**0.858**	**0.855**	**0.854**	**0.873**	**0.859**	**0.862**	**0.879**
MLP	Before	**6.532**	**7.386**	**9.526**	11.652	8.745	8.6	8.568	7.262	14.689	17.256
	After	14.04	8.212	9.977	**10.118**	**6.616**	**6.616**	**5.671**	**6.754**	**5.951**	**6.281**

Table 9: Shows the speed of each algorithm when similarity is 40%. Note that speed here means the running time, If the speed is high, it means running time is high. Finally, we need to select the lower values because takes less time. Thus, bolded values are the one running time is less.

algorithm	Data cleaning status	5%	10%	15%	20%	25%	30%	35%	40%	45%	50%
DecisionTree	Before	0.251	0.083	0.092	0.088	0.102	0.164	0.101	0.169	0.107	0.118
	After	**0.1**	**0.076**	**0.078**	**0.081**	**0.078**	**0.081**	**0.079**	**0.078**	**0.087**	**0.086**
LogisticRegression	Before	**0.034**	**0.028**	**0.03**	**0.036**	0.042	**0.033**	**0.036**	0.04	0.055	0.045
	After	0.036	0.037	0.033	0.043	**0.035**	0.034	0.037	**0.037**	**0.034**	**0.032**
KNN	Before	0.091	0.095	0.1	0.104	0.108	0.113	0.116	0.133	0.138	0.142
	After	**0.09**	**0.091**	**0.092**	**0.091**	**0.092**	**0.09**	**0.091**	**0.094**	**0.092**	**0.091**
RandomForest	Before	1.834	1.906	2.085	2.13	2.287	2.343	2.455	2.572	2.672	2.777
	After	**1.715**	**1.732**	**1.802**	**1.832**	**1.814**	**1.838**	**1.87**	**1.897**	**1.903**	**1.968**
AdaBoost	Before	0.922	0.903	0.912	0.939	0.957	0.998	1.098	1.12	1.04	1.065
	After	**0.866**	**0.859**	**0.854**	**0.854**	**0.892**	**0.877**	**0.885**	**0.874**	**0.857**	**0.856**
MLP	Before	**6.477**	**6.051**	8.484	6.29	6.637	6.864	**6.863**	9.988	7.582	9.521
	After	7.299	10.325	**5.654**	**5.84**	**5.449**	**5.633**	9.387	**5.769**	**5.525**	**6.4**

Table 10: Shows the speed of each algorithm when similarity is 80%. Note that speed here means the running time, If the speed is high, it means running time is high. Finally, we need to select the lower values because takes less time. Thus, bolded values are the one running time is less.

Finally, we plot all the results to compare both cases before and after data cleaning at different similarity levels. As observed, before data cleaning, we can draw conclusions aligning with the previously described ideas. However, after data cleaning, drawing conclusions becomes more challenging, as we aim to have clean data. Overall, after data cleaning, performance is higher in most cases, and the distance is lower.

Regarding speed, it can be stated that having duplicated data results in higher running times, implying slower speed. Nevertheless, as mentioned earlier, this is contingent on the nature of the algorithm and how it terminates in running time.

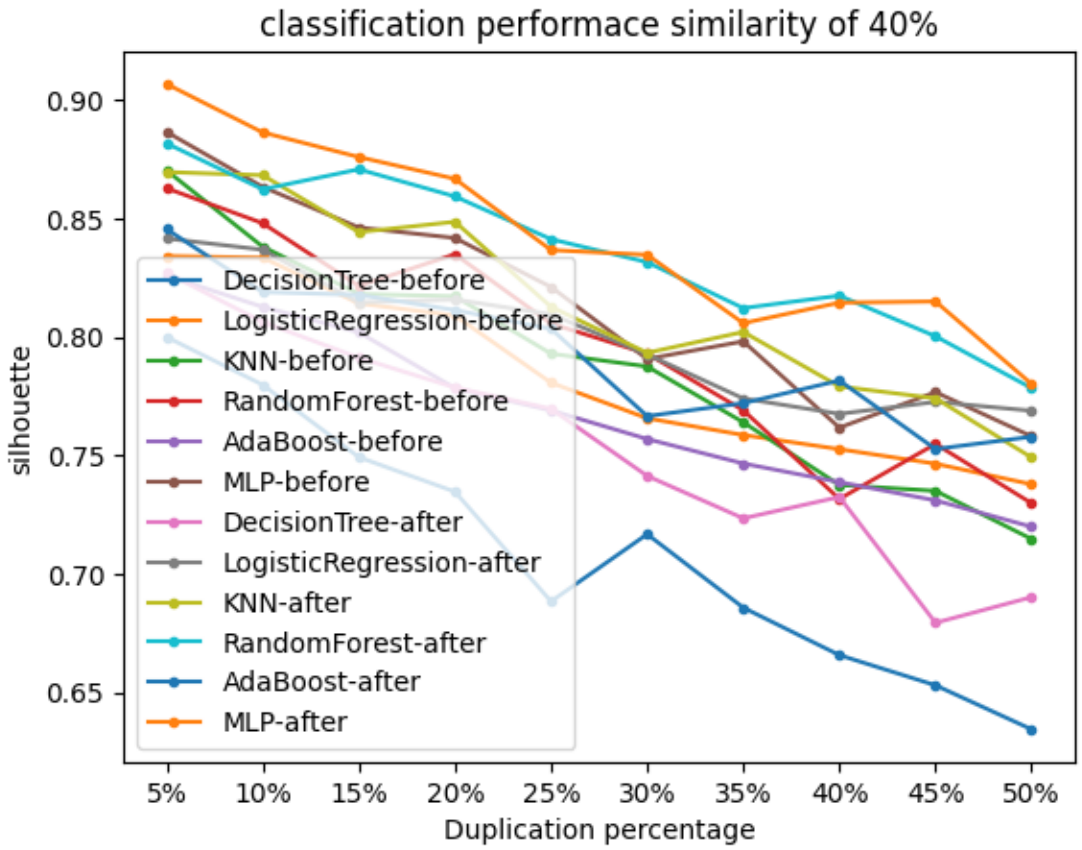


Image 4: mean performance before and after of data cleaning for all classification algorithms.

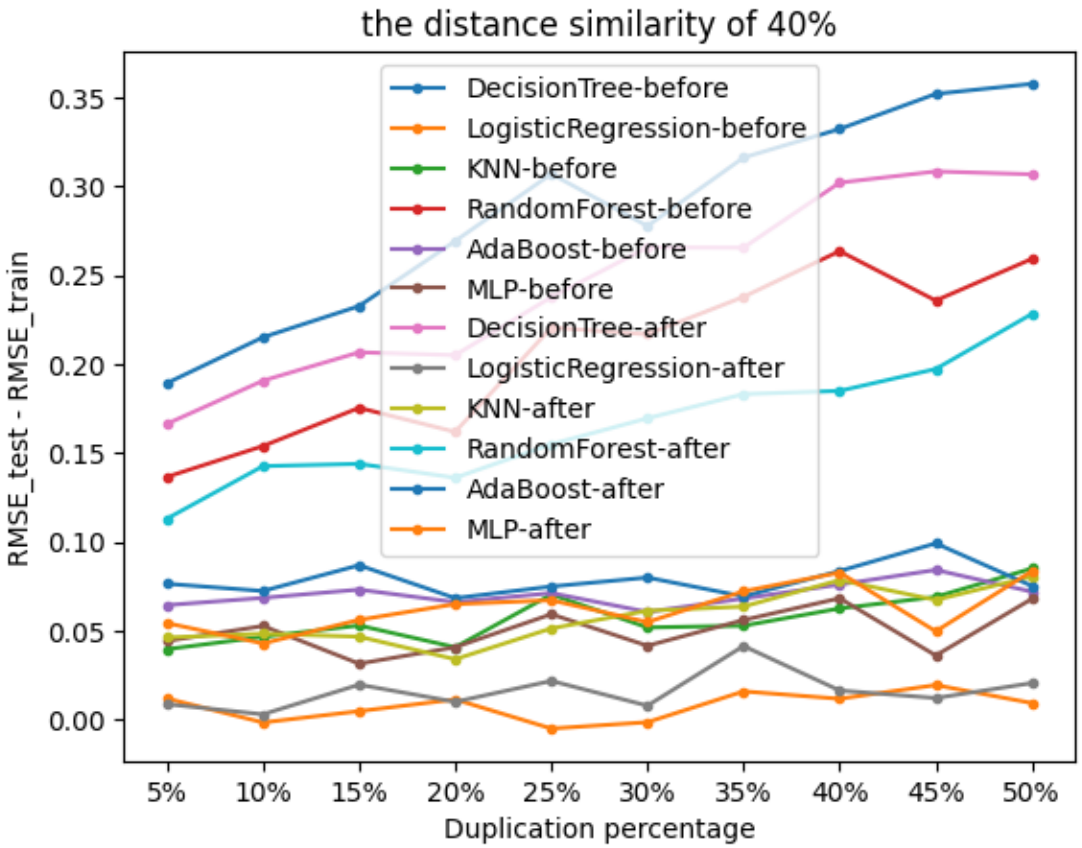


Image 5: Distance before and after of data cleaning for all classification algorithms.

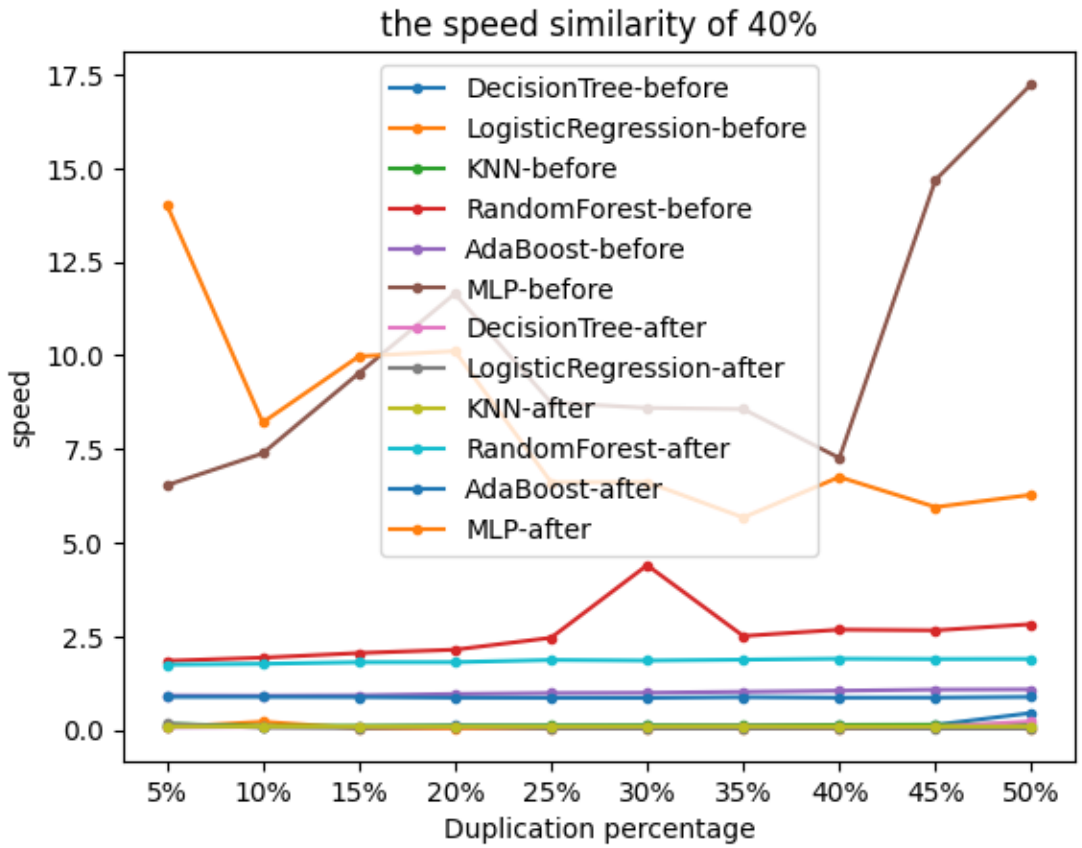


Image 6: speed of the classification methods before and after of data cleaning