Introduction to Machine Learning
Work 3: Lazy learning

Marc Ros Martí
Estel Ferrer Torres
Pol Roca Llaberia

December 21, 2020

Master in Artificial Intelligence

Universitat de Barcelona

Contents

1 Introduction

In this exercise, we apply a lazy learning method for a classification tasks over two different data sets (introduced in section 2).

Lazy learning, unlike eager learning (that generalize the training data before receiving a query), delays generalization of the training data until a query is made to the system.
This method has some advantage. The most important one would be that, since we are not learning any model with the training set, it deals successfully with changes in the problem domain (the algorithm used every time the whole data set for classifying a query).

One of the most well known lazy learning methods is the k-Nearest Neighbour (KNN), introduced in section 3.

In this work we do a comparative study between different variations of KNN algorithm: k value, distance metric, policy and feature weighting (all of them explained in subsection 3.1).

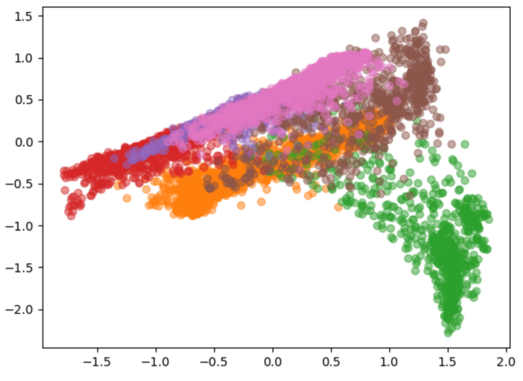
Finally, in section 4 by fixing the KNN parameters that perform the best, we introduce and compare the results of different sample reduction techniques.

2 Data sets description

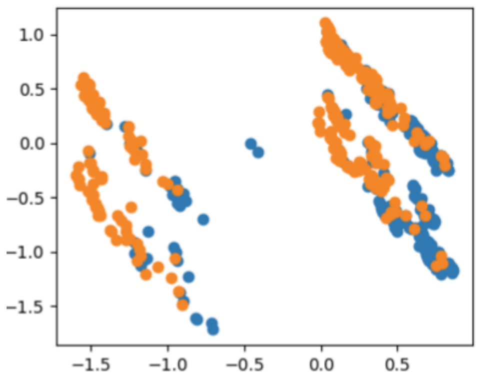
Two different data sets [1] are used to test our implementations and extract conclusions. The main characteristics of both data set are presented in table 1, while figure 1 shows some insight on the distribution of data in each data set by using PCA.

Domain	Cases	Num.	Nom	Cla.	Dev.Cla.	Maj.Cla.	Min.Cla	MV
SatImage	6,435	36	-	6	6.19 %	23.82 %	9.73 %	-
Credit-A	690	6	9	2	5.51 %	55.51 %	44.49 %	0.65 %

Table 1: Characteristics of each data set.



(a) SatImage



(b) Credit-A

Figure 1: PCA of the data sets employed in our experiments.

On the one hand, SatImage is a medium size data set which contains 6,435 numerical instances with 36 features each one. This data set does not need a pre-procesing since all values are numeric and compressed between zero and one.

On the other hand, Credit-A is a small data set that has a mixture of both numerical and categorical features, being 6 and 9 the respective amounts, while 690 is the total amount of instances and 2 the number of classes. Before carrying out the experiments, this data set needed some common pre-processing adjustments. Without going to much into detail, since we do not want to focus on that part in this report, we performed MinMax scaling on the numerical features and we one-hot encoded the categorical ones.

The 90% of both data sets will be used for training while the remaining 10% will be used for testing. However, notice that, without considering this 10% for the training, we risk losing important information in data set, which in turn increases error induced by bias. To overcome this possible problem, the data sets are provided in a 10-fold cross-validation sets: 10 different splits of the whole data set, containing different combinations of train and test samples but maintaining the 90% and 10% ratios. Therefore, 10 accuracy values are obtained for each data set and each KNN model. The final accuracy will be the average accuracy between all folds.

3 K-Nearest Neighbor algorithm

K-Nearest Neighbor algorithm (KNN) is a simple supervised learning algorithm, belonging to lazy learning family methods, that works on the basic assumption that data points of similar classes are closer to each other. Then, KNN assign the output of a new query to be the same as the most similar known case in the training set. Notice that 100% of the computational time is spend for consultation while 0% for training, delaying generalization until a query is made to the system. This method, requires a large space to store the entire training set (expensive in storage and resources) but, in contrast, it deals successfully with changes in the problem domain (changes over time does not affect the learning algorithm).

We can play with different parameters of the algorithm: the distance metric to be used, the policy to assign the class of a query when considering not only one but k nearest neighbours, the number of nearest neighbours to consider (k) and feature weighting (uniform or not).

All of this parameters are described in the following section.

3.1 Description of the parameters

We will concider three different distance metric, four different values of k, three different policies for deciding the solution of a query and three different ways to compute feature weighting. This ends up with 108 different KNN models.

- Distance metric. This parameter plays one of the most important roles since it is the base for deciding which training samples are closer to the query. We will compare the results between the three most well-known distance metrics:
  - Manhattan distance (1-norm): distance between two points measured along axes at right angles. It can also be refereed as order one Minkowski Metric.
  - Euclidean distance (2-norm): physical length of a line segment between the two points. It can also be refereed as order two Minkowski Metric.
  - Chebyshev distance: greatest distance along all coordinate dimension (Minkowski Metric of order ∞).

- k value. It is the number of nearest neighbours to a new query to be considered for classifying the given query. Four different values of k will be considered in the evaluation of the KNN model: 1, 3, 5 and 7.
- Policy. The policy is the way in which k nearest neighbours can be used to determine the class of the new query. We will obtain the results with three different policies:
  - Majority Class (simplest method): it assigns the majority class among the k neatest neighbours to be the class of the new query. In case of ties, we simply choose the first in the list (which will be the class of the closest instance between the tied instances).
  - Inverse Distance Weighted votes: it gives more importance to the class of those instances that are closer to the query in the search space. This method assigns more weight to the nearer neighbours by computing a vote for each possible class defined as the sum of the inverse distance of each nearest neighbour belonging to this particular class. The winning class is the one with the highest vote. In case of ties, we simply choose the first in the list (which will be the class of the closest instance between the tied instances).
  - Sheppard's work: similar to the Inverse Distance Weighted, but in this case, the vote of each class is defined as the sum of the inverse exponential of the distance. In case of ties, we simply choose the first in the list (which will be the class of the closest instance between the tied instances).
- Feature weighting. Probably not all the features are equally important in a data set. If this is the case, giving the same importance to all features can decreases the performance of the model. Therefore, is interesting to test the model with different feature weighting algorithms.
  - Uniform weights: simplest case, which assumes that all the feature have the same importance and assigns a unitary weight to each feature.
  - ReliefF: we apply the algorithm [2] to the train set and then modify all the data set with the obtained weights normalized between 0 and 1.
  - Mutual information: we apply the algorithm [3] to the train set and then modify the entire data set.

3.2 KNN performance

In this section we present the best accuracy results obtained with each data set (the average of correctly classified queries). Note that the accuracy used here is the average of the results per fold.

We compute the average accuracy between all models that used the same isolated parameter in order to extract information about how well a particular value for a particular parameter is working on average. This way, for a specific k within the 4 possible values, we compute the average accuracy between the 27 (108/4) models that used this particular k value. In addition, we also provide the efficiency for each model, representing the average problem-solving time.

3.2.1 Results for SatImage

Table 2 shows both accuracy and efficiency results for SatImage data set.

We can see that between all possible k values, the one that performs better is $k = 5$ since it provides the highest accuracy and the highest efficiency when averaging both results for all models with the same k value. However, notice that, even using a single nearest neighbour (in which case no policy is considered) the accuracy decreases less than 1%. From this we could say that no much noisy samples are present in the data set.

For distance metric (or similarity) KNN algorithm performs better in average when using the Manhattan distance (1-norm), followed by Euclidean similarity metric, with slightly lower accuracy value. Recall that Chebyshev is the worst option since it is the one that requires the higher computational time and at the same time the one providing considerably poorer results.

We do not observe differences in the performance of the model in terms of policy. Knowing that a voting schemes (like inverse or shepperd) are very useful when handling with noisy data sets, we can remark that SatImage is a noiseless data set.

Finally, for the weighting parameter, the highest accuracy is found with uniform weights (simplest case), which of course is the one that requires less computational time since no weights need to be computed. From this result we can say that more or less all attributes are equally important for classifying a new query.

Parameter	Value	Efficiency (ms)	Accuracy (%)
k	1	498	88.90
k	3	506	89.55
k	5	**489**	**89.67**
k	7	518	89.58
distance	1-norm	**439**	**90.70**
distance	2-norm	440	90.08
distance	chebyshev	630	87.49
policy	majority	**503**	89.40
policy	inverse	505	**89.44**
policy	sheppard	**503**	**89.44**
weighting	uniform	**495**	**90.14**
weighting	mutual_info	506	90.03
weighting	relief	508	88.10

Table 2: Average efficiency and accuracy metrics for each possible value of each parameter.

Moreover, table 3 present the best model in absolute, i.e., the one (within 108 different models) that performs better in terms of accuracy. We see that the maximum number of correctly classified queries obtained with KNN and Satiamge cross-vaidation (averaged for all the folds) is higher than 91 % when considering 1-norm similarity metric, Sheppard policy and no weighting.

k	Distance	Policy	Weighting	Efficiency	Accuracy
5	1-norm	Sheppard	uniform	419 ms	91.30 %

Table 3: Best model for SatImage data set

3.2.2 Results for Credit-A

Table 4 shows the summary of best averaged results by parameter values of KNN evaluated on Credit-A.

In this data set higher values of k show better results in general. This behavior could be because of noisy instances in the data set, but we do not think this is the reason to that because then there would not be much difference between $k = 5$ and $k = 7$. A reason to why higher k have better results could be because in this data set data of both classes is somewhat mixed, as we can see in figure 1(b).

Therefore, taking into account more nearest neighbors could be more useful in order to decide to which class a given instance belongs to.

For the distance measure parameter, we can see a clear difference (of about 4% in accuracy) between the 1-norm and 2-norm and the Chebyshev distance. The Euclidean distance seems to perform the best in this case but it is basically on-par with the Manhattan distance. Although the Chebyshev distance produces the worst results in terms of accuracy, here, it is the one achieving the lowest compute time among the three.

Besides, this data set is not big enough to take strong conclusions on the efficiency, and in fact, we will not make any assumption based on it because of that reason and because the differences in the results are very tight.

Regarding the voting policies, similarily to the other data set, we cannot observe big differences in the accuracy. Having said that, Sheppard policy yields the best results while the majority policy is just behind.

In the case of feature weighting is where we can see more different results with each method applied. The first naive method, which is simply not applying any weights, yields an average 80.28%. This is the lowest average accuracy on the table, and it means that generally any model with a combination of parameters that does not include feature weighting is deemed to perform worse than others. The second best accuracy considering this parameter is mutual information, with an increase of roughly a 3.5% in accuracy that results to an total average of 83.82%. However, results go even further with the third weighting method, ReliefF, with which models managed to achieve a remarkable 85.54% in accuracy on average. This being the highest accuracy score on the table leads us to firmly assume that, in some cases, feature weighting can be very helpful to improve the performance of a model. Despite, we also think that this phenomena strongly depends on the characteristics of the data set.

Parameter	Value	Efficiency (ms)	Accuracy (%)
k	1	**7.22**	81.30
k	3	7.33	83.34
k	5	7.38	83.83
k	7	7.41	**84.39**
distance	1-norm	7.36	84.47
distance	2-norm	7.53	**84.59**
distance	chebyshev	**7.12**	80.59
policy	majority	7.45	83.27
policy	inverse	7.29	82.84
policy	sheppard	**7.26**	**83.53**
weighting	uniform	**7.22**	80.28
weighting	mutual_info	7.38	83.82
weighting	relief	7.41	**85.54**

Table 4: Average efficiency and accuracy metrics for each possible value of each parameter.

Finally, table 5 shows the model that performed the best among all on Credit-A. This model considers 5 nearest neighbors, it uses the Chebyshev distance, the Sheppard policy and it obtains the best results after applying ReliefF weights to the data set. It is interesting to see that, even though the 2-norm yielded better results in general, the best model does not use it, and it uses the Chebyshev distance which is the worst in general. A similar case happens with the k, since $k = 5$ is used instead of $k = 7$, which is the best. That could be because each parameter does not affect the model separately, but rather a combined behavior between the parameters is observed, and as we can see, some parameters have more synergy when employed together than with some others.

k	Distance	Policy	Weighting	Efficiency	Accuracy
5	Chebyshev	Sheppard	ReliefF	7.35 ms	87.97 %

Table 5: Best model for Credit-A data set

3.3 Statistical Analysis

A statistical analysis or hypothesis test [4] is a methodology that allows to verify if the obtained experimental results are meaningful or not. In other words, checks the odds that the results have happened by chance. For that, it is necessary a statement from which the comparison has to be performed: the null hypothesis. In this concrete work, it is going to be used the null hypothesis that two related paired samples (of results -accuracy-) come from the same distribution. For that, several options have been considered, as explained in the following subsection.

We must mention that for the sake of the experiments (and the practical exercise), since we only have two data sets at our disposal and most of the methods for performing statistical analysis require more than 5 samples per model, we will consider each of the results on the 10 folds from each data set as a separate sample.

3.3.1 Selected Statistical Tests

To check the previously stated null hypothesis several tests can be implemented. On the one hand, if the evaluation is thought to be pair-wise, T-Test and Wilcoxon test have been considered. On the other hand, for evaluating multiple classifiers (i.e. more than two) ANOVA and Friedman tests have been studied.

T-Test and ANOVA are parametric hypothesis tests. This kind of tests require a particular environment to behave properly:

- Assume commensurability among differences of performance.
- They assume a normal distribution of the data.
- If not normal distribution, they require a big number of datasets (over 30) to work properly.

On the contrary, Wilcoxon and Friedman are non-parametric hypothesis tests, which are more suitable when the aforementioned conditions are not fulfilled.

In this particular work, the obtained performance data is not under a normal distribution and the number of available datasets is clearly below 30. Consequently, Wilcoxon and Friedman have been the selected tests to perform the pair-wise and multiple statistical analysis, respectively.

3.3.2 Methodology

In this first statistical analysis, the goal has been to determine for each of the parameters in isolation, i.e. fixing all the rest of parameters into a default configuration, which of them is performing better (in terms of accuracy) and afterwards determine if there is a statistical significant difference in their performance. Thus, per each isolated parameter, it could be determined whether the best performer is statistically significantly better than the other options. Finally, the winner of each parameter-based analysis is statistically compared with the overall best model, i.e. the one providing higher accuracy without fixing any of the parameters.

As stated, the first step has consisted in isolate each of the parameters in the model (k, distance, policy and weighting). For that, it has been proposed the following default configuration:

k	Distance	Policy	Weighting
5	2-norm	Majority	Uniform

Table 6: Default configuration to analyse isolated parameters

Finally, so as to evaluate the Null Hypothesis of the test, the obtained p-values are going to be considered. There is not a single rule to decide the p-value from which a null hypothesis has to be confirmed or rejected, as it may be domain dependent. Nevertheless, the typical value to reject the hypothesis is set to 0.05, which would ensure that with at least a 95% of probabilty the hypothesis would be correctly rejected. Following the standard criteria, a threshold of 0.05 has been selected to evaluate the hypothesis tests.

3.3.3 SatImage

Following the methodology previously described, the results obtained per each isolated parameter have been computed. Remember that this accuracies are found by testing the model for all values of an specific parameter when all other parameters are fixed as default (see table 6). The results obtained for SatImage data set are presented in the table 7 (in bold the parameters providing the highest accuracy).

Parameter	Value	Efficiency (ms)	Accuracy (%)
k	1	425	90.38
k	**3**	**425**	**91.02**
k	5	426	90.82
k	7	489	90.77
distance	**1-norm**	**423**	**91.13**
distance	2-norm	426	90.82
distance	chebyshev	634	88.90
policy	majority	426	90.82
policy	inverse	425	90.82
policy	sheppard	426	90.82
weighting	uniform	426	90.82
weighting	**mutual_info**	**442**	**90.93**
weighting	relief	430	89.09

Table 7: Average efficiency and accuracy metrics for each parameter in isolation.

Table 7 shows that the parameters that provides the best accuracy are $k = 3$, 1-norm distance metric and mutual info for weighting (regardless the policy).

In order to see if this parameters produce a statistical significant different in the performance of the model we perform a Friedman Test's with all 10 data sets. The results of this test are presented in table 8.

Parameter	p-values	Null-Hypothesis
k	0.240	confirmed
distance	0.0005	rejected
policy	0.507	confirmed
weighting	0.0009	rejected

Table 8: Friedman Test's results for SatImage

From table 8 we can see that the Null-Hypothesis is confirmed for the k value and, obviously, for the policy (sense we observed that the results were totally independent of this parameter). This means that neither k parameter nor policy parameter produces a statistical significant difference among the performances of the model in terms of accuracy.

However, we can not say the same with respect to distance metric and weighting parameters: the p-value of this two parameters is too low to confirm the Null-Hypothesis, which means that both parameters produces a statistical significant difference in terms of accuracy. Therefore, two Wilcoxon test have been done for distance metric and weighting parameters, with a pair-wise analysis between all possible values for each parameter. The results are presented in table 9 and table 10.

distance	1-norm	2-norm	Chebyshev
1-norm	-	confirmed	rejected
2-norm	confirmed	-	rejected
Chebyshev	rejected	rejected	-

Table 9: Wilcoxon test for isolated distance metrics

weighting	uniform	mutal info	relief
uniform	-	confirmed	rejected
mutal info	confirmed	-	rejected
relief	rejected	rejected	-

Table 10: Wilcoxon test for isolated distance metrics

As shown in table 9 the distance metric which produces a significant difference in performance is the Chebyshev metric. In fact, we already seen in table 7, that Chebyshev produces the lower accuracy values, with a difference higher than 2 % with respect to 1-norm metric, which can be considered significant. In contrast 1-norm and 2-norm produce very similar results and therefore the Null-Hypothesis is confirmed.

Table 10 informs that the significant difference in performance relays on relief, which is a non-uniform weighting.

Finally we make the statistical analysis to compare the 5 different models introduced in this section: the best four models derived form table 7 (which are the best model per each of the four parameters when fixing the others as default) and the best model in global (presented in table 5). Table 11 shows simply the characteristics of each of these models, where the last row belongs to the best model in global.

k	Policy	Distance	Weighting	Accuracy
3	majority	2-norm	uniform	91.02
5	majority	1-norm	uniform	91.13
5	inverse	2-norm	uniform	90.82
5	majority	2-norm	mutual info	90.93
5	Sheppard	1-norm	uniform	91.30

Table 11: Best models comparison for SatImage

p-values	Null-Hypothesis
0.123	confirmed

Table 12: Friedman Test's for best models in SatImage

We can see from Friedman Test (see table 12) that the Null-Hypothesis is satisfied since the p-value is higher than 0.05. Therefore, all this fives KNN models are quite similar in terms of performance (no statistically significant difference in accuracy) and we can not consider that one of this models is better than another.

3.3.4 Credit-A

Following the methodology described in section 3.3, the results obtained per each isolated parameter have been computed. The results for Credit-A data set can be found in the table 13.

Parameter	Value	Efficiency (ms)	Accuracy (%)
k	1	7.56	80.88
k	3	7.63	84.08
k	5	7.64	85.81
k	**7**	**7.64**	**86.25**
distance	**1-norm**	**7.46**	**86.25**
distance	2-norm	7.64	85.81
distance	chebyshev	6.85	64.76
policy	**majority**	**7.64**	**85.81**
policy	inverse	7.56	**84.45**
policy	sheppard	7.52	85.37
weighting	uniform	7.64	85.81
weighting	mutual_info	7.64	85.92
weighting	**relief**	**7.71**	**87.54**

Table 13: Average efficiency and accuracy metrics for each parameter in isolation.

Highlighted in bold in Table 13 we can see the isolated parameters providing the highest accuracy: K=7, 1-norm, majority and relief. Note that this table presents the average performance per each of the 10 folds (crossvalidation). However, to perform the statistical analysis, it is required to use the performance per each of the folds, which will be considered as different datasets evaluated per each of the studied models. Taken this into consideration, the Friedman test has been computed:

Parameter	p-values	Null-Hypothesis
k	0.029	rejected
distance	0.0002	rejected
policy	0.368	confirmed
weighting	0.165	confirmed

Table 14: Friedman Test's results for Credit-A

As it can be seen in Table 14, for the different policies and weights in isolation the null hypothesis is confirmed, i.e. there is not a statistical significant difference among the performances with the different tested parameters, which means that statistically speaking there is not a clear difference of performance among the different models despite their different average accuracies. On the other hand, for k and distance, the p-values are below 0.05, reason why the null hypothesis is rejected: the difference of performance is relevant, the different models cannot be considered to provide the same results. For this cases, a paired Wilcoxon test has been performed so as to detect which of the models are the ones differing the most:

K	1	3	5	7
1	-	confirmed	rejected	rejected
3	confirmed	-	rejected	confirmed
5	rejected	rejected	-	confirmed
7	rejected	confirmed	confirmed	-

Table 15: Wilcoxon test for isolated k's

distance	1-norm	2-norm	Chebyshev
1-norm	-	confirmed	rejected
2-norm	confirmed	-	rejected
Chebyshev	rejected	rejected	-

Table 16: Wilcoxon test for isolated distances

We can see how doing the pair-wise analysis can give us more light on which are the models that do not present an equally distributed performance. Concretely, the focus is going to be on k=7 and 1-norm as they are the models providing a higher average accuracy (see Table 13) for the isolated k's and distances respectively. When it comes to distances, there is a very clear difference between 1 and 2-norms and Chebyshev, while it is not the case for 1 and 2 norms (which present also really similar average accuracies). Concerning the k's, while it is true that the difference with k=1 is clearly statistically significant, it is not the case for k=3 and k=5, where the null hypothesis is confirmed meaning that the performances obtained from the different models come from the same distribution. Therefore, despite the average accuracy is considerably higher (especially with respect to k=3), the improve of performance has to be considered by chance. In the end, evaluating a metric with an average has always an important problem: outliers. Outliers can clearly affect the average final metric but analysing element-wise per each of the 10 folds, the performance of the different models is quite similar, i.e. their difference of performance is symmetric about zero.

After doing this complete analysis, it is time to perform a statistical analysis not only between the isolated metrics but also with their best representative and the global best combination of parameters. The selected winners among the isolated parameters are going to be finally the ones that were providing a higher accuracy. While it is true that the performed statistical test shows that some of them are not statistically different from other option, there is not a clear benefit of selecting the other options rather than the one providing the highest accuracy (the efficiencies, for instance, are practically the same among the comparable options). To sum up, the analysed models together with its accuracy metrics are shown in table 17.

k	Policy	Distance	Weighting	Accuracy
7	majority	2-norm	uniform	86.25
5	majority	1-norm	uniform	86.25
5	majority	2-norm	uniform	85.81
5	majority	2-norm	relief	87.54
5	sheppard	chebyshev	relief	87.97

Table 17: Best models comparison for Credit-A

Taking all of this into consideration, Friedman test has been computed among all of them:

p-values	Null-Hypothesis
0.434	confirmed

Table 18: Friedman Test's for best models in Credit-A

As can be seen, the obtained p-value is way above 0.05, reason why the null hypothesis for the Friedman test is confirmed. Consequently, all of the best models analysed are confirmed to have a symmetric difference of performances around 0, i.e. follow the same distribution. In that concrete case, was already visible in the values obtained in Table 17 that the average accuracies are quite similar. Nevertheless, an average metric is never a statistical guarantee. In this case, Friedman Test has confirmed the hypothesis. To sum up, the differences in accuracy seen in Table 17 cannot be considered as statistically significant, as it does not show the real difference when it comes to model performance.

4 Reduction KNN algorithm

In this section we are going to introduce three different algorithms used to reduce the number of samples in the train data set. Reduction techniques are very useful specially when we have to work with huge training sets, since they may require a large computational time (large training time when we talk about eager learning algorithms and large consultation time and storage when we talk about lazy learning algorithms, such as KNN). Therefore, it is interesting to analyze how the performance of the KNN (in terms of accuracy and efficiency) changes due to a reduction in the training set. For sure the efficiency will increase (since we will be working with less samples) but the accuracy may decrease slightly if the reduction is high. It is a tradeoff between efficiency and accuracy where in some applications we may decide the highest accuracy (no matter the computational cost) but in other applications we may prefer highest efficiency (if the accuracy has not been affected too much).

There are three main groups of reduction techniques: condensation techniques, edition techniques and hybrid techniques. We have implemented one algorithm from each group.

4.1 Selective Nearest Neighbor

Selective Nearest Neighbor (SNN) [5] is a quite old condensed reduction technique. Condensed techniques aim to retain border points while perceiving the accuracy over the training set but maybe affecting negatively to the generalization over the test set. The reduction rate of this techniques is normally high.

SNN iterates over a binary square matrix A which contains the information of the related neighbor (RN) of each sample in the training set: each column i contains the information of the related neighbours of the sample x_i (1 in the row j means that x_j is RN to x_i while 0 means that x_j its not a RN of x_i).
Two conditions must be satisfied for x_j to be a RN to x_i: 1) x_i and x_j must belong in the same class and 2) x_i is nearer to x_j than to any sample in the other class. Then, this matrix is iterated over 3 steps, where rows and columns are being deleted in each step, until the steps do not produce any change to the matrix. Step 1 eliminates all columns with a single 1, (found in a given row j) and adds to the reduced set S the instance x_j (row j is also deleted from matrix A). Step 2 deletes row j if for all (remaining) columns i and for some (remaining) row k, $A_{ji} \leq A_{ki}$. Step 3 deletes column i if for all (remaining) rows j and for some (remaining) column k, $A_{ji} \geq A_{jk}$. If some of the steps have produced a change in matrix A, return to step 1, otherwise the process ends and each instance x_j of each remaining row j in A are added to the reduced set S.

4.2 Extended Nearest Neighbors

Extended Nearest Neighbors (ENN) [6], also known as Wilson's Editing, relies on a basic idea: all the examples missclassified by the KNN algorithm have to be removed from the training set. The way to implement this is quite simple, as it is only necessary to execute the KNN algorithm for each of the training samples. In that process, each of the samples that is being evaluated is not included in the training (otherwise it would always be the nearest neighbour). After that, all those examples which are classified with a class different from their true label, are excluded from the train set.

In practical terms, this is not a very complex algorithm to implement. Nevertheless, it presents some challenges, specially if the data sets are big, as it is necessary to run the KNN for each of the examples in the train set. Finally, it is also necessary to specify the parameters for the initial KNN. In our implementation, so as to ensure consistency with the final experiment, it has been decided to apply the same parameters which are going to be used afterwards in the actual test.

4.3 Decremental Reduction Optimization Procedure 2

Decremental Reduction Optimization Procedure 2 (DROP2) is an algorithm based on DROP1 [7], which is equivalent to RNN [8].

The DROP1 algorithm reduces the inference set by iterating over the examples and removing only those that, after removing that example, the same number of associates can be correctly classified (that is, other examples having the first one as one of their nearest neighbors). For example, if 3 of the 5 associates of P are correctly classified initially, after removing P at least 3 of its associates must still be correctly classified afterwards. Only examples satisfying this condition are finally removed.

DROP2 is an improvement of DROP1. In the previous algorithm, notice that since it is iterative, the set in which the associates of a given sample P are found is constantly being reduced. DROP2 changes that behavior in such way that when a sample P is removed, the next nearest neighbor is found for each of its associates since P no longer accounts; however, in this case, P is still kept as an associate of its nearest neighbors and it will continue to help deciding if the remaining samples can be removed or not.

Moreover, DROP2 also changes the order of removal of samples. In DROP2, the first samples to be removed are those that are farther from their nearest enemy, which is considered the closest sample with a different label. This tends to remove points that are not in the decision boundary first, and that increases the chances to keep those border points which are more representative.

This algorithm does not have a difficult implementation either. Despite, it is theoretically more expensive in computing time because it has to perform KNN over the associates of every sample in the data set, and then find a new nearest neighbor for each associate every time. Since it also uses the KNN algorithm, we also decided to apply the same parameters that are going to be used with the KNN run on the reduced data set.

4.4 Reduction algorithms performance

In this section we present the evaluation of the three reduction algorithms applied to each data set. In both experiments, several accuracy and efficiency metrics are shown that are related to the reduction techniques and also to the KNN algorithms that are run and evaluated using each reduced data set. To do that, we measure the performance of the best KNN model corresponding to each data set obtained in the previous sections.

4.4.1 Results for SatImage

Algorithm	Reduction eff. (ms)	Reduced set (%)	KNN eff. (ms)	Accuracy (%)
None	**0**	100	1005	**91.30**
SNN	104478	**6.50**	**52**	76.55
ENN	3676	91.29	465	90.46
DROP2	82168	10.93	72	90.02

Table 19: Results of applying different reduction techniques, evaluating the reduction efficiency, the percentage of instances kept and the efficiency and accuracy of KNN using the reduced set.

A comparison of the different reduction techniques applied to SatImage can be seen in table 19. In this table, the most important metrics are the average accuracy and efficiency of models using each reduced set, and the relative size of those reduced sets.

In terms of accuracy, none of the reduction algorithms is able to improve the accuracy of the model by selecting a subset of instances of the data set. We can see a tendency related to that fact, that the smaller the reduced set is, the higher the loss in accuracy it will get. This is a common tradeoff since normally, the more examples the model has available the better it will be able to generalize. Nonetheless, as we have already stated, often the performance of a model is not only determined by its ability to make predictions, but also by the conditions in which it can perform that task. In particular, memory usage and response time. In that sense, reduction techniques are applied to data sets used for inference to attempt to improve these capabilities for lazy-learning models.

In relation to the results, the two algorithms that reduce the data set size more successfully are SNN and DROP2, with reduced sets of size 6.50% and 10.93% respectively and relatively to the total training set size. Apart from that, there is a significant difference when it comes to the accuracy achieved by the KNN model using these reduced sets. While DROP2 allows it to maintain a 90.02 score in accuracy, SNN decreases that score to 76.55. Moreover, DROP2 is able to perform the reduction 27% faster than SNN. Besides, we do not consider that improvement in time to be that important compared to other metrics, because it is a one-time action to be performed in practice.

ENN's results are neither surprising nor irrelevant. At first, we can see that it is the reduction technique that provides the best accuracy compared to the others. But that is presumable since it keeps 91.29% of the instances after the reduction. In spite of that, we must take into account that while it only reduces the data set by a mere 10%, the response time of the KNN model ends up being halved.

To sum up with these results, we could say DROP2 is the most effective algorithm to perform a reduction to the SatImage data set. We think that the accuracy-storage tradeoff is well justified here since at the cost of roughly 1.30% loss in accuracy, DROP2 is able to make KNN maintain that measure by using *only* 10.93% of the total amount of instances, which translates into almost 10 times less usage in memory. In addition to that, KNN are able to make predictions more efficiently, i.e. approximately 14x faster, which is also remarkable.

4.4.2 Results for Credit-A

Algorithm	Reduction eff. (ms)	Reduced set (%)	KNN eff. (ms)	Accuracy (%)
None	**0**	100	7.42	**87.97**
SNN	559	8.42	2.79	84.19
ENN	51	87.70	6.74	86.23
DROP2	1992	**5.57**	**2.59**	84.20

Table 20: Results of applying different reduction techniques, evaluating the reduction efficiency, the percentage of instances kept and the efficiency and accuracy of KNN using the reduced set.

The table above summarizes the comparative performance of the best none-reduced model against their reduced versions with the studied reduction algorithms. In this section several tradeoffs are going to be presented to evaluate them.

In terms of accuracy, none of the reduced techniques is able to surpass the none-reduced model. As observed in the previous data set, also in this case there is a certain pattern between the loss on accuracy and the reduction percentage: the more the trainset is reduced, the more the decay in accuracy. As explained, this is a common tradeoff, especially if there is a strong reduction, as some relevant data might be lost, making the model lose generalization capacity.

When analysing the reduction capacity of the algorithms, Credit-A goes quite in hand with SatImage: SNN and DROP2 are the methods that provide a higher reduction, achieving 8.42% and 5.57% respectively. Nonetheless, in this case both algorithms are able to provide practically the same average accuracy, around 84.2%. Moreover, in this data set SNN is able to perform the reduction much faster than DROP2. Note how, in any case, the reduction efficiency is better than in SatImage; mainly due to the data set having around 10 time less number of samples.

For its part, ENN is the reduction method providing less differences with respect to the none-reduced model. On the one hand, is the one presenting the lowest loss of average accuracy. However, it only reduces up to 88% of the train samples and fails to considerably reduce the KNN consultation time.

To conclude, it is difficult to discern whether DROP2 or SNN are providing the best performance for the reduction algorithms. On the one hand, they present practically the same average accuracy (84.2%) and efficiency (2.59 ms vs. 2.79 ms). The main differences come in the reduction capacity and the required time to achieve it: while DROP2 is able to reduce the data around 3pp more than SNN it requires more than three times to do it, although it is true that it is in the order of few seconds. The statistical analysis might help to finally choose among one of these two options.

4.5 Statistical Analysis

In this section we perform the statistical analysis with the tree different reduction techniques (SNN, ENN and DROP2) and with the performance with no reduction, in order to see if applying a reduction technique significantly improve or deteriorates the performance of the KNN model.

4.5.1 SatImage

So as to validate the results shown in Table 19, a Friedman test among the compared models has been computed:

p-values	Null-Hypothesis
0.000007	rejected

Table 21: Friedman Test's for reduced vs. best models in SatImage

The obtained p-value is tiny, way below 0.05; therefore, there is a statistical significant difference among the different models. As before, in order to understand where this difference come from, a Wilcoxon paired test is performed between the compared models:

Algorithm	No	SNN	ENN	DROP2
No	-	rejected	rejected	rejected
SNN	rejected	-	rejected	rejected
ENN	rejected	rejected	-	confirmed
DROP2	rejected	rejected	confirmed	-

Table 22: Wilcoxon test for reduction algorithms

From Table 22, we will concentrate on the tests involving the none reduced model, as it is our aim to evaluate if there is a statistical significant difference between it and their reduced versions. Analysing the table above, it can be seen how the Wilcoxon's null-hypothesis has been consistently rejected for all the pairs involving the none reduced model. Consequently, the statistical tests have confirmed what was already seen analysing the average results in Table 19: there is a significant worsening of performance when using the reduction algorithms. Still, the improvement in other areas, like the clear decrease in consultation time, could be used as a tradeoff with the accuracy loss; taking into consideration, in any case, that there is a significant decline in accuracy performance.

4.5.2 Credit-A

As a final step, the worthiness of the reduction algorithms is tested by performing a Friedman Test among the performances obtained with each of the reduced models and the best combination of parameters found for this data set. The results for the test can be found in the following table:

p-values	Null-Hypothesis
0.016	rejected

Table 23: Friedman Test's for reduced vs. best models in Credit-A

As previously explained, the p-value below 0.05 denotes that there is a statistical significant difference among the different models. Consequently, so as to have a clear picture between pairs of models, a Wilcoxon test is performed for all the possible pair combinations:

Algorithm	No	SNN	ENN	DROP2
No	-	confirmed	rejected	rejected
SNN	confirmed	-	confirmed	confirmed
ENN	rejected	confirmed	-	rejected
DROP2	rejected	confirmed	rejected	-

Table 24: Wilcoxon test between reduction and best model for Credit-A

From the results above, the interest is going to be focused mainly in the comparison of the best model and their reduced versions. In Credit-A, the results of the test are specially interesting as the only reduction model that fulfills the null-hypothesis (shares distribution with the not reduced model) is SNN, which is the one that has a higher average accuracy difference. The explanation for this has been already explained in this report: average metrics are heavily affected by outliers, while the hypothesis test performs a fold-by-fold comparison, being more able to detect the actual difference of performance. In this concrete case, the fact that there is not a statistical significant difference between the none reduced and the SNN model may be very relevant, as the later is able to provide an statistical equivalent accuracy with only an 8.42% of the required samples, which allows the consultation time to be more than halved. In addition, while it is true that it requires an extra time for performing the reduction; it is quite low (around 2 seconds), which makes feasible to implement this solution. To conclude, this final case shows us once again how the average metrics can lead to misleading conclusions and reinforce the importance of performing statistical analysis to validate the results.

5 Conclusions

In this work we have implemented and analyzed four different parameters for the KNN algorithm, from which we obtained 108 different models. We have learned the importance of performing an statistical analysis when working with different models and different data sets (coming from 10 fold cross-validation). In fact, we made the statistical analysis over five different models (the best in global and the best for each parameter in isolation) and we observe that no significant difference in performance exists. However, we have proven using both SatImage and Credit-A data sets that Chebyshev distance metric usually proves lower accuracy results with respect to 1-norm or 2-norm (with significantly difference proved by Wilcoxon test over the 3 distance metrics). We could not expect obtaining the best results with Chebyshev since this distance metric only consider the feature with larger difference and we have seen that in general all features have an important role (since no significant improvement has been observed when comparing uniform with no-uniform feature weighting).

Regarding the reduction technique we strongly consider to apply DROP2 since it significantly reduces the training set (maintaining approximately only the 10 % of the total samples) which increases the efficiency and reduces the storage space substantially, while maintaining a high accuracy level. This is very interesting in some applications where efficiency is more important than accuracy and specially when we are dealing with large data set.

In this document, even though we did not perform feature selection, we employed and analyzed feature weighting. In comparison to the reduction techniques, computing feature weights sometimes lead to better results, and sometimes worse results, in terms of accuracy. On the other hand, reduction techniques almost always produced lower accuracy scores on the KNN applied afterwards, sometimes by a small enough margin. It is in these cases where other capabilities measured by metrics such as the relative size of the reduced instance set or the response time of the classifier model, that come into play and would make us reason about what approach is most suitable for our scenario.

6 Reproducibility

In order to enable reproducibility on the experiments performed on this task, we have designed a CLI with different commands to run each case separately. In the following listing 1, several examples of command lines to execute the program are listed to help understanding its usage. All of them can be executed from the root folder of the project. Note that for the `analyze` and `compare` commands the input files are already provided with the source code in the *results* folder, which were obtained by running the `gridsearch` and `test-reduction` commands.

Listing 1: Command examples to execute the experiments

```
$ python3 main.py knn --help
$ python3 main.py gridsearch --help
$ python3 main.py test-reduction --help
$ python3 main.py analyze --help
$ python3 main.py compare --help

$ python3 main.py knn -d satimage -k 5 -s 1-norm \
    -p sheppard -w uniform -r no
$ python3 main.py knn -d credita -k 5 -s chebyshev \
    -p sheppard -w relief -r no
$ python3 main.py knn -d credita -k 1 -s 2-norm \
    -p inverse -w mutual_info -r snn

$ python3 main.py gridsearch -d satimage \
    -o results/satimage_gridsearch_cv.csv
$ python3 main.py gridsearch -d credita \
    -o results/credita_gridsearch_cv.csv
$ python3 main.py gridsearch -d credita -cv no \
    -o results/credita_gridsearch_folds.csv

$ python3 main.py test-reduction -d satimage -k 5 -s 1-norm \
    -p sheppard -w uniform -o results/satimage_reduction_cv.csv
$ python3 main.py test-reduction -d credita -k 5 -s chebyshev \
    -p sheppard -w relief -cv no -o results/credita_reduction_folds.csv

$ python3 main.py analyze -g results/satimage_gridsearch_folds.csv \
    -G results/satimage_gridsearch_cv.csv \
    -r results/satimage_reduction_folds.csv
$ python3 main.py analyze -g results/credita_gridsearch_folds.csv \
    -G results/credita_gridsearch_cv.csv \
    -r results/credita_reduction_folds.csv

$ python3 main.py compare -p k -m accuracy \
    -f results/satimage_gridsearch_cv.csv
$ python3 main.py compare -p distance -m efficiency \
    -f results/credita_gridsearch_cv.csv
$ python3 main.py compare -p algorithm -m percentage \
    -f results/satimage_reduction_cv.csv
$ python3 main.py compare -p algorithm -m accuracy \
    -f results/satimage_reduction_cv.csv
$ python3 main.py compare -p algorithm -m reduce_efficiency \
    -f results/satimage_reduction_cv.csv
```

References

[1] C. L. DuBois, "UCI network data repository." `http://networkdata.ics.uci.edu`, 2008.

[2] M. Robnik-Sikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine Learning*, vol. 53, pp. 23–69, 10 2003.

[3] L. F. Kozachenko, N. N. Leonenko, "Sample estimate of the entropy of a random vector," *Probl. Peredachi Inf.*, pp. 9–16, 1987.

[4] J. Demsar, "Statistical comparison of classifiers over multiple data sets," 2006.

[5] G. L. Ritter, H. B. Woodruff, S. R. Lowry, T. L Isenhour, "An algorithm for selective nearest neighbour decision rule," 1975.

[6] D. L. Wilson, "Asymptotic properties of nearest neighbor," 1972.

[7] D. Wilson and T. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, pp. 257–286, 01 2000.

[8] G. Gates, "The reduced nearest neighbor rule (corresp.)," *IEEE Transactions on Information Theory*, vol. 18, no. 3, pp. 431–433, 1972.