Introduction to Machine Learning:

Work 3 - Lazy learning exercise

Professor
SALAMO LLORENTE, Maria

Team members
DZMANASHVILI, Demetre
BEJARANO SEPULVEDA, Edison Jair
GLOWACZ, Jakub

Universitat de Barcelona
Facultat de Matemàtiques i Informàtica
Master's degree in Artificial Intelligence
Barcelona, 2021

Contents

1 Introduction to Lazy Learning

Lazy learning algorithms are a set of methods capable of improving their performance over time, which incorporate certain characteristics that differentiate them from their counterpart, the eager learning algorithms. Their main difference is that lazy learning methods delay most part of the computations until the moment when the prediction of a new instance is requested (1). In order to better understand this difference, the following two concepts are introduced (3):

- *Training time*: it consists of the first part of the process, where the system makes some inferences from the training data which are then stored for their use during consultation.
- *Consultation time*: it consists of the second part of the process, where new inputs are presented to the trained system, which replies by making the inferences corresponding to the inputs.

Thus, lazy learning refers to the machine learning techniques that spends most of its computational cost on consultation instead of training. In contrast, the aforementioned eager learning algorithms spend most of their processing time on training, during which they generate a rule-set or model of some kind, which is then able to make predictions of new inputs my means of simpler and rather fast processes.

Lazy learning techniques can be classified into two different groups, as follows: instance-based learning and lazy Bayesian rules. For the exercise under discussion in the present work, we focus on the former group of algorithms, instance-based learning.

2 Instance-based learning algorithms

This example learn the training examples by heart and then generalize to new instances using some similarity metric. It's called instance-based since the hypotheses are built from the training data. It's also known as lazy learning or memory-based learning. The size of the training data determines the time complexity of this technique. In this project we are gonna use a few of Instance-Based algorithms. Those are:

- **IB1**, **IB2**, **IB3**, **k-IBL**

In general all the algorithms mentioned above keeps the Content Description (CD) of instances that was added once during the training. The main differences between algorithms are how many and by what condition are they gonna keep instances and what will be the similarity metrics for comparing the algorithms. We are gonna describe those algorithms in detail in the following sections

2.1 IB1 algorithm

IB1 is the simplest method in Instance Based Learning. The algorithm behind is also very simple: At the start it is gonna add first element from training sate to CD. After that for every training instance the algorithm will look into CD and will find the instance that has the smallest distance between instance that is gonna classified and all instances that are in CD. After finding the most similar instance we will check the classes for both and will check if it was classified correctly or not. After classification we are gonna add classified instance in CD without any condition.So IB1 is gonna keep every instance in CD that was classified during the training. It does not look if it was correctly classified or not. The Distance metric for finding the most similar or the most closest one is Euclidean Distance that is described by the following formula:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

This algorithm is memory inefficient, but other than that is is known by high accuracy and fast execution time.

2.2 IB2 algorithm

IB2 is exactly the same but with just one difference. It is gonna keep misclassified instances in CD. So the entire process is the same, we are gonna add first instance in empty CD. We are gonna try to classify training instances and we are gonna add it into CD if the classification was incorrect. Like the IB1, the distance metric to find the nearest instance is Euclidean.

This Algorithm is improved version of IB1, it more than halved the number of instances that were keep in CD. Execution time did not changed that much. About the Accuracy it has a slightly worse results compared to IB1.

2.3 IB3 algorithm

IB3 is hardest algorithm among those that we already presented. IB3 is known as a noise tolerant algorithm. The idea of IB3 is "wait and see", it is collecting information during training and discarding some instances from CD. Now lets see the entire algorithm: Like others, on the first step we are adding first training instance into empty CD. After that we are picking *acceptable instances* out of entire CD. If *acceptable instances* are empty, we are picking random instance from CD and checking if the classes are same or not. If classes are not the same, we are gonna add that instance to CD like IB2. In addition to that we are keeping classification record of every instance in CD. We are saving how many classification attempts was successful for each instances and how many was not. After classification, we are checking if there is an instance as similar as we have chosen for classification, and if there is any we are gonna update classification record of such instances. After everything we are checking if there are any instances that classification record is *significantly poor*, if there is any we are gonna remove that from CD. Now lets clarify some terms:

Acceptable Instances are instances whose *classification accuracy* is *significantly greater* than its class' *observed frequency.*

Classification Accuracy is the number of successful attempts that the instance made during the training divided by number of all classification attempt

Class Observed Frequency is a proportion of instances so far that are of this class

By Significantly Greater we mean when instance‘s classification accuracy *confidence intervals* lower end-point is greater than class observed frequency confidence interval‘s higher end-point

By Significantly poor we mean when instance‘s classification accuracy *confidence interval‘s* higher end-point is less than class observed frequency *confidence interval‘s* lower end-point

Confidence Intervals are calculated by following formula:

$$\frac{p + z^2/2n \pm z\sqrt{\frac{p(1-p)}{n} + \frac{z^2}{4n^2}}}{1 + z^2/n}$$

where n is classification attempts, z is confidence level and it is 0.9 when we are calculating acceptance intervals and 0.7 when we are calculating dropable intervals. p is classification accuracy or observerd frequency, it depends for which one we are calculating boundaries.

IB3 compared to IB2 and IB1 uses significantly less memory but the execution time is much greater because we are doing far more calculations, Accuracies, in our case, has not improved much using IB3 so we can not say that this is the best method to use, because by far IB1 has achieved highest accuracy and IB3 used far less memory rather than that two.

2.4 k-IBL algorithm

Departing from the *IBL* algorithms, the next step was to develop a k-Nearest-Neighbours version of them. The only significant difference from the *IBL* algorithm is that instead of taking just one nearest neighbour into accout, we can vary the number specified as *k*. In the *k-IBL* the algorithms finds *k* nearest neighbours, like in the *k-NN* algorithm. Then voting protocols are used to determine the winning label that will be assigned to the classified datapoint. The voting protocols we used are:

- **Most-voted protocol** - item with most occurences wins. In case of ties, we choose the item which has the lowest mean distance from the core point.
- **Plurality protocol** - item with most occurences is assigned 1 point and the rest 0 points. The element with highest score wins, so in that case the element with most occurences. In case of ties, the k number is reduced for following iteration.
- **Borda protocol** - the *k* number of labels are ordered from the lowest distance to the highest. Then they are assigned points from k-1 to 0, in this ordering (5). The element with highest score wins. In the case of ties, number of *k* is reduced for the following iteration.

For the k-IBL we also used 4 different distance metrics to evaluate it:

- **Euclidean**

  The Euclidean distance is described by the following formula(6):

  $$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 ...}$$

  Where x and y mean attributes and the indices mean different sources of data.

- **Manhattan**

  The Manhattan distance is described by the following formula(6):

  $$|x_1 - x_2| + |y_1 - y_2| + ...$$

  Where x and y mean attributes and the indices mean different sources of data.

- **Canberra**

  The Canberra distance is described by the following formula(7):

  $$\sum_{i=1}^{n} \frac{|x_1 - x_2|}{|x_1| + |x_2|} + \frac{|y_1 - y_2|}{|y_1| + |y_2|} + ...$$

  Where x and y mean attributes and the indices mean different sources of data. The Canberra distance is a weighted version of the Manhattan distance.

- **HVDM**

  The HVDM distance is described by the following formula(8):

$$\sqrt{\sum_{a=1}^{m} d_a^2(x_1 a, x_2 a)}$$

  Where x_1 and x_2 mean different sources of data, a means attribute and m is the number of attributes. The d_a is interpreted in following way:

  - **If data is nominal**:

$$d_a(x_1, x_2) = \sum_{c=1}^{C} |\frac{N_a, x_1, c}{N_a, x_1}| - |\frac{N_a, x_2, c}{N_a, x_2}|$$

    Where N_a, x_1 is the number of incstances in the training set T that have value x_1 for attribute a, and N_a, x_1, c is the number of instances in T that have value x_1 for attribute a and output class c. C is the number of classes.

  - **If data is numerical**:

$$d_a(x_1, x_2) = \frac{|x_1 - x_2|}{4\delta}$$

    Where δ is the standard deviation of the numeric values of attribute a.

3 Feature selection

As can be observed in many machine learning problems, datasets containing many different features are common. Often, some of the features have great potential since they provide useful information for the problem, while others may contain less helpful data which might lead to overfitting or long time on training. For that reason, it can be very useful to choose the best features for the specific case under study, while ignoring the ones that are not as useful. In order to do that, many feature-selection techniques are available, which can be applied before the classification model is trained. As summarized by Cunningham et al. (2) feature-selection techniques can provide the following advantages:

- *Better classifiers*: the accuracy of the model might improve since noisy features can be ignored.
- *Knowledge discovery*: through feature selection one can identify influential features that are more useful than others and thus it can provide further understanding of the data and its features.
- *Less complexity*: thanks to feature selection, the dataset can be reduced significantly to a "minimal subset of features".
- *Computational cost*: if the model is simpler because it has less features, the cost will be less to set up and run.

Feature selection algorithms can be divided into three different categories: Wrappers, Filters and Embedded.

In the Wrapper-type methods, the classifier is wrapped in the feature selection process, which has the advantage of tying the feature selection to classifier performance; however, the computational cost increases significantly.

For the Filter methods, each feature is given a score and then a selection policy uses this ranking to select a feature subset. In other words, a basic filter gives a feature scoring mechanism and then the selection is based on the score. Filter-type methods can be divided into:

- Basic: I-Gain or Chi-Squared.
- Relief Algorithms: This technique have the advantage of the speed while scoring features.
- Correlation-Based algorithms.

For this method, some selection strategies might be selecting the top ranked k features, selecting the top 50% or selecting all the features with non-zero scores.

The third are the Embedded-type methods, which refer to the methods where the feature selection is made at the same time as the classifier training process takes place. Examples of classification algorithms that also perform feature selection are Random Forest, Decision Trees, and Linear Model.

For this project two of the here presented feature-selection methods were implemented: I-Gain and Relief. The baseline results from the best classifier using all the features in the dataset are compared with the results from the same classifier but using these feature-selection methods. These comparisons are presented and discussed for the datasets called Hypothyroid and SatImage in sections 4.1.3 and 4.2.3 respectively.

4 Results

To evaluate the performance of the algorithms we calculated the accuracy, variance, time and memory for each fold in. To evaluate the performance of each algorithm we calculated the average of the mean ranks for each fold. In case of IB1, IB2 and IB3, we also used t-test, to see if there are significant differences between the performances of the algorithms.

In case of comparing k-IBL, we evaluated 48 configurations with following properties:

- k-value: - 1, 3, 5, 7
- voting protocol: most-voted, plurality protocol, Borda protocol
- distance metrics: Euclidean, Manhattan, Canberra, HVDM

To see if there are significant differences between the configuration, we used Nemenyi test. We also computed the average mean rank of the folds for each configuration.

Statistical methods for comparing the differences between algorithms:

- **T-test**

  This statistical method is used to compare the statistical differences between two independent sample means(4). In our implementation we use it to compare IB1, IB2 and IB3 algorithms in pairs. The key hypothesis behind this test is called the null hypothesis. This statement assumes that there is no difference between the datasets. To measure if the hypothesis is true or not, the t-test returns a p-value. It is measured by taking into account the reference and compared value and a calculation of deviation between these values, with respect to the probability distribution of the statistic. The p-value varies between 0 and 1. When p-value is higher than 0,05 it means that the difference between two datasets is insignificant. If it is lower, it means that the null hypothesis could be rejected.

- **Nemenyi test**

  The Nemenyi test works similar to t-test, but it is a measure that is performed not only on pairs of datasets, but multiple datasets. In our case we use it to compare the 48 configurations of the

	Mean Accuracy	Variance	Avg. Memory used	Execution Time
IB1	93.29%	0.00019	3772/3772	0.9 sec
IB2	88.17%	0.0056	505/3772	8.7 sec
IB3	58.91%	0.012	102/3772	3 min

k-IBL algorithm. the Nemenyi test calcualte the differences between algorithms pairwise, but also takes into account the critical difference based on the number of algorithms and the student distribution of the statistics.

To calculate the rank of each algorithm, we compute mean accuracy and the mean rank, which is calculated based on *rankdata* class from *scipy* package. Besides the general performance of the algorithms it also takes into account the variability of the data in different folds of cross-validation. We pass the ranks for each folds to perform T-test in the case of IBL algorithms and Nemenyi test in the case of k-IBL.

4.1 Hypothyroid dataset

This is a mixed dataset with numerical and nominal data. It contains information about a medical condition named hypothyroid, and the occurences of it with attributes such as age, sex, pregnant, etc.

4.1.1 IBL algorithms

After performing IBL method on Hypothyroid dataset we got the following results

As we can deduce from the results, IB1 had the best accuracy, also variance is the lowest, execution time for IB1 was more than IB2 and IB2 used much less memory than IB1. In terms of memory IB3 used the fewest memory among them but the accuracy was significantly poor so we can not consider IB3 as a best method. Because IB1 got better accuracy and less variance we have chosen IB1 for k-IBL.

4.1.2 k-IBL algorithm

After performing the k-IB1 algorithm on *Hypothyroid* dataset we obtained the following results:

As it is visible on table 1, the mean ranks and accuracies are varying a lot. The best rank in this set is assigned to configuration 35. The highest accuracies however were obtained by configurations 26, 27 and 38. All of the configurations have one thing in common. They use k value over 1. In this dataset it is visible that *Most-voted* and *Plurality* voting policies work better than the *Borda* policy. Time of performing the algorithms differed from 1,1s to 1,4s in case of the *HVDM* metric which was the longest. The fastest metric was *Manhattan* which did times from 0,7s to 0,85s. The other metrics had similar execution time to the *Manhattan*, but were slightly slower. In terms of metrics performance, the best ones were *Manhattan* and *Canberra*. Execution time did not change when increasing k value and adding voting policies. After performing Nemenyi test, it is possible to find the most significant differences between algotrithms. Part of the Nemenyi matrix, which outputs 48x48 array with the *p-values* is shown in table 2.

As it is visible on table 2 a lot of the configurations reject the null Hypothesis, which means the differences are significant. The most significant differences were found between configurations from the beginning of the table 1 and the ending. That means, that with the increase of k, the differences are more and more significant.

Table 1 – Mean accuracy and mean rank for the 48 configurations of k-IBL for *Hypothyroid* dataset

config	**options**			**mean accuracy**	**mean rank**
	k	**distance metric**	**voting protocol**		
1	*1*	*HVDM*	*Borda*	91,8 %	44,2
2			*Most-Voted*	91,8 %	44,2
3			*Plurality*	91,8 %	44,2
4	*1*	*Euclidean*	*Borda*	93,3%	26
5			*Most-Voted*	93,3%	26
6			*Plurality*	93,3%	26
7	*1*	*Manhattan*	*Borda*	93,2%	28,3
8			*Most-Voted*	93,2%	28,3
9			*Plurality*	93,2%	28,3
10	1	*Canberra*	*Borda*	93%	32,4
11			*Most-Voted*	93%	32,4
12			*Plurality*	93%	32,4
13	*3*	*HVDM*	*Borda*	91,8 %	44,2
14			*Most-Voted*	91,8 %	17,4
15			*Plurality*	91,8 %	17,4
16	*3*	*Euclidean*	*Borda*	91,8 %	26
17			*Most-Voted*	93,9%	15,95
18			*Plurality*	93,9%	15,95
19	*3*	*Manhattan*	*Borda*	93,3%	28,3
20			*Most-Voted*	94%	8,1
21			*Plurality*	94%	8,1
22	*3*	*Canberra*	*Borda*	93,2%	32,4
23			*Most-Voted*	94,4%	8,2
24			*Plurality*	94,4%	8,2
25	*5*	*HVDM*	*Borda*	93%	44,2
26			*Most-Voted*	**94,5%**	19,15
27			*Plurality*	**94,5%**	18,35
28	*5*	*Euclidean*	*Borda*	91,8 %	26
29			*Most-Voted*	93,9%	16,45
30			*Plurality*	93,9%	16,25
31	*5*	*Manhattan*	*Borda*	93,3%	28,3
32			*Most-Voted*	94%	10,5
33			*Plurality*	94%	12
34	*5*	*Canberra*	*Borda*	93,2%	32,4
35			*Most-Voted*	94,2%	**7,45**
36			*Plurality*	94,1%	8,5
37	*7*	*HVDM*	*Borda*	93%	44,2
38			*Most-Voted*	**94,5%**	20,45
39			*Plurality*	94,4%	21,35
40	*7*	*Euclidean*	*Borda*	91,8%	26
41			*Most-Voted*	93,7%	21,2
42			*Plurality*	93,7%	21,55
43	*7*	*Manhattan*	*Borda*	93,2%	28,3
44			*Most-Voted*	94,1%	13,25
45			*Plurality*	94%	13,95
46	*7*	*Canberra*	*Borda*	93%	32,4
47			*Most-Voted*	94,2%	11,7
48			*Plurality*	94,2%	11,2

Table 2 – Nemenyi test for the *Hypothyroid* dataset with k-IBL

	1	**2**	**3**	**4**	**...**	**45**	**46**	**47**	**48**
1	1	0,9	0,9	0,65	...	0,001	0,9	0,001	0,001
2	0,9	1	0,9	0,65	...	0,001	0,9	0,001	0,001
3	0,9	0,9	1	0,65	...	0,001	0,9	0,001	0,001
4	0,65	0,65	0,65	1	...	0,9	0,9	0,9	0,9
...	...	...	...	...	1	...	...	...	...
45	0,001	0,001	0,001	0,9	...	1	0,62	0,9	0,9
46	0,9	0,9	0,9	0,9	...	0,62	1	0,35	0,29
47	0,001	0,001	0,001	0,001	...	0,9	0,35	1	0,9
48	0,001	0,001	0,001	0,001	...	0,9	0,29	0,9	1

4.1.3 k-IBL with Feature Selection

As it is possible to observe in the table 3, it was tested the I-Gain and Relief techniques for feature selection and it was compare without feature selection. Also we change the number of feature to observed the importance of the number of features in the results.

Table 3 – Comparison with and without Feature selection

Comparison of with and withaout Feature Selection algorithm				
FS algorithm	**Mean accuracy**	**memory**	**time used (Seconds)**	**Features**
Without	**89.74%**	100.00%	23.3145837160118	36
I-Gain	**89.42%**	**100.00%**	**18.0778215570608**	**25**
Without	89.74%	100.00%	22.3256329659489	36
relief	89.36%	100.00%	18.3161552590318	25
Without	89.74%	100.00%	22.0996954079892	36
I-Gain	89.04%	100.00%	17.1196567089937	20
Without	89.74%	100.00%	22.4422708940692	36
relief	88.55%	100.00%	17.7967225040484	20
Without	89.74%	100.00%	24.0428472210187	36
I-Gain	88.44%	100.00%	17.7124638280075	15
Without	89.74%	100.00%	24.0668900682398	36
relief	86.78%	100.00%	19.526246436988	15
Without	89.74%	100.00%	25.0099670709751	36
I-Gain	87.48%	100.00%	18.2044427930377	10
Without	89.74%	100.00%	24.0741096869751	36
relief	84.12%	100.00%	17.4824694099952	10
Without	89.74%	100.00%	21.6202808399685	36
I-Gain	**85.30%**	**100.00%**	16.114722052007	**5**
Without	89.74%	100.00%	21.1469809270638	36
relief	**60.92%**	**100.00%**	**15.3562649719825**	**5**

The parameters used for the k-IBL was:

- IB1 Algorithm
- Voting_Policy: most_voted
- K:3
- Distance: Euclidean

We can see that the best result for the accuracy was in the case where we doesn't use feature selection in where we obtained 89,74% and 23,314 seconds. The best combination for the time used was in the case where we use Relief as method in the feature selection with 5 features and the results was 15,356 seconds but its accuracy was 60,92%.

- For the *hypothyroid* dataset:

  Relief method:

Feature selection	Eucledian mean accuracy	Time (Seconds)	Features
Without	93,96%	6,809958964	29
relief	93,98%	5,786510452	25
relief	93,98%	6,019000875	20
relief	93,96%	6,431502169	15
relief	93,98%	5,464209219	10
relief	**94,81%**	**5,549279243**	**5**

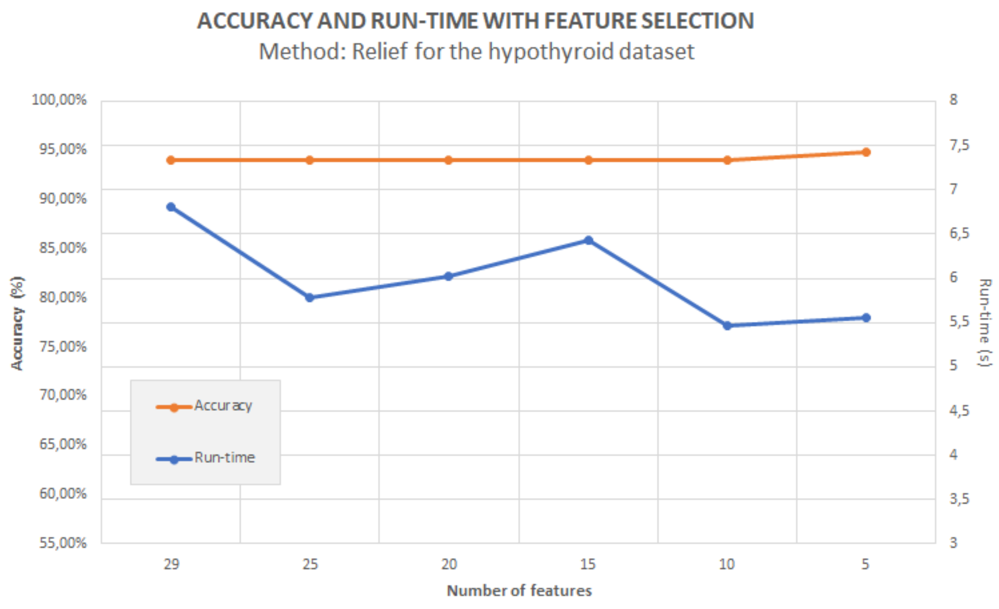

Figure 1 – Relief method for the hypothyroid dataset

I-Gain method:

Feature selection	Eucledian mean accuracy	Time (Seconds)	Features
Without	93,96%	6,809958964	29
I-Gain	94,20%	5,859446736	25
I-Gain	95,31%	5,60775805	20
I-Gain	95,63%	5,800794873	15
I-Gain	95,60%	6,064655818	10
I-Gain	**96,53%**	**5,789671323**	5

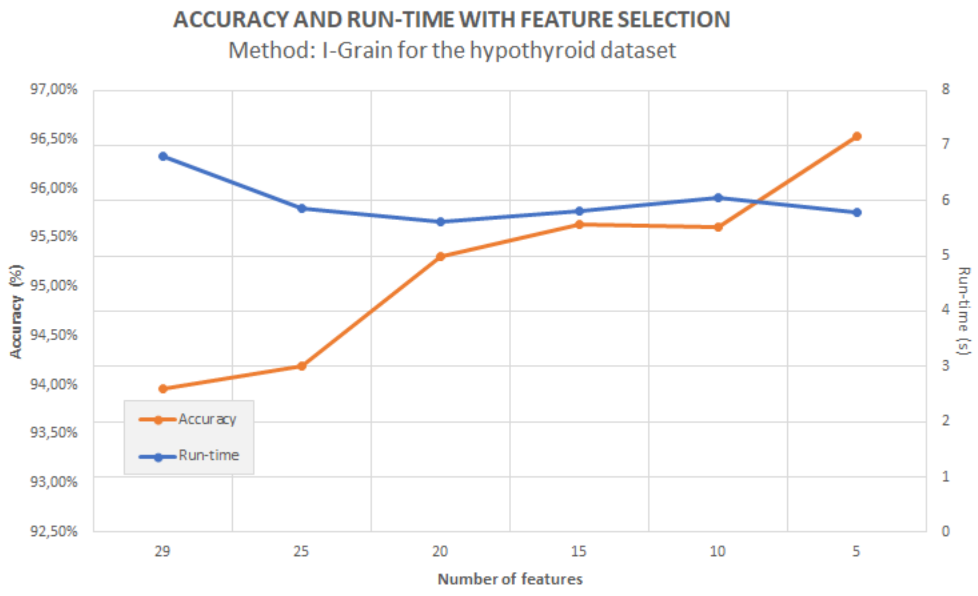

Figure 2 – I-Gain method for the hypothyroid dataset

4.2 SatImage dataset

This dataset contains information retrieved from satelite images. It contains information about multi-spacial pixels in 3x3 neighbourhoods, which are specified as attributes a1, a2, etc.

4.2.1 IBL algorithms

After performing IBL method on SatImage dataset we got the following results

	Mean Accuracy	**Variance**	**Avg. Memory used**	**Execution Time**
IB1	89.52%	0.0001	6435/6435	57 sec
IB2	85.59%	0.0003	1029/6435	12.6 sec
IB3	72.56%	0.035	145/6435	15 min

As we can deduce from the results, IB1 has still the best accuracy, also variance is the lowest and execution time for IB1 was more than IB2, on the other hand, IB2 used much less memory than IB1. In terms of memory IB3 still used the fewest memory among them but the accuracy was significantly poor so we can not consider IB3 as a best method. Because IB1 got the best accuracy and the least variance we have chosen IB1 for k-IBL.

4.2.2 k-IBL algorithm

After performing the k-IB1 algorithm on *SatImage* dataset we obtained the following results:

On table 4 it is visible that the results vary far less than in the case of the *Hypothyroid* dataset. The results do not seem to have any tendency when increasing the k. However the distance metrics still have similar effect as in the first dataset. The *Borda* voting protocol in some cases work better than the other two, which is different from the first dataset. The best algorithms in terms of the rank as well as the accuracy are configurations 20 and 21, with $k = 3$ *Manhattan* metric and *Most-voted*/*Plurality* voting protocol. To compare which of the configurations have significant differences in the mean rank,

Table 4 – Mean accuracy and mean rank for the 48 configurations of k-IBL for *SatImage* dataset

config	**options**			**mean accuracy**	**mean rank**
	k	**distance metric**	**voting protocol**		
1	*1*	*HVDM*	*Borda*	89,9%	17,5
2			*Most-Voted*	89,9%	17,5
3			*Plurality*	89,9%	17,5
4	*1*	*Euclidean*	*Borda*	89,5%	22,25
5			*Most-Voted*	89,5%	22,25
6			*Plurality*	89,5%	22,25
7	*1*	*Manhattan*	*Borda*	90%	13,3
8			*Most-Voted*	90%	13,3
9			*Plurality*	90%	13,3
10	1	*Canberra*	*Borda*	88,9%	33,65
11			*Most-Voted*	88,9%	33,65
12			*Plurality*	88,9%	33,65
13	*3*	*HVDM*	*Borda*	88,9%	17,5
14			*Most-Voted*	90%	13,3
15			*Plurality*	90%	13,3
16	*3*	*Euclidean*	*Borda*	89,5%	22,25
17			*Most-Voted*	89,7%	21,55
18			*Plurality*	89,7%	21,55
19	*3*	*Manhattan*	*Borda*	90%	13,3
20			*Most-Voted*	**90,1%**	**12,25**
21			*Plurality*	**90,1%**	**12,25**
22	*3*	*Canberra*	*Borda*	88,9%	33,65
23			*Most-Voted*	88,9%	34,75
24			*Plurality*	88,9%	34,75
25	*5*	*HVDM*	*Borda*	89,9%	17,5
26			*Most-Voted*	89,5%	23,85
27			*Plurality*	89,5%	23,9
28	*5*	*Euclidean*	*Borda*	89,5%	22,25
29			*Most-Voted*	89,5%	23
30			*Plurality*	89,6%	21,5
31	*5*	*Manhattan*	*Borda*	90%	13,3
32			*Most-Voted*	89,7%	19,9
33			*Plurality*	89,7%	20,75
34	*5*	*Canberra*	*Borda*	88,9%	33.65
35			*Most-Voted*	88,6%	35,95
36			*Plurality*	88,5%	37,6
37	*7*	*HVDM*	*Borda*	89,9%	17,5
38			*Most-Voted*	89,3%	29,05
39			*Plurality*	89,2%	28,65
40	*7*	*Euclidean*	*Borda*	89,5%	22.25
41			*Most-Voted*	89,5%	22,45
42			*Plurality*	89,5%	21,95
43	*7*	*Manhattan*	*Borda*	90%	13,3
44			*Most-Voted*	89,4%	26,7
45			*Plurality*	89,4%	27,7
46	*7*	*Canberra*	*Borda*	88,9%	33,65
47			*Most-Voted*	88,2%	40,85
48			*Plurality*	88,2%	40,3

we apply the Nemenyi test:

Table 5 – Nemenyi test for the *SatImage* dataset with k-IBL

	1	**2**	**3**	**4**	**...**	**45**	**46**	**47**	**48**
1	1	0,9	0,9	0,9	...	0,9	0,88	0,11	0,15
2	0,9	1	0,9	0,9	...	0,9	0,88	0,11	0,15
3	0,9	0,9	1	0,9	...	0,9	0,88	0,11	0,15
4	0,9	0,9	0,9	1	...	0,9	0,9	0,61	0,67
...	...	...	...	...	1	...	...	...	...
45	0,9	0,9	0,9	0,9	...	1	0,9	0,9	0,9
46	0,88	0,88	0,88	0,9	...	0,9	1	0,9	0,9
47	0,11	0,11	0,11	0,61	...	0,9	0,9	1	0,9
48	0,15	0,15	0,15	0,67	...	0,9	0,9	0,9	1

As presented on table 5, most of the comparisons prove the null hypothesis which means the differences between algorithms are insignificant. The lowest *p-value* was 0,007. It occured between configurations 20 and 47. That comparison proves the significance of the difference of these configuration as the *p-value* is below 0,05.

4.2.3 k-IBL with Feature Selection

As it is possible to observe in table 6, it was tested the I-Gain and Relief techniques for feature selection and it was compared without feature selection. Also, we change the number of features to observe the importance of the number of features in the results, this was varied in 5,10,15,20 and 25 features, taking into account that the total number of features is 36.

The parameters used for the k-IBL was:

- IB1 Algorithm
- Voting_Policy: most_voted
- K:3
- Distance: Euclidean

Table 6 – K-BIL with and without Feature selection

Comparison of with and withaout Feature Selection algorithm				
FS algorithm	**Mean accuracy**	**memory**	**time used (Seconds)**	**Features**
Without	93.96%	100.00%	6.18981055403856	36
I-Gain	94.20%	100.00%	5.85944673605263	25
Without	93.96%	100.00%	6.12852530102828	36
relief	93.98%	100.00%	5.78651045201696	25
Without	93.96%	100.00%	6.12852530102828	36
I-Gain	95.31%	100.00%	5.60775804996956	20
Without	93.96%	100.00%	6.10300320305396	36
relief	93.98%	100.00%	6.0190008750069	20
Without	93.96%	100.00%	6.26335457595997	36
I-Gain	95.63%	100.00%	5.80079487300827	15
Without	93.96%	100.00%	6.80995896403329	36
relief	93.96%	100.00%	6.43150216899812	15
Without	93.96%	100.00%	6.89471529997536	36
I-Gain	95.60%	100.00%	6.06465581798693	10
Without	93.96%	100.00%	5.83322073405725	36
relief	93.98%	100.00%	5.46420921894605	10
Without	93.96%	100.00%	8.48420922906371	36
I-Gain	**96.53%**	**100.00%**	**5.78967132302932**	**5**
Without	93.96%	100.00%	6.86058610095643	36
relief	94.81%	100.00%	5.54927924307413	**5**

For the satimage dataset as is showed in the table 3, the best result is 69.53% with a 5.789 seconds for the time used, this is achieved through feature selection with the $I - Gainmethod$ for 5 features.

- For the *satimage* dataset:

  Relief method:

Feature selection	**Eucledian mean accuracy**	**Time (Seconds)**	**Features**
Without	89,74%	23,31458372	36
relief	**89,36%**	**18,31615526**	**25**
relief	88,55%	17,7967225	20
relief	86,78%	19,52624644	15
relief	84,12%	17,48246941	10
relief	60,92%	15,35626497	5

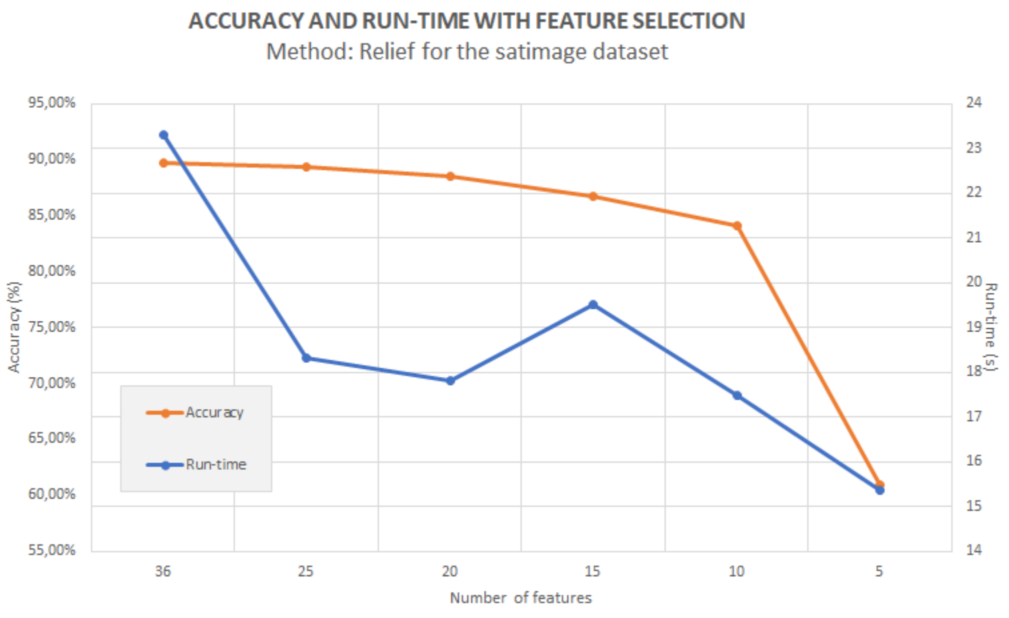

Figure 3 – Relief method for the satimage dataset

I-Gain method:

Feature selection	Eucledian mean accuracy	Time (Seconds)	Features
Without	89,74%	23,31458372	36
I-Gain	**89,42%**	**18,07782156**	**25**
I-Gain	89,04%	17,11965671	20
I-Gain	88,44%	17,71246383	15
I-Gain	87,48%	18,20444279	10
I-Gain	85,30%	16,11472205	5

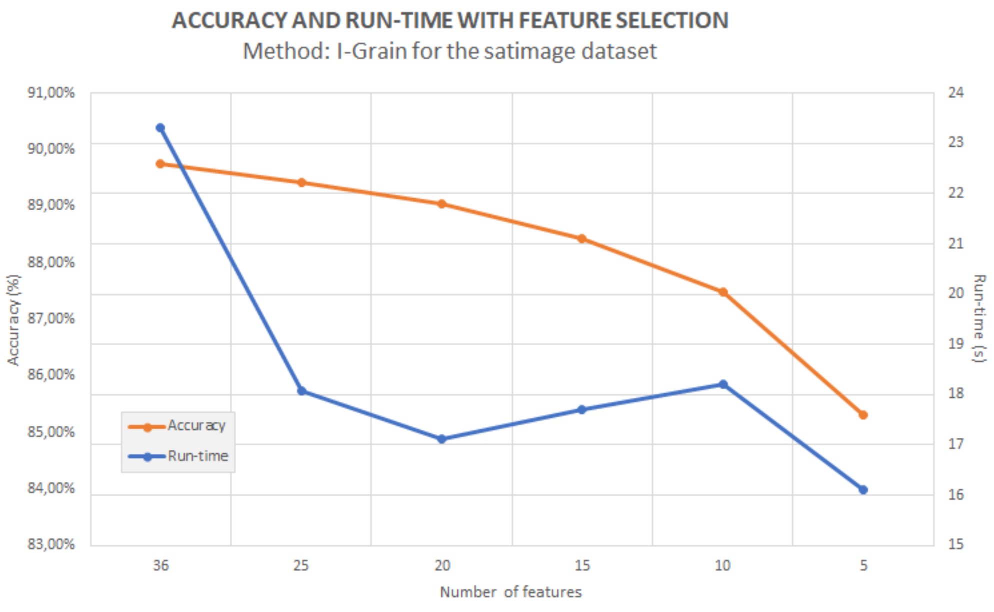

Figure 4 – I-Gain method for the satimage dataset

4.3 Discussion

4.3.1 Which is the best value of K at each dataset?

The best values of k we observed on our datasets, were 5 and 7. In the first datase we have seen a significant improvement when increasing the k. In the second one however, there is no clear tendency that it is improving. The best k value was 3. Even some cases prove that the accuracy is worse when we increase the k. It may be because of the dataset's complexity. To summarize, it is clear that increasing k has a crucial role in improving the algorithms performance. However not every dataset fits that description. In our implementation it did not have influence over the execution times of the algorithms. It is also important to choose the right voting policy to determine the winning label of the nearest neighbours.

4.3.2 Which IBL algorithm is the best one at each dataset?

We have run evalution algorithms for each dataset, For Hypothyroid IB1 and IB2 got very good results, both accuracy and variance was more than acceptable, execution time for IB1 was a little longer but the accuracy was better than IB2, From results, IB3 got worse results, only component that IB3 beat all other algorithms was memory, that is IB3's main trait.

Regarding SatImage dataset, from the results we can deduce that IB1 and IB2 got the best performance like in the previous case. Accuracies are very high and variance is still very low. IB3 is still out of competition because it got very low memory and high execution time. So between IB1 and IB2 we still chose IB1 because of high accuracy and very low variance.

4.3.3 Did you find useful the use of a voting scheme for deciding the solution of the query q?

In the case of $k > 1$, we found the voting protocols useful in deciding the winning label of the compared datapoint. The voting policies had a significant influence on the performance of the algorithms. In the first dataset, *Borda* voting policy was the worst in all of the configurations. *Most-voted* and *Plurality* were nearly the same and resulted in higher accuracy than the *Borda*. In case of the execution times, they were very similar. In the second dataset however, the results were not that clear and in some cases *Borda* protocol gave better results. Nevertheless, it is safe to say that *Most-voted* and *Plurality* protocols are the best, because in the second dataset the differences were not that significant. In the first one however, yes.

4.3.4 Which is the best similarity function for the k-IBL?

The best similarity function in our case was the *Canberra* and *Manhattan*. In both datasets, they were very close to each other in terms of the performance. The *Manhattan* metric however is faster in execution. That could be the reason to choose it over the *Canberra*.

4.3.5 Did you find differences in performance among the k-IBL and the selection k-IBL?

For the hypothyroidism data set it was observed in the table 3, that if we do not use the feature selection, our accuracy can be the maximum, but its time is also the maximum, on the other hand if we implement the feature selection, the time is reduced even precision is decreased. For that reason, the correct thing to do would be to use a configuration where you do not have to sacrifice much accuracy but, on the contrary, time is decreased. One of the possible configuration is used the method $I - Gain$ for feature selection with 25 features, with this configuration we can obtain 89.42% in the accuracy and 18.077 seconds, this means that the time is reduced 5.2367 seconds or in other words was improved 22,4%, in counterpart the accuracy decreased 0.23%.

For the satimage data set as is observed in the table 6 it was observed that the accuracy and the time could be improve when was used the feature selection, in this case this improvement was achieved thanks to use the I-Gain method with 5 features, as result the accuracy in the dataset managed to reach a 96.53% in 5.789 seconds, in other words the accuracy improve over 2.73% and the time was reduced in 0.779 seconds.

4.3.6 According to the data sets chosen, which feature selection method provides you more advice for knowing the underlying information in the data set? (look at the relative weight to each feature).

According our results in the table 7, in the hypothyroid dataset the Relief method give scores more high for the features than the I-Gain. Also for Relief the feature with more importance is the 19 $'T3\_measured'$ while in the I-Gain method is feature 17 $'TSH\_measured'$.

Table 7 – Scores of the I-Gain and Relief method in feature selection for Hypothyroid

Scores for hypothyroid dataset features			
	Features	**I-Gain**	**Relief**
1	’age’	0.0130091444266407	0.03636686
2	’sex’	0.0311109165536259	0.30339201
3	’on_thyroxine’	0.0431339678456506	0.2980505
4	’query_on_thyroxine’	0.0345563122549162	0.15603922
5	’on_antithyroid_medication’	0.0355977441167643	0.03267917
6	’sick’	0.0386840386272742	0.02435202
7	’pregnant’	0.0357013641215171	0.07300125
8	’thyroid_surgery’	0.0360767353937328	0.02395494
9	’I131_treatment’	0.0336720516886706	0.02976886
10	’query_hypothyroid’	0.03277105319121	0.03881331
11	’query_hyperthyroid’	0.0316658180546765	0.10514852
12	’lithium’	0.035755388204449	0.07821585
13	’goitre’	0.0353268298554872	0.00997517
14	’tumor’	0.03478372703303	0.01780273
15	’hypopituitary’	0.00235360902843507	0.05123276
16	’psych’	0.0372861144898693	0
17	**’TSH_measured’**	**0.0429337663558931**	0.07997609
18	’TSH’	0.230945049460737	0.14106723
19	**’T3_measured’**	0.0315305779504527	**0.88556323**
20	’T3’	0.03630859207682	0.21028597
21	’TT4_measured’	0.0477062843594864	0.09877779
22	’TT4’	0.0361867603596147	0.61262503
23	’T4U_measured’	0	0.11044633
24	’T4U’	0.00454154582135713	0.35037363
25	’FTI_measured’	0	0.11158327
26	’FTI’	0.036366862509186	0.65166401
27	’TBG_measured’	0.0629770001175045	0
28	’TBG’	0.00112337507793714	0.31303395
29	’referral_source’	0.0411676757400574	0
	Sum of total scores	1.083272304715	4.8441897

As is showed in the table 8 of Satimage dataset, it is possible to observed that for the I-Gain method the algorithm give us more score than the Relief method. Also, we can note that for the I-Gain the mos important feature is the $10.'a20'$, while for the Relief the most scored was the $29.'a29'$.

Table 8 – Scores of the I-Gain and Relief method in feature selection for Satimage

Scores for satimage dataset features			
	Features	**I-Gain**	**Relief**
1	a1	0.59851257	0.10344158
2	a2	0.5625994	0.08704191
3	a3	0.49926547	0.06720999
4	a4	0.6085746	0.08388752
5	a5	0.64724628	0.12208868
6	a6	0.59337043	0.09391883
7	a7	0.50912207	0.06340447
8	a8	0.63719826	0.08320472
9	a9	0.58758962	0.11207946
10	a10	0.56074372	0.08069772
11	a11	0.47869341	0.05948924
12	a12	0.59710687	0.07953566
13	a13	0.64717531	0.11772575
14	a14	0.62899618	0.08727048
15	a15	0.55690787	0.06878187
16	a16	0.70026047	0.10452375
17	a17	0.755317	0.12247018
18	a18	0.73019875	0.10036176
19	a19	0.59591646	0.07687616
20	**a20**	**0.74353961**	0.1040438
21	a21	0.69433719	0.11593018
22	a22	0.63851646	0.09179631
23	a23	0.52125608	0.05801426
24	a24	0.67726394	0.08376256
25	a25	0.58582501	0.11251084
26	a26	0.53717277	0.09213913
27	a27	0.48159425	0.06340091
28	a28	0.5945906	0.09058321
29	**a29**	0.63538507	**0.12568848**
30	a30	0.58804564	0.10324839
31	a31	0.51232062	0.06086269
32	a32	0.62738812	0.09122713
33	a33	0.61921258	0.10335728
34	a34	0.56606869	0.08851662
35	a35	0.45973646	0.05980933
36	a36	0.58727512	0.07399061
Sum of total scores		**21.56432295**	**3.23289146**

4.3.7 Do you think it will be much better to weight the features rather than removing some of them in the similarity function?

To answer this, we have two different approaches. The first, is where we have many features and at the same time, we don't have references in which features are the most important than others, in that point, it is better to bring scores for the features and in based on that extract the features that we need for our system.

The second case is where we have knowledge based on which datasets we don't need, in this case, it is most useful to remove it.

Acording to (9), the feature weighting is more appropriate for problems where the features vary in their relevance. Feature selection algorithms perform best when the features used to describe instances are either highly correlated with the class label or completely irrele- vant (Wettschereck et al., 1997).

5 Conclusions

- From all the results, IB1 got the best accuracy in all tries, IB2 got almost the same accuracy as IB1 but the execution time and memory was much better in all cases. For IB3 we got the least memory but the most execution time. It also could not beat neither Algorithm. So we can deduce that IB1 was the best algorithm with accuracy. By considering memory, variance, time and memory we used our formula to define the best IB algorithm and IB1 was the best in both databases.

- The k-IBL is a useful algorithm and it proved to be better than the regular IBL algorithms. When the k value was increased over 1, we have seen improvement of the results. In both datasets, the k-IBL was better in terms of performance over the regular IBLs. It also was not any slower in execution than the regular IBL algorithm. Summarizing, the k-IBL is a good alternative to the regular IBL, but we have to be careful with selecting the parameters such as voting policy. It can affect the performance of the algorithms as it has influence of choosing the right label for the datapoints, which is the most important part of the nearest neighbours version of the IBL.

- As has been previously discussed in the sections above, applying *Feature selection* results in considering less features, which can in some cases improve the accuracy because noisy data is ignored, while in other cases it can decrease the accuracy due to a loss of information. Because the machine learning algorithms that were used are already very robust in terms of their capability to handle noisy data, one would not expect a great improvement in the accuracy. Indeed, the only case where feature-selection proved to have a noticeably positive effect on the accuracy was when we used the Relief method on the Hypothyroid dataset; for the rest of the analysed cases, the accuracy remained almost the same or it decreased. From the analysis we inferred a great dependency of the effectiveness of feature-selection on the specific dataset that we are studying, which stems from the fact that some datasets contain some features that have less impact on the classification, while others are more relevant. On the other hand, the time used for the computations decreased in all of the cases as expected, since canceling out some features implies decreasing the amount of computations required by the classifier.

References

[1] David W. Aha. *Lazy Learning.* Washington D.C.: Springer, June 2013. ISBN 9780792345848.

[2] Cunningham, P., Kathirgamanathan, B., & Delany, S. (2021). Feature Selection Tutorial with Python Examples. ArXiv, abs/2106.06437.

[3] Lazy Learning: Springer US, pages: '571–572', Boston, MA, 2010 - Webb, Geoffrey I.

[4] T-test as a parametric statistic - Tae Kyun Kimcorresponding

[5] Interdisciplinary Description of Complex Systems : INDECS , Vol. 16 No. 3-B, 2018. - Aleksandar Hatzivelkos

[6] Shahid, Rizwan & Bertazzon, Stefania & Knudtson, Merril & Ghali, William. (2009). Comparison of distance measures in spatial analytical modeling for health service planning. *BMC health services research.* 9. 200. 10.1186/1472-6963-9-200.

[7] Canberra distance on ranked lists - Giuseppe Jurman, S. Riccadonna

[8] Improved Heterogeneous Distance Functions - D. Randall Wilson, Tony R. Martinez

[9] Wettschereck, D., Aha, D.W., Mohri, T., 1997. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. **Artif. Intell. Rev. 11 (1–5)**, 273–314.