Classification with k-NN, k+NN, Ria and Riona

A Preprint

Bogdan Krzysztof Jastrzębski
Faculty of Mathematics and Information Science
Warsaw University of Technology
Warsaw
bogdan.jastrzebski@dokt.pw.edu.pl

June 30, 2023

Abstract

The following is a technical report on the implementation of k-NN, k+NN, Ria and Riona classification algorithms.

Keywords Classification · k-NN · k+NN · Ria · Riona

1 Introduction

The project aim was to implement and evaluate the performance of classification algorithms:

- k-Nearest Neighbours (k-NN);
- k+Nearest Neighbours (k+NN);
- Ria;
- Riona;

The language of implementation is C++. We evaluated the aforementioned classification algorithms on one synthetic dataset and three real-world datasets from a provided repository:

- "iris.arff"
- "tae.arff"
- "zoo.arff"

and example dataset from the Riona paper Góra and Wojna [2002].

2 Implementation Focus

Since we are implementing the classifiers in C++, the implementation focus was to create an efficient script for serial use, such that we can repetitively classify new data in an online fashion. Because of that:

- implementation is mostly stack-based, or semi-stack based;
- we require data format to be known a priori during compilation time, e.g., types of data entries, which allows for automatic optimization of some parts of code, that otherwise would be dynamic and slow;

Moreover, the implementation is very generic, allowing for the implementation and use of different metrics.

The implementation does not focus on:

- data parsing;
- classifier evaluation;

3 Assumptions

Our assumptions are as follows:

- dataset is complete, i.e., there is no missing data;
- data can be loaded into memory all at once;
- there are two types of features:
  - categorical, represented as uint8_t;
  - real-valued, represented as float;
- no categorical attribute can have more than 256 different values (due to data format);
- input files are in space-separated values format (similar to standard comma-separated values format (CSV) with space instaed of comma);
- data has no headers;
- all data preprocessing is done by the user, either in C++ or before use in our script;

The input files we use have been transformed separately, we provide them with other project files.

4 Code Design

The code follows a quasi-functional design. The classes implement either:

- objects (intermediate values);
- transformations (functions);

Our code separates data from functionality. Objects do not implement methods other than constructors, they store data. Transformations implement a constructor for initialization and the operator(). We modify data "in place" only for values created inside a function. If a function does not have to be first-category, it is a simple C++ template function.

List of key classes:

- Observation (Object): stores a pointer to an observation structure;
- Metric (Transformation): abstract class for writing metric spaces. The operator() measures the distance between two variables. Subclasses:
  - ValueDifference: value difference metric, for a categorical variable;
  - AbsoluteDiffernce: absolute difference, for a numerical variable;
  - Max: composite max metric for tuples;
  - Manhattan: composite Manhattan metric for tuples;
- Rule (Transformation): implementation of a rule.

Composite metrics are metrics for tuples, that calculate 1-dimensional metrics (ValueDiffernce or AbsoluteDiffernce) for all entries and aggregate them. The Max takes the maximum distances for all features. The Manhattan adds distances for all features.

List of key functions:

- knn;
- ria;
- riona;

The template functions implement three classification methods. They return the class variable (uint8_t).

5 User Guide

The code allows repurposing code for new datasets. The data structure has to be implemented inside the code.

1. derive all data loader capabilities as provided in implementation files;
2. define metrics to use for your method;
3. calculate result;

The script can be more or less verbose. If you are interested in prediction only, define PRINT_LOG to be false, or true otherwise.

6 Result Reproduction

To reproduce results, simply run four scripts:

- zoo
- tae
- iris
- example

They print results in JSON format.

7 Versions of Implementation

The implementation is presented in two versions:

- original design: the proper version, that is clean and outputs only the prediction;
- design with logging: the same code with JSON logs, that allows for extracting metadata from the algorithm, for our analysis of its correctness;

We regard the first implementation as the default one.

8 Description of the Input and Output Data

The form of the input data is assumed to be:

- space-separated value file;
  - observations are written row-wise;
  - values in a row are separated by space;
- first column being class variable;
- no header names;
- all values are numerical;
  - categorical values should be labelled from 0 to the number of unique values;
  - numerical values are written in standard form;

Whether a column is categorical or real-valued is decided in the code, and is known a priori during compilation time for efficiency.

8.1 Example Dataset

A simple dataset from the Riona paper to show how Ria works.

(1) file_name : example.csv
(2) number_of_observations : 8
(3) number_of_attributes : 4
(4) decision_attribute_position : 1
(5) attributes :
  (I) 0 :
    (A) type: categorical
    (B) stats:
      (i) counts:
        (a) 1 : 4
        (b) 0 : 4
  (II) 1 :
    (A) type: numerical
    (B) stats:
      (i) min : -1.3858697343671664
      (ii) max : 1.6378460497066514
  (III) 2 :
    (A) type: numerical
    (B) stats:
      (i) min : -1.0502100630210074
      (ii) max : 2.450490147049017
  (IV) 3 :
    (A) type: categorical
    (B) stats:
      (i) counts:
        (a) 1 : 4
        (b) 0 : 4
      (ii) table:
        (a) 0 : [2, 2]
        (b) 1 : [2, 2]
  (V) 4 :
    (A) type: categorical
    (B) stats:
      (i) counts:
        (a) 3 : 3
        (b) 1 : 2
        (c) 2 : 2
        (d) 0 : 1
      (ii) table:
        (a) 0 : [1, 0]
        (b) 1 : [0, 2]
        (c) 2 : [1, 1]
        (d) 3 : [2, 1]

8.2 Iris Dataset

Dataset with only numerical attributes.

(1) file_name : iris.csv
(2) number_of_observations : 150
(3) number_of_attributes : 4
(4) decision_attribute_position : 1
(5) attributes :
  - (I) 0 :
    - (A) type: categorical
    - (B) stats:
      - (i) counts:
        - (a) 0 : 50
        - (b) 2 : 50
        - (c) 1 : 50
  - (II) 1 :
    - (A) type: numerical
    - (B) stats:
      - (i) min : -1.870024133847019
      - (ii) max : 2.4920192021244283
  - (III) 2 :
    - (A) type: numerical
    - (B) stats:
      - (i) min : -2.4389872524918412
      - (ii) max : 3.114683910677436
  - (IV) 3 :
    - (A) type: numerical
    - (B) stats:
      - (i) min : -1.5687352207168412
      - (ii) max : 1.7863413146490468
  - (V) 4 :
    - (A) type: numerical
    - (B) stats:
      - (i) min : -1.4444496972795187
      - (ii) max : 1.7109015831854497

8.3 tae Dataset

Dataset with both categorical and numerical attributes.

(1) file_name : tae.csv
(2) number_of_observations : 151
(3) number_of_attributes : 5
(4) decision_attribute_position : 1
(5) attributes :
  - (I) 0 :
    - (A) type: categorical
    - (B) stats:
      - (i) counts:
        - (a) 2 : 52
        - (b) 1 : 50
        - (c) 0 : 49
  - (II) 1 :
    - (A) type: categorical
    - (B) stats:
      - (i) counts:
        - (a) 1 : 122
        - (b) 0 : 29
      - (ii) table:
        - (a) 0 : [5, 6, 18]
        - (b) 1 : [44, 44, 34]
  - (III) 2 :
    - (A) type: numerical
    - (B) stats:
      - (i) min : -1.858316168217688
      - (ii) max : 1.6694668562039474
  - (IV) 3 :
    - (A) type: numerical
    - (B) stats:
      - (i) min : -1.0150476950160767
      - (ii) max : 2.5560660502641555
  - (V) 4 :
    - (A) type: categorical
    - (B) stats:
      - (i) counts:
        - (a) 1 : 128
        - (b) 0 : 23
      - (ii) table:
        - (a) 0 : [2, 6, 15]
        - (b) 1 : [47, 44, 37]
  - (VI) 5 :
    - (A) type: numerical
    - (B) stats:
      - (i) min : -1.9350683456068614
      - (ii) max : 2.9672765736363

8.4 Zoo Dataset

All categorical dataset.

(1) file_name : zoo.csv
(2) number_of_observations : 101
(3) number_of_attributes : 16
(4) decision_attribute_position : 1
(5) attributes :
    (I) 0 :
        (A) type: categorical
        (B) stats:
            (i) counts:
                (a) 0 : 41
                (b) 1 : 20
                (c) 3 : 13
                (d) 6 : 10
                (e) 5 : 8
                (f) 2 : 5
                (g) 4 : 4
    (II) 1 :
        (A) type: categorical
        (B) stats:
            (i) counts:
                (a) 0 : 58
                (b) 1 : 43
            (ii) table:
                (a) 0 : [2, 20, 5, 13, 4, 4, 10]
                (b) 1 : [39, 0, 0, 0, 0, 4, 0]
    (III) 10 :
        (A) type: categorical
        (B) stats:
            (i) counts:
                (a) 1 : 80
                (b) 0 : 21
            (ii) table:
                (a) 0 : [0, 0, 1, 13, 0, 0, 7]
                (b) 1 : [41, 20, 4, 0, 4, 8, 3]
    (IV) 11 :
        (A) type: categorical
        (B) stats:
            (i) counts:
                (a) 0 : 93
                (b) 1 : 8
            (ii) table:
                (a) 0 : [41, 20, 3, 12, 3, 6, 8]
                (b) 1 : [0, 0, 2, 1, 1, 2, 2]
    (V) 12 :
        (A) type: categorical
        (B) stats:
            (i) counts:
                (a) 0 : 84
                (b) 1 : 17

- (ii) table:
  - (a) 0 : [37, 20, 5, 0, 4, 8, 10]
  - (b) 1 : [4, 0, 0, 13, 0, 0, 0]

(VI) 13 :
- (A) type: categorical
- (B) stats:
  - (i) counts:
    - (a) 2 : 38
    - (b) 1 : 27
    - (c) 0 : 23
    - (d) 4 : 10
    - (e) 5 : 2
    - (f) 3 : 1
  - (ii) table:
    - (a) 0 : [3, 0, 3, 13, 0, 0, 4]
    - (b) 1 : [7, 20, 0, 0, 0, 0, 0]
    - (c) 2 : [31, 0, 2, 0, 4, 0, 1]
    - (d) 3 : [0, 0, 0, 0, 0, 0, 1]
    - (e) 4 : [0, 0, 0, 0, 0, 8, 2]
    - (f) 5 : [0, 0, 0, 0, 0, 0, 2]

(VII) 14 :
- (A) type: categorical
- (B) stats:
  - (i) counts:
    - (a) 1 : 75
    - (b) 0 : 26
  - (ii) table:
    - (a) 0 : [6, 0, 0, 0, 3, 8, 9]
    - (b) 1 : [35, 20, 5, 13, 1, 0, 1]

(VIII) 15 :
- (A) type: categorical
- (B) stats:
  - (i) counts:
    - (a) 0 : 88
    - (b) 1 : 13
  - (ii) table:
    - (a) 0 : [33, 17, 5, 12, 4, 7, 10]
    - (b) 1 : [8, 3, 0, 1, 0, 1, 0]

(IX) 16 :
- (A) type: categorical
- (B) stats:
  - (i) counts:
    - (a) 0 : 57
    - (b) 1 : 44
  - (ii) table:
    - (a) 0 : [9, 14, 4, 9, 4, 8, 9]
    - (b) 1 : [32, 6, 1, 4, 0, 0, 1]

(X) 2 :
- (A) type: categorical
- (B) stats:
  - (i) counts:
    - (a) 0 : 81
    - (b) 1 : 20
  - (ii) table:

(a) 0 : [41, 0, 5, 13, 4, 8, 10]
(b) 1 : [0, 20, 0, 0, 0, 0, 0]

(XI) 3 :
  (A) type: categorical
  (B) stats:
    (i) counts:
      (a) 1 : 59
      (b) 0 : 42
    (ii) table:
      (a) 0 : [40, 0, 1, 0, 0, 0, 1]
      (b) 1 : [1, 20, 4, 13, 4, 8, 9]

(XII) 4 :
  (A) type: categorical
  (B) stats:
    (i) counts:
      (a) 0 : 60
      (b) 1 : 41
    (ii) table:
      (a) 0 : [0, 20, 5, 13, 4, 8, 10]
      (b) 1 : [41, 0, 0, 0, 0, 0, 0]

(XIII) 5 :
  (A) type: categorical
  (B) stats:
    (i) counts:
      (a) 0 : 77
      (b) 1 : 24
    (ii) table:
      (a) 0 : [39, 4, 5, 13, 4, 2, 10]
      (b) 1 : [2, 16, 0, 0, 0, 6, 0]

(XIV) 6 :
  (A) type: categorical
  (B) stats:
    (i) counts:
      (a) 0 : 65
      (b) 1 : 36
    (ii) table:
      (a) 0 : [35, 14, 4, 0, 0, 8, 4]
      (b) 1 : [6, 6, 1, 13, 4, 0, 6]

(XV) 7 :
  (A) type: categorical
  (B) stats:
    (i) counts:
      (a) 1 : 56
      (b) 0 : 45
    (ii) table:
      (a) 0 : [19, 11, 1, 4, 1, 7, 2]
      (b) 1 : [22, 9, 4, 9, 3, 1, 8]

(XVI) 8 :
  (A) type: categorical
  (B) stats:
    (i) counts:
      (a) 1 : 61
      (b) 0 : 40
    (ii) table:

        (a) 0 : [1, 20, 1, 0, 0, 8, 10]
        (b) 1 : [40, 0, 4, 13, 4, 0, 0]

(XVII) 9 :
  (A) type: categorical
  (B) stats:
    (i) counts:
      (a) 1 : 83
      (b) 0 : 18
    (ii) table:
      (a) 0 : [0, 0, 0, 0, 0, 8, 10]
      (b) 1 : [41, 20, 5, 13, 4, 0, 0]

8.5 Output Data

The output is printed in JSON format. We include a notebook written in Python with an analysis of the output, which shows, how it can be loaded and processed.

id	Age	Weight	Sex	Blood Group	Class
0	35	90	M	A	Sick
1	40	65	F	AB	Sick
2	45	68	F	AB	Healthy
3	40	70	M	AB	Healthy
4	45	75	M	B	Sick
5	35	70	F	B	Healthy
6	45	70	M	0	Healthy
test	50	72	F	A	?

Table 1: Example dataset.

id	k±NN distance	RIA-consistent
0	5.54422	True
1	4.32934	False
2	2.90135	True
3	3.6292	True
4	2.427989	True
5	4.303772	True
6	3.287961	True

Table 2: Consistency of the Ria rules and distances for k±NN on the Example dataset.

9 Results

9.1 Dataset Preprocessing and Evaluation

Datasets have been preprocessed by:

- changing categorical values into consecutive ints from 0 to the number of unique values;
- normalizing numerical values;

Normalization is important, as we intentionally do not normalize the data internally. This way, the user is able to control the weight of variables also by rescaling the input. We assume an equal impact of all variables, and because of that, we normalize the data.

Evaluation is done by leave-one-out approach. k used for k-NN, k+NN and Riona is 5.

9.2 Synthetic Dataset

Tab. 1 shows an example dataset from the Riona paper Góra and Wojna [2002]. There are seven examples in the training dataset and we have one test example. In the paper, the authors show that for the Ria algorithm, the first training example creates a consistent rule, while the second is inconsistent. Results of consistency for Ria and Riona algorithms, as well as distance calculated for k±NN and Riona, are shown in Tab. 2. As we can see, the first rule is consistent, while the second is inconsistent as in the paper. Distances are calculated for normalized data, yet they seem to be a reasonable measure of differences between observations.

9.3 Iris Dataset

Figure 1 shows observations of the Iris dataset, presented in the space of two main principal components from the PCA. Figures 2, 3, 4 and 5 show predicted labels by k-NN, k+NN, Ria and Riona algorithms, respectively. Numerical results are presented in Tab. 3. k-NN, k+NN and Riona obtained the same results. Only Ria performed worse but still predicted the majority of the labels correctly.

measure	k-NN	k+NN	RIA	RIONA
accuracy	0.953	0.953	0.880	0.953
balanced accuracy	0.953	0.953	0.880	0.953

Table 3: Iris results: accuracy and balanced accuracy.

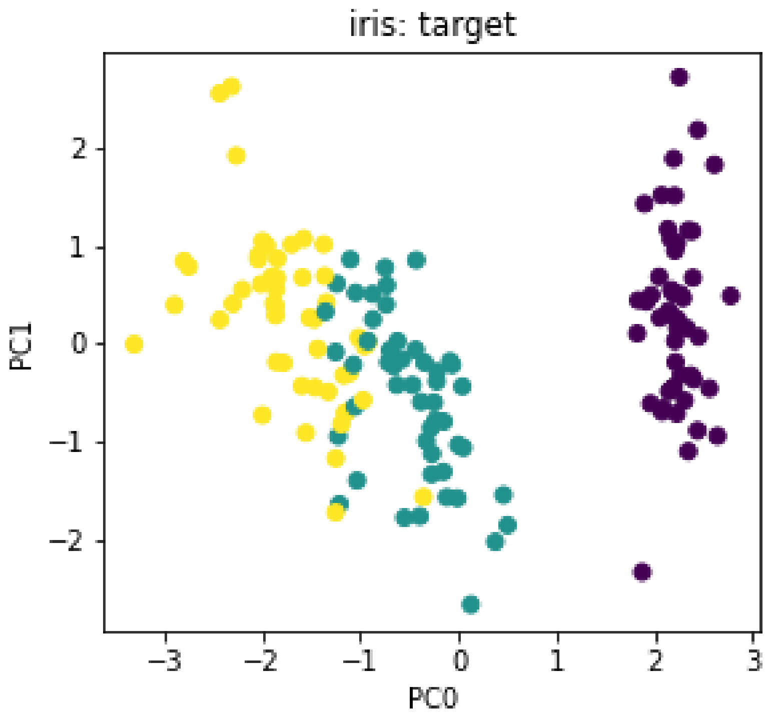

Figure 1: True class of observations of the Iris dataset: PCA plot.

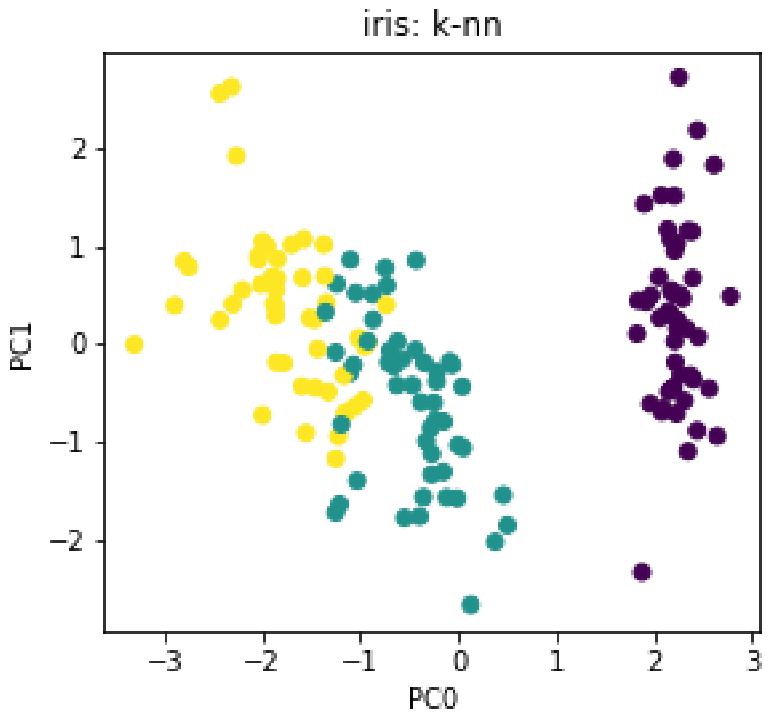

Figure 2: k-NN algorithm prediction on the Iris dataset: PCA plot.

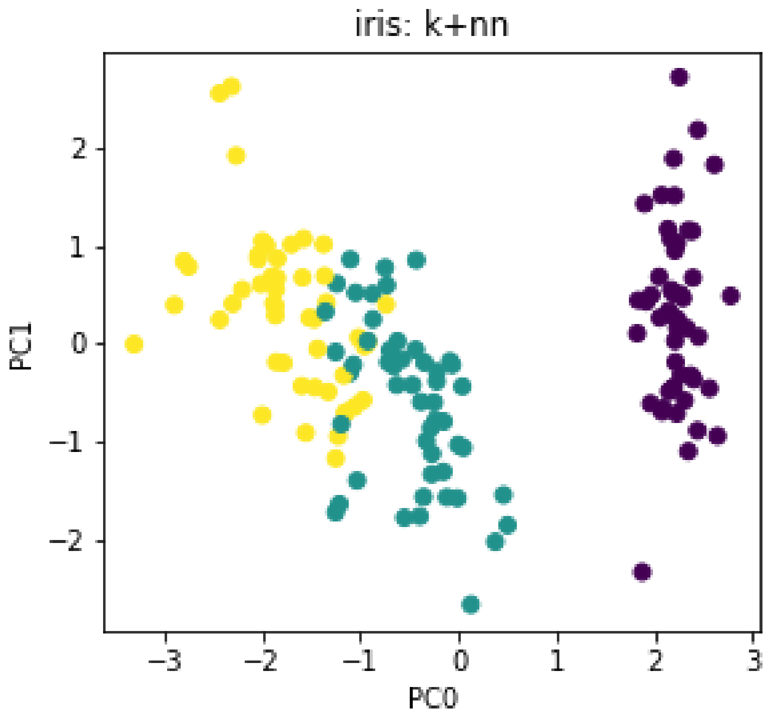

Figure 3: k+NN algorithm prediction on the Iris dataset: PCA plot.

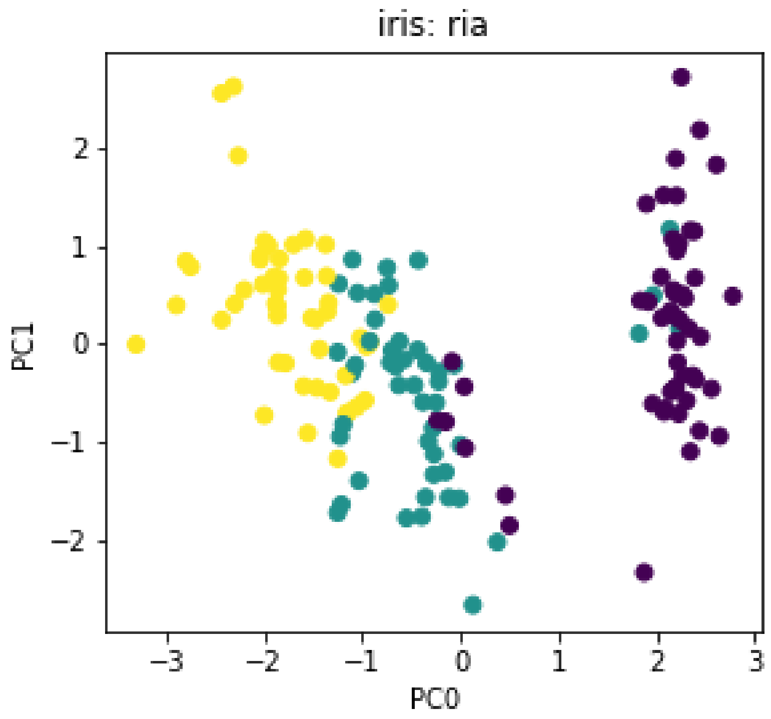

Figure 4: Ria algorithm prediction on the Iris dataset: PCA plot.

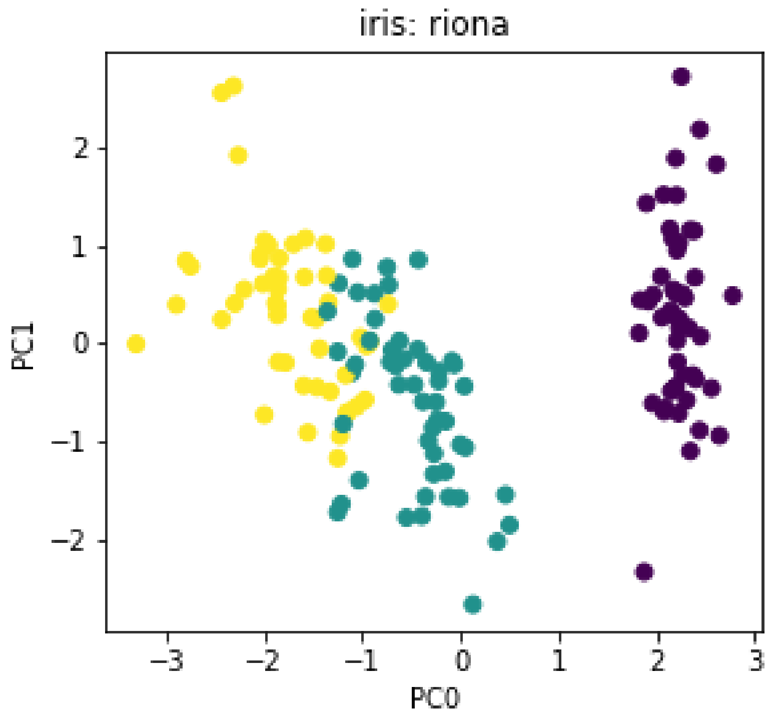

Figure 5: Riona algorithm prediction on the Iris dataset: PCA plot.

Figures 6 and 7 show neighbourhoods (yellow markers) of a particular example (shown in red) for the k-NN and k+NN algorithms, respectively. As we can see, the neighbourhood is correctly close to the example. The neighbourhoods for the two algorithms are almost always the same if our attributes feature randomized real-valued numerical data.

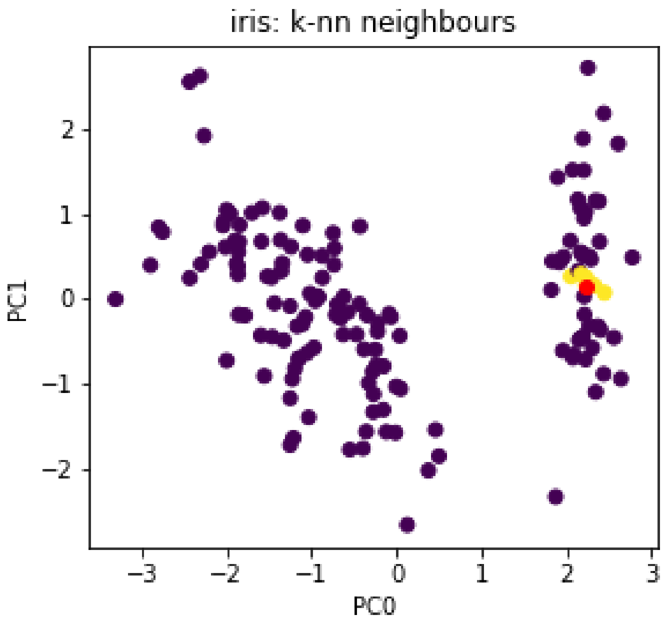

Figure 6: k-NN neighbours of a particular example in the Iris dataset.

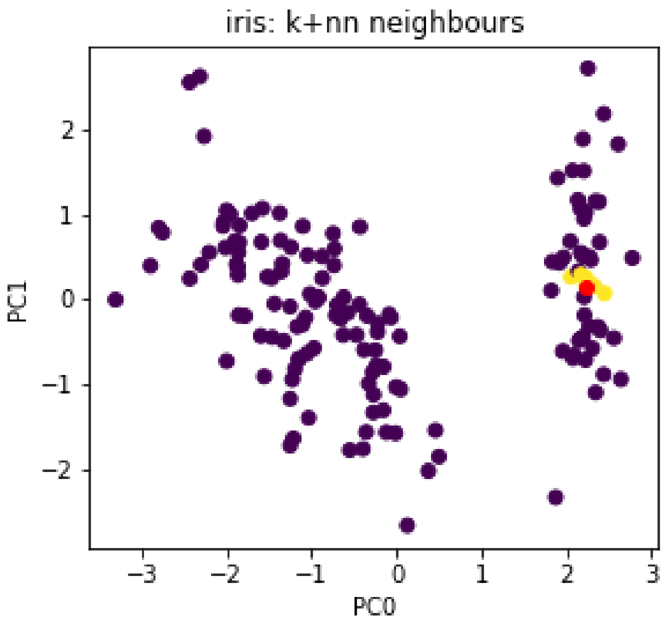

Figure 7: k+NN neighbours of a particular example in the Iris dataset.

Figure 8 shows distances calculated while predicting the label of the red test example. As we can see, the distance is calculated correctly.

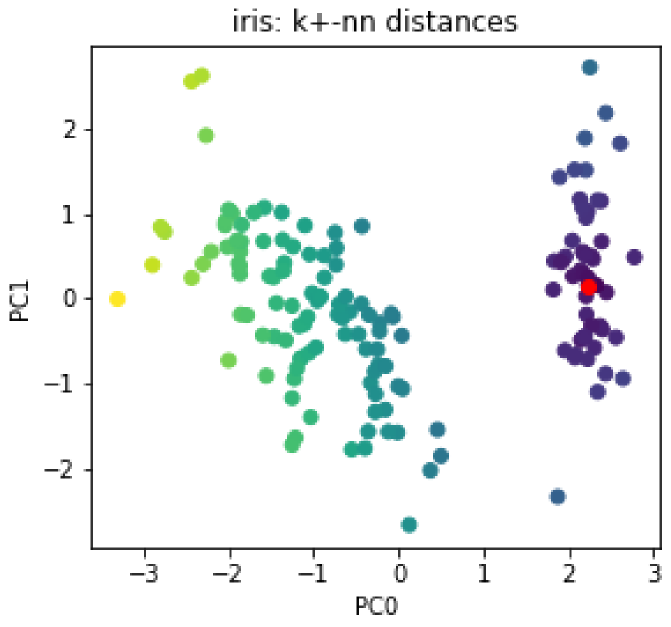

Figure 8: Calculated distance from a test observation (red) in the Iris dataset.

Figure 9 shows, which observations produce a consistent rule. As we can see, the majority of the observations are decisive for the Ria algorithm.

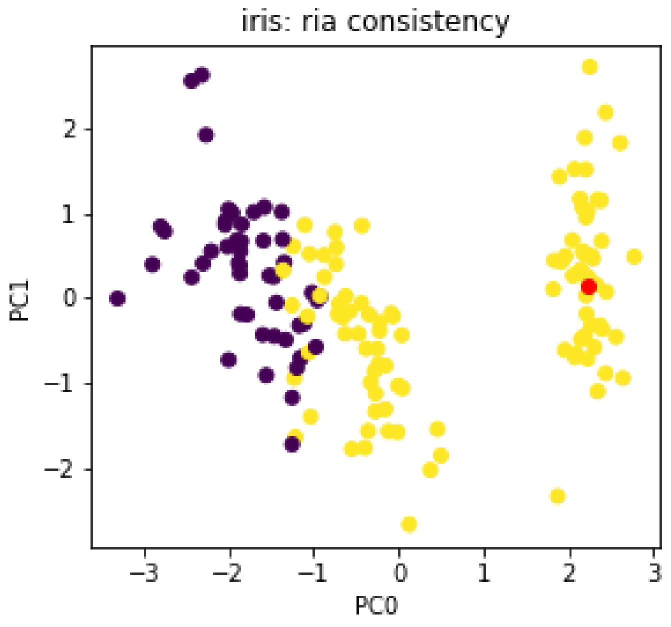

Figure 9: Consistent (yellow) observations/rules for a test observation (red) in the Iris dataset.

9.4 Tae Dataset

Figure 10 shows observations of the Tae dataset, presented in the space of two main principal components from the PCA. Figures 11, 12, 13 and 14 show predicted labels by k-NN, k+NN, Ria and Riona algorithms, respectively. Numerical results are presented in Tab. 4. Now the situation is different. Ria and Riona algorithms are much better than k-NN and k+NN by a large difference. The dataset has a large Bayes error, as compared to the Iris dataset.

measure	k-NN	k+NN	RIA	RIONA
accuracy	0.517	0.523	0.656	0.656
balanced accuracy	0.517	0.523	0.657	0.657

Table 4: Tae results: accuracy and balanced accuracy.

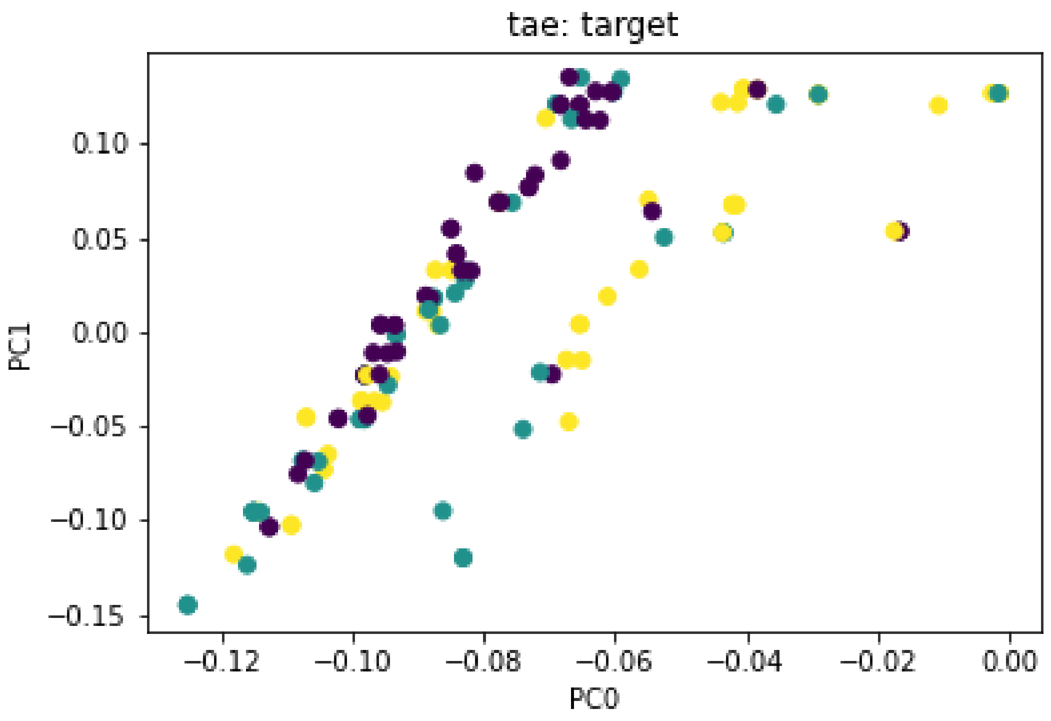

Figure 10: True class of observations of the Tae dataset: PCA plot.

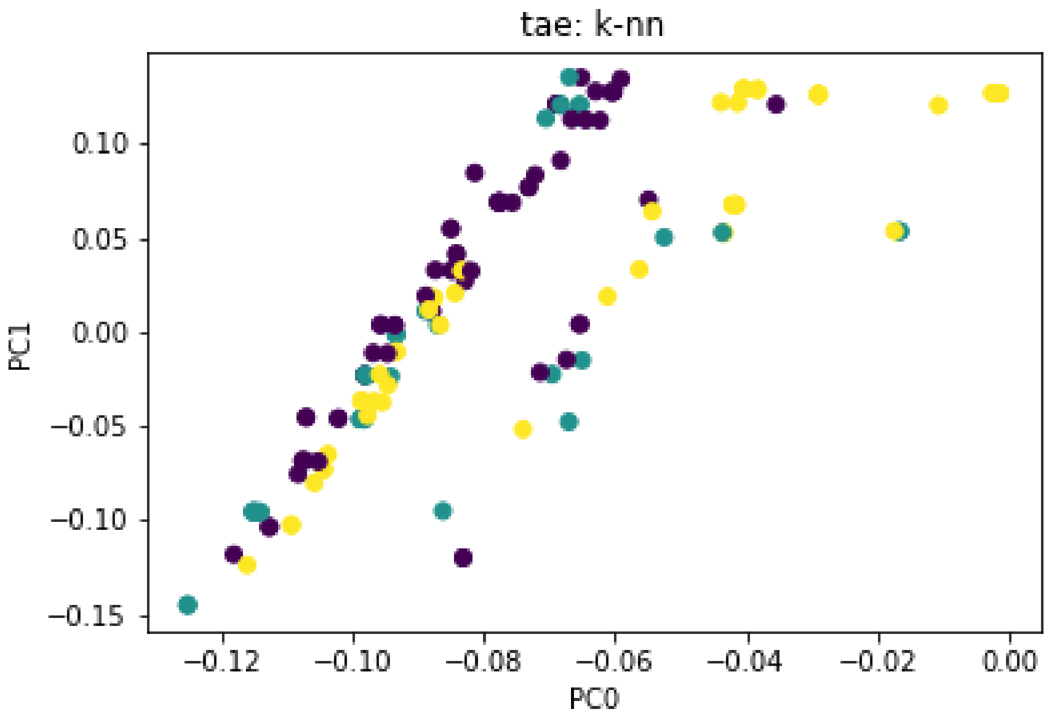

Figure 11: k-NN algorithm prediction on the Tae dataset: PCA plot.

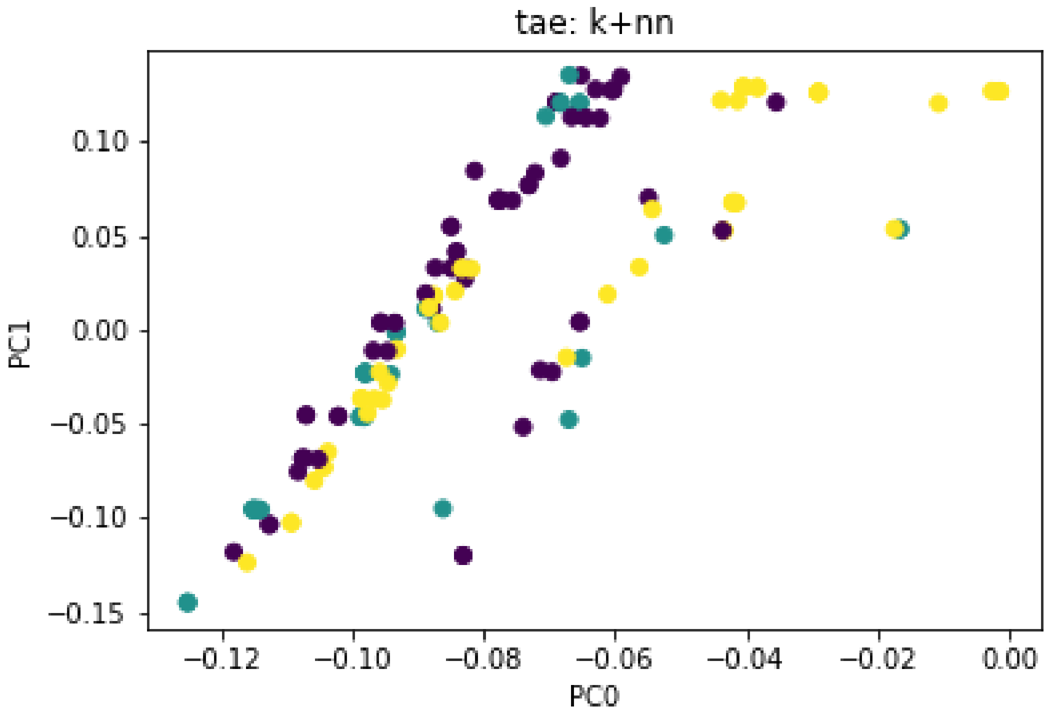

Figure 12: k+NN algorithm prediction on the Tae dataset: PCA plot.

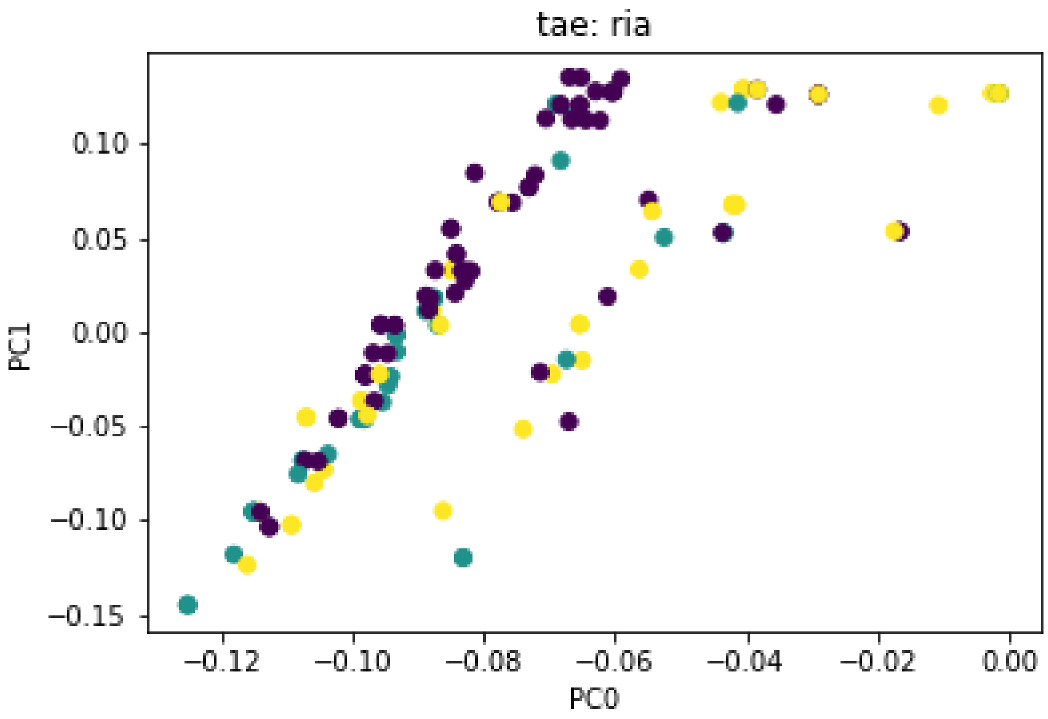

Figure 13: Ria algorithm prediction on the Tae dataset: PCA plot.

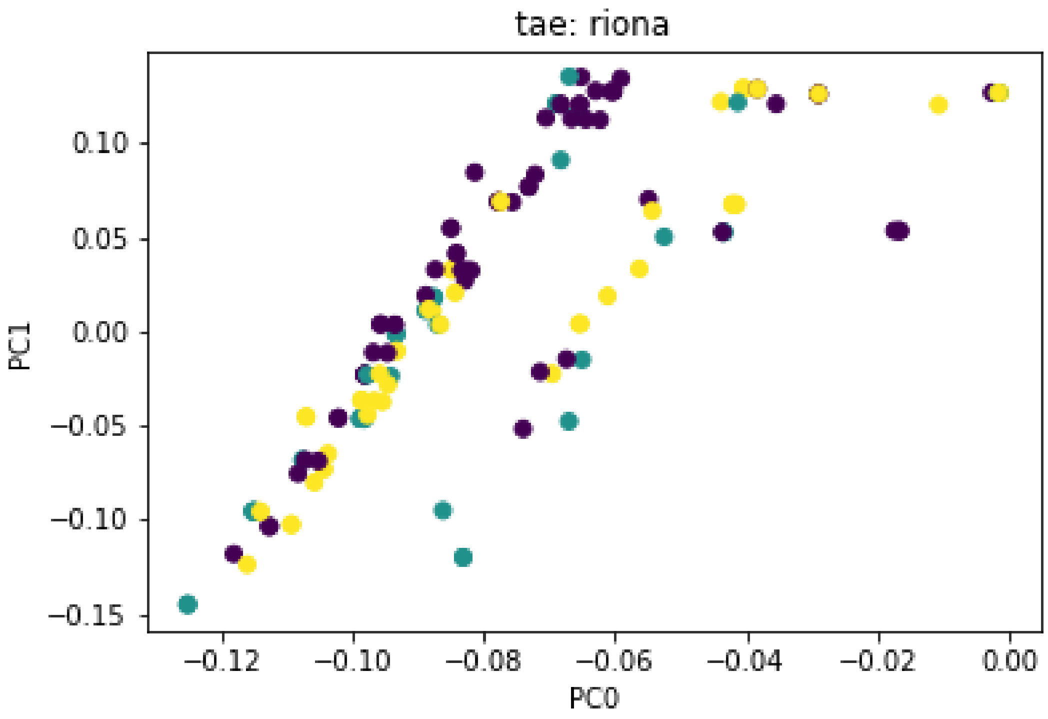

Figure 14: Riona algorithm prediction on the Tae dataset: PCA plot.

Figures 15 and 16 show neighbourhoods (yellow markers) of a particular example (shown in red) for the k-NN and k+NN algorithms, respectively. Results are largely the same as before.

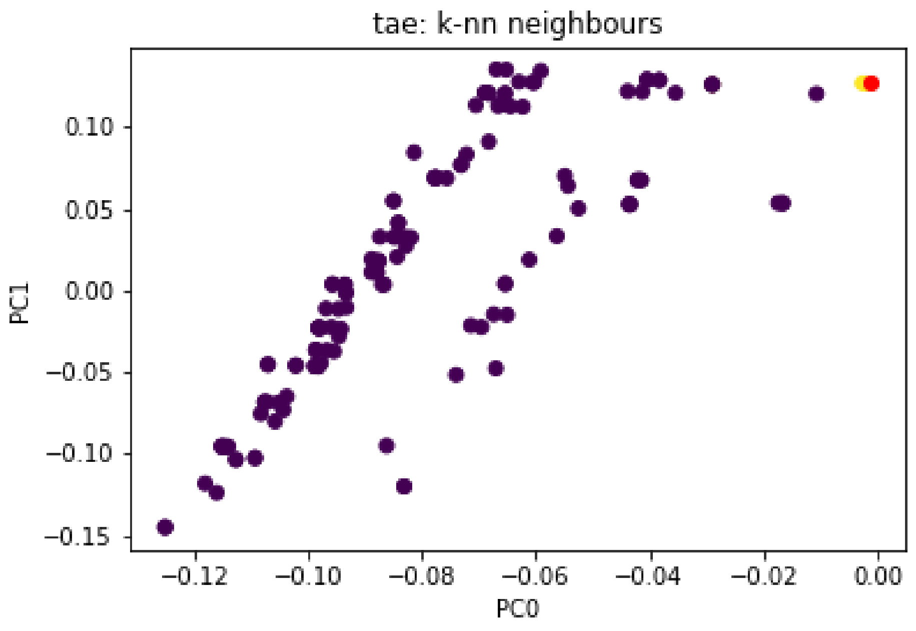

Figure 15: k-NN neighbours of a particular example in the Tae dataset.

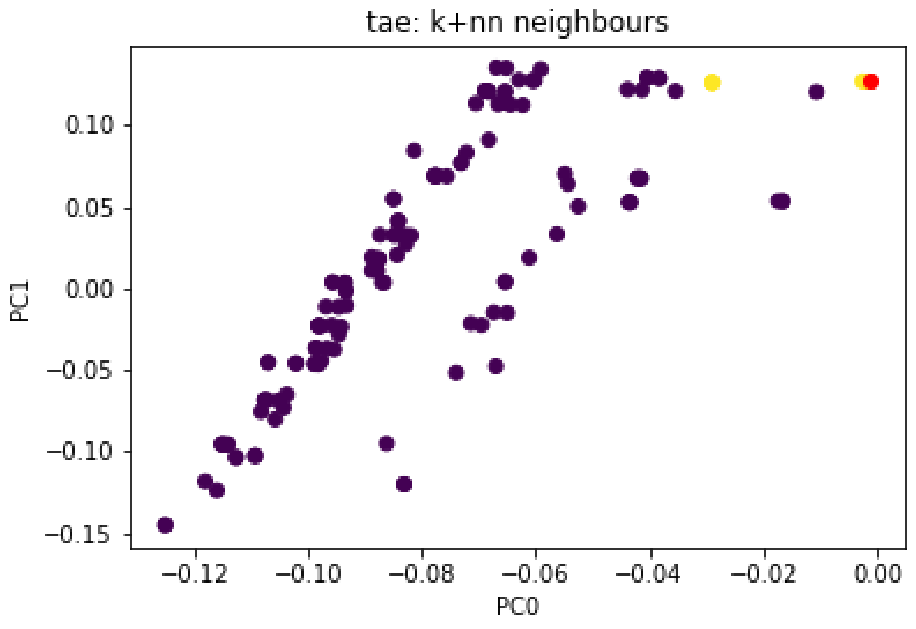

Figure 16: k+NN neighbours of a particular example in the Tae dataset.

Figure 17 shows distances calculated while predicting the label of the red test example. As we can see, the distance is calculated correctly.

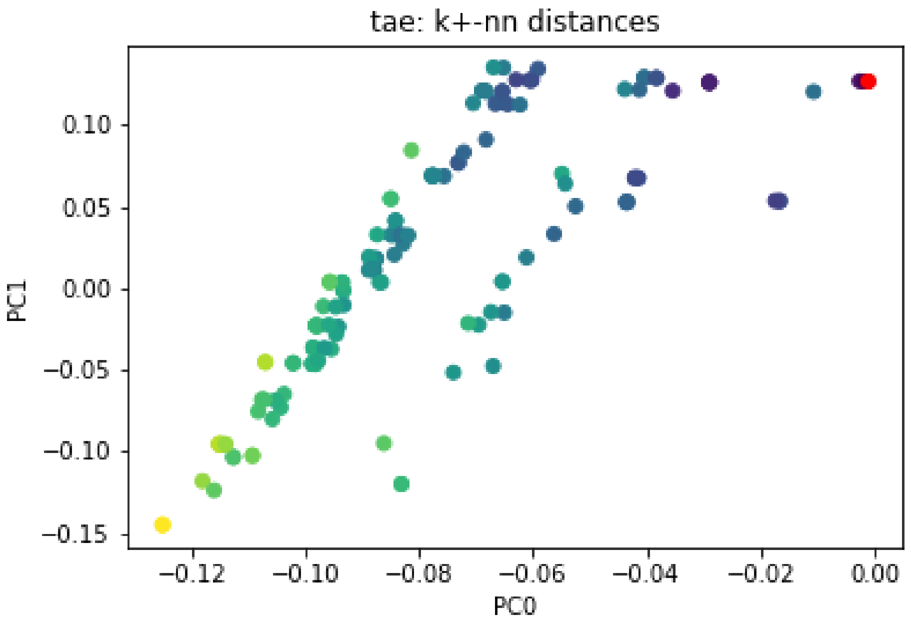

Figure 17: Calculated distance from a test observation (red) in the tae dataset.

Figure 18 shows, which observations produce a consistent rule. As we can see, this time minority of the observations are decisive for the Ria algorithm.

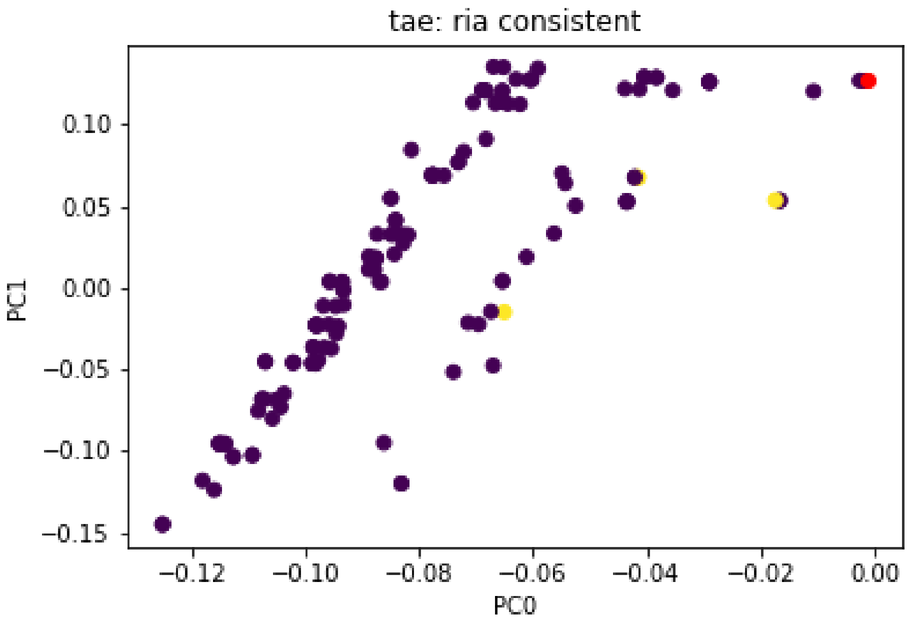

Figure 18: Consistent (yellow) observations/rules for a test observation (red) in the Zoo dataset.

9.5 Zoo Dataset

Figure 19 shows observations of the Zoo dataset, presented in the space of two main principal components from the PCA. Figures 20, 21, 22 and 23 show predicted labels by k-NN, k+NN, Ria and Riona algorithms, respectively. Numerical results are presented in Tab. 5. This time, the best results were obtained Riona. It is interesting, as Ria obtained the worst results.

measure	k-NN	k+NN	RIA	RIONA
accuracy	0.941	0.931	0.851	0.950
balanced accuracy	0.871	0.843	0.654	0.871

Table 5: Zoo results: accuracy and balanced accuracy.

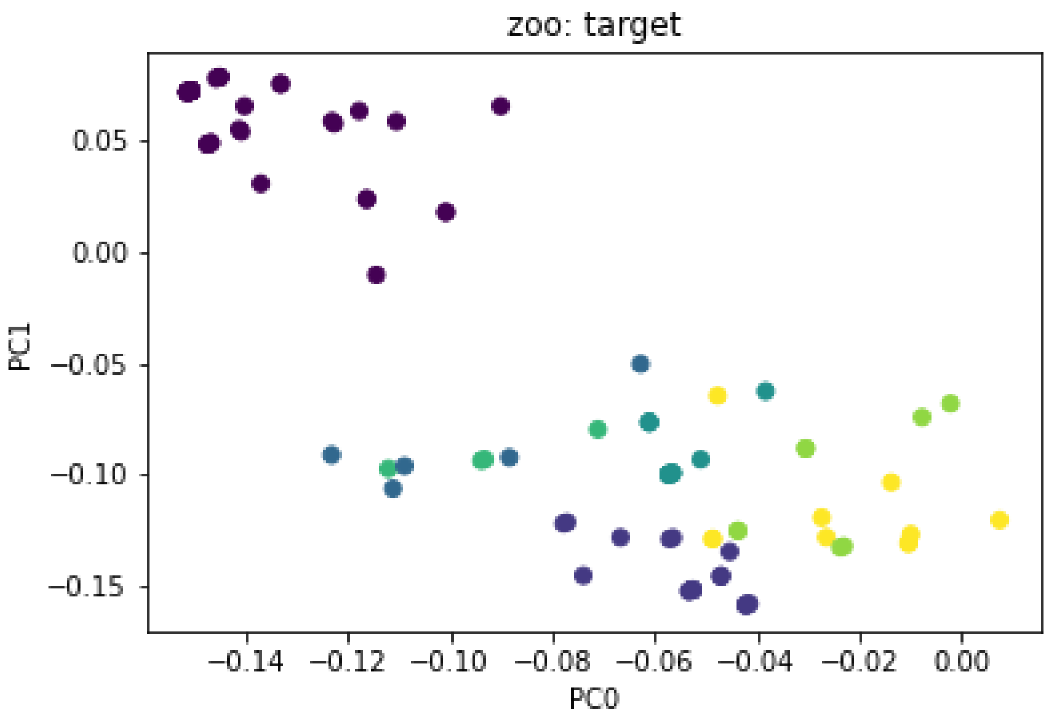

Figure 19: True class of observations of the Zoo dataset: PCA plot.

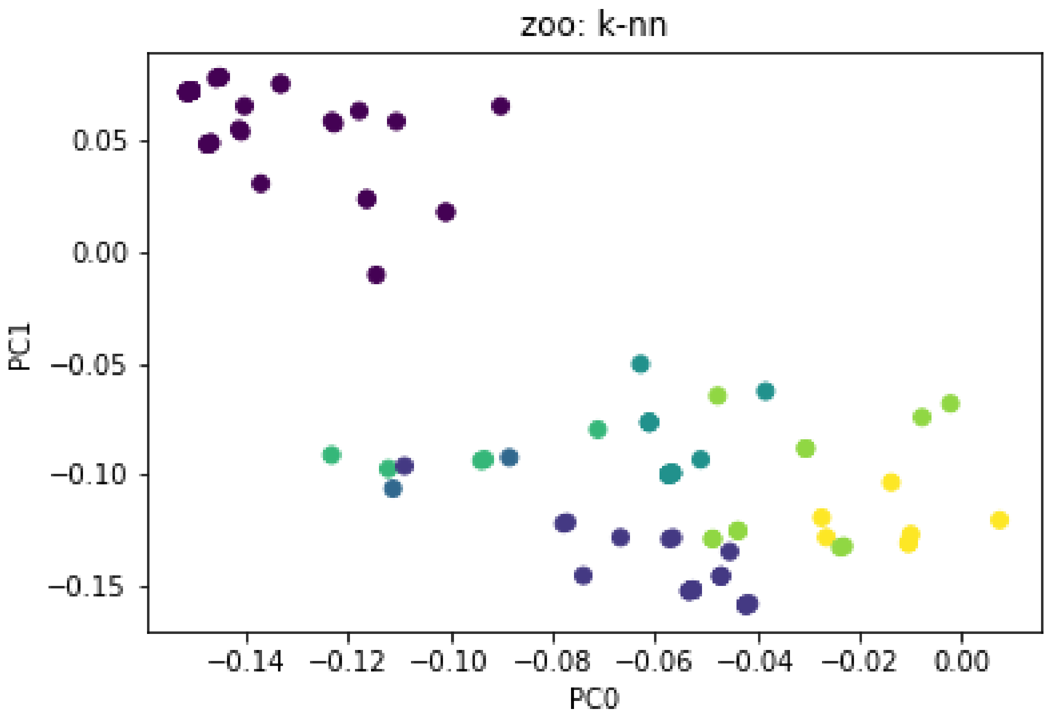

Figure 20: k-NN algorithm prediction on the Zoo dataset: PCA plot.

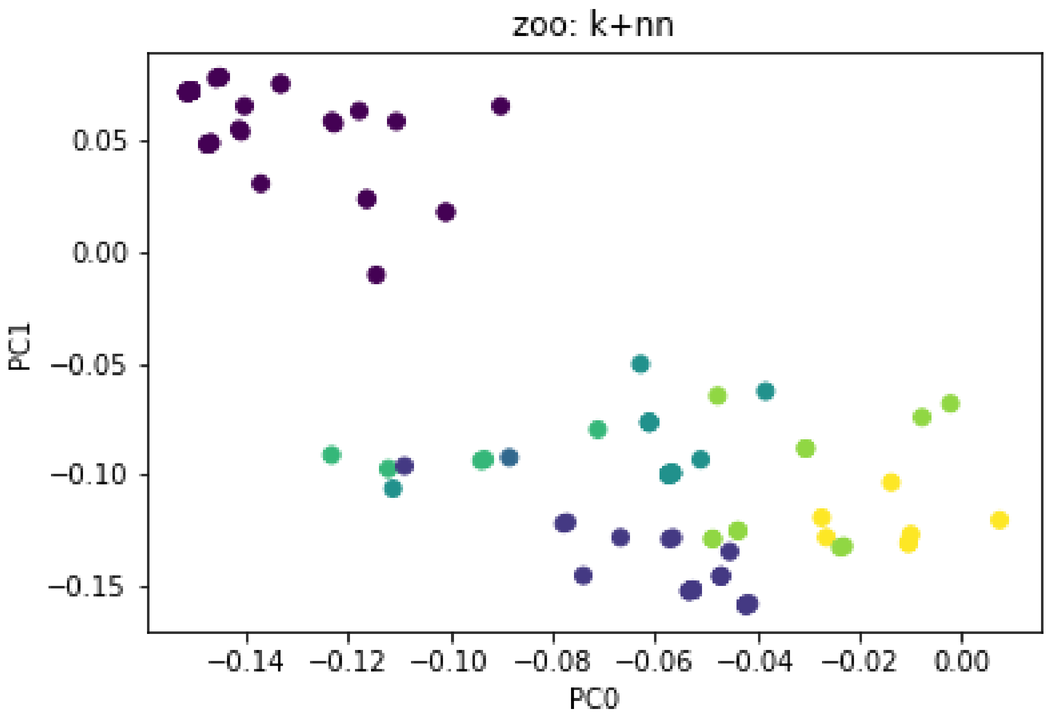

Figure 21: k+NN algorithm prediction on the Zoo dataset: PCA plot.

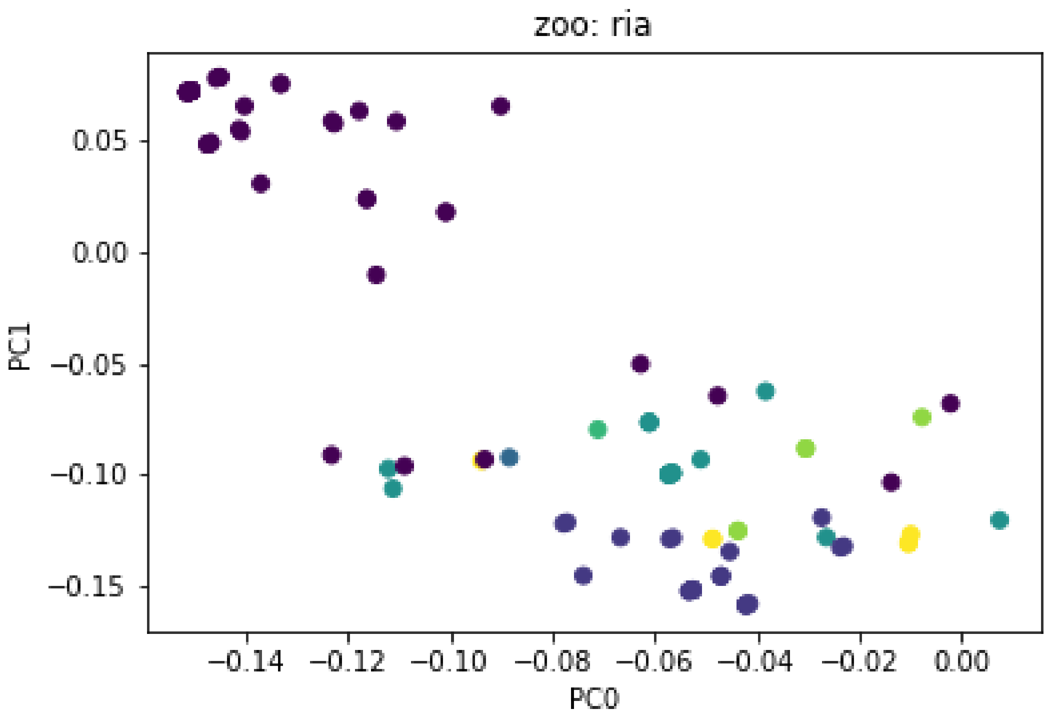

Figure 22: Ria algorithm prediction on the Zoo dataset: PCA plot.

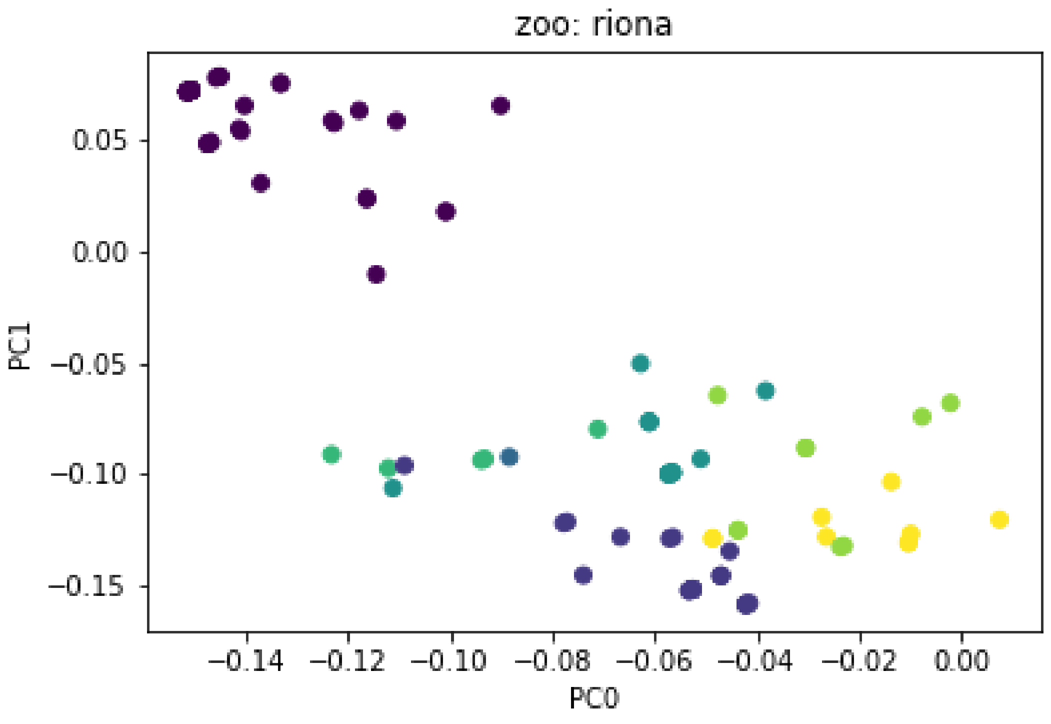

Figure 23: Riona algorithm prediction on the Zoo dataset: PCA plot.

Figures 24 and 25 show neighbourhoods (yellow markers) of a particular example (shown in red) for the k-NN and k+NN algorithms, respectively. As we can see, the neighbourhood is correctly close to the example. The neighbourhoods for the two algorithms are almost always the same if our attributes feature randomized real-valued numerical data.

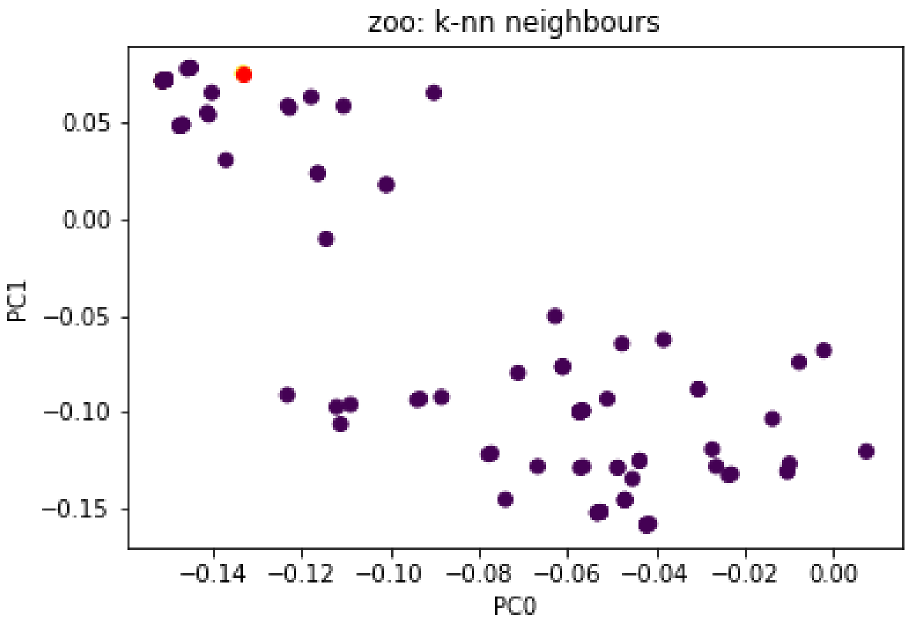

Figure 24: k-NN neighbours of a particular example in the Zoo dataset.

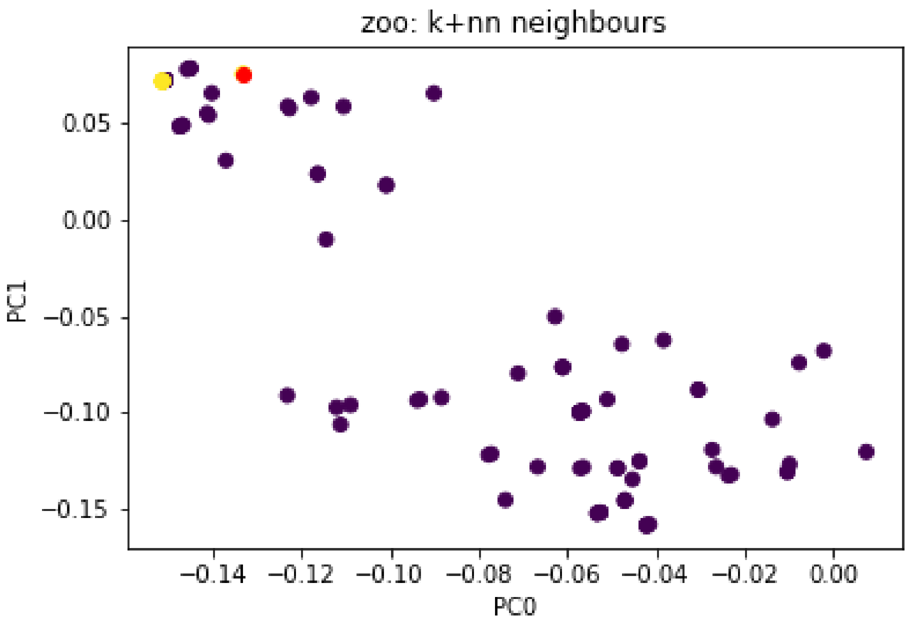

Figure 25: k+NN neighbours of a particular example in the Zoo dataset.

Figure 26 shows distances calculated while predicting the label of the red test example. As we can see, the distance is calculated correctly.

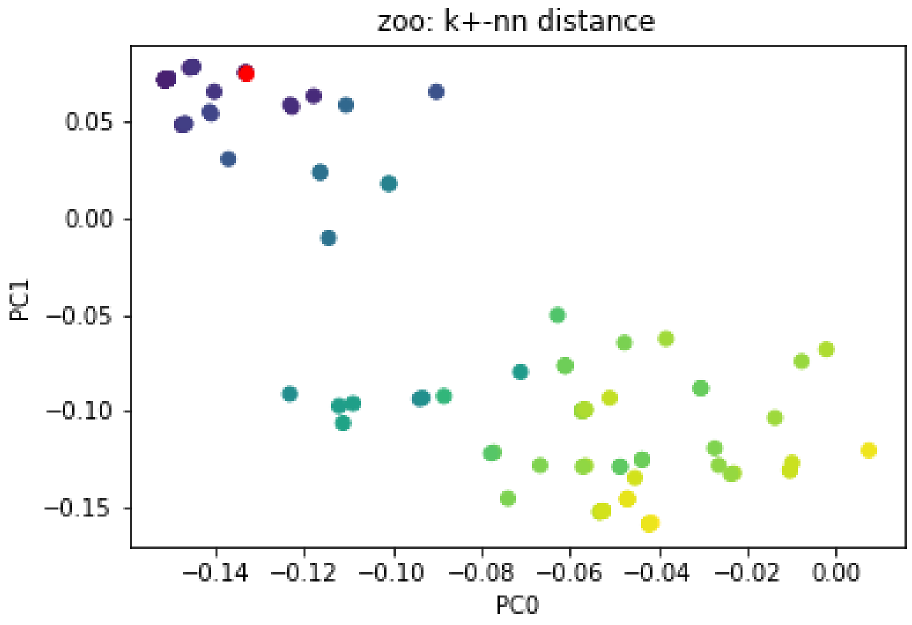

Figure 26: Calculated distance from a test observation (red) in the Zoo dataset.

Figure 27 shows, which observations produce a consistent rule. Here we can see, that consistent are all rules generated from the same cluster.

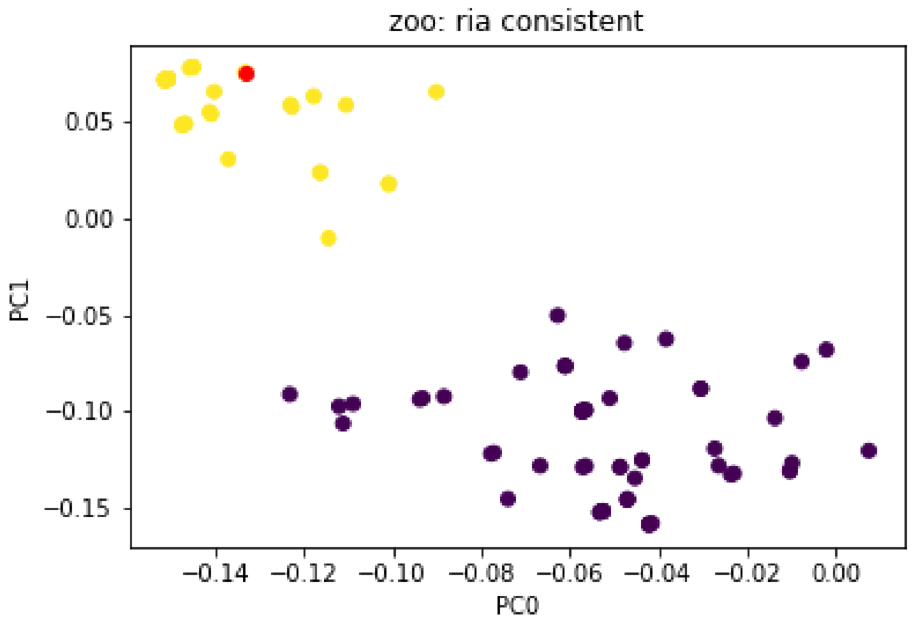

Figure 27: Consistent (yellow) observations/rules for a test observation (red) in the Zoo dataset.

10 Conclusions

In this work, we presented four classification algorithms: k-NN, k+NN, Ria and Riona. We showed that it is not decisive, which one is the best. Overall, the best performance was obtained by Riona and k-NN/k+NN.

In future work, an optimal value of k should be examined, as well as selecting a less naive metric for similarity measurements, e.g., by applying correct weights to the attributes.

References

Grzegorz Góra and Arkadiusz Wojna. RIONA: A new classification system combining rule induction and instance-based learning. *Fundam. Informaticae*, 51(4):369–390, 2002. URL `http://content.iospress.com/articles/fundamenta-informaticae/fi51-4-03`.