Universitat de Barcelona

Introduction to Machine Learning

Work 3: K-Nearest Neighbors exercise

Authors: Emer Rodriguez Formisano, Georgina Ballbè i Serra, Jorge Alexander
Supervisor: Maria Salamó
Date: 17th December 2017

Introduction

KNN classifier is a lazy learning algorithm which looks for the K nearest neighbours across the features of a dataset based on a defined distance metric. In this work a KNN algorithm was implemented with configuration for various distance metrics as well as the option of feature weighting and selection. The performance of the algorithm is then analysed.

The analysis of the performance of the algorithm consists of two parts. First, a search of the optimal KNN hyperparameters is performed by considering different K values and metric distances. Once a candidate model is selected, the second part consists of checking whether the feature weighted or feature selection methods improve the results.

All the mentioned evaluations are applied to two different datasets from the UCI repository[1]. The datasets chosen were the Hepatitis and Pen-based ones. The empirical results are exposed in the results section together with the main conclusions

Datasets

The analysis was performed using two datasets individually, without mixing the results. The intention of the analysis was to find good hyperparameters and test their performances with a given dataset, rather than finding robust hyperparameters across multiple datasets. The Hepatitis and Pen-based datasets from the UCI machine learning repository[1] were used and were provided having already been split into groups of 10 train and 10 test folds each.

Hepatitis

The Hepatitis data-set contains data regarding 155 patients who had the hepatitis disease. It includes several features such as age, sex,... (see the features list below). The feature to be classified is named “Class”, and represents whether that particular subject lived or died.

Features / dimensions

1. Class: DIE, LIVE
2. AGE: 10, 20, 30, 40, 50, 60, 70, 80
3. SEX: male, female
4. STEROID: no, yes
5. ANTIVIRALS: no, yes
6. FATIGUE: no, yes
7. MALAISE: no, yes
8. ANOREXIA: no, yes
9. LIVER BIG: no, yes
10. LIVER FIRM: no, yes
11. SPLEEN PALPABLE: no, yes
12. SPIDERS: no, yes
13. ASCITES: no, yes
14. VARICES: no, yes
15. BILIRUBIN: 0.39, 0.80, 1.20, 2.00, 3.00, 4.00 -- see the note below
16. ALK PHOSPHATE: 33, 80, 120, 160, 200, 250

17. SGOT: 13, 100, 200, 300, 400, 500,
18. ALBUMIN: 2.1, 3.0, 3.8, 4.5, 5.0, 6.0
19. PROTIME: 10, 20, 30, 40, 50, 60, 70, 80, 90
20. HISTOLOGY: no, yes

Training Size (Rows x Columns)
155 x 20

Pen-based

The pen-based dataset is a database of handwritten digits. The data contains 16 transformed coordinate values of a writing-tablet as writers wrote different individual digits on them. There is also one classification feature which is the digit written from 0 to 9.

Features
16 transformed coordinate values (between 0 to 100) plus one classification feature (0 to 9).

Training Size (Rows x Columns)
7494 x 17

Implementation

The Python programming language was used for the implementation and the code is split into the following files:

- **kNNAlgorithm.py**: file containing the KNN Algorithm
- **knn_utils.py**: contains auxiliary functions used for the analysis including: invoking knn algorithm; generating the weights; running the Friedman test$_{[2,4]}$; plotting the results.
- **parsing:py**: contains functions for loading the dataset and normalising their values.
- **main.py**: entrypoint script containing the steps of the entire analysis. The iterative steps are:
    1. Select a Dataset
    2. Run Part I: for each parsed fold, distance metric and k value, run the KNN Algorithm and compute the accuracy.
    3. Test the statistical differences of Part I models and select a candidate
    4. Run Part II: using the selected candidate hyperparameters, for each parsed fold and feature strategy, run the KNN Algorithm and compute the accuracy.
    5. Test the statistical significance using a Friedman test to calculate the p_values.

The following subsections explains the details of the process summarised previously.

Parser

The parser of the datasets receives the class field name as a parameter, and also a character/string used for the empty fields.

The parser returns a numpy matrix with all the instances features and a vector with the classes.

```
read_dataset (fileroute, classfield='class', emptyNomField='?' )
“”” Loads dataset information from file, computes values for missing features,
creates an all numeric matrix with all the instances features.
Creates a array containing the features names
Creates a array containing the class names
Creates a array containing the class values.

parameters:
fileroute: path to dataset file
classfield: name of the field containing the class value
emptyNomField: characters used in empty (missing information) in nominal features

returns:
x_allnumeric: n (number of instances) x m (number of features) numpy matrix
x_labels: 1x m array containing features labels
x_class: 1 x n array containing class assignment for each instance
x_class_names: 1 x c (number of different classes) array containing the names of
the classes
“”””
```

Normalization

Two normalization methods were implemented:

- Standardization values, zero mean and unit variance, using the function named: `normalize_mean_std(x)`
- Rescaling values between 0 and 1 using min and max in the function named: `normalize_min_max(x)`

`normalize_min_max` is used by default.

Cross-validation (10 folds)

To run the classifier for each one of the folds we use an iteration in our main script. We select what fold we are reading with variable `f`.

```
data_sets = [{'name': "hepatitis", 'dummy_value': "?", 'class_field': "Class"}]
,{'name': "pen-based", 'dummy_value': "", 'class_field': "a17"}]

for f in range(0, 10):
        path = 'datasets/{0}/{0}.fold.00000{1}'.format(dataset['name'], f)
        X_train, y_train, X_test, y_test = norm_train_test_split(path,
             dataset['class_field'], dataset['dummy_value'])
```

KNN Algorithm

The implementation the KNN algorithm is defined in a class called `kNNAlgorithm`. The constructor parameters are: the number of neighbours k, the metric (with an optional parameter), the selection policy and weights (optional).

Each metric and policy are implemented in its own method. As there is no training phase, the parameters are stored during the initialization together with the selection of the metric and policy function to be used.

The class follows the same usage pattern as the scikit-learn KNN. It has `fit` and `predict` methods. The `fit` method stores the training values for future predictions while the predict method performs the prediction of the class, it does it with the following steps whilst considering the selected metric and policy functions:

1. Computes the distances between train set and the samples
2. Compute the rank of the distance matrix row wise
3. Compute the prediction using the rank matrix
4. Return prediction, distance matrix and indexes

It is worth mentioning that a customised `mode` function used by the policy selection functions was also implemented. It efficiently finds the most frequent value in a vector and if more than one mode exists, it selects one value randomly.

Distance functions

KNN algorithm can be executed with the following similarity functions: Hamming, Euclidean, Cosine, Correlation and Minkowski (default p = 4). Correlation and Minkowski are the extra distance considered and each distance function is able to use weighted vectors of features.

Policy system

KNN algorithm selects which class to assign to the test sample by using the nearest sample (K=1) or voting (K=3, 5 or 7). The nearest sample was initially implemented as a separated policy function but it is in fact a particular case of the voting policy. Thus, the voting policy is the method used by default.

Weighted KNN Algorithm

The KNN Algorithm and its metrics are weighted feature compatible. In order to compute the weights of the features, the function `knn_weights` was implemented which acts as a filter before running the KNN algorithm. The filters used were two: **Information Gain** and the **RELIEF** (both versions, Relief for binary class and ReliefF for multiclass).

Information Gain

To obtain the weights of the features we call the function `mutual_info_classif` available in `scikit.feature_selection` module. This function measures the dependency between the variables.

Relief & ReliefF

To obtain the weights of the features we use the function belonging to the class `Relief` in `sklearn_relief`[8]. It implements Relief and ReliefF feature selection techniques returning a weight vector with values between 0 and 1. [5]

Selection KNN Algorithm

The selection method turns the weights into either 0 or 1. The filter considered are Information-Gain (IG) and Relief (the ones previously explained). The selection mechanism works as follows: it selects the most N relevant features based on weights obtained from the filter. For instance, if `num_features` = 3, the weight of the 3 features with the highest IG or Relief coefficient is set to 1 and the rest of features are set to 0.
In the results section is shown that this process is done for all the variables, all possible values of `num_features`.

Performance evaluation

In order to evaluate the performance of the algorithm we calculate the accuracy for each fold, using the accuracy function available in sklearn.metrics module. It simply calculates the **correctly classified over the total classified**.

In order to decide whether the performance of a certain configuration of the algorithm was significantly better or worst than others, a **Friedman test was used**[2,4]. The Friedman test calculates the p-value across groups of results, which can then be tested for rejecting the null hypothesis. The alpha value of **hypothesis rejection** was set at **0.05**.

Results & Discussion

The section contains the empirical results obtained from running the algorithms.

Hepatitis

The table 1 shows the parameters considered during the analysis of the Hepatitis dataset.

Tab.1 - Hepatitis parameters of the analysis

Parameter	**Values**
Significance level (alpha)	0.05

Num. of Neighbours	1, 3, 5 and 7
Distances	Euclidean, Cosine, Hamming, Minkowski (P=4), Correlation
Weighted Feature Method	Relief
Selection Feature Method	Information Gain with num of features from up to 1 to 18
Most important feature	Index 16 (Albumin)

Part I

It is observed that the **Euclidean distance metric with K = 7** gives the highest median and the lowest IQR (interquartile range) (See Fig.1). However, with the difference between the algorithms was found to have a Friedman **p_value of 0.920** , which means that the null hypothesis is not rejected, and the differences in performances are not statistically significant.

The following table shows the mean of the accuracy results of each fold:

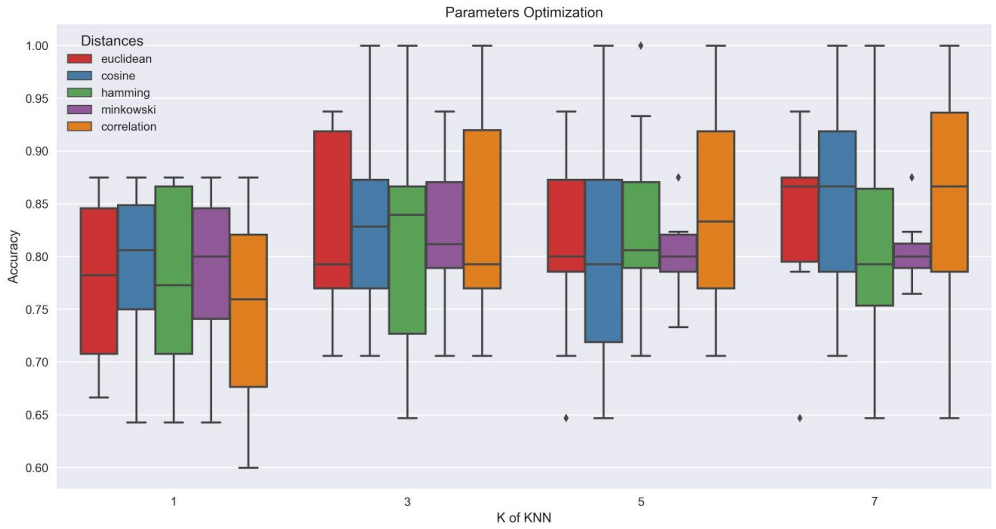
Tab 2. Mean accuracies

metric \ k	1	3	5	7
euclidean	0.774	0.826	0.808	0.840
cosine	0.787	0.833	0.807	0.846
hamming	0.780	0.819	0.833	0.820
minkowski	0.781	0.819	0.806	0.806
correlation	0.752	0.832	0.838	0.852

Tab 3. Descriptive statistics of the Accuracy for the selected model (Euclidean, 7)

N	Mean	Std. Dev.	Min	Q1 (25%)	Median (50%)	Q3 (75%)	Max
10	0.8396	0.0852	0.6470	0.7951	0.8666	0.8750	0.9375

Fig.1 - Hepatitis part 1 box plots



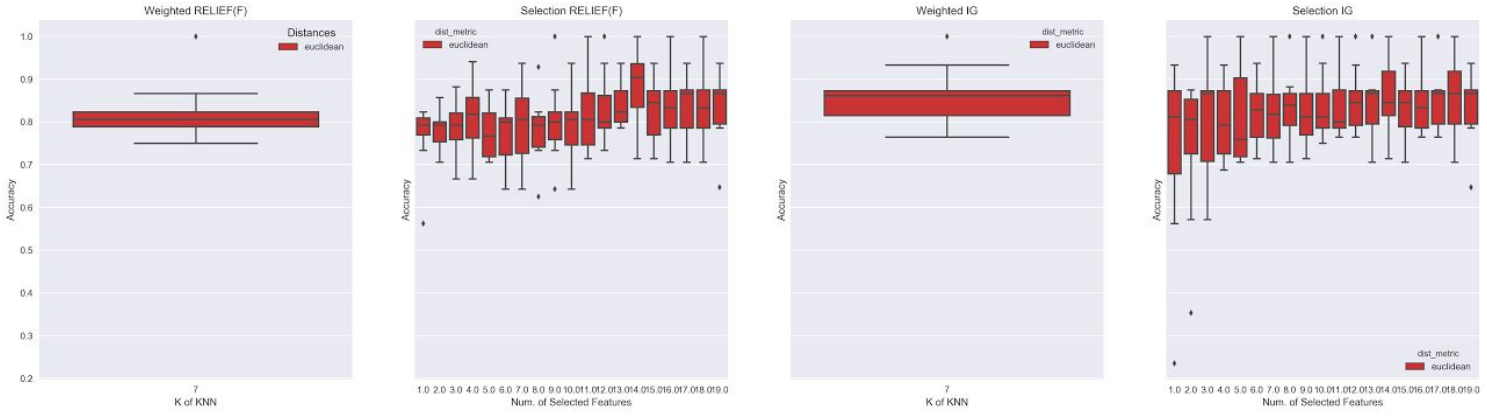
Part II

Hepatitis part II

When comparing all the feature weighting and selection (relief and IG) configurations, no statistical significant difference was found between the accuracies of the algorithms according to the Friedman test. The **p-value returned was 0.54**. The box plots of the spread of the accuracies show how similar they are (See Fig.2). In fact there's no clear one that has a higher median and lower IQR than the rest, although it's interesting to note that some of the selection accuracies achieve 100% accuracy as part of the spread.

It is also interesting to note that adding more than one feature, does not increase the performance of the model. It seems to be an extreme case, the performance of the model with a single feature with the highest Information Gain is statistically equivalent to the model with 19 features. This fact holds regardless the filtering strategy (IG or Relief)

Fig.2 - Hepatitis part 2 box plots



Pen-based

Table 4 shows the parameters considered during the analysis of the Pen-based dataset.

Tab.4 - Pen-based parameters of the analysis

Parameter	**Values**
Significance level (alpha)	0.05
Num. of Neighbours	1, 3, 5 and 7
Distances	Euclidean, Cosine, Hamming, Minkowski (P=4)
Weighted Feature Method	ReliefF
Selection Feature Method	Information Gain with num of features from up to 1 to 16

The correlation distance was dropped from the analysis due to a high computational time.

Part I

Taking a quick look at the average of the accuracies, table 5, a first thought is to consider the similarity of the results, excluding Hamming, which would suggest that the most suitable metric would be the one that has a lower computational cost. Furthermore, it is observed that the **Euclidean distance metric with K = 3** gives the highest median and the lowest interquartile range (IQR) (See Fig.4 and Tab 6.).

Tab 6. Descriptive statistics of the Accuracy for the selected model (Euclidean, 3)

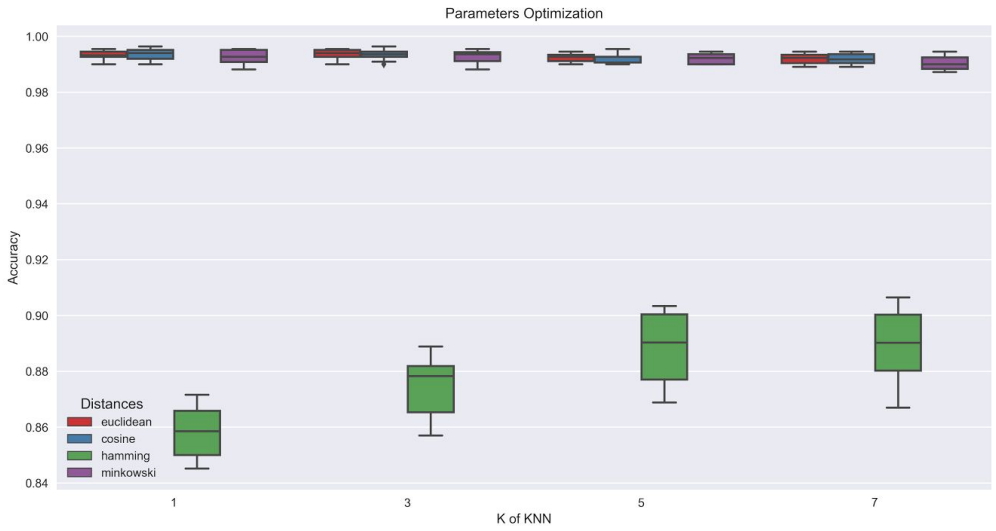
N	Mean	Std. Dev.	Min	Q1 (25%)	Median (50%)	Q3 (75%)	Max
10	0.9934	0.0017	0.9900	0.9927	0.9936	0.9945	0.9954

Tab 5. Mean accuracies

metric \ k	1	3	5	7
euclidean	0.993	0.993	0.992	0.992
cosine	0.994	0.994	0.993	0.992
hamming	0.858	0.876	0.886	0.890
minkowski	0.993	0.993	0.992	0.991
correlation	0.994	0.994	0.993	0.992

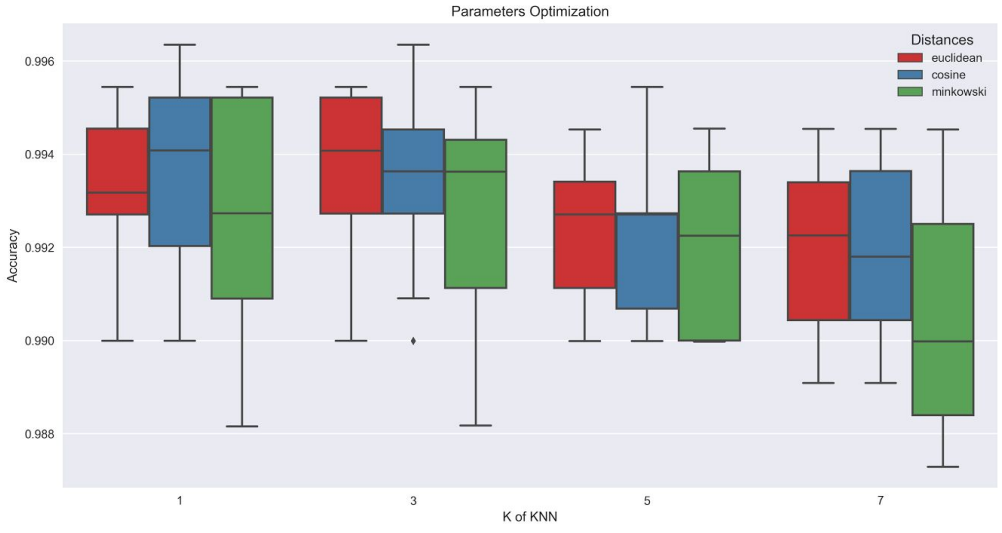
The results using Hamming distance are far below the other metrics (Fig. 3). Therefore, for better analysis, later figures shows the same results excluding Hamming distance.

Fig.3 - Pen-based part 1 box plots (all metrics)



Although the comparison tests among the three distances is not statistically different by the Friedman test (p-value 0.094), it doesn't make sense to analyse them because the accuracy range obtained is from 0.988 to 0.996. Therefore, regardless of the distance function and the K neighbours selected, the model will shows a good performance.

Fig.4 - Pen-based part 1 box plots - excluding Hamming



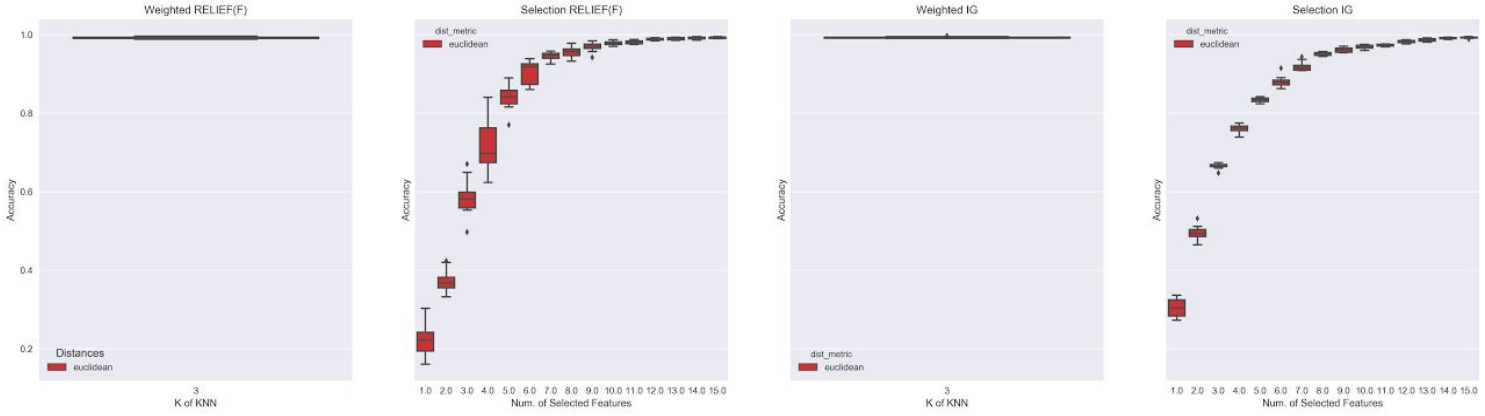
Part II

Pen-based part II

When comparing all the feature weighting and selection, relief and IG configurations, interesting patterns are observed.

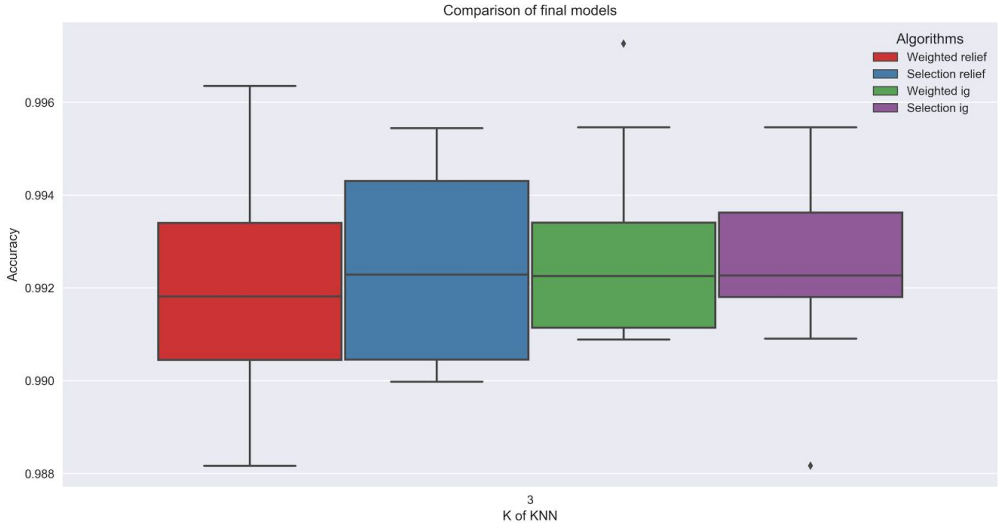
First, the selection strategy shows that the more features, the better the accuracy of the model. This is exactly the opposite extreme case shown in the previous dataset. The features of the Pen-based dataset represent coordinates on a writing tablet, it suggests that although there are certain areas of the writing area which may be more important in capturing information about a digit, the whole area needs to be included in order to achieve a high accuracy of digit recognition. Thus, we will consider the models with all the features.

Fig.5 - Pen-based part 2 box plots



Let's consider the best 4 models of each combination of filters and methods. The Fig. 6 shows the performances comparison and it is evident that no combination is better than another. The Friedman test confirms the hypothesis with a p-value of 0.821.

Fig.6 - Pen-based part 2 box plots of final models



Conclusions

KNN algorithm is a lazy learning algorithm and, as such, during the execution we could observe that the training computational cost is a minimum, just storing the training data, but then the running time for prediction is highly dependable on the number of training instances and number of features.

In the first part of this work, the KNN algorithm was analyzed for different distance metrics and K values. In the Hepatitis dataset it was found that a k value of 7 with a euclidean distance metric gave the best results. Where best is defined as the highest median average and lowest interquartile range. In the Pen-based dataset it was found that a k value of 3, also with a euclidean distance metric gave the best results.

When comparing the KNN class selection methods nearest neighbour (k=1) and voting (k=3,5 or 7) with Hepatitis dataset, using voting showed to be advantageous when observing the average accuracy of the folds, in order to see if that difference was statistically significant we did the friedman which showed there was no statistically significant, still euclidean distance and voting with 7 neighbours showed the highest accuracy median. This was also the case for Pen-based database and voting with 3 neighbours. As the voting vs neighbours method does not bring a significant increase of computational cost, voting models were chosen as the most suitable for these cases.

In the second part, it was found that the differences in performance between the KNN and the weighted KNN did not show enough statistical significance to suggest a rejection of the null hypothesis.

For selecting features, all but the top weighted feature of the current dataset were removed and then added back in order of weight descending. In the Hepatitis dataset, it was observed that the heaviest feature (according to information gain) had a similar contribution to the accuracy of the model as all the features combined (see Fig.2). This suggests that the Hepatitis dataset can be reduced in size dramatically. It was found that this important feature was the level of albumin, which makes sense as albumin is a protein made by the liver, which the hepatitis disease affects[9].

In the Pen-based dataset, it was observed that as the number of features increased, the contribution to accuracy increased too, logarithmically. As the features of the Pen-based dataset represent coordinates on a writing tablet, it suggests that although there are certain areas of the writing area which may be more important in capturing information about a digit, the whole area needs to be included in order to achieve a high accuracy of digit recognition. It would be interesting to see which coordinates are the most heavily weighted to suggest the most important area of a handwritten digit when it comes to resolving it to a digital one.

References

[1] Lichman, M. 2013. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. J. Mach. Learn. Res. 7 (December 2006), 1-30.

[3] R. Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the International Joint Conferences on Artificial Intelligence IJCAI-95.

[4] Wikipedia contributors. 2017. 'Friedman test', *Wikipedia, The Free Encyclopedia,* 24 November 2017, 11:13 UTC, <https://en.wikipedia.org/w/index.php?title=Friedman_test&oldid=811846519> [accessed 17 December 2017]

[5] Kira, Kenji and Rendell, Larry. 1992. The Feature Selection Problem: Traditional Methods and a New Algorithm. AAAI-92 Proceedings.

[6] SciKit documentation for sklearn feature selection. 2017. <http://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_selection> [accessed 17 December 2017]

[7] Pedregosa *et al.* 2011. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830.

[8] Alfredo Mungo. 2017. https://gitlab.com/qafir/sklearn-relief

[9] U.S. Department of Veterans Affairs. 2017. Albumin Hepatitis C. https://www.hepatitis.va.gov/patient/hcv/diagnosis/labtests-albumin.asp [accessed 17 December 2017]

Code execution

Code can be executed running the “main.py”.

The following files contain all the necessary functions:

`main.py`: main script to obtain performance data of KNN algorithms. Calls all necessary functions implemented in the files mentioned below.
`kNNAlgorithm.py` : implements the KNN classifier.
`knn_utils.py` : implements auxiliary functions to the algorithm.
`kw_nemenyi.py`: implements Nemenyi's multiple comparison test.
`Parser.py`: implements the parser of the dataset files.
`sklearn_relief.py`: implements Relief and ReliefF functions.

Extra packages may need to be installed, such as `seaborn` for the plots