COURSEWORK ASSIGNMENT **UNIVERSITY OF EAST ANGLIA**
School of Computing Sciences

UNIT : Machine Learning CMP-6002B
ASSIGNMENT TITLE : Classification Coursework

DATE SET **:** 25/02/19
DATE OF SUBMISSION **:** 01/05/19, 3 p.m. (Weds of Week 12)
RETURN DATE **:** 29/05/19
ASSIGNMENT VALUE **:** 40%
SET BY **:** Tony Bagnall **SIGNED:**
CHECKED BY **:** Gavin Cawley/Jason Lines **SIGNED:**

Aim:

Subject specific

The aim of this assignment is to implement variants of nearest neighbour classifiers and to evaluate/compare them on a range of real world data.

Transferable skills

The exercises will improve the following transferable skills: programming in Java; analysis of data and results; writing clearly and consisely; presenting data coherently.

Learning outcomes:

Subject specific

The student will learn about implementing variants of nearest neighbour classifiers and ensemble techniques and the general issues involved with implementing any classifier. They will understand better how to evaluate and compare classifiers on a range of data and for a specific problem. They will appreciate the difficulty in presenting technical results in a way to support or refute a hyptohesis.

Assessment criteria:

Part 1 will be assessed for correctness and the sensible presentation of working. Part 2 will be assessed on correctly following the specification, the use of data structures to improve efficiency, and the structure and style of code. Part 3 will be assessed on the presentation of the experimental design and results, the coherence of the paper and the conclusions drawn.

Description of the Assignment

The assignment is in three parts: Construct a classifier by hand. Implement variants of a well known classification algorithm; and to perform and write up experiments to test these variants against each other and other algorithms on a range of problems, including a case study of a specific problem.

1 Part 1: Nearest Neighbour Classifier by Hand (10%)

1. Table 1 shows a dataset describing the pitcher length and width of two species of pitcher plants, *Nepenthes raja* and *Nepenthes truncata*.

   Table 1 Pitcher lengths and widths of examples of *N. raja/truncata*.

width (cm)	length (cm)	species
9	13	*N. truncata*
5	14	*N. raja*
7	14	*N. truncata*
6	11	*N. raja*
6	12	*N. raja*
5	12	*N. raja*
8	15	*N. raja*
6	15	*N. truncata*
8	13	*N. raja*
7	12	*N. truncata*
4	13	*N. raja*
9	16	*N. truncata*

   Use the 3-nearest neighbour rule to identify the species of the examples shown in Table 2, showing all working in full.

   Table 2 Pitcher lengths and widths of unclassified *Nepenthes* plants.

width (cm)	length (cm)	species
5	13	*N. ?*
6	14	*N. ?*
7	15	*N. ?*
8	12	*N. ?*

2. Explain, with the aid of diagrams where appropriate, why the k-nearest neighbour classifier is sensitive to attribute scaling. Standardise the data in Table 1 (to zero mean and standard deviation one) and repeat the classification for the data in Table 2.

2 Part 2: Implement Nearest Neighbour Classifiers (50%)

This task involves implementing variants of nearest neighbour classifiers in Weka. Please note that this must be your original code. We are of course aware of existing implementations of NN classifiers in Weka and other packages. Copying from existing source code will be considered possible plagiarism and will be dealt with accordingly. Please note that you can assume that all attributes are real valued for all the implementations (i.e. you do not need to handle nominal attributes). You should check this using the Weka Capabilities mechanism.

2.1 Implement standard *k*-NN

Implement a class called `KNN` that extends the Weka class `AbstractClassifier`. The `buildClassifier` method simply needs to store the training data. The `classifyInstance` method must find the k closest neighbours to the test instance using Euclidean distance, then predict the value of the target variable as the mode of the target variable values of the k closest neighbours in the training set (i.e. by a simple voting scheme). Ties should be settled randomly. By default, k should be set to 1, but your class should also contain a method for the user to set k. The `distributionForInstance` method should return the proportion of the neighbours voting for each response variable value. The class should have the following additional functionality:

1. **Standardise attributes** (default to true). Each attribute should be standardised to zero mean and standard deviation one when the flag is set to true. This should be done in `buildClassifier`.
2. **Set k through leave one out cross validation** (default to false). You should evaluate all values of k between 1 and `maxK`, where `maxK` is 20% of the train set size or 100, whichever is smaller. The value of k you choose should be the one with the highest accuracy estimate on the training data (ties should be settled randomly). There are a range of ways to implement this. The simplest (and least efficient) is to just use another instance of `KNN` for each value of k. This is acceptable as an initial approach, but you should be able to work out a much more efficient way of doing this. Setting k should happen in the `buildClassifier` method.
3. **Use a weighted voting scheme** (default to false). Rather than allow all the k neighbours an equal vote, it is possible to weight their vote by the distance to the test instance. This means the closest neighbour has a greater influence on the final prediction than the k^{th} closest. Weights should be inversely proportional to distance. The weighted vote for neighbour instance y with test instance x should be $w = \frac{1}{1+d(x,y)}$ (there are other weighting functions of course, but you should use this for this coursework). If the weighting scheme is used it should be employed in all calculations in `buildClassifier`, `classifyInstance` and `distributionForInstance`.

Write a test harness to demonstrate that your code is correct and that when configured correctly the output of your classifier for the problem given in part 1 is the same as the results you obtained by hand. Add as much testing evidence as you deem necessary to demonstrate correctness. It is a good idea to work out expected answers by hand then test in code. If you do this, add the expected output in comments. Generally, remember to comment your code and conform to standard formatting conventions such as having accessor methods to set parameters.

2.2 Implement a *k*-NN ensemble

Ensembling can often improve the performance of a base classifier at the cost of greater memory and build time. You are required to implement an ensemble of *k*-NN classifiers, called `KnnEnsemble`. Note that you should implement the ensemble from scratch rather than extend or contain any other classifier in Weka. The classifiers should be stored as an array or List, with a default ensemble size to 50. Each member of the ensemble should be built on a different subset of the training data and a different set of attributes in order to inject diversity into the ensemble. The exact method of sampling the cases and attributes is up to you. However, it is preferable that you adopt a sampling method that has been proposed in the research literature (and provide a reference). Try to avoid cloning data for each classifier. It is up to you to decide how to configure the individual ensemble members. The `buildClassifier` method needs to set up each of the ensemble members in a way that new instances can be processed in `classifyInstance` and `distributionForInstance`.

Note that this is not a particularly detailed specification. There are some overall design decisions you need to make. There is not necessarily one correct and one incorrect way of doing this, and marking is not pass/fail. You need to decide on the way you think best. Please comment you code extensively to justify any decisions made. I am more interested in you appreciating the potential issues than I am in you following one set way of doing things.

3 Evaluation of Nearest Neighbour Classifiers (40%)

Your task is to perform a series of classification experiments and write them up as a research paper. Your experiments will address the question of whether the variations you have implemented for kNN improve performance over a range of problems and whether kNN is a good approach for a specific case study data set. You have been assigned a specific data set (see blackboard) and a link to further information. Please note that the for this part of the coursework we mark the paper itself, not the code used to generate the results. We advise you reserve significant time for reading and checking your paper, and recommend you ask someone else to read it before submission. Imagine you are writing a paper for a wide audience, not just for the marker. Aim for a tight focus on the specific questions the paper addresses. There are four issues to investigate:

1. Test whether one of the improvements you implemented in Part 2.1 (standardising attributes, setting K through CV, weighted vote) is better than 1-NN on the 30 classification problems we have provided.
2. Test whether your ensemble is better than a single base kNN classifier on the 30 classification problems.
3. Compare your ensemble against build in Weka classifiers on the 30 classification problems.
4. Perform a case study on your assigned data set to propose which classifier from those you used in part 3 would be best for this particular problem.

3.1 Experiments

You should compare classifiers based on accuracy (or, equivalently, error) and any other metrics described in the evaluation lecture that you feel are appropriate. The first two hypotheses require a comparison of two classifiers over multiple data sets. The third requires comparing multiple classifiers over multiple data sets. The final case study involves comparing multiple classifiers on a single dataset. You should think about the experimental design for each experiment, including deciding on a sampling method (train/test, cross validate or resample) and performance measures. The choice of classifiers for part 3 is up to you, but we recommend you include random forest in the list.

3.2 The Write Up

You should write a paper in Latex using the style file available on blackboard. The paper should be called "An experimental assessment of nearest neighbour classifiers". This should be around 4 pages, but there is no maximum or minimum limit. Your paper should include the following sections:

1. **Introduction**: start with the aims of the paper, a description of the hypotheses you are testing and an overview of the structure. Include in this your prior beliefs as to what you think the outcome of the test will be, with some rationalisation as to why. So, for example, do you think cross validation for k will make a difference? Can you find any published literature that claims to answer any of the questions?

2. **Data Description**: an overview describing the data, including data characteristics summarised in a table (number of attributes, number of train/test cases, number of classes, class distribution). For your case study data set you should have more detail, including a description of the source of the data and the nature of the problem the data describes. Look for references for similar types of problem.
3. **Classifier Description**: Include a description of your implementation of nearest neighbour classifiers, including details of design choices you made and data structures you employed. You should include an example of how to use the classifier with the refinements you have included. Also provide an overview of the other classifiers you use in the case study, including references.
4. **Results**: A description of the experimental procedure you used and details of the results. Remember, graphs are good. There should be a subsection for each hypothesis. The case study section should go into greater depth and remember, accuracy is not the only criteria.
5. **Conclusions**: how do you answer the questions posed? Are there any biases in your experiment and if so, how would you improve/refine them?

Presenting the output of experiments is a key transferable skill in any technical area. I stress again we mark the paper for this section, not the experimental code. Avoid writing a description of what you did. We do not need to know about any blind alleys you went down or problems you encountered, unless they are relevant to the experiments (e.g. memory constraints). Please also take care to include references where appropriate, and to check the formatting of references.

Assessment

Marks will be awarded as follows

10% **Part 1** will be assessed for correctness and the sensible presentation of working. You can do this by hand if you wish. Present all working and add comments to identify what you are doing.

50% **Part 2: 2.1: 30%, Part 2: 20%.** Part 2 will be assessed on correctly following the specification, the use of data structures to improve efficiency, and the structure and style of code.

40% **Part 3** will be assessed on the presentation of the experimental design and results, the coherence of the paper and the conclusions drawn.

 Submission instructions overleaf.

Submission Procedure

You should create a PDF for submission using the PASS system to include your solutions to part 1,2 and 3. If you do Part 1 by hand, simply take a picture of it and print to pdf. If you do this, please make sure the file is small. There is a size limit on pdf submission.

Via e:Vision: You should submit your PASS generated PDF online on e:Vision. This is the formal submission. They print them out for you and it is what we mark. If you do not submit via e:Vision it will be considered a non-submission.

via Blackboard: You should submit a zipped Netbeans project with your code **and** your PDF generated with PASS. We use this to verify the code works and to check for plagiarism. If you do not submit via blackboard you will be penalised. In addition to the classifier code for part 2, you should submit the code to generate the results for part 3. Structure the code to match the paper structure. So, for example, you may have a static method to run the experiments for the case study.

Written coursework should be submitted by following the standard CMP practice. Students are advised to refer to the Guidelines and Hints on Written Work in CMP (`https://intranet.uea.ac.uk/computing/Links/Reports?_ga=1.7481330.1383599.1413214592`).

Plagiarism: Plagiarism is the copying of close paraphrasing of published or unpublished work, including the work of another student; without due acknowledgement. Plagiarism is regarded a serious offence by the University, and all cases will be investigated. Possible consequences of plagiarism include deduction of marks and disciplinary action, as detailed by UEA's Policy on Plagiarism and Collusion.