COMP3308 Assignment 2

General Information

Deadline

9 May 2025 (Friday Week 10), 11:59

Late submissions are allowed up until 3 days after the deadline. A penalty of 5% per day late will apply. Assignments submitted more than 3 days late will not be accepted (i.e. will receive 0 marks). The cut-off time for a day is 11:59pm.

Overarching Goal

In this assignment, your goal is to implement and compare machine learning classifiers in the context of three prediction tasks, reporting on your findings and drawing general insights from these findings. In particular, you will need to:

1. Implement two classifiers from scratch (Naive Bayes and K-Nearest-Neighbours)
2. Implement 10-fold stratified cross-validation (to evaluate your classifiers)
3. Evaluate your classifiers' performances on two different datasets (relating to diabetes and room occupancy)
4. Evaluate additional classifiers on the same datasets using Weka (ZeroR, OneR, decision trees, random forest, support vector machines and neural networks) along with Weka's versions of the same classifiers (Naive Bayes and KNN)
5. Write a report (in the style of a scientific paper), which critically discusses your findings

✏ Marking

The assignment is worth 24 marks, and counts for **24% of your final course mark**. It consists of two parts: code (12 marks) and report (12 marks).

The code part is submitted through Ed (right here). It will be automarked against testcases. There is no manual marking.

The report part is submitted through Canvas and it will be manually marked. The marking rubric is provided on Canvas.

ℹ The report is worth **50%** - we recommend you leave plenty of time to complete it.

Groups

The assignment must be completed in **pairs** (groups of 2 students). No more than 2 students are allowed. Your partner does not need to be from the same tutorial, but must be from the same stream - COMP3308.

You need to register a group in Canvas, so that the code marks are correctly exported from Ed to Canvas and the report marks are correctly recorded in Canvas. There is a **separate group registration for Assignment 1 and Assignment 2**. Even if you work with the same partner as in Assignment 1, you need to register your group again, under Assignment 2. Both students will receive the same mark.

Go to People -> Groups and register your group under 3308 A2. (There is only one group-set. We found a solution to the problem with the number of groups in Canvas.)

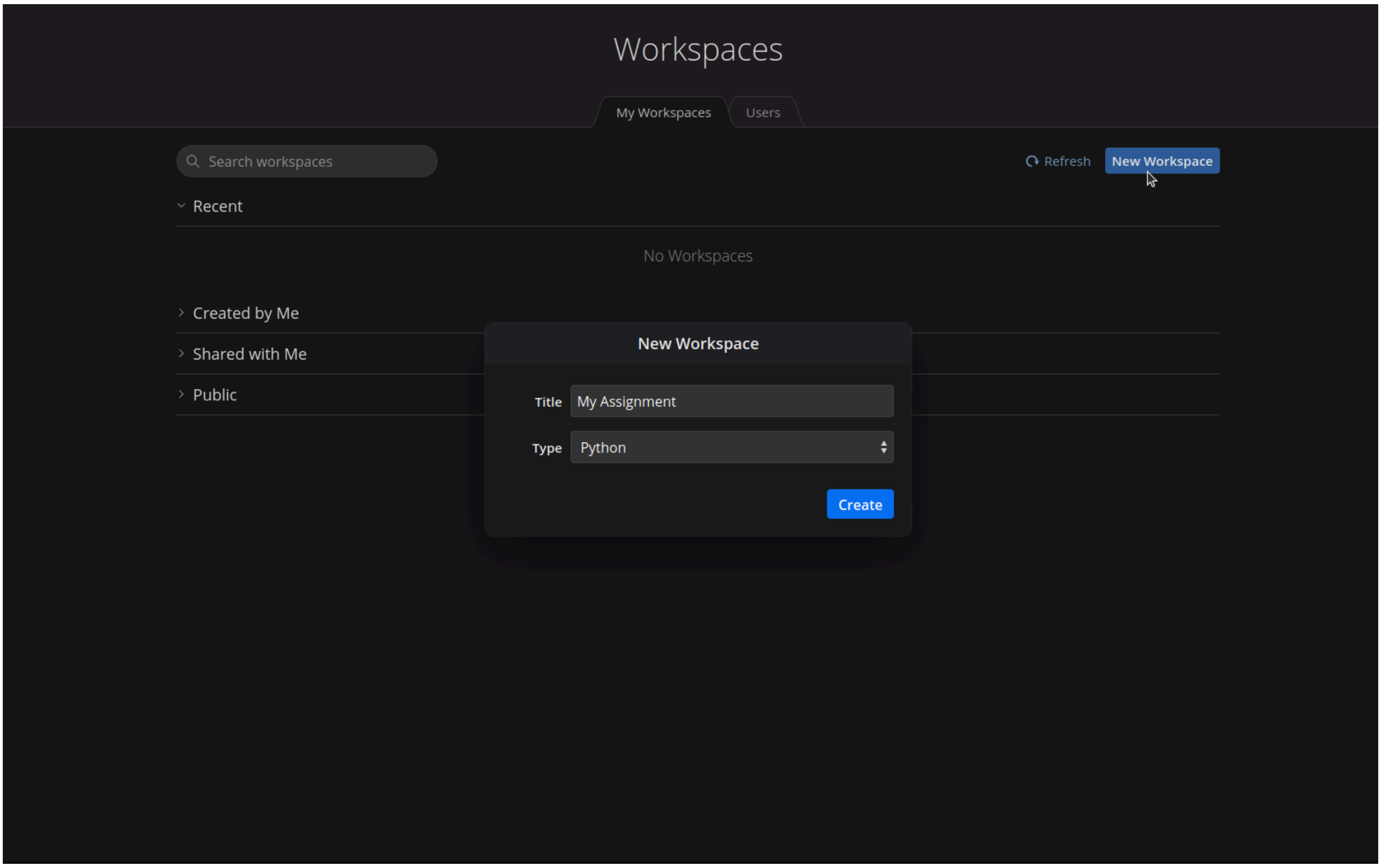
Language

Your implementation must be written in **Python**

Submission

The code part must be submitted through Ed. It consists of several parts, each with its own testcases. Submit your solution for each part to the relevant page. You can submit multiple times, each time you will receive feedback about the passed and failed testcases. If you pass all testcases, you will obtain the full marks.

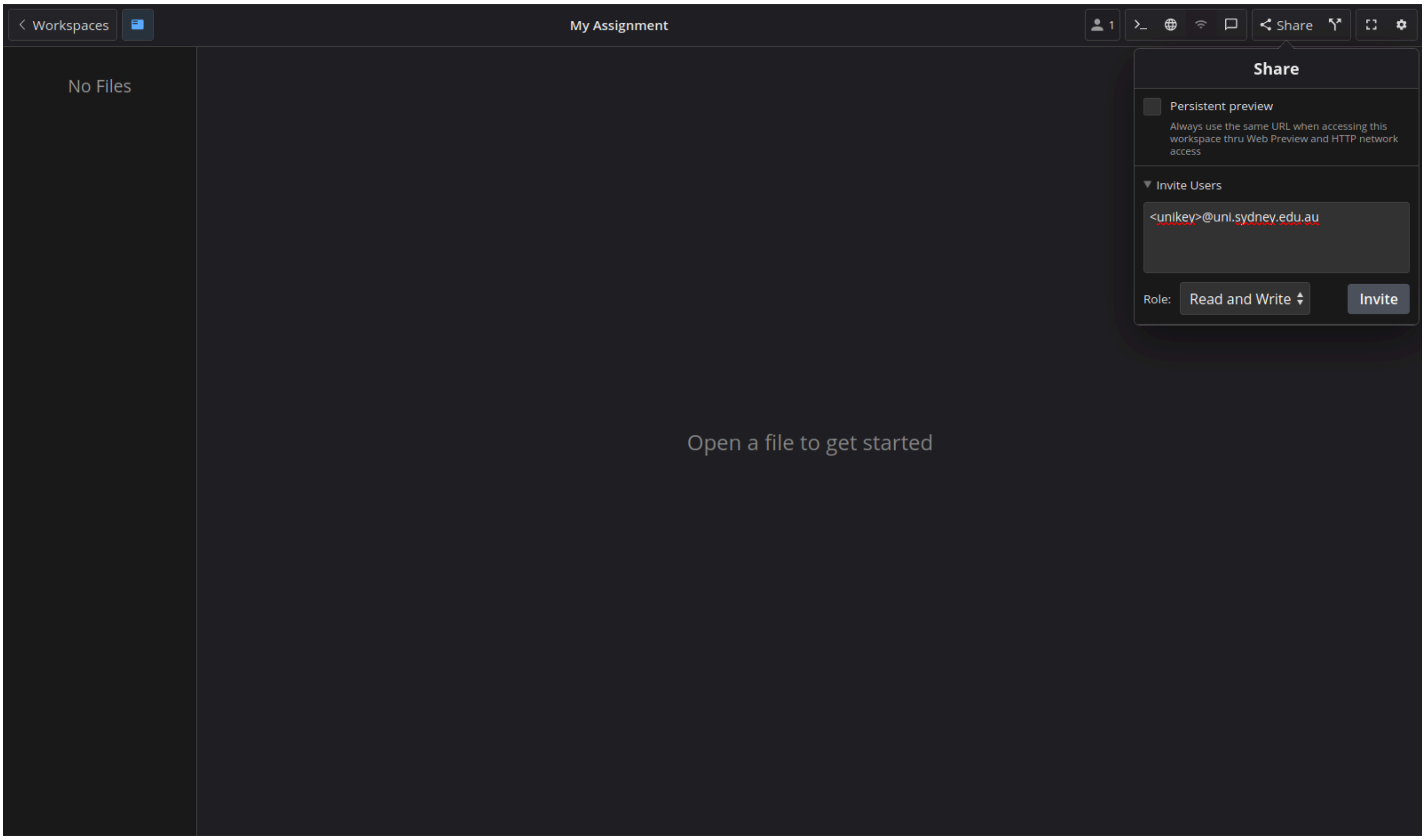
This assignment page is not shared - each student has their own workspace. You can however collaborate with your group member in real time through the **workspaces** page. To do so, create a **new workspace**:



Title: My Assignment

Type: Python

From there, invite your group member through the **share** tab:



□ Persistent preview

Always use the same URL when accessing this workspace thru Web Preview and HTTP network access

<unikey>@uni.sydney.edu.au

Role: Read and Write

The invited user can then access the workspace from the **Shared with Me** section on the

workspaces page.

When the assignment is finished, **both students should submit the whole assignment** (all parts) in their own assignment page. Finally, remember that the marks are based on the *latest* submission, so don't forget to press “Test”.

In the report part, you need to discuss your findings. Instructions for the report are included in this Ed module, but the report should be submitted to the Assignments page in Canvas.

Academic honesty

Please read the University policy on Academic Honesty very carefully.

Plagiarism (copying from another student, website or other sources), making your work available to another student to copy, engaging another person to complete the assignments instead of you (for payment or not) are all examples of academic dishonesty. Note that when there is copying between students, both students are penalised – the student who copies and the student who makes their work available for copying.

Do not confuse legitimate co-operation and cheating! This is a group assignment. You can discuss the assignment with other students. However, you need to form a group with one other student and work together to write the code and the report, and your work cannot be shared beyond this group.

You are not allowed to use any generative AI tools for the assignments in this course.

To detect similarity in this assignment, we will use TurnItIn and Ed's plagiarism detection system, which are extremely good. If you cheat, the chances that you will be caught are very high. Be smart and don’t risk your reputation by engaging in plagiarism and academic dishonesty!

Datasets

Well done!!

You now know a lot of general information about the assignment!

Next Step: Data

The next two slides will provide you with important information about the data you'll be using for this assignment (relating to diabetes and room occupancy), along with workspaces to explore the data!

We hope you're data-set on learning more!

Pima Indian Diabetes Dataset

If you measured a number of health-related factors (e.g., blood pressure, bmi, health), could you determine if someone had diabetes? Our first dataset, `pima-indians-diabetes.csv` will help us explore this!

The dataset consists of 768 instances described by 8 numeric attributes, relating to patients of Pima Indian heritage. Each entry in the dataset corresponds to a patient’s record; the attributes are personal characteristics and test measurements; and the class shows if the person tested positive for diabetes or not. More details about the features and class are given in the table below:

Column	**Description**
tp	Number of **times pregnant.**
gc	Plasma **glucose concentration** a 2 hours in an oral glucose tolerance test.
bp	Diastolic **blood pressure** $(mmHg)$
sft	Triceps **skin fold thickness** (mm)
si	2-Hour **serum insulin** (muU/ml)
bmi	**Body mass index** (kg/m^2)
dpf	**Diabetes pedigree function.**
age	**Age** (years)
class	Whether the individual **tested positive for diabetes** (yes) or not (no).

Further Information

Further information, including more context about the dataset is provided in the `pima-indians-diabetes-info.txt` file, also included in this workspace. We strongly recommend you read this carefully. **Note**: The original dataset can be sourced from UCI Machine Learning Repository. However, you need to use the dataset available here, which is modified version of the original dataset.

Your Task

To the right, we've provided you with a workspace to explore this dataset. **The workspace is unmarked**, but you will need to use the dataset later, and to also discuss it in your report, so it's a good idea to play around with it!

Room Occupancy Dataset

If you placed sensors in a room (e.g., light, temperature etc.), could you predict if the room was occupied? Our second dataset, `occupancy-estimation.csv`, will help us to explore exactly this!

The dataset consists of 2,025 instances, each corresponding to a point in time between December 2017 and January 2018 when sensor readings (light, temperature, sound and CO_2) were taken in a room, and the room occupancy was recorded. More specifically, the features and class are as follows:

Column	Description
temp	Temperature reading ($^\circ$C).
light	Light reading (Lux)
sound	Output of an amplifier attached to a microphone (Volts). The louder the sou
CO2	Carbon dioxide reading (PPM)
class	Whether or not there were people in the room (**no** = no people, **yes** = 1-3 pe

Your Task

To the right, we've provided you with a workspace to explore this dataset. **The workspace is unmarked**, but you will need to use the dataset later, and to also discuss it in your report, so it's a good idea to play around with it!

Acknowledgements

The dataset is adapted from the UCI Room Occupancy Estimation Dataset, donated to the UCI machine learning repository on 15/08/2023.

Normalisation

Data Preprocessing

Nice work! Now that you've explored the data, you'll need to perform some pre-processing on the two files (i.e., `pima-indians-diabetes.csv` and `occupancy-estimation.csv`) before you start implementing your own classifiers. The steps are detailed below:

Step 1: Normalise with Weka

After downloading the data files (from the two previous slides), use Weka's in-built normalisation filter to normalise the values of each attribute (excluding the class) to make sure they are in the range [0,1]. The normalisation should be done along each column (attribute), not each row (entry). Save the pre-processed files as `pima.csv` and `occupancy.csv`.

Step 2: Remove headers and paste file contents

Remove the first row of `pima.csv` and `occupancy.csv` (i.e., the column names), then copy their contents into this workspace, ensuring the pima data is pasted into the `pima.csv` section and the occupancy data is pasted into the `occupancy.csv` section.

Note: All further activities will be performed using these submitted files; if you have correctly processed the data, you will no longer need the original files.

Classifiers

Excellent work!! You've explored the data and performed some preprocessing! It's now time to implement two classifiers from class:

1. K-Nearest Neighbour
2. Naive Bayes

Function Signatures

You will write functions with the following signatures:

```
classify_nn(training_file, testing_file, k)
```

```
classify_nb(training_file, testing_file)
```

The first function will perform classification using the nearest neighbour algorithm; the second will use Naive Bayes

The functions will accept two filenames, given as strings. The first contains a dataset to be used to train the classifier; the second contains a dataset without class values for testing. The classifier functions must return their classifications for each example in the testing file.

The k parameter for `classify_nn` specifies how many of the nearest neighbours to use when classifying an example. It will be an integer.

Training data file

The input training file will consist of several rows of data, each with n attributes plus a single class value (yes or no). The file will not have a header row, will have one example per line, and each line will consist of a normalised value for each of the non-class attributes separated by commas, followed by a class value. This example has 8 attributes, like `pima.csv`:

```
0.084,0.192,0.569,0.274,0.105,0.179,0.090,0.284,yes
0.091,0.287,0.255,0.234,0.191,0.175,0.174,0.000,no
0.000,0.929,0.681,0.106,0.238,0.348,0.003,0.000,no
0.193,0.455,0.379,0.284,0.187,0.355,0.058,0.096,yes
0.489,0.774,0.578,0.218,0.122,0.829,0.104,0.000,no
0.378,0.839,0.489,0.118,0.173,0.885,0.045,0.691,yes
```

Testing Data File

The input testing data file will consist of several new examples to test your data on. The file will not have a header row, will have one example per line, and each line will consist of a normalised value for each of the non-class attributes separated by commas. An example input file could look as follows:

```
0.588,0.628,0.574,0.263,0.136,0.463,0.054,0.333
0.243,0.274,0.224,0.894,0.113,0.168,0.735,0.321
0.738,0.295,0.924,0.113,0.693,0.666,0.486,0.525
```

ℹ Note: Your functions should be able to handle any number of attributes; not just the 8 attributes from `pima.csv` or the 4 from `occupancy.csv`. You can assume that if the input training file has n attributes + a class column, then the testing file will also have n attributes.

Output

Your functions will return a list. Each element of the list, in order, corresponds to one of the examples in the given testing data file. The elements of the list should be strings representing the classification your function chose (i.e., "yes" or "no") for the corresponding line in the testing data file. An example return value could be as follows:

```
["yes", "no", "yes"]
```

ℹ Note: These outputs are in no way related to the sample inputs given above. If you have any questions or need any clarifications about program input or output, ask a question on Ed or ask your tutor. Since your program will be automatically tested by Ed, it is important that you follow the instructions exactly.

⚠ Warning: These implementations must be your own work - the use of libraries to perform ML tasks is not permitted.

K-Nearest Neighbour

The k-Nearest Neighbour algorithm should be implemented for any k value and should use Euclidean distance as the distance measure. In the case of ties, predict **yes**.

Naive Bayes

The Naive Bayes should be implemented for numeric attributes, using a probability density function. Assume a normal distribution, i.e. use the probability density function for a normal distribution. As before, if there is ever a tie between the two classes, choose class **yes**.

Ensemble

Well done! Now your task is to implement an ensemble that combines the predictions of three classifiers: two nearest neighbor (with different k) and one Naive Bayes. It combines the predictions by taking the majority vote, e.g. if the three predictions are **yes, yes**, **no**, the ensemble will predict **yes**.

You should use your implementation of Naive Bayes and k-Nearest Neighbor from the previous tasks.

The parameters k1 and k2 in the function below are the number of neighbors of the first and second k-Nearest Neighbor algorithms respectively.

Evaluating Classifiers

Brilliant Job!!! You've just finished implementing your own classifiers! It's now time to evaluate your classifiers; i.e. find out how well they actually perform as classifiers.

Evaluation Overview

This evaluation will consist of two steps:

- Dividing the datasets into **10 folds** (automatically marked, see the next slide)
- Implementing **10-fold cross-validation** using those folds to determine the performance of your classifiers, including the algorithm's average accuracy over the 10 folds (not automatically marked, but needed for the report)
- Extending the evaluation to include **Weka's implementations** of the same classifiers (NB and kNN), as well as many additional classifiers.

Stratified Folds

To show that you understand how 10-fold stratified cross-validation works, you will need to generate a file called `pima-folds.csv` from the original `pima.csv`. This file can be generated in any manner you choose (manually or using code). `pima-folds.csv` should contain 10 folds, each containing the approximately the same number of examples, and the ratio of yes examples to no examples should be approximately the same for each fold. Each fold should be in the following format:

- Name of the fold, fold1 to fold10.
- Contents of the fold, with each entry on a new line.
- A single blank line to separate the folds from each other.

An example of the `pima-folds.csv` file would look as follows (made up data):

```
fold1
0.588,0.628,0.574,0.263,0.136,0.463,0.054,0.333,yes
0.243,0.274,0.224,0.894,0.113,0.168,0.735,0.321,no

fold2
0.588,0.628,0.574,0.263,0.136,0.463,0.054,0.333,yes
0.243,0.274,0.224,0.894,0.113,0.168,0.735,0.321,no

...
fold10
0.588,0.628,0.574,0.263,0.136,0.463,0.054,0.333,yes
0.243,0.274,0.224,0.894,0.113,0.168,0.735,0.321,no
```

Note: The number of instances per fold should not vary by more than one. If the total number of instances is not divisible by ten, the remaining items should be distributed amongst the folds rather than being placed in one fold.

Copy your stratified folded data from `pima-folds.csv` into the corresponding file in this Ed problem and submit, and Ed will check that you have correctly created the folds and applied stratification.

Since this is an exercise to test your understanding, only `pima-folds.csv` is tested. However, you should still follow the same process to create `occupancy-folds.csv` in order to perform the evaluations on the next slide.

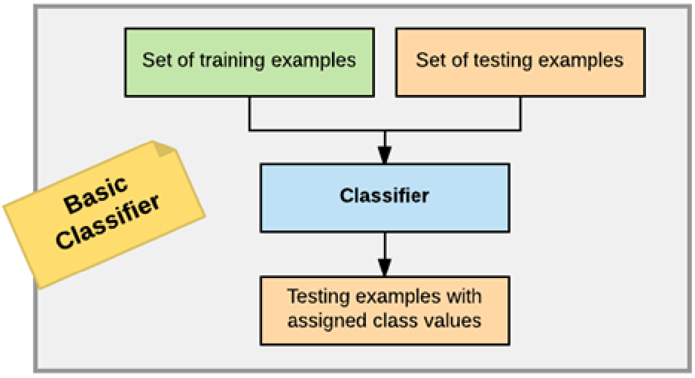
For more information on cross validation and stratification, see the following diagrams.

Cross Validation

Training Example
Contains:
1. a set of attributes
2. a class value that is already known

Testing Example
Contains:
1. a set of attributes
2. no class value; we have not seen this example before

Classifier
1. Use training data only to build the classifier
2. Run the testing data through the classifier one example at a time to assign a class value to each



Basic Classifier

Set of training examples

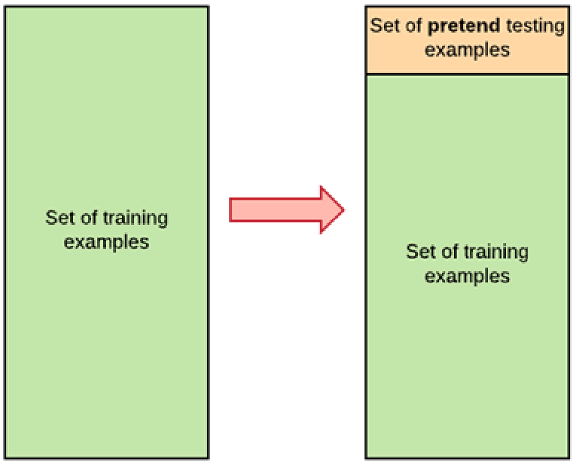
Set of testing examples

Classifier

Testing examples with assigned class values

Okay, so what if we only have a training dataset?
How do we know if the classifier is performing well?

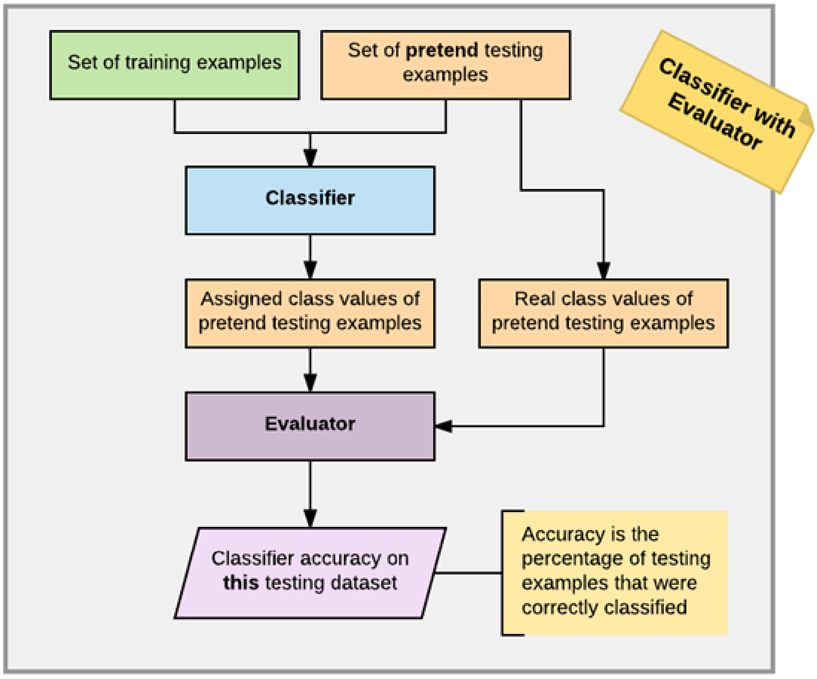
Evaluator
1. Determines how good a classifier is at assigning class values
2. Can only be used when you know the actual class values for your test data. How else would you know if you are correct?



Set of training examples

Set of **pretend** testing examples

Set of training examples



Classifier with Evaluator

Set of training examples

Set of **pretend** testing examples

Classifier

Assigned class values of pretend testing examples

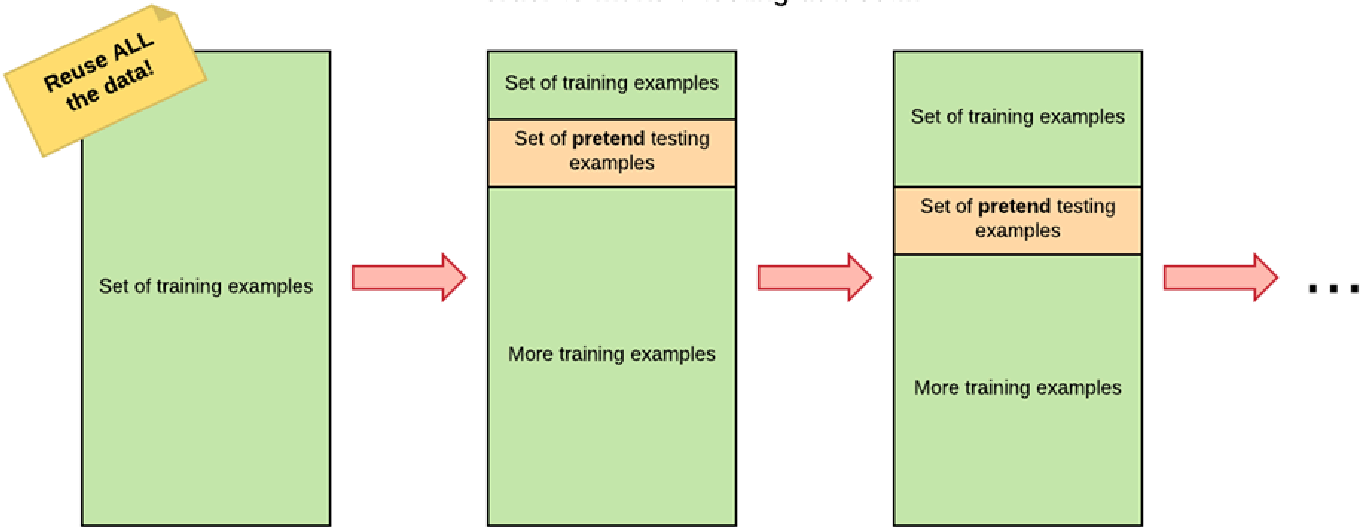
Real class values of pretend testing examples

Evaluator

Classifier accuracy on **this** testing dataset

Accuracy is the percentage of testing examples that were correctly classified

But now we've had to sacrifice some of the training data in order to make a testing dataset...



Reuse ALL the data!

Set of training examples

Set of training examples

Set of **pretend** testing examples

More training examples

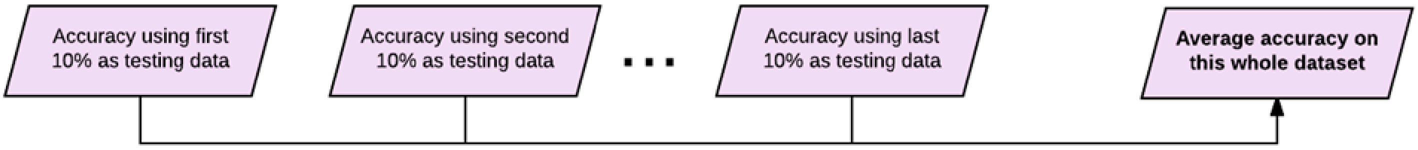
Set of training examples

Set of **pretend** testing examples

More training examples

. . .

After running ten "different" datasets through the classifiers, find the average accuracy.



Accuracy using first 10% as testing data

Accuracy using second 10% as testing data

. . .

Accuracy using last 10% as testing data

Average accuracy on this whole dataset

Cross Validation

Now that you've created your folds, write a program to evaluate your classifiers on the two datasets using those folds. In particular, you should perform 10-fold cross-validation to evaluate their accuracy (and any other performance measures you consider important) on both datasets.

> Note: There is no auto-marking for this section, but cross-validation is required to complete the report; you need to know the average accuracy of your NB and kNN algorithms (for various values of k).

Further Results - Weka

Congratulations!!

You've just finished all the programming tasks for this assignment! It's now time to use Weka to collect more results, so you'll be ready to write your final report.

Further Results with Weka

In Weka, select 10-fold cross validation (it is actually 10-fold stratified cross validation) and run the following algorithms: ZeroR, 1R, k-Nearest Neighbor (k-NN; IBk in Weka), Naive Bayes (NB), Decision Tree (DT; J48 in Weka), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM; SMO in Weka) and Random Forest (RF).

Compare the performance of the Weka’s classifiers with your k-Nearest Neighbour, Naive Bayes and Ensemble classifiers. Do this for both the `pima.csv` and `occupancy.csv` datasets.

Report

Describe your analysis and findings in a report which is similar in style to a research paper.

Your report should include the sections listed below. It should be written as if you were describing the study to someone who has not seen the data or this assignment before.

There is no minimum or maximum length for the report. However, please keep your report concise - you will be marked on the quality of the content not the length of the report.

1. Introduction

This section should briefly state the aim of your study and include a paragraph about why this study is important according to you.

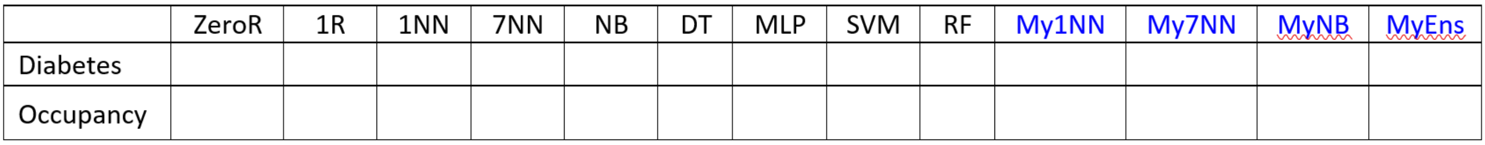
2. Data

This section should describe the datasets, mentioning the number of attributes and classes. You should also briefly summarise the similarities and differences between the datasets.

3. Results and discussion

Results

The accuracy results should be presented in the following table where My1NN, My7NN and MyNB are your implementations of the 1NN, 7NN and NB algorithms, and MyEns is your ensemble algorithm combining 1NN, 7NN and NB, evaluated using your stratified 10-fold cross validation.



	ZeroR	1R	1NN	7NN	NB	DT	MLP	SVM	RF	My1NN	My7NN	MyNB	MyEns
Diabetes													
Occupancy													

Discussion

- Compare the performance of all classifiers in terms of accuracy (and other performance measures if you have other measures)
- Compare your kNN and NB classifiers with Weka's
- Compare your ensemble MyEns with the individual classifiers it combines (My1NN, My7NN, MyNB)
- Discuss the changes in performance on the two datasets - did the classifiers perform differently and if so, did these differences make intuitive sense to you?

Include anything else that you consider important.

4. Conclusion

Summarise your main findings and suggest future work.

5. Reflection

Write one or two paragraphs describing the most important thing that you have learned throughout this assignment. Each group member should write their own reflection.

Submission

Submit your report through Canvas, using the appropriate assignments page. Reports will undergo plagiarism detection using TurnItIn, and will be marked manually in Canvas.

Marking Criteria

Code [12 marks]

Based on the tests passed in Ed; automatic marking.

Report [12 marks]

Manual marking. Please see the marking rubric in Canvas.