Assignment 2: Classification

Deadlines

Submission: 11:59pm, Friday 14th May, 2021 (week 10)

Late submissions policy

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

Marking

This assignment is worth 24 marks = 24% of your final mark. It includes two parts: code and report, worth 12 marks each. The code will be marked automatically by PASTA, the report will be marked manually.

The assignment can be completed individually or in pairs (no more than 2 students are allowed). See the submission details sections for more information about how to setup an individual or pair submission in PASTA.

Task description

In this assignment you will implement the Naïve Bayes and Decision Tree algorithms and evaluate them on a real dataset using the stratified cross validation method. You will also evaluate the performance of other classifiers on the same dataset using Weka. Finally, you will investigate the effect of feature selection, in particular the Correlation-based Feature Selection method (CFS) from Weka.

Programming languages

Your implementation can be written in Python, Java, C, C++ or MATLAB. The assignment will be tested automatically on PASTA, so your code must be compatible with the language versions listed in the "Input" section of this specification. You are not allowed to use any of the built-in classification libraries for the purposes of this assignment.

Submission

This assignment (code and report) will be submitted using PASTA. However, you also need to submit the report separately in Canvas for plagiarism checking. Please see the submission details sections for more information.

PASTA is located at https://comp3308.it.usyd.edu.au/PASTA/. In order to connect to the website, you'll need to be connected to the university VPN.

If you are off-campus, you'll need to be connected to the university VPN in order to access the PASTA website. You can read this page to find out how to connect to the University Cisco VPN. The students in China should use the new (FortiClient) VPN, which is the special VPN for accessing university resources from China. If you are on-campus and connected to the University wireless network, you don't need a VPN to access PASTA.

To access PASTA from China, it may be necessary to run both the FortiClient VPN and Cisco VPN; connect to FortiClient VPN first and then to Cisco VPN.

PASTA will allow you to make as many submissions as you wish, and each submission will provide you with feedback. Your last submission before the assignment deadline will be marked, and the mark displayed on PASTA will be the final mark for your code (12 marks).

Any assignment components that need to be submitted to PASTA will be included in a box like this one, describing what to submit.

1. Dataset preparation

The data

The dataset for this assignment is the Pima Indian Diabetes dataset. It contains 768 instances described by 8 numeric attributes. There are two classes - **yes** and **no**. Each entry in the dataset corresponds to a patient's record; the attributes are personal characteristics and test measurements; the class shows if the person shows signs of diabetes or not. The patients are from Pima Indian heritage, hence the name of the dataset.

A copy of the dataset can be downloaded from Canvas. There are 3 files associated with the dataset. The first file, **pima-indians-diabetes.names**, describes the data, including the number and the type of the attributes and classes, as well as their meaning. This file can be opened in any text editor and its purpose is to give you information about the dataset. Make sure you read the whole file to learn more about the meaning of the attributes and the classes.

The second file, **pima-indians-diabetes.data**, contains the numerical data itself, as described by the **names** file. The third file, **pima-indians-diabetes.discrete** is the same data after having all attributes discretised into nominal values. Your task is to predict the class, where the class can be **yes** or **no**.

Note: The original dataset can be sourced from UCI Machine Learning Repository. However, you need to use the dataset available on Canvas as it has been modified for consistency.

Data pre-processing

In order to use the numerical data in this assignment, you're going to need to do some pre-processing on the **pima-indians-diabetes.data** file.

Use Weka's in-built normalisation filter to normalise the values of each attribute to make sure they are in the range [0,1]. The normalisation should be done along each column (attribute), not each row (entry). The class attribute is not normalised – it should remain unchanged. Save the pre-processed file as **pima.csv**.

Warning: Weka assumes that any data file you give it has a header row (a row of column names at the top of the file). In order to ensure that Weka can process the data, you will need to add a header row to the data file and save it as a .csv file. It doesn't matter what you name the columns; just make sure you understand them! You can do this in any text editor. Make sure you remove the headers from **pima.csv** after pre-processing.

All further activities with the numerical data will be performed using **pima.csv** as the data file; if you have correctly processed the data, you will no longer need the **pima-indians-diabetes.data** file.

Include **pima.csv** in your submission to PASTA, and PASTA will check that you have correctly normalised the file.

You do not need to perform any pre-processing on the **pima-indians-diabetes.discrete** file.

2. Classification algorithms

You will now write two classifiers to predict the class (**yes** or **no**) given some new examples. These classifiers will be automatically tested by PASTA, so you need to make sure you follow the input and output instructions carefully.

Naïve Bayes

The Naïve 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. If there is ever a tie between the two classes, choose class **yes**.

Decision Tree

The Decision Tree classifier should be implemented for nominal attributes and will be evaluated using the nominal data provided to you. Use Information Gain to choose which attribute to split on at each stage of the tree, and do not implement any pruning. When building the Decision Tree, use the definitions of Information Gain and stopping conditions as provided to you in the lectures. As before, if there is ever a tie between the two classes, choose class **yes**.

Input

Your program will need to be named **MyClassifier**, however may be written in any of the languages mentioned in the "Programming languages" section.

Your program should take 3 command line arguments. The first argument is the path to the training data file, the second is the path to the testing data file, and the third is the name of the algorithm to be executed (**NB** for Naïve Bayes and **DT** for Decision Tree).

For example, if you were to make a submission in Java, your main class would be **MyClassifier.java**, and the following are examples of possible inputs to the program:

```
$ java MyClassifier pima.csv newexamples.csv NB
$ java MyClassifier pima-discretised.csv testing.csv DT
```

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 **NB**) or a discrete value (for **DT**) for each of the non-class attributes separated by commas, followed by a class value. The training data might look like this for numeric data:

```
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
```

Or like this for nominal (discrete) data:

```
high,medium,high,high,high,low,high,very high,yes
low,high,low,high,low,low,low,high,yes
high,very high,medium,high,high,medium,high,low,no
```

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 numerical input file could look like this:

```
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
```

An example discrete input file could look like this:

```
low,low,high,medium,high,low,medium,low,yes
high,high,low,high,low,high,low,high,yes
high,very high,medium,high,high,low,high,low,no
```

Note: your program should be able to handle any number of attributes; not just the 8 attributes from **`pima.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.

You can assume that your program will only be given numerical data for **NB** and discrete data for **DT**.

The following examples show how the program would be run for each of the submission languages, assuming we want to run the NB classifier, the training data is in a file called **`training.txt`**, and the testing data is in a file called **`testing.txt`**.

Python (version 3.7.0):

```
python MyClassifier.py training.txt testing.txt NB
```

Java (version 8):

```
javac MyClassifier.java
java MyClassifier training.txt testing.txt NB
```

C (gcc version 6.3.0):

```
gcc –lm -w -std=c99 –o MyClassifier MyClassifier.c *.c
./MyClassifier training.txt testing.txt NB
```

C++ (gcc version 6.3.0):

```
g++ –c MyClassifier.cpp *.cpp *.h
gcc –lstdc++ -lm –o MyClassifier *.o
./MyClassifier training.txt testing.txt NB
```

MATLAB (R2018a):

```
mcc -m -o MyClassifier -R -nodisplay -R -nojvm MyClassifier
./run_MyClassifier.sh <MATLAB_directory> training.txt testing.txt NB
```

Note: MATLAB must be run this way (compiled first) to speed up MATLAB running submissions. The arguments are passed to your **`MyClassifier`** function as strings. For example, the example above will be executed as a function call like this:

```
MyClassifier('training.txt', 'testing.txt', 'NB')
```

Output

Your program will output to standard output (a.k.a. "the console"). The output should be one class value (**yes** or **no**) per line – each line representing your program's classification of the corresponding line in the input file. An example output could look as follows:

```
yes
no
yes
```

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 Piazza or ask your tutor. Since your program will be automatically tested by PASTA, it is important that you follow the instructions exactly.

Include your classifier code in your submission to PASTA, and PASTA will test whether you have correctly implemented the two classifiers.

Make sure that your main code file (**`MyClassifier.*`**) is at the top level of the submission folder; i.e. don't put **`MyClassifier.*`** in a subfolder, or PASTA will not be able to find it.

PASTA will only be testing your code for correctness; i.e. does your code get the results that it should, given some expected inputs? Your classifiers are not expected to be able to classify new examples with some specified level of accuracy.

Your program should also be able to output the Decision Tree that is built when running the Decision Tree classifier. This output will NOT be tested automatically, however should only be outputted if the user requests it so (for example with a certain argument flag). DO NOT make your program output this diagram every time, as it will interfere with the testing system's ability to process your output. You will require this output for your report.

3. Evaluating your classifiers

Now that you have implemented some classifiers, you need to evaluate them; i.e. find out how well they actually perform as classifiers.

Implementation

In order to evaluate the performance of the classifiers, you will have to implement 10-fold stratified cross-validation as an extension to your classifier code. Your program should be able to show the algorithm's average accuracy over the 10 folds. This information will be required to complete the report; you need to know the average accuracy of your **NB** and **DT** algorithms.

If you are unsure how to modify your code to show average accuracy using 10-fold cross validation, see Appendix 1 – 10-fold cross validation for a diagram describing how 10-fold cross validation works as an evaluation method.

Make sure your folds are stratified, otherwise your accuracy results can be wrong. For more information see Appendix 2 – Stratification.

Your implementation of 10-fold stratified cross-validation is not required to be part of your PASTA submission, as it will not be automatically tested, however feel free to include it in your submission for reference.

Sample 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
```

You are **not** required to create an equivalent file for the discretised data set.

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.

Include **`pima-folds.csv`** in your submission to PASTA, and PASTA will check that you have correctly created the folds and applied stratification.

4. Feature selection

Correlation-based feature selection (CFS) is a method for selecting a subset of the original features (attributes). It searches for the best subset of features, where best is defined by a heuristic which considers how good the individual features are at predicting the class and how much they correlate with the other features. Good subsets of features contain features that are highly correlated with the class and uncorrelated with each other.

Load the **`pima.csv`** file in Weka, and apply CFS to reduce the number of features. It is available from the "Select attributes" tab in Weka. Use "Best-First Search" as the search method. Save the CSV file with the reduced number of attributes (this can be done in Weka) and name it **`pima-CFS.csv`**. Repeat the same for the discretised data and save the result as **`pima-discretised-CFS.csv`**.

Note: As before, in order to ensure Weka can understand the data, you'll need to add headers. Once you are done processing, remove the headers

Include **`pima-CFS.csv`** and **`pima-discretised-CFS.csv`** in your submission to PASTA, and PASTA will check that you have correctly applied CFS to the input files.

5. Evaluating Weka's classifiers

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), Naïve Bayes (NB), Decision Tree (DT; J48 in Weka), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM; SMO in Weka), Bagging (Bagg), Boosting (Boost, AdaBoostM1 in Weka) and Random Forest (RF). For Bagging and Boosting, combine decision trees (J48) not the default REPtrees or decision stumps.

Compare the performance of the Weka's classifiers with your Naïve Bayes and Decision Tree classifiers (again using nominal data for the DT and tree-based ensembles from Weka, and numeric data for the NB classifier from Weka – see the tables in the next section). Do this without feature selection (using **`pima.csv`** or relevant nominal data) and with CFS feature selection (using **`pima-CFS.csv`** or **`pima-discretised-CFS.csv`**).

6. Report

You will have to describe your analysis and findings in a report similar to a research paper. Your report should include (at least) the following 5 sections. There is no minimum or maximum length for the report – you will be marked on the quality of the content that you provide. Make sure you put enough detail in each section.

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

Aim

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

Data

This section should describe the dataset, mentioning the number of attributes and classes. It should also briefly describe the CFS method and list the attributes selected by the CFS.

Results and discussion

The accuracy results should be presented (in percentage, using 10-fold cross validation) in the following two tables where MyNB and MyDT (shown in blue) are your implementations of the NB and DT algorithms, evaluated using your stratified 10-fold cross validation. For the DT classifier in Weka, present two results – for an unpruned tree and for a pruned tree (using the default pruning option).

Classifier accuracy

Numeric Data	ZeroR	1R	1NN	5NN	NB	MLP	SVM	MyNB
No feature selection								
CFS								

Nominal Data	DT unpruned	DT pruned	MyDT	Bagg	Boost	RF
No feature selection						
CFS						

DT diagrams

Build DTs using the whole training data and include a DT diagram for each of the three DT classifiers (MyDT and the two Weka versions, pruned and unpruned). To do this in Weka, select "Test option: Use training set".

As the diagram for unpruned trees can be very large, leaving the diagram as a text-based diagram (like the ones produced in Weka) is acceptable.

Discussion

In the discussion, compare the performance of the classifiers, with and without feature selection. Compare your implementations of NB and DT with Weka's.

Discuss the effect of the feature selection – did CFS select a subset of the original features, and if so, did the selected subset make intuitive sense to you? Was the feature selection beneficial, i.e. did it improve accuracy, or have any other advantages? Why do you think this is the case?

How does your DT classifier compare with the unpruned and pruned DT generated by Weka? Discuss the role and effect of pruning.

Compare the accuracy of the tree-based classifiers (DT, Bagging, Boosting and RF).

Include anything else that you consider important.

Conclusion

Summarise your main findings and suggest future work.

Reflection

Write one or two paragraphs describing the most important thing that you have learned throughout this assignment.

> Include your report in your submission for PASTA, as that is where it will be marked. The report will be marked by hand after the due date of the assignment.
>
> Remember to also submit your report (only the report) in Canvas for plagiarism checking using TurnItIn.

7. Submission details - PASTA

This assignment is to be submitted electronically via the PASTA submission system. If you are submitting the assignment as a pair, only one of you needs to submit on PASTA, and it will show up for both of you. If you are submitting on your own, you still need to set up a group in PASTA; follow the steps below for instructions.

Individual submissions setup

The first thing you must do is create an individual group on PASTA. This is due to a limitation of PASTA. To create a group, follow the instructions below:

1. Click on the "Group Management" button (3 people icon), next to the submit button.
2. Click on the plus button in the bottom right to add a new group.
3. Scroll to the bottom of the list of groups and click on "Join Group" next to the group you just created.
4. Click on "Lock Group" to lock the group and stop others from joining the group (optional).

Pair submissions setup

The first thing you must do is create/join a group on PASTA. Follow the instructions below:

1. Click on the "Group Management" button (3 people icon), next to the submit button.
2. If your pair has not yet formed a group on PASTA, click on the plus button in the bottom right to add a new group, otherwise go to step 3.
3. Click on "Join Group" next to your group in the "Other Existing Groups" section.
4. If you wish to stop anyone from joining your group, click on "Lock Group".

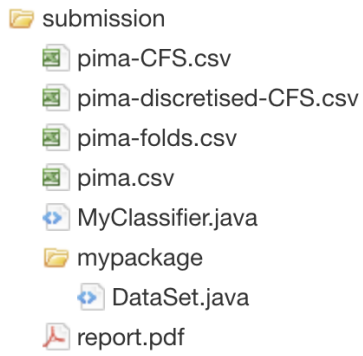
All submissions

Your submission should be zipped together in a single .zip file and include the following:

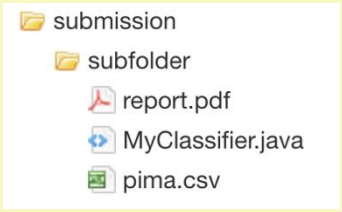
- The report in PDF format.
- The source code with a main program called **`MyClassifier`**. Valid extensions are `.java`, `.py`, `.c`, `.cpp`, `.cc`, and `.m`.
- Four data files: **`pima.csv`**, **`pima-CFS.csv`**, **`pima-folds.csv`** and **`pima-discretised-CFS.csv`**.

Upload your submission on PASTA under **Assignment 2 – Classification (Advanced)**. Make sure you tick the box saying that you're submitting on behalf of your group (even if you're working individually). The submission won't work if you don't.

When submitted to PASTA, a valid submission might look like this:



Note: Make sure you only zip up the files needed for the submission; not a folder containing these files. If your submission looks like this in PASTA, you have included a folder and PASTA will not be able to read your files:



8. Submission details – Canvas

You are also required to submit the report part of your assignment in Canvas (in the "Assignment2 Report" dropbox), for checking for plagiarism using TurnItIn. If you work in a pair, only one of you needs to submit.

9. Marking criteria

[12 marks] Code – based on the passed tests in PASTA; automatic marking

[12 marks] Report – manual marking

[1 mark] Introduction

- What is the aim of the study?
- Why is this study (or the problem) important?

[1 mark] Data – well explained

- Dataset – brief description of the dataset
- Attribute selection – brief summary of CFS and a list of the selected attributes

[6.5 marks] Results and discussion

- All results presented
- Correct and deep discussion of the results
- Effect of the feature selection – beneficial or not (accuracy, other advantages)
- Comparison between the classifiers (accuracy, other advantages)
- Comparison between the DT classifiers and discussion of pruning

- Comparison between the tree-based classifiers

[2 marks] Conclusions and future work

- Meaningful conclusions based on the results
- Meaningful future work suggested

[0.5 marks] Reflection (meaningful and relevant personal reflection)

[1 mark] English and presentation

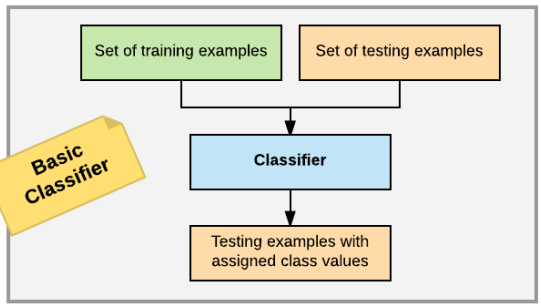
- Academic style, grammatical sentences, no spelling mistakes
- Good structure and layout; consistent formatting

10. Appendix 1 – 10-fold cross validation

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

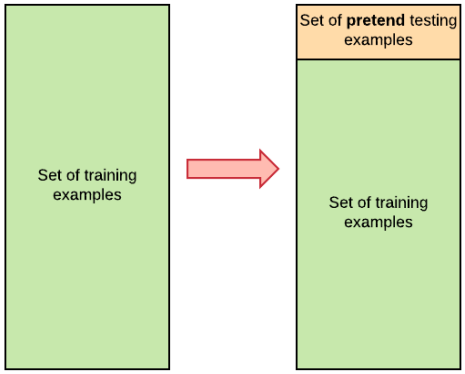
Testing Example
Contains:
1. a set of attributes
2. no class value; we have not seen this example before

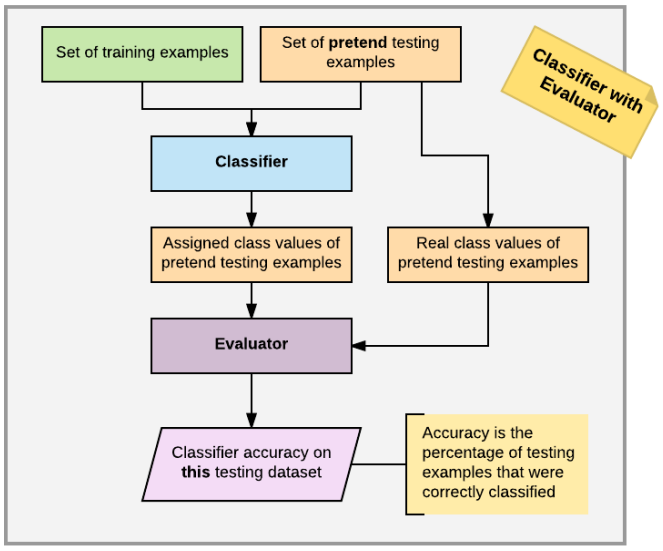
Classifier
1. Use training data only to build the classifier
2. Run the testing data through the classifier one example at a time to assign a class value to each



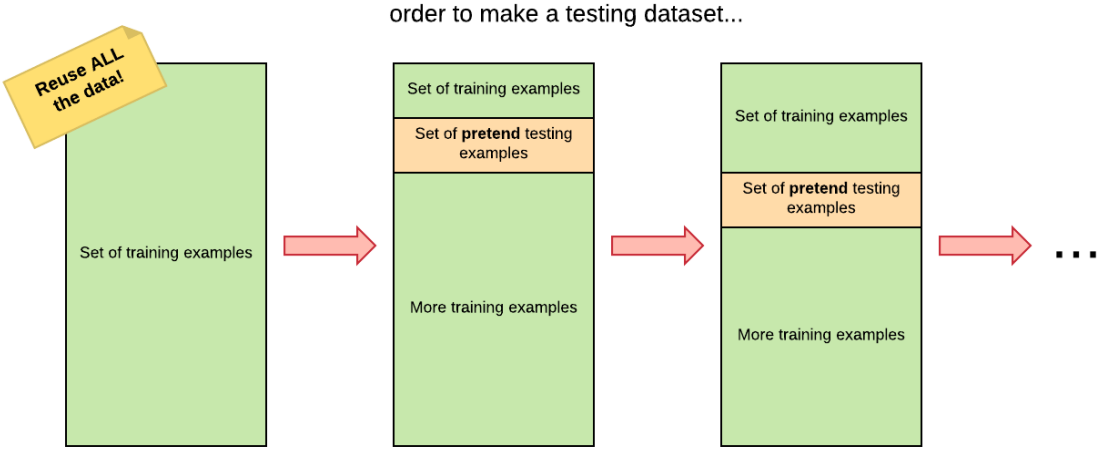
Okay, so what if we only have a training dataset?
How do we know if the classifier is performing well?

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?

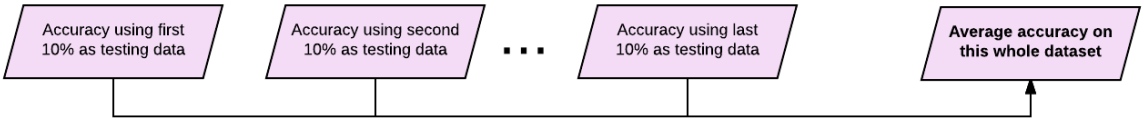




But now we've had to sacrifice some of the training data in order to make a testing dataset...



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



11. Appendix 2 – Stratification