# Assignment 1: Classification

## Key information

### Deadlines

**Submission**: 11:59pm, 16 September, 2022 (Friday week 7, Sydney time)

### 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 15 marks = 15% of your final mark.

Your code will be marked for correctness. A few marks will be allocated for style – meaningful variable names and comments.

The assignment can be completed individually or in groups of 2 students. No more than 2 students are allowed. See the submission details section for more information about how to submit.

### Submission

This assignment must be written in **Python** in the Jupyter Notebook environment. A Jupyter Notebook template is provided. Your implementation should use the same suite of libraries that we have used during the tutorials, such as **scikit-learn**, **numpy** and **pandas**.

The assignment will be submitted in Canvas. You need to submit two versions of your code: .ipynb and .pdf. There are two submission boxes – "Assignment 1 ipynb" for the .ipynb file and "Assignment 1 pdf" for the .pdf file.

Before you submit, if you work in a group, you will need to create a group in Canvas. Under the "People" page on Canvas, select the "A1 Group" tab. You and your group partner should choose one of the empty groups listed under this tab, and both join it. Groups have a maximum of 2 members. **If you are completing the assignment individually, you don't need to create a group.**

The submission file should contain the SID number(s) and should be named like this:

- a1-SID.ipynb (.pdf) for a student working individually, where SID is the student's SID number
- a1-SID1-SID2.ipynb (.pdf) for a group of 2 students, where SID1 and SID2 are the SIDs of the two students

## Task

In this assignment you will investigate a real dataset by implementing multiple classification algorithms. You will first pre-process the dataset by replacing missing values and normalising the dataset with a min-max scaler. You will then evaluate the performance of multiple classification algorithms: K-Nearest Neighbour, Logistic Regression, Naïve Bayes, Decision Tree, Support Vector Machine, Bagging, AdaBoost,

Gradient Boosting and Random Forest, using the stratified 10-fold cross validation method. You will also apply a grid search to find the best parameters for some of these classifiers.

## 1. Data loading, pre-processing and printing

The dataset for this assignment is the Breast Cancer Wisconsin. It contains 699 examples described by 9 numeric attributes. There are two classes – **class1,** corresponding to benign breast cancer tumours, and **class2**, corresponding to malignant breast cancer tumours. The features are computed from a digitized image of a biopsy sample of breast tissue for a subject.

The dataset should be downloaded from Canvas: **breast-cancer-wisconsin.csv**. This file includes the attribute (feature) headings and each row corresponds to one individual. Missing attributes in the dataset are recorded with a **'?'**.

You will need to pre-process the dataset, before you can apply the classification algorithms. Three types of pre-processing are required: filling in the missing values, normalisation and changing the class values. After this is done, you need to print the first 10 rows of the pre-processed dataset.

1. Filling in the missing attribute values - The **missing attribute values** should be replaces with the mean value of the column using sklearn.impute.**SimpleImputer**.
2. Normalising the data **- Normalisation** of each attribute should be performed using a min-max scaler to normalise the values between [0,1] with sklearn.preprocessing.**MinMaxScaler**.
3. Changing the class values - The classes **class1** and **class2** should be changed to **0** and **1** respectively.
4. Print the first 10 rows of the pre-processed dataset. The feature values should be formatted to 4 decimal places using .4f, the class value is an integer.

   For example, if your normalised data looks like this:

| Clump Thickness | Uniformity of Cell Size | Uniformity of Cell Shape | Marginal Adhesion | Single Epithelial Cell Size | Bare Nuclei | Bland Chromatin | Normal Nucleoli | Mitoses | Class |
|---|---|---|---|---|---|---|---|---|---|
| 0.1343 | 0.4333 | 0.5432 | 0.8589 | 0.3737 | 0.9485 | 0.4834 | 0.9456 | 0.4329 | 0 |
| 0.1345 | 0.4432 | 0.4567 | 0.4323 | 0.1111 | 0.3456 | 0.3213 | 0.8985 | 0.3456 | 1 |
| 0.4948 | 0.4798 | 0.2543 | 0.1876 | 0.9846 | 0.3345 | 0.4567 | 0.4983 | 0.2845 | 0 |

   Then your program should print:

```
0.1343,0.4333,0.5432,0.8589,0.3737,0.9485,0.4834,0.9456,0.4329,0
0.1345,0.4432,0.4567,0.4323,0.1111,0.3456,0.3213,0.8985,0.3456,1
0.4948,0.4798,0.2543,0.1876,0.9846,0.3345,0.4567,0.4983,0.2845,0
```

   (You need to print the first 10 rows not the first 3.)

**Your program must be able to correctly infer X and y from the file.** Do not hard-code the number of features and examples - do not set then to 699 and 9 as in the breast cancer dataset. We will test your code on a dataset with different number of features and examples than the given breast cancer dataset.

## 2. Defining functions for the classification algorithms

### Part 1: Cross validation without parameter tuning

You will now apply multiple classifiers to the pre-processed dataset, in particular: Nearest Neighbor, Logistic Regression, Naïve Bayes, Decision Tree, Bagging, Ada Boost and Gradient Boosting. All classifiers should use the **sklearn** modules from the tutorials**.** All random states in the classifiers should be set to **random_state=0**.

You need to evaluate the performance of these classifiers using 10-fold stratified cross validation from **`sklearn.model_selection.StratifiedKFold`** with these options:

**cvKFold=StratifiedKFold(n_splits=10, shuffle=True, random_state=0)**

**You will need to pass cvKFold (the stratified folds) as an argument when calculating the cross-validation accuracy, not cv=10 as in the tutorials. This ensures that random_state=0.**

For each classifier, write a function that accepts the required input and returns the average cross-validation score:

```
def exampleClassifier(X, y, [options]):
        …
        return scores.mean()
```

where **X** contains the attribute values and **y** contains the class (as in the tutorial exercises).

More specifically, the headers of the functions for the classifiers are given below:

#### K-Nearest Neighbour

```
def kNNClassifier(X, y, k)
        …
        return scores.mean()
```

It should use the `KNeighborsClassifier` from `sklearn.neighbors`.

#### Logistic Regression

```
def logregClassifier(X, y)
        …
        return scores.mean()
```

It should use `LogisticRegression` from `sklearn.linear_model`.

#### Naïve Bayes

```
def nbClassifier(X, y)
        …
        return scores.mean()
```

It should use `GaussianNB` from `sklearn.naive_bayes`

#### Decision Tree

```
def dtClassifier(X, y)
```

**…**
**return scores.mean()**

It should use DecisionTreeClassifier from sklearn.tree, with information gain (the entropy criterion)

### Ensembles: Bagging, Ada Boost and Gradient Boosting

**def bagDTClassifier(X, y, n_estimators, max_samples, max_depth)**
**…**
**return scores.mean()**

**def adaDTClassifier(X, y, n_estimators, learning_rate, max_depth)**
**…**
**return scores.mean()**

**def gbClassifier(X, y, n_estimators, learning_rate)**
**…**
**return scores.mean()**

These functions should implement Bagging, Ada Boost and Gradient Boosting using `BaggingClassifier`, `AdaBoostClassifier` and `GradientBoostingClassifier` from `sklearn.ensemble`. Bagging and Ada Boost should combine decision trees and use information gain.

## Part 2: Cross validation with parameter tuning

For two other classifiers, Linear SVM and Random Forest, we would like to find the best parameters using grid search with 10-fold stratified cross validation (`GridSearchCV` in sklearn).

The data should be split into training and test subsets using `train_test_split` from `sklearn.model_selection` with stratification and random_state=0 (as in the tutorials but with random_state=0).

**You will need to pass cvKFold (the stratified folds) as an argument to GridSearchCV, not cv=10 as in the tutorials. This ensures that random_state=0.**

Write the following functions:

### Linear SVM

**def bestLinClassifier(X,y)**
**…**
**return (appropriate values so that the required printing can be done)**

It should use SVC from `sklearn.svm`.

The grid search should consider the following values for the parameters C and gamma:

C = {0.001, 0.01, 0.1, 1, 10, 100}

gamma = {0.001, 0.01, 0.1, 1, 10, 100}

The function should return appropriate values, so that best parameters found, the best cross-validation accuracy and the test set accuracy can be printed when calling this function, see the next section.

### Random Forest

**def bestRFClassifier(X,y)**

It should use RandomForestClassifier from `sklearn.ensemble` with information gain and max_features set to 'sqrt'.

The grid search should consider the following values for the parameters n_estimators and max_leaf_nodes:

n_estimators = {10, 20, 30, 50, 100}

max_leaf_nodes = {4, 10, 16, 20, 30}

The function should return appropriate values, so that best parameters found, the best cross-validation accuracy and the test set accuracy can be printed when calling this function, see the next section.

## 3. Running the classifiers and printing the results

Run the classifiers from the previous section on the pre-processed dataset and print the results.

For Part1, set the parameters as follows (this is already done for you in the template):

```
#KNN
k=3

#Bagging
bag_n_estimators = 50
bag_max_samples = 100
bag_max_depth = 5

#AdaBoost
ada_n_estimators = 50
ada_learning_rate = 0.5
ada_bag_max_depth = 5

#GB
gb_n_estimators = 50
gb_learning_rate = 0.5
```

The printing should look like this but with the correct numbers (these are random numbers):

```
kNN average cross-validation accuracy: 0.8234
LR average cross-validation accuracy: 0.8123
NB average cross-validation accuracy: 0.7543
DT average cross-validation accuracy: 0.6345
Bagging average cross-validation accuracy: 0.8765
AdaBoost average cross-validation accuracy: 0.7165
```

```
GB average cross-validation accuracy: 0.9054

SVM best C: 0.0100
SVM best gamma: 10.0000
SVM cross-validation accuracy: 0.8676
SVM test set accuracy: 0.8098

RF best n_estimators: 10
RF best max_leaf_nodes: 16
RF cross-validation accuracy: 0.8600
RF test set accuracy: 0.8321
```

Format all numbers to 4 decimal places using .4f, except n_estimators and max_leaf_nodes which should be formatted as integers.

## Academic honesty – very important

Please read the University policy on Academic Honesty very carefully:
https://sydney.edu.au/students/academic-integrity.html

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 his/her work available for copying

The University penalties are severe and include: 1) a permanent record of academic dishonesty on your student file, 2) mark deduction, ranging from 0 for the assignment to Fail for the course and 3) expulsion from the University and cancelling of your student visa.

If there is a suspected case, the **investigation takes several months**. Your mark will not be finalised until the investigation is completed. This may create **problems enrolling in other courses next semester** (COMP5318 is a pre-requisite for many courses) or **delaying your graduation**. Going through the investigation is also **very stressful**.

In addition, the Australian Government passed a new legislation last year (Prohibiting Academic Cheating Services Bill) that makes it a **criminal offence** to provide or advertise academic cheating services - the provision or undertaking of work for students which forms a substantial part of a student's assessment task.

Do not confuse legitimate co-operation and cheating! You can discuss the assignment with other students, but you (if you work individually) or your group (if you work in pairs) must write your own code.

To detect code similarity in this assignment, we will use TurnItIn and MOSS which are **extremely good**. If you cheat, the chances that you will be caught are very high.

Do not even think about engaging in plagiarism or academic dishonesty, it is not worth it. **Be smart and don't risk your future or break the law by engaging in plagiarism and academic dishonesty!**