Assignment 1: Rice Classification

Key information

Deadlines

Submission: 11:59pm, 7 April 2025 (Monday week 7)

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 10 marks = 10% of your final mark.

Please see the marking rubric on Canvas. Most of the marks are for your code – it will be marked for correctness. A few marks will be allocated for code style – meaningful variable names and comments, and also for reflection.

We will run your code. If it doesn't run, you will get 0 marks for the parts that don't run.

Submission

The 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 submission is in Canvas.

Groups

The assignment should be completed in pairs (groups of 2 students). No more than 2 students are allowed.

Firstly (as announced on Ed in week 2), you need to create a group and register it in Canvas->People. There are 2 tabs (group sets) in People: A1 group-set1 and A1 group-set2. You must **register your group under one of these group-sets, not both.** We had to create 2 group-sets because Canvas has a limitation for the number of groups under one tab and our course is big.

Choose one of the empty groups listed under one of these tabs and join it. You and your partner must join the same group.

Submission instructions:

- When you are ready to submit your assignment, you need to submit it on behalf of the group in the corresponding submission box, not individually. Only one student from the group needs to submit, not both. When this is done, both students will be able to see the submission, and later see the mark (when the assignment is marked).
- You need to submit **two versions of your code: ipynb and pdf**.
- Make sure that you submit in the correct submission box! There may be penalties for incorrect submission. In summary:
    - If you have registered your group under "**A1 group-set1**", submit your ipynb code in "Submission: Assignment 1 ipynb for **A1 group-set1**" and your pdf in "Submission: Assignment 1 pdf for **A1 group-set1**"

- Similarly, if you have registered your group under "**A1 group-set2**", submit your ipynb code in "Submission: Assignment 1 ipynb for **A1 group-set2**" and your pdf in "Submission: Assignment 1 pdf for **A1 group-set2**"

It is important to follow the submission instructions carefully as otherwise **your mark may not be recorded correctly.**

File names and student names

- The file names should include your **group number**, e.g. a1-set1-group3.ipynb (.pdf), if you are from group 3 in group-set1 or a1-set2-group180.ipynb (.pdf) if you are from group 180 in group-set2.
- In the Jupyter Notebook, include your group name and your SIDs (as shown in the template). Do not include your name as the marking is anonymous.

A pdf of your Jupyter notebook can be generated using File>Download as>PDF or Print Preview > Save as PDF or File->Download as >HTML, then open the html file and save it as pdf.

Task

In this assignment you will implementing multiple classification algorithms and apply them to a real dataset. First, you will pre-process the dataset by replacing missing values and normalising the values. Second, you will implement and evaluate the performance of multiple classification algorithms: Logistic Regression, Naïve Bayes, Decision Tree, Bagging, AdaBoost and Gradient Boosting, using the stratified 10-fold cross-validation method. Third, you will apply grid search to find the best parameters for K-Nearest Neighbour, Support Vector Machine and Random Forest. Forth, you need to write a short reflection.

1. Data loading, pre-processing and printing

The dataset for this assignment is the Rice dataset which describes two varieties of rice grains. It contains 1400 examples, described with 7 features which have been extracted from rice grain images. There are two classes – class1, corresponding to rice variety "cammeo" and class2 – to rice variety "osmancik". The original dataset is bigger and available from the UCI Machine Learning repository: https://archive.ics.uci.edu/dataset/545/rice+cammeo+and+osmancik.

For this assignment, you should use the **rice-final2.csv** dataset provide on Canvas. It is a smaller and slightly modified version of the original dataset. This file includes the attribute (feature) headings and each row corresponds to one grain example. 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 replaced 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. A function **print_data** has been provided in the template to help you achieve this.

For example, you program should print the first 10 rows in **this format** (10 rows not 3, and the actual numbers, these are random numbers):

```
0.2861,0.2712,0.2086,0.4853,0.4296,0.2900,0.2247,1
0.5004,0.5781,0.5512,0.4854,0.7636,0.5374,0.7630,0
0.8077,0.8890,0.8694,0.6393,0.8541,0.8079,0.0747,0
```

Please note that we will test your code with another dataset, and your pre-processing should be written with this in mind. See the "Marking Criteria" section for more detail.

2. Defining functions for the classification algorithms

Part 1: Cross-validation without parameter tuning

You will now apply multiple classifiers to the processed dataset, in particular: 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).

Specifically, the headers of the functions for the classifiers are given below and provided in the template. You must not change them; your task is to write the body of these functions.

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`, respectively. Bagging and Ada Boost should combine decision trees and use information gain.

Part 2: Cross-validation for parameter tuning

For three other classifiers, K Nearest Neighbor, 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:

K-Nearest Neighbour

def bestKNNClassifier(X, y)

…

return (appropriate values so that the required printing can be done)

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

The grid search should consider the following values for the parameters k(n_neighbors) and p:

$k = \{1, 3, 5, 7\}$

p = {1, 2}

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.

SVM

def bestSVMClassifier(X,y)

…

return (appropriate values so that the required printing can be done)

It should use SVC from `sklearn.svm` with kernel set to 'rbf'.

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

C = {0.01, 0.1, 1, 5}

gamma = {0.01, 0.1, 1, 10}

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, 30, 60, 100}

max_leaf_nodes = {6, 12}

The function should return appropriate values, so that best parameters found, the best cross-validation accuracy, the test set accuracy, *the macro average F1 score, and the weighted average F1 score* 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 Part 1 of this assignment, set the parameters as follows (this is already done for you in the template):

```
#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):

```
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


KNN best k: 7
KNN best p: 2
KNN cross-validation accuracy: 0.7853
KNN test set accuracy: 0.5991

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
RF test set macro average F1: 0.8123
RF test set weighted average F1: 0.8261
```

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

4. Reflection

In the last cell of the template, each student needs to write a short reflection (1 paragraph):

What was the most important thing that you have learnt throughout this assignment?

Marking Criteria

The marking rubric is provided in Canvas.

We will test your program for correctness on the **rice** dataset (the given dataset) and on a second dataset, (**unknown**.csv). The unknown dataset will have the same format as the rice dataset but a different number of features and examples. Hence, do not hard-code the number of features and examples - do not set them to 1400 and 7 as in the rice dataset. For the unknown dataset, you may assume that the class variable is in the last column and that there are two classes - class1 and class2, as in the rice dataset.

To test your code before submission, we have made available another dataset (**test-before.csv**) and the correct results for it in **test-before-results.pdf**. You can run your program on this dataset and compare your results with the given correct results.

Due to differences in operating systems and environments, you may not be able to get the same results on the test-before dataset. When we mark your code, the final submission of your code and the solution code will be run on the same computer to minimise any such issues. If you are sure you have controlled for the different sources of randomness appropriately and are still getting slight differences on test-before due to your environment, this won't be an issue when your code is marked.

In your assignment submission, your Jupyter notebook needs to include **only the results on the rice dataset**, not on test-before dataset.

Academic honesty – very important

Please read the University policy on Academic Honesty 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 their work available for copying.

Do not confuse legitimate co-operation and cheating! You can discuss the assignment with other students but your group must write your own code.

You are **not allowed** to use generative AI tools for any assessment tasks in this course.

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

If there is a suspected case, the **investigation may take 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 other courses) or **delaying your graduation**. Going through the investigation is also **very stressful**.

We will use similarity detection software. 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 reputation by engaging in plagiarism and academic dishonesty!**