Assignment 1: Classification

Deadlines

Submission: 11:59pm, 23 April, 2021 (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. It must be completed individually. See the submission details sections for more information about submission in PASTA.

Your mark depends on the number of tests passed. There are no hidden tests. After each submission you can see which tests you have passed and your current mark. If you pass all tests, your mark will be 15/15.

Task description

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 multiple classification algorithms: K-Nearest Neighbour, Logistic Regression, Naïve Bayes, Decision Tree, Support Vector Machine and ensembles - Bagging, Boosting (AdaBoost, Gradient Boosting) and Random Forest. These will then be evaluated using the stratified 10-fold cross validation method. You will also apply a grid search to find the best parameters for some of the classifiers.

Programming language

This assignment must be written in **Python**. It should use the classification algorithms from the **sklearn** (scikit-learn) library used in the tutorials. You can also use the **numpy** and **pandas** libraries.

On PASTA we have the following versions:

1. Python version 3.7.0
2. Numpy: 1.18.2
3. Pandas: 1.0.3
4. Scikit-learn: 0.22.2.post1 – identical contents to 0.22.2. See: https://scikit-learn.org/stable/whats_new/v0.22.html

If you have a different version on your laptop or desktop computer, you must ensure that you code works with these versions.

Submission

This assignment must be submitted using PASTA.

VPN

PASTA is located at https://comp5318.it.usyd.edu.au/PASTA/. In order to connect to the website, you'll need to be connected to the university VPN. The University provides a Cisco VPN client https://vpn.sydney.edu.au; please read this page to find out how to connect to the VPN using the Cisco VPN client.

All students who are in in China should use the new VPN (FortiClient), which is the special VPN for accessing university resources from China. **Only students who are in China should use this VPN.** To access PASTA from China, it may also be necessary to run both the FortiClient VPN and Cisco VPN; connect to FortiClient VPN first and then to Cisco VPN.

If you are on-campus and connected to the University wireless network, you don't need a VPN to access PASTA.

Multiple submissions

PASTA will allow you to make as many submissions as you wish, and each submission will provide you with feedback on each of the components of the assignment. You **last** submission before the assignment deadline will be marked, and the mark displayed on PASTA will be the final mark for your assignment.

1. Dataset preparation

The data

The dataset for this assignment is the Breast Cancer Wisconsin. It contains 699 examples described by 9 numeric attributes. There are two classes – **class1** and **class2**. The features are computed from a digitized image of a fine needle aspirate of a breast mass of the subject. Benign breast cancer tumours correspond to **class1** and malignant breast cancer tumours correspond to **class2**.

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 **'?'**.

Data pre-processing

You will need to pre-process the dataset, before you can apply the classification algorithms. Three types of pre-processing are required:

5. The **missing attribute values** should be replaces with the mean value of the column using sklearn.impute.**SimpleImputer**.
6. **Normalisation** of each attribute should be performed using a min-max scaler to normalise the values between [0,1] with sklearn.preprocessing.**MinMaxScaler**.
7. The classes **class1** and **class2** should be changed to **0** and **1** respectively.
8. The value of each attribute should be formatted to 4 decimal places using .4f.

The correctness of your pre-processed data will be automatically tested by PASTA. Thus, you need to make sure that you follow the input and output instructions carefully.

2. Classification algorithms with 10-fold cross-validation

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 cross validation from **`sklearn.model_selection.StratifiedKFold`** with these options:

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

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

def exampleClassifier(X, y, [options]):

> **…**
>
> **return scores, 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)

It should use the `KNeighboursClassifier` from `sklearn.neighbours`.

Logistic Regression

def logregClassifier(X, y)

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

Naïve Bayes

def nbClassifier(X, y)

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

Decision Tree

def dtClassifier(X, y)

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

Ensembles

def bagDTClassifier(X, y, n_estimators, max_samples, max_depth)

def adaDTClassifier(X, y, n_estimators, learning_rate, max_depth)

def gbClassifier(X, y, n_estimators, learning_rate)

These functions should implement Bagging, Ada Boost and Gradient Boosting using `BaggingClassifier`, `AdaBoostClassifier` and `GradientBoostingClassifier` from `sklearn.ensemble`. All should combine Decision Trees with information gain.

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

Hint: You need to pass StratifiedKFold 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)

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 print the best parameters found, the best-cross validation accuracy score and the best test set accuracy score, see Section 4.

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}

max_leaf_nodes = {4, 16}

The function should print the best parameters found, best-cross validation accuracy score and best test set accuracy score, see Section 4.

4. Submission details - PASTA

This assignment is to be submitted electronically via the PASTA submission system.

How to submit to PASTA

Your program will need to be named MyClassifier. If you use Jupyter notebook to write your code, please do the following:

1. Ensure that you have a main method that is able to parse 3 command line arguments. PASTA will run your code with 3 different arguments. Details about the 3 arguments are given below.
2. Ensure that you export your Jupyter notebook as Python code and only submit the Python code to PASTA.

Before submitting to PASTA please make sure that all your Python programs (MyClassifier.py and all other supporting .py programs, if you have any) are zipped together. If you only have 1 Python program (MyClassifier.py), you do not need to zip it. Just submit that file to PASTA.

If you are zipping, please do **not** zip the folder containing your MyClassifier.py program (and your other supporting programs, if any). Zip only the .py files. To do this, select the MyClassifier.py file (and your other supporting scripts if any), right click, and compress/zip them. Then submit the resulting zip file to PASTA.

Detailed submission instructions:

- Find Assignment 1 and press "Submit" (the red icon on the right). Then press "Choose File" and attach the file, then press "I accept" after reading the University policy on academic honesty and agreeing with it.
- If you see a message indicating that your code is queued for testing, this means that your code has been uploaded successfully.
- If you see a red error box, you need to fix any problems indicated before your submission will be accepted.
- Once your program has been tested, the page will tell you to refresh for results, so refresh the page. You should see green and red boxes. Each box corresponds to a test; a red box indicates that your program has failed the respective test and a green box indicates that your program has passed the test.
- If you have red boxes, you can see which tests were not passed by clicking on Assignment 1. Correct the errors (go back to Jupyter, re-write your code and test it carefully) and then submit again in PASTA.

Important:

- Do not treat PASTA as your development environment! You must firstly test your code on your laptop or desktop computer to make sure it runs without errors and only after that submit to PASTA. If you don't do this, there will be too many submissions in the queue and the waiting time to run your code will become very long. In this case, we may need to limit the number of submissions per student; currently it is unlimited.
- Start working on the assignment as soon as possible and do not delay the submission till the last moment as PASTA will get very busy and you risk being late.

Input

Your program should take 3 command line arguments:

1. The first argument is the path to the data file.
2. The second argument is the name of the algorithm to be executed or the option for print the pre-processed dataset:
   a. **NN** for Nearest Neighbour.
   b. **LR** for Logistic Regression.

  c. **NB** for Naïve Bayes.
  d. **DT** for Decision Tree.
  e. **BAG** for Ensemble Bagging DT.
  f. **ADA** for Ensemble ADA boosting DT.
  g. **GB** for Ensemble Gradient Boosting.
  h. **RF** for Random Forest.
  i. **SVM** for Linear SVM.
  j. **P** for printing the pre-processed dataset

3. The third argument is **optional**, and should **only be supplied** to algorithms which require parameters, namely NN, BAG, ADA and GB. It is the path to the file containing the parameter values for the algorithm. The file should be formatted as a csv file like in the following examples:
   a. NN (note the capital K); an example for 5-Nearest Neighbour:
   ```
   K
   5
   ```
   b. BAG:
   ```
   n_estimators,max_samples,max_depth
   100,100,2
   ```
   c. ADA:
   ```
   n_estimators,learning_rate,max_depth
   100,0.2,3
   ```
   d. GB:
   ```
   n_estimators,learning_rate
   100,0.2
   ```

   For algorithms which do not require any parameters (LR, NB, DT, RF, SVM, P), the third argument should not be supplied.

The file paths (the first and third arguments) represent files that will be supplied to your program for reading. You can test your submission using any files you like, but PASTA will provide your submission with its own files for testing, so do not assume any specific filenames.

Your program must be able to correctly infer X and y from the file. Please do not hard-code the number of features and examples. For the last few tests we will use a dataset with different number of features and examples than the given breast cancer dataset. The new dataset will contain a header line and the last column will correspond to the class, as the given breast cancer dataset.

The following examples show how your program would be run:

1. We want to run the k-Nearest Neighbour classifier, the data is in a file called **breast-cancer-wisconsin-normalised**.csv, and the parameters are stored in a file called **param.csv**:

```
python MyClassifier.py breast-cancer-wisconsin-normalised.csv NN param.csv
```

2. We want to run Naïve Bayes and the data is in a file called breast-cancer-wisconsin-normalised.csv.

```
python MyClassifier.py breast-cancer-wisconsin-normalised.csv NB
```

3. We want to run the data pre-processing task and the data is in a file called **breast-cancer-wisconsin.csv:**

```
python MyClassifier.py breast-cancer-wisconsin.csv P
```

Only option P assumes non-pre-processed input data (and it needs to pre-process and print this data). All other options (NN, LR, etc) assume already pre-processed data (i.e. normalized, scaled, missing values replaced, etc) and don't need to do any data pre-processing.

Output

Your program will output to standard output (i.e. "the console").

Pre-processing task

For option P, you will need to output your pre-processed data onto the console using the print command.

Let's assume your normalised data looks like the following:

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 output should look like this:

```
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 must print the feature values to 4 decimal places as in the example above, e.g. 0.1 should be printed as 0.100. The class value is an integer.

Note also that there is no new line character after the last line. To print the last line, ensure you pass empty character as the end parameter for the print statement.

```
print(data, end='')
```

Classifiers with 10-fold cross validation task

Your classifier should only print the mean accuracy score with precision of 4 decimal places using .4f. Note that .4f does rounding too. For example, if your mean accuracy is 0.987689, your program output should look as follows:

```
0.9877
```

Note: There is no new line character after the accuracy. See above how to print like this.

Parameter tuning task

For Linear SVM, your program should output **exactly 4 lines**. The first line contains the optimal C value, the second line contains the optimal gamma value, and the third line contains the best cross-validation accuracy score formatted to 4 decimal places using .4f and the fourth line contains the test set accuracy score also formatted to 4 decimal places. For instance, if the optimal C and gamma values are 0.001 and 0.1 respectively, and the two accuracies are 0.999874 and 0.952512, your program output should look like this:

```
0.001
0.1
0.9999
0.9525
```

For Random Forest, your program should output **exactly 4 lines.** The first line contains the optimal n_estimators, the second line contains the optimal max_leaf_nodes, the third line contains the best cross validation accuracy score truncated to 4 decimal places using .4f and the fourth line contains the test set accuracy score also truncated to 4 decimal places. For instance, if the optimal n_estimators and max_leaf_nodes are 10 and 4 respectively, and the two accuracies are 0.997623 and 0.87243, your program output should look like this:

```
10
4
0.9976
0.8724
```

5. Academic honesty

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. 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 another student, this is a legitimate collaboration, but you cannot complete the assignment together – this is an individual assignment and everyone must write their own code.

To detect code similarity in this assignment, we will use the system MOSS which is **extremely good**. If you cheat, the chances that you will be caught are very high. **Be smart and don't risk your future or break the law by engaging in plagiarism and academic dishonesty!**