CITS5508 Machine Learning
Semester 1, 2020

Lab Sheet 3

Assessed, worth 10%. Due: 11:59pm, Monday 20th April 2020

1 Outline

This lab sheet consists of two small projects. The first project asks you to train an ensemble classifier (Voting classifier) and report its performance in terms of its F1 score on the test set. The second project asks you to train and test two Random Forest classifiers. This lab sheet is a good practical exercise to test your understanding of the techniques covered in Chapters 6–7.

2 Submission

Name your Jupyter Notebook files as **lab03proj1.ipynb** and **lab03proj2.ipynb** for the two projects below and submit them to LMS before the due date and time shown above. You can submit your files multiple times. Only the latest version will be marked.

3 Project 1

Your tasks for this project is to train and evaluate the performance of a Voting classifier in classifying the Cellular Localization Sites of Proteins on some E. Coli bacteria data. Your Voting classifier should comprise three individual classifiers: an SVM classifier, a Logistic Regression classifier, and a Gradient Descent classifier.

The data file and a brief description file are available from a web repository managed by the UCI (University of California at Irvine) Centre for Machine Learning and Intelligent Systems. See

https://archive.ics.uci.edu/ml/datasets/ecoli

You should be able to find the data in the *ecoli.data*[1] file and description about data in the *ecoli.names* file. You should save the data file in the same directory as your Jupyter Notebook file. Although the data file name does not end with "*.csv*", you should be able to read it in using the `read_csv` function as usual. You will need to hard code the column headings in your Python code as they are not in the data file. The data file has nine columns. The first eight columns are the features and the last column is the class label. There are 336 instances in total.

- You will need to think about what to do with non-numerical data. Out of the eight classes, you need to remove those classes having less than 10 instances as it is not possible to classify them. Provide some plots for data visualization after you have successfully read in the data.
- Perform an 80/20 split of the data to form your training set and test set. Once this is done, the same training set and test set should be used for all the classifiers.
- You will need to consider whether any data normalization is needed. Please describe your data normalization step in your markdown cell(s). There is no need to use the `Pipeline` class. Note that data normalization should be performed on the training set and test set separately.

[1]If you use the MacOS, the operating system may automatically rename the downloaded file as *ecoli.data.txt*. You will need to rename the file back to *ecoli.data*. Otherwise, your code would not work when we mark it.

- There is no need to do grid search for finding optimal hyperparameters for each classifier. You should be able to get reasonably good classification results for this dataset even using the default hyperparameter values. Based on the settings of these hyperparameters, you can then make small adjustment to see if the performance can be improved or not.
- For all the individual classifiers and the Voting classifier, report their classification accuracies, F1 scores, and confusion matrices on the test set. For each confusion matrix, you are required to display it in a diagram. You can use the `plot_confusion_matrix` function given by the author (see the `ipynb` file for Chapter 3) to show each confusion matrix.

Hints:

- When calling the function `read_csv`, there is an optional argument that you can use to specify the character that separates the columns of the data.
- The data cleaning process for this project involves removing some data instances. You can use various functions, such as `loc` and `drop`, from the `pandas.DataFrame` package to remove rows and columns of the `DataFrame` object. However, even after some rows are correctly removed, the index locations of the remaining rows in the `DataFrame` object are not automatically updated. This means that you would still be able to access the removed rows and you would see NAN (meaning *not a number*, i.e., *undefined*) for those rows. To overcome this problem, you will need to explicitly call the `reset_index` function to renumber the index locations. After performing the data split, you should check that you don't have any NAN values in both the training and test sets.

4 Project 2

In the UCI Machine Learning Repository website above, there is a Parkinsons dataset:

http://archive.ics.uci.edu/ml/datasets/Parkinsons

which has 23 attributes and two class labels: "healthy" and "unhealthy" (i.e., having the Parkinsons disease). The class labels appear under the *status* column. The data (as a text file in *csv* format) and description about the attributes can be downloaded, respectively, from

http://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/parkinsons.data
http://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/parkinsons.names

Same as project 1, you should save the data file in the same directory as your Jupyter Notebook file.

- Consider performing appropriate data cleaning. Are there any columns that should be removed? Perform this step in Python and provide some explanation.
- Divide the data into training and test sets (using 80/20 split). Investigate whether data normalization is essential for the Random Forest classifier. You may need to iterate the step in the next bullet point and the step here a few times to find out.
- Implement two Random Forest classifiers with different hyperparameter values. Report and compare the F1 scores of the two classifiers.

5 Penalty on late submissions

See the URL below about late submission of assignments:
https://ipoint.uwa.edu.au/app/answers/detail/a_id/2711/~/consequences-for-late-assignment-submission