CITS5508 Machine Learning
Semester 1, 2022

Lab Sheet 2

Assessed, worth 10%. Due: 11:59pm, Friday 25th March 2022

1 Outline

In this lab sheet, you will learn to develop Python code for a small classification task. You will also learn the *Markdown* syntax to put comments and explanation in your Jupyter Notebook file to improve the presentation of your work.

2 Submission

Name your Jupyter Notebook file as **lab02.ipynb** and submit it to cssubmit (https://secure.csse.uwa.edu.au/run/cssubmit) before the due date and time shown above. You can submit your file multiple times. Only the latest version will be marked.

3 Dataset

We will use the *training.csv* and *testing.csv* data files supplied on LMS for this labsheet. These files were downloaded and slightly modified from the **Forest type mapping dataset** on the UCI Machine Learning website. Please look at the link below:

http://archive.ics.uci.edu/ml/datasets/Forest+type+mapping#

for the description about this dataset. Please do not download the original dataset from the UCI ML website above. You should use the two *csv* files supplied on LMS for this labsheet. You should you save both *csv* files to the same directory with your *lab02.ipynb* file.
The training set (*training.csv*) contains 325 instances instances of multivariate remote sensing data of some forest areas in Japan. There are 4 different forest types labelled in the first column (the column heading is '*class*'), as described in the link above. The test set (*testing.csv*) has the same format as *training.csv* and contains 198 test instances.

4 Tasks

Your tasks for this lab sheet are:

1. Read in the contents of both csv files[1]. Inspect what the columns are by displaying the first few lines of the file. Use appropriate functions to display (visualise) the different features (or attributes / columns). Display some plots for visualising the data. Describe what you see in your markdown cells.
2. To simplify the classification task, write Python code to remove all the columns whose names begin with `pred_minus_obs`. You should have only 9 features (`b1`, `b2`, $\cdots$, `b9`) left for both the training and test sets.
3. Write Python code to count the number of instances for each class label. Do you have an imbalanced training set?
4. Perform an appropriate feature scaling step before doing the classification. You can use `MinMaxScaler`, `StandardardScaler`, or any suitable scaling function in the `sklearn.preprocessing` package. You can also write your own feature scaling code if you prefer. Whichever way, ensure that your feature scaling is applied to both the training data and the test data.

[1] Since both files are in the same directory as your Jupyter Notebook file, you should be able to read each one of them without any path name. For example, `pd.read_csv('training.csv')` should work just fine.

5. Use the *Support Vector Machine Classifier* implemented in the `sklearn.svm.SVC` class to perform multi-class classification using the *one-versus-one* strategy. You should look at the Scikit-learn API for this class and experiment with two hyperparameters. You should use grid search and 3-fold cross validation to find the optimal values for these two hyperparameters that maximise the classification accuracy.

   For other hyperparameters, you can manually set them to some reasonable values. Apart from the Python code, you should explain what you carried out in markdown cells, e.g., which two hyperparameters you have tried? what combination of the hyperparameter values gave the highest classification accuracy?

6. Repeat the above step using the *Stochastic Gradient Descent Classifier* from the `SGDClassifier` class. As the *one-versus-one* computation is not implemented in `SGDClassifier`, you can use the default *one-versus-all* strategy for this classifier.

7. Compare the performances of the two classifiers and give a brief discussion about your experimental results. You should show the confusion matrices and accuracies of the two classifiers for the testing set.

5 Presentation

A few tips on the presentation of your `ipynb` files:

- Present your `ipynb` file as a portfolio, with *Markdown* cells inserted at appropriate places to explain your code. See the following links if you are not familiar with *Markdown*:
  - https://www.markdownguide.org/cheat-sheet/ (basic)
  - https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Typesetting%20Equations.html (more advanced)
- Dividing the portfolio into suitable sections and subsections (with section and subsection numbers and meaningful headings) would make your portfolio easier to follow.
- Avoid having too many small *Markdown* cells that have only one short sentence. In addition to *Markdown* cells, some short comments can be put alongside the Python code.
- Use meaningful variable names.
- When printing out your results, provide some textual description so that the output is meaningful.

6 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