CITS5508 Machine Learning
Semester 1, 2021

Lab Sheet 4

Assessed, worth 10%. Due: 11:59pm, Friday 23rd April 2021

1 Outline

This lab sheet consists of two small projects. In the first project, you should train an AdaBoost Regressor for a small regression problem; in the second project, you should train two Random Forest models using the original data and using the reduced-dimensional data respectively. This lab sheet is a good practical exercise to test your understanding of the techniques covered in Chapters 5–7.

2 Submission

Put your implementation for the two projects below into a single Jupyter Notebook file. You should name your file as **lab04.ipynb** and submit it to LMS before the due date and time shown above. You can submit your file multiple times before the deadline. Only the latest version will be marked.

3 Project 1

The **Abalone** dataset

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

is a small dataset suitable for regressing (predicting) the numbers of rings of new abalone instances. The dataset contains 8 attributes. The ring values are what we want our AdaBoost regressor to predict.

The data and description about the attributes can be downloaded, respectively, from

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

You should download the file *abalone.data* and save it to the same directory with your .ipynb file. Do not rename the file. Although the file name ends with *.data*, you should be to read it the same way as a *.csv* file. **NOTE:** Do **NOT** download the file from your Python code.

This project includes the following tasks:

1. Inspect the data and perform the following data cleaning steps:
   i) from the correlations between pairs of features, determine which two features should be removed for the subsequent training and testing steps. Explain why the two features were chosen.
   ii) for the text column, relabel the `'M'` and `'F'` instances in your Python code as `'A'` (abbrev. for `Adult`). You should then use one-hot encoding[1] to convert this column into numerical columns.
2. Perform a random 85/15 split of the data to form a training set and a test set.

[1]See Page 67 of the textbook by Aurélien Géron, "Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems", O'Reilly, 2019, 2nd edition.

3. For the AdaBoost Regressor, use the Support Vector regressor `SVR` with an RBF kernel as the base estimator. As `SVR` is a reasonably good estimator, we shouldn't need many estimators under the ensemble. You can try 4 or 5 estimators in your code. Also, because of the sequential nature of the AdaBoost algorithm, having too many estimators would require lengthy training and prediction times.

4. Note that as the ring values **must be integers**, you should round and cast the predicted outputs from the ensemble regressor into integers. These integer values should then be considered to be the final prediction results. For instance, if the predicted ring value of an instance is 16.8 then it should be rounded to 17; if the predicted ring value is 16.2 then it should be rounded to 16.

5. Your Python code should report the *mean absolute errors* (MAEs) (both numerically and graphically) of **all the intermediate** and **the final models**. For instance, if there are 5 estimators in your ensemble, then you should have the predictions on the training and test sets for the first 4 intermediate models as well as those for the final (5th) model. To make it easier for comparison, you should put the MAEs for the training and test sets on the same plot.

6. Your Python code should graphically illustrate in two subplots how good the prediction results are on the training and test sets. You should write this as a function with appropriate input arguments so that the function can be reused for project 2. You should put in appropriate comments to describe the arguments to the function[2].

7. In addition, your code should show the distributions (using histogram(s) or bar chart(s)) of the **raw prediction errors** of the final model (no need to do this for the intermediate models) of the AdaBoost regressor for the training and test sets. In this last part of the illustration, you should use the *raw errors* (i.e., these errors can be negative, zero, or positive integers[3]. Again, use subplots for the illustration. As this part of the illustration is also needed for Project 2, you should write a Python function with appropriate arguments so that the function can be reused.

8. Compare the performance of the AdaBoost regressor on the training and test sets and provide a brief summary.

NOTE: You should be able to get an average of 7-10% relative error for the predictions on the test set, i.e., if the ground truth ring value is 20, then your prediction may be off by ± 2 (on average). You can consider fine tuning some hyperparameters for the base estimator, but please **do not include** any grid search code as it would lengthen the marking time of your labsheet.

4 Project 2

In the UCI Machine Learning Repository website above, there is a dataset on **red wine**:

https://archive.ics.uci.edu/ml/datasets/wine+quality

which has 12 attributes and a column that describes the *quality* rating of the wine. For any new test instance, we want our regressor to be able to predict this value. In this project, you should work on the *red wine* data file only. The `winequality-red.csv` can be downloaded directly from the link below:

[2] You should make sure that all your plots have the axes properly labelled; otherwise your plots do not make sense. These axis labels may therefore need to be arguments passed to your function.

[3] Given a test instance, if the error is negative (or positive), it means the regressor has underestimated (or overestimated) the number of rings of the instance. Using the raw errors instead of absolute errors, we can see whether the regressor tends to overestimate or underestimate and by how much. e.g., if the errors are $[+1, -2, 0, -2, -2]$ for a test set of 5 instances, it means the regressor has overestimated the number of rings by 1 for the 0th instance, underestimated the number of rings by 2 for the 1st, 3rd, and 4th instances, and made no mistake for the 2nd instance. For the histogram plot, the error -2 should have frequency equal to 3 (i.e., 3 counts); -1 should have frequency equal to 0; 0 should have frequency equal to 1; and $+1$ should have frequency equal to 1.

https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/

You should save the downloaded file (`winequality-red.csv`) to the same directory with your notebook file. Do not rename or modify the file in any way. As the file uses the semi-colon character (';') to separate the columns, you will need to set the `delimiter` or `sep` argument appropriately to read in the contents.

1. This project includes the tasks below. Read in the contents of the file and perform the usual data inspection and cleaning. Same as Project 1, do an 85/15 random split to form the training and testing sets. Implement a Random Forest regressor to predict the *wine quality* values of instances in the training and test sets. You should also set various hyperparameter values carefully to avoid overfitting the training data. However, this should be reported in your markdown cell(s) or commented-out code cell(s) to speed up the marking process.

   Similar to Project 1, as the wine quality values must be integers, a rounding and type casting operation on the regressor's outputs should be carried out. You should report the MAEs of the predictions on the training and test sets (numerically only, no graphs required). Finally, you should plot the prediction results and use histogram(s) or bar chart(s) to show the distributions of the raw errors of the predictions for both sets.

2. Your next task is to use the *feature importances* obtained from the training process to trim the feature dimension of the data. In your Python code, you should retain only those features whose *importance* values are above 5% (i.e., 0.05). You can either write your own Python code to work out which feature(s) should be removed or use the function `SelectFromModel` from the `sklearn.feature_selection` package[4].

   Report what features were retained and what features were removed in the above process. What is the total feature importance value that is retained after your dimension reduction step?

3. Repeat the training and prediction processes above on the reduced-dimensional data.

4. Finally, compare the performance of the two models of your regressor.

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

[4]Do not use scikit-learn above version 0.23.2, as the newer versions of this library are not yet available on all platforms and your code may not run when we mark it.