CITS5508 Machine Learning
Semester 1, 2024
Lab sheet 6
(Not assessed)

You will develop Python code to perform regression tasks using ensemble models. Certify that the presentation of your Python notebook is good and that you used the Markdown cells well. Make sure you properly format your plots and results. For instance, all your diagrams/plots should have proper axis labels and titles to help the reader understand what you are plotting. Another example is the confusion matrix; not showing the class names makes the confusion matrix completely useless. Use the lab sheets to learn how to improve the presentation of your notebook, as you will need this in the assessments.

Concrete Slump Test

The Concrete Slump Test dataset

https://archive.ics.uci.edu/ml/datasets/Concrete+Slump+Test

is a small dataset suitable for regression. For any new test instance, our aim is to use our trained model to predict its *28-day Compressive Strength* (Mpa) value. The *slump_test.data* dataset can be downloaded directly from the link below:

https://archive.ics.uci.edu/ml/machine-learning-databases/concrete/slump/

Although the name of the file ends with *.data*, it is actually a *csv* file.

NOTE: Save the downloaded file (*slump_test.data*) to the same directory with your Jupyter Notebook file.

Tasks

This part of the lab sheet includes the following tasks:

1. Firstly, you should inspect the data and clean it if needed. As described on the web page above, there are seven input variables (i.e., feature columns) and three output variables. Our interest is the *28-day Compressive Strength* output, so the other two output variables should be dropped. Perform some basic visualisation and determine whether any additional features should be removed. You should write a Python function for this data-cleaning step.
2. Perform an 80/20 random split on the dataset to form a training set and a test set. Implement a Voting regressor, using the following as the three base estimators with default hyperparameters: (i) a linear SVM regressor, (ii) a linear regressor (using the `LinearRegression` class), and (iii) a Stochastic Gradient Descent (SGD) Regressor with Ridge regularisation.
3. Train the base estimators and the Voting regressor on the training set, compare their performance on the training and test set.
4. Illustrate the predicted values versus the ground truth values of all the test instances for all models. Is there any consistency in the models' mistakes?
5. Individually tune a few hyperparameters for the SVM and SGD base estimators. You can use 3-fold cross-validation with Grid-Search to select the best hyperparameters. Then, repeat the previous task with the selected set of hyperparameters for each base estimator. Compare and comment on the results.

Abalone dataset

Considering the Abalone dataset you used in the previous lab. Your task now is to inspect the performance of Random Forest and Bagging models.

Tasks

1. Implement a Random Forest regressor with 100 estimators. Train your Random Forest regressor on the training set and test it on the test set. Report the performance on both set. Note: as the ring values must be integers, your Random Forest regressor's predicted results must be rounded to the nearest integer first.

2. Repeat the previous task but use the Decision Trees hyperparameter values selected in the previous lab sheet tasks. What was the impact (in any) of using the fine-tuned hyperparameters for the Random Forest model?

3. Do you think increasing the number of estimators could improve the model's performance?

4. 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)[1].

5. 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?

6. Repeat the training and prediction processes above on the reduced-dimensional data. We expect to see a slight change in the performance for the reduced-dimensional data. In many real applications, the feature dimension of the data may be reduced drastically with only a slight increase in the prediction error. We won't see this in this small dataset as the feature dimension is already quite small.

7. Finally, compare the performance on the training and test sets of the Random Forest regressors.

8. Illustrate in a diagram or two the prediction errors of all the test instances from the best of the above models, e.g., you can try computing the average prediction error for each ring value.

   Do large or small (or both) ring values tend to have large average errors? Is a large prediction error for a ring value related to insufficient training instances for that ring value?

9. Implement a Bagging regressor with 100 Decision Tree regressors as the base estimators. Experiment with and without using fine-tuned hyperparameters. For the Bagging regressor, be careful with the values you choose for *max_features*, *max_samples*, and *bootstrap* and how they compare with the Random Forest regressor.

10. Train your Bagging regressors using the full-dimensional training set and test it using the full-dimensional test set. Report the performance of the predictions for the test set. Illustrate in a diagram the predicted ring values versus the ground truth ring values of all the test instances.

11. Compare the performance of your Random Forest regressors with the Bagging regressors.

[1] If no features are removed with the 0.05 threshold value, then increase the threshold slightly. For the exercise in this lab sheet, we want to experiment with the effect of reducing feature dimension.