$$\begin{tikzcd}
 & & & d \arrow[dlll] \arrow[dl] \arrow[dr] \arrow[drrr] & & & \\
t_1 \arrow[rr] \arrow[rrrr, bend right] \arrow[rrrrrr, bend right] & & t_2 \arrow[rr] \arrow[rrrr, bend right] & & t_3 \arrow[rr] & & t_4
\end{tikzcd}$$

COMP47590
Advanced Machine Learning
Assignment 1: Multi-Label Classifiers

Introduction

In contrast to **multi-class classification** in which instances can only belong to a single class, in **multi-label classification**[1] problems instances can belong to more than one class at a time. For example, an image might be classified as containing all of *mountains*, *sky*, and a *house* or a piece of music might be classified as belonging to both the *rock* and *jazz* genres. A selection of multi-label classification approaches based on ensemble methods exist in the literature.

The task in this assignment is to implement a selection of multi-label classification approaches yourself. Although you can use scikit-learn base estimator implementations (e.g. decision trees, logistic regression, or support vector machine models) you must implement the ensemble methods yourself.

Tasks

Perform the following tasks:

0. For all of the tasks in this assignment use the **Yeast dataset**[2]. The Yeast dataset is formed by micro-array expression data and phylogenetic pro-files with 2,417 included. There are 103 descriptive features per gene. Each gene is associated with a set of functional classes. In this version of the dataset there are 14 functional classes and a gene can be associated with any number of these.
1. The simplest approach to multi-label classification is the **binary relevance**[3] algorithm in which individual binary base classifiers are implemented for each label. This uses a one-vs-all approach to generate the training sets for each base classifier. Implement the binary relevance algorithm. Allow the base classifier type to be set as a parameter to this algorithm[4].
2. One of the issues with the one-vs-all approach to generating training datasets used in the binary relevance algorithm is that the training datasets for base classifiers can be very imbalanced. Add an option to the binary relevance training implementation that uses **under-sampling** to create a balanced training dataset for each base classifier. You must implement the under-sampling approach yourself.

[1] For a good overview of multi-label classification see Gibaja, E. and Ventura, S. (2014), Multi-label learning: a review of the state of the art and ongoing research. WIREs Data Mining Knowl Discov, 4: 411-444. https://onlinelibrary.wiley.com/doi/full/10.1002/widm.1139?casa_token=2tuYiUJKCdcAAAAA%3AyOdbOLg13y2cH-YCuJO2GoCoR2z2Rhv2CCJNLwT2x7B2BJqLAGFu7zq0NFBKThTy2EFL4oQhqyczWi-w

[2] Elisseeff, A., & Weston, J. (2002). A kernel method for multi-labelled classification. In *Advances in neural information processing systems* (pp. 681-687). http://papers.nips.cc/paper/1964-a-kernel-method-for-multi-labelled-classification.pdf

[3] Tsoumakas G, Katakis I, Vlahavas I. Mining multi-label data. In: Data Mining and Knowledge Discovery Handbook, Part 6. Springer; 2010, 667–685. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.324.2716&rep=rep1&type=pdf

[4] For an example of how to pass base classifier type as a parameter see the scikit-learn BaggingClassifier implementation: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

3. Compare the performance of the two different versions of the binary relevance classifier (with or without under-sampling).
    - Design and use an appropriate evaluation strategy. Think about the most evaluation experiment and the most appropriate performance measure for the evaluation[5].
    - For all models use homogenous ensembles.
    - Try out a couple of different model types at the base layer (e.g. logistic regression models or decision trees).
4. One of the criticisms of the simple binary relevance classifier approach is that it does not take advantage of associations between labels in a multi-label classification scenario. For example, the presence of sea in an image increases the likelihood of a boat also being present, but decreases the likelihood of a giraffe being present. The **classifier chains**[6] algorithm is an effective multi-label classification algorithm that takes advantage of label associations. A classifier chain model generates a chain of binary classifiers each of predicts the presence or absence of a specific label. The in input to each classifier in the chain, however, includes the original descriptive features plus the outputs of the classifiers so far in the chain. This allows label associations to be taken into account. Implement the classifier chains algorithm.
5. Using the same experimental approach designed previously compare the performance of the classifier chains algorithm to the versions of the binary relevance algorithm.
6. Reflect on the performance of the different models evaluated in this assignment. Consider model performance as well as computational requirements, and model complexity. For this, write a commentary of no more than 300 words.

[5] **Hamming loss** or **macro-averaged f-score** are useful performance measures for multi-label classification problems, and both are implemented in scikit-learrn.
https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

[6] Read, Jesse, et al. "Classifier chains for multi-label classification." Machine learning 85.3 (2011): 333.
https://link.springer.com/content/pdf/10.1007/s10994-011-5256-5.pdf

Notes

The following notes may be useful:

- **Can I Use Scikit-Learn and Other Python Packages?** One of the goals of this assignment is to gain experience writing original machine learning algorithms, rather than simply using existing packages. We recognise, however, that very good implementations of machine learning algorithms exist in packages like **scikit-learn** and re-implementing things needlessly is often of limited value. So, for this assignment we seek to strike a balance. You must write your own implementation the binary relevance and classifier chain algorithms, but can use scikit-learn implementations for the base estimators. You should also use scikit-learn functions for performing tasks like cross validation, grid searches, and measuring performance. Similarly, all scikit-learn models are built on top of **numpy** and it provides many useful utility functions. If in doubt about the use of a package please just ask.
- **What Hardware Should I Use?** None of the datasets or models to be used in this assignment are so big that they shouldn't work fine on your laptop. If you are having problems, however, you should consider using Google Colab[7], a great online platform for machine learning.
- **It's Still Taking Forever!** While the dataset used in this assignment is not enormous, things like cross validations and grid searches can be slow. If you find things are taking too long feel free to down-sample the dataset. Submissions will not be penalised for using down-sampled datasets. Ensure, however, that the target feature levels remain stratified.
- **Teams.** You can complete this assignment as an individual or as a team of two. Submissions will be graded in the same way whether submitted by an individual or by a group. All members of a team will receive the same grade.

[7] Google colab https://colab.research.google.com

Submission

The key submission details for the assignment are as follows:

- **Submission date:** Sunday 8th March 2020 before 23:59.
- **Submission method:** Submissions should be made through the module Moodle site.
- **Submission format:** Submissions should compose a zip file containing the following:
    - a completed version of the template Jupyter notebook provided in .ipynb format (the Jupyter notebook and code should include adequate comments to understand your implementation);
    - a html export of your Jupyter notebook after execution that contains all output;
    - any other files required to execute your code.
- **Group submission:** For group submissions only one member of the group should submit.
- **Late submissions:** Late submissions will be penalised at 5% penalty per day.

Marking

Marking of tasks will be based on the following weighting.

Task 1	25%	Implement the **Binary Relevance** classifier.
Task 2	15%	Implement the **Binary Relevance with Under-Sampling** classifier.
Task 3	10%	Compare the performance of the Binary Relevance implementations.
Task 4	35%	Implement the **ClassifierChains** classifier.
Task 5	10%	Compare the Classifier Chains and Binary Relevance models.
Task 6	5%	Reflect on the experiment.