Description of the exercise

Study for the classification of 2 datasets from the UCI MachineLearning repository. One dataset is small (Small) with less than 1000 samples and the other large (Big) with more than 1000 samples. There are 12 S and 12 B datasets to study and none of the 60+ groups in the lab has the same combination of (S, B) datasets with another.

You can find the (S, B) assigned to your team in the UCI Datasets panel (shifts are on more sheets). To see which UCI datasets match your passwords and which data files to work with, consult the UCI classification datasets.

Objective: With the exception of dummy classifiers, your objective is for the other classifiers to find for each one separately in each dataset a) the optimal transformer architecture (pre-processing stages) and b) the optimal hyper-parameters (of both transformers and of the classifier) via grid search and crossvalidation.

Small dataset (S)

Basic information

1. Brief presentation of the dataset (what it describes).
2. Number of samples and characteristics, type of characteristics. Are there any unordered features and what are they?
3. Are there headings? Line numbering?
4. What are the labels of the classes and in which column are they located?
5. Did you need to make text file conversions and which ones?
6. Are there any missing values? What are the samples with absent values and what is their percentage of the total?
7. What is the number of classes and their sample percentages in total? If we consider that a dataset is unbalanced if any one class is 1.5 times more common than another (60% -40% in binary datasets) estimate the balance of the dataset.
8. Separate into train and test set. If there are missing values and unordered features, manage them and justify your choices.

Classification

- The classifiers you will look at in the small dataset are: dummy, Gaussian Naive Bayes, kNN.
- You will use 20% for test set and 10-fold format for cross-validation.
- You will use 2 different performance metrics in cross-validation to select the model: a) f1_micro and b) f1_macro.

Baseline classification

1. Manage any missing values. Train the classifiers in the train with default values (simple initialization). Evaluate the test set (including dummy) and print for each estimator: confusion matrix, f1-micro average and f1-macro average.

2. For each averaged metric, print a bar plot comparison with the values of this f1 for all classifiers.
3. Comment on the results of the plots and the precision, recall, f1 values of the confusion tables.

Optimizer of classifiers

1. For each classifier, optimize its performance in the training set through the pre-processing process and finding optimal hyperparameters (not all classifiers have hyperparameters). Evaluate the test set (including dummy) and print for each estimator: confusion matrix, f1-micro average and f1-macro average.
2. For the final fit of each classifier in the whole training set and for the predict in the test set print tables with the execution times.
3. For each averaged metric, print a comparison bar plot with the values of this f1 for all classifiers.
4. Print a table by changing the classifier performance before and after optimizing it.
5. Comment on the results of plots and the values of precision, recall, f1 of the confusion tables, the change in performance and the execution times.

Hyperparameters to optimize

kNN: n_neighbors.

Large dataset (B)

Basic information

1. Brief presentation of the dataset (what it describes).
2. Number of samples and characteristics, type of characteristics. Are there any unordered features and what are they?
3. Are there headings? Line numbering?
4. What are the labels of the classes and in which column are they located?
5. Did you need to make text file conversions and which ones?
6. Are there any missing values? What are the samples with absent values and what is their percentage of the total?
7. What is the number of classes and their sample percentages in total? If we consider that a dataset is unbalanced if any one class is 1.5 times more common than another (60% -40% in binary datasets) estimate the balance of the dataset.
8. Separate into train and test set. If there are missing values and unordered features, manage them and justify your choices.

Classification

- The classifiers you will look at in the large dataset are: dummy, Gaussian Naive Bayes, kNN, Multi-Layer Perceptron (MLP), Support Vector Machines (SVM)
- You will use 30% for test set and 5-fold format for cross-validation.
- You will use 2 different performance metrics in cross-validation to select the model: a) f1_micro and b) f1_macro.

Baseline classification

1. Manage any missing values. Train the classifiers in the train with default values (simple initialization). Evaluate the test set (including dummy) and print for each estimator: confusion matrix, f1-micro average and f1-macro average.
2. For each averaged metric, print a bar plot comparison with the values of this f1 for all classifiers.
3. Comment on the results of the plots and the precision, recall, f1 values of the confusion tables.

Optimizer of classifiers

1. For each classifier, optimize its performance in the training set through the pre-processing process and finding optimal hyperparameters (not all classifiers have hyperparameters). Evaluate the test set (including dummy) and print for each estimator: confusion matrix, f1-micro average and f1-macroaverage.
2. For the final fit of each classifier in the whole training set and for the predict in the test set print tables with the execution times.
3. For each averaged metric, print a comparison bar plot with the values of this f1 for all classifiers.
4. Print a table by changing the performance of the classifiers before and after optimizing them.
5. Comment on the results of plots and the values of precision, recall, f1 of the confusion tables, the change in performance and the execution times.

Hyperparameters to optimize

kNN: n_neighbors, metric, weights.
MLP: hidden_layer_sizes (use only one level of hidden neurons), activation, solver, max_iter, learning_rate, alpha.
SVM: For linear kernel you will use LinearSVC with basic hyperparameters for optimization: loss, tol, C. For poly and rbf cores you will use SVC with basic hyperparameters for optimization C, degree (for poly core), gamma, tol.

You can also experiment with the other classifier hyperparameters.

Multiclass datasets: kNN and MLP are inherently multiclass classifiers, while SVMs are not. However, SVC implements a one-vs-one multiclass strategy while LinearSVC also has a multiclass SVM strategy.

Unbalanced datasets: note that in SVM instead of facing the problem in the preprocessing stage you can use the class_weight parameter in training (without knowing from the beginning which of the two approaches gives the best results.