CSE176 Introduction to Machine Learning — Project
Fall semester 2025 — Miguel Á. Carreira-Perpiñán

This project is about training and exploring algorithms for learning ensembles of decision trees (**decision forests**) in several real-world classification and regression datasets, and evaluating their performance as a written report and an oral presentation at the end of the course. You'll have to download code available online implementing some specific algorithm and apply it to some datasets. Typically, the code will be in Python, although it can be in other languages (R, Matlab, C/C++). You'll have to use it no matter what language it is in.

The project will help you become familiar with a specific but widely applied and very successful machine learning model (tree ensembles), and to face the many decisions and issues that arise when working with real data.

The project has 3 parts. Part 1 is a simple, small binary classification problem using logistic regression and random forests. It is intended to make sure you get familiar with machine learning problems and, after optionally receiving our comments, that you correct any problems so you do parts 2 and 3 better. Part 2 is MNIST classification with XGBoost. Part 3 is group-specific in terms of the dataset and algorithm, and we expect a more extensive study. Having a common dataset and algorithm (MNIST, XGBoost) makes it possible to compare the performance (accuracy, number of parameters, inference time, training time, etc.) of each group's work on the same basis. But then, in order to have variability in the project presentations, the other forest algorithm and datasets will be unique to each group of students. Part 4 is optional.

IMPORTANT: see the course web page for deadlines, group size, etc.

I Project parts

Part 1: binary classification using logistic regression and random forests

Dataset: MNISTmini digits.
These are grayscale images of 10×10 pixels of handwritten digits in 10 classes (digit-0 to digit-9). Note: use as training, validation and test sets images 1–1000, 1001–2000 and 2001–3000, respectively. The reason to use MNISTmini (and only a subset of images from it) is just so that your experiments are faster.

Task: binary classification, e.g. digit-3 vs digit-7. We will assign each group a different pair of digits.

Algorithms: logistic regression and random forests, both in the scikit-learn implementation (Python), described in the links below. Each has various parameters or formulations, some of which were explained in class. Feel free to explore things, but we suggest:

- Logistic regression: use ℓ_2 regularization, with the hyperparameter determined via cross-validation. Any training algorithm (solver) should work but we sugest `"liblinear"`.
- Random forests: determine the number of trees by cross-validation. The parameters that control how each tree is learned (split criterion, `max_depth`, `max_features`, etc. can be set to the defaults).

Scikit-learn algorithms:

- Logistic regression:
  https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
  https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
- Random forests:
  https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
  https://scikit-learn.org/stable/modules/ensemble.html#random-forests

Do your best to understand the scikit-learn descriptions and experiment with the options available. If you get stuck, ask the TA in the office hours.

Part 2: MNIST multiclass classification with XGBoost

Dataset: the MNIST dataset of handwritten digits (whole training set, all 10 classes), using two types of features:

- Pixel features: use our Matlab file `MNIST.mat`, which uses as features the $28 \times 28 = 784$ grayscale pixel values in $[0, 255]$. It is partitioned into training (60k images) and test (10k images). Use the test set only to report test errors. Use the training set for training and (if needed) validation, in whichever way you want (e.g. a random split of 55k for training and 5k for validation).
- LeNet5 features: each image is represented as an 800-dimensional feature vector, obtained from a pretrained LeNet5 convolutional neural net (the output of the neurons at layer conv2). Use our Matlab file `MNIST-LeNet5.mat`. The images are the same as for the pixel-based images and appear in the same order.

Task: multiclass classification.

Algorithm: XGBoost.

Proceed similarly to part 1.

Part 3: multiclass or binary imbalanced classification, and regression

This is similar to parts 1 and 2 (training and evaluating models) but more extensive in its evaluation, and using group-specific datasets and algorithms. We give you considerable freedom on how to analyze the data, (possibly) transform it, explore hyperparameters, compare with additional baselines (e.g. logistic regression), etc. The better your exploration, the higher the grade.

Datasets and tasks:

1. One classification dataset, either multiclass classification or binary imbalanced classification.
2. One regression dataset.

Algorithm: a forest-based algorithm.

As for the specific dataset and algorithm, we can either assign them to you, or you can propose us a choice and we will consider it. The dataset should be neither too simple nor too difficult (in terms of the number of instances, features and classes), and also somewhat diverse in terms of the subject matter (for example, don't pick any other handwritten digit dataset). The files `datasets.txt` and `forest-packages.txt` (in CatCourses) give suggestions, but you are not limited to that. Contact the TA asap to determine your assignment.

For part 3, the minimum required is to do a similar study as in parts 1 and 2 for the datasets/algorithm you are assigned, but going beyond this will increase significantly your grade. There are many possibilities, these are just some suggestions:

- Evaluate the effect of the model's hyperparameters (beyond basic ones such as the number of trees).
- Apply some transformation to the data (e.g. making them zero-mean unit-variance, Gaussianizing them, using one-hot or label encoding for categorical features, etc.).
- Reduce dimensionality in advance (e.g. with PCA or with some form of feature selection).
- Manipulate the datasets more generally (e.g. via image deskewing, data augmentation…).
- Report things beyond the test accuracy, e.g. training time, ROC and AUC (for binary classification), number of parameters, etc.
- Investigate the model beyond its raw accuracy performance, e.g. try to understand which features are relevant.
- Use K-fold cross-validation.
- Try more algorithms, forest-based or otherwise (linear models, kernel SVMs, etc.).
- Try more datasets.
- Include interesting findings, observations and irregularities you find. Make sure plots are properly labeled and legible.
- Etc.

Many algorithms to help you explore the possibilities above are available in scikit-learn, R, etc. and you are free to use them if you want. Remember that you need to split your available data into training, validation and test, and that the primary goal is to achieve a forest with the highest accuracy in the test set (but don't use the test set other than to report the accuracy of the final model).

We will value your effort, creativity and insights achieved in the project grade. The more extensive and insighful your experimental exploration, the more you will learn about the algorithm and the higher the grade in the project.

Part 4: comparing the decision forest with a foundation model for tabular data

This part is entirely optional, but we will value it accordingly as an extra grade. A foundation model is based on neural nets, not decision trees. It is an extremely large neural net that has been pretrained on an enormous number of (in this case) tabular datasets, real and synthetic. You don't need to train anything, instead you feed both the training set and the test instance to the neural net and it produces a prediction directly. This type of models are highly experimental and are limited to small datasets (so you may have to apply to a subset of your dataset). You'll have to investigate the details on your own and you may need to run the foundation model in a GPU; see file `foundation-models.txt` in CatCourses. If you do try this, describe it in to your report (as described above).

II What you have to do

II.1 Project presentations

Logistics:

- Each group has 10 minutes to present followed by 5 minutes of questions.
- The groups will present in alphabetical order of UCM id (of the group member with first UCM id).
- Each group presentation should be done by having each member present some part of the work (rather than having a single member present everything).
- We'll use a single laptop for all presentations (to minimize time wasted changing laptops, projector issues, etc.). Send your slides (PDF file) in advance to the TA.

Each group will present a subset of their work, as follows:

Part 1 MNISTmini results: you don't have to present them.

Part 2 MNIST results with XGBoost: you don't have to present them. Instead, send the TA your results, so we can put the results from all groups together in a single PDF file and look at them during the presentation day. Specifically, for each dataset (MNIST pixel features and MNIST LeNet5 features):

- A plot (EPS or PDF file) of the test error as a function of the number of trees (for whatever hyperparameter values you settled on).
- A text file containing the best test error (found by your best cross-validated forest), the number of trees it used, and any other important hyperparameter values. For example:
  `test error 1.57%, 1500 trees, maximum depth 6`
  The TA will combine the files from all the groups into a "leaderboard".

Part 3 Group-specific datasets/algorithms:

- Briefly describe the dataset, noting its sample size, dimensionality and (for classification) number of classes.
- Briefly explain the algorithm you used.
- Explain what you did, for example how you handled the features (any preprocessing? outlier removal? missing values? etc.), how you did the cross-validation, what hyperparameters you tuned, etc.
- Show your results in whatever form you prefer. An important plot for forests is the test error as a function of the number of trees, but there are other plots you can generate.
- Explain any insights you obtained.

But, keep it to 10 minutes total. Presenting your work well is part of the evaluation. We will also take into account how well designed your presentation is; aim for clarity and simplicity.

Part 4 Foundation model: you don't have to present this.

II.2 Report (and any code you need to write)

Follow these instructions strictly.

Part 1 Upload a single file `part1.tar.gz` to CatCourses containing:

- A report (`report1.pdf`, max. 2 pages) describing what you did precisely but concisely. Don't include any code in the report (or, at most, only include short code snippets).
  For each algorithm separately, plot the training error and the validation error for a range of hyperparameter values and select as final model the one with lowest validation error. Then, give the test error for this model. How does it compare to the training/validation error?
- Any (Python) scripts your wrote (to train the forest, plot things, etc.) with brief instructions of how to run it (don't include any external code needed but do tell us what you used). Don't include any data files but give a link to them in the report.

Optional but strongly suggested: finish part 1 and upload it directly to CatCourses as soon as you can. The TA will give you comments, which will be helpful for parts 2 and 3.

Part 2 Upload a single file `part2.tar.gz` to CatCourses containing, as in part 1:

- A report (`report2.pdf`, max. 2 pages). Also include a confusion matrix for the final model on the test set.
- Code.

Part 3 Upload a single file `part3.tar.gz` to CatCourses containing:

- A report (`report3.pdf`, max. 10 pages) containing the following sections, in this order:
  1. A concise description of the algorithms you used.
  2. One section per dataset describing the dataset and your experimental results, carefully describing how you did things: selection of training, validation and test sets; choice of hyperparameters; etc. Be concise; no need to include every plot you did.
  3. A conclusion section where you comment on the lessons learned about the algorithm based on your results.
- Code.

Part 4 Don't include any code or model, just describe your results and upload a single file `report4.pdf` to CatCourses.

Apart from the technical quality of your work, in evaluating it we'll also consider the quality of the organization and writing of the report and code. Again, aim for clarity and simplicity.

IMPORTANT: also upload a 1-page file `contributors.pdf` with a brief description of what each group member contributed to the project and of the sources you consulted (books, papers, web pages, code, generative AI, etc.). See examples of how to do this in the course web page: https://faculty.ucmerced.edu/mcarreira-perpinan/teaching/CSE176 → Course grading.