Assignment
FEM31002 & FEM21045

September 2022

Welcome to the asisgnment of the machine learning course! This assignment should be made in groups of four students, so please team up with your fellow students. The setup of this assignment is quite non-standard, so please read this document carefully.

In this assignment, you are asked to apply the Machine Learning algorithms discussed in this course to predict the poverty risk of households in Costa Rica. This task was designed by the Inter-American Development Bank as a Kaggle challenge. In the following, we provide some context on this challenge: In order to qualify for social welfare programs in Costa Rica, households need to provide certain documents on their income and expense records. However, the poorest households often cannot provide these documents, so a different method is required to assess eligibility for social welfare programs. In this challenge, you will use characteristics of households and individuals to build a binary classifier that decides whether or not Costa Rican individuals are poor. With this information, Costa Rican government agencies can identify the individuals with the greatest need for social welfare, which will allow them to allocate social welfare payments more efficiently.

1 Overarching Task

Your overarching task is to train a machine learning algorithm that predicts whether or not an individual is poor. You will train the algorithm(s) for a preprocessed version of the *Costa Rican Household Poverty Level Prediction* dataset, made public on Kaggle.[1] Each instance of this dataset consists of one binary target and a set of nondescript predictor variables. The binary target takes the value 1 if a given individual is poor, and takes the value 0 if they are not poor.[2] The predictive values are a mix of information on the household and individual level that may be exhibit explanatory power for the target variable. A detailed description of each variable can be found in the enclosed file `codebook.csv`.

[1] `https://www.kaggle.com/competitions/costa-rican-household-poverty-prediction/overview`.

[2] Specifically, an individual who is classified as poor according to this binary target lives in either extreme or moderate poverty or is vulnerable to such poverty levels. Individuals who are not classified as poor according to the binary target do not live in poverty and are also not vulnerable to any level of poverty.

The preprocessing was performed in a way such that you can immediately apply your machine learning algorithm of choice without (much) further data manipulation. A description of the preprocessing that was applied can be found in Appendix A.

In designing your predictive algorithm, please adhere to the following requirements:

1. You are allowed to choose among the following algorithms: (logistic) linear regression with regularized training (ridge, lasso, elastic net), a (classification) tree, a random forest, gradient boosting, or a neural network (support vector machines are excluded from the assignment since their introduction in the course is too close to the deadline).[3]

2. You are **not** allowed to combine the algorithms listed in the above point into an ensemble for your final predictive algorithm. Although doing so is popular in practice, such ensembles are beyond the scope of this course and therefore excluded from the assignment.

3. The algorithm you train should maximize the accuracy measure: the number of correctly labeled instances, both 0 and 1, over the total number of observations.

4. When you are estimating and tuning your algorithms, we ask you to optimize hyperparameters and evaluate the accuracy of the final prediction algorithm. This requires you to come up with a training, validation, and test strategy. It is up to you to decide what strategy you choose, and as you have learned in the first week, there are many good choices.

5. Which or even how many hyperparameters to optimize is also up to you.

6. You may assume that accurately predicting individuals that are poor is equally as important as accurately predicting individuals that are not poor.

Hint: since this assignment is based on a data mining competition on Kaggle, you don't have to completely reinvent the wheel. We highly recommend you to check out the approaches taken by the winners of the Kaggle competition to see if there are methods you can adopt to improve your accuracy. Do keep in mind several differences between the competition and the prediction task given here:

1. We use an adapted and preprocessed dataset for your convenience. For this reason, several methods used to address problems in the Kaggle competition are irrelevant in the context of this assignment. Most notably, the Kaggle competition has the goal of multilabel classification (multiple levels of poverty and non-poverty), while we focus on binary classification (poverty vs. non-poverty).

2. There is a difference in performance metric, as in the Kaggle competition the macro F_1 score is used instead of the accuracy measure.

[3]Of course, you are very welcome to try a variety of different algorithms, including perhaps algorithms that have not been introduced in this course to see how they compare. But for your final report and predictive performance, please use one of the algorithms listed above.

2 Graded Subtasks (30 pt)

While your task is fundamentally singular, you are graded on the basis of three subtasks: the performance of your algorithm (coined *prediction challenge*), your design choices, and your ability to report on these choices concisely, and your conceptual understanding of the problem. These subtasks are elaborated on below. A more detailed grading rubric can be found in Appendix B.

2.1 Predictive Performance

2.1.1 Setup

The utility of your predictive algorithm is for a large part determined by the accuracy you achieve. We therefore grade your predictive performance based on an affine combination of two benchmarks, capped from below by 0 and from above by 5.

Lower Benchmark 1 pt if your algorithm predicts as well as a random number generator, generating numbers 0 and 1 with equal probability.

Upper Benchmark: 5 pt if your algorithm beats the best predictions by Max Welz' algorithm.

In other words: you are asked to beat the teaching staff! While this may seem like a daunting task, in fact, this should be very doable as we won't spend nearly the same amount of time optimizing our algorithms. We will only reveal the teacher's accuracy after the final deadline, so it is up to you to decide when your score is good enough. You may want to consider comparing your score with other teams to see how you are doing!

You are provided with two datasets: a training set and a test set, called `train.csv` and `test.csv`, respectively. The training set is a set of instances with binary targets (variable `target`) and features which can be used to train and test your machine learning algorithm yourself. The test set only includes features but no targets. You can find these two datasets in the Assignment page on Canvas and in the GitHub Classroom repository you will be using in this assignment; details on this follow in Section 3.

2.1.2 Prediction Challenge

Since we will grade your algorithm on its predictive accuracy against predictions made by us, we call this first part of the assignment a *prediction challenge*. In this prediction challenge, you are asked to predict the binary targets for the instances in the test data `test.csv` and provide them to us in a file `predictions.txt` so we can compute your predictive accuracy and grade its performance based on the two benchmarks above. The prediction file `predictions.txt` should contain all predicted target values in the same ordering as the instances in `test.csv` in a single bitstring. For example: if you predict for the first few instances of the test dataset the targets 0 1 1 0 0 0 1 1 0 1 1 1 1 1 0 1 0 1 0 1 1, respectively, then your submission should look as follows:

```
011000110111110101011
```

Please put this string in the plain text file `predictions.txt`. Your predictions file will be graded automatically in CodeGrade, so it is important that you strictly adhere to the above instructions. For example, do not change the file name `predictions.txt` in any form or put any other information into the text file.

Some warnings: please triple check that your prediction file is error-free, applied to the right dataset, and in the right order! The score we compute is final, so it is your responsibility to make sure your file is created correctly.

Please note that you are **not** allowed to use the test data in `test.csv` in any way or form during the training of your algorithm; any attempt of doing so will be considered fraud! We therefore recommend that you *yourselves* partition the data in `train.csv` into an estimation set (which you use for training) and a validation set (which you use for assessing out-of-sample accuracy). The only time you use `test.csv` should be when you make the predictions you intend to submit for the prediction challenge; these predictions are supposed to be the output of a machine learning technique that has been trained on `train.csv`.

Moreover, trying to retrieve targets from Kaggle and using these values in some way for your estimation will be also be treated as fraud. The same holds for directly copy-pasting code from the discussion forums or from elsewhere. Using packages is, of course, allowed.

You will be asked submit the complete code of your algorithm. We will not grade your code in any form. It merely serves as a check for us that you have not engaged in plagiarism or the forms of fraud described above. After the deadline, we may ask you to replicate your submitted predictions in our presence by using your submitted code.

2.2 Report (15 pt)

Your second subtask is reporting on your design choices/methodology and your main findings. Please refer to the overarching task description and the rubric to determine what information you should include. In general, your report has three purposes:

First, motivate your choice of algorithm and clearly describe how you have trained it. Your description of the training process should be concise enough such that it is fully replicable[4] for peers following this course. For example: "we used a grid-search" is not replicable because you didn't share which candidate values of the hyperparameters you used exactly. Please note that you are not asked to describe how the algorithm itself works, so there is no need to describe how, say, random forests work.

[4]Up to stochastics and programming details, meaning that a stochastic algorithm will not give the same result twice. In the report you don't have to share in detail how you programmed your code.

Second, you should show a clear motivation for the choices you made. This can either be based on a good rationale or cited relevant scientific sources.

Third, you should arrive to clear conclusions. For example, "our predictive algorithm achieves an out-of-sample accuracy of 50%." We again emphasize that you are not allowed to use the data in `test.csv` in your training of the algorithm.

Write your report in the provided LaTeX template, called `report.tex`. You are free to add packages of your choosing to create your report, but do not change the font, margins, or any other aspect of the template that determines how much text can be in your document. Your report—returned as the file `report.pdf` by the template—may not exceed one page. A list of references and an appendix may exceed the one page limit. In the appendix, you may **only** list the optimized hyperparameters and indicate the evaluated grid values. Moreover, please enter your names and group numbers at the top of the template page `report.tex`. You may ctrl-f or cmd-f "#HERE" in the template to find where you should enter these details.

2.3 Questions (10 pt)

For the final part of the assignment, we ask you to reflect on your results from the previous parts of the assignment by answering five questions, which are listed below. Please **do not** refer to tables or other results from the main task part of your hand-in when answering them, as the report and the questions will be graded by two separate TAs. Just like the report, you are supposed to write the answers to the questions in a provided LaTeX template, named `questions.tex`. Its output, `questions.pdf`, may not exceed one page, except for a list of references (appendices are not allowed). Again, you are not allowed to change the settings of this template. Again again, please enter your names and group numbers at the top of the template page `questions.tex`. You may ctrl-f or cmd-f "#HERE" in the template to find where you should enter these details.

Here are the five questions:

Q1: The estimation procedure requires you to maximize accuracy, which is a popular measure in binary classification. Describe a dataset on which an algorithm that maximizes the accuracy measure will not yield a useful (non-trivial) binary classification rule.

Note: This question does *not* ask to discuss alternative evaluation metrics to the accuracy measure or discussing a specific machine learning algorithm; it asks for describing characteristics of a dataset (with binary targets) on which maximizing the accuracy measure will lead to a useless decision rule for binary classification.

Q2: Overfitting is a constant threat for any machine learner. It is therefore advised to take preventive measures against overfitting, for instance regularization. Describe a procedure that can be used to check if a machine learning algorithm has overfitted on the training data.

Apply this procedure on your final machine learning algorithm (that has been trained on the training data `train.csv`) and conclude whether or not you find evidence for overfitting.

Q3: Many software packages for regularized linear regression silently standardize the features to have mean zero and variance one before fitting the model. (1) Explain why feature standardization is done when using regularized linear regression. (2) Should feature standardization also be done when using regression random forests? Explain why or why not.

For our next question, we consider the *universal approximation theorem*, which reads as follows (adapted from Theorem 3.2 in Kidger and Lyons, 2019).

Theorem 1 *Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^d$. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuously differentiable activation function. Denote by $\mathcal{N}^{\sigma}_{d,D,d+D+2}$ the space of feedforward neural networks with d input neurons, D output neurons, and an arbitrary number of hidden layers each with $d+D+2$ neurons, such that every neuron in the hidden layers has activation function σ and every neuron in the output layer has the identity activation function. Then, given any $\varepsilon > 0$ and any continuous function $f : \mathcal{X} \to \mathbb{R}^D$, there exists a neural network $\widehat{f} \in \mathcal{N}^{\sigma}_{d,D,d+D+2}$ such that*

$$\sup_{x\in\mathcal{X}} \left\| \widehat{f}(x) - f(x) \right\| < \varepsilon.$$

Heuristically, the universal approximation theorem states that a feedforward neural network of sufficient flexibility (that is, sufficiently many hidden layers and neurons) is able to approximate any continuous function.

Q4: Does the universal approximation theorem imply that learning a given target function by using a sufficiently rich training dataset and a sufficiently flexible feedforward neural network that has appropriately chosen activation functions will always succeed in regression problems? Discuss.

Q5 Suppose that we have a dataset that consists of continuous targets and a mix of continuous and binary features. It is a well-known phenomenon that when fitting regression trees, the fitting algorithm favors splits over continuous features to splits over binary features, even when both features have similar explanatory power for the target. Explain why this phenomenon occurs.

3 Workflow and Group Registration

3.1 Workflow

GitHub is playing a major role for this assignment. The idea is that all group members work on the assignment in a dedicated GitHub repository. Also, the submission of the assignment takes place through GitHub, so make sure that you become acquainted with Git and GitHub soon by going throught the introductory material we provide (in particular the videos on Canvas).

One of your group's members (and only one!) is supposed to create a team for your group in GitHub Classroom. The remaining members of your group should join this team. This process is shown in detail in the introductory video and the corresponding slides `Introduction_to_GitHub.pdf`. Once a team as been created, GitHub Classroom will automatically generate a repository for this. This will be your group's repository. The group member who has created the team on GitHub Classroom automatically becomes the owner of the repository. This repository automatically contains the two datasets, `train.csv` and `test.csv`, a codebook, an example of `predictions.txt`, as well as the LaTeX templates `report.tex` and `questions.tex`. In addition, it contains the output files `report.pdf` and `questions.pdf` that you obtain when you run the templates in a LaTeX editor of your choice.

After the automatic generation of the GitHub repository in GitHub classroom, the owner of this GitHub repository (and only the owner!) is requested to connect the repository to CodeGrade so that there is **one repository per group**. We provide instructions on linking a GitHub repository in a separate document, `instructions_codegrade.pdf`. A linked repository means that there is an *automatic submission* process: the current version of the main branch in your GitHub repository at the time of the assignment deadline will be your submission! Also note that we, the teachers, have access to the main branch of your repository (and only the main branch). You are strongly encouraged to work on branches other than the main branch (in fact, it is good practice to put experimental code in other branches), but make sure that the final version of your assignment is available on the main branch by the deadline.

3.1.1 Group Registration

The assignment is supposed to be done in teams of four. You are supposed to register your group in CodeGrade. You can do so after you have linked your repository to CodeGrade. We describe the registration process in `instructions_codegrade.pdf`. Please keep in mind the following two aspects.

1. Please link your repository and register in groups via CodeGrade until **September 15, 2022, at 10:00 AM**, which is the registration deadline. If you do not yet have a group by this date, please form a group of just yourself in CodeGrade. **Do not form groups in Canvas; we will disregard those.**

2. You are only guaranteed to work with group members of your choice if your group consists of four members. If there are groups of less than four members (including groups consisting of one single member) after the registration deadline, we may break up or rearrange these groups into groups of four. We will do so on September 15 after the registration deadline and notify you of your final group in case you are affected by this rearrangement.

4 Assignment Submission

4.1 Deliverables

The assignment submission consists of delivering the following four files; please **keep these file names in your submission!**

1. `predictions.txt` is a plain text file that contains only a single bitstring of your predictions for the prediction challenge, as described in Section 2.1.2.
2. `report.pdf` is the report described in Section 2.2. This file must be the output of the template `report.tex`.
3. `questions.pdf` contains the answers to the questions in Section 2.3. This file must be the output of the template `questions.tex`.
4. Replication code of the method that generated the predictions for the prediction challenge (submitted in `predictions.txt`). This replication code may consist of several individual files or folders. This code will not be graded, but make sure that it replicates the predictions when run in the intended way.

The deadline for providing these four deliverables is **October 9, 2022, at 11:59 PM**.

4.2 Submission Procedure

You will not submit the deliverables via Canvas. Instead, the assignment submission is done automatically through the GitHub repository you have linked to CodeGrade. **Your assignment submission is your GitHub repository's main branch in its version at the assignment deadline** on October 9, 11:59 PM. Hence, make sure that the final versions of all four deliverables (`predictions.txt`, `report.pdf`, `questions.pdf`, replication files) are pushed to the **main branch** before the assignment deadline. In the moment of the deadline, we will make a copy of the main branch of the repository. We will grade the deliverables in this version of the repository and this version only. You can still make pushes to the repository after the deadline, but we will not receive these changes since we will only grade the files as they were in the repository at the deadline. In addition, we will not have access to files on branches other than the main branch, so please make sure that all assignment components you wish to submit are present on the main branch by the deadline.

To facilitate our work, please make sure that the four deliverables are in the root directory on the main branch. For your convenicence, you will find templates is this location in the repository when it is generated by GitHub Classroom. The replication code may be organized in a folder (in the root directory).

If you have questions about the assignment, its intended workflow, or the submission format, please post them in the corresponding discussion forum on Canvas.

Finally, there is a checklist for this document's main takeaways in Appendix C.

Good luck with the assignment!

References

Kidger, P. and Lyons, T. (2019). Universal approximation with deep narrow networks. *arXiv preprint: arXiv.1905.08539*.

A The Applied Preprocessing

We have performed the preprocessing as follows:

1. Irrelevant, unpredictive, repetitive features or such without clear interpretation were removed.
2. A handful of instances with contradictory characteristics were dropped from the data.
3. Categorical features are transformed to hot-encoded features. That is to say: any categorical features with n categories have been transformed to $n-1$ binary features indicating whether or not an instance is of that particular category or not. If an instance does not belong to any of the $n-1$ categories and all the features are zero, then it belongs to the n-th category.
4. We have *not* standardized, normalized, or mean-centered any variables. Doing so may help your algorithm achieve a better accuracy.

B Rubric

Subtask	**Description**	**pts**
Performance	Capped affine combination between two benchmarks	5
	See main assignment description for more info	
Report	Validity of training strategy	4
	Well-argued selection of hyperparameters	2
	Correct hyperparameter optimization	3
	Conclusion	2
	Overall clarity, motivation and completeness	4
Questions	1	2
	2	2
	3	2
	4	2
	5	2
		30

C Checklist

- ☐ I have read through the file `instructions_codegrade.pdf`. Afterwards, I have registered myself in my corresponding group in CodeGrade, and my group has a GitHub repository that is linked to CodeGrade.
- ☐ I can commit to my group's GitHub repository.
- ☐ By the assignment deadline (October 9, 11:59PM), the following deliverables are on the main branch of my group's repository in the versions that we want to submit:
  - ☐ `predictions.txt` (in the root directory)
  - ☐ `report.pdf` (in the root directory)
  - ☐ `questions.pdf` (in the root directory)
  - ☐ Replication code for the predictions in `predictions.txt` (anywhere on the main branch, possibly in own directory).
- ☐ I have double-checked that I can perfectly replicate the predictions in `predictions.txt` with the replication code we intend to submit.
- ☐ I have double-checked that the files we intend to submit are on the main branch of my group's repository.