Universitat Oberta de Catalunya

Minimum Sample Size Estimation in Machine Learning

Guillermo Prol Castelo

Area 2
Master's degree in Bioinformatics and Biostatistics

Jose Luis Mosquera Mayo
(Carles Ventura Royo)

June 2, 2022

FINAL WORK CARD

Title:	Minimum Sample Size Estimation in Machine Learning
Author:	Guillermo Prol Castelo
Tutor:	Jose Luis Mosquera Mayo
SRP:	Carles Ventura Royo
Date of delivery:	June 2, 2022
Studies:	Master's degree in Bioinformatics and Biostatistics
Area:	Area 2
Language:	English
Number of credits:	15
Keywords:	sample size estimation, machine learning, comparison of algorithms, learning curve.

Abstract

La estimación del tamaño mínimo de la muestra es una técnica útil para producir experimentos rentables y, al mismo tiempo, significativos. Sin embargo, esta estimación no es sencilla en *machine learning*, donde tenemos una etapa de entrenamiento y otra de prueba. La primera es el objeto de este estudio, ya que establece una relación entre los datos predictores y los datos predichos. En este proyecto abordamos el problema del *machine learning* supervisado para la clasificación, considerando varias técnicas de aprendizaje automático (kNN, regresión logística, *naive* Bayes y *random forest*), mediante el desarrollo de un algoritmo, basado en metodologías ya existentes, que obtiene una curva de aprendizaje en la que el tamaño del conjunto de entrenamiento es la variable independiente y la variable dependiente es una métrica: la precisión o la κ de Cohen, obtenida en el paso de test. El algoritmo ajusta una ley de potencia inversa a la curva de aprendizaje y resulta eficaz para estimar el tamaño mínimo de la muestra en el paso de entrenamiento para algunos conjuntos de datos y algoritmos de aprendizaje automático. Sin embargo, cuando aumentamos el tamaño muestral en el paso de entrenamiento vemos como la curva de aprendizaje tiende a desviarse de la fórmula de potencia inversa inicial, que aumenta sin parar. Proponemos otras direcciones posibles para mejorar la disminución de la precisión y la κ de Cohen a medida que el tamaño de la muestra supera un umbral. Sin embargo, está claro que existe un ”punto óptimo” tras el cual aumentar el tamaño de la muestra de entrenamiento no mejora las predicciones.

Abstract

Minimum sample size estimation is a useful technique to produce cost-effective and, at the same time, significant experiments. However, this estimation is not straightforward in machine learning, where we have a training and a testing step. The former is the object of this study, as it establishes a relationship between the predictor and predicted data. In this project we tackle the supervised machine learning problem for classification, considering several machine learning techniques (kNN, logistic regression, naive Bayes, and random forest), by developing an algorithm, based on already-existing methodologies, that obtains a learning curve where the training set size is the independent variable and the dependent variable is a metric: either accuracy or Cohen's κ, obtained in the testing step. The algorithm fits an inverse power law to the learning curve and proves effective in estimating minimum sample size in the training step for some datasets and machine learning algorithms. However, when we increase the number of different training sizes we see the tendency of the learning curve to deviate from the ever-increasing initial inverse power formula. We propose other possible directions to improve the decrease in accuracy and Cohen's κ as the sample size surpasses a threshold. However, it is clear that there is a "sweet spot" where increasing the training sample size does not improve predictions further.

Contents

List of Figures

List of Tables

Chapter 1

Summary

Machine learning has recently experienced a surge in use [1]. Recent advancements in computational power have made these techniques more affordable for researchers. However, machine learning implementation is usually treated as a black-box problem, where little is known about its inner work. This fact makes it necessary to develop tools to make a useful set of techniques such as machine learning easier to use for researchers from different backgrounds.

In this project, we tackle the problem of minimum sample size estimation in machine learning. In general, minimum sample size estimation is useful to produce experiments that are both conclusive and economically and time-efficient. However, the equivalent problem in machine learning has not been fully developed, even though a set of techniques have been implemented in order to study this problem [2] the question remains open.

We choose to follow the methodology of [3], where a learning curve is obtained that plots the accuracy of a machine learning classification problem (kNN, logistic regression, naive Bayes, and random forest) against the corresponding training set size. We extend this learning technique to a more general metric, Cohen's κ, which is useful in multiple class classification problems. This learning curve is fitted with a formula that gives the shape of an inverse power law curve. Once we have estimated the formula through regression, we can calculate the minimum sample size given a threshold metric.

Said formula is shown to be useful to estimate the minimum sample size in some machine learning algorithms and datasets. On the other hand, it fails to adjust to the metric and sample size spread tends to decrease as the sample size reaches a certain, large value. We propose other formulas that may be useful to improve the generalization of fitting our data, such as high-order polynomials.

Even though our fitting curve fails to predict the tendency at large training set sizes, it is clear that there is an optimum point, or ”sweet spot”. This means that after reaching a specific training set size, using larger sizes in the training step either does not improve the performance of the algorithm or even tends to decrease its performance, mainly due to overfitting.

Our work sets a precedent in the field by compiling our techniques into an `R` package that is open-source and publicly available as a GitHub repository. This package, which we name `MinSizeML` may be easily installed and has mainly one function, including its documentation, intended to determine the minimum sample size in classification problems for machine learning. All in all, researchers may use it to design their own experiments, or even improve on our codes and follow the ideas given for future directions of this project.

Chapter 2

Introduction

2.1 Thesis context and justification

When designing an experiment, one of the main hurdles consists of determining the minimum possible sample size [4]. If there are too many subjects in a medical study or too many sequences in an 'omics' study, then data acquisition, as well as the subsequent analysis, becomes time-consuming and expensive. On the other hand, a study that does not take into account a large-enough sample will lack significance—i.e., results from such a study will not be conclusive.

The field of minimum sample size determination is extensive and well-known: the problem is presented as a hypothesis test (usually Mann-Whitney for a non-parametric study, or Student's t test for a parametric study) where type I and II errors are pre-specified, and the optimal size n is determined [4, 5, 6]. Even so, its extension into machine learning (ML) is not straightforward [7].

The main objective of an ML algorithm is to predict a variable (Y) that may be continuous or categorical, given a single or a set of predictor variables (X) [1]. Given these sets of data, the ML algorithm returns a function $f(X)$ to which one passes new predictor data X to infer unknown instances of the variable Y—all of this, without adding any extra information about Y.

There are plenty of different ML algorithms used in either continuous or categorical predictions [1]. However, given their inherent differences, it is not possible to tackle the minimum sample size problem in ML with a one-size-fits-all approach—i.e., there are some metrics like accuracy and Cohen's κ [8], but these very within algorithms, so each algorithm needs to be independently assessed. It is important to address this issue, since ML algorithms are, in general, computationally expensive. Hence, determining the minimum sample size would contribute to saving money, time, and carbon emissions [9].

2.2 Thesis objectives

The **main objectives** of this project are:

1. Implement an algorithm to study the minimum sample size—given a set of parameters—required for different learning algorithms in a classification problem.

2. Build an R package that enables systematic calculation of minimum sample size in different machine learning algorithms.

In order to achieve our goals we have considered the following **specific objectives**:

1. Compile state-of-the-art techniques to estimate minimum sample size in ML.

2. Select which ML algorithms are feasible to study during this thesis.

3. Apply different assessment metrics for sample size estimation, namely: ROC curve, ROC AUC, and learning curve.

4. Make the resulting R package publicly available.

The first three specific objectives are related to the first main objective, since the former are basically the generalized steps needed to achieve the latter. Meanwhile, the fourth specific objective is a further step once the second main objective is achieved.

2.3 Approach and methodology

The algorithm which we are based on [3] is intended to reduce the computational expense of supervised ML algorithms for classification by fitting a power-law curve, i.e. the relation of predictive power and sample size. This fitting allows a considerable reduction in the number of times one needs to run the ML algorithm at different sample sizes. Importantly, we may save time on very large sample sizes since the predictive power does not improve much after a certain sample size threshold [3].

Another acceptable approach may be to generate artificial cohorts of data based on a given dataset [10], then divide it into a development set and a validation set. The former is used for predictions, while the latter is intended as a reference for the comparison of predictions. The problem with this design is the high computational expense and its inherent complexity. Due to the limited time available, we choose to undertake the approach described in the previous paragraph.

2.4 Thesis planning

Here we describe the different tasks and steps taken to complete this thesis. We provide a calendar in the form of a Gantt chart (figure 2.1) and, in table 2.1, we break down the chronological details of the main objectives, as well as their degree of completion.

Objective	**Begin date**	**End date**	**Allocated hours (from PEC 1)**	**Hours spent**	**Completion %**
Algorithm implementation	08/03	25/03	60	60	100
R package	12/04	25/04	44	33	100
Shiny	26/04	—	22	0	0
Compile techniques	23/02	01/03	20	20	100
Select ML algorithms	21/03	01/04	20	20	100
Metrics	29/03	06/04	20	20	100
Improvements	26/04	09/05	—	33	100

Table 2.1: Objectives and their chronology.

2.4.1 Main tasks

1. Literature research. Divided up during the first two *PECs*, it is necessary to get acknowledged with the state-of-the-art methods for minimum sample size estimation, both in general and, specifically, in ML.

2. Develop an algorithm for the analysis to undertake. This shall be based on [3].

3. Select which algorithms fit our study, namely, supervised ML algorithms used for binary classification. These algorithms will be examined through the algorithm described in the previous bullet-point.

4. Compare the performance of different algorithms through different metrics.

2.4.2 Calendar

The previous tasks, as well as other more general tasks (writing reports, thesis, etc.), have been planned for and details are shown in the attached Gantt chart (page 4). Following, there is an estimation of the number of hours for each *PEC* and main tasks, at around 22 h per week and taking into account tasks overlapping in time:

- PEC 0: 20 h
  - Discussion with supervisor: 2 h
  - Main literature compilation: 10 h
  - Write report: 8 h
- PEC 1: 32 h
  - In-depth literature search: 20 h
  - Review objectives with supervisor: 2 h
  - Write report: 10 h
- PEC 2: 110 h
  - Develop code: 57 h
  - Select ML algorithms: 20 h
  - Compare algorithms with metrics: 20 h
  - Write report: 13 h
- PEC 3: 77 h
  - Code improvements: 11 h
  - Compile codes into R package: 33 h
  - Add other methods of data fitting: 22 h
  - Write report: 11 h
- PEC 4 (write thesis): 90 h
- PEC 5a (video presentation): 9 h
- PEC 5b (rehearse public presentation): 9 h

2.4.3 Milestones

1. By the end of *PEC 2*, the main structure of our pipeline should be finished.
2. By the end of *PEC 3*, all of our codes should be compiled in a package that may be distributed.

2.4.4 Risk analysis

- We should take into account that unexpected issues always arise during a project. This becomes especially relevant in a project that lasts only four months. Hence, the key steps have been scheduled during the development of *PEC 2*. If it is finally not possible to end the corresponding objectives on time, more time will be allocated during the development of *PEC 3*.
- In line with the previous risk factor is the relevance that a new package may have to the public. Falling short on the tasks set during *PEC 2* may mean that producing a relevant package would require further work.
- ML tends to be computationally expensive, hence the number of algorithms we are able to study may not be extensive. However, the fact that our code will be publicly available means it may be expanded by other developers.

2.5 Brief summary of contributions

We have already presented a summary of our work on Chapter 1, and the present Chapter 2 has served to illustrate the usefulness of our project, and our main objectives. In Chapter 3 we describe the current state of the art, so that the reader may see how our project fits into the global context of sample size estimation in machine learning. We describe the methods used in Chapter 4, focusing on what packages and functions we have used to build our main algorithm. Chapter 5 presents our main results: we describe the algorithm that we have implemented and which allows us to obtain different learning curves for accuracy and Cohen's κ for a varying training set size. We discuss these results in Chapter 6, considering which aspects are lacking and which are useful in the field. Chapter 7 consists on an economic assessment, but we also focus on the environmental impact of our project. In Chapter 8 we give the main conclusions of our work, point possible directions for future related projects, and critically study the accomplishment of our initial goals. Finally, Chapter 9 consists of a glossary of the relevant terminology.

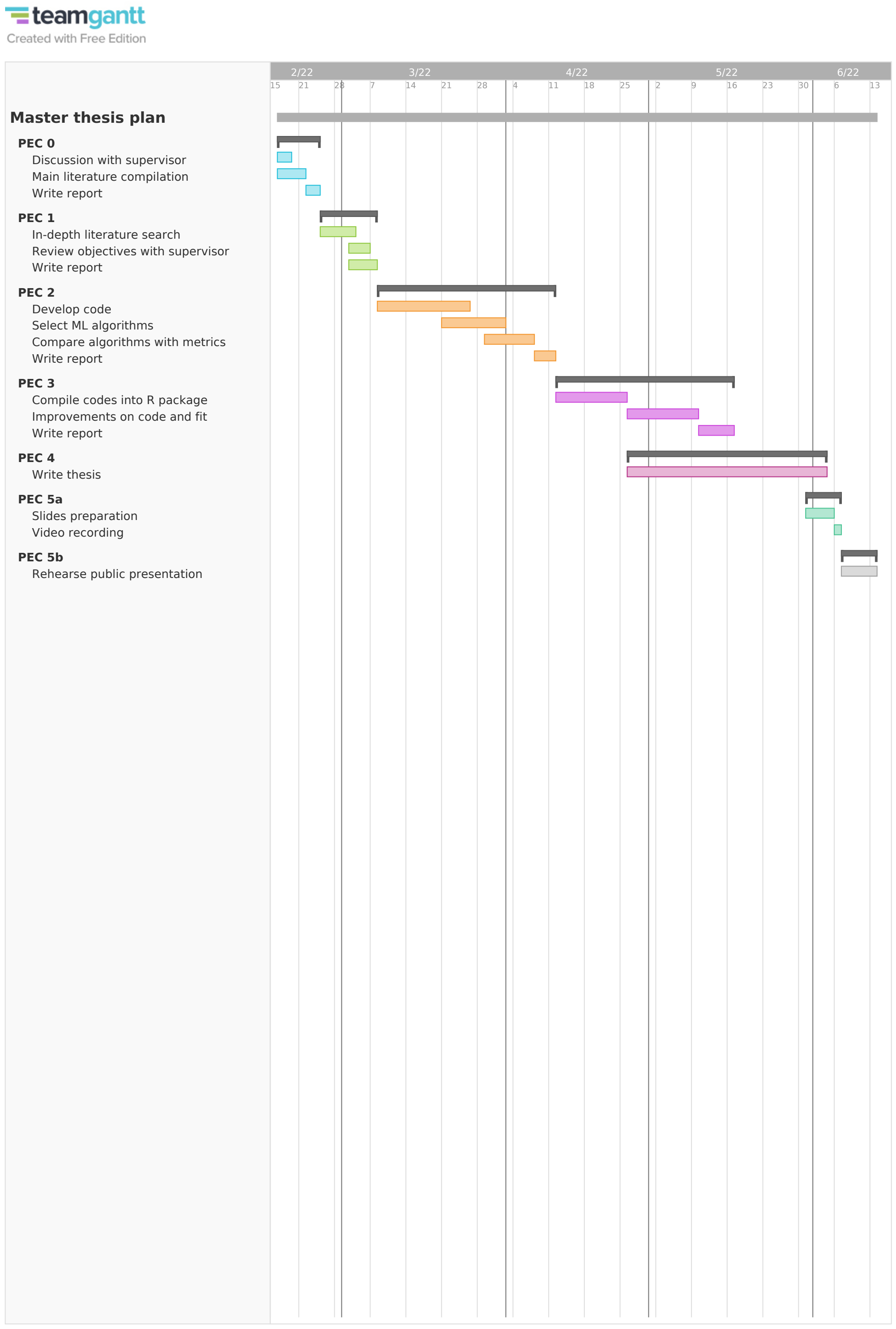


Figure 2.1: Updated Gantt Chart, showing how time has been assigned to each task.

Chapter 3

State of the art

The world of sample size estimation has been extensively studied to produce robust methodologies to determine the minimum sample size needed in a study. These methodologies are based on a hypothesis test where, in the simplest scenario, we need only know one pre-determined value of the usual spread of our data (i.e. the standard deviation of the population in question) and three parameters we can choose: α or type I error, β or type II error (alternatively, we may provide the power, i.e. $1 - \beta$—see table 3.1), and the magnitude of a significant difference [4]. In R, there are several packages which contain functions to determine the minimum sample size given the aforementioned parameters. These functions are useful since they allow researchers to better plan their experiments, since knowing the exact number of repetitions and/or samples an experiment requires to be relevant will save them time and money.

		Truth	
		Positive	Negative
Test	Positive	True Positive	False Positive Type I Error α
	Negative	False Negative Type II Error β	True Negative $1 - \beta$

Table 3.1: Possible outcomes of a hypothesis test.

Machine learning (ML) has been rapidly gaining importance in many fields related to information management. Recent advancements in computer power have also contributed to an ever-increasing interest in ML within many disciplines: from omics data analysis, to market investment, or improving fishing [11]. These algorithms allow us to predict rather complex phenomena given an input data and, optionally, the corresponding output. Within machine learning, we find supervised and unsupervised learning. The former is characterized by the lack of a known output of the problem, while in the latter we do know the corresponding output. Furthermore, ML may be used to solve either classification problems, where a finite number of categories are being predicted, or regression, where continuous data is being predicted.

While it may be tempting to assume the same minimum sample size estimation strategy in ML as in a classical study, there is a fundamental difference. ML requires a training step where the algorithm uses a ratio of the input data to find the best ”function” that relates predictor and predicted data. Therefore, the effectiveness of any ML algorithm depends on the training step. This fundamental difference requires a rather different approach when estimating the optimal or minimum sample size for training data in ML while producing a robust prediction.

To estimate minimum sample size in ML we need a metric that describes the robustness of the prediction. We can find approaches in the literature that use Receiver Operating Characteristic (ROC) curve [12] and its ROC Area Under the Curve (ROC AUC)[13]. On the other hand, we may also find learning curves useful, where a metric, usually accuracy, is calculated given a sample size [14]. Both of these metrics are useful when working with binary classifications (compared in figure 3.1). Accuracy may even be used in non-binary classification. However, it may be misleading when working with class-imbalanced data. Therefore, a more general metric, such as Cohen's κ [8], may be used to obtain a learning curve for a varying sample size in ML. Cohen's κ is given by the formula

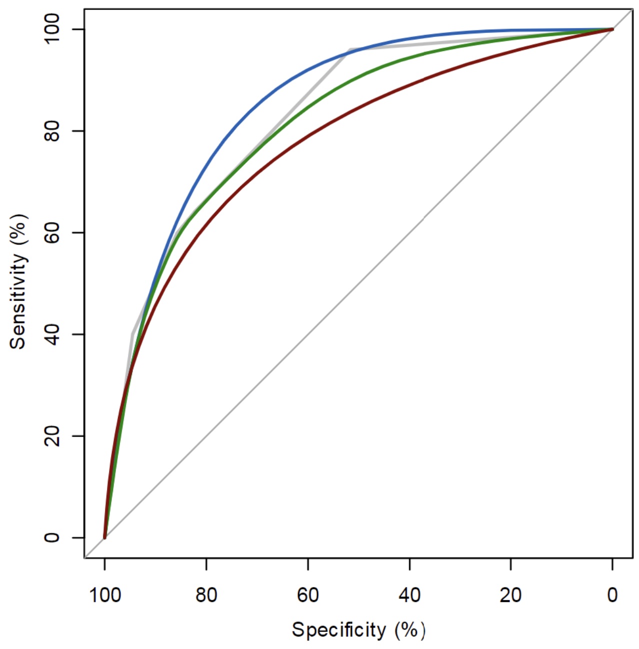
$$\kappa = \frac{p_0 - p_e}{1 - p_e}, \tag{3.1}$$

where p_0 is the observed agreement (this is the same as the accuracy), and p_e is the chance agreement (probability that the categories in a study coincide when drawn at random). The possible values of Cohen's κ are $\kappa \in [-1, 1]$, where $\kappa = 1$ indicates complete agreement, $\kappa = 0$ indicates no agreement other than what would be expected by chance, and $\kappa = -1$ would indicate a configuration such that the agreement of the predictions is worse than what would be achieved by drawing random predictions.

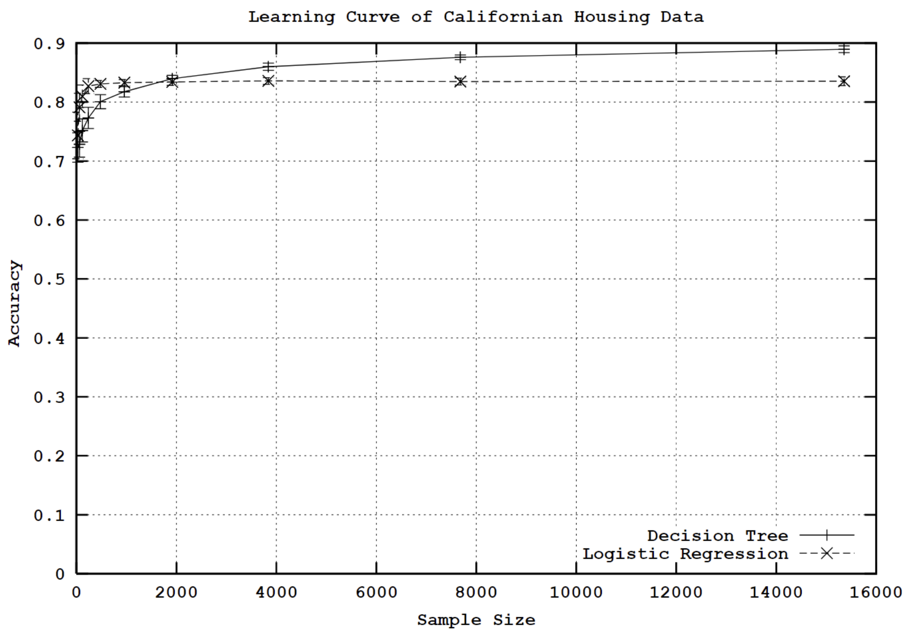
There are already algorithms described in the literature that allow analysis based on a learning curve that plots accuracy of predictions at different sample sizes for training, and fits the data to a fixed function [3]. Other studies tackle the problem through ROC AUC [10].

Besides, there are other, more recent and complex techniques to optimize the sampling during the training step of several ML algorithms [15, 16], treated as a multiple comparison problem, where methods like the Bonferroni correction or Holm's step-down procedure in order to estimate the minimum sample size of the different algorithms under study, given a specific problem.

Due to the time constraints in our project, we chose to tackle a rather simple problem: can we determine the minimum sample size for a specific classification problem and through different classification algorithms? For this reason, we chose to base our procedure on [3] to obtain a learning curve after using different sample sizes in the training step of an algorithm. Furthermore, in order to overcome the limitations of using accuracy as a metric, we consider expanding the procedure to also calculate Cohen's kappa when the classification problem is non-binary. Therefore, each ML algorithm will be assessed individually, without the multiple comparisons of [15, 16].



(a) ROC. Source: [17].



(b) Learning curve. Source: [14].

Figure 3.1: ROC and learning curve. (a) Receiver operating characteristics (ROC): from a binary classification problem, we can plot the sensitivity angainst specificity. The area under the curve (AUC) determines the degree of goodness of the experiment; values closer to 1 have less errors. (b) Learning curve: a metric, such as accuracy, may be plotted against the sample size. We can fit a curve to the data and, given a threshold metric, calculate the corresponding sample size.

Chapter 4

Methodology

During the development of this thesis we have used the `R` programming language (version 4.1.3) [18] and the following packages: `caret` (version 6.0.91) [19], `doParallel` (version 1.0.17) [20], `foreach` (version 1.5.2) [21], `stats` (version 4.1.3) [18], and `synthpop` (version 1.7.0) [22].

In the rest of this section, we detail the functions and procedures used to create the algorithm with which we have obtained the results of this thesis.

4.1 Data division

The first step in our thesis was to find adequate data to apply different ML classification algorithms. By adequate, we mean that data must contain one categorical variable, either binary (e.g., Yes/No, True/False, A/B...) or with multiple categories (e.g., different species of plants or a discrete ranking from 1 to 5). This variable is the predicted variable, which we may call Y.

On the other hand, we need the data to contain one or several predictor variables X, which may be any type of data, either categorical or continuous. The predictor variables X and predicted variable Y are divided into a dataset and vector, respectively, and passed to our algorithm.

Besides, these data we have found may need some pre-processing from the original dataset. First, to simplify our work, we simply removed instances of null or missing data. Second, we have also done some transformations of the data, namely normalization, and z-transformation, since this may help the ML algorithms to improve their predictions. Lastly, we have also considered the size of the datasets. Since some of the datasets we found have a small number of samples (for example, about 100), the predictions made on them may not be representative and we also want to make our algorithm as general as possible, which means we must work with large datasets (we considered that 1,000 samples to be ideal). For these reasons, we have created synthetic data from datasets from the original `iris` using the `R` package `synthpop`.

In the end, we have worked with four datasets: `Smarket` [23], which aims to predict the direction of different market assets (i.e., predict whether the asset will move up or down in value), `iris` [24], where several petals and sepal measurements are used to predict the corresponding species of iris (*setosa*, *versicolor*, and *virginica*), and the Breast Cancer Wisconsin (Diagnostic) Data Set [25] (downloaded on 14/03/2022), which contains measurements of several features of cell nuclei, and the corresponding cancer diagnostic (i.e., malignant or benign).

4.2 Data training and testing

The next step in the design of our algorithm incurred mainly in the use of the `R` package for machine learning called `caret`. At this stage, as usual with ML algorithms, we need to divide the input data, both predictor and predicted variables, into a training and a testing dataset. Thus, we will have four sets of variables: two sets of predictor variables, one for training and one for testing, and two sets of predicted variables, also one for training and one for testing.

The former, the training sets, are passed to `caret`'s `train` function, which obtains the relationship between the predictors and predicted variables. Next, the testing set is used in a two-fold manner. First, new instances of the predicted variable are acquired using the outcome of `train` with the `predict` function and the testing set of predictor variables. Second, the new predicted variables are compared to the early testing set for the predicted variable. This way, we can calculate a metric for the prediction step: either accuracy or Cohen's κ (formula 3.1).

Since we aim to estimate the minimum sample size needed to obtain a certain value of accuracy or Cohen's κ, we need to systematically change the amount of data that is used in the training step. For this, we use the `createDataPartition` function, also from `caret`, where we can state the percentage of training data into which split the input predictor and predicted variables. Furthermore, we need to change said percentage of training data step by step, so we design a loop where the steps described in this section are repeated with an increased portion of the input data assigned as training data.

Lastly, we must consider the high computational cost of ML algorithms. For this reason, we choose to parallelize the loop on the percentage of training data. We may do so through the `R` packages `foreach` and `doParallel`, which allows us to create a parallel `for` loop and use several logical threads available in our computer.

4.3 Fitting data

Once the loops are finished, we have created a large amount of new data, that is the sample size used during the training step, and the corresponding calculated values of accuracy and Cohen's κ. We can therefore build a learning curve by plotting the distribution of either metric against the sample size.

We can also calculate the corresponding fit of the curve that we use, proposed by [3]. To do so, we use the `nls` function available in the `stats` package. In the same way, we can calculate a 95% confidence interval (CI) of the fitted curve, resulting in a total of three curves.

By solving the fit equation with the estimated parameters, we can now know the sample size required to obtain a desired, minimum value of either metric (accuracy or Cohen's κ), i.e., we may estimate the minimum sample size for a given value of a metric.

Chapter 5

Results

5.1 Algorithm

We have been able to reproduce the procedure used in [3] by creating our own algorithm for classification, using mainly functions from the *caret*[19] package in R. To do so, we need some input data, divided into a set of variables X used to predict and a single variable Y which we want to predict. We then followed these steps.

First, we divide the input data into training and testing data sets. This is done by randomly selecting a pre-specified ratio of the elements in the input data to use in the training step and the remaining data for the testing step. Thus, we now have two subsets of each input data X (which we may call $trainX$ and $testX$) and Y (which we may call $trainY$ and $testY$). Second, we train the model using the corresponding training data subsets. In this step we used four different algorithms: k-nearest neighbors (kNN) [26], logistic regression [27], naive Bayes [28], and random forest (RF) [29]. Third, we use the trained model and the $testX$ set to make new predictions. We may assess the metric (accuracy or Cohen's κ) of these predictions by directly comparing them to the $testY$ data. Fourth, we save the metric of this prediction and the length of the training subset we obtained in the first step.

We repeat this process by varying the ratio of the input data to be used in the training step (e.g., we start with a 10% ratio which selects 10% of the input data for the training step and keeps the remaining 90% for testing; this ratio is increased in the subsequent iterations). Hence, we obtain two vectors: one with the sample size used in each iteration, and the corresponding accuracy of the predictions. We can fit this data to a curve of the form:

$$Y_{metric}(X_{size}) = (1 - a) - b \cdot X_{size}^{c}, \tag{5.1}$$

where a,b, and c are the parameters we want to obtain from fitting the data $Y_{metric} \in [0, 1]$ or metric and X_{size} or training sample size. Besides, we also obtain the 95% confidence interval of the fitted curve as two new curves, with their corresponding set of parameters $\{a', b', c'\}$ and $\{a'', b'', c''\}$. From these fittings, we can solve the minimum sample size N_{min} and the corresponding upper and lower bounds, given the desired accuracy.

5.2 Accuracy

For this section, we have used as input the Breast Cancer Wisconsin (Diagnostic) Data Set [25] and the prediction problem consists of determining, given different features of a breast mass, whether the diagnosis is benign or malignant. Figure 5.1 shows the result of the predicted accuracies for varying training set sizes, and the corresponding fitted curve of accuracy (using formula 5.1), including its 95% CI curves. We also write down the estimated minimum sample sizes and their confidence intervals for a 90% accuracy.

5.3 Cohen's kappa

We can also perform a study on the minimum sample size based on the minimum desired value of Cohen's kappa, instead of just the accuracy, using the same formula 5.1 as previously.

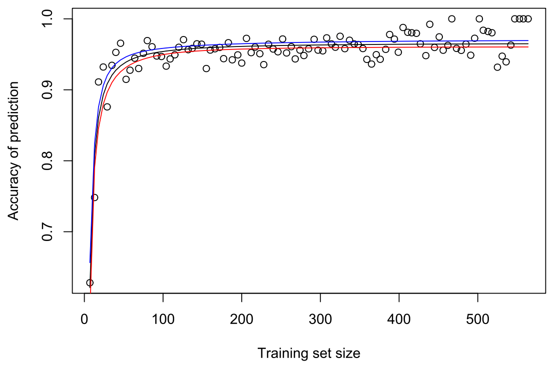
To corroborate its correct functioning we have used the `iris` dataset, included in base `R`, where the classification problem consists of assigning one of three species of iris, based on measurements from their sepals and petals. However, the original dataset contains only 108 plants measured, so we proceeded to create a synthetic dataset of 1,000 plants starting from the original set and using the package `synthpop`.

We then performed our method on the synthetic data using the kNN and random forest algorithms for multiple class classification. Figure 5.2 shows the result of the predicted Cohen's kappa for varying training set sizes, and the corresponding fitted curve of Cohen's kappa, including its 95% CI curves. We also write down the estimated minimum sample sizes and their confidence intervals for a 90% Cohen's kappa.

As we may see, the adjustment of the fitted curve for the metric, i.e.

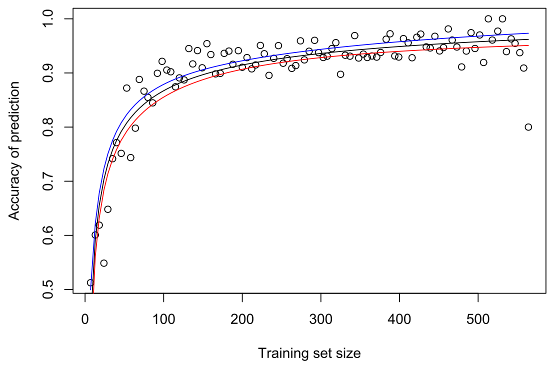
$$Y_{metric}(X_{size}) = (1 - a) - b \cdot X_{size}^{c},$$

does not coincide with the actual spread of the data ($R^2 = 0.178$ for kNN, and $R^2 = 0.0695$ for random forest). This shows that the formula proposed by [3] is not always recommended to extrapolate the value of a metric to high values. Looking at the right side of the plots, we see that Cohen's kappa tends to fall as the training set size increases, which may be due to overfitting.



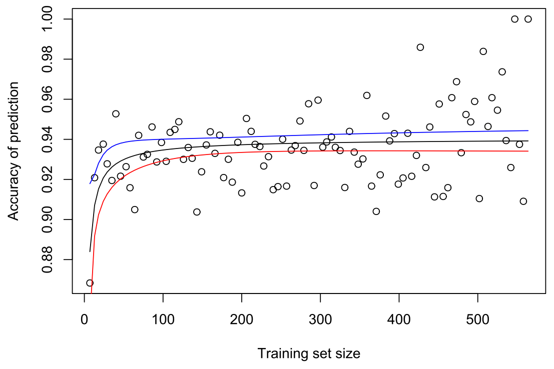
(a) kNN.

$N_{min} = 26.1$, with 95% CI: [22.8, 29.6]



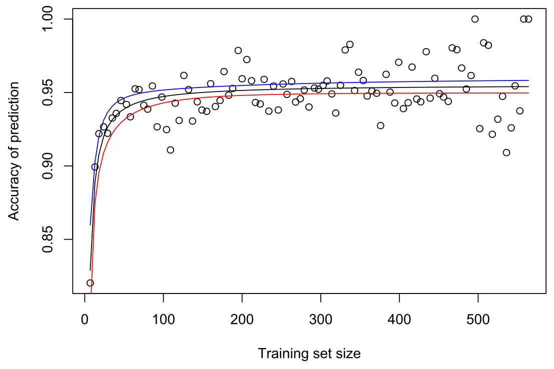
(b) Logistic regression.

$N_{min} = 157.4$, with 95% CI: $[145.9, 170.3]$



(c) Naive Bayes.

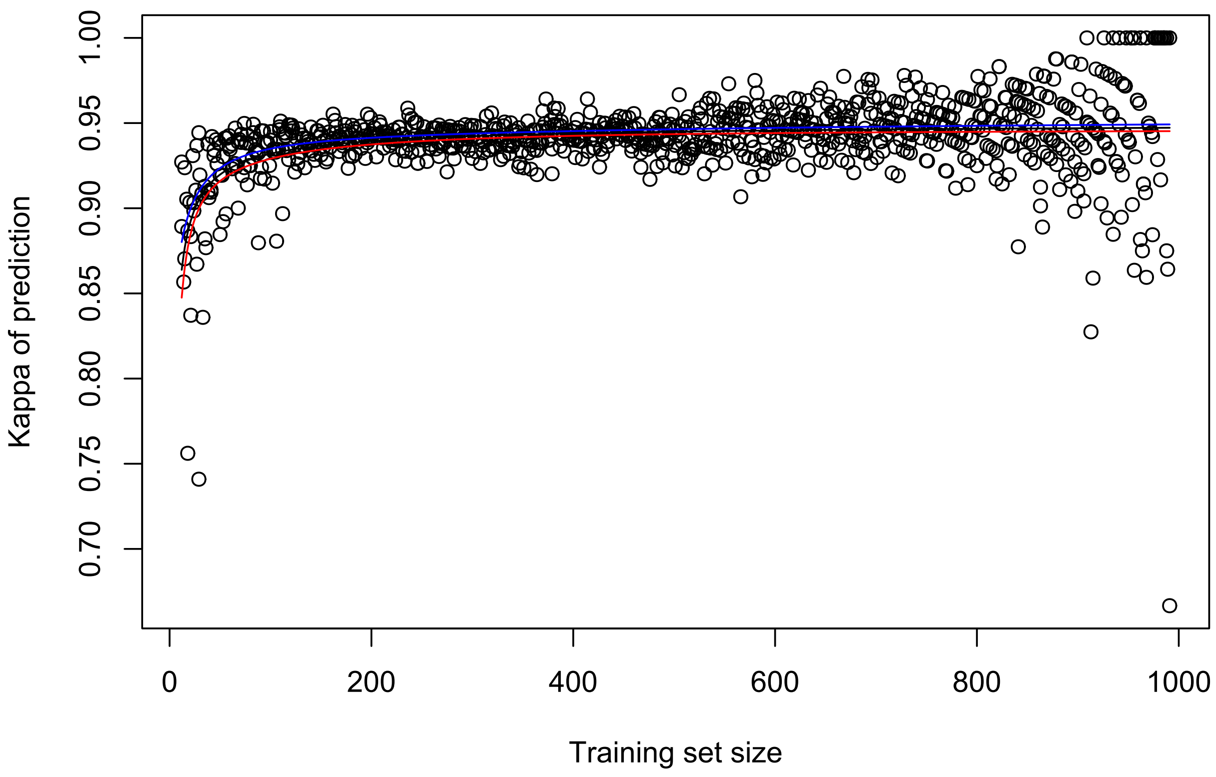
$N_{min} = 10.4$, with 95% CI: [3.3, 17.4]



(d) Random forest.

$N_{min} = 15.6$, with 95% CI: [11.8, 19.6]

Figure 5.1: Result of accuracy vs. training set size, using several machine learning algorithms for classification of the Wisconsin breast cancer database. Each graph shows a scatter plot of dots, representing the level of accuracy of the classification with a corresponding sample size for the training step. There are three continuous lines on each plot. The middle one (black) corresponds to the fitted data, given by the scatter plot. The upper one (blue) shows the upper limit of the 95% confidence interval for the fitted curve, while the lower one (red) shows the lower limit of the same confidence interval. N_{min} represents the minimum sample size for 90% accuracy, and the CI is the corresponding sample size calculated with the upper and lower curves.

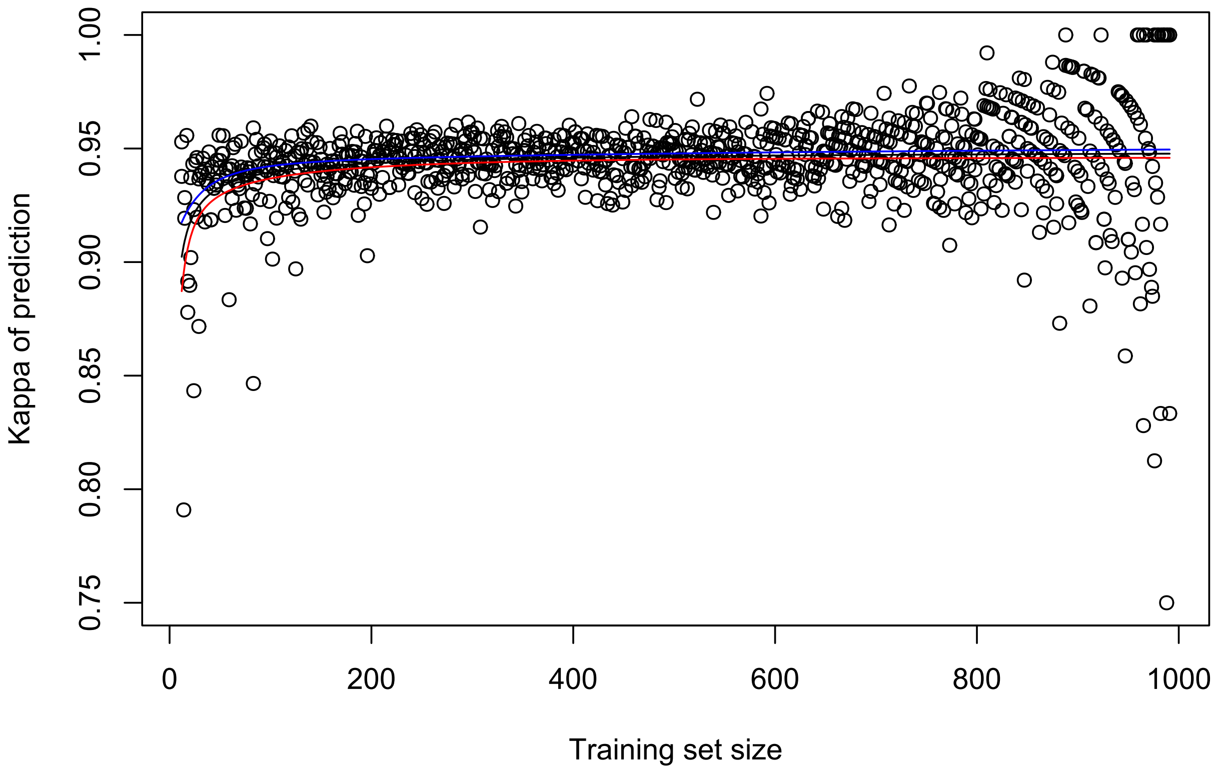


(a) kNN.

$N_{min} = 25.3$, with 95% CI: [20.4, 30.0]

$R^2 = 0.178$

Figure 5.2: Cohen's kappa vs training set size. Continues on next page.



(b) Random forest.

$N_{min} = 11.2$, with 95% CI: [6.8, 15.5]

$R^2 = 0.0695$

Figure 5.2: Cohen's kappa vs training set size. Result of Cohen's kappa vs. training set size, using kNN and random forest for multi-class classification of the synthetic *iris* database. Each graph shows a scatter plot of dots, representing the level of Cohen's kappa of the classification with a corresponding sample size for the training step. There are three continuous lines on each plot. The middle one (black) corresponds to the fitted data, given by the scatter plot. The upper one (blue) shows the upper limit of the 95% confidence interval for the fitted curve, while the lower one (red) shows the lower limit of the same confidence interval. N_{min} represents the minimum sample size for 90% Cohen's kappa, and the CI is the corresponding sample size calculated with the upper and lower curves. We also provide the R^2 of the middle line fit.

5.4 R package

In the previous sections, we described the functioning of our algorithm and used it to predict accuracy or Cohen's κ given a varying sample size of a dataset. We now concentrate on describing our package, which implements said methodology, using a variety of machine learning algorithms for supervised learning classification: k-nearest neighbors (kNN) [26], logistic regression [27], naive Bayes [28], and random forest (RF) [29].

We have compiled all of our codes into a distributable, open-source R package which is already available on GitHub (`https://github.com/gpcastelo/MinSizeML`) under an MIT license. Furthermore, our package may be easily installed through the command

```
devtools::install_github(MinSizeML)
```

To make our analysis easier for other users, we have encompassed the method into a single function:

```
MinSizeClassification(
  X,
  Y,
  algorithm,
  metric,
  thr_metric,
  formula_rhs = "(1-a)-b*X^c",
  start_parameters,
  p_vec = 1:99/100,
  cv_number = 5,
  show_plot = T,
  n.cores = 1
),
```

where the user provides the variable or set of variables `X` used as predictors, a vector of factors to predict `Y`, the machine learning `algorithm`, which `metric` to use (either "Accuracy" or "Kappa"), the minimum metric value for which the minimum sample size is calculated `thr_metric`, the right-hand side of the formula, `formula_rhs`, to fit the spread of metric values vs their corresponding sample size, a vector `p_vec` of ratios to divide training data into (i.e., the code loops through the different ratios to get a sample size and calculate the corresponding metric), the n-fold cross-validation `cv_number`, whether or not to show the plot of metric vs sample size and fitted curves with `show_plot`, and the number of cores to use with `n.cores`.

All in all, we have seen that the simple formula 5.1 for predicting accuracy or Cohen's kappa, with which we have worked all during this project, may be useful in specific cases (as seen in the previous section), but a more general procedure would help improve the correctness of minimum sample size estimation. Namely, our package provides an option for the user to use their own custom functions to fit the spread of metric values and sample size.

Chapter 6

Discussion

We had started out this project with the objective of reproducing the methods used in [3]. As may be seen in figure 5.1, we have indeed been able to reproduce their methodology. Importantly, the formula 5.1 they proposed for the accuracy as a function of the train step sample size has proven to be valid for some datasets and ML classification algorithms. Furthermore, this formula may also be extended to a more general metric, Cohen's κ, to allow the minimum size estimation in multiple-category classification problems.

Nevertheless, we have seen some inadequacies and limitations of said methodology. Firstly, not all the algorithms for binary classification seemed to be properly fit for the given formula 5.1 of accuracy. Especially when using naive Bayes as the predicting algorithm (figure 5.1c), the formula is not descriptive of the real spread of the data. Secondly, we have observed that the data tends to increase its spread as the training set size increases, showing a funnel-like shape, where smaller training set sizes yield accuracies closer to the fitted curve, and larger training set sizes yield accuracies that spread out far from the fitted curve.

Both of these effects are more obviously seen when increasing the number of points when training the algorithms, as seen in figure 5.2. Again, we observe a close resemblance of the fitted curve at low- to midpoints of the training size, but a further spread at larger sizes. Moreover, we see that the tendency of the metric is downward, starting at around a training test size of 800.

From the larger spread and downward tendency of the metrics for larger training sizes, we reason that this effect has a two-fold cause. On the one hand, the larger the training size we use, the smaller the testing set size will be. Therefore, as the testing size is decreased, the metric (either accuracy or Cohen's κ) is obtained from a reduced set of values, rendering the outcome not significant. That is why in figure 5.2 we observe both the downward tendency of the metric, but also points where the prediction is almost perfect ($\kappa \sim 1$)—i.e., the calculation of the κ is random, depending on which few specific instances of the testing set are being used to calculate the metric.

On the other hand, the downward tendency of the metric is clear when the training size reaches a certain value. This is due to the training data being overfitted. This means that the ML algorithm has "learned" the data too well, so much so, that it replicates the same noise as the data used for training. Hence, there is a larger error rate of classification and, when comparing the predicted and testing models, overfitting is reflected as a larger error, conversely, a lower κ of Cohen.

All in all, we see that the formula (equation 5.1) and procedure used in our project and in [3], may be useful in certain scenarios (figures 5.2a, 5.1b, and 5.2b). However, the formula for fitting metrics and training sample size is not always reflective of the measured metrics (figure 5.1c and 5.2). Thus, the need for a more general method to estimate minimum sample size in machine learning problems for classification.

Chapter 7

Economic assessment

There have been no direct costs associated with the development of this project. In fact, both the programming language `R` used to implement our algorithm, the suite RStudio where all the codes and documentation were written, and the `R` packages implied in the development (namely `caret`, `doParallel`, `foreach`, `stats`, and `synthpop`) are all either free to use or even open-source.

The resulting product, our `MinSizeML` package is hosted on the free-to-use platform GitHub. It is publicly and freely available to anyone who wants to either use it or develop upon it. Hence, there are no associated economic benefits to our project.

However, we should take into account several factors on why this project has been ”free”. First, the author has not had any income associated with the development of the project at any stage. According to the website Glassdoor, the mean gross salary of a bioinformatician in the Barcelona area is 30,732 € per year (link here). In the approximately four months this project has taken to develop, that would correspond to gross spending of 10,244 € for the employer.

Second, the use of electricity is another cost assumed by the student. The exact cost of the electricity used in this project is rather cumbersome to calculate given the daily and hourly price variations of electricity. We can, however, state that our main tool for using machine learning algorithms has been the central processing unit (CPU) AMD Ryzen 7 5800HS, with a Thermal Design Power (TDP) of 35 W.

Machine learning is certainly power-hungry, and recent developments in graphic processing units (GPU) have made these preferred over CPUs for machine learning [9]. GPUs are even more power-hungry than CPUs. For example, a popular GPU is the Nvidia RTX 3080, with a TDP of 350 W.

All of this power consumption implies not only economic costs but also environmental costs, as they have associated CO_2 emissions. Thanks to the Machine Learning Emission Calculator (MLEC, `https://mlco2.github.io/impact/`) developed in [9], we can estimate that the emission of some of the most popular GPUs. As a rough estimation, out of the 60 hours used for algorithm implementation (as stated in table 2.1) we can say that about 30 hours were used on running the machine learning algorithms. According to the MLEC, an RTX 3080 GPU used for 30 hours would emit 5.38 kg of CO_2 (the equivalent of driving for 21.7 km in an average car). Since our CPU's TDP is about a tenth of said GPU, we can roughly estimate that our emissions have been a tenth of those emitted by an RTX 3080 GPU for the duration of our project.

To the economic and environmental cost, we would need to add the rest of the computer parts and the computers themselves. That is to say, the cost of establishing a research group from scratch to tackle the same problem as our project would be high. Nevertheless, we believe our project would be profitable in the long run since it implements a systematic way to get a minimum sample size, instead of a process of trial-and-error to get the same result.

Chapter 8

Conclusions

8.1 Conclusions

Here we describe the conclusions of our study and relate them to the main and specific objectives described in section 2.2.

1. We have seen that it is sometimes possible to predict the minimum sample size following the approach of [3], to estimate metrics using equation 5.1. However, this methodology is not generalizable to every possible ML classification problem. Hence, a more general approach should be established to account for overfitting when using large sample sizes (where the training set is a lot larger than the testing set).

2. It is clear, however, that there is a "sweet spot": once we reach a certain training set size, the calculated metrics do not improve much or even decrease with larger sizes. One possible solution may be to use some high-degree polynomial (e.g., a sixth-degree polynomial) as a possible function for fitting. This new formula could be compared to the performance of equation 5.1 through R^2, $RMSE$, and MAE to obtain the most fitting for a given problem. Furthermore, we could consider other more general regressions, like `loess`.

3. We have collected all of the codes developed during these months and created an `R` package that is open source and may be easily installed. This package allows for the systematic calculation of minimum sample size in different machine learning algorithms. Besides, we provide all of the datasets used during the package development so that our results may be reproducible.

Conclusion number 1 marks the accomplishment of our first main objective (to implement an algorithm to estimate minimum sample size in ML classification problems), and the specific objectives 1, 2, and 3. The second conclusion is a by-product of our first conclusion. It was not an achievement that we were expecting from the beginning. However, it may be the most useful contribution of this thesis regarding future developments in the field. Lastly, the third conclusion marks also the accomplishment of our second main objective and our forth specific objective.

Besides, we initially set ourselves to build an online application using Shiny so that researchers without expertise in machine learning may easily calculate a minimum sample size for their experiments. We decided not to pursue this objective due to the more interest being in making our package more versatile with fittings of metrics and formulas data that do not align with those given by [3].

8.2 Future directions

We have not been able to fully explore alternative formulas that would better fit the metrics vs training sample size data spreads. Future projects should explore high-order polynomials and more general regression methods like `loess`.

We shall mention that the machine learning problems treated in this project are only for classification. In continuous-data prediction, we cannot use the same metrics as we have used, but rather R^2, $RMSE$, or MAE. Hence, the proposed equations would not be valid in the case of estimating the minimum sample size for machine learning using continuous data.

Besides, even though we decided to drop the development of a Shiny application, developing such an easily accessible tool online would be helpful for researchers that are not experienced in machine learning. After all, machine learning has become increasingly popular in recent years, but its intricate functioning means that researchers not specialized in computing would see any tool that helps them overcome the hurdle of learning how ML works as beneficial.

8.3 Planning follow-up

Our objectives have remained mainly unchanged, and we have fulfilled the most important ones during this work phase. Some minor comments should be taken into account.

The development of *PEC 0* and *1* complied with their planning. These *PECs* involved the phase of bibliography research and planning for the remainder of the project. Even though the bibliography on our topic is scarce, we managed to get a general idea of the state of the art from the beginning. We took into account different strategies for developing our algorithm and decided to follow [3] strategy since its simplicity would fit into a project of a length like this one.

During the first working phase, i.e. *PEC 2*, we focused on implementing the methodology (as described in Chapter 4) and began to build learning curves. We mostly used the `Smarket` dataset, which includes financial data but decided to look for a more biologically-related dataset.

Thus, we ended up working with the Breast Cancer Wisconsin (Diagnostic) Data Set, for which we showed the corresponding learning curves in *PEC 2* and here in figure 5.1. The major time drawbacks during this time were related to the use of different metrics when estimating the minimum sample size. We had initially thought ROC would be appropriate, but it is only intended to assess binary data.

In *PEC 3*, the second working phase, we realized some of the data did not fit the unlimited growth in accuracy or Cohen's κ we had initially proposed. Besides, we had initially intended to development of an online-available Shiny application, which would have eased the access of our calculations to researchers. This did not come to fruition after a cost vs usefulness study. That is, it would have taken very long to develop, while we had developed an `R` package that is already user-friendly by itself. Hence, our decision to drop the Shiny app development and focus on making our code as intuitive as possible.

Chapter 9

Glossary

In this chapter we present the most relevant terminology and acronyms, alongside their definition.

- Machine learning (ML): usually classified as a field within artificial intelligence, it is a discipline that builds algorithms to model a given set of data (training data). The resulting model (or predicted data) is the result of one or several decisions that the algorithm is programmed to undertake, even though the algorithm is not explicitly programmed to obtain said model.
- Supervised learning: branch of ML that requires known observations of the predicted dataset. Thus, the output from a ML algorithm may be quantitatively assessed.
- Metric: in this work we use this term to a function that indicates the difference between two points in a set. We consider two of these, applied to the predictions from ML:
  - Accuracy: it tells us how close is our prediction to the true value. We calculate it as a ratio between true positives and the total number of observations.
  - Cohen's κ: this metric is considered to be more robust when dealing with multiple-category classification [8]. It is also calculated as a ratio, as given by equation 3.1.
- Learning curve: in the context of our ML predictions, it represents the relationship between how good a prediction is (given by a metric) and the amount of its experience (given by the training set size) [14].
- PEC: acronym in Spanish, stands for *Prueba de Evaluación Continuada*, which are the standard deliverables used by the Universitat Oberta de Catalunya to grade their students performance along the duration of the school year.

Bibliography

[1] Davide Chicco. "Ten quick tips for machine learning in computational biology". In: *Bio-Data Mining* 10.35 (Dec. 2017). DOI: 0.1186/s13040-017-0155-3.

[2] Indranil Balki et al. *Sample-Size Determination Methodologies for Machine Learning in Medical Imaging Research: A Systematic Review*. Nov. 2019. DOI: 10.1016/j.carj.2019.06.002.

[3] Rosa L Figueroa and Qing Zeng-treitler. "Predicting sample size required for classification performance". In: *BMC medical informatics and decision making* 12.8 (2012). DOI: 10.1186/1472-6947-12-8.

[4] S. R. Jones, S. Carley, and M. Harrison. "An introduction to power and sample size estimation". In: *Emergency Medicine Journal* 20.5 (2003), pp. 453–458. DOI: 10.1136/emj.20.5.453.

[5] Nancy M. Fenn Buderer. "Statistical Methodology : I . Incorporating the Prevalence of Disease into the Sample Size Calculation for Sensitivity and Specificity". In: *Academic Emergency Medicine* 3 (1996), pp. 895–900. DOI: 10.1111/j.1553-2712.1996.tb03538.x.

[6] Mohamad Adam Bujang and Tassha Hilda Adnan. "Requirements for Minimum Sample Size for Sensitivity and Specificity Analysis". In: *Journal of Clinical and Diagnostic Research* 10.10 (2016), YE01–YE06. DOI: 10.7860/JCDR/2016/18129.8744.

[7] Yu Guo et al. "Sample size and statistical power considerations in high-dimensionality data settings : a comparative study of classification algorithms". In: *BMC Bioinformatics* 11.447 (2010). DOI: 10.1186/1471-2105-11-447.

[8] Tarald O. Kvålseth. "Note on Cohen's Kappa". In: *Psychological Reports* 65.1 (1989), pp. 223–226. ISSN: 0033-2941. DOI: 10.2466/pr0.1989.65.1.223.

[9] Alexandre Lacoste et al. "Quantifying the Carbon Emissions of Machine Learning". In: *arXiv* 1910.09700 (2019).

[10] Tjeerd Van Der Ploeg, Peter C Austin, and Ewout W Steyerberg. "Modern modelling techniques are data hungry : a simulation study for predicting dichotomous endpoints". In: *BMC Med Res Methodol* 14.137 (2014). DOI: 10.1186/1471-2288-14-137.

[11] Jing Luan et al. "The predictive performances of random forest models with limited sample size and different species traits". In: *Fisheries Research* 227.February (2020), p. 105534. ISSN: 01657836. DOI: 10.1016/j.fishres.2020.105534. URL: https://doi.org/10.1016/j.fishres.2020.105534.

[12] John A Swets. "Measuring the Accuracy of Diagnostic Systems". In: *Science* 240.45857 (1988), pp. 1285–1293. DOI: 10.1126/science.3287615.

[13] Andrew P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern Recognition* 30.7 (1997), pp. 1145–1159. DOI: 10.1016/S0031-3203(96)00142-2.

[14] Claudia Perlich, Foster Provost, and Jeffrey S Simonoff. "Tree Induction vs. Logistic Regression: A Learning-Curve Analysis". In: *Journal of Machine Learning Research* 4 (2003), pp. 211–255.

[15] Felipe Campelo and Fernanda Takahashi. "Sample size estimation for power and accuracy in the experimental comparison of algorithms". In: *Journal of Heuristics* 25.2 (2019), pp. 305–338. DOI: 10.1007/s10732-018-9396-7.

[16] Felipe Campelo and Elizabeth F. Wanner. "Sample size calculations for the experimental comparison of multiple algorithms on multiple problem instances". In: *Journal of Heuristics* 26.6 (2020), pp. 851–883. DOI: 10.1007/s10732-020-09454-w.

[17] Xavier Robin et al. "pROC: An open-source package for R and S+ to analyze and compare ROC curves". In: *BMC Bioinformatics* 12 (Mar. 2011). ISSN: 14712105. DOI: 10.1186/1471-2105-12-77.

[18] R Core Team. *R: A Language and Environment for Statistical Computing*. 2022. URL: https://www.R-project.org/.

[19] Max Kuhn. "Building Predictive Models in R Using the caret Package". In: *Journal of Statistical Software* 28.5 (2008), pp. 1–26. DOI: 10.18637/jss.v028.i05.

[20] Corporation Microsoft and Steve Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. Jan. 2022. URL: https://CRAN.R-project.org/package=doParallel.

[21] Corporation Microsoft and Steve Weston. *foreach: Provides Foreach Looping Construct*. 2022. URL: https://CRAN.R-project.org/package=foreach.

[22] Beata Nowok, Gillian M. Raab, and Chris Dibben. "Synthpop: Bespoke creation of synthetic data in R". In: *Journal of Statistical Software* 74 (Oct. 2016). ISSN: 15487660. DOI: 10.18637/jss.v074.i11.

[23] Gareth James et al. *An Introduction to Statistical Learning with Applications in R*. New York: Springer-Verlag, 2013.

[24] R A Becker, J M Chambers, and A R Wilks. *The New S Language*. Wadsworth & Brooks/Cole, 1988.

[25] D. Dua and C Graff. *UCI Machine Learning Repository*. Irvine, CA, 2017. URL: http://archive.ics.uci.edu/ml.

[26] Evelyn Fix and J. L. Hodges. “Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties”. In: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), pp. 238–247. DOI: 10.2307/1403797.

[27] D.R Cox. “The Regression Analysis of Binary Sequences”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 20 (1958), pp. 215–232.

[28] David J Hand and Kerning Yu. “Idiot’s Bayes-Not So Stupid After All?” In: *International Statistical Review* 69 (2001), pp. 385–398. DOI: 10.1111/j.1751-5823.2001.tb00465.x.

[29] Tin Kam Ho. “Random decision forests”. In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. IEEE Computer Society, 1995, pp. 278–282. DOI: 10.1109/ICDAR.1995.598994.