ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Econometrics & Machine Learning

Amal Elfassihi
Nicolas Khadivi
Bastien van Delft

March 3, 2017

Contents

1 Introduction

1.1 Motivations

The aim of this project was to investigate how Machine Learning algorithms (e.g. decision trees, random forests, boosting) compared to more traditional econometric tools such as linear or logistic regressions, using a real-world dataset [1] to study a binary variable with these various algorithms.

Our objective was twofold:

1. compare prediction quality between the various algorithms
2. compare interpretability of the models and recover marginal effects if possible

The project was supervised by Dr. Romain Aeberhardt and Thomas Larrieu.

1.2 Structure

The remainder of this report is laid out as follows:

- **Part 2** provides a theoretical overview of usual Econometric and Machine Learning methods;
- **Part 3** gives a short overview of our experimental setup;
- **Part 4** focuses on the experiments, the results that we find and their interpretation;
- **Part 5** contains concluding remarks about our project, as well as ideas for future work.

2 Background

2.1 Econometrics

This section relies heavily on the theoretical background provided by [2] and [5].

2.1.1 Linear Regression

Given n distinct *explanatory variables* (also known as *regressors*, *predictors* or *independent variables*), the **multiple linear regression model** takes the form:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n + \epsilon \tag{2.1}$$

where x_i is the ith predictor and β_i quantifies the association between that variable and the response. We interpret β_i as the *average effect* of a one-unit increase in x_i on y, holding all other predictors fixed.

The regression coefficients $\beta_0, \beta_1, ..., \beta_n$ are unknown, and must be estimated from the data. This is usually done through a *least-squares* approach (although other fitting methods exists), where the estimates $\hat{\beta_0}, \hat{\beta_1}, ..., \hat{\beta_n}$ are chosen to minimise the sum of squared residuals:

$$\sum_{i=1}^{n} (y_i - x_i^T b)^2 \tag{2.2}$$

Given estimates $\hat{\beta_0}, \hat{\beta_1}, ..., \hat{\beta_n}$, it is then straightforward to predict the response $\hat{y}$ on the basis of a set of values for the predictors $x_1, x_2, ..., x_n$, using the *fitted value*:

$$\hat{y} = \hat{\beta_0} + \hat{\beta_1} x_1 + \hat{\beta_2} x_2 + \; + \hat{\beta_n} x_n \tag{2.3}$$

However, there are three types of uncertainties associated with this prediction:

1. The coefficient estimates $\hat{\beta_0}, \hat{\beta_1}, ..., \hat{\beta_n}$ are estimates of the true values $\beta_0, \beta_1, ..., \beta_n$. That is, the least squares plane $\hat{y}$ is only an estimate for the true population regression plane $f(x) = \beta_0 + \beta_1 x_1 + \; + \beta_n x_n$. The inaccuracy in the coefficient estimates is related to the *reducible error*. We can compute a confidence interval in order to determine how close $\hat{y}$ will be to y.
2. In practice, assuming a linear model for y is almost always an approximation of reality, so there is an additional source of potentially reducible error which we call *model bias*. When we use a linear model, we are in fact estimating the best linear approximation to the true surface. However, here we will ignore this discrepancy, and operate as if the linear model were correct.
3. Even if we knew y — that is, even if we knew the true values for $\beta_0, \beta_1, ..., \beta_n$ — the response value cannot be predicted perfectly because of the random error ϵ.

We use *prediction intervals* to measure how much $\hat{y}$ varies from y. Prediction intervals are always wider than confidence intervals, because they incorporate both the error in the estimate for y (the *reducible error*) and the uncertainty as to how much an individual point will differ from the population regression plane (the *irreducible error*).

2.1.2 Classifiers: Logit & Probit Models

The linear regression model assumes that the response variable y is purely *quantitative*, but in many situations, the response variable is in fact *qualitative* (or *categorical*). For example, eye color takes on values such as blue, brown or green. Predicting a qualitative response for an observation is also sometimes referred as *classifying* that observation, since the observation is assigned to a category.

Two of the most widely-used classifiers are the **logistic** (or *logit*) and **probit** regressions. In contrast with the linear model, they are used to regress for the *probability* of a categorical outcome (e.g. $y = 0$ or $y = 1$).

Recall that for the linear regression model, we have:

$$y = x'\beta + \epsilon \tag{2.4}$$

Binary outcome models instead estimate the *probability* that $y = 1$ as a function of the independent variables:

$$p = P(y = 1|x) = F(x'\beta) \tag{2.5}$$

Since the probit and logit models are estimated using the maximum likelihood method, an increase in x will increase (decrease) the *likelihood* that $y = 1$. In other words, an increase in x makes the outcome of 1 more or less likely.

The distinction between logit and probit comes from the functional form (or *link function*) of F that each methods uses.

- **Logit**: F is the cdf of the logistic distribution

$$F(x'\beta) = \Lambda(x'\beta) = \frac{e^{x'\beta}}{1 + e^{x'\beta}}$$

- **Probit**: F is the cdf of the standard normal distribution

$$F(x'\beta) = \Phi(x'\beta) = \int_{-\infty}^{+\infty} \phi(z)dz$$

where $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$

Note that since both link functions are cumulative distribution functions of probability distributions, the predicted values are limited between 0 and 1.

Marginal Effects

The marginal effect (i.e. the effect of one-unit increase in x_i on y, *ceteris paribus*) can be obtained by simply taking the partial derivative of y with respect to the variable of interest x_i:

$$EM(x_i) = \frac{\partial y}{\partial x_i} \tag{2.6}$$

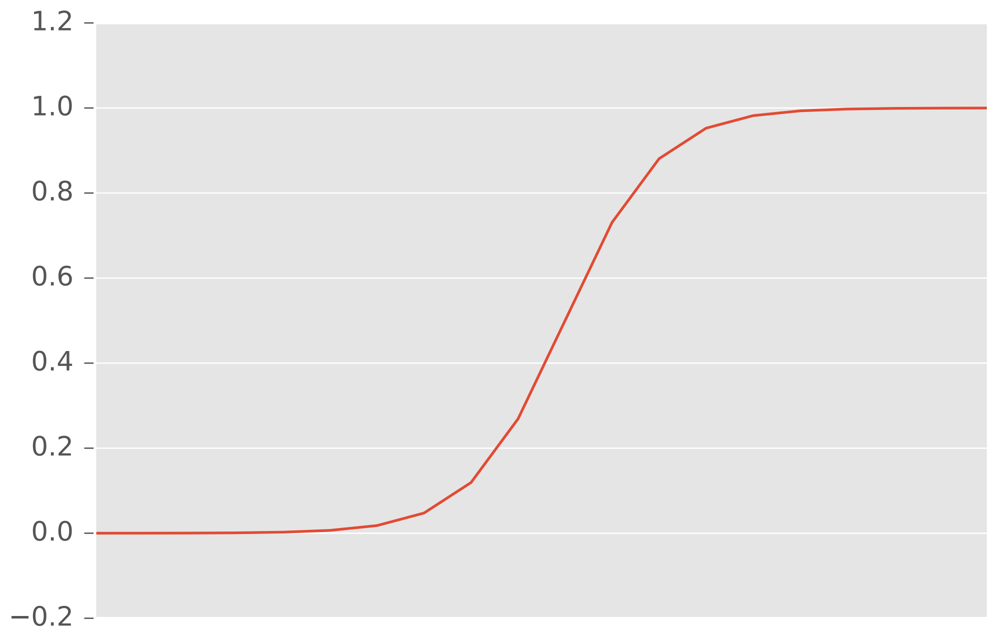


Figure 2.1: The `logit` function

With the linear model, we trivially obtain β_1 as the marginal effect of x_1, that is, the marginal effects are simply the coefficients and they do not depend on x.

For the logit and probit models, the marginal effects are slighly more complicated because of the non-linearity introduced by the link function:

$$EM(x_i) = \frac{\partial \mathbf{E}[y|x]}{\partial x_i} = \beta_i F'(x'\beta) \tag{2.7}$$

The marginal effects thus depend on $x = (x_1, ..., x_n)$, so we need to estimate them at a specific value of x, which is typically chosen to be the mean value - in other words, the vector of characteristics of the "average observation".

Thus, for the logit model, we have that:

$$EM(x_i) = \beta_i \Lambda(x'\beta)(1 - \Lambda(x'\beta)) = \beta_i \frac{e^{x'\beta}}{(1 + e^{x'\beta})^2}$$

and for the probit model:

$$EM(x_i) = \beta_i \Phi'(x'\beta) = \beta_i \phi(x'\beta)$$

where $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$, the pdf of the standard normal distribution.

Interpretation

- An increase in x increases (decreases) the probability that $y = 1$ by the marginal effect expressed as a percentage
  - For dummy independent variables, the marginal effect is expressed in comparison to the *base category* ($x = 0$).
  - For continuous independent variables, the marginal effect is expressed for a one-unit change in x.

- We interpret both the sign and the magnitude of the marginal effects.
- The probit and logit models produce almost identical marginal effects (up to a constant).

Coefficients

A nice property of the linear regression model is that *the marginal effects are the coefficients*, which means that we can evaluate the marginal effect of each regressor x_i simply by looking at the summary output from the regression.

This does not hold for logit and probit models, because of the presence of the link function, as discussed in the previous section. However, we can devise a simple "rule-of-thumb" to approximate the marginal effect from the coefficients. Recall that for logit and probit models:

$$EM(x_i) = \frac{\partial \mathbf{E}[y|x]}{\partial x_i} = \beta_i F'(x'\beta) \tag{2.8}$$

Now, using the maximum value of the F' function, we obtain an upper-bound approximation for the marginal effect directly from the coefficients in the summary table:

$$EM(x_i) \approx \beta_i \max F'(x'\beta) \tag{2.9}$$

For the logit model, $\max \Lambda'(x) = \Lambda'(0) = \frac{1}{4}$ and for the probit model, $\max \phi(x) = \phi(0) = \frac{1}{\sqrt{2\pi}} = \frac{1}{2.5}$ — thus, our simple "rule-of-thumb" for finding marginal effects can be summarised as follows:

- **Logit**: $EM(x_i) \approx \beta_i/4$
- **Probit**: $EM(x_i) \approx \beta_i/2.5$

2.2 Machine Learning

This section relies heavily on the theoretical background provided by [5], and applications in Python found in [3].

2.2.1 Decision Trees

Simple decision trees for regression and classification involve segmenting the predictor space into a number of simple regions. Then, in order to make a prediction for a given observation, we typically use the mean or the mode of the training observations in the region to which it belongs. Since the set of splits used to segment the space can be summarised in a tree, these approaches are known as *decision tree methods*.

Regression trees

The process of building a regression tree typically involves two steps:

1. First, we divide the predictor space – that is, the set of possible values for $x_1, x_2, ..., x_n$ – into J distinct and non-overlapping regions $R_1, R_2, ..., R_J$.
2. For every observation that falls into the region R_j, we predict the response value as the mean of the response values for the training observations in R_j.

How are the regions $R_1, ..., R_J$ constructed? In theory, the regions $R_1, ..., R_J$ could have any shape, but in practice, the predictor space is typically divided into high-dimensional rectangles, or *boxes* for simplicity and for ease of interpretation of the resulting model. The goal is then to find boxes $R_1, ..., R_J$ that minimise the RSS, given by:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{2.10}$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the j^{th} box.

While this method may produce good predictions on the training set, it is likely to overfit the data because the resulting tree might be too complex. A smaller tree with fewer splits (that is, fewer regions $R_1, ..., R_J$) may lead to lower variance and better interpretation at the cost of a little bias.

There are two strategies to make the resulting tree simpler:

1. Build the tree only so long as the decrease in the RSS due to each split exceeds some threshold; this method is considered somewhat "short-sighted" as an early below-threshold split might be followed by a very good split, i.e. a split that leads to a large reduction in RSS;
2. Grow a very large tree T_0, and then prune it back in order to obtain a simpler subtree.

Classification trees

The distinction between regression and classification trees is very similar to the one between the linear and logit/probit regressions. As we have seen in the previous section, in a regression tree, the predicted response for an observation is given by the (mean) response of the training observations that belong to the same region.

In contrast, the classification tree predicts that each observation belongs to the *most commonly occuring class* among training observations in the region to which it belongs.

Within this setting, the RSS criterion cannot be used for making the binary splits, so the *classification error rate* (i.e. the fraction of the training observations in that region that do not belong to the most common class) can be used instead:

$$E = 1 - \max_k \hat{p}_{mk} \tag{2.11}$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the m^{th} region that are from the k_{th} class.

However, it turns out that classification error is not sufficiently sensitive for tree-growing, and in practice the *Gini index* is preferred:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{2.12}$$

The Gini index is a measure of total variance across the K classes, and takes on a small value if all of the $\hat{p}_{mk}$s are close to zero or one. As such, it is also sometimes referred to as a measure of *node purity*, where a small value indicates that a node mostly contains observations from a single class.

Advantages & drawbacks

- Trees are very easy to explain to people, even easier than linear regression.
- Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches ;
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

However, trees usually do not have the same level of accuracy as some regression and classification approaches (as we will see later in the *Results* section).

2.2.2 Bagging, random forests, boosting

Simple trees typically suffer from high variance — that is, the results of a decision tree fit on distinct data sets are likely to be very different. In constrast, methods such as linear regression tend to have low variance. A popular and substantial improvement over simple trees is the aggregation of many such trees, using methods like bagging, random forests, and boosting.

Bagging

Bootstrap aggregation, or *bagging*, is a procedure for reducing the variance of a statistical learning method. Recall that given a set of n independent observations Z_1, ..., Z_n, each with variance σ^2, the variance of the mean of Z of the observations is given by $\frac{\sigma^2}{n}$. In other words, averaging a set of observations reduces variance.

Hence, a natural way to reduce the variance and increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. In other words, we calculate $\hat{f}_1(x), \hat{f}_2(x), ..., \hat{f}_B(x)$ using B separate training sets, and average them in order to obtain a single low-variance statistical learning model, given by:

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(x)$$

In practice, we do not have access to multiple training sets, but we can use a procedure called *bootstrap* to generate such sets, by taking repeated samples from the (single) training data set. In this approach we generate B different bootstrapped training data sets. We then train our method on the b^{th} bootstrapped training set in order to get $\hat{f}^b(x)$, and finally average all the predictions, to obtain:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f^*}^b(x)$$

Bagging has been demonstrated to give impressive improvements in accuracy by combining together hundreds or even thousands of trees into a single procedure.

Random Forests

Random forests provide an improvement over bagging by way of a small tweak that "decorrelates" the trees. As in bagging, we build a number of decision trees on samples bootstrapped from the original dataset (i.e. created from random sampling with replacement). The only distinction between bagging and random forests is that the latter selects, at each candidate split, a random subset of the features.

In bagging, if some of the features are very strong predictors for the output, they will be selected in many of the splits. As a consequence, all of the bagged trees will look quite similar to each other and the predictions from the bagged trees will be highly correlated. In particular, this means that bagging will not lead to a reduction in variance over a simple tree.

Random forests overcome this problem by forcing every split to pick predictors from a random subset, thus decorrelating the trees and reducing the variance of the resulting trees.

Boosting

Boosting is similar to bagging, but instead of building each tree on top of a bootstrapped dataset (independently of other trees), the trees are grown sequentially, using information from previous trees.

Note that in boosting, unlike in bagging, the construction of each tree depends strongly on the trees that have already been grown. From a computational point of view, this means that it is impossible to calculate boosting-based predictions in parallel, thus considerably slowing down the computation.

Boosting has three tuning parameters:

1. The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
2. The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
3. The number d of splits in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split. In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable. More generally, d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

3 Experimental Setup

As our objective is to compare classical econometric methods to machine learning algorithms, we had to pick a real dataset with a binary variable of interest y. We also had to ensure that we had an adequate experimental setup that would allow us to implement the algorithms and experiment in a relatively flexible way, while also allowing us to record and analyse our results with relative ease.

3.1 Database

On the advice of our supervisors, we turned to the *Enquête emploi en continu 2015* database [1] from the INSEE website. *Enquête Emploi* is the canonical database for the statistical system of knowledge of employment and unemployment in France, and one of the only sources that implements the definition of unemployment as defined by the International Labor Office [6].

The survey contains 431,678 observations taken in 2015, and provides original data on more than 100 variables, including occupation, female and youth activity, hours of work, precarious employment and wages. As such, using this data, we can better understand the situation of the unemployed as well as changes in the situation with regard to work: change from school to work, retirement activity, changes in occupation.

After downloading the dataset, we performed some basic manipulations to make it easier to work with (conversion from `DBF` to `SQLite` format), and then proceeded to create a "custom" subset of the database, such that we don't need to load all the data for each experiment, but only the parameters we are interested in. When these parameters were *categorical*, we split them up into purely *binary* variables to make our analyses easier.

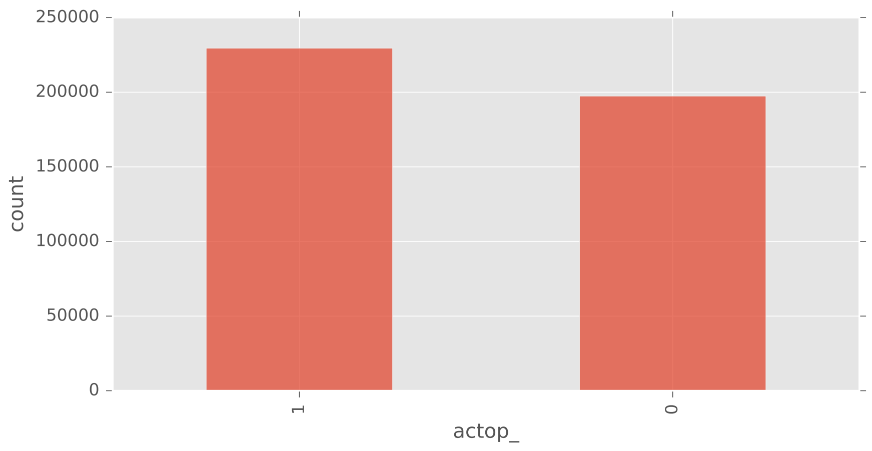


Figure 3.1: Number of active (`actop_=0`) and inactive (`actop_=1`) persons

3.2 Programming

Our setup for programming, running and analysing the results of our experiments revolved around various tools that we have put in place:

- The `Python` and `R` programming languages, along with the `scikit-learn` machine learning package [7][8][9]
- The `Git` source control manager [11], which enables us to keep track of different versions of our code (and thus project), while also letting us easily rollback changes, or run multiple versions of the code (e.g. with different parameters)
- `Jupyter` [10], an interactive computing platform, which enables us to run code directly in the browser, and mix text with code snippets to document our work properly

When our experiments reached the point where computation on our personal laptops was too slow/not possible anymore, we turned to Amazon Web Services [12], a cloud computing platform, which enabled us to provision very powerful machines almost immediately, and at a relatively small cost (~20$ in total).

3.3 Source

All of our work and results can be found on the project's GitHub repository[1] and in the Jupyter Notebook[2].

[1] `https://github.com/ncocacola/econml/`
[2] `https://github.com/ncocacola/econml/blob/master/econml.ipynb`

4 Experiments & Results

In this section, we provide an overview of some of the results that we found. We remind the reader that all the code and results of our experiments can be found in the Jupyter Notebook.

Throughout this section, the *reference group* is defined as:

- male;
- French;
- 50+ years old;
- no degree;
- no children;
- living in a rural area in metropolitan France.

4.1 Econometrics

We used the following classical econometric methods on the `eec15` database:

- Linear Regression
- Logistic Regression
- Probit Regression

For each of these methods, we used the same methodology: first, fit the model on `trim=1` data, then use the model to predict the probability that the `actop_` variable is equal to 1 (i.e. that the observed person is *unemployed*) for `trim=[2,3,4]` observations, and compare these results with the true values to assess the accuracy of the method.

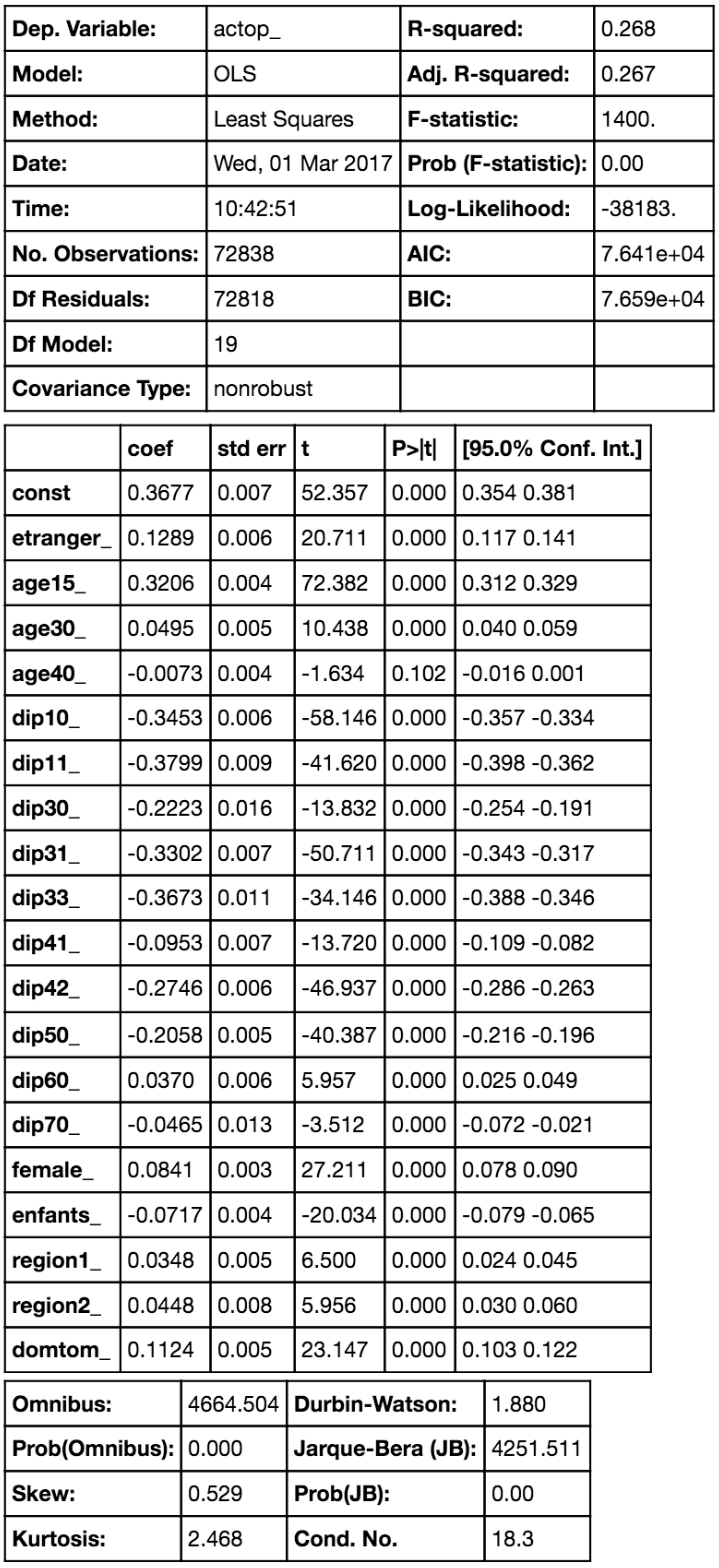
Due to the close proximity of the logit and probit methods, we will only discuss the results of the logit model. We recommend the interested reader to read through the code and results of the probit method online.

For the sake of clarity, for each model, we will first give the results, then provide an interpretation, focusing on the advantages and disadvantages of each method.

4.1.1 Linear Regression

As described in the *Background* section, linear regression is arguably the simplest method we can use. The results of the regression can be found in Figure 4.1.

Recall that when used to classify a binary variable, the result of a regression will be a *probability of success*, which we can then be used to conclude whether an individual is more or less likely to belong to a category or the other. The R-squared from this regression tells us that approximately 26% of the



Dep. Variable:	actop_	**R-squared:**	0.268
Model:	OLS	**Adj. R-squared:**	0.267
Method:	Least Squares	**F-statistic:**	1400.
Date:	Wed, 01 Mar 2017	**Prob (F-statistic):**	0.00
Time:	10:42:51	**Log-Likelihood:**	-38183.
No. Observations:	72838	**AIC:**	7.641e+04
Df Residuals:	72818	**BIC:**	7.659e+04
Df Model:	19		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
const	0.3677	0.007	52.357	0.000	0.354 0.381
etranger_	0.1289	0.006	20.711	0.000	0.117 0.141
age15_	0.3206	0.004	72.382	0.000	0.312 0.329
age30_	0.0495	0.005	10.438	0.000	0.040 0.059
age40_	-0.0073	0.004	-1.634	0.102	-0.016 0.001
dip10_	-0.3453	0.006	-58.146	0.000	-0.357 -0.334
dip11_	-0.3799	0.009	-41.620	0.000	-0.398 -0.362
dip30_	-0.2223	0.016	-13.832	0.000	-0.254 -0.191
dip31_	-0.3302	0.007	-50.711	0.000	-0.343 -0.317
dip33_	-0.3673	0.011	-34.146	0.000	-0.388 -0.346
dip41_	-0.0953	0.007	-13.720	0.000	-0.109 -0.082
dip42_	-0.2746	0.006	-46.937	0.000	-0.286 -0.263
dip50_	-0.2058	0.005	-40.387	0.000	-0.216 -0.196
dip60_	0.0370	0.006	5.957	0.000	0.025 0.049
dip70_	-0.0465	0.013	-3.512	0.000	-0.072 -0.021
female_	0.0841	0.003	27.211	0.000	0.078 0.090
enfants_	-0.0717	0.004	-20.034	0.000	-0.079 -0.065
region1_	0.0348	0.005	6.500	0.000	0.024 0.045
region2_	0.0448	0.008	5.956	0.000	0.030 0.060
domtom_	0.1124	0.005	23.147	0.000	0.103 0.122

Omnibus:	4664.504	**Durbin-Watson:**	1.880
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4251.511
Skew:	0.529	**Prob(JB):**	0.00
Kurtosis:	2.468	**Cond. No.**	18.3

Figure 4.1: Linear Regression Summary

explanation for an individual being active or not is explained by our set of independant variables. This number may seem low, but the regression is truly interesting if we look at the number/proportion of *true positives* (or *sensitivity*) and *true negatives* (or *specificity*):

	t1	t2	t3	t4
`actop_=0`	0.8805	0.8815	0.8802	0.8835
`actop_=1`	0.5455	0.5541	0.5322	0.5383

The method predicts people who are employed (`actop_=0`) very well, getting the right result approximately 88% of the time. In particular, since the proportion of employed is approximately 65% in our database, a uniform guess would be correct 65% of the time, so this method does in fact improve accuracy.

Also of note is the fact that all of our independent variables proved to be statistically signifiant, which is why we chose the same set of variables for all of our regressions.

Marginal Effects

We now turn to the information we were really looking for with this regression, namely *how* our predictions are built and *why* we obtain such results — we thus look at marginal effects. As we have mentioned before, the linear regression model has the property that the coefficients of the regression are the marginal effects (with no further transformation needed).

We can see from the summary table (in Figure 4.1) that the most important variables are `dip10_`, `dip11_`, `dip30_`, `dip31_` and `age15_`. In other words, the level/degree of education, and age are important parameters in determining whether someone is unemployed or not.

Since all of our regressors are binary (dummy) variables, the marginal effect is the increase/decrease in probability of being unemployed (`actop_=1`) relative to the reference group.

Limits

The limits of the linear regression model are somewhat obvious: it is good if the relation between variables is (approximately) linear, but if the underlying relationship is of any other kind, the model will be inaccurate. Another limit, in our particular case, is that we *may* obtain values superior > 1 or < 0, when we are in fact only dealing with binary values and hoping to get probabilities.

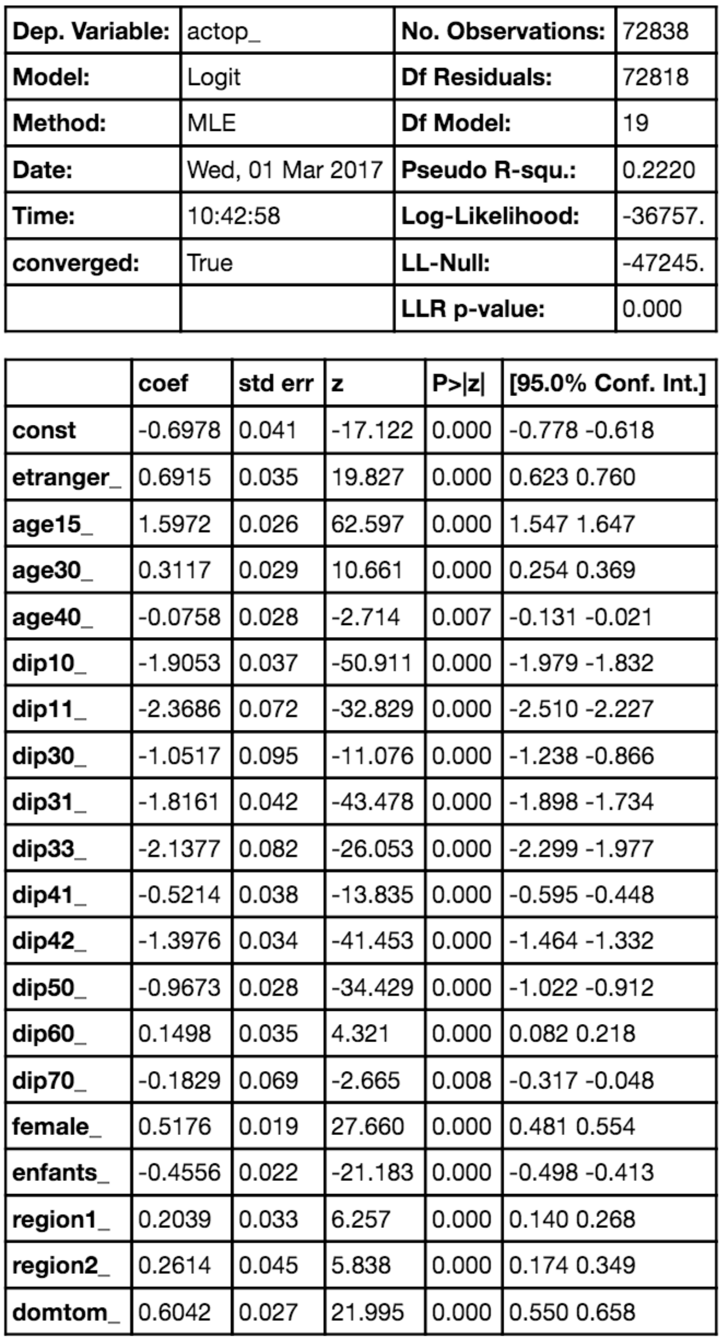
However, the main advantage that justifies the wide usage of linear regressions is the simplicity of interpretation. We can directly infer the causality relationships, compute statistical significance, and the R-squared to get an idea of how close we are to explaining the dependent variable. Another advantage is the low cost of computation, which makes this method both easy and fast to apply to large databases.

4.1.2 Logistic Regression

Logistic (or *logit*) regression is slightly more complicated, but also better suited to our problem of determining a probability for a binary response variable. We ran the logit regression on the same set of regressors as for the linear regression, and the summary table can be found in Figure 4.2.

Once again, we look at the accuracy of predicting $y = 1$ for people that are actually unemployed, and $y = 0$ for those that are employed.

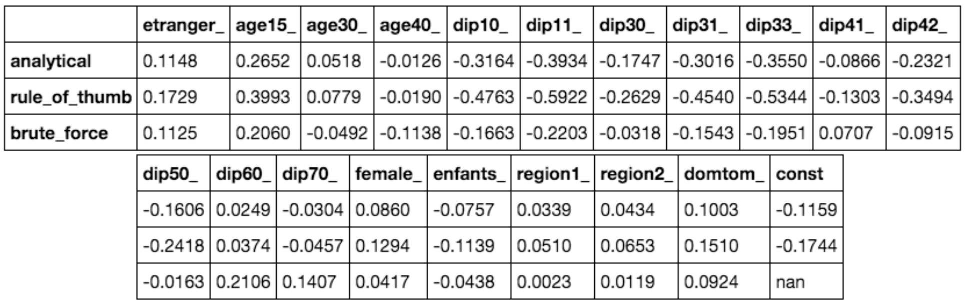
	t1	t2	t3	t4
`actop_=0`	0.8729	0.8737	0.8723	0.8757
`actop_=1`	0.5591	0.5672	0.5459	0.5524



Dep. Variable:	actop_	**No. Observations:**	72838
Model:	Logit	**Df Residuals:**	72818
Method:	MLE	**Df Model:**	19
Date:	Wed, 01 Mar 2017	**Pseudo R-squ.:**	0.2220
Time:	10:42:58	**Log-Likelihood:**	-36757.
converged:	True	**LL-Null:**	-47245.
		LLR p-value:	0.000

	coef	**std err**	**z**	**P>|z|**	**[95.0% Conf. Int.]**
const	-0.6978	0.041	-17.122	0.000	-0.778 -0.618
etranger_	0.6915	0.035	19.827	0.000	0.623 0.760
age15_	1.5972	0.026	62.597	0.000	1.547 1.647
age30_	0.3117	0.029	10.661	0.000	0.254 0.369
age40_	-0.0758	0.028	-2.714	0.007	-0.131 -0.021
dip10_	-1.9053	0.037	-50.911	0.000	-1.979 -1.832
dip11_	-2.3686	0.072	-32.829	0.000	-2.510 -2.227
dip30_	-1.0517	0.095	-11.076	0.000	-1.238 -0.866
dip31_	-1.8161	0.042	-43.478	0.000	-1.898 -1.734
dip33_	-2.1377	0.082	-26.053	0.000	-2.299 -1.977
dip41_	-0.5214	0.038	-13.835	0.000	-0.595 -0.448
dip42_	-1.3976	0.034	-41.453	0.000	-1.464 -1.332
dip50_	-0.9673	0.028	-34.429	0.000	-1.022 -0.912
dip60_	0.1498	0.035	4.321	0.000	0.082 0.218
dip70_	-0.1829	0.069	-2.665	0.008	-0.317 -0.048
female_	0.5176	0.019	27.660	0.000	0.481 0.554
enfants_	-0.4556	0.022	-21.183	0.000	-0.498 -0.413
region1_	0.2039	0.033	6.257	0.000	0.140 0.268
region2_	0.2614	0.045	5.838	0.000	0.174 0.349
domtom_	0.6042	0.027	21.995	0.000	0.550 0.658

Figure 4.2: Logit Regression Summary



	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_	dip50_	dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_	const
analytical	0.1148	0.2652	0.0518	-0.0126	-0.3164	-0.3934	-0.1747	-0.3016	-0.3550	-0.0866	-0.2321	-0.1606	0.0249	-0.0304	0.0860	-0.0757	0.0339	0.0434	0.1003	-0.1159
rule_of_thumb	0.1729	0.3993	0.0779	-0.0190	-0.4763	-0.5922	-0.2629	-0.4540	-0.5344	-0.1303	-0.3494	-0.2418	0.0374	-0.0457	0.1294	-0.1139	0.0510	0.0653	0.1510	-0.1744
brute_force	0.1125	0.2060	-0.0492	-0.1138	-0.1663	-0.2203	-0.0318	-0.1543	-0.1951	0.0707	-0.0915	-0.0163	0.2106	0.1407	0.0417	-0.0438	0.0023	0.0119	0.0924	nan

Figure 4.3: Logit — Marginal Effects

The prediction accuracy is very similar to what we get with a linear regression, in fact they are extremely close. Now, the question is how to interpret the results, and how to sense causality from this method.

Marginal Effects

While it is possible, mathematically, to derive the marginal effect of an independant variable (as we have seen in the *Background* section), we only have discrete (binary) regressors in our specific case. Thus, while the derivative gives us a sense of the marginal effect, this effect may be different in reality.

In addition to this, the marginal effect of a variable is linked to the value of other independent variables, so we focus on the *mean* marginal effects for each variable. More precisely, we look at the marginal effect of a given variable assuming that the other variables' are valued at their mean. The method we use to calculate this empirically (which we call "brute force") is as follows:

1. First, duplicate the original database and work on the new database
2. Pick a regressor (e.g. `dip11_`) and set it to 1 for all observations
3. Find the probability of $y = 1$ for all observations in both databases, using the model fitted to the original database.
4. For each observation, take the difference in $P(y = 1)$ between the two databases
5. Take the mean of these differences
6. Repeat this procedure for all regressors

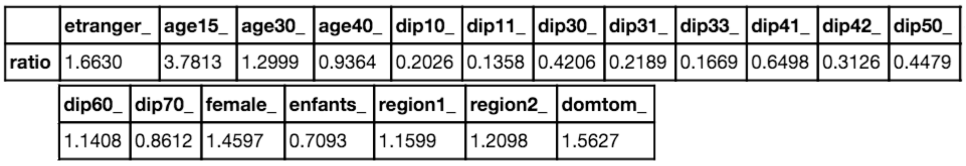
This method yields an empirical mean marginal effect for each of the independent variables.

Finally, we also wanted to get a sense of the accuracy of "rule-of-thumb" used to recover marginal effects from the coefficients of a logit regression (recall, this is $EM(x_1) \approx$ $beta_1^{logit}/4$. We find that this rule is not very accurate (the constant we divide by should be closer to 6).

The values we obtained mathematically (and, up to a constant, through the "rule-of-thumb") are different from the empirical ones, which we explain by the non-continuity of the independent variable. However, for independent variables where we have enough data, the values are close, or differ in magnitude but not in sign, which gives us confidence as to the quality of our marginal effects (see Figure 4.3), and enables us to order the variables according to their importance and thus, to sense what determines the prediction we make.

Odds Ratios

For the logit method, and in addition to the marginal effects, we also calculate the odds ratios to get an idea of the difference that having a given parameters or not makes on the response variable



	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_	dip50_
ratio	1.6630	3.7813	1.2999	0.9364	0.2026	0.1358	0.4206	0.2189	0.1669	0.6498	0.3126	0.4479

dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_
1.1408	0.8612	1.4597	0.7093	1.1599	1.2098	1.5627

Figure 4.4: Logit — Odds Ratios

(see Figure 4.4). This will be interesting to compare with odds ratios from the Machine Learning algorithms. We use the following odds ratios formula:

$$OR = \frac{P(y=1|x=1,X_{-1})/P(y=0|x=1,X_{-1})}{P(y=1|x=0,X_{-1})/P(y=0|x=0,X_{-1})} \tag{4.1}$$

An odds ratio close to 1 means that the variable doesnt add a lot of information. If it is close to zero, it means the variable strongly increases the probability that the individual works, and a ratio far beyond one indicates the opposite.

As a result we can conclude that being young makes it more likely that you are inactive, while having a higher degree of education makes it more likely that the person works.

Limits

The limits of logit exist by definition of the model. If the underlying relationship between variables is far from the logit curve, the model cannot predict accurately. In contrast, the model is significantly better than the linear model to predict binary/categorical variables — its main advantage is the combination of fair accuracy and simplicity.

This makes it cheap to apply to large databases, while also providing results that are easy to interpet and explain, even to people with no background in statistics — a great advantage that applies to regression models in general.

4.2 Machine Learning: tree-based methods

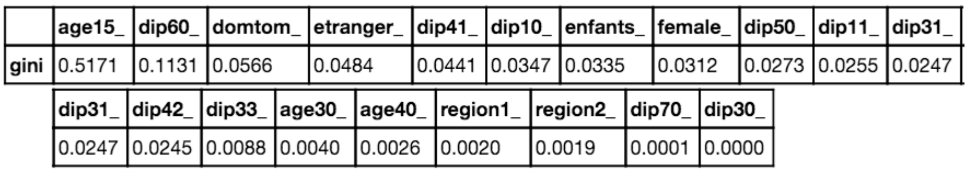
We now turn out attention to *tree-based methods*, completely different types of statistical methods, both in their setting and the way they operate. We use three major algorithms:

- (Simple) Decision Tree
- Random Forests
- Boosting

As for the regressions, we split our database into trimesters, and first fit the models on `trim=1` data, before using the fitted model for predictions on `trim=[2,3,4]` observations.

4.2.1 Simple Tree

Following the exact setting we have for regressions (i.e. keeping all variables binary), the initial results we obtain for *accuracy* seem to suggest that the simple tree performs as well but no better than linear or logit regressions, with an average accuracy of 76%, very close to what we had before.



	age15_	dip60_	domtom_	etranger_	dip41_	dip10_	enfants_	female_	dip50_	dip11_	dip31_
gini	0.5171	0.1131	0.0566	0.0484	0.0441	0.0347	0.0335	0.0312	0.0273	0.0255	0.0247

dip31_	dip42_	dip33_	age30_	age40_	region1_	region2_	dip70_	dip30_
0.0247	0.0245	0.0088	0.0040	0.0026	0.0020	0.0019	0.0001	0.0000

Figure 4.5: Simple Tree — Gini Feature Importance

Behind this result, however, lies a very sharp difference in terms of *prediction*. The simple tree model accurately predicts 91.6% of true positives, but only 47.5% of true negatives (recall that, rather confusingly, `actop_=1` should be interpreted as *unemployed*). Therefore, in spite of its apparent simplicity, the simple tree is so far the best model if we want to capture active people.

	t1	t2	t3	t4
`actop_=0`	0.9160	0.9170	0.9169	0.9200
`actop_=1`	0.4898	0.4997	0.4722	0.4803

Gini Index

Now, the question we are really trying to answer relates to the marginal effect of each variable, and more generally, to the *relative importance* of each variable. One method which is broadly used for measuring the variables importances is the *Gini index gain per variable*. Recall (from the *Background* section) that the Gini index is defined by:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{4.2}$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the m^{th} region that are from the k_{th} class.

The Gini index is a measure of the total variance accross the K classes (in our case $K = 2$), so in a sense, Gini is an indication of *node purity*. The Gini index takes on small values whenever the probabilities are close to zero or one, that is if the node is *pure*. At each split, we use the variable that maximises the gain in terms of Gini index.

The way to assess a feature's importance is then made easy and relatively straightforward: we simply look at the total amount that the Gini is decreased by due to splits over a given feature (see Figure 4.5).

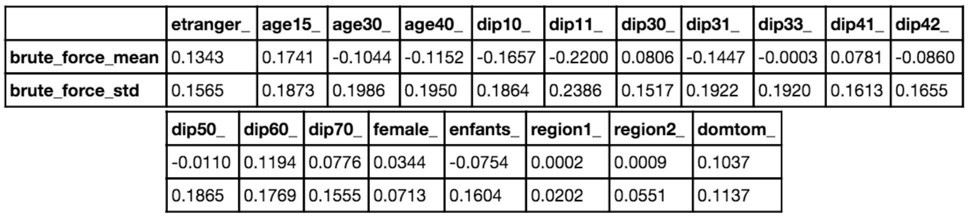
This value gives us an idea of features' importance, and it is also widely used with random forests and boosting methods, because it is easy and computed directly when running the algorithms (that is, without the need to perform more calculations),

Marginal Effects

Now that we have an idea of the relative importance of each feature/variable, we would like to measure a more accurate and specific measure of a marginal effect, i.e. what is the effect of a one-unit change in a given variable on the output?

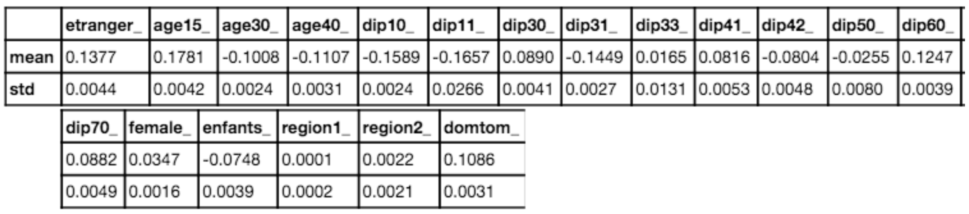
While regressions allowed us to calculate this effect analytically, this is not possible with tree-based methods, so we replicate the "brute force" approach: duplicate the database and modify one parameter from 0 to 1 for everybody, use the fitted model to predict the probabilities for both databases, and take the difference.

We thus obtain a marginal effect per individual and then take the mean marginal effect over all individuals. This gives a good approximation of the marginal effect for the average individual (see



	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_	dip50_	dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_
brute_force_mean	0.1343	0.1741	-0.1044	-0.1152	-0.1657	-0.2200	0.0806	-0.1447	-0.0003	0.0781	-0.0860	-0.0110	0.1194	0.0776	0.0344	-0.0754	0.0002	0.0009	0.1037
brute_force_std	0.1565	0.1873	0.1986	0.1950	0.1864	0.2386	0.1517	0.1922	0.1920	0.1613	0.1655	0.1865	0.1769	0.1555	0.0713	0.1604	0.0202	0.0551	0.1137

Figure 4.6: Simple Tree — "Brute Force" marginal effects



	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_	dip50_	dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_
mean	0.1377	0.1781	-0.1008	-0.1107	-0.1589	-0.1657	0.0890	-0.1449	0.0165	0.0816	-0.0804	-0.0255	0.1247	0.0882	0.0347	-0.0748	0.0001	0.0022	0.1086
std	0.0044	0.0042	0.0024	0.0031	0.0024	0.0266	0.0041	0.0027	0.0131	0.0053	0.0048	0.0080	0.0039	0.0049	0.0016	0.0039	0.0002	0.0021	0.0031

Figure 4.7: Simple Tree — 95% confidence interval for mean marginal effects ("Bootstrap")

Figure 4.6). However, note that the marginal effect of one variable is still dependent on the values of the other parameters.

We observe here that that the parameters with high Gini importance do have a fairly high marginal effect but it is impossible to infer the marginal effect of a change in one variable based only on Gini importance. Gini importance provides a measure of *relative importance* but it is not directly linked with marginal effects.

With the "brute force" method, we find a mean value and a standard deviation. This standard deviation measures the dispersion of the marginal effect across individuals, and is in fact not the dispersion we are looking for. What we would like is to build a 95% confidence interval for the value of the mean marginal effect.

In order to find this, we use the *bootstrap* method, which consists in the following procedure repeated n times (see results in Figure 4.7):

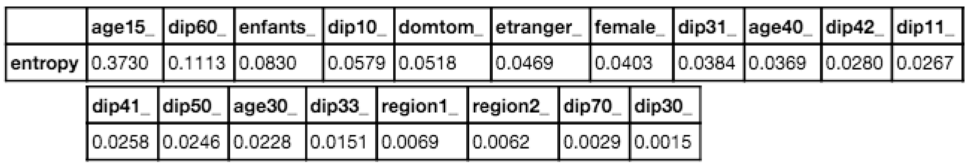
1. Create a "new" dataset by picking observations from the original dataset *with replacement*
2. Compute the mean marginal effect for each variable in the new dataset
3. Construct the interval by excluding the 2.5% smallest/largest values

We used $n = 1000$ for simple trees, but for later methods (e.g. boosting), we had to reduce this value to limit computation time.

The results we obtained for a simple tree are relatively satisfying — it is easy to infer a sense of causality in the model.

At the same time, it is clear that the simple tree is limited. The quality of the predictions are not any better in our case and don't justify the computation time spent calculating the marginal effects. However, simple trees classify the individual in a very different way than regression methods and even though it is a relatively simple method, it will still fit better to certain classes of problems. As a result, growing a tree or two to see how they fit the data should be done most of the time.

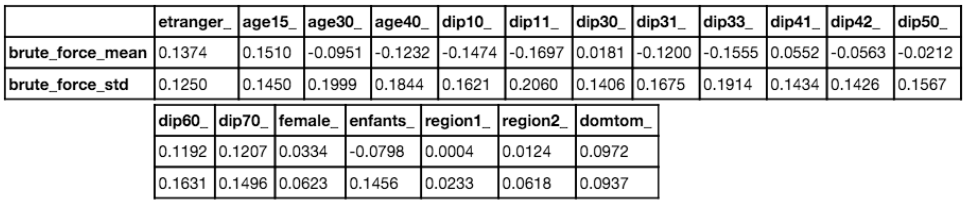
In order to exploit the nature of the tree better, we also tried to grow a tree directly using categorical variables (as opposed to converting all variables to a set of binary variables), as long as the variables could be ordered. We did obtain slightly better result on average (77.5% accuracy) but the most striking part was that we predicted the people who are active (i.e. `actop_=0`) even better, 93% of the time.



	age15_	dip60_	enfants_	dip10_	domtom_	etranger_	female_	dip31_	age40_	dip42_	dip11_
entropy	0.3730	0.1113	0.0830	0.0579	0.0518	0.0469	0.0403	0.0384	0.0369	0.0280	0.0267

dip41_	dip50_	age30_	dip33_	region1_	region2_	dip70_	dip30_
0.0258	0.0246	0.0228	0.0151	0.0069	0.0062	0.0029	0.0015

Figure 4.8: Random Forests — Gini Feature Importance



	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_	dip50_
brute_force_mean	0.1374	0.1510	-0.0951	-0.1232	-0.1474	-0.1697	0.0181	-0.1200	-0.1555	0.0552	-0.0563	-0.0212
brute_force_std	0.1250	0.1450	0.1999	0.1844	0.1621	0.2060	0.1406	0.1675	0.1914	0.1434	0.1426	0.1567

dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_
0.1192	0.1207	0.0334	-0.0798	0.0004	0.0124	0.0972
0.1631	0.1496	0.0623	0.1456	0.0233	0.0618	0.0937

Figure 4.9: Random Forests — "Brute Force" marginal effects

4.2.2 Random Forests

We will not expand here on how random forests work — we point the reader to the theory in *Background*. However, it is important to point out that the random forest algorithm is an aggregation of simple trees, and that it is expected to improve prediction quality by introducing some randomness.

In our very specific case, with only a few independent variables to choose from, all of which are binary or categorical, the prediction accuracy we get is rather disappointing when compared with the complexity of the algorithm: the overall accuracy we can reach is below 78%, a result similar to what we get with a simple tree.

However, using (the original) categorical variables for `age` and `diploma` instead of binary variables, and using a maximum depth of tree of 9 and a maximum feature per tree of 5, we managed to reach a 93.5% accuracy for prediction of active people, the best weve found so far. The cost of this improved accuracy is that we are below 50% for predicting inactive people (approximately 46% over the four quarters).

The results we obtain here are somewhat disappointing — nevertheless, we would like to emphasise that this we believe this to be mainly due to our setting. In general, random forests do give better predictions than simple trees, but as the algorithm is still based on simple trees, it is expected to conserve some its the main features.

Gini index

We used the exact same methods to calculate Gini importance as for simple trees. We expect to find some similarities between the Gini index with simple trees and random forests — at the very least, we expect the least important parameters in the simple tree to be more important in random forests due to the randomisation of the choice of independent variables (see Figure 4.8).

As we can see the main features are still the same even though the values vary a little bit. The gap between the most important and the least important variables is smaller, and a few small changes have occured but overall values are somewhat comparable.

Marginal Effects

We now take a closer at the real mean marginal effects, computed empirically. We expect to find some relationship between the results found for simple tree and and those we have now obtained.

	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_
lower	0.1258	0.1408	-0.0992	-0.1288	-0.1487	-0.1703	0.0037	-0.1271	-0.1696	0.0511	-0.0618
upper	0.1435	0.1539	-0.0898	-0.1201	-0.1345	-0.1408	0.0351	-0.1113	-0.1305	0.0638	-0.0465

dip50_	dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_
-0.0275	0.1119	0.1099	0.0322	-0.0897	-0.0005	0.0047	0.0966
-0.0157	0.1250	0.1471	0.0360	-0.0798	0.0009	0.0165	0.1091

Figure 4.10: Random Forests — 95% confidence interval for mean marginal effects ("Bootstrap")

	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_	dip50_
ratio	1.8528	2.9195	0.9969	0.8557	0.3612	0.3129	0.8248	0.4239	0.3432	0.9642	0.5895	0.6939

dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_
1.2507	1.2583	1.4018	0.6231	1.0663	1.1235	1.6085

Figure 4.11: Random Forests — Odds Ratios

As we can see here, the marginal effect of dip11_ (highest degree of education) is largest in absolute value, however dip11_ is not very important in terms of its Gini index, which means that it wasn't significant in terms of purity gain. One reason could actually be that because few people have dip11_, using it at a split at the very beginning does not produce a later gain in purity compared to other variable such as age15_ for instance. As a result dip11_ is only used as a very late split and so its purity gain may be regarded as somewhat negligible.

This actually emphasises the point we already made earlier about the simple tree. The Gini index is somewhat interesting for its simplicity but it only gives us an idea about relative marginal effects and it is only relevant when the dataset of people with the two features are approximately of the same size.

As for the mean marginal effect values we obtained, we can emphasise once again that the values are very different across parameters, both in terms of sign or magnitude. This means that it is somewhat easy to deduce a relevant sense of causality from these values, i.e. we can picture what feature is important in the prediction we make for a given individual.

In order to give some statistical foundations to our mean marginal effects, we also used the bootstrap method (with $n = 1000$ iterations) to calculate a 95% Confidence Interval (see Figure 4.10). Note that while this computation was relatively fast for simple trees (3.5 seconds per iteration), it was much slower for the random forest (4 minutes per iteration), due to the nature of the algorithm.

Odds Ratios

As we did for the logit method, we also calculate odds ratios (see Figure 4.11), using the following formula:

$$OR = \frac{P(y=1|x=1,X_{-1})/P(y=0|x=1,X_{-1})}{P(y=1|x=0,X_{-1})/P(y=0|x=0,X_{-1})} \tag{4.3}$$

Recall that an odds ratio close to 1 means that the variable doesnt add a lot of information. If it is close to zero, it means the variable strongly increases the probability that the individual works, and a ratio far beyond one indicates the opposite.

It is important to notice that in some cases, we did find a high mean marginal effect but in the particular case where we look at the reference individual the marginal effect is neglectable. This is the case for instance with `age30_` — which means that for an individual, being in his/her 30s is very similar in terms of prediction to being in his/her 50s (the reference variable).

Another interesting point is that odds ratios are easily compared between models. Unsurprisingly, we find that the ratios are very similar between the simple tree and the random forest, slightly less spread out in the case of the random forest. This makes sense when we think about the random forest algorithm, since a variable that may not be used much in a single tree may take more importance when it comes to the random forest due to the randomisation of predictors' selection.

We find that it is indeed possible to infer causality with the random forest algorithm, and to assess the importance of each predictor according to other predictor values. The main limits of this approach are the computation time and the amount of data that are required to find interesting results. However, assuming we have access to large datasets and very powerful machines, it would make sense to use random forests if the prediction it makes is more accurate.

Given that the functional form (e.g. logit) and tree-based approaches are very different, random forest will prove better than any regression in many cases, and it should not be ruled out because it lacks a (direct) sense of causality. Indeed, if it is needed and worth it, causality is computable.

4.2.3 Boosting

As our last Machine Learning algorithm pick, we opted for boosting — we used the AdaBoost algorithm available in `scikit-learn`, and took a simple tree as a base estimator.

The first aspect of boosting we would like to emphasise is that it is a relatively complex algorithm. Indeed, the fact that the computation of each new tree relies on the previous trees makes it impossible to parallelise fully, which means that computation time is very long. In contrast, with random forest, we can take advantage of large multi-core machines to speed up the computation, which is impossible with boosting.

The results we obtained for boosting are somewhat disappointing. The prediction is hardly any better than what we would obtain with a logistic regression. The general prediction accuracy is around 78%. Once again, the algorithm is very good at predicting 1s (~91,5%) but not very good at predicting 0s (~52%). As the boosting algorithm is also based on simple trees, it makes sense that we find once again, the main characteristics of the simple tree we grew earlier, especially considering we used the very same predictors.

However disappointing the results, in many cases, boosting algorithms outperform all the other models we have applied. The fact that this does not happen in this specific situation is not a contradiction. Our model is very specific with only binary variables and we do not possess enough informative variable to truly make a difference.

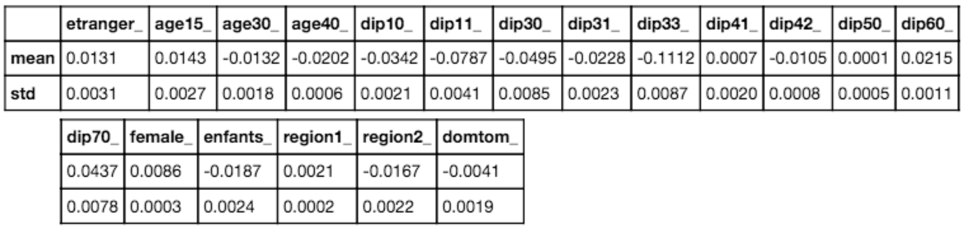
With that being said, it is still relevant to see (1) whether the marginal effects we obtained are relevant, (2) how can we make them statistically significant, and (3) how can we interpret and use them to infer the hidden causality relationships.

Marginal Effects

We have applied to this model, the very same methods as before in order to find out marginal effects per variable. The results can be found in Figure 4.12.

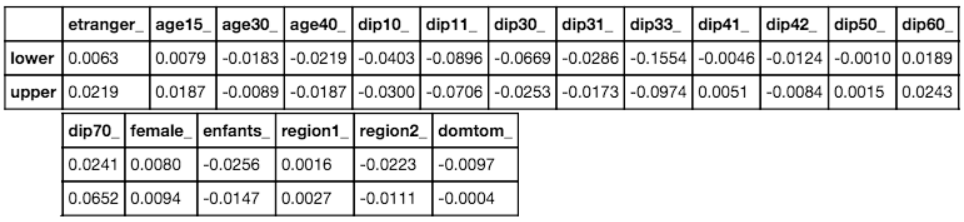
We first notice that the absolute values we obtain are very small, which means that that the marginal effects seem less spread towards extreme values. The dispersion around the mean is still very consequent but the means themselves are much smaller than what we obtained for random forests or simple trees for instance.

A plausible explanation for this can be found by looking more closely at the algorithm itself. Boosting grows trees recursively and based on previous trees' residuals — then, it makes sense to infer that



	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_	dip50_	dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_
mean	0.0131	0.0143	-0.0132	-0.0202	-0.0342	-0.0787	-0.0495	-0.0228	-0.1112	0.0007	-0.0105	0.0001	0.0215	0.0437	0.0086	-0.0187	0.0021	-0.0167	-0.0041
std	0.0031	0.0027	0.0018	0.0006	0.0021	0.0041	0.0085	0.0023	0.0087	0.0020	0.0008	0.0005	0.0011	0.0078	0.0003	0.0024	0.0002	0.0022	0.0019

Figure 4.12: Boosting — "Brute Force" marginal effects



	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_	dip50_	dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_
lower	0.0063	0.0079	-0.0183	-0.0219	-0.0403	-0.0896	-0.0669	-0.0286	-0.1554	-0.0046	-0.0124	-0.0010	0.0189	0.0241	0.0080	-0.0256	0.0016	-0.0223	-0.0097
upper	0.0219	0.0187	-0.0089	-0.0187	-0.0300	-0.0706	-0.0253	-0.0173	-0.0974	0.0051	-0.0084	0.0015	0.0243	0.0652	0.0094	-0.0147	0.0027	-0.0111	-0.0004

Figure 4.13: Boosting — 95% confidence interval for mean marginal effects ("Bootstrap")

more variables are involved in the prediction process and especially that the most important variables (in previous models) would lose some of their dominance. As a result, boosting seems to "even out" the impact of each variable.

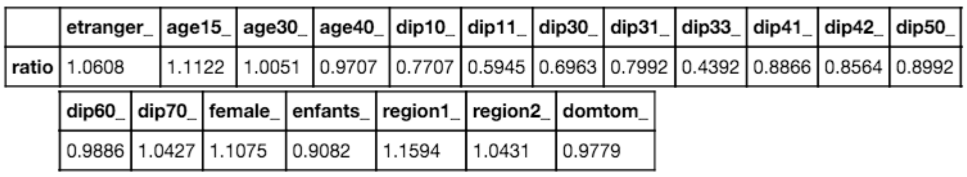
In order to ensure that the means we found are statistically significant, we also computed a 95% confidence interval using the bootstrap method. It is important to note however, that due to the nature and complexity of the boosting algorithm, we were not able to run more than $n = 500$ iterations, and that even by limiting the number of iterations, the computation took more than 30 hours in total and required very large amounts of memory.

This is a very important limit of the boosting model — because of the great amount of computation required, it is difficult to practice "trial-and-error" and thus, the boosting algorithm is less interesting than other methods.

Odds Ratios

Finally, we take a quick look at the odds ratios for the boosting model (Figure 4.14. As expected, we find that the odds ratios are as spread out as before, and this reinforces the general idea that marginal effects are less distinct than before and that they range within smaller absolute values, leading in turn to odds ratios that are closer to 1.

This is an interesting distinction between random forest and boosting, which seems to indicate that boosting trie to look at the "bigger picture" of an individual's characteristics, while random forests give more importance to the main characteristics.



	etranger_	age15_	age30_	age40_	dip10_	dip11_	dip30_	dip31_	dip33_	dip41_	dip42_	dip50_	dip60_	dip70_	female_	enfants_	region1_	region2_	domtom_
ratio	1.0608	1.1122	1.0051	0.9707	0.7707	0.5945	0.6963	0.7992	0.4392	0.8866	0.8564	0.8992	0.9886	1.0427	1.1075	0.9082	1.1594	1.0431	0.9779

Figure 4.14: Boosting — Odds Ratios

5 Conclusion

Using the *Enquête emploi en continu 2015* database [1], we studied the binary variable `actop_` and attempted to recover the main parameters that impact its value and their marginal effects, using various Econometric and Machine Learning methods.

Our results have not allowed us to draw any definitive conclusions, but enabled us to gain a lot of insight into the workings of the various methods, and their advantages and disadvantages:

Our findings can be summarised in the following points:

- No algorithm is better than others in every aspect — there are always situations in which one is better than others and vice-versa.
- Predictions can be accurate on different aspects (e.g. good at predicting 1s, but not 0s), which means that it is important to be specific about the objective that we pursue.
- Alongside their accuracy, models have different costs, in terms of computational power required, interpretability and simplicity. In general, it is better to follow Occam's razor and stick with the simplest method that provides adequate results.
- Marginal effects are always inferrable, albeit at vastly different costs and with different accuracies. In other works, it is always possible to get a sense of whats going on, through various methods.

5.1 Future Work

Due to various constraints (mainly time and computational power), we were only able to complete a subset of the experiments we have in mind. While our results provided us with some answers, they also opened up many other questions. It would be very interesting to pick a different database and/or set of parameters (in particular, continuous variables), and test some of the conclusions we made from our experiments to either validate or invalidate them, for example by shedding more light on how the various algorithms assign importance to the features.

At the same time, we must not forget that perfect prediction is metaphysically impossible, and that the more effort is put into finding an accurate model, the likelier it is that the complexity of the model makes it very hard to interpret. The key is thus to always keep in mind the tradeoff between accuracy and interpretability.

List of Figures

Bibliography

[1] INSEE, *Enquête Emploi en Continu*. 2015

[2] J. M. Wooldridge, *Introductory Econometrics: A Modern Approach*, CENGAGE Learning, 2015

[3] S. Raschka, *Python Machine Learning*, Packt Publishing, 2015

[4] AnalyticsVidhya.com, *Essentials of Machine Learning Algorithms (with Python and R Codes)*. 2015

[5] G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*. Springer, 2013

[6] International Labour Office.

[7] The Python Programming Language.

[8] The R Project.

[9] scikit-learn: machine learning in Python.

[10] Jupyter Notebook.

[11] Git SCM.

[12] Amazon Web Services