Universidad de San Andrés

Departamento de Economía

Maestría en Economía

An Approach to Train Machine Leaning Algorithms

Diego Hernán Álvarez

31.662.212

Mentor: Manuel Maurette

Buenos Aires

18 de Noviembre, 2020

Tesis de Maestría en Economía de

Diego Hernán Álvarez

"Un Enfoque para Entrenar Algoritmos de Aprendizaje"

Resumen

La Minería de Datos ha evolucionado y la tendencia se mueve hacia el entrenamiento de algoritmos de aprendizaje cada vez más complejos. Dichos algoritmos son utilizados en distintos campos para tomar decisiones importantes. Por ejemplo, en el mundo de la economía y las finanzas se entrenan modelos de puntuación de créditos, detección de fraudes, campañas de marketing, selección de candidatos en los trabajos, etc. Sin embargo, la elección del algoritmo de aprendizaje no necesariamente tiene que estar sesgada hacia lo complejo y sofisticado. Antes de entrenar un algoritmo de aprendizaje debería diseñarse un plan de trabajo. Una posible estrategia podría ser entrenar un modelo simple e interpretable como base, y luego entrenar algoritmos más sofisticados y testear cual es la mejora en el desempeño de estos últimos con respecto al modelo base. El objetivo de mi tesis es entrenar una regresión logística como modelo base y comparar su desempeño con los modelos de ensamble de árboles y testear si siempre vale la pena entrenar algoritmos sofisticados.

Palabras clave: Sofisticación, Interpretación, Balance, Desafío, Plan de trabajo

"An Approach to Train Machine Learning Algorithms"

Abstract
With the evolution of data mining the trend is to train more complex algorithms. Those sophisticated machine learning algorithms are being used in several fields to take important decisions. Examples in economics include credit scoring models, fraud detection, marketing campaigns, job applications, etc. However, modeling approach choices should not be biased towards complex machine learning algorithms necessarily. Before training machine learning algorithms a working plan should be designed. A strategy could be training a baseline, simple, and interpretable model and, then, rely on more complex ones to ascertain the extent of

Universidad de SanAndrés

performance improvements. The purpose of this thesis is to train a logistic regression as a baseline model and challenge it with tree-based ensemble machine learning algorithms to test how much those more complex models improve performance and determine whether it is always worth training complex machine learning algorithms.

Keywords: Sophistication, Interpretability, Balance, Challenge, Working plan

Códigos JEL: C49, C52, C55

QUAERERE VERUM
Universidad de SanAndrés

I would like to thank Professor Enrique Kawamura. He always supported me, provided me with opportunities, encouraged me to write this thesis, and trusted myself with faith and affection.

I would also like to thank Manuel Maurette and Yanina Gimenez. I could have not faced this challenge without them. Yanina has supported me with her knowledge and expertise during the whole process.

Universidad de SanAndrés

Table of Contents

QUAERERE VERUM
Universidad de
SanAndrés

1.0 – Introduction

Supervised classification machine learning algorithms have evolved from simple and interpretable models to more sophisticated models that better represent complex relationships between the features and the target variable. However, modeling approach choices should not be biased towards complex machine learning algorithms necessarily. There are many aspects to choose a modeling approach. First, the purpose of the model should be clearly stated before training a machine learning algorithm. For example, is the aim of the model to be as accurate as possible even at the expense of simplicity and interpretability? Is the purpose of the model to provide recommendations as well as their justifications? Those questions can also lead to other important ones: is a linear relationship expected between the features and the target variable? Is training time an issue to be considered? If training a complex model is time-consuming, but a simpler and more interpretable one such as a logistic regression performs slightly worse than the complex model, should the complex model still be chosen?

All these questions help design a working plan before training machine learning algorithms. A useful approach could be to start by training a baseline simple and interpretable model. The baseline model should not be time-consuming within the data preprocessing field. In addition, an empirical analysis of the dataset can lead to an expectation about the existence of linear relationships between the features and the target variable. Then, depending on the purpose of the model, the performance of its baseline, and the existence of training time constraints; the baseline model could be challenged by training more complex models. The purpose of this thesis is to train a logistic regression as a baseline model and challenge it with tree-based ensemble machine learning algorithms to ascertain the performance improvements and to answer the question if it is always worth training complex machine learning algorithms. I trained a set of models on the United States' Census Income dataset (Kohav and Becker 1996), which consists of 48,842 instances and whose target is to predict whether a person earns more than $50 thousand per year. Before fitting the models, I clearly stated my purpose: I want to find the best classification model, while maintaining a balance between training and interpretability efforts.

The performance of a machine learning algorithm must be evaluated considering the 'bias-variance trade-off'. On the one hand, a learning method presents variance issues when it is too specialized on the data it has been trained and fails to predict new observations. In this case, the algorithm overfits the training data. On the other hand, a learning method has a bias problem when the modeling approach is not flexible enough to match the patterns in the data. For example, a Logistic Regression model has a bias issue when fitting data that has non-linear patterns even though its predictions are more stable on new observations. The 'bias-variance trade-off' implies that a machine learning algorithm with a bias or variance issue should be treated carefully to prevent solving one problem at the expense of the other.

I have chosen a logistic regression as a baseline model because its training is not excessively time-consuming. Additionally, it is not a black-box model, since the estimated coefficients explain the drivers of the probability. If the challenger models improve the performance of the logistic regression, the baseline model may not necessarily be discarded. In such case, I would consider how much these models improved performance and the time required not only to train them but also to connect cause and effect.

The challenger models I have trained in this thesis are tree-based ensemble algorithms. There are two branches of ensemble models that were developed in parallel: 'Bagging' and 'Boosting.' These machine learning algorithms improved the 'Classification and Regression Trees' (CART) approach developed by Breiman et al. (1984). This method produces hierarchical partitions using binary conditions over the feature space. The output is very easy to interpret and should not be used if the nature of the data is expected to be linear. In addition, this learning method has a variance issue. 'Bagging' or 'Bootstrap aggregating' was developed to address the variance of a learning method by bootstrapping the learning set multiple times, training the learning method on each of the bootstrapped sets, and producing an 'aggregated predictor' with the average of the outcomes when the target variable is numerical or with "a plurality vote when predicting a class" (Breiman 1994). Breiman (1994) concluded "what one loses, with the trees, is a simple and interpretable structure. What one gains is increased accuracy." Ho from AT&T Bell Laboratories —concerned about accuracy losses, which resulted from limiting the complexity when training decision trees to gain accuracy on unseen data— proposed a method to build multiple trees using "randomly selected subspaces of the feature space" (Ho 1995) to improve the accuracy on unseen data without losing the accuracy of the classifier on the training data. Within the shape recognition field, Amit and Geman explored a tree-based approach that randomly selected a small sample of informative features at each node "on a virtually infinite family of binary features . . . of the image data" (Amit and Geman 1997). Later, Breiman (2001) proposed the 'Random Forests' approach that combined the random selection of features to split each decision tree node with Bagging (Breiman 2001). On

Universidad de San Andrés

the Boosting side, Kearns and Valiant introduced the famous question of whether a “weak” learning model, which is slightly better than random guessing, can be ‘boosted’ to produce a strong learning algorithm, in the distribution-free or probably approximately correct (PAC) model (Freund and Schapire 1999). In 1989, Schapire introduced a polynomial-time boosting algorithm and showed that “a model of learnability in which the learner is only required to perform slightly better than guessing is as strong as a model in which the learner’s error can be made arbitrarily small” (Schapire 1990). Freund (1990) developed a more efficient boosting algorithm that presented some practical weaknesses, and in 1995 Freund and Schapire developed the ‘AdaBoost’ algorithm, which overcame the practical weaknesses of the previous boosting algorithms (Freund and Schapire 1999). In 2001, Friedman developed a general gradient descent ‘boosting’ paradigm making a connection between stage-wise additive expansions and steepest-descent minimization in function space (Friedman 2001).

I organized the rest of the thesis as follows: In Section 2, I further explain the ‘bias-variance’ trade-off and provide thorough details of the mathematics behind logistic regression and tree-based ensemble models. In Section 3, I describe the ‘Receiver Operating Characteristic’ and ‘Precision-Recall’ curves, which I have used to assess the performance of the models. In Section 4, I analyze the dataset and describe the preprocessing I have conducted on the features and the target variable. In Section 5, I provide a description and a comparison of the estimation of each model. Finally, I provide my conclusions in Section 6.

Universidad de SanAndrés

2.0 – Methodologies

In this section, I introduce the notation[1] I used in this thesis, the bias-variance tradeoff, and two resampling approaches. In the subsections herein, I describe the methodologies of the learning methods trained.

Let x_{ij} represent the value of the jth variable for the ith observation (James, Witten, et al. 2013) where:

- $i = 1,2, \dots, n$.
- $j = 1,2, \dots, p$.

Matrix $\boldsymbol{X}$ has dimension nxp and its (i, j) element is x_{ij} (James, Witten, et al. 2013):

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}. \quad [1]$$

The rows from matrix $\boldsymbol{X}$ are $x_1, x_2, \dots, x_n$. Vector $\boldsymbol{x}_i$ has the p variable values for the ith observation (James, Witten, et al. 2013):

$$\boldsymbol{x}_i = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix}. \quad [2]$$

The columns of $\boldsymbol{X}$ are $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_p$. Each is a vector of length n (James, Witten, et al. 2013):

$$\boldsymbol{x}_j = \begin{pmatrix} x_{1j} \\ \vdots \\ x_{nj} \end{pmatrix}. \quad [3]$$

The ith observation of the target variable is y_i. The vector of all n observations is the following (James, Witten, et al. 2013):

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}. \quad [4]$$

[1] I used the notation from James, Witten, et al. 2013.

QUAERERE VERUM Universidad de SanAndrés

To further explain the bias-variance trade-off, $D = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$ is the set of data points that are assumed to be $i.i.d.$ drawn from an unknown distribution $P(\boldsymbol{X}, \boldsymbol{Y})$ and $\boldsymbol{y} \in \mathbb{R}$ — I assumed, for the sake of simplicity, a regression setting that uses a squared loss function. There may not be a unique y label associated to that vector for a given input vector of features $\boldsymbol{x}$. Therefore, there is a distribution over all possible y values given vector $\boldsymbol{x}$ (Weinberger 2018). The expected value $\bar{y}$ given $\boldsymbol{x} \in \mathbb{R}^p$ is defined as (Weinberger 2018):

$$\bar{y}(\boldsymbol{x}) = \int y \, Pr(y|\boldsymbol{x}) dy. \qquad [5]$$

The output of algorithm $\mathcal{A}$ after trained on D is denoted as (Weinberger 2018):

$$h_D = \mathcal{A}(D). \qquad [6]$$

The training error is the expected error of the learning method in the training set D. However, the aim of the algorithm is to accurately predict the label of new data points drawn from distribution P —i.e., $(\boldsymbol{x}, y) \sim P$. The test or generalization error is the expected error of the classifier (trained on D) over new data points (Weinberger 2018):

$$E_{x,y\sim p}[(h_D(x) - y)^2] = \iint [h_D(\boldsymbol{x}) - y]^2 \, P(\boldsymbol{x}, y) \, d\boldsymbol{x} \, dy. \qquad [7]$$

Each $\boldsymbol{x}$ and y from D are random variables drawn n times from distribution P. Therefore, dataset D, as well as h_D, is also a random variable. This is because h_D is the output of algorithm $\mathcal{A}$ trained on D (i.e., a function of a random variable). The expected classifier $\bar{h}$ is the weighted average of algorithm $\mathcal{A}$'s output trained on datasets D drawn n times from distribution P (Weinberger 2018):

$$\bar{h} = E_{D\sim P^n}[\mathcal{A}(D)] = \int h_D P(D) dD. \qquad [8]$$

Since h_D is a random variable, the expected error (given $\mathcal{A}$) over many drawn datasets is the following (Weinberger 2018):

$$E_{\substack{(x,y)\sim P \\ D\sim P^n}}[(h_D(x) - y)^2] = \iiint [(h_D(\boldsymbol{x}) - y)^2] \, P(\boldsymbol{x}, y) \, P(D) dy \, d\boldsymbol{x} \, dD. \qquad [9]$$

The aim is to choose an algorithm with the lowest generalization error. Equation [9] can be broken down into the following three terms (Weinberger 2018):

$$E_{\boldsymbol{x},y,D}[(h_D(x) - y)^2] \\ = E_{\boldsymbol{x}}\left[\left(\bar{h}(\boldsymbol{x}) - \bar{y}(\boldsymbol{x})\right)^2\right] + E_{\boldsymbol{x},y}[(\bar{y}(\boldsymbol{x}) - y)^2] + E_{\boldsymbol{x},D}\left[\left(h_D(\boldsymbol{x}) - \bar{h}(\boldsymbol{x})\right)^2\right]. \quad [10]$$

The variance of a classifier is $E_{\boldsymbol{x},D}\left[\left(h_D(\boldsymbol{x}) - \bar{h}(\boldsymbol{x})\right)^2\right]$. If different datasets are drawn, it measures (on average) how much the outputs of the classifier changes with respect to the mean classifier. If a learning method has a high variance, small changes with respect to the training dataset can result in large changes in the predictions (Weinberger 2018). The learning method is overfitting the training data when the expected training error is small but has a large test error.

The noise is $E_{\boldsymbol{x},y}[(\bar{y}(\boldsymbol{x}) - y)^2]$. In the regression setting, the aim of the algorithm is to minimize the squared loss. The minimum of the squared loss is achieved with the expected label $\bar{y}(\boldsymbol{x})$. This error lies in $\bar{y}(\boldsymbol{x})$ not matching the actual label y and, therefore, being irreducible (Weinberger 2018).

The squared bias is $E_{\boldsymbol{x}}\left[\left(\bar{h}(\boldsymbol{x}) - \bar{y}(\boldsymbol{x})\right)^2\right]$. If an infinite number of datasets allows obtaining the expected classifier, the term measures the degree of bias of the learning method towards some explanation that does not match the real patterns in the data (Weinberger 2018). For example, a linear regression is a simple model with less flexibility than smoothing splines or classifiers, such as Random Forests or Gradient Boosting (James, Witten, et al. 2013), and may have a bias problem fitted on data with nonlinear patterns.

This involves a trade-off because when a learning method tries to bring one of the three terms down to zero, the other ones may go up. The bias-variance trade-off also holds when the response variable is qualitative.

If a classifier is trained on $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ where the response variable is qualitative, the training error rate is the proportion of observations that are wrongly classified (Weinberger 2018):

$$\frac{1}{n}\sum_{i=1}^{n} I\big(y_i \neq h_D(x_i)\big). \quad [11]$$

Where

- $h_D(x_i)$ is the predicted label for the ith observation
- I is the indicator function that equals 1 if $y_i \neq h_D(x_i)$ and 0 otherwise.

In practice, it is not possible to draw infinite datasets. Resampling methods, such as validation set approach and k-fold cross validation, address these issues. The validation set approach randomly divides the available set of observations D into training and validation sets (James, Witten, et al., An Introduction to Statistical Learning with Applications in R 2013). Algorithm $\mathcal{A}$ is fit on the training set, and this model is used to predict the target variable in the validation set. The validation error rate provides an estimation of the error rate when the algorithm is trained on new observations (James, Witten, et al. 2013). However, this estimation can be highly variable, because it depends on the splitting of D (James, Witten, et al. 2013). With k-fold cross-validation D is randomly divided "into k groups, or folds, of approximately equal size" (James, Witten, et al. 2013). The first fold is considered as a validation set, and $\mathcal{A}$ is fit on the $k-1$ folds (James, Witten, et al. 2013). The error estimation is computed using the observations in the hold-out fold (James, Witten, et al. 2013). The k-fold cross-validation estimation is calculated by averaging the k estimations of the test error (James, Witten, et al. 2013).

In practice, D is split into training and test sets. The algorithm is fit in the training set. In addition, generally k-fold cross-validation is used over the training set to calibrate the specific parameters of the model. The test set is used to evaluate model performance.

Universidad de SanAndrés

2.1 – Logistic Regression

Let us denote Y as a binary-response variable that can only take two values (Sosa Escudero 1999). Each value is associated with the occurrence of a specific event. Therefore,

$$Y = \begin{cases} 1 & If\ the\ event\ occurs, \\ 0 & If\ the\ event\ does\ not\ occur. \end{cases} \qquad [12]$$

The realization of the random variable Y is y_i and takes values 1 and 0 with probabilities p and $(1-p)$, respectively. Therefore, Y follows a Bernoulli distribution with parameter p and can be written in compact form as follows:

$$Pr(Y = y_i) = p^{y_i}(1-p)^{1-y_i}. \qquad [13]$$

The expected value and variance of Y_i are

$$E(Y) = p. \quad [14]$$

$$Var(Y) = p * (1 - p). \quad [15]$$

Both mean and variance depend on the underlying probability (Murphy 2012). Therefore, any factor that affects the probability will alter the mean and the variance.

Let us $\{y_1, \dots, y_n\}$ denote a random sample of n independent observations and x_i, a vector of p observed covariates. The aim is to estimate the probability of occurrence of the event conditioned on the set of information x_i (Sosa Escudero 1999):

$$p_i = Pr(y_i = 1 | x_i). \quad [16]$$

The probability is also the expectation of y_i conditioned on x_i (Sosa Escudero 1999):

$$E(y_i|x_i) = 1 * p_i + 0 * (1 - p_i) = p_{i.} \quad [17]$$

A simple modeling approach could assume the existence of a linear relationship between x_i and y_i (Sosa Escudero 1999):

$$y_i = x_i\beta + u_i \qquad \text{with } E(u_i|x_i) = 0. \quad [18]$$

Therefore,

$$E(y_i|x_i) = p_i = x_i\beta. \quad [19]$$

Where β is a vector of regression coefficients that is often estimated through the Ordinary Least Squares (OLS) approach. However, conditional probabilities estimated using the above specification present issues (Sosa Escudero 1999). These issues lie in $p_i \in [0,1]$, but the linear predictor $x_i\beta$ can take any real value.

A simple solution to this problem is to transform the probability and model such transformation as a linear function of the covariates. Such transformation consists of two steps.

First, p_i is transformed into odds:

$$odds_i = \frac{p_i}{1 - p_i}. \quad [20]$$

QUAERERE VERUM Universidad de SanAndrés

The odds are the ratio of favorable cases to unfavorable cases. For example, if $p_i = \frac{1}{2}$, the odds are one-to-one; or if $p_i = \frac{1}{3}$, the odds are one-to-two.

Second, logits or log-odds are calculated by taking the logarithms of the odds ratio:

$$\eta_i = logit(p_i) = log\left(\frac{p_i}{1-p_i}\right). \qquad [21]$$

Logits map probabilities from range (0,1) to the entire real line on the following grounds:

- The odds approach 0 and the logit tends to $-\infty$, as p_i goes to 0.
- The odds tend to $+\infty$ and the logit, as p_i approaches 1.
- The odds are one-to-one and the logit is 0 when $p_i = \frac{1}{2}$.
- Negative logits occur when $p_i < \frac{1}{2}$, and positive logits occur when $p_i > \frac{1}{2}$.

The antilogit is the inverse transformation of the logit:

$$p_i = logit^{-1}(\eta_i) = \frac{e^{\eta_i}}{1 + e^{\eta_i}}. \qquad [22]$$

The logistic regression approach assumes that $logit(p_i)$ is a linear function of the explanatory variables:

$$logit(p_i) = x_i\beta. \qquad [23]$$

So that

$$p_i = F(x_i\beta) = \frac{e^{x_i\beta}}{1 + e^{x_i\beta}}. \qquad [24]$$

$F(.)$ is the logistic distribution function that has the following properties (Sosa Escudero 1999):

- $F(-\infty) = 0,$
- $F(\infty) = 1,$
- $f(x) = \frac{dF(x)}{dx} > 0.$

Therefore, $F(.)$ transforms the linear index $x_i\beta$ into probability-consistent values (Sosa Escudero 1999).

The coefficients measure the change in the conditional probability when any of the p explanatory variables experiences a marginal change (Sosa Escudero 1999):

$$\frac{\partial p_i}{\partial x_{ip}} = \beta_p * F(x_i\beta). \quad [25]$$

Therefore, the marginal effect comprises two components (Sosa Escudero 1999):

- β_p measures the impact of a marginal variation of x_{ip} on the linear index $x_i\beta$.
- $F(x_i\beta)$ quantifies the change in the conditional probability due to changes in the linear index $x_i\beta$.

Logistic regression models are often estimated through the maximum likelihood approach. The logarithm of the maximum likelihood function is the following (Sosa Escudero 1999):

$$L(\boldsymbol{y}, \beta) = \sum_{i=1}^{n} \left[y_i * log\big(F(\boldsymbol{X}\beta)\big) + (1 - y_i) * log\big(1 - F(\boldsymbol{X}\beta)\big) \right]. \quad [26]$$

Since the likelihood function of logit models is strictly concave, the solution $\hat{\beta}$ of the maximization, if it exists, defines a unique maximum (Sosa Escudero 1999).

2.2 – Decision Trees

Tree-based methods stratify or segment the predictor space into a set of rectangles and fit a simple model in each (James, Witten, et al. 2013). A tree consists of:

- A root node at the top of the tree.
- Terminal nodes or leaves that correspond to a set of regions $R_1, \dots, R_M$.
- Internal nodes (the points along the tree).
- Branches, which are the segments of the tree that connect the nodes.

A decision tree consists of a set of splitting rules that start in the root node and assign observations of the full dataset to different branches. Classification and Regression Trees (CART) is a supervised hierarchical method that partitions the feature space using binary conditions (Breiman, Friedman, et al. 1984). In this section, I focus on classification trees.

QUAERERE VERUM Universidad de SanAndrés

Decision Trees are algorithms that work iteratively. Let us denote $(Y, \boldsymbol{X})$ as an aleatory vector with $\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{p}}$**.** Starting in the root node, the first step is to find a feature $j \in \{1, \dots, p\}$ and a split point s $(s \in \mathbb{R})$ for feature j. Feature j and split point s define the following pair of half-planes (James, Witten, et al. 2013):

$$R_L(j,s) = \left\{\boldsymbol{x} \in \mathbb{R}^p : x_j \leq s\right\} \text{ and } R_R(j,s) = \left\{\boldsymbol{x} \in \mathbb{R}^p : x_j > s\right\}. \quad [27]$$

Given a training dataset $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$ and assuming that feature j and s have been chosen, D can be divided into two subsets:

$$S_L = \left\{(x_i, y_i) \in D : x_{ij} \leq s\right\} \text{and } S_R = \left\{(x_i, y_i) \in D : x_{ij} > s\right\}. \quad [28]$$

Assuming that the classes of the target variable are $k = 1, \dots, K$, the following subsets are also defined (Weinberger 2018):

$$S_{mk} = \{(x_i, y_i) \in S_m : y_i = k\} \text{ with } m = L, R. \quad [29]$$

Therefore, the probability of label k in subset S_{mk} is the following:

$$p_{mk} = \frac{N_{mk}}{N_m}. \quad [30]$$

Where

- N_{mk} is the number of observations from S_{mk} with label k.
- N_m is total number of observations in S_{mk}.

p_{mk} is calculated for each of the k labels. The observations in node S_m are classified as class k according to the majority class (Hastie, Tibshirani and Friedman 2013):

$$k(S_m) = argmax_k \, p_{mk}. \quad [31]$$

The distribution of the classes within node m should not be equally-likely:

$$p_{m1} = p_{m2} = \cdots = p_{mK}. \quad [32]$$

QUAERERE VERUM
Universidad de
SanAndrés

Therefore, the classification tree algorithm needs purity/impurity functions such as 'Gini' or 'Entropy' (or 'Deviance') to measure the homogeneity within each node[2]:

$$Gini = \sum_{k=1}^{K} p_{mk} * (1 - p_{mk}). \tag{33}$$

$$D = -\sum_{k=1}^{K} p_{mk} * log(p_{mk}). \tag{34}$$

For example, Entropy is maximized when $p_{m1} = p_{m2} = \cdots = p_{mK}$, while it reaches its lowest when the set is biased to a particular class k (Kuhn and Johnson 2013). Therefore, the algorithm will seek to minimize the purity within each node.

Starting in the root node (given feature j and split point s) the purity/impurity of D is measured in the following way (Weinberger 2018):

$$H(D) = \frac{N_L}{N_D} * H(S_L) + \frac{N_R}{N_D} * H(S_R). \tag{35}$$

Where

- H could be Gini, Entropy, or any impurity measure.
- N_L is the number of observations from the left set S_L.
- N_R is the number of observations from the right set N_R.
- N_D is the total number of observations of the training set D (since the algorithm starts in the root node).

Therefore, the purity/impurity of D is measured as a weighted average between the purity/impurity of the subsets S_L and S_R.

The algorithm works in a greedy-way. Given feature j, the algorithm finds the optimal split by trying out every single point. For each split, the algorithm calculates the purities of the two leafs. The optimal split s is the one that outputs the lowest weighted average. The same process

QUAERERE VERUM Universidad de San Andrés

[2] Another impurity measure is "Misclassification Error," but it is not differentiable. By contrast, Gini and Entropy are more sensitive to changes in the probabilities (Hastie, Tibshirani and Friedman 2013).

is repeated for every single feature among the p predictors. The optimal feature, with optimal split s, is the one that outputs the lowest weighted average among all p features.

The tree grows recursively in the same way; that is, each node is split into two more regions. In order to do so, the optimal feature j with optimal split point s is the one with the lowest weighted average, after trying out every single split point for each feature among the p predictors. The tree is grown until some stopping rule is applied—e.g., limiting the depth of the tree— (Bishop 2006).

2.3 – Random Forests

Decision trees have a high variance problem, since splitting can be applied repeatedly until every single leaf is pure. In such case, the classifier memorizes the data leading to a high testing error. Limiting the depth of the tree or the number of terminal nodes (i.e., 'Cost Complexity Pruning') to overcome the variance issue can introduce bias in the model. 'Bagging' or 'Bootstrap Aggregating' is a general procedure that addresses the high variance problem effectively (James, Witten, et al. 2013).

Bagging consist in creating M bootstrapped training samples out of the original dataset D (Breiman 1994). Each dataset $D_1, \dots D_M$ contains the same number of observations as D. The resampling is made randomly with replacement so that one data point can be repeated in dataset D_m many times. Then, an algorithm $\mathcal{A}$ is trained in each dataset resulting in a sequence of outputs $\{h_{D_1}, \dots h_{D_M}\}$. In a regression setting the final prediction is the average of the sequence:

$$\hat{h}(x) = \frac{1}{M}\sum_{j=1}^{M} h_{D_j}(x). \qquad [36]$$

Bagging is very useful for mitigating variance in decision trees (James, Witten, et al. 2013). This mitigation is done by fitting M decision trees without pruning, so that each tree has low bias; the variance issue is addressed by averaging the outcomes of the M trees. In a classification setting, each M tree predicts a class, and the final prediction is the most frequent class. This procedure is called the 'Majority vote.'

Random Forests combine bagging with a random selection of a subset of features $m \ll p$ whenever a split is done in each of the M trees. The random selection of a subset of features decorrelates the trees. Typically, in a classification setting the number of features to split each node is approximately the square root of the total number of features (Hastie, Tibshirani and Friedman 2013). Consequently, decorrelating the trees allows Random Forests to improve the bagging procedure. It could be the case in which a particular feature has a high correlation with the target variable. By bagging the original dataset, each Decision Tree trained on the bagged data sets would be biased towards selecting the variable with a high correlation with the target variable (James, Witten, et al. 2013).

In addition, Random Forests, as well as any learning method using Bagging, enables the estimation of the test error directly from the training dataset. On average, each bagged tree uses around 2/3 of the observations (James, Witten, et al. 2013). Therefore, the error in every single training point of the original dataset can be measured using the classifiers that were not trained with such point. Those errors are averaged to compute the so-called 'Out-of-Bag error'(Weinberger 2018):

$$\epsilon_{OOB}=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{z_i}\sum_{j:(x_i,y_i)\notin D_j} I\big(h_j(x_i),y_i\big). \tag{37}$$

Where z_i is the number of classifiers that were not trained using x_i.

QUAERERE VERUM Universidad de SanAndrés

Bagging improves accuracy at the expense of interpretability. By bagging decision trees, the aggregated predictor cannot be represented using a single tree (James, Witten, et al. 2013).

2.4 – Gradient Boosting

In this subsection, I describe the general framework for Gradient Boosting Algorithms. Boosting is a procedure that combines weak learners; that is, classifiers whose error rate is "only slightly better than random guessing" (Hastie, Tibshirani and Friedman 2013) and outputs a stronger classifier that significantly reduces the bias issue. In 2001, Friedman made a connection "between stagewise and steepest descent minimization" (Friedman 2001). The following steps details the 'Forward Stagewise Additive' algorithm (Hastie, Tibshirani and Friedman 2013):

1) Initialize $h^0 = 0$
2) For $t = 1, \dots, M$:

   2.1) Compute:

$$(\alpha^t, \boldsymbol{\eta}^t) = argmin_{\alpha,\boldsymbol{\eta}} \sum_{i-1}^{N} L\big(y_i, H^{t-1}(\boldsymbol{x}_i) + \alpha h(\boldsymbol{x}_i; \boldsymbol{\eta})\big)\,. \qquad [38]$$

2.2) Set:

$$H^t(\boldsymbol{x}) = H^{t-1}(\boldsymbol{x}) + \alpha^t h(\boldsymbol{x}; \boldsymbol{\eta}^t). \qquad [39]$$

Where $\boldsymbol{\eta}$ is the vector of parameters that characterizes the weak learners and α is the weight that each weak learner has in the ensemble. For trees, $\boldsymbol{\eta}$ "parameterizes the split variables and split points at the internal nodes, and the predictions at the terminal nodes" (Hastie, Tibshirani and Friedman 2013). The term 'forward' refers to the fact that when a new $\alpha^t h(\boldsymbol{x}; \boldsymbol{\eta}^t)$ is added to the ensemble $H^t(\boldsymbol{x})$, previous α and $\boldsymbol{\eta}$ (embedded in $H^{t-1}(\boldsymbol{x})$) are not adjusted.

To develop the 'Forward Stagewise Additive' strategy, Friedman stated a function estimation problem where the setting was (Friedman 2001):

- A random response variable y
- A set of random explanatory variables $\boldsymbol{x} = \{x_1, \dots, x_n\}$
- A loss function $L\big(y, H(\boldsymbol{x})\big)$

His goal was to find an approximation $\widehat{H}(\boldsymbol{x})$ of H^*, where

$$H^* = argmin_H E_{y,\boldsymbol{x}} L\big(y, H(\boldsymbol{x})\big) = argmin_H E_{\boldsymbol{x}} \left[E_y \left(L\big(y, H(\boldsymbol{x})\big) \right) \middle| \, \boldsymbol{x} \right]. \qquad [40]$$

Function $H^*(\boldsymbol{x})$ minimizes the expectation of the loss function over the joint distribution of all $(y, \boldsymbol{x})$-values (Friedman 2001). He focused on the following parameterized form of 'additive' expansions (Friedman 2001):

$$H(\boldsymbol{x}, \{\alpha^t, \boldsymbol{\eta}^t\}_{t=1}^{M}) = \textstyle\sum_{t=1}^{M} \alpha^t \, h(\boldsymbol{x}; \boldsymbol{\eta}^t). \qquad [41]$$

Friedman wanted to find an approximation $\widehat{H}(\boldsymbol{x})$ of H^* using training observations. First, he formulated a parameterized model and chose 'Steepest-descent' as the numerical minimization method. Then, he used a non-parametric approach and applied numerical optimization in function space to minimize the following expectation (Friedman 2001):

$$\phi\big(H(\boldsymbol{x})\big) = E_y\big[L\big(y, H(\boldsymbol{x})\big)\big|\boldsymbol{x}\big]. \qquad [42]$$

Where $H(\boldsymbol{x})$ "evaluated at each point $\boldsymbol{x}$" was considered to be a parameter (Friedman 2001). Friedman expressed the solution as follows:

$$H^*(\boldsymbol{x}) = \sum_{t=0}^{M} h^t(\boldsymbol{x}). \qquad [43]$$

Where $h^0(\boldsymbol{x})$ "is an initial guess" and $\{h^t(\boldsymbol{x})\}_{t=1}^{M}$ "are incremental functions ('steps' or 'boosts')" (Friedman 2001). The steepest-descend approach consist in calculating the gradient of $\phi\big(H(\boldsymbol{x})\big)$ evaluated at $H^{t-1}(\boldsymbol{x}) = \sum_{t=0}^{t-1} h^t(\boldsymbol{x})$. The gradient $g^t(\boldsymbol{x})$ of the expected value of the loss function evaluated at $H^{t-1}(\boldsymbol{x})$ defines the direction in which the expected value of the loss function is minimized:

$$g^t(\boldsymbol{x}) = \left[\frac{\partial E_y\big[L\big(y, H(\boldsymbol{x})\big)\big|\boldsymbol{x}\big]}{\partial H(\boldsymbol{x})}\right]_{H(x)=H^{t-1}(x)}. \qquad [44]$$

QUAERERE VERUM
Universidad de
SanAndrés

Then, α^t determines the step that needs to be taken along that direction and is obtained using the line-search approach:

$$\alpha^t = argmin_{\alpha} E_{y,x} L\big(y, H^{t-1}(\boldsymbol{x}) - \alpha g^t(\boldsymbol{x})\big). \qquad [45]$$

With

$$h^t(\boldsymbol{x}) = -\alpha^t g^t(\boldsymbol{x}). \qquad [46]$$

However, this nonparametric approach cannot be used with finite training data, and Friedman parameterized the optimization problem to minimize a data-based estimation of the expected loss (Friedman 2001):

$$\{\alpha^t, \boldsymbol{\eta}^t\}_{t=1}^{M} = argmin_{\{\alpha'^t, \boldsymbol{\eta}'^t\}_{t=1}^{M}} \sum_{i=1}^{N} L\left(y_i, \sum_{t=1}^{M} \alpha'^t h(\boldsymbol{x}_i; \boldsymbol{\eta}'^t)\right). \qquad [47]$$

Then, Friedman proposed adopting the 'greedy stagewise' approach whenever the above minimization problem cannot be solved. The 'greedy stagewise' approach is the one described at the beginning of this section; that is, initialize h^0 and for $t = 1,2, \dots M$ and solve for (Friedman 2001):

$$(\alpha^t, \boldsymbol{\eta}^t) = argmin_{\alpha,\boldsymbol{\eta}} \sum_{i-1}^{N} L\big(y_i, H^{t-1}(\boldsymbol{x}_i) + \alpha h(\boldsymbol{x}_i; \boldsymbol{\eta})\big). \quad [48]$$

And then set:

$$H^t(\boldsymbol{x}) = H^{t-1}(\boldsymbol{x}) + \alpha^t h(\boldsymbol{x}; \boldsymbol{\eta}^t). \quad [49]$$

Depending on the loss function or base learner $h(\boldsymbol{x}; \boldsymbol{\eta})$ the solution to the minimization problem can be broken into two parts. Given $H^{t-1}(\boldsymbol{x})$, the gradient using finite training data is the following:

$$g^t(x_i) = \left[\frac{\partial L\big(y_i, H(x_i)\big)}{\partial H(x_i)}\right]_{H(x_i)=H^{t-1}(x)}. \quad [50]$$

Moreover, $\boldsymbol{\nabla}^t = \{g^t(x_i)\}_{i=1}^{N}$ determines the "best steepest-descent step direction … in the N −dimensional data space" (J. Friedman 2001). The weak learner $h(\boldsymbol{x}; \boldsymbol{\eta})$ is the "most highly correlated" learner with the gradient of the loss function (Friedman 2001). Assuming α is a small constant, the problem is solved in the same way as the gradient descent by taking a first order Taylor approximation (Weinberger 2018):

$$L\big(H^{t-1}(\boldsymbol{x}) + \alpha h^t(\boldsymbol{x}; \boldsymbol{\eta})\big) \approx L\big(H^{t-1}(\boldsymbol{x})\big) + \alpha\langle \boldsymbol{\nabla}^t, h^t(\boldsymbol{x}; \boldsymbol{\eta})\rangle. \quad [51]$$

Where $\langle \boldsymbol{\nabla}^t, h^t(\boldsymbol{x}; \boldsymbol{\eta})\rangle$ is the inner product between the gradient of the loss function and $h^t(\boldsymbol{x}; \boldsymbol{\eta})$. The linear approximation of the loss linear function only holds for small α values and within a small region around $L\big(H^{t-1}(\boldsymbol{x})\big)$ (Weinberger 2018). Since $L\big(H^{t-1}(\boldsymbol{x})\big)$ is a constant that is not affected by $h^t(\boldsymbol{x}; \boldsymbol{\eta})$, the minimization problem approximately becomes

$$h^t(\boldsymbol{x};\boldsymbol{\eta^t}) = argmin_{\boldsymbol{\eta}}\, \alpha \sum_{i=1}^{n} \frac{\partial L\big(y_i, H^{t-1}(x_i)\big)}{\partial H^{t-1}(x_i)} * h^t(\boldsymbol{x};\boldsymbol{\eta}). \quad [52]$$

Then, the line search is performed, as follows:

$$\alpha^t = argmin_{\alpha} \sum_{i=1}^{N} L\big(y_i, H^{t-1}(\boldsymbol{x}) + \alpha h^t(\boldsymbol{x};\boldsymbol{\eta^t})\big). \quad [53]$$

The approximation is updated (Friedman 2001), as follows:

$$H^t(\boldsymbol{x}) = H^{t-1}(\boldsymbol{x}) + \alpha^t h^t(\boldsymbol{x};\boldsymbol{\eta^t}). \quad [54]$$

2.4.1 – AdaBoost (Adaptive Boosting)

In this section, I describe the 'AdaBoost' algorithm. The general setting is the following:

- $D = \{(x_1, y_1), \ldots, (x_1, y_n)\}$ is the set of samples.
- The classification setting is binary: $y_i \in \{+1, -1\}$.
- α^t is an adaptive parameter that is optimized in every iteration t.
- $h^t(\boldsymbol{x})$ is a binary weak learner that outputs $\{-1, +1\}$.
- The loss function is exponential: $l(H^t) = \sum_{i=1}^{n} e^{-y_i H^t(x_i)}$.

QUAERERE VERUM Universidad de SanAndrés

The algorithm works in an iterative-way and the ensemble classifier is a linear combination of the weak learners. The final classification, after performing T iterations, is (Hastie, Tibshirani and Friedman 2013):

$$H^T(x) = sign[\alpha^1 h^1(x) + \alpha^2 h^2(x) + \cdots + \alpha^T h^T(x)]. \quad [55]$$

Where α^t represents the weight assigned to weak learner $h^t(\boldsymbol{x})$. The weights gives higher influence in the final prediction to the more accurate weak learners in the sequence.

The exponential loss function yields higher values than the correctly classified ones for the misclassified samples. After iteration t:

- H^t classifies the ith observation correctly if $y_i = H^t(x_i)$, which is equivalent to $y_i H^t(x_i) = 1$. In this case, the loss function outputs a negative value $l[H^t(x_i), y_i] = e^{-1}$.

- H^t classifies the ith observation incorrectly if $y_i \neq H^t(x_i)$, which is equivalent to $y_i H^t(x_i) = -1$, and the loss function output a positive value $l[H^t(x_i), y_i] = e^1$.

At each boosting step t, the algorithm assigns a weight w_i^t to each sample observation. The weights of all the observations that have been incorrectly classified by $H^{t-1}(x) = \sum_{t=1}^{t-1} \alpha^t h^t(x)$ increase, whereas the weights decrease for those that were correctly classified. Therefore, the algorithm spots the next weak learner in the observations that have been incorrectly classified (Hastie, Tibshirani and Friedman 2013).

First, all the samples $i = 1,2, \dots N$ are assumed to have the same weight (Hastie, Tibshirani and Friedman 2013):

$$w_i = \frac{1}{N}. \tag{56}$$

When the weights are updated, the following constraint must hold in order to enforce a distribution:

$$\sum_{i=1}^{N} w_i^t = 1. \tag{57}$$

This constraint holds at the beginning, since all the samples are equally weighted.

Assuming that $t-1$ iterations have been performed, the next weak learner $h^t(x)$ and α^t are the result of the following minimization problem (Hastie, Tibshirani and Friedman 2013):

$$(\alpha^t, h^t) = argmin_{\alpha,h} \sum_{i=1}^{N} e^{\left[-y_i\left(H^{t-1}(x_i)+\alpha^t h^t(x_i)\right)\right]}. \tag{58}$$

This can be solved through gradient descent in function space by fixing a small $\alpha > 0$ and using a first order Taylor approximation of the loss function with respect to vector $H^{t-1}(\boldsymbol{x})$ (Weinberger 2018):

$$l\left(H^{t-1}(\boldsymbol{x}) + \alpha h^t(\boldsymbol{x})\right) \approx l\left(H^{t-1}(\boldsymbol{x})\right) + \alpha \langle \nabla l\left(H^{t-1}(\boldsymbol{x})\right), h^t \rangle. \tag{59}$$

Using training data to find the next weak learner the minimization problem becomes (Weinberger 2018):

$$h^t = argmin_h \sum_{i=1}^{N} \frac{\partial l(y_i, H^{t-1}(x_i))}{\partial [H^{t-1}(x_i)]} h^t(x_i). \qquad [60]$$

$$h^t = argmin_h \sum_{i=1}^{N} -y_i e^{-y_i H^{t-1}(x_i)}\, h^t(x_i). \qquad [61]$$

In the expression above, $e^{-y_i H^{t-1}(x_i)}$ shows the extent to which a particular data point i contributes to the loss function (Weinberger 2018). To enforce a distribution of weights, the following normalization factor is defined (Weinberger 2018):

$$Z = \sum_{i=1}^{N} e^{-y_i H^{t-1}(x_i)}. \qquad [62]$$

The minimization problem becomes

$$h^t = argmin_h \sum_{i=1}^{N} -y_i \frac{e^{-y_i H^{t-1}(x_i)}}{Z} h^t(x_i), \qquad [63]$$

denoting $w_i^t(x_i) = \frac{e^{-y_i H^{t-1}(x_i)}}{Z}$

$$h^t = argmin_h \sum_{i=1}^{n} -y_i\, h^t(x_i)\, w_i^t(x_i). \qquad [64]$$

QUAERERE VERUM
Universidad de
SanAndrés

As mentioned above:

- $h^t(x_i)\, y_i = 1 \leftrightarrow h^t(x_i) = y_i$, which means that $h^t(x_i)$ classifies y_i correctly.
- $h^t(x_i)\, y_i = -1 \leftrightarrow h^t(x_i) \neq y_i$, which means that $h^t(x_i)$ classifies y_i incorrectly.

Therefore, the summation can be divided between the correctly and incorrectly classified observations:

$$h^t = argmin_h \sum_{i:h^t(x_i)\neq y_i} w_i^t - \sum_{i:h^t(x_i)=y_i} w_i^t. \qquad [65]$$

Since $1 - \epsilon^t = \sum_{i:h^t(x_i)\neq y_i} w_i^t$ is the weighted accuracy, and $\epsilon^t = \sum_{i:h^t(x_i)=y_i} w_i^t$ is the weighted error, the next weak learner $h^t(x_i)$ is the one that minimizes the error rate (Weinberger 2018):

$$h^t = argmin_h \sum_{i:h^t(x_i)\neq y_i} w_i^t. \quad [66]$$

After finding h^t, in AdaBoost the weight α^t associated with weak learner h^t is adaptive. The optimal weight is the one that minimizes the exponential loss function:

$$\alpha^t = argmin_\alpha \sum_{i=1}^{n} e^{-y_i\left[H^{t-1}(x_i)+\alpha^t\, h^t(x_i)\right]}. \quad [67]$$

To find the optimal weight the expression above is differentiated with respect to α and equated to 0:

$$\sum_{i=1}^{n} -y_i h^t(x_i) e^{-y_i\left[H^{t-1}(x_i)+\alpha^t\, h^t(x_i)\right]} = 0. \quad [68]$$

Given $h^t(x_i)$, the sum over n observations can be again split between the ones that were correctly ($h^t(x_i)\; y_i = 1$) and incorrectly classified($h^t(x_i) y_i \neq 1$) (Weinberger 2018):

$$\sum_{i:h^t(x_i)y_i=1} -y_i h^t(x_i)\; e^{-\left[y_i H^{t-1}(x_i)+\alpha^t h^t(x_i) y_i\right]} + \sum_{i:h^t(x_i)y_i\neq 1} -y_i h^t(x_i)\; e^{-\left[y_i H^{t-1}(x_i)+\alpha^t h^t(x_i) y_i\right]} = 0, \quad [69]$$

dividing by the normalizing factor Z:

$$-e^{-\alpha^t} \sum_{i:h^t(x_i)y_i=1} w_i^t \; + e^{\alpha^t} \sum_{i:h^t(x_i)y_i\neq 1} w_i^t \; = 0, \quad [70]$$

$$-(1-\epsilon^t)\; e^{-\alpha^t} + \epsilon^t\; e^{\alpha^t} = 0. \quad [71]$$

Therefore, the optimal alpha for iteration t is

$$\alpha^t = \frac{1}{2} ln\left(\frac{1-\epsilon^t}{\epsilon^t}\right), \quad [72]$$

With h^t and α^t, the weights for each sample must be updated to find in the next iteration another weak learner and α:

$$w_i^{t+1}(x_i) = \frac{w_i^t}{Z}\, e^{-\alpha^t h^t(x_i) y_i}. \qquad [73]$$

Replacing α^t, the updated weight for each of the samples that have been correctly classified by $h^t(x)$ is:

$$w_i^{t+1}(x_i) = \frac{w_i^t}{Z}\sqrt{\frac{\epsilon^t}{1-\epsilon^t}}, \qquad [74]$$

while the updated weight for each of the incorrectly classified samples is

$$w_i^{t+1}(x_i) = \frac{w_i^t}{Z}\sqrt{\frac{1-\epsilon^t}{\epsilon^t}}. \qquad [75]$$

Z is the normalizing factor that makes the sum of the updated weights total 1:

$$\sqrt{\frac{\epsilon^t}{1-\epsilon^t}} \sum_{i:h_i^t = y_i} w_i^t + \sqrt{\frac{1-\epsilon^t}{\epsilon^t}} \sum_{i:h_i^t \neq y_i} w_i^t = Z, \qquad [76]$$

QUAERERE VERUM
Universidad de
SanAndrés

recalling that $(1-\epsilon^t) = \sum_{i:h_i^t = y_i} w_i^t$ and $\epsilon^t = \sum_{i:h_i^t \neq y_i} w_i^t$:

$$Z = 2\sqrt{\epsilon^t\,(1-\epsilon^t)}, \qquad [77]$$

therefore, replacing the normalizing factor by the updated weights for the correctly classified observations yields:

$$w_i^{t+1} = \frac{w_i^t}{2}\,\frac{1}{1-\epsilon^t}, \qquad [78]$$

while doing the same for the incorrectly classified observation:

$$w_i^{t+1} = \frac{w_i^t}{2} \frac{1}{\epsilon^t}. \qquad [79]$$

The weights will be higher for the samples that have been incorrectly classified in the previous round and lower for the correctly classified ones. The new weak learners are found iteratively by taking into account the mistakes from the previous round. Therefore, AdaBoost recursively corrects the bias problem (Freund and Schapire 1996).

QUAERERE VERUM
Universidad de
SanAndrés

3.0 – Performance measures

In this section, I describe the 'Receiver Operating Characteristics' (ROC) and 'Precision-Recall' graphs that I used in section 5 to evaluate the performance of the models estimated. "A receiver operating characteristics (ROC) graph is a technique for visualizing, organizing and selecting classifiers based on their performance." (Fawcett 2005).

Let us assume a classification problem with only two possible classes $y_i \in \{0,1\}$ and denote the actual classes (Fawcett 2005) as:

- 'Positive' (p) when $y_i = 1$,
- 'Negative' (n) when $y_i = 0$.

For example, if a classifier $h(\boldsymbol{x})$ wants to predict whether a client from a financial institution will default, the positive class is default (i.e., $y_i = 1$) if the client actually defaulted, and $y_i = 0$ if the client did not default.

Let us also assume the predicted classes are denoted as (Fawcett 2005):

- 'Yes' (Y) when $\hat{y}_i = h(x_i) = 1$,
- 'No' (N) when $\hat{y}_i = h(x_i) = 0$.

Given a classifier and a test set, a $2x2$ confusion matrix (or contingency table) can be built (Fawcett 2005). A confusion matrix is a squared matrix, whose dimensions reflect the number of classes.

Figure 3.0.1 – Confusion matrix

			Actual	
			P	**N**
			1	**0**
Predicted	**P**	**1**	TP	FP
	N	**0**	FN	TN
Total actual			P	N

Source: Table prepared based on Fawcett, 2005

The following bullet points detail the metrics displayed in the confusion matrix (Sosa Escudero 2020):

QUAERERE VERUM
Universidad de
San Andrés

- $P = \sum_{i=1}^{n} y_i$ is the total number of observations with actual positive class from the test, where n is the total number of observations.
- $N = \sum_{i=1}^{n} (1 - y_i)$ is the total number of observations with actual negative class.
- True positives (TP) is the total number of observations whose predicted classes coincide with their actual positive ones. Therefore, $TP = \sum_{i=1}^{n} \hat{y}_i \, y_i$ because $\hat{y}_i \, y_i = 1$ when $\hat{y}_i = y_i = 1$; and $\hat{y}_i \, y_i = 0$ when $y_i = 1$ and the predicted class is $\hat{y}_i = 0$.
- True negatives TN is the total number of observations whose predicted classes coincide with their actual negative ones. Therefore, $TN = \sum_{i=1}^{n}(1 - \hat{y}_i)\,(1 - y_i)$.
- False positives FP is the total number of observations with actual class $y_i = 0$, and predicted class $\hat{y}_i = 1$ (i.e., $(1 - y_i) = 1$). Therefore, $FP = \sum_{i=1}^{n} \hat{y}_i \,(1 - y_i)$.
- False negatives FN is the total number of observations with actual class $y_i = 1$ and predicted class $\hat{y}_i = 0$. Therefore, $FN = \sum_{i=1}^{n}(1 - \hat{y}_i)\, y_i$.

The numbers along the major diagonal of the matrix represent the observations that were correctly classified (Fawcett 2005). Those numbers are the inputs to compute the accuracy of a classifier:

$$Accuracy = \frac{TP + TN}{P + N}. \quad [80]$$

The true positive rate, recall, or sensitivity is the ratio between the number of true positives and the total number of positives:

$$TP\ rate = Recall = Sensitivity = \frac{TP}{P}. \quad [81]$$

The false positive rate is

$$FP\ rate = \frac{FP}{N}. \quad [82]$$

The precision of a classifier is the ratio between true positives and the total number of predicted positive labels, and it is used to measure the ability of a model to predict the positive class:

$$Precision = \frac{TP}{TP + FP}. \quad [83]$$

QUAERERE VERUM
Universidad de
SanAndrés

And the F1-score is the harmonic mean between precision and recall:

$$F1 - score = 2\frac{(Precision * Recall)}{(Precision + Recall)}. \quad [84]$$

Decision trees (among others) are discrete classifiers because they output a class decision (i.e., a Y or N) for each observation. When discrete classifiers are trained on the test set, they output a single confusion matrix (Fawcett 2005).

Classifiers such as logistic regressions yield a probability for each observation (Fawcett 2005). In such case, a threshold $c \in [0,1]$ must be set to produce a discrete classifier and output a single confusion matrix. Therefore, when $p_i > c$, then $\hat{y}_i = 1$. The extreme cases are the following (Sosa Escudero 2020):

- When $c = 1$, none of the observations are predicted as positive, but all of them are predicted as negative (i.e., $\hat{y}_i = 0$). In that event, $TP\ rate = FP\ rate = 0$. This strategy never outputs false positives, because it never predicts positives.
- When $c = 0$, all of the observations in the test set are classified as positives. In that event, all the actual positives are classified as positives, but all the actual negatives are classified as positives as well. Therefore, $TP\ rate = FP\ rate = 1$.
- A threshold set to $c = 1 - \varepsilon$ allows the classifier to predict positive classes (i.e., $TP\ rate > 0$) and also enables the prediction of false positives (i.e., $FP\ rate > 0$.

Universidad de San Andrés

A ROC curve plots the relationship between the true positive and the false positive rates for every single threshold $c \in [0,1]$ (Sosa Escudero 2020). The number of true positives and false negatives depend on the threshold c, as well as on the total predicted positives $\sum_{i=1}^{n} \hat{y}_i\ (c)$:

$$\sum_{i=1}^{n} \hat{y}_i\ y_i\ (c) + \sum_{i=1}^{n} \hat{y}_i\ (1 - y_i)\ (c) = \sum_{i=1}^{n} \hat{y}_i\ (c), \quad [85]$$

$$\frac{\sum_{i=1}^{n} \hat{y}_i\ y_i\ (c)}{P} + \frac{\sum_{i=1}^{n} \hat{y}_i\ (1 - y_i)\ (c)}{P} = \frac{\sum_{i=1}^{n} \hat{y}_i\ (c)}{P}. \quad [86]$$

Re-arranging:

$$\frac{\sum_{i=1}^{n} \hat{y}_i\ y_i\ (c)}{P} = \frac{\sum_{i=1}^{n} \hat{y}_i\ (c)}{P} - \frac{N}{N}\frac{\sum_{i=1}^{n} \hat{y}_i\ (1 - y_i)\ (c)}{P}, \quad [87]$$

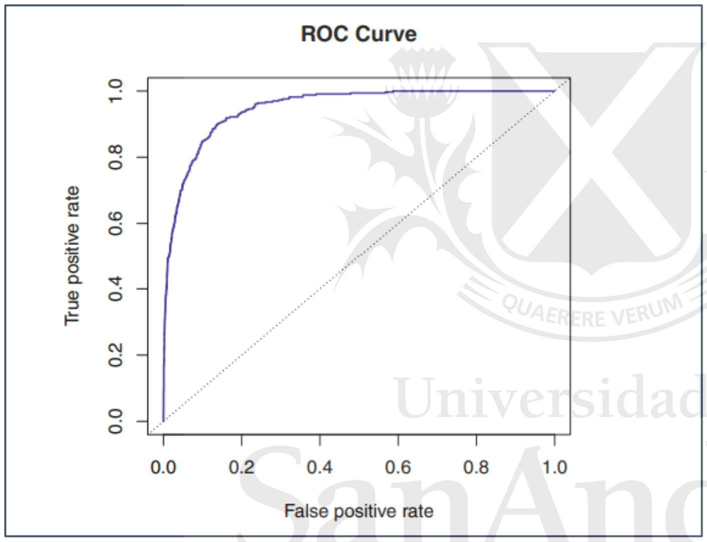
$$\frac{\sum_{i=1}^{n} \hat{y}_i \, y_i \, (c)}{P} = \frac{\sum_{i=1}^{n} \hat{y}_i \ (c)}{P} - \frac{N}{P} \frac{\sum_{i=1}^{n} \hat{y}_i \, (1 - y_i) \ (c)}{N}. \qquad [88]$$

Therefore,

$$TP\ rate\ (c) = \frac{\sum_{i=1}^{n} \hat{y}_i \ (c)}{P} - \frac{N}{P}\ FP\ rate\ (c). \qquad [89]$$

ROC plots the true positive rates on the y-axis and the false positive rates on the x-axis (Fawcett 2005). The point (0,0) corresponds to a threshold $c = 1$, while (1,1) corresponds to $c = 0$.

Figure 3.0.2 – Illustration of a ROC Curve



Source: (James, Witten, et al., An Introduction to Statistical Learning with Applications in R 2013)

QUAERERE VERUM Universidad de San Andrés

A perfect classifier would be one with $TP\ rate\ (c) = 1$ and $rate\ (c) = 0$. This classifier would represent point (0,1) in the ROC space. Therefore, the most desirable point in the ROC space is the closest to (0,1). The diagonal line $y = x$ represents a classifier that randomly guesses the class (Fawcett 2005). Any classifier that outputs a ROC curve below the diagonal line performs worse than a random guessing classifier because it reverses the classifications (Fawcett 2005). The true positive and false positive rates of the reverse classifier ($TP\ rate^R$ and $FP\ rate^R$, respectively) are (Sosa Escudero 2020)

$$TP\ rate^R = \frac{\sum_{i=1}^{n} (1 - \hat{y}_i)\, y_i}{P}. \qquad [90]$$

$$FP\ rate^R = \frac{\sum_{i=1}^{n}(1-\hat{y}_i)\,(1-y_i)}{N}. \quad [91]$$

In addition, $FP\ rate - TP\ rate = TP\ rate^R - FP\ rate^R$ (Sosa Escudero, Análisis ROC 2020). On the one hand, $FP\ rate - TP\ rate$ equals to:

$$\frac{\sum_{i=1}^{n}\hat{y}_i\,(1-y_i)}{N} - \frac{\sum_{i=1}^{n}\hat{y}_i\,y_i}{P} = -\left[\frac{\sum_{i=1}^{n}\hat{y}_i\,y_i}{P} - \frac{\sum_{i=1}^{n}\hat{y}_i\,(1-y_i)}{N}\right]. \quad [92]$$

On the other hand, $TP\ rate^R - FP\ rate^R$:

$$\frac{\sum_{i=1}^{n}(1-\hat{y}_i)\,y_i}{P} - \frac{\sum_{i=1}^{n}(1-\hat{y}_i)\,(1-y_i)}{N}. \quad [93]$$

Rearranging the equation above:

$$\frac{\sum_{i=1}^{n}\ y_i}{P} - \frac{\sum_{i=1}^{n}\hat{y}_i\,y_i}{P} - \frac{\sum_{i=1}^{n}\ (1-y_i)}{N} + \frac{\sum_{i=1}^{n}\hat{y}_i\,(1-y_i)}{N}. \quad [94]$$

$$-\left[\frac{\sum_{i=1}^{n}\hat{y}_i\,y_i}{P} - \frac{\sum_{i=1}^{n}\hat{y}_i\,(1-y_i)}{N}\right] + \frac{\sum_{i=1}^{n}\ y_i}{P} - \frac{\sum_{i=1}^{n}\ (1-y_i)}{N}. \quad [95]$$

Since $\sum_{i=1}^{n}\ y_i$ represents the number of positives P, and $\sum_{i=1}^{n}\ (1-y_i)$, the number of negatives N:

$$-\left[\frac{\sum_{i=1}^{n}\hat{y}_i\,y_i}{P} - \frac{\sum_{i=1}^{n}\hat{y}_i\,(1-y_i)}{N}\right] + \frac{P}{P} - \frac{N}{N}. \quad [96]$$

Therefore,

$$FP\ rate - TP\ rate = -\left[\frac{\sum_{i=1}^{n}\hat{y}_i\,y_i}{P} - \frac{\sum_{i=1}^{n}\hat{y}_i\,(1-y_i)}{N}\right]. \quad [97]$$

Consequently, the ROC of a classifier is under the diagonal line when $FP\ rate > TP\ rate$ (Sosa Escudero 2020).

The ROC curve of a perfect classifier connects points (0,0), (0,1), and (1,1) in the ROC space. The ROC curve can also be used to set a classification threshold. This depends on the classification setting. For example, if the purpose of the classifier is to predict whether new

QUAERERE VERUM Universidad de San Andrés

observations might have a serious health disease, then a threshold lower than 0.5 could be chosen even at the expense of increasing the $FP\ rate(c)$.

Calculating the area under the ROC curve (AUC) can help measure how far the ROC is from the ideal. The AUC is used to compare classifiers, since it reduces the ROC performance to a single scalar value (Fawcett 2005). The AUC values of classifiers that are better than random guessing are between 0.5 and 1.

AUCs can be used to compare the performance of different classifiers and choose the one with the highest AUC. However, this depends on whether the sample is balanced. A sample is balanced when the proportion of positive observations is approximately 50%. When the sample is unbalanced and there are more negative observations relative to positive ones, 'Precision-Recall' curves can also be used to evaluate the performance of machine learning models. Since the calculation of precision does not require the number of true negatives observations, it is not affected by the unbalanced sample.

QUAERERE VERUM
Universidad de
SanAndrés

4.0 – Data and inputs

The dataset used in this thesis is the United States Census Income. It was extracted by Barry Becker from the 1994 Census database (Kohav and Becker 1996). It consists of 48,842 observations, each representing an individual taking part in the census. Table 4.0.1 shows the variables from the dataset and its description.

Table 4.0.1.1 – Variables description[3]

Feature	Description	Type of feature
age	The age of the census person	Continuous
workclass	Shows whether the census person is self-employed, works in the private sector, in any stage of the government, or receives no payment	Categorical
education	Contains the educational level: high school, bachelor, doctorade, etc.	Categorical
educational-num	Shows the number of years the census person received a formal education	Continuous
marital-status	Denotes the marital status condition: married, not-married, widowed, etc.	Categorical
occupation	Details the job of the census person: protective services, farming, machine operator, etc.	Categorical
relationship	Describes the family role of the census person: husband, wife, unmarried, etc.	Categorical
race	Denotes the race: black, white, etc.	Categorical
gender	Shows whether the census person is male or female	Categorical
capital-gain	The amount of profits received from an investment	Continuous
capital-loss	The amount of losses resulted from an investment	Continuous
hours-per-week	The number of hours worked per week	Continuous
native-country	Shows the native country of the census person: USA, England, Cuba, etc.	Categorical
income	Denotes whether the census person earns less or more than $50K per year	Binary target

Source: Table prepared on Census Income dataset

The target variable is 'income'. This classification problem aims to predict whether a person earns more or less than $50 thousand per year.

To analyze the dataset, I transformed 'income' into a dummy variable. I assigned 0 values to samples totaling $50 thousand or amounting to less than that sum (i.e., the negative class or

[3] There is a feature named ´fnlwgt´ which I did not use in the analysis.

low income class) and 1 to the rest of the samples (i.e., the positive class or high income). Next, I classified samples according to 'gender': I assigned 1 to females and 0 to males.

Before preprocessing the features, I used Python's library 'sklearn.model_selection.train_test_split' (setting a random seed of 7) to divide the samples into train and test sets. I set the train set to 70% of the total samples (totaling 34,189 observations out of 48,842). I split the set at this stage to prevent overfitting because I imputed missing values and grouped some categories using the target variable from the training set. With respect to categorical variables, I only created categories, whose proportion of observation was (approximately) less than 1% over the total number of samples from the training set. At this stage, I conducted the transformations on the features I used to estimate all the models. The dataset is moderately unbalanced with approximately 25% of the individuals earning more than $50 thousand per year.

Table 4.0.1.2 displays 'marital-status' before grouping categories. To read the table, e.g., 'Married-AF-spouse' proportion over the total number of samples (from the train set) was 0,1%. Within such category 63% of the observations (i.e., 17 out of 27) represent individuals earning $50thousand per year or less (i.e., 'Low income'). I grouped that category with 'Married-civ-spouse' into 'Married'. I grouped that category with 'Married-civ-spouse', since both categories are characterized by a similar income distribution, and 'Married-AF-spouse' would have comprised only 27 observations.

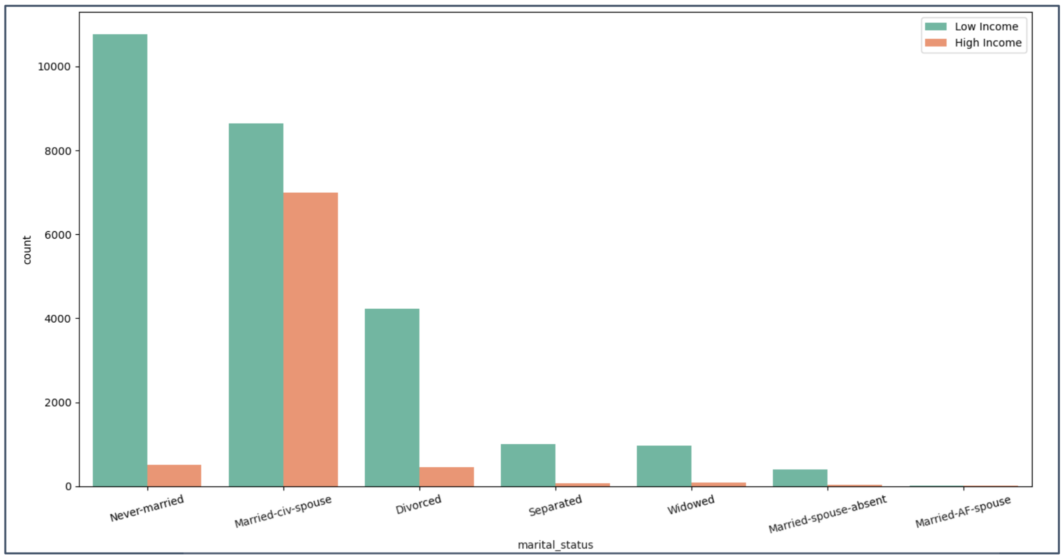
Universidad de San Andrés

Table 4.0.1.2 – Number and proportion of low and high income observations (before grouping)

Marital status	Number of observations			Proportion of observations of each category over total number of samples	Proportion of low and high income observations over total observations within each category	
	Low income	High income	Total		Low income	High income
Divorced	4.226	465	4.691	13,7%	90,1%	9,9%
Married-AF-spouse	17	10	27	0,1%	63,0%	37,0%
Married-civ-spouse	8.634	6.986	15.620	45,7%	55,3%	44,7%
Married-spouse-absent	396	40	436	1,3%	90,8%	9,2%
Never-married	10.758	514	11.272	33,0%	95,4%	4,6%
Separated	1.000	69	1.069	3,1%	93,5%	6,5%
Widowed	977	97	1.074	3,1%	91,0%	9,0%
Total	26.008	8.181	34.189	100,0%	76,1%	23,9%

Source: Table prepared on Census Income dataset

Figure 4.0.1.1 – Number low and high income observations from each category



Source: Figure prepared on Census Income dataset

I renamed some of the categories of all of the categorical features. The following table shows the 'mapping' between the original and renamed categories.

Table 4.0.1.3 – Mapping of 'marital-status' categories

Category	
Before grouping	**After Grouping**
Divorced	Divorced
Married-AF-spouse	Married
Married-civ-spouse	Married
Married-spouse-absent	Married-sa
Never-married	Never_married
Separated	Separated
Widowed	Widowed

Source: Table prepared on Census Income dataset

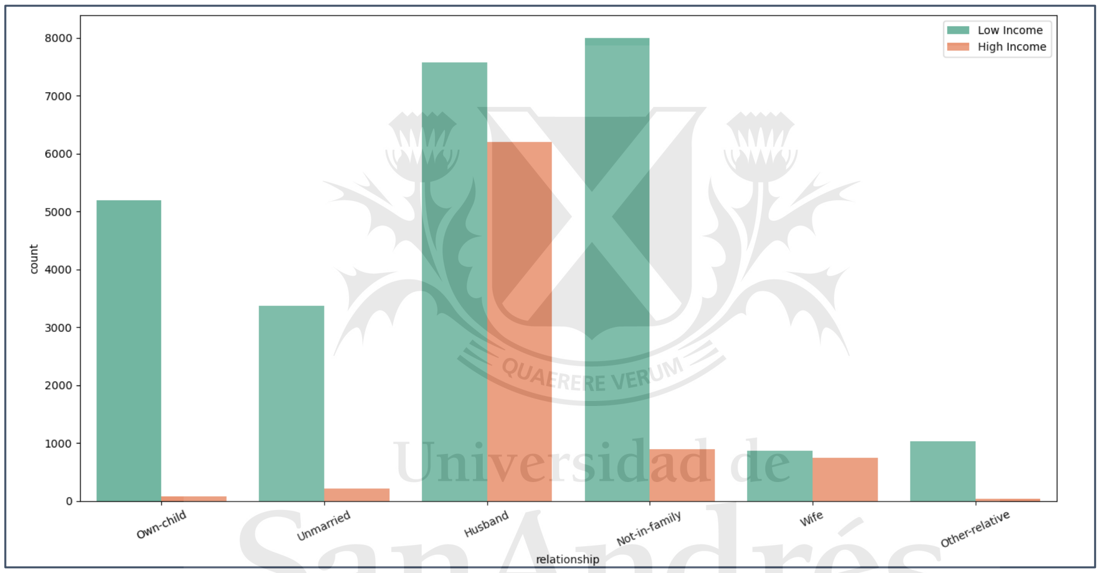
In terms of 'relationship', I only grouped 'Husband' and 'Wife' into 'Hus_Wif'. Although the proportion of 'Wife' over total number of samples based on the income distribution is 4.8%, I considered it logical to group it into 'Husband.'

Table 4.0.1.4 – Number and proportion of low and high income observations (before grouping)

Relationship	Number of observations			Proportion of observations of each category over total number of samples	Proportion of low and high income observations over total observations within each category	
	Low income	High income	Total		Low income	High income
Husband	7.574	6.205	13.779	40,3%	55,0%	45,0%
Not-in-family	7.987	895	8.882	26,0%	89,9%	10,1%
Other-relative	1.027	34	1.061	3,1%	96,8%	3,2%
Own-child	5.189	85	5.274	15,4%	98,4%	1,6%
Unmarried	3.365	211	3.576	10,5%	94,1%	5,9%
Wife	866	751	1.617	4,7%	53,6%	46,4%
Total	26.008	8.181	34.189	100,0%	76,1%	23,9%

Source: Table prepared on Census Income dataset

Figure 4.0.1.2 – Number low and high income observations from each category



QUAERERE VERUM
Universidad de
SanAndrés

Source: Figure prepared on Census Income dataset

Table 4.0.1.5 – Mapping of 'relationship' categories

Category	
Before grouping	After Grouping
Husband	Hus_Wif
Not-in-family	NIF (not in family)
Other-relative	Other
Own-child	Own_ch
Unmarried	Unmarried
Wife	Hus_Wif

Source: Table prepared on Census Income dataset

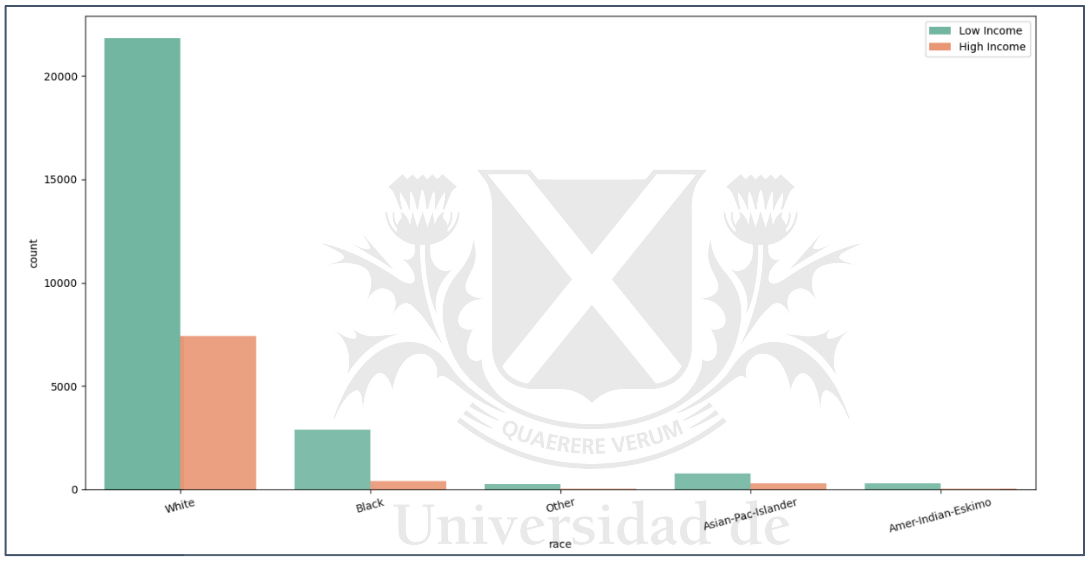
In terms of "race", 'White' represents 85.5% of the total number of samples.

Table 4.0.1.6 – Number and proportion of low and high income observations (before grouping)

Race	Number of observations			Proportion of observations of each category over total number of samples	Proportion of low and high income observations over total observations within each category	
	Low income	High income	Total		Low income	High income
Amer-Indian-Eskimo	289	43	332	1,0%	87,0%	13,0%
Asian-Pac-Islander	765	286	1.051	3,1%	72,8%	27,2%
Black	2.899	397	3.296	9,6%	88,0%	12,0%
Other	249	35	284	0,8%	87,7%	12,3%
White	21.806	7.420	29.226	85,5%	74,6%	25,4%
Total	26.008	8.181	34.189	100,0%	76,1%	23,9%

Source: Table prepared on Census Income dataset

Figure 4.0.1.3 – Number low and high income observations from each category



Source: Figure prepared on Census Income dataset

Universidad de
SanAndrés

Therefore, 'White' remained a single category, and the rest of the races were grouped into 'Other'. Next, I assigned 1 values to 'White' and 0 values to 'Other'.

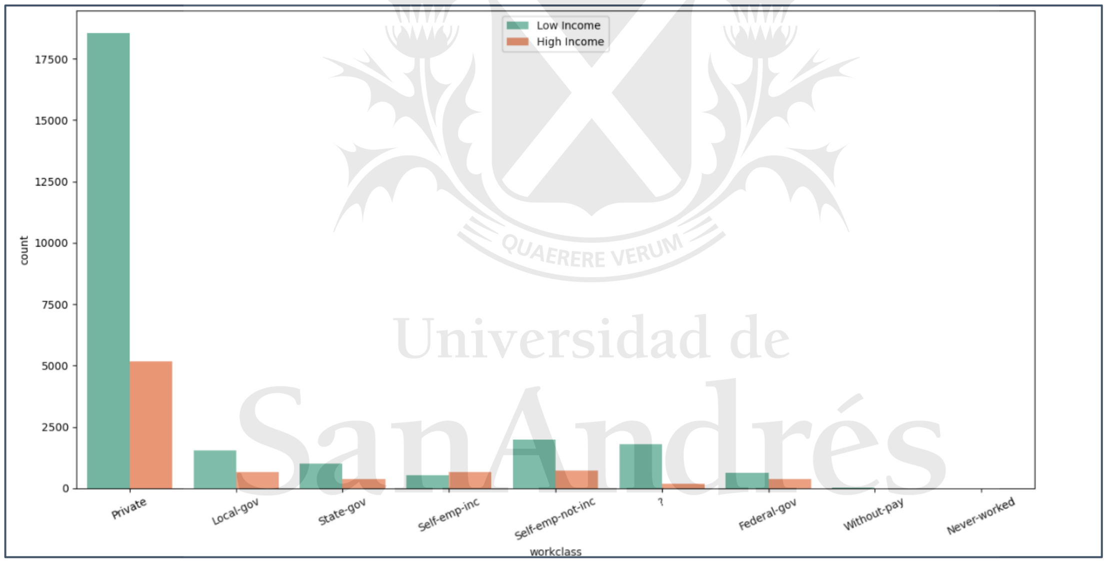
For 'workclass' feature, I renamed missing values '?' as 'Other'. I also grouped 'Without-pay' and 'Never-worked' into 'Other', as both categories had a very small proportion of observations. I do not reckon this grouping decision can distort the information because most of the times missing values might represent a mix of categories.

Table 4.0.1.7 – Number and Proportion of Low and High Income Observations (before grouping)

Workclass	Number of observations			Proportion of observations of each category over total number of samples	Proportion of low and high income observations over total observations within each category	
	Low income	High income	Total		Low income	High income
?	1.783	193	1.976	5,8%	90,2%	9,8%
Federal-gov	622	380	1.002	2,9%	62,1%	37,9%
Local-gov	1.550	658	2.208	6,5%	70,2%	29,8%
Never-worked	2	0	2	0,0%	100,0%	0,0%
Private	18.528	5.184	23.712	69,4%	78,1%	21,9%
Self-emp-inc	540	661	1.201	3,5%	45,0%	55,0%
Self-emp-not-inc	1.968	735	2.703	7,9%	72,8%	27,2%
State-gov	999	368	1.367	4,0%	73,1%	26,9%
Without-pay	16	2	18	0,1%	88,9%	11,1%
Total	26.008	8.181	34.189	100,0%	76,1%	23,9%

Source: Table prepared on Census Income dataset

Figure 4.0.1.4 – Number low and high income observations from each category



QUAERERE VERUM
Universidad de
SanAndrés

Source: Figure prepared on Census Income dataset

Table 4.0.1.8 – Mapping of 'workclass' categories

Category	
Before grouping	**After Grouping**
?	Other
Federal-gov	F_GOV
Local-gov	L_GOV
Never-worked	Other
Private	Priv
Self-emp-inc	SEI
Self-emp-not-inc	SENI
State-gov	S_GOV
Without-pay	Other

Source: Table prepared on Census Income dataset

The categorical feature 'education' comprises the same information as the continuous feature 'educational-num'. For example, people whose education category is 'Preschool' studied one year. Therefore, I dropped 'educational-num' and used 'education'.

Table 4.0.1.9 –'education' against 'educational-num'

educational-num	education
1	Preschool
2	1st-4th
3	5th-6th
4	7th-8th
5	9th
6	10th
7	11th
8	12th
9	HS-grad
10	Some-college
11	Assoc-voc
12	Assoc-acdm
13	Bachelors
14	Masters
15	Prof-school
16	Doctorate

Source: Table prepared on Census Income dataset

I only grouped 'Preschool' with '1st-4st' given its small number of observations.

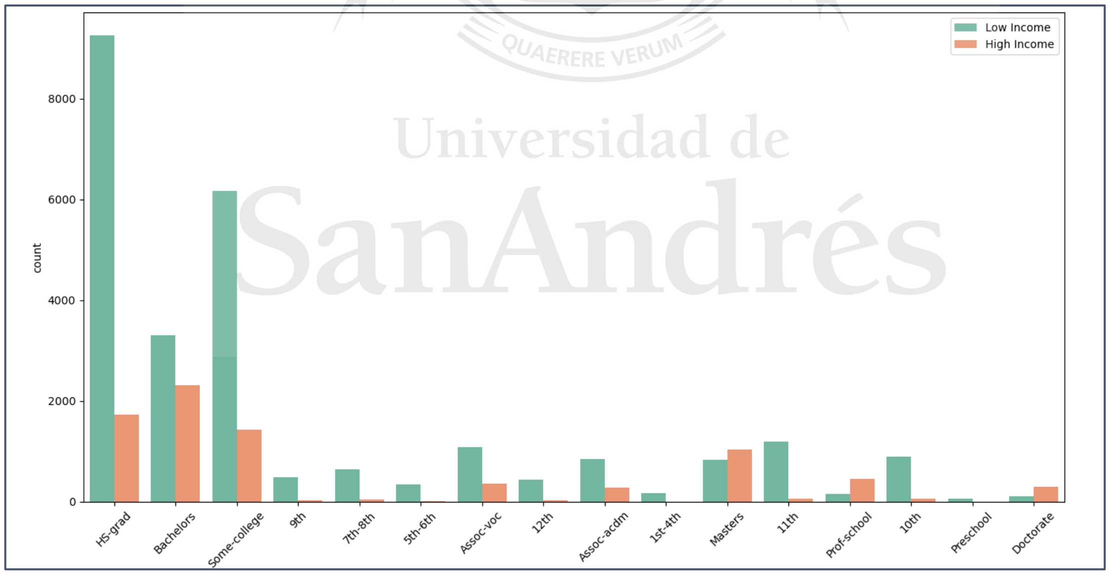
QUAERERE VERUM Universidad de SanAndrés

Table 4.0.1.10 – Number and proportion of low and high income observations (before grouping)

Education	Number of observations			Proportion of observations of each category over total number of samples	Proportion of low and high income observations over total observations within each category	
	Low income	High income	Total		Low income	High income
10th	900	69	969	2,8%	92,9%	7,1%
11th	1.199	60	1.259	3,7%	95,2%	4,8%
12th	436	33	469	1,4%	93,0%	7,0%
1st-4th	167	5	172	0,5%	97,1%	2,9%
5th-6th	347	16	363	1,1%	95,6%	4,4%
7th-8th	643	40	683	2,0%	94,1%	5,9%
9th	493	29	522	1,5%	94,4%	5,6%
Assoc-acdm	855	289	1.144	3,3%	74,7%	25,3%
Assoc-voc	1.081	361	1.442	4,2%	75,0%	25,0%
Bachelors	3.300	2.318	5.618	16,4%	58,7%	41,3%
Doctorate	109	297	406	1,2%	26,8%	73,2%
HS-grad	9.251	1.731	10.982	32,1%	84,2%	15,8%
Masters	838	1.044	1.882	5,5%	44,5%	55,5%
Preschool	59	1	60	0,2%	98,3%	1,7%
Prof-school	155	456	611	1,8%	25,4%	74,6%
Some-college	6.175	1.432	7.607	22,2%	81,2%	18,8%
Total	26.008	8.181	34.189	100,0%	76,1%	23,9%

Source: Table prepared on Census Income dataset

Figure 4.0.1.5 – Number low and high income observations from each category



Source: Figure prepared on Census Income dataset

QUAERERE VERUM
Universidad de SanAndrés

Groupings into “occupation” were based on the income distribution and logical criteria:

- ‘Other-service’ with ‘Priv-house-serv’ into ‘Other_Serv’ because ‘Other-service’ comprised a small number of observations.
- ‘Protective-serv’ with ‘Armed-Forces’ into ‘Security’ because of the small number of observations of ‘Armed-Forces’ category

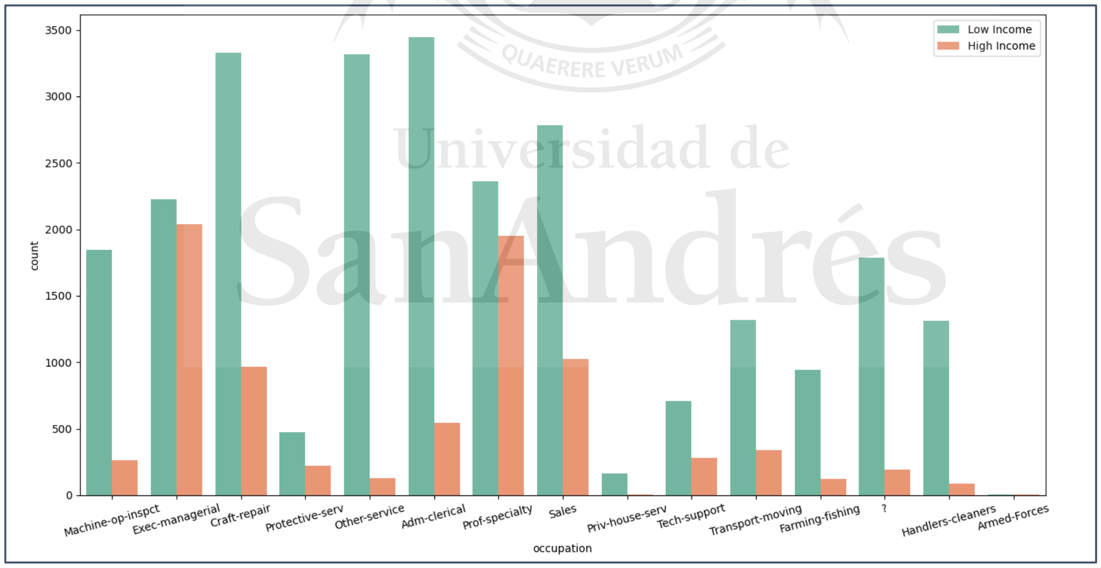
Then, I renamed missing values '?' as 'Other'.

Table 4.0.1.11– Number and Proportion of Low and High Income Observations (before grouping)

Occupation	Number of observations			Proportion of observations of each category over total number of samples	Proportion of low and high income observations over total observations within each category	
	Low income	High income	Total		Low income	High income
?	1.785,00	193,00	1.978,00	5,8%	90,2%	9,8%
Adm-clerical	3.442,00	544,00	3.986,00	11,7%	86,4%	13,6%
Armed-Forces	8,00	4,00	12,00	0,0%	66,7%	33,3%
Craft-repair	3.327,00	966,00	4.293,00	12,6%	77,5%	22,5%
Exec-managerial	2.226,00	2.039,00	4.265,00	12,5%	52,2%	47,8%
Farming-fishing	944,00	124,00	1.068,00	3,1%	88,4%	11,6%
Handlers-cleaners	1.313,00	88,00	1.401,00	4,1%	93,7%	6,3%
Machine-op-inspct	1.845,00	264,00	2.109,00	6,2%	87,5%	12,5%
Other-service	3.318,00	129,00	3.447,00	10,1%	96,3%	3,7%
Priv-house-serv	162,00	3,00	165,00	0,5%	98,2%	1,8%
Prof-specialty	2.360,00	1.952,00	4.312,00	12,6%	54,7%	45,3%
Protective-serv	472,00	224,00	696,00	2,0%	67,8%	32,2%
Sales	2.781,00	1.026,00	3.807,00	11,1%	73,0%	27,0%
Tech-support	709,00	283,00	992,00	2,9%	71,5%	28,5%
Transport-moving	1.316,00	342,00	1.658,00	4,8%	79,4%	20,6%
Total	26.008	8.181	34.189,00	100,0%	76,1%	23,9%

Source: Table prepared on Census Income dataset

Figure 4.0.1.6 – Number low and high income observations from each category



Source: Figure prepared on Census Income dataset

Table 4.0.1.12 – Mapping of ‘occupation’ categories

Category	
Before grouping	**After Grouping**
?	Others
Adm-clerical	A_Cler
Armed-Forces	Security
Craft-repair	Craft
Exec-managerial	Exec
Farming-fishing	Farming
Handlers-cleaners	Cleaners
Machine-op-inspct	Mach_insp
Other-service	Other_Serv
Priv-house-serv	Other_Serv
Prof-specialty	Prof
Protective-serv	Security
Sales	Sales
Tech-support	Tech
Transport-moving	Transp

Source: Table prepared on Census Income dataset

The ‘native_country’ feature comprises individuals, whose countries represent a small proportion of the total samples.

QUAERERE VERUM
Universidad de
SanAndrés

Table 4.0.1.13– Number and proportion of low and high income observations (before grouping)

Native country	Number of observations			Proportion of observations of each category over total number of samples	Proportion of low and high income observations over total observations within each category	
	Low income	High income	Total		Low income	High income
?	447,00	150,00	597,00	1,7%	74,9%	25,1%
Cambodia	16,00	6,00	22,00	0,1%	72,7%	27,3%
Canada	83,00	45,00	128,00	0,4%	64,8%	35,2%
China	60,00	27,00	87,00	0,3%	69,0%	31,0%
Columbia	53,00	4,00	57,00	0,2%	93,0%	7,0%
Cuba	67,00	23,00	90,00	0,3%	74,4%	25,6%
Dominican-Republic	67,00	3,00	70,00	0,2%	95,7%	4,3%
Ecuador	25,00	1,00	26,00	0,1%	96,2%	3,8%
El-Salvador	106,00	7,00	113,00	0,3%	93,8%	6,2%
England	55,00	35,00	90,00	0,3%	61,1%	38,9%
France	19,00	9,00	28,00	0,1%	67,9%	32,1%
Germany	113,00	40,00	153,00	0,4%	73,9%	26,1%
Greece	18,00	9,00	27,00	0,1%	66,7%	33,3%
Guatemala	68,00	2,00	70,00	0,2%	97,1%	2,9%
Haiti	45,00	5,00	50,00	0,1%	90,0%	10,0%
Holand-Netherlands	1,00	0,00	1,00	0,0%	100,0%	0,0%
Honduras	14,00	0,00	14,00	0,0%	100,0%	0,0%
Hong	16,00	6,00	22,00	0,1%	72,7%	27,3%
Hungary	8,00	5,00	13,00	0,0%	61,5%	38,5%
India	59,00	44,00	103,00	0,3%	57,3%	42,7%
Iran	23,00	15,00	38,00	0,1%	60,5%	39,5%
Ireland	21,00	11,00	32,00	0,1%	65,6%	34,4%
Italy	52,00	26,00	78,00	0,2%	66,7%	33,3%
Jamaica	59,00	7,00	66,00	0,2%	89,4%	10,6%
Japan	42,00	23,00	65,00	0,2%	64,6%	35,4%
Laos	13,00	2,00	15,00	0,0%	86,7%	13,3%
Mexico	643,00	31,00	674,00	2,0%	95,4%	4,6%
Nicaragua	28,00	2,00	30,00	0,1%	93,3%	6,7%
Outlying-US(Guam-USVI-et	17,00	1,00	18,00	0,1%	94,4%	5,6%
Peru	36,00	3,00	39,00	0,1%	92,3%	7,7%
Philippines	150,00	61,00	211,00	0,6%	71,1%	28,9%
Poland	46,00	12,00	58,00	0,2%	79,3%	20,7%
Portugal	41,00	11,00	52,00	0,2%	78,8%	21,2%
Puerto-Rico	118,00	13,00	131,00	0,4%	90,1%	9,9%
Scotland	12,00	3,00	15,00	0,0%	80,0%	20,0%
South	69,00	11,00	80,00	0,2%	86,3%	13,8%
Taiwan	27,00	15,00	42,00	0,1%	64,3%	35,7%
Thailand	16,00	3,00	19,00	0,1%	84,2%	15,8%
Trinadad&Tobago	16,00	2,00	18,00	0,1%	88,9%	11,1%
United-States	23.181,00	7.497,00	30.678,00	89,7%	75,6%	24,4%
Vietnam	47,00	4,00	51,00	0,1%	92,2%	7,8%
Yugoslavia	11,00	7,00	18,00	0,1%	61,1%	38,9%
Total	26.008	8.181	34.189,00	100,0%	76,1%	23,9%

Source: Table prepared on Census Income dataset

Thus, I assigned missing values based on the mode 'United-States.' Then, I grouped the categories based on a geographical location:

- 'North': 'United-States,' 'Mexico,' 'Puerto-Rico,' 'El-Salvador,' 'Cuba,' 'Jamaica,' 'Dominican-Republic,' 'Guatemala,' 'Haiti,' 'Nicaragua,' 'Trinidad&Tobago,' 'Outlying-US(Guam-USVI-etc),' and 'Honduras'

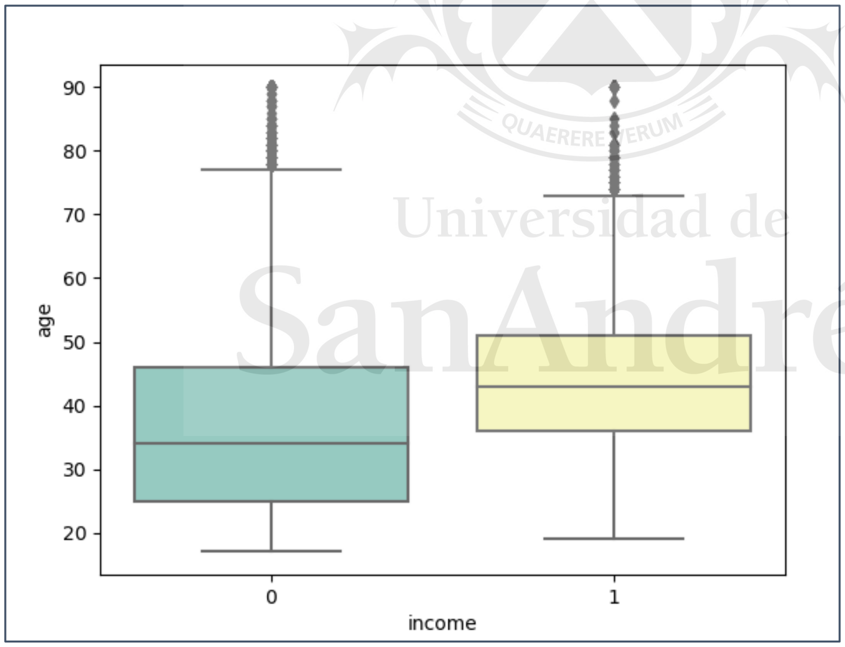
- ‘Europe’: ‘Germany,’ ‘England,’ ‘Italy,’ ‘Poland,’ ‘Portugal,’ ‘Greece,’ ‘France,’ ‘Ireland,’ ‘Yugoslavia,’ ‘Laos,’ ‘Scotland,’ ‘Hungary,’ and ‘Holand-Netherlands’
- ‘Asia’: ‘Philippines,’ ‘India,’ ‘China,’ ‘Japan,’ ‘Vietnam,’ ‘Taiwan,’ ‘Iran,’ ‘Hong,’ ‘Thailand,’ and ‘Cambodia’
- ‘South’: ‘South,’[4] ‘Columbia,’[5] ‘Peru,’ and ‘Ecuador’

In the final dataset, the categorical features are represented by 0/1 dummy variables coded by a 1-of-K scheme.

Then, I continued analyzing the numerical features. The ‘age’ boxplots show that

- 50% of the low income samples age is approximately between 25 and 45 years old. The median is 35 years, and the distribution is skewed.
- 50% of the high income samples age is approximately between 35 and 55 years old. The median is 45 years, and the distribution is moderately skewed.

Figure 4.0.1.7 – The ‘age’ boxplots of low and high income samples

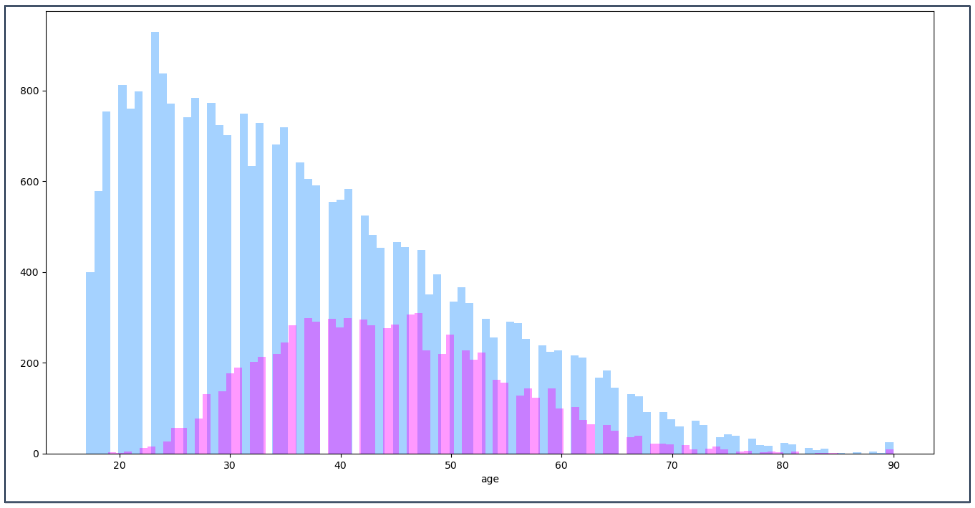


Source: Figure prepared on Census Income dataset

[4] This is not a country by I assumed it should be grouped within ‘South’ category.
[5] Based on the income distribution I assumed it referred to ‘Colombia’.

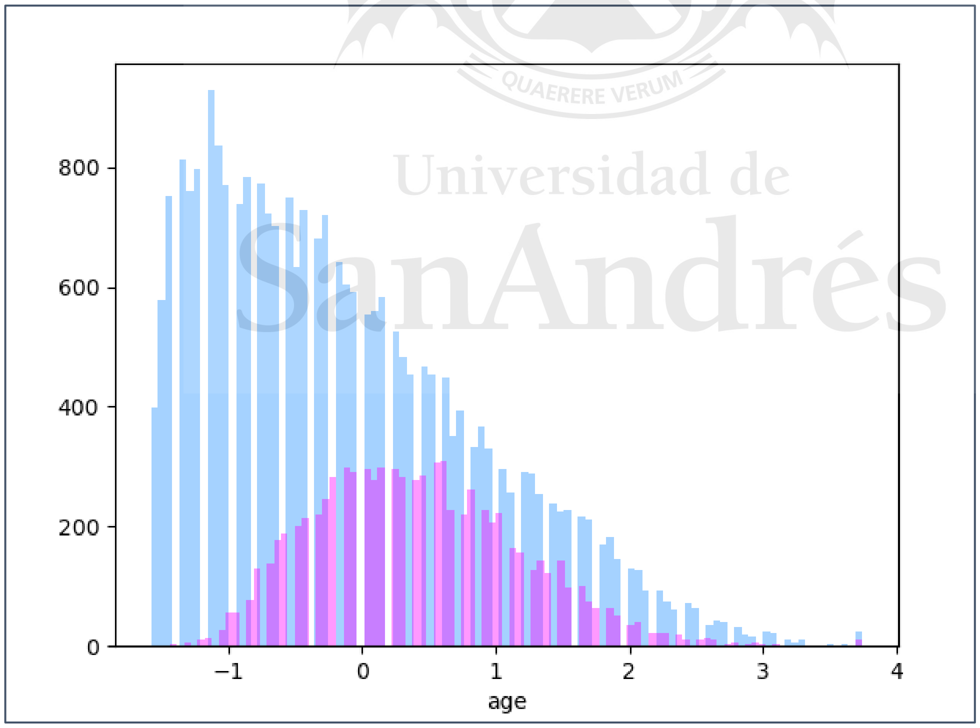
QUAERERE VERUM
Universidad de SanAndrés

Figure 4.0.1.8 – The 'age' Histograms for Low and High Income Samples (before standardization)



Source: Figure prepared on Census Income dataset

Figure 4.0.1.9 – The 'age' Histograms for Low (blue) and High Income (purple) Samples (after standardization)

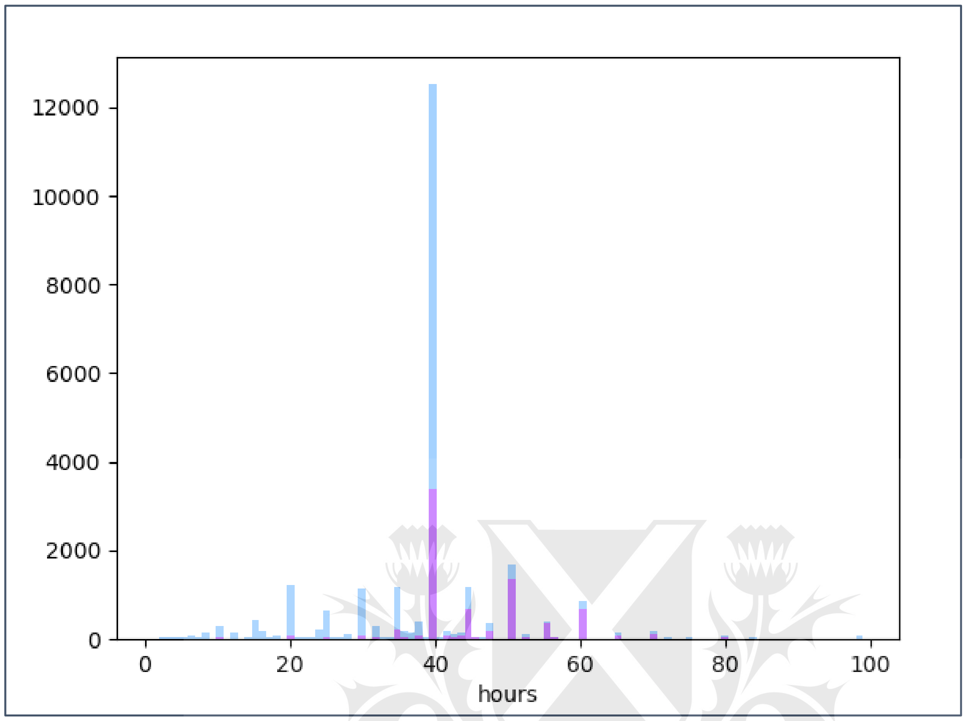


Source: Figure prepared on Census Income dataset

The following two figures show the 'hours-per-week' histograms before and after standardization.

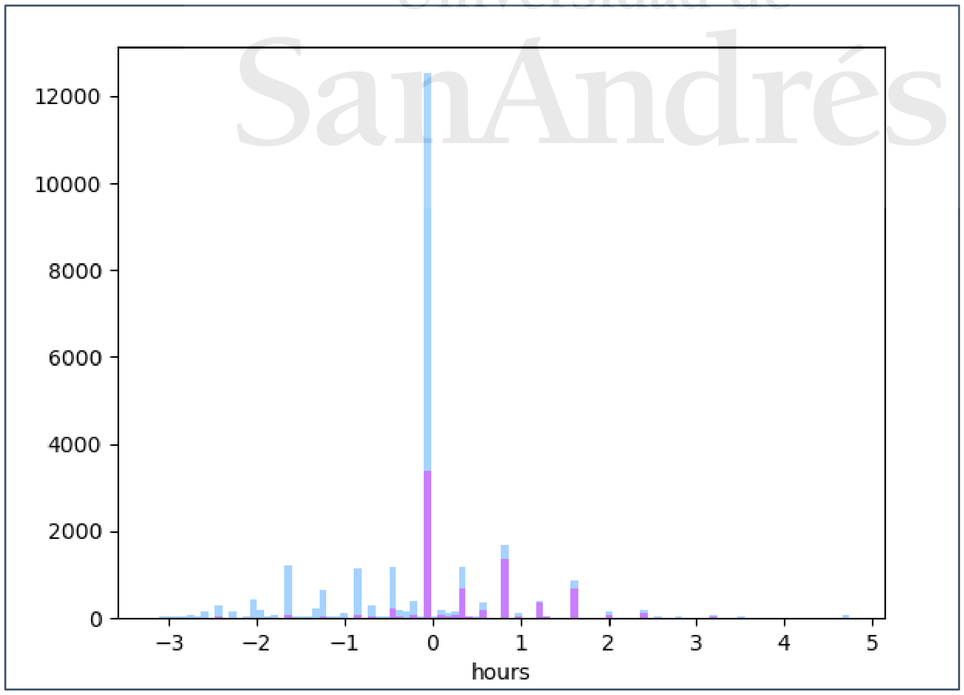
QUAERERE VERUM
Universidad de
SanAndrés

Figure 4.0.1.10 – The 'hours-per-week' Histograms for Low (blue) and High Income (purple) Samples (before standardization)



Source: Figure prepared on Census Income dataset

Figure 4.0.1.11 – The 'hours-per-week' Histograms for Low (blue) and High Income (purple) Samples (after standardization)

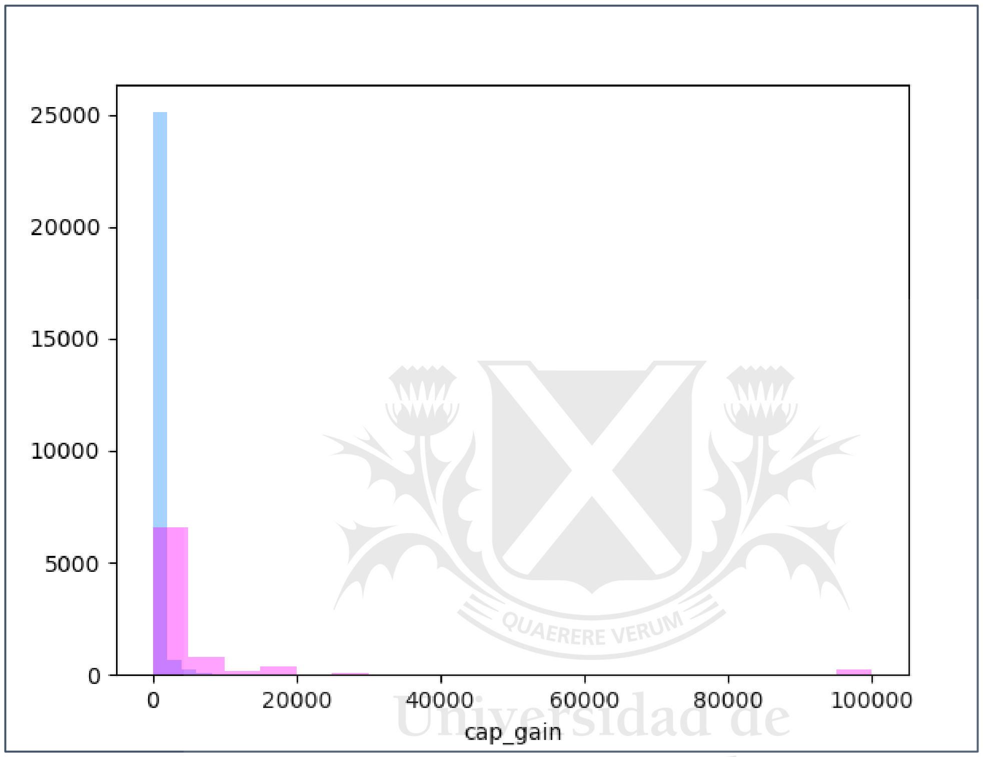


Source: Figure prepared on Census Income dataset

QUAERERE VERUM
Universidad de
SanAndrés

In terms of 'capital-gain' and 'capital-loss,' the following histograms shows that most of the observations has 0 capital loss and gains. In addition, 'capital-gain' contains approximately 200 observations with capital gains around $100 thousand, which could be considered outliers.

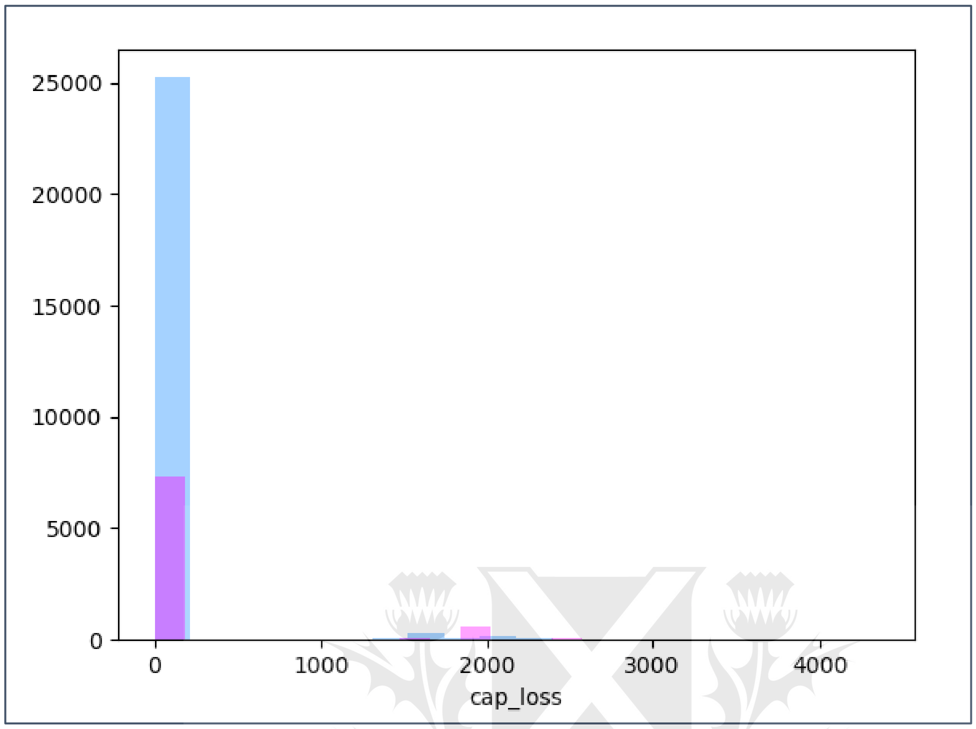
Figure 4.0.1.12 – The 'capital-gain' Histograms for Low (blue) and High Income (purple)



Source: Figure prepared on Census Income dataset

Universidad de SanAndrés

Figure 4.0.1.13 – The 'capital-loss' Histograms for Low (blue) and High Income (purple)

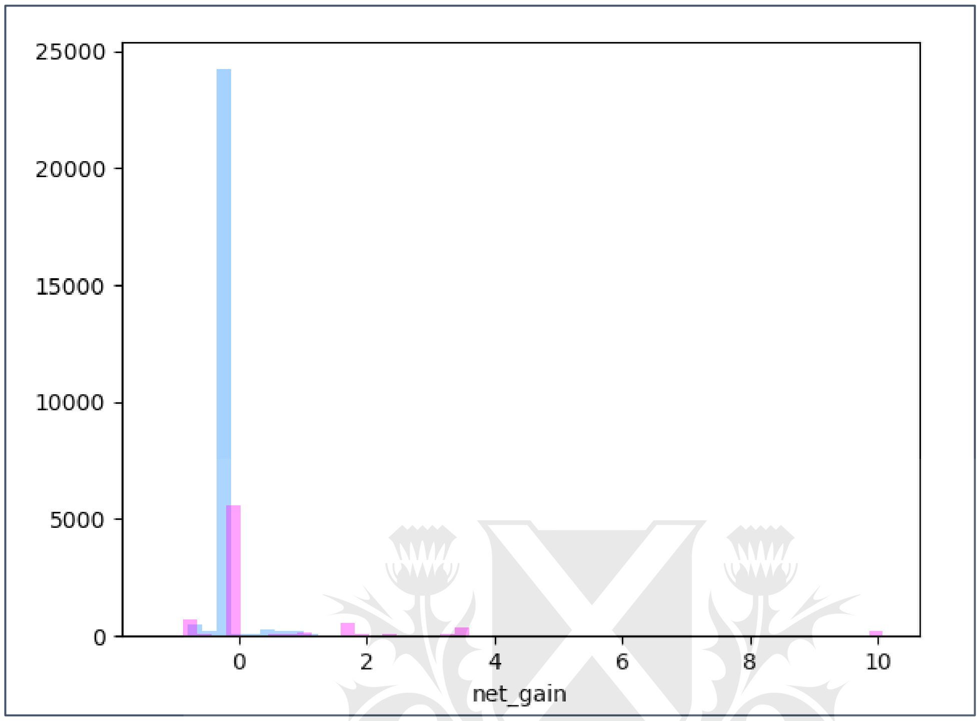


Source: Figure prepared on Census Income dataset

QUAERERE VERUM Universidad de SanAndrés

I did not want to discard these variables and I wanted to minimize the distortion they could produce in the Logistic Regression model. Therefore, I created a variable named 'net_gain,' which reflects the difference between capital gains and losses. Nevertheless, before creating the variable, I imputed the second highest gain (approximately $41 thousand) to the samples that earned $100 thousand. Then, I standardized 'net_gain' and dropped the capital gain and loss features.

Figure 4.0.1.11 – The 'net_gain' histograms for low (blue) and high income (purple) samples (after standardization)



Source: Figure prepared on Census Income dataset

QUAERERE VERUM
Universidad de
SanAndrés

5.0 – Model estimation and Performance Assessment

As I described in Section 4, I used Python's package 'sklearn.model_selection.train_test_split' to split the samples into training and test sets. I specified a random seed equal to 7 and used 30% of the observations for testing purposes and 70% for the training of the models.

To choose each of the parameters of the model and the shrinkage methods, I used Python's package 'sklearn.model_selection.GridSearchCV'. This library performs k-fold cross-validation on the training data. I have built grids, set 5 folds, and used 'roc_auc' as a scoring metric for each parameter and shrinkage method. The algorithm chooses the best combination in the following way:

1) The specified model uses 80% of the training data (since I chose 5 folds) and computes the AUC in the remaining 20% of the training data for each combination it fits.
2) The algorithm repeats step 1 five times.
3) It computes the mean AUC (i.e., the mean of the five AUCs).
4) The best combination is the one that outputs the highest mean AUC.

5.1 – Logistic Regression

To estimate the logistic regression, I have used 'sklearn.linear_model.LogisticRegression' Python's library. To choose the parameters and regularization method, I included in the following terms in the grid-search:

- Both Lasso ('l1') and Ridge ('l2') penalties
- C =[1/10000, 1/5000, 1/1000, 1/500, 1/100, 1/50, 1/10, 1, 5, 10, 50, 100, 500, 1000, 5000, 10000]
- Maximum number of iterations: [100, 200, 300, 400, 500, 600, 700]

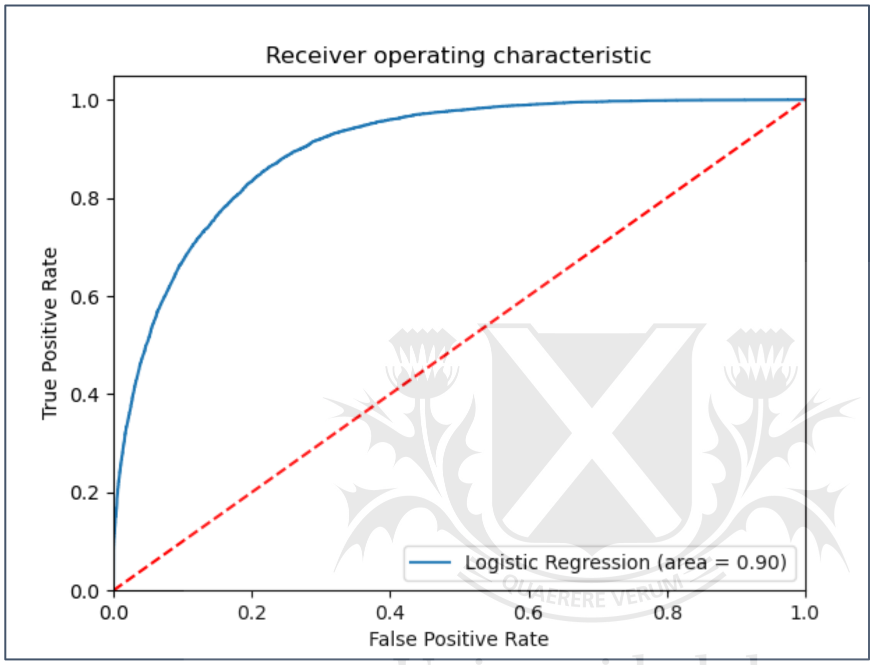
The algorithm I chose for the optimization problem was 'saga', which is a variant of the 'sag' solver and stands for Stochastic Average Gradient Descent. The solver 'saga' can handle both Lasso and Ridge penalties and is a time-efficient solver with large datasets. I fit the logistic regression with an intercept.

The grid search yielded the following results:

- Penalty: Lasso (L1)
- $C = 1$ (which coincided with the default value of the Python's library)
- 300 maximum number of iterations

The Lasso shrinkage parameter λ equals $\frac{1}{C}$. As λ approaches 0, the regularization effect weakens. The grid search output was $\lambda = 1$. Thus, the coefficients of some variables were nulled with this shrinkage parameter. The following two figures show the ROC curves for both the train and test sets.

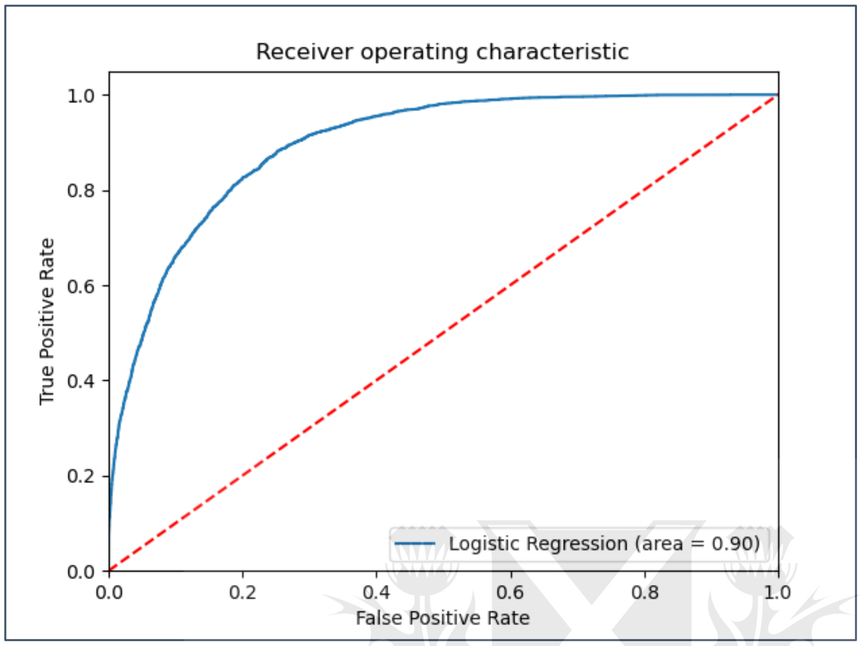
Figure 5.1.1 – ROC for train set



Source: Figure prepared on Census Income dataset

Universidad de SanAndrés

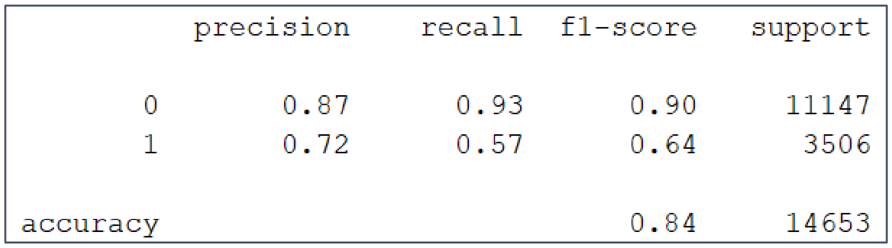
Figure 5.1.2 – ROC for test set



Source: Figure prepared on Census Income dataset

The train and test AUCs are approximately the same. As expected, the Logistic Regression model does not experience variance. Based on the area under the ROC curve, the room for improvement of the challenger models is quite limited, as the value of the high test AUC is approximately 0.90. However, based on each class F1-score, the model accurately predicts low income observations, but it performs poorly when classifying high income observations. The following figure shows the classification report using the 0.5 default threshold.

Figure 5.1.3 – Classification report on test set



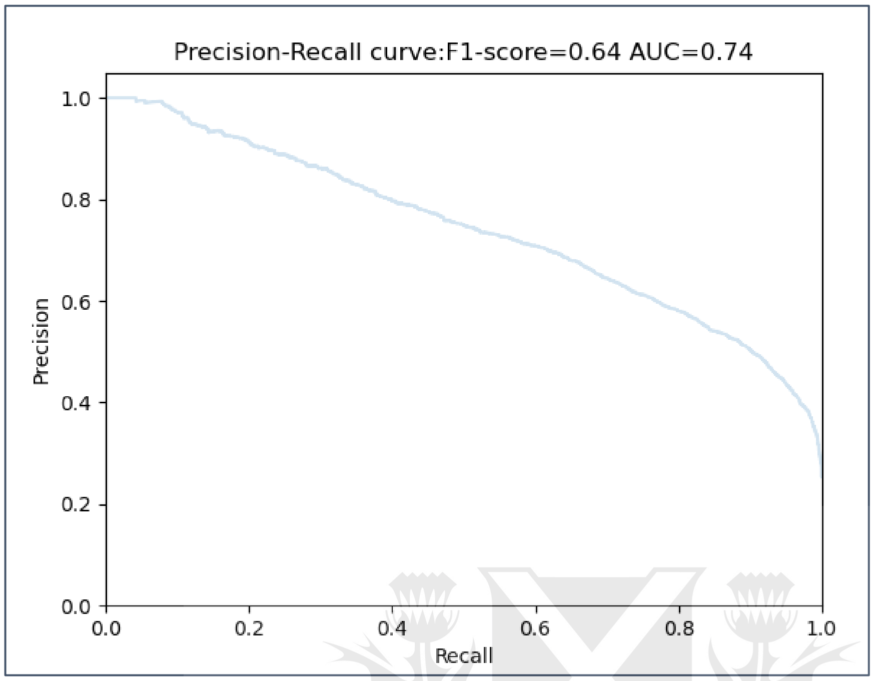
```
              precision    recall  f1-score   support

           0       0.87      0.93      0.90     11147
           1       0.72      0.57      0.64      3506

    accuracy                           0.84     14653
```

Source: Table prepared on Census Income dataset

Since the sample is moderately unbalanced, the performance of the model can also be evaluated with a ‘Precision-Recall’ curve. With an AUC of 0.74, the performance of the model is less optimistic, but the challenger models might improve performance.

QUAERERE VERUM Universidad de San Andrés

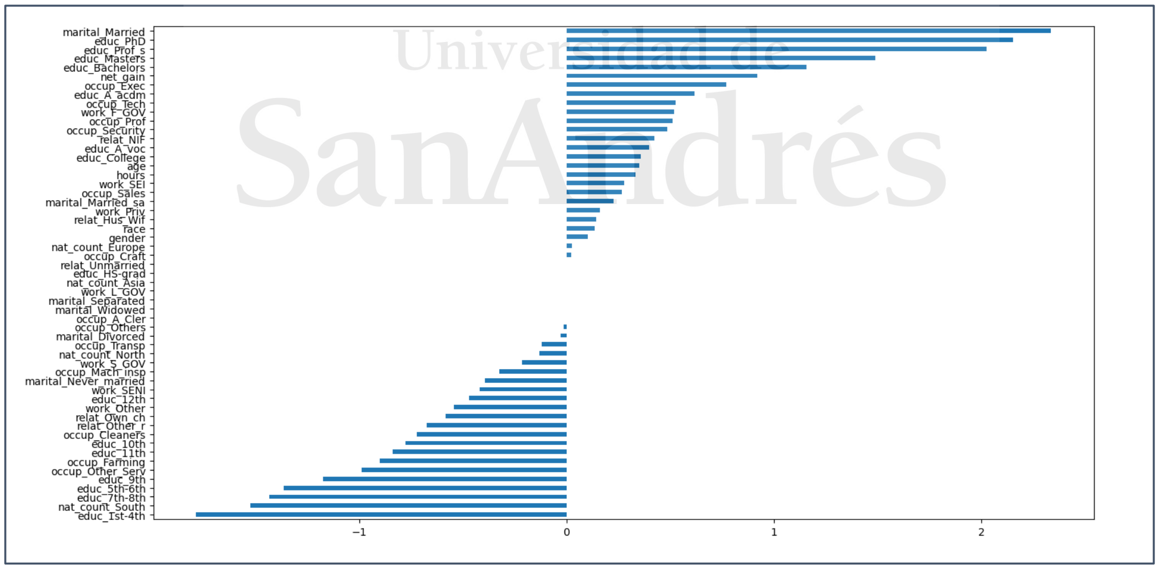
Figure 5.1.4 – Precision-recall curve on test set



Source: Figure prepared on Census Income dataset

The following figure displays the estimated coefficients.

Figure 5.1.5 – Logistic Regression Coefficients



Source: Figure prepared on Census Income dataset

The Lasso penalty nulled the coefficients of ‘relat_Unmarried’, ‘educ_HS-grad’, ‘nat_count_Asia’, ‘work_L_GOV’, ‘marital_Separated’, ‘marital_Widowed’, and ‘occup_A_Cler’ features. I consider that the signs and values of the coefficients that were not nulled are appropriate. For example,

QUAERERE VERUM
Universidad de SanAndrés

‘age’ and ‘hours’ have positive effects on the probability of earning more than $50 thousand per year. On the one hand, a higher level of education, being married, and professional skills increases the probability of earning more than $50 thousand per year. On the other hand, lower levels of education, being unmarried, and having low-skilled occupations decreases the probability.

5.2 – Gradient Boosting

To estimate the Gradient Boosting model, I have used Python’s library ‘sklearn.ensemble.GradientBoostingClassifier’. First, I describe the tree-specific parameters:

- ‘min_samples_split’: it is the minimum number of samples that the algorithm will require for splitting a node.
- ‘min_samples_leaf’: it is the minimum number of samples that the algorithm will include in the terminal nodes.
- ‘max_depth’: it is the maximum depth of each boosting tree.
- ‘max_features’: it is the maximum number of features that the tree will consider to look for the best split.

The gradient boosting specific parameters are the following:

- ‘learning_rate’: it is the rate that shrinks the contribution of the weak learners in the ensemble.
- ‘n_estimators’: it is the number of trees that are included in the ensemble. There is a trade-off between this parameter and the learning rate: lowering learning rates requires a larger number of trees for increasing the scoring metric.

These parameters are used to control overfitting.

Table 5.2.1 – Relation between parameters and overfitting

	Controls for overfitting with:
min_samples_split	High values
min_samples_leaf	High values
max_depth	low values
max_features	low values
learning_rate	low values
n_estimators	low values

Source: Table prepared on Gradient Boosting model research

To estimate the model, I used the following approach:

Universidad de San Andrés

1) I set initial tree-specific parameters.
2) I chose a high learning rate of 0.1 and found the optimal number of trees for that learning rate and the initial tree-specific parameters from step 1.
3) I recalibrated the tree-specific parameters.
4) I lowered the learning rate 0.01 and increased the number of trees.

I used a high learning rate to choose a conservative (with respect to overfitting) tree-specific parameters. If I had chosen the tree parameters using the 0.01 learning rate, they would have been higher than the ones I used in the final model estimation.

I set the following initial tree-specific parameters (step 1):

- 'min_samples_split' = 500 (approximately 1.5% of the total number of training observations).
- 'min_samples_leaf' = 100.
- 'max_depth' = 9.
- 'max_features' = 'sqrt' (which is the square root of the total number of features in the dataset).

Using these initial values, I performed a grid search to determine the initial optimal number of trees (please refer to step 2). I tested values from 200 to 1000 taking steps of 20. The 'n_estimators' output was 460.

Next, I recalibrated the tree-specific parameters (step 3). I started with 'max_depth' and 'min_samples_split', since these are the most sensitive (tree-specific) parameters for the model. For 'max_depth' I tested values from 5 to 25 taking steps of 2. I applied the same procedure to 'min_samples_split' with values from 3000 to 6000 taking steps of 200. The output values were 13 and 4800 for 'max_depth' and 'min_samples_split', respectively.

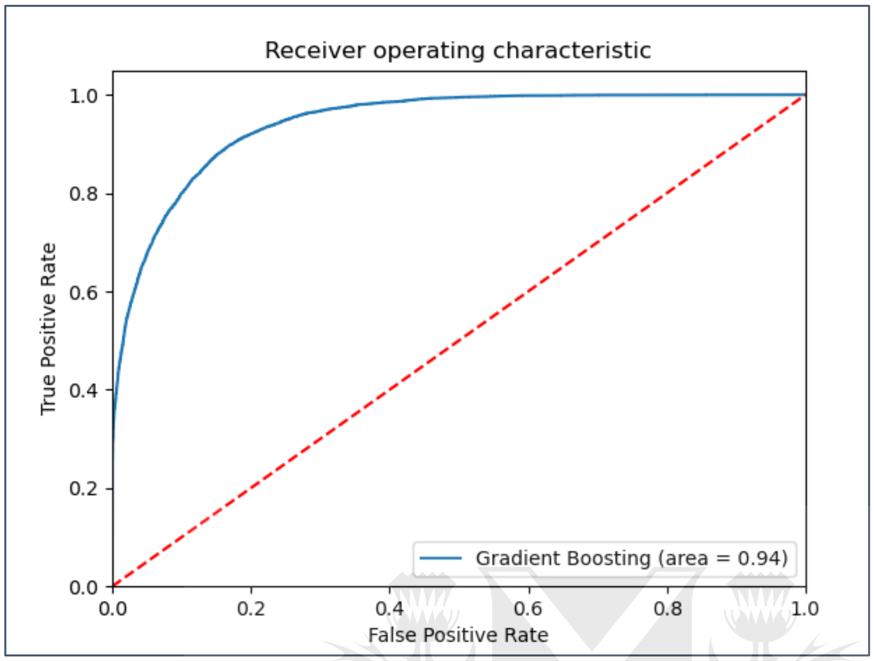
Then, I tuned 'min_samples_leaf' testing values from 60 to 151 taking steps of 10. The output value was 70. I also checked 'max_features' from 7 to 21 taking steps of 2. The output was 15.

Regarding the Gradient Boosting parameters, I decreased the learning rate to 0.01. To choose the number of trees with the updated learning rate, I built a grid to try with different numbers of trees [500, 2500, 3500, 4500, 5500, 6500, 7500]. The output value was 4500.

The following two figures show the ROC curves for both the train and test sets.

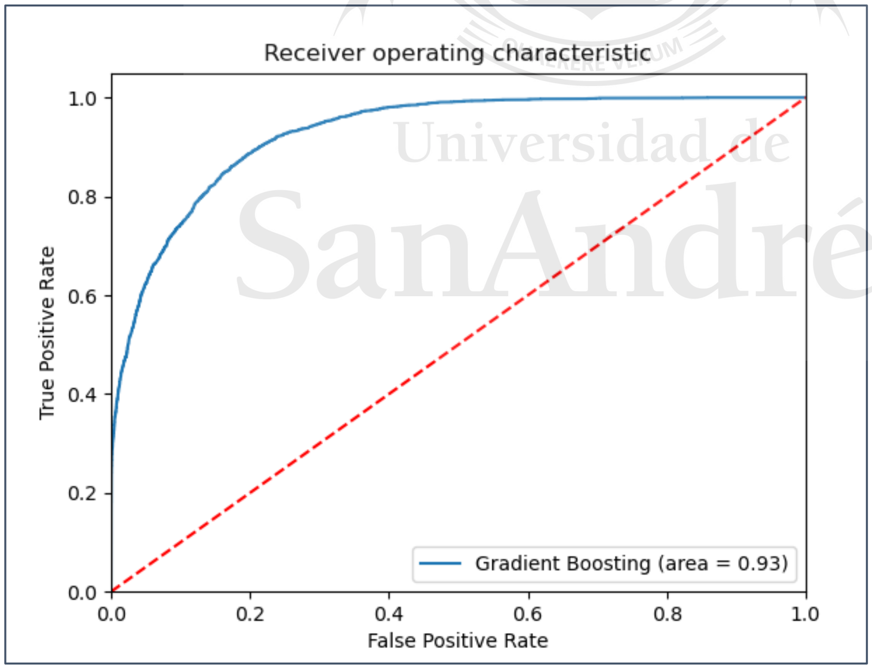
Universidad de San Andrés

Figure 5.2.1 – ROC for train set



Source: Figure prepared on Census Income dataset

Figure 5.2.2 – ROC for test set



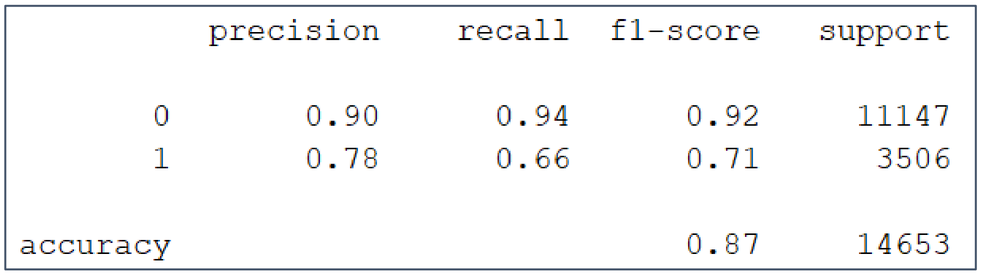
Source: Figure prepared on Census Income dataset

The AUCs between train and test datasets are similar and I do not consider the model has a variance issue. Based on the 'F1-score,' the Gradient Boosting model is better than the Logistic

Universidad de SanAndrés

Regression model in terms of predicting new high-income observations. However, as in the baseline model, the gap between precision and recall in both classes is significant.

Table 5.2.2 – Classification report on test set

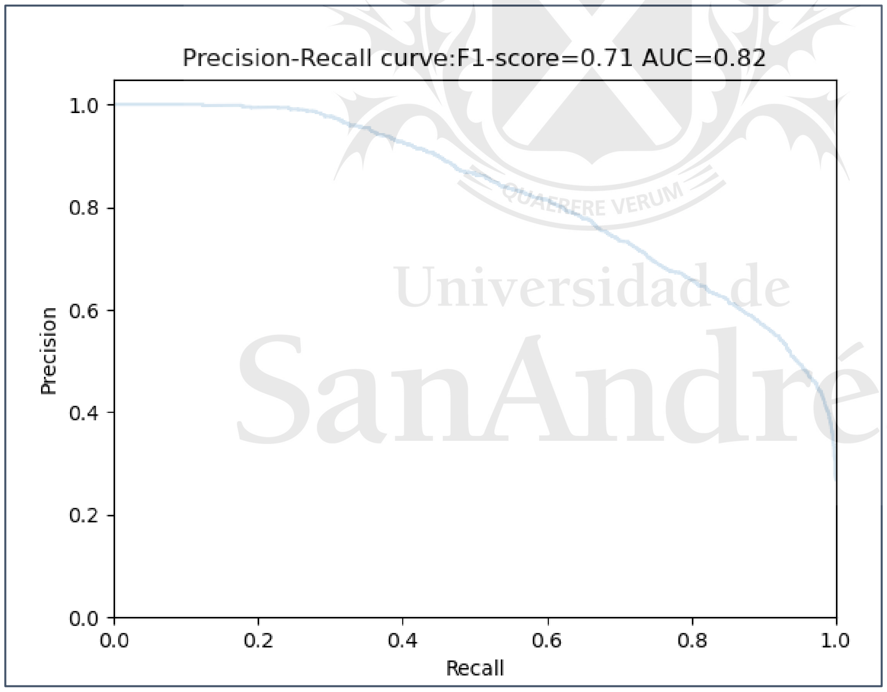


```
              precision    recall  f1-score   support

           0       0.90      0.94      0.92     11147
           1       0.78      0.66      0.71      3506

    accuracy                           0.87     14653
```

Source: Table prepared on Census Income dataset

The area under the 'Precision-Recall' curve reflects the improvement with respect to the baseline model (0.82 against 0.74).

Figure 5.2.3 – 'Precision-Recall' curve on test set



Source: Figure prepared on Census Income dataset

QUAERERE VERUM
Universidad de SanAndrés

5.3 – AdaBoost

To estimate the AdaBoost model, I have used Python's library 'CatBoost'. I set the following parameters:

- 'min_data_in_leaf'= 4800
- 'Loss_function'= 'Logloss'

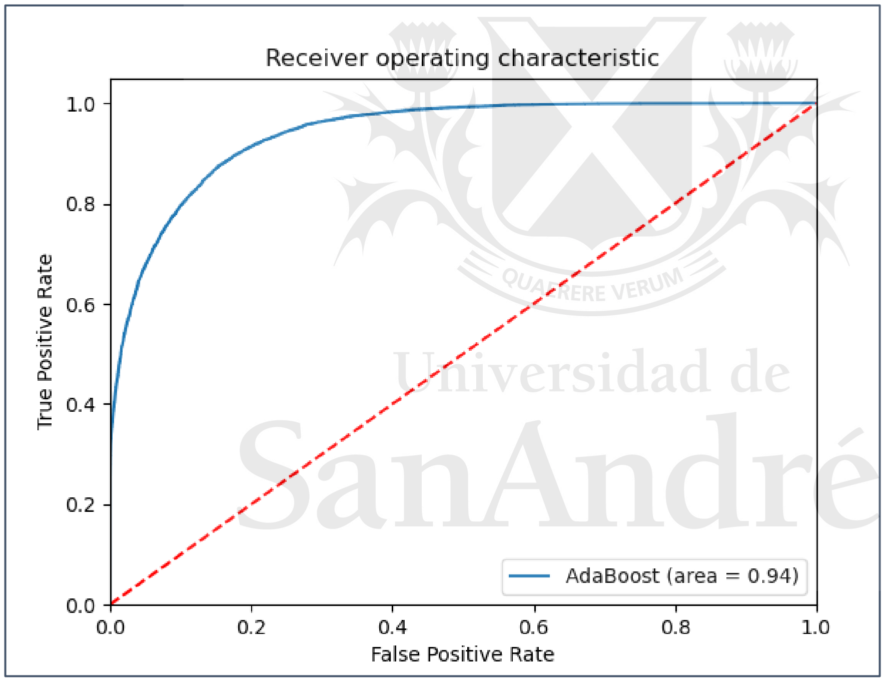
- 'Learning_rate'= 0.1
- 'depth' = 5

I performed grid search to determine the initial optimal number of trees. I tested values from 100 to 1500 taking steps of 50. The 'n_estimators' output was 700.

Next, I recalibrated 'depth'. I tested values from 1 to 10 taking steps of 2 and the output was 5. Finally, I decreased the learning rate to 0.01 and performed grid search to determine the optimal number of trees. I tested values from 5000 to 10000 taking steps of 500 and the output was 7000.

The following two figures show the ROC curves for both the train and test sets.

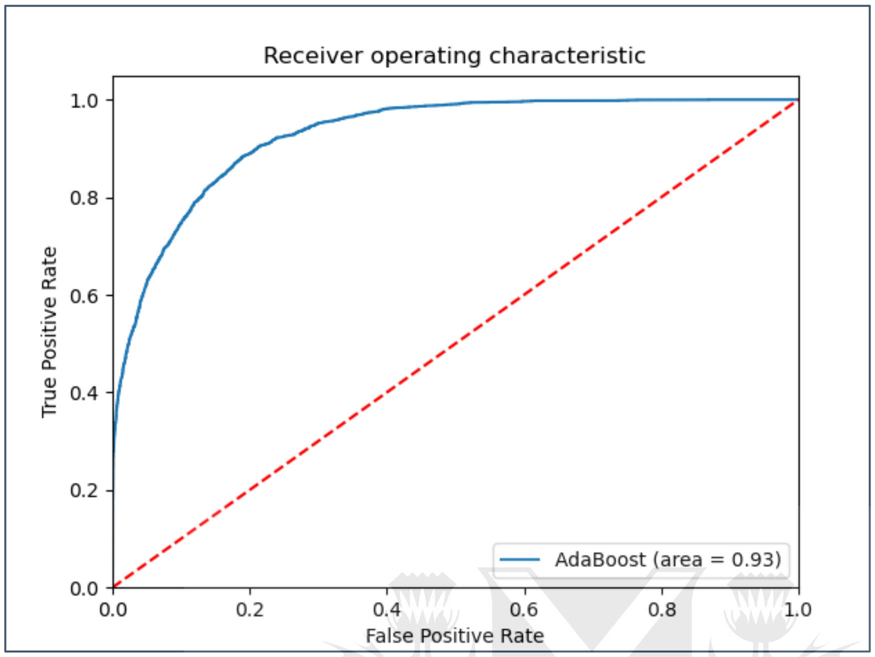
Figure 5.3.1 – ROC for train set



Source: Figure prepared on Census Income dataset

QUAERERE VERUM
Universidad de SanAndrés

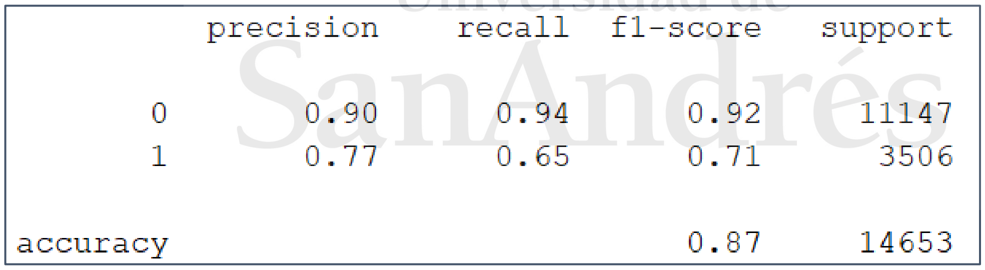
Figure 5.3.2 – ROC for test set



Source: Figure prepared on Census Income dataset

The AUCs between train and test datasets are similar and I do not consider the model has a variance issue. Based on the 'F1-score,' the AdaBoost and Gradient Boosting model results are similar.

Table 5.3.1 – Classification report on test set



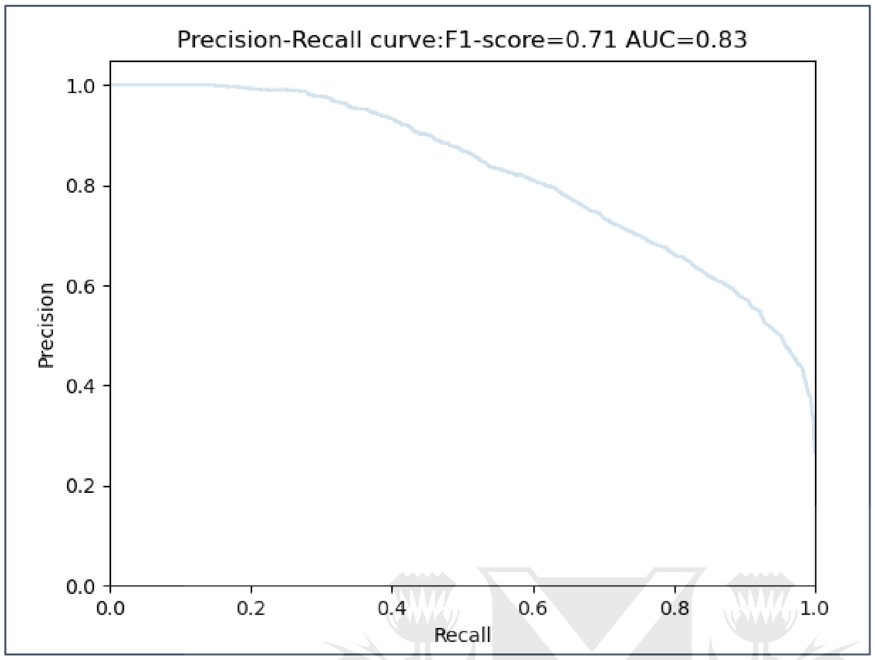
```
              precision    recall  f1-score   support

           0       0.90      0.94      0.92     11147
           1       0.77      0.65      0.71      3506

    accuracy                           0.87     14653
```

Source: Table prepared on Census Income dataset

The area under the 'Precision-Recall' curve reflects a slight improvement with respect to the Gradient Boosting model (0.83 against 0.82).

QUAERERE VERUM
Universidad de
SanAndrés

Figure 5.2.3 – 'Precision-Recall' curve on test set



Source: Figure prepared on Census Income dataset

5.4 – Random Forest

To estimate the Random Forest model, I have used Python's library 'sklearn.ensemble.RandomForestClassifier'. I set the following initial parameters:

- 'criterion': 'Gini' as the impurity function to measure the quality of a split
- 'bootstrap': 'True' for the algorithm to build the bootstrap samples with replacement
- 'n_estimators': 500 trees
- 'min_samples_split': 500
- 'max_leaf_nodes': 100

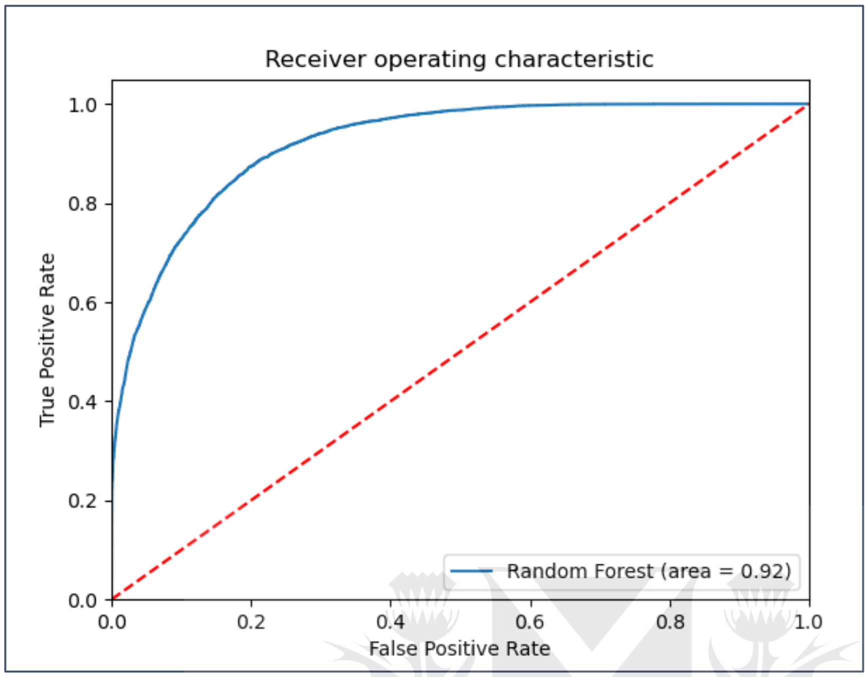
Next, I calibrated 'max_depth' and 'max_features'. Regarding 'max_depth' I tested values from 10 to 70 taking steps of 5, and the output was 45. I followed the same procedure for 'max_features' with values from 7 to 21 taking steps of 2, and the output value was 17.

Then, I calibrated 'max_leaf_nodes' testing values from 10 to 400 taking steps of 10 and the output was 130. Finally, I recalibrated the number of trees using the following grid [100, 200, 300, 500, 1000, 2000]. The output value was 500.

The following two figures show the ROC curves for both the train and test sets.

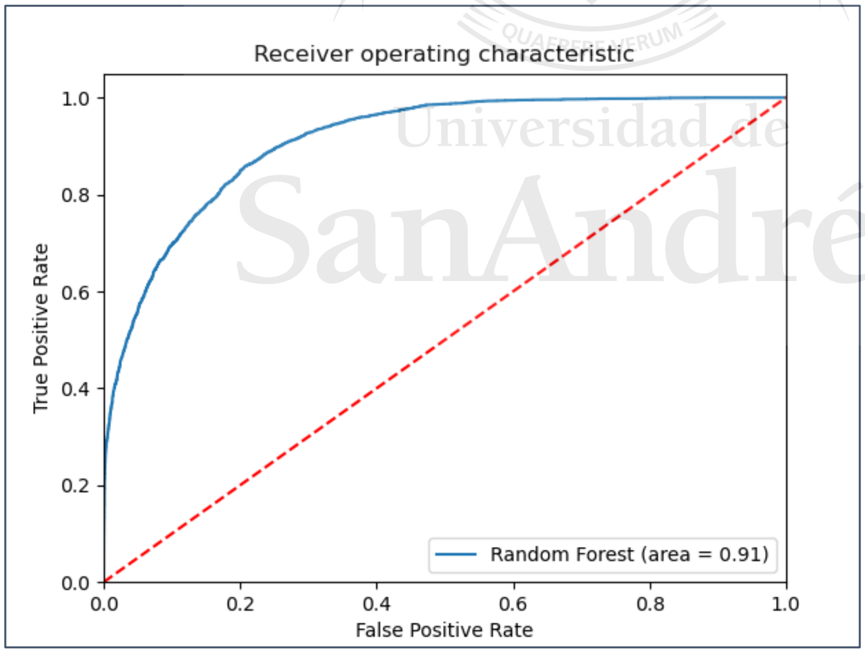
QUAERERE VERUM Universidad de SanAndrés

Figure 5.4.1 – ROC for train set



Source: Figure prepared on Census Income dataset

Figure 5.4.2 – ROC for test set

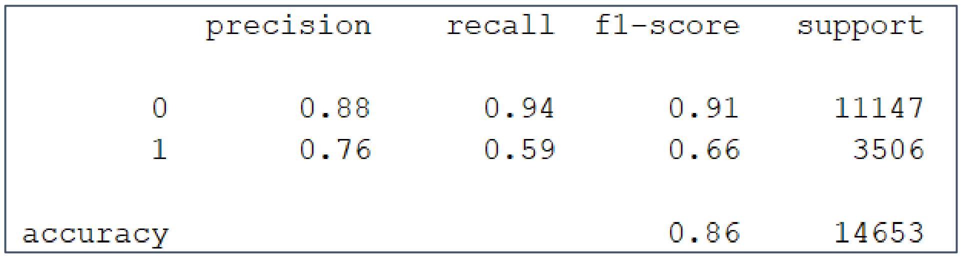


Source: Figure prepared on Census Income dataset

The AUCs between train and test datasets are similar. As in the Logistic Regression and the Gradient Boosting, the performance of the Random Forest model is less optimistic when assessing the 'Precision-Recall' curve.

Universidad de SanAndrés

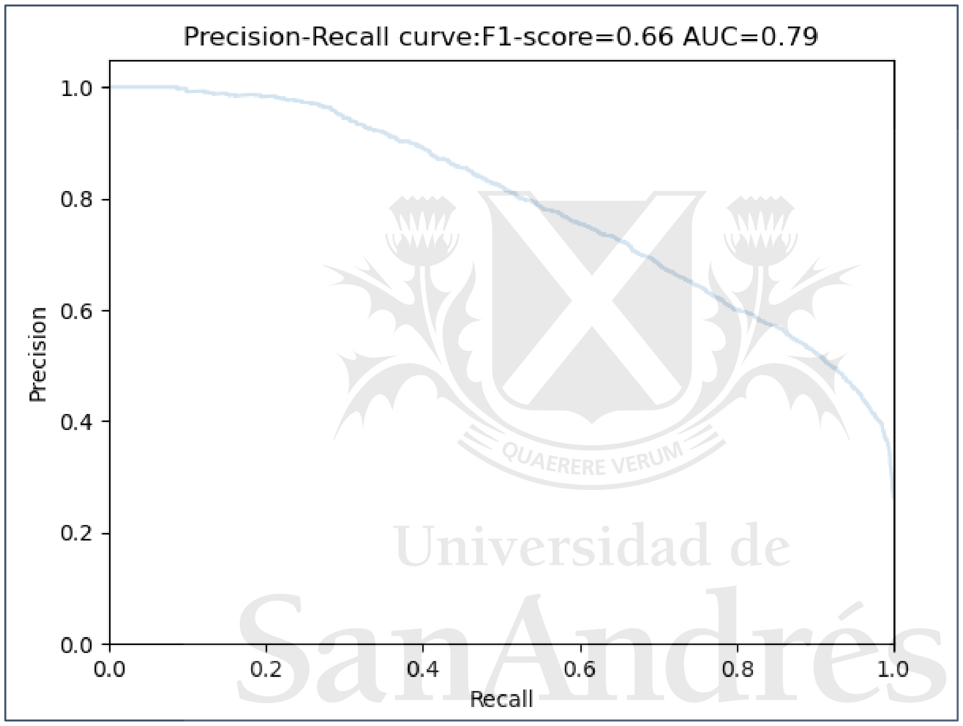
Table 5.1.1 – Classification Report



```
              precision    recall  f1-score   support

           0       0.88      0.94      0.91     11147
           1       0.76      0.59      0.66      3506

    accuracy                           0.86     14653
```

Source: Table prepared on Census Income dataset

Figure 5.4.3 – Precision-recall curve



QUAERERE VERUM
Universidad de San Andrés

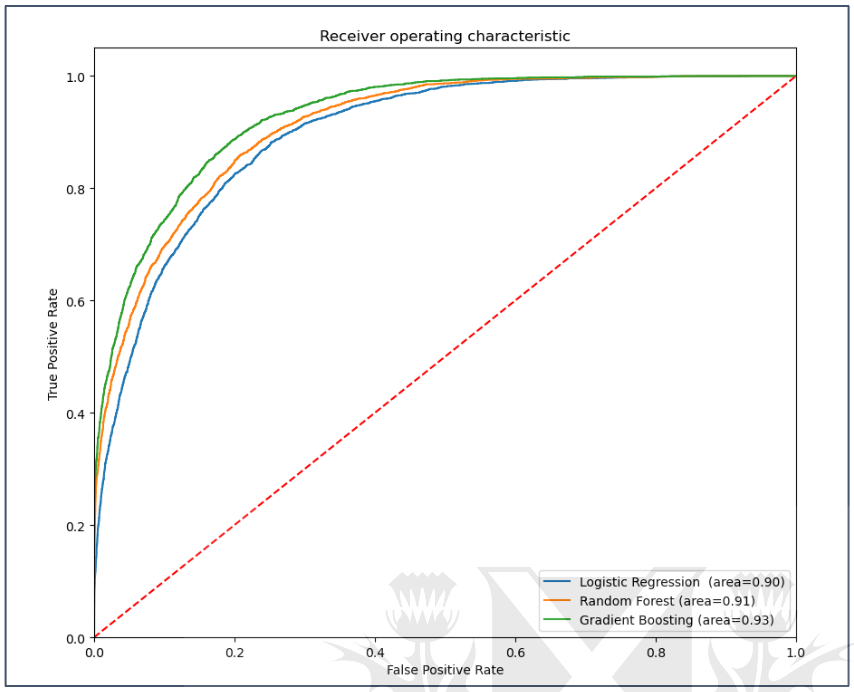
Source: Figure prepared on Census Income dataset

5.5 – Model Comparison

Based on the ROC AUC, the baseline Logistic Regression model I have estimated shows a satisfactory performance (with test AUC of 0.9). The challenger models improved the performance with the Gradient Boosting model showing the highest AUC. The following figure shows the ROC curves of the three models. The Baseline model set a high AUC floor so I did not expect a significant increase on the challenger model[6].

[6] I did not include the AdaBoost model because the results were similar to the Gradient Boosting model

Figure 5.5.1 – ROC curve comparison

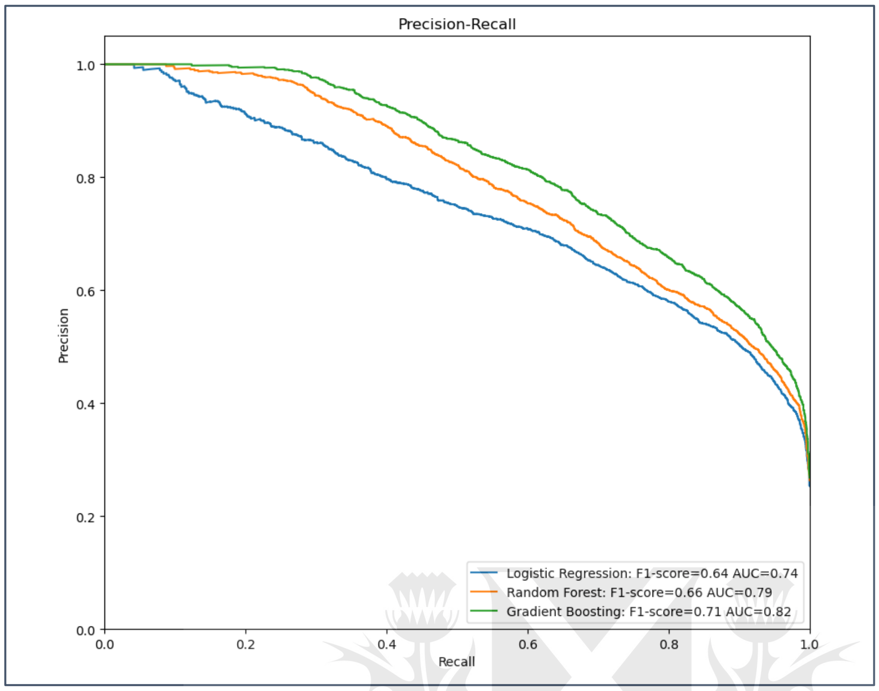


Source: Figure prepared on Census Income dataset

Analyzing the 'Precision-Recall' curves and F1-scores, all the models predict low income samples satisfactorily, but their performance is poor when predicting high income observations. The following figure shows the 'Precision-Recall' curves of the three models. In this case, the Gradient Boosting model outperforms the Logistic Regression.

Universidad de SanAndrés

Figure 5.5.2 – 'Precision-Recall' curves comparison

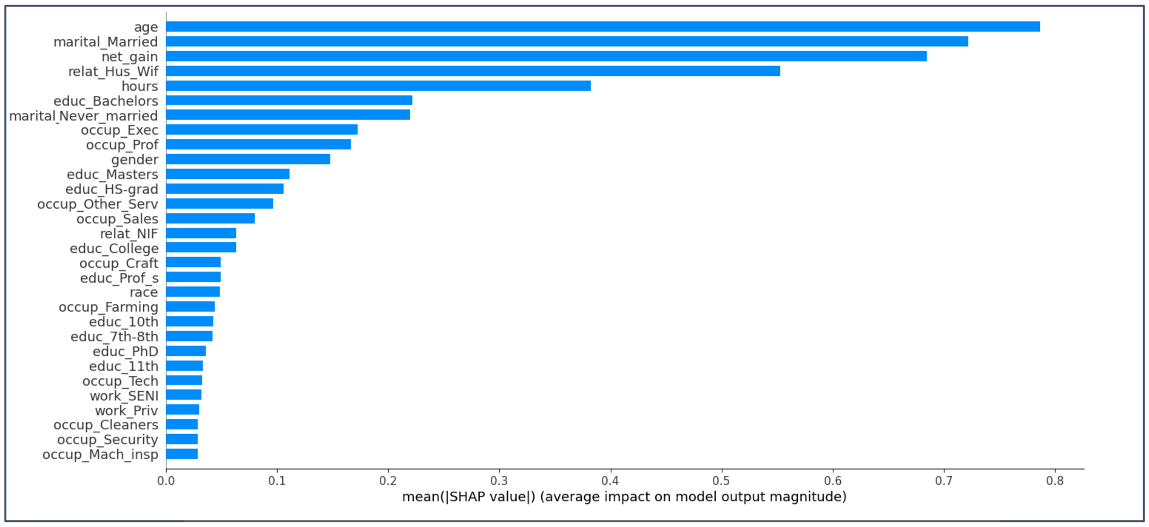


Source: Figure prepared on Census Income dataset

The coefficients of the logistic regression model are aligned with the empirical patterns observed. The following two figures show the importance of the feature I estimated using 'SHAP' Python's Library for the Gradient Boosting Model (I only included the thirstiest and most important features).

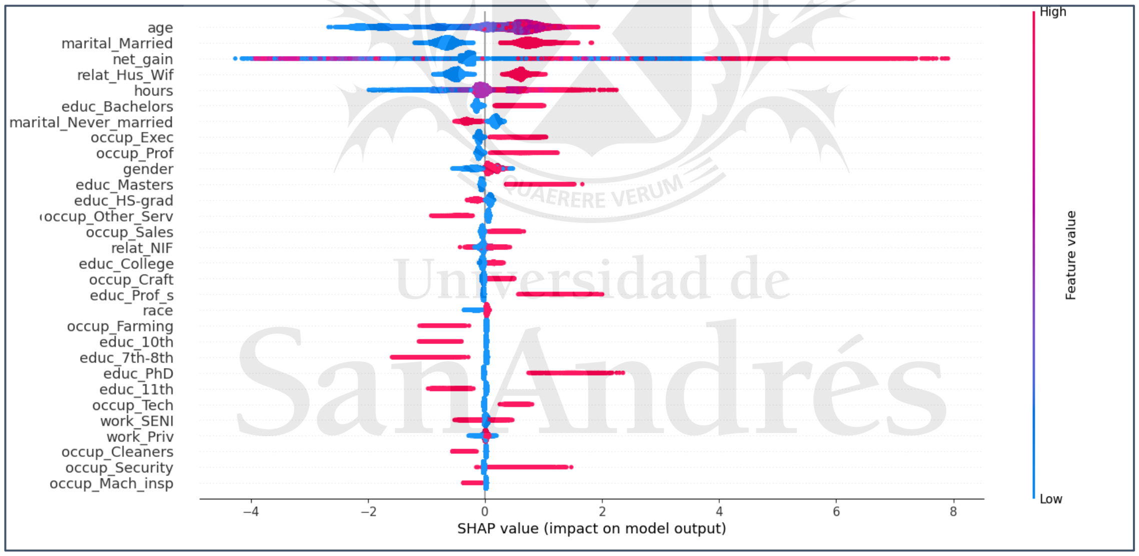
Universidad de SanAndrés

Figure 5.5.3 – Average impact on Gradient Boosting model output



Source: Figure prepared on Census Income dataset

Figure 5.5.4 – SHAP value impact on Gradient Boosting model output



Source: Figure prepared on Census Income dataset

The most important features of the Gradient Boosting model are age, hours, net gain, marital status, professional skills, and gender. The logistic regression coefficient for gender was low, but this lies in the fact that there are other features correlated with gender. In fact, this dataset is very famous in the 'Fair Lending' field in which females are the unfavorable class (since most of the high income people are males) and the correlation between gender with other features is known as 'red-lining effect' (Toshihiro Kamishima 2012). Figure 5.5.4 shows that a higher level of education (e.g., Bachelors', Masters', PhD, and professional schools), and skilled professions (e.g., executives, professionals) are important drivers in determining if incomes exceed $50 thousand per year.

QUAERERE VERUM
Universidad de
SanAndrés

6.0 – Concluding remarks

Before training machine learning algorithms, a working plan should be designed. I used the United States Census Income dataset and clearly stated my purpose before fitting the models: I wanted to find the best classification model while maintaining a balance between training and interpretability efforts. In the data preprocessing field, I could find clear and intuitive patterns in the categorical features: the incomes of individuals with high educational levels or skilled professions who are married exceed $50 thousand per year. Regarding numerical features, the wealthiest people were over 30 years and worked over 40 hours per week. Performing a deeply feature engineering process and adopting the approaches required to deal with unbalanced samples (the Census Income was moderately unbalanced) was out of the scope of my thesis. I imputed missing values, grouped categories for some of the categorical features, rescaled age and hour-per-week continuous variables, and transformed capital gain and loss features into a single variable. I fitted a baseline Logistic Regression model. Based on the area under the ROC curve the performance of the model was acceptable, leaving the challenger models little room for improvement. However, the Logistic Regression model performance was less optimal to classify high income observations. Then, I trained tree-ensemble challenger models, which slightly improved the area under the ROC curve but were more appropriate for classifying high income observations. In general, the feature importance of the Gradient Boosting model was in line with the coefficients of the Logistic Regression.

The trend in data mining is to train more sophisticated algorithms. However, even if training time (or computational requirements) was not an issue, I would still start by fitting a relatively simple model and then move to a more complex one, but not to the most complicated one. The working plan might not be linear. After fitting a baseline model or even a relatively more sophisticated one, new insights may suggest moving a step back to the feature engineering field. Considering the balance between training and interpretability efforts and the patterns I have observed in the data, I would choose and improve the Logistic Regression model for the Census Income dataset.

Universidad de San Andrés

References

Amit, Yali, and Geman, Donald. "Shape Quantization and Recognition with Randomized Trees." Massachusetts Institute of Technology, 1997.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Breiman, Leo. "Bagging Predictors." University of California, September 1994.

—. "Random Forests." University of California, January 2001.

Breiman, Leo, Friedman, Jerome, Olshen, Richard, and Stone, Charles. 1984. *Classification and Regression Trees*. Taylor & Francis Group, LLC.

Fawcett, Tom. "An introduction to ROC analysis." Palo Alto: Institute for the Study of Learning and Expertise, ELSEVIER, December 19, 2005.

Freund, Yoav, and Schapire, Robert E. "Experiments with a New Boosting Algorithm."AT&T Laboratories, 1996.

Freund, Yoav, and Schapire, Robert E. "A Short Introduction to Boosting." AT&T Labs - Research, September 1999.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. "Additive Logistic Regression: A Statistical View of Boosting." The Annals of Statistics, 2000.

Friedman, Jerome. "Greedy Function Approximation: A Gradient Boosting Machine." The Annals of Statistics, Vol. 29, No. 5, 1189–1232, 2001.

Gareth, James, Witten, Daniela, Hastie, Trevor, and Tibshirani, Robert. 2013. *An Introduction to Statistical Learning with Applications in R*. New York: Springer.

Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. 2013. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Ho, Tin Kam. "Random Decision Forests ." AT&T Bell Laboratories, 1995.

Kohav, Ronny, and Becker, Barry. UCI Machine Learning Repository (California: University of California). May 1, 1996. https://archive.ics.uci.edu/ml/datasets/adult (accessed 2020).

Kuhn, Max, and Johnson, Kjell. 2013. Applied Predictive Modeling. Springer.

Mason, L., J. Baxter, P. Bartlett, and M. Frean. Boosting Algorithms as Gradient Descent. MIT Press, 2000.

Murphy, Kevin. 2012. *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press.

Schapire, Robert E. "The Strength of Weak Learnability." Kluwer Academic Publishers, Boston, 1990.

Universidad de San Andrés

Sosa Escudero, Walter. Análisis ROC. August 24, 2020.
https://www.youtube.com/watch?v=_yzJQ-vp5uI (accessed 2020).

—. "Tópicos de Econometría Aplicada (Notas de Clase) Trabajo Docente Nro. 2 Universidad Nacional de La Plata." Septiembre 1999.

Kamishima, Toshihiro, Okaho, Shotaro, Asoh, Hideki, and Sakuma, Jun. "Fairness-aware Classifier with Prejudice Remover Regularizer." 2012.

Weinberger, Kilian. 17: Decision Trees. July 2018.
http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote17.html (accessed 2020).

—. 19: Boosting. July 2018.
http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote19.html (accessed 2020).

—. Lecture 12: Bias-Variance Tradeoff. July 2018.
http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote12.html (accessed 2020).

QUAERERE VERUM
Universidad de
SanAndrés