School Dropouts Prediction

Pablo Dinamarca

23/6/2020

Contents

Chapter 1

Introduction

My name is Pablo Dinamarca and I am a university student in Paraguay. I am currently working on a Public Policy project to improve the educational and health system in Paraguay.

Data science is entering different sectors within public entities. In the case of Paraguay, prediction models can be applied for the sector of the economy, health, politics, and many other fields, which still do not have the resources to prepare them. Undoubtedly, having statistics based on information is key to planning different policies in a more efficient framework or an improvement in the management of State resources, from an educational reform to a change in current monetary policy.

When we study data from developing countries, it is normal to have a limited amount of resources available. Normally the information is usually scarce and incomplete and makes it difficult to analyze them. With this in mind, focus this work on the **prediction of school dropouts** from the data available on the DGEEC page, which is the General Directorate of Statistics, Surveys and Censuses of Paraguay.

This entity generates and integrates statistical information and makes it available for free use of the available data. In our case **We will take the defections of the year 2017 as a base and we will try to predict those of 2018.** Available in the DGEEC.

We will make predictions of school dropouts applying statistical models and using the caret's package to create the algorithms in RStudio. Below, I leave you some interesting statistics of Paraguay with current data.

Paraguay is one of the smallest countries in Latin America with only 406,752 km² of territory approx. and with 7,252,672 inhabitants approx. The population of women is approximately 3,599,516. and men 3,653,156 approx. 62.5% of the population lives in the urban area and 37.5% in the rural area.

When analyzing the population by age groups, there are 2,096,464 girls, boys and adolescents (0-14 years old), 1,954,150 young people (15-29 years old), 2,715,396 adults (30-64 years old) and 486,662 adults. older (65 years and over).

Life expectancy at birth will be 74.7 years old: 77.7 years old for women and 71.8 years old for men. Data from the dgeec.

1.1 REG02_EPHC_ANUAL Dataset

The database that we will use is called REG02_EPHC_ANUAL_2017 for predictors variables and REG02_EPHC_ANUAL_2018 for validation set, which has a Spanish dictionary with the information of the parameters and the variables it contains.

1. The EPHC_2017 count with 73643 observation and 218 variables.
2. The EPHC_2018 count with 55453 observation and 218 variables.

Of the 218 variables, we will take some that we consider most relevant to create our Machine Learning model. Keep in mind that the dataset is completely in Spanish so we advise you to have a web tool such as the Google Translator at hand.

1.2 Model Evaluation

To evaluate any machine learning algorithms we must compare the predicted value with the actual result and measure the error between them using a loss function.

In our case, the EPHC dataset is unbalanced (as we will see later), so we must use a function that minimizes the error taking that into account.

The function that we will use is called **F1-Score** and the idea is that the precision of the models must be as close as possible to 1.

1.3 Process and workflow

Our work process will consist of the following steps:

1. Data Preparation: Download, analyze, modify and prepare the data to be processed and modeled.
2. Data exploration: explore the data by creating graphs, tables and stradistic summaries to understand the characteristics, the relationship and the predictors it contains.
3. Data Cleaning: In this section, unnecessary data is removed and data sets are ready to start modeling.

4. Data analysis and modeling: Models are created from the modified data sets that are evaluated with the F1-Score and the final results are presented.

5. Final report: the report, its implications and possible future uses are exposed.

Chapter 2

Data preparation

In this section, we download and divide the dataset for use in the analysis. Please note that we will use two datasets.

1. EPHC_2017 for training data.
2. EPHC_2018 for test data.

2.1 Load the data

We first downloaded the data sets from the DGEEC website. The training set will be called "Data" with the 2017 data and the validation set will be called "Validation" with the 2018 data.

We select some variables that seem relevant to us to generate our models. Note that the number of variables will depend on the computing capacity, so we only selected 9 in this project of which 7 we use to predict and 2 to create the variable to predict `Drop_out`.

```r
# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",
                                     repos = "http://cran.us.r-project.org")
if(!require(hrbrthemes)) install.packages("data.table",
                                          repos = "http://cran.us.r-project.org")
if(!require(haven)) install.packages("lubridate",
                                     repos = "http://cran.us.r-project.org")
if(!require(plotly)) install.packages("plotly",
                                      repos = "http://cran.us.r-project.org")
```

```r
if(!require(lubridate)) install.packages("tidyverse",
                                         repos = "http://cran.us.r-project.org")
if(!require(MLmetrics)) install.packages("plotly",
                                         repos = "http://cran.us.r-project.org")

# REG02_EPHC_ANUAL_2017.SAV dataset and 2018:
# https://www.dgeec.gov.py/microdatos/microdatos.php

dl <- tempfile()
download.file("https://www.dgeec.gov.py/microdatos/register/EPHC-ANUAL/REG02_EPHC_ANUAL_

dl <- tempfile()
download.file("https://www.dgeec.gov.py/microdatos/register/EPHC-ANUAL/REG02_EPHC_ANUAL_

EPHC_2017 <- read_sav("REG02_EPHC_ANUAL_2017.SAV")
EPHC_2018 <- read_sav("REG02_EPHC_ANUAL_2018.SAV")

#---------------------------------#
# Create Data and Validation set #
#---------------------------------#

# Select some variables.

Data <- EPHC_2017 %>%
  select(c("DPTO","AREA","P02","P06","ED01",
           "ED0504","ED10","e01aimde", "ipcm")) %>%
  mutate(Drop_out = ifelse(ED0504 < 503 &
                             ED0504 != 409 & !is.na(ED10) & !is.na(ED0504),
                           "Desertor", "No_Desertor")) %>%
  filter(!is.na(ipcm) & !is.na(ED01))

Validation <- EPHC_2018 %>%
  select(c("DPTO","AREA","P02","P06","ED01",
           "ED0504","ED10", "e01aimde", "ipcm")) %>%
  mutate(Drop_out = ifelse((ED0504 < 503 &
                              ED0504 != 409 & !is.na(ED10)) & !is.na(ED0504),
                           "Desertor", "No_Desertor")) %>%
  filter(!is.na(ipcm) & !is.na(ED01))


# Observe the Data_set
glimpse(Data)
```

```
## Rows: 66,553
## Columns: 10
## $ DPTO     <dbl+lbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
## $ AREA     <dbl+lbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
## $ P02      <dbl+lbl> 39, 23, 17, 78, 41, 33, 41, 8, 49, 47, 13, 55, 28, 50,...
## $ P06      <dbl+lbl> 6, 1, 6, 6, 1, 1, 6, 1, 6, 1, 6, 6, 1, 1, 6, 1, 6, 6, ...
## $ ED01     <dbl+lbl> 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, ...
## $ ED0504   <dbl+lbl> 606, 2401, 901, 2404, 2406, 2406, 2406, 302, 606, 606,...
## $ ED10     <dbl+lbl> 14, NA, NA, NA, 5, 5, 5, NA, 1, 1, NA, NA, 2, 2, 5, NA...
## $ e01aimde <dbl+lbl> 0, 1996045, 0, 0, 2994067, 2994067, 5988134, 0, 528951...
## $ ipcm     <dbl> 2328719, 2328719, 2328719, 10329532, 10329532, 2482385, 24...
## $ Drop_out <chr> "No_Desertor", "No_Desertor", "No_Desertor", "No_Desertor"...
```

2.2 Modify the data

If we look at the dataset, we can see that it contains some coded variables that we could convert into factors. This helps us generate graphs by levels and makes it easier for us to carry out our modeling.

```r
#------------------#
# Modify the data #
#------------------#

# As Factor some variables
Data <- Data %>%
  mutate_at(vars(c("DPTO","P06","AREA", "ED01", "ED0504","ED10","Drop_out")),
            funs(as_factor(.)))

Validation <- Validation %>%
  mutate_at(vars(c("DPTO","P06","AREA", "ED01", "ED0504","ED10","Drop_out")),
            funs(as_factor(.)))


# Relevel the possitive variable
Data$Drop_out <- relevel(Data$Drop_out, "Desertor")
Validation$Drop_out <- relevel(Validation$Drop_out, "Desertor")


# Rename the Variables
names(Data) <- c("Department","Area", "Age", "Sex", "Language",
                 "Grade","Cause","Income", "Income_pc","Drop_out")

names(Validation) <- c("Department","Area", "Age", "Sex", "Language",
                       "Grade","Cause","Income", "Income_pc","Drop_out")
```

Finally we create the training and test set to generate our models.

```r
# Create the test set and train set
set.seed(1, sample.kind = "Rounding")
index <- createDataPartition(Data$Drop_out, times = 1, p = 0.1, list = FALSE)
test_set <- Data[index,]
train_set <- Data[-index,]


# Remove Extra Datasets
rm(EPHC_2017,EPHC_2018, index)
```

When the model reaches the `F1-Score` goal in the training set, we will apply the model we created with the *Validation* set for the final test.

Chapter 3

Data Exploration

Before starting to build our model, we must study how our data is composed from different points of view and with different variables.

3.1 Pre-visualize the Data

```
# Pre-visualize the Data
names(Data)
```

```
##  [1] "Department" "Area"       "Age"        "Sex"        "Language"
##  [6] "Grade"      "Cause"      "Income"     "Income_pc"  "Drop_out"
```

```
head(Data, 5)
```

Department	Area	Age	Sex	Language	Grade	Cause
Asunción	Urbana	39	Mujeres	Castellano	Bachillerato Humanístico / Científico 6°	Motivo
Asunción	Urbana	23	Hombres	Castellano	Superior Universitario 1°	NA
Asunción	Urbana	17	Mujeres	Castellano	Educ. Media Científica 1°	NA
Asunción	Urbana	78	Mujeres	Castellano	Superior Universitario 4°	NA
Asunción	Urbana	41	Hombres	Castellano	Superior Universitario 6°	Consic

```
dim(Data)
```

```
## [1] 66553    10
```

```r
class(Data)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```r
summary(Data)
```

```
##        Department        Area             Age               Sex
##  Central     : 9558   Urbana:37263   Min.   :  5.00   Hombres:33222
##  Alto Paraná: 8638   Rural :29290   1st Qu.: 16.00   Mujeres:33331
##  Itapúa      : 5404                  Median : 30.00
##  Asunción    : 4500                  Mean   : 33.37
##  Cordillera : 4284                  3rd Qu.: 48.00
##  San Pedro  : 4054                  Max.   :107.00
##  (Other)    :30115
##                  Language                                   Grade
##  Guaraní               :30808   Educ. Escolar Básica 6°   :10661
##  Guaraní y Castellano:18286   Educ. Escolar Básica 3°   : 4454
##  Castellano           :15063   Educ. Escolar Básica 4°   : 4083
##  Otro idioma          : 2172   Educ. Escolar Básica 5°   : 3936
##  No habla             :   221   Educ. Media Científica 3°: 3719
##  NR                   :     3   (Other)                   :38061
##                                 NA's                      : 1639
##                                   Cause            Income             Income_pc
##  Sin recursos en el hogar      : 9627   Min.   :        0   Min.   :        0
##  Necesidad de trabajar         : 8678   1st Qu.:        0   1st Qu.:   505131
##  Motivos familiares            : 3762   Median :        0   Median :   885524
##  No quiere estudiar            : 2377   Mean   :  1034624   Mean   :  1407934
##  No existe institución cercana: 2265   3rd Qu.:  1490157   3rd Qu.:  1539108
##  (Other)                       : 4108   Max.   :490602722   Max.   :686031500
##  NA's                          :35736
##        Drop_out
##  Desertor   :13005
##  No_Desertor:53548
##
##
##
##
##
```

```r
glimpse(Data)
```

```
## Rows: 66,553
## Columns: 10
## $ Department <fct> Asunción, Asunción, Asunción, Asunción, Asunción, Asunci...
## $ Area       <fct> Urbana, Urbana, Urbana, Urbana, Urbana, Urbana, Urbana, ...
## $ Age        <dbl+lbl> 39, 23, 17, 78, 41, 33, 41, 8, 49, 47, 13, 55, 28, 5...
## $ Sex        <fct> Mujeres, Hombres, Mujeres, Mujeres, Hombres, Hombres, Mu...
## $ Language   <fct> Castellano, Castellano, Castellano, Castellano, Castella...
## $ Grade      <fct> Bachillerato Humanístico / Científico 6°, Superior Unive...
## $ Cause      <fct> Motivos familiares, NA, NA, NA, Considera que terminó lo...
## $ Income     <dbl+lbl> 0, 1996045, 0, 0, 2994067, 2994067, 5988134, 0, 5289...
## $ Income_pc  <dbl> 2328719, 2328719, 2328719, 10329532, 10329532, 2482385, ...
## $ Drop_out   <fct> No_Desertor, No_Desertor, No_Desertor, No_Desertor, No_D...
```

We can describe the variables that we will use in our model:

1. Department: Paraguay is divided into 17 departments, to give you an idea, the departments are like the states in the United States.
2. Area: We can notice two important areas, urban and rural.
3. Sex: In this case Men = `Hombres` and Women = `Mujeres`.
4. Language: In Paraguay we have two important languages, Spanish = `Castellano` and Guaraní. Some people speak a mix of them and we call it jopara.
5. Grade: The different levels of education are defined in this variable. In our case we will focus on school levels.
6. Cause: It refers to the reasons why they drop out of school. Where we will see variables such as lack of resources or family reasons.
7. Income: They are the income declared monthly.
8. Income_pc: They are the income distributed per capita.
9. Drop_out: This is the variable that we will try to predict for the year 2018. In this is found who are deserters `Desertor` and who are not `No_Desertor`.

```r
# View some tables
table(Data$Sex)
```

```
##
## Hombres Mujeres
##   33222   33331
```

```
table(Data$Area)
```

```
##
## Urbana  Rural
##  37263  29290
```

```
Datades <- table(Data$Drop_out)
Valdes <- table(Validation$Drop_out)

Desertor_Data <- Datades[1]/Datades[2]
Desertor_Val <- Valdes[1]/Valdes[2]
Desertor_Data
```

```
##  Desertor
## 0.2428662
```

```
Desertor_Val
```

```
##  Desertor
## 0.2258787
```

```
rm(Datades, Valdes)
```

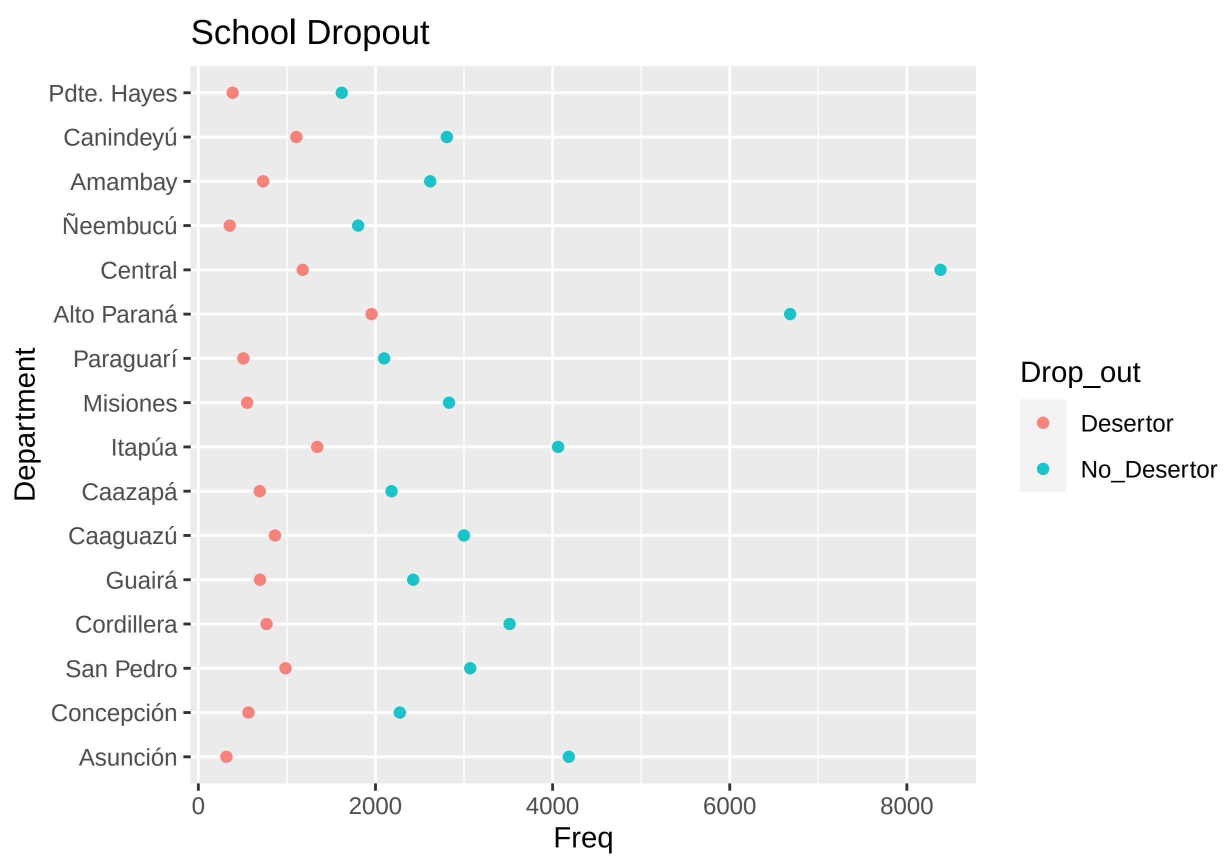
When we look at the Data tables and dropout validation, we notice that the dataset is clearly unbalanced.

3.2 Exploration by each feature

3.2.1 Desertion by Department

If we look at the graph and table below, we will see that the departments with the largest population also tend to have the highest number of deserters.

```
data.frame(table(select(Data, Department, Drop_out))) %>%
  ggplot(aes(Freq, Department, color=Drop_out)) +
  geom_point(alpha=0.9) +
  ggtitle("School Dropout")
```

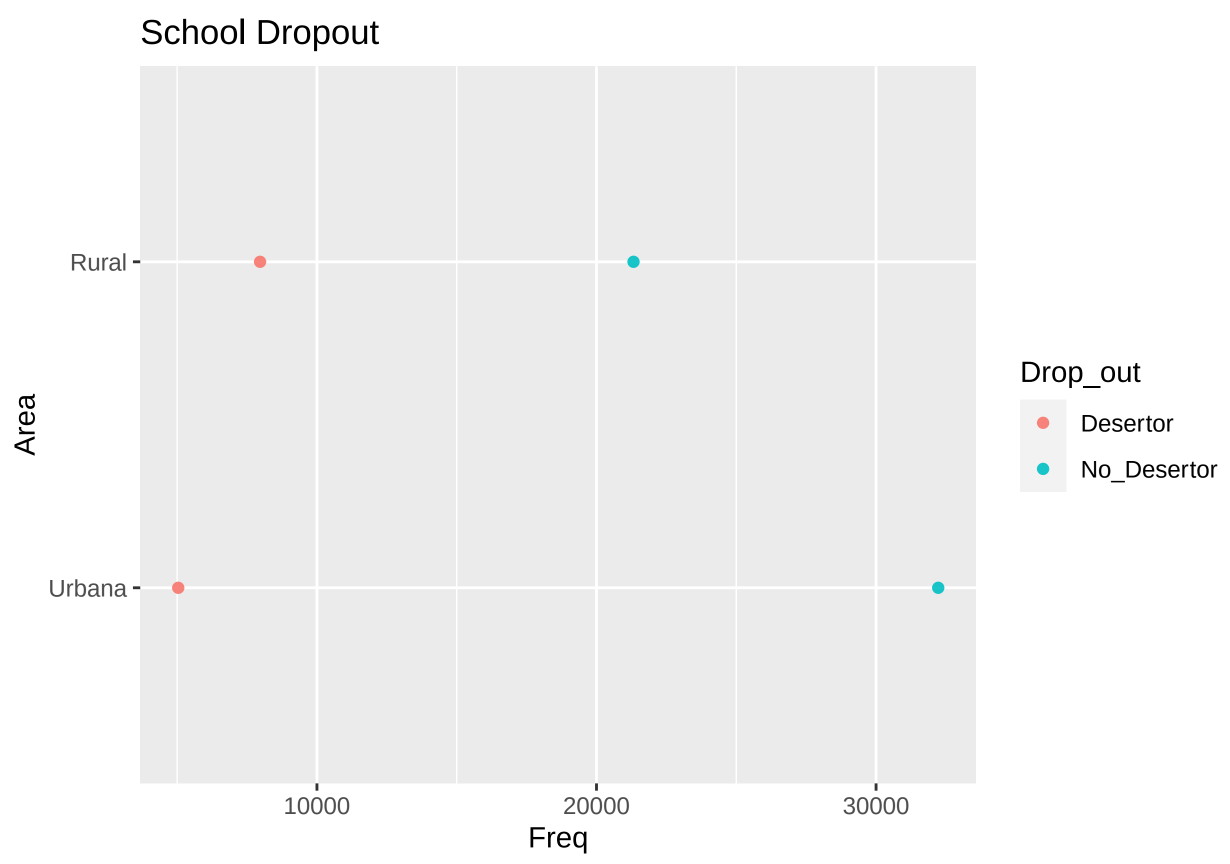


```r
# Department by population
table(as.data.frame(Data$Department))
```

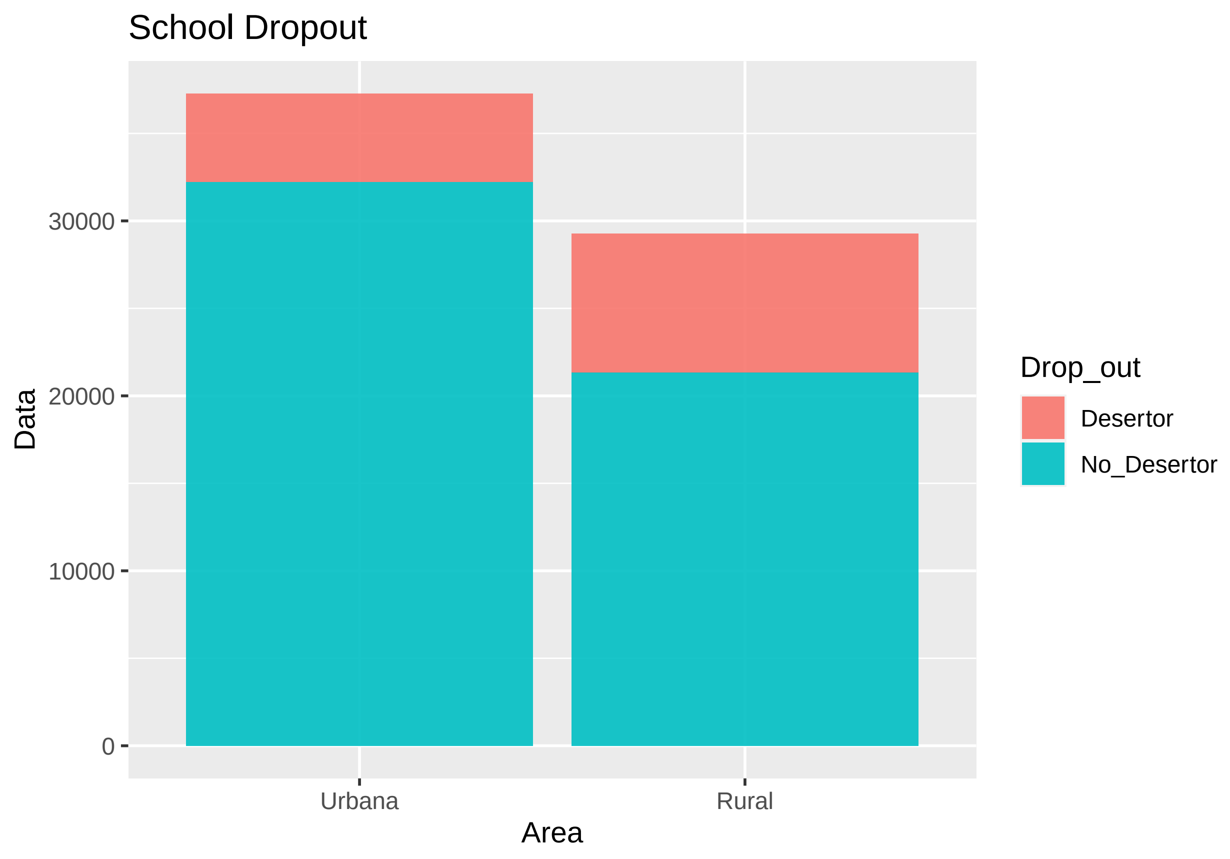
```
##
##    Asunción  Concepción   San Pedro  Cordillera      Guairá    Caaguazú
##        4500        2841        4054        4284        3122        3864
##     Caazapá      Itapúa    Misiones   Paraguarí Alto Paraná     Central
##        2874        5404        3382        2606        8638        9558
##    Ñeembucú     Amambay   Canindeyú Pdte. Hayes
##        2158        3349        3913        2006
```

3.2.2 Desertion by Area

```r
as.data.frame(table(select(Data, Area, Drop_out))) %>%
  ggplot(aes(Freq, Area, color=Drop_out)) +
  geom_point(alpha=0.9) +
  ggtitle("School Dropout")
```



```
Data %>% ggplot(aes(Area, fill=Drop_out)) + geom_bar(alpha=0.9) +
  ggtitle("School Dropout") +
  xlab("Area") +
  ylab("Data")
```



```
Des <- as.data.frame(table(select(Data, Area, Drop_out)))
Des
```

Area	Drop_out	Freq
Urbana	Desertor	5039
Rural	Desertor	7966
Urbana	No_Desertor	32224
Rural	No_Desertor	21324

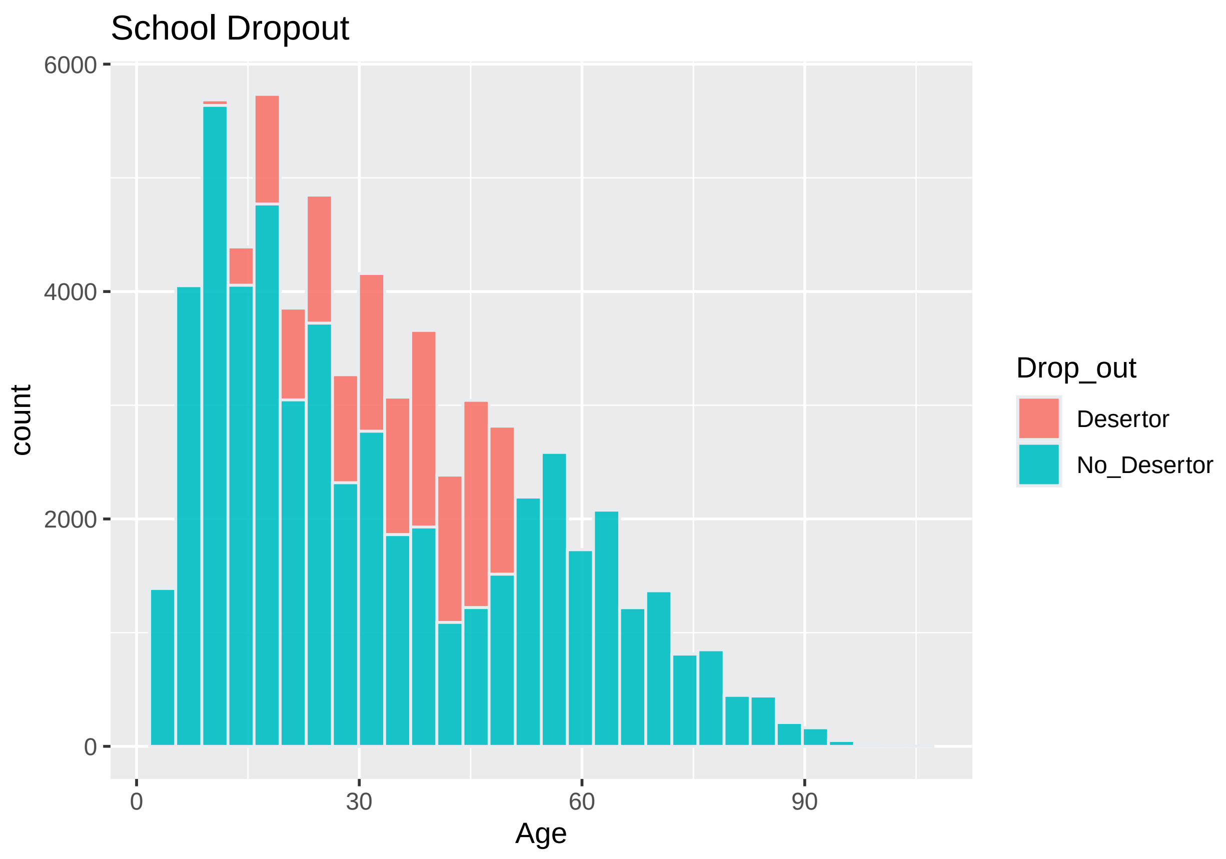
```
# Desertor in percentage by Area
data.frame(Urbana = (Des$Freq[1]/(Des$Freq[1]+Des$Freq[3]))*100,
           Rural = (Des$Freq[2]/(Des$Freq[2]+Des$Freq[4]))*100)
```

Urbana	Rural
13.5228	27.197

When looking at the graphs we clearly notice that the rural area tends to have more school dropouts and with the tables we affirm our observations.

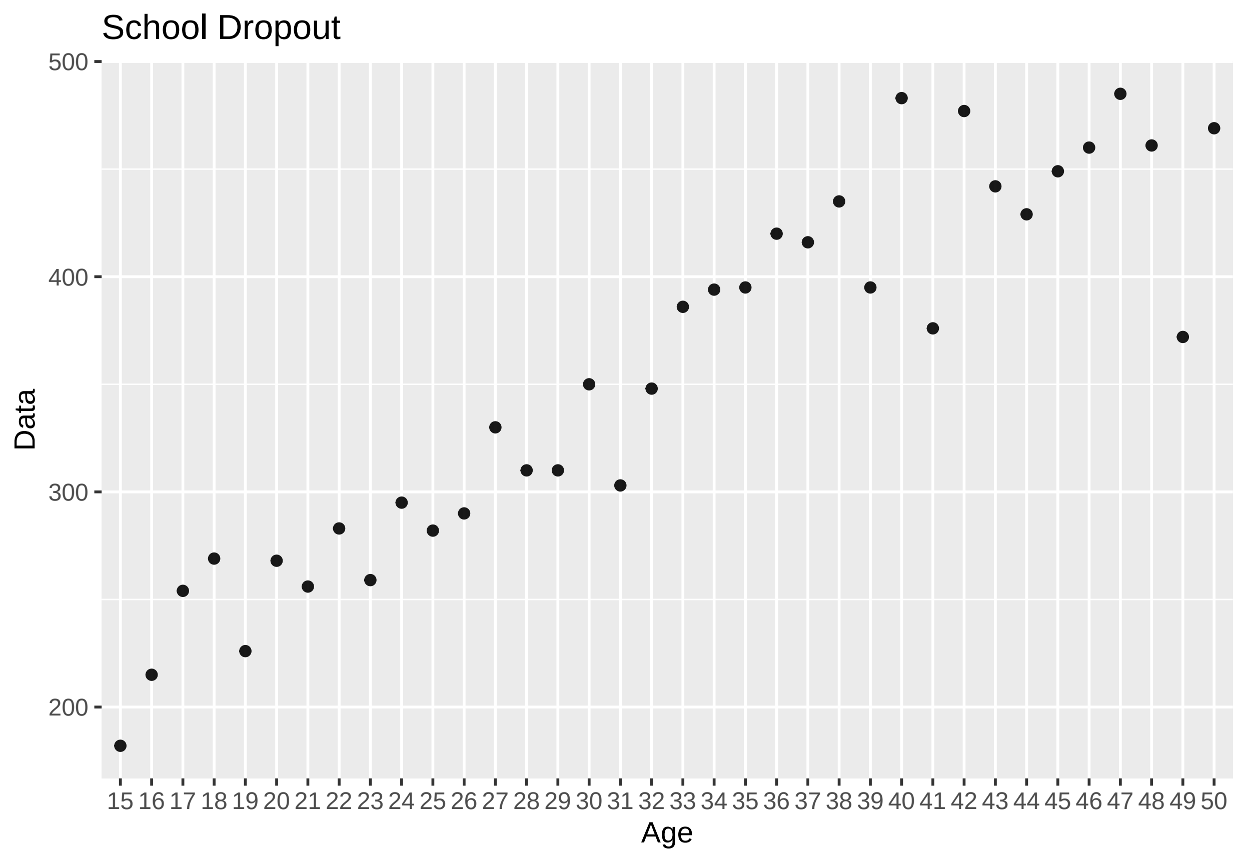
3.2.3 Desertion by Age

```
Data %>%
  ggplot(aes(Age, fill=Drop_out)) + geom_histogram(color="#e9ecef", alpha=0.9) +
  ggtitle("School Dropout")
```



```
only_Des <- Data %>% filter(Drop_out=="Desertor")

as.data.frame(table(only_Des$Age)) %>%
  filter(Freq > 100) %>%
  ggplot(aes(Var1,Freq)) + geom_point(fill="#69b3a2", alpha=0.9) +
  ggtitle("School Dropout") +
  xlab("Age") +
  ylab("Data")
```



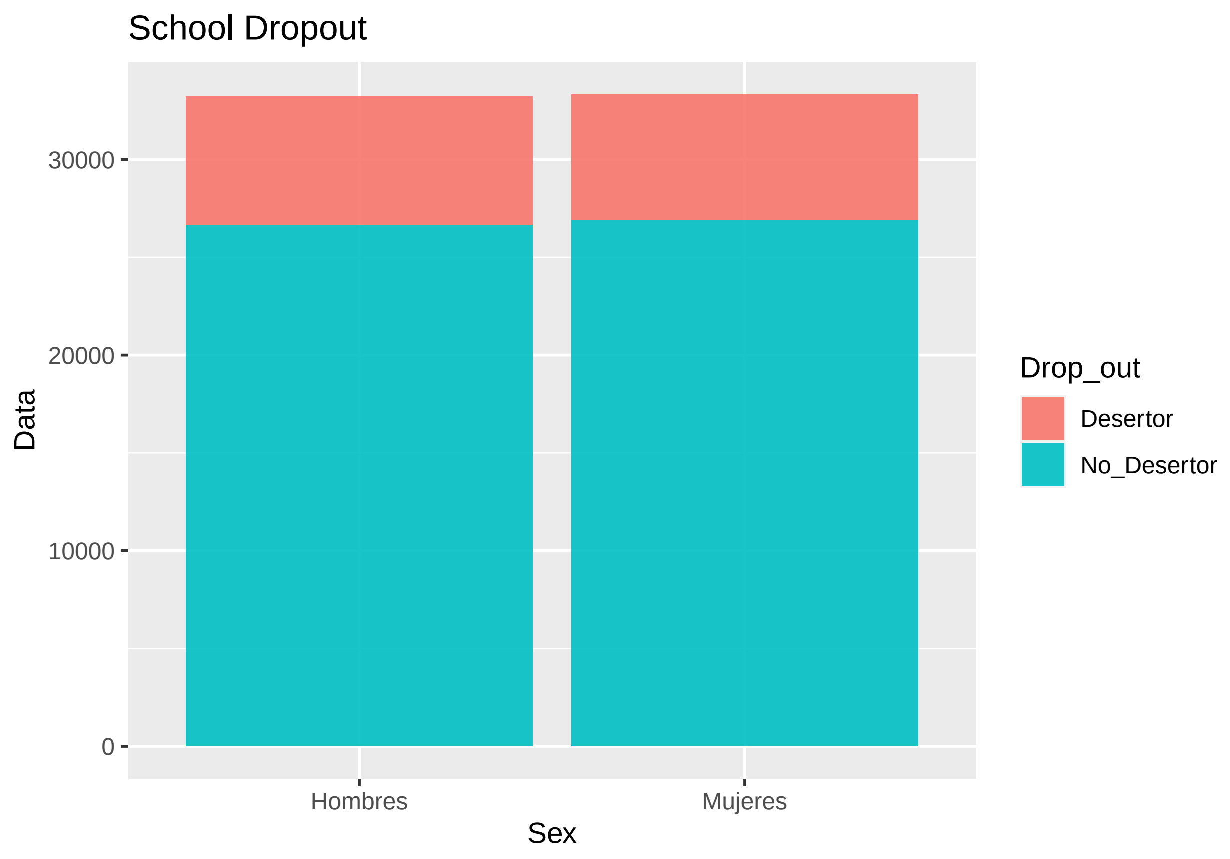
```
as.data.frame(table(only_Des$Age)) %>%
  filter(Freq > 100) %>% arrange(-Freq) %>% head(5)
```

Var1	Freq
47	485
40	483
42	477
50	469
48	461

This graph is interesting since we visualize a timeline created with the age of the people. We can calculate the number of deserters by age and, as expected, the generation that currently has 20 years deserted less than the generation that currently has 40.

3.2.4 Desertion by Sex

```r
Data %>% ggplot(aes(Sex, fill=Drop_out)) + geom_bar(alpha=0.9) +
  ggtitle("School Dropout") +
  xlab("Sex") +
  ylab("Data")
```



```r
Des <- as.data.frame(table(select(Data, Sex, Drop_out)))
Des
```

Sex	Drop_out	Freq
Hombres	Desertor	6579
Mujeres	Desertor	6426
Hombres	No_Desertor	26643
Mujeres	No_Desertor	26905

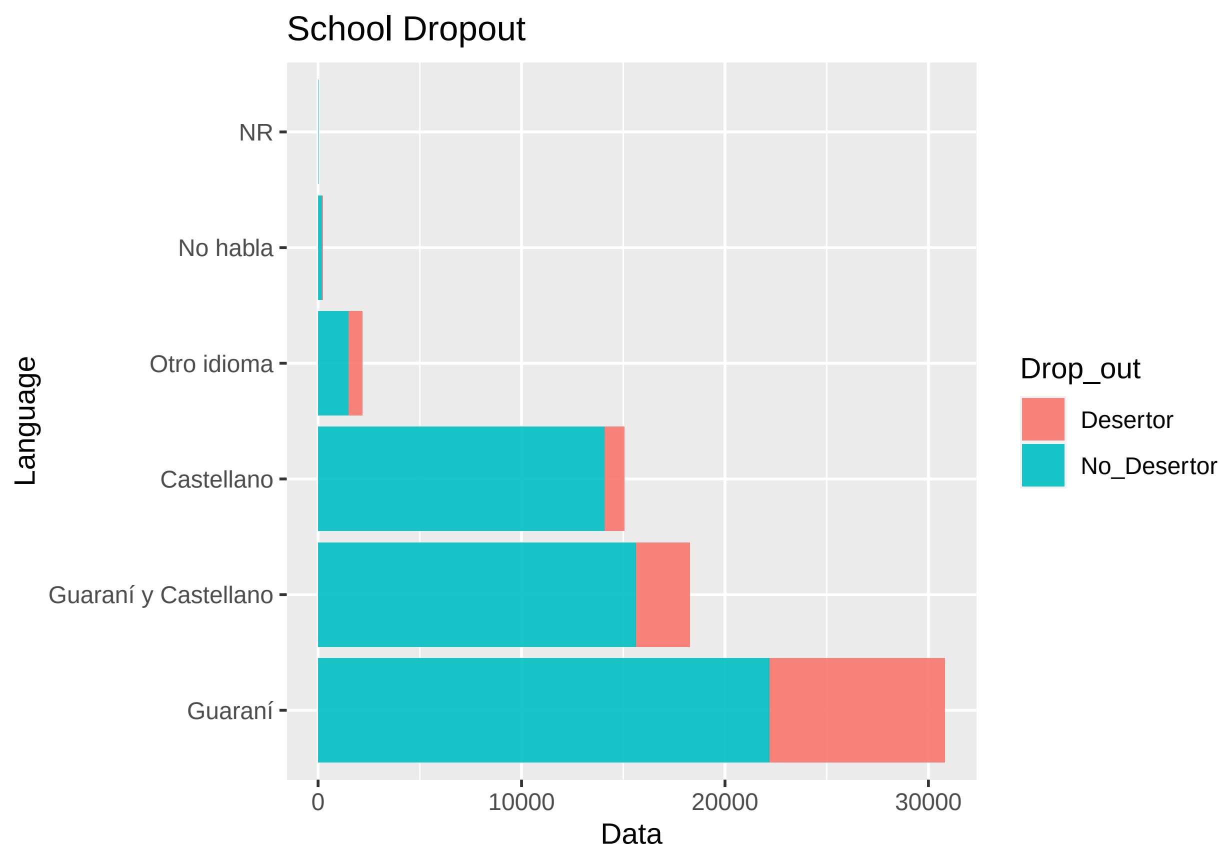
```r
# Desertor in percentage
data.frame(Man = (Des$Freq[1]/(Des$Freq[1]+Des$Freq[3]))*100,
           Woman = (Des$Freq[2]/(Des$Freq[2]+Des$Freq[4]))*100)
```

Man	Woman
19.80314	19.27935

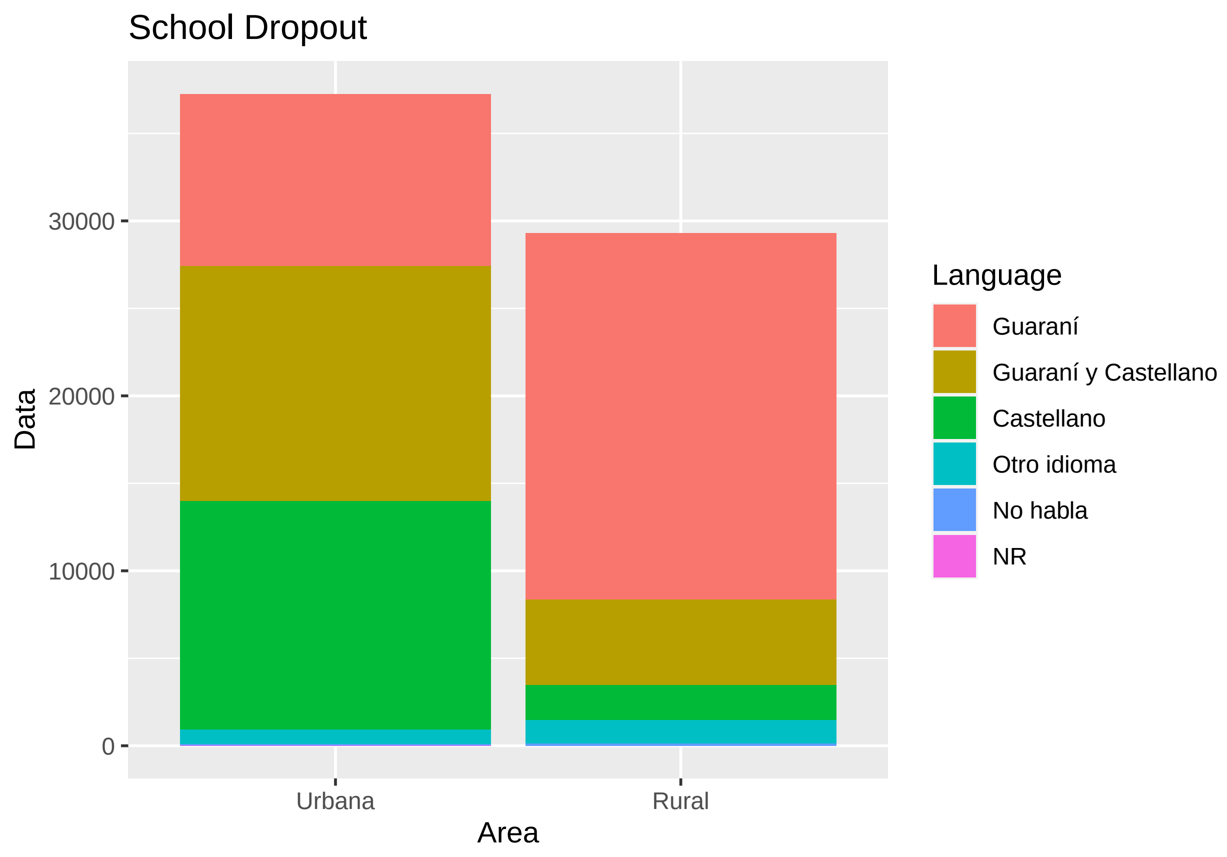
We can note that school dropouts are not due to gender, although we also note that men dropped out a little more than women.

3.2.5 Desertion by Language

```
Data %>% ggplot(aes(Language, fill=Drop_out)) + geom_bar(alpha=0.9) +
  ggtitle("School Dropout") +
  xlab("Language") +
  ylab("Data") + coord_flip()
```



```
Data %>% ggplot(aes(Area, fill=Language)) + geom_bar() +
  ggtitle("School Dropout") +
  xlab("Area") +
  ylab("Data")
```



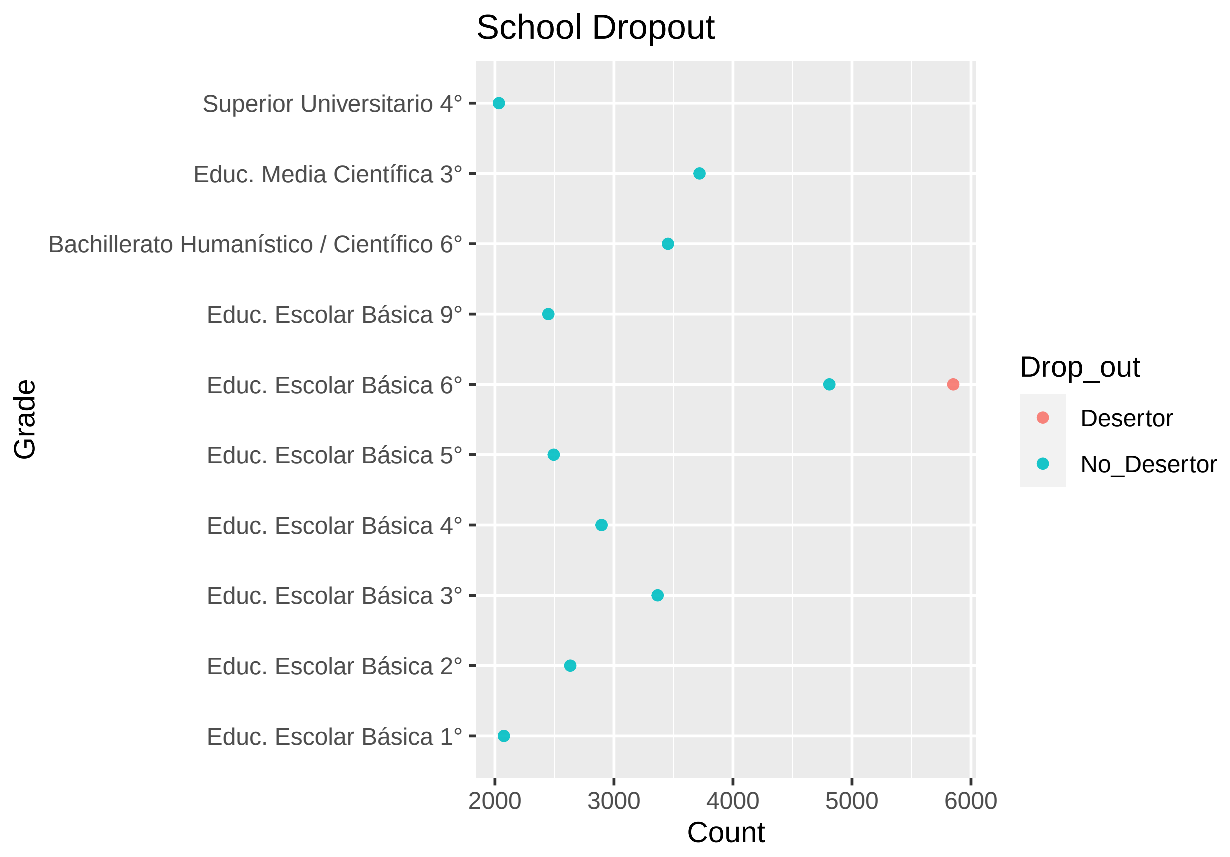
The Language by Area chart demonstrates an interesting reality in Paraguay. Most of the population living in rural areas tend to speak in more closed Guarani and those in urban areas tend to speak jopara or Spanish.

3.2.6 Desertion by Grade

```
Grade_Des <- as.data.frame(table(Data$Grade, Data$Drop_out)) %>%
  filter(Freq > 2000)

names(Grade_Des) <- c("Grade","Drop_out", "Count")

Grade_Des %>%
  ggplot(aes(Count, Grade, color=Drop_out)) +
  geom_point(alpha=0.9) +
  ggtitle("School Dropout")
```



```
Grade_Des %>% filter(Grade=="Educ. Escolar Básica 6°")
```

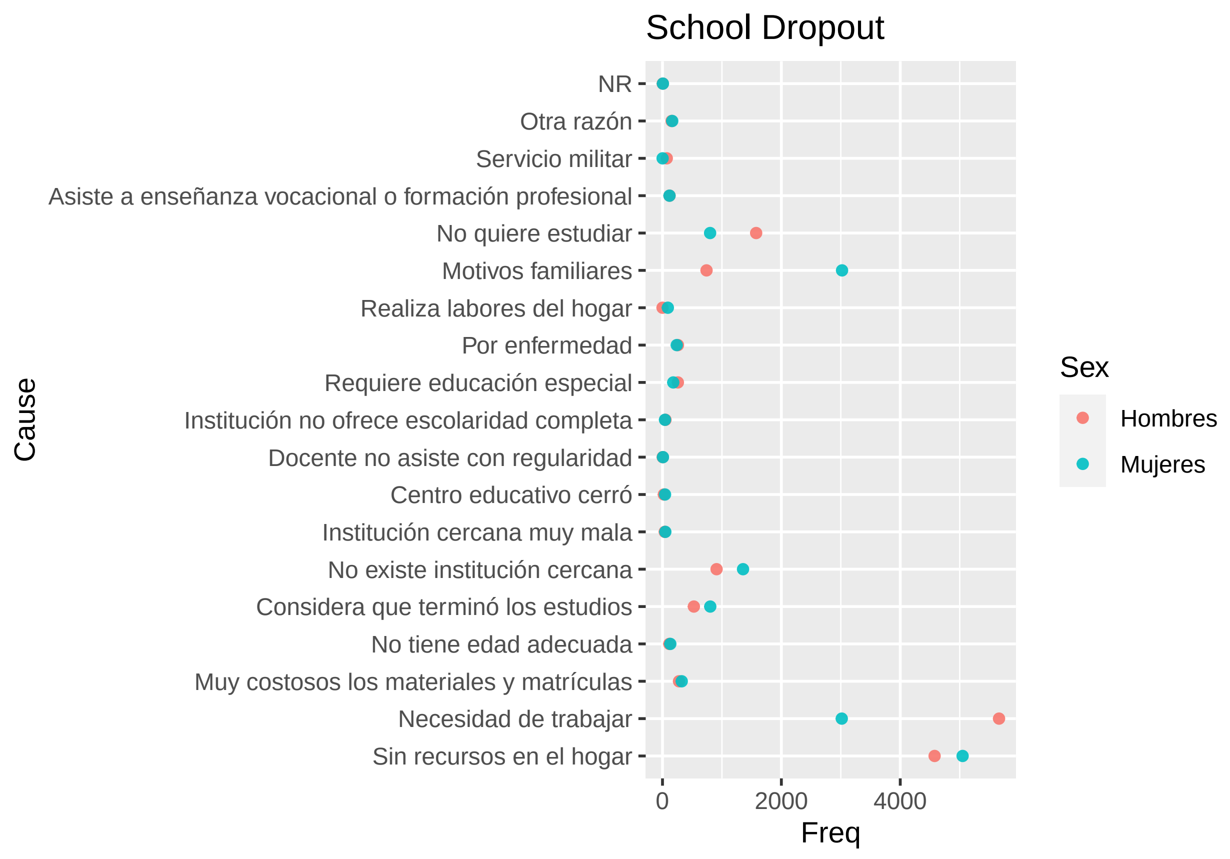
Grade	Drop_out	Count
Educ. Escolar Básica 6°	Desertor	5851
Educ. Escolar Básica 6°	No_Desertor	4810

We note that for some reason people tend to drop out more in 6th grade than in others, in fact, there are more 6th grade dropouts(Desertores) than No_Dropouts(No_Desertores).

3.2.7 Desertion by Cause

```
Causa <- Data %>% select(Cause,Sex) %>% table()  %>% as.data.frame()

Causa %>%
  ggplot(aes(Freq, Cause, color=Sex)) +
  geom_point(alpha=0.9) +
  ggtitle("School Dropout")
```



```r
as.data.frame(table(Data$Cause, Data$Sex)) %>%
  filter(Freq>1000) %>% arrange(Var2, -Freq)
```

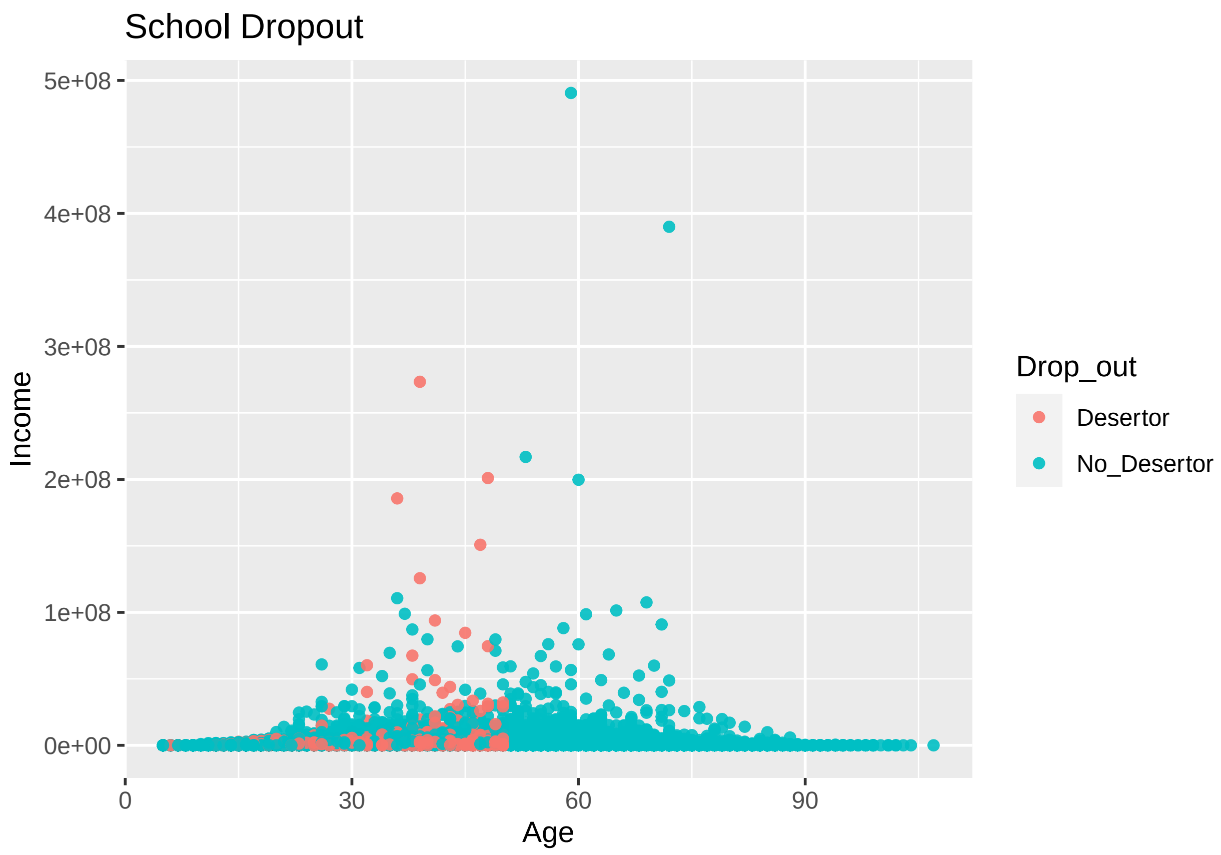
Var1	Var2	Freq
Necesidad de trabajar	Hombres	5662
Sin recursos en el hogar	Hombres	4578
No quiere estudiar	Hombres	1576
Sin recursos en el hogar	Mujeres	5049
Motivos familiares	Mujeres	3021
Necesidad de trabajar	Mujeres	3016
No existe institución cercana	Mujeres	1355

Here we list the main reasons why people drop out of school and note some gender differences.

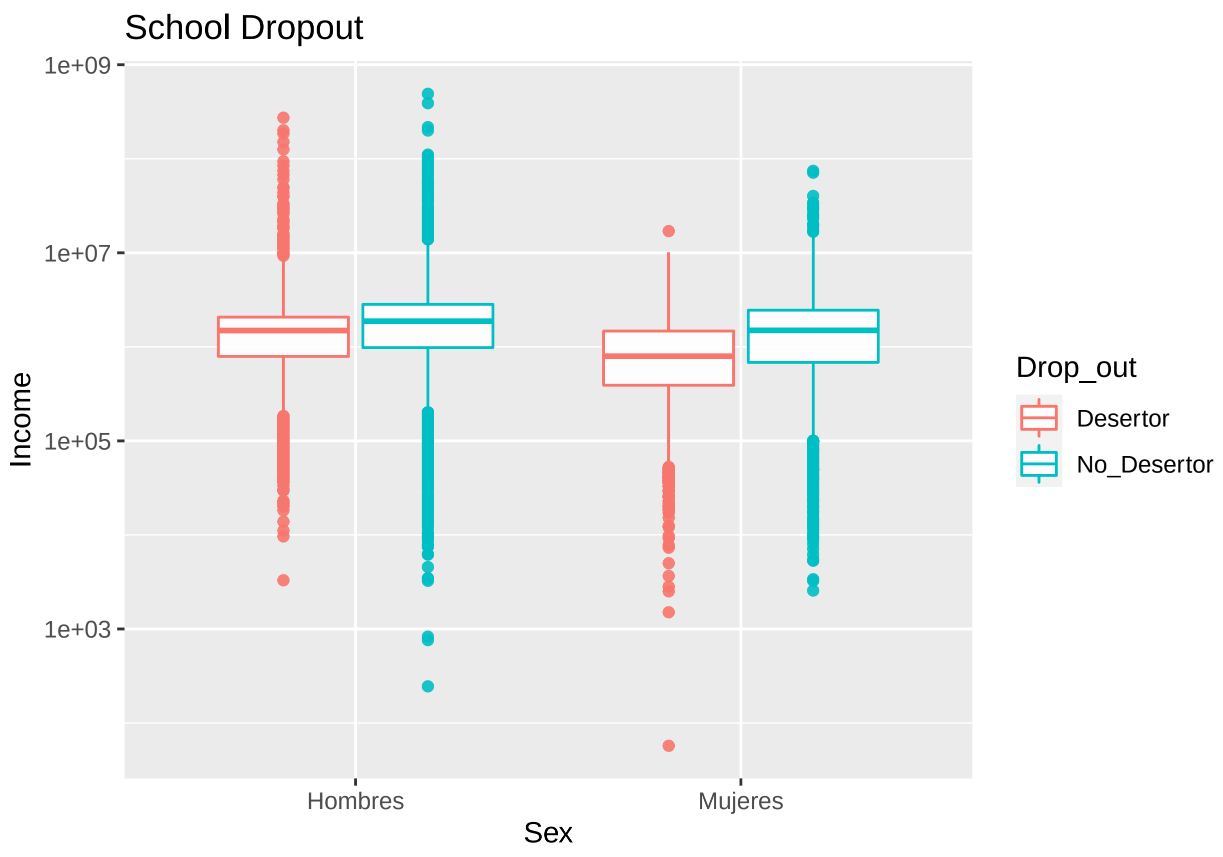
We note that both men and women drop out for reasons such as lack of resources at home or need to work, however, there are many men who do not want to study and women who do not have an institution nearby or for family reasons.

3.2.8 Desertion by Income

```r
Data %>%
  ggplot(aes(Age, Income, color=Drop_out)) +
  geom_point(alpha=0.9) +
  ggtitle("School Dropout")
```



```r
Data %>%
  ggplot(aes(x=Sex, y=Income, color=Drop_out)) +
  geom_boxplot(alpha=0.9) +
  ggtitle("School Dropout") +
  scale_y_log10()
```

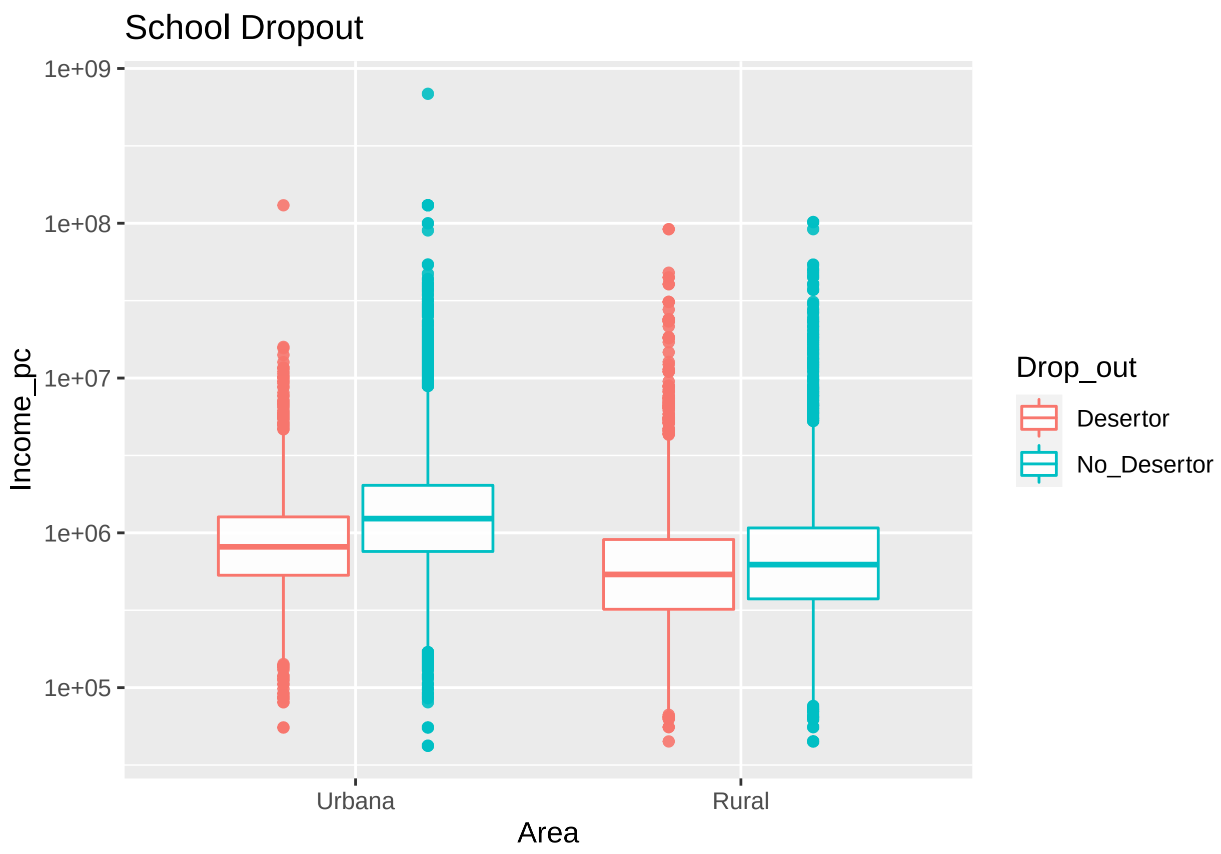


In the first graph we can see the distribution of income by age and classified by dropout. We note that dropouts tend to have less income than non-dropouts.

In the second graph, we note a slight difference between men's wages compared to women's. We can see that male dropouts tend to earn slightly more than women.

3.2.9 Desertion by Income per capita

```
Data %>%
  ggplot(aes(x=Area, y=Income_pc, color=Drop_out)) +
  geom_boxplot(alpha=0.9) +
  ggtitle("School Dropout") +
  scale_y_log10()
```



In this comparison, the Rural area tend to have less income than in the urban area.

Chapter 4

Data Cleaning

Now it is time to remove extra variables that will not be useful in our modeling.

4.1 Remove extra variable

```r
# Remove extra variable
rm(Causa, only_Des, Grade_Des, Des)
```

Chapter 5

Modeling to Use

We will describe a little the different statistical models that we will use in our model. We will not go into details about each of them since the train function of the caret's package does the most work for us, but if you want to delve further I recommend you visit the Available Models.

5.1 Logistic Regression

$$p(x) = Pr(Y = 1|X = x) = \beta_0 + \beta_1 x$$

Where the probability of an event x occurring is given by the probability that the event $Y = 1$ as long as $X = x$ (the available data) which is equal to β_0 (which is the starting point) and β_1 (which is the data classifier).

It is used to model the probability that a certain class or event exists $p(x) = Pr(Y = 1|X = x)$. This can be extended to model various kinds of events $\beta_0 + \beta_1 x + ... + \beta_n x$. Each object detected in the image will be assigned a probability between 0 and 1, with a sum of one. For examples when you need to predict a Binary outcome like pass or fail, you can take pass=1 and fail=0 and calculate the $Pr(Y = 1|X = x)$.

Representation: Straight (or plane or hyperplane)

Evaluation: Logistical loss

Optimization: Estimation by maximum likelihood

5.2 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition, and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events.

The resulting combination may be used as a linear classifier, or, more commonly, for dimensionality reduction before later classification.

In a few words LDA is a dimensionality reduction technique. As the name implies dimensionality reduction techniques reduce the number of dimensions (variables) in a dataset while retaining as much information as possible.

This article show a great example about this.

5.3 Quadratic Discriminant Analysis

The QDA is statistical classifier that uses a quadratic decision surface to separate measurements of two or more classes of objects or events.

Quadratic discriminant analysis (QDA) is closely related to linear discriminant analysis (LDA), where it is assumed that the measurements from each class are normally distributed. Unlike LDA however, in QDA there is no assumption that the covariance of each of the classes is identical.

5.4 Flexible Discriminant Analysis

Flexible Discriminant Analysis is a classification model based on a mixture of linear regression models, which uses optimal scoring to transform the response variable so that the data are in a better form for linear separation, and multiple adaptive regression splines to generate the discriminant surface.

Fisher's linear discriminant analysis is a valuable tool for multigroup classi cation. With a large number of predictors, one can nd a reduced number of discriminant coordinate functions that are optimal for separating the groups.

You can find more information here.

5.5 Random Forest

The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

What do we need in order for our random forest to make accurate class predictions?

1. We need features that have at least some predictive power. After all, if we put garbage in then we will get garbage out.

2. The trees of the forest and more importantly their predictions need to be uncorrelated (or at least have low correlations with each other). While the algorithm itself via feature randomness tries to engineer these low correlations for us, the features we select and the hyper-parameters we choose will impact the ultimate correlations as well.

5.6 Evaluation Results

5.6.1 F1-Score

In statistical analysis of binary classification, the `F1-Score` is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

Here you can see a little example.

```
# Create random variables
set.seed(1, sample.kind = "Rounding")
Example <- sample(seq(0,1,0.01), replace = TRUE, 20)
Actual <- sample(c("Correct", "Incorrect"), 20, replace = TRUE)

# Create the model prediction
Prediction <- ifelse(Example > 0.50,"Correct","Incorrect")

# Test your model
Matrix <- confusionMatrix(as.factor(Prediction), as.factor(Actual))$table
Matrix
```

```
##            Reference
## Prediction  Correct Incorrect
##   Correct         7         4
##   Incorrect       5         4
```

```
# Calculate F1-Score
Precision_Ex <- Matrix[1]/(Matrix[1]+Matrix[3])
Recall_Ex <- Matrix[1]/(Matrix[1]+Matrix[2])

F1 <- 2*((Precision_Ex*Recall_Ex)/(Precision_Ex+Recall_Ex))
F1
```

```
## [1] 0.6086957
```

```r
# This formula do the same
F1_Score(as.factor(Prediction), as.factor(Actual))
```

```
## [1] 0.6086957
```

We have:

```r
Matrix
```

```
##             Reference
## Prediction  Correct Incorrect
##   Correct         7         4
##   Incorrect       5         4
```

Who contain TP True positive 7, FP False positive 4,TN True negative 4, FN False negative 5

You can calculate:

$$\text{precision} = \frac{TP}{TP + FP}$$

and also:

$$\text{recall} = \frac{TP}{TP + FN}$$

and apply the F1_Score to evaluate your model presition.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Here we define the loss functions:

```r
# F1-Score
f1 <- function(data, lev = NULL, model = NULL) {
  f1_val <- F1_Score(y_pred = data$pred, y_true = data$obs, positive = lev[1])
  c(F1 = f1_val)
}

# Remove example variables
rm(Matrix, Example, Actual, Prediction, Precision_Ex, Recall_Ex, F1)
```

5.7 Parameters

5.7.1 Model - Class Weights

When faced with classification tasks in the real world, it can be challenging to deal with an outcome where one class heavily outweighs the other (imbalanced classes). So we need to apply methods to improve performance on imbalanced data.

In this project we include Class weights: impose a heavier cost when errors are made in the minority class.

Here we define the Class Weights:

```
Model_W <- function(data) {
  ifelse(data$Drop_out == "Desertor",
         (1/table(data$Drop_out)[1]) * 0.5,
         (1/table(data$Drop_out)[2]) * 0.5)
}
```

5.7.2 Cross Validation

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, in most methods multiple rounds of cross-validation are performed using different partitions, and the validation results are combined (averaged) over the rounds to give an estimate of the model's predictive performance. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.

5.8 Caret's - train function

This function sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure.

When we run the model, the train function takes the values we gave it to predict and converts them to numerical factors to run the models. For example, if the variable is sex, the train function automatically takes 1 = male and 2 = female to train the classification models.

Another parameter it uses is trcontrol, which is a list of values that define how this function works. Here we define the use of cross validation for example.

Chapter 6

Data analysis and Modeling

We will train the models using the caret package and the training function. We will follow the following modeling order:

1. First we will create a model with simple logistic regression and another adjusted by weights to compare the improvement of the algorithm.
2. We will apply a formula to train 4 of the 5 available models. Due to limitations in computing capacity, we calculate the fda without taking into account the weights and separately. Then we compare it with the other models.
3. Once you have selected the best prediction model, we train the final algorithm with the Validation set. To finish, we compared the model with random sampling to compare it.

Keep in mind that while training the model we will also compare other metric such as Accuracy with F1_Score.

In addition, we will expose the metrics that the train function selected for some models, using cross validation of 3 fold. This method is called **K-fold crossvalidation**.

6.1 Logistic Regression

```r
# Create the fit model with glm.
fit_glm <- train(Drop_out ~ Age + Department + Sex + Area +
                   Income + Income_pc + Language,
                 data = train_set,
                 method = "glm",
                 metric = "F1",
                 trControl = trainControl(method = "cv",
                                          number = 3,
                                          classProbs = TRUE,
                                          summaryFunction = f1))


fit_glm$result
```

parameter	F1	F1SD
none	0.0104241	0.0083723

```
# Test the model with Test_data.
Matrix_glm <- confusionMatrix(predict(fit_glm, test_set),
                              test_set$Drop_out, mode = "prec_recall")

Model_Results <- tibble(Method="Logistic_Regression",
                        F1_Score = Matrix_glm$byClass["F1"],
                        Accuracy = Matrix_glm$byClass["Balanced Accuracy"])

# Saw the Results
data.frame(Model_Results)
```

Method	F1_Score	Accuracy
Logistic_Regression	0.0091463	0.5018391

6.2 Logistic Regression by Weights

```
# Create the fit model with glm.
fit_glm <- train(Drop_out ~ Age + Department + Sex +
                   Area + Income + Income_pc + Language,
                 data = train_set,
                 method = "glm",
                 metric = "F1",
                 weights = Model_W(train_set),
                 trControl = trainControl(method = "cv",
                                          number = 3,
                                          classProbs = TRUE,
                                        summaryFunction = f1))


fit_glm$result
```

parameter	F1	F1SD
none	0.4176995	0.006653

```
# Test the model with Test_data.
Matrix_glm <- confusionMatrix(predict(fit_glm, test_set),
                              test_set$Drop_out, mode = "prec_recall")

Model_Results <- bind_rows(Model_Results,
                          tibble(Method="LR_Weights",
                                 F1_Score = Matrix_glm$byClass["F1"],
                                 Accuracy = Matrix_glm$byClass["Balanced Accuracy"]))

# Saw the Results
data.frame(Model_Results)
```

Method	F1_Score	Accuracy
Logistic_Regression	0.0091463	0.5018391
LR_Weights	0.4274265	0.6606370

We noticed a significant improvement when adjusting the model with weights.

6.3 All Models

As you can see, all the models will follow a similar encoding parameter so we will adjust the algorithms with functions that all the models generate.

```
# Note: this process could take a couple of minutes

# Select the model to predict.
models <- c("glm", "lda", "qda", "rf")


# Create the fit model with different models.
```

```
fits <- lapply(models, function(model){
  print(model)
  train(Drop_out ~ Age + Department + Sex +
          Area + Income + Income_pc,
        data = train_set,
        method = model,
        metric = "F1",
        weights = Model_W(train_set),
        trControl = trainControl(method = "cv",
                                 number = 3,
                                 classProbs = TRUE,
                                 summaryFunction = f1))
})
```

```
## [1] "glm"
## [1] "lda"
## [1] "qda"
## [1] "rf"
```

```
names(fits) <- models

# Test the model with Validation set.
Matrix_Models <- sapply(fits, function(fits){
  confusionMatrix(predict(fits, newdata = test_set),
                  test_set$Drop_out, mode = "prec_recall")$byClass["F1"]
})

Matrix_Acc <- sapply(fits, function(fits){
  confusionMatrix(predict(fits, newdata = test_set),
                  test_set$Drop_out, mode = "prec_recall")$byClass["Balanced Accuracy"]
})


Model_Results <- data.frame(Model=models,
                            F1_Score=as.factor(Matrix_Models),
                            Accuracy=as.factor(Matrix_Acc)) %>%
  arrange(desc(F1_Score))

# Saw the Results
Model_Results
```

Model	F1_Score	Accuracy
rf	0.615257731958763	0.751315048181712
qda	0.468683651804671	0.691455183149355
glm	0.414422638064811	0.646067845534319
lda	0.00153609831029186	0.500384319754035

6.4 Flexible Discriminant Analysis

Note: For computing problems we need to do the fda modeling without the weights.

```
# Create the fit model with fda.
fit_fda <- train(Drop_out ~ Age + Department + Sex +
                   Area + Income + Income_pc + Language,
                 data = train_set,
                 method = "fda",
                 metric = "F1",
                 trControl = trainControl(method = "cv",
                                          number = 3,
                                          classProbs = TRUE,
                                          summaryFunction = f1))


fit_fda$result
```

nprune	degree	F1	F1SD
2	1	NaN	NA
9	1	0.6104482	0.0070590
17	1	0.6228312	0.0073364

```
# Test the model with Test_data.
Matrix_fda <- confusionMatrix(predict(fit_fda, test_set),
                              test_set$Drop_out, mode = "prec_recall")

Model_Results <- bind_rows(Model_Results,
                           data.frame(Model="fda",
                                      F1_Score = as.factor(Matrix_fda$byClass["F1"]),
                                      Accuracy = as.factor(Matrix_fda$byClass["Balanced Accu

# Saw the Results
```

```
Model_Results
```

Model	F1_Score	Accuracy
fda	0.626605884790717	0.757305771399003
rf	0.615257731958763	0.751315048181712
qda	0.468683651804671	0.691455183149355
glm	0.414422638064811	0.646067845534319
lda	0.00153609831029186	0.500384319754035

As you can see, Accuracy overestimate the model by not taking into account the data imbalance. 0.2428662 for deserters and`r 1-Desertor_Data` for non-deserters from the Data set.

6.5 Remove extra variables

Remove all unnecessary variables to assess the final model with Validation set.

```
rm(fit_glm, fit_s, fits, F1_s, fit_fda,
   Matrix_glm, Matrix_s, Matrix_Models, Matrix_fda,
   train_set, test_set, models,
   Model_Results)
```

Chapter 7

Final Validation

Now is time to train the best model with all Data and test with Validation.

```
# Create the fit model with fda.
fit <- train(Drop_out ~ Age + Department + Sex +
               Area + Income + Income_pc + Language,
             data = Data,
             method = "fda",
             metric = "F1",
             trControl = trainControl(method = "cv",
                                      number = 3,
                                      classProbs = TRUE,
                                      summaryFunction = f1))


fit$result
```

nprune	degree	F1	F1SD
2	1	NaN	NA
9	1	0.6126384	0.0043281
17	1	0.6258989	0.0065286

```
# Test the model with Validation set.
Matrix <- confusionMatrix(predict(fit, Validation),
                          Validation$Drop_out, mode = "prec_recall")

Model_Result <- tibble(Method="FDA",
                       F1_Score = Matrix$byClass["F1"])

# Saw the Results
```

```
data.frame(Model_Result)
```

Method	F1_Score
FDA	0.6274714

7.1 Compare the model with a sample.

We take a base line model to compare the results.

```
# By Sample
set.seed(1, sample.kind = "Rounding")

# Create the fit model with a sample.
fit_s <- sample(c("Desertor", "No_Desertor"), replace = TRUE, nrow(Validation))

F1_s <- F1_Score(Validation$Drop_out, fit_s)

Model_Result <- bind_rows(Model_Result,
                          tibble(Method="By Sample",
                       F1_Score = F1_s))

# Saw the Results
data.frame(Model_Result)
```

Method	F1_Score
FDA	0.6274714
By Sample	0.2703410

Chapter 8

Final Report

8.1 Conclusion

With this project, we present a 2018 dropout prediction with a hit of 0.6274714 on a scale of 0 to 1.

The first thing we can notice is that the data is clearly unbalanced with 0.2258787 of deserters in the Validation data, so if we make a prediction based on random samples we only get a precision of 0.270341 so our model clearly improves the prediction of this dataset.

In addition to this, the expansion factor must be taken into account, since, as these datasets are based on surveys, it must be adjusted by weight to apply it to the population.

In the end we implemented a total of 5 different models of which the fda obtained a higher F1 score, so we implemented this model in the validation dataset to obtain the most optimal result.

In addition, we added parameters such as weight adjustment and cross validation to improve our algorithm. We select a total of 9 variables of which 7 we use to predict and 2 to create the variable to predict `Drop_out`.

It is key to understand that we do not optimize any variable manually, since the train function of the Caret's package performs this process for us. In addition, we compare different metrics with the F1_Score results to demonstrate once again the imbalance of the dataset.

8.2 Limitations

Some of the observations, such as prediction with more variables, can be computationally expensive to perform, in addition to not having all the necessary data.

Please note that the EPHC_2017 data set only has data for approximately 74,000 people. of the 7,000,000 approx. from Paraguay in addition to that EPHC_2018 has even less data (55,000 approx.).

This is a big problem predicting different variables, and therefore perhaps this study is overestimating the precision of the algorithms. To make a more accurate prediction we need to accumulate a larger amount of data.

It should also be noted that this model uses supervised learning, so we must run the model every time the data is updated.

8.3 Future work

We can still improve the model much more, we can add more variables or use other effective techniques such as k-Neighbors or the Naive Bayes, which could obtain better metrics for our model.

Although it is true that the main thing would be to improve Paraguay's data management system, the reality is that there is still a long way to go.

In addition to Caret's package, there are many R libraries that can be downloaded and applied to generate algorithms, some of the most widely used are listed here R Libraries.

Finally, this project can serve as a basis to implement a school dropout prediction system. But keep in mind that the data should be further explored, consider more variables to predict the models, and adjust the expansion factor of the data with other more effective techniques.