Modelo Room 8

Juan José y Juan Carlos Jimenez

2024-02-17

Preliminares

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
library(skimr)
library(janitor)
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```r
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.1 --
## v broom        1.0.5     v rsample      1.2.0
## v dials        1.2.0     v tune         1.1.2
## v infer        1.0.6     v workflows    1.1.3
## v modeldata    1.2.0     v workflowsets 1.0.1
```

```
## v parsnip      1.2.0     v yardstick    1.3.0
## v recipes      1.0.9
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard()     masks purrr::discard()
## x magrittr::extract()   masks tidyr::extract()
## x dplyr::filter()       masks stats::filter()
## x recipes::fixed()      masks stringr::fixed()
## x dplyr::lag()          masks stats::lag()
## x magrittr::set_names() masks purrr::set_names()
## x yardstick::spec()     masks readr::spec()
## x recipes::step()       masks stats::step()
## * Learn how to get started at https://www.tidymodels.org/start/
```

```r
library(ranger)
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(vip)
```

```
##
## Attaching package: 'vip'
##
## The following object is masked from 'package:utils':
##
##     vi
```

Importacion

```r
data <- read_csv('Data/WA_Fn-UseC_-Telco-Customer-Churn.csv')
```

```
## Rows: 7043 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): customerID, gender, Partner, Dependents, PhoneService, MultipleLin...
## dbl  (4): SeniorCitizen, tenure, MonthlyCharges, TotalCharges
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
data %>% glimpse
```

Verificar y corregir columnas

```
## Rows: 7,043
## Columns: 21
## $ customerID       <chr> "7590-VHVEG", "5575-GNVDE", "3668-QPYBK", "7795-CFOCW~
## $ gender           <chr> "Female", "Male", "Male", "Male", "Female", "Female",~
## $ SeniorCitizen    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Partner          <chr> "Yes", "No", "No", "No", "No", "No", "No", "No", "Yes~
```

```
## $ Dependents       <chr> "No", "No", "No", "No", "No", "No", "Yes", "No", "No"~
## $ tenure           <dbl> 1, 34, 2, 45, 2, 8, 22, 10, 28, 62, 13, 16, 58, 49, 2~
## $ PhoneService     <chr> "No", "Yes", "Yes", "No", "Yes", "Yes", "Yes", "No", ~
## $ MultipleLines    <chr> "No phone service", "No", "No", "No phone service", "~
## $ InternetService  <chr> "DSL", "DSL", "DSL", "DSL", "Fiber optic", "Fiber opt~
## $ OnlineSecurity   <chr> "No", "Yes", "Yes", "Yes", "No", "No", "No", "Yes", "~
## $ OnlineBackup     <chr> "Yes", "No", "Yes", "No", "No", "No", "Yes", "No", "N~
## $ DeviceProtection <chr> "No", "Yes", "No", "Yes", "No", "Yes", "No", "No", "Y~
## $ TechSupport      <chr> "No", "No", "No", "Yes", "No", "No", "No", "No", "Yes~
## $ StreamingTV      <chr> "No", "No", "No", "No", "No", "Yes", "Yes", "No", "Ye~
## $ StreamingMovies  <chr> "No", "No", "No", "No", "No", "Yes", "No", "No", "Yes~
## $ Contract         <chr> "Month-to-month", "One year", "Month-to-month", "One ~
## $ PaperlessBilling <chr> "Yes", "No", "Yes", "No", "Yes", "Yes", "Yes", "No", ~
## $ PaymentMethod    <chr> "Electronic check", "Mailed check", "Mailed check", "~
## $ MonthlyCharges   <dbl> 29.85, 56.95, 53.85, 42.30, 70.70, 99.65, 89.10, 29.7~
## $ TotalCharges     <dbl> 29.85, 1889.50, 108.15, 1840.75, 151.65, 820.50, 1949~
## $ Churn            <chr> "No", "No", "Yes", "No", "Yes", "Yes", "No", "No", "Y~
```

```
data %>%
        summarise_all(list(
                .n=~sum(!is.na(.)),
                .na=~sum(is.na(.)),
                .min=~min(.,na.rm = T),
                .max=~max(.,na.rm = T),
                .clase=~class(.),
                .valor_distinto=~n_distinct(.)
        )) %>% mutate(across(everything(),~as.character(.))) %>%
        pivot_longer(everything(),
                     names_to = c("varible",".value"),
                     names_sep = c("_\\.")) %>%
        print(n="all")
```

```
## # A tibble: 21 x 7
##    varible          n     na    min                  max   clase valor_distinto
##    <chr>            <chr> <chr> <chr>                <chr> <chr> <chr>
##  1 customerID       7043  0     0002-ORFBO           9995~ char~ 7043
##  2 gender           7043  0     Female               Male  char~ 2
##  3 SeniorCitizen    7043  0     0                    1     nume~ 2
##  4 Partner          7043  0     No                   Yes   char~ 2
##  5 Dependents       7043  0     No                   Yes   char~ 2
##  6 tenure           7043  0     0                    72    nume~ 73
##  7 PhoneService     7043  0     No                   Yes   char~ 2
##  8 MultipleLines    7043  0     No                   Yes   char~ 3
##  9 InternetService  7043  0     DSL                  No    char~ 3
## 10 OnlineSecurity   7043  0     No                   Yes   char~ 3
## 11 OnlineBackup     7043  0     No                   Yes   char~ 3
## 12 DeviceProtection 7043  0     No                   Yes   char~ 3
## 13 TechSupport      7043  0     No                   Yes   char~ 3
## 14 StreamingTV      7043  0     No                   Yes   char~ 3
## 15 StreamingMovies  7043  0     No                   Yes   char~ 3
## 16 Contract         7043  0     Month-to-month       Two ~ char~ 3
## 17 PaperlessBilling 7043  0     No                   Yes   char~ 2
## 18 PaymentMethod    7043  0     Bank transfer (autom~ Mail~ char~ 4
## 19 MonthlyCharges   7043  0     18.25                118.~ nume~ 1585
## 20 TotalCharges     7032  11    18.8                 8684~ nume~ 6531
```

```
## 21 Churn           7043  0     No                   Yes   char~ 2
```

```
#Convertir a factor
data %>%
  mutate( Churn = factor(Churn,
  levels= c("Yes","No"),
  labels= c("si", "no"))
  ) -> data



data %>%
  mutate(
    SeniorCitizen = factor(SeniorCitizen, levels = c(0, 1), labels = c('no', 'si')),
    gender = factor(gender, levels = c('Female', 'Male'), labels = c('Mujeres', 'Hombres')),
    Partner = factor(Partner, levels = c('Yes', 'No'), labels = c('si', 'No')),
    Dependents = factor(Dependents, levels = c('Yes', 'No'), labels = c('si', 'No')),
    PhoneService = factor(PhoneService, levels = c('Yes', 'No'), labels = c('si', 'No')),
    PaperlessBilling = factor(PaperlessBilling, levels = c('Yes', 'No'), labels = c('si', 'No')),
    MultipleLines = factor(MultipleLines, levels = c('Yes', 'No', "No phone service"), labels = c('si',
    InternetService = factor(InternetService, levels = c('DSL', 'Fiber optic', 'No'), labels = c('DSL',
    OnlineSecurity = factor(OnlineSecurity, levels = c('No', 'Yes', 'No internet service'), labels = c(
    OnlineBackup = factor(OnlineBackup, levels = c('Yes', 'No', 'No internet service'), labels = c('si'
    DeviceProtection = factor(DeviceProtection, levels = c('Yes', 'No', 'No internet service'), labels =
    TechSupport = factor(TechSupport, levels = c('Yes', 'No', 'No internet service'), labels = c('si',
    StreamingTV = factor(StreamingTV, levels = c('Yes', 'No', 'No internet service'), labels = c('si',
    StreamingMovies = factor(StreamingMovies, levels = c('Yes', 'No', 'No internet service'), labels = c
    Contract = factor(Contract, levels = c('Month-to-month', 'One year', 'Two year'), labels = c('mes a
    PaymentMethod = factor(PaymentMethod, levels = c('Electronic check', 'Mailed check', 'Bank transfer
  ) -> data
```

EDA

EDA Univariado

```
skim(data)
```

Todas las variables

Table 1: Data summary

Name	data
Number of rows	7043
Number of columns	21
Column type frequency:	
character	1
factor	17
numeric	3
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
customerID	0	1	10	10	0	7043	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	0	1	FALSE	2	Hom: 3555, Muj: 3488
SeniorCitizen	0	1	FALSE	2	no: 5901, si: 1142
Partner	0	1	FALSE	2	No: 3641, si: 3402
Dependents	0	1	FALSE	2	No: 4933, si: 2110
PhoneService	0	1	FALSE	2	si: 6361, No: 682
MultipleLines	0	1	FALSE	3	no: 3390, si: 2971, sin: 682
InternetService	0	1	FALSE	3	Fib: 3096, DSL: 2421, No: 1526
OnlineSecurity	0	1	FALSE	3	No: 3498, Yes: 2019, sin: 1526
OnlineBackup	0	1	FALSE	3	no: 3088, si: 2429, sin: 1526
DeviceProtection	0	1	FALSE	3	no: 3095, si: 2422, sin: 1526
TechSupport	0	1	FALSE	3	no: 3473, si: 2044, sin: 1526
StreamingTV	0	1	FALSE	3	no: 2810, si: 2707, sin: 1526
StreamingMovies	0	1	FALSE	3	no: 2785, si: 2732, sin: 1526
Contract	0	1	FALSE	3	mes: 3875, dos: 1695, un : 1473
PaperlessBilling	0	1	FALSE	2	si: 4171, No: 2872
PaymentMethod	0	1	FALSE	4	che: 2365, che: 1612, tra: 1544, tra: 1522
Churn	0	1	FALSE	2	no: 5174, si: 1869

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
tenure	0	1	32.37	24.56	0.00	9.00	29.00	55.00	72.00	
MonthlyCharges	0	1	64.76	30.09	18.25	35.50	70.35	89.85	118.75	
TotalCharges	11	1	2283.30	2266.77	18.80	401.45	1397.47	3794.74	8684.80	

Posibles outliers

```
data %>%
  reframe(
    tibble(
      Descrip= c('P_0', 'P_02', 'P_25', 'P_50' , 'P_75', 'P_98', 'P_100') ,
      Valor= quantile( TotalCharges, c(0, 0.2, 0.25, 0.50 ,0.75, 0.98, 1), na.rm= T)
    )
    )
```

```
## # A tibble: 7 x 2
##   Descrip  Valor
##   <chr>    <dbl>
## 1 P_0       18.8
## 2 P_02     267.
## 3 P_25     401.
## 4 P_50    1397.
## 5 P_75    3795.
```

```
## 6 P_98    7721.
## 7 P_100   8685.
```

```
#Media por Partner
data %>%
        group_by(Partner) %>%
        summarise(media=mean(TotalCharges,
                          na.rm =T ))
```

```
## # A tibble: 2 x 2
##   Partner media
##   <fct>   <dbl>
## 1 si      3032.
## 2 No      1585.
```

```
#Distribución de TotalCharges por Partner
data %>%
        ggplot(aes(TotalCharges,
                   color=Partner))+
        geom_density()+
        scale_y_continuous(labels = scales::number_format())
```

model_Room_8_files/figure-latex/unnamed-chunk-6-1.pdf

Existe una diferencia importante en los registros de dinero cobrado a los clientes al considerar aquellos que tienen pareja. Al revisar el gráfico anterior, se evidencia que las personas que informaron tener pareja son las que presentan los cobros más altos. Esto se refleja en la distribución, donde se observa una densidad mayor para valores altos en comparación con las personas que declararon no tener pareja.

Balanceo de datos

```
data %>%
  group_by(Churn) %>%
  count(name = "frec") %>%
ungroup() %>%
mutate( Porc= frec/sum(frec))
```

```
## # A tibble: 2 x 3
##   Churn  frec  Porc
##   <fct> <int> <dbl>
## 1 si     1869 0.265
## 2 no     5174 0.735
```

```
data %>%
group_by(Churn) %>%
count( name = 'frec') %>%
ungroup() %>%
mutate( Porc= frec/sum(frec)) %>%
ggplot( aes(x= Churn, y= Porc)) +
geom_segment( aes(xend= Churn, y=0, yend=Porc),
color= "steelblue", linewidth= 1) +
geom_point( size=5, color= "steelblue") +
```

```
coord_flip() +
scale_y_continuous( labels = percent_format()) +
labs(title= 'Porcentaje de Clientes que Abandonan',
y= "Porcentaje", x= "Churn") +
theme_bw()
```

model_Room_8_files/figure-latex/unnamed-chunk-7-1.pdf

El porcentaje de clientes que abandonan el servicio de telecomunicaciones es aproximadamente 3 a 1 en el muestra de análisis.

EDA Bivariado

Room 2

Genero vs Churn

Tabla

Grafico

Interpretacion

PhoneService vs Churn

Interpretacion

Room 4

SeniorCitizen vs Churn

Tabla

```
data %>%
  group_by(SeniorCitizen, Churn) %>%
  summarise(
    N = n(),
    Porc = round(100*N/nrow(data),2)
  ) %>%
  mutate(Porc_grupo = round(100*N/sum(N),2)) -> valores_Sc
```

```
## `summarise()` has grouped output by 'SeniorCitizen'. You can override using the
## `.groups` argument.
```

```
valores_Sc
```

```
## # A tibble: 4 x 5
## # Groups:   SeniorCitizen [2]
##   SeniorCitizen Churn     N  Porc Porc_grupo
##   <fct>         <fct> <int> <dbl>      <dbl>
## 1 no            si     1393 19.8        23.6
## 2 no            no     4508 64.0        76.4
## 3 si            si      476  6.76       41.7
## 4 si            no      666  9.46       58.3
```

```
# Comentado por conflictos con paquete
# tigerstats::rowPerc(xtabs(~SeniorCitizen+Churn, data=data) )
```

Gráfico

```r
ggplot(valores_Sc, aes(x = SeniorCitizen, y = Porc_grupo, fill = Churn)) +
  geom_col(stat = "identity", position = "dodge") +
  geom_text(aes(label = Porc_grupo), vjust = 1.5,
            position = position_dodge(.9))
```

```
## Warning in geom_col(stat = "identity", position = "dodge"): Ignoring unknown
## parameters: `stat`
```

model_Room_8_files/figure-latex/unnamed-chunk-13-1.pdf

Interpretación

De los adultos mayores el 41% abandonó el servicio, mientras que de los no adultos mayores apenas un 24% abandonó el servicio. Aparentemente, un adulto mayor tiene mayor probabilidad de abandonar el servicio.

MultipleLines vs Churn   Tabla

```r
data %>%
  group_by(MultipleLines, Churn) %>%
  summarise(
    N = n(),
    Porc = round(100*N/nrow(data),2)
  ) %>%
  mutate(Porc_grupo = round(100*N/sum(N),2)) -> valores_Ml
```

```
## `summarise()` has grouped output by 'MultipleLines'. You can override using the
## `.groups` argument.
```

```r
valores_Ml
```

```
## # A tibble: 6 x 5
## # Groups:   MultipleLines [3]
##   MultipleLines Churn     N  Porc Porc_grupo
##   <fct>         <fct> <int> <dbl>      <dbl>
## 1 si            si      850 12.1        28.6
## 2 si            no     2121 30.1        71.4
## 3 no            si      849 12.0        25.0
## 4 no            no     2541 36.1        75.0
## 5 sin servicio  si      170  2.41       24.9
## 6 sin servicio  no      512  7.27       75.1
```

```r
# Comentado por conflictos con paquete
# tigerstats::rowPerc(xtabs(~MultipleLines+Churn, data=data) )
```

Gráfico

```r
ggplot(valores_Ml, aes(x = MultipleLines, y = N, fill = Churn)) +
  geom_col(stat = "identity", position = "dodge") +
  geom_text(aes(label = N), vjust = 1.5,
            position = position_dodge(.9))
```

```
## Warning in geom_col(stat = "identity", position = "dodge"): Ignoring unknown
```

```
## parameters: `stat`
```

model_Room_8_files/figure-latex/unnamed-chunk-16-1.pdf

Interpretación

Entre las categorías de MultipleLines no hay mayor diferencia, entre los que abandonan o no el servicio. Los porcentajes son muy parecidos.

```r
data %>%
  group_by(TechSupport, Churn) %>%
  summarise(
    N = n(),
    Porc = round(100*N/nrow(data),2)
  ) %>%
  mutate(Porc_grupo = round(100*N/sum(N),2)) -> valores_Ts
```

TechSupport vs Churn

```
## `summarise()` has grouped output by 'TechSupport'. You can override using the
## `.groups` argument.
```

```r
valores_Ts
```

```
## # A tibble: 6 x 5
## # Groups:   TechSupport [3]
##   TechSupport  Churn     N  Porc Porc_grupo
##   <fct>        <fct> <int> <dbl>      <dbl>
## 1 si           si      310   4.4       15.2
## 2 si           no     1734  24.6       84.8
## 3 no           si     1446  20.5       41.6
## 4 no           no     2027  28.8       58.4
## 5 sin servicio si      113   1.6        7.4
## 6 sin servicio no     1413  20.1       92.6
```

```r
# Comentado por conflictos con paquete
# tigerstats::rowPerc(xtabs(~TechSupport+Churn, data=data) )
```

Gráfico

```r
ggplot(valores_Ts, aes(x = TechSupport, y = N, fill = Churn)) +
  geom_col(stat = "identity", position = "dodge") +
  geom_text(aes(label = N), vjust = 1.5,
            position = position_dodge(.9))
```

```
## Warning in geom_col(stat = "identity", position = "dodge"): Ignoring unknown
## parameters: `stat`
```

model_Room_8_files/figure-latex/unnamed-chunk-19-1.pdf

Interpretación

Los que no cuentan con soporte técnico, el 41.68% abandona el servicio. Por otro lado, los que si cuentan con soporte apenas un 15.17% abandona el servicio y los que no tienen internet contratado solo un 7.40% abandona el servicio.

```r
data %>%
  group_by(PaymentMethod, Churn) %>%
  summarise(
    N = n(),
    Porc = round(100*N/nrow(data),2)
  ) %>%
  mutate(Porc_grupo = round(100*N/sum(N),2)) -> valores_Pm
```

PaymentMethod vs Churn

```
## `summarise()` has grouped output by 'PaymentMethod'. You can override using the
## `.groups` argument.
```

```r
valores_Pm
```

```
## # A tibble: 8 x 5
## # Groups:   PaymentMethod [4]
##   PaymentMethod            Churn     N  Porc Porc_grupo
##   <fct>                    <fct> <int> <dbl>      <dbl>
## 1 cheque electronico       si     1071 15.2        45.3
## 2 cheque electronico       no     1294 18.4        54.7
## 3 cheque mail              si      308  4.37       19.1
## 4 cheque mail              no     1304 18.5        80.9
## 5 transferencia bancaria   si      258  3.66       16.7
## 6 transferencia bancaria   no     1286 18.3        83.3
## 7 transferencia automatica si      232  3.29       15.2
## 8 transferencia automatica no     1290 18.3        84.8
```

```r
# Comentado por conflictos con paquete
# tigerstats::rowPerc(xtabs(~PaymentMethod+Churn, data=data) )
```

Gráfico

```r
ggplot(valores_Pm, aes(x = PaymentMethod, y = N, fill = Churn)) +
  geom_col(stat = "identity", position = "dodge") +
  geom_text(aes(label = N), vjust = 1.5,
            position = position_dodge(.9))
```

```
## Warning in geom_col(stat = "identity", position = "dodge"): Ignoring unknown
## parameters: `stat`
```

model_Room_8_files/figure-latex/unnamed-chunk-22-1.pdf

Interpretación

A excepción de los que pagan con cheque electrónico los porcentajes de abandono son muy parecidos. En el caso de los que pagan con cheque electrónico un 45.29% abandona el servicio.

EDA Multivariado

MonthlyCharges vs PaymentMethod vs Churn Gráfico 1

```r
ggplot(data, aes(x = PaymentMethod, y = MonthlyCharges, fill = Churn)) +
  geom_boxplot()
```

model_Room_8_files/figure-latex/unnamed-chunk-23-1.pdf

Interpretación

Gráfico 2

```r
ggplot(data, aes(x = Churn, y = MonthlyCharges, fill = PaymentMethod)) +
  geom_boxplot()
```

model_Room_8_files/figure-latex/unnamed-chunk-24-1.pdf

Interpretación

Los montos de pago de aquellos que abandonan el servicio son superiores a aquellos que permanecen con excepción de los que pagan por cheque electrónico.

Room 6

Room 7

Room 8

```r

data %>% tabyl(Churn, Dependents ) -> t1

t1 %>% adorn_totals(c("row", "col")) %>% adorn_percentages("all") %>%
adorn_pct_formatting(rounding = "half up", digits = 0) %>% adorn_ns() %>%
  adorn_title("combined") %>% knitr::kable()
```

Eda Bivariado entre churn y Dependents

Churn/Dependents	si	No	Total
si	5% (326)	22% (1,543)	27% (1,869)
no	25% (1,784)	48% (3,390)	73% (5,174)
Total	30% (2,110)	70% (4,933)	100% (7,043)

```
data %>% count(Churn, Dependents) %>%
  mutate(porc = n / sum(n)) %>%
  ggplot(aes(fill=Dependents, y=porc, x=Churn)) +
    geom_col(position="stack") +
    geom_text(aes(label=scales::percent(porc)),position = position_stack(vjust=0.5))+
  scale_y_continuous(labels = scales::percent_format())
```

model_Room_8_files/figure-latex/unnamed-chunk-25-1.pdf

De la muestra analizada alrededor del 27% corresponde a individuos que dejaron sus planes en el último mes. Además se conoce que el 4,6% de estos individuos contaban con personas dependientes.

```
data %>% tabyl(Churn,OnlineSecurity ) -> t2

t2 %>% adorn_totals(c("row", "col")) %>% adorn_percentages("all") %>%
adorn_pct_formatting(rounding = "half up", digits = 0) %>% adorn_ns()%>% adorn_title("combined")%>% kni
```

Eda Bivariado entre churn y OnlineSecurity

Churn/OnlineSecurity	No	Yes	sin servicio	Total
si	21% (1,461)	4% (295)	2% (113)	27% (1,869)
no	29% (2,037)	24% (1,724)	20% (1,413)	73% (5,174)
Total	50% (3,498)	29% (2,019)	22% (1,526)	100% (7,043)

```
data %>% count(Churn, OnlineSecurity) %>%
  mutate(porc = n / sum(n)) %>%
  ggplot(aes(fill=OnlineSecurity, y=porc, x=Churn)) +
    geom_col(position="stack") +
    geom_text(aes(label=scales::percent(porc)),position = position_stack(vjust=0.5))+
  scale_y_continuous(labels = scales::percent_format())
```

model_Room_8_files/figure-latex/unnamed-chunk-26-1.pdf

El 20,74% de los indviduos analizados y que abandonaron el servicio no contaban con seguridad en línea. Por otro lado aquellos que no salieron del plan reflejaron una participación similar con relación al servicio de seguridad en línea.

```r
data %>% tabyl(Churn,StreamingMovies) -> t3

t3 %>% adorn_totals(c("row", "col")) %>% adorn_percentages("all") %>%
adorn_pct_formatting(rounding = "half up", digits = 0) %>% adorn_ns()%>% adorn_title("combined")%>% kni
```

Eda Bivariado entre Churn y StreamingMovies

Churn/StreamingMovies	si	no	sin servicio	Total
si	12% (818)	13% (938)	2% (113)	27% (1,869)
no	27% (1,914)	26% (1,847)	20% (1,413)	73% (5,174)
Total	39% (2,732)	40% (2,785)	22% (1,526)	100% (7,043)

```r
data %>% count(Churn, StreamingMovies) %>%
  mutate(porc = n / sum(n)) %>%
  ggplot(aes(fill=StreamingMovies, y=porc, x=Churn)) +
    geom_col(position="stack") +
    geom_text(aes(label=scales::percent(porc)),position = position_stack(vjust=0.5))+
  scale_y_continuous(labels = scales::percent_format())
```

model_Room_8_files/figure-latex/unnamed-chunk-27-1.pdf

El 13,32% de los indviduos analizados y que abandonaron el servicio no contaban con servicio de Streaming Movies. Sin embargo, tanto para el grupo que abandonaron o mantuvieron el servicio no se evidencia una importancia relevante para su salidad del plan.

```r
data %>%  group_by(Churn) %>%
  summarise(n = n(),
            promedio = mean(TotalCharges,na.rm = T),
            n_missing = sum(is.na(TotalCharges)),
            desv = sd(TotalCharges, na.rm =T)) -> t4
t4
```

Eda Bivariado entre churn y Total Charges

```
## # A tibble: 2 x 5
##   Churn     n promedio n_missing  desv
##   <fct> <int>    <dbl>     <int> <dbl>
## 1 si     1869    1532.         0 1891.
## 2 no     5174    2555.        11 2329.
```

```r
data %>%
  ggplot(aes(x = Churn, y = TotalCharges, fill = Churn)) +
  geom_boxplot() +
  stat_summary(fun = mean, geom = "point", shape = 3, size = 3,
               color = "white", position = position_dodge(width = 0.75)) +
  labs(title = "Churn vs Total Charges",
       x = "Churn",
```

```
       y = "Total Charges") +
        theme_minimal()
```

model_Room_8_files/figure-latex/unnamed-chunk-28-1.pdf

En promedio el gasto total en el servicio de telecomunicaciones para los individuos que salieron del plan es inferior por usuario en alrededor de USD 1000. Para los usuarios que salieron del servicio se evidencia datos atípicos elevados que alcanzan los valores máximos de los usarios que no salieron del servicio. Cabe señalar que, para los usuarios que no abandoron el servicio su extipendio total del plan se concentra entre el rango intercuartílico. Adicionalmente, se observa que la variable gasto total cuenta con 11 valores perdidos.

```
data %>%  group_by(Churn,Dependents) %>%
    summarise(n = n(),
            promedio = mean(MonthlyCharges,na.rm = T),
            n_missing = sum(is.na(MonthlyCharges)),
            desv = sd(MonthlyCharges, na.rm =T)) -> t5
```

Eda Multivariado entre MonthlyCharges vs Dependents vs Churn

```
## `summarise()` has grouped output by 'Churn'. You can override using the
## `.groups` argument.
```

```
t5
```

```
## # A tibble: 4 x 6
## # Groups:   Churn [2]
##   Churn Dependents     n promedio n_missing  desv
##   <fct> <fct>      <int>    <dbl>     <int> <dbl>
## 1 si    si           326     72.9         0  25.8
## 2 si    No          1543     74.8         0  24.4
## 3 no    si          1784     57.1         0  31.6
## 4 no    No          3390     63.5         0  30.6
```

```
data %>%
  ggplot(aes(x = Churn, y = MonthlyCharges, fill = Dependents)) +
  geom_boxplot() +
  stat_summary(fun = mean, geom = "point", shape = 3, size = 3, color = "white", position = position_do
  labs(title = "Monthtly Charges for Dependents vs Churn",
       x = "Churn",
       y = "Monthly Charges")
```

model_Room_8_files/figure-latex/unnamed-chunk-29-1.pdf

La distribución del gasto mensual de aquellos individuos que no salieron del plan se concentra dentro del cuartil 1 y cuartil 3, además entre el máximo y mínimo de los que dejaron el servicio respecto de los que se quedaron no se verifica mayor diferencia. Es importante destacar que, tanto el promedio como la mediana del pago mensual es superior en los individuos que salieron del plan de los que se quedaron, no obstante no se verifica mayor diferencia respecto de contar o no con dependientes en los usuarios que abandonaron el plan.

Room 9

Room 10

Matriz de Correlación

```r
#data %>%  select(-customerID) %>% GGally::ggpairs()
data %>%  select_if(where(is.numeric)) %>% GGally::ggpairs()
```

model_Room_8_files/figure-latex/unnamed-chunk-30-1.pdf

Según el gráfico anterior existe una alta correlación positiva entre las variables TotalChanges y tenure, con 0.826, seguida de la correlación entre las variables TotalChages y MonthyChanges, con una correlación moderada de 0.651.

##MODELAMIENTO

###Train - Test Split

```r
#data %>% select(-customerID) -> data

set.seed(1234) # Semilla para aleatorios
split <- data %>%
initial_split(
prop = 0.8, # Porcentaje al train
strata = Churn # Estratificación del muestreo
)
```

###Data de entrenamiento

```r
train <- training(split)
dim(train)
```

```
## [1] 5634   21
```

###Data de prueba (test)

```r
test <- testing(split)
dim(test)
```

```
## [1] 1409   21
```

###Preprocesamiento

####Receipe y Balanceo de datos

```r
receta <- train %>%
recipe(Churn ~ . ) %>% ## Crea la receta
## Eliminar variables que no usaremos
step_rm(customerID) %>%
## Crear nuevas variables (insight desde el EDA)
# step_mutate( account_length_anio= account_length/12 )
## Imputar los datos
# step_impute_mean()
```

```r
step_impute_knn(TotalCharges ) %>%
## Estandarizacion/Normalizacion de numericas
step_normalize( all_numeric(), -all_outcomes()) %>%
## Crear una categoría "otros" que agrupe a categorias pequeñas
step_other(all_nominal(), -all_outcomes() , threshold = 0.07, other = "otros") %>%
## Crear una categoría "new" para observaciones con labels "no muestreados"
step_novel(all_nominal(), -all_outcomes() , new_level = "new") %>%
## Crear variables indicadoras para cada categoría
step_dummy(all_nominal(), -all_outcomes() ) %>% # Dummy
## Eliminar automáticamente variables con alta correlacion
## para evitar la multicolinealidad xi ~ xj
# step_corr(all_numeric(), -all_outcomes(), threshold = 0.9) %>%
## Tambien podemos eliminar variables con multicolinealidad "a mano"
#step_rm(total_day_charge, total_eve_charge,
#total_night_charge, total_intl_charge) %>% # Eliminar
## Eliminar columnas con varianza cercana a cero
step_nzv(all_predictors()) %>%
themis::step_upsample(Churn, over_ratio = 0.9, skip= TRUE, seed= 123)
```

####Entrenamiento y ajuste de Hiperparámetro

```r
set.seed(1234)
cv <- vfold_cv(train, v = 5, repeats = 1, strata = Churn)
cv
```

```
## #  5-fold cross-validation using stratification
## # A tibble: 5 x 2
##   splits              id
##   <list>              <chr>
## 1 <split [4507/1127]> Fold1
## 2 <split [4507/1127]> Fold2
## 3 <split [4507/1127]> Fold3
## 4 <split [4507/1127]> Fold4
## 5 <split [4508/1126]> Fold5
```

###Métricas

```r
metricas <- metric_set(accuracy, sens, spec, bal_accuracy)
metricas
```

```
## A metric set, consisting of:
## - `accuracy()`, a class metric     | direction: maximize
## - `sens()`, a class metric         | direction: maximize
## - `spec()`, a class metric         | direction: maximize
## - `bal_accuracy()`, a class metric | direction: maximize
```

Modelamiento - Random Forest

###Especificacion del modelo

```r
rf_sp <-
  rand_forest(
  mtry = tune(), trees = tune(), min_n = tune() ) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification")
```

###Work Flow

```r
rf_wflow <-
  workflow() %>%
  add_recipe(receta) %>%
  add_model(rf_sp)
  rf_wflow
```

```
## == Workflow ==================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor --------------------------------------------------------------
## 8 Recipe Steps
##
## * step_rm()
## * step_impute_knn()
## * step_normalize()
## * step_other()
## * step_novel()
## * step_dummy()
## * step_nzv()
## * step_upsample()
##
## -- Model ---------------------------------------------------------------------
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = tune()
##   min_n = tune()
##
## Engine-Specific Arguments:
##   importance = impurity
##
## Computational engine: ranger
```

###Afinamiento de Hiperpárametros

```r
set.seed(123)
rf_grid <- rf_sp %>%
## preguntamos los parametros tuneables del modelo
parameters() %>%
## Vamos a definir un rango para el min_n y mtry
update(min_n= min_n( range= c(70, 170)),
mtry= mtry( range= c(4, 7))) %>%
grid_latin_hypercube(size = 10) #preguntar como se construye la malla
```

```
## Warning: `parameters.model_spec()` was deprecated in tune 0.1.6.9003.
## i Please use `hardhat::extract_parameter_set_dials()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

###Paralelización

```r
#parallel::detectCores(logical=FALSE)
```

###Entrenamiento de Malla de Busqueda en la Crossvalidation

```r
set.seed(123)
rf_tuned <- tune_grid(
rf_wflow, ## Modelo
resamples= cv, ## Crossvalidation
grid = rf_grid, ## Malla de Busqueda
metrics = metricas, ## Metricas
control= control_grid(allow_par = T, save_pred = T) ## Paralel y Pred
)
rf_tuned
```

```
## # Tuning results
## # 5-fold cross-validation using stratification
## # A tibble: 5 x 5
##   splits              id    .metrics          .notes           .predictions
##   <list>              <chr> <list>            <list>           <list>
## 1 <split [4507/1127]> Fold1 <tibble [40 x 7]> <tibble [0 x 3]> <tibble>
## 2 <split [4507/1127]> Fold2 <tibble [40 x 7]> <tibble [0 x 3]> <tibble>
## 3 <split [4507/1127]> Fold3 <tibble [40 x 7]> <tibble [0 x 3]> <tibble>
## 4 <split [4507/1127]> Fold4 <tibble [40 x 7]> <tibble [0 x 3]> <tibble>
## 5 <split [4508/1126]> Fold5 <tibble [40 x 7]> <tibble [0 x 3]> <tibble>
```

###Evaluación de modelos Evaluamos que modelo resulto mejor

```r
show_best(rf_tuned, metric = 'accuracy', n = 10)
```

```
## # A tibble: 10 x 9
##     mtry trees min_n .metric  .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1     4  1370   116 accuracy binary     0.769     5 0.00960 Preprocessor1_Mode~
##  2     5   325    83 accuracy binary     0.768     5 0.00838 Preprocessor1_Mode~
##  3     6   467    91 accuracy binary     0.767     5 0.00997 Preprocessor1_Mode~
##  4     7   773    80 accuracy binary     0.767     5 0.00904 Preprocessor1_Mode~
##  5     5    70   144 accuracy binary     0.767     5 0.00933 Preprocessor1_Mode~
##  6     5  1904   126 accuracy binary     0.767     5 0.0100  Preprocessor1_Mode~
##  7     6   950   130 accuracy binary     0.766     5 0.00975 Preprocessor1_Mode~
##  8     5  1441   106 accuracy binary     0.765     5 0.00994 Preprocessor1_Mode~
##  9     6  1124   162 accuracy binary     0.765     5 0.00997 Preprocessor1_Mode~
## 10     6  1737   158 accuracy binary     0.765     5 0.0100  Preprocessor1_Mode~
```

```r
show_best(rf_tuned, metric = 'sens', n = 10)
```

```
## # A tibble: 10 x 9
##     mtry trees min_n .metric .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1     5    70   144 sens    binary     0.738     5 0.0130  Preprocessor1_Model~
##  2     4  1370   116 sens    binary     0.736     5 0.0168  Preprocessor1_Model~
##  3     6  1737   158 sens    binary     0.735     5 0.0126  Preprocessor1_Model~
##  4     6  1124   162 sens    binary     0.735     5 0.0126  Preprocessor1_Model~
##  5     6   950   130 sens    binary     0.734     5 0.0117  Preprocessor1_Model~
##  6     5  1904   126 sens    binary     0.734     5 0.0129  Preprocessor1_Model~
##  7     5  1441   106 sens    binary     0.732     5 0.0144  Preprocessor1_Model~
##  8     5   325    83 sens    binary     0.728     5 0.0113  Preprocessor1_Model~
##  9     6   467    91 sens    binary     0.728     5 0.0115  Preprocessor1_Model~
## 10     7   773    80 sens    binary     0.720     5 0.00913 Preprocessor1_Model~
```

```r
show_best(rf_tuned, metric = 'spec', n = 10)
```

```
## # A tibble: 10 x 9
##     mtry trees min_n .metric .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1     7   773    80 spec    binary     0.784     5 0.00913 Preprocessor1_Model~
##  2     5   325    83 spec    binary     0.782     5 0.00749 Preprocessor1_Model~
##  3     6   467    91 spec    binary     0.782     5 0.00951 Preprocessor1_Model~
##  4     4  1370   116 spec    binary     0.780     5 0.00756 Preprocessor1_Model~
##  5     5  1904   126 spec    binary     0.779     5 0.00920 Preprocessor1_Model~
##  6     5    70   144 spec    binary     0.778     5 0.00869 Preprocessor1_Model~
##  7     6   950   130 spec    binary     0.777     5 0.00939 Preprocessor1_Model~
##  8     5  1441   106 spec    binary     0.777     5 0.00881 Preprocessor1_Model~
##  9     6  1124   162 spec    binary     0.776     5 0.00916 Preprocessor1_Model~
## 10     6  1737   158 spec    binary     0.776     5 0.00932 Preprocessor1_Model~
```

```r
show_best(rf_tuned, metric = 'bal_accuracy', n = 10)
```

```
## # A tibble: 10 x 9
##     mtry trees min_n .metric      .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>        <chr>      <dbl> <int>   <dbl> <chr>
##  1     4  1370   116 bal_accuracy binary     0.758     5 0.0117  Preprocessor1_~
##  2     5    70   144 bal_accuracy binary     0.758     5 0.0103  Preprocessor1_~
##  3     5  1904   126 bal_accuracy binary     0.757     5 0.0109  Preprocessor1_~
##  4     6   950   130 bal_accuracy binary     0.756     5 0.0102  Preprocessor1_~
##  5     6  1124   162 bal_accuracy binary     0.755     5 0.0108  Preprocessor1_~
##  6     5   325    83 bal_accuracy binary     0.755     5 0.00925 Preprocessor1_~
##  7     6  1737   158 bal_accuracy binary     0.755     5 0.0108  Preprocessor1_~
##  8     5  1441   106 bal_accuracy binary     0.755     5 0.0112  Preprocessor1_~
##  9     6   467    91 bal_accuracy binary     0.755     5 0.0104  Preprocessor1_~
## 10     7   773    80 bal_accuracy binary     0.752     5 0.00902 Preprocessor1_~
```

Revisando los resultados de las métricas de evaluación del modelo y considerando que es de interés para le empresa contar con la mejor estimación de los individuos que realmente abandonan el servicio se selecciona como referencia los hiperpárametros del modelo número 6 que presenta los mejores resultados en sensibilidad, para ajustar el modelo final.

###Nueva Malla de Búsqueda

```r
set.seed(123)

rf_grid_2 <- crossing(
  min_n = seq(114, 118, 2),
  mtry = c(4, 5),
  trees= seq(1270, 1570, 100)
)

rf_grid_2
```

```
## # A tibble: 24 x 3
##    min_n  mtry trees
##    <dbl> <dbl> <dbl>
##  1   114     4  1270
##  2   114     4  1370
##  3   114     4  1470
##  4   114     4  1570
```

```
##  5   114     5  1270
##  6   114     5  1370
##  7   114     5  1470
##  8   114     5  1570
##  9   116     4  1270
## 10   116     4  1370
## # i 14 more rows
```

####Entrenamiento de Malla de Busqueda en la Crossvalidation

```r
set.seed(123)
rf_tuned_2 <- tune_grid(
  rf_wflow, ## Modelo
  resamples= cv, ## Crossvalidation
  grid = rf_grid_2, ## Malla de Busqueda
  metrics = metricas, ## Metricas
  control= control_grid(allow_par = T, save_pred = T) ## Paralel y Pred
)
rf_tuned_2
```

```
## # Tuning results
## # 5-fold cross-validation using stratification
## # A tibble: 5 x 5
##   splits             id    .metrics          .notes           .predictions
##   <list>             <chr> <list>            <list>           <list>
## 1 <split [4507/1127]> Fold1 <tibble [96 x 7]> <tibble [0 x 3]> <tibble>
## 2 <split [4507/1127]> Fold2 <tibble [96 x 7]> <tibble [0 x 3]> <tibble>
## 3 <split [4507/1127]> Fold3 <tibble [96 x 7]> <tibble [0 x 3]> <tibble>
## 4 <split [4507/1127]> Fold4 <tibble [96 x 7]> <tibble [0 x 3]> <tibble>
## 5 <split [4508/1126]> Fold5 <tibble [96 x 7]> <tibble [0 x 3]> <tibble>
```

Evaluamos que modelo resulto mejor de la segunda grilla de hiperpárametros

```r
show_best(rf_tuned_2, metric = 'accuracy', n = 10)
```

```
## # A tibble: 10 x 9
##     mtry trees min_n .metric  .estimator  mean     n std_err .config
##    <dbl> <dbl> <dbl> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
##  1     4  1470   114 accuracy binary     0.770     5 0.00949 Preprocessor1_Mode~
##  2     5  1270   114 accuracy binary     0.769     5 0.00982 Preprocessor1_Mode~
##  3     4  1270   118 accuracy binary     0.768     5 0.0100  Preprocessor1_Mode~
##  4     4  1470   116 accuracy binary     0.768     5 0.00969 Preprocessor1_Mode~
##  5     4  1370   118 accuracy binary     0.768     5 0.00987 Preprocessor1_Mode~
##  6     4  1370   114 accuracy binary     0.768     5 0.00868 Preprocessor1_Mode~
##  7     4  1370   116 accuracy binary     0.768     5 0.00856 Preprocessor1_Mode~
##  8     5  1570   118 accuracy binary     0.768     5 0.0106  Preprocessor1_Mode~
##  9     4  1470   118 accuracy binary     0.768     5 0.00947 Preprocessor1_Mode~
## 10     5  1570   114 accuracy binary     0.768     5 0.0104  Preprocessor1_Mode~
```

```r
show_best(rf_tuned_2, metric = 'sens', n = 10)
```

```
## # A tibble: 10 x 9
##     mtry trees min_n .metric .estimator  mean     n std_err .config
##    <dbl> <dbl> <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1     4  1470   114 sens    binary     0.738     5  0.0154 Preprocessor1_Model~
##  2     4  1270   116 sens    binary     0.738     5  0.0144 Preprocessor1_Model~
##  3     4  1370   116 sens    binary     0.738     5  0.0137 Preprocessor1_Model~
```

```
## 4     4  1470   116 sens    binary     0.738     5  0.0156 Preprocessor1_Model~
## 5     5  1270   114 sens    binary     0.737     5  0.0119 Preprocessor1_Model~
## 6     4  1570   118 sens    binary     0.737     5  0.0174 Preprocessor1_Model~
## 7     4  1270   114 sens    binary     0.736     5  0.0148 Preprocessor1_Model~
## 8     4  1570   116 sens    binary     0.736     5  0.0136 Preprocessor1_Model~
## 9     4  1370   118 sens    binary     0.736     5  0.0154 Preprocessor1_Model~
## 10    4  1270   118 sens    binary     0.736     5  0.0157 Preprocessor1_Model~
```

```r
show_best(rf_tuned_2, metric = 'spec', n = 10)
```

```
## # A tibble: 10 x 9
##      mtry trees min_n .metric .estimator  mean     n std_err .config
##     <dbl> <dbl> <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1      5  1570   118 spec    binary     0.782     5 0.00941 Preprocessor1_Model~
## 2      4  1470   114 spec    binary     0.781     5 0.00787 Preprocessor1_Model~
## 3      5  1570   114 spec    binary     0.780     5 0.00949 Preprocessor1_Model~
## 4      4  1370   114 spec    binary     0.780     5 0.00669 Preprocessor1_Model~
## 5      4  1470   118 spec    binary     0.780     5 0.00754 Preprocessor1_Model~
## 6      4  1270   118 spec    binary     0.780     5 0.00852 Preprocessor1_Model~
## 7      5  1270   114 spec    binary     0.780     5 0.00939 Preprocessor1_Model~
## 8      4  1370   118 spec    binary     0.780     5 0.00826 Preprocessor1_Model~
## 9      4  1570   114 spec    binary     0.779     5 0.00676 Preprocessor1_Model~
## 10     5  1470   114 spec    binary     0.779     5 0.00939 Preprocessor1_Model~
```

```r
show_best(rf_tuned_2, metric = 'bal_accuracy', n = 10)
```

```
## # A tibble: 10 x 9
##      mtry trees min_n .metric      .estimator  mean     n std_err .config
##     <dbl> <dbl> <dbl> <chr>        <chr>      <dbl> <int>   <dbl> <chr>
## 1      4  1470   114 bal_accuracy binary     0.760     5 0.0112  Preprocessor1_~
## 2      5  1270   114 bal_accuracy binary     0.759     5 0.0104  Preprocessor1_~
## 3      4  1470   116 bal_accuracy binary     0.758     5 0.0115  Preprocessor1_~
## 4      4  1370   116 bal_accuracy binary     0.758     5 0.00988 Preprocessor1_~
## 5      4  1370   118 bal_accuracy binary     0.758     5 0.0115  Preprocessor1_~
## 6      4  1270   116 bal_accuracy binary     0.758     5 0.0107  Preprocessor1_~
## 7      4  1270   118 bal_accuracy binary     0.758     5 0.0117  Preprocessor1_~
## 8      4  1270   114 bal_accuracy binary     0.758     5 0.0109  Preprocessor1_~
## 9      4  1570   118 bal_accuracy binary     0.757     5 0.0126  Preprocessor1_~
## 10     4  1370   114 bal_accuracy binary     0.757     5 0.0108  Preprocessor1_~
```

###Modelo final

```r
## Definir la mejor combinacion
rf_pars_fin <- select_best(rf_tuned_2, metric = 'sens')
## Finalizar (darle valores a parametros tuneables) el workflow
rf_wflow_fin <-
rf_wflow %>%
  finalize_workflow(rf_pars_fin)
rf_wflow_fin
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 8 Recipe Steps
##
```

```
## * step_rm()
## * step_impute_knn()
## * step_normalize()
## * step_other()
## * step_novel()
## * step_dummy()
## * step_nzv()
## * step_upsample()
##
## -- Model -----------------------------------------------------------------------
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = 4
##   trees = 1470
##   min_n = 114
##
## Engine-Specific Arguments:
##   importance = impurity
##
## Computational engine: ranger
```

###Entrenamiento del modelo

Entrenar el modelo final

```r
rf_fitted <- fit(rf_wflow_fin, train)
rf_fitted
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: rand_forest()
##
## -- Preprocessor ----------------------------------------------------------------
## 8 Recipe Steps
##
## * step_rm()
## * step_impute_knn()
## * step_normalize()
## * step_other()
## * step_novel()
## * step_dummy()
## * step_nzv()
## * step_upsample()
##
## -- Model -----------------------------------------------------------------------
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, mtry = min_cols(~4,      x), num.trees = ~1470, min.n
##
## Type:                             Probability estimation
## Number of trees:                  1470
## Sample size:                      7864
## Number of independent variables:  30
## Mtry:                             4
```

```
## Target node size:                  114
## Variable importance mode:          impurity
## Splitrule:                         gini
## OOB prediction error (Brier s.):   0.1476555
```

###Modelo sin workflow

```r
rf_model_fin <- extract_fit_parsnip(rf_fitted)
```

###Evaluación del Modelo

Evaluación en la data de entrenamiento

```r
train %>%
predict(rf_fitted , new_data = . ) %>%
mutate(Real= train$Churn) %>%
conf_mat(truth = Real, estimate = .pred_class ) %>%
summary
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.793
##  2 kap                  binary         0.524
##  3 sens                 binary         0.791
##  4 spec                 binary         0.794
##  5 ppv                  binary         0.581
##  6 npv                  binary         0.913
##  7 mcc                  binary         0.537
##  8 j_index              binary         0.585
##  9 bal_accuracy         binary         0.792
## 10 detection_prevalence binary         0.361
## 11 precision            binary         0.581
## 12 recall               binary         0.791
## 13 f_meas               binary         0.670
```

###Evaluación en la data de prueba

```r
test %>%
predict(rf_fitted, new_data = . ) %>%
mutate(Real= test$Churn) %>%
  conf_mat(truth = Real, estimate = .pred_class ) %>%
summary
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.779
##  2 kap                  binary         0.481
##  3 sens                 binary         0.727
##  4 spec                 binary         0.798
##  5 ppv                  binary         0.565
##  6 npv                  binary         0.890
##  7 mcc                  binary         0.489
##  8 j_index              binary         0.525
##  9 bal_accuracy         binary         0.763
## 10 detection_prevalence binary         0.341
## 11 precision            binary         0.565
```

```
## 12 recall               binary         0.727
## 13 f_meas               binary         0.636
```

###¿Que variables pueden estar relacionadas más con el abandono de clientes?

```
library(vip)
rf_model_fin %>%
  vip(geom = "point")
```

model_Room_8_files/figure-latex/unnamed-chunk-57-1.pdf

Modelo Boosting XGBoost

Como parte del ejercicio de selección del mejor modelo para la determinación de la mejor estrategia para retención de clientes, se utiliza el algoritmo XGboost con la finalidad de elegir el modelo con mayor poder predictivo.

Para tal efecto, la aplicación del algoritmo XGboost inicia a partir de la receta establecida para los datos de abandono de clientes del servicio de telecomunicaciones.

###Especificación del modelo

```
xgb_sp <- boost_tree(mtry = tune(), trees = tune(),
  loss_reduction = tune(), learn_rate= tune() ) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

xgb_sp %>%
  translate()
```

```
## Boosted Tree Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = tune()
##   learn_rate = tune()
##   loss_reduction = tune()
##
## Computational engine: xgboost
##
## Model fit template:
## parsnip::xgb_train(x = missing_arg(), y = missing_arg(), weights = missing_arg(),
##     colsample_bynode = tune(), nrounds = tune(), eta = tune(),
##     gamma = tune(), nthread = 1, verbose = 0)
```

###Afinamiento de Malla de búsqueda

Previo a establecer la malla de búsqueda el algoritmo requiere la data incorporada las funciones establecidas en la receta, por tal motivo se aplica prep y bake. Posteriormente, se obtiene la malla de búsqueda

```
receta_prep = prep(receta, train)
finalize(mtry(), bake(receta_prep, new_data = NULL ))
```

```
## # Randomly Selected Predictors (quantitative)
## Range: [1, 31]
```

```
set.seed(123)
xgb_grid <- xgb_sp %>%
  parameters() %>%
  finalize(bake(receta_prep, new_data = NULL)) %>%
  grid_latin_hypercube(size = 10)

xgb_grid
```

```
## # A tibble: 10 x 4
##     mtry trees learn_rate loss_reduction
##    <int> <int>      <dbl>          <dbl>
##  1    20   670    0.00125        3.86e-5
##  2    25   866    0.108          6.57e-5
##  3    23   338    0.0284         3.16e-3
##  4    13  1324    0.00197        1.10e-6
##  5     4   573    0.307          2.46e-8
##  6     9  1170    0.0456         1.10e-9
##  7    28  1750    0.0566         3.05e-9
##  8     6   125    0.00670        1.22e-1
##  9    12  1904    0.00443        1.98e+1
## 10    18  1441    0.0142         1.06e+0
```

###Work Flow

```
xgb_wflow <-
  workflow() %>%
  add_recipe(receta) %>%
  add_model(xgb_sp)

xgb_wflow
```

```
## == Workflow ===================================================================
## Preprocessor: Recipe
## Model: boost_tree()
##
## -- Preprocessor ---------------------------------------------------------------
## 8 Recipe Steps
##
## * step_rm()
## * step_impute_knn()
## * step_normalize()
## * step_other()
## * step_novel()
## * step_dummy()
## * step_nzv()
## * step_upsample()
##
## -- Model ----------------------------------------------------------------------
## Boosted Tree Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = tune()
```

```
##   learn_rate = tune()
##   loss_reduction = tune()
##
## Computational engine: xgboost
```

###Entrenamiento de Malla de Busqueda en la Crossvalidation - Xgboost

```r
set.seed(123)

xgb_tuned <- tune_grid(
  xgb_wflow,
  resamples= cv,
  grid = xgb_grid,
  metrics = metricas,
  control= control_grid(allow_par = T, save_pred = T)
  )

xgb_tuned
```

```
## # Tuning results
## # 5-fold cross-validation using stratification
## # A tibble: 5 x 5
##   splits              id    .metrics          .notes           .predictions
##   <list>              <chr> <list>            <list>           <list>
## 1 <split [4507/1127]> Fold1 <tibble [40 x 8]> <tibble [0 x 3]> <tibble>
## 2 <split [4507/1127]> Fold2 <tibble [40 x 8]> <tibble [0 x 3]> <tibble>
## 3 <split [4507/1127]> Fold3 <tibble [40 x 8]> <tibble [0 x 3]> <tibble>
## 4 <split [4507/1127]> Fold4 <tibble [40 x 8]> <tibble [0 x 3]> <tibble>
## 5 <split [4508/1126]> Fold5 <tibble [40 x 8]> <tibble [0 x 3]> <tibble>
```

###Mejor modelo

Evaluamos que modelo resulto mejor

```r
show_best(xgb_tuned, metric = 'accuracy', n = 10)
```

```
## # A tibble: 10 x 10
##     mtry trees learn_rate loss_reduction .metric  .estimator  mean     n std_err
##    <int> <int>      <dbl>          <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
##  1     9  1170    0.0456         1.10e-9 accuracy binary     0.758     5 0.00439
##  2    13  1324    0.00197        1.10e-6 accuracy binary     0.757     5 0.00891
##  3    12  1904    0.00443        1.98e+1 accuracy binary     0.757     5 0.00622
##  4    18  1441    0.0142         1.06e+0 accuracy binary     0.757     5 0.00483
##  5     6   125    0.00670        1.22e-1 accuracy binary     0.756     5 0.0106
##  6    23   338    0.0284         3.16e-3 accuracy binary     0.755     5 0.00716
##  7    28  1750    0.0566         3.05e-9 accuracy binary     0.752     5 0.00301
##  8    20   670    0.00125        3.86e-5 accuracy binary     0.752     5 0.00673
##  9    25   866    0.108          6.57e-5 accuracy binary     0.752     5 0.00425
## 10     4   573    0.307          2.46e-8 accuracy binary     0.748     5 0.00541
## # i 1 more variable: .config <chr>
```

```r
show_best(xgb_tuned, metric = 'sens', n = 10)
```

```
## # A tibble: 10 x 10
##     mtry trees learn_rate loss_reduction .metric .estimator  mean     n std_err
##    <int> <int>      <dbl>          <dbl> <chr>   <chr>      <dbl> <int>   <dbl>
##  1    12  1904    0.00443        1.98e+1 sens    binary     0.771     5 0.00640
##  2     6   125    0.00670        1.22e-1 sens    binary     0.759     5 0.0123
```

```
## 3 13 1324 0.00197 1.10e-6 sens binary 0.752 5 0.0137
## 4 20 670 0.00125 3.86e-5 sens binary 0.747 5 0.0137
## 5 23 338 0.0284 3.16e-3 sens binary 0.719 5 0.0126
## 6 18 1441 0.0142 1.06e+0 sens binary 0.699 5 0.00872
## 7 9 1170 0.0456 1.10e-9 sens binary 0.648 5 0.00936
## 8 28 1750 0.0566 3.05e-9 sens binary 0.629 5 0.00512
## 9 25 866 0.108 6.57e-5 sens binary 0.628 5 0.00881
## 10 4 573 0.307 2.46e-8 sens binary 0.613 5 0.00932
## # i 1 more variable: .config <chr>
```

```r
show_best(xgb_tuned, metric = 'spec', n = 10)
```

```
## # A tibble: 10 x 10
##     mtry trees learn_rate loss_reduction .metric .estimator  mean     n std_err
##    <int> <int>      <dbl>          <dbl> <chr>   <chr>      <dbl> <int>   <dbl>
##  1     9  1170    0.0456         1.10e-9 spec    binary     0.797     5 0.00292
##  2    28  1750    0.0566         3.05e-9 spec    binary     0.797     5 0.00347
##  3     4   573    0.307          2.46e-8 spec    binary     0.797     5 0.00432
##  4    25   866    0.108          6.57e-5 spec    binary     0.796     5 0.00269
##  5    18  1441    0.0142         1.06e+0 spec    binary     0.777     5 0.00361
##  6    23   338    0.0284         3.16e-3 spec    binary     0.769     5 0.00536
##  7    13  1324    0.00197        1.10e-6 spec    binary     0.758     5 0.00790
##  8     6   125    0.00670        1.22e-1 spec    binary     0.756     5 0.0112
##  9    20   670    0.00125        3.86e-5 spec    binary     0.754     5 0.00503
## 10    12  1904    0.00443        1.98e+1 spec    binary     0.752     5 0.00639
## # i 1 more variable: .config <chr>
```

```r
show_best(xgb_tuned, metric = 'bal_accuracy', n = 10)
```

```
## # A tibble: 10 x 10
##     mtry trees learn_rate loss_reduction .metric  .estimator  mean     n std_err
##    <int> <int>      <dbl>          <dbl> <chr>    <chr>      <dbl> <int>   <dbl>
##  1    12  1904    0.00443        1.98e+1 bal_acc~ binary     0.761     5 0.00617
##  2     6   125    0.00670        1.22e-1 bal_acc~ binary     0.757     5 0.0106
##  3    13  1324    0.00197        1.10e-6 bal_acc~ binary     0.755     5 0.0102
##  4    20   670    0.00125        3.86e-5 bal_acc~ binary     0.750     5 0.00876
##  5    23   338    0.0284         3.16e-3 bal_acc~ binary     0.744     5 0.00886
##  6    18  1441    0.0142         1.06e+0 bal_acc~ binary     0.738     5 0.00603
##  7     9  1170    0.0456         1.10e-9 bal_acc~ binary     0.723     5 0.00592
##  8    28  1750    0.0566         3.05e-9 bal_acc~ binary     0.713     5 0.00325
##  9    25   866    0.108          6.57e-5 bal_acc~ binary     0.712     5 0.00569
## 10     4   573    0.307          2.46e-8 bal_acc~ binary     0.705     5 0.00657
## # i 1 more variable: .config <chr>
```

###Selección del modelo final

La selección de los hipepáramestros es con base al mejor modelo de sensibilidad y "bal_accuracy", por tanto se el modelo final resulta ser el modelo 04, no se realizá un proceso búsqueda manual del mejor modelo debido a la amplitud de valores que puede tomar el learn_rate y la función de costo, así como la discrecionalidad de la amplitud de búsqueda.

```r
xgb_pars_fin <- select_best(xgb_tuned, metric = 'sens')
xgb_wflow_fin <-
  xgb_wflow %>%
  finalize_workflow(xgb_pars_fin)

xgb_wflow_fin
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: boost_tree()
##
## -- Preprocessor ----------------------------------------------------------------
## 8 Recipe Steps
##
## * step_rm()
## * step_impute_knn()
## * step_normalize()
## * step_other()
## * step_novel()
## * step_dummy()
## * step_nzv()
## * step_upsample()
##
## -- Model -----------------------------------------------------------------------
## Boosted Tree Model Specification (classification)
##
## Main Arguments:
##   mtry = 12
##   trees = 1904
##   learn_rate = 0.00443167801773591
##   loss_reduction = 19.7893446768168
##
## Computational engine: xgboost
```

###Entrenar el modelo final

```
xgb_fitted <- fit(xgb_wflow_fin, train)
xgb_fitted
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: boost_tree()
##
## -- Preprocessor ----------------------------------------------------------------
## 8 Recipe Steps
##
## * step_rm()
## * step_impute_knn()
## * step_normalize()
## * step_other()
## * step_novel()
## * step_dummy()
## * step_nzv()
## * step_upsample()
##
## -- Model -----------------------------------------------------------------------
## ##### xgb.Booster
## raw: 6.3 Mb
## call:
##   xgboost::xgb.train(params = list(eta = 0.00443167801773591, max_depth = 6,
##     gamma = 19.7893446768168, colsample_bytree = 1, colsample_bynode = 0.4,
##     min_child_weight = 1, subsample = 1), data = x$data, nrounds = 1904L,
```

```
##     watchlist = x$watchlist, verbose = 0, nthread = 1, objective = "binary:logistic")
## params (as set within xgb.train):
##   eta = "0.00443167801773591", max_depth = "6", gamma = "19.7893446768168", colsample_bytree = "1", c
## xgb.attributes:
##   niter
## callbacks:
##   cb.evaluation.log()
## # of features: 30
## niter: 1904
## nfeatures : 30
## evaluation_log:
##     iter training_logloss
##        1        0.6916595
##        2        0.6901715
## ---
##     1903        0.4539354
##     1904        0.4539354
```

###Selección del modelo

```
xgb_model_fin <- pull_workflow_fit(xgb_fitted)
```

```
## Warning: `pull_workflow_fit()` was deprecated in workflows 0.2.3.
## i Please use `extract_fit_parsnip()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
xgb_model_fin
```

```
## parsnip model object
##
## ##### xgb.Booster
## raw: 6.3 Mb
## call:
##   xgboost::xgb.train(params = list(eta = 0.00443167801773591, max_depth = 6,
##     gamma = 19.7893446768168, colsample_bytree = 1, colsample_bynode = 0.4,
##     min_child_weight = 1, subsample = 1), data = x$data, nrounds = 1904L,
##     watchlist = x$watchlist, verbose = 0, nthread = 1, objective = "binary:logistic")
## params (as set within xgb.train):
##   eta = "0.00443167801773591", max_depth = "6", gamma = "19.7893446768168", colsample_bytree = "1", c
## xgb.attributes:
##   niter
## callbacks:
##   cb.evaluation.log()
## # of features: 30
## niter: 1904
## nfeatures : 30
## evaluation_log:
##     iter training_logloss
##        1        0.6916595
##        2        0.6901715
## ---
##     1903        0.4539354
##     1904        0.4539354
```

###Evaluación del modelo en la data de entrenamiento y prueba

```
train %>%
  predict(xgb_fitted , new_data = . ) %>%
  mutate(Real= train$Churn) %>%
  conf_mat(truth = Real, estimate = .pred_class ) %>%
  summary
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.772
##  2 kap                  binary         0.487
##  3 sens                 binary         0.787
##  4 spec                 binary         0.767
##  5 ppv                  binary         0.550
##  6 npv                  binary         0.909
##  7 mcc                  binary         0.504
##  8 j_index              binary         0.554
##  9 bal_accuracy         binary         0.777
## 10 detection_prevalence binary         0.380
## 11 precision            binary         0.550
## 12 recall               binary         0.787
## 13 f_meas               binary         0.647
```

```
test %>%
  predict(xgb_fitted , new_data = . ) %>%
  mutate(Real= test$Churn) %>%
  conf_mat(truth = Real, estimate = .pred_class ) %>%
  summary
```

```
## # A tibble: 13 x 3
##    .metric              .estimator .estimate
##    <chr>                <chr>          <dbl>
##  1 accuracy             binary         0.772
##  2 kap                  binary         0.477
##  3 sens                 binary         0.754
##  4 spec                 binary         0.779
##  5 ppv                  binary         0.552
##  6 npv                  binary         0.898
##  7 mcc                  binary         0.489
##  8 j_index              binary         0.533
##  9 bal_accuracy         binary         0.766
## 10 detection_prevalence binary         0.363
## 11 precision            binary         0.552
## 12 recall               binary         0.754
## 13 f_meas               binary         0.637
```

###Importancia de las variables en el modelo

```
vip(xgb_model_fin)
```

model_Room_8_files/figure-latex/unnamed-chunk-71-1.pdf

Analizando la importancia de las variables en el modelo se verifica que el gasto mensual y total presentan una alta importancia en las variables sin embargo el gasto total es una combinación lineal del gasto mensual, por lo que se podría excluir del modelo una de las variabales para probar si incrementa su capacidad de predicción con respecto a sensibilidad y bal_accuracy.

###Conclusión

Del análisis realizado se evidencia que ambos modelos (Random Forest y Xgboost) presentan muy buena estimación con relación a sensibilidad, es decir con la capacidad de predicción de los individuos que abandonan el servicio, no obstante Xgboost presenta un resultado superior en el test de prueba (0.74 > 0.72), por tanto el modelo seleccionado es Xgboost.