Proyecto Final Aprendizaje

Laura Lopez S. Jacome (144089) y Dante Ruiz Martínez (183340)

12/7/2018

Contents

1. Introducción

1.1 Problema

Según la Organización de las Naciones Unidas (ONU), la pobreza "va más allá de la falta de ingresos y recursos para garantizar unos medios de vida sostenibles. Entre sus manifestaciones se incluyen el hambre y la malnutrición, el acceso limitado a la educación y a otros servicios básicos, la discriminación y la exclusión sociales y la falta de participación en la adopción de decisiones". Pese a los avances en la reducción de la pobreza, la cantidad de personas que viven en condiciones de pobreza extrema en el mundo sigue siendo inaceptablemente alta. Es por esto que gobiernos e instituciones como el Inter-American Development Bank tienen programas de ayuda social. Sin embargo, muchos programas sociales tienen dificultades para asegurarse de que las personas adecuadas reciban la ayuda necesaria. Es especialmente difícil cuando un programa se enfoca en el segmento más pobre de la población. Los más pobres del mundo normalmente no pueden proporcionar los registros de ingresos y gastos necesarios para demostrar que califican. ¿Entonces cómo se puede saber si una persona es parte de la población objetivo a la que va dirigido el programa de apoyo?

En América Latina, un método popular utiliza un algoritmo para verificar la calificación de ingresos. Se llama la prueba de medios de proxy (o PMT). Con PMT, las agencias utilizan un modelo que considera los atributos del hogar observables de una familia como el material de sus paredes y techo, o los activos que se encuentran en el hogar para clarificarlos y predecir su nivel de necesidad. Si bien esto es una mejora, la precisión sigue siendo un problema a medida que la población de la región crece y la pobreza disminuye.

1.2 Objetivo

El objetivo de este proyecto es desarrollar un modelo de aprendizaje de máquina que pueda predecir el nivel de pobreza de los hogares utilizando las características tanto del hogar como de los individuales.

2. Datos

2.1 Visualiación de Datos

Los datos se obtuvieron de la competencia de Kaggle **"Costa Rican Household Poverty Level Prediction"**. Los datos de la competencia se proporcionan en dos archivos: **train.csv y test.csv**. El **conjunto de entrenamiento tiene 9,557 filas y 143 columnas**, mientras que el **conjunto de pruebas tiene 23,856 filas y 142 columnas**. **Cada fila representa a un individuo** y cada columna es una **característica**, ya sea única **para el individuo o del hogar del individuo**. El conjunto de entrenamiento tiene una columna adicional, **Target, que representa el nivel de pobreza en una escala del 1 al 4 y representa nuestra variable objetivo**.

- 1 = pobreza extrema
- 2 = pobreza moderada
- 3 = hogares vulnerables
- 4 = hogares no vulnerables

Cabe destacar que **el objetivo de la competencia es predecir el nivel de pobreza del hogar**. Sin embargo, **los datos proporcionados son de individuos**, cada uno con características tanto individuales como del hogar. Además, por instrucciones de la competencia, se debe hacer **una predicción por cada individuo** del conejunto de prueba pero se debe **entrenar el modelo únicamente con los datos de los jefes de familia**. Por lo que tendremos que hacer agregaciones de los datos para poder hacer las predicciones correctas.

2.2 Descripción del conjunto de entrenamiento

A continuación se presenta una lista de las 143 variables originales en el conjunto de entrenamiento.

```
names(data_train)
```

```
##   [1] "Id"              "v2a1"            "hacdor"
##   [4] "rooms"           "hacapo"          "v14a"
##   [7] "refrig"          "v18q"            "v18q1"
##  [10] "r4h1"            "r4h2"            "r4h3"
##  [13] "r4m1"            "r4m2"            "r4m3"
##  [16] "r4t1"            "r4t2"            "r4t3"
##  [19] "tamhog"          "tamviv"          "escolari"
##  [22] "rez_esc"         "hhsize"          "paredblolad"
##  [25] "paredzocalo"     "paredpreb"       "pareddes"
##  [28] "paredmad"        "paredzinc"       "paredfibras"
##  [31] "paredother"      "pisomoscer"      "pisocemento"
##  [34] "pisoother"       "pisonatur"       "pisonotiene"
##  [37] "pisomadera"      "techozinc"       "techoentrepiso"
##  [40] "techocane"       "techootro"       "cielorazo"
##  [43] "abastaguadentro" "abastaguafuera"  "abastaguano"
##  [46] "public"          "planpri"         "noelec"
##  [49] "coopele"         "sanitario1"      "sanitario2"
##  [52] "sanitario3"      "sanitario5"      "sanitario6"
##  [55] "energcocinar1"   "energcocinar2"   "energcocinar3"
##  [58] "energcocinar4"   "elimbasu1"       "elimbasu2"
##  [61] "elimbasu3"       "elimbasu4"       "elimbasu5"
##  [64] "elimbasu6"       "epared1"         "epared2"
##  [67] "epared3"         "etecho1"         "etecho2"
##  [70] "etecho3"         "eviv1"           "eviv2"
##  [73] "eviv3"           "dis"             "male"
##  [76] "female"          "estadocivil1"    "estadocivil2"
##  [79] "estadocivil3"    "estadocivil4"    "estadocivil5"
##  [82] "estadocivil6"    "estadocivil7"    "parentesco1"
##  [85] "parentesco2"     "parentesco3"     "parentesco4"
##  [88] "parentesco5"     "parentesco6"     "parentesco7"
##  [91] "parentesco8"     "parentesco9"     "parentesco10"
##  [94] "parentesco11"    "parentesco12"    "idhogar"
##  [97] "hogar_nin"       "hogar_adul"      "hogar_mayor"
## [100] "hogar_total"     "dependency"      "edjefe"
## [103] "edjefa"          "meaneduc"        "instlevel1"
```

```
## [106] "instlevel2"      "instlevel3"      "instlevel4"
## [109] "instlevel5"      "instlevel6"      "instlevel7"
## [112] "instlevel8"      "instlevel9"      "bedrooms"
## [115] "overcrowding"    "tipovivi1"       "tipovivi2"
## [118] "tipovivi3"       "tipovivi4"       "tipovivi5"
## [121] "computer"        "television"      "mobilephone"
## [124] "qmobilephone"    "lugar1"          "lugar2"
## [127] "lugar3"          "lugar4"          "lugar5"
## [130] "lugar6"          "area1"           "area2"
## [133] "age"             "SQBescolari"     "SQBage"
## [136] "SQBhogar_total"  "SQBedjefe"       "SQBhogar_nin"
## [139] "SQBovercrowding" "SQBdependency"   "SQBmeaned"
## [142] "agesq"           "Target"
```

El dataset de entrenamiento tiene 9,557 observaciones con 143 variables. Abajo se observa que las variables son de dos tipos: continuas y categóricas. La mayoría de las características son categóricas pues responden a preguntas como "tiene o no" alguna característica el hogar o el individuo. Las variables categóricas algunas son binarias y otras multiclase.

El conjunto de entrenamiento tiene los siguientes tipos de datos: cadenas de caracteres, números enteros y números continuos.

```
table(sapply(data_train, class)) %>% knitr::kable(caption = "Número de Variables por tipo en el conjunto
```

Table 1: Número de Variables por tipo en el conjunto de entrenamiento

Var1	Freq
character	5
integer	132
numeric	6

También hay presencia de NAs.

- "v2a1": 6860
- "v18q1": 7342
- "rez_esc": 7928
- "meaneduc": 5
- "SQBmeaned": 5

Las variables en el conjunto de entrenamiento con NAs son el ingreso mensual (v2a1), el número de tabletas que tiene el hogar (v18q1), los años de resago escolar (rez_esc), años promedio de educación en los adultos (meaneduc) y años de escolaridad de miembros del hogar mayores de 18 años al cuadrado (SQBmeaned).

2.3 Descripción del conjunto de prueba

Se tienen los mismos tipos de variables que en el conjunto de prueba.

```
table(sapply(data_test, class))  %>% knitr::kable(caption = "Número de Variables por tipo en el conjunto
```

Table 2: Número de Variables por tipo en el conjunto de prueba

Var1	Freq
character	5
integer	131
numeric	6

El conjunto de prueba tiene los mismos tipos de variables que el conjunto de entrenamiento. Solo no incluye la variable Target o variable respuesta, lo cual era de esperarse.

Asimismo, el mismo número de variables son las que presentan datos nulos. Sin embargo, es de notar que las variables v2a1, v18q1 y rez_esc tienen miles de datos nulos que habrá que tratar más adelante.

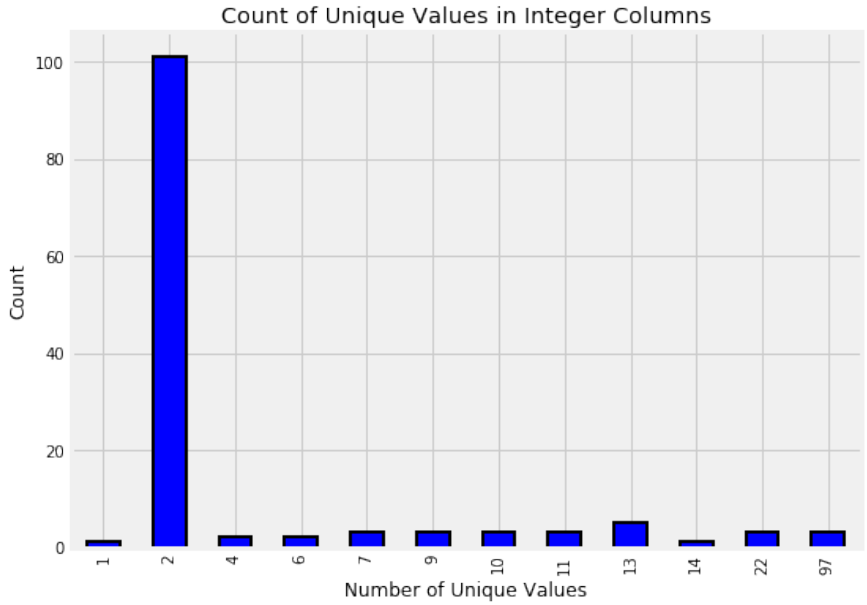
- "v2a1": 17403
- "v18q1": 18126
- "rez_esc": 19653
- "meaneduc" : 31
- "SQBmeaned" : 31

En el conjunto de prueba las variables con datos faltantes (NAs) son el ingreso mensual (v2a1), número de tabletas que el hogar tiene (v18q1), los años de rezago escolar (rez_esc), años promedio de educación en los adultos (meaneduc) y años de escolaridad de miembros del hogar mayores de 18 años al cuadrado (SQBmeaned).

3. Análisis exploratorio de los datos

3.1 Variables numéricas

La siguiente gráfica muestra el número de valores únicos por cada una de las variables con tipo numérico.

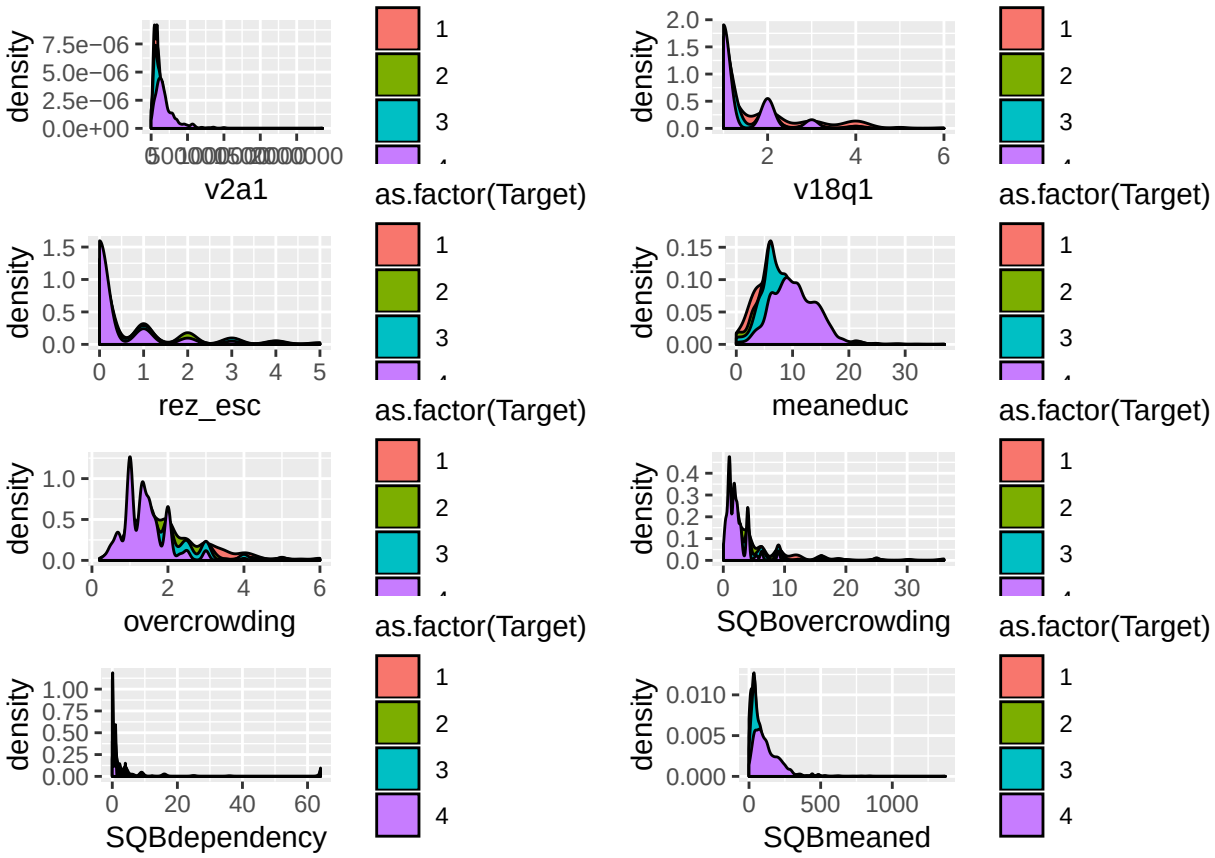


Se concluye que las variables con dos valores únicos son booleanas y las que tienen más de dos valores son multicategóricas.

3.2 Variables continuas

Las siguientes gráficas muestran la distribución de los campos con tipo continuo. Estas gráficas, nos van a ayudar a ver si hay una diferencia significativa en la distribución de la variable según el nivel de pobreza de los hogares.

```
##v2a1
g1 <- ggplot(data = data_train, aes(x = v2a1, fill = as.factor(Target))) +
  geom_density()
##v18q1
g2 <- ggplot(data = data_train, aes(x = v18q1, fill = as.factor(Target))) +
  geom_density()
##rez_esc
g3 <- ggplot(data = data_train, aes(x = rez_esc, fill = as.factor(Target))) +
  geom_density()
##meaneduc
g4 <- ggplot(data = data_train, aes(x = meaneduc, fill = as.factor(Target))) +
  geom_density()
##overcrowding
g5 <- ggplot(data = data_train, aes(x = overcrowding, fill = as.factor(Target))) +
  geom_density()
##SQBovercrowding
g6 <- ggplot(data = data_train, aes(x = SQBovercrowding, fill = as.factor(Target))) +
  geom_density()
##SQBdependency
g7 <- ggplot(data = data_train, aes(x = SQBdependency, fill = as.factor(Target))) +
  geom_density()
##SQBmeaned
g8 <-ggplot(data = data_train, aes(x = SQBmeaned, fill = as.factor(Target))) +
  geom_density()

library(gridExtra)
grid.arrange(g1,g2,g3,g4,g5,g6,g7,g8, ncol = 2)
```



Más adelante, calcularemos las correlaciones entre las variables y la variable de interés "Target" para medir las relaciones entre las características, pero estas gráficas ya pueden darnos una idea de qué variables pueden ser más "relevantes" para un modelo. Por ejemplo, el meaneduc, que representa la educación promedio de los adultos en el hogar, parece estar relacionado con el nivel de pobreza: un promedio más alto de la educación de adultos conduce a valores más altos de la meta, que son niveles de pobreza menos severos.

3. Datos con caracteres

Las siguientes características vienen en formato de cadena de caracteres.

```
data_train %>% select(Id,idhogar, dependency, edjefe, edjefa) %>% head() %>% knitr::kable(caption = "Ca
```

Table 3: Características con múltiples tipos de entrada

Id	idhogar	dependency	edjefe	edjefa
ID_279628684	21eb7fcc1	no	10	no
ID_f29eb3ddd	0e5d7a658	8	12	no
ID_68de51c94	2c7317ea8	8	no	11
ID_d671db89c	2b58d945f	yes	11	no
ID_d56d6f5f5	2b58d945f	yes	11	no
ID_ec05b1a7b	2b58d945f	yes	11	no

En las variables de dependency, edjefe y edjefa se observa que hay una combinación de datos numéricos con

caracteres si/no. Según la documentación de estas columnas:

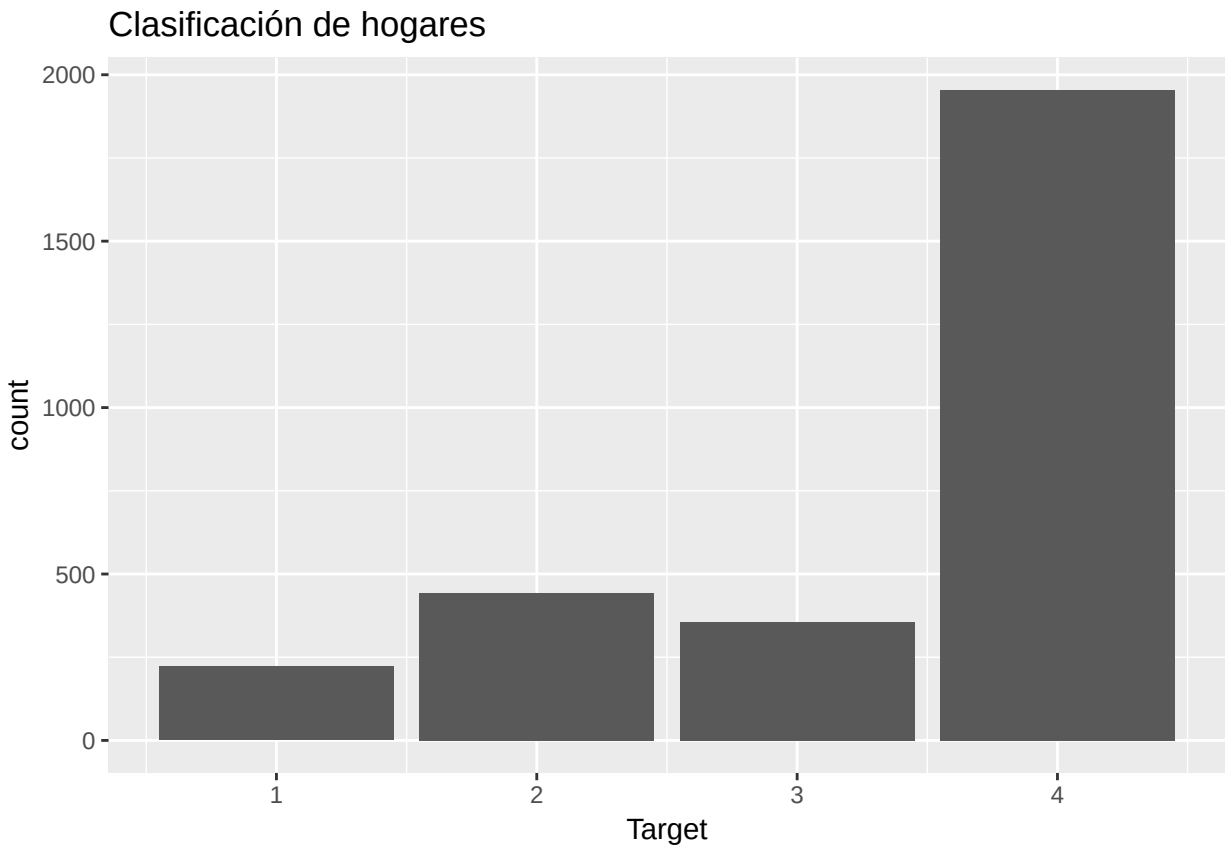
- dependencia: tasa de dependencia, calculada = (número de miembros del hogar menores de 19 años o mayores de 64) / (número de miembros del hogar entre 19 y 64 años)
- edjefe: años de educación del jefe de hogar masculino, basados en la interacción de escolari (años de educación), jefe de hogar y género, sí = 1 y no = 0
- edjefa: años de educación de mujeres jefas de hogar, basados en la interacción de escolari (años de educación), cabeza de familia y género, sí = 1 y no = 0

Estas explicaciones aclaran el problema. Para estas tres variables, "sí" = 1 y "no" = 0. Podemos corregir las variables mediante una asignación y convertirlas en numéricas.

3.4 Variable respuesta (Target)

Gratificamos la distribución de Target.

```
data_train %>% select(idhogar,parentesco1,Target) %>%
  filter(parentesco1 == 1) %>%
  ggplot(aes(Target)) +
  geom_bar(stat = 'count') +
  ggtitle("Clasificación de hogares")
```



Se puede observar que tenemos un conjunto de entrenamiento desbalanceado. Donde tenemos 63% de hogares no vulnerables (4), 17% de hogares en pobreza moderada (2), 13% de hogares vulnerables y 7% de hogares en extrema pobreza.

```r
table(data_train$Target) %>% knitr::kable(caption = "Frecuencias de la variable Target")
```

Table 4: Frecuencias de la variable Target

Var1	Freq
1	755
2	1597
3	1209
4	5996

```r
prop.table(table(data_train$Target)) %>% knitr::kable(caption = "Proporciones de la variable Target")
```

Table 5: Proporciones de la variable Target

Var1	Freq
1	0.0789997
2	0.1671026
3	0.1265041
4	0.6273935

4. Limpieza de datos

Para la limpieza de datos se juntaron el conjunto de entrenamiento y el de prueba. Para hacer esto posible se agregó una variable Target con valores nulos para el conjunto de prueba.

```r
data_train <- data_train %>% mutate(conjunto = "train")
data_test <- data_test %>% mutate(Target = NA,
                                  conjunto = "test")
data_complete <- rbind(data_train, data_test)
```

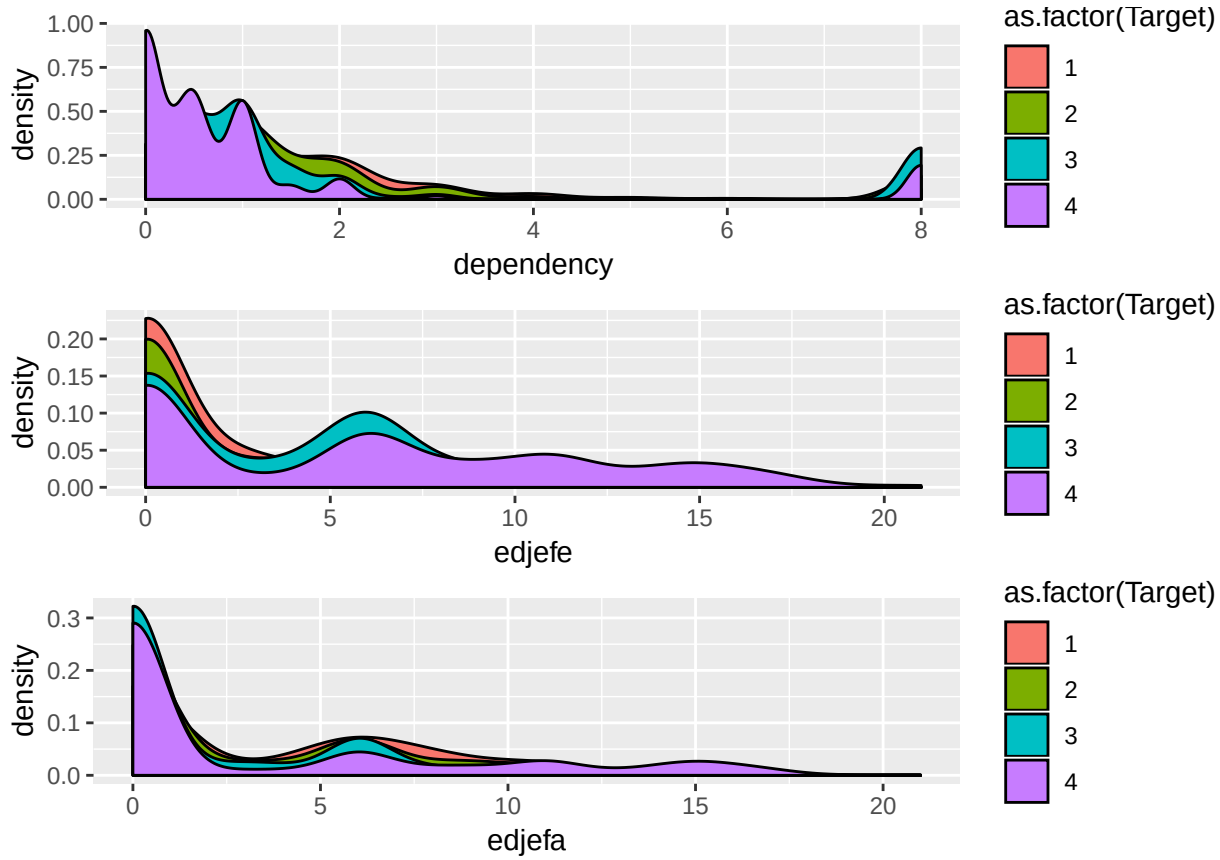
4.1 Corrección de variables con caracteres

Revisando la documentación se identificó que las características de dependency, edjefe y edjefa con valores yes y no, corresponden a 1 y 0 respectivamente. Por lo que se corrige esto.

```r
data_complete <- data_complete %>%
  mutate(dependency = recode(dependency, `yes`= "1", `no`= "0"),
         edjefe = recode(edjefe, `yes`= "1", `no`= "0"),
         edjefa = recode(edjefa, `yes`= "1", `no`= "0")) %>%
  mutate(dependency = as.numeric(dependency),
         edjefe = as.numeric(edjefe),
         edjefa = as.numeric(edjefa))

campos_limpios <- data_complete %>%
  select(Target, dependency,edjefe,edjefa, parentesco1) %>%
  filter(!is.na(Target)) %>%
  filter(parentesco1 == 1)
```

La distribución de estas características de tipo numérico discreto se ven así.

```
d1 <- ggplot(data = campos_limpios,
             aes(x = dependency, fill = as.factor(Target),
                 group = Target)) +
  geom_density()

d2 <- ggplot(data = campos_limpios,
             aes(x = edjefe, fill = as.factor(Target))) +
  geom_density()

d3 <- ggplot(data = campos_limpios,
             aes(x = edjefa, fill = as.factor(Target))) +
  geom_density()

grid.arrange(d1,d2,d3)
```



4.2 Corrección del label Target

En los grupos de discusión de la competencia se menciona que en el conjunto de datos hay errores, porque en los hogares hay individuos que pertenecen a un mismo hogar y tienen diferente nivel de pobreza ("Target").

Se identificaron 335 individuos mal clasificados:

```
hogares_mal_clasificados <- data_complete %>%
  select(idhogar, Target, conjunto) %>%
  filter(conjunto == "train") %>%
  group_by(idhogar) %>%
  unique() %>%
  summarise(categorias = n()) %>%
  arrange(desc(categorias)) %>%
  filter(categorias > 1)

data_complete %>%
  select(idhogar, Target, parentesco1, conjunto) %>%
  filter(conjunto == "train", idhogar %in% hogares_mal_clasificados$idhogar) %>%
  head(10) %>%
  knitr::kable(caption = "La variable target tiene errores de clasificación a nivel individuo")
```

Table 6: La variable target tiene errores de clasificación a nivel individuo

idhogar	Target	parentesco1	conjunto
4b6077882	1	1	train
4b6077882	2	0	train
4b6077882	2	0	train
6833ac5dc	2	0	train
6833ac5dc	2	0	train
6833ac5dc	2	0	train
6833ac5dc	2	1	train
6833ac5dc	2	0	train
6833ac5dc	1	0	train
43b9c83e5	2	0	train

Se encontraron 85 hogares mal clasificados. En la tabla arriba se observa que hay hogares hasta con tres valores diferentes de target. Para corregir esto, los anfitriones de la competencia dicen que se debe asignar el valor de "Target" del jefe de familia a lo demás integrantes del hogar.

```
## Para arreglar este problema se saca el Target del hogar para el jefe de hogar.
targets_correctos <- data_complete %>%
  select(idhogar, Target, conjunto, parentesco1) %>%
  filter(conjunto == "train", idhogar %in% hogares_mal_clasificados$idhogar,  parentesco1 == 1) %>%
  select(idhogar, Target)

## Hacemos un left join de los targets_correctos a data_complete
data_complete <- left_join(data_complete, targets_correctos, by="idhogar") %>%
  mutate(Target = ifelse(!is.na(Target.y),Target.y,Target.x))

## Se observa que se corrigió el error en las etiquetas de target
data_complete %>%
  select(idhogar, Target, conjunto) %>%
  filter(conjunto == "train") %>%
  group_by(idhogar) %>%
  unique() %>%
  summarise(categorias = n()) %>%
```

```
  arrange(desc(categorias)) %>%
  filter(categorias > 1)
```

```
## # A tibble: 0 x 2
## # ... with 2 variables: idhogar <chr>, categorias <int>
```

```
## Se eliminan variables auxiliares
data_complete$Target.x <- NULL
data_complete$Target.y <- NULL
```

4.3 Corrección de valores faltantes

A continuación se limpian los valores NA de las variables.

4.3.1 Limpiar NAs en variable v2a1

Juntando los datos de entrenamiento y prueba en la variable de pago de renta mensual tenemos 24,263 NAs. Esta variable es de las más importantes para determinar el nivel de pobreza, por lo que deberemos imputar valores en esta variable, para poder correr nuestros modelos más.

Estadísticas descriptivas de la variable Pago de Renta Mensual

```
summary(data_complete$v2a1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       0   80000  135000  172031  200000 2852700   24263
```

```
data_complete <- data_complete %>%
  mutate(tipovivi_ = ifelse(data_complete$tipovivi1 == 1, 1,
                            ifelse(data_complete$tipovivi2 == 1, 2,
                                   ifelse(data_complete$tipovivi3 == 1, 3,
                                          ifelse(data_complete$tipovivi4 == 1, 4,
                                                 ifelse(data_complete$tipovivi5 == 1, 5,8))))))
```

En la siguiente gráfica se presenta el número de datos faltantes por tipo de vivienda para encontrar algún patrón.

```
data_complete %>%
  select(tipovivi_, v2a1) %>%
  mutate(tipovivi_ = factor(tipovivi_),
         missing_val = is.na(v2a1)) %>%
  group_by(missing_val, tipovivi_) %>%
  summarise(conteo = n()) %>%
  ggplot(aes(x= factor(tipovivi_), y= conteo, fill = missing_val)) +
  geom_bar(stat = "identity") +
  ggtitle("Datos faltantes en la variable en pago mensual de la renta por tipo de vivienda") +
  xlab("Tipo de vivienda (tipovivi_)")
```

Cuando agrupamos la variable de pago mensual de la renta por posesión del hogar, se puede ver un patrón. Las observaciones que tienen NAs son solamente los hogares que tienen una casa pero ya la pagaron (tipovivi1 = 1), tienen un hogar precario (tipovivi4 = 1) y tienen un hogar asignado o prestado (tipovivi5 = 1).

Con los resultados anteriores, se tomó la decisión de imputar un cero en la variable v2a1 donde hay NAs para el caso donde el hogar es propietario de la casa, el hogar es precario y cuando el hogar es prestado o asignado. Esto porque no afecta el sentido de la información.

```
# Imputar ceros donde hay NAs
data_complete <- data_complete %>%
  mutate(v2a1 = ifelse(is.na(v2a1), 0, v2a1))
```

4.3.2 Limpiar NAs en variable v18q1

Construimos una tabla cruzada entre los valores faltantes de la variable tiene tableta y número de tabletas.

```
missing_v18q1_df <- data_complete %>%
  select(parentesco1,v18q, v18q1) %>%
  filter(parentesco1 == 1) %>%
  mutate(is_missing_tablet = is.na(v18q1))

table(missing_v18q1_df$v18q, missing_v18q1_df$is_missing_tablet) %>% knitr::kable(caption = "Tabla cruza
```

Table 7: Tabla cruzada de valores faltantes de la variable tiene tableta vs número de tabletas

	FALSE	TRUE
0	0	8044
1	2263	0

En la tabla anterior se puede ver que todos los individuos que no poseen una tableta tienen missing values en el campo de número de tabletas que tiene el hogar. De esta manera tomamos la decisión de imputar un cero a estas variable cuando hay NAs.

```
data_complete <- data_complete %>% mutate(v18q1 = ifelse(is.na(v18q1),0,v18q1))
```

4.3.3 Limpiar NAs en variable rez_esc

La variable de rezago escolar presenta 1161 valores faltantes.

Estadísticas descriptivas de rezago escolar

```
summary(data_complete$rez_esc)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.000   0.000   0.435   0.000  99.000   27581
```

Por la información del dataset se sabe que esta variable solo esta definida para edades de los individuos entre 7 y 19 años por lo que se imputaron ceros para los valores de la variable rez_esc que estaban fuera del rango de edad. Asimismo se corrigió el valor del nivel de 99 en rez_esc que debía ser 5. Los missing values que quedan son los que corresponden a las edades de 7 a 19.

```
data_complete <-  data_complete %>%
  mutate(rez_esc = ifelse(rez_esc > 5, 5, rez_esc),
         rez_esc = ifelse(!is.na(rez_esc), rez_esc,
                          ifelse(age > 19,0,
                                 ifelse(age < 7, 0, rez_esc))))

data_complete %>%
  select(rez_esc, age) %>%
  group_by(rez_esc) %>%
  summarise(edad_promedio = max(age)) %>%
  knitr::kable(caption = "Edad promedio por rezago escolar")
```

Table 8: Edad promedio por rezago escolar

rez_esc	edad_promedio
0	97
1	17
2	17
3	17
4	17
5	17
NA	19

5. Ingeniería de características

En esta parte vamos a proceder a agregar la base de datos a nivel hogar el cual es requerido para hacer predicciones. Para hacer esto, empezamos clasificando las variables de acuerdo a si son características a nivel individual o a nivel hogar y de acuerdo a su tipo (numérico, booleano, ordenado).

```
### Variables id a nivel hogar
id_ = c('Id', 'idhogar', 'Target')

### Variables booleanas a nivel individuo
ind_bool = c('v18q', 'dis', 'male', 'female', 'estadocivil1', 'estadocivil2', 'estadocivil3',
             'estadocivil4', 'estadocivil5', 'estadocivil6', 'estadocivil7',
             'parentesco1', 'parentesco2',  'parentesco3', 'parentesco4', 'parentesco5',
             'parentesco6', 'parentesco7', 'parentesco8',  'parentesco9', 'parentesco10',
             'parentesco11', 'parentesco12', 'instlevel1', 'instlevel2', 'instlevel3',
             'instlevel4', 'instlevel5', 'instlevel6', 'instlevel7', 'instlevel8',
             'instlevel9', 'mobilephone')
### Variables categóricas ordenadas
ind_ordered = c('rez_esc', 'escolari', 'age')

## Variables booleanas a nivel hogar
hh_bool = c('hacdor', 'hacapo', 'v14a', 'refrig', 'paredblolad', 'paredzocalo',
            'paredpreb','pisocemento', 'pareddes', 'paredmad',
            'paredzinc', 'paredfibras', 'paredother', 'pisomoscer', 'pisoother',
            'pisonatur', 'pisonotiene', 'pisomadera',
            'techozinc', 'techoentrepiso', 'techocane', 'techootro', 'cielorazo',
            'abastaguadentro', 'abastaguafuera', 'abastaguano',
            'public', 'planpri', 'noelec', 'coopele', 'sanitario1',
            'sanitario2', 'sanitario3', 'sanitario5',   'sanitario6',
            'energcocinar1', 'energcocinar2', 'energcocinar3', 'energcocinar4',
            'elimbasu1', 'elimbasu2', 'elimbasu3', 'elimbasu4',
            'elimbasu5', 'elimbasu6', 'epared1', 'epared2', 'epared3',
            'etecho1', 'etecho2', 'etecho3', 'eviv1', 'eviv2', 'eviv3',
            'tipovivi1', 'tipovivi2', 'tipovivi3', 'tipovivi4', 'tipovivi5',
            'computer', 'television', 'lugar1', 'lugar2', 'lugar3',
            'lugar4', 'lugar5', 'lugar6', 'area1', 'area2')
### Variables ordinales a nivel house hold
hh_ordered = c('rooms', 'r4h1', 'r4h2', 'r4h3', 'r4m1','r4m2','r4m3', 'r4t1',  'r4t2',
               'r4t3', 'v18q1', 'tamhog','tamviv','hhsize','hogar_nin',
               'hogar_adul','hogar_mayor','hogar_total',  'bedrooms', 'qmobilephone')

### Variables continuas a nivel hogar
hh_cont = c('v2a1', 'dependency', 'edjefe', 'edjefa', 'meaneduc', 'overcrowding')

### Variables cuadráticas
sqr_ = c('SQBescolari', 'SQBage', 'SQBhogar_total', 'SQBedjefe',
         'SQBhogar_nin', 'SQBovercrowding', 'SQBdependency', 'SQBmeaned', 'agesq')
```

Una vez agrupadas las variables, decidimos quitar las variables cuadráticas ya que no las vamos a utilizar en nuestros modelos.

```
data_complete <- data_complete %>% select(-one_of(sqr_))
```

5.1 Correlación de variables a nivel hogar

Revisamos aquellas variables a nivel hogar que están altamente correlacionadas. Se encontró que las variables tamviv, hogar_total, hhsize, r4t3, area1, public.

```
## Correlación de variables a nivel hogar
data_household <- data_complete %>% filter(parentesco1 == 1) %>%
  select(-one_of( ind_bool, ind_ordered))

data_household_num <- data_household %>% select(-id_,-conjunto)

df_cor <- rquery.cormat(data_household_num, type="flatten", graph=FALSE)
df_cor[["r"]]  %>%
  filter(abs(cor) > 0.95) %>%
  knitr::kable(caption = "Variables del hogar con alta correlación")
```

Table 9: Variables del hogar con alta correlación

row	column	cor	p
hogar_total	hhsize	1.00	0
hogar_total	r4t3	1.00	0
hhsize	r4t3	1.00	0
hogar_total	tamhog	1.00	0
hhsize	tamhog	1.00	0
r4t3	tamhog	1.00	0
area1	area2	-1.00	0
public	coopele	-0.97	0

Al revisar la correlación de variables a nivel hogar se encontró que las variables tamviv, hogar_total, hhsize, r4t3, area1, public están altamente correlacionadas.

5.2 Correlación con hhsize

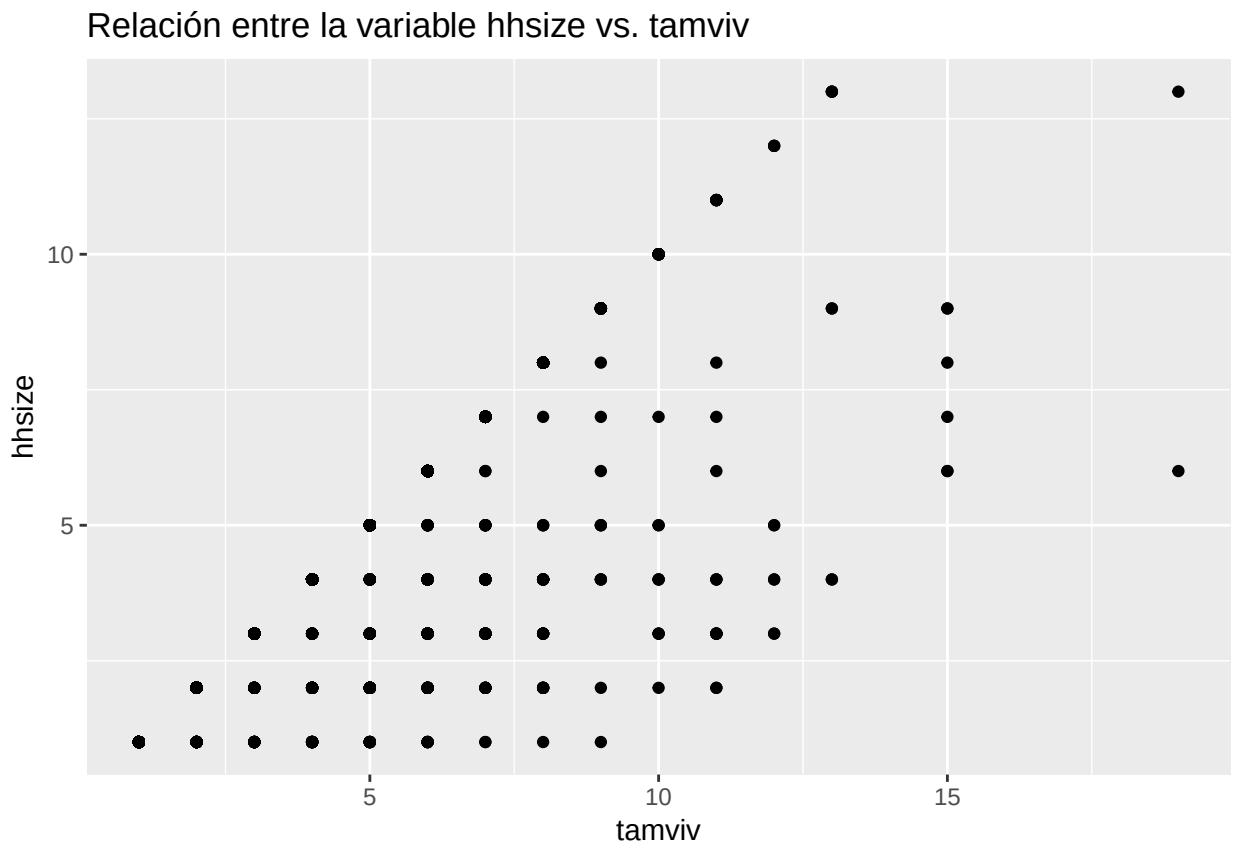
Por descripción de la base de datos, hhsize, tamhog, hogar_total y r4t3 nos dan información acerca del número de individuos viviendo en el hogar. Es por esto que presentan una correlación casi perfecta.

Concluimos que de esas 4 variables nos vamos a quedar con una sola ya que las demás son redundantes en el modelo. Es por esto que nos vamos a quedar con hhsize y quitamos tamhog, hogar_total y r4t3

```
data_household <- data_household %>% select(-tamhog, -hogar_total, -r4t3)
```

En el caso de tamviv, no se excluyó ya que aunque tiene una correlación alta con hhsize son cosas diferentes. tamviv es la cantidad de personas que viven en la casa y hhsize es la cantidad de miembros de la familia. Es por esto que vamos a investigar más a fondo la diferencia entre estas variables.

```
ggplot(data_household,aes(x=tamviv, y=hhsize))+geom_point() +
  ggtitle("Relación entre la variable hhsize vs. tamviv")
```



En la gráfica anterior se puede observar que en muchos casos hay más personas viviendo en la casa que integrantes de la familia. La diferencia entre estas variables puede ser indicativa por lo que la vamos a incluir en los datos.

```
data_household <- data_household%>% mutate(hhsize_diff= tamviv-hhsize)
```

5.3 Correlación de variables con area1

```
df_cor[["r"]] %>% filter(row == "area1") %>%
  filter(abs(cor) > 0.95) %>%
  knitr::kable(caption = "Correlación de variables con area1")
```

Table 10: Correlación de variables con area1

row	column	cor	p
area1	area2	-1	0

Area1 está relacionada con area 1. Area1 indica si la vivienda está en una urbe lo cual es redundante con area2 que indica si la vivienda se encuentra en zona rural. Es por esto que vamos a quitar area2.

```
data_household<- data_household %>% select(-area2)
```

5.4 Creación de variables ordinales

5.4.1 Correlacion de variables con public

```
df_cor[["r"]]  %>% filter(row == "public") %>%
  filter(abs(cor) > 0.95) %>%
  knitr::kable(caption = "Correlación de variables con public")
```

Table 11: Correlación de variables con public

row	column	cor	p
public	coopele	-0.97	0

Se puede observar que la variable public esta relacionada con coopele. Estas variables hacen referencia a la fuente de electricidad de la casa.

Hay 4 variables que nos hablan de la electricidad en la casa: public (electricidad publica), coopele (electricidad cooperativa), noelec (no electricidad) y planpri(electricidad de fuente privada)

Estas 4 variables las vamos a juntar en 1 sola variable ordinal

1) no electricidad
2) electricidad de cooperativo
3) electricidad publica
4) electricidad privada

```
data_household <- data_household %>%
  mutate(elec = ifelse(data_household$noelec == 1, 1,
                       ifelse(data_household$coopele == 1, 2,
                              ifelse(data_household$public == 1, 3,
                                     ifelse(data_household$planpri == 1, 4,NA)))))%>%
  mutate(elec_missing = is.na(elec))%>%
  select(-public, -coopele, -noelec, -planpri)
```

5.4.2 Correlación de variables de pared

El estado de las paredes se divide en 3 variables booleanas que indican el estado bueno, regular y malo.

Vamos a juntar estas 3 variables en una sola que indique el estado de la pared 1) malo 2) regular 3) bueno

```
data_household <- data_household %>%
  mutate(e_pared_ = ifelse(data_household$epared1 == 1, 1,
                           ifelse(data_household$epared2 == 1, 2,
                                  ifelse(data_household$epared3 == 1, 3,NA))))%>%
  select(-epared1, -epared2, -epared3)
```

5.4.3 Correlación de variables de techo

El estado del TECHO se divide en 3 boolean que indican el estado bueno, regular y malo.

Vamos a juntar estas 3 variables en una sola que indique el estado de la pared 1) malo 2) regular 3) buen

```
data_household <- data_household %>%
  mutate(e_techo_ = ifelse(data_household$etecho1 == 1, 1,
                           ifelse(data_household$etecho2 == 1, 2,
                                  ifelse(data_household$etecho3 == 1, 3,NA))))%>%

  select(-etecho1, -etecho2, -etecho3)
```

5.4.4 Correlación de variables de suelo

El estado del SUELO se divide en 3 boolean que indican el estado bueno, regular y malo.

Vamos a juntar estas 3 variables en una sola que indique el estado de la pared 1) malo 2) regular 3) bueno

```
data_household <- data_household %>%
  mutate(e_suelo_ = ifelse(data_household$eviv1 == 1, 1,
                           ifelse(data_household$eviv2 == 1, 2,
                                  ifelse(data_household$eviv3 == 1, 3,NA)))) %>%

  select(-eviv1, -eviv2, -eviv3)
```

5.5 Correlación de variables a nivel individuo

A continuación se presenta la ingeniería de características a nivel individuo.

```
data_ind <- data_complete  %>%
  select(-one_of( hh_cont, hh_ordered, hh_bool))
data_ind_num <- data_ind %>% select(-id_,-conjunto)

df_cor_id <- rquery.cormat(data_ind_num, type="flatten", graph=FALSE)[["r"]]
df_cor_id %>%
  filter(abs(cor) > 0.95) %>%
  knitr::kable(caption = "Correlación de la variable male")
```

Table 12: Correlación de la variable male

row	column	cor	p
male	female	-1	0

5.5.1 Creación de variables ordinales

En el caso de variables individuales, también hay variables que podemos reemplazar por una sola variable ordinal.

5.5.1.1 Creación de variable ordinal para nivel de educación

Nueva variable ordinal para el nivel de educación:

1) no tiene ningún nivel de educación
2) primaria incompleta
3) primaria completa
4) secundaria incompleta
5) secundaria completa
6) secundaria técnica incompletas
7) secundaria técnica completa
8) licenciatura
9) posgrado

```
data_ind <- data_ind %>%
  mutate(inst_level = ifelse(data_ind$instlevel1 == 1, 1,
                        ifelse(data_ind$instlevel2 == 1, 2,
                               ifelse(data_ind$instlevel3 == 1, 3,
                                      ifelse(data_ind$instlevel4 == 1, 4,
                                             ifelse(data_ind$instlevel5==1,5,
                                                    ifelse(data_ind$instlevel6==1,6,
                                                           ifelse(data_ind$instlevel7==1,7,
                                                                  ifelse(data_ind$instlevel8==1,8,
                                                                         ifelse(data_ind$instlevel9==1,9
  select(-instlevel1, -instlevel2, -instlevel3, -instlevel4,-instlevel5,-instlevel6,-instlevel7,-instlev
```

6. Feature Construction

6.1 Variables a nivel hogar

6.1.1 Calidad del hogar

Agregaremos una variable que me indique el estado total de la vivienda.

```
data_household <- data_household%>%
  mutate(house_quality = e_pared_+e_techo_+e_suelo_)%>%
  mutate(house_quality_2 = e_pared_*e_techo_*e_suelo_)

data_household%>% select(house_quality, house_quality_2, Target)%>%
  group_by(Target)%>%
  summarise(prom_house_qual= mean(house_quality),
            prom_house_qual2 = mean(house_quality_2)) %>%
  knitr::kable(caption = "Variables que resumen la calidad de las viviendas por nivel de pobreza")
```

Table 13: Variables que resumen la calidad de las viviendas por nivel de pobreza

Target	prom_house_qual	prom_house_qual2
1	6.265766	11.32432
2	6.590498	12.91176
3	7.095775	15.36620
4	7.923234	20.11617
NA	7.475729	17.64890

Se puede observar que los Targets con menor nivel de pobreza tienen los valores más altos de la calidad de la vivienda.

6.1.2 Calidad de servicios

Ya que tenemos una variable de la calidad del hogar, se va a crear una variable que nos indica si la casa está en mal estado (no tiene piso, no tiene agua, no tiene techo).

```
data_household <- data_household %>%
  mutate(noelec = ifelse(elec==1,1,0))%>%
  mutate(noTecho = ifelse(cielorazo==0, 1, 0))%>%
  mutate(worst_qual = (sanitario1 + pisonotiene + abastaguano + noelec + noTecho)*-1)%>%
    select(-noelec, -noTecho)
```

6.1.3 Propiedad de electrodomésticos

Creamos una variable de propiedad de electrodomésticos por hogar. Esta variable resume la tenencia de refrigeradores, computadoras, televisión y tabletas.

```
data_household<- data_household%>%
   mutate(tabletas = ifelse(v18q1>0,1,0))%>%
  mutate(electrodomesticos = refrig+computer+television + tabletas)%>%
  select(-tabletas)
```

6.1.4 Variables indicadoras per cápita

Creamos las siguientes variables por personas que integran el tamaño del hogar.

- Teléfonos por miembros del hogar
- Tabletas por miembros del hogar
- Cuartos por miembros del hogar
- Renta mensual por miembros del hogar

```
data_household <- data_household %>%
  mutate(tel_pc = qmobilephone/tamviv)%>%
  mutate(tabletas_pc = v18q1/tamviv)%>%
  mutate(cuartos_pc = rooms/tamviv)%>%
  mutate(renta_pc = v2a1/tamviv)
```

6.2 Construcción de variables a nivel individuo

6.2.1 Variables de años de escolaridad, nivel educativo y tecnología

A nivel de individuo se crearon las siguientes variables:

- Escolari_age: Proporción de años de escolaridad respecto a la edad del individuo.
- Inst_age: Nivel educativo proporcional a la edad del individuo
- Tech: Número de tabletas y teléfonos con los que dispone el individuo.

```
data_ind<- data_ind%>%
  mutate(escolari_age = ifelse(age ==0, NA, escolari/age))%>%
  mutate(inst_age = ifelse(age ==0, NA, inst_level/age))%>%
  mutate(tech=v18q+mobilephone)

#cbind(colSums(is.na(data_ind)))
```

6.2.3 Agregación de las características a nivel individuo

Agregamos las variables a nivel individual para contar solo con características a nivel hogar para poder hacer las predicciones.

Se crearon distintas agregaciones para cada una de las características individuales utilizando las siguientes operaciones: mínimo, máximo, promedio, frecuencia y rango.

```
# Crear función de rango
rango <- function(x){
  max(x)-min(x)
}
# Agregar las características individuales por min, max, mean, length, rango
data_ind_agg<- data_ind %>%
  select( -Target, -Id, -conjunto) %>%
  group_by(idhogar) %>%
  summarise_all(c("min","max","mean","length","rango"))
```

```
## Correlacion de las variables agregadas
# df_cor_ind <- rquery.cormat(select(data_ind_agg,-idhogar), type="flatten", graph=FALSE)[["r"]]
# #df_cor_ind %>% filter(abs(cor) > 0.95)
```

6.3 Construcción de la base de datos a nivel hogar para modelación

Se concatenan las bases de datos con características a nivel hogar, con la de características a nivel individual previamente agregadas a nivel hogar. Esta nueva base de datos contiene 10,307 observaciones y 257 variables. Es preciso mencionar que hasta este punto esta base de datos con características limpias considera ambos conjuntos entrenamiento y prueba.

```
# Unir bases de datos
data_final_hh <- left_join(x = data_household ,
                           y = data_ind_agg,
                           by = 'idhogar')
dim(data_final_hh)
```

```
## [1] 10307   257
```

Por último se construye una nueva variable donde el jefe de familia es mujer, ya que esta variable puede tener alguna relación con el nivel de pobreza.

```
## Agregar variable de cuando el jefe de familia es mujer
jefes_mujeres_df <- data_ind %>%
  select(idhogar, parentesco1, female) %>%
  filter(parentesco1 == 1) %>%
```

```
  mutate(is_jefe_mujer = ifelse(female == 1, 1, 0)) %>%
  select(idhogar, is_jefe_mujer)

data_final_hh <- left_join(x = data_final_hh ,
                           y = jefes_mujeres_df,
                           by = 'idhogar')
```

Asimismo, producto de la ingeniería de características se agregaron 115 nuevas variables.

Estimación de Modelos

A continuación se presenta el ajuste de diferentes modelos, con el objetivo de encontrar alguno que se desempeñe mejor con el conjunto de entrenamiento. Para comparar los modelos se utiliza la misma métrica que se utiliza la en la competencia de Kaggle, el F-Macro Score.

$MacroF1 = (F1Class1 + F2Class2 + F3Class3 + F4Class4)/4$

Esta medida consiste en el promedio armónico del F-Score para cada una de las categorías del nivel de pobreza.

Considerando lo anterior se construyen las funciones auxiliares para estimar el F-Macro-Score.

```
## Función F-Macro
ConfusionMatrix <- function(y_pred, y_true) {
  Confusion_Mat <- table(y_true, y_pred)
  return(Confusion_Mat)
}


ConfusionDF <- function(y_pred, y_true) {
  Confusion_DF <- transform(as.data.frame(ConfusionMatrix(y_pred, y_true)),
                            y_true = as.character(y_true),
                            y_pred = as.character(y_pred),
                            Freq = as.integer(Freq))
  return(Confusion_DF)
}

Precision_macro <- function(y_true, y_pred, labels = NULL) {
  Confusion_DF <- ConfusionDF(y_pred, y_true)

  if (is.null(labels) == TRUE) labels <- unique(c(y_true, y_pred))

  Prec <- c()
  for (i in c(1:length(labels))) {
    positive <- labels[i]

    tmp <- Confusion_DF[which(Confusion_DF$y_true==positive & Confusion_DF$y_pred==positive), "Freq"]
    TP <- if (length(tmp)==0) 0 else as.integer(tmp)
    tmp <- Confusion_DF[which(Confusion_DF$y_true!=positive & Confusion_DF$y_pred==positive), "Freq"]
    FP <- if (length(tmp)==0) 0 else as.integer(sum(tmp))

    Prec[i] <- TP/(TP+FP)
  }
```

```
  Prec[is.na(Prec)] <- 0
  Precision_macro <- mean(Prec) # sum(Prec) / length(labels)
  return(Precision_macro)
}



Recall_macro <- function(y_true, y_pred, labels = NULL) {
  Confusion_DF <- ConfusionDF(y_pred, y_true)

  if (is.null(labels) == TRUE) labels <- unique(c(y_true, y_pred))


  Rec <- c()
  for (i in c(1:length(labels))) {
    positive <- labels[i]

    tmp <- Confusion_DF[which(Confusion_DF$y_true==positive & Confusion_DF$y_pred==positive), "Freq"]
    TP <- if (length(tmp)==0) 0 else as.integer(tmp)

    tmp <- Confusion_DF[which(Confusion_DF$y_true==positive & Confusion_DF$y_pred!=positive), "Freq"]
    FN <- if (length(tmp)==0) 0 else as.integer(sum(tmp))

    Rec[i] <- TP/(TP+FN)
  }

  Rec[is.na(Rec)] <- 0
  Recall_macro <- mean(Rec) # sum(Rec) / length(labels)
  return(Recall_macro)
}


F1_Score_macro <- function(y_true, y_pred, labels = NULL) {
  if (is.null(labels) == TRUE) labels <- unique(c(y_true, y_pred))

  Precision <- Precision_macro(y_true, y_pred, labels)
  Recall <- Recall_macro(y_true, y_pred, labels)
  F1_Score_macro <- 2 * (Precision * Recall) / (Precision + Recall)
  return(F1_Score_macro)
}
```

Separamos nuevamente el conjunto de entrenamiento y prueba a partir del conjunto de datos limpios.

```
## Después de obtener la base de datos limpia,
#ahora se separa en conjunto de entrenamiento y prueba

data_train_hh_clean <- data_final_hh %>% filter(conjunto == "train")
data_test_hh_clean <- data_final_hh %>% filter(conjunto == "test")
```

Imputar datos con la mediana

Como se identificó en el análisis de datos exploratorios la base de datos limpia aún contiene datos faltantes, que deben recibir un tratamiento antes de comenzar a ajustar modelos.

Considerando que la mayoría de los datos tienen variables categóricas y que la variable continua más importante es el monto de la renta mensual, se optó por imputar los datos utilizando la mediana para cada una de las variables.

```
#### Imputación en el conjunto de Entrenamiento

median_meaneduc = median(data_train_hh_clean$meaneduc, na.rm = TRUE)
median_rez_esc_min = median(data_train_hh_clean$rez_esc_min, na.rm = TRUE)
median_inst_level_min = median(data_train_hh_clean$inst_level_min, na.rm = TRUE)
median_escolari_age_min = median(data_train_hh_clean$escolari_age_min, na.rm = TRUE)
median_inst_age_min = median(data_train_hh_clean$inst_age_min, na.rm = TRUE)
median_rez_esc_max = median(data_train_hh_clean$rez_esc_max, na.rm = TRUE)
median_escolari_age_max = median(data_train_hh_clean$escolari_age_max, na.rm = TRUE)
median_inst_age_max = median(data_train_hh_clean$inst_age_max, na.rm = TRUE)
median_rez_esc_mean = median(data_train_hh_clean$rez_esc_mean, na.rm = TRUE)
median_inst_level_mean = median(data_train_hh_clean$inst_level_mean, na.rm = TRUE)
median_escolari_age_mean = median(data_train_hh_clean$escolari_age_mean, na.rm = TRUE)
median_inst_age_mean = median(data_train_hh_clean$inst_age_mean, na.rm = TRUE)
median_rez_esc_rango = median(data_train_hh_clean$rez_esc_rango, na.rm = TRUE)
median_escolari_age_rango = median(data_train_hh_clean$escolari_age_rango, na.rm = TRUE)
median_inst_age_rango = median(data_train_hh_clean$inst_age_rango, na.rm = TRUE)
median_elec = median(data_train_hh_clean$elec, na.rm = TRUE)
median_worst_qual = median(data_train_hh_clean$worst_qual, na.rm = TRUE)
median_inst_level_rango = median(data_train_hh_clean$inst_level_rango, na.rm = TRUE)
median_inst_level_max = median(data_train_hh_clean$inst_level_max, na.rm = TRUE)

#### Vemos los siguientes NA en el conjunto de entrenamiento
#cbind(apply(is.na(data_train_hh_clean),2,sum))

data_train_imputada <- data_train_hh_clean %>%
  mutate(meaneduc = ifelse( is.na(meaneduc) ,
                            median_meaneduc , meaneduc ),
         rez_esc_min = ifelse( is.na(rez_esc_min) ,
                               median_rez_esc_min , rez_esc_min ),
         inst_level_min = ifelse( is.na(inst_level_min),
                                  median_inst_level_min ,
                                  inst_level_min ),
         escolari_age_min = ifelse( is.na(escolari_age_min) ,
                                    median_escolari_age_min ,
                                    escolari_age_min ),
         inst_age_min = ifelse( is.na(inst_age_min) ,
                                median_inst_age_min , inst_age_min ),
         rez_esc_max = ifelse( is.na(rez_esc_max) ,
                               median_rez_esc_max , rez_esc_max ),
         escolari_age_max = ifelse( is.na(escolari_age_max) ,
                                    median_escolari_age_max ,
                                    escolari_age_max ),
         inst_age_max = ifelse( is.na(inst_age_max) ,
                                median_inst_age_max ,
                                inst_age_max ),
         rez_esc_mean = ifelse( is.na(rez_esc_mean) ,
                                median_rez_esc_mean ,
                                rez_esc_mean ),
         inst_level_mean = ifelse( is.na(inst_level_mean) ,
```

```
                                median_inst_level_mean,
                                inst_level_mean),
        escolari_age_mean = ifelse( is.na(escolari_age_mean) ,
                                  median_escolari_age_mean ,
                                  escolari_age_mean ),
        inst_age_mean = ifelse(is.na(inst_age_mean),
                               median_inst_age_mean ,
                               inst_age_mean ),
        rez_esc_rango = ifelse(is.na(rez_esc_rango),
                               median_rez_esc_rango ,
                               rez_esc_rango ),
        escolari_age_rango = ifelse(is.na(escolari_age_rango),
                                    median_escolari_age_rango ,
                                    escolari_age_rango ),
        inst_age_rango = ifelse( is.na(inst_age_rango) ,
                                 median_inst_age_rango ,
                                 inst_age_rango),
        elec = ifelse( is.na(elec) , median_elec , elec ),
        worst_qual = ifelse( is.na(worst_qual) ,
                             median_worst_qual, worst_qual ),
        inst_level_rango = ifelse( is.na(inst_level_rango) ,
                                   median_inst_level_rango ,
                                   inst_level_rango ),
        inst_level_max = ifelse( is.na(inst_level_max) ,
                                 median_inst_level_max ,
                                 inst_level_max ))

#### Ya no hay NA en el conjunto de entrenamiento
#cbind(apply(is.na(data_train_imputada),2,sum))
```

Se procede a imputar los datos de prueba. Para evitar contaminar el conjunto de prueba, se imputaron los datos faltantes con las medianas calculadas en el conjunto de entrenamiento.

```
#### Prubea inst_level_max
#cbind(apply(is.na(data_test_hh_clean),2,sum))

data_test_imputada <- data_test_hh_clean %>%
  mutate(meaneduc = ifelse(is.na(meaneduc),
                           median(meaneduc, na.rm = TRUE),
                           meaneduc),
        rez_esc_min = ifelse(is.na(rez_esc_min),
                             median(rez_esc_min, na.rm = TRUE),
                             rez_esc_min),
        inst_level_min = ifelse(is.na(inst_level_min),
                                median(inst_level_min,
                                       na.rm = TRUE),
                                inst_level_min),
        escolari_age_min = ifelse(is.na(escolari_age_min),
                                  median(escolari_age_min,
                                         na.rm = TRUE),
                                  escolari_age_min),
        inst_age_min = ifelse(is.na(inst_age_min),
                              median(inst_age_min, na.rm = TRUE),
```

```
                              inst_age_min),
         rez_esc_max = ifelse(is.na(rez_esc_max),
                              median(rez_esc_max, na.rm = TRUE),
                              rez_esc_max),
         escolari_age_max = ifelse(is.na(escolari_age_max),
                                   median(escolari_age_max,
                                          na.rm = TRUE),
                                   escolari_age_max),
         inst_age_max = ifelse(is.na(inst_age_max),
                               median(inst_age_max, na.rm = TRUE),
                               inst_age_max),
         rez_esc_mean = ifelse(is.na(rez_esc_mean),
                               median(rez_esc_mean,
                                      na.rm = TRUE),
                               rez_esc_mean),
         inst_level_mean = ifelse(is.na(inst_level_mean),
                                  median(inst_level_mean,
                                         na.rm = TRUE),
                                  inst_level_mean),
         escolari_age_mean = ifelse(is.na(escolari_age_mean),
                                    median(escolari_age_mean,
                                           na.rm = TRUE),
                                    escolari_age_mean),
         inst_age_mean = ifelse(is.na(inst_age_mean),
                                median(inst_age_mean,
                                       na.rm = TRUE),
                                inst_age_mean),
         rez_esc_rango = ifelse(is.na(rez_esc_rango),
                                median(rez_esc_rango,
                                       na.rm = TRUE),
                                rez_esc_rango),
         escolari_age_rango = ifelse(is.na(escolari_age_rango),
                                     median(escolari_age_rango,
                                            na.rm = TRUE),
                                     escolari_age_rango),
         inst_age_rango = ifelse(is.na(inst_age_rango),
                                 median(inst_age_rango,
                                        na.rm = TRUE),
                                 inst_age_rango),
         elec = ifelse(is.na(elec),
                       median(elec, na.rm = TRUE),elec),
         worst_qual = ifelse(is.na(worst_qual),
                             median(worst_qual, na.rm = TRUE),
                             worst_qual),
         inst_level_rango = ifelse(is.na(inst_level_rango),
                                   median(inst_level_rango,
                                          na.rm = TRUE),
                                   inst_level_rango),
         inst_level_max = ifelse( is.na(inst_level_max) ,
                                  median_inst_level_max ,
                                  inst_level_max ))

### Vemos que ya no hay datos NA en el conjunto de prueba
```

```
#cbind(apply(is.na(data_test_imputada),2,sum))
```

Estandarizar datos

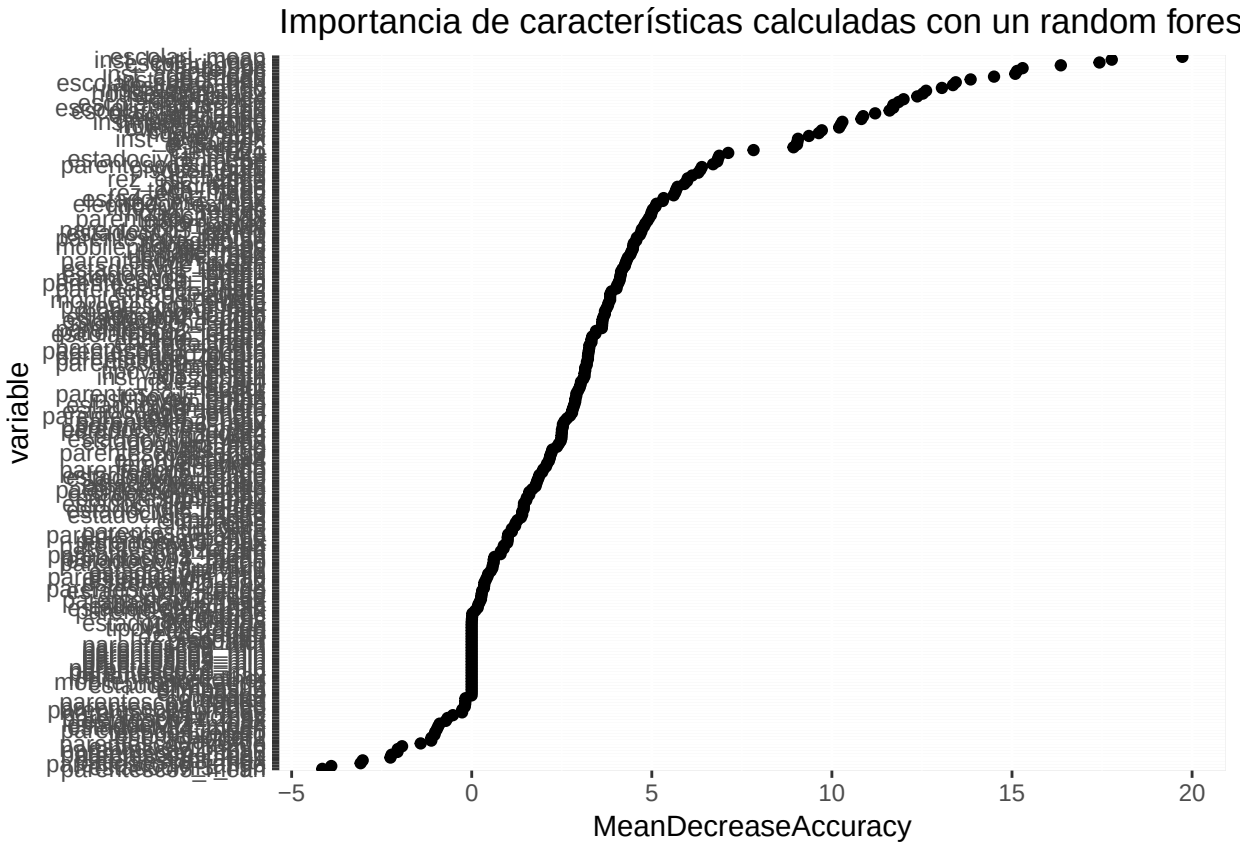
Al final no se estandarizaron los datos porque metían problemas al algoritmo de regresión logística, asimismo, el los métodos de árboles aleatorios y gboost no requieren utilizar la información estandarizada.

```
## Estandarizar bases de datos
train_transformed <- data_train_imputada
#### Conjunto de prueba
test_transformed <- data_test_imputada
```

Selección de variables

Se utilizó un random forest con el propósito de calcular las importancias de las características, a fin de identificar aquellas que más aportan para explicar el nivel de pobreza de los hogares.

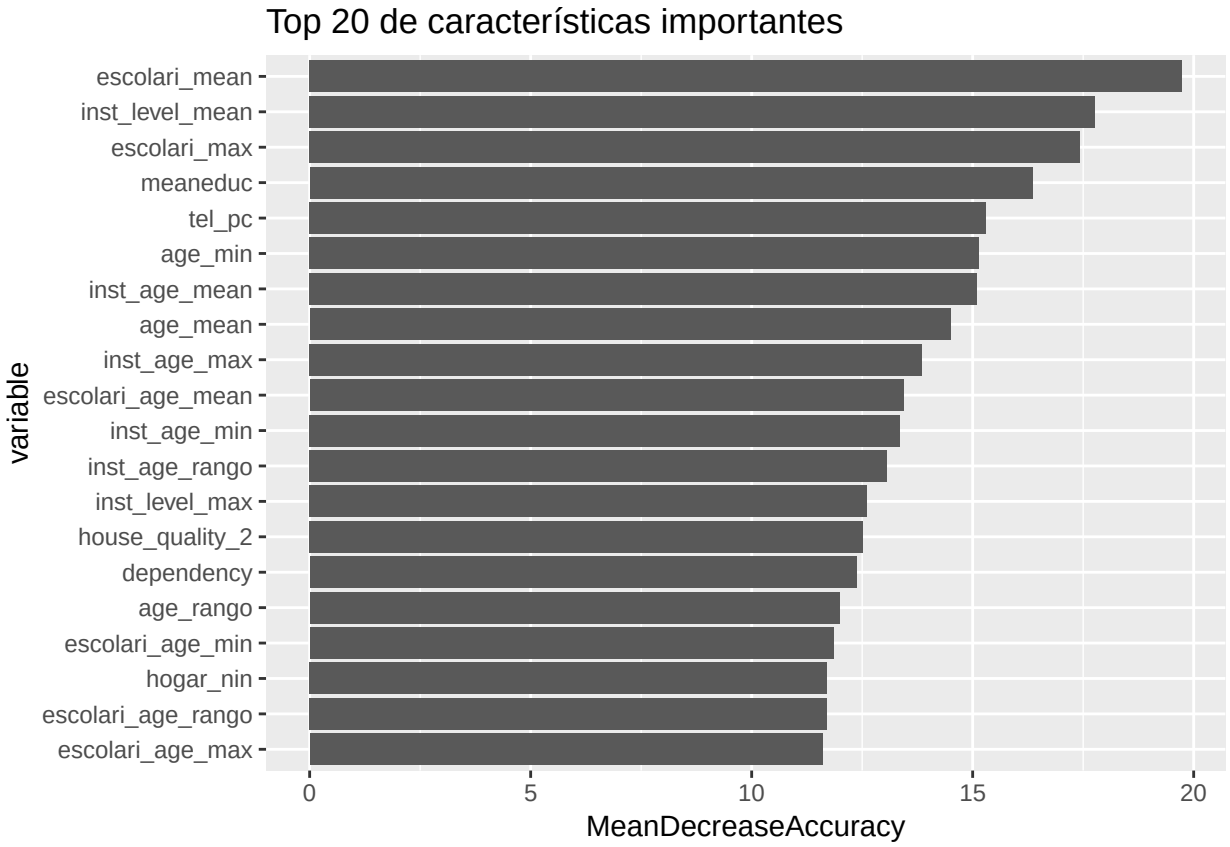
```
# Selección de variables
modelo_seleccion_var <-randomForest(factor(Target) ~ .,
                                    data = select(train_transformed,
                                                  -conjunto,
                                                  -elec_missing,
                                                  -idhogar, -Id),
                            ntree = 500, importance=TRUE)

imp <- importance(modelo_seleccion_var, type=1, scale = TRUE)
importancia_df <- data_frame(variable = rownames(imp),
                             MeanDecreaseAccuracy = imp[,1]) %>%
  arrange(desc(MeanDecreaseAccuracy))

importancia_df <- importancia_df %>%
  mutate(variable = reorder(variable, MeanDecreaseAccuracy))

ggplot(importancia_df , aes(x=variable, y= MeanDecreaseAccuracy)) +
  geom_point() +
  coord_flip() +
  ggtitle("Importancia de características calculadas con un random forest")
```



En la gráfica de importancias se aprecian en orden descendiente las características que mayor peso tienen para explicar el problema de la pobreza a nivel hogar. Podemos ver que hay variables con un mean decrease accuracy menor a cero, estas variables si se quitaran ayudarían a mejorar el modelo. Aquellas arriba de 0 si se quitan bajaría el performance del ajuste, sin embargo, no todas pesan lo mismo.

Las primeras 20 variables más importantes, tienen un mean decrease accuracy mayor a 11 son:

```
importancia_df %>%
  top_n(20) %>%
  ggplot(aes(x=variable,y=MeanDecreaseAccuracy)) +
      geom_bar(stat = 'identity') +
      coord_flip() +
      ggtitle("Top 20 de características importantes")
```



- años de escolaridad promedio (escolari_mean),
- grado académico promedio (inst_level_mean),
- años de escolaridad máxima (escolari_max),
- promedio de años de educación en mayores de 18 años (meaneduc),
- número de teléfonos (tel_pc), años del miembro más joven del hogar (age_min),
- promedio del nivel educativo como proporción de la edad (inst_age_mean),
- máximo del nivel educativo como proporción de la edad (inst_age_max),
- promedio de años de escolaridad como proporción de la edad de los individuos (escolari_age_mean).
- calidad de la casa en función del tipo de pared, techo y suelo (house_quality_2)
- dependency económica del hogar calculada en función del número de menores de edad y mayores de 64 años (dependency)
- Número de niños menores de edad (hogar_nin)
- Demás variables relacionadas con años de educación y nivel de estudios (inst_age_min, inst_age_rango, inst_level_max, escolari_age_min, age_rango, escolari_age_rango, escolari_age_max)

Del análisis de las 20 características más importantes, se puede confirmar que las variables relacionadas con la educación, años de escolaridad y nivel educactivo, en el hogar son los principales determinantes de la pobreza del hogar. Por otro lado, también aparecen variables como número de teléfonos, calidad de la casa, dependencia económica y número de menores de edad.

Aquellas variables que resultaron menos importantes, mean decrease accuracy negativo fueron aquellas que tienen que ver con:

- Parentesco
- Estado civil

- Si cuenta con sanitario en la propiedad
- Televisión
- Región de Chorotega
- No contar con cocina
- Si el material del techo es de fibra de cemento, metal o zinc
- Si el material de las paredes es de residuos
- Si la propiedad tiene una computadora
- Cuartos hacinados

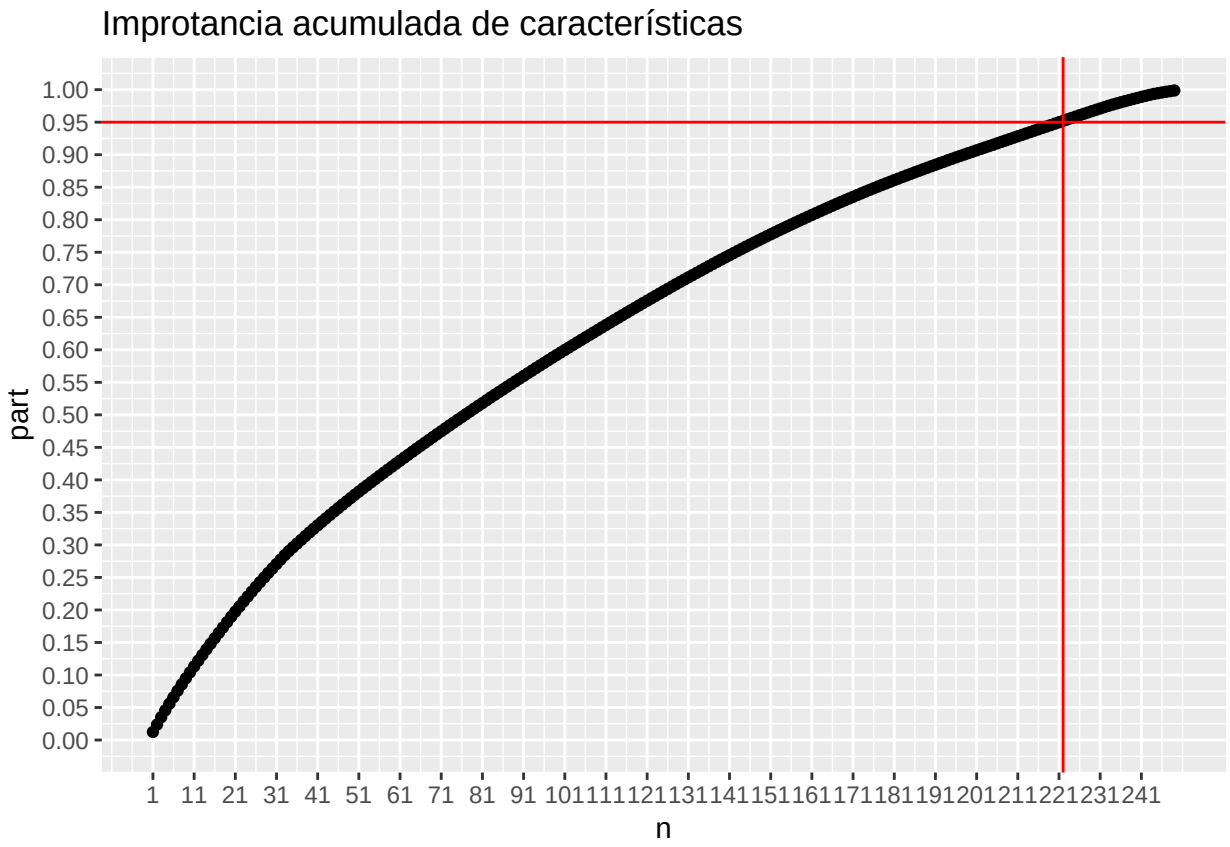
La mayoría de las variables tenían que ver con el parentesco de los individuos y estado civil. Algunas variables que resultaron sorprendentemente estar aquí fueron la disponibilidad de sanitario en el hogar, televisión, computadora, cuartos hacinados y materiales de construcción.

Importancia de características acumulativa.

En la siguiente gráfica se observa la importancia acumulativa de las variables con el propósito ver como se comporta la importancia en la medida que se incrementa el número de características.

```
importancia_df %>%
  mutate(z = (MeanDecreaseAccuracy - min(MeanDecreaseAccuracy))/(max(MeanDecreaseAccuracy)-min(MeanDecr
  mutate(acum = cumsum(z),
         part = acum/sum(z),
         n = 1:nrow(importancia_df)) %>%
  ggplot(aes(x = n, y = part)) +
    geom_point() +
    geom_vline(xintercept = 222, color = "red") +
    geom_hline(yintercept = 0.95, color = "red") +
    scale_x_continuous(limits = c(1,249), breaks = seq(1,249,10)) +
    scale_y_continuous(limits = c(0,1), breaks = seq(0,1,0.05)) +
    ggtitle("Improtancia acumulada de características")
```

```
## Warning: Removed 4 rows containing missing values (geom_point).
```



Se observa que para lograr una importancia acumulada de 95% de importancia se necesitan las 222 variables más importantes.

Una posibilidad para no utilizar todas las variables es utilizar estas 222 variables.

Construcción de conjuntos de entrenamiento y validación.

Con el propósito de entrenar y evaluar el desempeño de diferentes modelos para predecir el nivel de pobreza de los hogares se va a utilizar validación cruzada. Para lo cual se dividió la muestra de entrenamiento en 10 splits de entrenamiento y 10 de validación.

En la construcción de estos conjuntos se utilizaron las 222 características más importantes.

```
# Construir 5 conjuntos de validación cruzada a partir del conjunto de entrenamiento

## Seleccionar las variables de entrada

variables_modelo <- c("escolari_mean","inst_level_mean","inst_age_max","inst_age_mean",
                      "meaneduc","escolari_max","age_mean","dependency","escolari_age_mean",
                      "age_min","age_rango","tel_pc","house_quality_2","inst_age_min",
                      "escolari_age_max","inst_age_rango","inst_level_max","age_max",
                      "escolari_age_rango","house_quality","escolari_age_min","hogar_nin",
                      "qmobilephone","escolari_rango","hogar_adul","e_suelo_","escolari_min",
                      "inst_level_rango","e_techo_","overcrowding","e_pared_",
                      "inst_level_min","cuartos_pc","parentesco3_mean","r4h2","rez_esc_mean",
                      "r4t2","worst_qual","r4t1","cielorazo","estadocivil1_mean",
```

```
                      "electrodomesticos","bedrooms","renta_pc","estadocivil1_rango","v2a1",
                      "r4m3","hogar_mayor","r4m2","tamviv","lugar1","tech_mean","dis_mean",
                      "rez_esc_rango","rooms","tipovivi__min","parentesco7_length","r4m1",
                      "dis_rango","tech_min","energcocinar4","mobilephone_mean","tech_max",
                      "parentesco3_max","pisocemento","edjefe","tipovivi__mean","paredmad",
                      "rez_esc_max","parentesco3_rango","parentesco1_mean","female_rango",
                      "pisomoscer","estadocivil5_length","female_mean","paredblolad",
                      "lugar3","estadocivil1_max","estadocivil7_rango","estadocivil2_length",
                      "v18q_min","inst_level_length","paredpreb","v18q_max","edjefa",
                      "parentesco6_mean","parentesco3_length","r4h1","r4h3","tabletas_pc",
                      "v18q1","estadocivil6_mean","parentesco9_length","dis_length",
                      "escolari_length","tipovivi__max","parentesco2_rango","tipovivi5",
                      "parentesco12_length","estadocivil5_mean","parentesco8_length",
                      "parentesco6_length","dis_min","estadocivil4_length",
                      "tipovivi__length","energcocinar2","inst_age_length","hhsize",
                      "pisomadera","age_length","mobilephone_max","mobilephone_length",
                      "is_jefe_mujer","parentesco2_length","estadocivil7_length",
                      "parentesco5_length","hacdor","tipovivi3","female_length",
                      "parentesco6_max","female_min","estadocivil7_mean","mobilephone_min",
                      "v18q_mean","parentesco4_length","parentesco10_length","female_max",
                      "estadocivil6_rango","parentesco11_length","tipovivi4","elimbasu1",
                      "energcocinar3","area1","refrig","parentesco6_rango",
                      "estadocivil6_length","tipovivi_","parentesco1_length","lugar5",
                      "tech_length","dis_max","parentesco2_max","v18q_length",
                      "parentesco11_mean","estadocivil1_length","estadocivil7_max",
                      "parentesco2_mean","hacapo","estadocivil4_mean","parentesco5_rango",
                      "escolari_age_length","estadocivil6_max","rez_esc_length",
                      "abastaguano","estadocivil5_max","parentesco12_rango",
                      "estadocivil3_length","pisonotiene","pareddes","estadocivil2_rango",
                      "parentesco9_mean","tipovivi2","estadocivil5_rango",
                      "parentesco9_rango","paredfibras","lugar6","sanitario2","sanitario5",
                      "parentesco4_max","estadocivil4_rango","tipovivi1","lugar4",
                      "estadocivil3_rango","elimbasu3","parentesco1_rango",
                      "parentesco10_max","techoentrepiso","estadocivil2_min",
                      "parentesco12_mean","parentesco12_max","hhsize_diff",
                      "abastaguafuera","paredzinc","v14a","estadocivil2_max",
                      "parentesco5_max","paredother","pisoother","pisonatur","techootro",
                      "sanitario6","elimbasu4","elimbasu5","elimbasu6","rez_esc_min",
                      "estadocivil1_min","parentesco2_min","parentesco3_min",
                      "parentesco4_min","parentesco5_min","parentesco6_min",
                      "parentesco7_min","parentesco8_min","parentesco9_min",
                      "parentesco10_min","parentesco11_min","parentesco12_min",
                      "parentesco1_max")

variables_para_quitar<- c("conjunto", "elec_missing", "idhogar", "Id")
variables_ids <- c("idhogar", "Id")

train_final <- select(train_transformed, -variables_para_quitar)
test_ids <- select(test_transformed, variables_ids)
test_final <- select(test_transformed, -variables_para_quitar)


cv_split <- vfold_cv(train_final, v = 10)
```

```r
cv_data <- cv_split %>%
  mutate(
    train = map(splits, ~training(.x)),
    validate = map(splits,~testing(.x))
  )
cv_data
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 4
##    splits       id     train                 validate
##  * <list>       <chr>  <list>                <list>
##  1 <S3: rsplit> Fold01 <tibble [2,675 x 254]> <tibble [298 x 254]>
##  2 <S3: rsplit> Fold02 <tibble [2,675 x 254]> <tibble [298 x 254]>
##  3 <S3: rsplit> Fold03 <tibble [2,675 x 254]> <tibble [298 x 254]>
##  4 <S3: rsplit> Fold04 <tibble [2,676 x 254]> <tibble [297 x 254]>
##  5 <S3: rsplit> Fold05 <tibble [2,676 x 254]> <tibble [297 x 254]>
##  6 <S3: rsplit> Fold06 <tibble [2,676 x 254]> <tibble [297 x 254]>
##  7 <S3: rsplit> Fold07 <tibble [2,676 x 254]> <tibble [297 x 254]>
##  8 <S3: rsplit> Fold08 <tibble [2,676 x 254]> <tibble [297 x 254]>
##  9 <S3: rsplit> Fold09 <tibble [2,676 x 254]> <tibble [297 x 254]>
## 10 <S3: rsplit> Fold10 <tibble [2,676 x 254]> <tibble [297 x 254]>
```

Modelación

A continuación se presenta la especificación de los modelos, su ajuste y como coparan sus diferentes estimaciones.

Modelo logístico multinomial

La siguientes especificación es una de las 3 que se hicieron para un modelo de regresión logística multinomial. Este caso es el de las 222 variables.

```r
# Entrenar el modelo para cada uno de los 5 conjuntos de validación cruzada
cv_models_logistic_reg <- cv_data %>%
  mutate(model = map(train, ~ multinom(formula = factor(Target) ~ escolari_mean+inst_level_mean+inst_ag

                                       , data = .x)))
```

```
## # weights:  840 (627 variable)
## initial  value 3708.337416
## iter  10 value 2748.905728
## iter  20 value 2478.431848
## iter  30 value 2382.822990
## iter  40 value 2316.539881
## iter  50 value 2195.357621
## iter  60 value 2103.235035
## iter  70 value 2008.417277
## iter  80 value 1971.971703
## iter  90 value 1962.004798
## iter 100 value 1956.374981
```

```
## final  value 1956.374981
## stopped after 100 iterations
## # weights:  840 (627 variable)
## initial  value 3708.337416
## iter  10 value 2752.863011
## iter  20 value 2462.918063
## iter  30 value 2354.688586
## iter  40 value 2243.363235
## iter  50 value 2120.300676
## iter  60 value 2026.749793
## iter  70 value 1966.730983
## iter  80 value 1942.127948
## iter  90 value 1932.770500
## iter 100 value 1926.972380
## final  value 1926.972380
## stopped after 100 iterations
## # weights:  840 (627 variable)
## initial  value 3708.337416
## iter  10 value 2562.791000
## iter  20 value 2308.933369
## iter  30 value 2216.194115
## iter  40 value 2156.257417
## iter  50 value 2079.776121
## iter  60 value 2019.860146
## iter  70 value 1967.952673
## iter  80 value 1945.099932
## iter  90 value 1938.185920
## iter 100 value 1931.302783
## final  value 1931.302783
## stopped after 100 iterations
## # weights:  840 (627 variable)
## initial  value 3709.723710
## iter  10 value 2703.611321
## iter  20 value 2371.910826
## iter  30 value 2254.065953
## iter  40 value 2175.509102
## iter  50 value 2088.678330
## iter  60 value 2018.228057
## iter  70 value 1961.602856
## iter  80 value 1938.920056
## iter  90 value 1931.150348
## iter 100 value 1924.823839
## final  value 1924.823839
## stopped after 100 iterations
## # weights:  840 (627 variable)
## initial  value 3709.723710
## iter  10 value 2661.796771
## iter  20 value 2392.592929
## iter  30 value 2259.595314
## iter  40 value 2183.468329
## iter  50 value 2080.516271
## iter  60 value 1987.186498
## iter  70 value 1925.529360
## iter  80 value 1898.716379
```

```
## iter  90 value 1891.859782
## iter 100 value 1885.130400
## final  value 1885.130400
## stopped after 100 iterations
## # weights:  840 (627 variable)
## initial  value 3709.723710
## iter  10 value 2922.885558
## iter  20 value 2563.264229
## iter  30 value 2444.157592
## iter  40 value 2281.618354
## iter  50 value 2156.020202
## iter  60 value 2059.309499
## iter  70 value 1972.502480
## iter  80 value 1943.479854
## iter  90 value 1933.202458
## iter 100 value 1925.510127
## final  value 1925.510127
## stopped after 100 iterations
## # weights:  840 (627 variable)
## initial  value 3709.723710
## iter  10 value 2691.246333
## iter  20 value 2486.045617
## iter  30 value 2386.488232
## iter  40 value 2299.772118
## iter  50 value 2174.971044
## iter  60 value 2048.738045
## iter  70 value 1953.682621
## iter  80 value 1914.550940
## iter  90 value 1906.021279
## iter 100 value 1899.015944
## final  value 1899.015944
## stopped after 100 iterations
## # weights:  840 (627 variable)
## initial  value 3709.723710
## iter  10 value 2678.088207
## iter  20 value 2435.380967
## iter  30 value 2338.783033
## iter  40 value 2269.255140
## iter  50 value 2150.363263
## iter  60 value 2034.850719
## iter  70 value 1951.174611
## iter  80 value 1913.946026
## iter  90 value 1904.168084
## iter 100 value 1897.659242
## final  value 1897.659242
## stopped after 100 iterations
## # weights:  840 (627 variable)
## initial  value 3709.723710
## iter  10 value 2672.104660
## iter  20 value 2451.293903
## iter  30 value 2354.470182
## iter  40 value 2275.162646
## iter  50 value 2177.328469
## iter  60 value 2077.530265
```

```
## iter  70 value 1998.375651
## iter  80 value 1956.146543
## iter  90 value 1943.357208
## iter 100 value 1936.383451
## final  value 1936.383451
## stopped after 100 iterations
## # weights:  840 (627 variable)
## initial  value 3709.723710
## iter  10 value 2848.604801
## iter  20 value 2437.941770
## iter  30 value 2289.415065
## iter  40 value 2173.863482
## iter  50 value 2078.302881
## iter  60 value 1990.643413
## iter  70 value 1934.614072
## iter  80 value 1917.371554
## iter  90 value 1907.385120
## iter 100 value 1902.195176
## final  value 1902.195176
## stopped after 100 iterations
```

```
cv_models_logistic_reg
```

```
## #  10-fold cross-validation
## # A tibble: 10 x 5
##    splits       id     train               validate           model
##  * <list>       <chr>  <list>              <list>             <list>
##  1 <S3: rsplit> Fold01 <tibble [2,675 x 2~ <tibble [298 x 25~ <S3: multin~
##  2 <S3: rsplit> Fold02 <tibble [2,675 x 2~ <tibble [298 x 25~ <S3: multin~
##  3 <S3: rsplit> Fold03 <tibble [2,675 x 2~ <tibble [298 x 25~ <S3: multin~
##  4 <S3: rsplit> Fold04 <tibble [2,676 x 2~ <tibble [297 x 25~ <S3: multin~
##  5 <S3: rsplit> Fold05 <tibble [2,676 x 2~ <tibble [297 x 25~ <S3: multin~
##  6 <S3: rsplit> Fold06 <tibble [2,676 x 2~ <tibble [297 x 25~ <S3: multin~
##  7 <S3: rsplit> Fold07 <tibble [2,676 x 2~ <tibble [297 x 25~ <S3: multin~
##  8 <S3: rsplit> Fold08 <tibble [2,676 x 2~ <tibble [297 x 25~ <S3: multin~
##  9 <S3: rsplit> Fold09 <tibble [2,676 x 2~ <tibble [297 x 25~ <S3: multin~
## 10 <S3: rsplit> Fold10 <tibble [2,676 x 2~ <tibble [297 x 25~ <S3: multin~
```

Ajuste del modelo y predicción.

```
# Ajuste el modelo
cv_prep_logistic_reg <- cv_models_logistic_reg %>%
  mutate(
    # Extraer Target2 de los 10 conjuntos de validación
    validate_actual = map(validate, ~.x$Target),
    # Predict life expectancy for each validate set using its corresponding model
    validate_predicted = map2(.x = model, .y = validate, ~predict(.x, .y))
  )
```

F-Score-Macro promedio

El Macro F-Score de este modelo es el siguiente.

```r
# Extraer las métricas de evaluación y calcular el error de validación cruzada promedio para cada uno d

cv_perf_metrics <- cv_prep_logistic_reg %>%
  mutate(f1_score_macro = map2_dbl(validate_actual, validate_predicted,
                                   ~ F1_Score_macro(as.numeric(.x), as.numeric(.y))))

mean(cv_perf_metrics$f1_score_macro)
```

```
## [1] 0.3815761
```

En la siguiente tabla se comparan los tres modelos logísticos multinomiales que se estimaron..

Table 14: Resultados de los 5 mejores modelos con Regresión Logística

n_variables	F1_Score_Macro_CV	F1_Score_Macro_Prueba
10	0.337	0.298
40	0.375	0.310
222	0.389	0.385

En general se puede ver que la regresión logística multinomial logró un F1 Macro Score bastante bueno en la medida que se iban incluyendo más variables importantes. La mejor estimación que se obtuvo fue la de 0.385

Modelo Random Forest

Los modelos de bosques aleatorios tienen dos hiper parámetros el número de árboles y el número de variables que ven de manera aleatoria. Tomando en cuenta lo anterior se entrenará bosques aleatorios con:

- mtry : 2, 4, 8, 16, 32, 64 y 128
- ntree : 64, 100, 128, 500, 1000

```r
# construir un grid search de parámetros
cv_tune <- cv_data %>%
  crossing(mtry = c(2,4,8,16,32,64,128))
```

A continuación se muestra la especificación para uno de los modelos Random Forest que se estimaron con las 222 características más importantes.

```r
cv_models_rf_64 <- cv_tune %>%
   mutate(model = map2(train,mtry, ~ randomForest(factor(Target) ~ escolari_mean+inst_level_mean+inst_ag
                                                  data = .x, mtry = .y, ntree = 64)))
```

A continuación se presentan los resultados del F-Score Macro para todos los modelos ajustados:

```r
# Modelo cv_prep_rf_32
cv_prep_rf_32 <- cv_models_rf_32 %>%
  mutate(
    # Extraer Target2 de los 5 conjuntos de validación
```

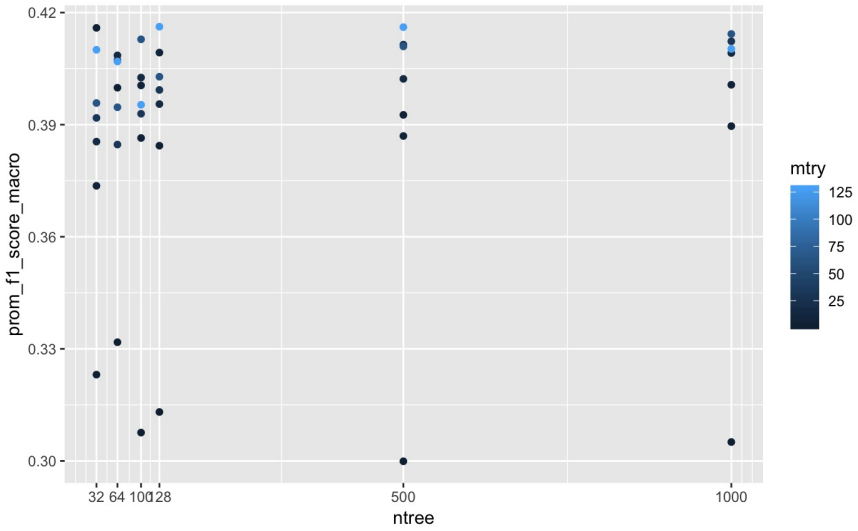
```
    validate_actual = map(validate, ~.x$Target),
    # Predict life expectancy for each validate set using its corresponding model
    validate_predicted = map2(.x = model, .y = validate, ~predict(.x, .y))
  )

# Modelo cv_prep_rf_64
cv_prep_rf_64 <- cv_models_rf_64 %>%
  mutate(
    # Extraer Target2 de los 5 conjuntos de validación
    validate_actual = map(validate, ~.x$Target),
    # Predict life expectancy for each validate set using its corresponding model
    validate_predicted = map2(.x = model, .y = validate, ~predict(.x, .y))
  )

# Modelo cv_models_rf_100
cv_prep_rf_100 <- cv_models_rf_100 %>%
  mutate(
    # Extraer Target2 de los 5 conjuntos de validación
    validate_actual = map(validate, ~.x$Target),
    # Predict life expectancy for each validate set using its corresponding model
    validate_predicted = map2(.x = model, .y = validate, ~predict(.x, .y))
  )

# Modelo cv_models_rf_128
cv_prep_rf_128 <- cv_models_rf_128 %>%
  mutate(
    # Extraer Target2 de los 5 conjuntos de validación
    validate_actual = map(validate, ~.x$Target),
    # Predict life expectancy for each validate set using its corresponding model
    validate_predicted = map2(.x = model, .y = validate, ~predict(.x, .y))
  )

# Modelo cv_models_rf_500
cv_prep_rf_500 <- cv_models_rf_500 %>%
  mutate(
    # Extraer Target2 de los 5 conjuntos de validación
    validate_actual = map(validate, ~.x$Target),
    # Predict life expectancy for each validate set using its corresponding model
    validate_predicted = map2(.x = model, .y = validate, ~predict(.x, .y))
  )

# Modelo cv_models_rf_1000
cv_prep_rf_1000 <- cv_models_rf_1000 %>%
  mutate(
    # Extraer Target2 de los 5 conjuntos de validación
    validate_actual = map(validate, ~.x$Target),
    # Predict life expectancy for each validate set using its corresponding model
    validate_predicted = map2(.x = model, .y = validate, ~predict(.x, .y))
  )
```

Los resultados son los siguientes

```r
# Extraer las métricas de evaluación y calcular el error de validación cruzada promedio para cada uno d

# Modelo cv_prep_rf_32
cv_perf_metrics_rf_32 <- cv_prep_rf_32 %>%
  mutate(f1_score_macro = map2_dbl(validate_actual, validate_predicted,
                                   ~ F1_Score_macro(as.numeric(.x), as.numeric(.y))))
f1_score_macro_rf_32 <- cv_perf_metrics_rf_32 %>%
  select(id, mtry, f1_score_macro) %>%
  group_by(mtry) %>%
  summarise(prom_f1_score_macro = mean(f1_score_macro),
            min_f1_score_macro = min(f1_score_macro),
            max_f1_score_macro = max(f1_score_macro)) %>%
  mutate(ntree = 32)

# Modelo cv_prep_rf_64
cv_perf_metrics_rf_64 <- cv_prep_rf_64 %>%
  mutate(f1_score_macro = map2_dbl(validate_actual, validate_predicted,
                                   ~ F1_Score_macro(as.numeric(.x), as.numeric(.y))))
f1_score_macro_rf_64 <- cv_perf_metrics_rf_64 %>%
  select(id, mtry, f1_score_macro) %>%
  group_by(mtry) %>%
  summarise(prom_f1_score_macro = mean(f1_score_macro),
            min_f1_score_macro = min(f1_score_macro),
            max_f1_score_macro = max(f1_score_macro)) %>%
  mutate(ntree = 64)

# Modelo cv_prep_rf_100
cv_perf_metrics_rf_100 <- cv_prep_rf_100 %>%
  mutate(f1_score_macro = map2_dbl(validate_actual, validate_predicted,
                                   ~ F1_Score_macro(as.numeric(.x), as.numeric(.y))))
f1_score_macro_rf_100 <- cv_perf_metrics_rf_100 %>%
  select(id, mtry, f1_score_macro) %>%
  group_by(mtry) %>%
  summarise(prom_f1_score_macro = mean(f1_score_macro),
            min_f1_score_macro = min(f1_score_macro),
            max_f1_score_macro = max(f1_score_macro)) %>%
  mutate(ntree = 100)

# Modelo cv_prep_rf_128
cv_perf_metrics_rf_128 <- cv_prep_rf_128 %>%
  mutate(f1_score_macro = map2_dbl(validate_actual, validate_predicted,
                                   ~ F1_Score_macro(as.numeric(.x), as.numeric(.y))))
f1_score_macro_rf_128 <- cv_perf_metrics_rf_128 %>%
  select(id, mtry, f1_score_macro) %>%
  group_by(mtry) %>%
  summarise(prom_f1_score_macro = mean(f1_score_macro),
            min_f1_score_macro = min(f1_score_macro),
            max_f1_score_macro = max(f1_score_macro)) %>%
  mutate(ntree = 128)

# Modelo cv_prep_rf_500
cv_perf_metrics_rf_500 <- cv_prep_rf_500 %>%
  mutate(f1_score_macro = map2_dbl(validate_actual, validate_predicted,
```

```
                                    ~ F1_Score_macro(as.numeric(.x), as.numeric(.y))))
f1_score_macro_rf_500 <- cv_perf_metrics_rf_500 %>%
  select(id, mtry, f1_score_macro) %>%
  group_by(mtry) %>%
  summarise(prom_f1_score_macro = mean(f1_score_macro),
            min_f1_score_macro = min(f1_score_macro),
            max_f1_score_macro = max(f1_score_macro)) %>%
  mutate(ntree = 500)

# Modelo cv_prep_rf_1000
cv_perf_metrics_rf_1000 <- cv_prep_rf_1000 %>%
  mutate(f1_score_macro = map2_dbl(validate_actual, validate_predicted,
                                    ~ F1_Score_macro(as.numeric(.x), as.numeric(.y))))
f1_score_macro_rf_1000 <- cv_perf_metrics_rf_1000 %>%
  select(id, mtry, f1_score_macro) %>%
  group_by(mtry) %>%
  summarise(prom_f1_score_macro = mean(f1_score_macro),
            min_f1_score_macro = min(f1_score_macro),
            max_f1_score_macro = max(f1_score_macro)) %>%
  mutate(ntree = 1000)
```

Se estimaron 42 modelos con diferentes especificaciones de números de árboles y mtry. En la siguiente gráfica se muestra su desempeño.



Estos fueron los 5 mejores modelos con random forest que encontramos en el proceso de validación cruzada utilizando la métrica de evaluación F1 Score Macro. Asimismo, se presenta el F1 Score Macro del conjunto de prueba obtenido en Kaggle.

Table 15: Resultados de los 5 mejores modelos con random forest

ntree	mtry	cv	kaggle
128	128	0.416	0.376
500	128	0.416	0.373
32	4	0.416	0.347
1000	64	0.414	0.374
100	64	0.413	0.376

De los resultados anteriores se puede concluir que el model random forest para amplio espectro de hiper parámetros no supera la barrera del 0.40 en el F Socre Macro de la competencia. Asimismo, se puede apreciar que el score de validación cruzada está sobrestimando al score de prueba. No se ve un patrón claro en la selección de hiper parámetros. Pero se puede confirmar que con 500 o menos arboles es posible obtener un score alto.

XBoost

Considerando los resultados del modelo logístico y el random forest, en esta sección se implementa un modelo Xboost el cual tiene la ventaja de lidiar con el problema de clases desbalanceadas como es este problema de la pobreza a nivel hogar.

A manera de ejemplo se presenta una de las especificaciones de los modelos XGBoost que se ajustaron en este proyecto.

A continuación se preparan los datos en el formato que requiere XGboost:

```
# Preparamos las base de datos en el formato que requiere el paquete
# Convertir a valor numérico la variable dependiente y aplicar el one hot encoding

label_train <- (unlist(select(train_final, Target)) %>% as.numeric()) - 1
X_train <- select(train_final,-Target)

label_test <- (unlist(select(test_final, Target)) %>% as.numeric()) - 1
X_test <- select(test_final,-Target)

# Pasamos la matriz de covariables de data frame a matrix
# Rows: 2,973
# Cols: 253
new_X_train <- data.matrix(X_train)
new_X_test <- data.matrix(X_test)

# Preparamos las matrices de covariables al formato de xgboost
xgb_train <- xgb.DMatrix(data = new_X_train, label = label_train)
xgb_test <- xgb.DMatrix(data = new_X_test, label = label_test)

# Watch list
watchlist <- list(train = xgb_train)
```

Se fijan los parámetros del XGboost para un problema de clasificación de múltiples categorías:

- Max depth
- Eta
- num_class
- subsample
- lambda
- gamma
- colsample_bytree
- objective: multi:softprob
- eval_metric: mlogloss

Destaca la métrica de evaluación que se utilizó para estos modelos, la Multiclass Log Loss.

```
# Set parameters
params <- list(booster = "gbtree",
               max_depth = 3,
               eta = 0.03,
               nthread = 1,
               num_class = 4,
               lambda = 0.001,
               objective = "multi:softprob",
               eval_metric = "mlogloss")
```

Considerando los resultados que obtuvimos con la validación cruzada utilizando el modelo logístico y el modelo de arboles aleatorios, en este caso utilizamos una validación cruzada estratificada con 10 splits. El número de iteraciones en este boost fueron 200, con un early stopping de 20.

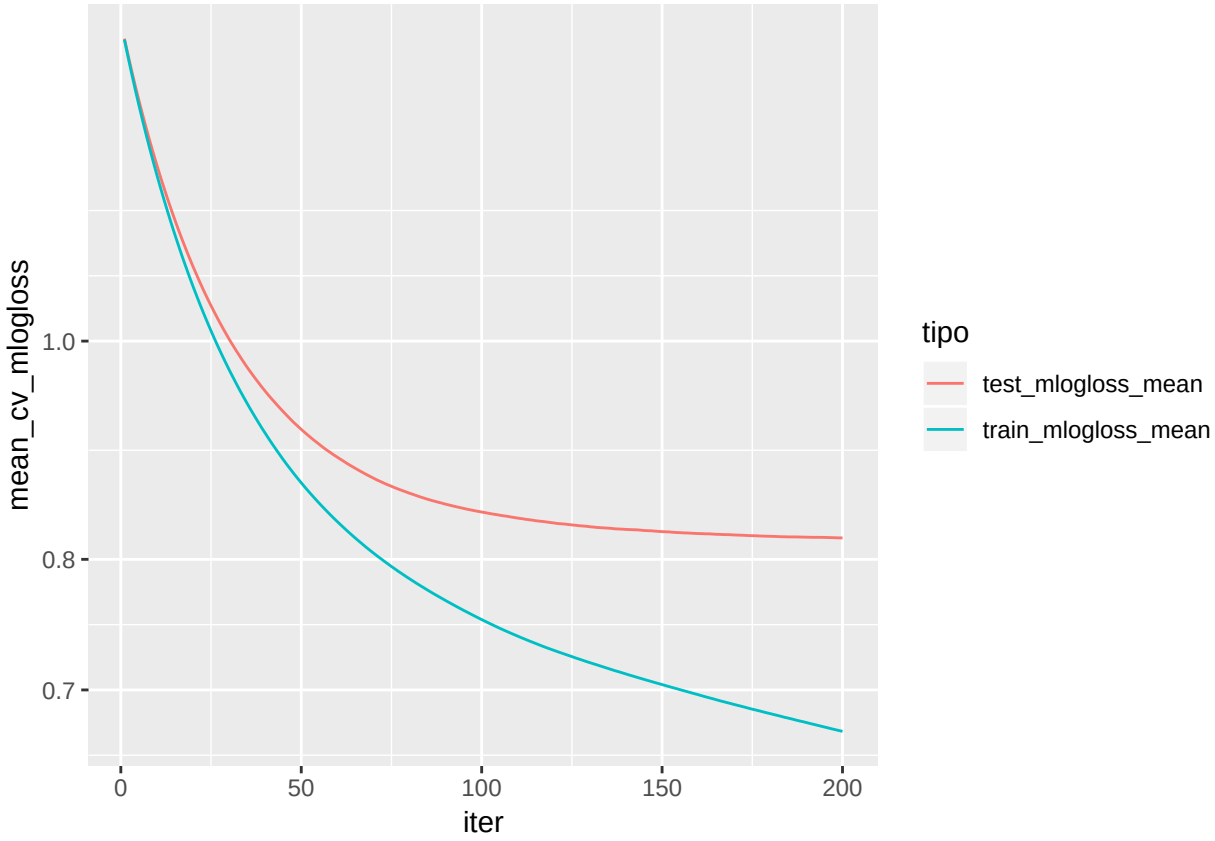
```
# Calculate # of folds for cross-validation

iteraciones <- 200

xgbcv <- xgb.cv(params = params,
                data = xgb_train,
                nrounds = iteraciones,
                nfold = 10,
                showsd = TRUE,
                stratified = TRUE,
                print.every.n = 10,
                early_stop_round = iteraciones * .1,
                maximize = FALSE,
                prediction = TRUE,
                gamma = 0.5)
```

```
## Warning: 'print.every.n' is deprecated.
## Use 'print_every_n' instead.
## See help("Deprecated") and help("xgboost-deprecated").

## [1]  train-mlogloss:1.361014+0.000324    test-mlogloss:1.362352+0.001821
## [11] train-mlogloss:1.169782+0.002637    test-mlogloss:1.182700+0.016261
## [21] train-mlogloss:1.047479+0.003913    test-mlogloss:1.070119+0.025987
## [31] train-mlogloss:0.964241+0.004653    test-mlogloss:0.996037+0.033138
## [41] train-mlogloss:0.905023+0.005020    test-mlogloss:0.945864+0.038203
## [51] train-mlogloss:0.861314+0.005457    test-mlogloss:0.910720+0.041645
## [61] train-mlogloss:0.828544+0.005583    test-mlogloss:0.885968+0.044352
## [71] train-mlogloss:0.802877+0.005682    test-mlogloss:0.867625+0.046804
## [81] train-mlogloss:0.782389+0.005696    test-mlogloss:0.854883+0.048886
## [91] train-mlogloss:0.765451+0.005791    test-mlogloss:0.845661+0.051005
## [101]    train-mlogloss:0.750843+0.005855    test-mlogloss:0.839122+0.052433
## [111]    train-mlogloss:0.738394+0.006029    test-mlogloss:0.834021+0.053183
## [121]    train-mlogloss:0.727950+0.006040    test-mlogloss:0.829882+0.054004
## [131]    train-mlogloss:0.718936+0.006018    test-mlogloss:0.826829+0.054695
## [141]    train-mlogloss:0.710680+0.006074    test-mlogloss:0.824714+0.055526
## [151]    train-mlogloss:0.703080+0.006137    test-mlogloss:0.822855+0.055941
## [161]    train-mlogloss:0.695821+0.006255    test-mlogloss:0.821304+0.056191
## [171]    train-mlogloss:0.689010+0.006307    test-mlogloss:0.820063+0.056279
## [181]    train-mlogloss:0.682562+0.006332    test-mlogloss:0.818999+0.056226
```

```
## [191]    train-mlogloss:0.676373+0.006363    test-mlogloss:0.818258+0.056405
## [200]    train-mlogloss:0.670953+0.006453    test-mlogloss:0.817722+0.056530
```

Se monitoreó el ajuste de los XGboost utilizando las curvas de desempeño del error. Se procuró cerrar la brecha entre el entrenamiento y la validación

```
xgbcv$evaluation_log %>%
  select(iter, train_mlogloss_mean, test_mlogloss_mean) %>%
  gather(tipo,mean_cv_mlogloss, 2:3) %>%
  ggplot(aes(x = iter, y = mean_cv_mlogloss, colour = tipo, group = tipo)) +
    geom_line() +
    scale_y_log10()
```



Se ajustaron 20 modelos utilizando este algoritmo y modificando los siguientes parámetros para mejorar el puntaje:

- eta: La tasa de aprendizaje
- Lambda: Regularización sobre los coeficientes
- Subsamples: Submuestras de la información para evitar sobreajuste
- Max_Depth: Profundidad del arbol
- Feature subsample: Proporción de las características utilizadas para ajustar los arboles

La mayoría de estos parámetros contribuyeron a combatir el sobreajuste del modelo y el costo comunicacional.

En la siguiente tabla se muestran los resultados ordenados de mayor a menor por el F Score Macro obtenido en el conjunto de prueba.

```
resultados_boost <- read_csv("resultados_boost.csv")

resultados_boost <- resultados_boost %>% arrange(desc(Prueba))

knitr::kable(resultados_boost, caption = "Resultados del XGBoost")
```

Table 16: Resultados del XGBoost

Entrena	Prueba	iteraciones	Eta	Lambda	Subsamples	Max_depth	Gamma	feature_subsample
0.9499116	0.400	2000	0.030	0.20	1.00	3	0.0	1.00
0.9900000	0.398	3000	0.030	0.20	0.20	3	0.0	1.00
0.8900000	0.398	3000	0.030	0.20	0.20	4	0.5	0.25
0.9500000	0.396	2000	0.030	0.20	0.20	3	0.0	1.00
1.0000000	0.396	3000	0.030	0.20	0.20	4	0.0	1.00
0.7900000	0.396	2500	0.030	0.20	0.20	4	0.5	0.05
0.7600000	0.396	2500	0.030	0.20	0.20	4	0.5	0.04
0.9900000	0.395	3000	0.030	0.20	0.20	4	0.5	1.00
0.9790000	0.394	2000	0.030	0.19	0.54	3	0.0	1.00
0.9500000	0.393	2000	0.030	0.19	1.00	3	0.0	1.00
0.9490000	0.392	2000	0.030	0.21	1.00	3	0.0	1.00
0.9383542	0.391	2000	0.030	0.00	1.00	3	0.0	1.00
0.7883000	0.389	1000	0.030	0.00	1.00	3	0.0	1.00
0.7800000	0.389	500	0.030	0.20	0.20	4	0.5	0.05
0.6699000	0.388	1000	0.030	0.20	0.20	4	0.5	0.05
0.9497000	0.386	2000	0.030	0.15	1.00	3	0.0	1.00
0.5700000	0.371	500	0.030	0.20	0.20	4	0.5	0.05
0.4849563	0.344	1000	0.003	0.00	1.00	3	0.0	1.00
0.4466259	0.337	200	0.003	0.00	1.00	3	0.0	1.00
0.4405086	0.331	100	0.003	0.00	1.00	3	0.0	1.00

Se puede observar que se exploró un amplio espacio de hiper parámetros para superar el F-Score Macro obtenido en Kaggle. El ganador de la competencia obtuvo un score de 0.448. En ese sentido logramos acercarnos lo más que pudimos al obtener un 0.4.

Conclusiones

El objetivo de este proyecto era producir un algoritmo que ayudara a predecir el nivel de pobreza en los hogares de Costa Rica. Se probaron diferentes iteraciones de tres algoritmos Regresión Logística, Random Forest y XGboost. La métrica que se utilizó en este proyecto fue la media armónica de la precisión y el recall para todas las categorías, que en problemas de clasificación corresponde al promedio de los F1 Score de cada clase. Utilizando este criterio, el algoritmo que mejor de desempeñó de manera consistente fue el XGboost que alcanzó un F Score Macro de score de 0.4. Este indicador lo que nos está diciendo es que tan bueno fue el algoritmo para acertar al nivel correcto de pobreza en los hogares de las predicciones positivas que se hicieron, y que tan bueno fue el algoritmo para encontrar todos los casos correctos.

La clave en este problema para obtener un mejor score es la ingeniería de características. Dado que este conjunto de datos se compone de variables a nivel hogar y a nivel individuo, hay que saber cómo agregar de manera correcta esta información. En este proyecto se agregó la información utilizando promedios, mínimos, máximos y rango. Asimismo, se crearon nuevas variables para capturar la interacción de ellas y reducir también el espacio de características. Las que más éxito tuvieron fueron aquellas relacionadas con la educación y el nivel de escolaridad, en un segundo plano las variables asociadas a la calidad de la vivienda y

dispositivos electrónicos. Intuitivamente esto tiene mucho sentido en la vida real y es lo que nuestro algoritmo tomó en cuenta.

Otro de los grandes retos de este problema fue la limpieza y la agregación de los datos, ya que si no se hace de manera correcta, el modelo no sirve para nada. En esta etapa uno de los principales problemas fueron los datos nulos, los cuáles eran muchos. En nuestro caso los datos fueron imputados utilizando la mediana.