Prueba técnica: presentación de resultados

Jorge Humberto Moncayo Bravo *(jmoncayo@unal.edu.co)*

18 de marzo de 2023

1. Introducción

En el presente informe se ponen de manifiesto los resultados de cada uno de los puntos asignados en la prueba técnica. Para los puntos **1** y **2** se presenta tanto la metodología como el componente de resultados (desde el punto de vista técnico y de negocio); así mismo, para el punto **3**, la explicación se realiza sin andamiajes a través de un breve párrafo con sus respectivas referencias bibliográficas. El análisis técnico se llevó a cabo en `RStudio` en su versión `2022.12.0`, empleando `R` en su versión `4.1.2` sobre una máquina `GNU-Linux` con distribución `Ubuntu 22.04.2`.

2. Metodología: primer punto

En base al requerimiento del cliente interno, el propósito es crear un modelo de predictivo de clasificación que permita seleccionar los clientes de acuerdo con su probabilidad de incumplimiento, por tal razón, se decide realizar un análisis supervisado empleando clasificación binaria sobre el atributo `Mora30` porque de la proposición inicial se tiene que el producto se espera ofrecer a "un segmento de la población de riesgo bajo", de ahí que este atributo resulte un clasificador más sensato, en comparación con `Mora60`, para determinar los clientes que pueden caer en mora en el **corto plazo**. Elegido el enfoque de resolución del problema de negocio, el flujo de trabajo para dar respuesta a este punto tiene los siguientes pasos esenciales basados en el flujo de trabajo sugerido por el *framework* `tidymodels`, el cual es una colección de parquetes para modelado y machine learning enfocado en seguir los principios del `tidyverse` (reutilización, composición de funciones simples a través de *pipelines*, programación funcional y diseño para humanos):

- Extracción y limpieza de datos
- Creación de particiones de entrenamiento, prueba y validación
- Definición de modelos
- Definición de receta de preprocesamiento
- Definición de flujo de trabajo y búsqueda de hiperparámetros
- Evaluación de métricas y entrenamiento de modelo

La descripción detallada de los pasos anteriormente descritos se describe en la siguiente sección.

3. Resultados: primer punto

3.1. Extracción y limpieza de datos

En este paso, se extrae la información proveniente de un archivo en formato .xlsx para almacenarlo en un objeto de clase `tibble` (una versión moderna de la clase `data.frame` que se encuentra en la base de R).

```
library(ggplot2)
library(tidymodels)
library(tidyverse)
library(readxl)
library(tibble)
library(parsnip)
```

```
library(vip)

df <- read_excel(ruta,
                 col_types = c("skip", "skip", "numeric",
                               "numeric", "numeric", "numeric",
                               "numeric", "numeric", "numeric",
                               "numeric", "text", "text", "numeric",
                               "text", "text", "numeric", "text",
                               "text", "numeric", "numeric", "numeric",
                               "numeric")) %>%
  as_tibble()

df
```

```
## # A tibble: 4,315 x 20
##    Mora30 Mora60 MoraM~1 Exper~2 Perso~3 Gasto~4 Gasto~5 Tiemp~6 OCUPA~7 TIPCO~8
##     <dbl>  <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <chr>   <chr>
##  1      0      0       0       0       0   7838.       0      10 JUBILA~ TÉRMIN~
##  2      1      0      59       0       1   9194.       0       1 EMPLEA~ TÉRMIN~
##  3      0      0      30       1       1   8450        0       1 EMPLEA~ TÉRMIN~
##  4      1      0      47       0       0   7472        0       1 EMPLEA~ TÉRMIN~
##  5      0      0       0       0       0   6160        0       2 EMPLEA~ TÉRMIN~
##  6      0      0      30       0       1   6160        0       2 EMPLEA~ TÉRMIN~
##  7      0      0      29       1       0   6160        0      15 JUBILA~ OTROS
##  8      1      1      91       0       0   9000        0       1 EMPLEA~ TÉRMIN~
##  9      0      0       0       0       1   6160        0       3 JUBILA~ OTROS
## 10      0      0       0       1       0   6900        0       2 EMPLEA~ TÉRMIN~
## # ... with 4,305 more rows, 10 more variables: Edad <dbl>, Estado_Civil <chr>,
## #   Genero <chr>, Ingresos <dbl>, Nivel_Academico <chr>, Tipo_Vivienda <chr>,
## #   TiempoCliente <dbl>, Tiempo_SistemaFro <dbl>, PORCEND <dbl>,
## #   Obligaciones_SistemaFro <dbl>, and abbreviated variable names
## #   1: MoraMax_UltimoSemestre, 2: Experiencia, 3: PersonasCargo,
## #   4: GastosFamiliares, 5: GastoArriendo, 6: TiempoActividad, 7: OCUPACION,
## #   8: TIPCONTRATO
```

A continuación, se modifican los nombres de las columnas para hacerlos más accesibles:

```
colnames(df) <- c("mora30", "mora60", "moramax_ultimosemestre",
                  "experiencia", "personas_cargo", "gastos_familiares",
                  "gastos_arriendo", "tiempo_actividad", "ocupacion",
                  "tipo_contrato", "edad", "estado_civil",
                  "genero", "ingresos", "nivel_academico",
                  "tipo_vivienda", "tiempo_cliente","tiempo_sistema",
                  "porcend", "obligaciones_sistema")
```

Se modifican los tipos de datos de algunas columnas y se efectúa un breve análisis descriptivo de los datos:

```
df <- df %>%
  mutate(mora30 = as.factor(mora30),
         mora60 = as.factor(mora60),
         moramax_ultimosemestre = as.integer(moramax_ultimosemestre),
         experiencia = as.integer(experiencia),
         personas_cargo = as.integer(personas_cargo),
         gastos_familiares = as.integer(gastos_familiares),
         gastos_arriendo = as.integer(gastos_arriendo),
```

```
        edad = as.integer(edad),
        ingresos = as.integer(ingresos),
        tiempo_cliente = as.integer(tiempo_cliente),
        tiempo_sistema = as.integer(tiempo_sistema),
        obligaciones_sistema = as.integer(obligaciones_sistema))
summary(df)
```

```
##  mora30   mora60   moramax_ultimosemestre  experiencia     personas_cargo
##  0:3264   0:3811   Min.   :  0.0          Min.   :0.0000   Min.   :0.000
##  1:1051   1: 504   1st Qu.:  0.0          1st Qu.:0.0000   1st Qu.:0.000
##                    Median : 17.0          Median :0.0000   Median :0.000
##                    Mean   : 26.9          Mean   :0.3395   Mean   :0.372
##                    3rd Qu.: 30.0          3rd Qu.:1.0000   3rd Qu.:0.000
##                    Max.   :364.0          Max.   :1.0000   Max.   :6.000
##  gastos_familiares gastos_arriendo  tiempo_actividad    ocupacion
##  Min.   : 6000     Min.   :     0   Min.   :     0.0   Length:4315
##  1st Qu.: 6200     1st Qu.:     0   1st Qu.:     1.0   Class :character
##  Median : 7308     Median :     0   Median :     1.0   Mode  :character
##  Mean   : 7500     Mean   :  7101   Mean   :   203.7
##  3rd Qu.: 8490     3rd Qu.:     0   3rd Qu.:     3.0
##  Max.   :13000     Max.   :300000   Max.   :866880.0
##  tipo_contrato           edad       estado_civil          genero
##  Length:4315        Min.   :18.00   Length:4315        Length:4315
##  Class :character   1st Qu.:27.00   Class :character   Class :character
##  Mode  :character   Median :30.00   Mode  :character   Mode  :character
##                     Mean   :33.75
##                     3rd Qu.:38.00
##                     Max.   :69.00
##     ingresos     nivel_academico    tipo_vivienda      tiempo_cliente
##  Min.   : 8955   Length:4315        Length:4315        Min.   :  0.0
##  1st Qu.: 9204   Class :character   Class :character   1st Qu.:  0.0
##  Median :10813   Mode  :character   Mode  :character   Median :  4.0
##  Mean   :11125                                         Mean   : 12.3
##  3rd Qu.:12546                                         3rd Qu.: 14.0
##  Max.   :14916                                         Max.   :339.0
##  tiempo_sistema      porcend       obligaciones_sistema
##  Min.   :  0.00   Min.   :0.2041   Min.   :0.0000
##  1st Qu.:  0.00   1st Qu.:0.4870   1st Qu.:0.0000
##  Median :  0.00   Median :0.9463   Median :0.0000
##  Mean   : 16.49   Mean   :0.7667   Mean   :0.3145
##  3rd Qu.: 20.50   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :339.00   Max.   :1.0769   Max.   :4.0000
```

De lo anterior, se tiene que para `tiempo_actividad` se cuenta con un valor atípico (866880), el cuál se imputa por la mediana para evitar afectación en el desempeño de los modelos que se desarrollan más adelante.

```
mediana <- median(df$tiempo_actividad, na.rm = TRUE)
outlier <- 866880.0
df$tiempo_actividad[df$tiempo_actividad == outlier] <- mediana
```

Del enfoque de resolución del problema se estableció que `mora30` es el clasificador o variable de respuesta, de ahí que se proceda a efectuar la eliminación de `mora60` y `moramax_ultimosemestre`.

```
df <- df %>%
  select(-mora60, -moramax_ultimosemestre)
```

3.2. Creación de particiones de datos

Una vez obtenido el conjunto de datos a tratar, limpiado y transformado, es menester crear conjuntos de entrenamiento, prueba y validación que son insumo en las etapas de modelamiento y validación. Para esto, se aprovechan las funciones de `tidymodels` destinadas a este propósito y se crean los conjuntos de datos estratificados (para balancear la proporción de muestras pertenecientes a individuos que entran o no en mora).

```
#Definición de semilla
set.seed(1) #Definición de semilla
# Parámetro strata en mora30 para estrafificación
splits <- initial_split(df, strata = mora30)
df_training <- training(splits)
df_testing <- testing(splits)
set.seed(2)
df_validation <- validation_split(df_training,
                            strata = mora30,
                            prop = 0.80)
```

3.3. Definición de modelos

El paso siguiente es definir los modelos candidatos que más adelante se comparan en función de su desempeño y métricas. Para este, fin se opta por escoger dos modelos clásicos y uno derivado del aprendizaje profundo: regresión logística con penalización, bosque aletario y una red neuronal (perceptrón multicapa). La construcción de estos modelos a través de `tidymodels` es sencilla: se define el modelo y sus hiperparámetros, se escoge el `engine` o librería que va a realizar el modelado, así como su respectiva configuración y, por último, se define el tipo de modelo, que en este caso es `classification`. Adicionalmente se define una variable llamada `cores` que se invoca dentro de la función `set_engine` y que la utiliza el parámetro `num.threads` y sirve para entrenar modelos aprovechando la capacidad multinúcleo del procesador. Nótese que los hiperparámetros no se definen explícitamente, sino a través del objeto `tune()`, dado que sobre cada modelo candidato se pretende efectuar un *grid search* para encontrar los hiperparámetros óptimos en cada uno de estos.

```
# Para detectar el número de hilos del procesador
cores <- parallel::detectCores()

model_lr <- logistic_reg(penalty = tune(), mixture = 1) %>%
  set_engine("glmnet", num.threads = cores)

model_rf <- rand_forest(trees = tune(), mtry = tune(), min_n = tune()) %>%
  set_engine("ranger", num.threads = cores) %>%
  set_mode("classification")

model_neural <- mlp(hidden_units = tune(),
                    activation = tune(),
                    dropout = tune(),
                    epochs = 30) %>% # 30 épocas
  set_engine("keras", num.threads = cores) %>%
  set_mode("classification")
```

3.4. Definición de receta de preprocesamiento

Del conjunto de datos de entrenamiento, se tienen variables categóricas (o factores) y variables numéricas. La estructura en que se encuentra este *set* de datos no es la adecuada para efectuar el modelado, por tal motivo las variables numéricas deben ser sometidas a un proceso de normalización y escalado y las variables

categóricas ser puestas a punto a través de un proceso de *one hot encoding*, esto con el fin de acelerar el aprendizaje de los algoritmos y mejorar la precisión e interpretabilidad de los modelos. Para tal fin, se define sobre un *pipeline* de `tidymodels` la receta `recipe()` de preprocesamiento de datos.

```
df_rec <- recipe(mora30 ~ ., data = df_training) %>%
  # Se aplica codificación one hot sobre variables categóricas
  step_dummy(all_nominal_predictors()) %>%
  # Eliminar variables redundantes que están altamente correlacionadas
  step_corr(all_numeric_predictors(), threshold = 0.9) %>%
  # Normalización de las variables numéricas
  step_normalize(all_numeric_predictors()) %>%
  # Escalamiento de las variables numéricas
  step_scale(all_numeric_predictors())
```

3.5. Flujo de trabajo y búsqueda de hiperparámetros

A continuación, se definen los flujos de trabajo *worflows* empleando el objeto `workflow()` de `tidymodels` a través de un *pipeline* donde se iniciliza el mismo y se añaden los modelos y las recetas.

```
# Flujo de trabajo: regresión logística
lr_workflow <-
  workflow() %>%
  add_model(model_lr) %>%
  add_recipe(df_rec)

# Flujo de trabajo: bosque aleatorio
rf_workflow <-
  workflow() %>%
  add_model(model_rf) %>%
  add_recipe(df_rec)

# Flujo de trabajo: red neuronal
neural_workflow <-
  workflow() %>%
  add_model(model_neural) %>%
  add_recipe(df_rec)
```

3.5.1. Definición de cuadrículas de hiperparámetros

Como la intención es que sobre cada modelo se efectúe un *grid search* aprovechando el `tune_grid()` de `tidymodels` entonces se definen las cuadrículas de búsqueda, se almacenan y se pasan sobre los *pipelines* definidos para los flujos de trabajo y se efectuan así los procesos de búsqueda de hiperparámetros óptimos.

```
# Definición de cuadrícula para
# los hiperparámetros del bosque aleatorio
# Se evalúan trees, min_n y mtry
n_combinations <- 50 # Número de combinaciones
trees_bounds_rf <- c(10, 100) # trees varía de 10 a 100
min_n_bounds_rf <- c(1, 50) # min_ varía de 1 a 50
mtry_bounds_rf <- c(1, 10) # mtry varía de 1 a 10
trees_values_rf <- sample.int(trees_bounds_rf[2] - trees_bounds_rf[1],
                              n_combinations, replace = TRUE) + trees_bounds_rf[1]
min_n_values_rf <- sample.int(min_n_bounds_rf[2] - min_n_bounds_rf[1],
                            n_combinations, replace = TRUE) + min_n_bounds_rf[1]
mtry_values_rf <- runif(n_combinations, mtry_bounds_rf[1], mtry_bounds_rf[2])
```

```
# Cuadrícula de rf
grid_rf <- tibble(
  trees = trees_values_rf,
  min_n = min_n_values_rf,
  mtry = mtry_values_rf
)

# Definición de cuadrícula para
# el hiperparámetro de penalización en
# regresión logística
grid_lr <- tibble(penalty = 10^seq(-4, -1, length.out = 30))

# Definición de cuadrícula para
# hiperparámetros en perceptrón multicapa
# Capas ocultas, función de activación y dilución
grid_neural <- crossing(
  hidden_units = c(4, 8, 16, 32, 64),
  activation = c("relu"),
  dropout = c(0.01, 0.03, 0.05, 0.07, 0.1, 0.3, 0.5, 0.7, 0.9, 0.99),
)
```

3.5.2. *Grid search* sobre hiperparámetros

```
# Obtención de resultados a través de
# métricas de roc_auc, precisión y sensibilidad
# Grid search sobre conjunto de validación

# Grid search sobre regresión logística
lr_res <-
  lr_workflow %>%
  tune_grid(df_validation,
            grid = grid_lr,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(roc_auc, precision, sensitivity))

# Grid search sobre bosque aleato
rf_res <-
  rf_workflow %>%
  tune_grid(df_validation,
            grid = grid_rf,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(roc_auc, precision, sensitivity))

# Grid search sobre perceptrón multicapa
neural_res <-
  neural_workflow %>%
  tune_grid(df_validation,
            grid = grid_neural,
            control = control_grid(save_pred = TRUE),
            metrics = metric_set(roc_auc, precision, sensitivity))
```

Posteriormente, se almacenan los resultados de cada uno de los *grid search*:

```
rf_best <-
  rf_res %>%
  collect_metrics() %>%
  spread(.metric, mean) %>%
  arrange(desc(roc_auc))
rf_top <-
  rf_res %>%
  select_best(metric = "roc_auc")
rf_auc <-
  rf_res %>%
  collect_predictions(parameters = rf_top) %>%
  roc_curve(mora30, .pred_0) %>%
  mutate(model = "Random Forest")

lr_best <-
  lr_res %>%
  collect_metrics() %>%
  spread(.metric, mean) %>%
  arrange(desc(roc_auc))
lr_top <-
  lr_res %>%
  select_best(metric = "roc_auc")
lr_auc <-
  lr_res %>%
  collect_predictions(parameters = lr_top) %>%
  roc_curve(mora30, .pred_0) %>%
  mutate(model = "Logistic Regression")

neural_best <-
  neural_res %>%
  collect_metrics() %>%
  spread(.metric, mean) %>%
  arrange(desc(roc_auc))
neural_top <-
  neural_res %>%
  select_best(metric = "roc_auc")
neural_auc <-
  neural_res %>%
  collect_predictions(parameters = neural_top) %>%
  roc_curve(mora30, .pred_0) %>%
  mutate(model = "Neural Network")
neural_best_auc <-neural_best$roc_auc[1]
```

3.5.3. Evaluación de métricas y entrenamiento de modelo

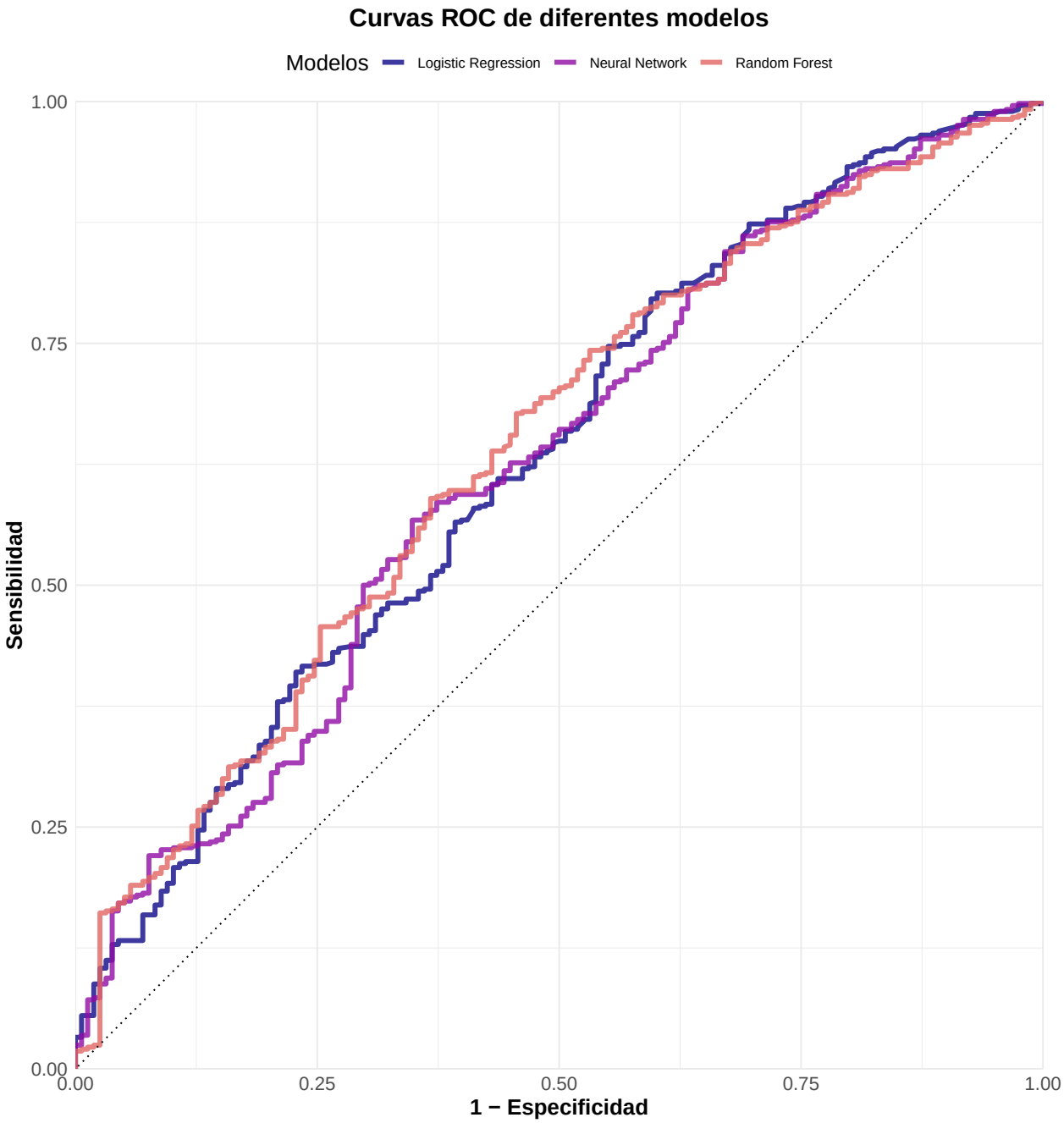
A continuación, en base a los resultados de los *grid search*, se tiene que:

- El `roc_auc` para la regresión logística es de: 0.629
- El `roc_auc` para el bosque aletorio es de: 0.64
- El `roc_auc` para la red neuronal es de: 0.624

Así mismo, se presenta la gráfica de las curvas ROC para cada uno de los modelos candidatos:

```
bind_rows(rf_auc, lr_auc, neural_auc) %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity, col = model)) +
```

```
  geom_path(linewidth = 1.5, alpha = 0.8) +
  geom_abline(lty = 3) +
  coord_equal() +
  scale_color_viridis_d(option = "plasma", end = .6) +
  labs(title = "Curvas ROC de diferentes modelos",
       x = "1 - Especificidad", y = "Sensibilidad", col = "Modelos") +
  scale_x_continuous(limits = c(0, 1), expand = c(0, 0)) +
  scale_y_continuous(limits = c(0, 1), expand = c(0, 0)) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"),
        legend.position = "top", # Ubica la leyenda debajo de la gráfica
        legend.justification = "center", # Centra la leyenda
        legend.title = element_text(size = 14),
        axis.title = element_text(size = 14, face = "bold"),
        axis.text = element_text(size = 12))
```



En base a lo anterior, el mejor modelo, de entre los evaluados, es el **bosque aleatorio** con `roc_auc` de 0.64, precisión de 0.757 y sensibilidad de 0.998. De todas formas, es de aclarar, que el rendimiento de los modelos candidatos es bastante moderado, ligeramente superior al azar.

3.5.4. Prueba de modelo escogido con hiperparámetros óptimos

```
# Se establece modelo de validación
last_rf_mod <-
  rand_forest(mtry = rf_best$mtry[1],
              min_n = rf_best$min_n[1],
              trees = rf_best$trees[1]) %>%
  set_engine("ranger", num.threads = cores,
```

```
              importance = "impurity") %>%
  set_mode("classification")

# Se establece worflow
last_rf_workflow <-
  rf_workflow %>%
  update_model(last_rf_mod)

# Se establece ajuste
set.seed(123)
last_rf_fit <-
  last_rf_workflow %>%
  last_fit(splits)
```

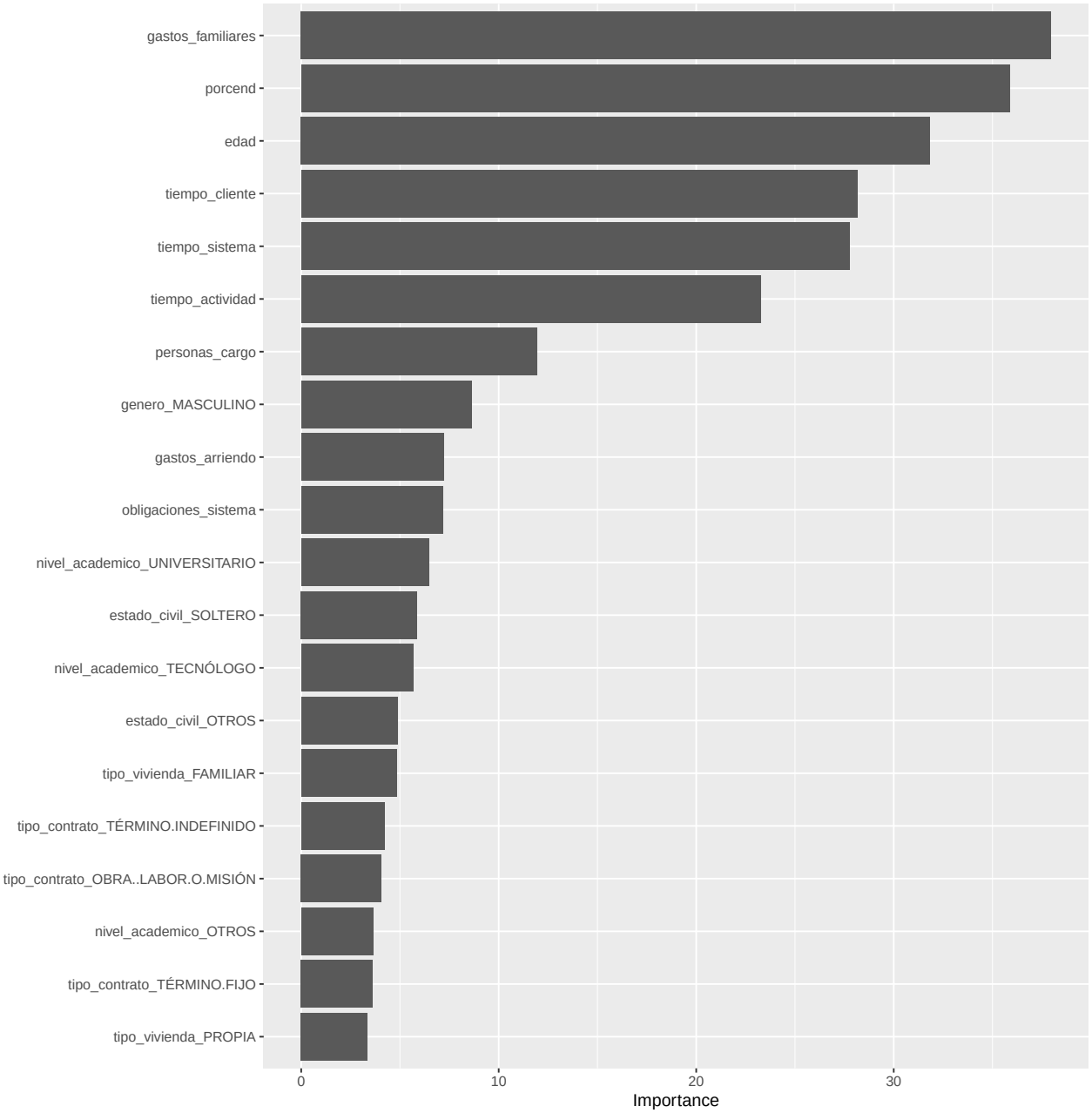
Se visualizan los resultados de la predicción sobre datos no vistos:

```
last_rf_metrics <- last_rf_fit %>%
  collect_metrics()
```

De lo anterior, se tiene que la métrica de `roc_auc` sobre datos no vistos es de 0.615918325505107. El cual, como es de esperar, es inferior al obtenido en el entrenamiento. Así mismo, la precisión es de 0.753 o, lo que es lo mismo, el modelo acertó un 75.3 % de las veces.

Por curiosidad, se examinan las 20 variables que más influyen en la predicción:

```
last_rf_fit %>%
  extract_fit_parsnip() %>%
  vip(num_features = 20)
```



4. Metodología: segundo punto

Teniendo en cuenta los requerimientos de este punto, la intención es elegir de entre dos modelos de clasificación de clientes en base a su puntaje e incumplimiento. Los criterios de elección se presentan en el enunciado: capacidad discriminante, estabilidad y distribución de buenos y malos. Con esto en mente, el enfoque es, una vez extraído el conjunto de datos, partirlo en dos *sets*: cliente AB y cliente XY. Posteriormente, dado que tenemos solo dos variables de importancia en cada *set* (puntuación e incumplimiento), se halla la correlación entre estas dos variables para tener un primer acercamiento respecto a la relación de cada variable de puntuación con el incumplimiento predicho. Más adelante, se efectúa un proceso de entrenamiento de un modelo de regresión logística sin penalización sobre cada *set* de datos a través de validación cruzada (*cross validation*) para determinar la capacidad discriminante y estabilidad de cada modelo y así tener herramientas suficientes para comparar los modelos en cuestión.

5. Resultados: segundo punto

```
# Se cargan las librerías necesarias
library(ltm)
library(naniar)

# Se importa el archivo en formato .xlsx a un data frame en formato tibble
df <- read_excel(ruta, col_types = c("skip", "skip", "numeric",
                                                    "text", "numeric")) %>%
  as_tibble()

colnames(df) <- c("puntaje_AB", "puntaje_XY", "default")
df$puntaje_XY[df$puntaje_XY == "."] <- NA
df$puntaje_XY <- as.integer(df$puntaje_XY)
df$puntaje_AB <- as.integer(df$puntaje_AB)
df$default <- as.factor(df$default)
summary(df)
```

```
##    puntaje_AB      puntaje_XY    default
##  Min.   : 6.00   Min.   : 0.00   0:45135
##  1st Qu.: 9.00   1st Qu.:17.00   1:68974
##  Median :10.00   Median :21.00
##  Mean   : 9.66   Mean   :19.52
##  3rd Qu.:10.00   3rd Qu.:23.00
##  Max.   :12.00   Max.   :27.00
##                  NA's   :12800
```

El porcentaje de datos nulos en la columna `puntaje_XY` es de 11.22 %. Por tal motivo, se evalúa si la procedencia de los datos nulos se debe a factores aleatorios o no, de ahí que un enfoque que se aplica en el presente análisis es la realización de una prueba MCAR (*Missing Completely at Random*) en donde la hipótesis nula es que los datos faltantes son un MCAR, es decir, que no hay un patrón específico en los datos faltantes. Valores $p < 0.05$ indican, por el contrario que los datos faltantes no son un MCAR.

```
mcar_test(df[c("puntaje_XY", "default")])
```

```
## # A tibble: 1 x 4
##   statistic    df p.value missing.patterns
##       <dbl> <dbl>   <dbl>            <int>
## 1     0.476     1   0.490                2
```

De la prueba anterior se desprende que, con un valor p » 0.05, los datos faltantes son un MCAR. Así pues, se procede con la eliminación de dichos valores faltantes y con la división en los dos *sets* de datos.

```
# Eliminación de valores faltantes
df <- na.omit(df)
# Partición de los sets de datos
df_AB <- df[c("puntaje_AB", "default")]
df_XY <- df[c("puntaje_XY", "default")]
```

A continuación, se hallan las correlaciones para cada *set* de datos entre la puntuación y el *default*. Para esto, dado que tenemos una variable continua y otra categórica binaria, empleamos la correlación biserial puntual, así pues, tenemos que para el *set* cliente AB, la correlación biserial puntual es de $-8{,}5741894 \times 10^{-5}$ y la correlación para el *set* del cliente XY es de -0.0026784. Ambas correlaciones son muy bajas, para efectos prácticos 0, de ahí que en cada uno de los modelos planteados no se tenga correlación entre el puntaje y el *default*, por lo que se puede intuir, de primera mano, que desarrollando un modelo de predicción sobre cada *set* de datos, sus capacidades predictivas son equiparables a las del mero azar. Aún así, para dar seguimiento

a la metodología planteada, se continúa con el ejercicio.

5.1. Particionado y modelado

```
# e define semilla
set.seed(123)

# Se particionan datos
AB_split <- initial_split(df_AB, prop = 0.8)
AB_train <- training(AB_split)
AB_test <- testing(AB_split)
XY_split <- initial_split(df_XY, prop = 0.8)
XY_train <- training(XY_split)
XY_test <- testing(XY_split)

# Se define modelo regresión logística
## Válido para ambos sets AB y XY
lr_mod <-
  logistic_reg(mode = "classification") %>%
  set_engine("glm")

# Se ajustan modelos
lr_fit_AB <-
  lr_mod %>%
  fit(default ~ ., data = AB_train)

lr_fit_XY <-
  lr_mod %>%
  fit(default ~ ., data = XY_train)

lr_training_pred_AB <-
  predict(lr_fit_AB, AB_train) %>%
  bind_cols(predict(lr_fit_AB, AB_train, type = "prob")) %>%
  # Add the true outcome data back in
  bind_cols(AB_train %>%
              dplyr::select(default))

lr_training_pred_XY <-
  predict(lr_fit_XY, XY_train) %>%
  bind_cols(predict(lr_fit_XY, XY_train, type = "prob")) %>%
  # Add the true outcome data back in
  bind_cols(XY_train %>%
              dplyr::select(default))
```

5.2. Verificación métricas y *cross validation*

Se presenta el resultado de la métrica `roc_auc`, sobre el conjunto de entrenamiento, para el *set* cliente AB:

```
lr_training_pred_AB %>%
  roc_auc(truth = default, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
```

```
## 1 roc_auc binary         0.500
```

Así mismo, se presenta el resultado de la métrica `roc_auc`, sobre el conjunto de entrenamiento, para el *set* cliente XY:

```
lr_training_pred_XY %>%
  roc_auc(truth = default, .pred_0)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.503
```

Con valores de `roc_auc` de aproximadamente 0.5 sobre cada conjunto de entrenamiento, se puede decir con certeza que ambos modelos de clasificación no tienen capacidad predictiva y se comportan similar a la elección aleatorio de la variable de predicción. Así pues, se continúa con el ejercicio, aunque las conclusiones derivadas del entrenamiento saltan a la vista.

```
# Se establece semilla
set.seed(789)
# Se establecen los folds (5)
folds <- vfold_cv(AB_train, v = 5)
# Se genera el worflow
lr_wf_AB <-
  workflow() %>%
  add_model(lr_mod) %>%
  add_formula(default ~ .)
# Se establece nueva semilla
# Se ajusta aplicando remuestreo
set.seed(321)
lr_fit_rs_AB <-
  lr_wf_AB %>%
  fit_resamples(folds)

# Se establece semilla
set.seed(123)
folds <- vfold_cv(XY_train, v = 5)
# Se genera el worflow
lr_wf_XY <-
  workflow() %>%
  add_model(lr_mod) %>%
  add_formula(default ~ .)
# Se establece nueva semilla
# Se ajusta aplicando remuestreo
set.seed(456)
lr_fit_rs_XY <-
  lr_wf_XY %>%
  fit_resamples(folds)
```

Se aplica `collect_metrics` sobre los modelos sometidos a validación cruzada y se inspeccionan sus métricas:

Para el *set* AB:

```
collect_metrics(lr_fit_rs_AB)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n  std_err .config
##   <chr>    <chr>      <dbl> <int>    <dbl> <chr>
```

```
## 1 accuracy binary     0.604     5 0.00174  Preprocessor1_Model1
## 2 roc_auc  binary     0.496     5 0.000581 Preprocessor1_Model1
```

Y, para el *set* XY:

```
collect_metrics(lr_fit_rs_XY)
```

```
## # A tibble: 2 x 6
##   .metric  .estimator  mean     n std_err .config
##   <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1 accuracy binary     0.605     5 0.00164 Preprocessor1_Model1
## 2 roc_auc  binary     0.499     5 0.00201 Preprocessor1_Model1
```

De lo anterior, se desprende que incluso siguiendo el enfoque de validación cruzada las métricas, en general, no difieren y los modelos planteados por los clientes no tienen capacidad predictiva significativamente mejor que el azar y, por tanto, no se puede elegir entre ninguno de los dos modelos.

6. Solución tercer punto

Las principales variables macroeconómicas que, considero, se deben estar monitoreando y, de ser posible, ensamblar en modelos de toma de decisiones son:

- Crecimiento del PIB: es quizá la variable macroeconómica de referencia, puesto que indica la tasa de crecimiento (o decrecimiento) en la suma de bienes y servicios producidos en el país. En valor positivo y elevado en esta variable significa, de forma muy general, que los individuos están percibiendo mayores ingresos en su trabajo, lo que les permite aumentar su ritmo de consumo y ahorro.
- Índice de precios del consumidor (IPC) e inflación: el IPC es un indicador que hace referencia a la variación en el promedio de precios para un grupo de bienes y servicios y representa, en términos prácticos, el costo de vida. El crecimiento del IPC es lo que se conoce como inflación, y es de suma importancia porque la variación en este indicador afecta la capacidad de compra de los individuos, la redistribución de los ingresos y la competitividad de las empresas.

- Informe sobre Mercado Laboral del DANE y Gran Encuesta Integrada de Hogares: con esta información se puede tener un panorama sobre la tasa de desempleo a nivel nacional, regional, departamental y a nivel de capitales.
- Productividad Total de los Factores: es una variable económica vital para medir el crecimiento y desarrollo de la economía. Se calcula relacionando el PIB con los factores que lo generan a nivel de capital y trabajo. Es útil monitorear su valor porque da una mejor idea sobre la competitividad, el crecimiento económico sostenible a largo plazo y el bienestar social general en la población.
- Reportes de Inclusión Financiera: estos reportes son interesantes porque realizan diagnósticos sobre la evolución en el acceso y aceptación de servicios financieros.
- Tasa Representativa del Mercado (TRM): el aumento o disminución de la relación dólar/peso colombiano favorecen o desfavorecen a distintos sectores de la economía.
- Tipos de interés Fed: el aumento en los tipos de interés de lal Sistema de Reserva Federal suele ser un indicador global de enfriamiento de la economía con fines de reducción de la inflación en los Estados Unidos, lo que suele traducirse en disminución en la demanda de materias primas colombianas que son pagadas con divisas extranjeras, así mismo la subida de tipos puede presional el alza en el precio del dólar y en la política monetaria del Banco de la República.
- Tipos de interés BCC: la subida en los tipos de interés del Banco de la República suele traducirse en intentos por contener la inflación y "enfriar" la economía reduciendo la oferta de dinero, lo cual tiene un impacto directo porque encarece el acceso al crédito, puede aumentar el desempleo puesto que las empresas están menos dispuestas a contratar personal debido a que sus gastos operacionales se destinan a pago de deuda, aumenta la morosidad por cartera vencida y disminuye el margen de las empresas fuertemente apalancadas en el crédito.

7. Conclusiones

En el caso de la solución del primer punto:

- Se desarrolló un modelo de clasificación de clientes basado en su probabilidad de mora. El modelo escogido, de entre tres candidatos, fue un bosque aleatorio. Las métricas de este modelo a penas estuvieron del orden de 1.5 puntos porcentuales por encima de los otros dos modelos candidatos (regresión logística con penalización y una red neuronal)
- Las métricas obtenidas en los modelos de clasificación resultaron bastante modestas a pesar de haber desarrollado un enfoque basado en optimización de hiperparámetros, por lo que es menester revisar la metodología y optar por enfoques alternativos que mejoren las métricas en la clasificación.

En el caso de la solución del segundo punto:

- Se determinó una metodología para el tratamiento de los datos faltantes derivados de la presencia del caracter . (punto) que fueron convertidos a valores nulos. Dicha metodología probó que la naturaleza de dichos valores ausentes se debe a la aleatoriedad.
- Se pudo realizar una prueba de correlación entre el puntaje y el *default* para los *sets* de datos de los clientes y se determinó que la correlación es aproximadamente 0.
- En base a la correlación de aproximadamente 0 se probó, en conjunto con el análisis predictivo, que un modelo de clasificación basado en el puntaje propuesto no puede tener un mejor rendimiento que uno debido al mero azar.
- La solución al problema probablemente resultó tautológica, sin embargo, esto pudo deberse a la incomprensión del problema, por parte del autor de este análisis, por lo que se recomienda una mejor acotación del mismo para evitar ambiguedades en la solución del problema.

8. Información sobre sesión

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.2 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
##
## locale:
## [1] en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] naniar_1.0.0       ltm_1.2-0          polycor_0.8-1      msm_1.7
##  [5] MASS_7.3-58.1      magrittr_2.0.3     keras_2.11.0       ranger_0.14.1
##  [9] glmnet_4.1-6       Matrix_1.5-1       vip_0.3.2          readxl_1.4.2
## [13] lubridate_1.9.2    forcats_1.0.0      stringr_1.5.0      readr_2.1.4
## [17] tidyverse_2.0.0    yardstick_1.1.0    workflowsets_1.0.0 workflows_1.1.3
## [21] tune_1.0.1         tidyr_1.3.0        tibble_3.1.8       rsample_1.1.1
## [25] recipes_1.0.2      purrr_1.0.1        parsnip_1.0.4      modeldata_1.1.0
## [29] infer_1.0.4        dplyr_1.1.0        dials_1.1.0        scales_1.2.1
```

```
## [33] broom_1.0.4        tidymodels_1.0.0   ggplot2_3.4.1
##
## loaded via a namespace (and not attached):
##  [1] colorspace_2.0-3   ellipsis_0.3.2     class_7.3-20       visdat_0.6.0
##  [5] rprojroot_2.0.3    base64enc_0.1-3    rstudioapi_0.14    listenv_0.8.0
##  [9] furrr_0.3.1        farver_2.1.1       prodlim_2019.11.13 fansi_1.0.3
## [13] mvtnorm_1.1-3      codetools_0.2-18   splines_4.1.2      knitr_1.40
## [17] zeallot_0.1.0      jsonlite_1.8.4     png_0.1-7          tfruns_1.5.1
## [21] compiler_4.1.2     backports_1.4.1    fastmap_1.1.0      cli_3.6.0
## [25] admisc_0.31        htmltools_0.5.3    tools_4.1.2        gtable_0.3.1
## [29] glue_1.6.2         Rcpp_1.0.9         cellranger_1.1.0   DiceDesign_1.9
## [33] vctrs_0.6.0        iterators_1.0.14   timeDate_4021.106  modelenv_0.1.1
## [37] gower_1.0.0        xfun_0.34          globals_0.16.1     timechange_0.2.0
## [41] lifecycle_1.0.3    future_1.28.0      ipred_0.9-13       hms_1.1.2
## [45] parallel_4.1.2     expm_0.999-7       yaml_2.3.6         reticulate_1.26
## [49] gridExtra_2.3      rpart_4.1.16       stringi_1.7.8      highr_0.9
## [53] tensorflow_2.9.0   foreach_1.5.2      lhs_1.1.6          hardhat_1.2.0
## [57] lava_1.6.10        shape_1.4.6        rlang_1.1.0        pkgconfig_2.0.3
## [61] evaluate_0.17      lattice_0.20-45    labeling_0.4.2     tidyselect_1.2.0
## [65] norm_1.0-10.0      here_1.0.1         parallelly_1.32.1  R6_2.5.1
## [69] generics_0.1.3     pillar_1.8.1       whisker_0.4        withr_2.5.0
## [73] survival_3.2-13    nnet_7.3-17        future.apply_1.9.1 utf8_1.2.2
## [77] tzdb_0.3.0         rmarkdown_2.17     grid_4.1.2         digest_0.6.30
## [81] GPfit_1.0-8        munsell_0.5.0      viridisLite_0.4.1
```