Modelo de clasificación: Suscripción

Samuel Calderon, Kevin Arenas, Jose Torres

INF648-PUCP-2024-01 - Aprendizaje Automático

1 Introducción

1.1 Cargado de librerías

Se decidió usar R en lugar de Python. Para este trabajo solo se necesitan dos librerías. `tidyverse` para la lectura, limpieza, y análisis de data, y `tidymodels` para la definición y ejecución de modelos de machine learning.

```
library(tidyverse)
library(tidymodels)
```

1.2 Lectura de datos

Al leer los datos, se aprovecha para hacer la limpieza respectiva. En este caso solo se necesita convertir las variables de tipo texto en “factor” (categóricas).

```
# Data de entrenamiento
suscripcion <- read_csv("data/train.csv") |>
  mutate_if(is.character, as.factor)

# Data de test
validacion <- read_csv("data/test_x.csv") |>
  mutate_if(is.character, as.factor)
```

Las funciones de R suelen soportar el operador pipe (| >), que permite pasar el output de una función como primer parámetro de la siguiente.

```
dim(suscripcion)
```

```
[1] 32538    22
```

```
dim(validacion)
```

```
[1] 12769    21
```

Se puede ver las dimensiones de cada conjunto de datos.

1.3 Flujo de trabajo de tidymodels

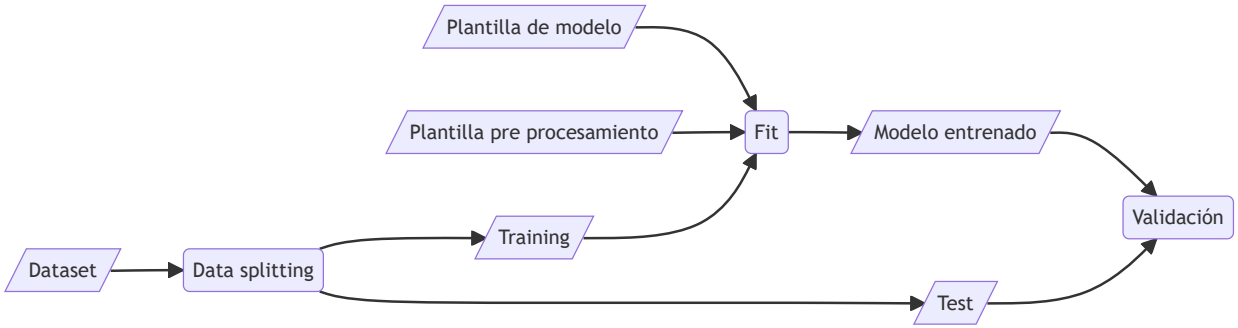


Figure 1

1.4 Data splitting

Dividimos `suscripcion` en un conjunto de entrenamiento (75%) y uno de test (25%).

```
set.seed(42)

data_split <- initial_split(suscripcion, prop = 3/4)

train_data <- training(data_split)
test_data  <- testing(data_split)
```

Alternativamente, preparamos un conjunto para validación cruzada de 10 sub-grupos. Estratificamos según la variable dependiente para evitar demasiado *inbalance*. Esto se utilizará más adelante.

```
folds_cv <- vfold_cv(suscripcion, strata = `Subscripcion Deposito`, v = 10)
```

2 Regresión logística

2.1 Plantilla de preprocesamiento

Para el pre procesamiento empezamos definiendo la fórmula del modelo. El punto hace referencia a todo el resto de variables.

```
my_recipe <- recipe(`Subscripcion Deposito` ~ ., data = suscripcion) |>
  update_role(Id, Duracion, new_role = "ignored") |>
  step_dummy(all_nominal_predictors()) |>
  step_normalize(all_numeric_predictors()) |>
  step_zv(all_predictors())
```

Esta plantilla se va a utilizar también para otros modelos.

2.2 Plantilla de modelo

Para el primer ejemplo, usamos la regresión logística. `set_engine()` y `set_mode()` permiten personalizar la implementación del modelo.

```
my_model <- logistic_reg() |>
  set_engine("glm") |>
  set_mode("classification")
```

2.3 Definición de workflow

Para poder combinar ambas plantillas, las agregamos a un mismo *workflow*.

```
my_workflow <- workflow() |>
  add_model(my_model) |>
  add_recipe(my_recipe)
```

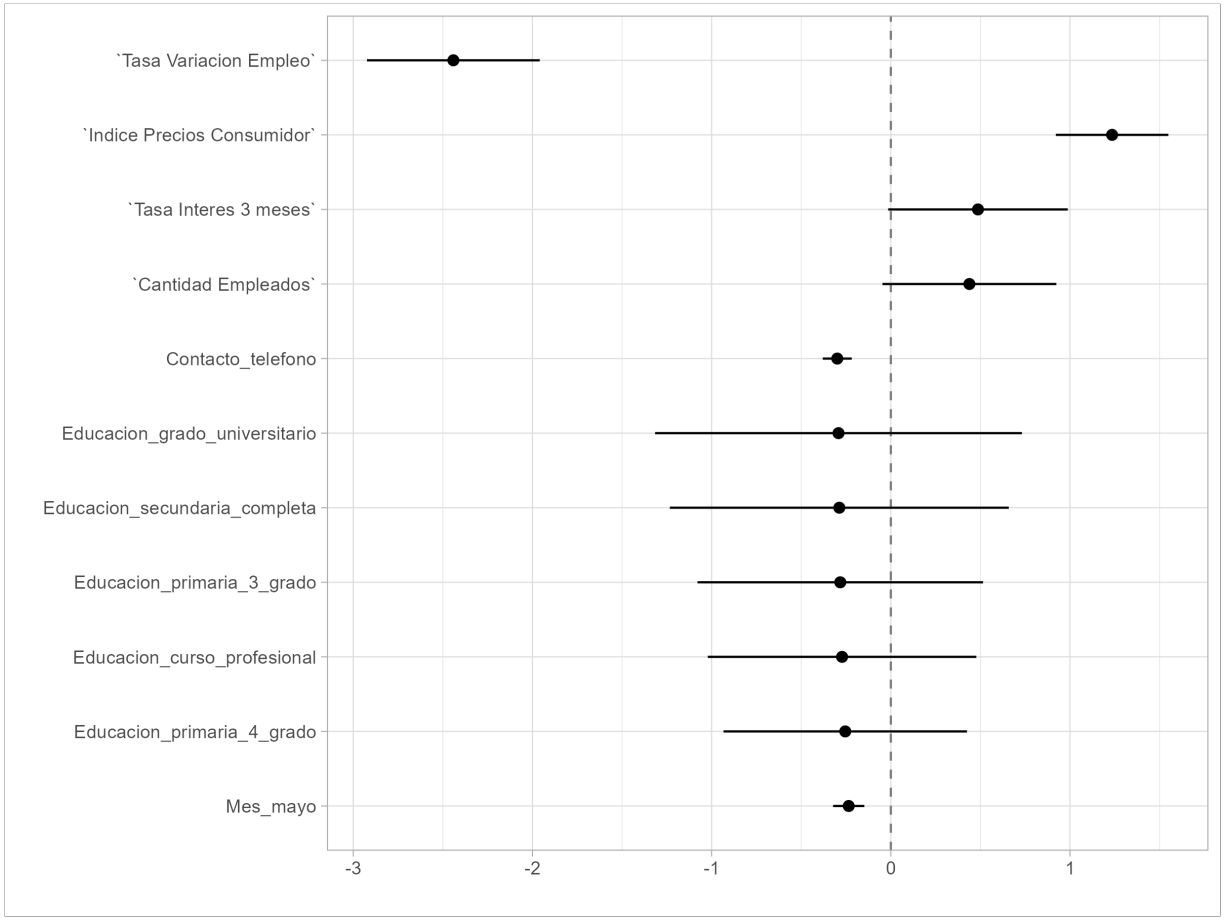
2.4 Fit

Para entrenar el modelo, tomamos el *workflow* como punto de partida, y le hacemos `fit()` usando la data de entrenamiento.

```
suscripcion_fit <- my_workflow |>
  fit(data = train_data)
```

El siguiente gráfico muestra las 12 variables con mayores coeficientes.

▶ Code



INF648-PUCP-2024-01 - Aprendizaje Automático

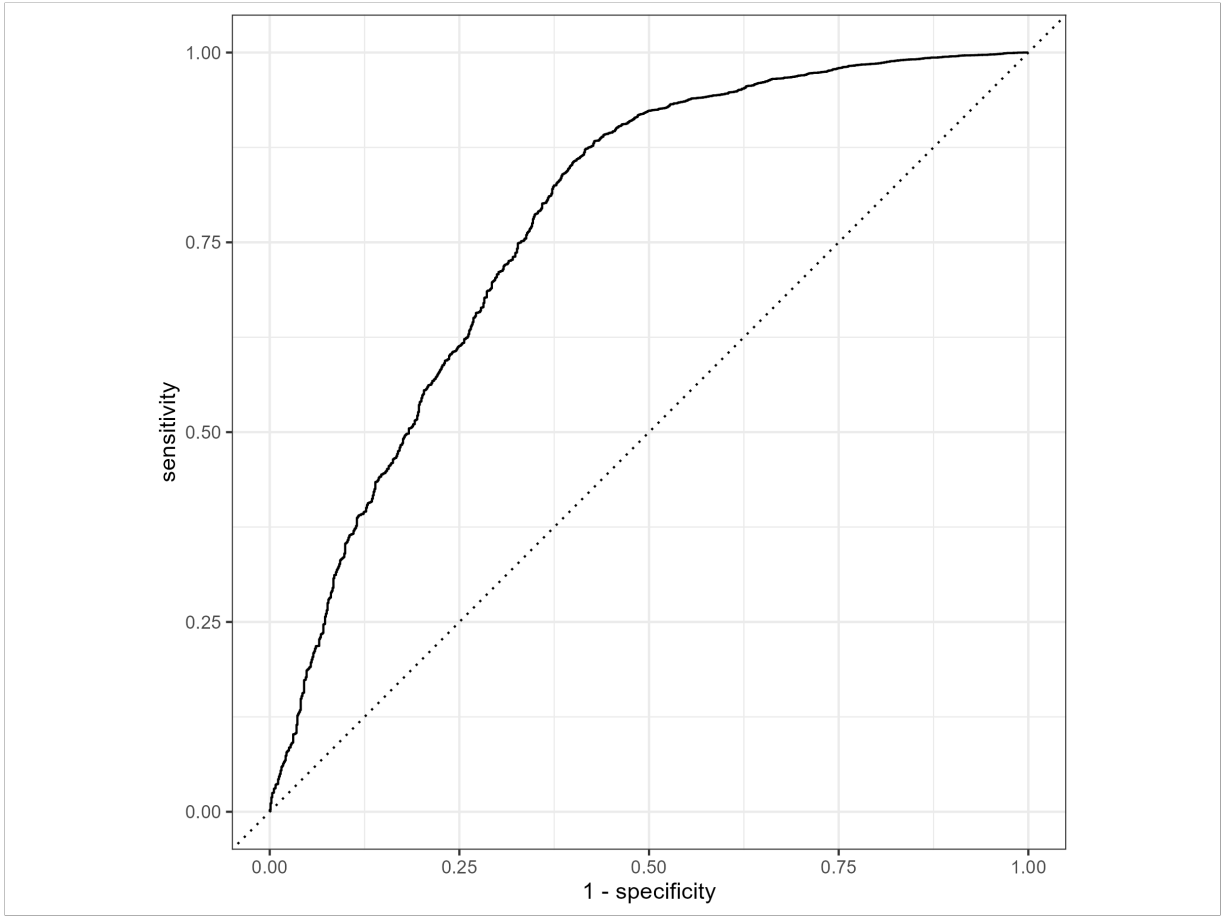
2.5 Validación

Para esto, mostramos el cálculo del AUC y su gráfico correspondiente. Esto se obtiene a partir de hacer predicciones con el modelo entrenado, usando la data de test.

▶ Code

.metric	.estimator	.estimate
roc_auc	binary	0.7773887

▶ Code


1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 - specificity

3 Prueba con otros modelos

3.1 Workflow múltiple

Se mantiene la plantilla de pre-procesamiento, pero creamos un listado de plantillas de modelos. Para poder, comparar, incluímos la plantilla de regresión logística.

```
nuevo_workflow <- workflow_set(
  preproc = list(
    recipe = my_recipe
  ),
  models = list(
    logistica = logistic_reg(mode = "classification"),
    arbol_decision = decision_tree(mode = "classification"),
    random_forest = rand_forest(mode = "classification"),
    boosted_tree = boost_tree(mode = "classification")
  )
)
```

3.2 Entrenamiento múltiple

Para el entrenamiento, se usa el workflow múltiple y el split hecho para cross validation.

```
set.seed(42) # volvemos a usar semilla por el random forest

suscripcion_fit_multiple <- workflow_map(
  object = nuevo_workflow,
  fn = "fit_resamples",
  resamples = folds_cv, # generado en el data splitting
  control = control_resamples(save_workflow = TRUE),
  verbose = TRUE
)
```

Con ello, se obtuvieron 10 resultados para evaluar cada modelo.

El siguiente gráfico muestra el desempeño según tres indicadores.

▶ Code

También es posible mirar a los promedios (con barra de error) de los tres indicadores. En general, los cuatro están bastante cercanos entre sí, y en varios casos sus errores se superponen.

▶ Code

Nos podemos concentrar en el promedio del indicador **AUC** para elegir el modelo. En este caso, *boosted_tree* ocupa el primer lugar.

▶ Code

modelo	roc_auc	rank
boosted_tree	0.7975378	1
logistica	0.7858089	2
random_forest	0.7840145	3
arbol_decision	0.7029308	4

4 Entrenamiento final

4.1 Nuevo entrenamiento

Ahora que sabemos que `boosted_tree` tuvo mejor desempeño. Entrenamos el 100% de los datos con este algoritmo.

```
my_boosted_tree <- workflow() |>
  add_recipe(my_recipe) |>
  add_model(boost_tree(mode = "classification")) |>
  fit(data = suscripcion)
```

4.2 Predicciones

Para predecir, usamos la función `augment()`, que añade las predicciones al como nuevas columnas en el conjunto de datos

```
resultados <- my_boosted_tree |>
  augment(validacion)
```

```
# A tibble: 12,769 × 24
   .pred_class .pred_no .pred_si    Id  Edad Trabajo    `Estado Civil` Educacion
   <fct>          <dbl>    <dbl> <dbl> <dbl> <fct>      <fct>          <fct>
 1 no             0.914   0.0859     0    30 trabajado… casado         primaria…
 2 no             0.952   0.0482     1    39 servicios  soltero        secundar…
 3 no             0.931   0.0692     2    25 servicios  casado         secundar…
 4 no             0.940   0.0596     3    38 servicios  casado         primaria…
 5 no             0.958   0.0419     4    47 administr… casado         grado_un…
 6 no             0.739   0.261      5    32 servicios  soltero        grado_un…
 7 no             0.652   0.348      6    32 administr… soltero        grado_un…
 8 no             0.958   0.0424     7    41 emprended… casado         grado_un…
 9 no             0.952   0.0479     8    31 servicios  divorciado     curso_pr…
10 no             0.972   0.0280     9    35 trabajado… casado         primaria…
# ℹ 12,759 more rows
# ℹ 16 more variables: `Credito por Defecto` <fct>, `Prestamo Vivienda` <fct>,
#   `Prestamo Personal` <fct>, Contacto <fct>, Mes <fct>, `Dia Semana` <fct>,
```

▶ Code

4.3 Guardar resultados

Guardamos los resultados en un nuevo archivo para ser subidos a Kaggle.

```
resultados |>
  select(Id, `Subscripcion Deposito` = .pred_si) |>
  write_csv("resultados.csv")
```

4.4 Posibles mejoras

- *Tunear* los parámetros del modelo
- Selección de características post comparación de modelos
- Limpieza/imputación de datos
  En realidad se predijo con todos los modelos presentados. En la plataforma, los resultados de `random_forest` obtuvieron mayor puntaje.

Gracias!