# Modelos de clasificación para predecir ventas de pólizas de seguro

Jordi Vanrell Forteza

30/1/2021

## 1. Marco general, base de datos y estructura

La finalidad de este estudio es encontrar el mejor modelo predictivo para que determinada compañía pueda clasificar, entre los clientes que previamente tienen contratado un seguro médico, a los interesados en una póliza para su vehículo.

La base de datos original procede de Kaggle. Únicamente se hace uso de una parte de los datos del fichero `train.csv`, los datos de entrenamiento, que contiene información anonimizada de unos 380,000 clientes respecto de las siguientes variables:

- *id*: número de identificación del cliente.
- *Gender*: género del cliente (valores *Male*, *Female*)
- *Age*: edad del cliente (en años)
- *Driving_License*: el cliente tiene (1) o no (0) carnet de conducir.
- *Region_Code*: código numérico de la región donde vive el cliente (53 distintos)
- *Previously_Insured*: el cliente ya tiene (1) o no póliza de seguro.
- *Vehicle_Age*: tiempo que tiene el vehículo, en tres categorías (*<1 year*, *1-2 years* y *>2 years*)
- *Vehicle_Damage*: si el cliente tuvo (*Yes*) o no (*No*) un accidente en el pasado.
- *Annual_Premium*: la prima anual que pagaría el cliente.
- *PolicySalesChannel*: código del canal de ventas (155 distintos)
- *Vintage*: antigüedad del cliente en la compañía (en días)
- *Response*: si finalmete contrató (1) o no (0) la póliza de seguro de vehículos.

Esta última es la variable dependiente en los modelos.

## 2. Preparación de la base de datos

Tras la carga de datos, por limitaciones computacionales, lo primero que se hace es extraer una submuestra del 3% de las observaciones del *data set*.

Luego se procede a codificar a valores numéricos las variables *Gender*, *Vehicle_Damage* y *Vehicle_Age*, que se transforma en dos *dummies* (la categoría de referencia es *<1 year*).

Después se hacen dos *target_encoding* con *Region_Code* y *Policy_Sales_Channel*. Cabe señalar que, de las 155 categorías de esta última, las categorías *26*, *124* y *152* suponen más del 75% de las observaciones. Por esta razón se opta por generar dos *dummies* (*Channel_124* y *Channel_26*; *152* se convierte en la categoría de referencia) y conservar solamente el *target encoding* de todas las demás. Finalmente, anticipando la necesidad de hacer *SMOTE* se transforman temporalmente *todas* las variables a factores para evitar valores fuera de los rangos razonables de las variables (p.e. menores de edad, pólizas negativas, etc.)

Luego se divide la base de datos en dos conjuntos, el 70% de entrenamiento y el 30% de test. Se dispone de 11433 observaciones; se considera que la mitad de la submuestra es suficiente para modelizar en una primera etapa y se reserva un 20% del entrenamiento como datos de validación. Esto no sería realmente necesario porque pretende aplicarse validación cruzada, pero los valores de la variable dependiente están muy desbalanceados y será necesario hacer un SMOTE sobre los datos de entrenamiento. Con la muestra de validación se asegura testear efectivamente la exactitud predictiva sobre observaciones reales y no sobre observaciones sintéticas fruto del proceso SMOTE.

Table 1: Distribución de la variable dependiente (datos de entrenamiento antes de SMOTE)

| Real 0 | Real 1 |
|---|---|
| 4996 | 720 |

# 3. Métodos de aprendizaje supervisado

## 3.1. Modelo de clasificación logística (Logit)

Se desea estimar el primer modelo sencillo en base a una regresión logística de clasificción. Para ello primero se transforman los datos para tenerlos en formato numérico. A continuación se efectúa la estimación por máxima verosimilitud de los coeficientes del modelo con el fin de maximizar la probabilidad de que una observación se encuentre efectivamente en la categoría de la variable *Response* que recogen los datos, usando la función `glm` del paquete `caret`.

Después se usa la muestra de validación para predecir *Response* y compararla con los datos observados de esta variable. Puede decirse que, en este caso, el modelo *Logit* arroja exactitud predictiva del 70.84% sobre la muestra de validación. Véase la matriz de confusión:

Table 2: Matriz de confusión (Logit)

| | Real 0 | Real 1 |
|---|---|---|
| Pred 0 | 1360 | 15 |
| Pred 1 | 652 | 260 |

## 3.2 Análisis discriminante lineal (ADL)

Otro modelo sencillo aplicable a problemas de clasificación es el ADL. Este método modeliza la distribución de las variables explicativas por separado (funciones de densidad por categorías) y luego se sirve de las probabilidades de las categorías de *Response* condicionadas a las variables explicativas para clasificar las observaciones en la categoría con mayor puntuación discriminante, de acuerdo con sus valores y las funciones de densidades de las variables explicativas.

Tras usar la muestra de validación para predecir *Response* y compararla con los datos observados, el ADL arroja una exactitud predictiva del 65.19% sobre la muestra de validación. Véase la matriz de confusión:

Table 3: Matriz de confusión (LDA)

| | Real 0 | Real 1 |
|---|---|---|
| Pred 0 | 1222 | 6 |
| Pred 1 | 790 | 269 |

## 3.3. Métodos de clasificación basados en árboles

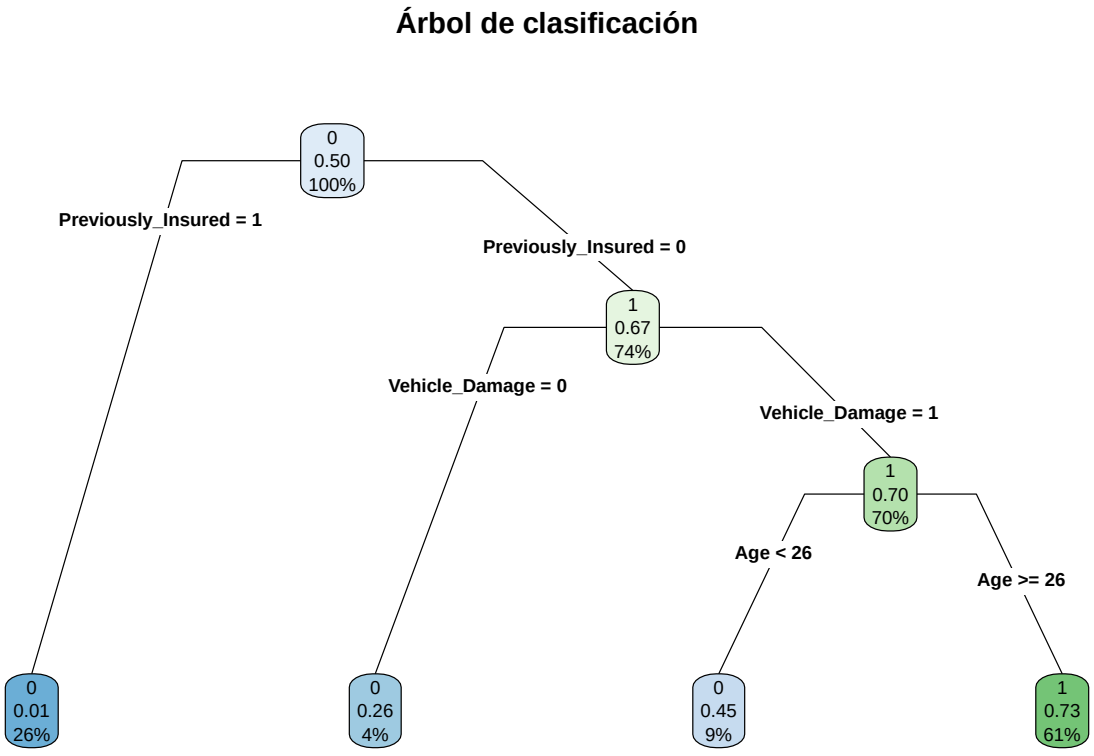
En clasificación, los métodos basados en árboles estratifican las observaciones en regiones de espacio predictor con el criterio de minimizar el error de clasificación.

### 3.3.1. Árbol de clasificación

Los árboles individuales son el método más sencillo posible dentro de este tipo de métodos. Como prerrequisito, la variable dependiente se transforma a factor en ambas muestras. La construcción de los árboles depende de tres parámetros controlables:

- `minsplit` es la cantidad mínima de observaciones que se contemplan para hacer una división en un nodo.
- `minbucket` es la cantidad mínima de observaciones que se contemplan en un nodo terminal.
- `cp` es el parámetro que marca el umbral de complejidad, es decir, la mejora mínima en el proceso de clasificación para que pueda contemplarse hacer una división adicional.

Limitado `cp` a 0.01 y `minsplit` y `minbucket` a 10 se genera un árbol y luego se evalúa la necesidad de poda.




Parece suficientemente sencillo como para no suponer un problema de sobreajuste, así que definitivamente se prescinde de la poda.

Con este árbole de clasificación se obtiene una exactitud predictiva del 70.88% sobre la muestra de validación. Véase la matriz de confusión:

Table 4: Matriz de confusión (Árbol clas.)

| | Real 0 | Real 1 |
|---|---|---|
| Pred 0 | 1374 | 28 |
| Pred 1 | 638 | 247 |

### 3.3.2. Bosques aleatorios

Se pretende obtener mejores resultados a través del método de *bosques aleatorios*.

Para aplicar el método se hace uso el método `ranger` del paquete `caret`. El funcionamiento básico consiste en combinar una serie de árboles construidos con una selección aleatoria de cierto número de predictores (`mtry`) entre los disponibles. Se registra la clase predicha por cada árbol para cada observación y se guarda la categoría más frecuente de las predichas.

El modelo se *sintoniza* de acuerdo con los siguientes parámetros:

- `min.node.size`: la cantidad mínima de observaciones que se contemplan para hacer una división en un nodo. Se estable arbitrariamente en 10.
- `num.trees`: número de árboles con que se construye el bosque. En principio se obtienen mejores resultados cuanto más elevado sea. Se considera que 4000 árboles son suficientes y que, a partir de esa cifra, las ganancias son mínimas.
- `mtry`: número de variables que entran en juego en la construcción de los árboles. Se sabe que un valor demasiado elevado en este caso produce sobrejuste en el modelo. Como se usa validación cruzada en *5-fold* se presume que los resultados no estarán sobreajustados.

Table 5: Mejor Bosque aleatorio

| | mtry | splitrule | min.node.size |
|---|---|---|---|
| 8 | 8 | gini | 10 |

Tras estimar bosques aleatorios con `mtry` entre 1 y 13, el mejor ha resultado ser el bosque aleatorio con `mtry` = 8 (bajo las condiciones de 4000 árboles, `min.node.size` de 10 y criterio de división *Gini*).

Con este método se obtiene una exactitud predictiva del 80.41% sobre la muestra de validación. Véase la matriz de confusión:

Table 6: Matriz de confusión (Bosques Aleatorios)

| | Real 0 | Real 1 |
|---|---|---|
| Pred 0 | 1721 | 157 |
| Pred 1 | 291 | 118 |

### 3.3.3. Boosting

Con la técnica de *boosting* se crean múltiples copias de remuestreo en entrenamiento mediante *bootstrap* y se estima un árbol para cada copia con la información de los previos para luego combinarlos en una única predicción.

El modelo se *sintoniza* de acuerdo con los siguientes parámetros:

- `num.trees`: número de árboles o iteraciones. En *boosting* los resultados mejoran con muchos árboles, pero un número de árboles demasiado grande puede generar sobreajuste; la validación cruzada sirve para evitar este efecto.
- `interact.depth`: profundidad máxima del árbol. En general se sabe que los modelos con más profundidad de árbol tienden a sobreajustar más. A la vez, los modelos menos profundos son computacionalmente más eficientes pero requieren un número más elevado de árboles.
- `shrinkage`: la contribución de cada árbol al resultado final o, dicho de otra manera, la velocidad a la que boosting aprende. Valores más pequeños facilitan que el algoritmo se detenga antes de sobreajustar. Toma posibles valores entre 0 y 1, pero los valores más pequeños permiten generalizar bien (aunque requieren más árboles).
- `n.minobsinnode`: número mínimo de observaciones en los nodos terminales (y la complejidad de cada árbol). Por simplicidad se establece en todo caso en 10, que es el valor por defecto.

De cara a sintonizar el modelo, para evitar dinámicas de sobreajuste, el análisis se efectúa con *5-fold CV*.

Table 7: Mejor Boosting

| | n.trees | interaction.depth | shrinkage | n.minobsinnode |
|---|---|---|---|---|
| 23 | 4000 | 18 | 0.3 | 10 |

Contemplados valores de `interaction.depth` entre 14 y 26, de `n.trees` entre 2000 y 5000 y de `shrinkage` de 0.05 a 0.5, los mejores resultados se han obtenido con 4000 árboles, `interaction.depth`de 18 y `shrinkage` de 0.3.

Con este modelo se obtiene una exactitud predictiva del 82.51%. Véase la matriz de confusión:

Table 8: Matriz de confusión (Boosting)

| | Real 0 | Real 1 |
|---|---|---|
| Pred 0 | 1784 | 172 |
| Pred 1 | 228 | 103 |

# 4. Selección de modelo y estimación sobre los datos de test

## 4.1. Selección del modelo

Recapitúlense las cifras generales de los modelos contemplados.

Table 9: Resumen de modelos

| Modelo | Accuracy (validation) |
|---|---|
| Logit | 0.7083516 |
| LDA | 0.6519458 |
| Class. Tree | 0.7087888 |
| Random Forest | 0.8041102 |
| Boosting | 0.8250984 |

*Boosting* es claramente superior al resto de modelos.

## 4.2. Integración de las muestras de entrenamiento y validación

Como la separación anterior en *training* y *validación* se efectuó antes de aplicar SMOTE, se rescata el conjunto original de los datos de entrenamiento (previo a la extracción del conjunto de validación) y se somete al mismo procedimiento. Por simplicidad se establece que el número de observaciones tras el SMOTE sea el mismo que en el conjunto de entrenamiento anterior sin la muestra de validación, pero esta vez el SMOTE habrá tenido en cuenta también los datos de validación para sintetizar las observaciones que reequilibran la muestra. Luego se comprueba el equilibrio entre las observaciones de la variable dependiente.

Table 10: Distribución de la variable dependiente (Conjuntos de entrenamiento original y validación tras SMOTE)

| Real 0 | Real 1 |
|---|---|
| 4996 | 4996 |

## 4.3. Resintonización del modelo ganador

Por último, como en este caso se dispone de los valores ciertos de la variable dependiente en la muestra de test, puede aplicarse el modelo ganador para predecir sobre este conjunto y ajustarlo para mejorar, si cabe, su exactitud predictiva.

Contemplados valores de `interaction.depth` entre 34 y 46, de `n.trees` entre 2500 y 10000 y de `shrinkage` de 0.02 a 0.5, los mejores resultados se han obtenido con 7500 árboles, `interaction.depth`de 42 y `shrinkage` de 0.05.

Table 11: Modelo ganador (sobre los datos de test)

| | n.trees | interaction.depth | shrinkage | n.minobsinnode |
|---|---|---|---|---|
| 27 | 7500 | 42 | 0.05 | 10 |

Con este modelo se obtiene una exactitud predictiva del 82.59% sobre la muestra de test, que ya es superior a los resultados obtenidos sobre la muestra de validación para este modelo. Véase la matriz de confusión:

Table 12: Matriz de confusión (Boosting, modelo ganador)

| | Real 0 | Real 1 |
|---|---|---|
| Pred 0 | 2653 | 215 |
| Pred 1 | 382 | 180 |

En cuanto a la importancia de las variables, según el modelo ganador las más importantes son `Vehicle_Damage`, `Vintage`, `Annual_Premium`, `Age` y `Previously_Insured`, tres de los cuales aparecían ya en el árbol de clasificación del apartado 3.1.

Table 13: Importancia relativa

| | Overall |
|---|---|
| Gender | 8.802068 |
| Age | 73.282920 |
| Driving_License | 0.000000 |
| Previously_Insured | 71.653151 |
| Vehicle_Damage | 100.000000 |
| Annual_Premium | 83.144022 |
| Vintage | 86.708791 |
| Vehicle_Age_plus_2Y | 2.536882 |
| Vehicle_Age_1Y_to_2Y | 3.108472 |
| Encoded_Region | 42.019065 |
| Channel_124 | 5.236226 |
| Channel_26 | 5.202984 |
| Channel_other | 16.113349 |

## 4.4. Notas finales

En una nota final, cabe señalar que podría haberse hecho una resintonización más precisa de los modelos en los puntos 3.3.3. y 4.3. pero los medios informáticos con los que se cuenta dificultan el proceso aun a pesar de haber prescindido de una gran parte de la muestra original de los datos.

Además, en una nota a parte, cabe notar que el número de falsos positivos y falsos negativos es muy alto con respecto al de verdaderos positivos cuando se aplica el modelo ganador. El modelo de clasificación de Boosting clasifica los casos lo mejor que puede según el criterio dado (*Accuracy*). Sin embargo, en la práctica podría ser más útil un modelo que sacrificara puntos de exactitud en favor de una menor tasa de falsos negativos si el objetivo de la campaña no fuera tanto clasificar correctamente el mayor número de casos como vender pólizas de seguro al mayor número de clientes posible y con el mínimo coste. En esta línea podría ser más razonable seleccionar los modelos en base a indicadores de la familia *F-Score*, que penalizan según el tipo de error.