Ecommerce: Predicción de intención de compra en linea

Loncón Joaquín

27 de mayo 2022

Contenido

Introducción **2**

Objetivo . . . 2

Entendiendo los datos . . . 2

Preparación **3**

Missing values . . . 3

Errores . . . 3

Análisis **3**

Inspección visual . . . 3

Conclusiones importantes . . . 4

Otras posibles variables a analizar . . . 4

Conceptos básicos de Machine Learning **8**

Train set y Test set . . . 8

Pre-Processing . . . 8

Cross-Validation . . . 8

Creación del modelo **9**

Problema del predominio . . . 9

Soluciones al problema del predominio . . . 9

Algoritmos de Machine Learning . . . 9

Ensemble . . . 11

Resultados **11**

Modelo Final . . . 11

Conclusiones . . . 11

Introducción

Objetivo

El objetivo de este proyecto es construir un **Modelo de Machine Learning** capaz de predecir si las personas que visitan el sitio web de nuestro negocio van a **realizar una compra o no**.

Entendiendo los datos

Nuestra base de datos contiene algunas métricas obtenidas mediante **Google Analytics** para cada página particular del sitio web.

Estas métricas son:

- **Bounce Rate**: Tasa de rebote. Representa el porcentaje de visitantes que entran al sitio web desde una página en particular y luego se van (rebotan) sin desencadenar ninguna otra solicitud al servidor durante esa sesión.
- **Exit Rate**: Tasa de salida. Representa el porcentaje de visitantes que hacen click en otro sitio desde una página determinada después de haber visitado cualquier número de páginas del sitio.
- **Page Value**: Valor de página. Es el valor medio de una página que ha visitado un usuario antes de acceder a la página objetivo o de completar una transacción de comercio (o ambas).

Sumado a las métricas de Google Analytics, se utilizan medidas que se derivan de la información de la URL de las páginas visitadas por el usuario, las cuales se actualizan en tiempo real cuando un usuario realiza una acción (como puede ser pasar de una página a otra). Estas medidas representan el número de los diferentes tipos de páginas visitadas por el usuario en esa sesión y el tiempo total invertido (medido en segundos) en cada una de estas categorías de páginas.

- **Administrative**: Número de páginas administrativas que el usuario visita
- **Administrative Duration**: Tiempo que pasa el usuario en páginas administrativas
- **Informational**: Número de páginas informativas que el usuario visita
- **Informational Duration**: Tiempo que pasa el usuario en páginas informativas
- **Product Related**: Número de páginas relacionadas al producto que el usuario visita
- **Product Related Duration**: Tiempo que pasa el usuario en páginas relacionadas al producto

Por último, el negocio toma las variables que cree necesarias y las mide de las siguiente manera:

- **Special Day**: Hace referencia a la proximidad de un día especial o festivo como puede ser el Día de la madre o Navidad
- **Operating system**: El sistema operativo que utilizó el usuario al entrar sitio web
- **Browser**: Identifica el navegador que utilizó el usuario al entrar al sitio web
- **Region**: Región desde la que se conecta el usuario al sitio
- **traffic type**: Representa como categoriza la empresa el tráfico del usuario
- **visitor type**: Informa si es la primera vez que entra el usuario al sitio (`New_Visitor`) o si ya habia accedido (`Returning_Visitor`)
- **Weekend**: Identifica si es fin de semana o no
- **Revenue**: Es nuestra variable de interés y representa si el usuario completa una compra o no

Preparación

Una vez entendidos los datos pasamos a dejar la base lista para utilizarse, esto incluye el tratamiento de *missing values* (un valor perdido, el equivalente a una celda vacia), además de limpiar la base de posibles errores y cambiar las clases de las variables según corresponda.

Missing values

creamos el objeto `online_shoppers_intention` donde queda guardada nuestra base de datos, cambiamos las clases de las variables según corresponda (por ejemplo, de `"numeric"` a `"factor"` si fuese una variable numérica que debería ser categórica). Ahora si queremos saber si hay algun *missing value* usamos la siguiente linea de código:

```
any(is.na(online_shoppers_intention))
```

```
## [1] FALSE
```

y vemos que nuestra base de datos está completa.

Errores

Sin embargo, notamos que tenemos un posible error en la variable `VisitorType`:

```
levels(online_shoppers_intention$VisitorType)
```

```
## [1] "New_Visitor"       "Other"             "Returning_Visitor"
```

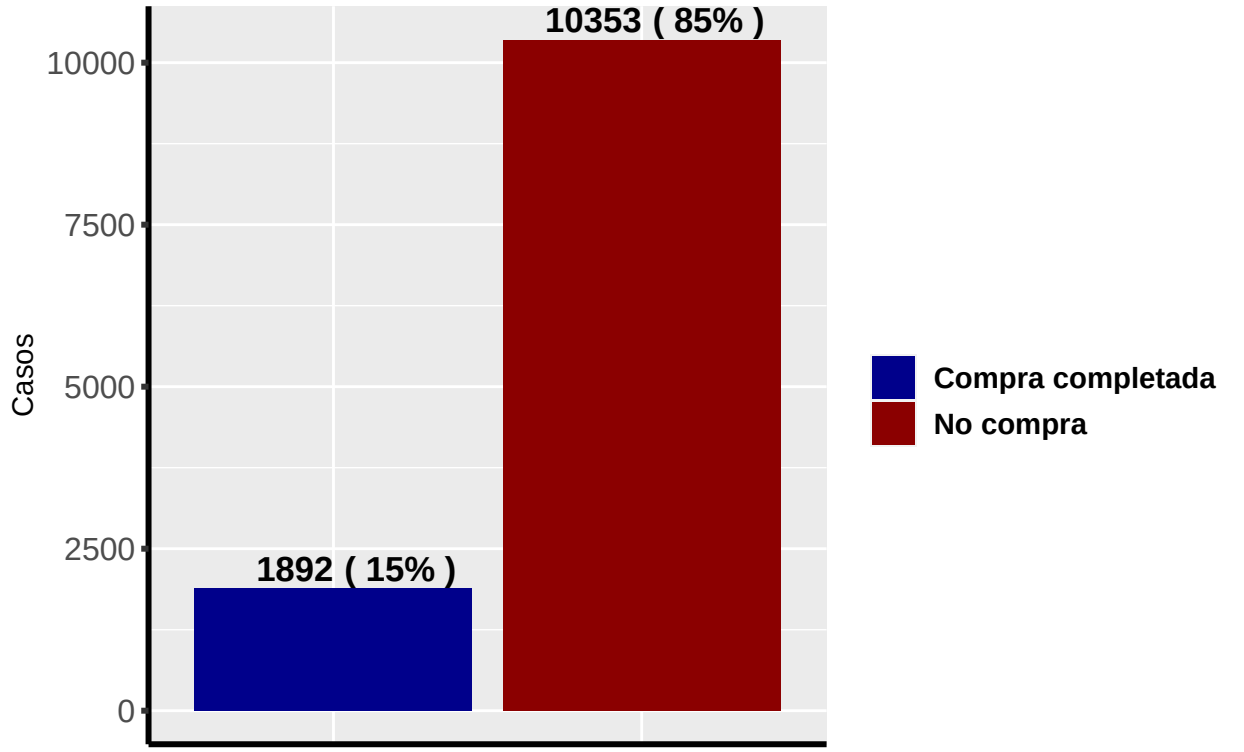
Vemos que tiene tres categorias: `New_Visitor`, `Returning_Visitor` y `Other`. Cuando sabemos por la definición previa de esta variable que deberia tener solo dos. `Other` podría ser por ejemplo, algun programador que entró a testear el sitio, o simplemente algun error que no permitió registrar de forma adecuada al usuario. Ya que son muy pocos los casos donde tenemos `Other` y no son el objetivo de este proyecto, vamos a borrar las observaciones con esta clase.

Análisis

Recordemos que nuestro objetivo era construir un **Modelo de Machine Learning** capaz de predecir si las personas que visitan el sitio web de nuestro negocio van a **realizar una compra o no**. Si bien el análisis de datos no es la parte principal del proyecto, nos sirve para entender el problema y poder abordarlo de una mejor manera.

Inspección visual

Un claro ejemplo de como el análisis nos puede ahorrar muchos problemas lo podemos ver en el siguiente gráfico:



Conclusiones importantes

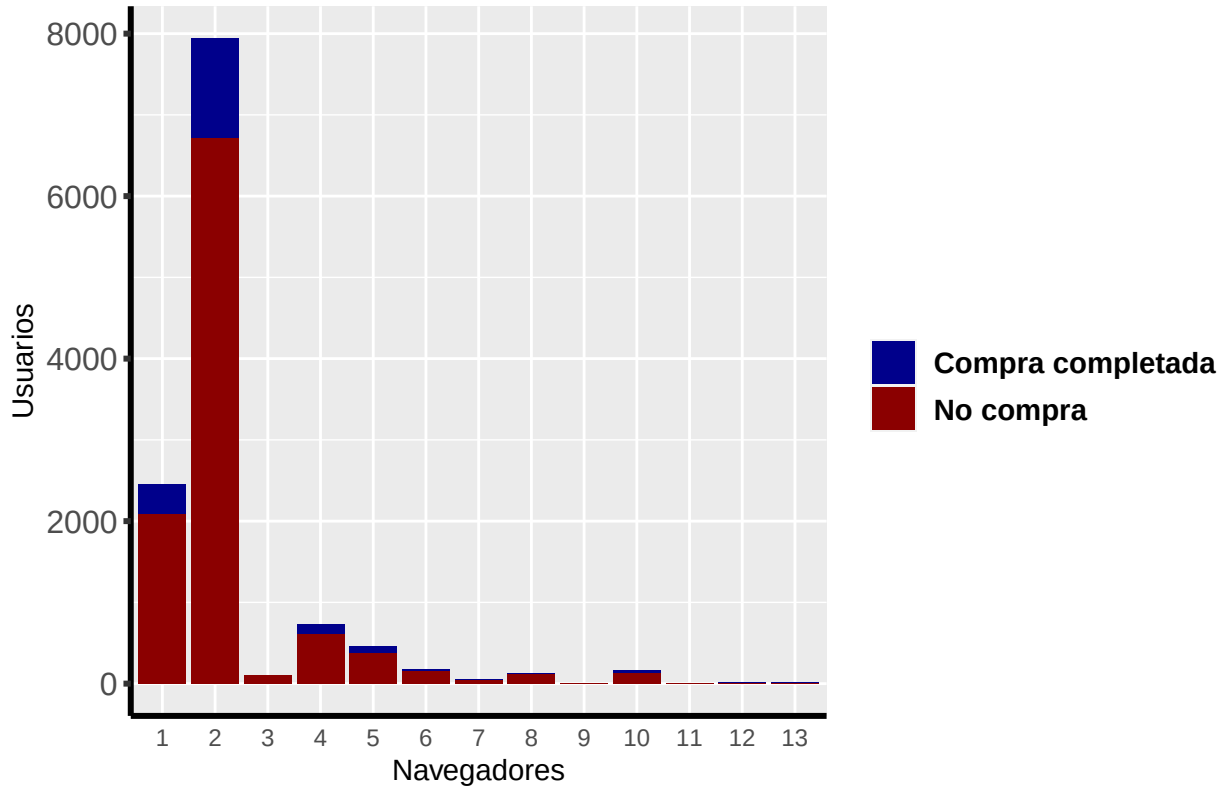
Gracias al análisis observamos que claramente hay pocos usuarios que compran (15%) en relación a los que no compran (85%). Si ignoráramos este hecho tendríamos problemas mas adelante, ya que dado el **predominio** de los usuarios que no compran, un modelo fácil pero totalmente inútil seria predecir siempre "No compra" y de esta forma tendríamos una **precisión** alta, ya que el 85% de las veces acertaríamos con nuestra predicción. La dificultad con estos casos es que con los modelos de machine learning buscamos predecir lo mejor posible, entonces el modelo ajustará a los datos entendiendo que el hecho de completar una compra es "raro" y por lo tanto buscará maximizar su capacidad predictiva prediciendo principalmente la No compra.

El tratamiento para el problema del **predominio** lo veremos al crear los modelos.

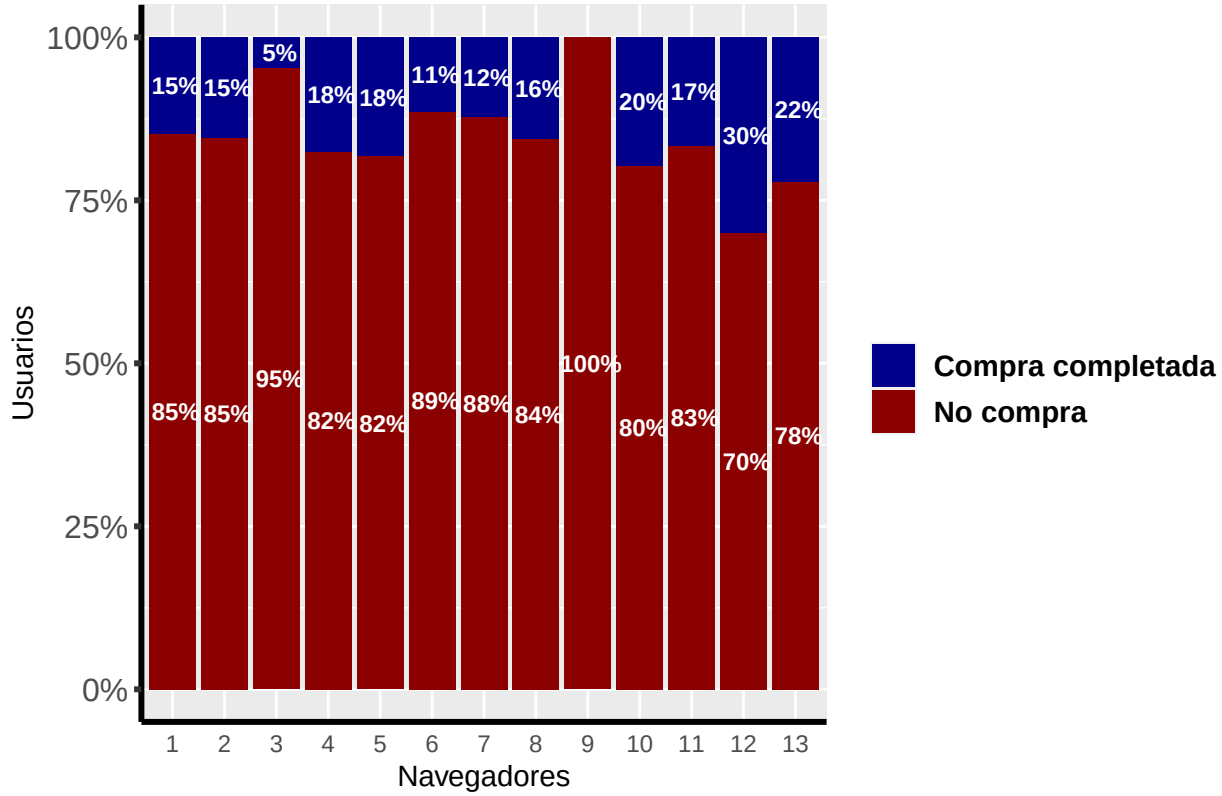
Otras posibles variables a analizar

Una pregunta válida para el análisis podría ser: ¿puede ser que según el navegador que utiliza el usuario se afecte la compra en nuestro sitio?

Esto se podría dar por distintas razones, por ejemplo, quizás algunos navegadores son más lentos y entonces los usuarios que entran a nuestro sitio a través de ellos ven mas dificultoso el proceso de comprar y deciden salir del sitio. En este caso los navegadores no llevan un nombre sino que se los identifica con un número, pasemos a ver como se distribuyen los usuarios a través de los distintos navegadores.

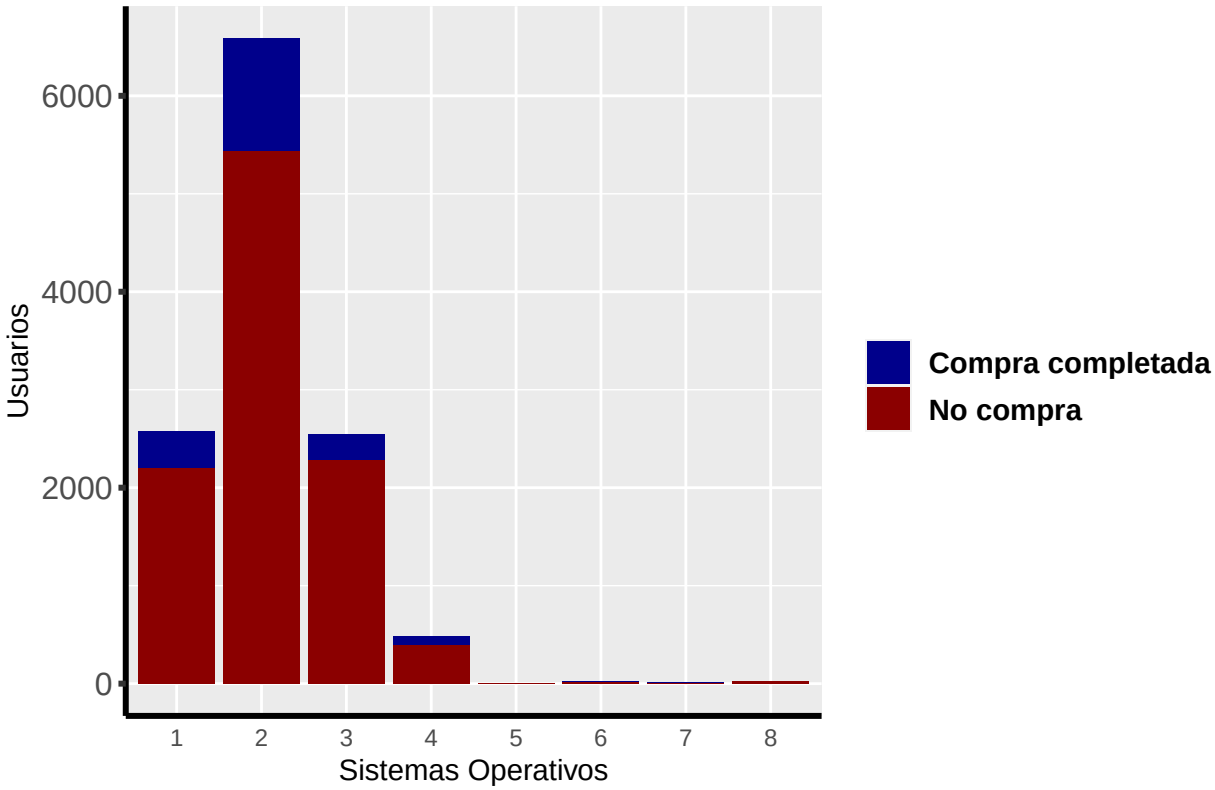


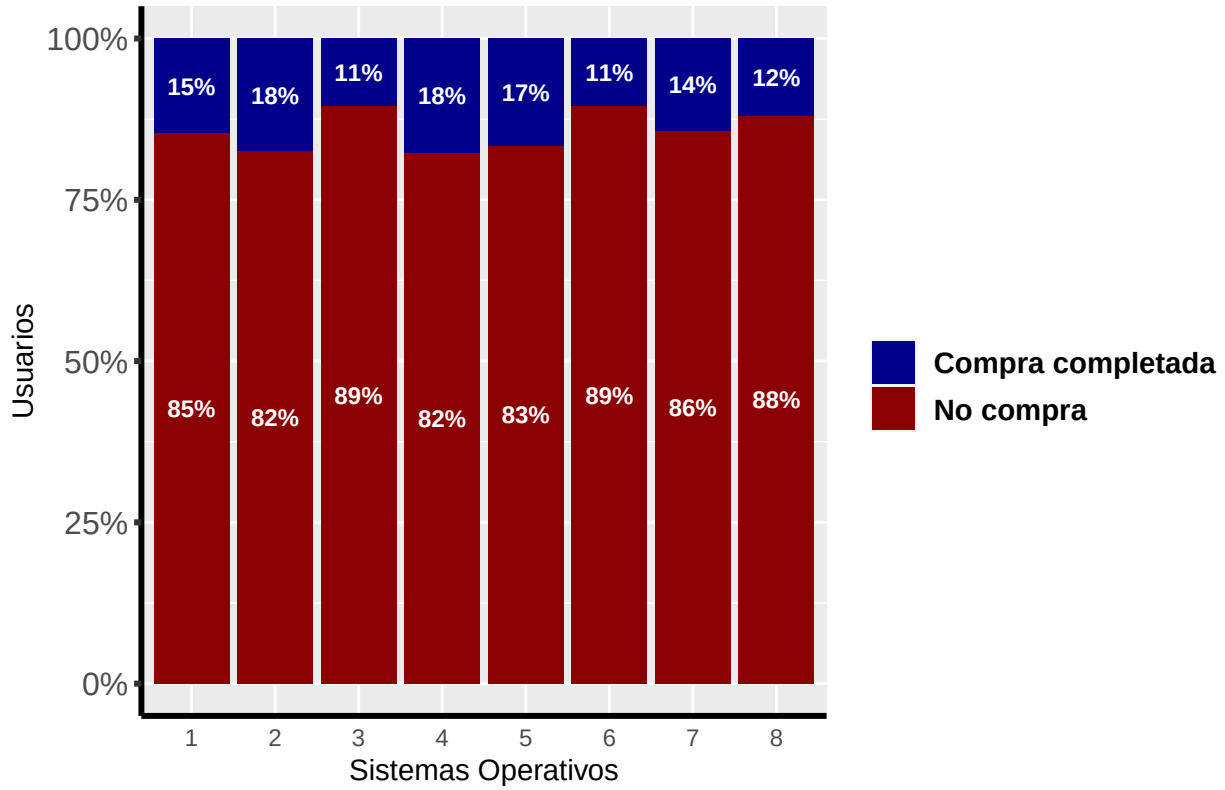
Vemos que los usuarios utilizan principalmente dos navegadores para entrar a nuestro sitio, pero para poder comparar gráficamente si esto afecta a la compra tendríamos que ver según el porcentaje de usuarios en cada navegador.



Si bien parece que existe un problema con el navegador 9, recordemos por el gráfico anterior que casi ningun usuario usa este navegador, específicamente lo utilizó un solo usuario, así que no representa un contratiempo, algo similar pasa con el navegador 3, a priori no parece existir un problema con los navegadores.

Podemos realizar el mismo análisis pero respecto a los sistemas operativos que utilizan los usuarios.


6000
4000
2000
0
Usuarios
1
2
3
4
5
6
7
8
Sistemas Operativos
Compra completada
No compra



Conceptos básicos de Machine Learning

Train set y Test set

Para poder entrenar los algoritmos de **machine learning** se dividen los datos en dos subconjuntos. El conjunto de entrenamiento (**train set**), donde se estiman los parámetros y el algoritmo "Aprende" de los datos (usualmente se utiliza 80% de los datos), y el conjunto de testeo (**test set**), donde se observa la performance del modelo con datos nuevos (el restante 20%).

Pre-Processing

En este caso, para la creación de los modelos en general se usarán distintos métodos de pre-procesamiento de los datos como son **escalar** y **centrar** (solo cuando corresponda), esto es equivalente a normalizar, es decir, restar la media (centrar) y dividir por el desvío estándar (escalar) para lograr hacer las variables comparables entre sí. También se agrega la opción de omitir las variables que tienen **varianza cercana a cero**, ya que si el predictor no varía entonces no aporta información para la creación del modelo.

Estos 3 métodos los usaremos en R mediante las opciones `scale` `center` y `nzv` respectivamente.

Cross-Validation

con el objetivo de optimizar los parámetros de los distintos modelos y garantizar que sus resultados no dependen de como se separaron los datos en un principio (en train set y test set), se utiliza el instrumento

de **Cross-Validation** o **K-Fold Cross Validation** (validación cruzada), el cual consiste en tomar el train set y dividirlo "k" veces (obteniendo k bloques/pliegues de datos), de forma de entrenar el algoritmo en k-1 de los bloques y con el bloque restante cumplir los objetivos mencionados.

Este procedimiento se repite intercambiando los bloques y dejando siempre uno distinto para testear, es decir, si `k=3` entonces el proceso se lleva a cabo 3 veces, y el test set sería el primer bloque (se entrena en 2 y 3), luego el segundo (se entrena en 1 y 3) y por último el tercero (se entrena en 1 y 2), no necesariamente en ese orden.

Creación del modelo

Una vez vistos los conceptos básicos pasamos a crear el modelo con el cual buscamos predecir si las personas que visitan el sitio web de nuestro negocio van a **realizar una compra o no**.

Problema del predominio

Como se mostró anteriormente, existe un gran predominio de una clase ("No compra"). Ahora se verá los problemas que conlleva ignorar esto.

A continuación se entrena un **arbol de decisión** en el train set. Con la parte restante (test set) **solo se observa** la performance del modelo, para ver que tan bien predice con datos nuevos.

En este caso se obtiene una **alta precisión** como esperaríamos de un modelo entrenado en datos con problemas de **predominio**, la precisión fue de un 0.8921569, es decir, el 89.22% de las veces predecimos correctamente la clase (compra completada o No compra), pero sabemos que como la mayoria son "No compra" es mas fácil predecir No compra (ver la seccion "Conclusiones importantes"). Entonces, recordando nuestro objetivo, ¿qué pasa si observamos que tan bien predecimos la compra completada?

Vemos que del total de compras completadas (omitiendo la No compra) solo pudimos predecir el 45.77% de las mismas. De hecho, si nos importara de la misma forma predecir la No compra (sobre el total de No compras) como la compra completada (sobre el total de compras completadas), en vez de utilizar la precisión podríamos usar la métrica **precisión balanceada**, la cual pondera de la misma forma ambas clases. Por lo tanto, si predecimos muy bien la clase predominante pero muy mal la clase minoritaria, nos baja la precisión. Vemos que la precisión balanceada fue del 71.46%, mucho menor que la precisión antes vista.

Soluciones al problema del predominio

Existen varias formas de tratar el problema del predominio, en nuestro caso podemos utilizar una técnica llamada **up-sampling**. Esta consiste en incrementar el tamaño de la clase minoritaria (compra completada) mediante un muestreo con reemplazo, es decir, la muestra extraída se repone cada vez para que la población sea idéntica en cada extracción, esto se realiza hasta que ambas clases tengan el mismo tamaño. Notemos que al hacer esto, las observaciones se balancean (tendremos la misma cantidad de observaciones de No compra y de Compra completada) y por lo tanto, el predominio desaparece.

Algoritmos de Machine Learning

Una vez solucionado el predominio, entrenamos diferentes algoritmos en el train set (balanceado mediante up-sampling), si bien no se ahondará en como funciona exactamente cada modelo se deja explícito el código de R para poder ver como se programan estos algoritmos con todo lo visto hasta ahora.

```
fit_rpart <- train(Revenue ~ .,
                   data = trainup,
                   method = "rpart",
                   trControl = trainControl(method = "cv", number = 10),
                   preProcess = "nzv")

fit_glm <- train(Revenue ~ .,
                 data = trainup,
                 method = "glm",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale", "nzv"))

fit_lda <- train(Revenue ~ .,
                data = trainup,
                method = "lda",
                trControl = trainControl(method = "cv", number = 10),
                preProcess = c("center", "scale", "nzv"))

fit_pam <- train(Revenue ~ .,
                 data = trainup,
                 method = "pam",
                 trControl = trainControl(method = "cv", number = 10),
                 preProcess = c("center", "scale", "nzv"))

fit_nb <- train(Revenue ~ .,
                data = trainup,
                method = "naive_bayes",
                trControl = trainControl(method = "cv", number = 10),
                preProcess = c("center", "scale", "nzv"))
```

cada algoritmo queda guardado en el objeto `fit_` seguido de su abreviación. La función `train()` del paquete `caret` nos permite entrenar los modelos bajo la siguiente sintaxis:

- `Y ~ x1 + x2 ...` : `Y` representa la variable que queremos predecir, en nuestro caso se llama **Revenue** la cual contiene dos valores `No compra` y `Compra completada`. la expresión `~` seguido de las `x` significa "dependiendo de estos predictores/variables", si se quiere usar todas las variables de la base se deja simplemente `~ .` . Por lo tanto, nuestra expresión `Revenue ~ .` significa que queremos predecir `Revenue` (compra completada y no compra) y utilizamos todas las variables de nuestra base de datos para lograrlo.
- `data = trainup`: representa en que data se va a entrenar el algoritmo, en nuestro caso `trainup` que es el train set balanceado mediante up-sampling.
- `method = "algoritmo"`: en la opción `method` especificamos que algoritmo queremos usar, en nuestro caso usamos:
  - Classification Tree (Árbol de decisión)
  - Generalized Linear Model
  - Linear Discriminant Analysis
  - Nearest Shrunken Centroids
  - Naive Bayes
- `trControl = trainControl()`: En esta sección explicitamos que instrumentos vamos a utilizar en el entrenamiento, como vimos antes utilizaremos **cross-validation** (CV) con k = 10, por lo tanto terminamos con un *10-Fold Cross Validation*.

- `preProcess = "opciones"`: se especifica que métodos se le aplicarán a los predictores (ver PreProcess).

Nota: En los algoritmos basados en árboles (tree-based) no es necesario escalar y centrar los predictores.

Ensemble

Un *Ensemble* es el concepto de combinar los resultados de los diferentes algoritmos para hacer predicciones, es decir, "juntar" muchos algoritmos en uno solo, esto generalmente mejora mucho los resultados finales. Una forma de hacer esto es tomar la **probabilidad condicional** asignada a cada clase a lo largo de todos los algoritmos y tomar un promedio, de esta forma obtendremos la **probabilidad condicional promedio** para cada observación.

Veamos como funciona la predicción de un algoritmo para enterlo mejor:

vemos que para las observaciones 1 y 2 hay mayor probabilidad de que sea `No_compra` entonces el algoritmo predice `No_compra` ya que la probabilidad es mayor al 50%. Por otro lado, para las observaciones 31 y 38 la probabilidad es mayor para `Compra_completada`, por lo tanto el algoritmo predecirá esa clase.

Como los modelos que entrenamos tienen probabilidades distintas podemos tomar un promedio y de esta forma tener el equivalente a "mas opiniones" respecto a si el usuario va a comprar o no.

Resultados

Modelo Final

Una vez creado nuestro Modelo de Machine Learning mediente un *Ensemble* de distintos algoritmos vamos a observar que tan bien predice nuestro modelo final con nuevos datos.

Obtuvimos una precisión final del 82.43%, pero como corregimos el predominio, tuvimos una precisión balanceada mayor, del 84.21% (equivalente a darle la misma importancia a predecir bien ambas clases).

Particularmente, podemos predecir correctamente cuando un usuario va a comprar en nuestro sitio el 86.77% de las veces (sobre el total real de compras) y por otro lado tambien podemos predecir cuando un usuario no realiza ninguna compra, en este caso predecimos correctamente la "No compra" el 81.64% de las veces (sobre el total de no compras).

Conclusiones

Vimos que los métodos de Machine Learning pueden ser aplicables a proyectos de Ecommerce. Con esta capacidad predictiva se podrían llevar a cabo distintas estrategias en lo que respecta a las compras online con un gran abanico de objetivos. Se podrían tomar decisiones sobre **que hacer** dado que tenemos un gran conocimiento sobre **que van a hacer** los usuarios en nuestro sitio web.

Por ejemplo, dado que tenemos gran certeza sobre la realizacion de las compras de los usuarios (86.77%), ¿Podemos mejorar la logística para realizar las entregas mas rápido o evitar congestiones en el servicio de reparto?. Este y otros problemas pueden ser solucionados mediante un enfoque orientado a los datos y la predicción.