

# Spaceship Titanic

## CDD & AA - Proyecto final Kaggle

Germán C. Paz Méndez

## 1 Descripción

Bienvenido al año 2912, donde se necesitan tus conocimientos de ciencia de datos para resolver un misterio cósmico. Hemos recibido una transmisión desde cuatro años luz de distancia y las cosas no pintan bien.

La nave espacial Titanic era un transatlántico interestelar de pasajeros botado hace un mes. Con casi 13.000 pasajeros a bordo, la nave partió en su viaje inaugural transportando emigrantes de nuestro sistema solar a tres exoplanetas recientemente habitables en órbita alrededor de estrellas cercanas.

Mientras rodeaba Alfa Centauri camino de su primer destino -el tórrido 55 Cancri E-, la incauta nave espacial Titanic colisionó con una anomalía del espacio-tiempo oculta en una nube de polvo. Por desgracia, su destino fue similar al de su homónima de 1000 años antes. Aunque la nave permaneció intacta, ¡casi la mitad de los pasajeros fueron transportados a una dimensión alternativa!

Para ayudar a los equipos de rescate y recuperar a los pasajeros perdidos, tienes el reto de predecir qué pasajeros fueron transportados por la anomalía utilizando los registros recuperados del sistema informático dañado de la nave espacial.

¡Ayuda a salvarlos y cambia la historia!

## 2 Conjunto de datos

El conjunto de datos contiene 8.693 registros de datos con 14 variables de entrada numéricas y una variable objetivo con dos valores de clase, lo que implica una clasificación binaria. **El objetivo es predecir si los pasajeros fueron transportados a otra dimensión.**

Variable	Descripción
PassengerId	Identificador único del pasajero.
HomePlanet	Planeta de origen del pasajero
CryoSleep	Indica si el pasajero ha elegido ser puesto en animación suspendida durante la duración del viaje. Los pasajeros en criosueño están confinados a sus cabinas.
Cabin	Número de cabina donde se aloja el pasajero. Tiene la forma cubierta/número/lado, donde el lado puede ser P para Puerto o S para Estribor.
Destination	El planeta al que el pasajero desembarcará.
Age	La edad del pasajero.
VIP	Si el pasajero ha pagado por un servicio VIP especial durante el viaje.
RoomService, FoodCourt, ShoppingMall, Spa, VRDeck	Cantidad que el pasajero ha facturado en cada una de las muchas comodidades de lujo del Titanic espacial.
Name	Los nombres y apellidos del pasajero.
Transported	<b>Si el pasajero fue transportado a otra dimensión.</b>

## 3 Análisis exploratorio de los datos

### 3.1 Variables que podemos descartar

Entre el conjunto de variables predictoras encontramos las variables **PassengerId** y **Name** las cuales son variables candidatas para ser descartadas ya que son identificadores únicos y no aportan valor predictivo. En principio en el preprocesamiento descartaremos la variable **Name** y más adelante veremos si podemos extraer información de la variable **PassengerId** ya que posee un formato específico que puede aportar valor predictivo.

### 3.2 Variable objetivo

Ahora vamos a analizar la variable objetivo **Transported** para ver si está balanceada o no.

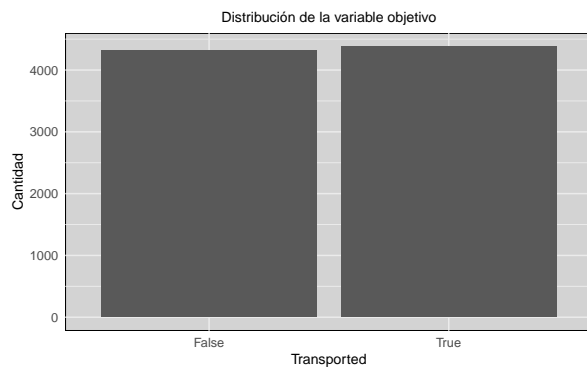


Figura 1

La variable objetivo, como podemos observar, está balanceada. En general un variable objetivo o clase balanceada permite que el modelo de aprendizaje automático se entrene de manera más efectiva, aprendiendo a reconocer patrones en todas las clases por igual. Esto conduce a un mejor rendimiento general y a una mayor fiabilidad en las predicciones del modelo, especialmente en casos de uso donde cada clase tiene importancia crítica y no se puede permitir un sesgo hacia una clase en particular.

### 3.3 Variable predictora Cabin

La variable **Cabin** tiene la forma cubierta/número/lado, donde el lado puede ser P para Puerto o S para Estribor.

Vamos a dividir esta variable en tres variables nuevas: **CabinDeck**, **CabinNumber** y **CabinSide**. Hacemos esto debido a que la variable **Cabin** tiene demasiados niveles y con ello identificadores únicos por lo que no aporta valor predictivo como tal.

La **cubierta de la cabina** puede estar relacionada con características como la clase de la cabina, la proximidad a ciertas áreas de la nave, y posiblemente la seguridad o el lujo.

El **número de cabina** es demasiado específica para tener un valor predictivo directo por lo que lo agruparemos en rangos y se podrá inferir la posición dentro de la nave.

El **lado de la cabina** podría indicar diferencias en la experiencia del pasajero o en otros factores relevantes.

### 3.4 Valores nulos (missing values)

Vamos a ver cuantos valores nulos tenemos en cada variable en nuestro conjunto de datos de entrenamiento.

##	HomePlanet	CryoSleep	Destination	Age	VIP	RoomService
##	167	169	141	148	160	149
##	FoodCourt	ShoppingMall	Spa	VRDeck	CabinDeck	CabinNumber
##	152	157	146	148	159	159
##	CabinSide	Transported				
##	159	0				

Podemos observar que todas las variables predictoras tienen valores nulos, tanto las variables categóricas como las numéricas. Hay que tener especial cuidado con las variables categóricas ya que no hay NA presentes si no cadenas vacías “”. Usaremos **recipes** para imputar los valores nulos.

Todas las variables comentadas se encuentran en el rango del 2% de valores nulos por lo que reemplazar los valores faltantes puede ser razonable, ya que es poco probable que distorsione significativamente las características generales de los datos.

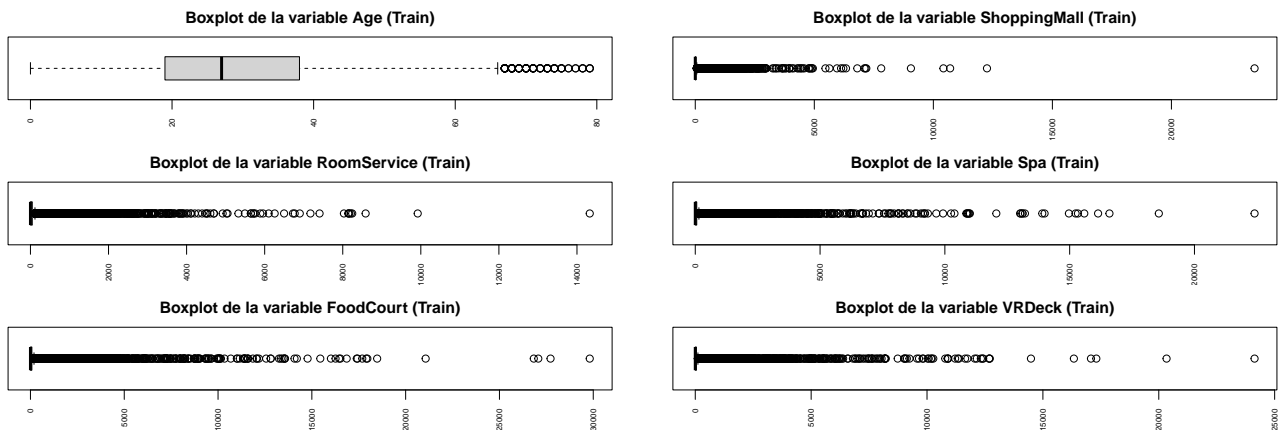
Para ello usaremos la mediana para las variables predictoras numéricas y la moda para las variables predictoras categóricas.

La mediana es una buena opción especialmente si la distribución de la variable predictora es asimétrica o si hay valores atípicos. La mediana es menos sensible a los valores atípicos que la media.

La moda es generalmente una opción segura para imputar valores faltantes en variables categóricas, ya que es el valor más frecuente y, por lo tanto, es menos probable que sesgue la distribución.

### 3.5 Valores atípicos (Outliers)

Pasemos ahora a ver los outliers o valores atípicos en el conjunto de variables del conjunto de datos de entrenamiento. Para ello usaremos el diagrama de cajas y bigotes con aquellas variables numéricas.



Podemos verificar mediante el diagrama de cajas y bigotes que todas las variables numéricas presentan valores atípicos lo que sugiere también una variabilidad considerable en los patrones de gasto. Esto indica que algunos individuos gastaron significativamente más que la mayoría.

En cuanto a los servicios, podemos además observar que la mediana se sitúa en el cero lo que indica que al menos la mitad de los pasajeros no gastaron nada en estos servicios.

En cuanto a la edad, podemos observar que la mediana indica que la población de pasajeros es relativamente joven y el tercer cuartil indica que la mayoría de pasajeros tiene menos de 40 años.

No veo necesario eliminar estos valores atípicos ya que no son valores erróneos y pueden aportar valor predictivo al modelo. Además, la eliminación de valores atípicos puede ser peligrosa ya que puede sesgar la distribución de los datos y afectar negativamente al rendimiento del modelo.

### 3.6 Distribución de las variables predictoras numéricas con respecto a la clase

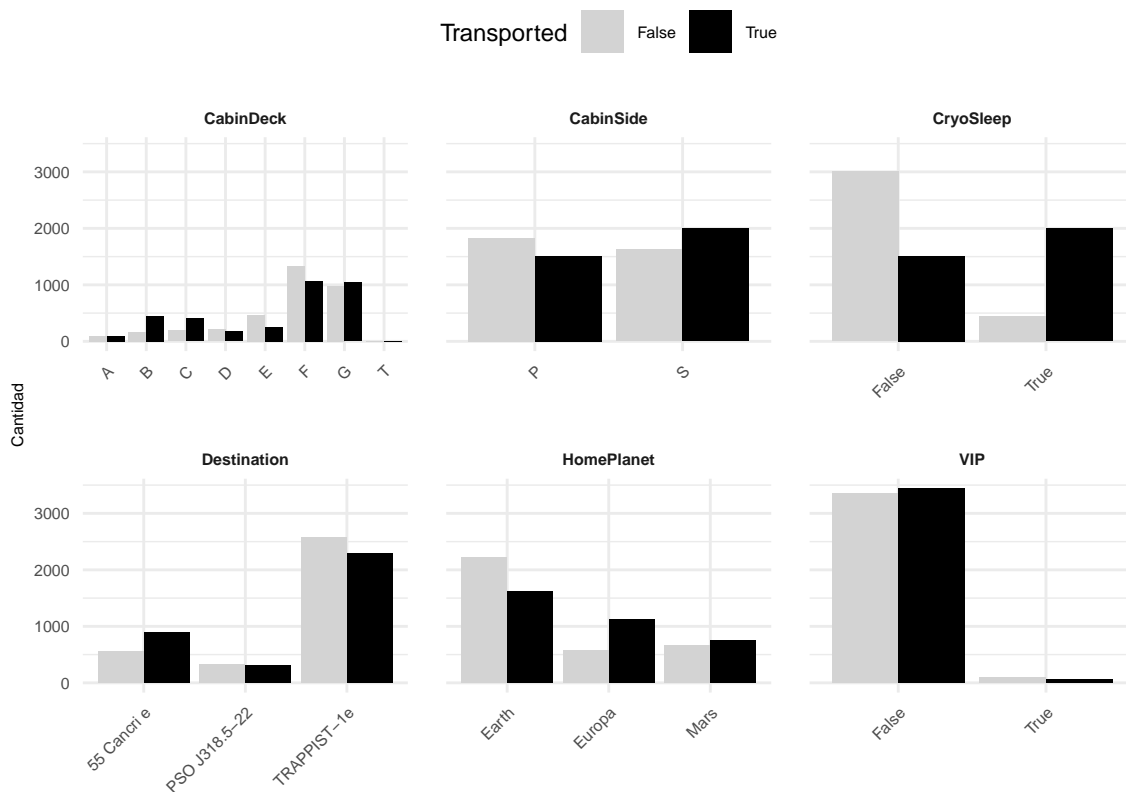


Figura 3

Viendo las gráficas de barras de las diferentes variables predictoras con respecto a la variable objetivo podemos observar lo siguiente:

- Tienes más posibilidades de ser transportado si has elegido el **criosueño** aunque la mayoría de pasajeros no eligieron esta opción.
- Tienes más posibilidades de ser transportado si te alojas en la **cubierta F** o en la **cubierta G** por cantidad aunque porcentualmente las **cubiertas B** y **C** presentan más pasajeros transportados que no transportados.
- Tienes más posibilidades de ser transportado si eliges como destino **TRAPPIST-1e** ya que es el destino más frecuente.
- Tienes más posibilidades de ser transportado si no eres **VIP** aunque la mayoría de pasajeros no son VIP.
- Tienes las mismas posibilidades de ser transportado si te alojas en el **lado P** o en el **lado S**.
- Tienes las mismas posibilidades de ser transportado si eres de **Europa**, de la **Tierra** o de **Marte** aunque el origen más frecuente es la **Tierra** y el menos frecuente **Marte**.

### 3.7 Correlación entre las variables predictoras

Dado que nuestro conjunto de variables predictoras está compuesto por variables numéricas y categóricas, vamos a convertir los datos categóricos en valores numéricos usando factorización. Este método es una aproximación y proporciona una medida de correlación que no es la correlación de Pearson estándar, sino una medida de asociación que puede ser útil para explorar relaciones en datos mixtos (categóricos y numéricos).

Del gráfico de correlaciones podemos obtener las siguientes Conclusiones:

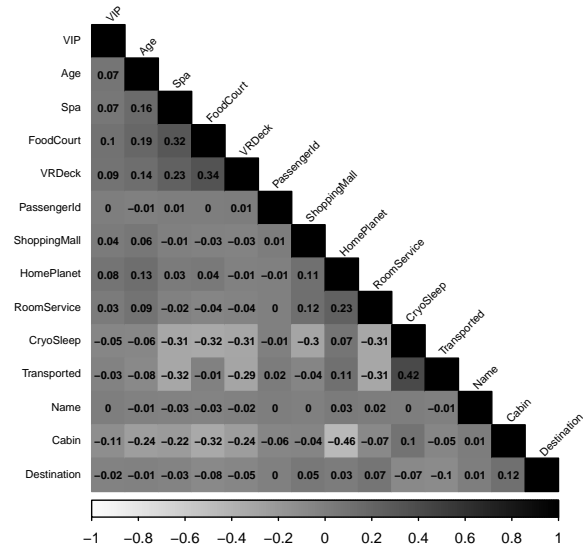
- La variable objetivo **Transported** parece tener una correlación positiva moderada con **CryoSleep** (0.46), lo que sugiere que hay una tendencia de que los pasajeros en criosueño tengan más probabilidades de ser transportados como ya hemos comentado anteriormente.

- La variable **CabinDeck** muestra una correlación negativa significativa con **Transported** (-0.41), lo que podría indicar que ciertos niveles de cubierta están asociados con una menor probabilidad de ser transportados como también hemos podido apreciar anteriormente.

- La variable **Age** tiene una correlación positiva ligera con **RoomService** y **Spa** (0.13 y 0.15, respectivamente), lo que puede implicar que los pasajeros de mayor edad tienden a gastar más en estos servicios.

- La variable **CabinNumber** tiene una correlación fuerte con **CabinSide** (0.52). Esto podría ser el resultado de cómo se asignan los números de las cabinas y su ubicación dependiendo del lado de la cabina.

- La variable **VRDeck** tiene correlaciones moderadas con **FoodCourt** y **Spa** (0.31 y 0.34, respectivamente), lo que podría reflejar un patrón de gasto donde aquellos que gastan en una categoría también gastan en otras.



### 3.8 Variable predictora PassengerId

La variable **PassengerId** es un identificador único para cada pasajero y no aporta valor predictivo tal y como está. Por ello vamos a procesarla para ver si podemos extraer información y valor predictivo de ella.

Cada identificador tiene la forma **gggg\_pp**, donde **gggg** indica el grupo con el que viaja el pasajero y **pp** es su número dentro del grupo. Los miembros de un grupo suelen ser familiares, pero no siempre.

Creamos la variable **group\_size** con el número de personas pertenecientes al grupo a partir de **PassengerId**. Luego eliminaremos la variable **PassengerId** ya que no aporta valor predictivo tal y como está.

## 4 Preprocesamiento usando recipes

El uso de **recipes** o recetas nos permite crear un flujo de trabajo de preprocesamiento que se puede aplicar a los datos de entrenamiento, de validación y de prueba. Esto es importante porque los datos de prueba no deben influir en el preprocesamiento de los datos de entrenamiento. Básicamente una receta es una lista de pasos que se pueden aplicar a los datos. Cada paso puede ser una transformación, una selección de variables, una ingeniería de características o una combinación de estas.

Podemos ver algunos uso básicos en el siguiente sitio web: <https://recipes.tidymodels.org/articles/recipes.html>.

Muchas veces nos equivocamos al aplicar los pasos de preprocesamiento a los datos de validación y de prueba. Por ejemplo, si calculamos la media de una variable numérica en los datos de entrenamiento y luego la aplicamos a los datos de validación y de prueba, esto puede introducir un sesgo en los datos de validación y de prueba. Esto se debe a que los datos de validación y de prueba no deben influir en el preprocesamiento de los datos de entrenamiento. Por lo tanto, es importante que los pasos de preprocesamiento se apliquen solo a los datos de entrenamiento y luego se apliquen a los datos de validación y de prueba. Esto se puede hacer fácilmente con **recipes**.

## 5 Modelos

A continuación describiré los modelos habituales que se usan para este tipo de problemas de clasificación binaria. Estos cinco modelos ofrecen una buena mezcla de simplicidad, capacidad de manejo de datos complejos y rendimiento:

1. **Regresión Logística:** Aunque es un modelo más simple, es un buen punto de partida para cualquier problema de clasificación binaria. Es rápido, fácil de interpretar y te puede dar una línea de base para el rendimiento que puedes esperar.
2. **Random Forest:** Este modelo es un ensamble que puede manejar tanto características numéricas como categóricas y es capaz de modelar interacciones complejas entre características. Es menos probable que se sobreajuste en comparación con un solo árbol de decisión.
3. **XGBoost (Extreme Gradient Boosting):** Este es un modelo de ensamble que ha ganado muchas competiciones de Kaggle debido a su rendimiento y velocidad. Es eficaz para manejar una variedad de tipos de datos y puede ser ajustado para mejorar aún más su rendimiento.
4. **Support Vector Machines (SVM):** Con un kernel adecuado, SVM puede ser muy efectivo en espacios de alta dimensión y es particularmente bueno cuando hay un margen claro de separación entre las clases.
5. **Redes Neuronales Artificiales (ANN):** Las redes neuronales proporcionan una gran flexibilidad y capacidad para modelar relaciones no lineales y complejas. Con la cantidad adecuada de datos y la configuración correcta, pueden superar a los modelos más tradicionales.

De entre ellos elegí que mejor rendimiento me ofreció.

### 5.1 Comparativa

##	models	accuracies	kappas
## 1	Logistic Regression	0.7980437	0.5958338
## 2	Random Forest	0.7865362	0.5734148
## 3	XGBoost	0.7951669	0.5903012
## 4	SVM	0.7940161	0.5879013
## 5	ANN	0.8032221	0.6062815

El modelo ANN es el modelo que a primera vista ha obtenido un rendimiento mejor. Sería interesante ponerlo a punto para ver si puede mejorar su rendimiento pero en este caso vamos a usar **XGBoost** ya que creemos que nos dará un buen rendimiento en este problema. A continuación vamos a ajustar sus hiperparámetros para ver si podemos mejorar su rendimiento. Además habrá que hacer un preprocesamiento con las variables categóricas para que el modelo pueda manejarlas.

También es importante señalar el tema del tiempo de cómputo que ha necesitado cada modelo. Conforme nos adentramos en intentar obtener el mejor rendimiento que el modelo puede ofrecernos el tiempo de cómputo podría aumentar considerablemente. Estos tiempo son una perspectiva inicial de lo que puede suponer el entrenamiento básico de cada modelo.

la hora de ir probando los hiperparámetros de un modelo, el tiempo de cómputo puede aumentar considerablemente. Por ello, es importante tener en cuenta los hiperparámetros que se modifican y la diferencia de tiempo de cómputo que supone.

Modelo	Tiempo de cómputo
Regresión Logística	1 segundo
Random Forest	27 segundos
XGBoost	20 segundos
SVM	19 segundos
ANN	5 segundos

A

### 5.2 XGBoost

Sabemos que **XGBoost**, como la mayoría de los algoritmos basados en árboles de decisión, maneja eficientemente variables numéricas. Sin embargo, no maneja directamente variables categóricas sin ser codificadas.

Entre las opciones habituales de codificación de variables categóricas se encuentran:

- **One-Hot Encoding:** Esta es una codificación simple que crea una columna separada para cada valor de la variable categórica. Por ejemplo, si la variable categórica es `HomePlanet` y tiene tres valores posibles: `Tierra`, `Marte` y `Europa`, entonces se crearán tres columnas: `HomePlanet_Earth`, `HomePlanet_Mars` y `HomePlanet_Europa`. Si un pasajero es de la `Tierra`, entonces `HomePlanet_Earth` será 1 y las otras dos columnas serán 0. Esta codificación es simple y funciona bien para variables categóricas con pocos valores únicos. Sin embargo, puede ser ineficiente para variables categóricas con muchos valores únicos.

- **Binary Encoding:** Esta codificación es similar a la codificación One-Hot, pero en lugar de crear una columna separada para cada valor de la variable categórica, crea una columna para cada bit de la codificación binaria de los valores. Por ejemplo, si la variable categórica es `HomePlanet` y tiene tres valores posibles: `Tierra`, `Marte` y `Europa`, entonces se crearán dos columnas: `HomePlanet_0` y `HomePlanet_1`. Si un pasajero es de la `Tierra`, entonces `HomePlanet_0` será 0 y `HomePlanet_1` será 0. Esta codificación es más eficiente que la codificación One-Hot, pero puede ser ineficiente para variables categóricas con muchos valores únicos.

- **Target Encoding:** Esta codificación reemplaza cada valor de la variable categórica con la media de la variable objetivo para ese valor. Por ejemplo, si la variable categórica es `HomePlanet` y tiene tres valores posibles: `Tierra`, `Marte` y `Europa`, entonces se reemplazará cada valor de `Tierra` con la media de la variable objetivo para los pasajeros de la `Tierra`, cada valor de `Marte` con la media de la variable objetivo para los pasajeros de `Marte`, y cada valor de `Europa` con la media de la variable objetivo para los pasajeros de `Europa`. Esta codificación es más eficiente que la codificación One-Hot y la codificación binaria, pero puede ser ineficiente para variables categóricas con muchos valores únicos.

Hay otras codificaciones que podrían ser más eficientes para variables categóricas pero estas tres son las más comunes y son un buen punto de partida.

Para codificar las variables categoricas haremos uso de `dummyVars` incluida en la librería de `caret`. `dummyVars` es una función que crea una fórmula para codificar variables categóricas. Luego usamos `predict` para codificar las variables categóricas en los conjuntos de datos de entrenamiento y test.

### 5.3 Exploración de los hiperparámetros de XGBoost

Una forma de optimizar el modelo `XGBoost` es ajustar sus hiperparámetros. Los hiperparámetros son parámetros que no se aprenden durante el entrenamiento del modelo, sino que se establecen antes de comenzar el entrenamiento.

Vamos a ver algunas configuraciones generales de hiperparámetros en `XGBoost` que suelen ser efectivas para mejorar el rendimiento:

Hiperparámetro	Definición	Posibles valores
nrounds	Número de árboles a construir. Valores más altos pueden mejorar el rendimiento, pero también aumentan el riesgo de sobreajuste	Un rango típico podría ser entre 100 y 1000.
eta	Tasa de aprendizaje que controla el peso de las nuevas árboles añadidas al modelo.	Valores más bajos, como 0.01 a 0.1, pueden ser más eficaces, especialmente en combinación con un mayor número de árboles.
max_depth	Profundidad máxima de cada árbol.	Valores entre 3 y 10 suelen ser eficientes, pero depende del tamaño y la dimensionalidad de tus datos.
subsample	Porcentaje de muestras utilizadas para entrenar cada árbol.	Valores alrededor de 0.8 a 1 pueden ser efectivos.
colsample_bytree	Porcentaje de características utilizadas para entrenar cada árbol.	Valores comunes están entre 0.3 y 0.8.
gamma	Mínima pérdida necesaria para realizar una partición adicional en un nodo del árbol.	Valores mayores hacen al modelo más conservador.

Hiperparámetro	Definición	Posibles valores
min_child_weight	Mínima suma de pesos de todas las observaciones requeridas en un niño.	Un valor más grande puede evitar el sobreajuste en datasets con desequilibrio de clases.
lambda y alpha	Parámetros de regularización L2 y L1, respectivamente, que pueden ayudar a prevenir el sobreajuste.	

Después he probado diferentes técnicas de búsqueda de hiperparámetros como **Grid Search** y **Random Search**. Al final he obtenido por **Random Search** ya que **Grid Search** es muy costoso computacionalmente y **Random Search** suele obtener resultados similares en menos tiempo.

```
# XGBoost usando Grid Search
tc <- trainControl(method = "cv",
  number = 10)

grid <- expand.grid(
  nrounds = c(100, 130, 150),
  max_depth = c(3, 4, 5),
  gamma = c(0.5, 1, 1.5),
  colsample_bytree = c(0.6, 0.8, 1),
  subsample = c(0.6, 0.8, 1)
)

t <- Sys.time()
set.seed(1234)
modelXgboost_tuned <- train(
  Transported ~ .,
  data = train_data,
  method = "xgbTree",
  trControl = tc,
  tuneGrid = grid
)
Sys.time() - t

# XGBoost usando Random Search
tc <- trainControl(method = "cv",
  number = 10, search = "random")

random_grid <- list(
  nrounds = c(100, 130, 150),
  max_depth = c(3, 4, 5),
  gamma = c(0.5, 1, 1.5),
  colsample_bytree = c(0.6, 0.8, 1),
  subsample = c(0.6, 0.8, 1)
)

t <- Sys.time()
set.seed(1234)
modelXgboost_tuned <- train(
  Transported ~ .,
  data = train_data,
  method = "xgbTree",
  trControl = tc,
  tuneLength = 10
)
Sys.time() - t
```

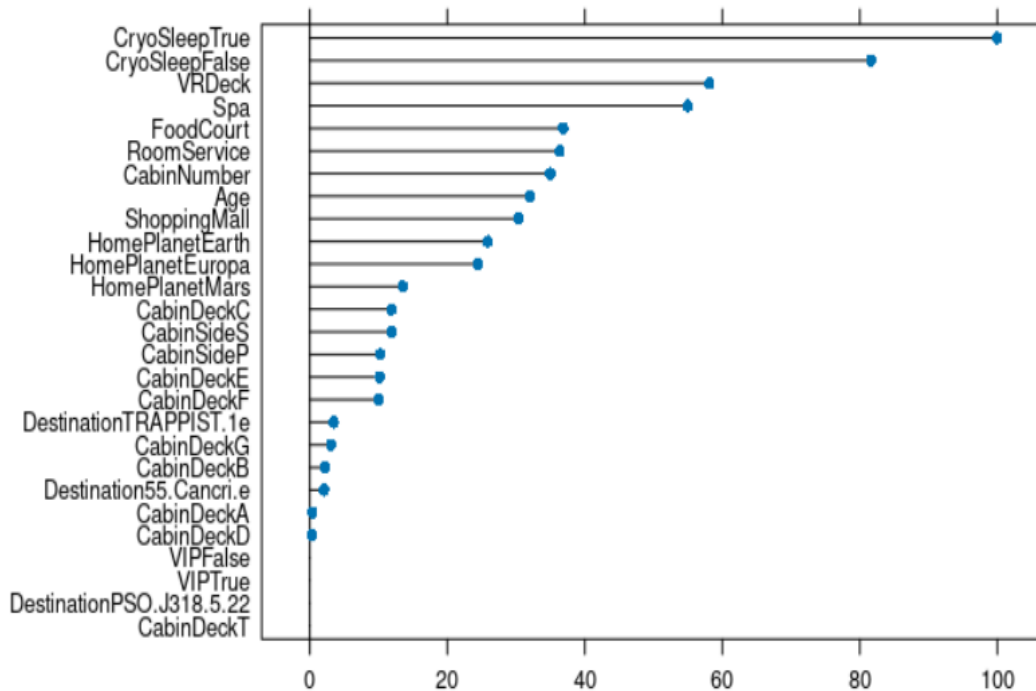
Encontrar los mejores hiperparámetros conlleva prueba y error. No hay una configuración de hiperparámetros que funcione mejor para todos los conjuntos de datos. Por lo tanto, es importante probar diferentes configuraciones y ver cuál funciona mejor.

## 6 Importancia del conjunto de variables en XGBoost (Feature selection)

Una vez que hemos ajustado los hiperparámetros, podemos ver la importancia de las variables en el modelo XGBoost. La importancia de las variables es una medida de cuánto contribuye cada variable a la precisión del modelo. Una variable con una importancia alta contribuye más a la precisión del modelo que una variable con una importancia baja.

En **caret** podemos hacer uso de la función **varImp()** para obtener la importancia de las variables en el modelo XGBoost.





En este caso podríamos eliminar las variables con una importancia menor a 15% y tener un modelo más simple con un rendimiento similar.

## 7 Conclusiones

En este proyecto he realizado un análisis exploratorio de los datos, he realizado un preprocesamiento de los datos, he comparado varios modelos, he seleccionado el modelo que mejor rendimiento ha obtenido y he ajustado sus hiperparámetros para obtener un mejor rendimiento.

Personalmente me ha sorprendido el rendimiento de **XGBoost** ya que es un modelo que no había usado anteriormente y he podido comprobar que es un modelo muy potente y que ofrece un rendimiento muy bueno.

También he podido comprobar que el preprocesamiento de los datos es una parte muy importante del proyecto ya que puede mejorar el rendimiento de los modelos considerablemente.

Por último, he podido comprobar que el tiempo de cómputo es un factor muy importante a tener en cuenta ya que conforme nos adentramos en intentar obtener el mejor rendimiento que el modelo puede ofrecernos el tiempo de cómputo puede aumentar considerablemente. Por ello es importante tener en cuenta el tiempo de cómputo que puede suponer el entrenamiento de cada modelo.


En la entrevista se responderán todas las dudas y les explicaré el código R usado en el proyecto.

## 8 Github

El código R usado en el proyecto se encuentra disponible en el siguiente repositorio de Github: [https://github.com/gcpmendez/CDD\\_R\\_kaggle\\_uimp](https://github.com/gcpmendez/CDD_R_kaggle_uimp)


## 9 Resultados en Kaggle

YOUR RECENT SUBMISSION

 **submission.csv** Score: 0.79962


Submitted by Germán C. Paz Méndez · Submitted 20 seconds ago

↓ Jump to your leaderboard position



**Germán C. Paz Méndez**

HPC engineer at ITER S.A.  
Santa Cruz de Tenerife, Canary Islands, Spain  
Joined a month ago · last seen in the past day




Competitions  
Novice

Home Competitions (1) Datasets Code Discussion Followers Notifications Account Edit Public Profile

Active Completed Hosted Community Bookmarks

Default ▾



**Spaceship Titanic**

Predict which passengers are transported to an alternate dimension  
Getting Started · 2891 Teams · Ongoing

946/2891

