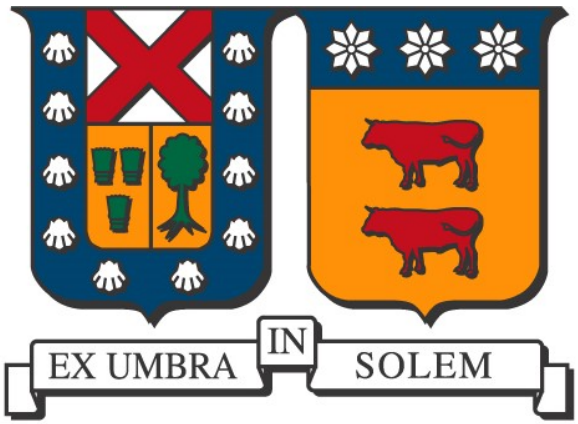
Spaceship Titanic

Proyecto final - MAT281

Lucas Daniel Vargas Arroyo

Departamento de matemáticas
Universidad Técnica Federico Santa María

04 de Diciembre de 2023



Definición del problema

Contexto

Nos situamos en 2912 donde la nave espacial Titanic fue lanzada hace un mes y al sufrir un accidente, alrededor de la mitad de los pasajeros fueron transportados a una dimension alternativa.

Problema

Se nos pide predecir el destino de los pasajeros mediante Machine Learning, para lo cual es necesario analizar y procesar los datos para aplicar algún modelo.

Dataset

Conjunto de datos

Contamos con 2 conjuntos de datos, el conjunto de entrenamiento (train df) y el conjunto de prueba (test df), que tienen como unica diferencia la presencia de la columna "Transported", la cual nos dice si el pasajero fue teletransportado o no.

Atributos (Columnas)

Ambos conjuntos poseen los atributos:

Atributos categóricos

"PassengerId", "HomePlanet", "CryoSleep", "Cabin", "Destination", "VIP" y "Name".

Atributos continuos

"Age", "RoomService", "FoodCourt", "ShoppingMall", "Spa" y "VRDeck".

Estadística descriptiva

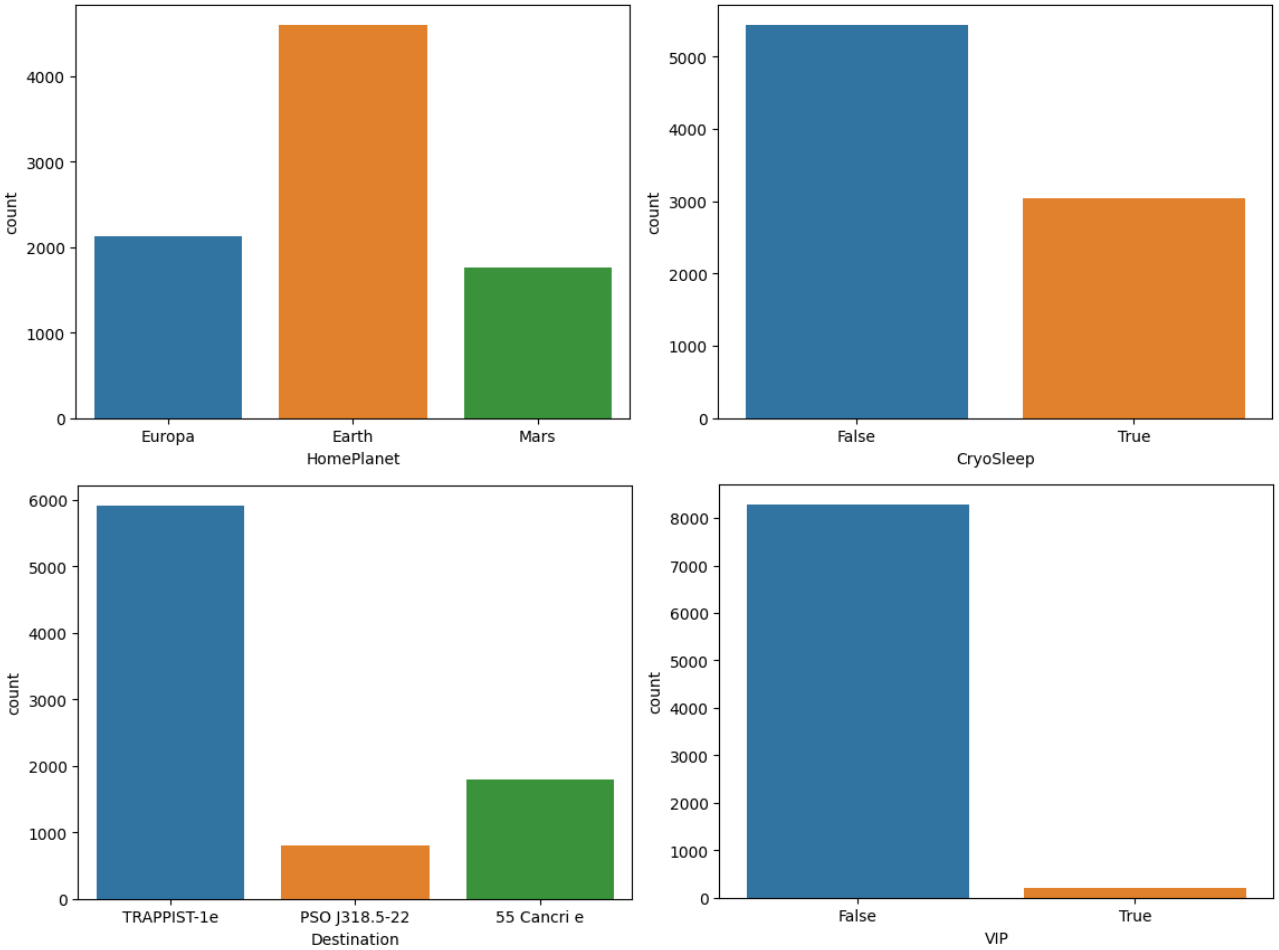
train df:

	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	8514.000000	8512.000000	8510.000000	8485.000000	8510.000000	8505.000000
mean	28.827930	224.687617	458.077203	173.729169	311.138778	304.854791
std	14.489021	666.717663	1611.489240	604.696458	1136.705535	1145.717189
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	27.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	38.000000	47.000000	76.000000	27.000000	59.000000	46.000000
max	79.000000	14327.000000	29813.000000	23492.000000	22408.000000	24133.000000

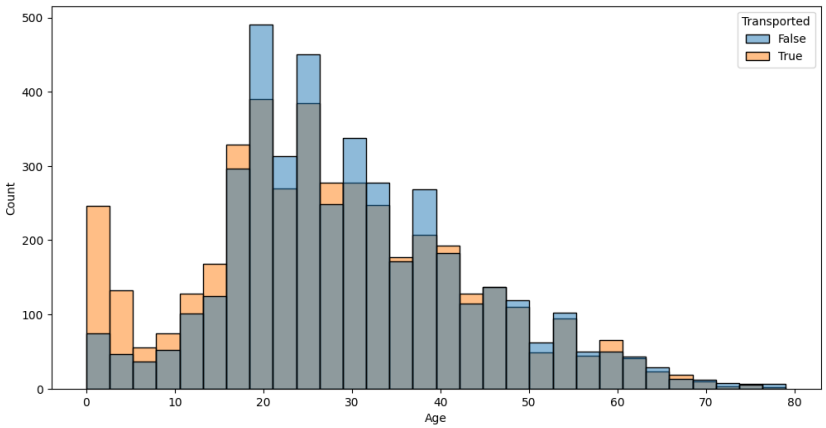
test df:

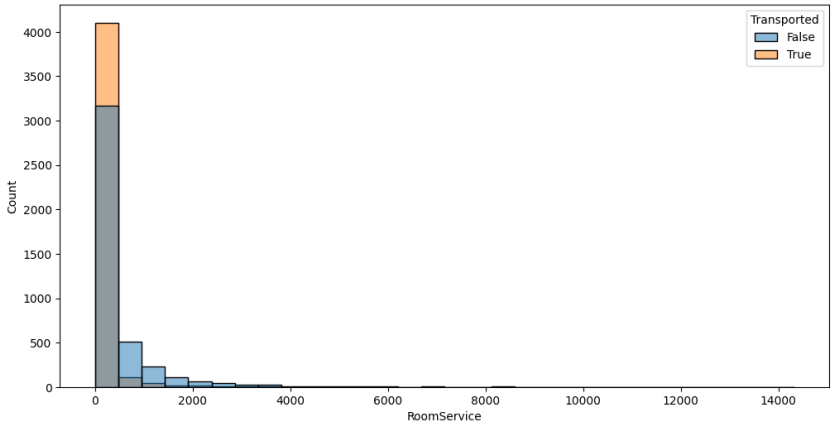
	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	4186.000000	4195.000000	4171.000000	4179.000000	4176.000000	4197.000000
mean	28.658146	219.266269	439.484296	177.295525	303.052443	310.710031
std	14.179072	607.011289	1527.663045	560.821123	1117.186015	1246.994742
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	26.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	37.000000	53.000000	78.000000	33.000000	50.000000	36.000000
max	79.000000	11567.000000	25273.000000	8292.000000	19844.000000	22272.000000

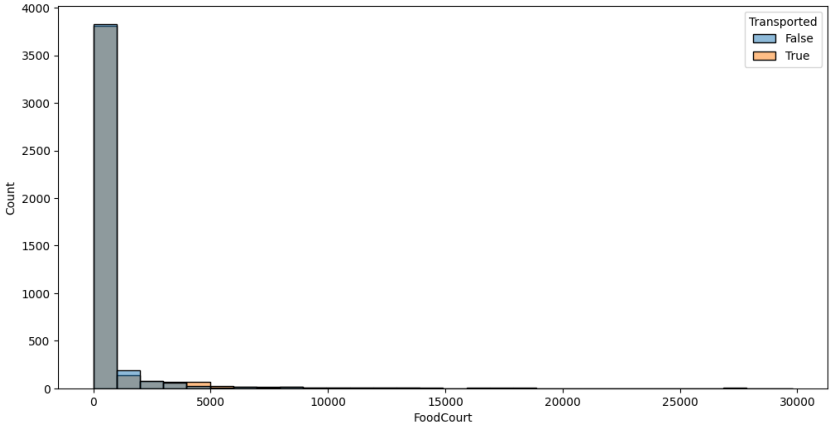
HomePlanet, CryoSleep, Destination, VIP



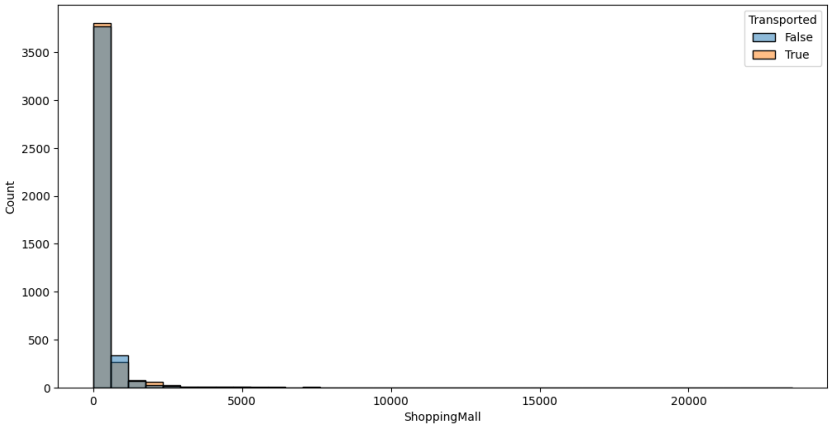
Age/RoomService/FoodCourt vs Transported

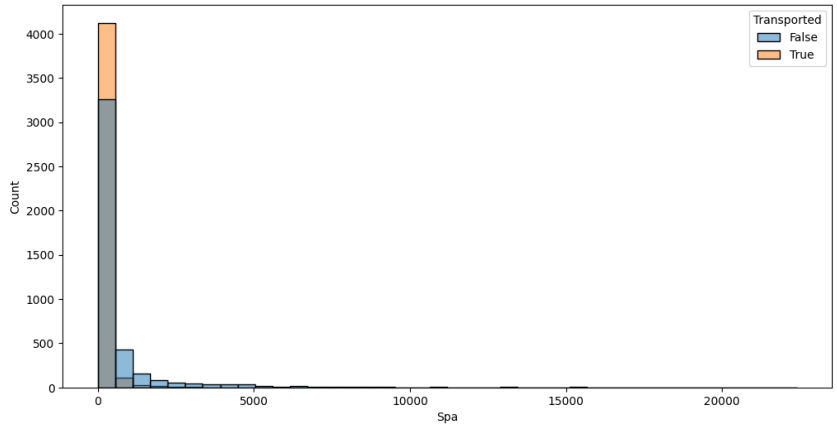


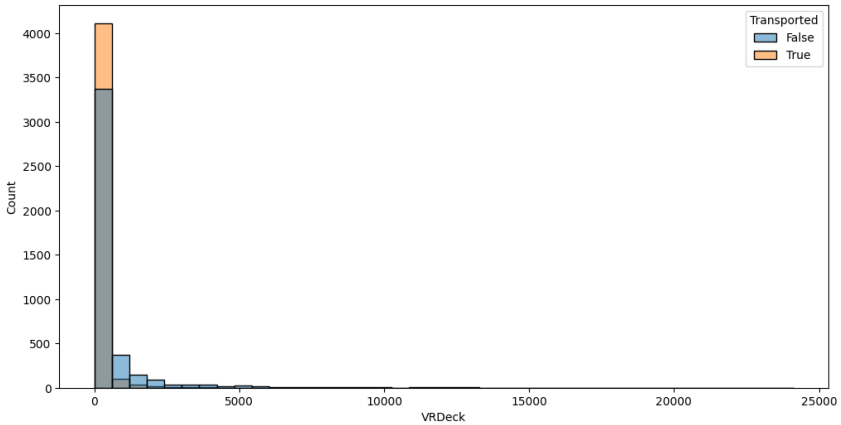




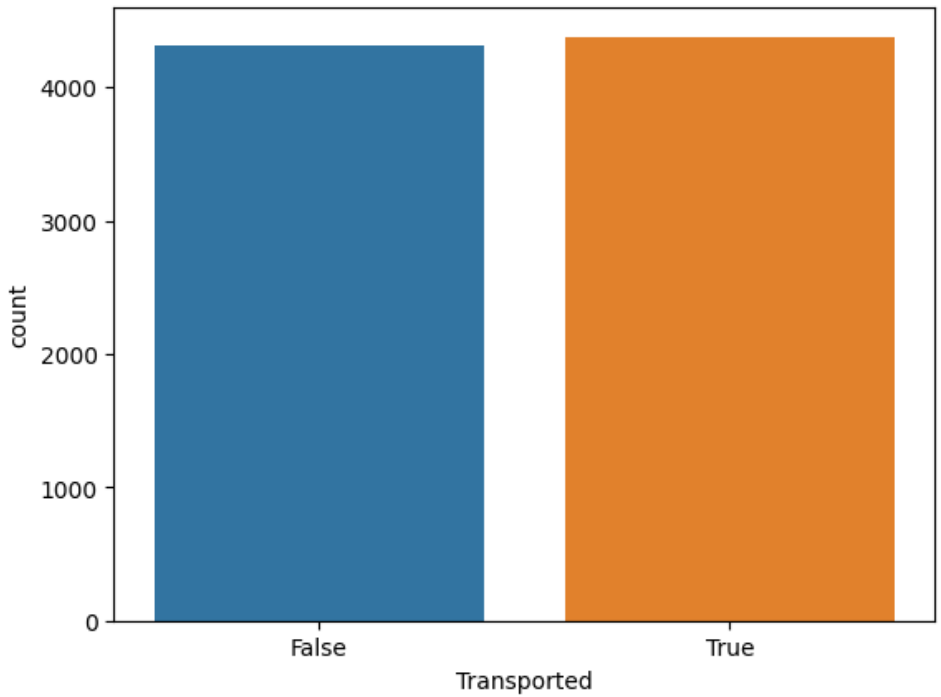
ShoppingMall/Spa/VRDeck vs Transported







Transported



Limpieza de Datos

Para comenzar el preprocesamiento se empezó limpiando los datos, haciendo lo siguiente:

- Dividir el conjunto entre datos de entrada y salida (solo para *train df*).
- Se revisan los datos que presentan NaN.
- Atributos categóricos con NaN: Se revisa la cantidad de clases por atributo, se sacan atributos que no aporten y en los demás se reemplazan los valores NaN por la moda de la columna.
- Atributos continuos con NaN: Se reemplaza el valor por la media de la columna.

División de conjuntos

Se dividen train/test en:

- *train cat/test cat*: se agrupan atributos categoricos que deben pasarse por un encoder para poder ser utilizados en un modelo.
- *train cont/ test cont*: se agrupan atributos continuos que deben pasarse por un scaler para poder ser utilizados en un modelo.
- En otros conjuntos particulares se guardaron los atributos "VIP" y "CryoSleep".

Encoders y Scalers

- Encoder: One Hot Encoder, este encoder sirve para atributos categóricos que no tengan un orden entre ellos a costa de aumentar la dimensionalidad del problema.
- Scaler: Standard Scaler, este scaler sirve para eliminar la media y cambiar la varianza a 1, con lo que no habrán problemas para entrenar un modelo.

Modelos a entrenar

En este proyecto se usaron 3 modelos de la librería *sklearn* de Python y una red neuronal a partir de la librería tensorflow:

- Support Vector Classifier (SVC)
- Decision Tree Classifier
- Random Forest Classifier
- Red Neuronal Densa

Optimización de Hiperparámetros y entrenamiento del modelo: Preliminares

Se ejecuto la función *train test split* de *sklearn* para así entrenar y probar el modelo con el conjunto *train df*.

Optimización de Hiperparámetros y entrenamiento del modelo: SVC

Para el modelo SVC se probaron los siguientes valores:

- C: 0.1, 1, 10.
- *kernel*: linear, rbf.

Resultando como mejores parámetros $C = 10$ y *kernel* = 'rbf'.

Optimización de Hiperparámetros y entrenamiento del modelo: Decision Tree Classifier

Para el modelo Decision Tree Classifier se probaron los siguientes valores:

- *max depth*: 1, 5, 10, 15, 20, 25, 30.
- *min samples split*: 20, 50, 100, 150, 200.

Resultando como mejores parámetros *max depth* = 10 y *min samples split* = 100.

Optimización de Hiperparámetros y entrenamiento del modelo: Random Forest Classifier

Para el modelo Random Forest Classifier se probaron los siguientes valores:

- *n estimators*: 50, 100, 150.
- *max depth*: 1, 5, 10, 15.
- *min samples split*: 20, 50, 100.

Resultando como mejores parámetros *n estimators* = 100, *max depth* = 15 y *min samples split* = 50.

Métricas en *train df*

Como métrica se usó la precisión pues ésta es la usada por kaggle para este desafío, de forma que, las métricas obtenidas para los modelos en *train df* fueron las siguientes:

- SVC: Se obtuvo una precisión de 0.77918.
- Decision Tree Classifier: Se obtuvo una precisión de 0.77976.
- Random Forest Classifier: Se obtuvo una precisión de 0.78838.
- Red Neuronal: Se obtuvo una precisión de 0.77378.

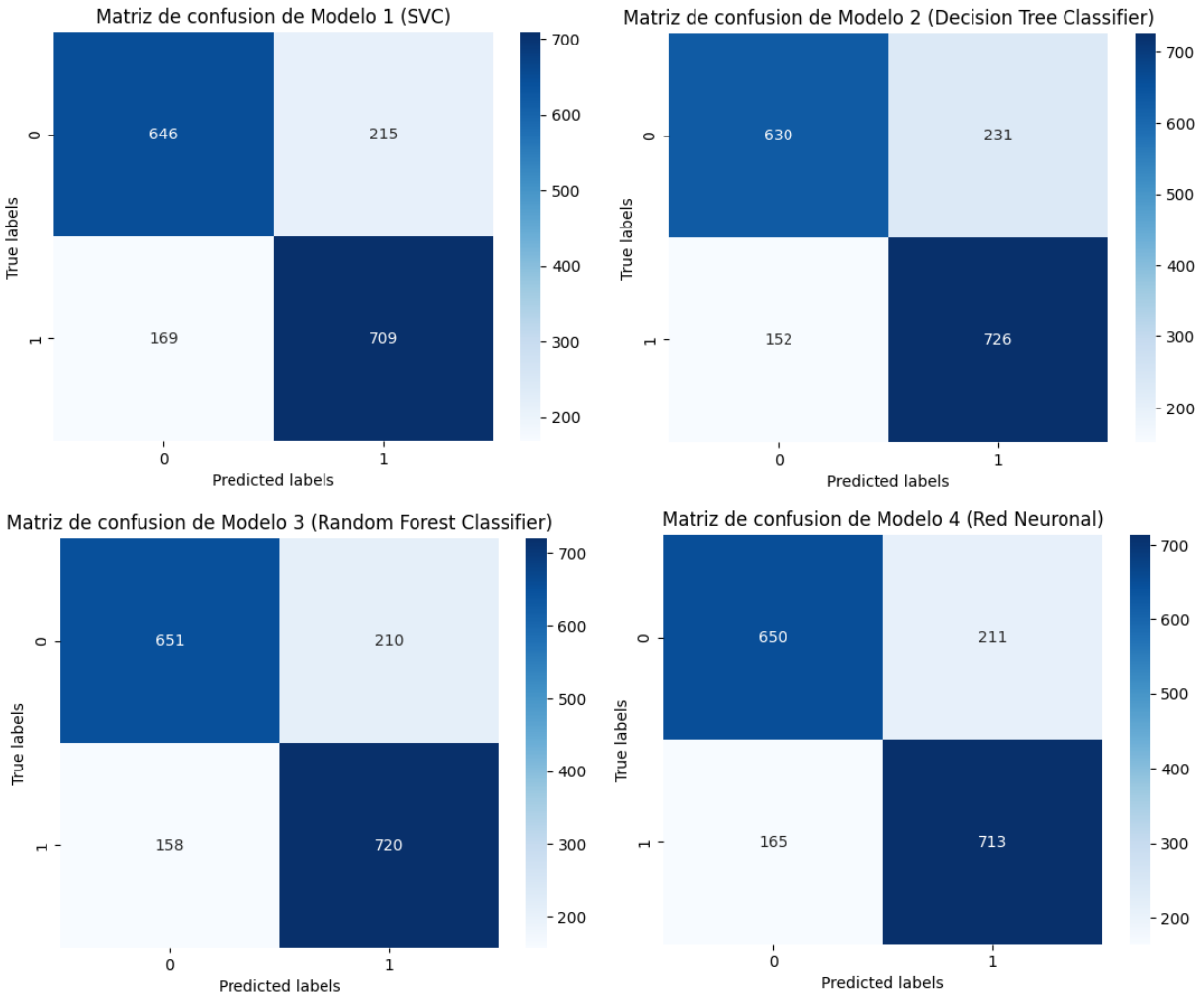
Métricas en *test df*

Se realizaron las predicciones de cada modelo sobre *test df*, se subieron a *Kaggle* y se obtuvieron los siguientes resultados:

- SVC: Se obtuvo una precisión de 0.79167.
- Decision Tree Classifier: Se obtuvo una precisión de 0.78115.
- Random Forest Classifier: Se obtuvo una precisión de 0.79003.
- Red Neuronal: Se obtuvo una precisión de 0.79261.

Puesto que se presenta una precisión similar a la obtenida en el conjunto de entrenamiento, entonces no se presenta overfitting en ninguno de los modelos.

Matrices de Confusión



Conclusión

¡Muchas gracias por su atención!