Proyecto final: Spaceship Titanic

Tomás Torres Lira

Departamento de Matemática
Universidad Técnica Federico Santa María

December 5, 2023

Contenidos

Contenidos

Introducción

Descripción del problema

Es el año 2912 y se recibió una transmisión desde cuatro años luz de distancia y las cosas no parecen ir del todo bien.
La nave espacial Titanic fue un transatlántico de pasajeros interestelar lanzado hace un mes. Alrededor de 13.000 pasajeros iban a bordo, la nave emprendió su viaje inaugural transportando emigrantes de nuestro sistema solar a tres exoplanetas recientemente habitables que orbitan estrellas cercanas.

Introducción

Descripción del problema

Mientras rodeaba Alpha Centauri en ruta hacia su primer destino, el tórrido 55 Cancri E, la desprevenida nave espacial Titanic chocó con una anomalía del espacio-tiempo escondida dentro de una nube de polvo. Lamentablemente, tuvo un destino similar al de su homónimo de 1000 años antes. Aunque la nave permaneció intacta, ¡casi la mitad de los pasajeros fueron transportados a una dimensión alternativa!

Desafío

Consiste en predecir hacía donde fueron los pasajeros desaparecidos mediante Machine Learning, para esto se requiere un análisis y procesamientos de los datos para aplicar algún modelo visto en clases.

Contenidos

Conjunto de datos

Dataset

Los datos obtenidos en la plataforma de Kaggle corresponderán a dos datasets los cuales son:

- Conjunto de entrenamiento (train.csv / train df)
- Conjunto de prueba (test.csv / test df)

La diferencia entre ellas es la presencia de la columna "Transported".

Conjunto de datos

Formato

La manera correcta de presentar y cargar la predicción en Kaggle es utilizando un archivo .csv que contenga las siguientes columnas:

- `PassengerID`: Identificación de cada pasajero a bordo en el conjunto de pruebas.
- `Transported`: Corresponde al target para cada pasajero, este predice si el valor es verdadero o falso (booleano)

Datos

Atributos categóricos

- "PassengerId"
- "HomePlanet"
- "CryoSleep"
- "Cabin"
- "Destination"
- "VIP"
- "Name"

Datos

Atributos continuos

- "Age"
- "RoomService"
- "FoodCourt"
- "ShoppingMall"
- "Spa"
- "VRDeck"

Estadística descriptiva

Para ambos conjuntos de datos se tendrá:

- train df:

```
train_df.describe()
```

	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

Figure: Valores del conjunto de entrenamiento

- test df:

```
test_df.describe()
```

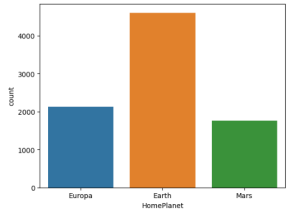
	Age	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
count	4186.000000	4195.000000	4171.000000	4179.000000	4176.000000	4197.000000
mean	28.658146	219.266269	439.484296	177.295525	303.052443	310.710031
std	14.179072	607.011289	1527.663045	560.821123	1117.186015	1246.994742
min	0.000000	0.000000	0.000000	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

Figure: Valores del conjunto de prueba

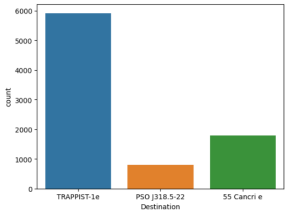
Contenidos

Visualización descriptiva

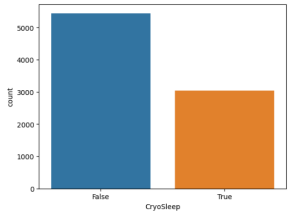
Procedecencia:



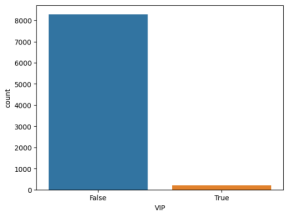
Destino:



Estado:



Condición:



Relación con respecto a teletransporte

Edad:

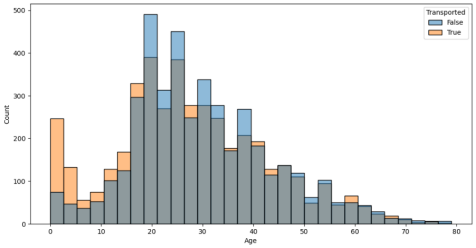


Figure: Edad vs teletransportados

Servicio habitación:

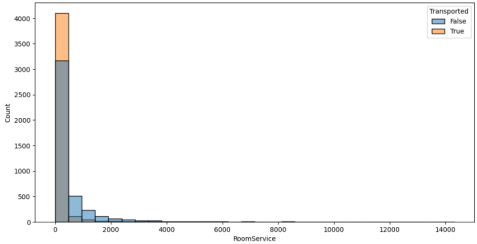


Figure: Servicio habitación vs teletransportados

Relación con respecto a teletransporte

Servicio comida:

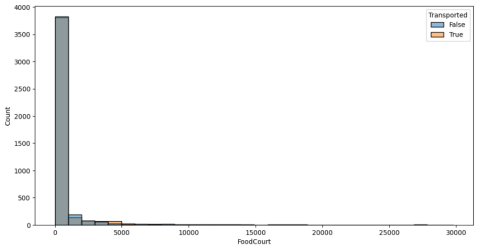


Figure: Servicio comida vs teletransportados

Gastos compras:

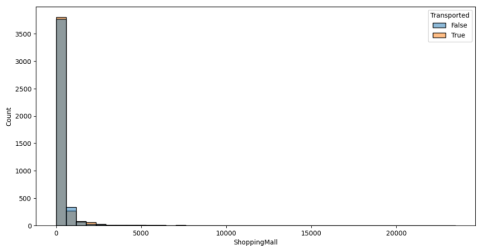


Figure: Gastos compras vs teletransportados

Relación con respecto a teletransporte

Spa:

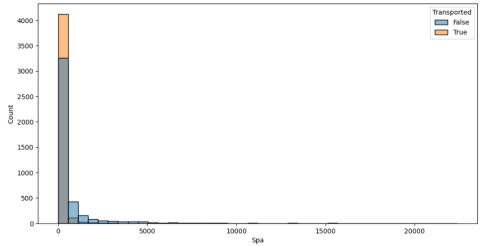


Figure: Servicio Spa vs teletransportados

VRDeck:

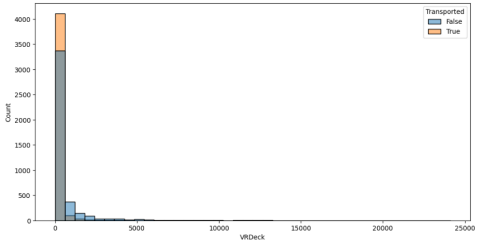


Figure: VRDeck vs teletransportados

Cantidad teletransportados

Pasajeros teletransportados vs no teletransportados

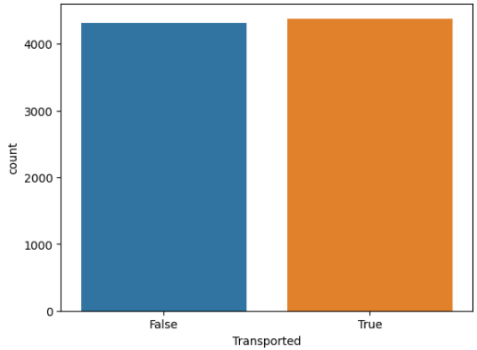


Figure: Comparación cantidad pasajeros

Contenidos

Tratamiento de datos

El proceso de preprocesamiento comenzó con la limpieza de datos, siguiendo estos pasos:

- Separación del conjunto en datos de entrada y salida (solo para train df).
- Revisión de los datos que contienen valores NaN.

Para los atributos se realizará lo siguiente:

- Para los atributos categóricos con valores NaN se evaluó la cantidad de clases por atributo, se eliminaron los atributos irrelevantes y para los restantes se reemplazaron los valores NaN por la moda de la columna.
- En caso de ser atributos numéricos con valores NaN se sustituyeron los valores por la media de la columna.

Tratamiento de datos

Se llevaron a cabo las divisiones de train/test en:

- train cat/test cat: agrupando los atributos categóricos que necesitan pasar por un encoder para su uso en el modelo.
- train cont/test cont: agrupando los atributos continuos que deben pasar por un escalador (scaler) para su uso en el modelo.
- Además, se crearon conjuntos separados para los atributos "VIP" y "CryoSleep".

Entrenamiento de modelo

One hot encoder

El One Hot Encoder se emplea para atributos categóricos que no tienen un orden inherente entre ellos, lo que resulta en un aumento en la dimensión del problema.

Standard scaler

Por otro lado, el Standard Scaler se utiliza para eliminar la media y ajustar la varianza a 1, lo que garantiza que no existan inconvenientes al entrenar un modelo.

Contenidos

Selección de modelo

En este proyecto, se emplearon tres modelos de la biblioteca sklearn de Python y una red neuronal desarrollada a partir de la biblioteca TensorFlow, los cuales son:

- **Modelo 1:** Decision Tree Classifier
- **Modelo 2:** Random Forest Classifier
- **Modelo 3:** Support Vector Classifier (SVC)
- **Modelo 4:** Red Neuronal

Entrenamiento del modelo

Preliminares

La función "train test split" de sklearn se utilizó para dividir el conjunto de datos train df en conjuntos de entrenamiento y prueba, permitiendo entrenar y probar los modelos.

Optimización de Hiperparámetros

Modelo 1: Decision Tree Classifier

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

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

Los mejores parámetros resultaron ser max depth = 10 y min samples split = 100.

Modelo 2: Random Forest Classifier

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

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

Los mejores parámetros resultaron ser n estimators = 100, max depth = 15 y min samples split = 50.

Optimización de Hiperparámetros

Modelo 3: Support Vector Classifier (SVC)

Para el modelo SVC se evaluaron los siguientes valores:

- C: 0.1, 1, 10.
- kernel: lineal, rbf.

Los mejores parámetros resultaron ser C = 10 y kernel = 'rbf'.

Contenidos

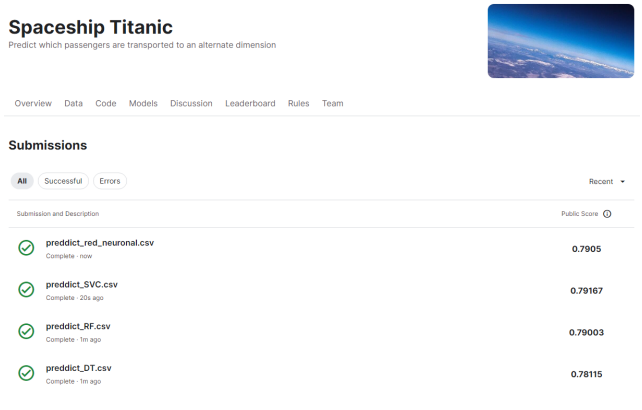
Métricas para conjuntos de entrenamiento

La métrica utilizada fue la precisión, ya que es la métrica empleada por Kaggle para este desafío. A continuación, se presentan las métricas obtenidas para los modelos en el conjunto de datos de entrenamiento:

- **Modelo 1:** Decision Tree Classifier: Se obtuvo una precisión de 0.77976
- **Modelo 2:** Random Forest Classifier: Se obtuvo una precisión de 0.78953.
- **Modelo 3:** SVC: Se obtuvo una precisión de 0.77918.
- **Modelo 4:** Red Neuronal: Se obtuvo una precisión de 0.77688.

Métricas para conjuntos de testeo

Se realizaron las predicciones para cada uno de los modelos sobre test df, posteriormente se subieron a la plataforma Kaggle donde se obtuvieron los siguientes resultados:



Métricas para conjuntos de testeo

Al tabular los valores obtenidos se tendrá lo siguiente:

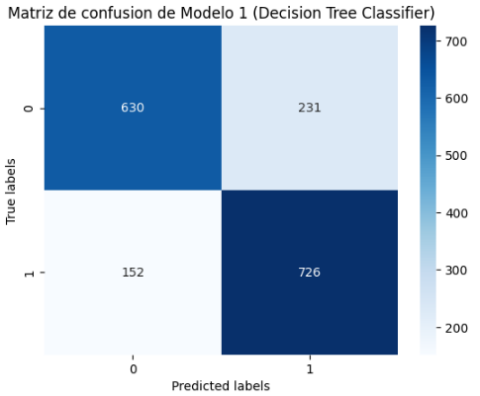
Modelos / Métricas	Proyecto	Kaggle
DT	0.77976	0.78115
RT	0.78953	0.79003
SVC	0.77918	0.79167
Red Neu	0.77688	0.7905

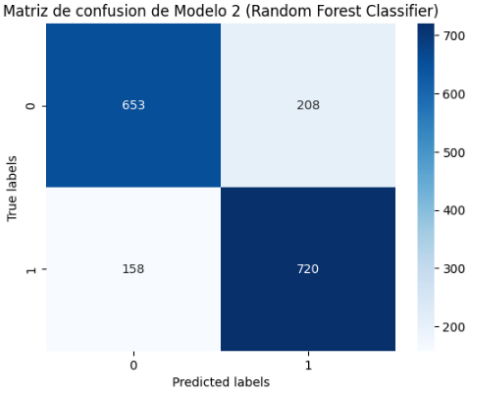
Al presentarse una precisión similar a la obtenida en el conjunto de entrenamiento, se puede determinar entonces que no se presenta overfitting para ninguno de los modelos planteados.

Contenidos

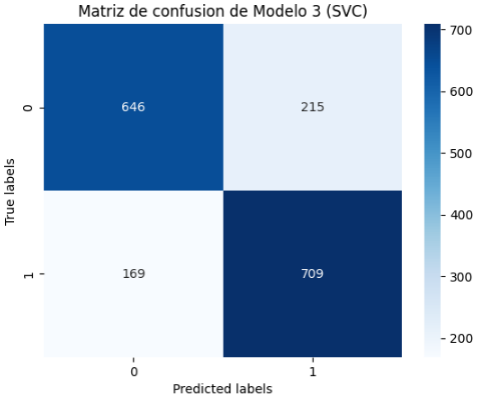
Matriz de confusión

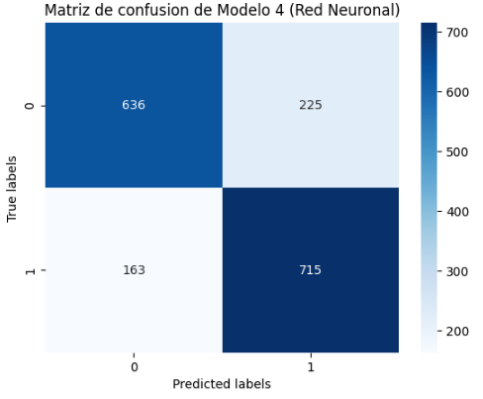
Por medio de matrices de confusión es posible notar el comportamiento de los modelos, es decir:





Matriz de confusión





Se observa que los modelos en la mayoría de los casos predicen correctamente. También, se ve que los modelos tienden a no cometer más un tipo de error.

Contenidos

Conclusión

Para el proyecto, aunque los modelos propuestos mostraron resultados similares, se recomienda elegir el modelo más eficiente en términos computacionales. Además, se observó un fenómeno interesante: el error en los datos de prueba resultó ser menor que en los datos de entrenamiento.

Esta situación podría explicarse por la capacidad de los datos de entrenamiento para proporcionar un alto nivel de información a los modelos, lo que permitió una mejor generalización y predicción precisa en los datos de prueba.

Contenidos

• Kaggle. (2023). Spaceship Titanic Competition. Recuperado de https://www.kaggle.com/competitions/spaceship-titanic/

Proyecto final: Spaceship Titanic

Tomás Torres Lira

Departamento de Matemática
Universidad Técnica Federico Santa María

December 5, 2023