

Titanic 2012


Proyecto de MAT281

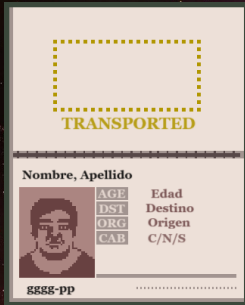
Ignacio Allendes

Universidad Técnica Federico Santa María

1 de Diciembre de 2023

Tabla de Contenidos

- 
- 1 Introducción
 - 2 Análisis exploratorio de datos
 - 3 Procesamiento de datos
 - 4 Modelos y Métricas
 - 5 Conclusiones



- PassengerId (gggg-pp)
- Homeplanet
- CryoSleep
- Cabin (deck/num/side)
- Destination
- Age
- VIP
- RoomService, ShoppingMall, Spa, VRDeck
- Name

Un poco de estadísticas

| | Age | RoomService | FoodCourt | ShoppingMall | Spa | VRDeck |
|-------|-------|-------------|-----------|--------------|---------|---------|
| count | 8514 | 8512 | 8510 | 8485 | 8510 | 8505 |
| mean | 28.83 | 224.69 | 458.08 | 173.73 | 311.14 | 304.85 |
| std | 14.49 | 666.72 | 1611.49 | 604.70 | 1136.71 | 1145.72 |
| min | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 % | 19 | 0 | 0 | 0 | 0 | 0 |
| 50 % | 27 | 0 | 0 | 0 | 0 | 0 |
| 75 % | 38 | 47 | 76 | 27 | 59 | 46 |
| max | 79 | 14327 | 29813 | 23492 | 22408 | 24133 |

Tabla: Estadísticas del conjunto de entrenamiento

señalar que por cada atributo hay alrededor de 200 NaNs

Visualización descriptiva

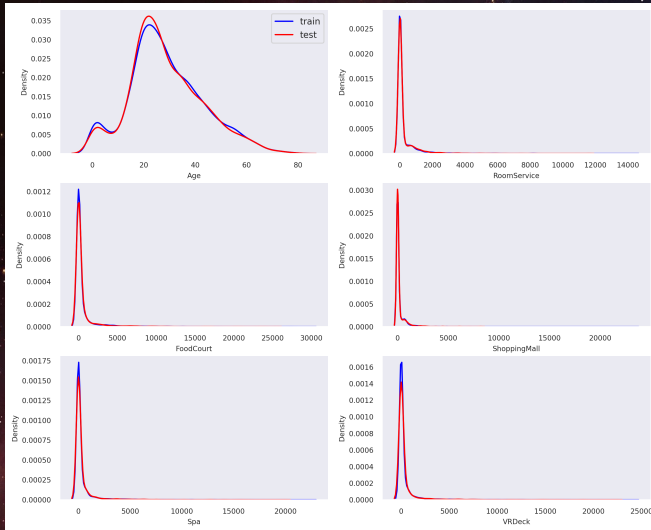


Figura: Distribucion atributos continuos

Visualización descriptiva

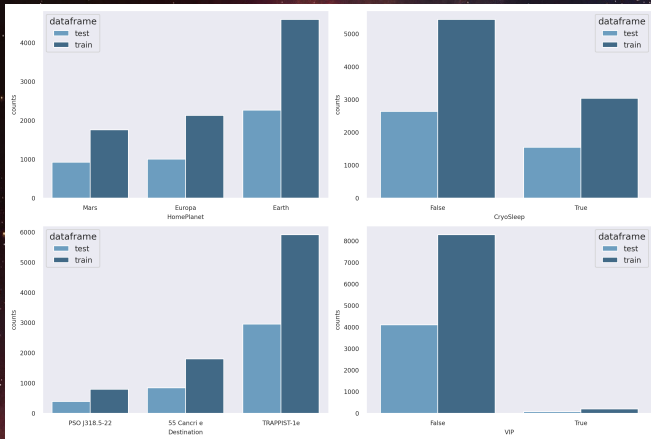


Figura: Distribucion atributos categoricos

Visualización descriptiva

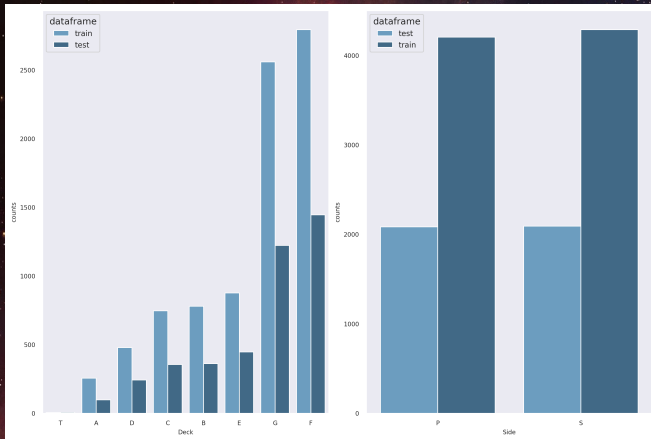


Figura: Distribucion atributos categoricos

Visualización descriptiva

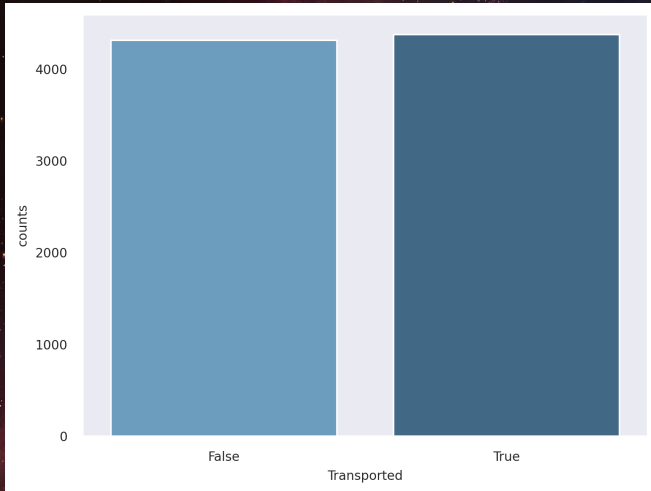


Figura: Distribucion atributos categoricos

- Para los atributos continuos faltantes, los rellenaremos con la media
- Para los categoricos, con la categoria '?'
- separamos Cabin en Deck/Num/Side
- Del PassengerId sacamos el GroupSize
- Botamos los atributos que no usaremos, como el Name.

Con esto tratamos todos los NaNs del conjunto de entrenamiento y pruebas.

Finalmente, usamos MinMaxScaler y OneHotEncoder para el escalamiento y codificación de los datos.

Consideramos los modelos LogisticRegression, SVC, GradientBoosting y RandomForest.

| | Train | Test |
|--------------------|-------|------|
| LogisticRegression | 0.76 | 0.78 |
| SVC | 0.76 | 0.73 |
| GradientBoosting | 0.82 | 0.82 |
| RandomForest | 1 | 0.8 |

Tabla: Accuracy de los modelos base

Optimización de Hiperparametros

Ahora, mediante GridSearch, realizamos optimización de hiperparametros a cada modelo.

Para LogisticRegression consideramos la siguiente red

- $C = 0.001, 0.01, 0.1, 1, 10, 100$
- $\text{penalty} = l1, l2$

Para SVC,

- $C = 0.1, 1, 10, 100$
- $\text{kernel} = \text{linear}, \text{rbf}, \text{poly}$

Para GradientBoosting,

- $n_estimators = 10, 100, 500, 1000$
- $\text{max_depth} = 1, 3, 5, 7$

Para RandomForest,

- $n_estimators = 10, 100, 500, 1000$
- $\text{max_depth} = 1, 3, 5, 7$

Optimización de Hiperparametros

- Para LogisticRegression, obtuvimos $C = 100$ y $\text{penalty} = l2$.
- Para SVC, obtuvimos $C = 100$, $\text{kernel} = \text{poly}$.
- Para GradientBoosting, obtuvimos $n_estimators = 500$ y $\text{max_depth} = 3$.
- Para RandomForest, obtuvimos $n_estimators = 1000$ y $\text{max_depth} = 7$.

Optimización de Hiperparametros

| | Test |
|--------------------|------|
| LogisticRegression | 0.80 |
| SVC | 0.80 |
| GradientBoosting | 0.82 |
| RandomForest | 0.80 |

Tabla: Accuracy de los mejores modelos en GridSearch

Por el desempeño antes y despues de la optimización, nos quedamos con GradientBoosting

Usamos classification report para obtener varias metricas.

| | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|---------------------|
| No Transportado | 0.84 | 0.80 | 0.82 | 3893 |
| Transportado | 0.81 | 0.85 | 0.83 | 3930 |
| accuracy | | | | 0.82 7823 |
| macro avg | | | | 0.82 0.82 0.82 7823 |
| weighted avg | | | | 0.82 0.82 0.82 7823 |

Tabla: Classification Report de GradientBoosting en el conjunto de entrenamiento

Usamos classification report para obtener varias metricas.

| | precision | recall | f1-score | support |
|-----------------|-----------|--------|----------|--------------------|
| No Transportado | 0.81 | 0.81 | 0.81 | 422 |
| Transportado | 0.82 | 0.83 | 0.82 | 448 |
| accuracy | | | | 0.82 870 |
| macro avg | | | | 0.82 0.82 0.82 870 |
| weighted avg | | | | 0.82 0.82 0.82 870 |

Tabla: Classification Report de GradientBoosting en el conjunto de pruebas

También tenemos las siguientes matrices de confusión.

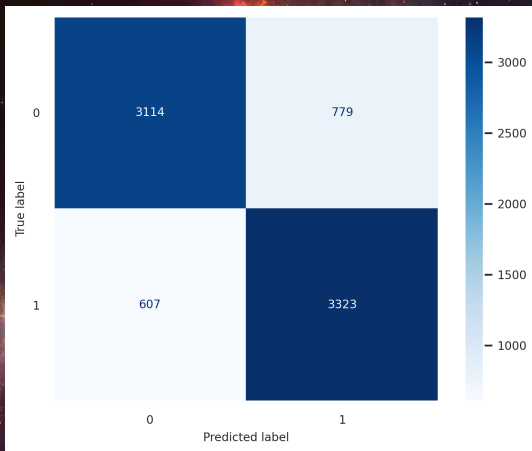


Figura: Matriz de confusión en el conjunto de entrenamiento

También tenemos las siguientes matrices de confusión.

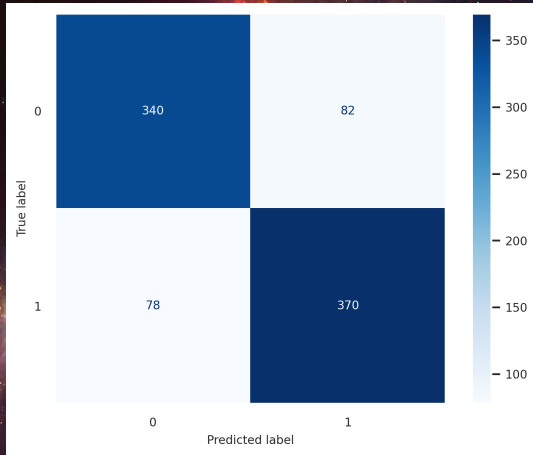


Figura: Matriz de confusión en el conjunto de pruebas

Notemos que aplicando un PCA de 2 componentes, los datos no son separables a simple vista

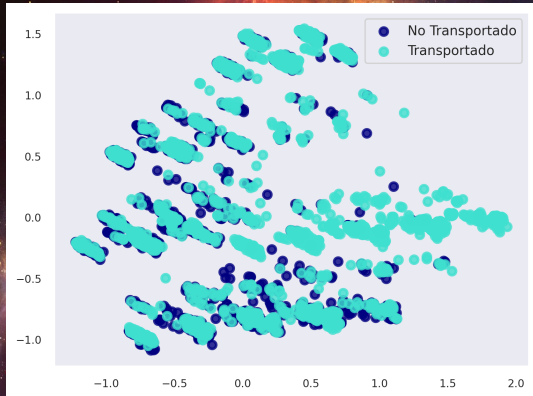


Figura: PCA de 2 componentes

Y aplicando un PCA de 3 componentes, los datos tampoco son separables a simple vista

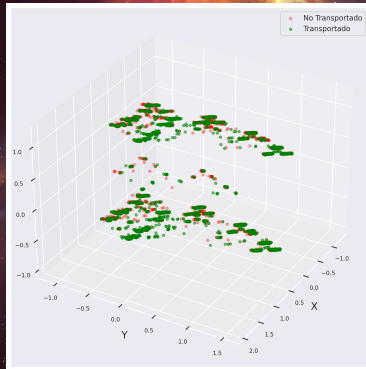


Figura: PCA de 3 componentes

El siguiente diagrama indica como funciona GradientBoosting

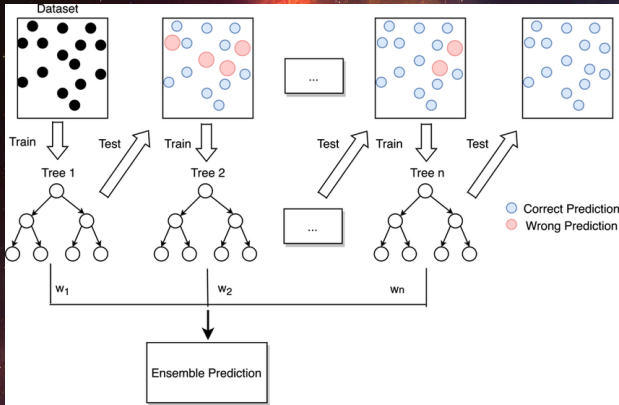


Figura: diagrama de flujo del modelo

Debido a la naturaleza de GradientBoosting, podemos visualizar los arboles que crea, por ejemplo, este es el arbol 99

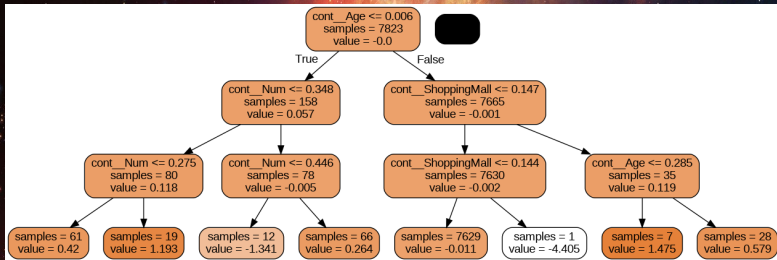


Figura: Arbol 99 del modelo

- En este proyecto, atacamos un problema de clasificacion de inicio a fin
- Aplicamos todo lo visto en el curso
- Obtuvimos un clasificador consistente
- Obtuvimos un score en el Kaggle de 0.80476