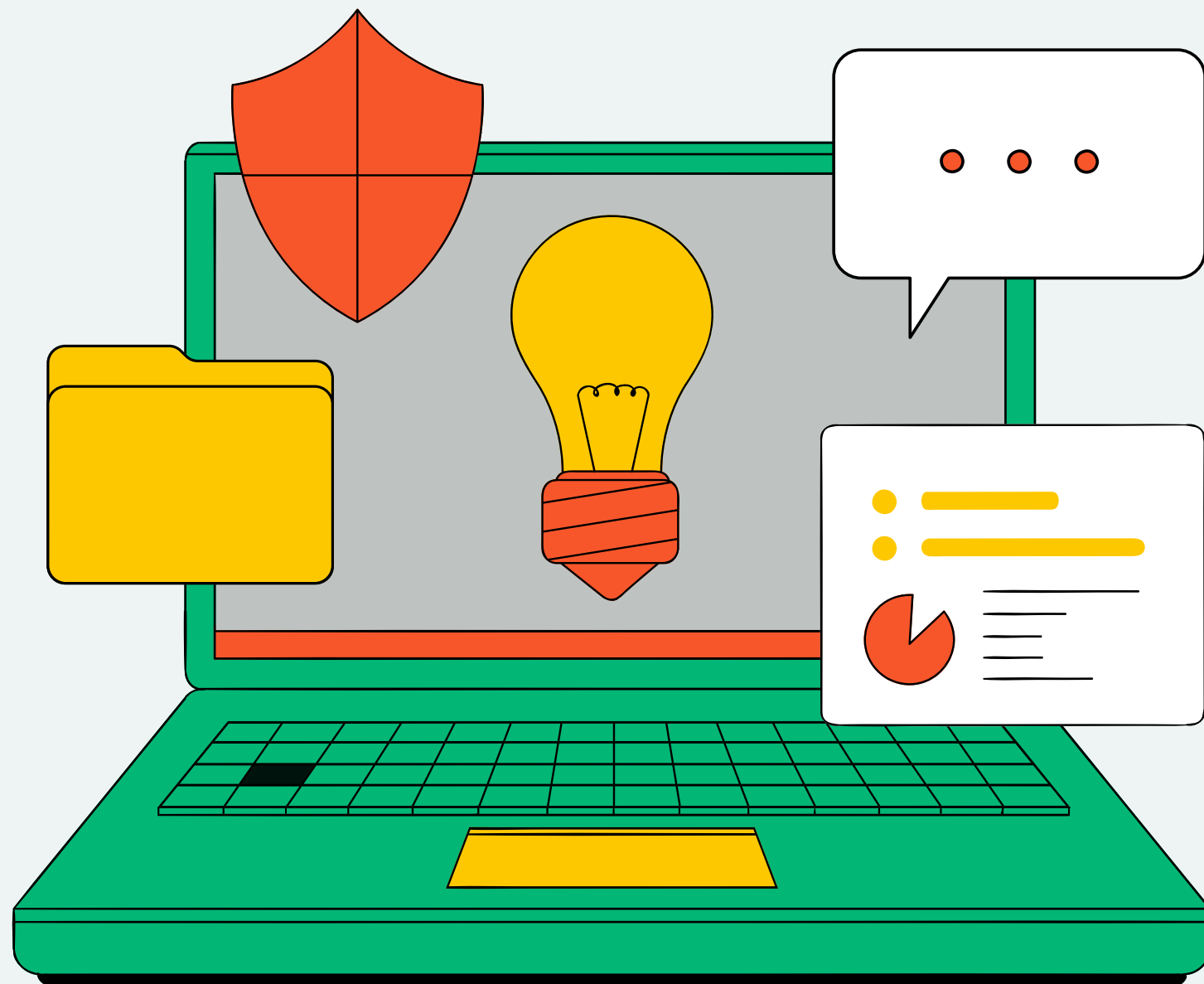


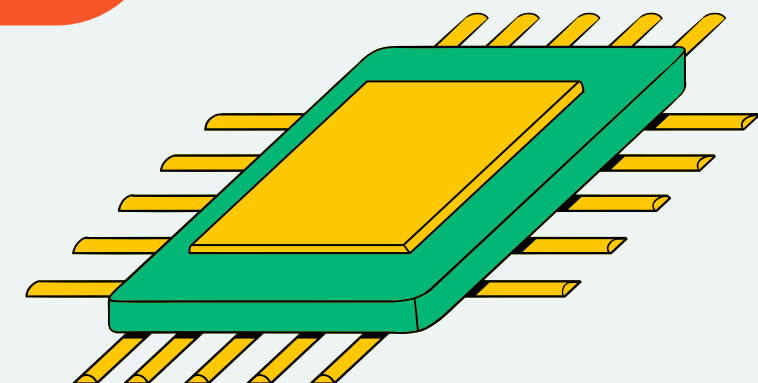
SPACESHIP TITANIC
KAGGLE · PREDICTION COMPETITION



ROMULUS SPACESHIP

PRESENTADO POR:

ERIC MANUEL NAVARRO MARTÍNEZ
GUSTAVO ALEJANDRO GUTIERREZ VALDÉS
GUSTAVO TÉLLEZ MIRELES
MARILUZ DANIELA SÁNCHEZ MORALES
PABLO SPÍNOLA LÓPEZ



ANÁLISIS - PROBLEMA CLASIFICATORIO



La nave espacial Titanic fue un transatlántico interestelar lanzado hace un mes. Con **casi 13.000 pasajeros a bordo**, la nave emprendió su viaje inaugural transportando emigrantes de nuestro sistema solar a tres exoplanetas recientemente habitables que orbitan estrellas cercanas.

Mientras rodeaba Alpha Centauri en ruta hacia su primer destino, el tórrido 55 Cancri E, la **incauta nave espacial Titanic chocó con una anomalía del espacio-tiempo** escondida dentro de una nube de polvo.

Lamentablemente, corrió un destino similar al de su homónimo de 1000 años antes. Aunque el barco permaneció intacto, **¡casi la mitad de los pasajeros fueron transportados a una dimensión alternativa!**



ANÁLISIS DE LAS COLUMNAS



PassengerId: Cada ID tiene la forma gggg_pp donde gggg indica un grupo con el que viaja el pasajero y pp es su número dentro del grupo.

HomePlanet: El planeta del que partió el pasajero.

CryoSleep: Indica si el pasajero está en animación suspendida durante la duración del viaje.

Cabin: El número de cabina donde se hospeda el pasajero. Toma el formato cubierta/número/lado.

Destination: Planeta donde desembarcará el pasajero.

Age: Edad del pasajero.

VIP: Si el pasajero ha pagado VIP.

RoomService, FoodCourt, ShoppingMall, Spa,

VRDeck: Monto que el pasajero ha pagado en cada una de las amenidades del Spaceship Titanic.

Name: Nombre y apellido.

Transported: Si el pasajero fue transportado a otra dimensión.





ANÁLISIS - PARA TENER VALORES NUMÉRICOS Y LLENAR VACÍOS

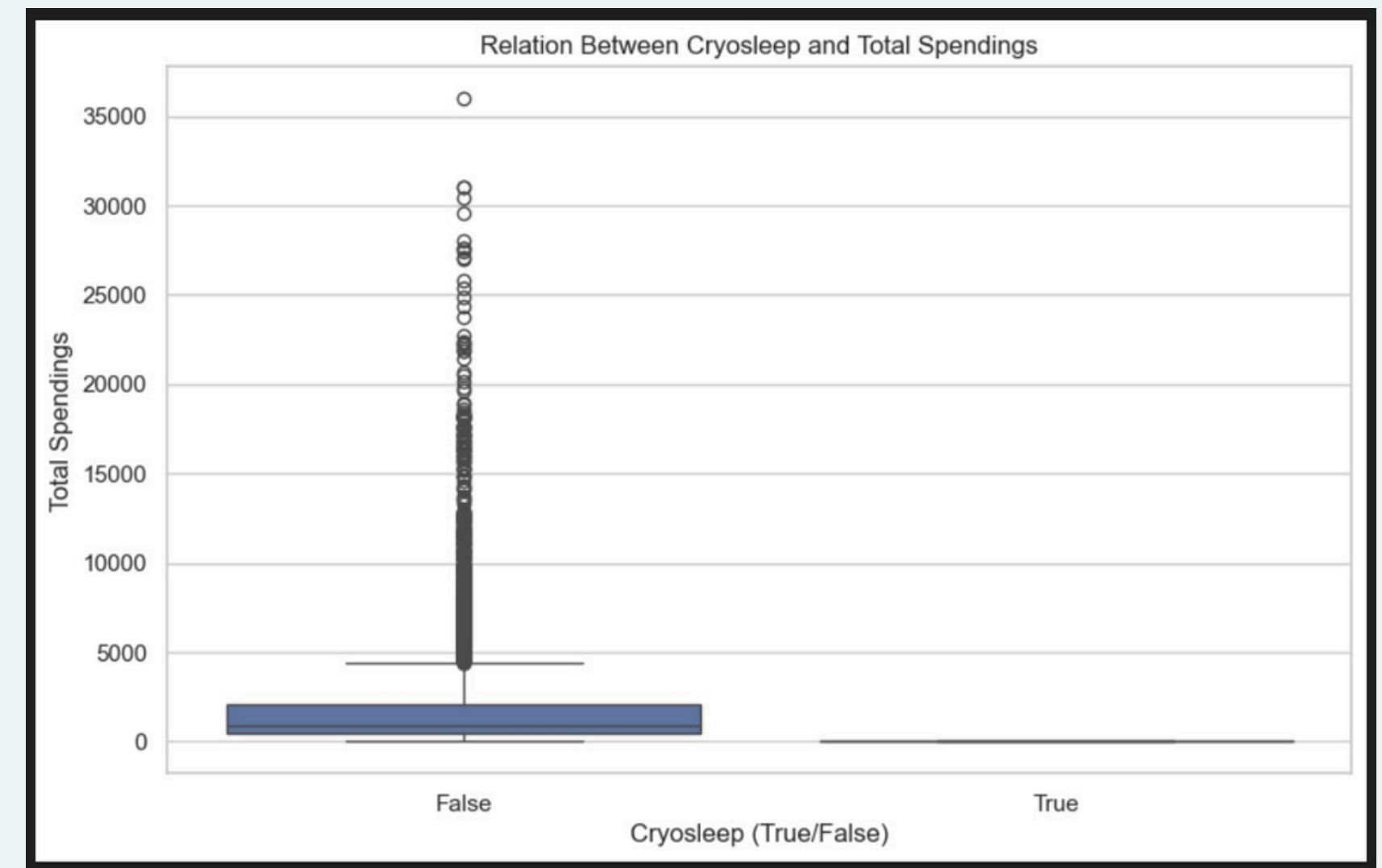
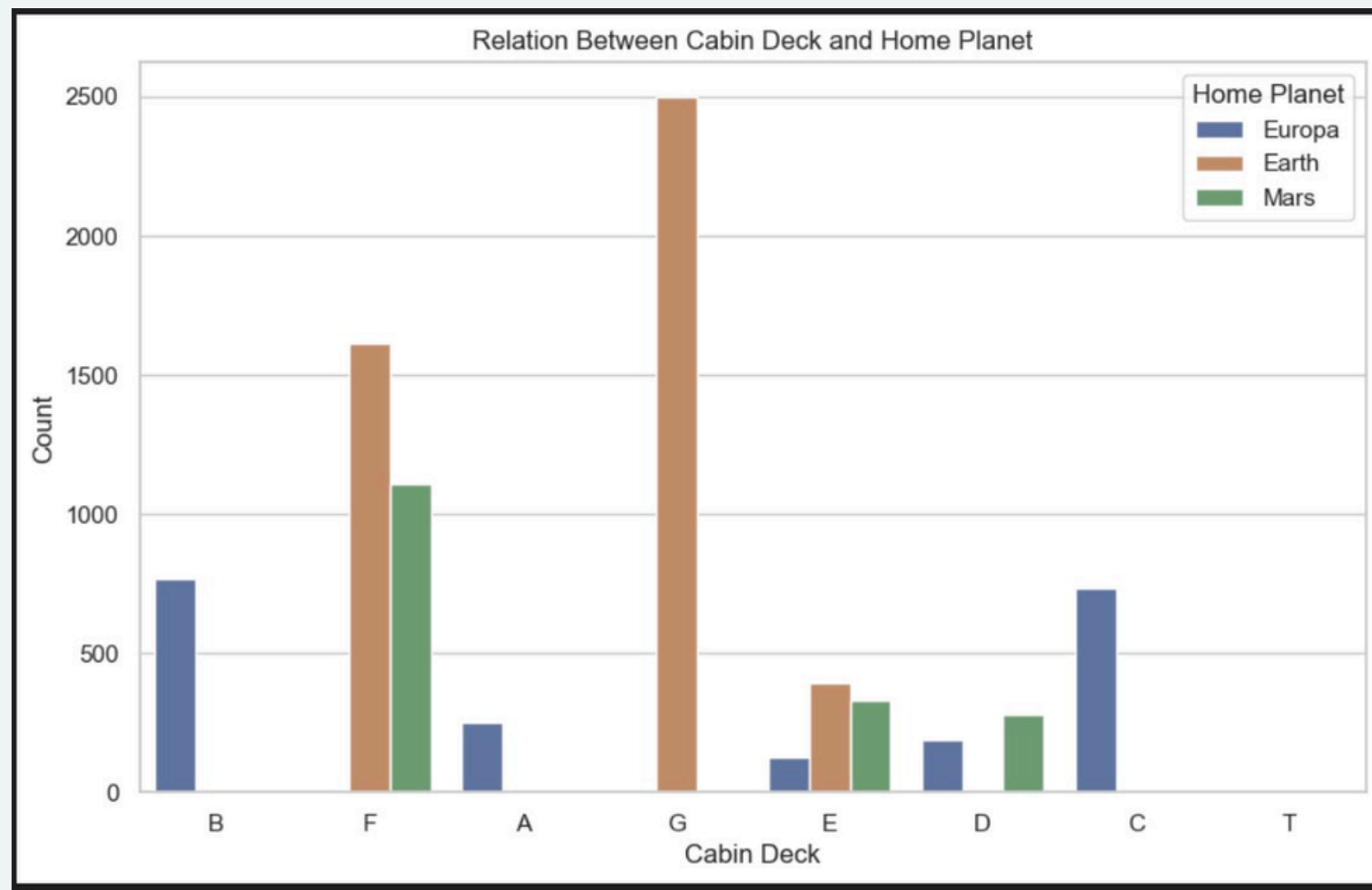
PassengerId: Se separa en 2 enteros: **Número de grupo** y **Número de pasajero**.

HomePlanet: Label encoding: Tierra -> 0, Marte -> 1, Europa: -> 2. Imputar con moda.

CryoSleep: En caso de haber gastado, CryoSleep es 0. En otro caso llenar con 1.

Cabin: Se crean las columnas **Deck**, **CabinNum** y **CabinSide**. Se elimina **Cabin**.

Deck: Label encoding: A-G -> 0-6, T -> 7. Si viene de la Tierra, deck es G. Imputar con moda.





ANÁLISIS - PARA TENER VALORES NUMÉRICOS Y LLENAR VACÍOS

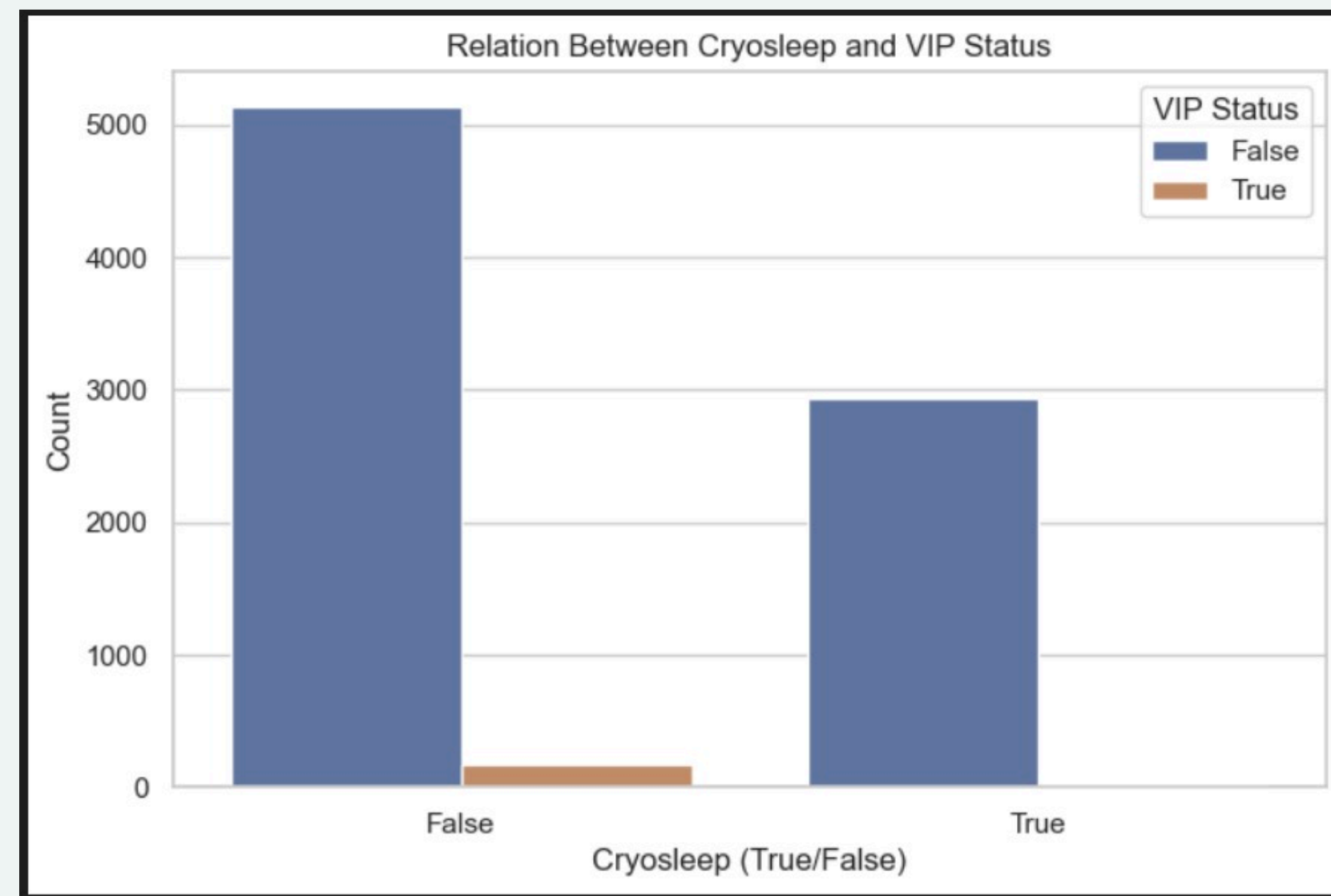
CabinNum: Si hay simetría, imputar con media. Si no, imputar con mediana.

CabinSide: Label encode: P → 0, S → 1. Imputar con moda.

Destination: Label encode: 0 – 2. Imputar con moda.

Age: Si el gasto total es 0, se llena con 6. Se observa simetría, imputar con media o mediana.

VIP: Si se encuentra en CryoSleep, no es VIP. De lo contrario imputar con moda.





ANÁLISIS - PARA TENER VALORES NUMÉRICOS Y LLENAR VACÍOS

RoomService, FoodCourt, ShoppingMall, Spa, VRDeck:

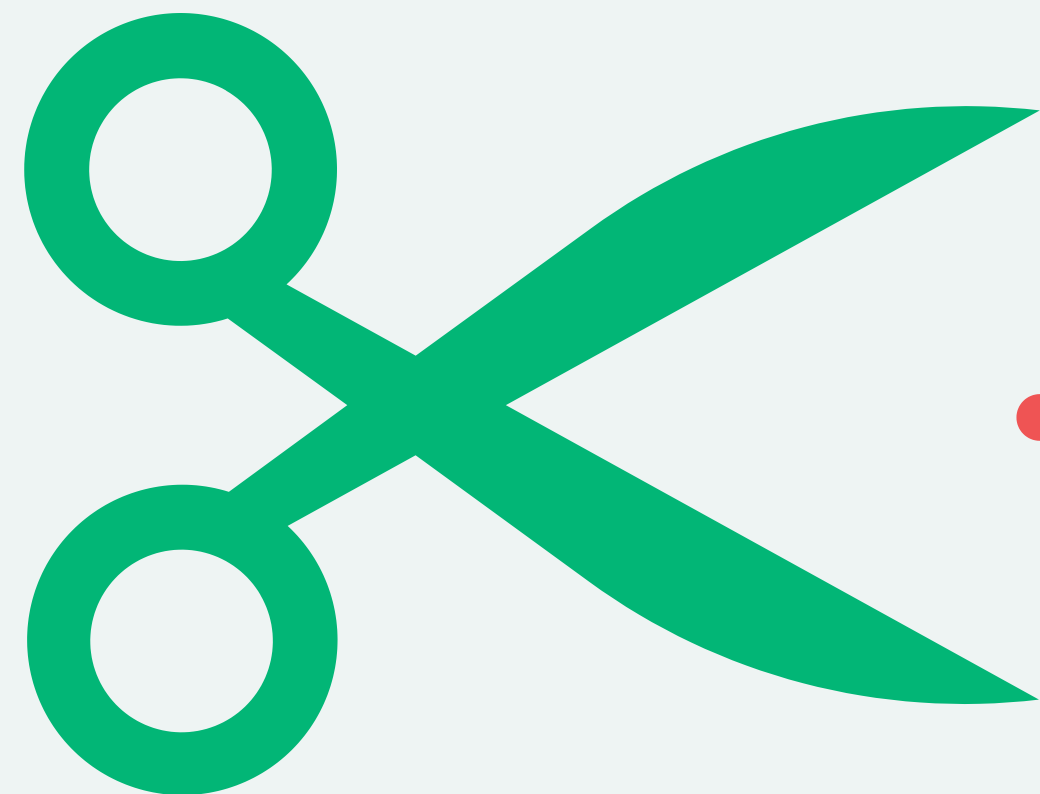
Si está en CryoSleep, el gasto es 0.

Si no, se observa simetría y se imputa con media o mediana.

TotalSpent: Es la suma de lo gastado en amenidades.

Name: Se elimina la columna, la información familiar se encuentra en el número de grupo.

HomePlanet_Destination_Interaction: Relación entre origen y destino.





COLUMNAS FINALES - ESTANDARIZACIÓN

- HomePlanet
- CryoSleep
- Destination
- Age
- VIP
- RoomService
- FoodCourt
- ShoppingMall
- Spa
- VRDeck
- Transported
- GroupNum
- PassNum
- Deck
- CabinNumber
- CabinSide
- TotalSpent
- HomePlanet_Destination_Interaction
- Transported

HomePlar	CryoSleep	Destinatic	Age	VIP	RoomService	FoodCourt	ShoppingMall	Spa
1.558989	-0.73273	-0.60128	0.716426	-0.15305	-0.3330855	-0.281010566	-0.283562321	-0.27061
-0.82593	-0.73273	-0.60128	-0.31648	-0.15305	-0.168063764	-0.275370728	-0.241756802	0.217146
1.558989	-0.73273	-0.60128	2.024779	6.532879	-0.267985182	1.959884912	-0.283562321	5.695295
1.558989	-0.73273	-0.60128	0.303263	-0.15305	-0.3330855	0.522979618	0.336831584	2.687022
-0.82593	-0.73273	-0.60128	-0.86737	-0.15305	0.125644646	-0.237145162	-0.031056985	0.231361
-0.82593	-0.73273	2.480648	1.06073	-0.15305	-0.3330855	0.021660719	-0.283562321	-0.01207
-0.82593	-0.73273	-0.60128	-0.17876	-0.15305	-0.269499143	0.683401666	-0.278545659	-0.27061
-0.82593	1.364606	-0.60128	-0.04104	-0.15305	-0.3330855	-0.281010566	-0.283562321	-0.27061
-0.82593	-0.73273	-0.60128	0.440984	-0.15305	-0.3330855	0.210908604	-0.255134568	-0.07871
1.558989	1.364606	0.939682	-1.00509	-0.15305	-0.3330855	-0.281010566	-0.283562321	-0.27061
1.558989	1.364606	-0.60128	0.372123	-0.15305	-0.3330855	-0.281010566	-0.283562321	-0.27061
1.558989	-0.73273	0.939682	1.12959	-0.15305	-0.274041025	4.290391144	0.701375712	-0.17288
0.366531	-0.73273	-0.60128	0.234402	-0.15305	-0.222566356	-0.281010566	1.594341603	-0.27061
-0.82593	-0.73273	-0.60128	1.336172	-0.15305	0.75545237	-0.280383917	-0.174867971	-0.27061
-0.82593	-0.73273	-0.60128	-0.04104	-0.15305	-0.320973813	0.329345193	-0.263495672	-0.26883

VRDeck	GroupNum	PassNum	TotalSpent	Deck	CabinNumber	CabinSide	HomePlanet_Destination_Interaction	Transported	
-0.26299	-1.73430909	-0.491133206	-0.514035976	-1.884640596	-1.186542178	-1.032805209	-0.376594831	0	
-0.22419	-1.733934702	-0.491133206	-0.251464445	0.377682834	-1.186542178	0.968125409	-0.376594831	1	
-0.21978	-1.733560315	-0.491133206	3.190149091	-2.450221453	-1.186542178	0.968125409	-0.376594831	0	
-0.09281	-1.733560315	0.457416218	1.332526858	-2.450221453	-1.186542178	0.968125409	-0.376594831	0	
-0.26122	-1.733185927	-0.491133206	-0.124816492	0.377682834	-1.184565791	0.968125409	-0.376594831	1	
-0.26299	-1.73281154	-0.491133206	-0.237907763	0.377682834	-1.186542178	-1.032805209	-0.376594831	1	
-0.26299	-1.732437152	-0.491133206	0.051063624	0.377682834	-1.182589404	0.968125409	-0.376594831	1	
-0.26299	-1.732437152	0.457416218	-0.514035976	0.943263692	-1.186542178	0.968125409	-0.376594831	1	
-0.26299	-1.732062764	-0.491133206	-0.150859592	0.377682834	-1.180613017	0.968125409	-0.376594831	1	
-0.26299	-1.731688377	-0.491133206	-0.514035976	-1.884640596	-1.184565791	-1.032805209	2.684404578	1	
-0.26299	-1.731688377	0.457416218	-0.514035976	-1.884640596	-1.184565791	-1.032805209	-0.376594831	1	
-0.15365	-1.731688377	1.405965642	2.39601291	-1.884640596	-1.184565791	-1.032805209	2.684404578	1	
-0.16335	-1.731313989	-0.491133206	-0.047043946	0.377682834	-1.184565791	-1.032805209	-0.376594831	1	
-0.24183	-1.730939602	-0.491133206	-0.225421345	0.943263692	-1.184565791	0.968125409	-0.376594831	0	
-0.25682	-1.730565214	-0.491133206	-0.156210914	0.377682834	-1.182589404	-1.032805209	-0.376594831	1	

MODELOS



Regresión Logística

Random Forest



CatBoost



EVALUACIÓN DE MODELOS (BASE)

RANDOM FOREST:

ACCURACY: 0.8021851638872916

PRECISION: 0.8091236494597839

RECALL: 0.7846332945285215

F1 SCORE: 0.7966903073286052

LOGISTIC REGRESSION:

ACCURACY: 0.7906843013225991

PRECISION: 0.7722772277227723

RECALL: 0.8172293364377182

F1 SCORE: 0.7941176470588235

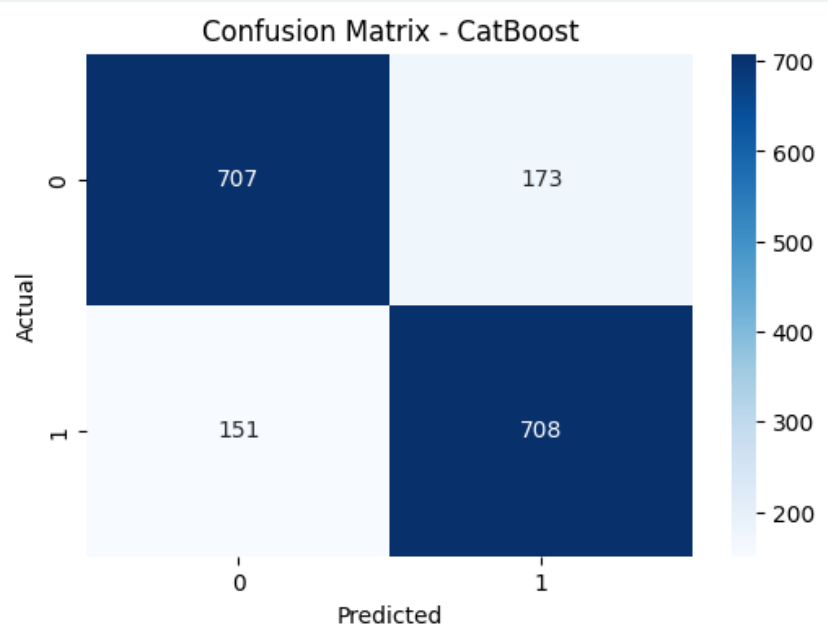
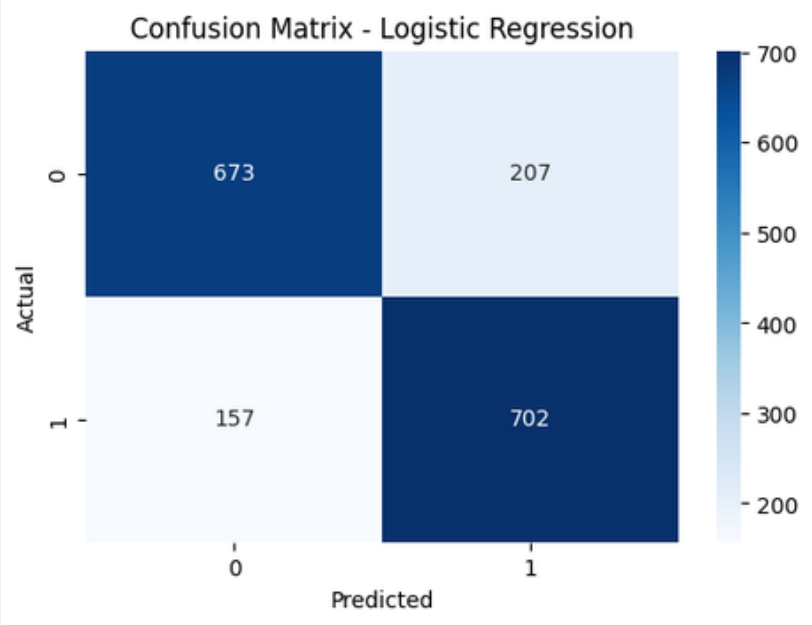
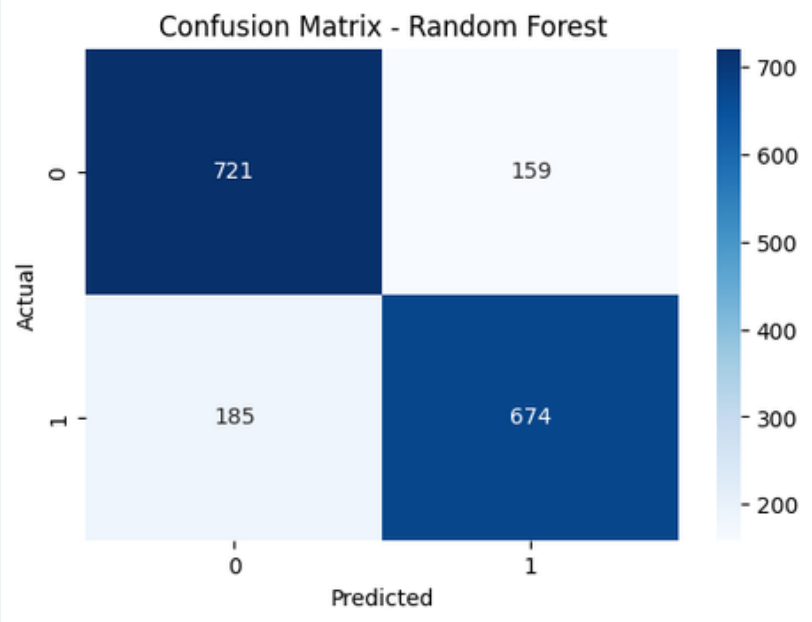
CATBOOST:

ACCURACY: 0.8136860264519838

PRECISION: 0.8036322360953462

RECALL: 0.8242142025611175

F1 SCORE: 0.8137931034482758





AJUSTE DE HIPERPARÁMETROS

Se realizó un **Grid Search** para buscar los mejores hiperparámetros y posteriormente entrenar con cada modelo con sus mejores hiperparámetros. CV = 5



RANDOM FOREST BEST PARAMETERS:

```
'BOOTSTRAP': FALSE,  
'MAX_DEPTH': NONE,  
'MIN_SAMPLES_LEAF': 4,  
'MIN_SAMPLES_SPLIT': 10,  
'N_ESTIMATORS': 200
```

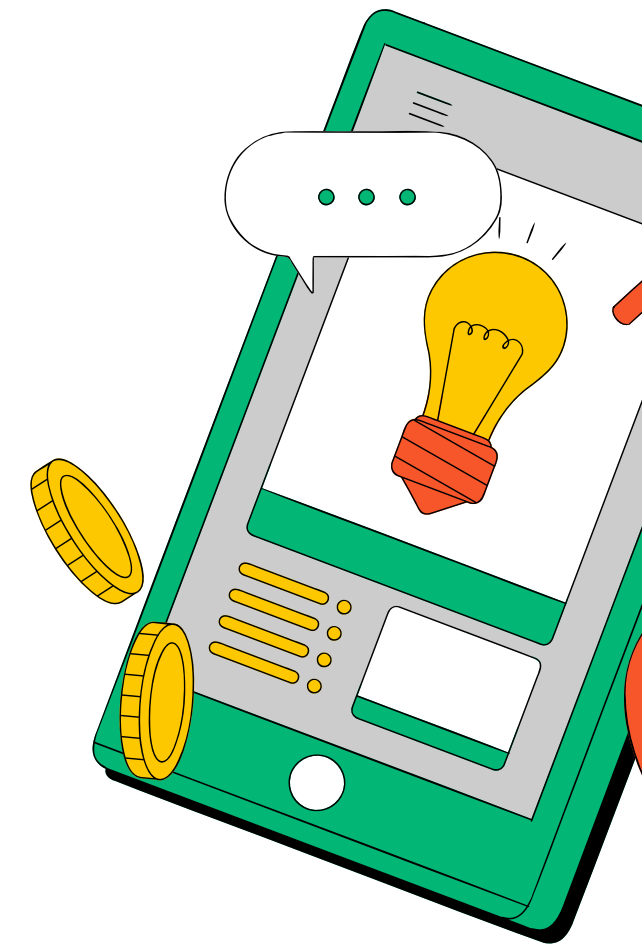
LOGISTIC REGRESSION BEST PARAMETERS:

```
'LEARNING_RATE': 0.01,  
'PENALTY': 'L2',  
'SOLVER': 'LIBLINEAR'
```



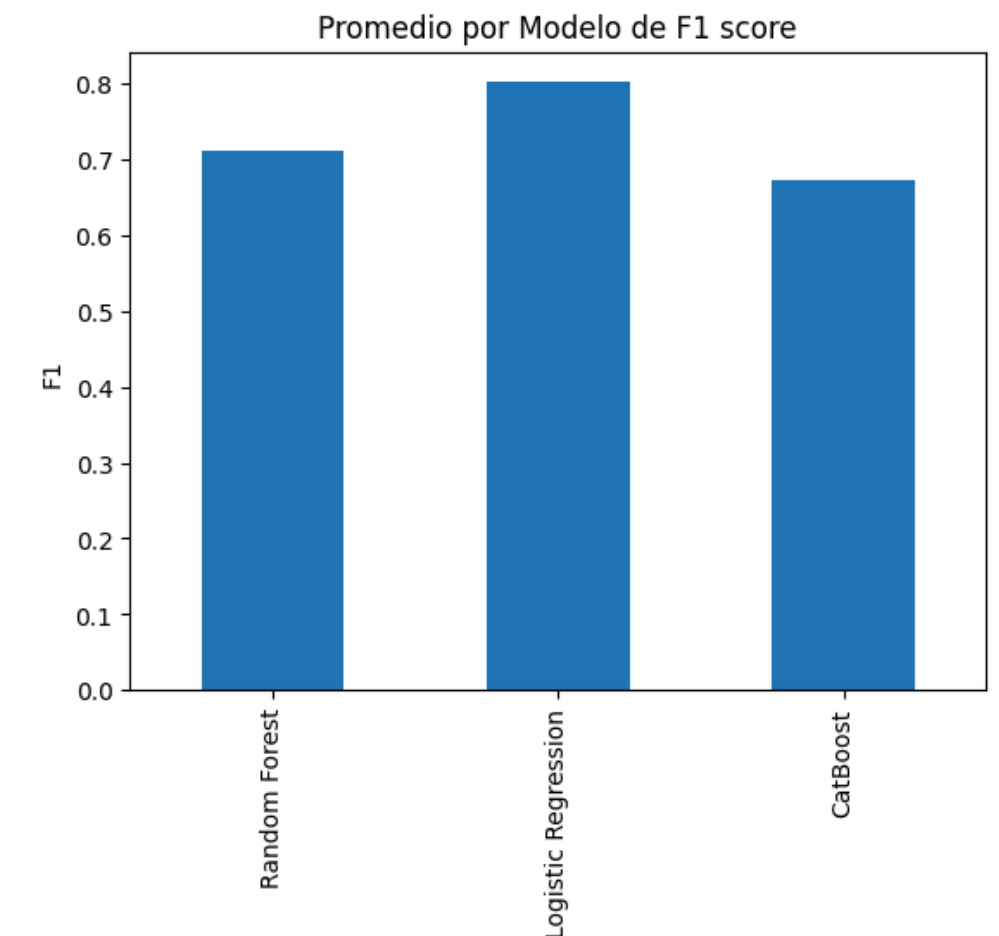
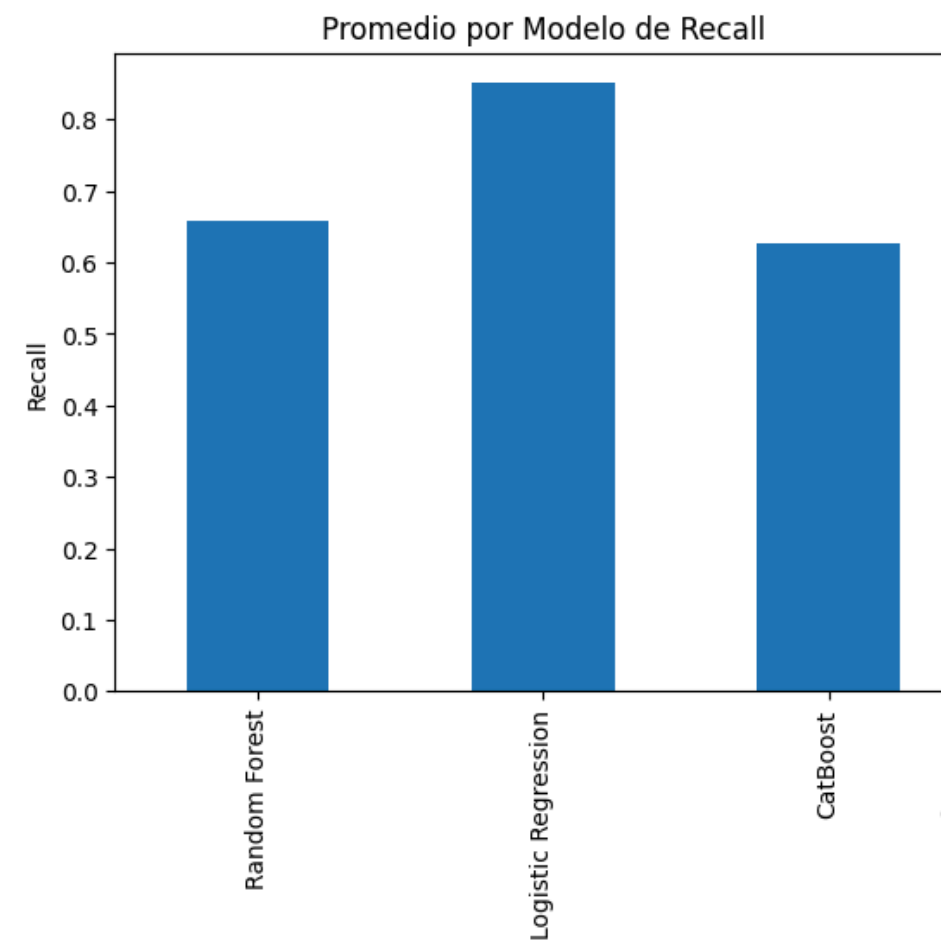
CATBOOST BEST PARAMETERS:

```
'DEPTH': 8,  
'L2_LEAF_REG': 1,  
'LEARNING_RATE': 0.01
```



CROSS VALIDATION - K-FOLDS ESTRATIFICADO

	Random Forest	Logistic Regression	CatBoost
0	0.661875	0.780334	0.641748
1	0.734330	0.781484	0.707303
2	0.795285	0.801035	0.785509
3	0.818757	0.800345	0.793441
4	0.759494	0.779056	0.720944
Precision df			
	Random Forest	Logistic Regression	CatBoost
0	0.964516	0.757829	0.992218
1	0.783562	0.751521	0.760284
2	0.774841	0.793792	0.755330
3	0.788066	0.780851	0.733273
4	0.900175	0.718611	0.977941
Recall df			
	Random Forest	Logistic Regression	CatBoost
0	0.341324	0.828767	0.291096
1	0.652968	0.845890	0.611872
2	0.836758	0.817352	0.849315
3	0.875429	0.838857	0.926857
4	0.587429	0.922286	0.456000
F1 df			
	Random Forest	Logistic Regression	CatBoost
0	0.504216	0.791712	0.450132
1	0.712329	0.795918	0.678052
2	0.804610	0.805399	0.799570
3	0.829453	0.808815	0.818778
4	0.710927	0.807808	0.621980



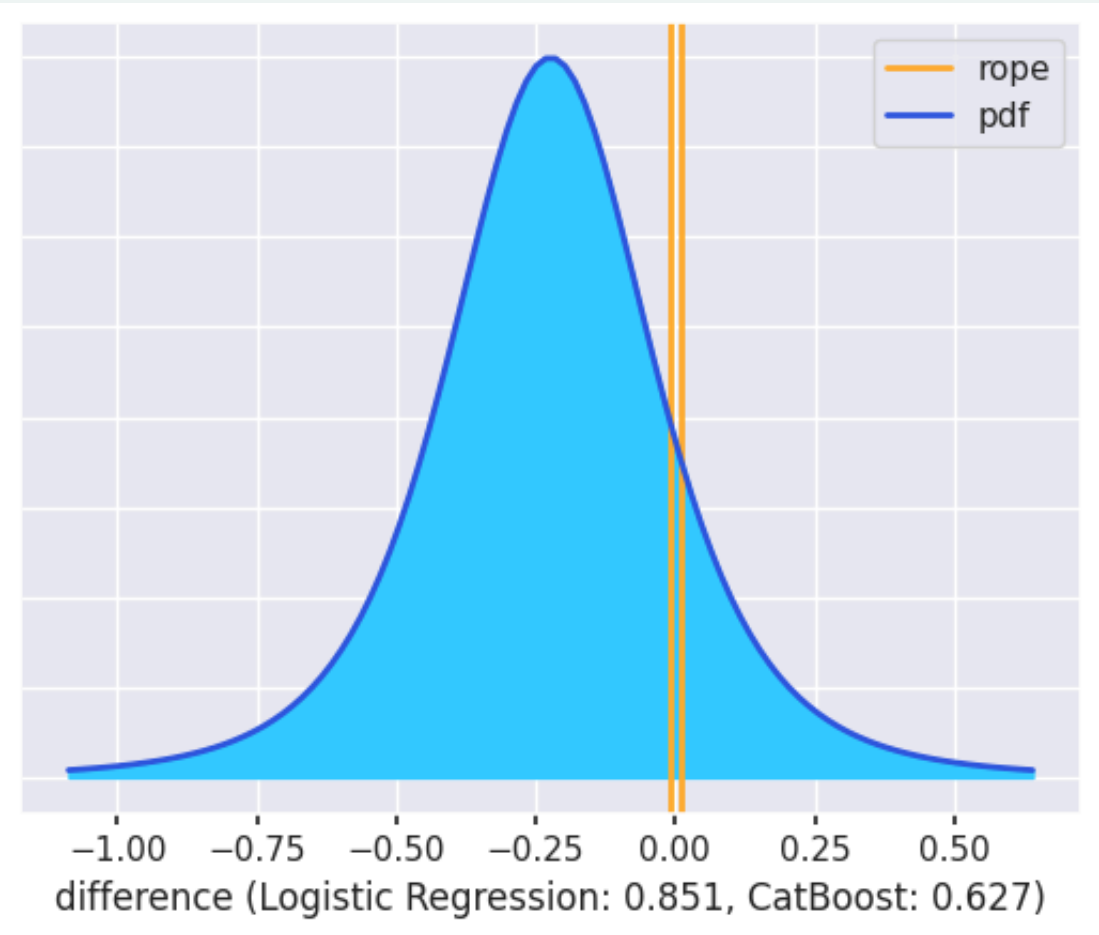
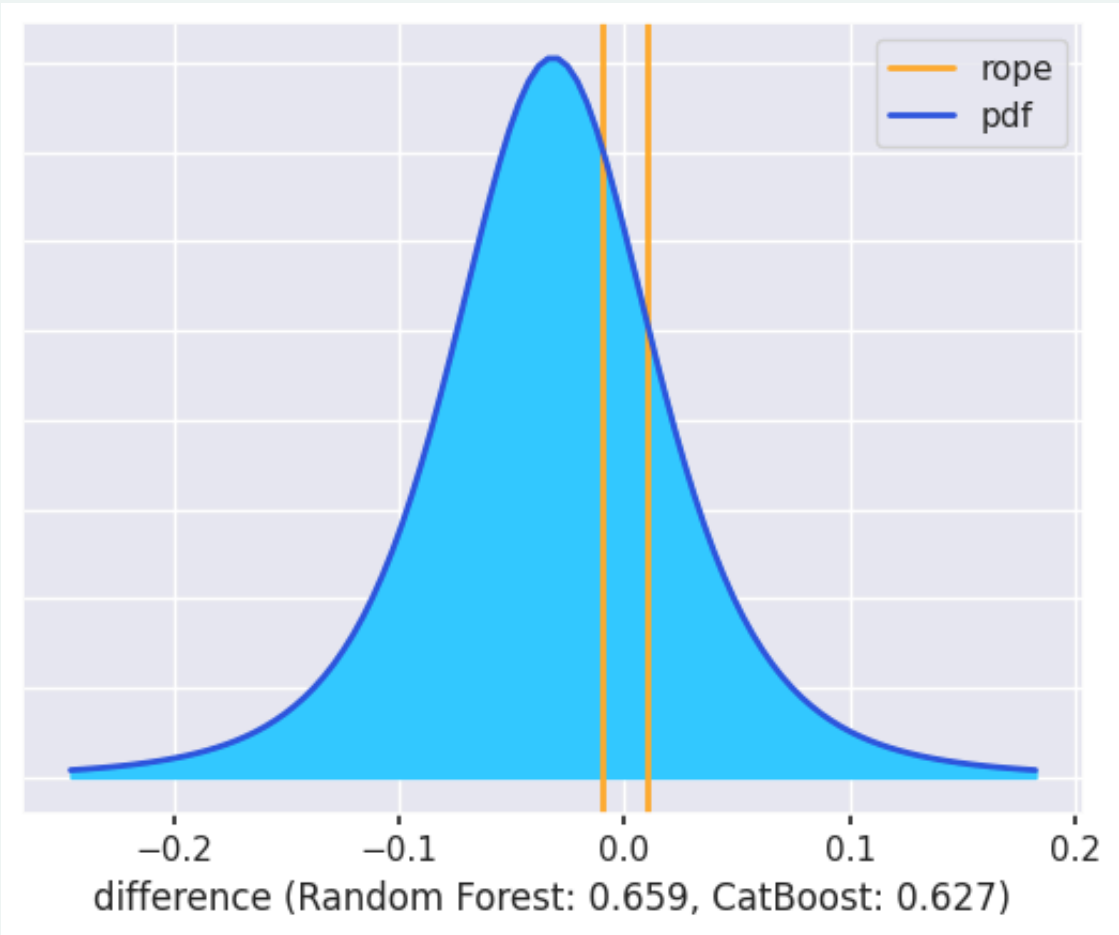
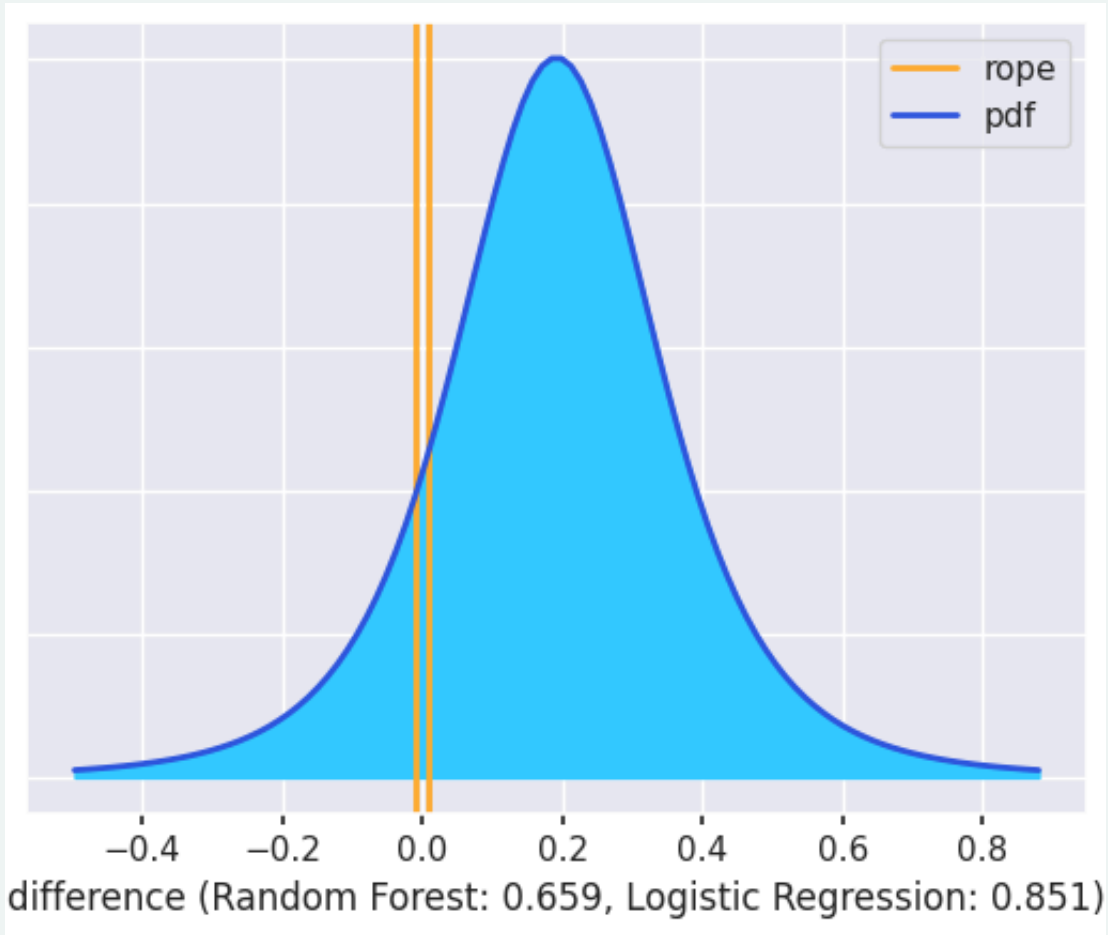


COMPARACIÓN BAYESIANA

Comparación bayesiana entre Random Forest y Logistic Regression en Recall:
(0.12420522888084852,
0.021203951140909227,
0.8545908199782423)

Comparación bayesiana entre Random Forest y CatBoost en Recall:
(0.6677609286371811,
0.12209863205878269,
0.21014043930403625)

Comparación bayesiana entre Logistic Regression y CatBoost en Recall:
(0.8406563361183341,
0.018710586839676635,
0.14063307704198924)

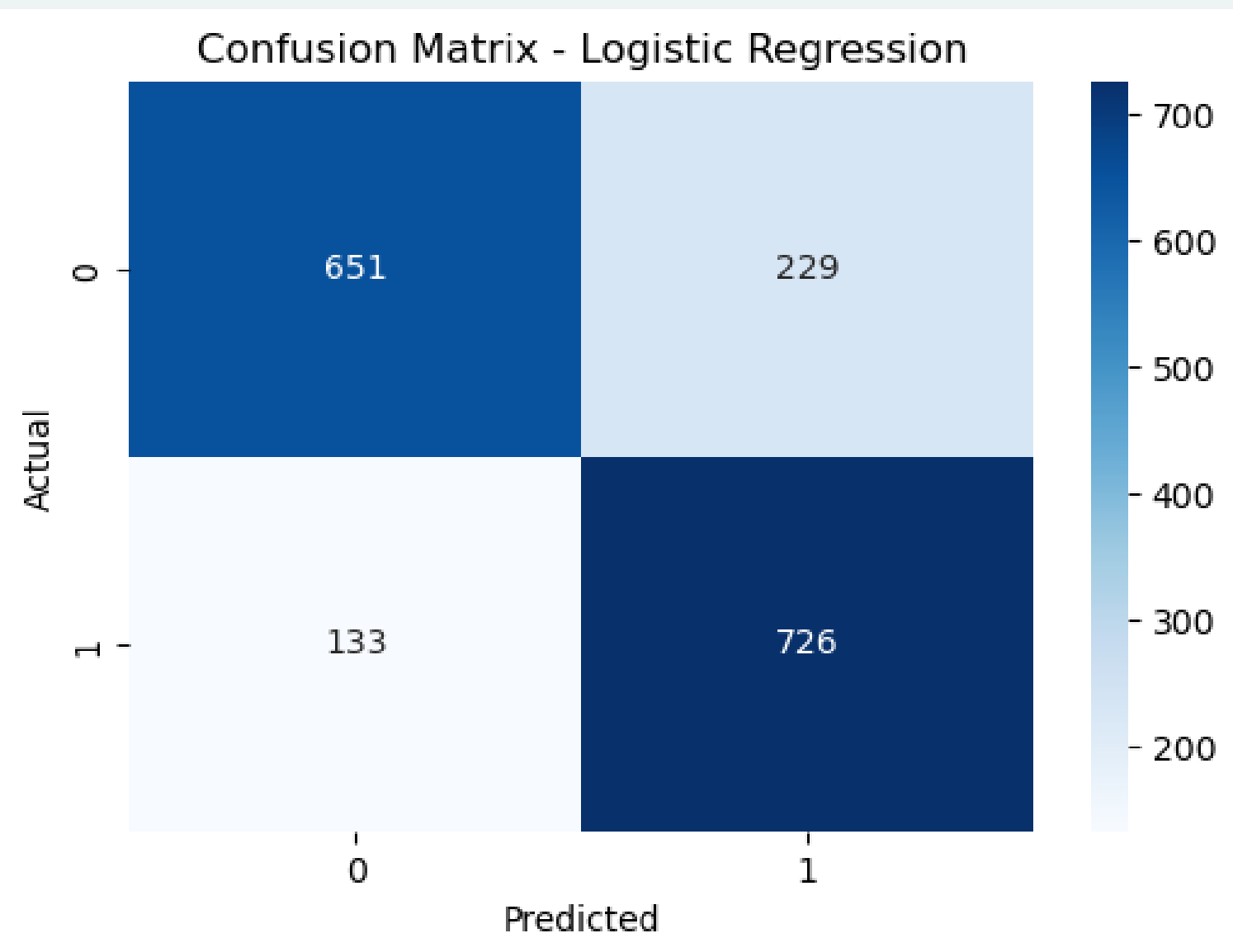




SELECCIÓN DE MODELO



Regresión Logística



El objetivo clave es reducir los falsos negativos y las falsas alarmas que son cruciales para minimizar las aprobaciones incorrectas de viajes.

El random forest y la regresión logística tienen un desempeño de manera similar, pero la regresión logística se destaca por ser ligeramente mejor en el recall, que es más crítica para la seguridad de los pasajeros. Por tanto, la **regresión logística se considera el mejor modelo** para las predicciones.



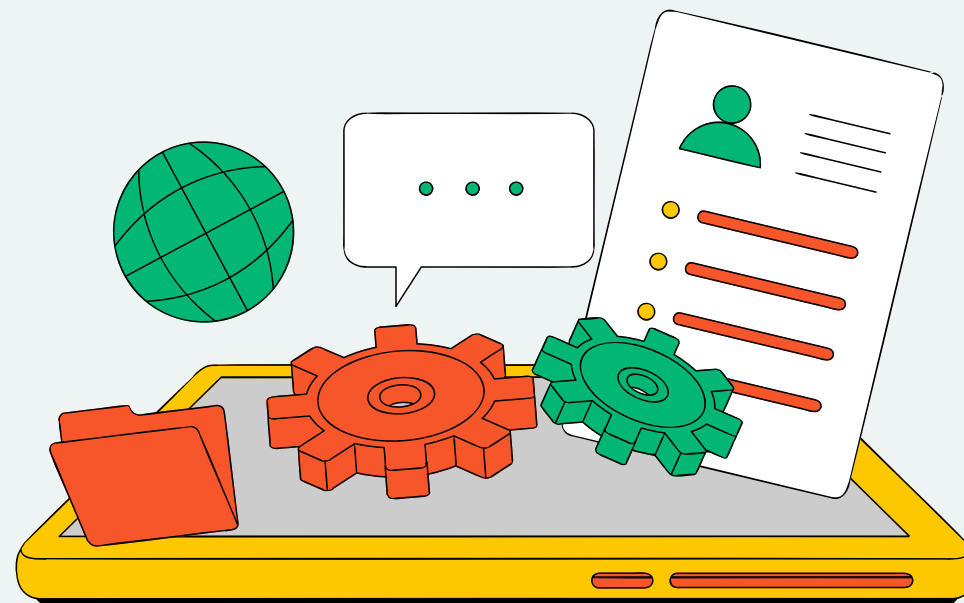


APLICACIÓN

Backend

Frontend

Demostración en vivo



CONCLUSIONES

TRABAJO A FUTURO

01

Mejoras de aplicación:

- Mejoras de seguridad
- Guardar en BD
- Retroalimentación de App

02

Mejorar el análisis de datos:

- Análisis de Nombres (Bayes)

03

Mejorar modelos:

- Análisis de importancia de columnas
- Mejora de hiperparámetros
- Diferentes métricas de evaluación








PUNTAJE EN KAGGLE

Puntaje

0.80617

182	bivba		0.80617	6	13d
183	Gustavo Téllez Mireles		0.80617	14	5m
184	aramosc03		0.80617	10	13h

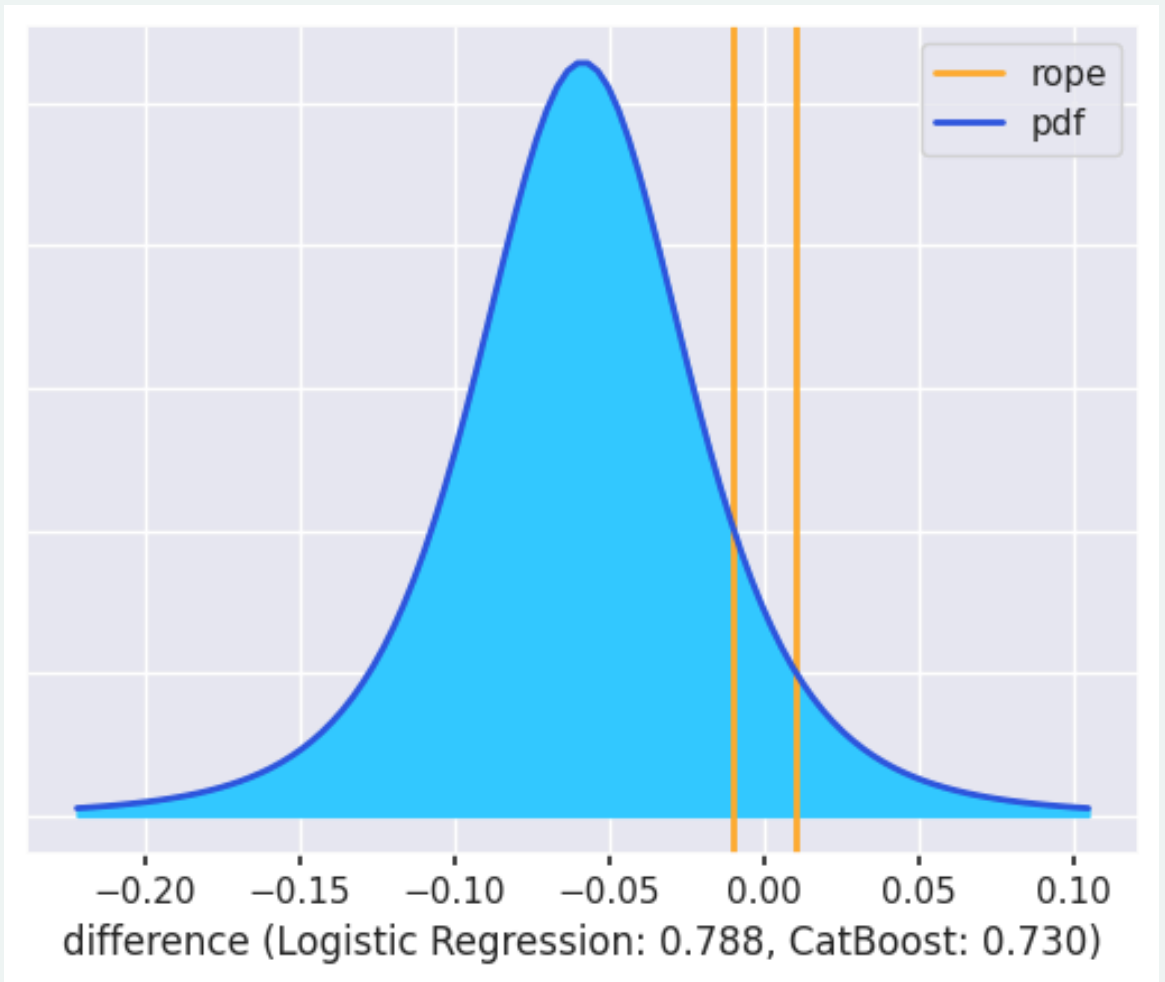
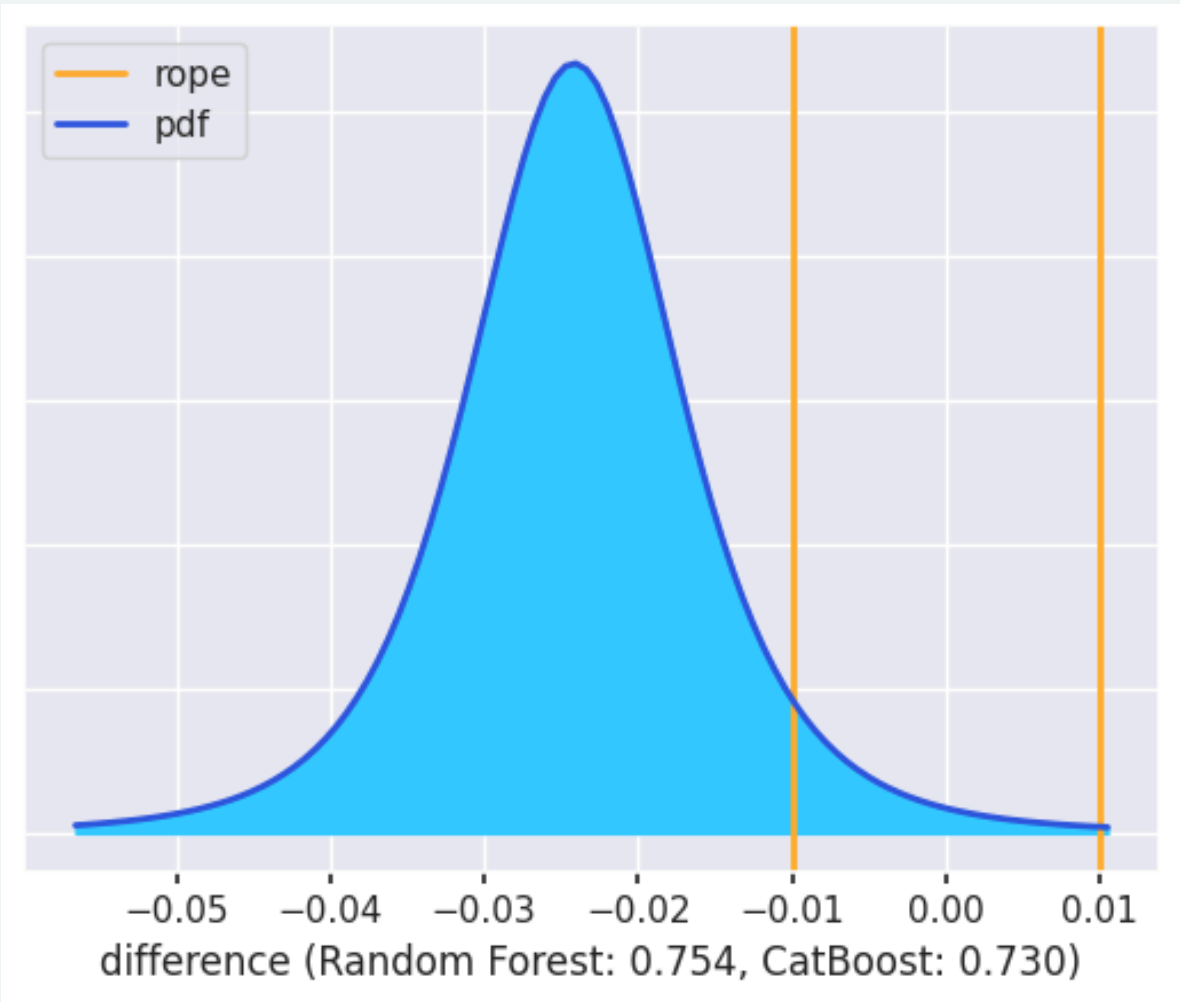
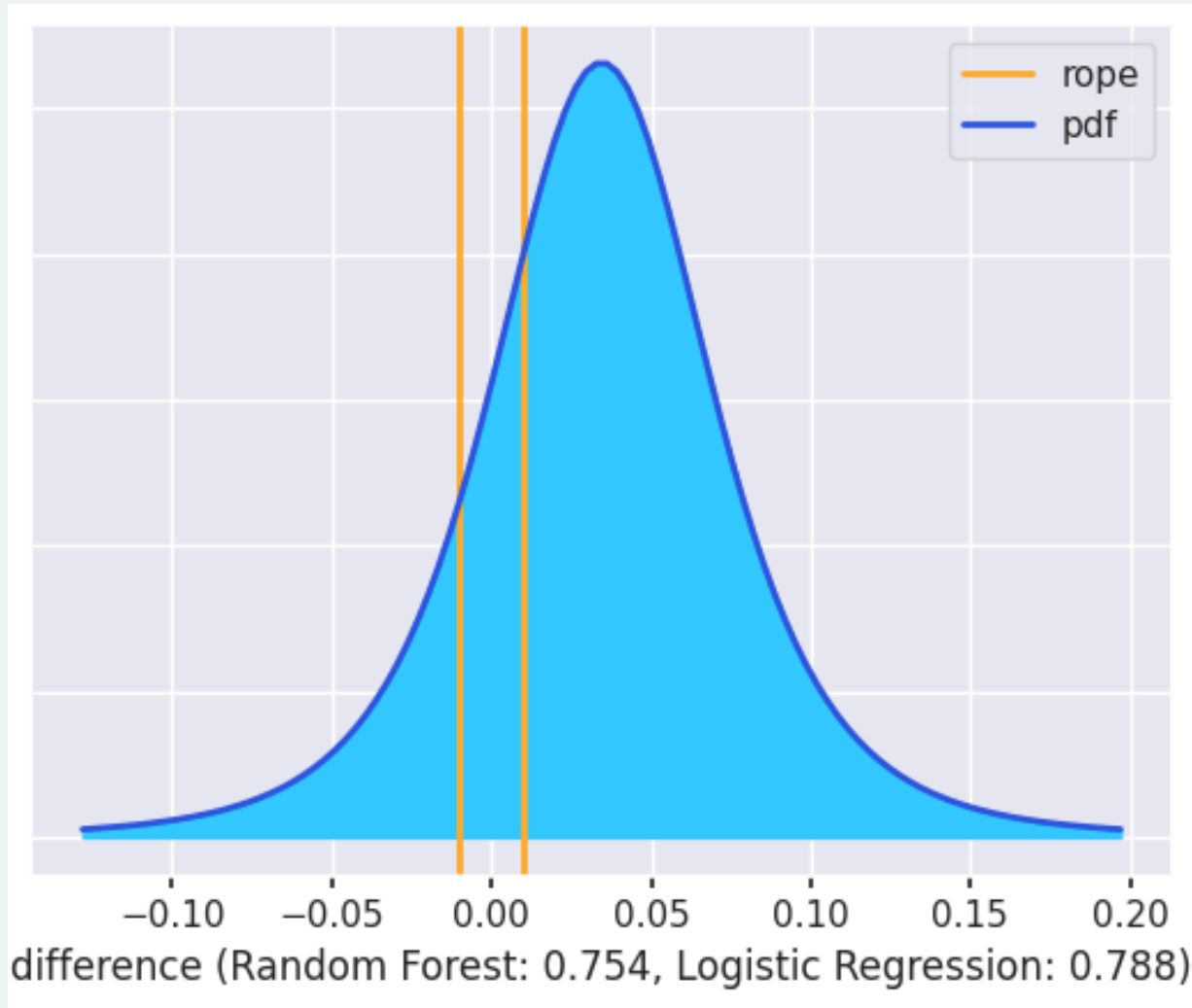


APÉNDICES - COMPARACIÓN BAYESIANA ACCURACY

COMPARACIÓN BAYESIANA ENTRE RANDOM FOREST Y LOGISTIC REGRESSION EN ACCURACY: (0.13780601585031496, 0.124939662619866, 0.7372543215298191)

COMPARACIÓN BAYESIANA ENTRE RANDOM FOREST Y CATBOOST EN ACCURACY: (0.9427923880726922, 0.05305667868893682, 0.004150933238371013)

COMPARACIÓN BAYESIANA ENTRE LOGISTIC REGRESSION Y CATBOOST EN ACCURACY: (0.8791294869983376, 0.05847154241225583, 0.06239897058940658)

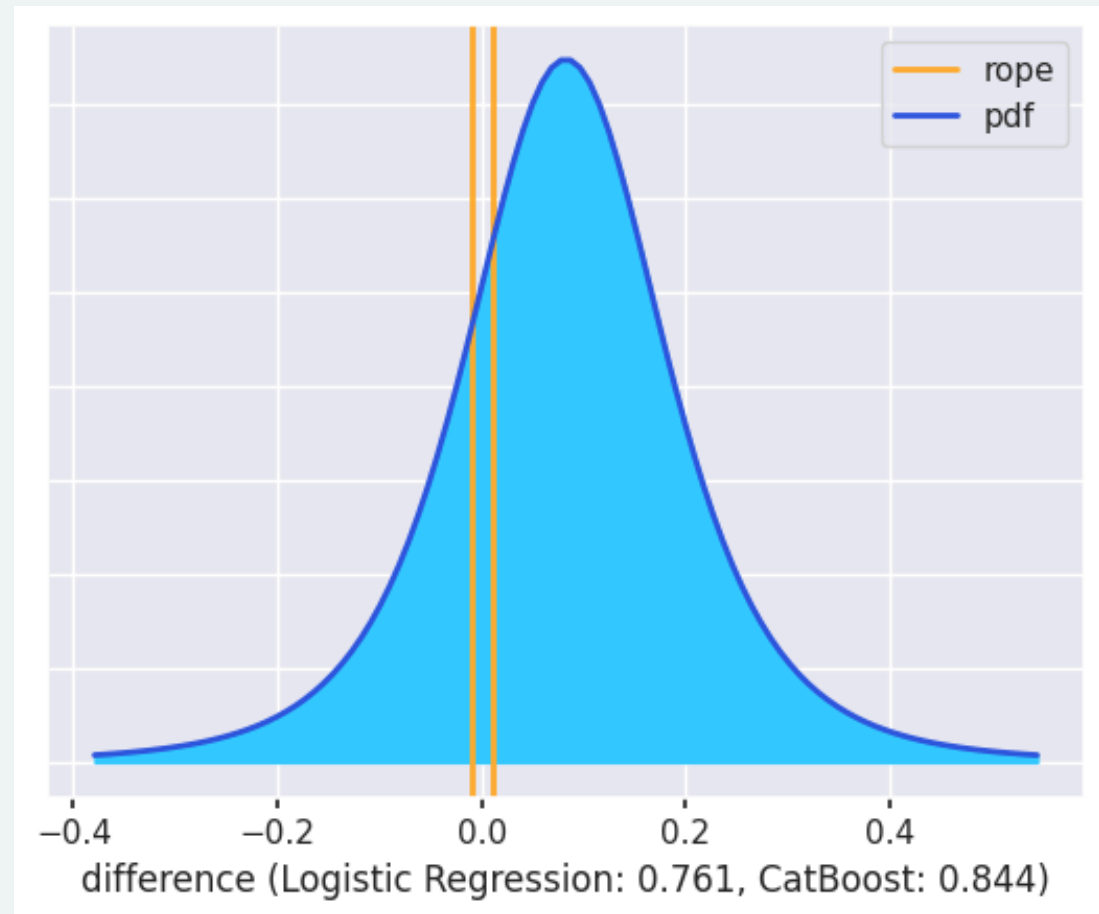
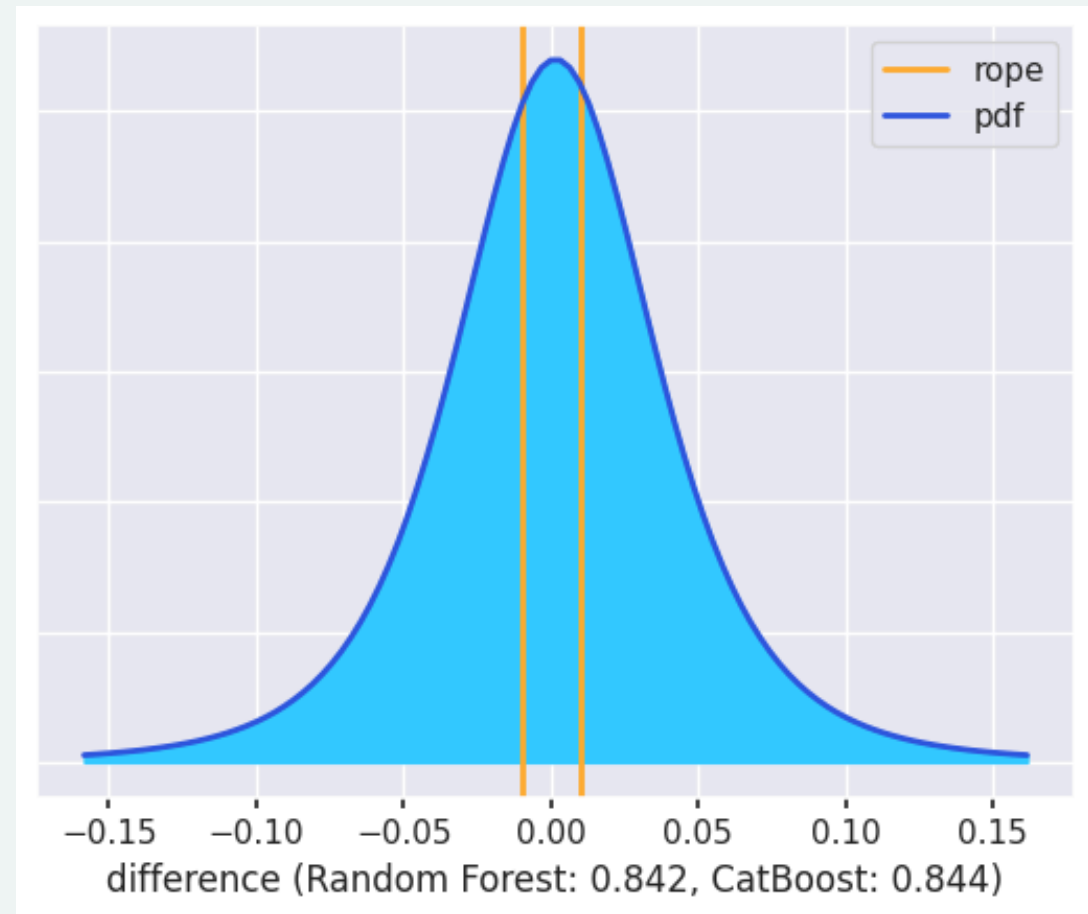
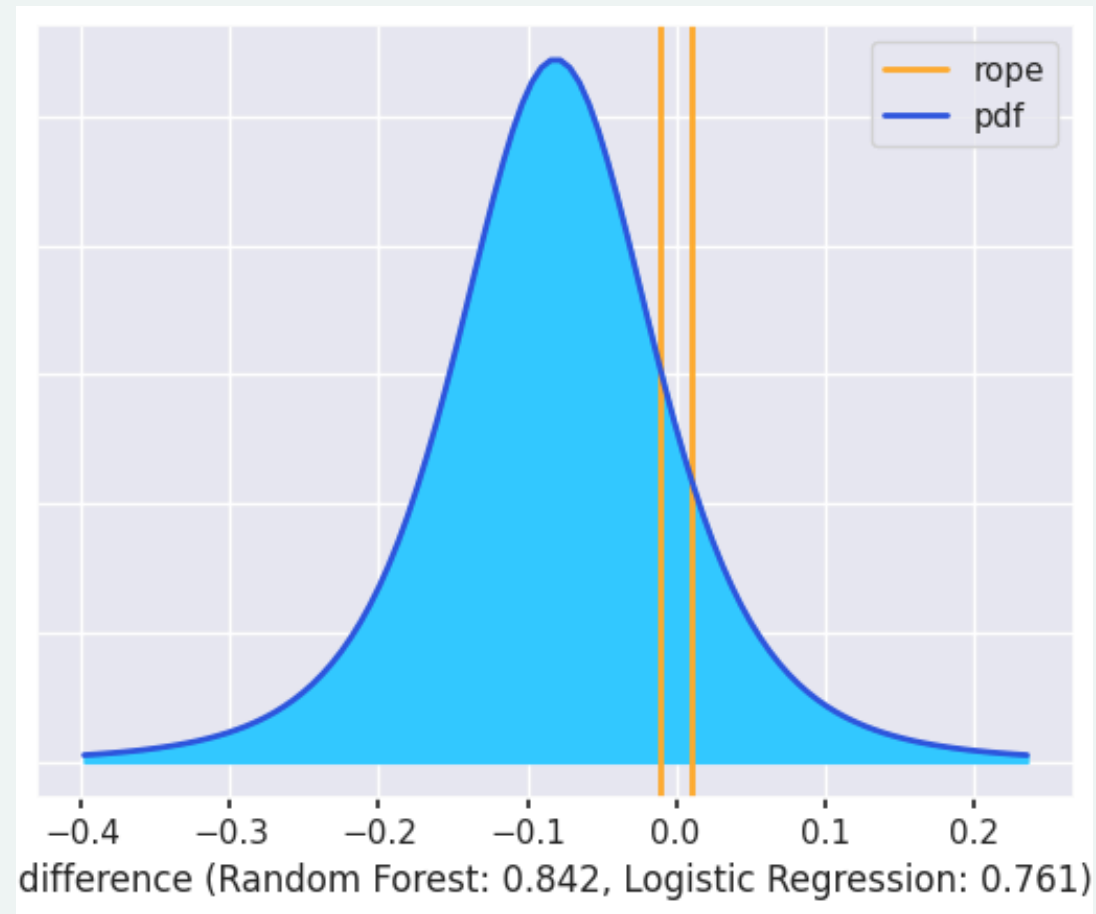


APÉNDICES - COMPARACIÓN BAYESIANA PRECISION

COMPARACIÓN BAYESIANA ENTRE RANDOM FOREST Y LOGISTIC REGRESSION EN PRECISION: (0.8194771402108476, 0.05196151387844672, 0.12856134591070567)

COMPARACIÓN BAYESIANA ENTRE RANDOM FOREST Y CATBOOST EN PRECISION: (0.3778384975552414, 0.21198712636854167, 0.410174376076217)

COMPARACIÓN BAYESIANA ENTRE LOGISTIC REGRESSION Y CATBOOST EN PRECISION: (0.2041461876340363, 0.05055130580849121, 0.7453025065574725)

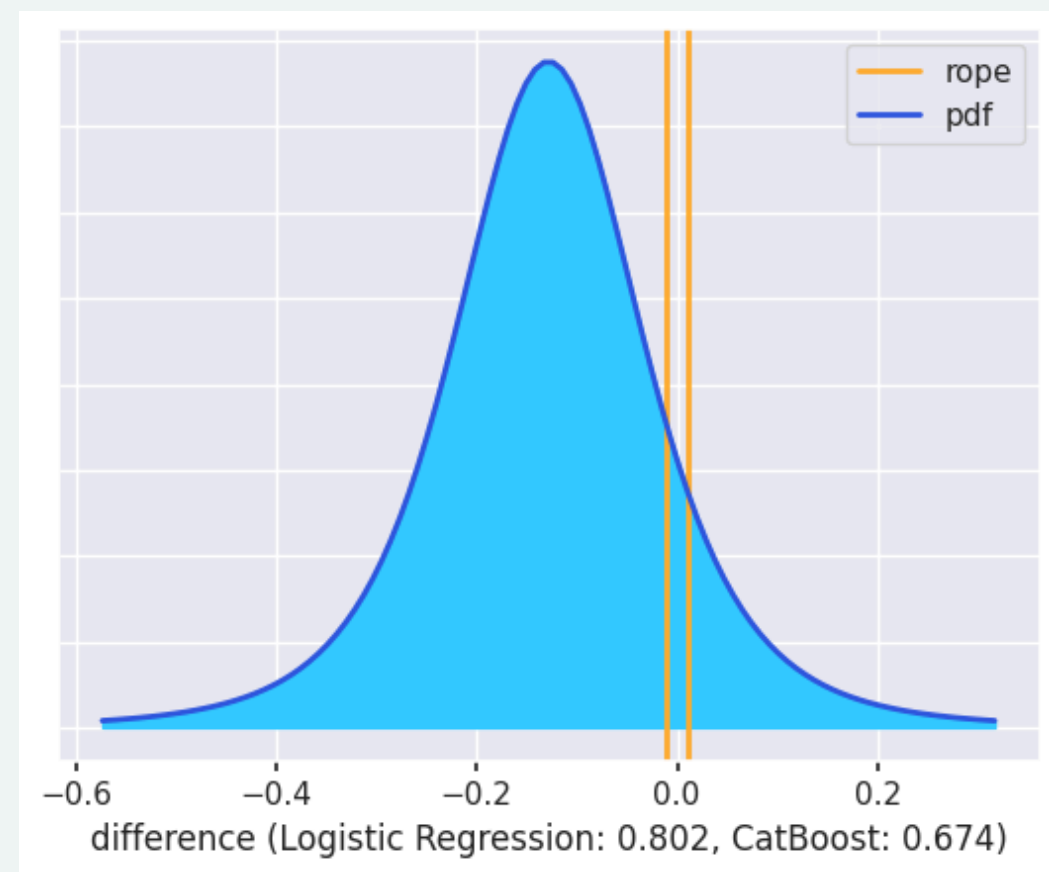
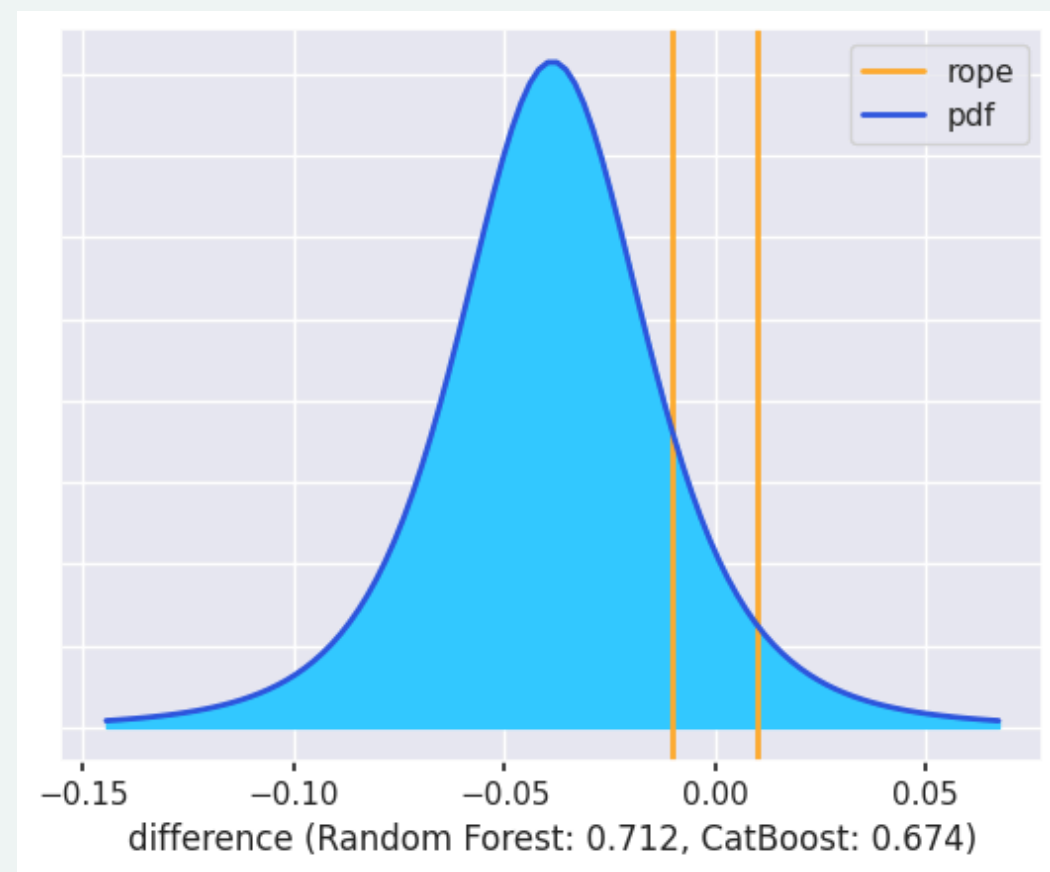
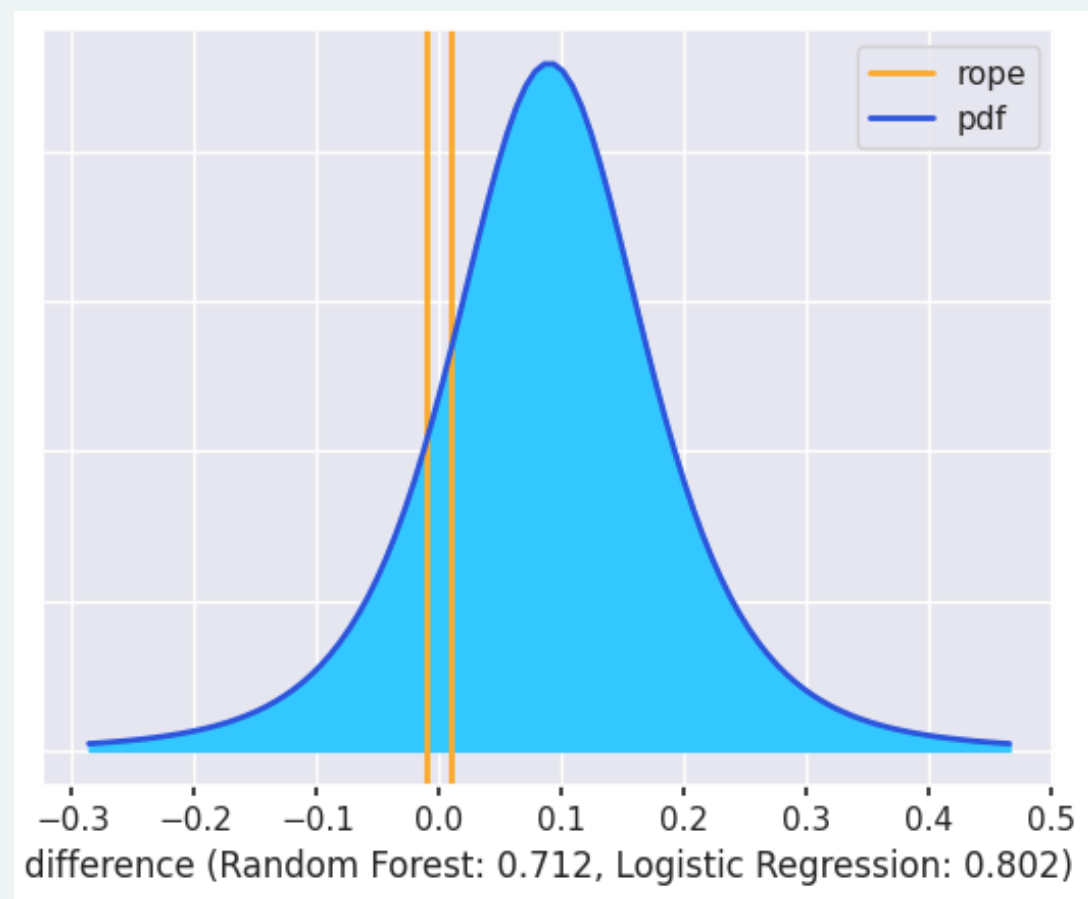


APÉNDICES - COMPARACIÓN BAYESIANA F1 SCORE

COMPARACIÓN BAYESIANA ENTRE RANDOM FOREST Y LOGISTIC REGRESSION EN F1 SCORE: (0.14477072146834083, 0.04763150228940427, 0.8075977762422549)

COMPARACIÓN BAYESIANA ENTRE RANDOM FOREST Y CATBOOST EN F1 SCORE: (0.8592994488646477, 0.08967949586077606, 0.05102105527457623)

COMPARACIÓN BAYESIANA ENTRE LOGISTIC REGRESSION Y CATBOOST EN F1 SCORE: (0.8558300774971028, 0.03123660527688632, 0.11293331722601085)



APPENDICES - FUNCIONAMIENTO SERVIDOR Y MODELO



```
1 # Load the trained model
2 with open('finalized_model.sav', 'rb') as file:
3     model = pickle.load(file)
```



```
1 def predict(df):
2     # Make predictions
3     prediction = model.predict(df)
4     return prediction
```

```
{
  "PASSENGERID": "G_1",
  "CRYOSLEEP": TRUE,
  "VIP": FALSE,
  "HOMEPLANET": "EARTH",
  "DESTINATION": "TRAPPIST-1E",
  "DECK": "A",
  "CABIN": "A/123/P",
  "ROOMSERVICE": 100,
  "FOODCOURT": 200,
  "SHOPPINGMALL": 10450,
  "SPA": 300,
  "VRDECK": 400,
  "AGE": 25,
  "NAME": "GUSTBAO"
}
```



```
1 def process_df(df):
2     # Binarize the booleans
3     df['CryoSleep'] = df['CryoSleep'].replace({False: 0, True: 1})
4     df['VIP'] = df['VIP'].replace({False: 0, True: 1})
5
6     # Create new columns
7     df[['GroupNum', 'PassNum']] = df['PassengerId'].str.split('_', expand=True)
8     df['GroupNum'] = pd.to_numeric(df['GroupNum'], errors='coerce')
9     df['PassNum'] = pd.to_numeric(df['PassNum'], errors='coerce')
10    df.drop(columns=["PassengerId"], inplace=True)
11
12    df['TotalSpent'] = df[["RoomService", "FoodCourt", "ShoppingMall", "Spa", "VRDeck"]].sum(axis=1)
13
14    # Split Cabin into components
15    df[['Deck', 'CabinNumber', 'CabinSide']] = df['Cabin'].str.split('/', expand=True)
16    df['CabinNumber'] = pd.to_numeric(df['CabinNumber'], errors='coerce') # Ensure CabinNumber is numeric
17    df.drop(columns=["Cabin"], inplace=True)
18
19    # Label encoding for categorical data
20    df['HomePlanet'] = df['HomePlanet'].replace({'Earth': 0, 'Mars': 1, 'Europa': 2})
21    df['Destination'] = df['Destination'].replace({'TRAPPIST-1e': 0, '55 Cancri e': 1, 'PSO J318.5-22': 2})
22    df['Deck'] = df['Deck'].replace({'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'T': 7})
23    df['CabinSide'] = df['CabinSide'].replace({'P': 0, 'S': 1})
24
25    # Handle special cases for missing values (if any were theoretically provided)
26    df.loc[(df['HomePlanet'] == 0) & (df['Deck'].isnull()), 'Deck'] = 6
27    df.loc[(df['TotalSpent'] > 0) & (df['CryoSleep'].isnull()), 'CryoSleep'] = 0
28    df.loc[(df['CryoSleep'] == 1) & (df['VIP'].isnull()), 'VIP'] = 0
29
30    # Create interaction columns
31    df['HomePlanet_Destination_Interaction'] = df['HomePlanet'] * df['Destination']
32
33    # Drop unnecessary columns
34    df.drop(columns=["Name"], inplace=True)
35
36    # Ensure all columns are numeric
37    df = df.apply(pd.to_numeric, errors='coerce')
38
39    return df
```