EXPLORACION Y APLICACIÓN DE MACHE LEARNING PARA ANÁLISIS DE DATOS (CASO SPACE X)

De Willer Gianni Torrico Arispe

RESUMEN EJECUTIVO

- El proyecto se centra en la implementación de algoritmos de Machine Learning para la resolución de problemas complejos. Utilizando técnicas como Regresión Logística, Árboles de Decisión y k-Vecinos más Cercanos, se exploraron conjuntos de datos detallados relacionados con predicciones de vuelos espaciales. Se abordaron desafíos en la optimización de modelos y la visualización de datos para proporcionar un análisis predictivo preciso y comprensible.

INTRODUCCION

- El proyecto se enfoca en el análisis predictivo de vuelos espaciales mediante técnicas de Machine Learning. Con el objetivo de comprender y predecir el éxito de lanzamientos espaciales, se exploraron y aplicaron diversos algoritmos. Este trabajo se sumerge en la aplicación de Regresión Logística, Árboles de Decisión y k-Vecinos más Cercanos para modelar y prever resultados, ofreciendo un análisis detallado del rendimiento de estas técnicas en la predicción de resultados de misiones espaciales.

RECOPILACION Y MANEJO DE DATOS

1. Obtención y Análisis Inicial de los Datos de SpaceX:

- Se realizó una solicitud GET a la URL proporcionada para obtener datos de lanzamiento de SpaceX.
- Se identificó que varios campos eran IDs sin información descriptiva.
- Se utilizó nuevamente la API para obtener detalles adicionales de los lanzamientos utilizando las columnas `rocket`, `payloads`, `launchpad`, y `cores`.
- Se filtró el conjunto de datos conservando solo las características relevantes y la fecha de vuelo.
- Se eliminaron filas con múltiples núcleos o múltiples cargas útiles en un solo cohete.

Para hacer que los resultados JSON solicitados sean más consistentes, utilizaremos el siguiente objeto de respuesta estático para este proyecto:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

[17] ✓ 0.0s MagicPython

Eso indica que la solicitud fue exitosa, con un código de respuesta 200.

+ Code + Markdown

```
response.status_code
```

[18] ✓ 0.0s MagicPython

```
200
```

```
# Muestra las primeras filas del DataFrame
launch_data.head()
```

MagicPython

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003

RECOPILACION Y MANEJO DE DATOS

2. Filtrado de Datos para Incluir Solo Lanzamientos de Falcon 9:

- Se filtraron los datos para incluir solo los lanzamientos de Falcon 9.
- Se creó un nuevo DataFrame llamado `data_falcon9` para almacenar estos datos.

```
# Hint data['BoosterVersion'] != 'Falcon 1'
data_falcon9 = launch_data[launch_data['BoosterVersion'] != 'Falcon 1']
```

[89] MagicPython

Ahora que hemos eliminado algunos valores, deberíamos restablecer la columna de números de vuelo (FlightNumber).

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

[90] MagicPython

...

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	Reused
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	

```
data_falcon9.isnull().sum()
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       5
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad       26
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

RECOPILACION Y MANEJO DE DATOS

3. Tratamiento de Valores Faltantes:

- Se calculó el valor promedio de la columna `PayloadMass`.
- Se reemplazaron los valores faltantes (`np.nan`) en `PayloadMass` con su valor promedio.
- Es importante tener en cuenta que al reemplazar los valores NaN en `PayloadMass`, se generó un aviso `SettingWithCopyWarning` debido a cómo se realizó la operación. Es recomendable revisar la documentación de Pandas para abordar estas advertencias si es necesario.

Calcula a continuación la media para la columna `PayloadMass` utilizando `.mean()`. Luego, usa la media y la función `.replace()` para reemplazar los valore `np.nan` en los datos con la media que calculaste.

```
# Calcula el valor promedio de la columna PayloadMass
mean_payload_mass = data_falcon9['PayloadMass'].mean()

# Reemplaza los valores np.nan con su valor promedio
data_falcon9['PayloadMass'].replace(np.nan, mean_payload_mass, inplace=True)
```

4] ✓ 0.0s MagicPyth

```
C:\Users\wille\AppData\Local\Temp\ipykernel_5240\2873490337.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_falcon9['PayloadMass'].replace(np.nan, mean_payload_mass, inplace=True)
```

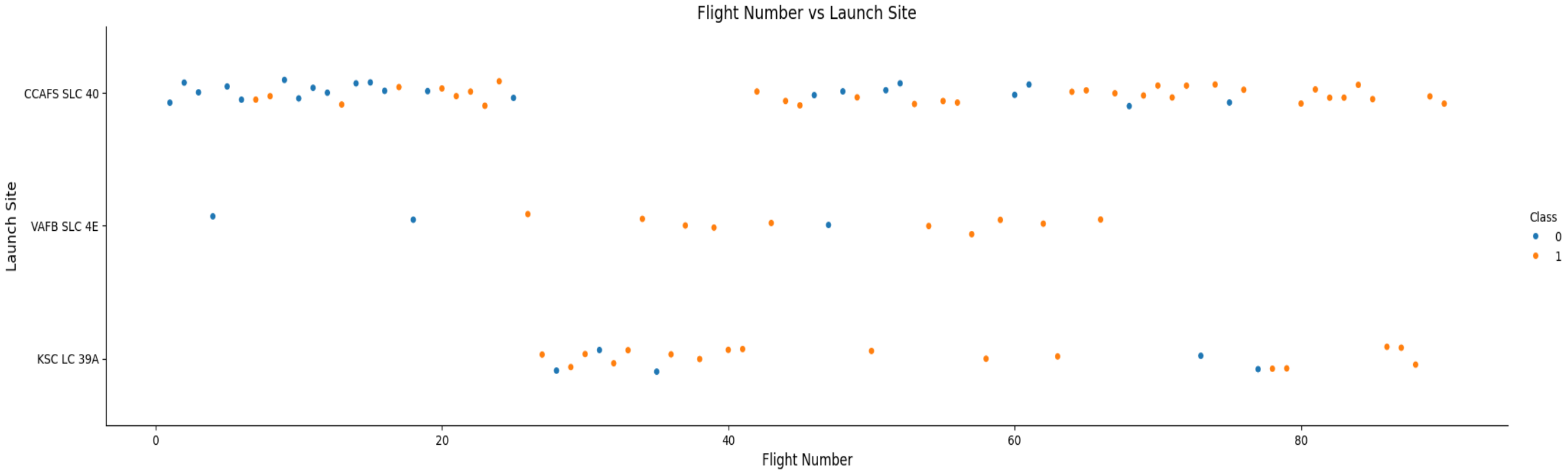
```
data_falcon9
```

5] ✓ 0.0s MagicPyth

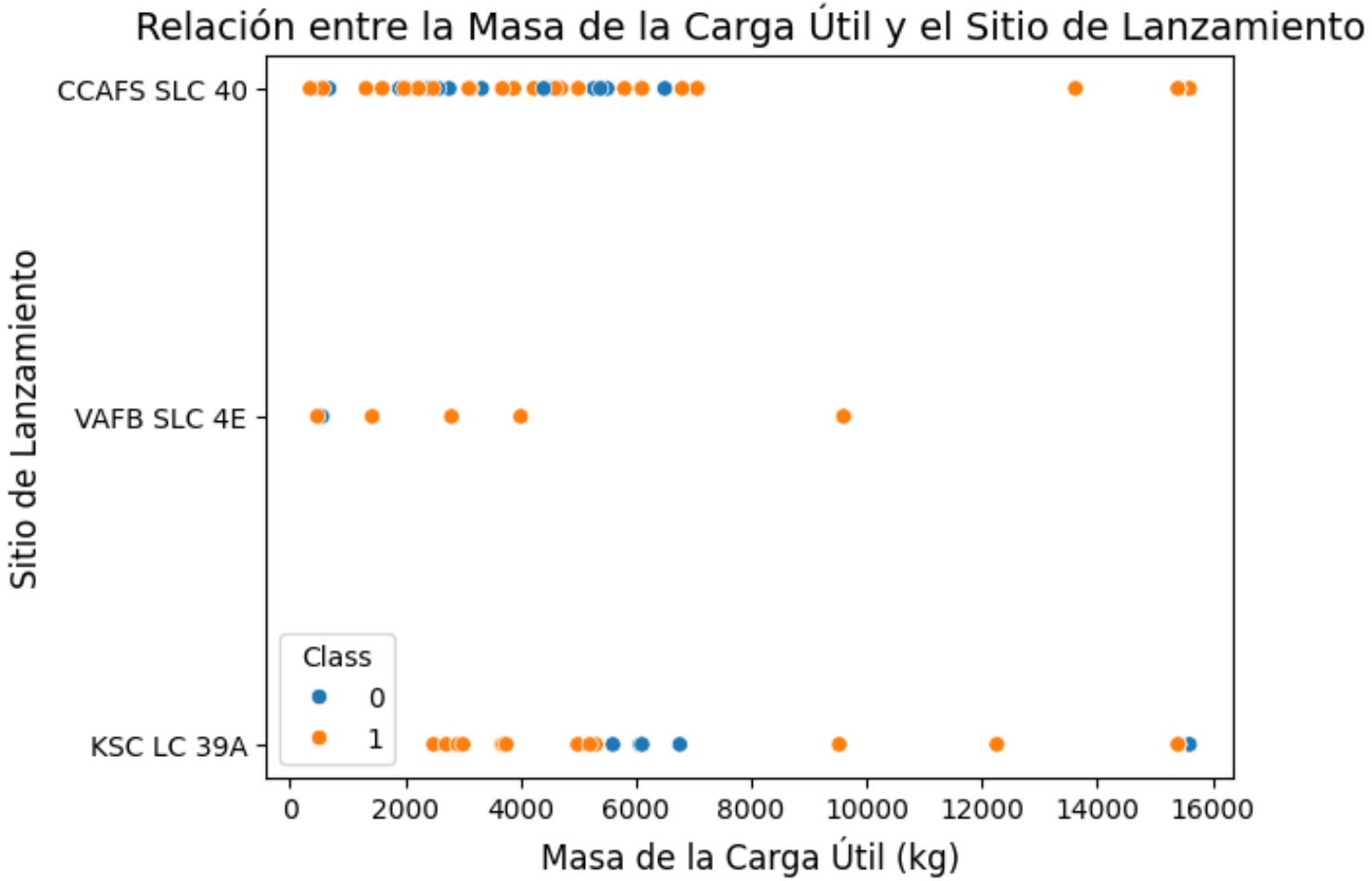
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	Reu
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	

ANALISIS EXPLORATORIO DE DATOS

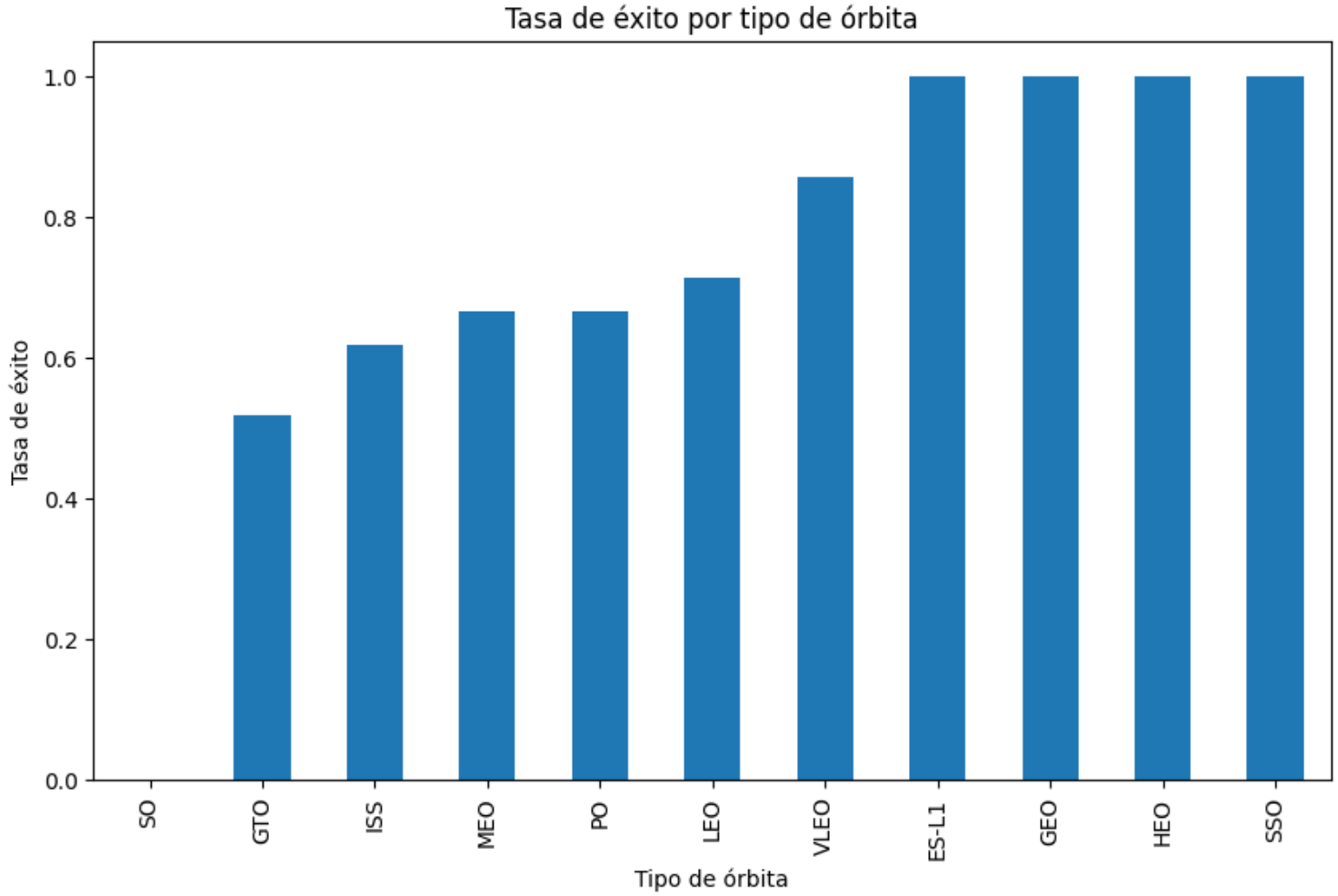
Visualización de la relación entre el número de vuelo y el sitio de lanzamiento.



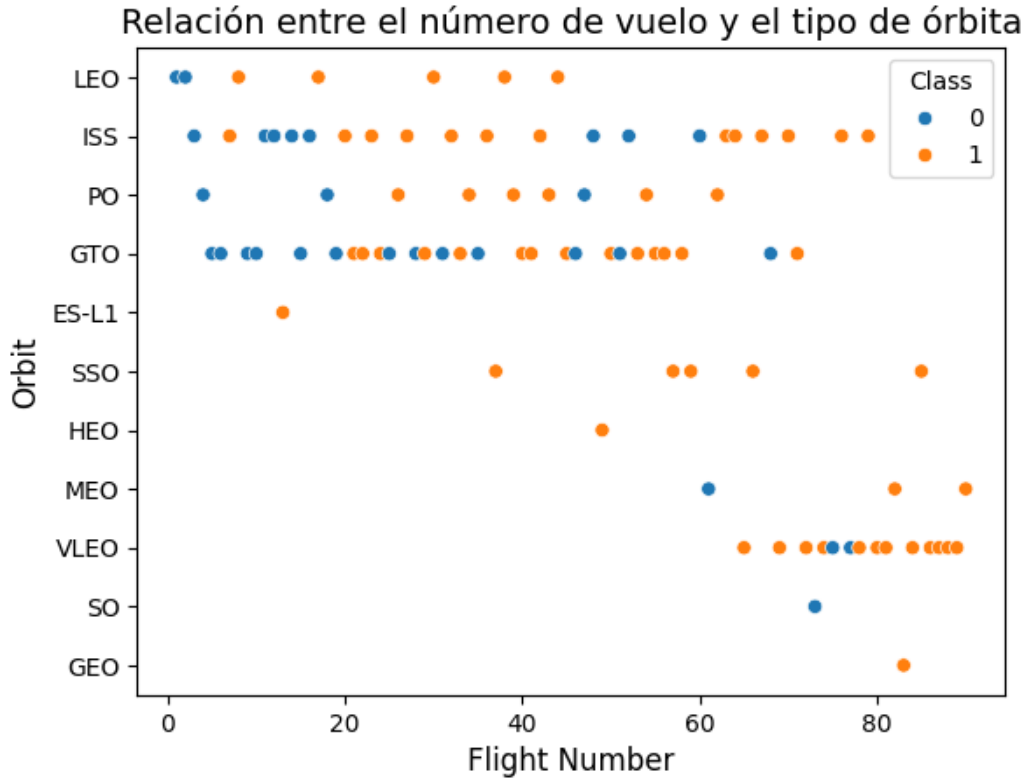
ANALISIS EXPLORATORIO DE DATOS



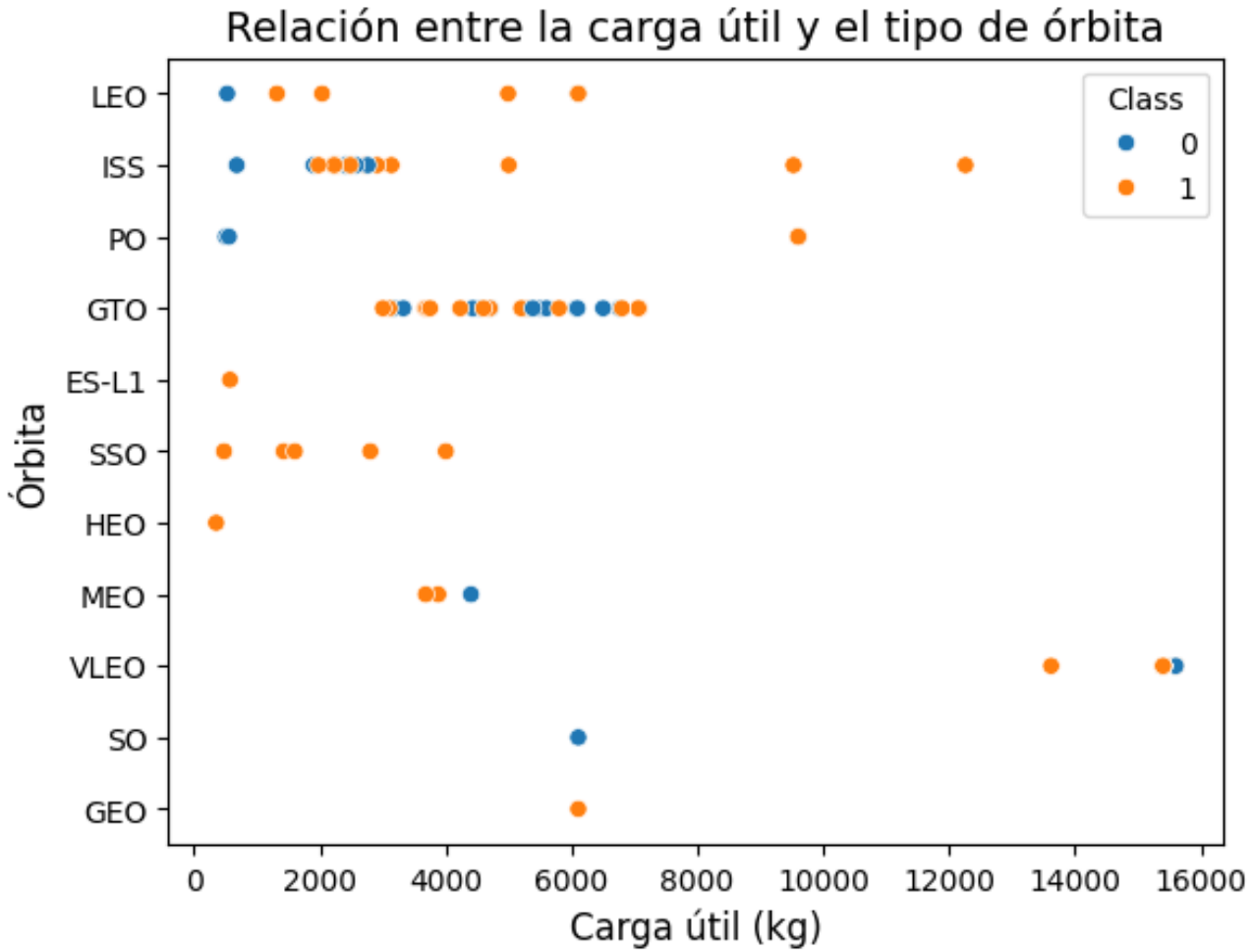
ANALISIS EXPLORATORIO DE DATOS



ANALISIS EXPLORATORIO DE DATOS



ANALISIS EXPLORATORIO DE DATOS



ANALISIS EXPLORATORIO DE DATOS

Análisis Exploratorio de Datos SQL

```
# Muestra los nombres de las columnas
%sql PRAGMA table_info(SPACEXTABLE);
```

```
 * sqlite:///my_data1.db
Done.
```

cid	name	type	notnull	dflt_value	pk
0	Date	TEXT	0	None	0
1	Time (UTC)	TEXT	0	None	0
2	Booster_Version	TEXT	0	None	0
3	Launch_Site	TEXT	0	None	0
4	Payload	TEXT	0	None	0
5	PAYLOAD_MASS__KG_	INT	0	None	0
6	Orbit	TEXT	0	None	0
7	Customer	TEXT	0	None	0
8	Mission_Outcome	TEXT	0	None	0
9	Landing_Outcome	TEXT	0	None	0

Análisis Exploratorio de Datos SQL

Mostrar 5 registros donde los sitios de lanzamiento comiencen con la cadena 'CCA'

```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

MagicPytho

* sqlite:///my_data1.db
Done.

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Análisis Exploratorio de Datos SQL

Muestra el peso total de la carga útil transportada por cohetes lanzados por NASA (CRS).

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE '%NASA (CRS)%';
```

```
 * sqlite:///my_data1.db
Done.
```

SUM(PAYLOAD_MASS__KG_)
48213

Análisis Exploratorio de Datos SQL

Mostrar el promedio de la masa de carga útil transportada por la versión del propulsor F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';
```

```
 * sqlite:///my_data1.db
Done.
```

AVG(PAYLOAD_MASS__KG_)
2928.4

Análisis Exploratorio de Datos SQL

Enumera la fecha en la que se logró el primer resultado exitoso de aterrizaje en una plataforma terrestre.

Consejo: Usa la función ***min****.*

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%ground%' AND Landing_Outcome LIKE '%Success%';
```

```
 * sqlite:///my_data1.db
Done.
```

MIN(Date)
2015-12-22

Análisis Exploratorio de Datos SQL

Lista los nombres de los propulsores que han tenido éxito en el barco no tripulado y tienen una masa de carga útil mayor a 4000 pero menor a 6000

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome='Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

13]

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Análisis Exploratorio de Datos SQL

Lista el número total de resultados exitosos y fallidos de misiones.

```
:T COUNT(CASE WHEN Mission_Outcome = 'Success' THEN 1 END) AS Successful_Outcomes, COUNT(CASE WHEN Mission_Outcome = 'Failure' THEN 1 END) AS Failure_Outcomes FROM SPACEXTABLE;
```

[14]

[14]

MagicPy

```
 * sqlite:///my_data1.db
Done.
```

Successful_Outcomes	Failure_Outcomes
98	0

+ Code + Markdown

Análisis Exploratorio de Datos SQL

Enumera los nombres de las `booster_versions` que han transportado la carga útil máxima. Utiliza una subconsulta.

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE );
```

5]

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

Análisis Exploratorio de Datos SQL

Lista los registros que mostrarán los nombres de los meses, los fallos en los aterrizajes en la plataforma flotante, las versiones del cohete y el lugar de lanzamiento para los meses del año 2015.

Nota: SQLite no admite nombres de meses. Por lo tanto, necesitas usar `substr(Date, 6,2)` como mes para obtener los meses y `substr(Date,0,5)='2015'` para el año.

```
%sql SELECT substr(Date, 6, 2) AS month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date, 0, 5) = '2015' AND Landing_Outcome LIKE '%Failure (drone s
```

MagicPython

```
 * sqlite:///my_data1.db
Done.
```

month	Landing_Outcome	Booster_Version	Launch_Site
10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Análisis Exploratorio de Datos SQL

Clasifica el recuento de resultados de aterrizaje (como Falla (plataforma flotante) o Éxito (plataforma terrestre)) entre las fechas 2010-06-04 y 2017-03-20, en orden descendente.

```
%sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' AND (Landing_Outcome = 'Failure (drone ship)' OR Landing_Ou
```

MagicPythor

```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	Outcome_Count
Success (ground pad)	5
Failure (drone ship)	5

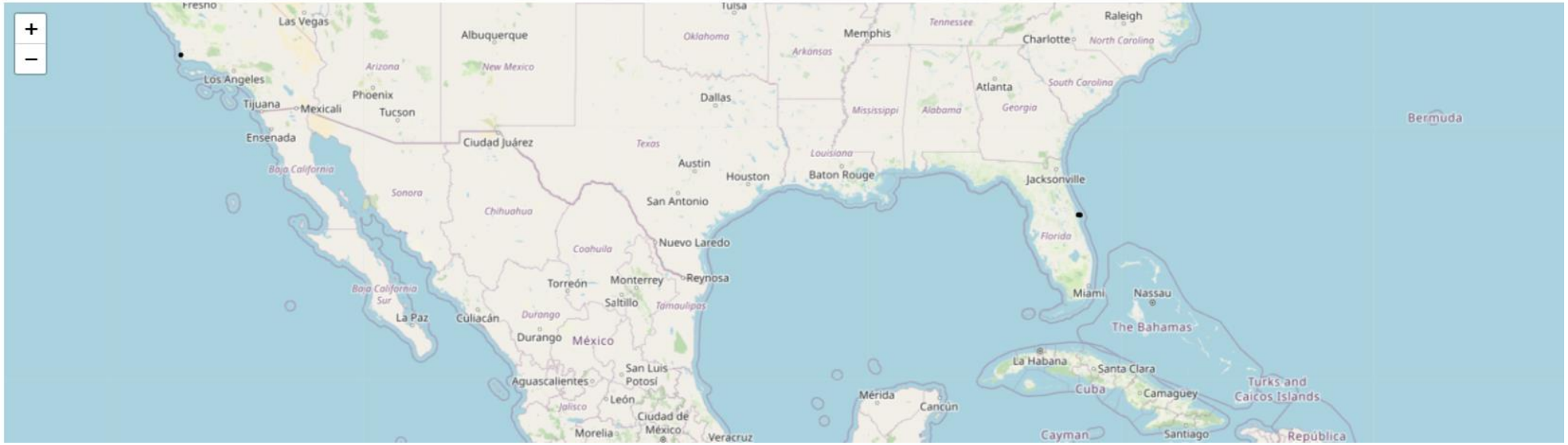
MAPA INTERACTIVO CON FOLLIUM

```
# Crea un círculo azul en las coordenadas del Centro Espacial Johnson de la NASA con una etiqueta emergente que muestra su nombre
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('Centro Espacial Johnson de la NASA'))
# Crea un círculo azul en las coordenadas del Centro Espacial Johnson de la NASA con un ícono que muestra su nombre
marker = folium.map.Marker(
    nasa_coordinate,
    # Crea un ícono como una etiqueta de texto
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

✓ 0.0s

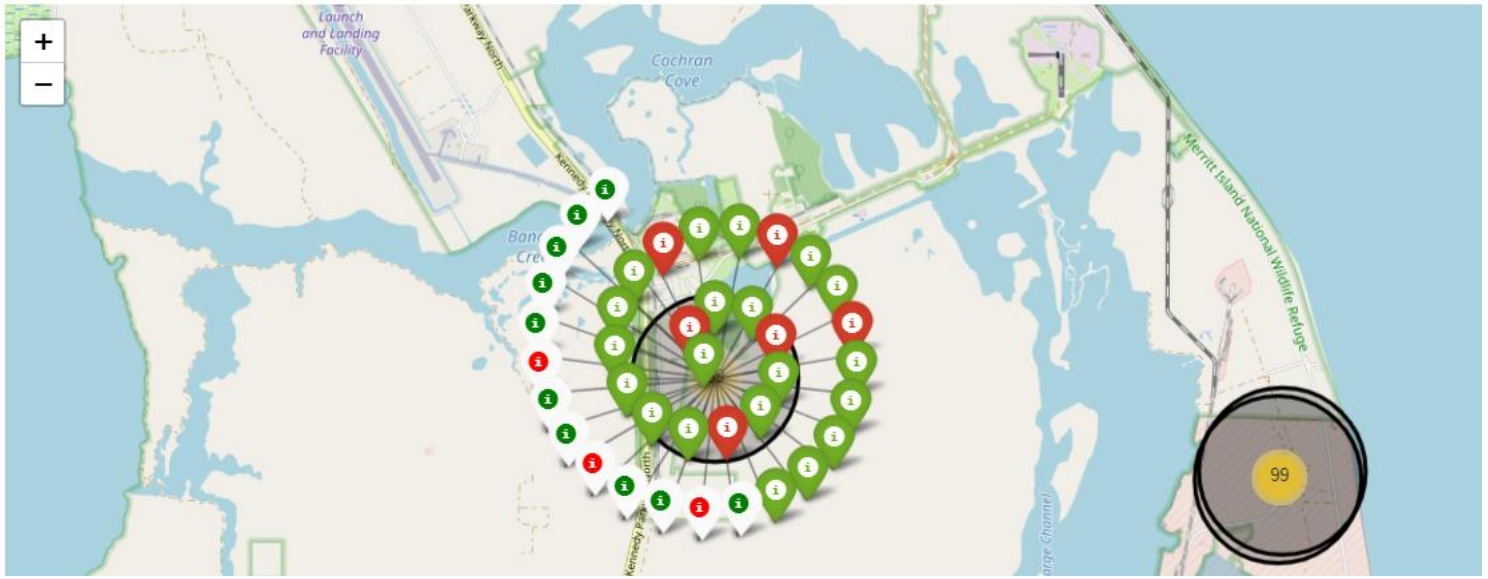
MAPA INTERACTIVO CON FOLLIUM

```
# Inicializa el mapa
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# Para cada sitio de lanzamiento, agrega un objeto de tipo Circle basado en sus coordenadas (Latitud, Longitud). Además, añade el nombre del sitio de lanzamiento como etiqueta emergente (popup label).
# Itera sobre los sitios de lanzamiento y agrega un objeto Circle y un popup label para cada uno
for index, site in launch_sites_df.iterrows():
    # Crea un objeto Circle basado en las coordenadas del sitio de lanzamiento
    circle = folium.Circle([site['Lat'], site['Long']], radius=1000, color='#000000', fill=True).add_child(folium.Popup(site['Launch Site']))

    # Añade el objeto Circle al mapa
    site_map.add_child(circle)
site_map
```



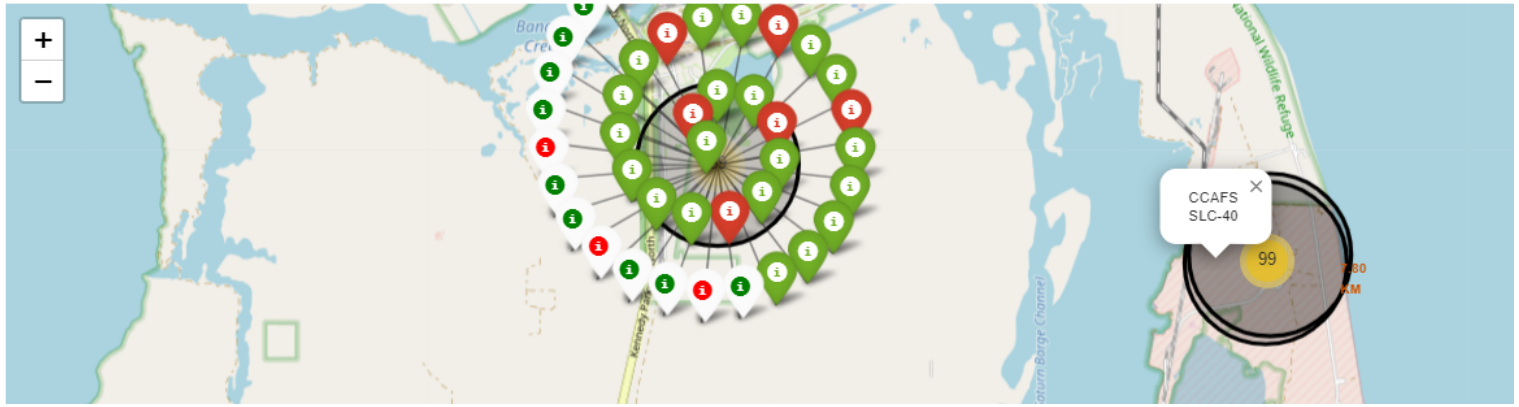
MAPA INTERACTIVO CON FOLLIUM

```
# Agregar marker_cluster al site_map actual
site_map.add_child(marker_cluster)

# Por cada fila en el DataFrame spacex_df
# Crear un objeto marcador con su coordenada
# y personalizar la propiedad icon del marcador para indicar si el lanzamiento fue exitoso o fallido
for index, record in spacex_df.iterrows():
    # Crear un marcador con la coordenada actual y el ícono basado en el color del lanzamiento
    marker = folium.Marker(location=[record['Lat'], record['Long']],
                           icon=folium.Icon(color='white', icon_color=record['marker_color']))
    # Agregar el marcador al clúster de marcadores
    marker_cluster.add_child(marker)


site_map
```



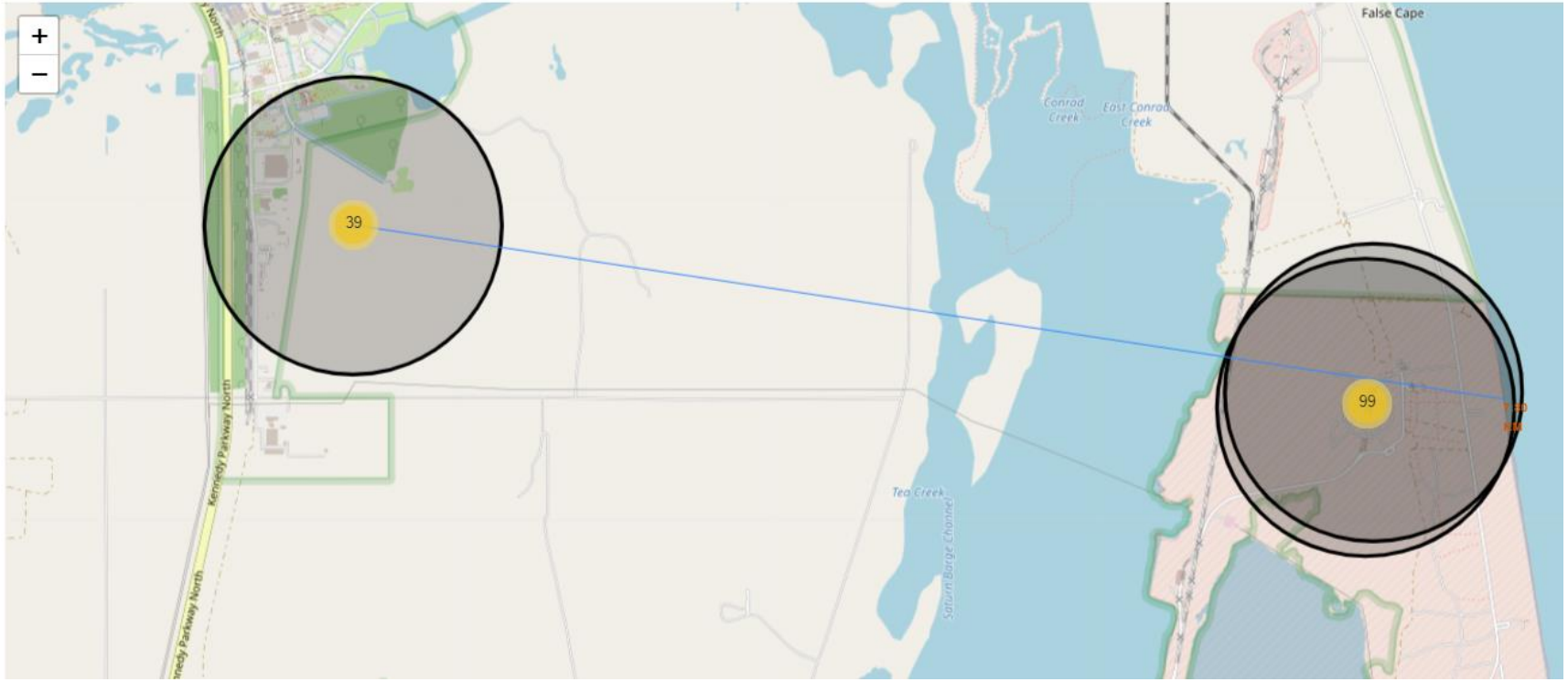
MAPA INTERACTIVO CON FOLLIUM

```
# Coordenadas del punto de la costa más cercano que seleccionaste
coastline_coordinate = [28.5628, -80.5679]

# Coordenadas del sitio de lanzamiento que estás evaluando
launch_site_lat = 28.573255  # Por ejemplo, reemplaza con las coordenadas reales
launch_site_lon = -80.646895  # Por ejemplo, reemplaza con las coordenadas reales

# Calcular la distancia entre el punto de la costa y el sitio de lanzamiento
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_coordinate[0], coastline_coordinate[1])

# Crear y agregar un marcador de distancia al mapa
distance_marker = folium.Marker(
    location=coastline_coordinate,
    icon=DivIcon(
        icon_size=(20, 20),
        icon_anchor=(0, 0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_coastline),
    )
)

# Agregar el marcador de distancia al mapa
site_map.add_child(distance_marker)
```



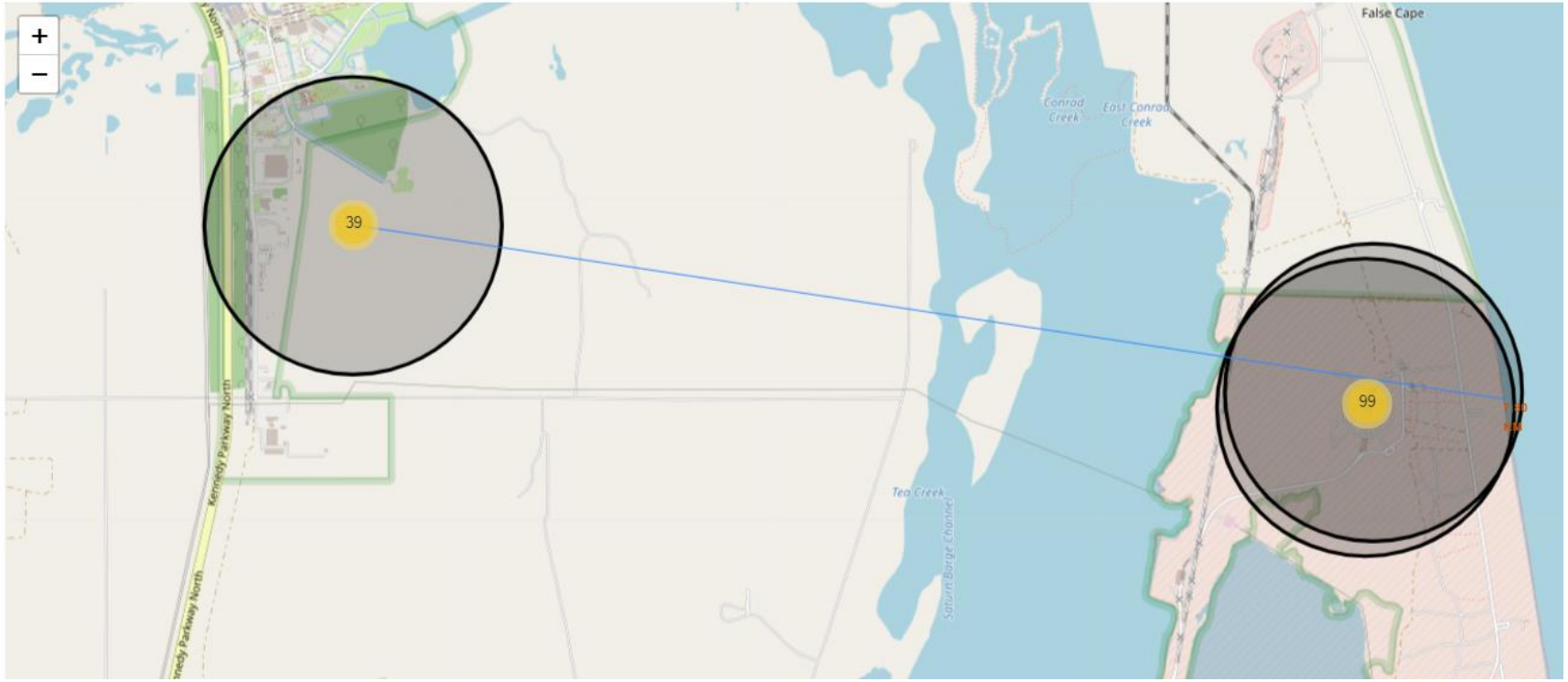
MAPA INTERACTIVO CON FOLLIUM

```
# Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
# lines=folium.PolyLine(locations=coordinates, weight=1df
linea = folium.PolyLine(locations=[coastline_coordinate, [launch_site_lat, launch_site_lon]], weight=1)
site_map.add_child(linea)
```

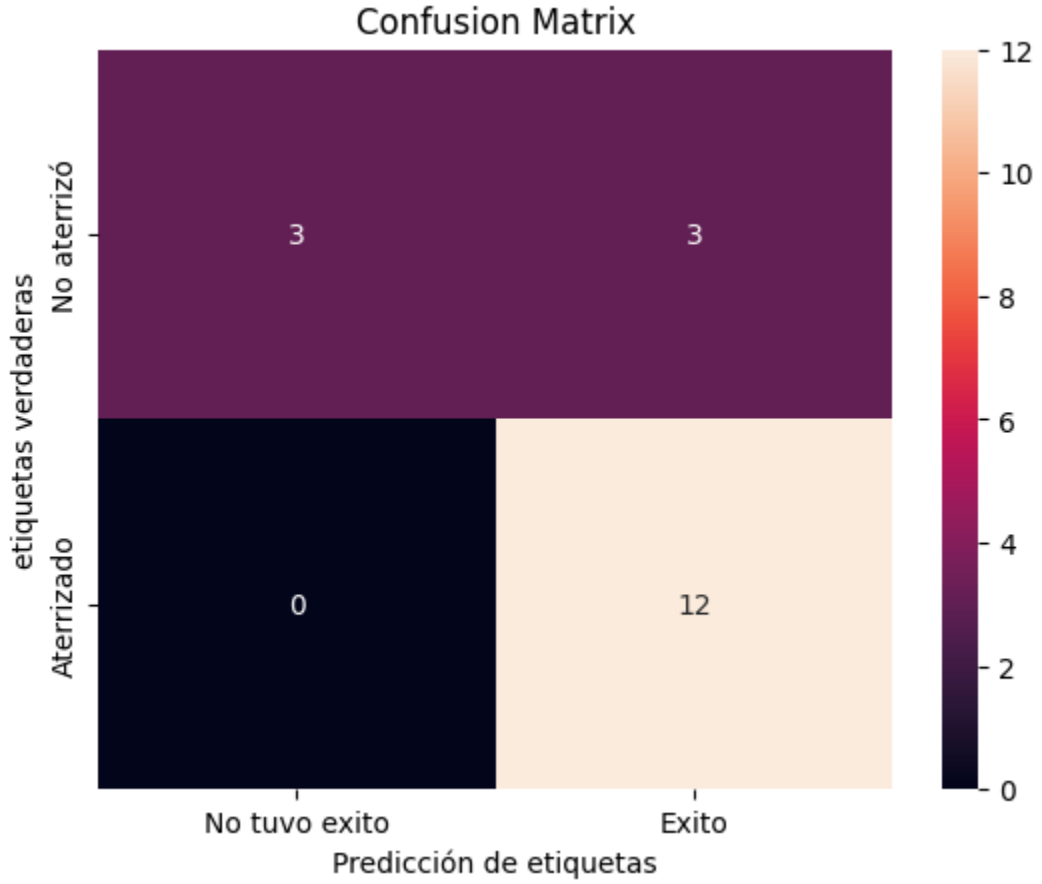


MAPA INTERACTIVO CON FOLLIUM

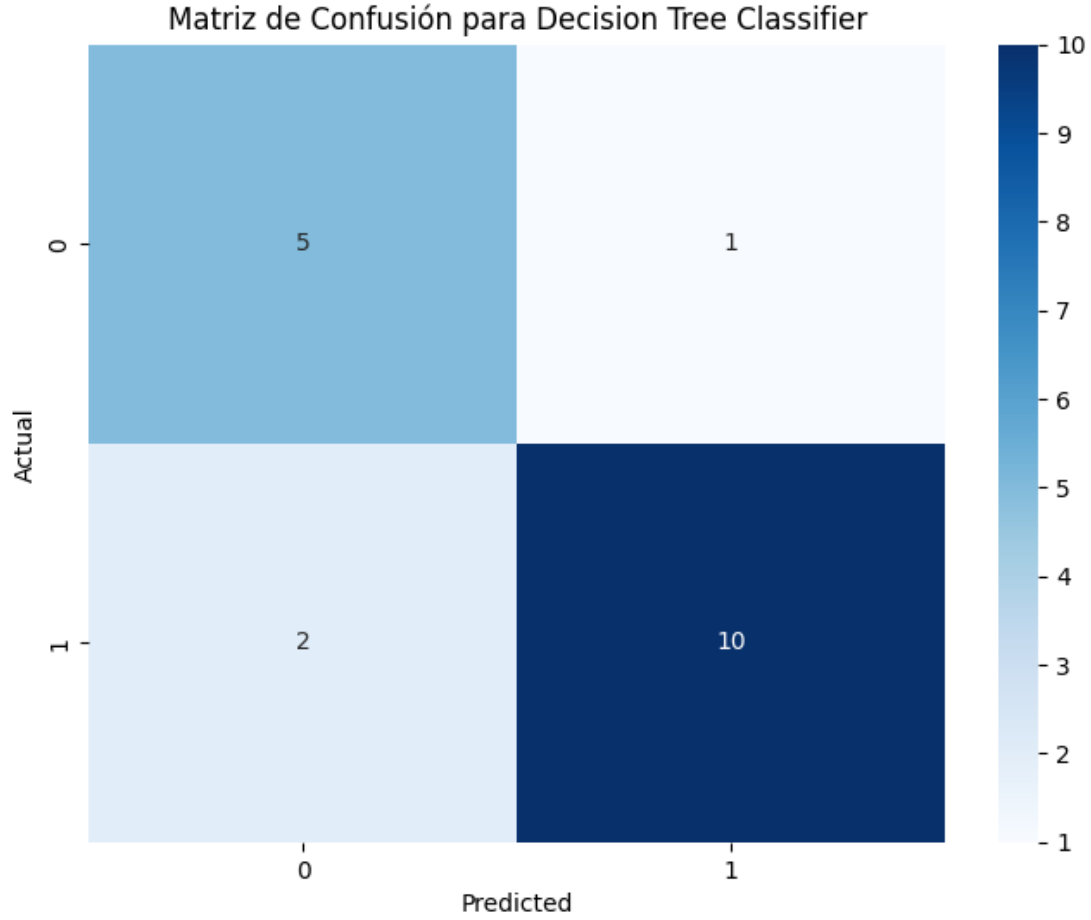
```
# Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
# lines=folium.PolyLine(locations=coordinates, weight=1df
linea = folium.PolyLine(locations=[coastline_coordinate, [launch_site_lat, launch_site_lon]], weight=1)
site_map.add_child(linea)
```



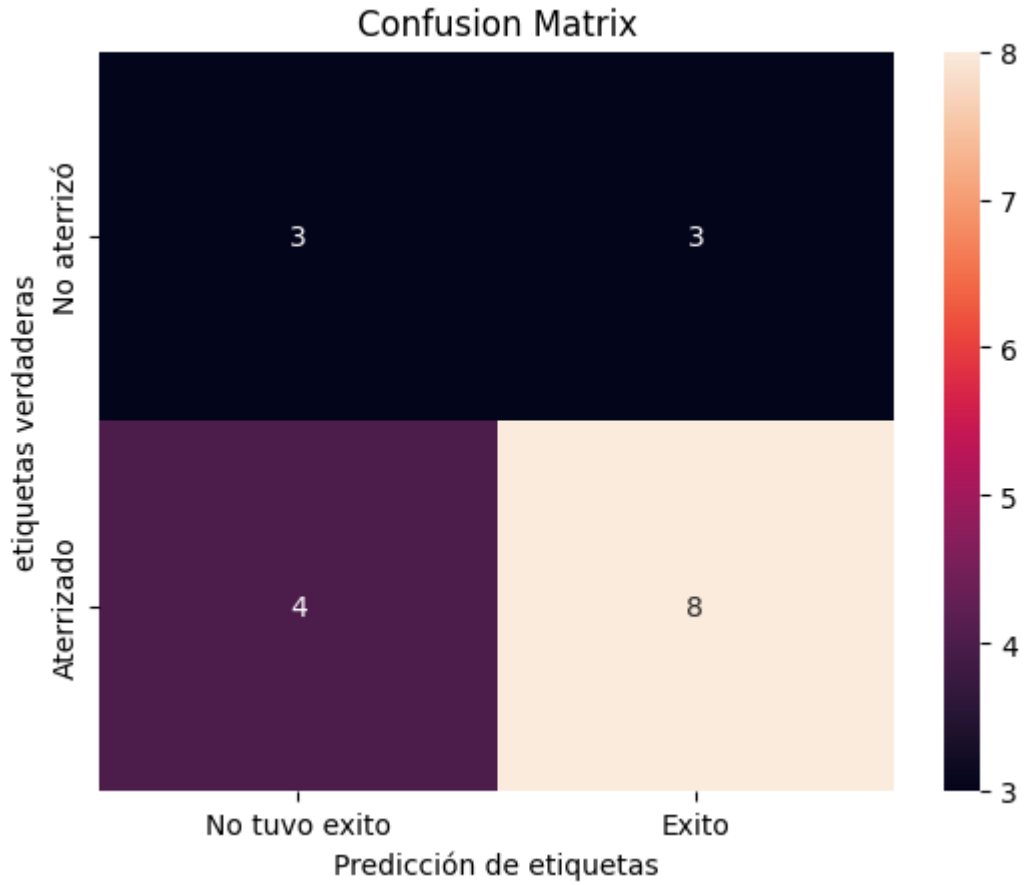
PREDICCION MEDIANTE APRENDIZAJE AUTOMATICO



PREDICCION MEDIANTE APRENDIZAJE AUTOMATICO



PREDICCION MEDIANTE APRENDIZAJE AUTOMATICO



PREDICCION MEDIANTE APRENDIZAJE AUTOMATICO

```
# Aquí se asume que ya has entrenado y ajustado los modelos: lr, tree_cv y knn_cv

# Calcula la precisión de cada modelo en los datos de prueba
accuracy_lr = lr.score(X_test, Y_test)
accuracy_tree = tree_cv.score(X_test, Y_test)
accuracy_knn = knn_cv.score(X_test, Y_test)

# Imprime las puntuaciones de precisión para cada modelo
print("Precisión de Regresión Logística:", accuracy_lr)
print("Precisión de Árbol de Decisión:", accuracy_tree)
print("Precisión de k-Vecinos más Cercanos:", accuracy_knn)
```

[37] ✓ 0.0s

```
Precisión de Regresión Logística: 0.8333333333333334
Precisión de Árbol de Decisión: 0.9444444444444444
Precisión de k-Vecinos más Cercanos: 0.6111111111111112
```

CONCLUSIONES

- 1. Crecimiento Sostenido: Se observa un crecimiento constante en la frecuencia de los lanzamientos de SpaceX a lo largo del tiempo, lo que sugiere una mayor actividad en la industria espacial y un crecimiento continuo de la empresa.

- 2. Éxito Operativo: A pesar de algunos fallos ocasionales, la tasa general de éxito en los lanzamientos es alta, lo que resalta la fiabilidad de los cohetes Falcon 9 y el éxito en la reutilización de componentes.

- 3.Falcon 9 como Vehículo Principal: La mayoría de los lanzamientos se realizan utilizando el cohete Falcon 9, lo que sugiere su papel predominante en la cartera de lanzamientos de SpaceX.

- 4. Diversidad de Órbitas: Se observa una diversidad considerable en las órbitas utilizadas, desde órbitas bajas cercanas a la Tierra hasta órbitas geoestacionarias y de transferencia.

CONCLUSIONES

- 5.Reutilización y Sostenibilidad: La reutilización de componentes, como los cohetes y las cápsulas, es una estrategia clave de SpaceX, lo que indica un enfoque hacia la sostenibilidad y la reducción de costos en la industria espacial.

- 6. Evolución Tecnológica: El análisis revela una evolución constante en la tecnología y la capacidad de carga útil, lo que muestra el progreso tecnológico continuo de la empresa a lo largo del tiempo.

- 7. Impacto en la Industria: SpaceX ha tenido un impacto significativo en la industria aeroespacial al desafiar las convenciones existentes y establecer nuevos estándares en términos de eficiencia, costos y tecnología.