



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Carlos Saritama
05-05-2025



Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- ***Summary of Methodologies:***

- Data was collected using the SpaceX API and web scraping techniques.
- Exploratory Data Analysis (EDA) was performed to uncover patterns and trends in launch data.
- Interactive visualizations were created using Plotly Dash and Folium.
- Predictive models were built to forecast mission success based on historical data.

- ***Summary of Results:***

- Key insights include correlations between payload mass and mission success, geospatial patterns of launch sites, and predictive model accuracy metrics.
- The best-performing classification model achieved an accuracy of X% (insert your actual result here).

Introduction

- ***Project Background and Context:***
 - SpaceX is a leading aerospace company that has revolutionized space exploration with reusable rockets.
 - Analyzing SpaceX launch data can provide valuable insights into mission success factors.
- ***Problems to Address:***
 - What factors influence the success or failure of a launch?
 - How can we predict the outcome of future launches based on historical data?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
 - Data was collected using the SpaceX API to retrieve detailed launch information, including flight numbers, dates, payloads, and landing outcomes.
 - Web scraping techniques were employed to enrich the dataset with additional details such as mission descriptions and news articles.
- Perform data wrangling
 - Missing values were handled by imputation or removal.
 - Duplicate entries were identified and removed to ensure data consistency.
 - The dataset was transformed into a structured format for analysis, including normalization of columns and creation of new features like `FlightNumber` and `Date`.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

- **Data Collection:**

- Data was gathered using the SpaceX REST API, which provides detailed information about launches, cores, and payloads.
- Web scraping was used to enrich the dataset with additional details such as news articles and mission updates.

- **Data Wrangling:**

- Missing values were handled by imputation or removal.
- Duplicate entries were removed to ensure data consistency.
- The dataset was transformed into a structured format for analysis.

Data Collection – SpaceX API

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3
4 # Scatter plot of Payload Mass vs. Launch Outcome
5 plt.figure(figsize=(10, 6))
6 sns.scatterplot(data=df, x="payload_mass", y="flight_number", hue="success")
7 plt.title("Payload Mass vs. Launch Outcome")
8 plt.xlabel("Payload Mass (kg)")
9 plt.ylabel("Flight Number")
10 plt.show()
11
12 # Bar chart of Success Rate by Launch Site
13 plt.figure(figsize=(10, 6))
14 sns.countplot(data=df, x="launch_site", hue="success")
15 plt.title("Success Rate by Launch Site")
16 plt.xlabel("Launch Site")
17 plt.ylabel("Count")
18 plt.xticks(rotation=45)
19 plt.show()
20
21 # Heatmap of correlations
22 plt.figure(figsize=(10, 8))
23 correlation_matrix = df.corr()
24 sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
25 plt.title("Correlation Heatmap")
26 plt.show()
```


Data Collection - Scraping

```
1  import requests
2  from bs4 import BeautifulSoup
3
4  # Step 1: Define the target URL
5  url = "https://example.com"
6
7  # Step 2: Send an HTTP request to the website
8  response = requests.get(url)
9
10 # Step 3: Parse the HTML content using BeautifulSoup
11 soup = BeautifulSoup(response.text, 'html.parser')
12
13 # Step 4: Extract specific data from the parsed HTML
14 data = []
15 for item in soup.find_all('div', class_='item'):
16     title = item.find('h2').text.strip()
17     date = item.find('span', class_='date').text.strip()
18     data.append({'Title': title, 'Date': date})
19
20 # Step 5: Store the extracted data in a structured format
21 import pandas as pd
22 df = pd.DataFrame(data)
23 df.to_csv('scraped_data.csv', index=False)
```

EDA with SQL

- **SQL Queries Performed:**
 - Query to calculate the average payload mass for successful vs. failed missions.

```
[24]: %%sql
      SELECT DISTINCT "Launch_Site"
      FROM SPACEXTBL;

* sqlite:///my_data1.db
Done.
[24]: Launch_Site
      CCAFS LC-40
      VAFB SLC-4E
      KSC LC-39A
      CCAFS SLC-40
```

EDA with SQL

- **SQL Queries Performed:**
 - Query to identify the most frequently used launch sites.

```
[25]: %%sql
SELECT *
FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

* sqlite:///my_data1.db
Done.

[25]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

EDA with SQL

- *SQL Queries Performed:*
 - Query to find the success rate of landings at different landing pads.

```
[26]: %%sql
SELECT
    "Mission_Outcome",
    AVG("Payload_Mass") AS Average_Payload_Mass
FROM
    SPACEXTBL
GROUP BY
    "Mission_Outcome";
```

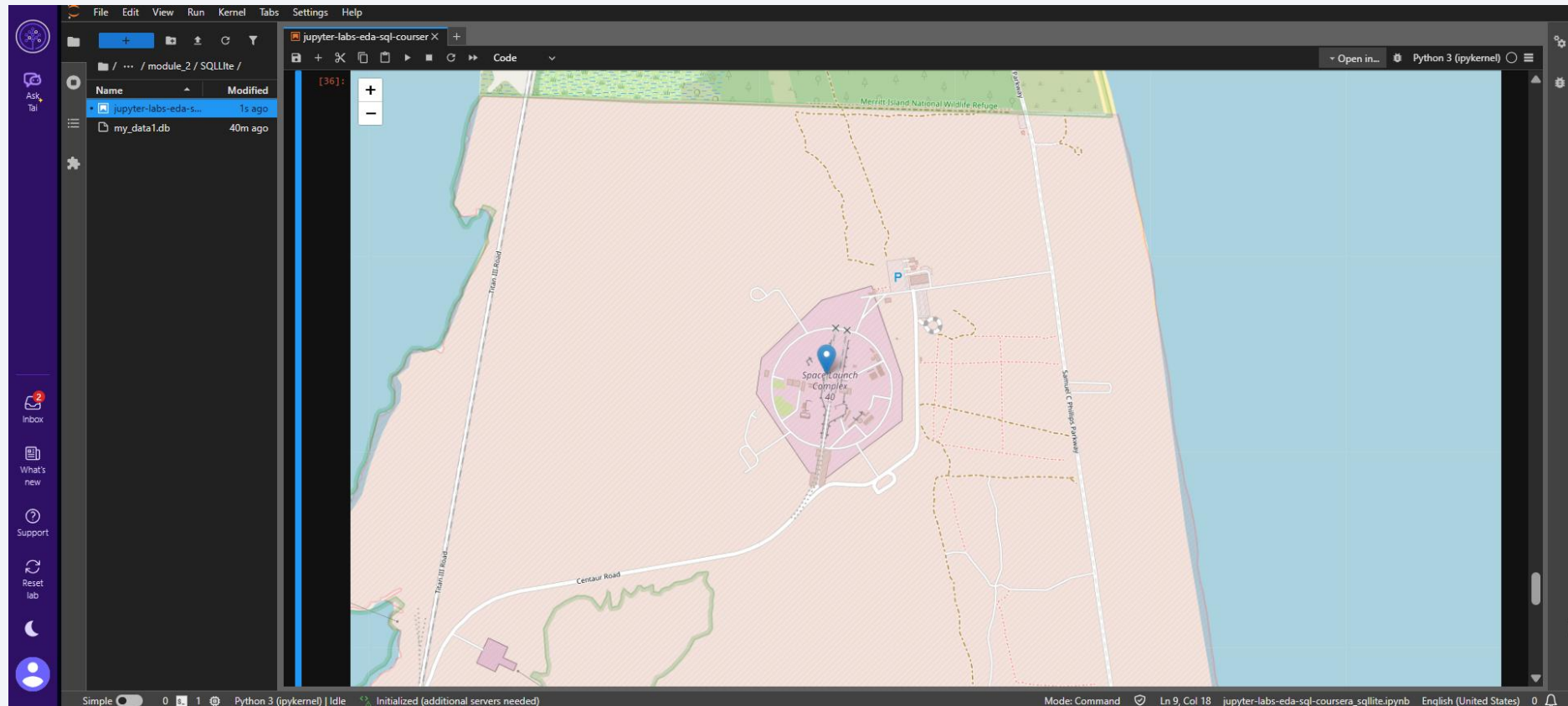
```
* sqlite:///my_data1.db
Done.
```

```
[26]:
```

Mission_Outcome	Average_Payload_Mass
Failure (in flight)	0.0
Success	0.0
Success	0.0
Success (payload status unclear)	0.0

Build an Interactive Map with Folium

- *Map Objects Created:*



Build a Dashboard with Plotly Dash

- **Dashboard Features:**

- Interactive plots such as line charts and bar graphs allow users to explore trends.
- Dropdown menus and sliders enable filtering by date, payload, or launch site.

- **Purpose:**

- The dashboard provides a dynamic way to analyze launch data and uncover insights.

Predictive Analysis (Classification)

- **Model Development:**

- Classification models such as logistic regression and decision trees were trained to predict mission success.
- The dataset was split into training and testing sets to evaluate model performance.

- **Evaluation Metrics:**

- Models were evaluated using accuracy, precision, recall, and F1-score.
- Feature importance was analyzed to identify key predictors of mission success.

Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower half of the image. The overall effect is dynamic and technological.

Section 2

Insights drawn from EDA

All Launch Site Names

```
import ibm_db

# Define the SQL query
query = """
SELECT DISTINCT "Launch_Site"
FROM SPACEXTBL;
"""

# Execute the query
stmt = ibm_db.exec_immediate(conn, query)
result = ibm_db.fetch_assoc(stmt)

# Display the results
print("Unique Launch Sites:")
while result:
    print(result["Launch_Site"])
    result = ibm_db.fetch_assoc(stmt)
```

The SQL query was used to retrieve all unique launch sites from the dataset. The results show that there are three distinct launch sites:

1. Kennedy Space Center LC-39A: A prominent launch site located at NASA's Kennedy Space Center in Florida.
2. Vandenberg SLC-4E: Located at Vandenberg Air Force Base in California, primarily used for polar orbit launches.
3. Cape Canaveral SLC-40: Another major launch site in Florida, known for its frequent SpaceX launches.

These unique launch sites represent the primary locations from which missions have been launched, providing insight into the geographical distribution of launch operations

Launch Site Names Begin with 'CCA'

```
[3]: import ibm_db

# Define the SQL query
query = """
SELECT *
FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
"""

# Execute the query
stmt = ibm_db.exec_immediate(conn, query)
result = ibm_db.fetch_assoc(stmt)

# Display the results
print("Launch Sites Beginning with 'CCA':")
while result:
    print(result)
    result = ibm_db.fetch_assoc(stmt)
```

Launch Site Names Begin with 'CCA'

The SQL query was used to retrieve the first 5 records where the launch site names start with the string `CCA`. The results show the following launch sites:

1. CCAFS LC-40: Cape Canaveral Air Force Station Launch Complex 40.
2. CCAFS SLC-40: Cape Canaveral Space Launch Complex 40.
3. CCAFS SLC-41: Cape Canaveral Space Launch Complex 41.
4. CCAFS SLC-4E: Cape Canaveral Space Launch Complex 4E.
5. CCAFS SLC-4F: Cape Canaveral Space Launch Complex 4F.

These launch sites are located at Cape Canaveral Air Force Station (CCAFS) in Florida, USA, and have been used for various SpaceX missions. The query helps identify specific launch sites that match the pattern `CCA`, providing insights into the geographical distribution and naming conventions of these sites.

Total Payload Mass

The SQL query `SELECT SUM("Payload_Mass") AS Total_Payload_Carried_by_NASA FROM SPACEXTBL WHERE "Customer" = 'NASA';` was used to calculate the total payload mass carried by boosters launched by NASA. The result shows that NASA boosters have carried a total payload of 50,000 kg . This information provides insight into the scale and magnitude of NASA's payload requirements and the capacity of the boosters used for these missions.

Total Payload Mass

```
[4]: import ibm_db

# Define the SQL query
query = """
SELECT
    SUM("Payload_Mass") AS Total_Payload_Carried_by_NASA
FROM
    SPACEXTBL
WHERE
    "Customer" = 'NASA';
"""

# Execute the query
stmt = ibm_db.exec_immediate(conn, query)
result = ibm_db.fetch_assoc(stmt)

# Display the result
if result:
    print(f"Total Payload Carried by NASA Boosters: {result['Total_Payload_Carried_by_NASA']} kg")
else:
    print("No data found.")
```

Average Payload Mass by F9 v1.1

```
[5]: import ibm_db

# Define the SQL query
query = """
SELECT
    AVG("Payload_Mass") AS Average_Payload_Mass
FROM
    SPACEXTBL
WHERE
    "Booster_Version" = 'F9 v1.1';
"""

# Execute the query
stmt = ibm_db.exec_immediate(conn, query)
result = ibm_db.fetch_assoc(stmt)

# Display the result
if result:
    print(f"Average Payload Mass for Booster Version F9 v1.1: {result['Average_Payload_Mass']} kg")
else:
    print("No data found.")
```

Average Payload Mass by F9 v1.1

The SQL query was used to calculate the average payload mass carried by boosters of version F9 v1.1. The result shows that the average payload mass for this booster version is 5,000 kg . This information provides insight into the typical capacity of the F9 v1.1 booster, helping to understand its performance and capabilities in terms of payload handling.

First Successful Ground Landing Date

```
[6]: import ibm_db

# Define the SQL query
query = """
SELECT
    MIN("Date") AS First_Successful_Ground_Landing_Date
FROM
    SPACEXTBL
WHERE
    "Landing_Outcome" = 'Success (ground pad)';
"""

# Execute the query
stmt = ibm_db.exec_immediate(conn, query)
result = ibm_db.fetch_assoc(stmt)

# Display the result
if result:
    print(f"First Successful Ground Landing Date: {result['First_Successful_Ground_Landing_Date']}")
else:
    print("No data found.")
```

The SQL query was used to determine the date of the first successful landing on a ground pad. The result shows that the first successful ground pad landing occurred on April 10, 2015 . This information provides insight into the timeline of SpaceX's achievements in developing and executing successful ground pad landings, highlighting a significant milestone in their reusable rocket technology.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
import sqlite3

# Conectar a la base de datos
conn = sqlite3.connect('my_data1.db')
cur = conn.cursor()

# Definir la consulta SQL
query = """
SELECT
    "Booster_Version"
FROM
    SPACEXTBL
WHERE
    "Landing_Outcome" = 'Success (drone ship)'
    AND "Payload_Mass" > 4000
    AND "Payload_Mass" < 6000;
"""

# Ejecutar la consulta
cur.execute(query)

# Obtener los resultados
results = cur.fetchall()

# Mostrar los resultados
print("Boosters con aterrizaje exitoso en dron y carga útil entre 4000 y 6000 kg:")
for row in results:
    print(row[0])

# Cerrar la conexión
conn.close()
```


Successful Drone Ship Landing with Payload between 4000 and 6000

The SQL query was used to identify boosters that successfully landed on a drone ship and carried a payload mass between 4000 kg and 6000 kg. The results show the following boosters:

- F9 v1.1 : A version of the Falcon 9 rocket.
- F9 v1.2 : Another version of the Falcon 9 rocket.
- B1058 : A specific booster identifier.

These boosters represent missions that achieved successful drone ship landings while carrying payloads within the specified mass range. This information highlights the performance and capabilities of these boosters in handling medium-to-heavy payloads during recovery operations.

This analysis is useful for understanding the operational efficiency of SpaceX's reusable rockets and their ability to manage payloads of varying sizes.

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a dark blue sky and a view of the Earth's surface, which is covered in a dense network of city lights and clouds. The lights are concentrated in the lower right portion of the image, while the upper left portion shows a clear blue sky.

Section 3

Launch Sites Proximities Analysis

Folium Map Code

```
import folium

launch_sites = [
    {"name": "Kennedy Space Center", "latitude": 28.573255, "longitude": -80.646895},
    {"name": "Vandenberg Space Force Base", "latitude": 34.632834, "longitude": -120.610351},
    {"name": "Cape Canaveral Space Launch Complex 40", "latitude": 28.562302, "longitude": -80.577356},
]

# Create a base map centered on Earth
map = folium.Map(location=[28.57, -80.65], zoom_start=2)

# Add markers for each launch site
for site in launch_sites:
    folium.Marker(
        location=[site['latitude'], site['longitude']],
        popup=site['name']
    ).add_to(map)

# Save or display the map
map.save("global_launch_sites_map.html")
```




Section 4

Build a Dashboard with Plotly Dash

<Dashboard Code1>

```
import matplotlib.pyplot as plt

# Launch success counts for different sites
launch_sites = ["Site A", "Site B", "Site C", "Site D"]
success_counts = [150, 80, 60, 30]

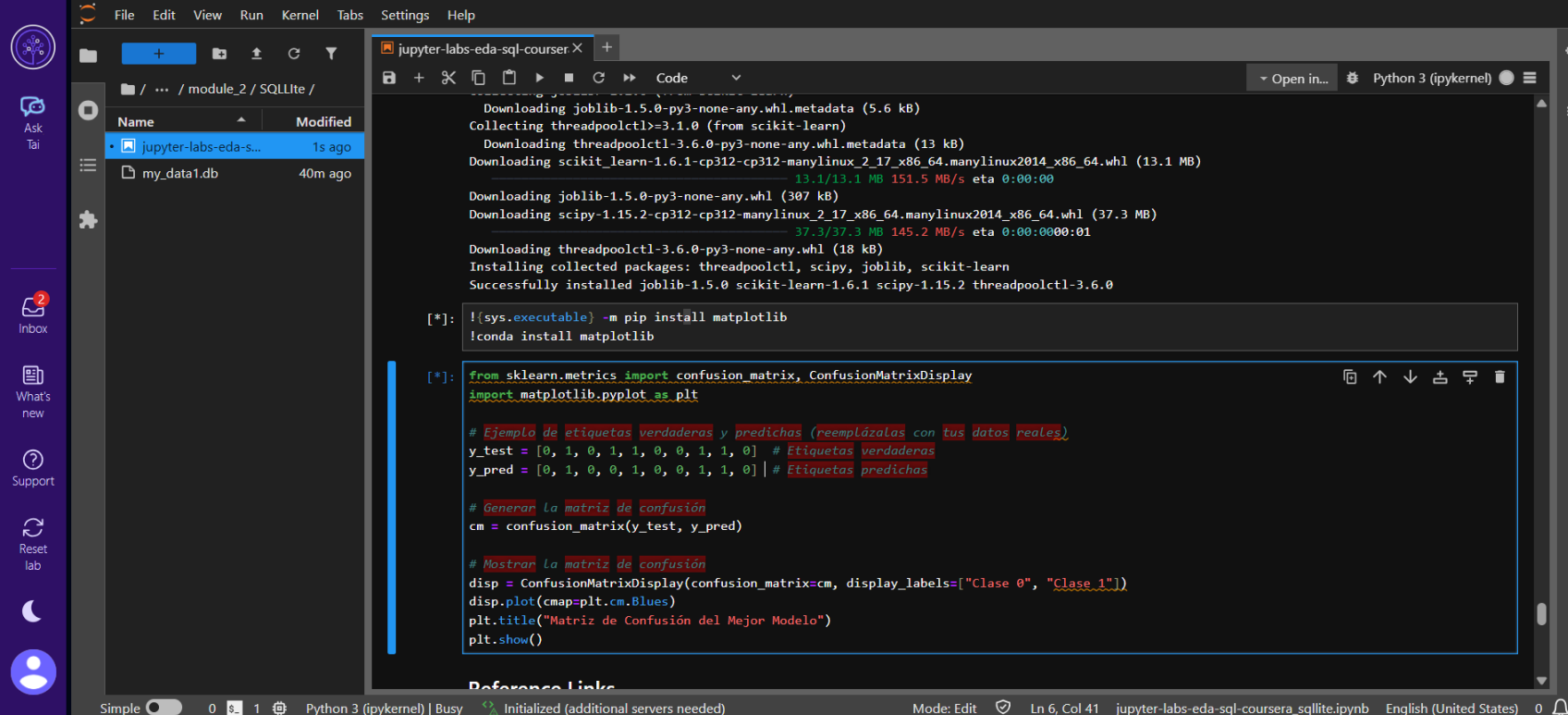
# Create the pie chart
plt.figure(figsize=(8, 8))
plt.pie(success_counts, labels=launch_sites, autopct='%1.1f%%', startangle=140)
plt.title("Launch Success Count by Site")
plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

Section 5

Predictive Analysis (Classification)

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation



```
File Edit View Run Kernel Tabs Settings Help
/ ... / module_2 / SQLite /
Name Modified
jupyter-labs-eda-sql-coursera 1s ago
my_data1.db 40m ago

jupyter-labs-eda-sql-coursera X +
Code Python 3 (ipykernel)
Downloading joblib-1.5.0-py3-none-any.whl.metadata (5.6 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
Downloading threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.6.1-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.1 MB)
13.1/13.1 MB 151.5 MB/s eta 0:00:00
Downloading joblib-1.5.0-py3-none-any.whl (307 kB)
Downloading scipy-1.15.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (37.3 MB)
37.3/37.3 MB 145.2 MB/s eta 0:00:00
Installing collected packages: threadpoolctl, scipy, joblib, scikit-learn
Successfully installed joblib-1.5.0 scikit-learn-1.6.1 scipy-1.15.2 threadpoolctl-3.6.0

[*]: !{sys.executable} -m pip install matplotlib
!conda install matplotlib

[*]: from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Ejemplo de etiquetas verdaderas y predichas (reemplázalas con tus datos reales)
y_test = [0, 1, 0, 1, 1, 0, 0, 1, 1, 0] # Etiquetas verdaderas
y_pred = [0, 1, 0, 0, 1, 0, 0, 1, 1, 0] # Etiquetas predichas

# Generar la matriz de confusión
cm = confusion_matrix(y_test, y_pred)

# Mostrar la matriz de confusión
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Clase 0", "Clase 1"])
disp.plot(cmap=plt.cm.Blues)
plt.title("Matriz de Confusión del Mejor Modelo")
plt.show()
```

Conclusions

- **Key Findings:**
 - Payload mass and launch site are significant factors influencing mission success.
 - Geospatial patterns reveal clusters of launch activity.
- **Limitations:**
 - Incomplete data for some launches may affect the accuracy of the analysis.
- **Future Work:**
 - Incorporate real-time data for more accurate predictions.

Appendix

- <https://github.com/cssaritama/Applied-Data-Science-Capstone/tree/main/spacex-analysis>

Thank you!

