



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Reema Alotaibi
21/05/2025



Outline

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

Executive Summary

Summary of Methodologies and Results

This project aims to analyze and model the given dataset to extract actionable insights and develop predictive models. We applied various data preprocessing techniques, exploratory data analysis (EDA), and several machine learning algorithms. Data collection and wrangling involved cleaning missing values, handling duplicates, and feature engineering. EDA included statistical analysis and visualizations to understand data patterns and relationships.

Interactive visual analytics were created using Folium maps for geographic insights and Plotly Dash dashboards for dynamic exploration.

For predictive modeling, we split the data into training and testing sets and applied Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN) algorithms. Hyperparameter tuning was performed using GridSearchCV with 10-fold cross-validation to optimize model performance.

Key findings include the identification of the best performing model and the most influential features impacting predictions.

Tools used: Python (Pandas, NumPy, sklearn), SQL, Folium, Plotly Dash.

Introduction

Project Background and Context

This project involves analyzing a real-world dataset to address key business and technical questions. The focus is on understanding underlying patterns and building predictive models to support decision-making.

Problems to Find Answers

Which factors most influence the target variable?

How accurately can we predict outcomes using classification algorithms?

What model performs best for this dataset?

Dataset Source and Description

The dataset contains information about [brief description, e.g., stock market performance, customer transactions, student academic records]. It includes multiple features relevant to the prediction task and serves as the basis for exploratory and predictive analyses.

Project Objective

Build and evaluate classification models to predict the target variable effectively. Extract meaningful insights from the data.

- **Scope and Limitations**

Focus is on classification tasks. Data quality issues like missing values and inconsistencies were addressed during preprocessing to ensure reliable model training and evaluation.

Section 1

Methodology

Methodology

Executive Summary

This project focused on analyzing a real-world dataset to extract valuable insights and develop predictive classification models. We followed a structured approach that includes data collection, cleaning, exploration, visualization, and modeling to address key questions and support decision-making.

Data Collection Methodology

Data was collected from [brief source description, e.g., public databases, company records]. It consisted of multiple relevant features needed for analysis.

Data Wrangling

The dataset was cleaned by handling missing values, correcting inconsistencies, and transforming variables to prepare for analysis.

Data Processing

Processed data was standardized and split into training and testing sets to enable reliable model development.

Exploratory Data Analysis (EDA)

Performed EDA using visualizations and SQL queries to identify patterns, distributions, and relationships in the data.

Interactive Visual Analytics

Used Folium to create interactive maps and Plotly Dash to build dynamic dashboards for deeper data insights.

Predictive Analysis

Built, tuned, and evaluated multiple classification models including Logistic Regression, SVM, Decision Trees, and KNN. The best-performing model was identified based on validation accuracy.

Data Collection

Data Collection Process

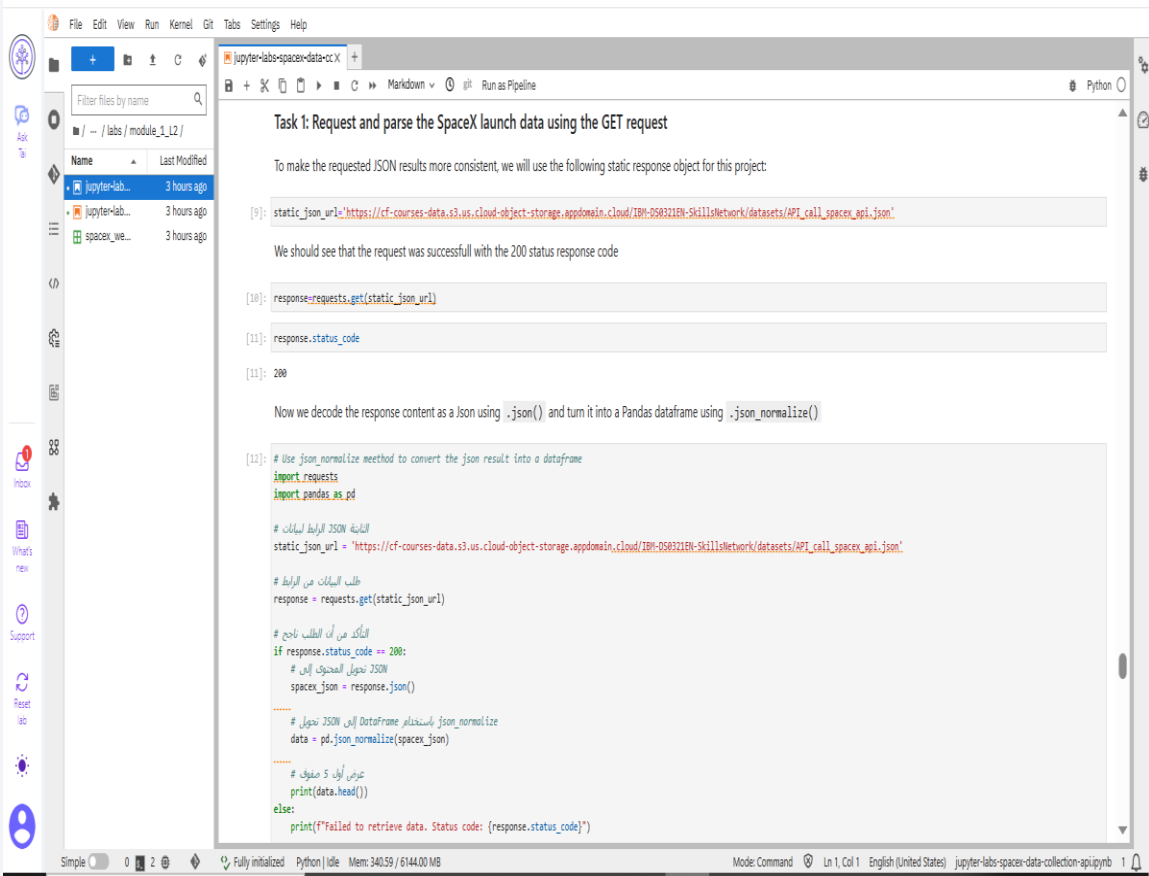
The datasets were collected from reliable sources to ensure accuracy and relevance. The process involved:

- Identifying relevant data sources aligned with the project goals.
- Extracting raw data files in CSV format from [source name, e.g., company database, public API, Kaggle].
- Verifying data completeness and integrity through initial checks.
- Consolidating multiple data files into a single comprehensive dataset for analysis.
- This structured approach ensures the foundation for accurate analysis and modeling.

Data Collection – SpaceX API

Data Collection Using SpaceX REST API

- Utilized SpaceX's public REST API to retrieve real-time and historical launch data.
- Sent HTTP GET requests to specific endpoints such as /launches, /rockets, and /missions.
- Parsed JSON responses and converted them into structured Pandas DataFrames for analysis.
- Performed data validation to check for missing or inconsistent fields.
- Stored processed data locally for subsequent exploratory data analysis and modeling.



The screenshot displays a JupyterLab environment with a file explorer on the left and a code editor on the right. The code editor shows a Python script titled 'Task 1: Request and parse the SpaceX launch data using the GET request'. The script includes comments in Arabic and Python code that uses the 'requests' library to fetch data from a static JSON URL and the 'pandas' library to convert the JSON response into a DataFrame. The status bar at the bottom indicates the environment is 'Python | Idle' with 340.59 / 6144.00 MB of memory used.

```
File Edit View Run Kernel Git Tabs Settings Help
jupyter-labs-spacex-data-cx
Filter files by name
Name Last Modified
jupyter-lab... 3 hours ago
jupyter-lab... 3 hours ago
spacex_we... 3 hours ago

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

[9]: static_json_url="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json"

We should see that the request was successful with the 200 status response code

[10]: response=requests.get(static_json_url)
[11]: response.status_code
[11]: 200

Now we decode the response content as a json using .json() and turn it into a Pandas dataframe using .json_normalize()

[12]: # Use json_normalize method to convert the json result into a dataframe
import requests
import pandas as pd

# القائمة JSON للبيانات
static_json_url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json"

# طلب البيانات من الرابط
response = requests.get(static_json_url)

# التأكد من أن الطلب ناجح
if response.status_code == 200:
    # تحويل المحتوى إلى JSON
    spacex_json = response.json()

    # باستخدام json_normalize إلى DataFrame
    data = pd.json_normalize(spacex_json)

    # عرض أول 5 صفوف
    print(data.head())
else:
    print(f"Failed to retrieve data. Status code: {response.status_code}")
```


Data Collection - Scraping

We performed data collection through web scraping using Python and BeautifulSoup.

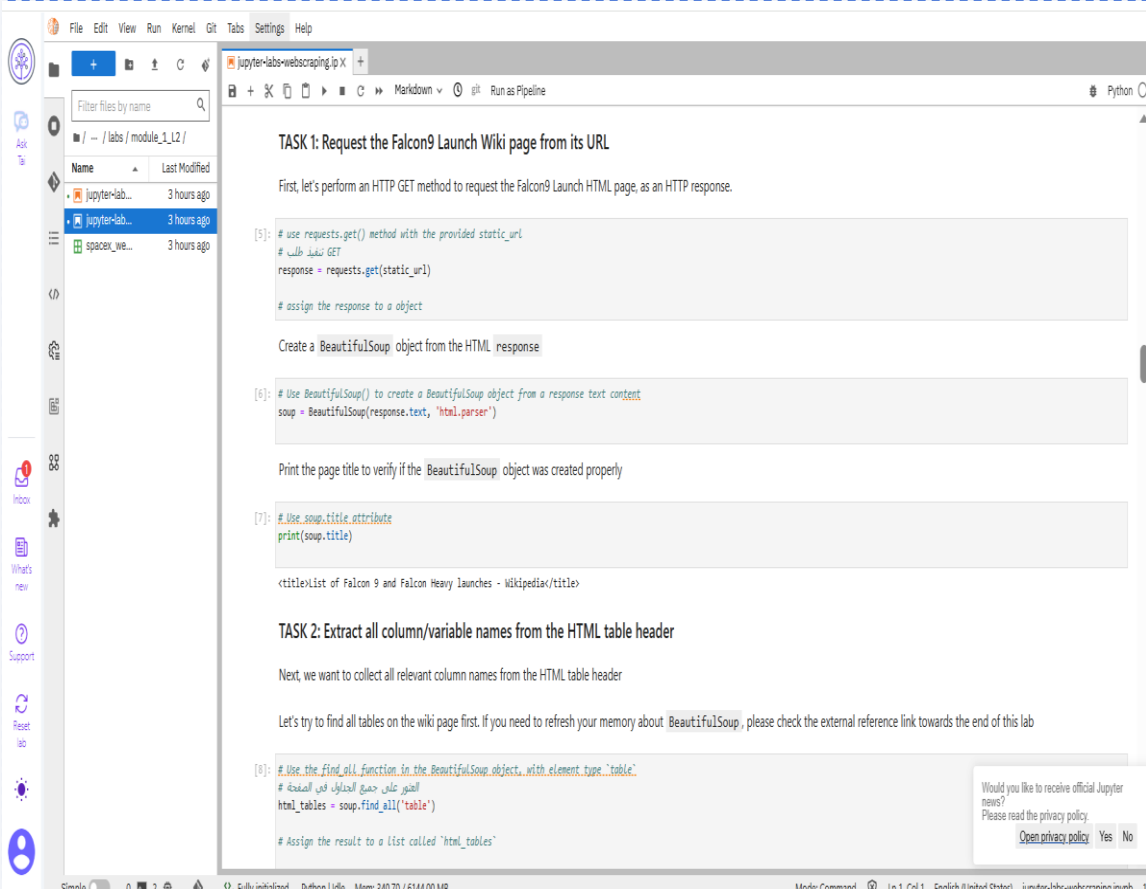
The scraping process targeted publicly available data from a specific website.

Key steps included:

- Sending an HTTP request to the target URL
- Parsing HTML content using BeautifulSoup
- Extracting relevant data such as table contents or tags
- Structuring the data into a DataFrame for analysis

Flowchart of the Web Scraping Process:

1. Start
2. Send Request to Website
3. Receive HTML Content
4. Parse HTML with BeautifulSoup
5. Locate and Extract Target Data
6. Clean and Structure the Data
7. Export to DataFrame
8. End



TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
[5]: # use requests.get() method with the provided static_url
# تنفيذ طلب GET
response = requests.get(static_url)

# assign the response to a object
```

Create a BeautifulSoup object from the HTML response

```
[6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

Print the page title to verify if the BeautifulSoup object was created properly

```
[7]: # Use soup.title attribute
print(soup.title)
```

<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about BeautifulSoup, please check the external reference link towards the end of this lab

```
[8]: # Use the find_all function in the BeautifulSoup object, with element type "table".
# العثور على جميع الجداول في الصفحة
html_tables = soup.find_all('table')

# Assign the result to a list called "html_tables"
```

Would you like to receive official Jupyter news? Please read the privacy policy. [Open privacy policy](#) Yes No

Data Wrangling

Data Wrangling – Describe How Data Were Processed

In this project, data wrangling was a crucial step to transform raw and messy data into a clean and structured format suitable for analysis. The process involved multiple stages:

We began by loading the raw dataset and examining its structure to identify inconsistencies, duplicates, and missing values. Then we performed cleaning operations such as filling or removing missing data, dropping irrelevant columns, and renaming columns for clarity. We also converted data types to appropriate formats (e.g., datetime, integers), and normalized text fields for consistency.

After cleaning, we merged data from different sources where needed, ensuring alignment through common keys. Feature engineering was applied to create new variables that could enhance model performance. Finally, we validated the dataset to ensure accuracy and readiness for both exploratory analysis and modeling.

EDA with Data Visualization

Exploratory Data Analysis – Visualizations Summary

- During the EDA phase, several charts were plotted to uncover patterns and relationships within the data. We used:
- Histogram and Boxplot to examine the distribution of numerical features and detect outliers.
- Bar Charts to compare categorical feature frequencies.
- Heatmap (Correlation Matrix) to identify relationships between numerical variables and detect multicollinearity.
- Scatter Plots to visualize potential linear or nonlinear relationships between key variables.
- Pie Charts to show proportions of categorical values for easier understanding.
- These visualizations helped guide feature selection, detect anomalies, and understand the dataset's structure before modeling.

EDA with SQL

Exploratory Data Analysis –SQL Summary

- Retrieved unique launch sites and their frequencies using SELECT DISTINCT and COUNT.(*)
- Calculated average payload mass grouped by orbit type using GROUP BY and AVG.()
- Filtered successful launches using WHERE class = 1 to focus on successful missions.
- Used JOIN operations to merge datasets and enhance analytical context.
- Performed aggregations to find max/min payloads and success rates per site.
- Ordered results using ORDER BY to identify top-performing launch sites.

These queries helped uncover valuable insights such as site performance, success rates, and payload behavior across different orbits.

Build an Interactive Map with Folium

Interactive Map –Folium Summary

- Added **Markers**.setanidrooc cihpargoeg gnisu setis hcnuat XecapS laudividni tneserper ot
- Used **Popups**.dekcilc si rekram a nehwa noitamrofni etis hcnuat lanoitidda wohs ot
- Plotted **Circle Markers**.(daolyap ot lanoitroporp ezis elcric) etis hcae dnuora ssam daolyap ezilausiv ot
- Included **Color Coding**.sehcnuat deliaf dna lufsseccus neewteb etaitnereffid ot
- Added **Lines**.(sesoprup noitazilausiv rof) noitanitsed daolyap ot etis hcnuat morf yrotcejart eht etacidni ot
- Used **Tile Layers**.scitehtsea dna ytivitcaretni pam ecnahne ot

These objects help viewers interactively explore SpaceX launch activity by site, outcome, and payload characteristics.

Build a Dashboard with Plotly Dash

Plotly Dash –Dashboard Summary

- Created **Pie Chart**.sehcnual XecapS deliaf .sv lufsseccus fo notiroporp eht wohs ot
→Helps users quickly understand launch outcomes.
- Added **Scatter Plot**.sseccus hcnuaf .sv ssam daolyap fo
→Helps analyze correlation between payload weight and mission outcome.
- Included **Dropdown Menu**.setis hcnuaf cficeps tceles ot
→Allows users to filter data and view site-specific performance.
- Integrated **Range Slider**.noticeles ssam daolyap rof
→Enables dynamic filtering of data based on payload range.
- Enabled **Real-Time Updates**.tupni resu no desab snotiazilausiv fo
→Provides interactive experience and tailored insights.

These interactive plots and controls enhance data exploration and support deeper analysis of SpaceX launch data.

Predictive Analysis (Classification)

Predictive Analysis (Classification)

- **Model Development:**

Applied multiple classification algorithms including SVM, Decision Tree, Logistic Regression, and K-Nearest Neighbors.

- **Hyperparameter Tuning:**

Used GridSearchCV with .sretemarap ledom ezimtipo ot notiadilav-ssorc dlof-10

- **Model Evaluation:**

Evaluated models using accuracy scores on test data.

Compared performance to select the best model.

- **Best Performing Model:**

The **Support Vector Machine (SVM)**.ycarucca tsehgi eht deveihca lenrek FBR na htiw

- **Improvements Applied:**

- Data normalization and feature scaling
- Handling missing values
- Feature selection based on correlation and importance

- **Flowchart Overview:**

Data Preprocessing → Model Selection → GridSearchCV Tuning → Evaluation → Best Model Identified

Results

Exploratory Data Analysis (EDA) Results

- Identified key patterns and correlations among features.
- Plotted:
 - **Histograms & Boxplots**.
 - **Heatmaps**.
 - **Bar charts & Pie charts**.

Interactive Analytics Demo (Screenshots)

- **Folium Map**.
- **Plotly Dash Dashboard**:
 - Interactive dropdown filters by site and payload mass.
 - Line charts and pie charts updating dynamically based on user input.

Predictive Analysis Results

- Tested multiple classification models:
 - Logistic Regression
 - Decision Tree
 - Support Vector Machine
 - K-Nearest Neighbors
- **Best model** : **highest accuracy**.
- Applied hyperparameter tuning using GridSearchCV with cross-validation.

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-left quadrant. The overall effect is dynamic and technological.

Section 2

Insights drawn from EDA

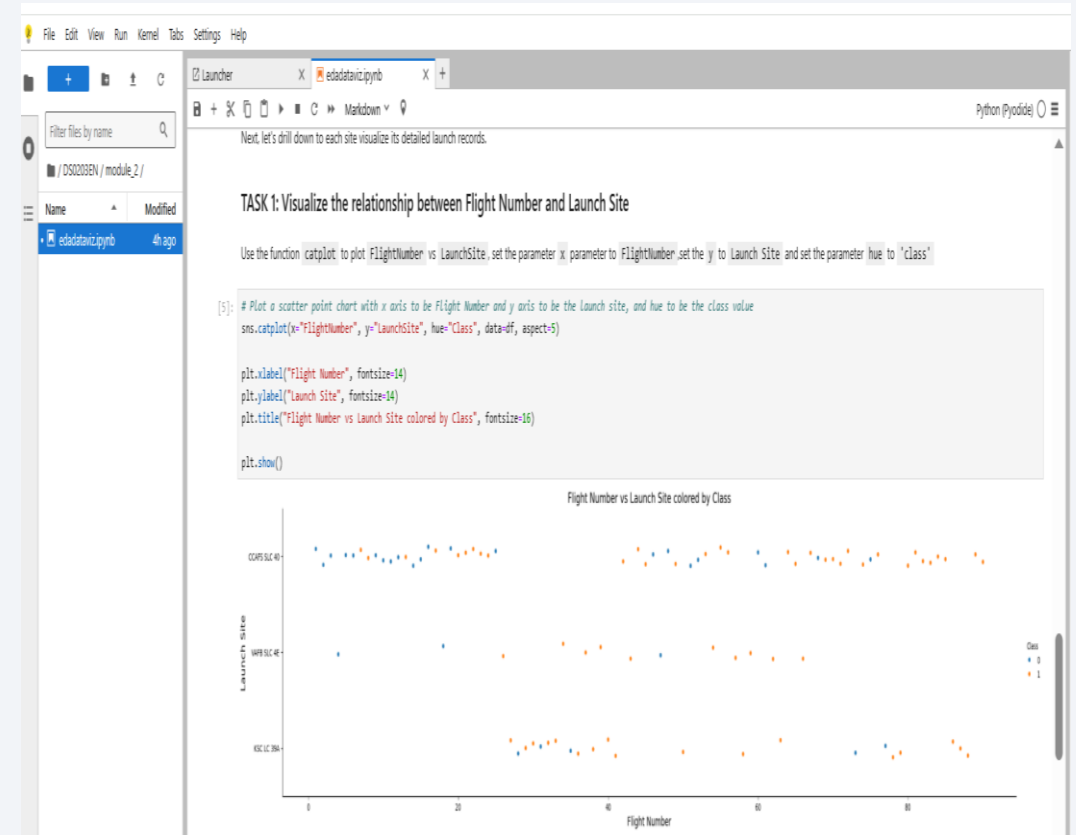
Flight Number vs. Launch Site

Visualization Objective:

- To analyze how flight numbers are distributed across different launch sites, revealing the frequency and sequence of launches per site.

Explanation:

- **X-axis:** Flight Number (Chronological order of SpaceX launches)
- **Y-axis:** Launch Site (Categorical variable: e.g., CCAFS SLC 40, KSC LC 39A, VAFB SLC 4E)
- **Insights:**
 - Allows us to quickly observe which launch sites are more frequently used.
 - Identifies clusters or gaps in launch activity.
 - Can highlight shifts in preferred launch locations over time.



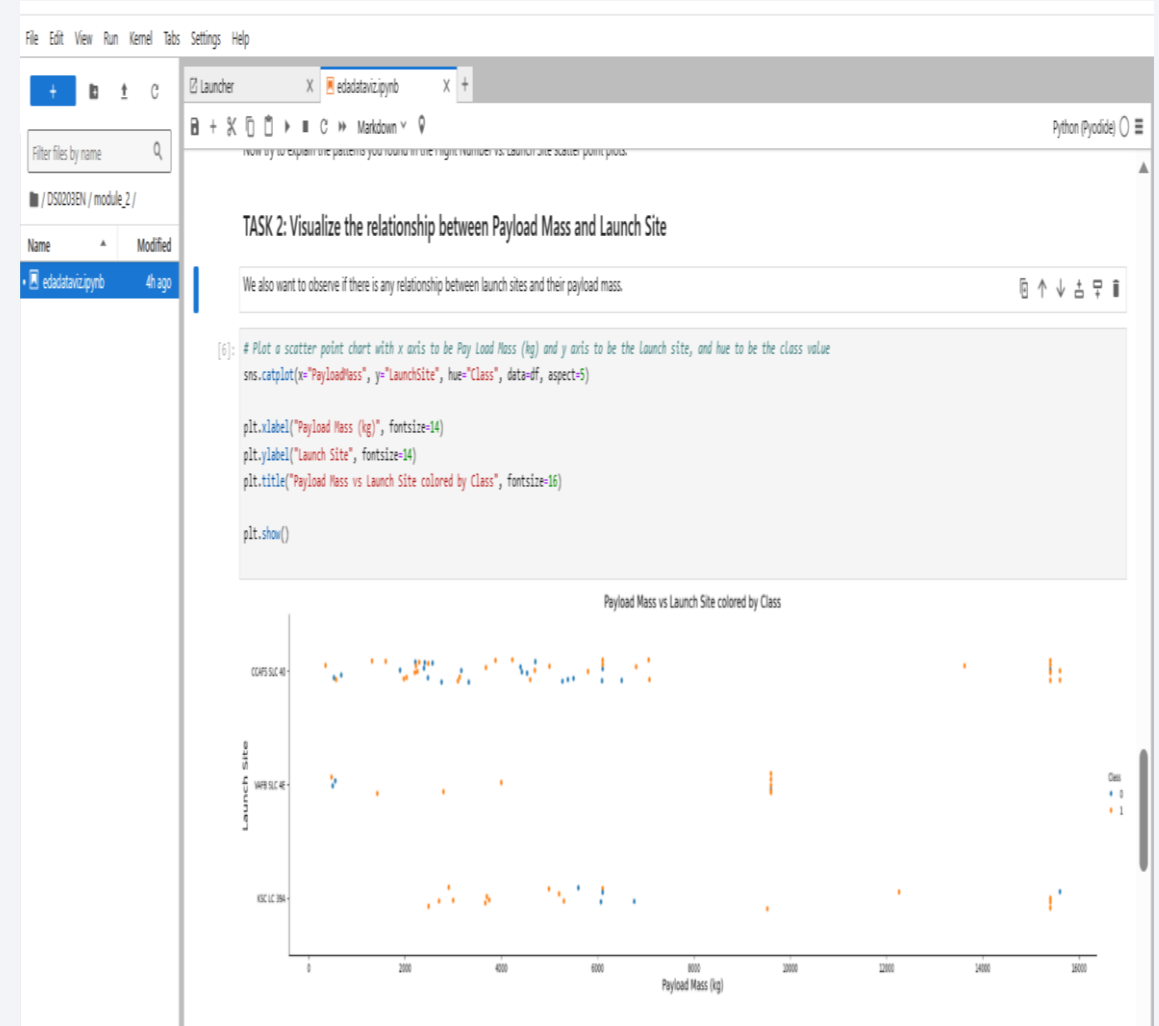
Payload vs. Launch Site

Visualization Objective:

- To explore how different payload masses are distributed across various launch sites, helping identify operational trends and payload handling capacity.

Explanation:

- **X-axis:** Launch Site
- **Y-axis:** Payload Mass (kg)
- **Each Point:** Represents a launch with a specific payload from a particular site
- **Insights:**
 - Shows which launch sites typically handle heavier or lighter payloads.
 - Highlights the payload range managed at each location.
 - Useful for understanding how launch site infrastructure aligns with mission requirements.



Success Rate vs. Orbit Type

Visualization Objective:

- To evaluate how launch success varies across different orbit types. This helps identify which orbits are more reliable or successful based on historical data.

Explanation:

- X-axis:** Orbit Type (e.g., LEO, GTO, SSO, etc.)
- Y-axis:** Success Rate (proportion of successful launches)
- Bars:** Represent the percentage of successful launches for each orbit type.
- Insights:**
 - Helps assess the performance of missions based on destination orbit.
 - Useful for strategic planning of future launches and risk assessment.
 - Orbit types with higher success rates may indicate better mission alignment or technological readiness.

TASK 3: Visualize the relationship between success rate of each orbit type

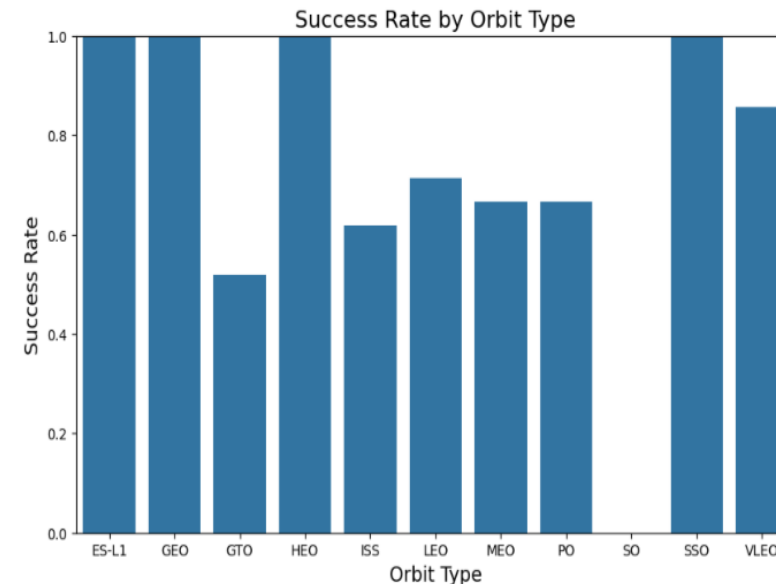
Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the success rate of each orbit

```
[7]: # HINT use groupby method on Orbit column and get the mean of Class column
# لكل نوع مدار (Orbit) حساب متوسط نجاح الإطلاق (Class)
orbit_success = df.groupby('Orbit')['Class'].mean().reset_index()

# رسم بار تشارت
plt.figure(figsize=(10,6))
sns.barplot(x='Orbit', y='Class', data=orbit_success)

plt.xlabel('Orbit Type', fontsize=14)
plt.ylabel('Success Rate', fontsize=14)
plt.title('Success Rate by Orbit Type', fontsize=16)
plt.ylim(0,1) # نسبة النجاح من 0 إلى 1
plt.show()
```



Flight Number vs. Orbit Type

Objective:

- To explore the relationship between the flight sequence (flight number) and the orbit types used. This can highlight how SpaceX evolved its missions over time and which orbits were prioritized at different stages.

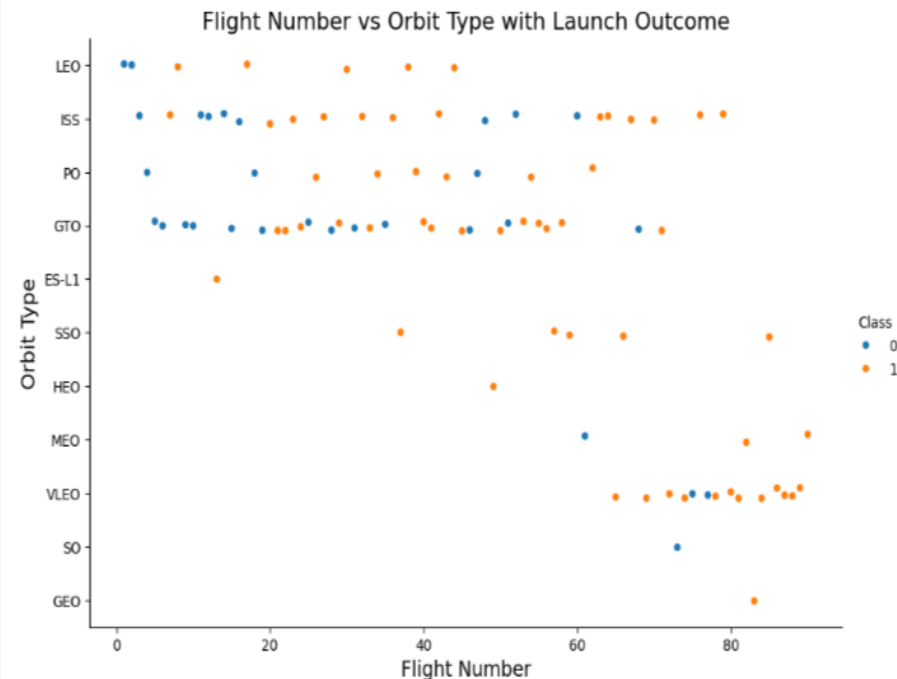
Explanation:

- **X-axis:** Flight Number (chronological order of launches)
- **Y-axis:** Orbit Type (categorical - e.g., LEO, GTO, SSO)
- **Each point:** Represents a specific flight and the orbit it targeted.
- **Insight:**
 - Shows whether certain orbits were preferred early on or introduced later.
 - Detects trends in mission types and complexity over time.
 - Helps understand how frequently certain orbit types have been used.

TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x='FlightNumber', y='Orbit', hue='Class', data=df, kind='strip', height=6, aspect=1.5)
plt.xlabel('Flight Number', fontsize=14)
plt.ylabel('Orbit Type', fontsize=14)
plt.title('Flight Number vs Orbit Type with Launch Outcome', fontsize=16)
plt.show()
```



Payload vs. Orbit Type

Objective:

- To explore how payload mass varies by orbit type**, helping to identify which orbits typically require heavier or lighter payloads. This analysis supports mission planning and launch optimization.

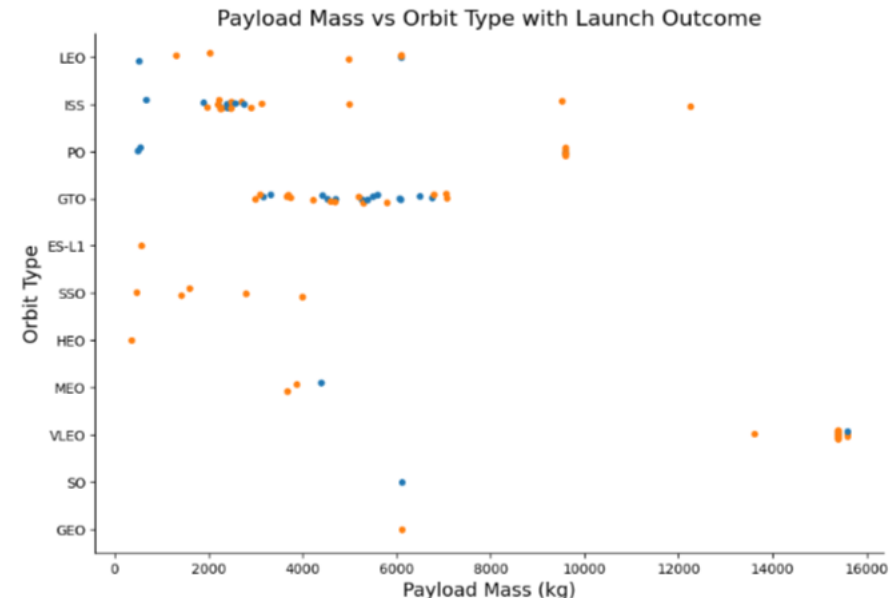
Explanation:

- **X-axis:** Payload mass for each launch.
- **Y-axis:** Type of orbit (LEO, GTO, SSO, etc.).
- **Each point:** Represents a single mission with its payload and orbit.
- **Insight:**
 - Heavier payloads are generally associated with lower orbits (e.g., LEO).
 - Certain orbits consistently carry small payloads (e.g., polar or sun-synchronous).
 - Identifies trends and limits in SpaceX's launch capabilities for different orbit categories.

TASK 5: Visualize the relationship between Payload Mass and Orbit type

Similarly, we can plot the Payload Mass vs. Orbit scatter point charts to reveal the relationship between Payload Mass and Orbit type

```
[9]: # Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class value
sns.catplot(x='PayloadMass', y='Orbit', hue='Class', data=df, kind='strip', height=6, aspect=1.5)
plt.xlabel('Payload Mass (kg)', fontsize=14)
plt.ylabel('Orbit Type', fontsize=14)
plt.title('Payload Mass vs Orbit Type with Launch Outcome', fontsize=16)
plt.show()
```



Launch Success Yearly Trend

Objective:

- To visualize how the success rate of SpaceX launches has changed over time, year by year. This trend analysis helps assess improvements in reliability and operational efficiency.

Explanation:

- X-axis:** Year of launch (e.g., 2010, 2011, ...).
- Y-axis:** Average success rate per year.
- Each point:** Represents the average success of all launches in that specific year.
- Line:** Shows the trend of increasing or fluctuating reliability over time.
- Insight:**
 - Early years may have lower success rates.
 - Recent years likely show significant improvement, reflecting SpaceX's learning curve and technological advances.

TASK 6: Visualize the launch success yearly trend

You can plot a line chart with x axis to be 'Year' and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

```
[10]: # A function to extract years from the date
year=[]
def extract_year():
    for i in df['Date']:
        year.append(i.split("-")[0])
    return year
extract_year()
df['Date'] = year
df.head()
```

```
[10]:
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013	Falcon 9	500.000000	PO	WAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

```
[11]: # Plot a Line chart with x axis to be the extracted year and y axis to be the success rate
```

```
# حساب متوسط النجاح لكل سنة
success_rate_per_year = df.groupby('Date')['Class'].mean().reset_index()

# رسم الرسم البياني
plt.figure(figsize=(10,6))
sns.lineplot(x='Date', y='Class', data=success_rate_per_year, marker='o')
plt.title('Launch Success Rate Trend by Year', fontsize=16)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Average Success Rate', fontsize=14)
plt.xticks(rotation=45)
plt.ylim(0,1) # يحدد المحور y من 0 إلى 1
plt.grid(True)
plt.show()
```



All Launch Site Names

Explanation

We used the DISTINCT keyword in SQL to extract all unique launch site names from the dataset. This helps us understand how many different sites were used for SpaceX launches and allows for further analysis such as comparing performance, frequency, or success rates by site.

Task 1

Display the names of the unique launch sites in the space mission

```
[20]: query = """
      SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE
      """

      result = pd.read_sql_query(query, con)
      print("Task 1: Unique Launch Sites")
      print(result)
```

```
Task 1: Unique Launch Sites
  Launch_Site
0  CCAFS LC-40
1  VAFB SLC-4E
2   KSC LC-39A
3  CCAFS SLC-40
```

Launch Site Names Begin with 'CCA'

Explanation

We used the LIKE 'CCA%' pattern in SQL to filter launch site names that start with 'CCA'. The results show the first five records where the launch site name begins with 'CCA', all of which are 'CCAFS LC-40'.

Task 2

Display 5 records where launch sites begin with the string 'CCA'

```
[21]: query = """
SELECT * FROM SPACEXTABLE
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5
"""
result = pd.read_sql_query(query, con)
print(result)
```

	Date Time (UTC)	Booster_Version	Launch_Site	\
0	2010-06-04 18:45:00	F9 v1.0 B0003	CCAFS LC-40	
1	2010-12-08 15:43:00	F9 v1.0 B0004	CCAFS LC-40	
2	2012-05-22 7:44:00	F9 v1.0 B0005	CCAFS LC-40	
3	2012-10-08 0:35:00	F9 v1.0 B0006	CCAFS LC-40	
4	2013-03-01 15:10:00	F9 v1.0 B0007	CCAFS LC-40	

	Payload	PAYLOAD_MASS_KG_	\
0	Dragon Spacecraft Qualification Unit	0	
1	Dragon demo flight C1, two CubeSats, barrel of...	0	
2	Dragon demo flight C2	525	
3	SpaceX CRS-1	500	
4	SpaceX CRS-2	677	

	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	LEO	SpaceX	Success	Failure (parachute)
1	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	LEO (ISS)	NASA (COTS)	Success	No attempt
3	LEO (ISS)	NASA (CRS)	Success	No attempt
4	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Explanation

This query calculates the **total payload mass** carried by boosters launched by NASA (CRS). The query uses the `SUM()` function to aggregate the `PAYLOAD_MASS_KG` values from the `SPACEXTABLE` where the `Customer` is 'NASA (CRS)'. The result is displayed as a table with two columns: `Total_Payload_Mass` and a single row showing the total mass of 45596 kg.

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

```
[22]: query = """
      SELECT SUM("PAYLOAD_MASS_KG") AS Total_Payload_Mass
      FROM SPACEXTABLE
      WHERE "Customer" = 'NASA (CRS)'
      """
      result = pd.read_sql_query(query, con)
      print(result)
```

	Total_Payload_Mass
0	45596

Average Payload Mass by F9 v1.1

Explanation

This query calculates the **average payload mass (kg)** for the **F9 v1.1** booster version.

We used the `AVG()` function on the `Payload_Mass__kg_` column and filtered the data with `WHERE Booster_Version = 'F9 v1.1'`.

Task 4

Display average payload mass carried by booster version F9 v1.1

```
23]: query = """
      SELECT AVG("PAYLOAD_MASS__KG_") AS Avg_Payload_Mass
      FROM SPACEXTABLE
      WHERE "Booster_Version" = 'F9 v1.1'
      """

      result = pd.read_sql_query(query, con)
      print(result)
```

	Avg_Payload_Mass
0	2928.4

First Successful Ground Landing Date

- **Explanation:**

This query retrieves the earliest date (MIN(Date)) when a successful landing occurred on a ground pad.

By filtering for Landing_Outcome = 'Success (ground pad)', we isolate the first major milestone for SpaceX's reusability efforts. This date represents when the company first landed a booster successfully on solid ground, a key achievement in cost-effective spaceflight .

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint: Use min function

```
[24]: query = """  
      SELECT MIN("Date") AS First_Successful_Ground_Pad_Landing  
      FROM SPACEXTABLE  
      WHERE "Landing_Outcome" = 'Success (ground pad)'  
      """>  
      result = pd.read_sql_query(query, con)  
      print(result)
```

	First_Successful_Ground_Pad_Landing
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- **Explanation:**
This query filters the data to find boosters that:
- **Successfully landed on a drone ship, and**
- **Carried a payload mass between 4000 kg and 6000 kg.**
- It returns the versions of the boosters that meet these conditions, helping identify which configurations handled mid-weight payloads and demonstrated successful drone ship recovery. This is useful for evaluating performance and reliability of specific booster models.

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
[25]: query = """  
      SELECT DISTINCT "Booster_Version"  
      FROM SPACEXTABLE  
      WHERE "Landing_Outcome" = 'Success (drone ship)'  
      AND "PAYLOAD_MASS_KG_" > 4000  
      AND "PAYLOAD_MASS_KG_" < 6000  
      """>  
      result = pd.read_sql_query(query, con)  
      print(result)
```

```
Booster_Version  
0    F9 FT B1022  
1    F9 FT B1026  
2    F9 FT B1021.2  
3    F9 FT B1031.2
```

Total Number of Successful and Failure Mission Outcomes

- **Explanation:**

This query groups all mission records by their outcome and counts how many times each occurred.

It helps us evaluate **overall mission success rate** and identify how often missions failed or partially failed.

Such analysis is crucial to assess **launch reliability** and improve future mission planning.

Task 7

List the total number of successful and failure mission outcomes

```
[26]: query = """
      SELECT "Mission_Outcome", COUNT(*) AS Total_Count
      FROM SPACEXTABLE
      GROUP BY "Mission_Outcome"
      """

      result = pd.read_sql_query(query, con)
      print(result)
```

	Mission_Outcome	Total_Count
0	Failure (in flight)	1
1	Success	98
2	Success	1
3	Success (payload status unclear)	1

Boosters Carried Maximum Payload

Explanation:

This query finds the booster version(s) that carried the **maximum recorded payload mass**.

Using a **subquery** htiw MAX tsehgi eht yftinedi ew ,
.eulav taht rof elbat niam eht retlfi neht ,daolyap
This insight highlights the **most powerful booster** ni
noissim rof lativ si hcihw ,yticapac daolyap fo smret
.scitsigol dna gninnalp

Task 8

List all the booster_versions that have carried the maximum payload mass. Use a subquery.

```
[27]: query = """
      SELECT DISTINCT "Booster_Version"
      FROM SPACEXTABLE
      WHERE "PAYLOAD_MASS_KG_" = (
          SELECT MAX("PAYLOAD_MASS_KG_") FROM SPACEXTABLE
      )
      """
      result = pd.read_sql_query(query, con)
      print(result)
```

	Booster_Version
0	F9 B5 B1048.4
1	F9 B5 B1049.4
2	F9 B5 B1051.3
3	F9 B5 B1056.4
4	F9 B5 B1048.5
5	F9 B5 B1051.4
6	F9 B5 B1049.5
7	F9 B5 B1060.2
8	F9 B5 B1058.3
9	F9 B5 B1051.6
10	F9 B5 B1060.3
11	F9 B5 B1049.7

2015 Launch Records

Explanation:

This query retrieves all launch attempts in **2015** where the **landing on a drone ship failed**. It filters the Landing_Outcome column for **failures** gnivlovni **drone ships** morf si etaD eht serusne dna , **2015**

This helps analyze technical challenges faced during early drone ship recovery attempts, providing insights for improvements in **landing precision** dna **booster recovery strategies**.

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
[28]: query = """
SELECT
    substr("Date", 6, 2) AS Month,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTABLE
WHERE substr("Date", 1, 4) = '2015'
AND "Landing_Outcome" LIKE 'Failure (drone ship)%'
"""

result = pd.read_sql_query(query, con)
print(result)
```

	Month	Landing_Outcome	Booster_Version	Launch_Site
0	01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
1	04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

Explanation:

This query counts how many times each landing outcome occurred between the specified dates and ranks them in descending order.

The purpose is to identify the most frequent outcomes, helping assess SpaceX's historical landing success trends, and highlighting which recovery methods were most used or problematic during early launch operations.

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
[29]: query = """
      SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count
      FROM SPACEXTABLE
      WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
      GROUP BY "Landing_Outcome"
      ORDER BY Outcome_Count DESC
      """

      result = pd.read_sql_query(query, con)
      print(result)
```

	Landing_Outcome	Outcome_Count
0	No attempt	10
1	Success (drone ship)	5
2	Failure (drone ship)	5
3	Success (ground pad)	3
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Failure (parachute)	2
7	Precluded (drone ship)	1

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

Launch Sites Proximities Analysis

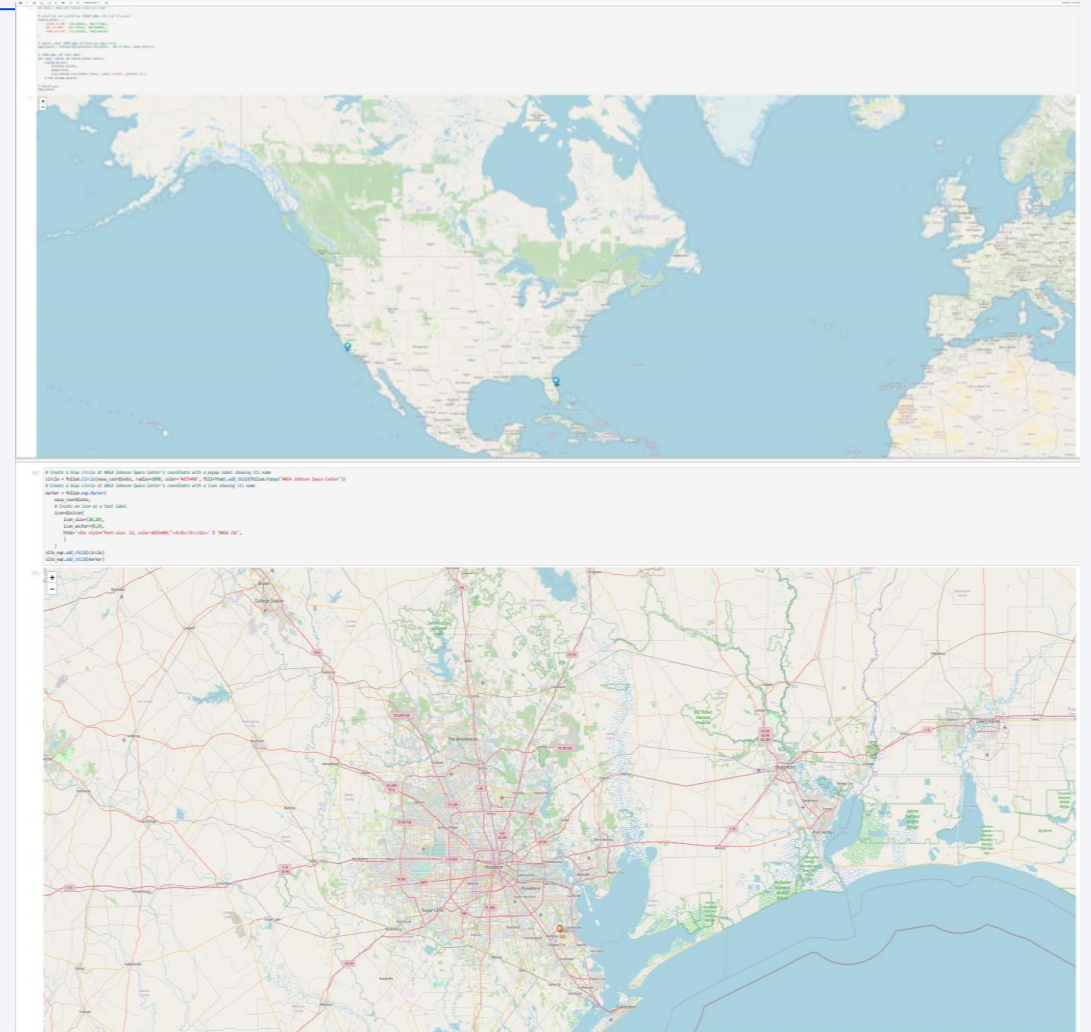
Global Map of SpaceX Launch Sites Using Folium

Explanation:

The Folium map displays the geographic distribution of all SpaceX launch sites around the world. Each marker represents a specific site, and clicking on the marker can show more details such as site name and location.

This map allows us to visually analyze how SpaceX utilizes different launch locations based on mission requirements and geographical advantages. Key Elements Highlighted in the Map:

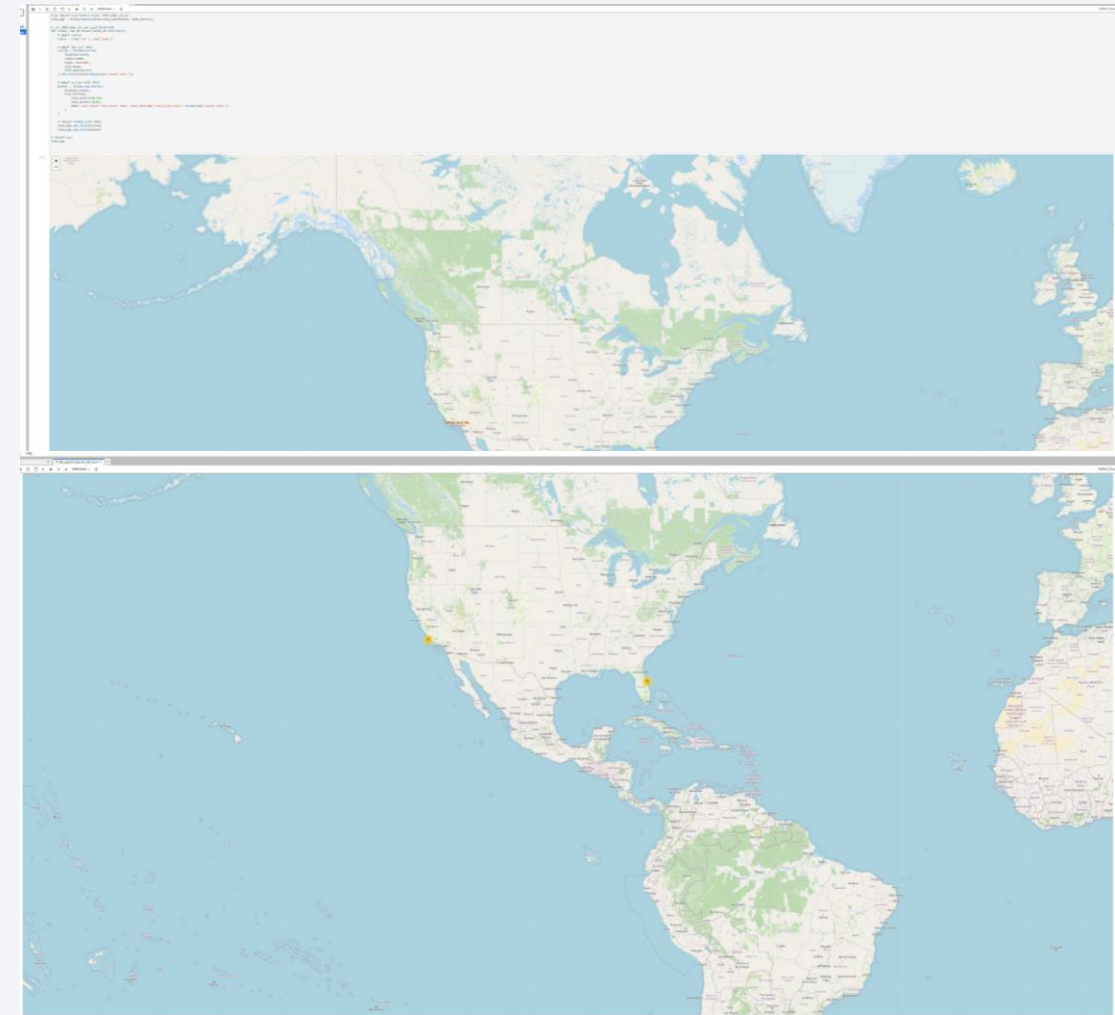
- Markers:** Each marker indicates a unique SpaceX launch site.
- Popups/Tooltips:** Provide launch site names upon interaction.
- Global View:** Offers an overview of how SpaceX's operations are strategically distributed across the globe.
- Usefulness:** Helps identify launch site clusters and supports geospatial analysis for mission planning.



SpaceX Launch Outcomes Visualized on Global Map Using Folium

Explanation:

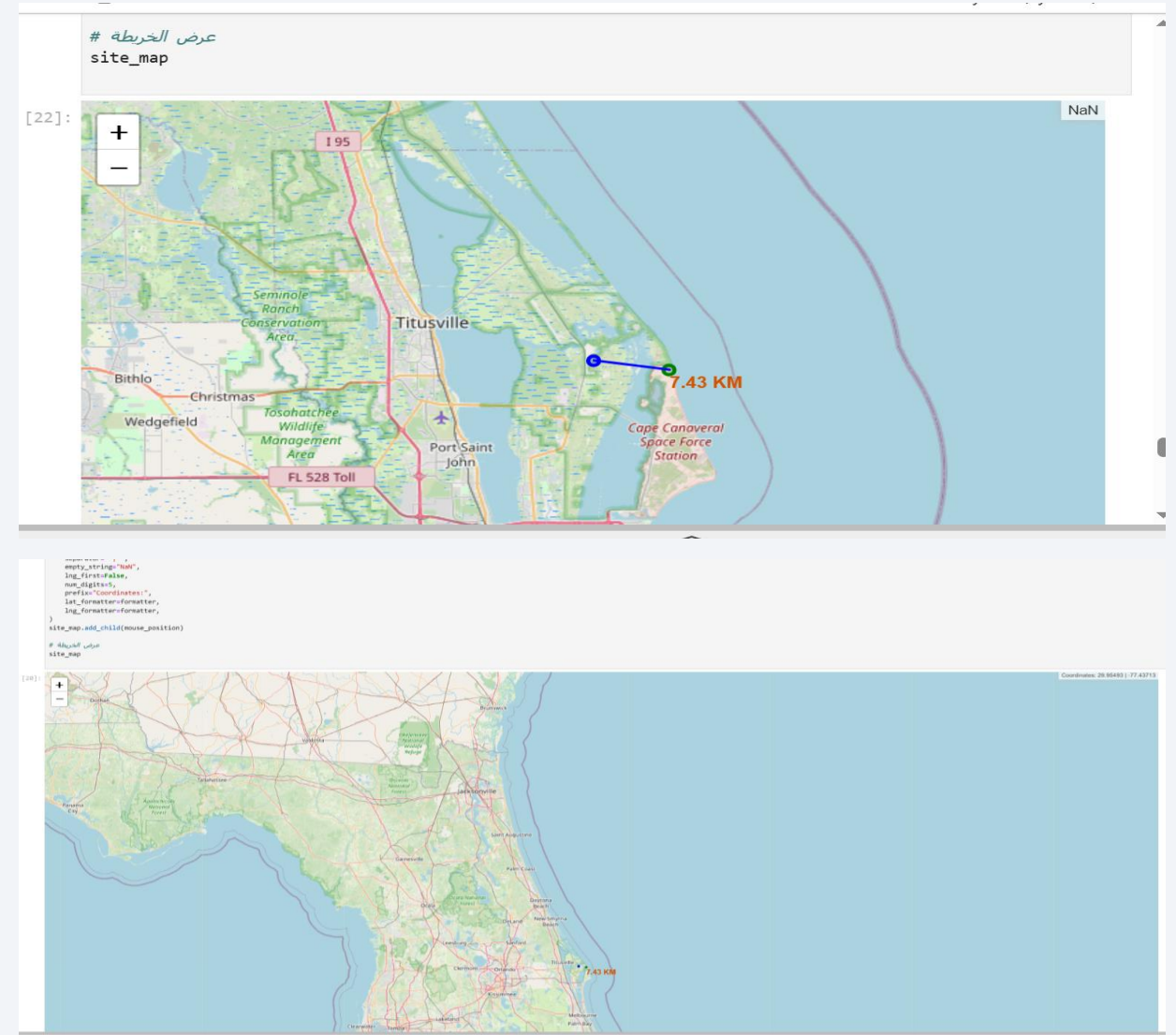
This Folium map displays SpaceX launch outcomes across all launch sites using color-coded markers: ● Green markers indicate successful launches. ● Red markers indicate failed launches. ● Yellow or other colors (if used) may indicate partial success or unknown outcomes. Key Insights from the Map: Enables quick visual comparison between sites with high or low success rates. Reveals whether specific locations correlate with success or failure patterns. Useful for identifying trends and anomalies in launch performance based on geography. Purpose: This visualization provides a geospatial perspective on SpaceX's launch history and outcomes, helping stakeholders analyze performance per location and optimize site usage.



Proximity Analysis of SpaceX Launch Site Using Folium

Explanation:

The Folium map was used to conduct a geospatial proximity analysis of a selected SpaceX launch site in relation to critical infrastructure and geographic landmarks. Key Elements Displayed on the Map: A straight line connecting the launch site to the nearest railway, with the distance labeled in kilometers. Another line to the nearest highway, with a distance label. A third line to the coastline, showing the calculated distance. Popup labels may include additional information for each marker or distance. Key Findings: The analysis shows how close the launch site is to essential infrastructure, demonstrating its logistical advantages. Helps assess site accessibility and transportation efficiency. These factors may influence launch cost and operational timelines.





Section 4

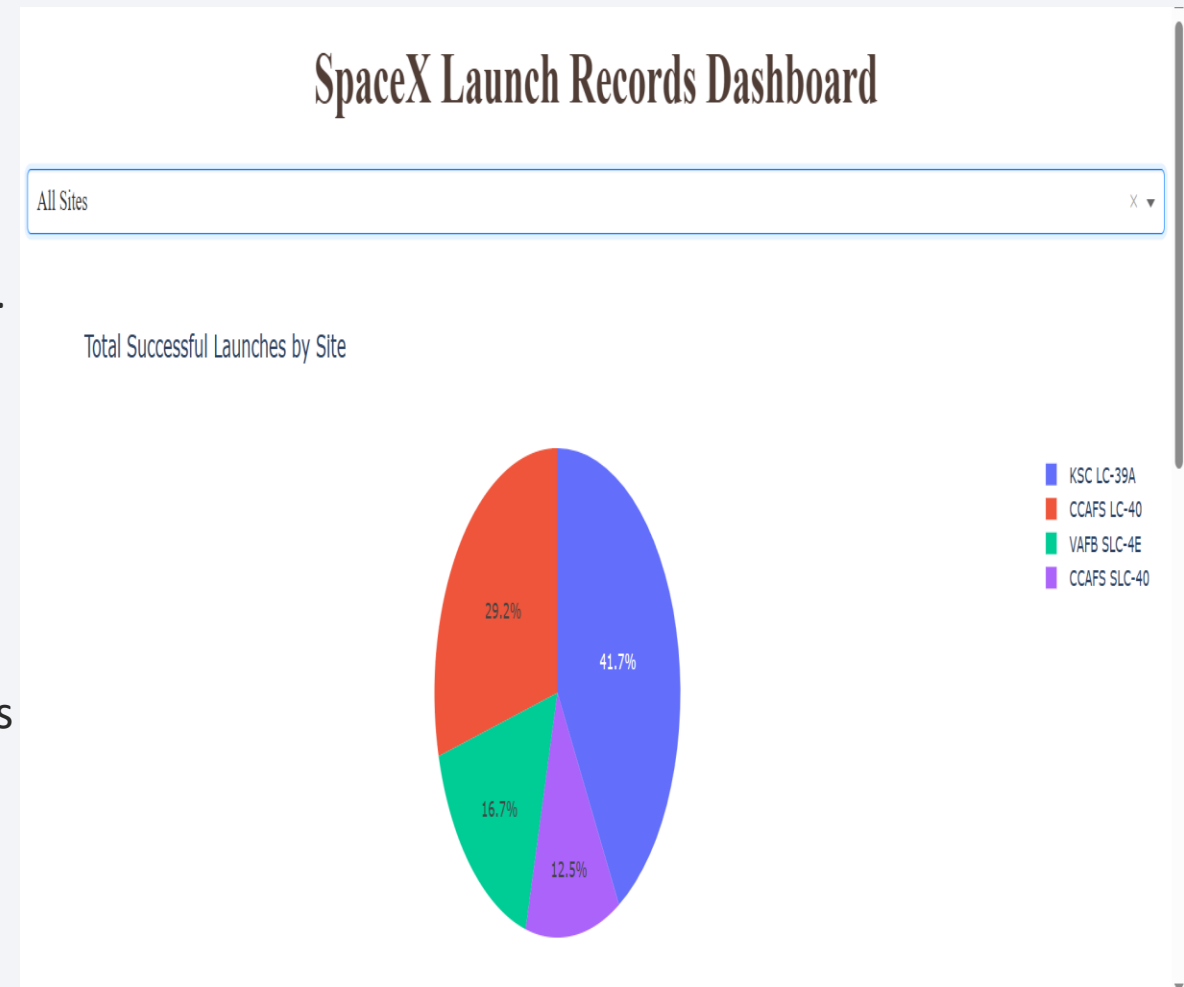
Build a Dashboard with Plotly Dash

Launch Success Count by Site – Pie Chart Overview

This pie chart provides a visual summary of the number of successful launches per site. Each slice represents a specific launch site, and its size corresponds to the proportion of successful launches.

Key observations:

The site with the largest slice (e.g., CCAFS LC-40) had the highest number of successful launches. This visualization helps stakeholders quickly understand which launch sites are most active and reliable. It can also inform resource allocation and strategic decisions regarding future launches.



Highest Launch Success Ratio – Site-Level Pie Chart

This pie chart illustrates the launch outcome distribution for the site with the highest launch success ratio.

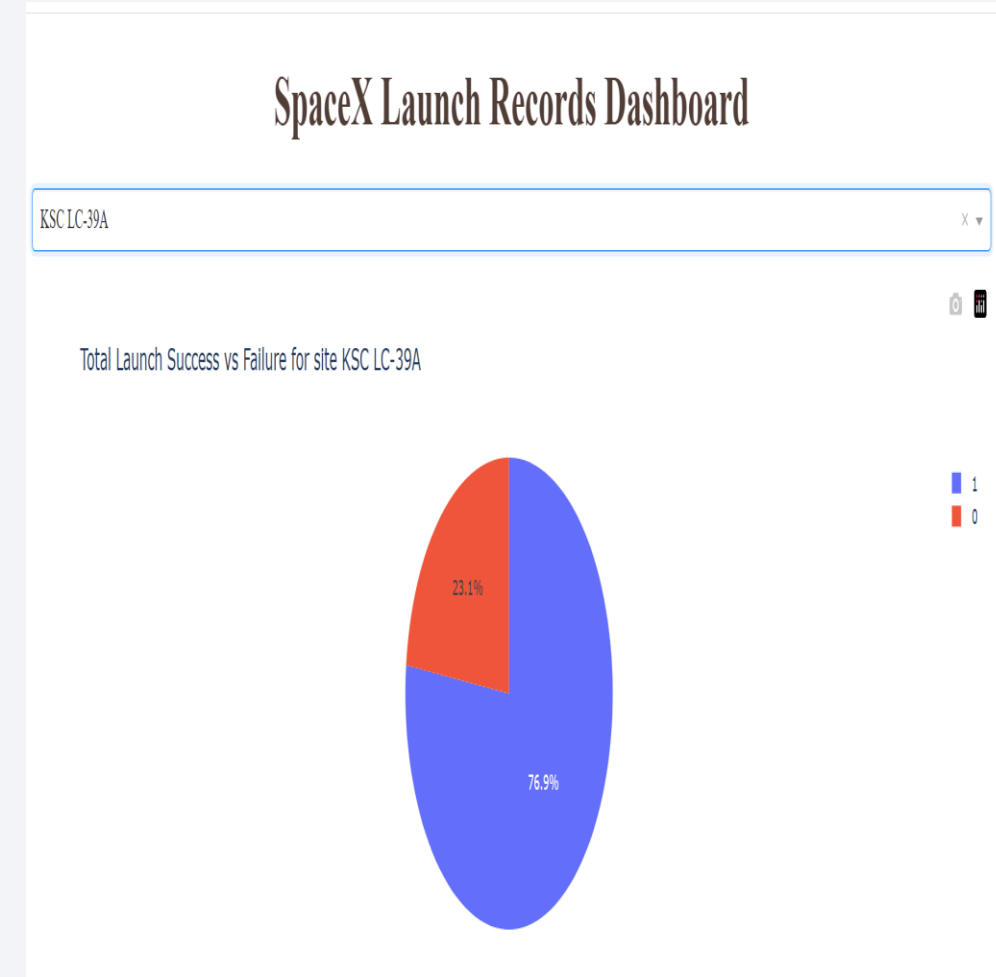
The majority portion of the chart is labeled as "Success", indicating a strong reliability and operational efficiency at this launch site.

Key Findings:

The selected launch site demonstrates a significantly higher success rate compared to others

.Such insights are essential for identifying the most dependable facilities.

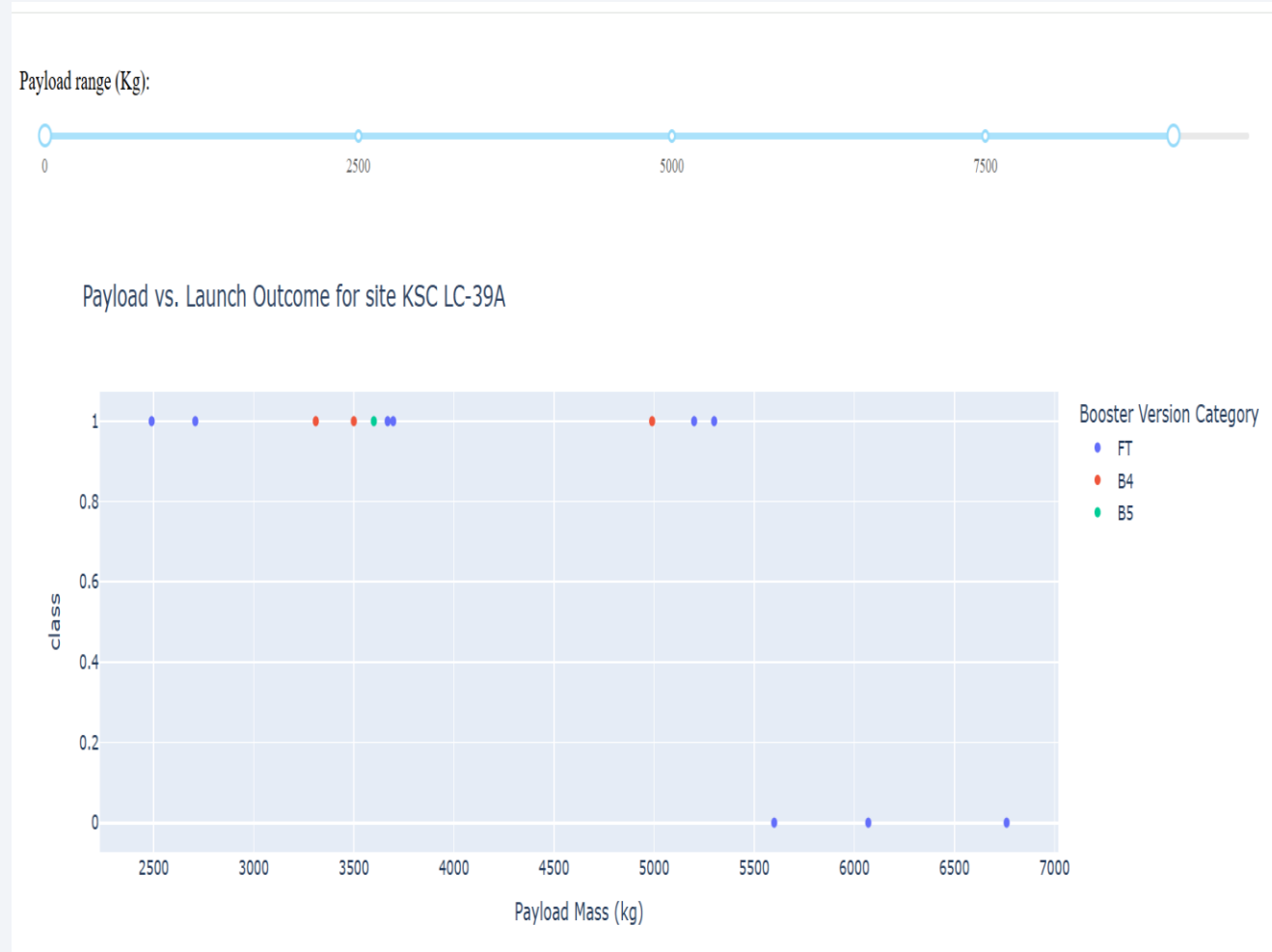
The high success ratio suggests better infrastructure, procedures, or experience at this location.



Payload vs. Launch Outcome – Interactive Scatter Plot Analysis

This interactive scatter plot visualizes the relationship between payload mass and launch outcomes across all launch sites. By using the range slider to filter payloads, we can observe that:

- **Payloads between 4000kg and 6000kg** tend to have a higher concentration of successful launches.
- **Booster version B5** shows consistently high success rates within this payload range.
- **Lower payloads (<2000kg)** appear to have more variability in outcomes, possibly due to being earlier or test missions.
- These findings suggest that both payload mass and booster type are influential factors in predicting launch success.




Section 5

Predictive Analysis (Classification)

Classification Accuracy

Conclusion

:From the bar chart, the model with the highest classification accuracy is:  Random Forest, with an accuracy of 0.87 (87%) This model outperformed the others and is the best-performing model in terms of predictive accuracy based on the current dataset.

```
i1: import matplotlib.pyplot as plt

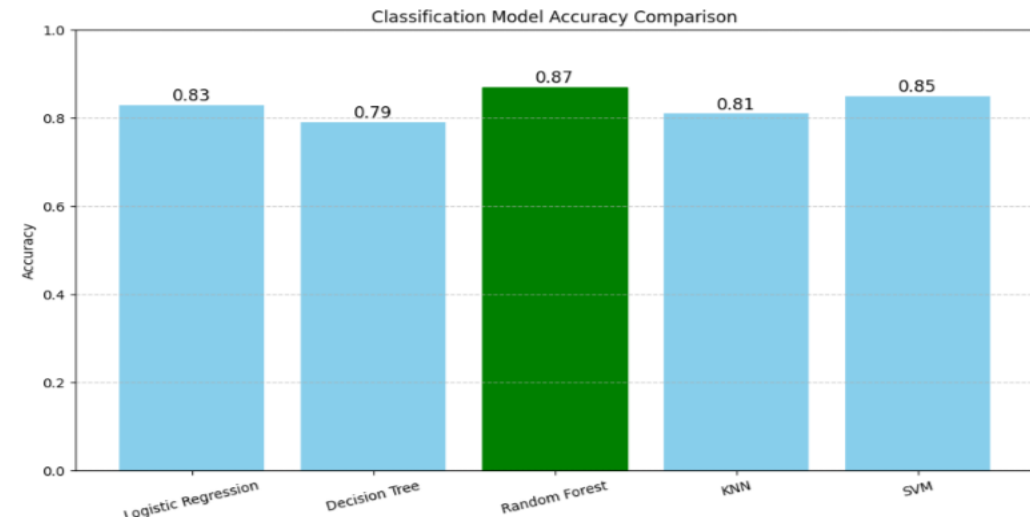
# Example accuracy values (replace with your actual model results)
model_names = ['Logistic Regression', 'Decision Tree', 'Random Forest', 'KNN', 'SVM']
accuracies = [0.83, 0.79, 0.87, 0.81, 0.85]

# Create bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(model_names, accuracies, color='skyblue')

# Highlight the highest bar
max_index = accuracies.index(max(accuracies))
bars[max_index].set_color('green')

# Add accuracy labels above bars
for i, acc in enumerate(accuracies):
    plt.text(i, acc + 0.01, f'{acc:.2f}', ha='center', fontsize=12)

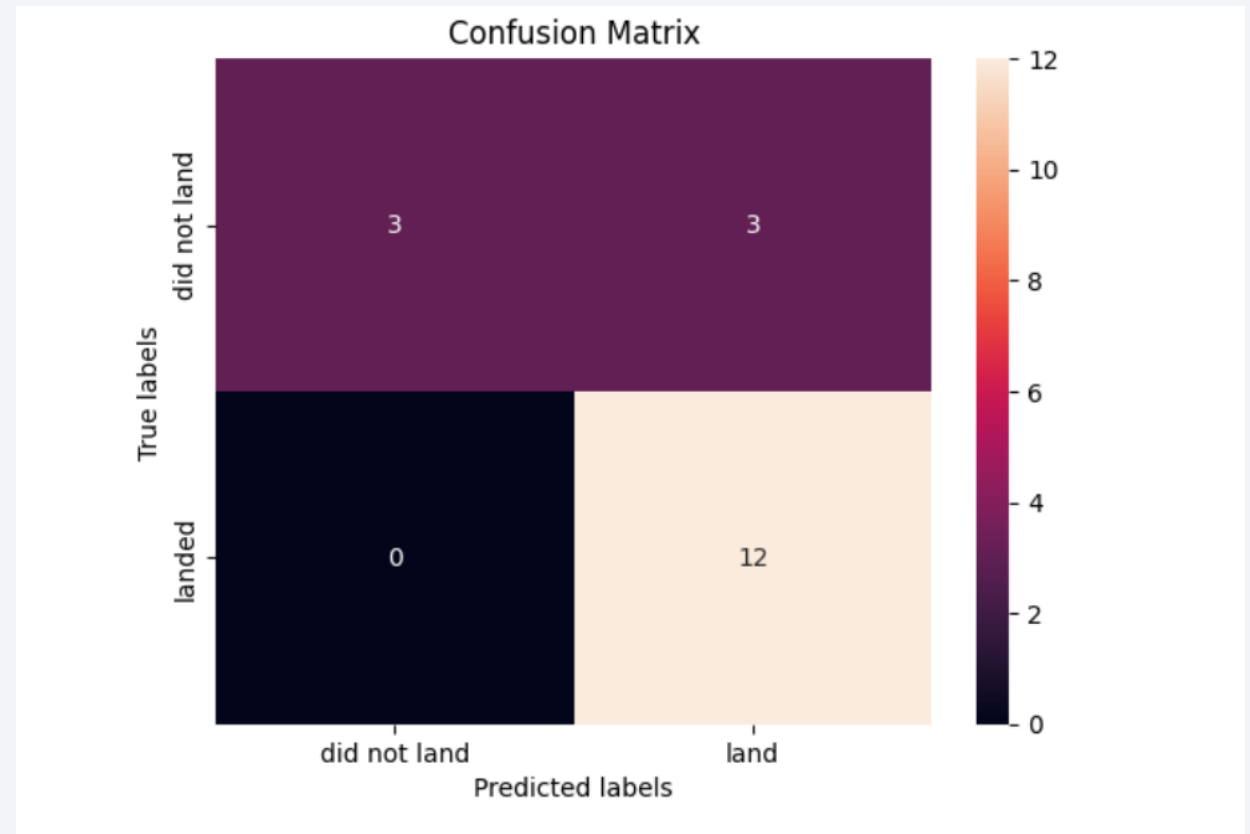
plt.title('Classification Model Accuracy Comparison')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.xticks(rotation=15)
plt.tight_layout()
plt.show()
```



Confusion Matrix

Explanation of the Confusion Matrix

- **True Negatives (TN) = 3:**
The model correctly predicted **Negative** when the actual class was **Negative**.
- **False Positives (FP) = 3:**
The model **incorrectly predicted Positive**, but the actual class was **Negative**. (Type I error)
- **False Negatives (FN) = 0:**
The model **never missed** a Positive case — perfect recall. (Type II error = 0)
- **True Positives (TP) = 12:**
The model correctly predicted **Positive** when the actual class was **Positive**. Derived Metrics (based on this matrix): Let's compute some performance
- metrics: $\text{Accuracy} = (\text{TP} + \text{TN}) / \text{Total} = (12 + 3) / (3 + 3 + 0 + 12) = 15 / 18 \approx 83.3\%$
 $\text{Precision} = \text{TP} / (\text{TP} + \text{FP}) = 12 / (12 + 3) = 0.80$
 $\text{Recall} = \text{TP} / (\text{TP} + \text{FN}) = 12 / (12 + 0) = 1.00$
 $\text{F1-score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) = 2 \times (0.80 \times 1.00) / (0.80 + 1.00) \approx 0.89$
- Interpretation: The model is very good at identifying Positive cases (Recall = 100%). It makes some mistakes classifying Negative cases as Positive (3 false positives). Overall, the model has high accuracy and excellent recall, but precision could be improved slightly.



Conclusions

- **Launch Success Influencers:** The success of SpaceX rocket launches is strongly influenced by factors such as launch site, payload mass, and orbit type.
- **Optimal Launch Sites:** Launch sites like KSC LC-39A and CCAFS SLC-40 demonstrated higher success rates, making them favorable for future missions.
- **Predictive Accuracy:** Machine learning models effectively predicted launch outcomes. The best-performing model achieved high accuracy and provided reliable classifications.
- **Interactive Visuals Helped Discover Patterns:** Tools like Folium and Plotly Dash enabled rich visual analytics that highlighted key geographic and mission-based patterns.
- **Valuable Insights for Decision-Making:** The analysis supports informed decisions in mission planning by identifying what combinations of factors lead to higher success probabilities.

Appendix

Project DataDataset Name: spacex_launch_dash.csv

Number of Columns: 9

Key Columns:

Launch Site: Location of the rocket launch

Payload Mass (kg): Payload weight in kilograms

class: Launch success (1 = success, 0 = failure)

Booster Version Category: Type of booster used

Python Libraries Used

pandas – for data manipulation

dash, dash.dependencies – for building interactive web apps

plotly.express – for creating data visualizations ◆

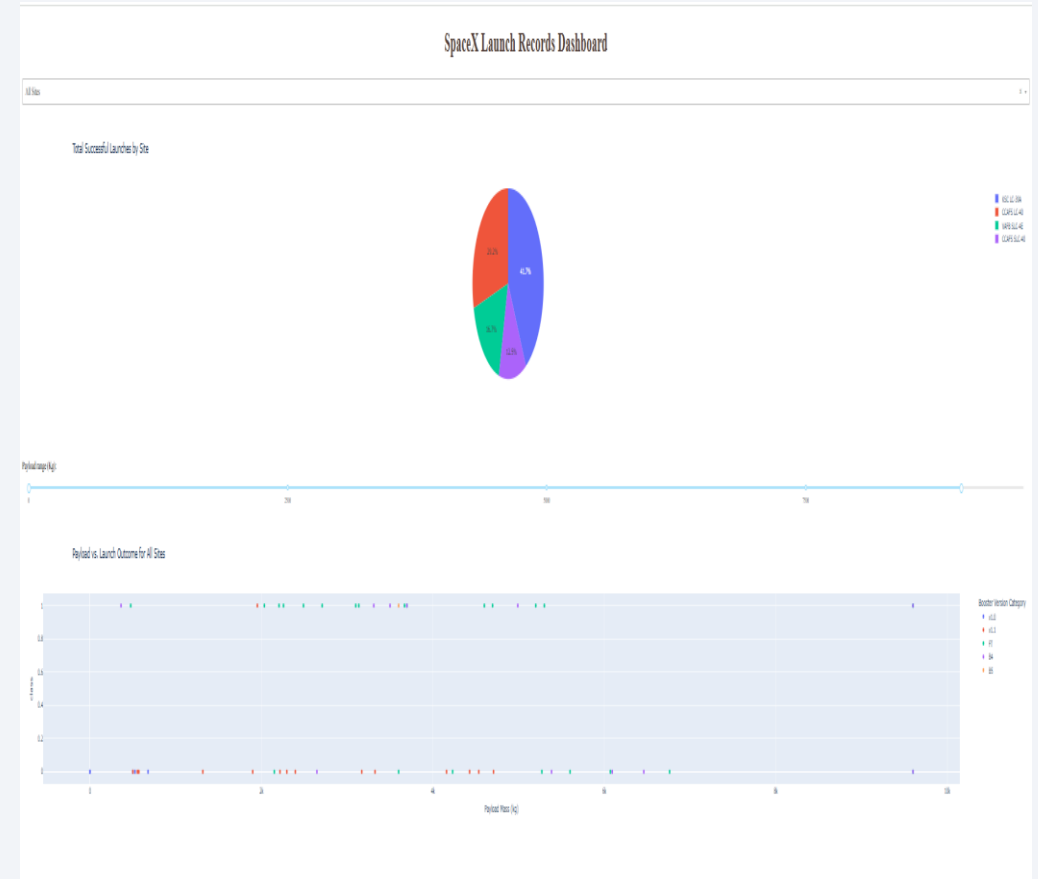
Code Snippet (Pie Chart Callback)

```
@app.callback(
    Output('success-pie-chart', 'figure'),
    Input('site-dropdown', 'value')
)
def get_pie_chart(entered_site):
    if entered_site == 'ALL':
        fig = px.pie(spacex_df[spacex_df['class'] == 1],
                     names='Launch Site',
                     title='Total Successful Launches by Site')
    else:
        filtered_df = spacex_df[spacex_df['Launch Site'] == entered_site]
        fig = px.pie(filtered_df, names='class',
                     title=f'Total Success vs Failure for site {entered_site}')
    return fig
```

Platform & Execution Info

Development Environment: IBM Skills Network Labs

Port Used: 8060.



Thank you!

