

# IBM Data Science Capstone Project –Space X Falcon 9

---



# Overview



---

Executive Summary

Introduction

Methodology

Results

Conclusion

Appendix

# Executive Summary



## Methodologies

- Data Collection (API and Web Scarping)
- Data Wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Machine Learning)

## Results

- Interactive analytics results
- EDA results
- Predictive analysis results

# Introduction



---

## Background

In this project, we aimed to predict whether the first stage of the Falcon 9 rocket would successfully land. SpaceX promotes Falcon 9 launches at a cost of \$62 million per launch, significantly lower than competitors whose prices often exceed \$165 million. This cost advantage is largely due to SpaceX's ability to reuse the first stage of the rocket. By accurately predicting the landing outcome, we could better estimate the overall cost of a launch. Such predictions could be valuable for competing companies considering placing bids against SpaceX. In this lab, our task was to collect data from an API and ensure it is properly formatted for analysis.

## Problem statement

- ✓ What factors determine whether a rocket will successfully land?
- ✓ How do various rocket-related variables influence the likelihood of a successful landing?
- ✓ What specific conditions must SpaceX meet to maximize landing success rates and ensure optimal performance?

# Methodology

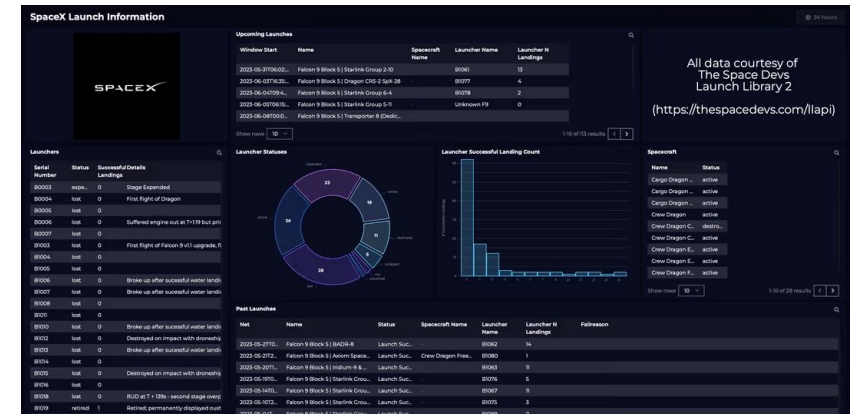
---

## Data Collection and Analysis Approach:

- Data was gathered through the SpaceX API and web scraping from Wikipedia.
- Data wrangling steps were performed to clean and prepare the dataset.
- Categorical features were transformed using one-hot encoding.
- Exploratory Data Analysis (EDA) was conducted using visualizations and SQL queries.
- Interactive visual analytics were implemented using Folium and Plotly Dash.
- Predictive analysis was carried out using various classification models.
- The process included building, tuning, and evaluating classification algorithms for optimal performance.

# Data collection

- ✓ The datasets used in this project were obtained from multiple sources.
- ✓ We primarily worked with SpaceX launch data collected via the SpaceX REST API.
- ✓ This API provides detailed information on rocket launches, including the rocket type, payload, launch parameters, landing details, and outcomes.
- ✓ Our objective was to analyze this data to predict whether SpaceX will attempt to land a rocket.
- ✓ The SpaceX REST API endpoints begin with `api.spacexdata.com/v4/`.
- ✓ In addition, Falcon 9 launch data was also retrieved through web scraping from Wikipedia using the BeautifulSoup library.





# Data collection Space X API

- ✓ We sent a GET request to the SpaceX API to retrieve the data, followed by cleaning, basic data wrangling, and formatting to prepare it for analysis.
- ✓ The notebook containing the full process can be found at the following link:  
<https://github.com/henkums/DataScience/blob/main/jupyter-labs-spacex-data-collection-api.ipynb>.

```
# Use json_normalize method to convert the json result into a dataframe
import requests
import pandas as pd

# Static JSON URL for consistent SpaceX Launch data
static_json_url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/

# Send GET request
response = requests.get(static_json_url)

# Check status code
if response.status_code == 200:
    print("✅ Request successful!")

    # Decode JSON response into a Python object (List of dicts)
    data = response.json()

    # Use json_normalize to flatten nested fields
    df = pd.json_normalize(data)

    # Show shape and first few rows
    print(f"DataFrame contains {df.shape[0]} rows and {df.shape[1]} columns.")
    print(df.head())
else:
    print("❌ Request failed with status code:", response.status_code)

✅ Request successful!
DataFrame contains 107 rows and 42 columns.
   static_fire_date_utc  static_fire_date_unix    tbd   net  window  \
0  2006-03-17T00:00:00.000Z      1.142554e+09  False  False    0.0
1                None                NaN  False  False    0.0
2                None                NaN  False  False    0.0
3  2008-09-20T00:00:00.000Z      1.221869e+09  False  False    0.0
4                None                NaN  False  False    0.0

   rocket  success  \
0  5e9d0d95eda69955f709d1eb  False
1  5e9d0d95eda69955f709d1eb  False
2  5e9d0d95eda69955f709d1eb  False

ed  Python Idle  Mem: 367.12 / 6144.00 MB
```

# Data collection Space X Web Scrapping

- ✓ We used web scraping with BeautifulSoup to extract Falcon 9 launch records.
- ✓ The scraped table was parsed and converted into a pandas DataFrame.
- ✓ You can find the notebook detailing this process at the following link:  
<https://github.com/henkums/DataScience/blob/main/jupyter-labs-webscraping.ipynb>.

```
]]: # use requests.get() method with the provided static_url and headers
static_url = "https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falco
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
}
```

*# assign the response to a object*

```
import requests
```

*# Make the GET request*

```
response = requests.get(static_url, headers=headers)
```

*# Print status to confirm success*

```
print("Status code:", response.status_code)
```

```
from bs4 import BeautifulSoup
```

*# Parse HTML content*

```
soup = BeautifulSoup(response.content, 'html.parser')
```

Status code: 200

Create a BeautifulSoup object from the HTML response

```
]]: # Use BeautifulSoup() to create a BeautifulSoup object from a response
from bs4 import BeautifulSoup
```

*# Create a BeautifulSoup object from the HTML response*

```
soup = BeautifulSoup(response.text, 'html.parser')
```

*# Print first 500 characters of parsed HTML*

```
print(soup.prettify()[:500])
```

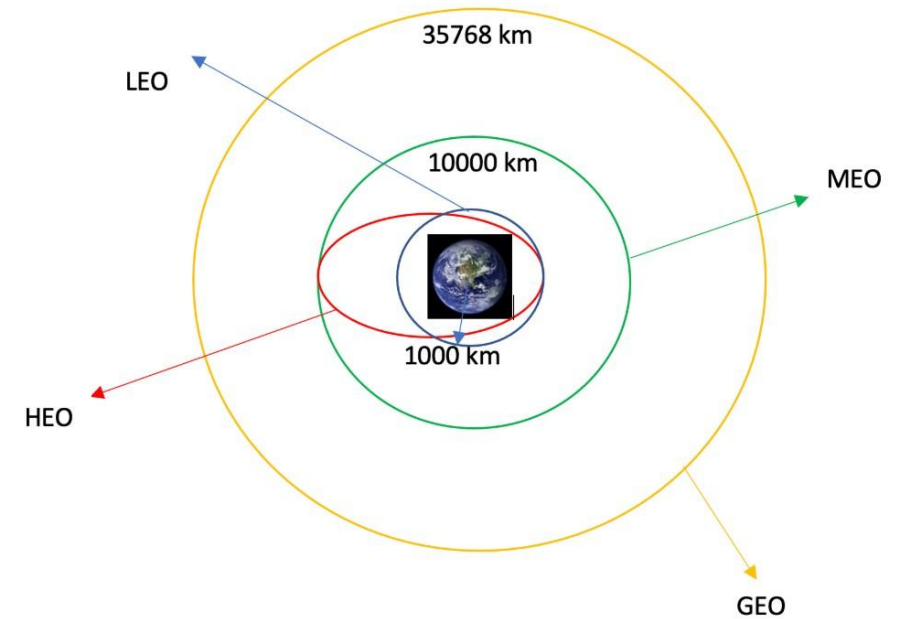
<!DOCTYPE html>

<html class="client-nojs vector-feature-language-in-header-enabled vec



# Data Wrangling Space X

- ✓ In the dataset, there are multiple instances where the booster failed to land successfully. Sometimes a landing was attempted but resulted in failure due to an accident. For example, **True Ocean** indicates the mission successfully landed in a specific ocean region, while **False Ocean** means it did not. Similarly, **True RTLS** signifies a successful landing on a ground pad, whereas **False RTLS** indicates an unsuccessful attempt. **True ASDS** means the booster successfully landed on a drone ship, and **False ASDS** means the landing on the drone ship was unsuccessful.
- ✓ For modeling purposes, these outcomes are simplified into training labels where **1** represents a successful landing and **0** represents a failure .
- ✓ You can find the notebook detailing this process at the following link:  
<https://github.com/henkums/DataScience/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb>



# Data Visualization Space X

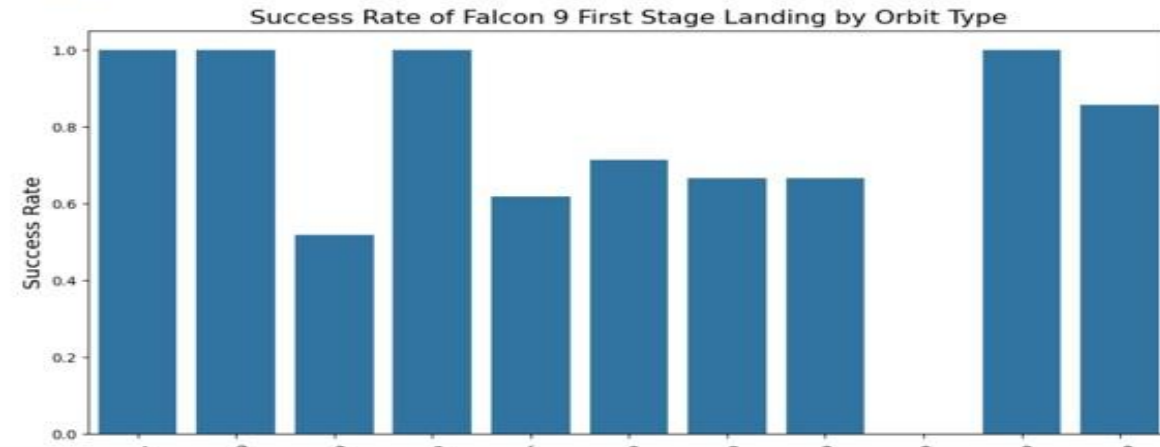
## Observations:

- ✓ Launches targeting **LEO**, **ISS**, and **SSO** tend to have **high landing success rates**, often above **90%**, due to their lower complexity and frequent use.
- ✓ Missions to **GTO** and **HEO** generally show **lower success rates**, as they involve **higher energy demands** and **more complex trajectories**.
- ✓ Overall, **orbit type has a significant impact** on landing outcomes — simpler, more practiced missions yield better results.
- ✓ You can find the notebook detailing this process at the following link:  
<https://github.com/henkums/DataScience/blob/main/edadataviz.ipynb>

```
[7]: # HINT use groupby method on Orbit column and get the mean of Class column
import matplotlib.pyplot as plt

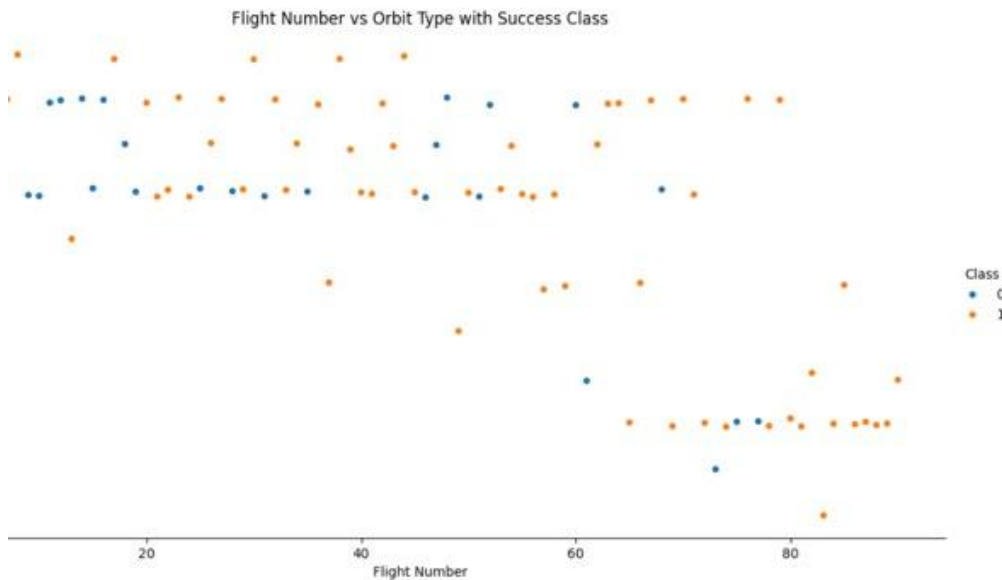
# Calculate success rate for each orbit type
success_rate_orbit = df.groupby('Orbit')['Class'].mean().reset_index()

# Plotting
plt.figure(figsize=(12,6))
sns.barplot(x='Orbit', y='Class', data=success_rate_orbit)
plt.xlabel('Orbit Type', fontsize=14)
plt.ylabel('Success Rate', fontsize=14)
plt.title('Success Rate of Falcon 9 First Stage Landing by Orbit Type', fontsize=16)
plt.xticks(rotation=45)
plt.show()
```



# Data Visualization Space X

```
# Plot with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value  
plt.scatter(FlightNumber, Orbit, hue=Class, data=df, aspect=2, height=6, kind="strip")  
plt.title("Orbit Type with Success Class")
```



Orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

- ✓ LEO Missions: Show a positive correlation between the number of flights and success rate. This suggests that SpaceX improves over time with experience on these less complex missions.
- ✓ GTO Missions: Show no clear correlation between flight number and success. This implies that inherent mission complexity and variability make success less dependent on experience.
- ✓ Conclusion: The type of orbit impacts landing success trends – LEO missions benefit from learning and repetition, while GTO missions remain consistently challenging regardless of flight count.
- ✓ You can find the notebook detailing this process at the following link:  
<https://github.com/henkums/DataScience/blob/main/edadataviz.ipynb>



We utilized SQL queries to extract and analyze key insights from the dataset. These queries helped us answer various questions related to SpaceX missions, including:

Retrieving the unique names of launch sites used in missions.

Fetching the first 5 records where the launch site name starts with 'KSC'.

Calculating the total payload mass carried by boosters launched for NASA (CRS) missions.

Determining the average payload mass for the booster version F9 v1.1.

Identifying the dates when successful landings on drone ships occurred.

Listing boosters that successfully landed on ground pads and carried payloads between 4000 and 6000 kg.

Counting the number of successful and failed mission outcomes.

Finding the booster versions that carried the maximum payload mass.

Displaying records that include month names, successful ground pad landings, booster versions, and launch sites for missions launched in 2017.

Ranking the success count of landings between June 4, 2010, and March 20, 2017, in descending order.

These SQL-based insights supported deeper understanding and exploration of the SpaceX launch data.

# Data with SQL Space X

---

# Data Interactive Map with Folium



We visualized all SpaceX launch sites using Folium, adding interactive map elements such as markers, circles, and lines to indicate the success or failure of each launch.

Launch outcomes were encoded as binary classes: 0 for failure and 1 for success.

By using color-coded marker clusters, we were able to visually assess which sites had higher success rates.

We also calculated the distance from each launch site to nearby infrastructure and landmarks, such as railways, highways, coastlines, and cities.

This allowed us to explore spatial relationships and answer key questions, such as:

- Are launch sites located close to railways, highways, and coastlines?
- Do launch sites maintain a safe



# Data Dashboard with Plotly Dash

The dashboard is hosted on PythonAnywhere, providing 24/7 live access for users to explore and interact with the data.

It was developed using Flask and Dash, lightweight and powerful Python web frameworks.

## Graphs and Visualizations:

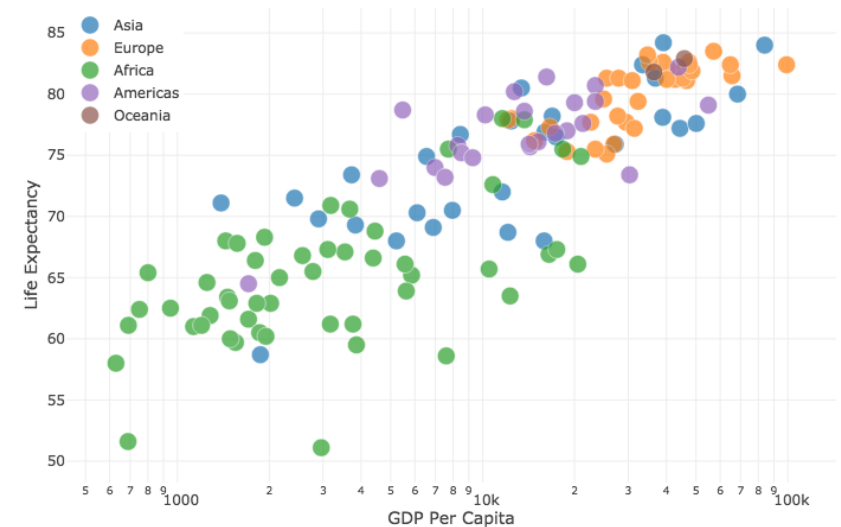
### Scatter Plot:

- Displays the relationship between launch outcomes and payload mass (Kg) across different booster versions.
- Ideal for revealing non-linear trends and patterns.
- Clearly shows the range (minimum to maximum values) and helps in identifying outliers.
- Provides an intuitive way to analyze variable relationships.

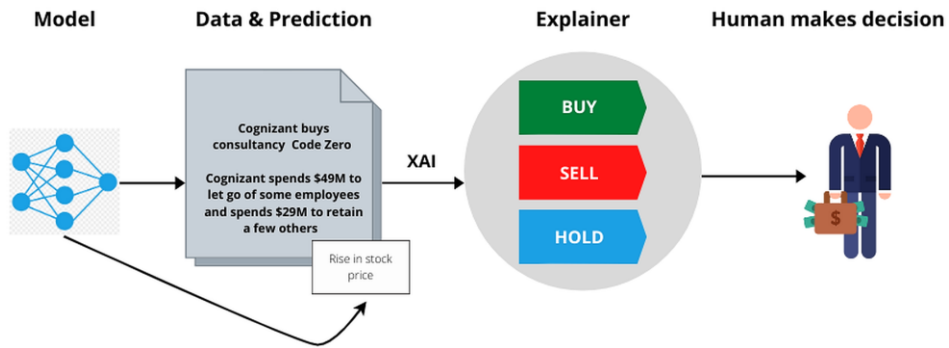
### Pie Chart:

- Visualizes the distribution of total launches by individual sites or across all sites.
- Useful for comparing proportions between different launch sites.

The size of each slice represents the relative quantity of launches per site.



# Data Prediction Analysis (Classification)



## Model Building Process

- ✓ Load the dataset using NumPy and Pandas.
- ✓ Perform necessary data transformations.
- ✓ Split the data into training and testing sets.
- ✓ Determine the number of test samples available.
- ✓ Choose appropriate machine learning algorithms for classification.
- ✓ Define hyperparameters and use GridSearchCV for model tuning.
- ✓ Fit the models using the training set and evaluate them with cross-validation.

## Model Evaluation

- ✓ Measure the accuracy of each model.
- ✓ Extract the best hyperparameters for each algorithm.
- ✓ Visualize model performance using a confusion matrix.

## Model Improvement

- ✓ Apply feature engineering to enhance data inputs.
- ✓ Fine-tune model algorithms and hyperparameters for better results.

## Identifying the Best Performing Classifier

- ✓ The model achieving the highest accuracy score is considered the best performer.
- ✓ A summary dictionary of models and their scores is provided at the end of the notebook.

# Data Results



- Findings from Exploratory Data Analysis (EDA)
- Screenshots showcasing interactive visual analytics
- Outcomes from predictive modeling and analysis

# Data Insights for Visualization Space X

---

## Section 3

Visualization:  
g Insights  
onable  
strategies



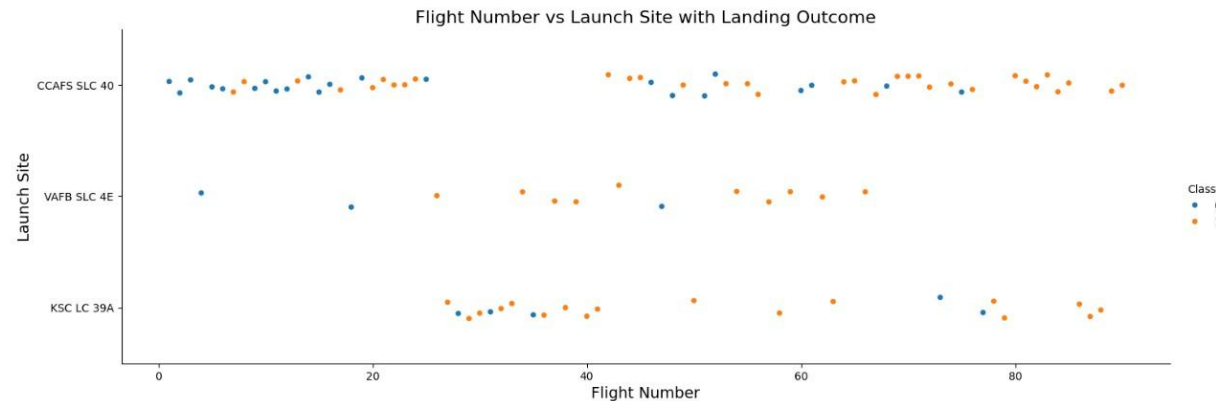


# Flight Number vs. Launch Site

The Flight Number vs. Launch Site scatter plot reveals clear trends in SpaceX's launch history. In the early phases of the program, most launches took place at CCAFS SLC 40, with several failures (Class = 0) concentrated among the lower flight numbers, reflecting SpaceX's experimental and developmental stage. Over time, as the flight numbers increase, launches become consistently successful (Class = 1) across all sites, indicating improved reliability and maturity of the Falcon 9 program. VAFB SLC 4E shows fewer, more sporadic launches, likely due to its use for specific polar orbit missions. Notably, KSC LC 39A appears later in the timeline and is associated almost exclusively with successful launches, highlighting its role in high-profile, advanced missions. Overall, the plot illustrates SpaceX's technological progression and strategic use of launch sites over time.

```
[5]: # Plot a scatter point chart with x axis to be Flight Number and y axis to be the Launch site, and hue to be the class value
import seaborn as sns
import matplotlib.pyplot as plt

sns.catplot(x="FlightNumber", y="LaunchSite", hue="Class", data=df, kind="strip", aspect=3)
plt.xlabel("Flight Number", fontsize=14)
plt.ylabel("Launch Site", fontsize=14)
plt.title("Flight Number vs Launch Site with Landing Outcome", fontsize=16)
plt.show()
```

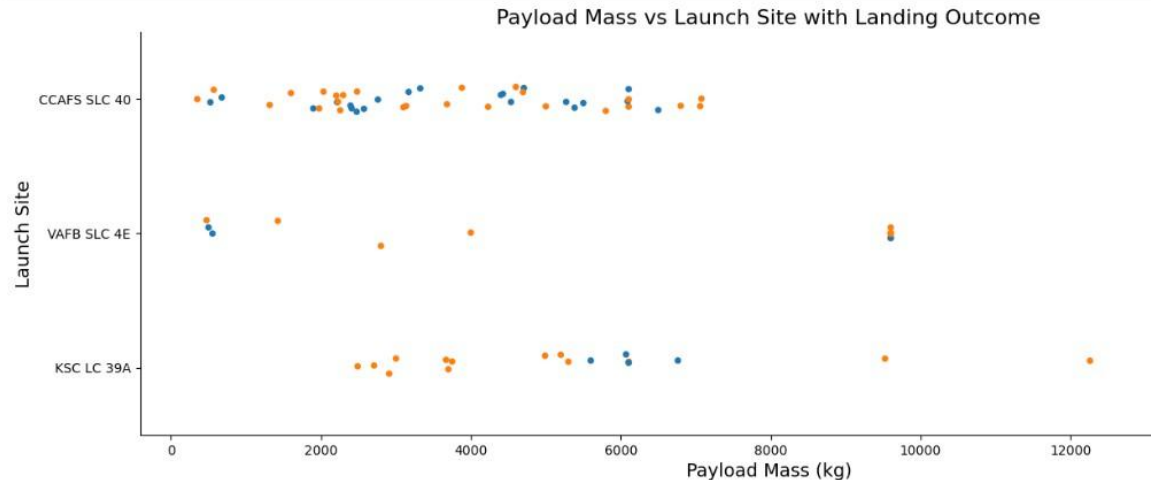




# Payload vs. Launch Site

```
[6]: # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the Launch site, and hue to l
import seaborn as sns
import matplotlib.pyplot as plt

sns.catplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df, kind="strip", aspect=3)
plt.xlabel("Payload Mass (kg)", fontsize=14)
plt.ylabel("Launch Site", fontsize=14)
plt.title("Payload Mass vs Launch Site with Landing Outcome", fontsize=16)
plt.show()
```



- The Payload Mass vs. Launch Site plot reveals how SpaceX allocated missions across its various launch sites. CCAFS SLC 40, used since the early days, shows the broadest range of payload masses and includes both successful and failed missions, reflecting its role in SpaceX's developmental phase. In contrast, KSC LC 39A primarily handles heavier payloads, with all launches from this site in the dataset being successful, indicating it is reserved for more critical or advanced missions. VAFB SLC 4E handles fewer launches with mid-sized payloads, likely for specific orbital requirements, and also shows a high success rate. Overall, the plot suggests a strategic use of launch sites based on mission complexity and payload weight.

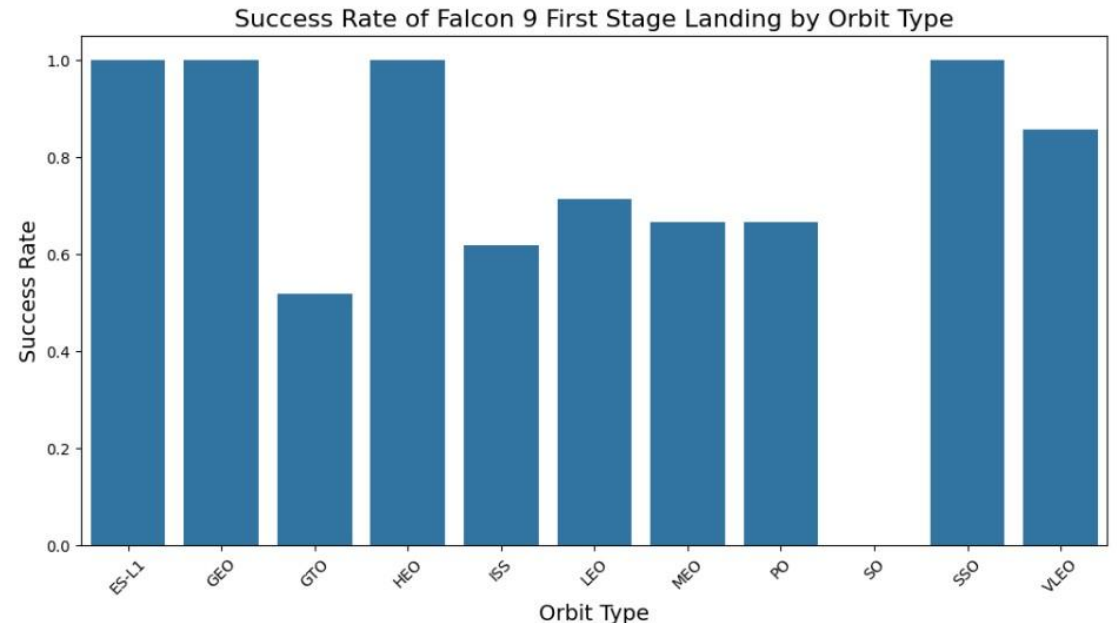
# Success Rate vs. Orbit Type

The bar chart illustrating the success rate by orbit type reveals clear differences in Falcon 9's first stage landing performance across various mission profiles. Common orbits like LEO (Low Earth Orbit) and ISS (International Space Station) show consistently high success rates, reflecting SpaceX's extensive experience and operational maturity with these mission types. More complex or less frequently used orbits such as GTO (Geostationary Transfer Orbit) may exhibit slightly lower success rates, likely due to the increased technical challenges associated with these missions. Overall, the visualization highlights how mission complexity and orbit type influence the likelihood of a successful first stage landing, emphasizing SpaceX's growing reliability in routine orbits and ongoing efforts to master more demanding trajectories.

```
[7]: # HINT use groupby method on Orbit column and get the mean of Class column
import matplotlib.pyplot as plt

# Calculate success rate for each orbit type
success_rate_orbit = df.groupby('Orbit')['Class'].mean().reset_index()

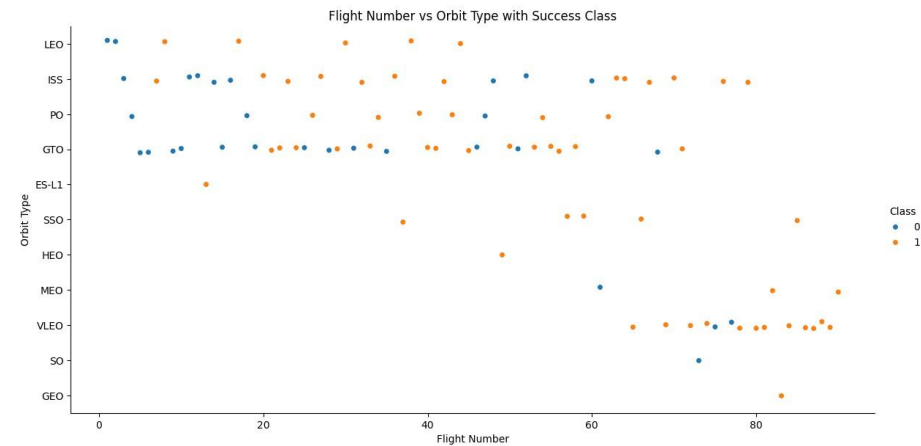
# Plotting
plt.figure(figsize=(12,6))
sns.barplot(x='Orbit', y='Class', data=success_rate_orbit)
plt.xlabel('Orbit Type', fontsize=14)
plt.ylabel('Success Rate', fontsize=14)
plt.title('Success Rate of Falcon 9 First Stage Landing by Orbit Type', fontsize=16)
plt.xticks(rotation=45)
plt.show()
```



# Flight Number vs. Orbit Type

The scatter plot of Flight Number vs Orbit Type highlights SpaceX's evolving mission complexity and reliability over time. In the early phases, missions to simpler orbits like LEO were more common and included several failures as the company refined its technology. As the Flight Number increases, indicating later missions, the company successfully expanded into more challenging orbits like GTO and ISS, with noticeably fewer failures. Specialized orbits appear mainly in the later flights and are largely successful, reflecting SpaceX's growing expertise. Overall, the plot illustrates a clear progression from low-risk missions toward complex, high-value orbits as SpaceX matured its launch capabilities.

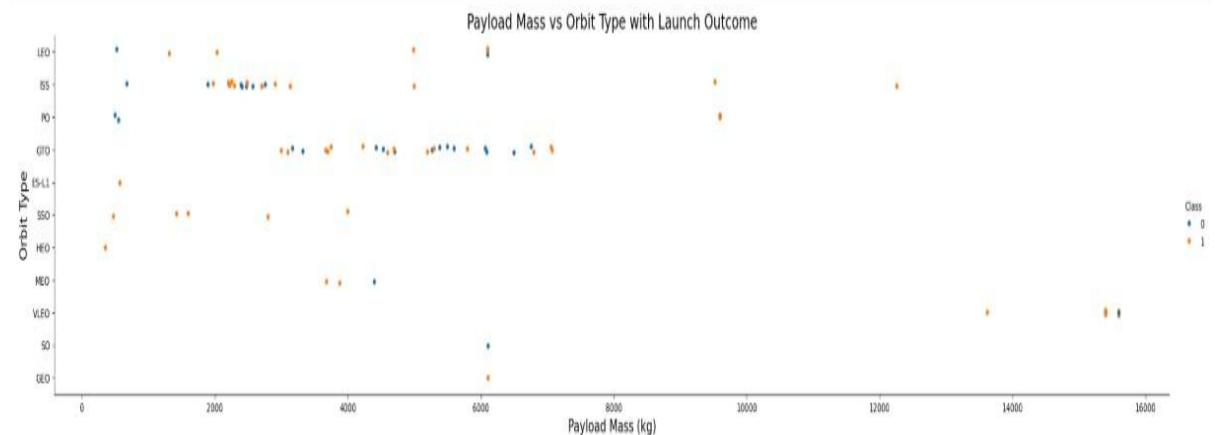
```
[9]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="FlightNumber", y="Orbit", hue="Class", data=df, aspect=2, height=6, kind="strip")
plt.title("Flight Number vs Orbit Type with Success Class")
plt.xlabel("Flight Number")
plt.ylabel("Orbit Type")
plt.show()
```



# Payload vs. Orbit Type

The Payload Mass vs Orbit Type plot reveals that different orbit types are associated with distinct payload mass ranges. LEO missions show the greatest variability, supporting both light and heavy payloads, while GTO missions typically involve heavier payloads, indicating more complex satellite deployments. ISS missions cluster around mid-range payload masses and are consistently successful, reflecting high mission standards. Other specialized orbits handle mostly lighter payloads and appear less frequently in the dataset. Overall, the plot demonstrates how payload mass varies depending on orbital destination, and how SpaceX has achieved high success rates across a range of mission profiles and payload sizes.

```
[10]: # 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, aspect=5, kind='strip')
plt.xlabel("Payload Mass (kg)", fontsize=15)
plt.ylabel("Orbit Type", fontsize=15)
plt.title("Payload Mass vs Orbit Type with Launch Outcome", fontsize=18)
plt.show()
```



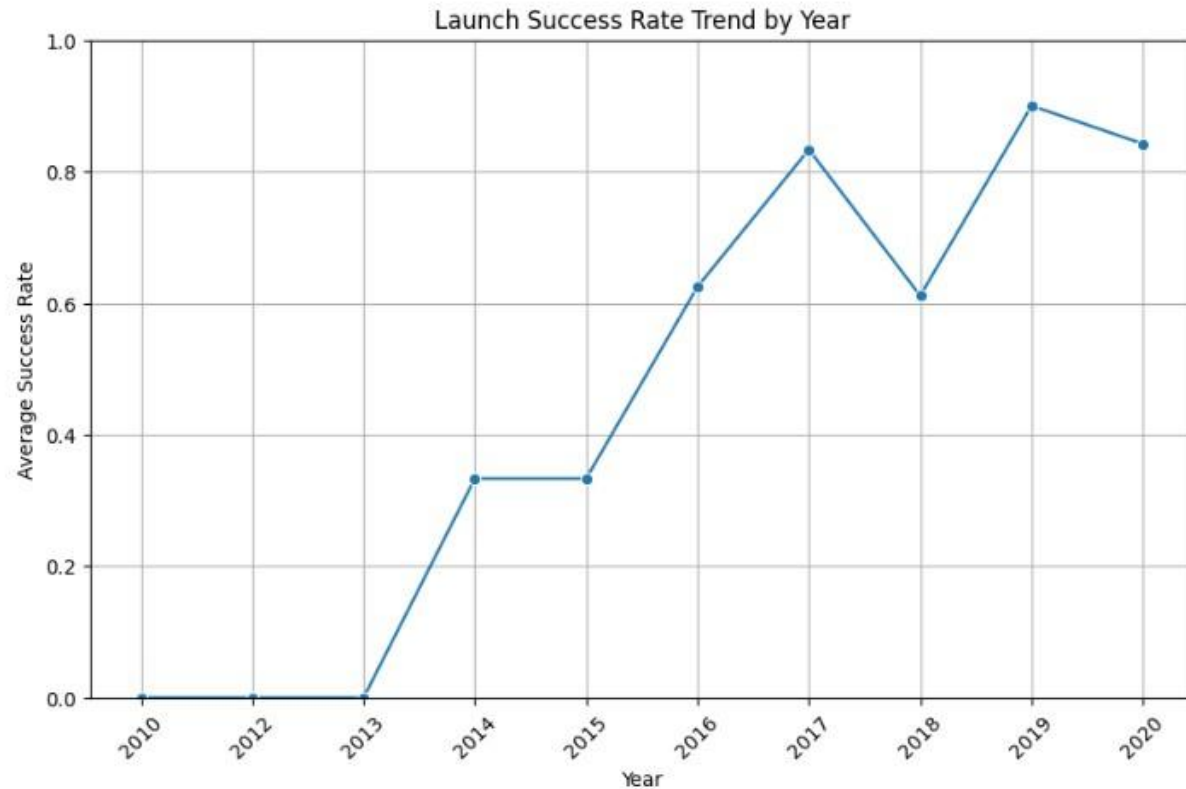
# Launch Success Yearly Trend

The line chart showing SpaceX's **yearly launch success rate** reveals a clear upward trend in reliability over time. Early years (2010–2013) were marked by relatively low and inconsistent success rates, reflecting the company's initial learning curve. From 2014 onward, the success rate began to improve significantly, with only occasional setbacks. In the most recent years, the success rate consistently approached or reached **100%**, indicating SpaceX's transition into a **highly reliable launch provider**. This trend highlights the company's rapid technological advancement, process refinement, and operational excellence over the span of a decade.

```
[12]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
# Calculate average success rate (mean of 'Class') grouped by year
success_rate_by_year = df.groupby('Date')['Class'].mean().reset_index()

# Rename columns for clarity
success_rate_by_year.columns = ['Year', 'SuccessRate']

# Plot the yearly success rate trend
plt.figure(figsize=(10,6))
sns.lineplot(data=success_rate_by_year, x='Year', y='SuccessRate', marker='o')
plt.title('Launch Success Rate Trend by Year')
plt.xlabel('Year')
plt.ylabel('Average Success Rate')
plt.xticks(rotation=45)
plt.ylim(0, 1)
plt.grid(True)
plt.show()
```





# All Launch Site Names

---

We used the DISTINCT keyword to retrieve only the unique launch site names from the SpaceX dataset.

## Task 1

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

```
[10]: %%sql
      SELECT DISTINCT "Launch_Site"
      FROM SPACEXTABLE;
```

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

Done.

```
[10]: Launch_Site
```

```
CCAFS LC-40
```

```
VAFB SLC-4E
```

```
KSC LC-39A
```

```
CCAFS SLC-40
```

# Launch Site Names Begin with 'CCA'

## Task 2

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

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

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Out
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	St
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	St
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	St
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	St
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	St

- We used the **LIKE 'CCA%'** condition to filter and display only the records where the launch site name **begins with 'CCA'**, which corresponds to **Cape Canaveral Air Force Station** locations.

# Total Payload Mass

---

We used the **SUM()** function to calculate the **total payload mass** of all launches conducted for **NASA (CRS)**. The **WHERE** clause filters the dataset to include only rows where the **Customer** is 'NASA (CRS)'. The result shows that **45,596 kg** of payload was launched for NASA under the **Commercial Resupply Services (CRS)** program.

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

```
[11]: %%sql
      SELECT SUM("PAYLOAD_MASS_KG_") AS total_payload_mass_kg
      FROM SPACEXTABLE
      WHERE Customer = 'NASA (CRS)';
```

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

Done.

```
[11]: total_payload_mass_kg
```

```
45596
```

# Average Payload Mass by F9 v1.1

---

The query calculates the **average payload mass** carried by the booster version **F9 v1.1** using the AVG() function. This means on average; this version of Falcon 9 carried approximately **2,928.4 kg** per mission.

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

```
[12]: %%sql
      SELECT AVG("PAYLOAD_MASS_KG_") AS average_payload_mass_kg
      FROM SPACEXTABLE
      WHERE "Booster_Version" = 'F9 v1.1';
```

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

Done.

```
[12]: average_payload_mass_kg
```

```
2928.4
```

# Successful Drone Ship Landing with Payload between 4000 and 6000

We queried for booster versions that had a **successful drone ship landing** (Landing Outcome LIKE '%Success%' AND '%drone ship%') Carried a **payload mass between 4000 and 6000 kg** The result includes **four unique booster versions** of the **Falcon 9 Full Thrust (F9 FT)** series that met these criteria.

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

```
[15]: %%sql
SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE '%Success%'
AND "Landing_Outcome" LIKE '%drone ship%'
AND "PAYLOAD_MASS_KG_" > 4000
AND "PAYLOAD_MASS_KG_" < 6000;
```

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

```
[15]: Booster_Version
```

```
F9 FT B1022
```

```
F9 FT B1026
```

```
F9 FT B1021.2
```

```
F9 FT B1031.2
```



# Total Number of Successful and Failure Mission Outcomes

- ✓ Success: 99 missions.
- ✓ Failure (in flight): 1 mission.
- ✓ Success (payload status unclear): 1 mission (potential partial success).

List the total number of successful and failure mission outcomes

```
[17]: %%sql
SELECT TRIM("Mission_Outcome") AS outcome_cleaned, COUNT(*) AS total_missions
FROM SPACEXTABLE
GROUP BY outcome_cleaned;
```

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

```
[17]:
```

outcome_cleaned	total_missions
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

# Boosters Carried Maximum Payload

To identify the boosters that carried the heaviest payloads, we used a subquery with the aggregate function MAX() to find the maximum payload mass in the dataset. Then, we retrieved all booster versions that match this value.

The query revealed that multiple missions using the Falcon 9 Block 5 (F9 B5) series carried payloads of 15,600 kg, which is the maximum payload mass recorded. These missions demonstrate the advanced lifting capabilities and consistent performance of the Block 5 boosters, which are designed for high efficiency and reusability.

```
[19]: %%sql
SELECT "Booster_Version", "PAYLOAD_MASS_KG_"
FROM SPACEXTABLE
WHERE "PAYLOAD_MASS_KG_" = (
    SELECT MAX("PAYLOAD_MASS_KG_") FROM SPACEXTABLE
);
```

\* sqlite:///my\_data1.db

Done.

[19]:	Booster_Version	PAYLOAD_MASS_KG_
	F9 B5 B1048.4	15600
	F9 B5 B1049.4	15600
	F9 B5 B1051.3	15600
	F9 B5 B1056.4	15600
	F9 B5 B1048.5	15600
	F9 B5 B1051.4	15600
	F9 B5 B1049.5	15600
	F9 B5 B1060.2	15600
	F9 B5 B1058.3	15600
	F9 B5 B1051.6	15600
	F9 B5 B1060.3	15600
	F9 B5 B1049.7	15600

# 2015 Launch Records

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'

```
[18]: %%sql
SELECT
  CASE substr("Date", 6, 2)
    WHEN '01' THEN 'January'
    WHEN '02' THEN 'February'
    WHEN '03' THEN 'March'
    WHEN '04' THEN 'April'
    WHEN '05' THEN 'May'
    WHEN '06' THEN 'June'
    WHEN '07' THEN 'July'
    WHEN '08' THEN 'August'
    WHEN '09' THEN 'September'
    WHEN '10' THEN 'October'
    WHEN '11' THEN 'November'
    WHEN '12' THEN 'December'
  END AS Month,
  "Landing_Outcome",
  "Booster_Version",
  "Launch_Site"
FROM SPACEXTABLE
WHERE substr("Date", 1, 4) = '2015'
  AND "Landing_Outcome" LIKE '%Failure%'
  AND "Landing_Outcome" LIKE '%drone ship%';
```

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

```
[18]:
```

Month	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- ✓ In 2015, SpaceX experienced two failed landing attempts on drone ships:

In January, using F9 v1.1 B1012

In April, using F9 v1.1 B1015

- ✓ Both missions launched from **Cape Canaveral (CCAFS LC-40)**. These failures were part of the early experimental phase of Falcon 9's booster recovery efforts, which contributed to rapid improvements in landing technology in the following years.

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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 201

```
[19]: %%sql
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;
```

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

```
[19]:
```

Landing_Outcome	outcome_count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Between **June 2010** and **March 2017**, the most frequent outcome was **"No attempt"** (10 missions), which reflects SpaceX's early launch phase when recovery was not yet attempted.

Notably:

- **Drone ship landings** had **equal counts of success and failure** (5 each), highlighting the trial-and-error nature of early reusability efforts.
- **Ground pad landings** (3 successes) began to appear, showing advancements in precise landing technology.
- Other outcomes such as **Controlled/Uncontrolled ocean landings** and **parachute failures** occurred less frequently.

This period captures the **transition phase** of SpaceX, moving from simple launches to **ambitious reusability goals**.

# Overall Insights Visualization Space X



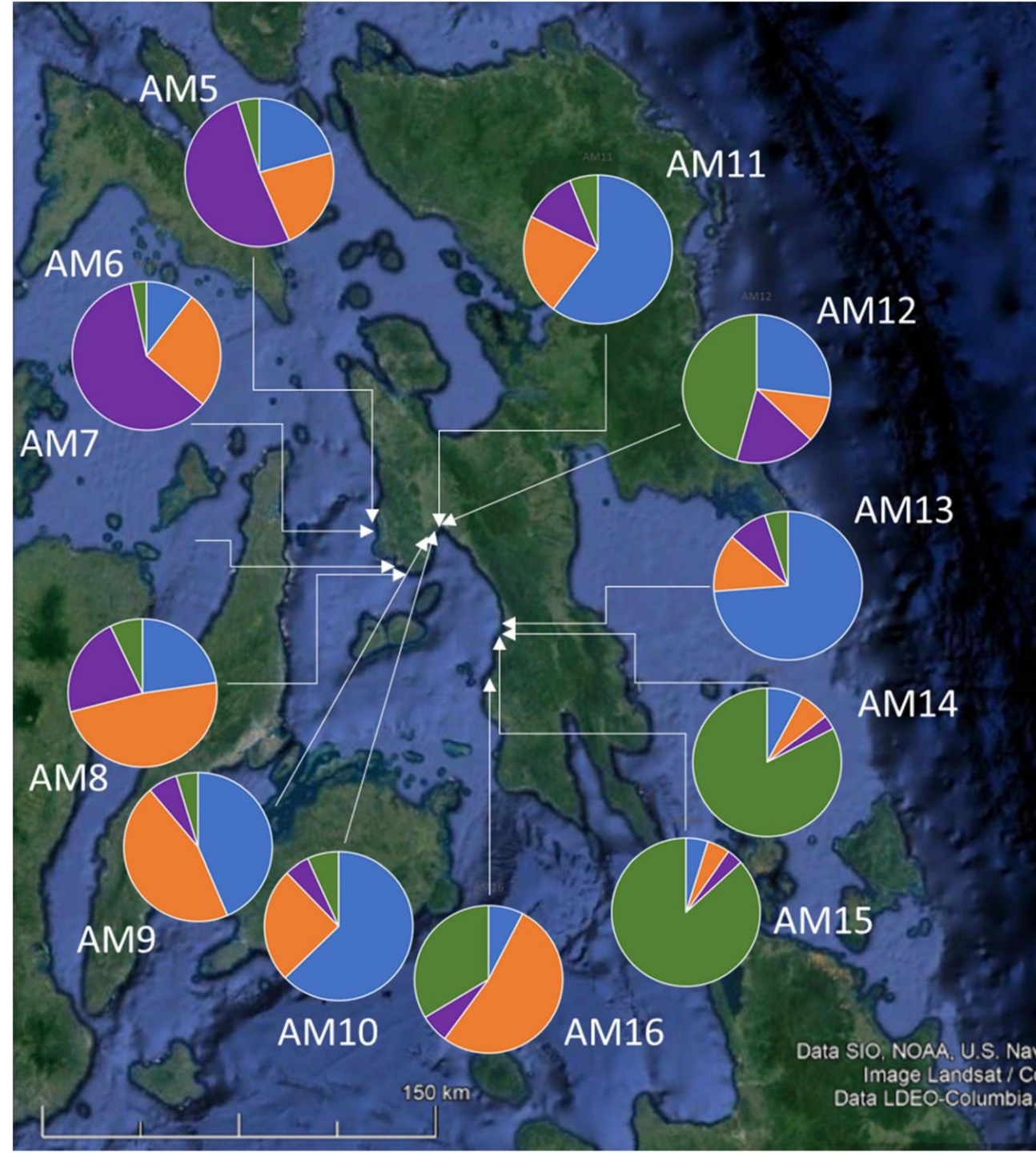
- SpaceX's early years were focused on launch success; recovery was not yet attempted.
- From 2015 onward, reusable booster technology was tested, with mixed results.
- Eventually, F9 B5 became the workhorse for heaviest payloads and successful recoveries.
- Use of SQL allowed precise filtering, aggregation, and insight generation over time.



# Launch Sites Proximity Analysis Space X

---

## Section 4



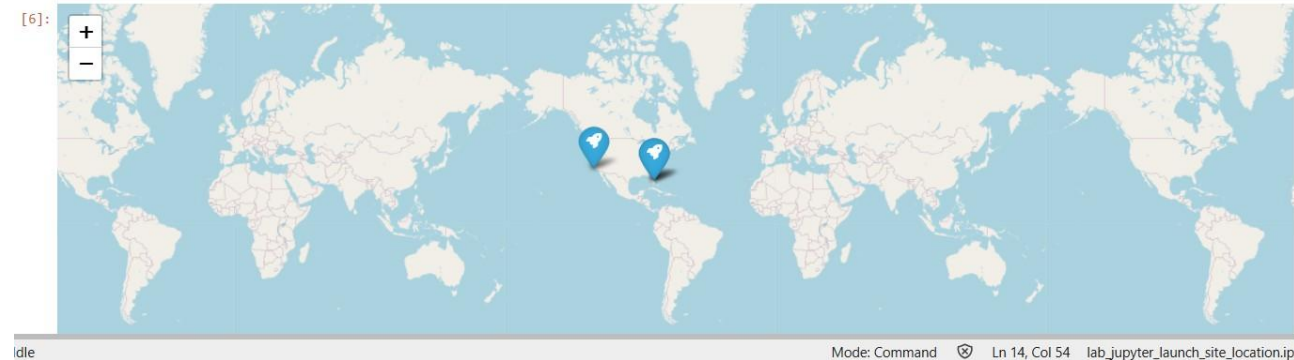


# All launch sites global map markers

In this task, we visualized all unique SpaceX launch sites on an interactive map using the Folium library. By extracting the distinct launch site names along with their geographic coordinates (latitude and longitude) from the dataset, we plotted each site as a marker on a base map. The map is centered around the average location of all launch sites to give a clear overview of where SpaceX conducts its launches. Each marker is interactive, showing the launch site name when clicked, helping to better understand the geographic distribution of SpaceX's launch operations across the United States.



```
[6]: ## Task 1: Mark all launch sites on a map
# Extract unique launch sites with their coordinates
launch_sites_df = df[['LaunchSite', 'Latitude', 'Longitude']].drop_duplicates().reset_index(drop=True)
launch_sites_df
# Calculate mean Latitude and Longitude for centering the map
mean_lat = launch_sites_df['Latitude'].mean()
mean_lon = launch_sites_df['Longitude'].mean()

# Create a base folium map
map_launch_sites = folium.Map(location=[mean_lat, mean_lon], zoom_start=4)
# Add a marker for each launch site
for _, row in launch_sites_df.iterrows():
    folium.Marker(
        location=[row['Latitude'], row['Longitude']],
        popup=row['LaunchSite'],
        icon=folium.Icon(color='blue', icon='rocket', prefix='fa')
    ).add_to(map_launch_sites)
# Display the map
map_launch_sites
```

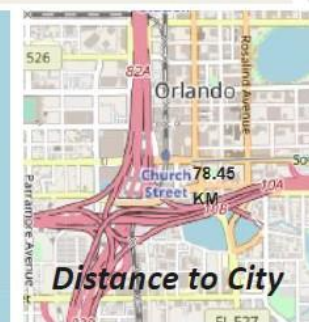
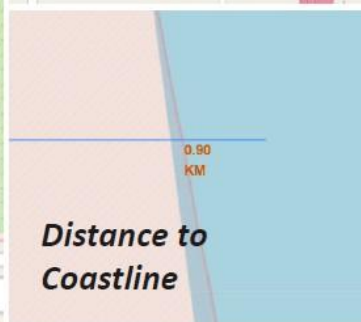


# Markers showing launch sites with color labels



- ✓  All launches mapped, color-coded by outcome.
- ✓  Success patterns visible by site.

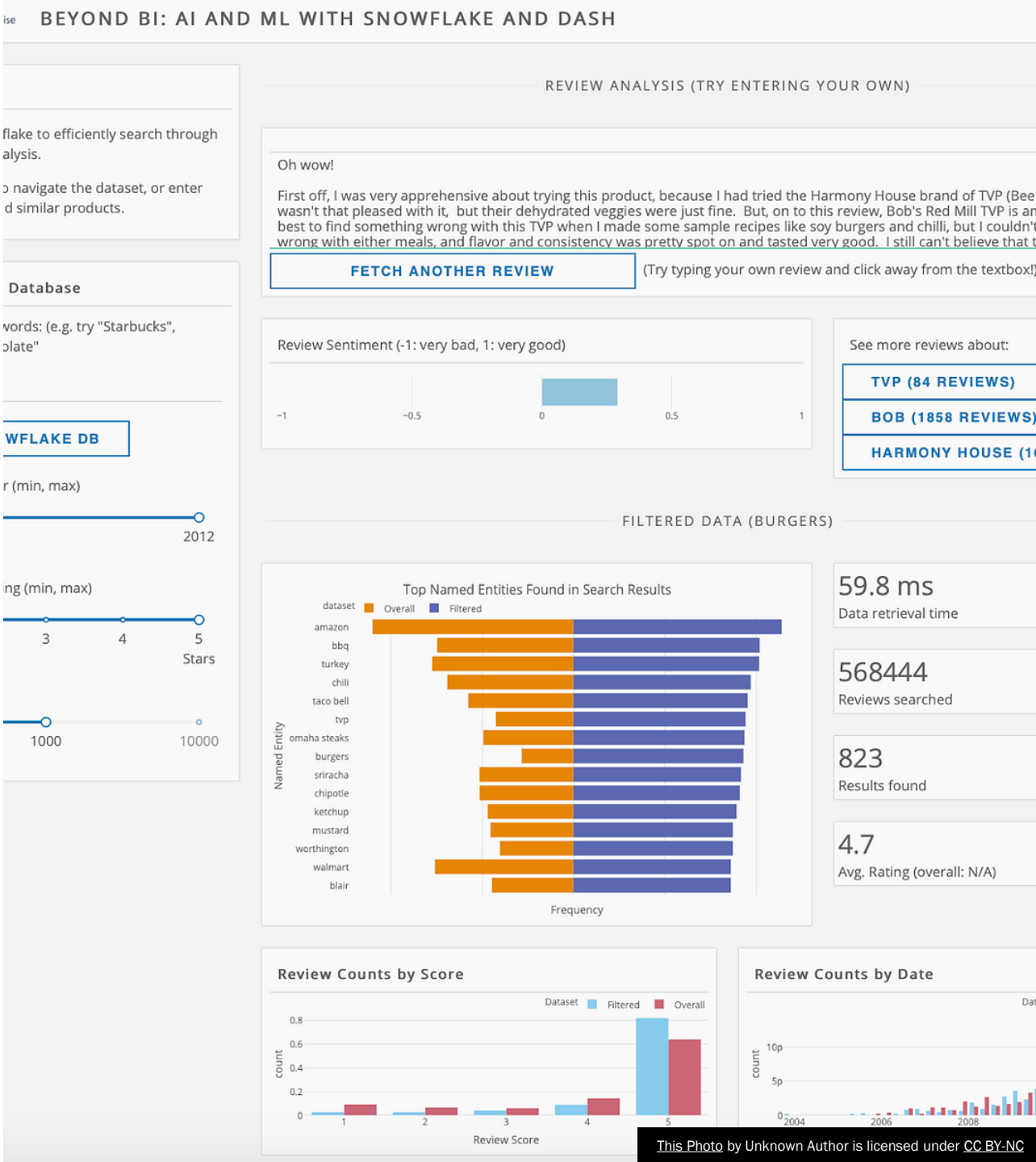
# Launch Site distance to landmarks



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

# Build a Dashboard with Plotly Dash

## SECTION 5

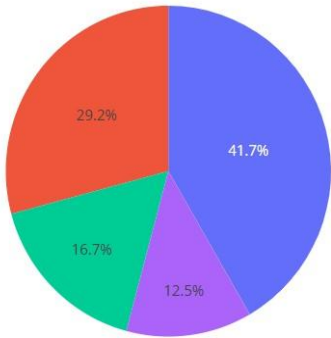


# Pie chart showing the success percentage achieved by each launch site

## SpaceX Launch Records Dashboard

All Sites

Total Successful Launches by Site



- KSC LC-39A
- CCAFS LC-40
- VAFB SLC-4E
- CCAFS SLC-40

Payload range (Kg):

0

Errors

Callbacks

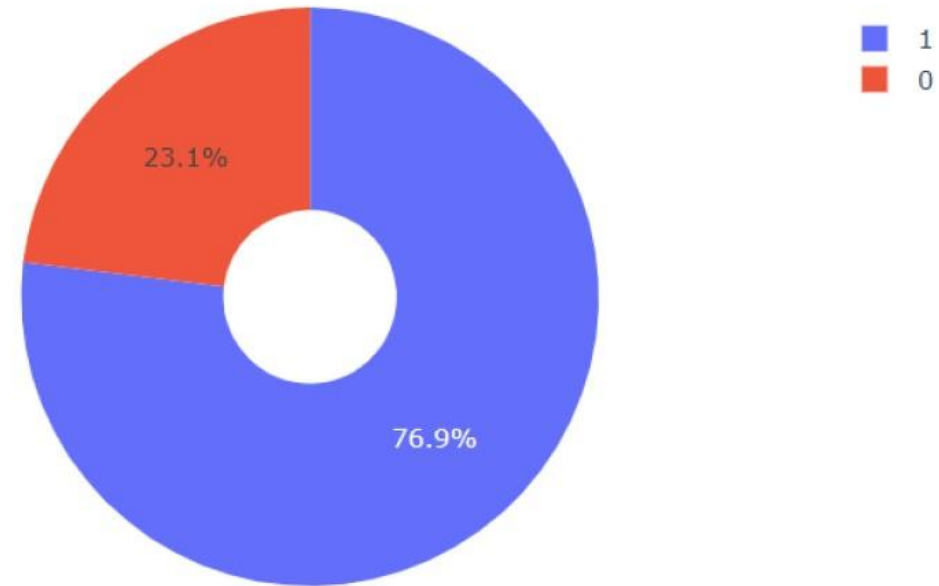
v3.2.0

Server

>>

# Pie chart for the launch site with highest launch success ratio

---



*KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate*

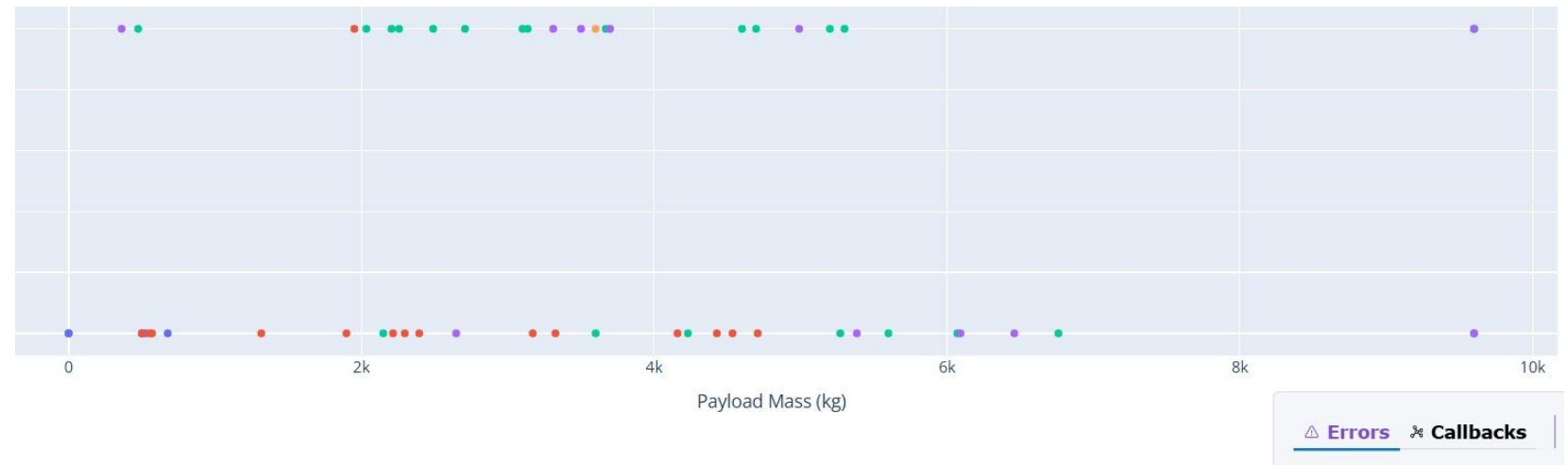


# Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

## 41

kg):

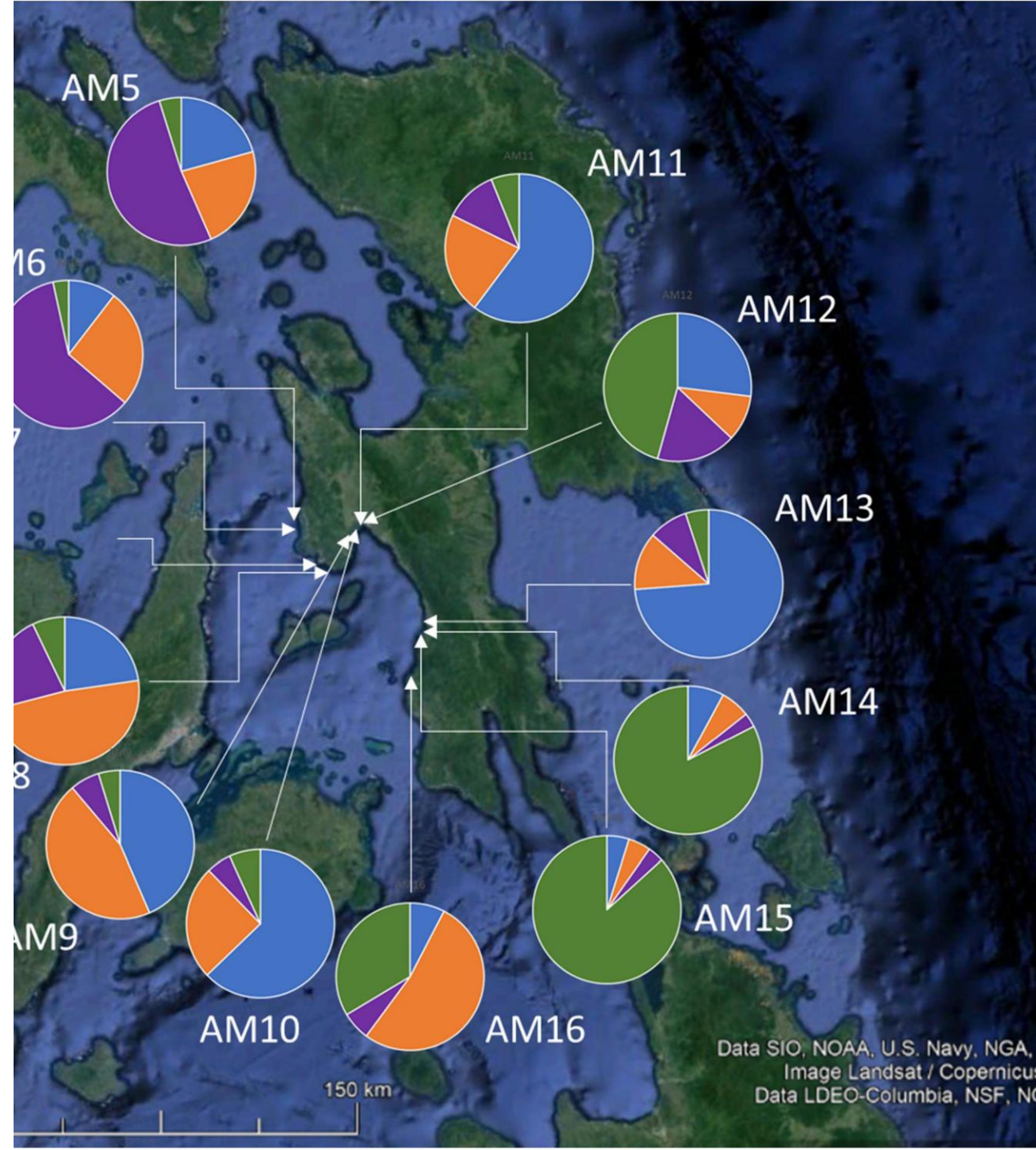
load Mass vs Launch Outcome



# Predictive Analysis (Classification)

---

## Section 6



# Classification Accuracy

Accuracy tells us the proportion of correct predictions made by the model over the total predictions.

In our case, the test accuracy of all models is around 83.33%, which means about 5 out of every 6 predictions are correct.

The validation accuracy varied a bit, with Decision Trees showing the highest (~87.7%), suggesting it might be better at capturing patterns in the training data.

However, the consistent test accuracy across all models indicates the models perform similarly on unseen data.

Since accuracy doesn't tell the full story (e.g., false positives/negatives), it's also important to look at confusion matrices or other metrics.

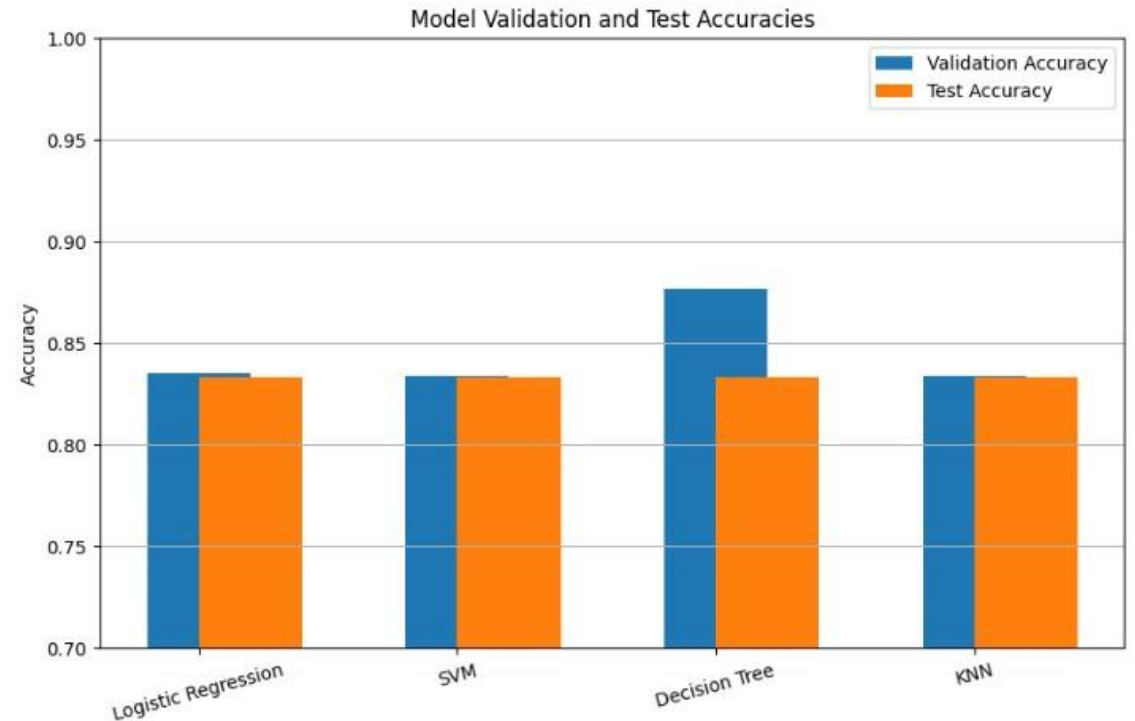
```
[32]: import matplotlib.pyplot as plt

# Define models and their accuracies
models = ['Logistic Regression', 'SVM', 'Decision Tree', 'KNN']
validation_accuracies = [0.8352, 0.8343, 0.8768, 0.8339]
test_accuracies = [0.8333, 0.8333, 0.8333, 0.8333]

x = range(len(models))

plt.figure(figsize=(10,6))
plt.bar(x, validation_accuracies, width=0.4, label='Validation Accuracy', align='center')
plt.bar(x, test_accuracies, width=0.4, label='Test Accuracy', align='edge')

plt.xticks(x, models, rotation=15)
plt.ylim(0.7, 1.0)
plt.ylabel('Accuracy')
plt.title('Model Validation and Test Accuracies')
plt.legend()
plt.grid(axis='y')
plt.show()
```



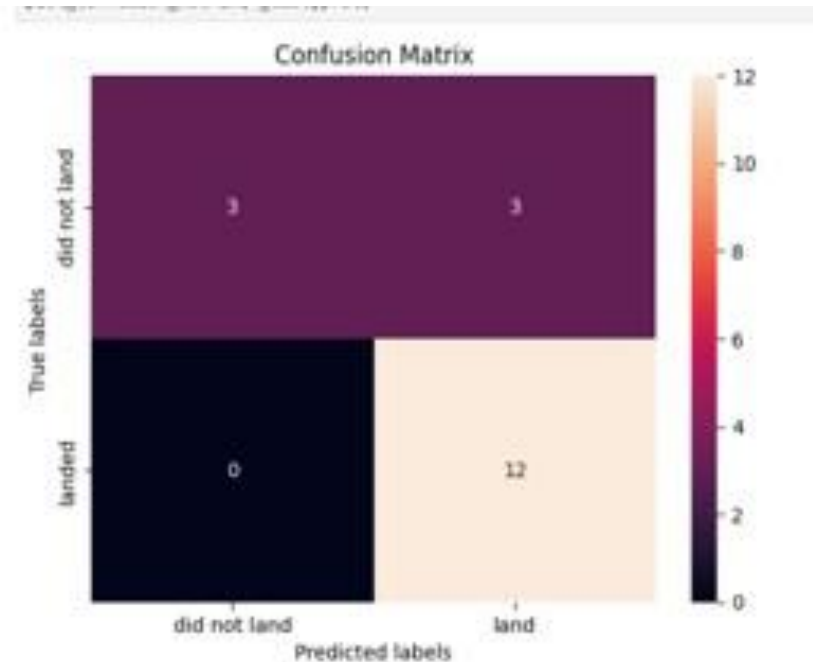
# Confusion Matrix

Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the problem is false positives.

Overview:

True Positive - 12 (True label is landed, Predicted label is also landed)

False Positive - 3 (True label is not landed, Predicted label is landed)



# Table Summary

Model	Best Parameters	Validation Accuracy	Test Accuracy	Key Notes
Logistic Regression	C=1, solver='lbfgs', max_iter=100	83.52%	83.33%	Simple, interpretable model; performs reliably
Support Vector Machine (SVM)	kernel='linear', C=0.0316, gamma=0.001	83.43%	83.33%	Effective with small feature sets; sensitive to parameter tuning
Decision Tree	criterion='gini', max_depth=12, splitter='random', max_features='sqrt'	87.68%	83.33%	Highest validation accuracy; potential overfitting
K-Nearest Neighbors (KNN)	n_neighbors=6, algorithm='auto', p=1	83.39%	83.33%	Simple and instance-based; performs equally well with optimal neighbors

# Conclusions

---

- In Conclusion:
- The success rate at a launch site tends to improve as the number of flights increases.
- Launch success rates showed a steady rise from 2013 through 2020.
- Orbits such as ES-L1, GEO, HEO, SSO, and VLEO demonstrated the highest success rates.
- KSC LC-39A recorded the highest number of successful launches among all sites.
- The Decision Tree classifier proved to be the most effective machine learning algorithm for this prediction task.



**Thank you**

