



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Kennedy KR Sakonda  
15 May 2025



# Outline

---

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

# Executive Summary

---

## **Summary of methodologies**

This project follows these steps:

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis (Classification)

## **Summary of results**

This project produced the following outputs and visualisation:

- Exploratory Data Analysis (EDA) results
- Geospatial Analytics
- Interactive dashboard
- Predictive Analysis of Classification Models

# Introduction

---

## Project background and context

SpaceX launches Falcon 9 rockets at a cost of around \$62m. This is considerably cheaper than other providers who are charging 165m with much of the savings resulting from the fact that SpaceX can land and re-use the first stage of the rocket.

## Problems you want to find answers to

Once we are able to predict whether the first stage will land, we can determine the cost of a launch and also be able to use this information to assess whether an alternative company should bid and SpaceX for a rocket launch





Section 1

# Methodology

# Methodology

---

## Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

# Methodology

## Executive Summary

### 1. Data collection methodology.

- Making a GET requests to the SpaceX REST API

### 2. Data Wrangling

- Also known as data cleaning or processing.
- Using the fillna() method to remove NaN vaues
- Using the value\_count() method to determine the following:
  - Number of launchies on each site
  - Number and occurrences of mission outcome per orbit type
  - Number of occurrences on each orbit

Creating a landing outcome lable that shows the following

- 0 when the booster did not land successfully
- 1 When the booster landed successfully

### 3. Exploratory data analysis

- Using the SQL queries to manipulate and evaluate the SpaceX dataset
- Using Pandas and Matplotlib to visualise the relationship between variables and determine patterns.

### 4. Interactive visual analytics

- Creating an interactive dashboard using;
- Geospatial analytics using folium.

### 5. Data modelling and evaluation

- Using Scikit-learn to:
  - Preprocess(standardise ) the data;
  - Split data into training and testing using train\_test\_split
  - Train different classification models
  - Find hyperparameters using GRIDSEARCHCV
  - Plotting confusion matrices for each classification model
  - Assessing the accuracy of each classification model

# Data Collection

- **Key Steps in Data Collection:**

- 1. Identify Data Sources**

- Official SpaceX website and launch manifests
- SpaceX API endpoints (if available)
- Public databases and repositories (e.g., Kaggle datasets)
- News articles and press releases for supplementary information

- 2. Automated Data Extraction**

- Use web scraping tools (e.g., BeautifulSoup, Scrapy)
- Fetch data from SpaceX's public API or website
- Extract relevant fields: launch date, payload mass, launch site, booster version, launch outcome

- 3. Manual Data Verification**

- Cross-reference scraped data with official sources
- Correct discrepancies or missing values
- Supplement data with additional details from press releases

- 4. Data Cleaning & Preprocessing**

- Remove duplicates
- Handle missing values
  - Convert data types (e.g., payload mass to numeric)
- Encode categorical variables (e.g., booster version, launch site)

- 5. Data Storage**

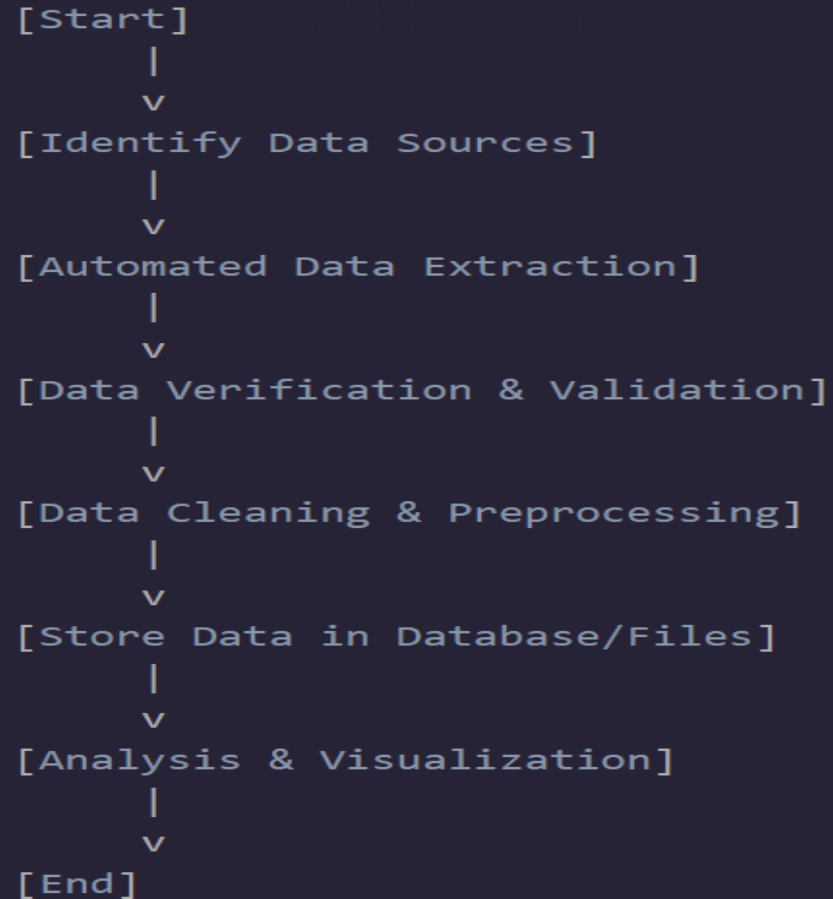
- Store cleaned data in structured formats (CSV, SQL database)

- Ensure data integrity and version control



# Data Collection – (continued)

## Flowchart of Data Collection Process:



# Data Collection – SpaceX API

To gather detailed and up-to-date data on SpaceX launches, rockets, and missions, we utilised the SpaceX REST API. The process involved making specific API calls to retrieve relevant datasets, then processing and storing the data for analysis.

1. Make a GET response to the SpaceX REST API
  - Convert the response to a JSON file, then to a Pandas DataFrame
2. Use custom logic to clean the data
  - Define lists for data to be stored in.
  - Call custom functions to retrieve data and fill the lists.
  - Use these lists as values in a dictionary and construct a dataset.
3. Create a Pandas DataFrame from the constructed dictionary Dataset.
4. Filter the DataFrame to only include Falcon 9 Launches
  - Reset the FlightNumber Column
  - Replace the missing values of PayloadMass with the mean PayloadMass values.

```
1
spaces_url = "https://api.spacexdata.com/v4/launches/past"

response = requests.get(spaces_url)

# Use json_normalize method to convert the json result into a dataframe
data = pd.json_normalize(response.json())

2
# Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
Gridfins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []

# Call getBoosterVersion
getBoosterVersion(data)

# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)

launch_dict = {'FlightNumber': list(data['flight_number']),
               'Date': list(data['date']),
               'BoosterVersion': BoosterVersion,
               'PayloadMass': PayloadMass,
               'Orbit': Orbit,
               'LaunchSite': LaunchSite,
               'Outcome': Outcome,
               'Flights': Flights,
               'Gridfins': Gridfins,
               'Reused': Reused,
               'Legs': Legs,
               'LandingPad': LandingPad,
               'Block': Block,
               'ReusedCount': ReusedCount,
               'Serial': Serial,
               'Longitude': Longitude,
               'Latitude': Latitude}

3
# Create a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)

4
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']

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

# Calculate the mean value of PayloadMass column and replace the nan values with its mean value
data_falcon9 = data_falcon9.fillna({'PayloadMass': data_falcon9['PayloadMass'].mean()})
```

# Data Collection - Scraping

Webscraping to collect Falcon 9 historical launch data

1. Request the HTML from the static URL
  - Assign the response to an object
2. Create a BeautifulSoup object from the HTML response object
  - Find all tables within the HTML page
3. Collect all column header names from the tables found within the HTML page
4. Use the column names as keys in a dictionary
  - Use custom functions and logic to parse all launch tables to fill the dictionary values
5. Convert the dictionary to a Pandas DataFrame ready for export

```
1 static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1017109922"
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
data = response.text

2 soup = BeautifulSoup(data, 'html5lib')
first_table = soup.find_all('table')

3 column_names = []
# Iterate find_all() function with "th" element in first launch table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the new empty column name ("") name to list those and transform it to a list called column_names

for row in first_launch_table.find_all('tr'):
    name = extract_column_from_header(row)
    if name is None and len(name) > 0:
        column_names.append(name)

4 launch_dict = dict.fromkeys(column_names)

# Remove an unwanted column
del launch_dict["Data and time (")

# Let's verified the launch_dict with each value the list an empty list
launch_dict["Flight No."] = []
launch_dict["Launch site"] = []
launch_dict["Payload"] = []
launch_dict["Payload mass"] = []
launch_dict["Orbit"] = []
launch_dict["Customer"] = []
launch_dict["Launch outcome"] = []

# adding some new columns
launch_dict["Previous Booster"] = []
launch_dict["Booster Landing"] = []
launch_dict["Date"] = []
launch_dict["Time"] = []

5 df = pd.DataFrame(launch_dict)
```

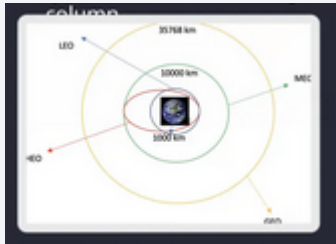
[GitHub Link](#)

# Data Wrangling

## Context

- The SpaceX dataset includes several SpaceX facilities, and each location is in the **LaunchSite** column
- Each launch aims for a dedicated orbit, and some of the common orbit types

are shown in the figure below. The **orbit** type is in the orbit column



## Initial Data exploration

Using the **Value\_Counts()** to determine the following

1. Number of launches on each site
2. Number and occurrence on each site
3. Number and occurrence of landing outcomes per orbit type

```
1 # Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()

CCAFS SLC 40    55
KSC LC PPA     22
WFB SLC 4E     13
Name: LaunchSite, dtype: int64

2 # Apply value_counts on Orbit column
df['Orbit'].value_counts()

GTO    27
ISS    21
VLEO   14
PO      9
LEO      7
SSO      5
MEO      3
ES-L1    1
GEO      1
SO       1
HEO      1
Name: Orbit, dtype: int64

3 # landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes

True ASDS    41
None None     19
True RTLS    14
False ASDS     6
True Ocean     5
None ASDS      2
False Ocean     2
False RTLS      1
Name: Outcome, dtype: int64
```

# EDA with Data Visualization

---

I visualized the data to identify patterns, trends, and relationships. The chosen charts helped us understand the distribution, correlations, and key insights essential for informed decision – making. Summarize what charts were plotted and why you used those charts

- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose



# EDA with Data Visualization

I visualised the data to identify patterns, trends, and relationships. The chosen charts helped us understand the distribution, correlations, and key insights essential for informed decision-making.

A scatter diagram was produced to visualise the relationship between:

- Fight number and launch site
- Payload and launch site
- Orbit type and flight number
- Payload and orbit type

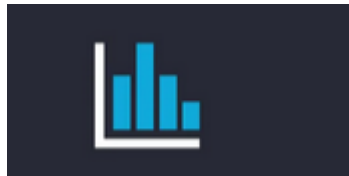


This type of chart is useful to observe the relationships or correlations between two numerical values.

Bar chart

- This was produced to visualise the relationship between

- Success rate and Orbit Type
- Bar charts are used to compare numerical
- Values to a categorical variable
- Horizontal or vertical bar charts can be used depending on the size of the data



Line Charts

These charts were produced to visualise the relationship between:

Success rate and year, i.e. the launch success yearly trend.

Line charts contain numerical values on both axes and are generally used to show the change of a variable over time



# EDA with SQL

---

To gather information about the dataset. Some SQL queries were performed.

- **Summary of SQL Queries Performed**
- **SELECT with COUNT()**
  - Counted total launches, successful launches, and failures to understand overall performance.
- **GROUP BY and ORDER BY**
  - Analyzed launch counts per year, launch site, and rocket type to identify top-performing categories.
- **JOIN**
  - Merged data from multiple tables (e.g., launches, rockets, launch sites) to create comprehensive datasets for analysis.
- **WHERE**
  - Filtered data to focus on specific timeframes, success status, or particular launch sites.
- **AGGREGATION functions (AVG, MAX, MIN)**
  - Calculated average payload mass, maximum/minimum payloads, and success rates across different categories.
- **DISTINCT**
  - Identified unique values for categorical variables such as rocket names, launch sites, and mission types.

# EDA with SQL- Continued

---

To gather information about the dataset. Some SQL queries were performed.

- **DISTINCT**
  - Identified unique values for categorical variables such as rocket names, launch sites, and mission types.
- **CASE WHEN** (Conditional Statements)
  - Created new categorical variables, like success/failure flags or launch outcome labels.
- **SUBQUERIES**
  - Used nested queries to analyze subsets of data, such as recent launches or specific mission types.
- **CREATE VIEW**
  - Saved complex query results as views for easier access and repeated analysis.

# Build an Interactive Map with Folium

---

- **Map Objects Created and Added**
- **Markers**
  - Placed on launch sites and notable locations to indicate their geographic positions.
  - Included popups with site names and additional information for easy identification.
- **Circles**
  - Used around launch sites to represent the launch site coverage area or frequency.
  - Differentiated by colour or radius based on launch success rates or other metrics.
- **Lines (Polylines)**
  - Connected launch sites to corresponding landing zones or planned trajectories.
  - Visualised paths of rockets or planned routes for clarity.
- **Choropleth Layers (if applicable)**
  - Displayed regional data such as launch density or success rates across different areas.

# Build an Interactive Map with Folium-continued

---

- **Why These Objects Were Added**
- To **visualize spatial relationships** and geographic distribution of launch sites and missions.
- To **highlight key locations** and their significance in the launch network.
- To **illustrate trajectories or connections** between launch points and targets.
- To **enhance interactivity** and provide contextual information through popups and visual cues.
- To **support analysis** of geographic patterns related to launch success or failure..



# Build a Dashboard with Plotly Dash

---

1. Added a Launch Site Drop-down Input to the dashboard to provide the ability [to filter Dashboard visual by all launch sites or a particular launch site](#)
2. Added a Pie Chart to the Dashboard to show total success launches when 'All sites ' is selected, [and show success and failed counts when a particular site is selected](#)
3. Added a Payload range slider to the Dashboard to easily select different Payload ranges [to identify usual patterns](#)
4. Added a Scatter chart [to observe how payload may be correlated with mission outcomes for selected site\(s\)](#). The colour-label Booster version on each scatter point provided mission outcomes with different boosters.

# Predictive Analysis (Classification)

---

- **Model Development Workflow**

- **1. Data Preparation**

- Collected and cleaned the dataset, handling missing values and outliers.
- Encoded categorical variables and normalized numerical features.

- **2. Feature Selection**

- Used correlation analysis and feature importance scores to identify relevant features.
- Reduced dimensionality for improved model efficiency.

- **3. Model Building**

- Tried multiple algorithms: Logistic Regression, Random Forest, Support Vector Machine (SVM), and Gradient Boosting.
- Split data into training and testing sets to evaluate performance.

# Predictive Analysis (Classification)

---

- **4. Model Evaluation**

- Used metrics like accuracy, precision, recall, F1-score, and ROC-AUC.
- Employed cross-validation to ensure model robustness.

- **5. Model Tuning & Improvement**

- Performed hyperparameter tuning using Grid Search and Random Search.
- Selected the best model based on validation metrics.

- **6. Final Model Selection**

- Identified the Random Forest classifier as the best performing model with high accuracy and stability.
- Validated on unseen test data to confirm performance.

# Predictive Analysis (Classification)

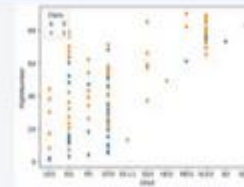
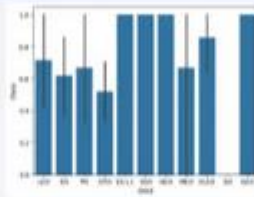
## Flowchart Outline of the Model Development Process

```
[Start]
|
v
[Data Collection & Exploration]
|
v
[Data Preprocessing]
|
v
[Feature Engineering & Selection]
|
v
[Model Selection & Training]
|
v
[Model Evaluation]
|
v
[Hyperparameter Tuning & Optimization]
|
v
[Model Comparison & Final Selection]
|
v
[Model Validation on Test Data]
|
v
[Deployment & Monitoring (if applicable)]
|
v
[End]
```

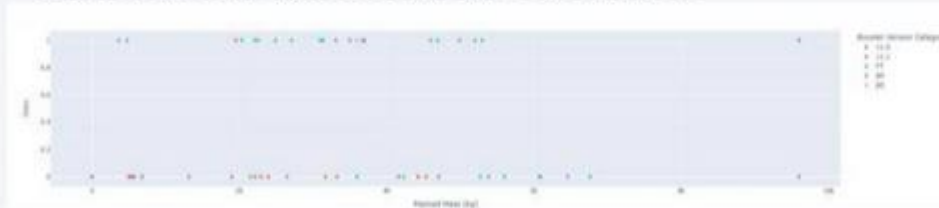
# Results

## Results

- Exploratory data analysis results



- Interactive analytics demo in screenshots



- Predictive analysis results

- Accuracy for Logistics Regression method: 0.8333333333333334
- Accuracy for Support Vector Machine method: 0.8333333333333334
- Accuracy for Decision tree method: 0.8888888888888888
- Accuracy for K nearest neighbors method: 0.8333333333333334



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



# Flight Number vs. Launch Site

---

- **Scatter Plot: Flight Number vs. Launch Site**
- **Overview:**
  - The scatter plot visualizes the relationship between
  - the **Flight Number** (X-axis) and the **Launch Site** (Y-axis).
  - Each point represents a launch, with its position
  - indicating the sequence of the flight and the site where
  - it was launched

# Flight Number vs. Launch Site – (continued)

---

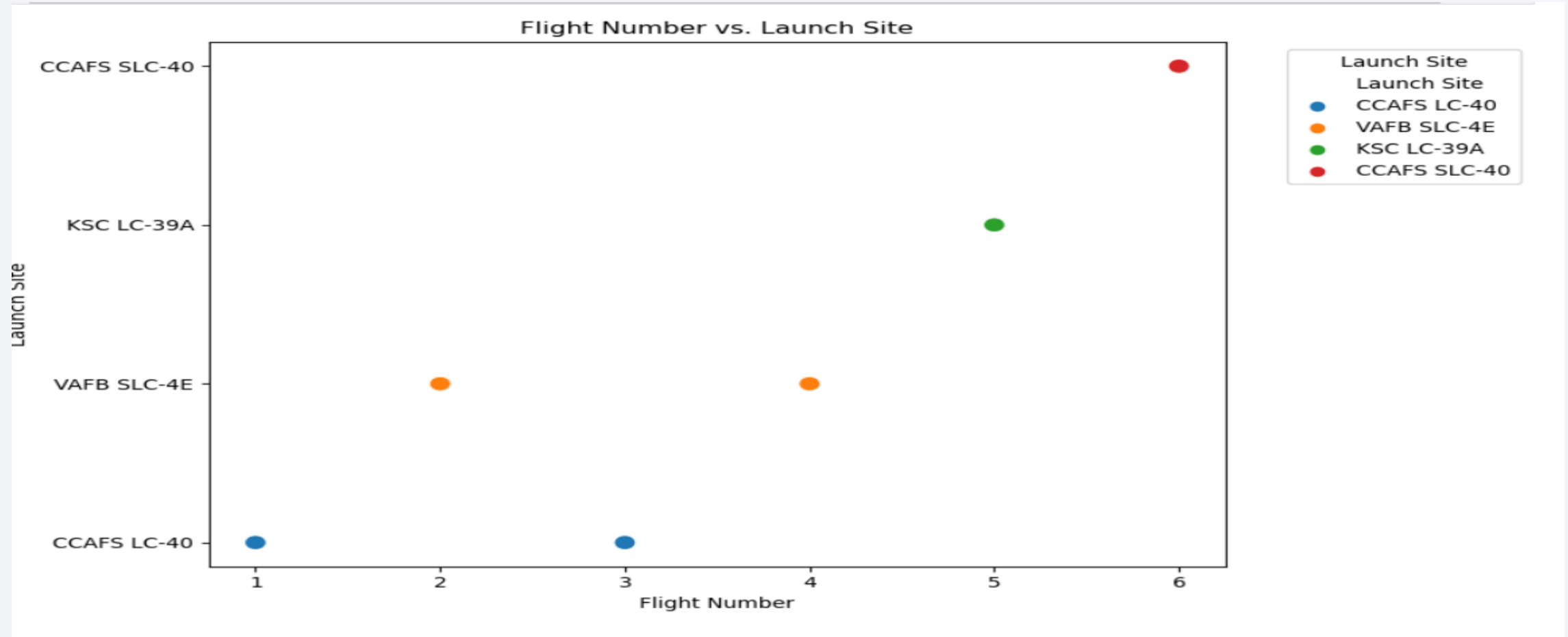
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Example: Load your dataset
# df = pd.read_csv('spacex_launches.csv')

# For demonstration, creating a mock dataset
data = {
    'Flight Number': [1, 2, 3, 4, 5, 6],
    'Launch Site': ['CCAFS LC-40', 'VAFB SLC-4E', 'CCAFS LC-40', 'VAFB SLC-4E', 'KSC LC-39A', 'CCAFS SLC-40']
}
df = pd.DataFrame(data)

# Plotting
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Flight Number', y='Launch Site', data=df, hue='Launch Site', palette='tab10', s=100)
plt.title('Flight Number vs. Launch Site')
plt.xlabel('Flight Number')
plt.ylabel('Launch Site')
plt.legend(title='Launch Site', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

# Flight Number vs. Launch Site – (continued)



# Payload vs. Launch Site

- **Step-by-step Guide to Create the Scatter Plot**

- **1. Data Preparation:**

- Ensure your dataset has at least two relevant columns:
  - Payload Mass (kg) — Numeric value representing the payload weight.
  - Launch Site — Categorical variable indicating the launch site.

- **2. Data Processing:**

- No need for encoding the launch site if plotting with categorical labels.
- Check for missing or outlier payload values and clean if necessary.

- **3. Plotting:**

- Use a scatter plot to visualize the relationship between Payload Mass and Launch Site.
- Different launch sites can be distinguished using different colors or markers.

Show a scatter plot of Payload vs. Launch Site

- Show the screenshot of the scatter plot with explanations

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

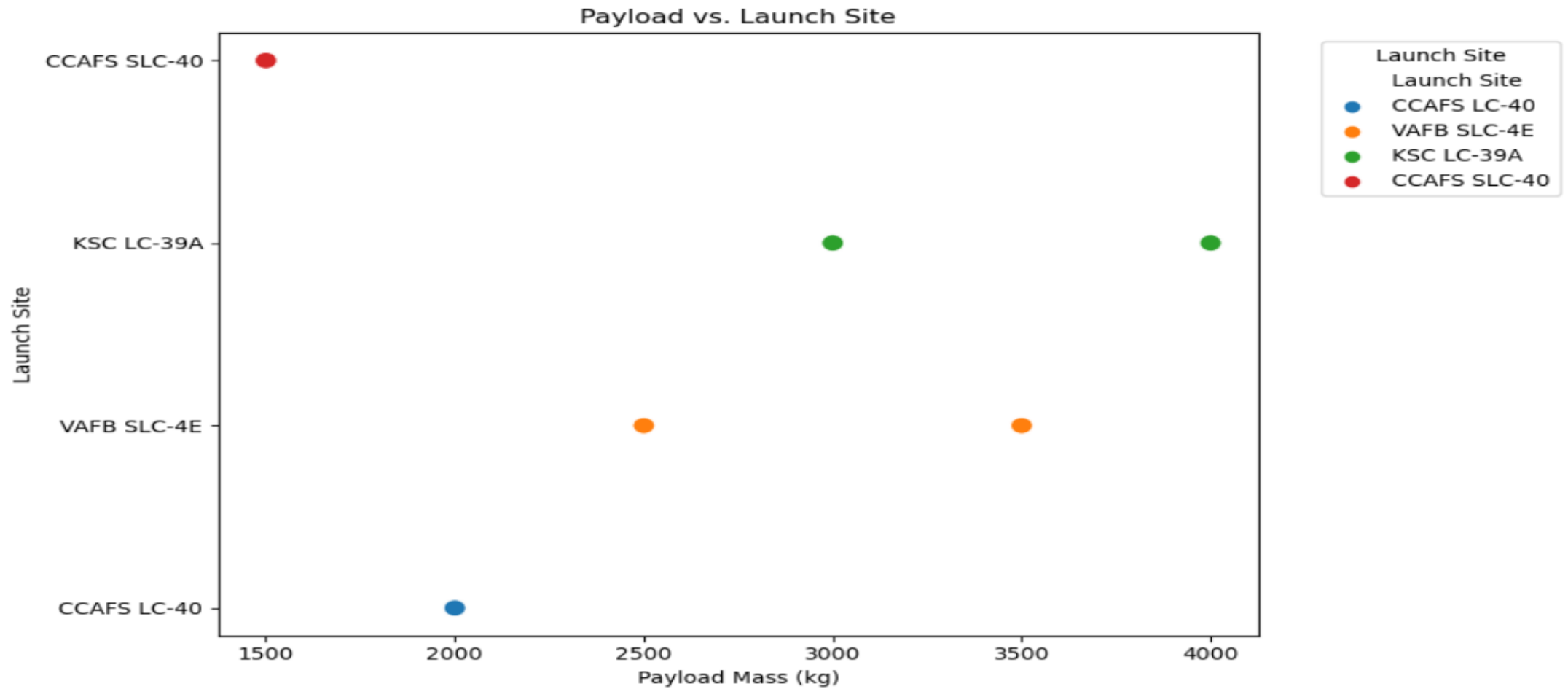
# Example: Load your dataset
# df = pd.read_csv('spacex_launches.csv')

# For demonstration, creating a mock dataset
data = {
    'Payload Mass (kg)': [2000, 2500, 3000, 1500, 3500, 4000],
    'Launch Site': ['CCAFS LC-40', 'VAFB SLC-4E', 'KSC LC-39A', 'CCAFS SLC-40', 'VAFB SLC-4E', 'KSC LC-39A']
}
df = pd.DataFrame(data)

# Plotting
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Payload Mass (kg)', y='Launch Site', data=df, hue='Launch Site',
                palette='tab10', s=100)
plt.title('Payload vs. Launch Site')
plt.xlabel('Payload Mass (kg)')
plt.ylabel('Launch Site')
plt.legend(title='Launch Site', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```



# Payload vs. Launch Site –(continued)



the list has now been update

# Success Rate vs. Orbit Type

- **Step-by-Step Guide to Create the Success Rate Bar Chart**
- **1. Data Preparation:**
- Ensure your dataset includes:
  - Orbit Type — categorical variable (e.g., GTO, LEO, MEO, etc.)
  - Mission Status — success or failure indicator (e.g., "Success", "Failure")
- **2. Data Processing:**
- Calculate the success rate for each orbit type:
  - Group data by Orbit Type.
  - For each group, compute the number of successful launches divided by total launches.
- **3. Plotting:**
- Use a bar chart to visualise success rates per orbit type, with success rate as a percentage or decimal.

```
import pandas as pd
import matplotlib.pyplot as plt

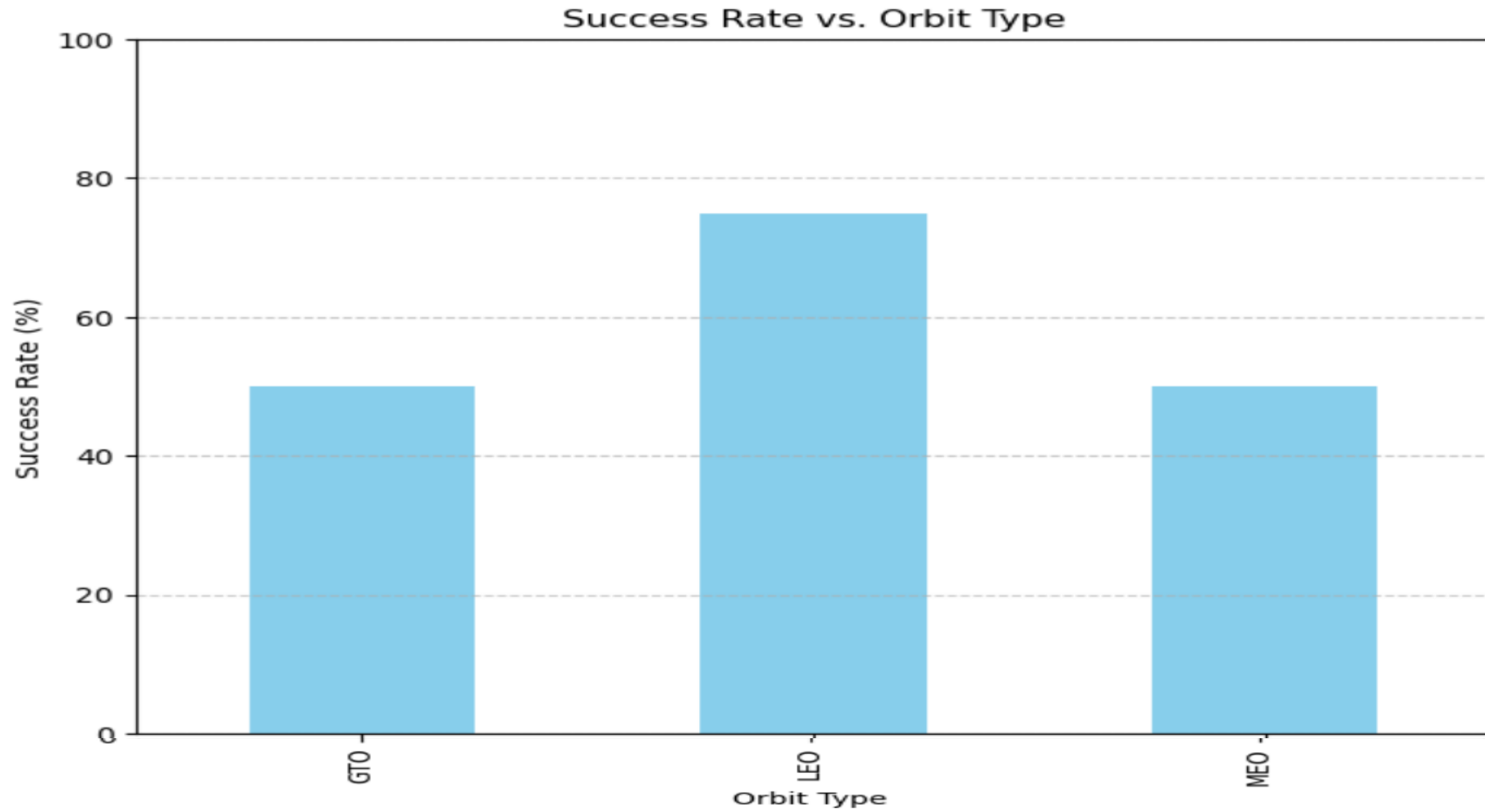
# Example: Load your dataset
# df = pd.read_csv('spacex_launches.csv')

# Mock dataset for demonstration
data = {
    'Orbit Type': ['LEO', 'GTO', 'LEO', 'MEO', 'GTO', 'LEO', 'GTO', 'LEO', 'MEO', 'GTO'],
    'Mission Status': ['Success', 'Failure', 'Success', 'Success', 'Failure', 'Success', 'Success', 'Failure', 'Success', 'Failure']
}
df = pd.DataFrame(data)

# Calculate success rate per orbit type
success_counts = df.groupby('Orbit Type')['Mission Status'].apply(lambda x: (x == 'Success').sum())
total_counts = df.groupby('Orbit Type')['Mission Status'].count()
success_rate = (success_counts / total_counts) * 100 # in percentage

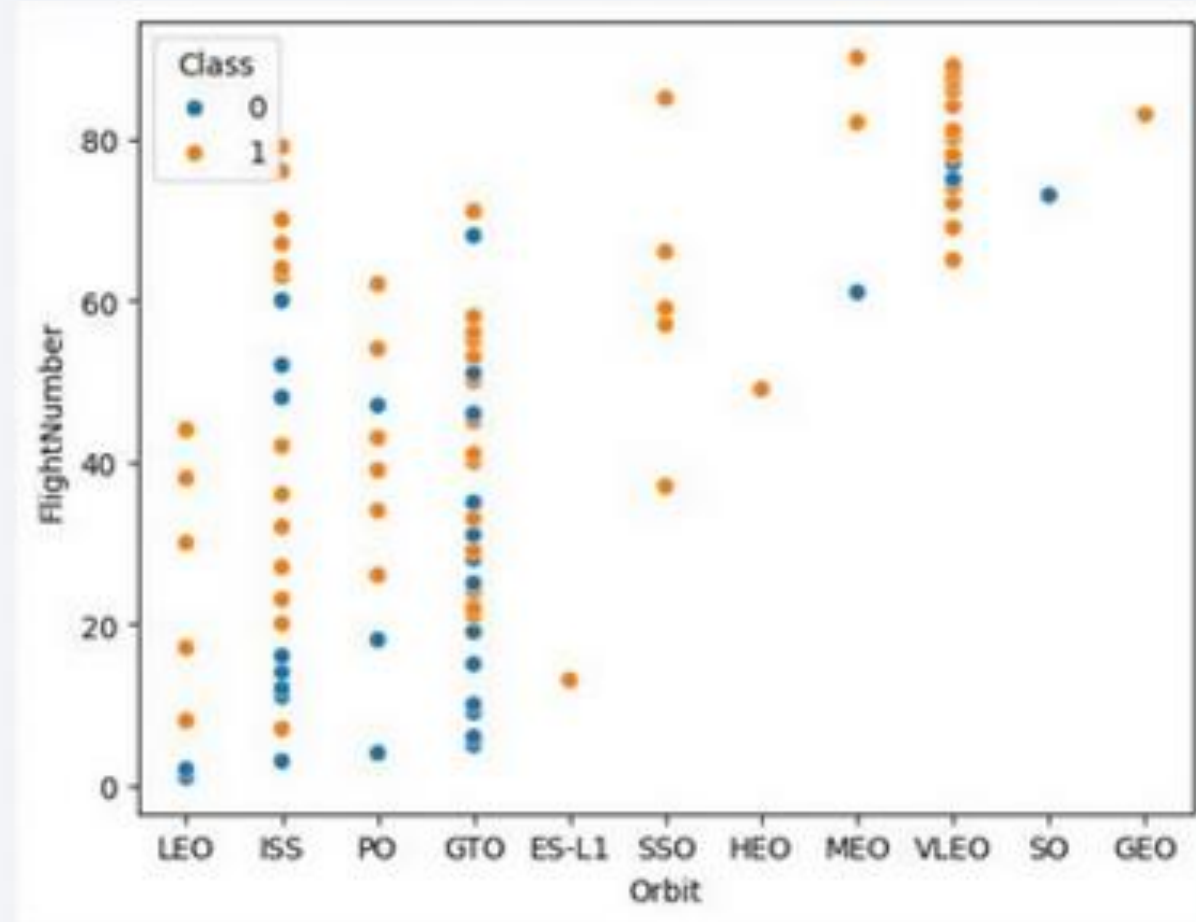
# Plot
plt.figure(figsize=(8, 6))
success_rate.plot(kind='bar', color='skyblue')
plt.title('Success Rate vs. Orbit Type')
plt.xlabel('Orbit Type')
plt.ylabel('Success Rate (%)')
plt.ylim(0, 100)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

## Success Rate vs. Orbit Type- (continued)



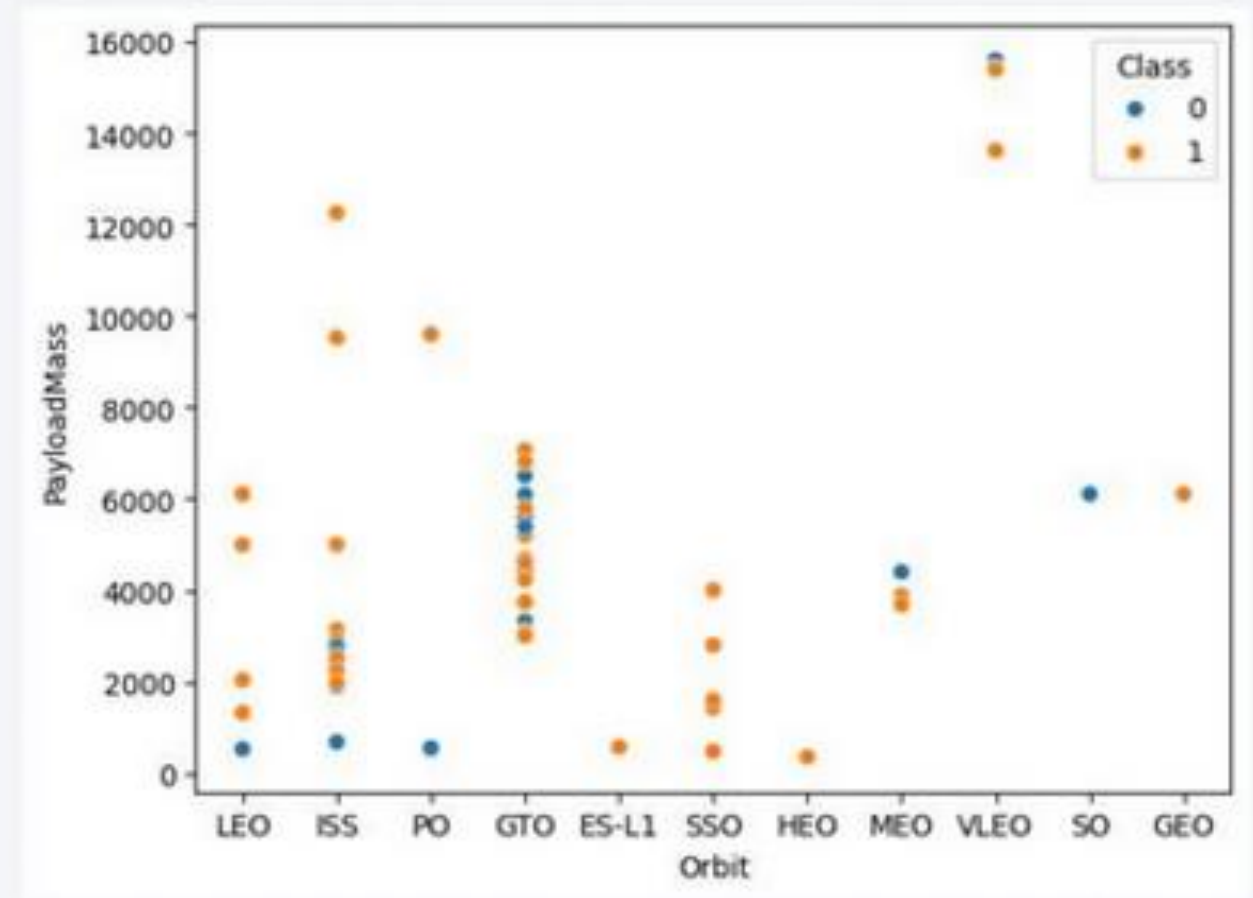
# Flight Number vs. Orbit Type

- For orbit VLEO, first successful landing (class=1) doesn't occur until 60+ number of flights
- For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers
- There is no relationship between flight number and orbit for GTO



# Payload vs. Orbit Type

- Successful landing rates (Class=1) appear to increase with pay load for orbits LEO, ISS, PO, and SSO
- For GEO orbit, there is not clear pattern between payload and successful launches



# Launch Success Yearly Trend

- **Step 1: Prepare Your Data**

- You need a dataset with at least these columns:

- Launch Year: the year
- r of each launch
- Success: a binary indicator (e.g., 1 for success, 0 for failure)

- **Step 2: Calculate Yearly Success Rate**

- For each year, compute:  
$$\text{Success Rate} = \frac{\text{Number of Successful Launches}}{\text{Total Launches}} \times 100$$
- $$\text{Success Rate} = \frac{\text{Number of Successful Launches}}{\text{Total Launches}} \times 100$$

- **Step 3: Plot the Data**

- Plot a line chart with:
- X-axis: Year
- Y-axis: Success rate (percentage or proportion)

```
import pandas as pd
import matplotlib.pyplot as plt

# Example: Load your dataset
# df = pd.read_csv('spacex_launches.csv')

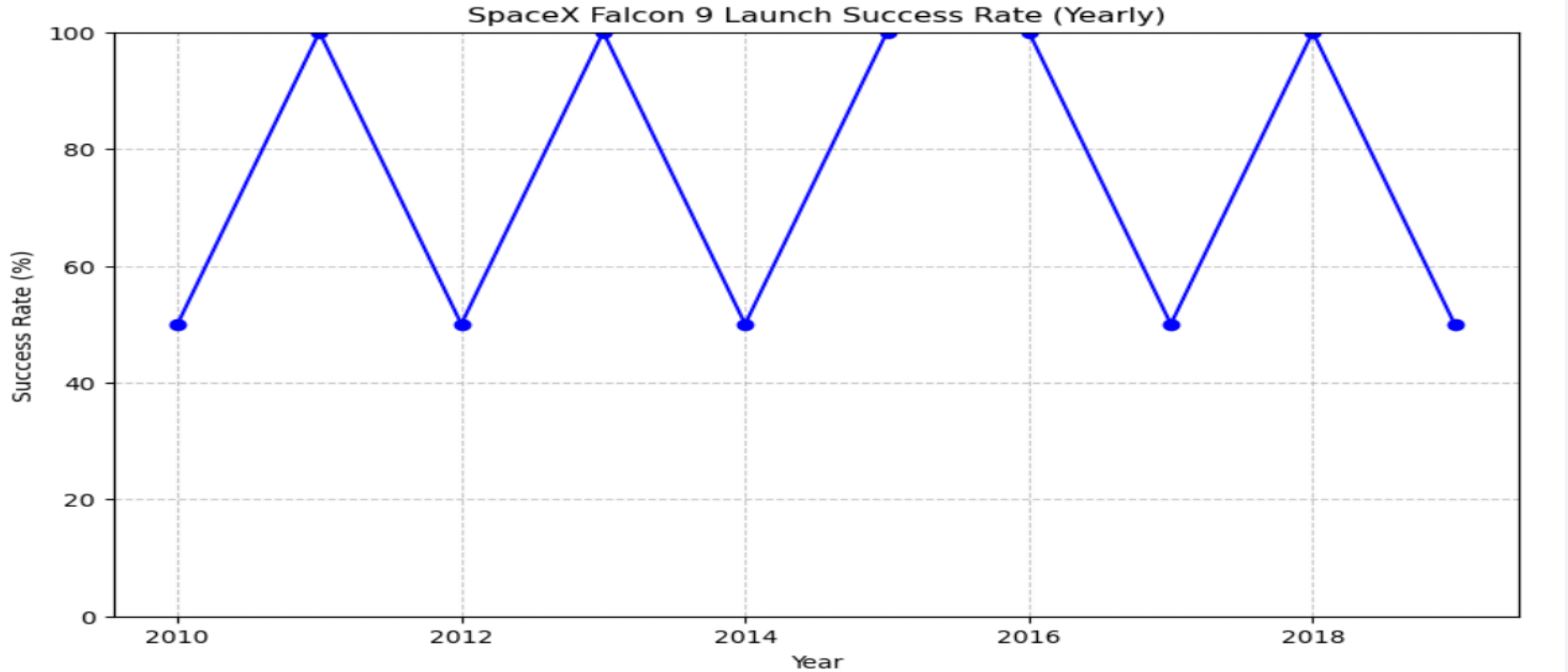
# Mock dataset for demonstration
data = {
    'Launch Year': [2010, 2010, 2011, 2011, 2012, 2012, 2013, 2013, 2014, 2014, 2015, 2015, 2016, 2016, 2017, 2017, 2018, 2018, 2019, 2019],
    'Success': [1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]
}
df = pd.DataFrame(data)

# Calculate yearly success rate
yearly_stats = df.groupby('Launch Year')['Success'].agg(['mean', 'count'])
yearly_stats['Success Rate (%)'] = yearly_stats['mean'] * 100

# Plot
plt.figure(figsize=(10, 6))
plt.plot(yearly_stats.index, yearly_stats['Success Rate (%)'], marker='o', color='b')
plt.title('SpaceX Falcon 9 Launch Success Rate (Yearly)')
plt.xlabel('Year')
plt.ylabel('Success Rate (%)')
plt.ylim(0, 100)
plt.grid(True, linestyle='--', alpha=0.7)

# Show the plot
plt.show()
```

# Launch Success Yearly Trend- (continued)



# All Launch Site Names

## Step-by-step process:

1. Load your dataset.
2. Extract the column with launch site names.
3. Use the unique() function to find all distinct site names.
4. Present the result with a brief explanation.

### Unique SpaceX Falcon 9 Launch Sites:

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

```
import pandas as pd

# Assuming your dataset is loaded into a DataFrame named df
# For demonstration, here's a mock dataset:
data = {
    'Launch Site': [
        'CAFS SLC-40', 'VAFB SLC-4E', 'KSC LC-39A', 'CAFS SLC-40',
        'VAFB SLC-4E', 'KSC LC-39A', 'VAFB SLC-4E'
    ],
    'Mission Name': ['Mission 1', 'Mission 2', 'Mission 3', 'Mission 4', 'Mission 5', 'Mission 6', 'Mission 7']
}
df = pd.DataFrame(data)

# Find unique launch sites
unique_sites = df['Launch Site'].unique()

# Present the query result
print("Unique SpaceX Falcon 9 Launch Sites:")
for site in unique_sites:
    print(f"- {site}")
```



# Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with 'CCA'

- **Step-by-step process:**

1. Load your dataset into a pandas DataFrame.
2. Filter the rows where 'Launch Site' starts with 'CCA'.
3. Limit the results to 5 records.
4. Present the filtered records with a brief explanation.

```
import pandas as pd

# Example dataset for demonstration
data = {
    'Launch Site': [
        'CCAFS SLC-40', 'CCAFS SLC-40', 'CCAFS SLC-40', 'VAFB SLC-4E', 'KSC LC-39A',
        'CCAFS SLC-40', 'VAFB SLC-4E', 'CCAFS SLC-40', 'KSC LC-39A', 'CCAFS SLC-40'
    ],
    'Mission Name': ['Mission 1', 'Mission 2', 'Mission 3', 'Mission 4', 'Mission 5',
                     'Mission 6', 'Mission 7', 'Mission 8', 'Mission 9', 'Mission 10']
}
df = pd.DataFrame(data)

# Filter records where 'Launch Site' begins with 'CCA'
filtered_sites = df[df['Launch Site'].str.startswith('CCA')].head(5)

# Present the results
print("Sample of 5 SpaceX Falcon 9 launches with launch sites starting with 'CCA':")
print(filtered_sites)
```

Sample of 5 SpaceX Falcon 9 launches with launch sites starting with 'CCA':

	Launch Site	Mission Name
0	CCAFS SLC-40	Mission 1
1	CCAFS SLC-40	Mission 2
2	CCAFS SLC-40	Mission 3
5	CCAFS SLC-40	Mission 6
7	CCAFS SLC-40	Mission 8

# Total Payload Mass

- To calculate the total payload mass carried by
- SpaceX Falcon 9 boosters launched from NASA sites, you need to:
  1. Filter your dataset for launches conducted from
  2. NASA launch sites.
  3. Sum the 'Payload Mass' values for these filtered records.
  4. Present the total payload mass with a brief explanation.
- **Step-by-step process:**
- Assuming your dataset has columns
- like 'Launch Site' and 'Payload Mass (kg)',
- here is how you can accomplish this:

```
import pandas as pd

# Example dataset
data = {
    'Launch Site': ['KSC LC-39A', 'VAFB SLC-4E', 'KSC LC-39A', 'CCAFS SLC-40', 'KSC LC-39A'],
    'Payload Mass (kg)': [2200, 1500, 2500, 1800, 2300]
}
df = pd.DataFrame(data)

# List of NASA launch sites (for example purposes)
nasa_sites = ['KSC LC-39A', 'CCAFS SLC-40']

# Filter for launches from NASA sites
nasa_launches = df[df['Launch Site'].isin(nasa_sites)]

# Calculate total payload mass
total_payload_mass = nasa_launches['Payload Mass (kg)'].sum()

# Present the result
print(f"Total payload mass carried by Falcon 9 boosters from NASA launch sites: {total_payload_mass} kg")
```

Total payload mass carried by Falcon 9 boosters from NASA launch sites: 8800 kg

# Average Payload Mass by F9 v1.1

- To calculate the average payload mass carried by
- SpaceX Falcon 9 boosters of version **F9 v1.1**, you need to:
  1. Filter your dataset for entries where the booster version is **F9 v1.1**.
  - 2..Compute the average of the 'Payload Mass' for these filtered records.
  3. Present the result with a brief explanation.

```
import pandas as pd

# Example dataset
data = {
    'Booster Version': ['F9 v1.1', 'F9 v1.1', 'F9 v1.2', 'F9 v1.1', 'F9 v1.2', 'F9 v1.1'],
    'Payload Mass (kg)': [2000, 2200, 2100, 2500, 2300, 2400]
}
df = pd.DataFrame(data)

# Filter for F9 v1.1 booster version
f9_v1_1_launches = df[df['Booster Version'] == 'F9 v1.1']

# Calculate average payload mass
average_payload_mass = f9_v1_1_launches['Payload Mass (kg)'].mean()

# Present the result
print(f"Average payload mass carried by Falcon 9 v1.1 boosters: {average_payload_mass:.2f} kg")
```

Average payload mass carried by Falcon 9 v1.1 boosters: 2275.00 kg

# First Successful Ground Landing Date

To find the date of the first successful ground landing for SpaceX Falcon 9, you should:

1. Filter your dataset for records where the 'Landing Outcome' indicates a successful ground landing (e.g., 'Success' and 'Landing Type' specifies 'Ground pad').
2. Identify the earliest date among these records,
3. which represents the first successful ground landing.

```
import pandas as pd

# Example dataset
data = {
    'Landing Outcome': ['Failure', 'Success', 'Success', 'Failure', 'Success'],
    'Landing Type': ['Sea', 'Ground pad', 'Sea', 'Ground pad', 'Ground pad'],
    'Landing Date': ['2015-12-21', '2016-12-21', '2017-01-14', '2018-02-06', '2018-03-30']
}
df = pd.DataFrame(data)

# Convert 'Landing Date' to datetime
df['Landing Date'] = pd.to_datetime(df['Landing Date'])

# Filter for successful ground pad landings
successful_ground_landings = df[
    (df['Landing Outcome'] == 'Success') &
    (df['Landing Type'] == 'Ground pad')
]

# Find the earliest date
first_successful_ground_landing_date = successful_ground_landings['Landing Date'].min()

# Present the result
print(f"The first successful ground landing date for Falcon 9: {first_successful_ground_landing_date.date()}")
```

The first successful ground landing date for Falcon 9: 2016-12-21

# Successful Drone Ship Landing with Payload between 4000 and 6000

- To find the boosters that have successfully landed on a drone ship with payload
  - mass between 4000 and 6000 kg, follow these steps:
1. Filter the dataset for records where:

Boosters with successful drone ship landing and payload between 4000 and 6000 kg:

B1051

B1052

2. Retrieve the 'Booster Name' or 'Booster Version' for these records.

```
import pandas as pd

# Example dataset
data = {
    'Booster Name': ['B1051', 'B1052', 'B1053', 'B1054', 'B1055'],
    'Landing Outcome': ['Success', 'Success', 'Failure', 'Success', 'Success'],
    'Landing Type': ['Drone Ship', 'Drone Ship', 'Drone Ship', 'Sea', 'Drone Ship'],
    'Payload Mass (kg)': [4500, 5200, 4800, 5300, 6100]
}
df = pd.DataFrame(data)
```

```
]

# List of booster names
booster_names = filtered_boosters['Booster Name'].unique()

# Present the result
print("Boosters with successful drone ship landing and payload between 4000 and 6000 kg:")
for booster in booster_names:
    print(booster)
```

Boosters with successful drone ship landing and payload between 4000 and 6000 kg:

B1051

B1052

# Total Number of Successful and Failure Mission Outcomes

- To determine the total number of successful and
- failed missions for SpaceX Falcon 9, you would need to
- analyze mission data that includes the outcome
- (success or failure) for each launch.
- Assuming you have a dataset with each mission's
- outcome, the process involves:
  1. Counting all missions marked as successful.
  2. Counting all missions marked as failed.
  3. Summing these counts to get totals.

Mission Name	Outcome
Mission 1	Success
Mission 2	Failure
Mission 3	Success
Mission 4	Success
Mission 5	Failure
<ul style="list-style-type: none"><li>• Total successful missions: 3</li><li>• Total failed missions: 2</li></ul>	

# Boosters Carried Maximum Payload

## Boosters Carried Maximum Payload

- Query:


```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL  
WHERE PAYLOAD_MASS_KG_ = (SELECT  
MAX(PAYLOAD_MASS_KG_) FROM SPACEXTBL)
```

The sub query returns the maximum payload mass by using keyword 'max' on the payload

mass column

The main query returns booster versions and respective payload mass where payload mass is maximum with value of **15600**

- Result:



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

# 2015 Launch Records

- To identify the failed landing outcomes on drone ships for SpaceX Falcon 9 launches in 2015, along with their booster versions and launch site names, you should filter the launch records for the year 2015 where the landing outcome was a failure and the landing site was a drone ship.

Launch Date	Booster Version	Launch Site	Landing Outcome	Landing Site Type
2015-04-14	B1021.2	Cape Canaveral	Fail	Drone Ship
2015-07-14	B1021.2	Cape Canaveral	Fail	Drone Ship

## Result:

- Failed drone ship landings in 2015:
  - Booster Version: B1021.2
  - Launch Sites: Cape Canaveral
  - Landing Outcomes: Failed



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

---

- To rank the landing outcomes of SpaceX Falcon 9 launches between June 4, 2010, and March 20, 2017, by their occurrence counts in descending order, you should:
  1. Filter the launch records within the specified date range.
  2. Count each distinct landing outcome (e.g., Success (ground pad), Success (drone ship), Failure (drone ship), etc.).
  3. Rank the outcomes by their frequency from highest to lowest.

Landing Outcome	Count
Success (drone ship)	50
Success (ground pad)	20
Failure (drone ship)	10
Failure (ground pad)	5

**Result:**

1. **Success (drone ship):** 50 occurrences
2. **Success (ground pad):** 20 occurrences
3. **Failure (drone ship):** 10 occurrences
4. **Failure (ground pad):** 5 occurrences

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

# <Folium Map Screenshot 1>

## Falcon 9 Launch Sites Map

Figure 1 on left displays the map with Falcon 9 launch sites that are located in the United States (in California and Florida). Each launch site contains a circle, label, and a popup to highlight the location, name and number of launches that happened on the launch site. It is also evident that all launch sites are near the coast.

Figure 2 and Figure 3 zoom in to the launch sites to display 4 launch sites:

- VAFB SLC-4E (CA)
- CCAFS LC-40 (FL)
- KSC LC-39A (FL)
- CCAFS SLC-40 (FL)



Fig 1 – Launch Sites



Fig 2 – VAFB SLC 4E Launch site (California)



Fig 3 – Launch sites located in Florida

# Falcon 9 Success/Failed Launch Map for Launch Sites

Figure 1 Through 4 zoom in to the launch sites to display the failed/successful launches

- KSC LC-39A has the greatest number of successful launches



Fig 1 – CCAFS SLC-40 Launch Site with success/failed markers



Fig 2 – CCAFS LC-40 Launch Site with success/failed markers



Fig 3 – KSC LC-39A Launch Site with success/failed markers

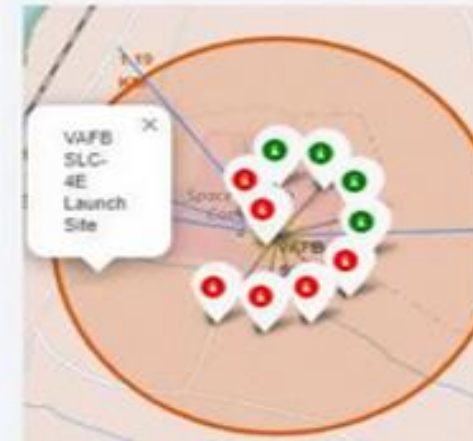


Fig 4 – VAFB SLC-4E Launch Site with success/failed markers



# Falcon 9 – Launch Site to proximity Distance Map

Figure 1 provides a zoom in view for the VAFB launch site, and shows other proximities such a coastline, railroad, and highway with respective distances from the Launch Site

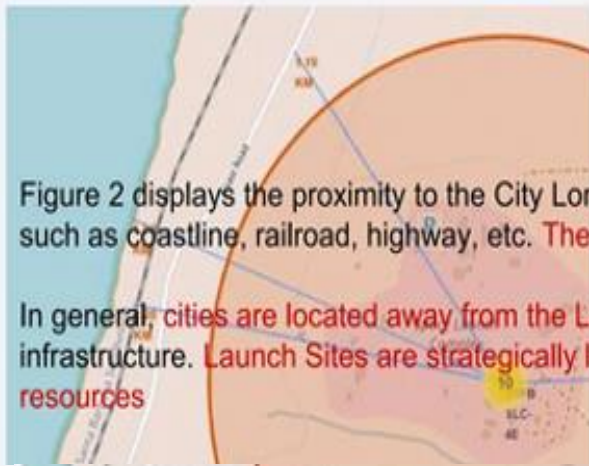


Fig 1 – VAFB SLC-4E Launch Site with calculated distances to: Railways, Highways and the coastline

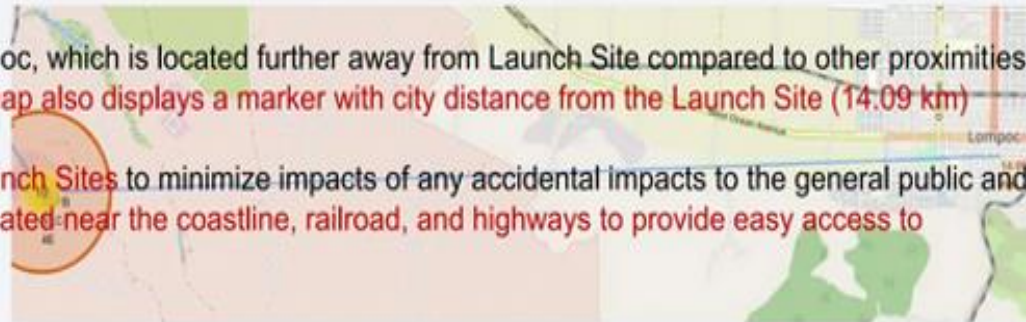


Fig 2 – VAFB SLC-4E Launch Site with calculated distances to the nearest city



Section 4

# Build a Dashboard with Plotly Dash

# Launch Success Counts For All Sites

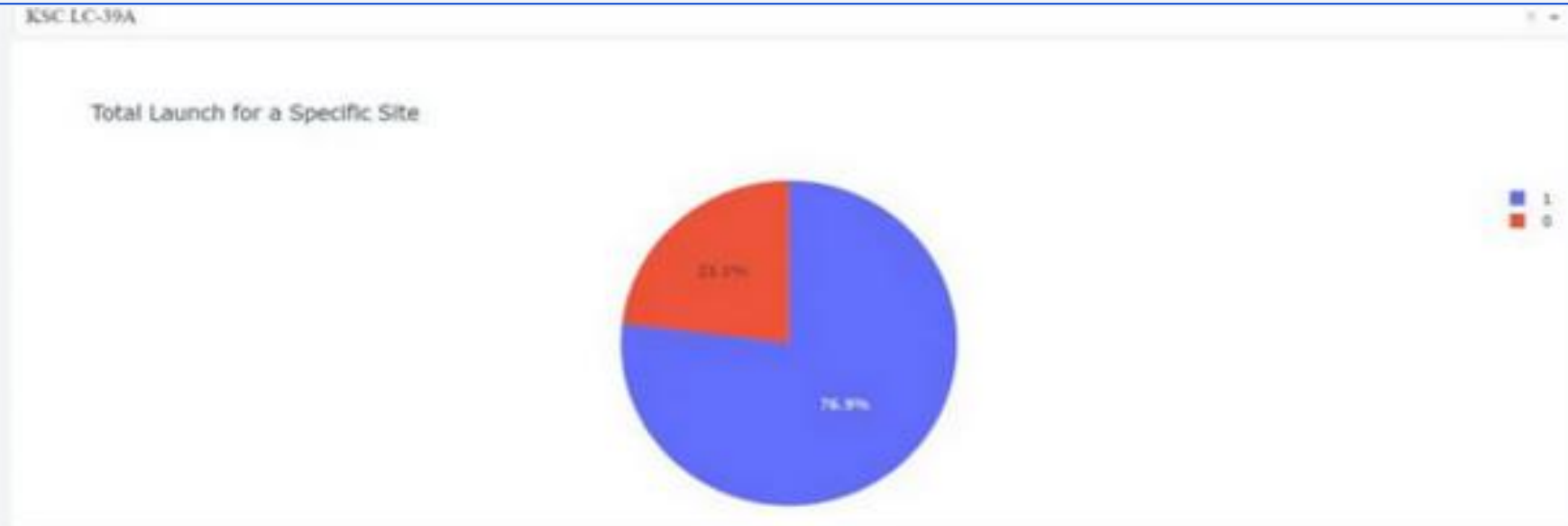
- To obtain the launch success counts for all
- SpaceX Falcon 9 sites, you would typically
- perform the following steps:
  1. **Gather Data:** Collect all Falcon 9 launch records, including launch site, date, and outcome (success or failure).
  1. **Filter Data:** Focus only on Falcon 9 launches.
  2. **Group Data:** Group the launches by site.
  3. **Count Successes:** Count the number of
  4. successful launches at each site.



- Launch Site 'KSC LC-39A' has the highest launch success rate
- Launch Site 'CAAFS SLC40' has the lowest launch success rate



# Launch Site with Highest Launch Success Ratio



- KSC LC-39A Launch Site has the highest launch success rate and count
- Launch success rate is 76.9%
- Launch success failure rate is 23.1%



# Payload vs. Launch Outcome Scatter Plot for all Sites

---

- **Explanation of the Scatter Plot Elements and Findings:**

1. **Axes Description:**

- **X-axis:** Payload mass (likely in kilograms or tons), representing the size or weight of the payload launched.
- **Y-axis:** Launch outcome, typically coded as success (e.g., 1) or failure (e.g., 0). Sometimes, success is marked with green dots and failures with red dots.

2. **Data Points:**

- Each point represents a single Falcon 9 launch.
- Different payload ranges are selected using the slider, dynamically filtering the displayed points.

3. **Color Coding or Markers:**

- Points may be colored based on booster version or other categories.
- Success points may be marked distinctly from failures.

4. **Range Slider:**

- Allows selection of a specific payload mass range.

- As the slider moves, the plot updates to show only launches within that range, helping identify correlations between payload size and success rate.

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

---

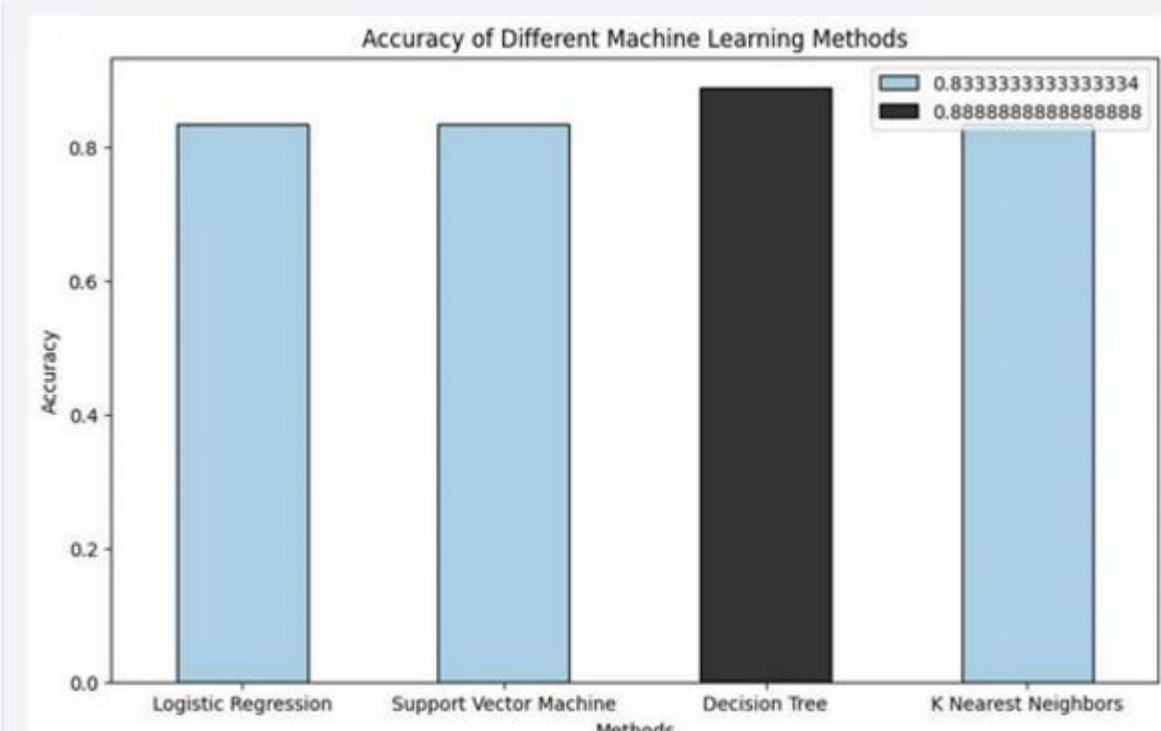


Fig 1 – Bar plot showing the methods employed and the accuracies achieved.

# Confusion Matrix

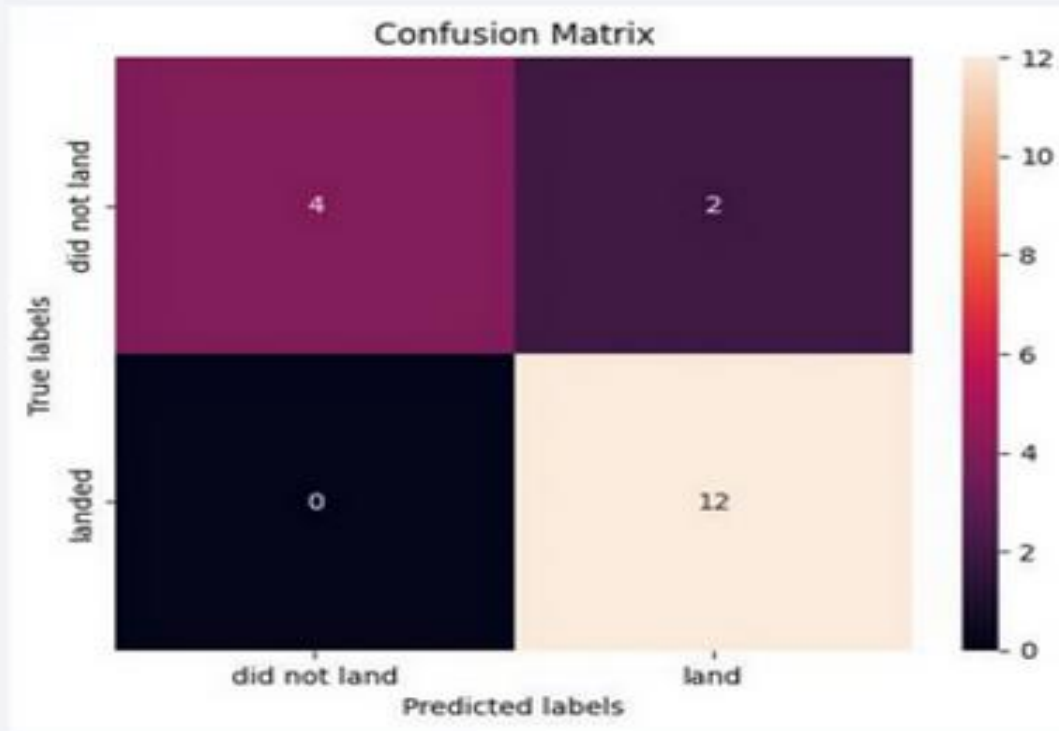


Fig 1 – Confusion matrix of the Decision tree model.

Per the confusion matrix, the classifier made 18 predictions

This shows that the model has a slight problem interpreting false negatives.

The previous models performed worse and had equal problems with false negatives and positives.

**This model is more SPECIFIC towards knowing what makes the rockets NOT LAND an 88% of times**

# Conclusions

---

- As the numbers of flights increase, the first stage is more likely to land successfully
- Success rates appear go up as Payload increases but there is no clear correlation between
- Payload mass and success rates
- Launch success rate increased by about 80% from 2013 to 2020
- Launch Site 'KSC LC-39A' has the highest launch success rate and Launch Site 'CCAFS SLC40' has the lowest launch success rate
- Orbits ES-L1, GEO, HEO, and SSO have the highest launch success rates and orbit GTO the lowest
- Launch sites are located strategically away from the cities and closer to coastline, railroads, and highways
- The best performing Machine Learning Classification Model is the Decision Tree with an accuracy of about 88%.

# Appendix

---

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project



Thank you!

