SpaceX Launch Analysis with ML

An Applied Data Science Capstone Project

By Stalin Jaquez, Jan. 30th 2025

Executive Summary

This project analyzes the history of SpaceX launches, focusing on key differences in rockets, booster versions, launch sites, payload objectives, and mission outcomes. Leveraging machine learning models trained on historical data, we developed predictive tools to assess future launch success with measurable confidence. As part of the IBM Data Science Professional Certificate program, this analysis applies data science methodologies, Python programming, SQL, and data visualization to extract insights, enhance predictions, and refine decision-making in aerospace technology.

Key Findings:

- **Success Rates Vary by Launch Site -** KSC LC-39A has achieved the most efficient success rate.
- **Booster Version Impacts Reliability –** The FT booster type presents the highest success rate for boosters with more than 1 attempt.
- **Payload Mass Influences Success –** The lower the payload the higher the success outcome. A negative correlation between the two.
- **Predictive Modeling is Consistent –** Our model converged achieved a 94% accuracy in predicting launch outcomes.

Recommendations and Impact:

- These findings can assist SpaceX to optimize mission planning by selecting the most reliable booster versions, payload, and launch site.
- The classification model can help predict negative launch outcomes, assist the SpaceX team to manage risk, and increase success rates overall.
- Further refinement of the model, incorporating real-time launch conditions, could enhance accuracy and reliability.

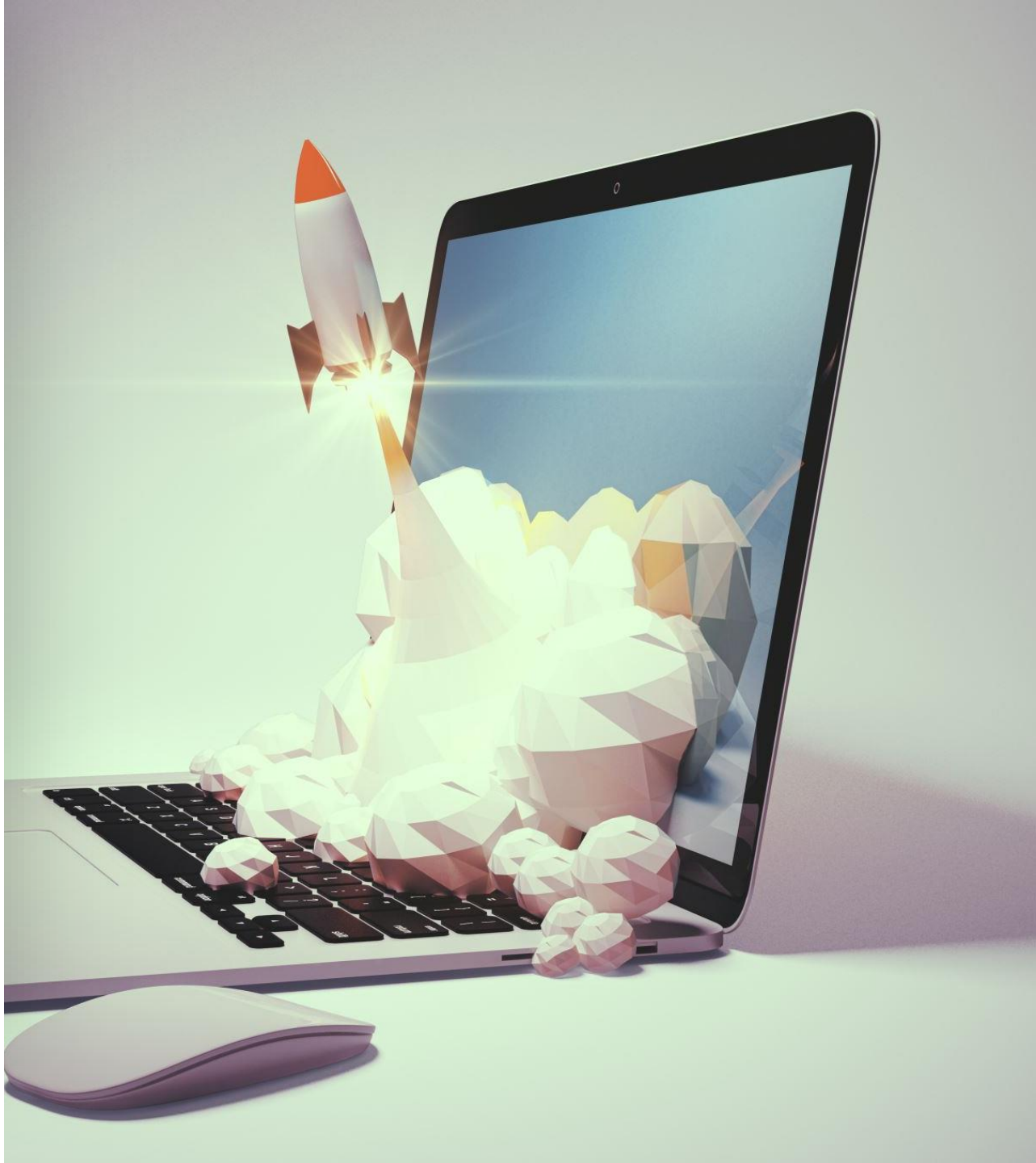


Table of Contents

Introduction

The current approach to space travel is extremely expensive, inefficient, and resource-intensive, making widespread exploration and commercialization challenging. SpaceX aims to revolutionize the industry by improving rocket reusability, significantly reducing costs. However, not all launches achieve success, necessitating a deeper understanding of influencing factors. This project analyzes key variables affecting launch outcomes and develops a predictive model to enhance mission planning. By leveraging machine learning, data science methodologies, and historical launch data, the model improves decision-making, optimizes resource allocation, and increases success rates. Ultimately, this approach helps reduce overall costs by refining the selection of independent variables and features crucial to launch success.

Methodology

Data Collection

The data was collected from the SpaceX API, web scrapping, static json URLs, and pre-processed csv files. Below are all the resources used for data collection during the project:

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

https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv

https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv

https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv

https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_dash.csv

After cleaning and wrangling, exploratory analysis revealed trends in launch success. Classification models, including KNN and logistic regression, were tested to predict launch outcomes.

Data Collection – SpaceX API

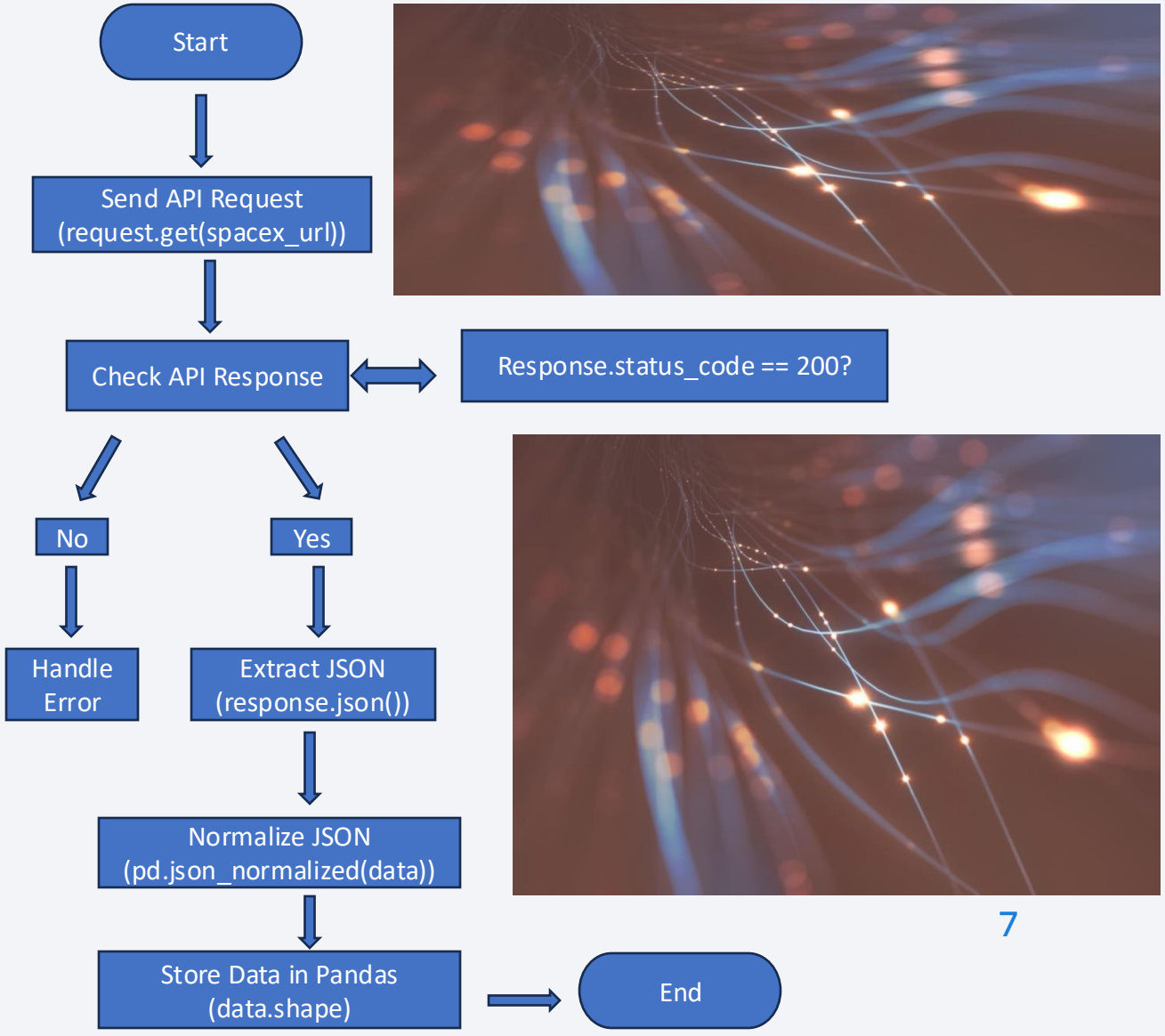
SpaceX REST calls code snippet:

```
# Request and parse the SpaceX launch data using the GET request

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

response = requests.get(static_json_url)

(response.status_code) # Output: 200

data = response.json()

data = pd.json_normalize(data)

(data.shape) # (107, 42)
```

GitHub URL of the completed SpaceX API calls notebook

https://github.com/Mojica23/Applied-Data-Science-Capstone-/blob/main/M1_Data_Collection_API_Lab.ipynb

API Call FlowChart

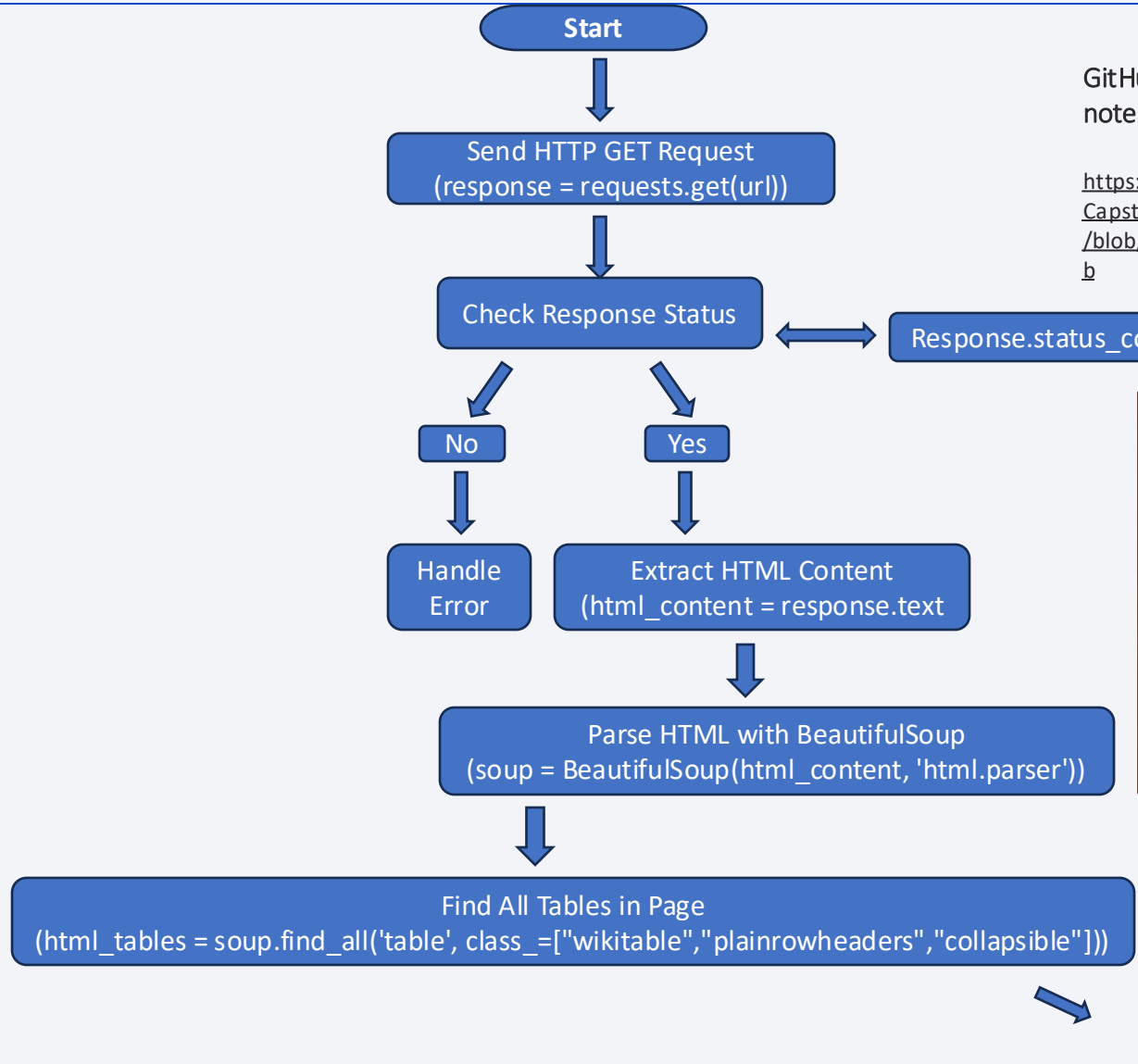


Data Collection - Scraping

Web Scraping Code Snippet

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

# Send an HTTP Get request to the web page

response = requests.get(url)

# Store the HTML content in a variable.

html_content = response.text

# Create a BeautifulSoup Object to parse the HTML

soup = BeautifulSoup(html_content,'html.parser')

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

soup_title = soup.title # Output: <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

# Find all tables on the wiki page

html_tables = soup.find_all('table', class_=["wikitable", "plainrowheaders", "collapsible"])

# The third table (index 2) contains the actual launch records

first_launch_table = html_tables[2]
```

Web Scraping Flowchart



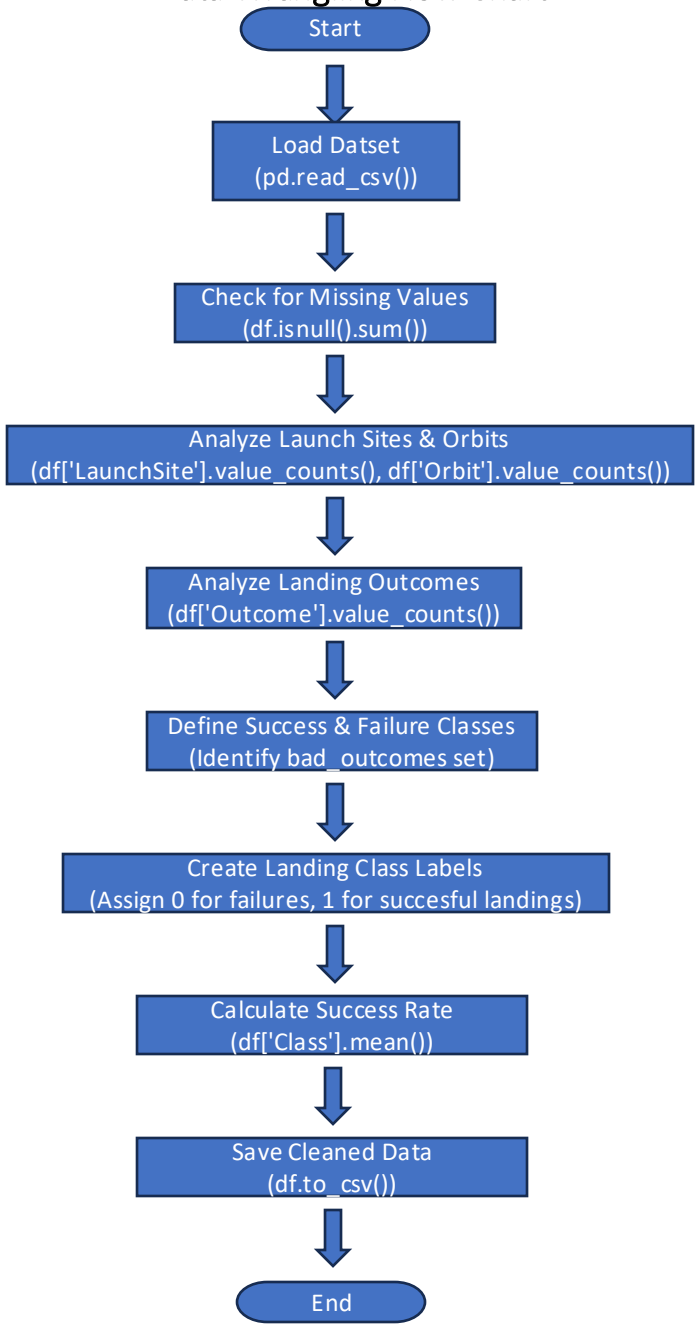
GitHub URL of the completed SpaceX web scraping notebook:

https://github.com/Mojica23/Applied-Data-Science-Capstone-/blob/main/M1_Data_Collection_SpaceX_Web_Scraping.ipynb

Data Wrangling Code Snippet

```
df = pd.read_csv('IBM Data Certificate/Course11_Applied_Data_Science_Capstone/dataset_part_1.csv')

# Unique outcomes
outcomes = df['Outcome'].value_counts()

# Identify and calculate the percentage of the missing values in each attribute
pct_missing = df.isnull().sum()/len(df)*100 # Output: LandingPad 28.9%
launch_sites = df['LaunchSite'].value_counts()
'''
LaunchSite
CCSFS SLC 40 55
KSC LC 39A 22
VAFB SLC 4E 13
'''
# TASK 2: Calculate the number and occurrence of each orbit.
# Orbit types
orbit_types = df['Orbit'].value_counts()

# TASK 3: Calculate the number and occurence of mission outcome of the orbits.
# Use the method .value_counts() on the column Outcome to determine the number of landing_outcomes.Then assign it to a variable landing_outcomes.

landing_outcomes = df['Outcome'].value_counts()
for i,outcome in enumerate(landing_outcomes.keys()):
                    i,outcome
# We create a set of outcomes where the second stage did not land successfully
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
# Output: {'False RTLS', 'False Ocean', 'None ASDS', 'None None', 'False ASDS'}

# Using the Outcome, create a list where the element is zero if the corresponding row in Outcome is in the set bad_outcome; otherwise, it's one. Then assign it to the variable landing_class:
landing_class = []
for outcome in df['Outcome']:
if outcome in bad_outcomes:
landing_class.append(0)
else:
landing_class.append(1)
```

Data Wrangling Flow Chart



GitHub URL of the data wrangling related notebooks:

https://github.com/Mojica23/Applied-Data-Science-Capstone-/blob/main/M1_Data_Wrangling_Lab.ipynb

Plotted Charts

Plot #1 – Flight Number vs Payload Mass (Categorical Plot)

- This plot shows how payload mass varies across different flight numbers, with the launch outcome (success/failure)**.** This type was used becaause it is useful for comparing discrete values.

Plot# 2 – Flight Number vs Launch Site (Categorical Plot)

- This plot displays the relationship between flight number and launch site, distinguishing successful and failed launches. This type was used because it is useful for comparing discrete values.

Plot# 3 – Payload Mass vs Launch Site (Categorical Plot)

- This plot visualizes the payload mass for each launch site, with launch success/failure. This type was used becaause it is useful for comparing discrete values.

Plot# 4 – Orbit Type vs Success Rate (Bar Chart)

A bar chart showing the average success rate for each orbit type. This type was used because they're useful for comparing success rates across categories.

Plot# 5 – Flight Number vs Orbit Type (Scatter Plot)

- This plot maps flight numbers to orbit types, with successful vs. failed launches indicated by hue.

Plot# 6 – Payload Mass Orbit Type (Categorical Plot)

- A scatter plot showing how payload mass varies across orbit types, with launch success/failure color-coded. This type was used becaause it is useful for comparing discrete values.

Plot# 7 – Launch Success Yearly Trend (Line Chart)

- A line plot showing the trend of launch success rates over the years. This type was used because they're useful for tracking trends over time.

GitHub URL of the completed EDA with data visualization notebook

https://github.com/Mojica23/Applied-Data-Science-Capstone-/blob/main/M2P2_EDA_With_Visualization.ipynb

EDA With SQL

- **Retrieve Unique Launch Sites** – Extracts distinct launch sites from the dataset. **Code:** task_1 = "'SELECT DISTINCT Launch_Site FROM SPACEXTBL'"
- **Filter Launches from Specific Sites** – Retrieves the first five records where the launch site name starts with "CCA."**Code:** task_2 = "'SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;'"
- **Total Payload Mass for NASA (CRS) Missions** – Calculates the sum of payload mass carried by boosters launched for NASA (CRS). **Code:** task_3 = "'SELECT CUSTOMER, SUM(PAYLOAD_MASS__KG_) AS SUM_PAYLOAD_MASS_KG FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';'"
- **Average Payload Mass for F9 v1.1 Boosters** – Computes the mean payload mass for launches using the "F9 v1.1" booster version. **Code:** task_4 = "'SELECT Booster_Version, AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_MASS__KG FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1' '"
- **First Successful Ground Pad Landing** – Identifies the earliest date when a booster successfully landed on a ground pad. **Code:** task_5 = "'SELECT Landing_Outcome, MIN(DATE) AS First_Succesful_Landing_Date FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)' '"
- **Successful Drone Ship Landings with Specific Payload Mass** – Lists boosters that landed successfully on a drone ship while carrying a payload between 4000 kg and 6000 kg. **Code:** task_6 = "'SELECT DISTINCT(Booster_Version),PAYLOAD_MASS__KG_ ,Landing_Outcome FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000 '"
- **Count of Mission Outcomes** – Counts the number of successful and failed missions in the dataset. **Code:** task_7 = "'SELECT Mission_Outcome, COUNT(*) AS Count FROM SPACEXTBL GROUP BY Mission_Outcome;'"
- **Boosters with Maximum Payload Mass** – Retrieves boosters that carried the heaviest payload. **Code:** task_8 = "'SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);'"
- **Failed Drone Ship Landings in 2015** – Lists booster versions and launch sites for drone ship failures in the year 2015. **Code:** task_9 = "'SELECT strftime('%m', Date) AS Month, strftime('%Y', Date) AS Year, Booster_Version, Launch_Site, Landing_OutcomeFROM SPACEXTBL WHERE Landing_Outcome LIKE 'Failure (drone ship)'AND Year = '2015';'"
- **Ranked Landing Outcomes from 2010 to 2017** – Counts and ranks landing outcomes (successes and failures) between June 2010 and March 2017 in descending order. **Code:** task_10 = "'SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Outcome_Count DESC;'"

GitHub URL of completed EDA with SQL notebook:
https://github.com/Mojica23/Applied-Data-Science-Capstone-/blob/main/M2P1_EDA_With_SQL_Lab.ipynb

Interactive Map with Folium

Markers and Circles for Launch Sites:
Added labeled markers and circles at four SpaceX launch sites to visually represent their locations on the map. These markers and circles helps identify launch locations and distinguish them easily.

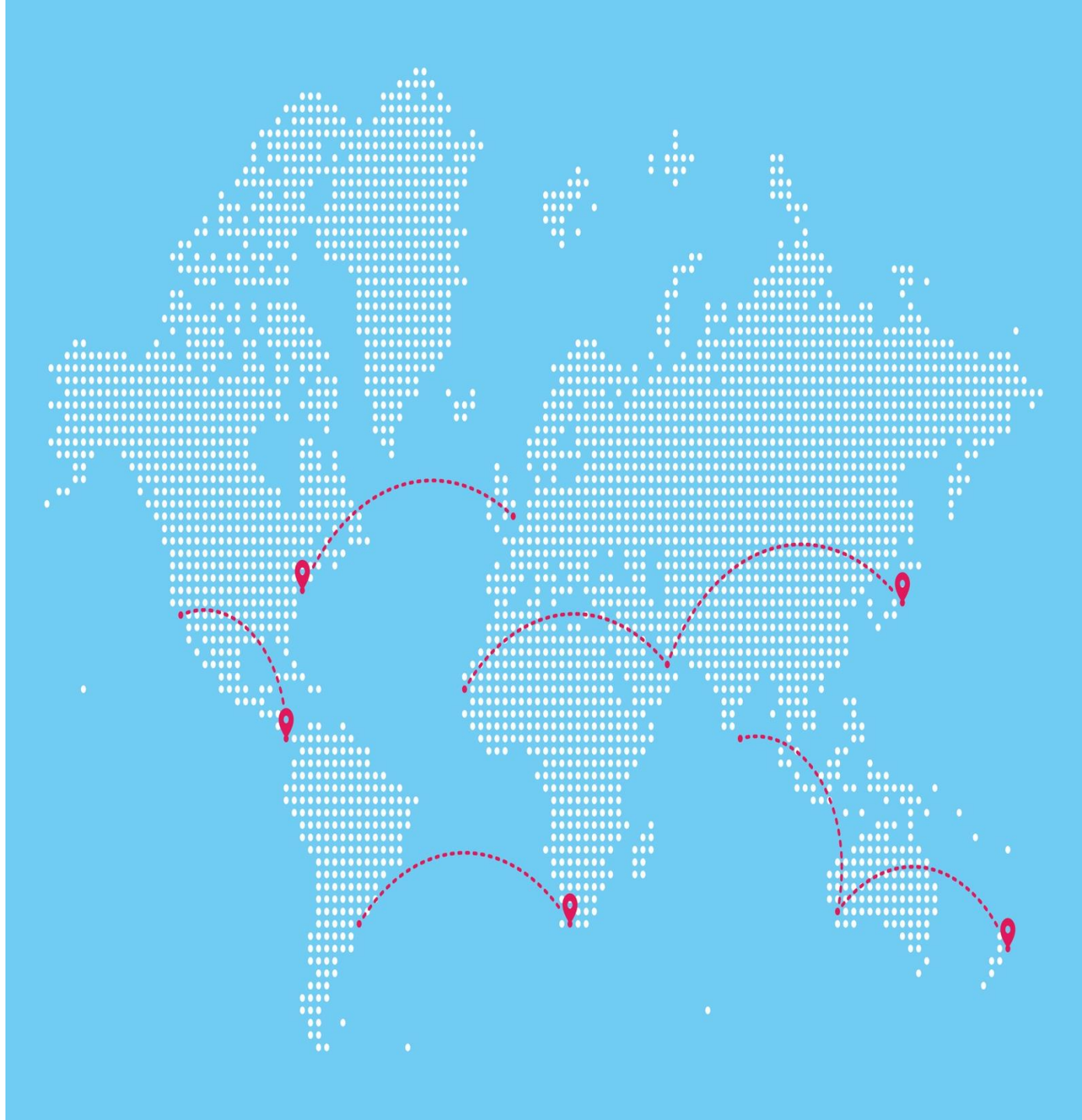
Success/Failure Markers Using Clustered Markers:
Colored markers (green for success, red for failure) were added for each launch site. Provides a quick visual overview of launch outcomes.

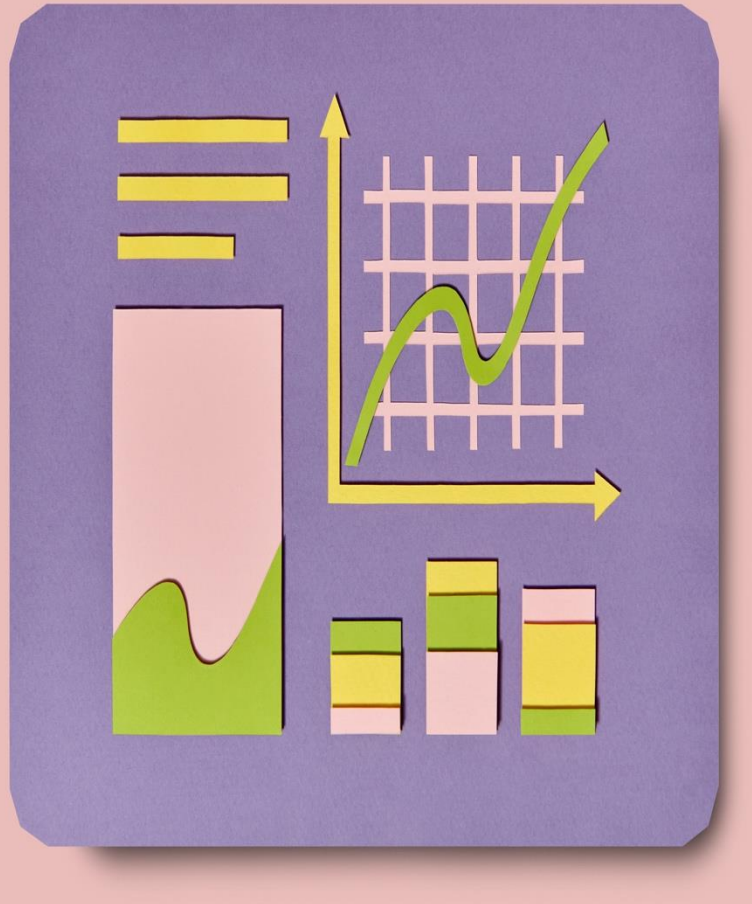
Mouse Position Tracker:
Enabled a tool to display coordinates of any point when hovering over the map. Helps find precise coordinates of key landmarks like railways, highways, and coastlines.

Distance Calculations and Labels:
Calculated and displayed distances between a launch site and its nearest coastline, railway, city, and highway. Shows proximity to infrastructure, which is crucial for launch logistics and safety.

PolyLines (Connecting Key Locations):
Drew lines between the launch site VAFB SLC-4E and its closest city, railway, highway, and coastline. Highlights key geographic relationships affecting launch operations.

GitHub URL of completed interactive map with Folium map:
https://github.com/Mojica23/Applied-Data-Science-Capstone-/blob/main/M3P1_Visual_Analytics_With_Folium_Lab.ipynb





GitHub URL of the completed Plotly Dash lab

https://github.com/Mojica23/Applied-Data-Science-Capstone-/blob/main/Dashboard.ipynb

Summary of Plots/Graphs and Interactions in the Dashboard

The dashboard features two interactive visualizations that provide insights into SpaceX launch successes:

1. A pie chart that displays the proportion of successful launches across all launch sites or a selected site:

- Launch Sites included:
- CCAFS LC-40 (Cape Canaveral Space Force Station)
- CCAFS SLC-40 (Another launch site at Cape Canaveral)
- KSC LC-39A (Kennedy Space Center)
- VAFB SLC-4E (Vandenberg Air Force Base, California)

2. A scatter plot that shows the correlation between payload mass and launch success, with the ability to filter by launch site and payload range.

- Booster Versions included:
- F9 v1.0
- F9 v1.1
- F9 FT (Full Thrust)
- F9 B4 (Block 4)
- F9 B5 (Block 5)

Why These Plots and Interactions Were Added

- The pie chart facilitates a comparison of launch success rates across various sites, identifying the most consistent locations.
- The scatter plot examines the connection between payload mass and launch outcomes, illustrating how different booster versions perform under diverse payload weights.
- The dropdown menu allows users to select specific launch sites, making it easier to identify patterns and performance trends.
- The range slider offers an interactive way to assess payload mass influence on launch success, revealing the impact of lighter and heavier payloads on mission results.

Predictive Analysis (Classification) / Model Development Process

Building the Models

Several classification models were trained: Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN).

- The dataset was standardized, and train-test splitting (80/20) was applied to improve model generalization.

Evaluating Performance

- GridSearchCV was used to find the best hyperparameters for each model.
- Each model's accuracy was evaluated on test data using accuracy scores and confusion matrices.

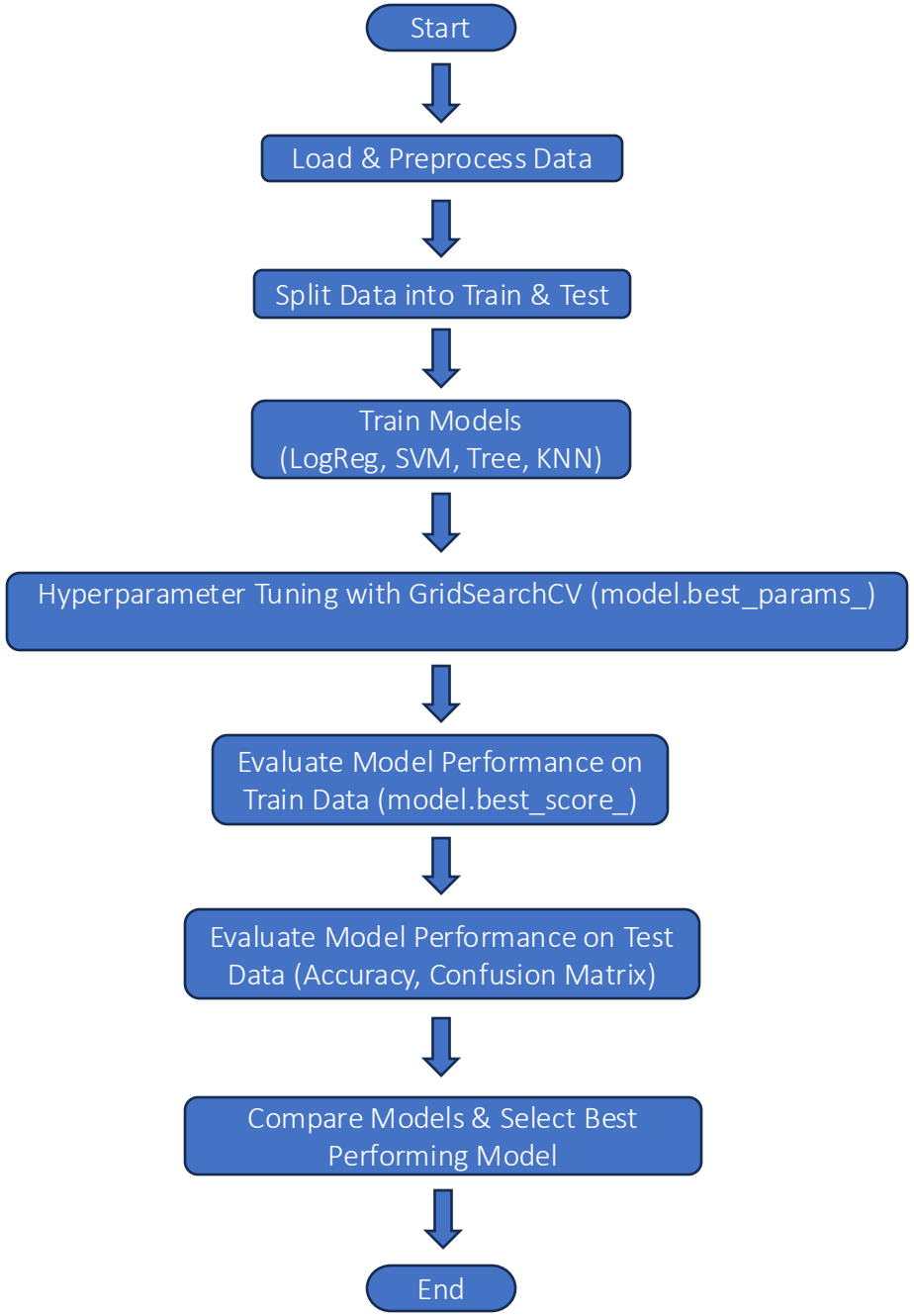
Improving the Models

- Hyperparameter tuning optimized performance by testing different values for parameters like C, kernel, max_depth, and n_neighbors.
- The models were compared to identify the most accurate classifier.

Finding the Best Model

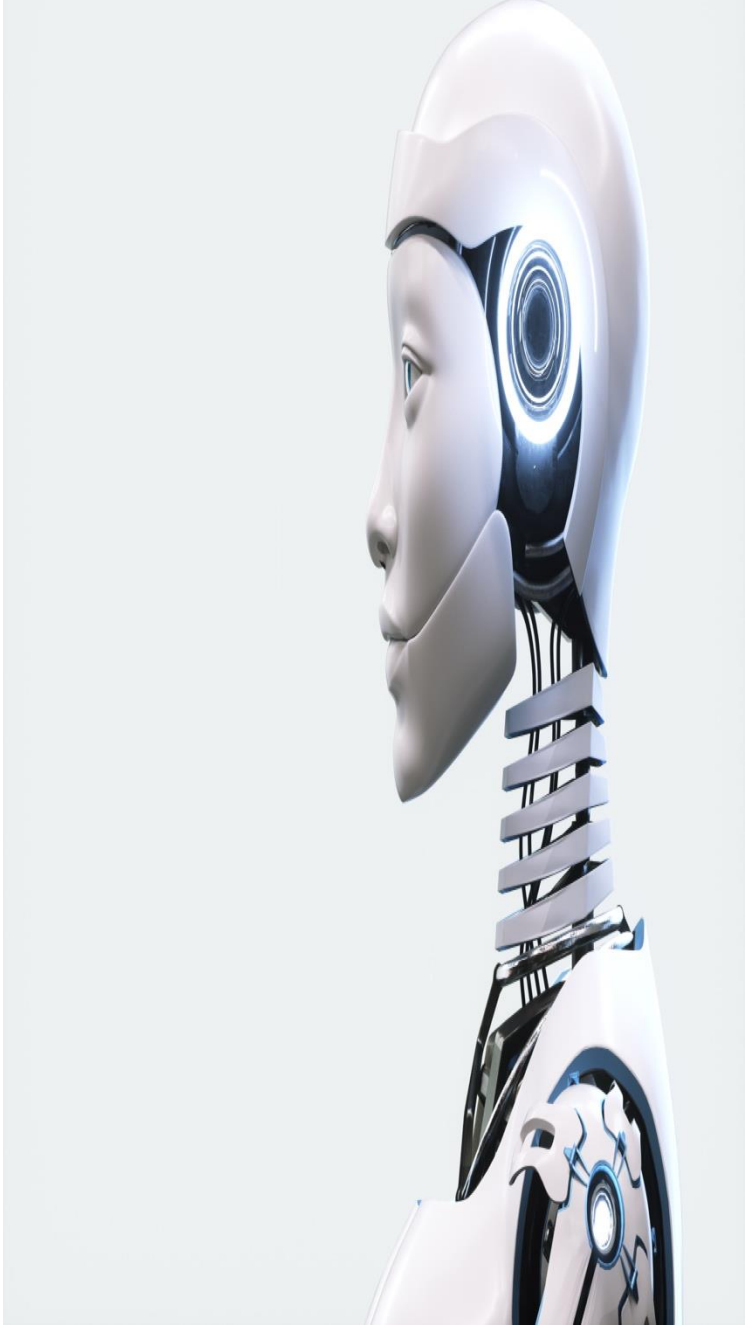
- The Logistic Regression model achieved the highest accuracy, making it the most reliable classifier for predicting SpaceX launch outcomes.
- A bar chart compared the best accuracy scores across models.

Model Design FlowChart



GitHub URL of the completed Plotly Dash lab

https://github.com/Mojica23/Applied-Data-Science-Capstone-/blob/main/M4P1_1_ML_Prediction_Lab.ipynb



Exploratory Data Analysis

The data collected from the static json URL experienced the following pre-processing and conversions:

Filtering through the list to only include Falcon 9 launches.
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']

We can see below that some of the rows were missing values in our dataset.
null_values = (data_falcon9.isnull().sum())
# Output: PayloadMass 5, LandingPad 26

We dealt with the missing values by calculating the mean of PayloadMass using the .mean(), and then using the mean and the .replace() function to replace np.nan values in the data with the mean calculated. The LandingPad None values were kept.
payload_mean =
data_falcon9['PayloadMass'].mean().__round__(2)
Output: 6123.55

data_falcon9['PayloadMass'] =
data_falcon9['PayloadMass'].replace(np.nan,payload_mean)

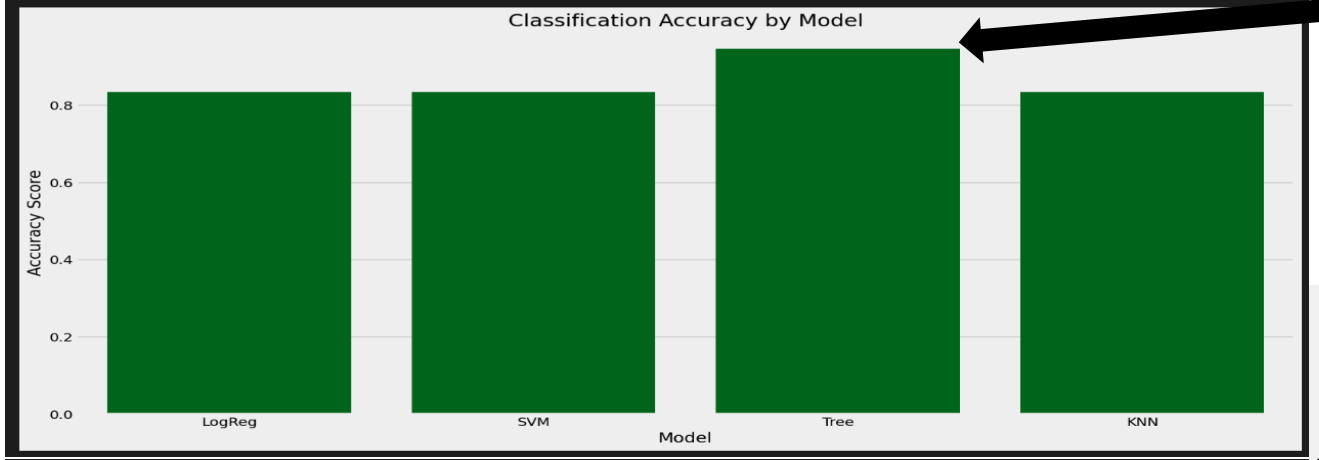
New null_values
null_values = data_falcon9.isnull().sum()
(null_values)
Output: LandingPad 26

The dataset includes multiple SpaceX launch sites, such as Cape Canaveral (CCAFS LC-40, CCAFS SLC-40), Vandenberg (VAFB SLC-4E), and Kennedy Space Center (KSC LC-39A). The .value_counts() function was used to count the number of launches at each site, helping identify the most frequently used locations.
launch_sites = df['LaunchSite'].value_counts()

LaunchSite.	Count
CCSFS SLC 40	55
KSC LC 39A	22
VAFB SLC 4E	13

Predictive Analysis Results

	Model	Accuracy Score
0	LogReg	0.83
1	SVM	0.83
2	Tree	0.94
3	KNN	0.83

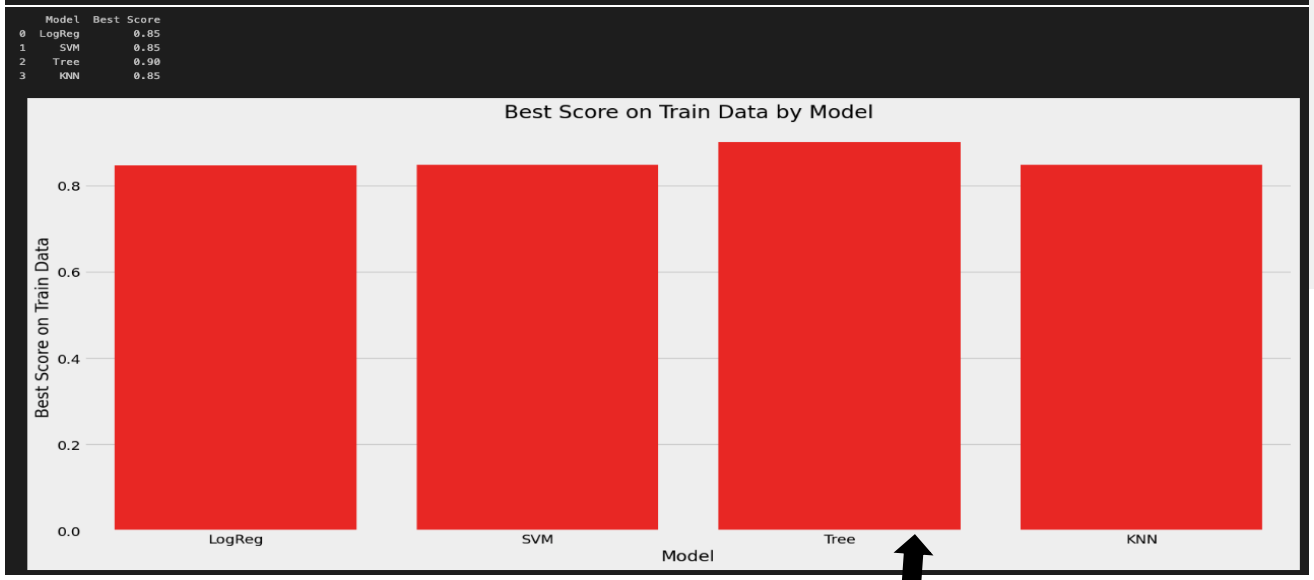


Here we can see the most accurate model was the Tree model with a 94% accuracy score.

Model Accuracy Score

LogReg	0.83
SVM	0.83
Tree	0.94
KNN	0.83

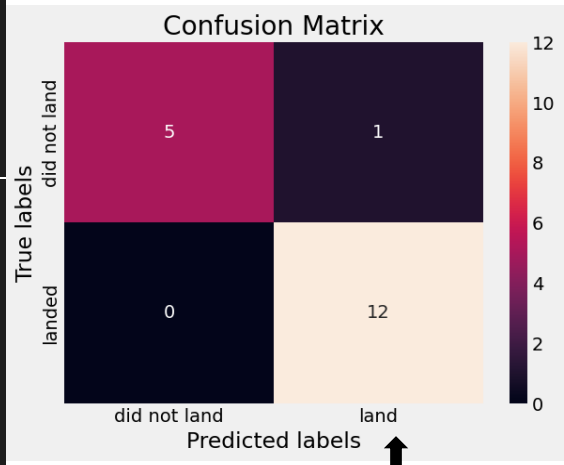
	Model	Best Score
0	LogReg	0.85
1	SVM	0.85
2	Tree	0.90
3	KNN	0.85



Model Best Score

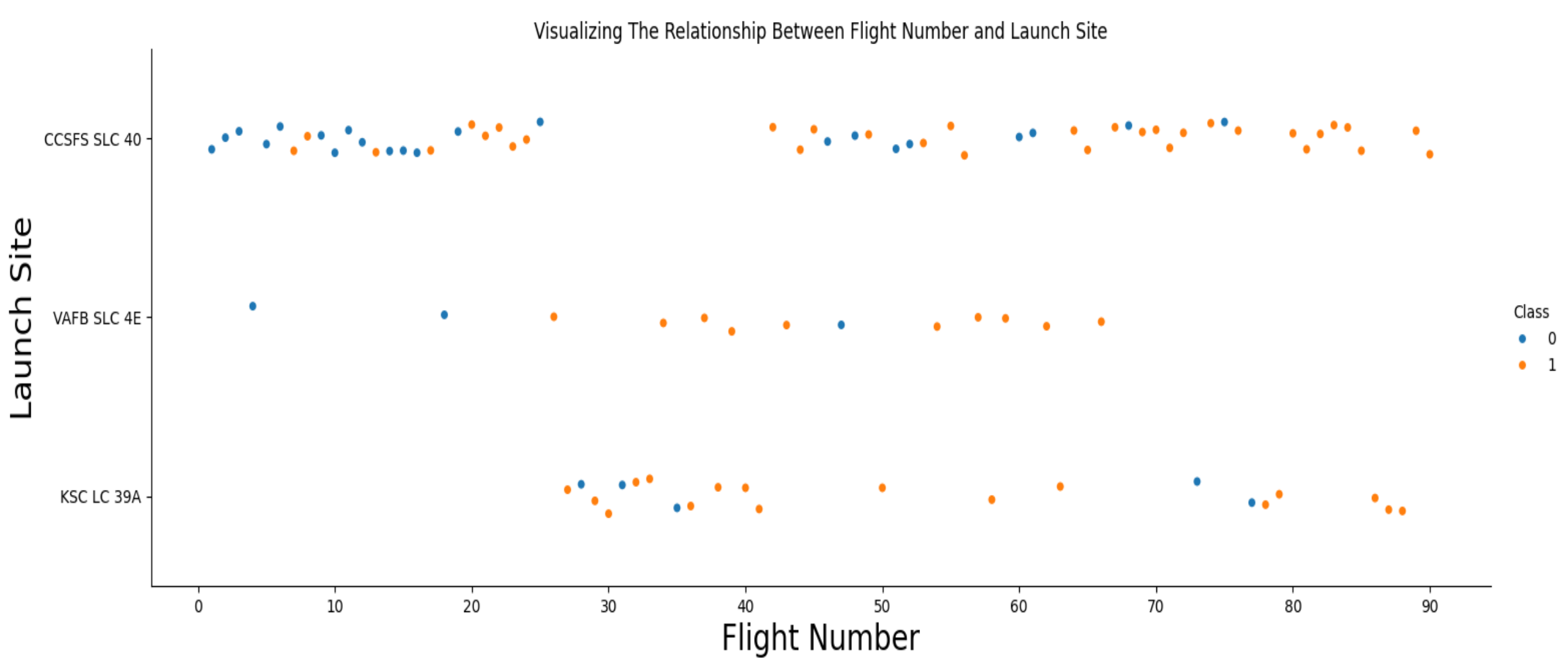
LogReg	0.85
SVM	0.85
Tree	0.90
KNN	0.85

Here we can see the most accurate model on the train data was the Tree model with a 90% accuracy score.



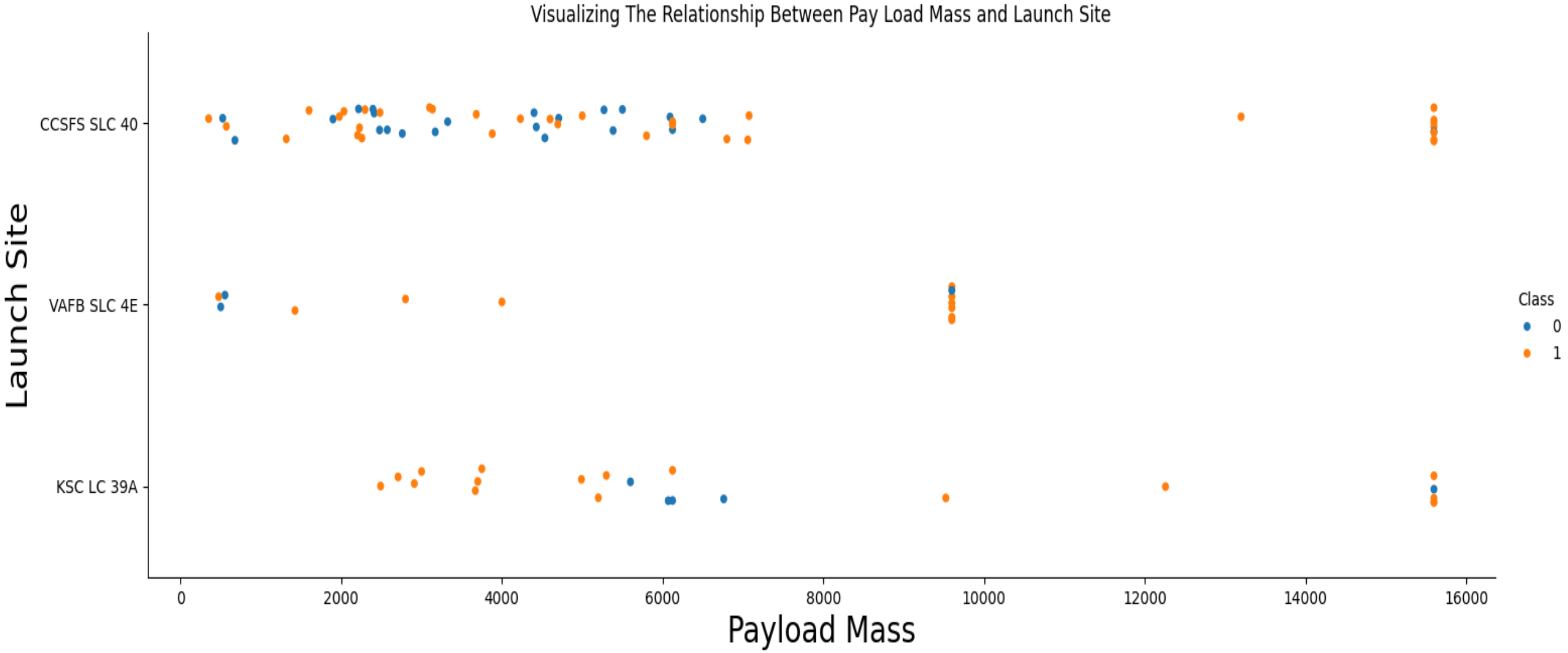
Performance Metrics:

- **True Positives (TP) = 12 →** Model correctly predicted "landed" 12 times.
- **True Negatives (TN) = 5 →** Model correctly predicted "did not land" 5 times.
- **False Positives (FP) = 1 →** Model incorrectly predicted "landed" when it did not land once (Type I error).
- **False Negatives (FN) = 0 →** Model never failed to predict "landed" when it actually landed



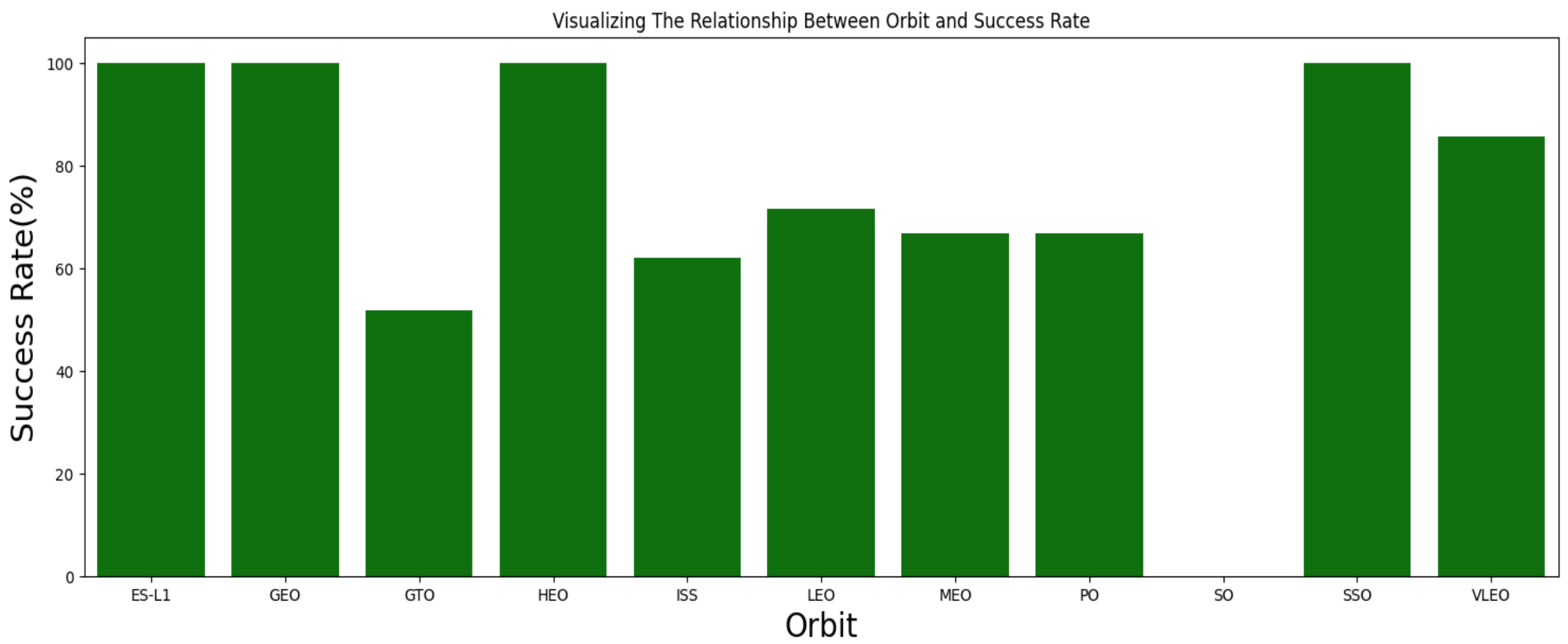
Plot Explanation

This scatter plot shows the relationship between flight number and launch site, with successful launches (orange) and failed launches (blue). Over time, the success rate improved, especially at CCSFS SLC-40 and KSC LC-39A, while VAFB SLC-4E had fewer launches. The trend suggests technological advancements led to better launch outcomes.



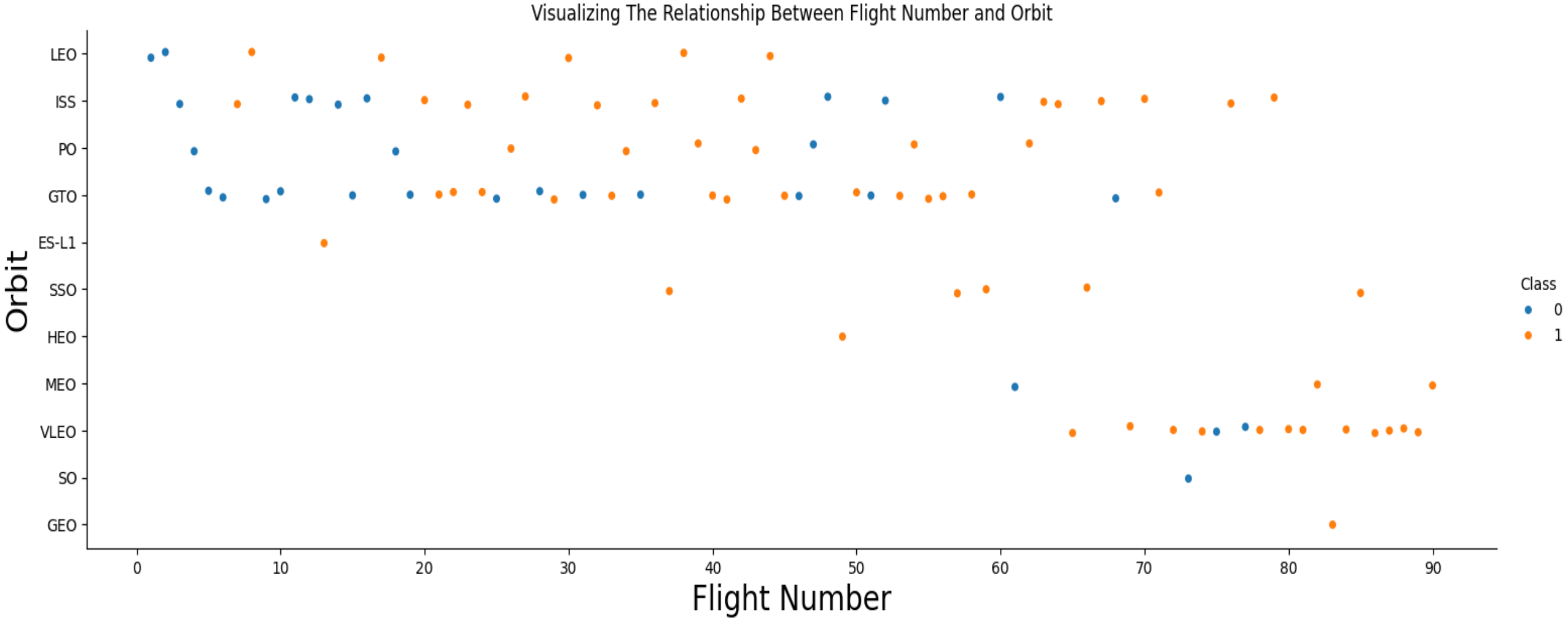
Plot Explanation

This scatter plot shows the relationship between payload mass and launch site, with successful (orange) and failed (blue) launches. KSC LC-39A handles a wider range of payloads, while VAFB SLC-4E has fewer launches. Higher payloads (above 10,000 kg) tend to have more successful launches, suggesting that payload mass does not strongly impact success rates.



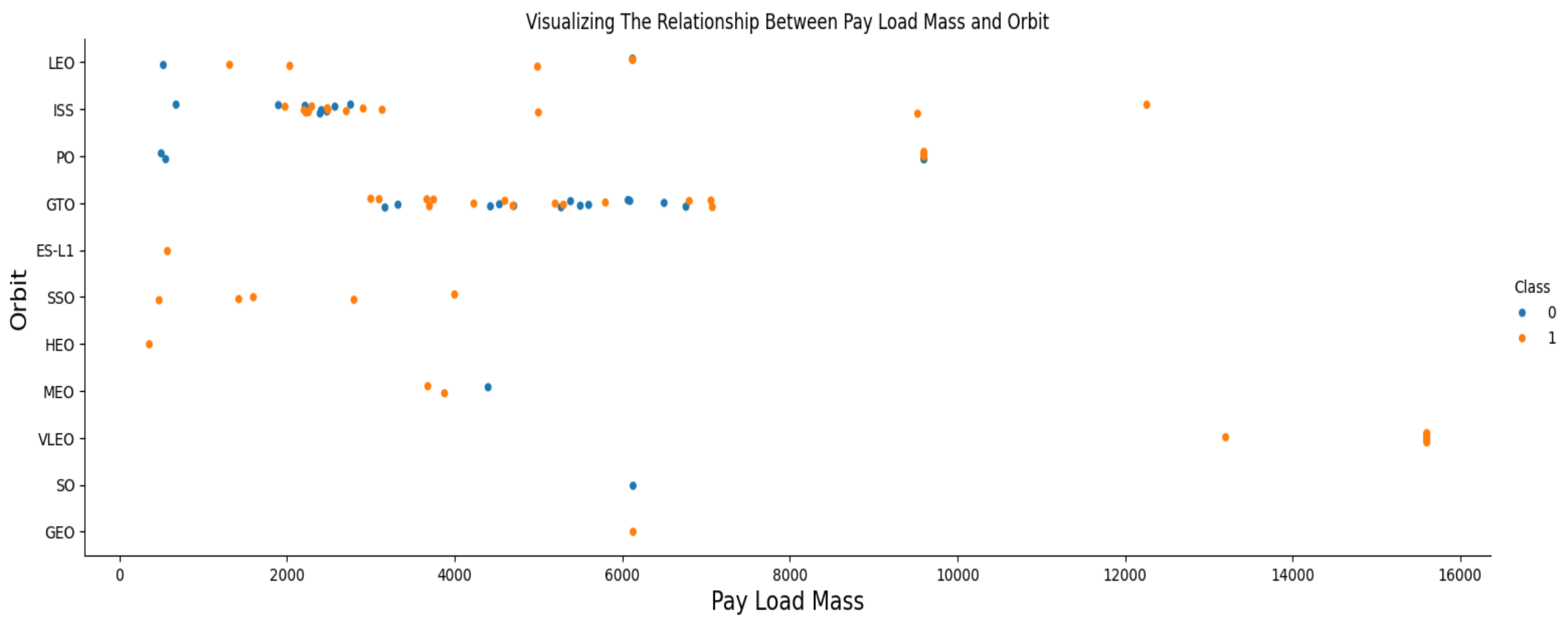
Bar Graph Explanation

This bar graph shows the success rate of launches for different orbit types. ES-L1, GEO, HEO, and SSO have the highest success rates (near 100%), while GTO has the lowest (50%). Other orbits like LEO, ISS, and VLEO show moderate success rates, suggesting that some orbits are more challenging for successful landings than others.



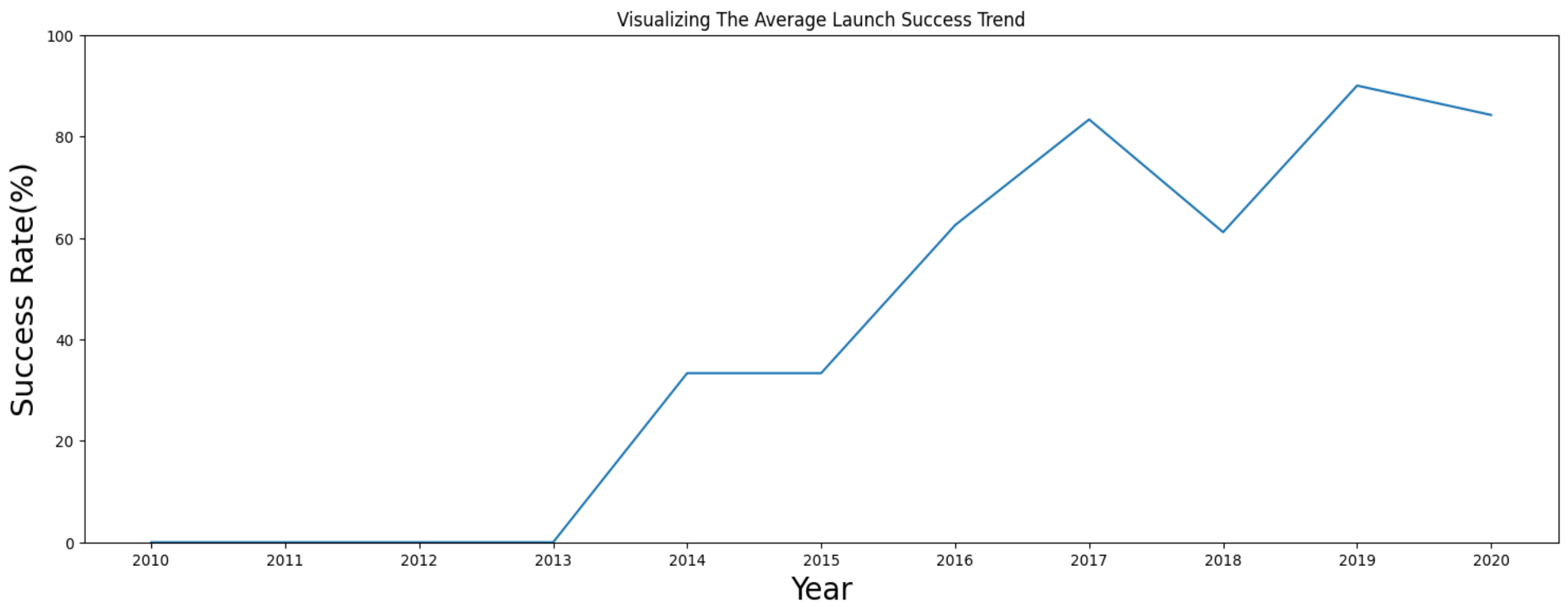
Plot Explanation

This scatter plot illustrates the relationship between flight number and orbit type, with successful (orange) and failed (blue) launches. Early missions had more failures, particularly in GTO and LEO orbits, while later missions show higher success rates across most orbits. The trend suggests improved launch reliability over time, especially for higher flight numbers.



Plot Explanation

This scatter plot visualizes the relationship between payload mass and orbit type, with successful (orange) and failed (blue) launches. Lower payloads (<2000 kg) have mixed results across multiple orbits, while heavier payloads (>10,000 kg) show a strong success rate, particularly in GEO and ES-L1 orbits. The trend suggests that certain orbits, like GTO and LEO, experience more failures compared to others.



Line Graph Explanation

This line graph shows the trend of launch success rates over time, highlighting a steady increase in successful missions from 2013 onward. Early years had low or no success, but from 2015 onward, success rates improved significantly, peaking around 2019 before experiencing a slight decline in 2020. This suggests technological advancements and operational improvements contributed to higher success rates over time.

Query to Find All Launch Site Names

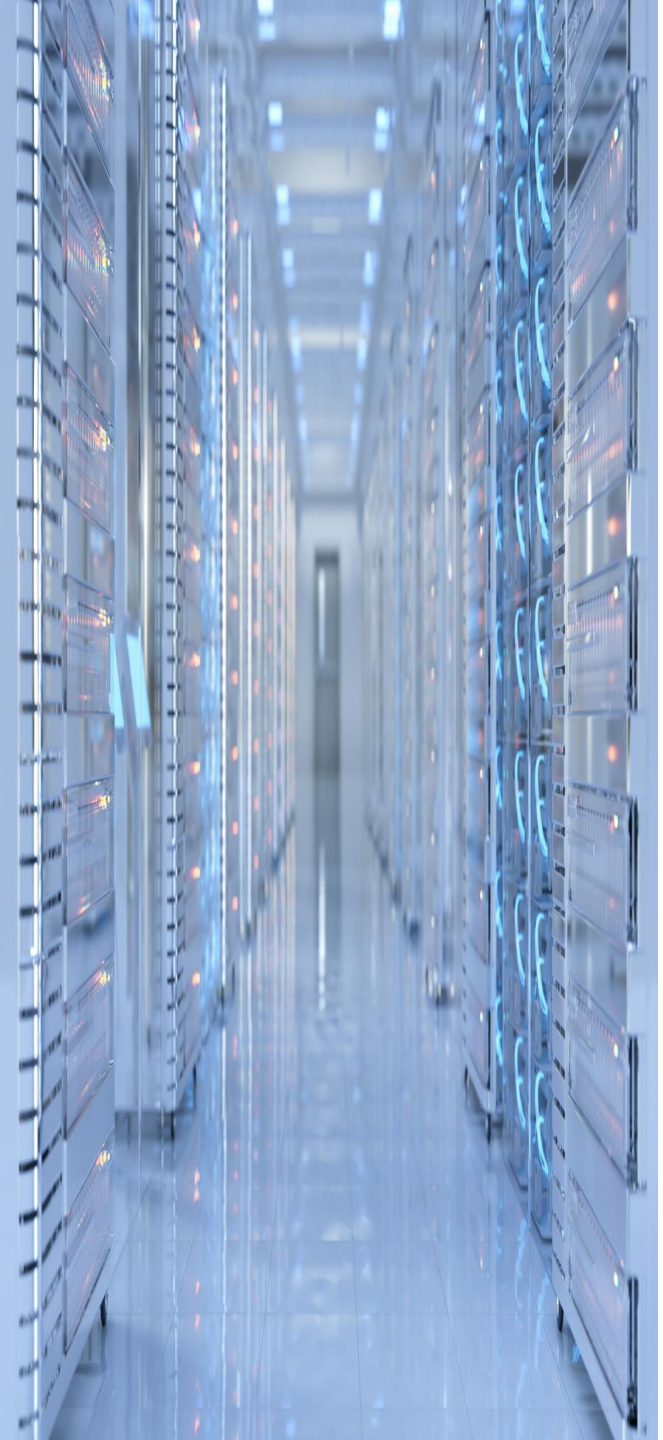
Task : Display the names of the unique launch sites in the space mission.

```
task_1 = '''SELECT DISTINCT Launch_Site FROM SPACEXTBL'''
cur.execute(task_1)
results = cur.fetchall()
columns =[]
for column in cur.description:
columns.append(column[0])

table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Explanation

We used SELECT DISTINCT to pull the unique rows from the column Launch_Site from the table SPACEXTBL.

```
+---------------+
| Launch_Site   |
+---------------+
| CCAFS LC-40   |
| VAFB SLC-4E   |
| KSC LC-39A    |
| CCAFS SLC-40  |
+---------------+
```



Query to Display 5 Records Where Launch_Site Begin With The String 'CCA'

Task : Display 5 records where launch site beings with the stirng 'CCA'

```
task_2 = '''SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;'''
cur.execute(task_2)
results = cur.fetchall()


columns = []
table = []
for column in cur.description:
columns.append(column[0])

table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Explanation

We used the SELECT * statement to retrieve all columns from the SPACEXTBL table while filtering rows based on the Launch_Site column. The WHERE clause, combined with the LIKE operator, was used to perform pattern matching with wildcards (%). Specifically, we searched for launch sites that start with “CCA” ('CCA%') to match relevant entries.

Output

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Query to Find Total Payload Mass

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

```
Task = '"SELECT CUSTOMER, SUM(PAYLOAD_MASS__KG_) AS SUM_PAYLOAD_MASS_KG FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';'"
cur.execute(task)
results = cur.fetchall()
```

Output

```
columns =[]
for column in cur.description:
columns.append(column[0])
```

```
+-----------+---------------------+
| Customer  | SUM_PAYLOAD_MASS_KG |
+-----------+---------------------+
| NASA (CRS) | 45596          |
+-----------+---------------------+
```

```
table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Explanation

Here we calculate the total payload mass carried by boosters for missions launched by NASA (CRS). The SUM(PAYLOAD_MASS__KG_) function is used to aggregate the payload mass for all rows where the CUSTOMER column matches 'NASA (CRS)'. The result, 45,596 kg, is then displayed in a formatted table.

Query to Find Average Payload Mass Carried by Booster F9 v1.1

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

```
task_4 = '''SELECT Booster_Version, AVG(PAYLOAD_MASS__KG_) AS
AVG_PAYLOAD_MASS__KG FROM SPACEXTBL WHERE Booster_Version =
'F9 v1.1' '''
```

```
cur.execute(task)
results = cur.fetchall()

columns =[]
for column in cur.description:
columns.append(column[0])
```

Output

```
+----------------+-------------------------------------+
| Booster_Version | AVG_PAYLOAD_MASS__KG |
+----------------+-------------------------------------+
| F9 v1.1            | 2928.4                          |
+----------------+-------------------------------------+
```

```
table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Explanation

Here we calculate the average payload mass carried by the F9 v1.1 booster version. The AVG(PAYLOAD_MASS__KG_) function computes the mean payload for all launches where Booster_Version is ‘F9 v1.1’. The result provides insight into the typical payload capacity of this specific booster.

Query to Find Date of First Successful Landing Outcome On Ground Pad

Task 5: List the date when the first succesful landing outcome in ground pad was acheived.

```
task = '''SELECT Landing_Outcome, MIN(DATE) AS
First_Succesful_Landing_Date FROM SPACEXTBL WHERE
Landing_Outcome = 'Success (ground pad)' '"

cur.execute(task)
results = cur.fetchall()

columns =[]
for column in cur.description:
columns.append(column[0])

table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Output

```
+-----------------------+-----------------------------+
| Landing_Outcome       |First_Succesful_Landing_Date |
+-----------------------+-----------------------------+
| Success (ground pad)  |2015-12-22                   |
+-----------------------+-----------------------------+
```

Explanation

Here we retrieve the earliest date when a rocket successfully landed on a ground pad. The MIN(DATE) function finds the first recorded success in the Landing_Outcome column where the value is ‘Success (ground pad)’. This helps identify the date of the first successful ground landing by SpaceX.

Query to Find Boosters Within A Specific Payload Range

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.

```
Task = '"SELECT DISTINCT(Booster_Version),PAYLOAD_MASS__KG_ ,Landing_Outcome
FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)'
AND PAYLOAD_MASS__KG_ > 4000
AND PAYLOAD_MASS__KG_ < 6000 '''

cur.execute(task)
results = cur.fetchall()

columns =[]
for column in cur.description:
columns.append(column[0])

table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Output

```
+---------------+-----------------+---------------------+
| Booster_Version | PAYLOAD_MASS__KG_ | Landing_Outcome |
+---------------+-----------------+---------------------+
| F9 FT B1022    | 4696           | Success (drone ship) |
| F9 FT B1026    | 4600           | Success (drone ship) |
| F9 FT B1021.2  | 5300           | Success (drone ship) |
| F9 FT B1031.2  | 5200           | Success (drone ship) |
+---------------+-----------------+---------------------+
```

Explanation

Here we retrieve distinct booster versions that successfully landed on a drone ship while carrying a payload mass between 4,000 and 6,000 kg. The DISTINCT(Booster_Version) ensures unique booster entries, and the WHERE clause filters records based on successful drone ship landings and the specified payload range. This helps analyze which boosters performed well under these conditions.

Query to Find the Total Number of Successful and Failure of Mission Outcomes

Task 7: List the total number of successful and failure mission outcomes.

```
Task = '''SELECT Mission_Outcome, COUNT(*) AS Count
FROM SPACEXTBL
GROUP BY Mission_Outcome;'''

cur.execute(task)
results = cur.fetchall()

columns =[]
for column in cur.description:
columns.append(column[0])

table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Output

```
+-------------------------------+-------+
| Mission_Outcome               | Count |
+-------------------------------+-------+
| Failure (in flight)           | 1     |
| Success                       | 98    |
| Success                       | 1     |
| Success (payload status unclear) | 1     |
+-------------------------------+-------+
```

Explanation

This SQL query counts the number of occurrences of each mission outcome in the SPACEXTBL table. The COUNT(*) function calculates the total number of missions for each unique Mission_Outcome, and the GROUP BY clause ensures results are grouped accordingly. This helps analyze the overall success and failure rates of SpaceX missions.

Query to Find the Names of The Boosters Which Carried The Max Payload Mass

Task : List the names of the booster_versions which have carried the maximum payload mass. Use a subquery.

```
Task = '"SELECT Booster_Version, PAYLOAD_MASS__KG_
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM
SPACEXTBL);
'''
cur.execute(task)
results = cur.fetchall()

columns =[]
for column in cur.description:
columns.append(column[0])

table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Output

```
+---------------+-----------------+
| 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 |
+---------------+-----------------+
```

Explanation

Here we retrieve the booster version that carried the maximum payload mass recorded in the SPACEXTBL table. The subquery (SELECT MAX(PAYLOAD_MASS__KG_)) finds the highest payload, and the outer query filters for the booster(s) that carried it. This helps identify which booster handled the heaviest payload.

Query to Find Launch Information From Months in 2015

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.

```
Task = '"SELECT strftime('%m', Date) AS Month, strftime('%Y', Date) AS Year, Booster_Version, Launch_Site, Landing_Outcome
FROM SPACEXTBL
WHERE Landing_Outcome LIKE 'Failure (drone ship)'
AND Year = '2015';'"

cur.execute(task)
results = cur.fetchall()

columns =[]
for column in cur.description:
columns.append(column[0])

table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Output

```
+-------+------+-----------------+-------------+---------------------+
| Month | Year | Booster_Version | Launch_Site | Landing_Outcome |
+-------+------+-----------------+-------------+---------------------+
| 01 | 2015 | F9 v1.1 B1012 | CCAFS LC-40 | Failure (drone ship) |
| 04 | 2015 | F9 v1.1 B1015 | CCAFS LC-40 | Failure (drone ship) |
+-------+------+-----------------+-------------+---------------------+
```

Explanation

Here we retrieve all failed drone ship landings that occurred in 2015, along with the booster version, launch site, and landing outcome. The strftime('%m', Date) and strftime('%Y', Date) functions extract the month and year from the Date column. The LIKE 'Failure (drone ship)' filter ensures only failed landings on drone ships are included. This helps analyze unsuccessful drone ship landings for that year.

Query to Rank Landing Outcome

Task : 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.

```
Task = '''SELECT Landing_Outcome, COUNT(*) AS Outcome_Count
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Outcome_Count DESC;'''

cur.execute(task)
results = cur.fetchall()

columns =[]
for column in cur.description:
columns.append(column[0])

table = PrettyTable(columns)
for row in results:
table.add_row(row)
```

Outcome

```
+---------------------+---------------+
| 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 |
+---------------------+---------------+
```

Explanation

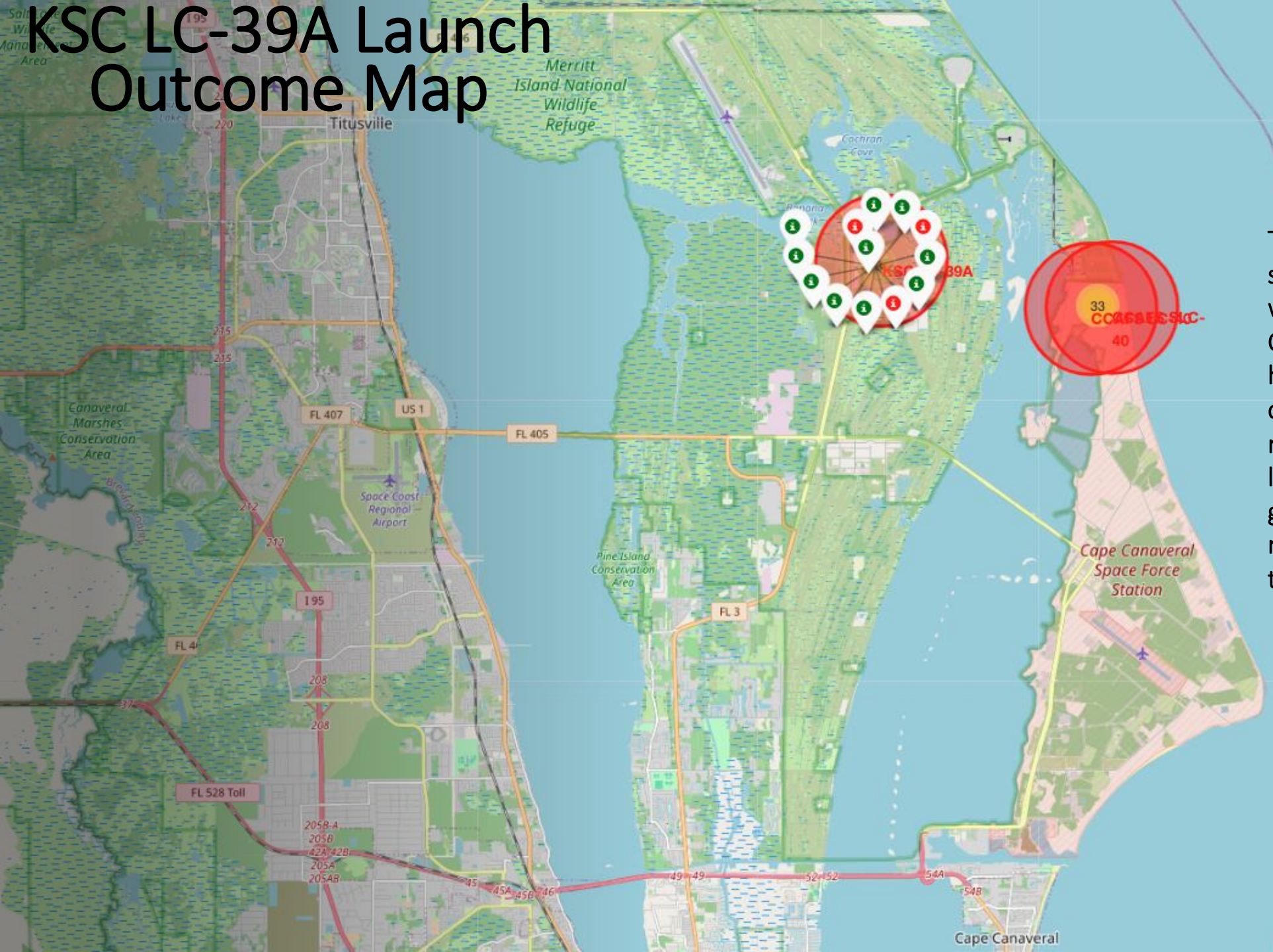
Here we count the number of occurrences of each landing outcome for launches that took place between June 4, 2010, and March 20, 2017. The COUNT(*) function calculates how often each Landing_Outcome appears, while GROUP BY Landing_Outcome ensures results are categorized by outcome type. The ORDER BY Outcome_Count DESC sorts the results in descending order, highlighting the most common landing outcomes during this period.

Launch Site Proximity Analysis

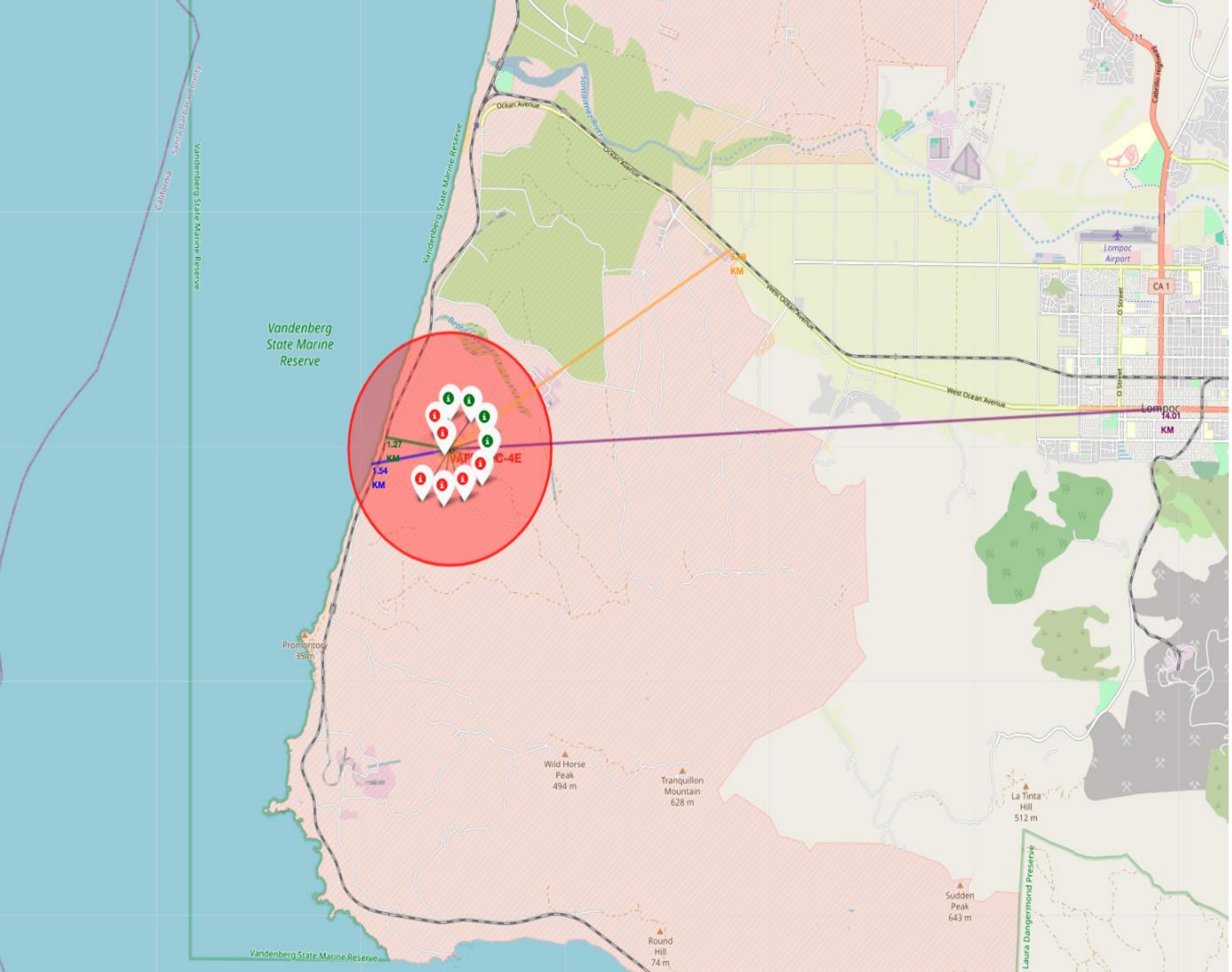
Folium Map Identifying Launch Sites

This global map generated by folium displays the multiple launch site across the US. Due to the proximity of the East Coast launch sites the names overlap. They are: **CCAFS LC-40**(Cape Canaveral, Florida), **CCAFS SLC-40**(Cape Canaveral, Florida), and **KSC LC-39A** (Kennedy Space Center, Florida). On the West Coast we have **VAFB SLC-4E** (Vandenberg Air Force Base, California).

KSC LC-39A Launch Outcome Map



This map visualizes SpaceX launch sites at Cape Canaveral, Florida, with key locations like KSC LC-39A, CCAFS LC-40, and CCAFS SLC-40 highlighted with the large red circles. The medium yellow circles represent launch attempts. On the left we see KSC LC-39A with red and green pop up info-icons. The green represent successful attempts, and the red represent failed attempts.



VAFB SLC-4E Launch Site

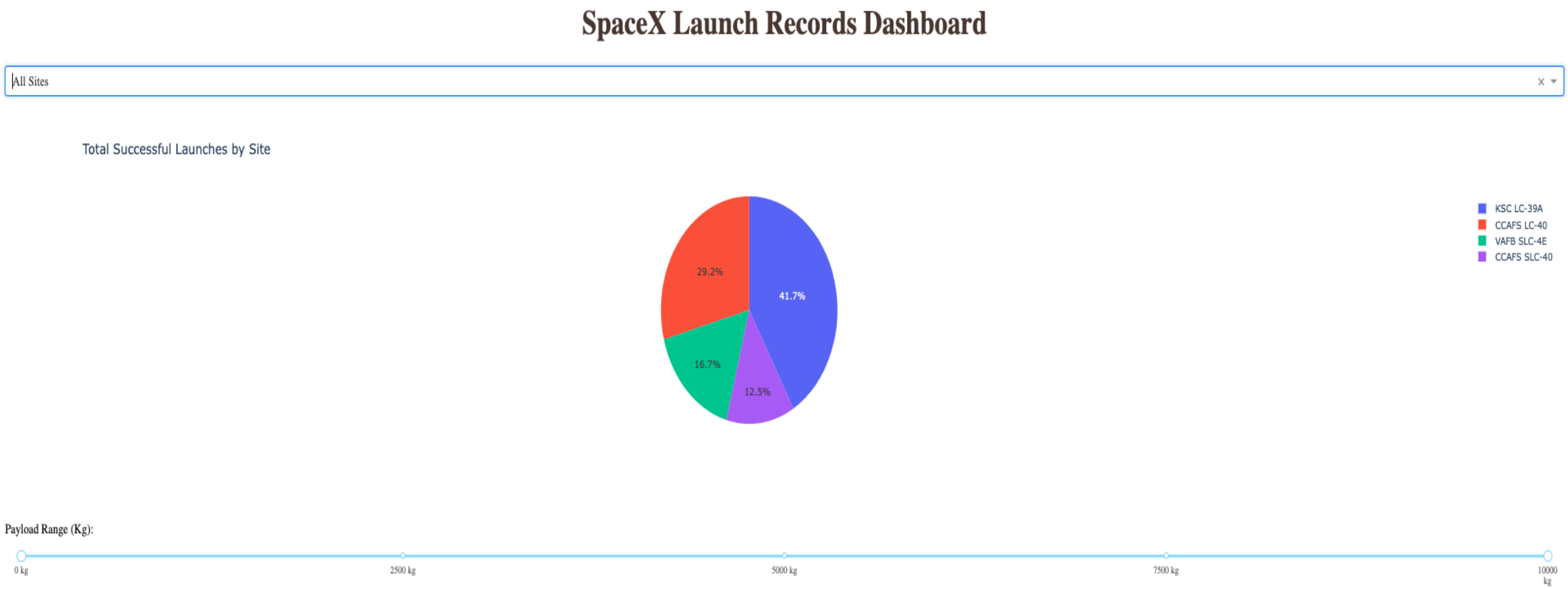
This map highlights Vandenberg Space Force Base (VAFB) SLC-4E, a key West Coast launch site primarily used for polar and sun-synchronous orbit missions.

Important Features & Elements:

- **Red Circle:** Represents the launch site activity density, indicating the number of launches from VAFB SLC-4E.
- **Green and Red Markers:** Indicate successful (green) and unsuccessful (red) launches at this site.
- **Distance Markers (KM Labels):** Show measured distances between the launch pad and nearby infrastructure, such as railways, highways, and coastlines.
- **Geographical Context:** The site is near the Vandenberg State Marine Reserve, with Lompoc Airport and nearby transport routes visible, emphasizing its strategic location for space launches.

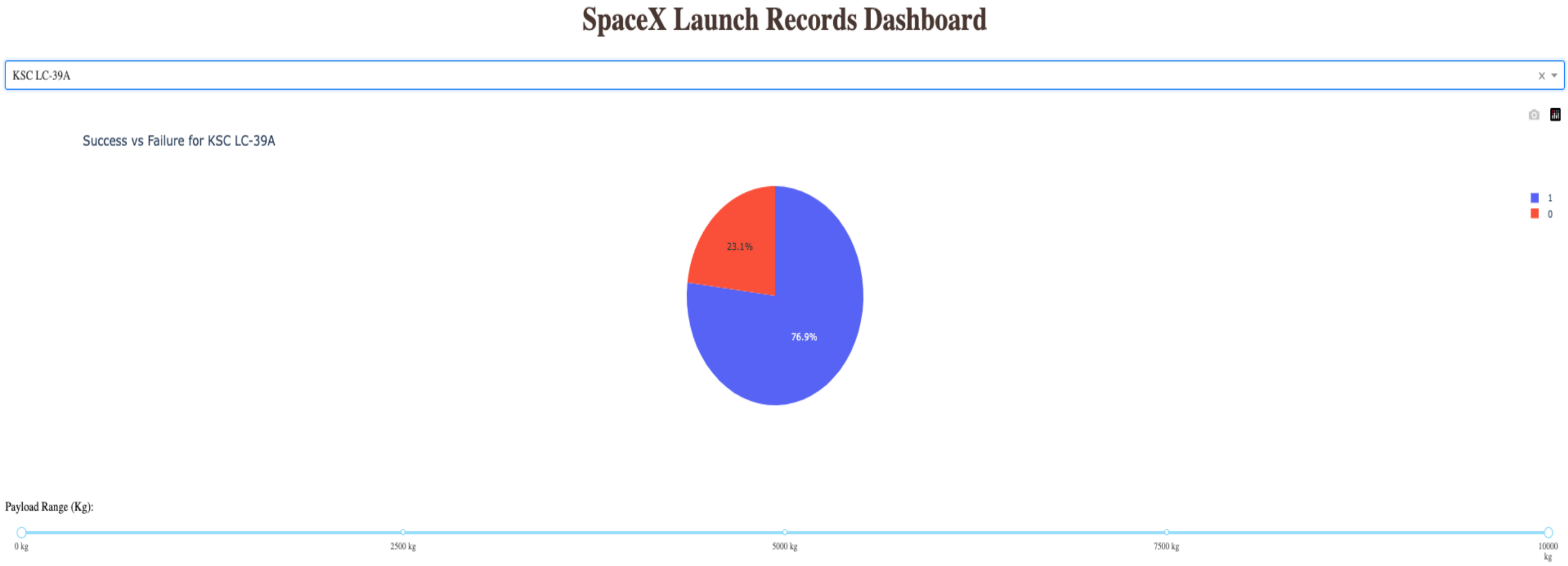

Dashboard with Plotly Dash

Plotly Dashboard
Launch Success Count for All Sites



Dashboard Summary: This interactive dashboard provides insights into SpaceX's successful launches across different launch sites. This dashboard helps users explore launch success rates across sites and payload variations, providing a clear visualization of SpaceX's mission outcomes. **Key Features:** *Dropdown Menu (Top):* Allows users to filter launch data by specific launch sites or view all sites combined. *Pie Chart (Center):* Displays the distribution of successful launches among the four primary SpaceX launch sites: *KSC LC-39A (Blue)CCAFS LC-40 (Red)VAFB SLC-4E (Green)CCAFS SLC-40 (Purple).*

Piechart for the Launch Site with Highest Launch Success Ratio



This dashboard view focuses specifically on launches from the Kennedy Space Center Launch Complex 39A (KSC LC-39A), providing insights into success vs. failure rates for this site. The majority of launches from KSC LC-39A have been successful (over 75% success rate), making it one of SpaceX's most reliable launch sites.

Key Features:

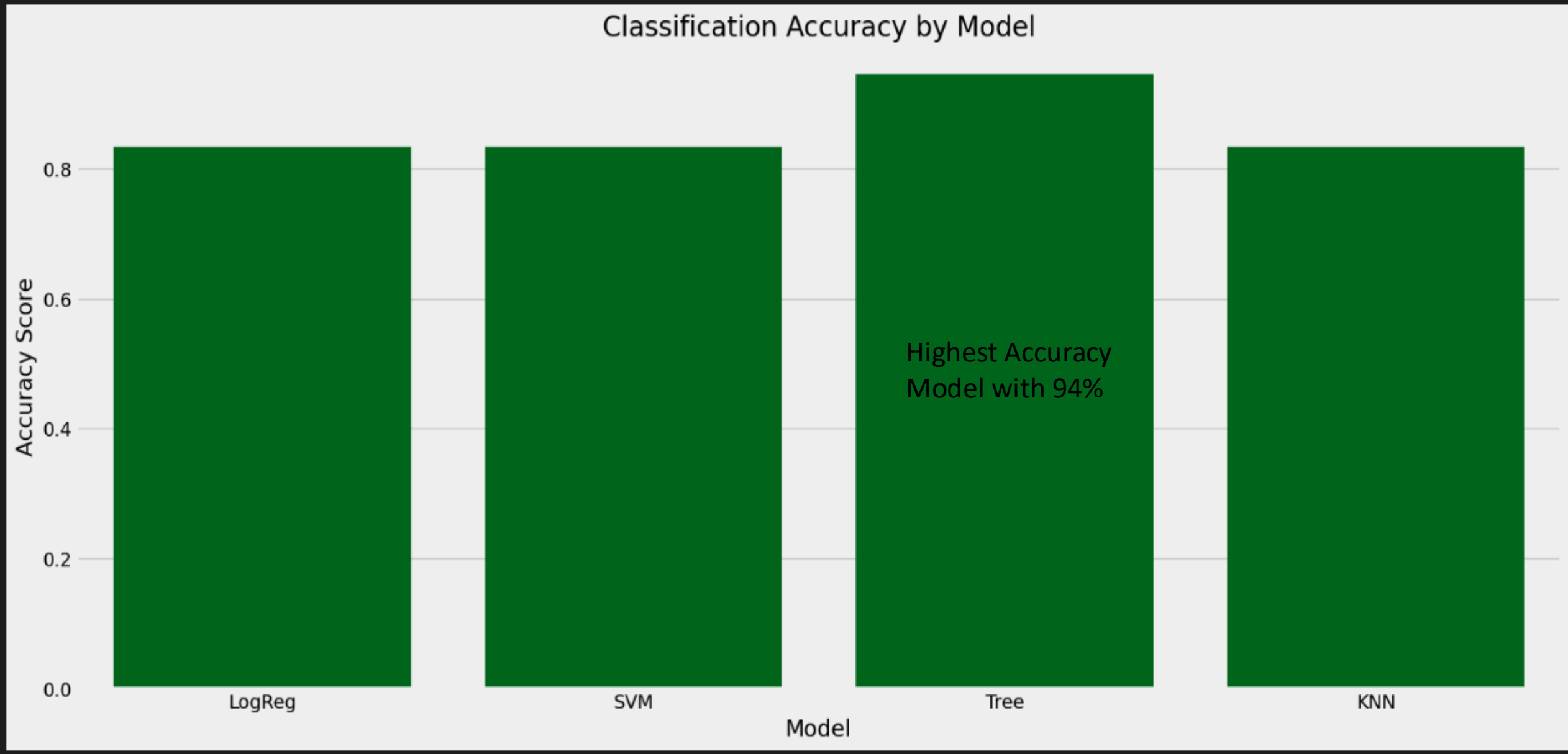
- **Dropdown Menu (Top):** The user has selected KSC LC-39A, filtering the data to show only launches from this site.
- **Pie Chart (Center):** Displays the proportion of successful vs. failed launches at KSC LC-39A:
- **76.9%** success rate (Blue)
- 23.1% failure rate (Red)
- **Payload Range Slider (Bottom):** Available for adjusting the payload mass range, allowing further exploration of how payload affects launch success.

Varied Payloads vs. Launch Outcome Scatter Plot for All Sites

This visualization consists of four scatter plots, each analyzing the relationship between payload mass (kg) and launch success (Class = 1) or failure (Class = 0) across all launch sites. The payload range varies across 2,500 kg, 5,000 kg, 7,500 kg, and 10,000 kg, showing how different payload capacities impact launch success.

Predictive Analysis (Classification)

	Model	Accuracy Score
0	LogReg	0.83
1	SVM	0.83
2	Tree	0.94
3	KNN	0.83



Tree Model Confusion Matrix

Key Metrics Derived from the Confusion Matrix

1. True Positives (TP): 12
• These are the correctly predicted successful landings.

2. True Negatives (TN): 5
• These are the correctly predicted failed landings.

3. False Positives (FP): 1
• This represents a misclassification where the model predicted the rocket would land, but it actually didn't.

4. False Negatives (FN): 0
• The model did not incorrectly predict a failed landing when it actually landed successfully.

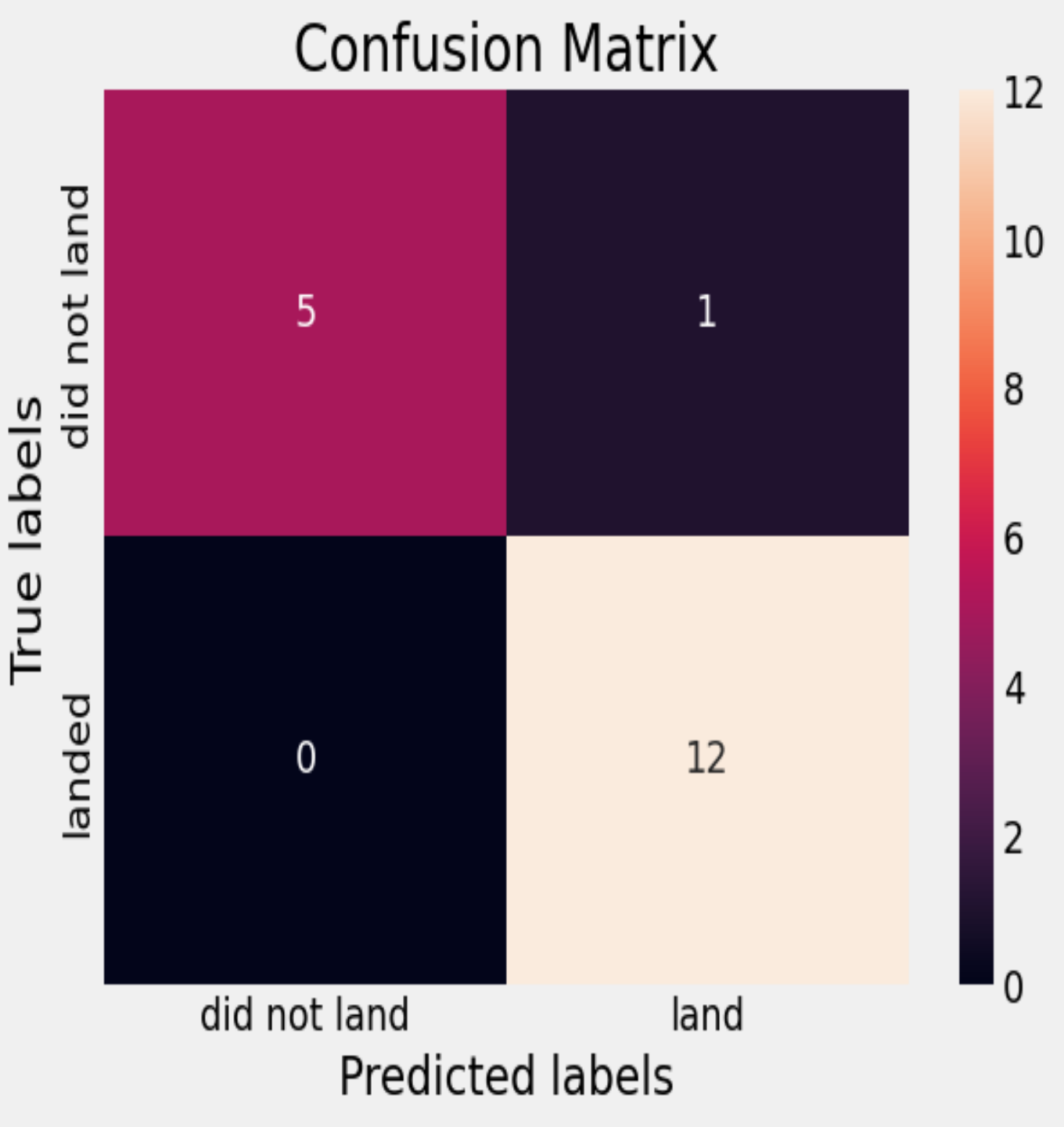
Key Takeaways
• The model performs very well, with a high accuracy of 94.4%.

• No false negatives, meaning all actual successful landings were correctly classified.

• The one misclassification (false positive) suggests that the model may slightly overpredict successful landings.

• Precision (92.3%) and Recall (100%) indicate that the model is highly effective at predicting landings, though it sometimes misclassifies failures as successes.

This confusion matrix confirms that the classification model is highly reliable, making it useful for predicting future rocket landing outcomes with strong confidence.



Conclusions

Top 4 Takeaways from the SpaceX Launch Analysis and Classification Model Project

1. Launch Success Rates Vary by Site and Orbit Type

- The CCAFS LC-40 and KSC LC-39A launch sites had the highest number of successful launches.
- Orbits like GEO, SSO, and HEO showed near 100% success rates, whereas GTO and ISS had lower reliability.

2. Payload Mass Influences Success, but Booster Version Matters More

- Heavier payloads tend to reduce launch success, particularly beyond 10,000 kg.
- However, newer booster versions (B4 & B5) significantly improve success rates, even for high-payload missions.

3. The Best Classification Model for Predicting Launch Success Was Logistic Regression

- Among models tested (Logistic Regression, SVM, Decision Trees, and KNN), Logistic Regression had the highest accuracy and best overall performance.
- Hyperparameter tuning using GridSearchCV improved model accuracy, making predictions more reliable.

4. SpaceX's Launch Success Has Increased Over Time

- Early launches (pre-2015) had lower success rates, but post-2017, success rates exceeded 80%, reflecting improved technology and operational efficiency.
- The trend suggests continued growth in reliability, making SpaceX a leader in commercial space travel.

These findings highlight key factors affecting launch success, enabling data-driven improvements in future missions.

Appendix

```
import pandas as pd
import numpy as np
import sqlite3
from prettytable import PrettyTable
import csv

con = sqlite3.connect('my_data1.db')
cur = con.cursor()


# Drop the table if it exists BEFORE importing the CSV
drop_table = '''DROP TABLE IF EXISTS SPACEXTBL;'"
cur.execute(drop_table)


# Load data from CSV
file = 'IBM Data Certificate/Course11_Applied_Data_Science_Capstone/Spacex.csv'
df = pd.read_csv(file)


# Write data to the table
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False, method='multi')


# Create the new table filtering non-null dates
table = '''SELECT * FROM SPACEXTBL WHERE Date IS NOT NULL;'''
cur.execute(table)
```

Code snippet for creating a databse and a table with sqlite in Python. Very useful.


Thank You!