Winning Space Race with Data Science

Christopher Kettell
2025-01-14

Outline

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

Executive Summary

This presentation sets out the methodology for a comprehensive data science workflow, from raw data collection through predictive modelling, integrating both statistical analysis and interactive visualisation techniques to understand the Falcon 9 and SpaceX launch success factors.

Data Collection & Analysis Process:

Data Sources: Combination of SpaceX API and web scraping
Processing Steps:
Data wrangling and cleaning
SQL-based exploration with visualization
Interactive mapping and dashboard creation
Machine learning classification modelling

Results Overview:

EDA Findings: Key patterns identified through exploratory data analysis
Visual Analytics: Insights derived from interactive maps and dashboards
Predictive Modelling:
Evaluation of various classification models
Selection of optimal model for launch success prediction

Introduction

Project Background and Context

In this capstone project, we present the Data Science approach, tools and methodology in the used to ultimately complete an analysis of SpaceX open data to answer the question of “will a Falcon 9 first stage land successfully?”

Why does this matter?

Unlike other space missions, the SpaceX approach enables the reuse of the first stage rocket in lifting a payload, and then returning to a landing pad, rather than falling into the sea (mostly).

Based on available data, a Falcon 9 rocket launch costs of around 62 million dollars; from other providers the launch cost is upward of 165 million dollars

Therefore if we can determine if the first stage will land, we can determine the cost of a launch.

It is also some pretty incredible engineering in motion

What is a Falcon 9?

Falcon 9 is a partially reusable, human-rated, two-stage-to-orbit, medium-lift launch vehicle[a] designed and manufactured in the United States by SpaceX.

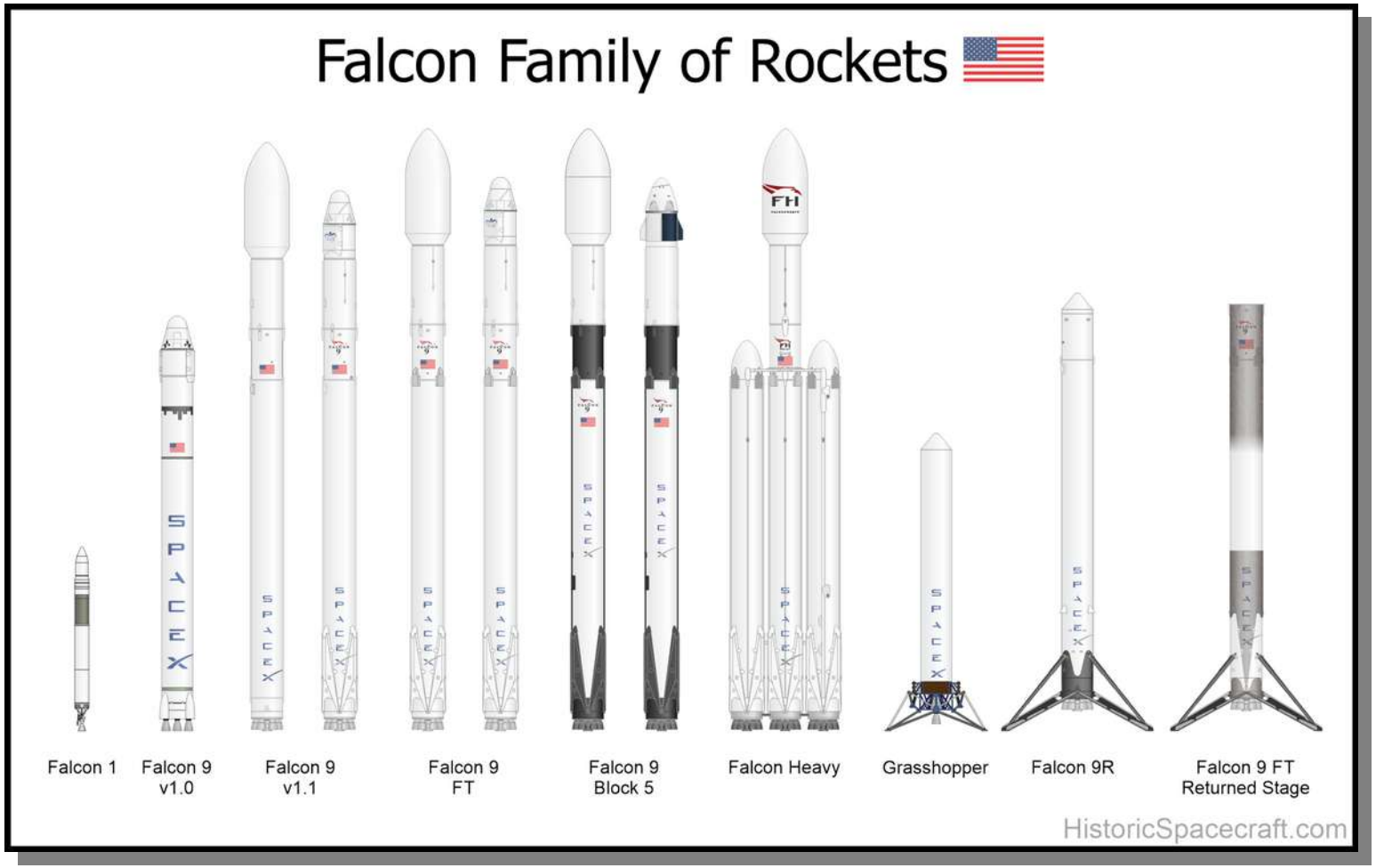
The first Falcon 9 launch was on 4 June 2010, and the first commercial resupply mission to the International Space Station (ISS) launched on 8 October 2012.[1]

In 2020, it became the first commercial rocket to launch humans to orbit.[2]

The Falcon 9 has an exceptional safety record,[3][4][5] with:

- 425 successful launches,
- two in-flight failures,
- one partial failure and
- one pre-flight destruction.

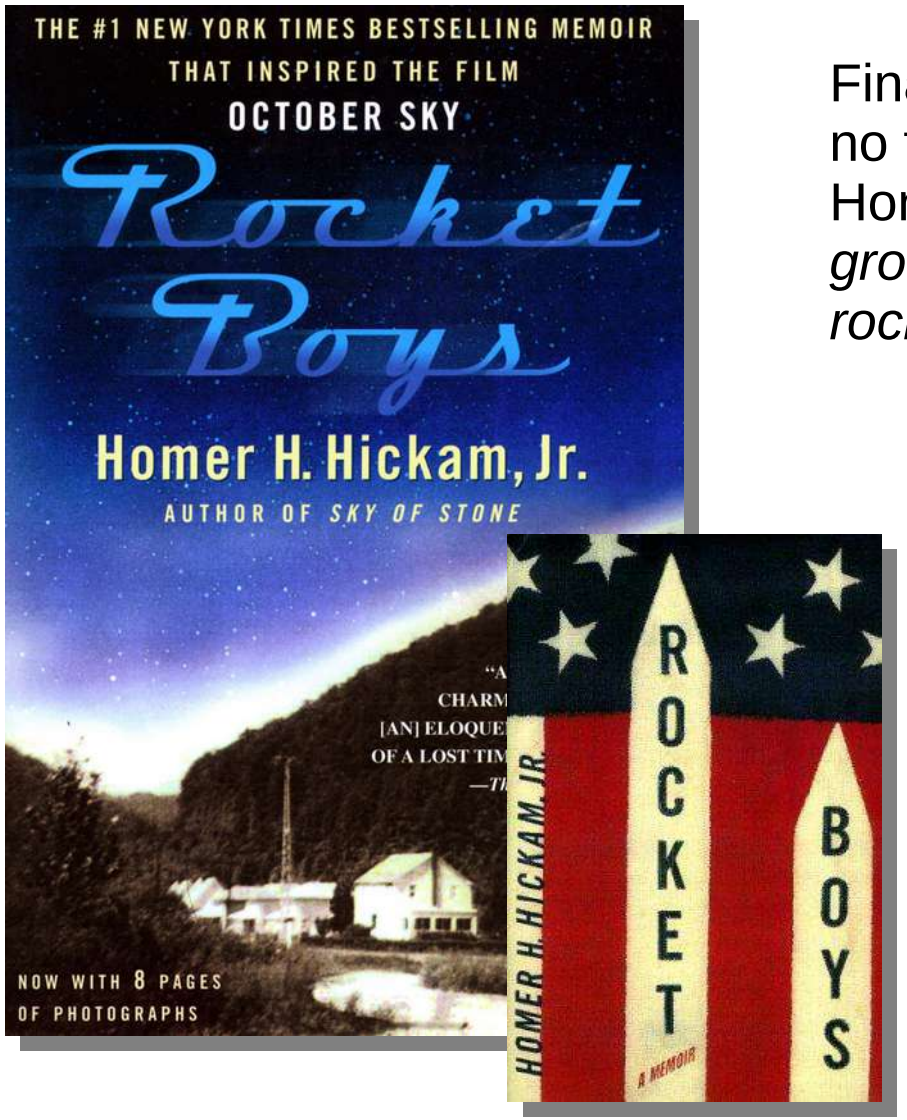
It is the most-launched American orbital rocket in history.



https://historicspacecraft.com/Rockets_SpaceX.html

[1] Amos, Jonathan (8 October 2012). "SpaceX lifts off with ISS cargo". BBC News. Archived from the original on 20 November 2018. Retrieved 3 June 2018. [2] "NASA and SpaceX launch astronauts into new era of private spaceflight". 30 May 2020. Archived from the original on 12 December 2020. Retrieved 8 December 2020. [3] Berger, Eric (3 February 2022). "The Falcon 9 may now be the safest rocket ever launched". Ars Technica. Archived from the original on 25 April 2023. Retrieved 21 May 2023. [4] "The Download: Falcon 9's future, and Big Tech's climate goals". 18 July 2024. Archived from the original on 19 August 2024. Retrieved 19 August 2024. [5]"SpaceX rocket failure highlights need for multiple launch options: 'Falcon 9 is not invulnerable'". 25 July 2024. Archived from the original on 19 August 2024. Retrieved 19 August 2024.

Further Reading



Finally, if you would care for a good read recommendation, look no further than The Rocket Boys, an autobiographical memoir by Homer Hickham. As described by Wikipedia best ...*"It is a story of growing up in a mining town, and a boy's pursuit of amateur rocketry in a coal mining town."*

- Its childhood excitement and discovery of Rockets
- Its meeting Wener Van Braun
- Its NASA
- Its space

- **Its a wonderful read.**

https://homerhickam.com/project/rocket-boys/

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

Perform data wrangling

Perform exploratory data analysis (EDA) using visualization and SQL

Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

Data Collection

The web scraping process began with setting up essential libraries, which utilised the requests library for HTTP communication, whilst employing BeautifulSoup from bs4 for HTML parsing, alongside pandas for data structuring and export purposes.

Data retrieval was performed using an HTTP GET request to the Falcon 9 Wikipedia page from which a BeautifulSoup object from the HTML response was first created and then parsed using the 'html.parser'.

Table identification, tables were identified using the find_all('table') method, subsequently identifying the third table (index 2) as the target launch records table, from which column headers using the th elements were extracted

Data Collection

Data structure preparation involved creating an empty dictionary (launch_dict) with keys derived from column names. Irrelevant columns, were removed, then empty lists were initialised for each data categoryD

Data extraction involved iterating through each table row using the tr elements, whilst validating rows by checking for flight numbers. Data points were extracted using a few different methods including direct string extraction, anchor tag text retrieval, and custom parsing for complex fields.

Data cleaning and structuring, required cleaning of missing values and annotations, unnecessary whitespace and formatting, parsing of dates and times into separate columns, and processed special cases such as mass measurements.

Data Collection

The final data processing stage involved converting the dictionary to a pandas DataFrame and exporting the structured data to 'spacex_web_scraped.csv', whilst preserving the index=False setting for clean CSV output.

This methodical approach to extracting, cleaning, and structuring the data has resulted in a dataset that is thoroughly prepared for subsequent analysis.

Github link: https://github.com/christophekey/Coursea_Data_Science_Capstone

All files for the Capstone project are located in the above repo

Data Collection – Too Long Didn't Read Version

Initial Setup and Libraries:

- Utilised **requests** library for HTTP communication
- Employed **BeautifulSoup** from bs4 for HTML parsing
- Used **pandas** for data structuring and export

Data Retrieval:

- Performed HTTP GET request to the Falcon 9 Wikipedia page
- Created a BeautifulSoup object from the HTML response
- Parsed using 'html.parser' for compatibility

Table Identification:

- Located all tables using find_all('table') method
- Identified the third table (index 2) as the target launch records table
- Extracted column headers using <th> elements

Data Structure Preparation

- Created an empty dictionary (launch_dict) with keys from column names
- Removed irrelevant columns (e.g., 'Date and time ( )')
- Initialised empty lists for each data category

Github link: https://github.com/christophekey/Coursea_Data_Science_Capstone

All files for the Capstone project are located in the above repo

Data Extraction Process:

- Iterated through each table row (<tr> elements)
- Validated rows by checking if they contain flight numbers
- Extracted data points using various methods:

Direct string extraction

- Anchor tag text retrieval (a.string)
- Custom parsing for complex fields

Data Cleaning and Structuring:

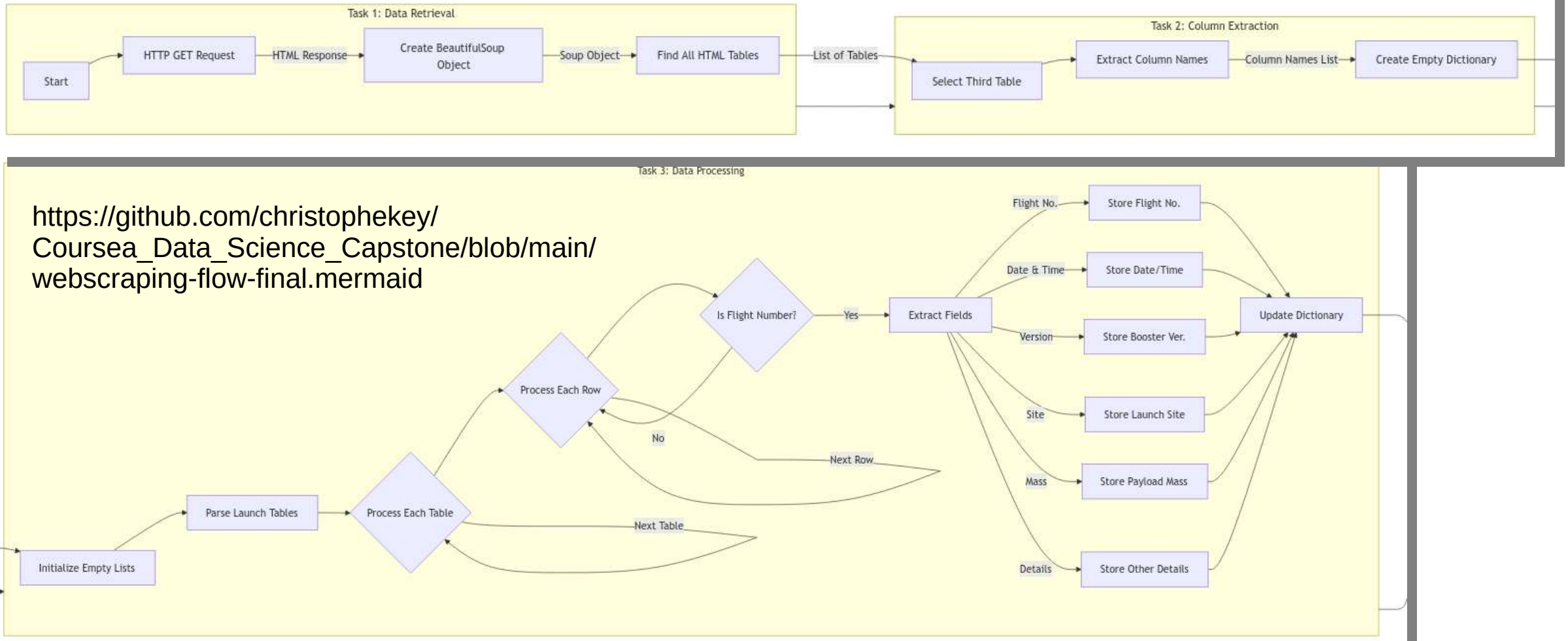
- Handled missing values and annotations
- Stripped unnecessary whitespace and formatting
- Parsed dates and times into separate columns
- Processed special cases like mass measurements

Final Data Processing:

- Converted the dictionary to a pandas DataFrame
- Exported the structured data to 'spacex_web_scraped.csv'
- Preserved index=False setting for clean CSV output

Data Collection – Flow Chart

https://github.com/christophekey/Coursea_Data_Science_Capstone/blob/main/webscraping-flow-final.mermaid



Data Collection – Falcon 9

2021

[4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

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

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

[7]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a objectsoup
soup = BeautifulSoup(response.text, 'html.parser')
```

Create a `BeautifulSoup` object from the HTML `response`

[8]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup
```

[8]:
```
<!DOCTYPE html>

<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-lang
eader-disabled vector-feature-p
age-tools-pinned-disabled vector-feature-toc-pinned-clientpref-1 vector-feature-main-menu-pinned-disabled vector-feature-limited-width-clientpref-1 vector-feature-limited-wid
th-content-enabled vector-feature-custom-font-size-clientpref-1 vector-feature-appearance-pinned-clientpref-1 vector-feature-night-mode-enabled skin-theme-clientpref-day vect
or-toc-available" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
<script>(function(){var className="client-js vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vector-feature-sticky-header-disab
led vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-clientpref-1 vector-feature-main-menu-pinned-disabled vector-feature-limited-width-clientpref-1 vecto
r-feature-limited-width-content-enabled vector-feature-custom-font-size-clientpref-1 vector-feature-appearance-pinned-clientpref-1 vector-feature-night-mode-enabled skin-them
e-clientpref-day vector-toc-available";var cookie=document.cookie.match(/(?:^|; )enwikimwclientpreferences=([^;]+)/);if(cookie){cookie[1].split('%2C').forEach(function(pref)
{className=className.replace(new RegExp('(^| )'+pref.replace(/-clientpref-\w+$|[^\w-]+/g,'')+'-clientpref-\\w+( |$)'),'$1'+pref+'$2');});}document.documentElement.className=c
lassName;}());RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy",
```

Source URL

requests.get

BS4 Object and Parser

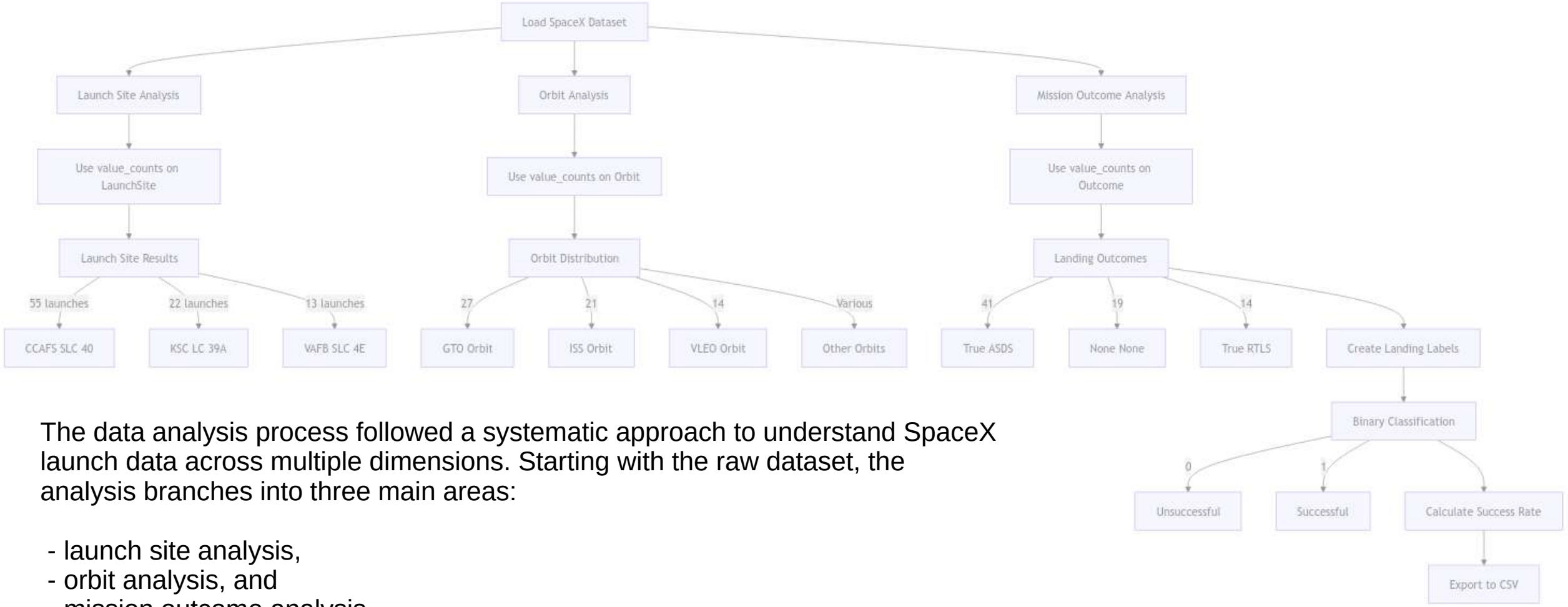
Html content

Data Wrangling

Describe how data were processed
You need to present your data wrangling process using key phrases and flowcharts
Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

Github link: https://github.com/christophekey/Coursea_Data_Science_Capstone/tree/main/visual_interactive_dashboard

Data Wrangling – Flow Chart



The data analysis process followed a systematic approach to understand SpaceX launch data across multiple dimensions. Starting with the raw dataset, the analysis branches into three main areas:

- launch site analysis,
- orbit analysis, and
- mission outcome analysis.

Each step uses the value_counts() method to aggregate and summarize data, culminating in a comprehensive view of SpaceX's launch operations and success rates.

EDA with Data Visualization

Key Visualisation Methods Used:

Scatter Plots: Examined relationships between:

Payload mass and flight number
Flight number and launch site
Payload and launch site
Flight number and orbit type
Payload and orbit type

Bar Charts: Visualized success rates across different orbit types

Line Plot: Tracked yearly trends in launch success rates over time

These visualizations helped identify patterns and correlations between various launch parameters and mission outcomes.

Github link:
https://github.com/christophekey/Coursea_Data_Science_Capstone/blob/main/eda_insights/jupyter-labs-eda-dataviz-v2.ipynb

EDA with SQL

Github link: LINK

The following SQL queries were used to explore the data:

Query 1 - Display the names of the unique launch sites in the space mission (uses **DISTINCT**)

Query 2 - Display 5 records where launch sites begin with the string 'CCA' (using **WHERE** "Launch_Site" **LIKE** 'CCA%')

Query 3 - Display the total payload mass carried by boosters launched by NASA (CRS) (using **SUM** and **WHERE**)

Query 4 - Display average payload mass carried by booster version F9 v1.1 (Using **AVG** and **WHERE**)

Query 5 - List the date when the first successful landing outcome in ground pad was achieved(Using **MIN**)

Query 6 - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000 (Using **WHERE** + **AND**)

Query 7 - List the total number of successful and failure mission outcomes (Using **COUNT** and **GROUPBY**)

Query 8 - List the names of the booster_versions which have carried the maximum payload mass. (USING **WHERE** as a Subquery)

Query 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. (Using substr for month and year and then **WHERE** + **AND**)

Query 10 -Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order. (Using **WHERE**, **GROUP BY** and **ORDER BY**)

Build an Interactive Map with Folium

Key Folium Components used:

Circle: Highlighted NASA JSC's area as the map's centre point
Marker: Pinpointed specific launch locations
Marker Cluster: Grouped successful and failed launches by site
Mouse Position: Enabled coordinate tracking for measuring coastal distances
Line: Drew connecting lines to show distances between:

- Launch sites and coastlines
- Launch sites and transportation routes (highways)
- Launch sites and major urban areas
- Launch sites and major airport

These mapping tools helped visualise spatial relationships between launch facilities and key geographical features.

Github link:
https://github.com/christophekey/Coursea_Data_Science_Capstone/blob/main/eda_insights/jupyter-labs-eda-dataviz-v2.ipynb

Build a Dashboard with Plotly Dash

Interactive Dashboard Elements:

Site Selector: Drop-down menu for launch site selection

Success Rate Visualization: Pie chart displaying success rates for selected launch sites

Payload Controls: Range slider for payload mass selection

Correlation Display: Scatter plot showing relationship between:

- Payload mass
- Launch success
- Selected launch sites

The interactive dashboard enables the exploration of relationships between launch sites, payload mass, and mission success rates through dynamic visualisation tools.

Github link:
https://github.com/christophekey/Coursea_Data_Science_Capstone/blob/main/eda_insights/jupyter-labs-eda-dataviz-v2.ipynb

Predictive Analysis (Classification)

Model Building & Evaluation Steps:

Data Preparation:

- Split data into features and target variables
- Normalized feature columns
- Converted target to NumPy array
- Created training and test sets

Model Optimization:

- Used GridSearchCV to:
  - Determine optimal parameters
  - Find best performance scores

Model Assessment:

- Calculated test set accuracy
- Visualized results through confusion matrix

Github link:
https://github.com/christophekey/Coursea_Data_Science_Capstone/blob/main/predictive_analytics/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

Results

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

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

Three Launch Sites:

- **CCAFS SLC 40** (Cape Canaveral Air Force Station Space Launch Complex 40)
- **VAFB SLC 4E** (Vandenberg Air Force Base Space Launch Complex 4E)
- K**SC LC 39A** (Kennedy Space Center Launch Complex 39A)

CCAFS SLC 40 appears to be the most frequently used site, with launches spread consistently across the entire range of flight numbers from 0 to ~90

VAFB SLC 4E shows much more sporadic usage, with only occasional launches scattered throughout the timeline

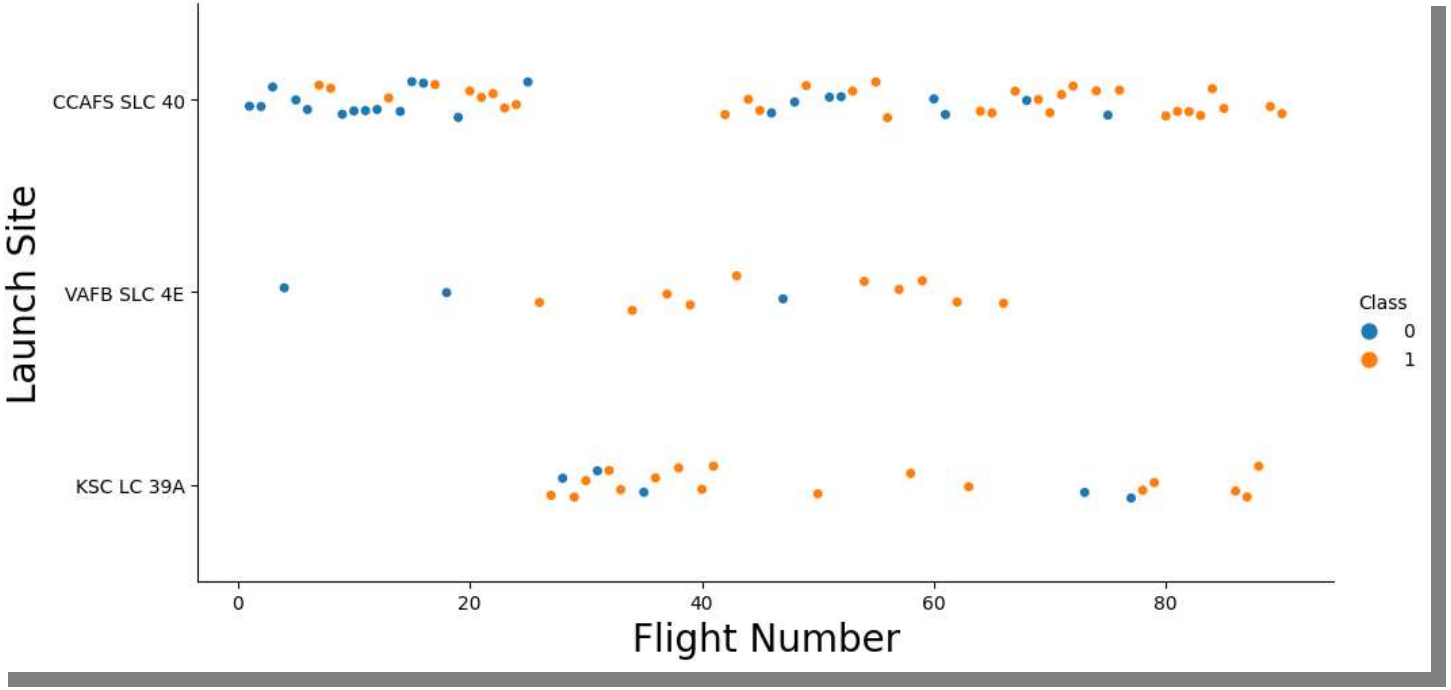
KSC LC 39A seems to have come into use later, starting around flight number 25, and shows regular but less frequent usage compared to CCAFS SLC 40

CCAFS SLC 40 shows the most even distribution between classes

First flights numbers 0-20 primarily used CCAFS SLC 40

Launch site diversity increased after flight number 20, with more regular use of KSC LC 39A

The use of VAFB SLC 4E maintains relatively consistent but sparse usage throughout the timeline



Key Finding: The pattern suggests a gradual expansion and diversification of launch operations across multiple sites, while maintaining CCAFS SLC 40 as the primary launch facility.

```
# 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
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 2)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

Payload vs. Launch Site

CCAFS SLC 40 appears most versatile, capable of handling the widest range of payload masses

KSC LC 39A seems specialized for medium to heavy payloads

VAFB SLC 4E appears to be used specifically for lighter payloads

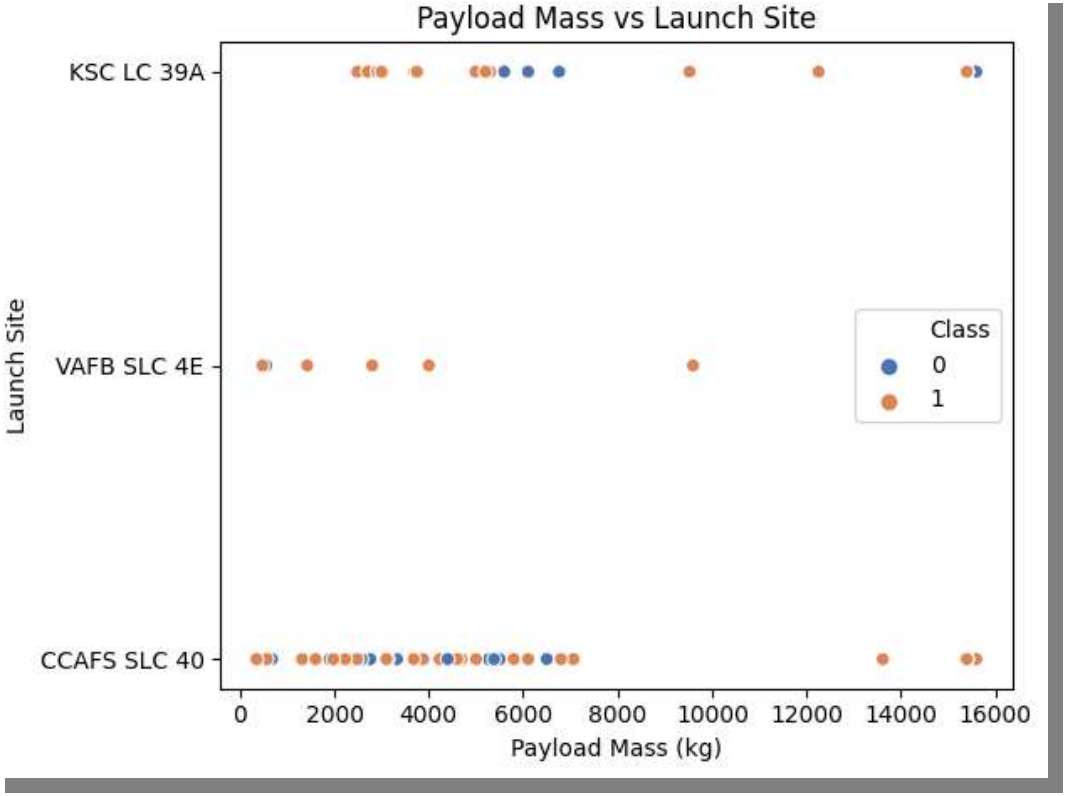
No clear correlation between payload mass and success/failure rates is immediately apparent with both heavy and light payloads showing a mix of outcomes

Most launches across all sites fall within the 2,000-8,000 kg range

There are relatively few very heavy payloads (>10,000 kg)

Very light payloads (<1,000 kg) are also uncommon

Key Finding: each launch site has somewhat specialized capabilities and preferred payload mass ranges, with CCAFS SLC 40 serving as the most versatile facility.




```
# 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 be the class value
sns.scatterplot(data=df, x='PayloadMass', y='LaunchSite', hue='Class', palette='deep')
plt.title('Payload Mass vs Launch Site')
plt.xlabel('Payload Mass (kg)')
plt.ylabel('Launch Site')

plt.tight_layout()
plt.show()
```

Success Rate vs. Orbit Type

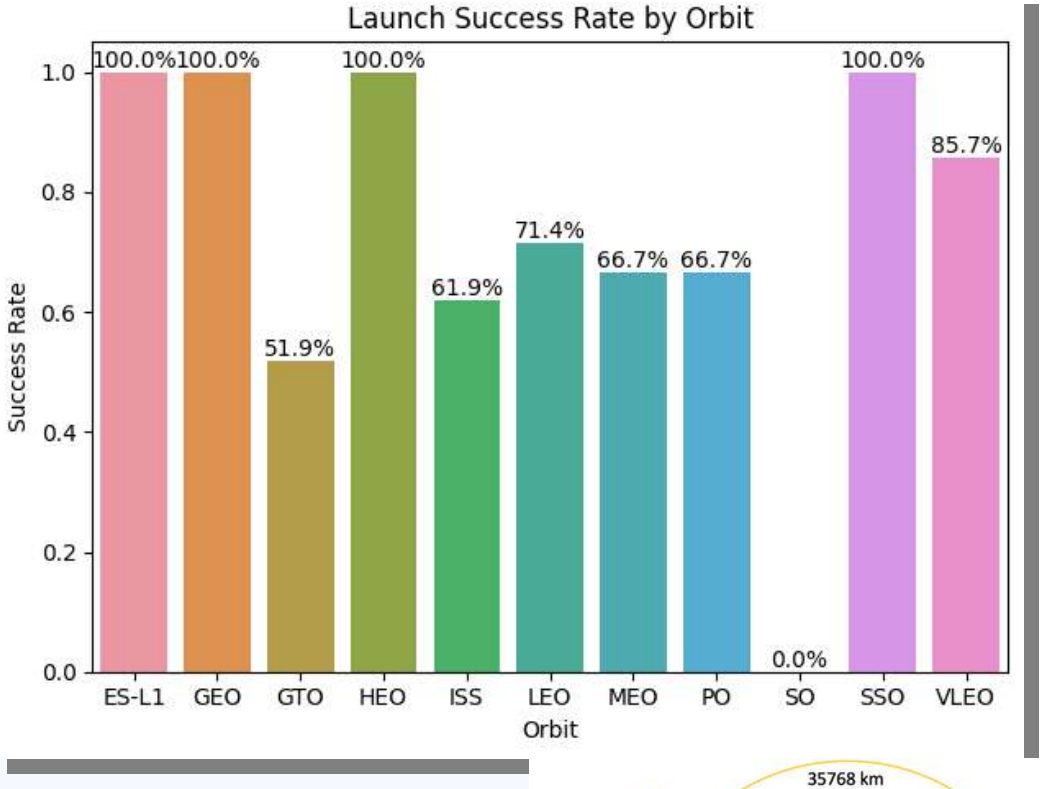
Missions to specialized orbits (ES-L1, SSO) tend to have higher success rates

Traditional GEO missions are very reliable

Transfer orbits like GTO are more challenging, with lower success rates

The majority of routine operational orbits (LEO, MEO, PO) maintain reasonable success rates above 65%

The very poor performance in SO launches might warrant investigation or could be due to a small sample size

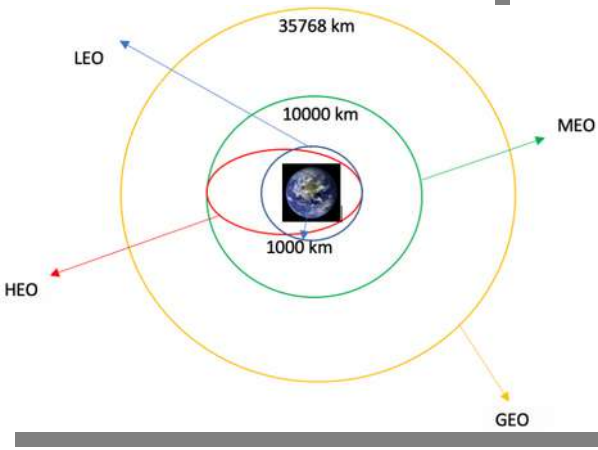



```
# HINT use groupby method on Orbit column and get the mean of Class column
success_rate = df.groupby('Orbit')['Class'].mean()

sns.barplot(x=success_rate.index, y=success_rate.values)
plt.title('Launch Success Rate by Orbit')
plt.xlabel('Orbit')
plt.ylabel('Success Rate')

for i, v in enumerate(success_rate.values):
    plt.text(i, v, f'{v:.1%}', ha='center', va='bottom')

plt.tight_layout()
plt.show()
```



Flight Number vs. Orbit Type

Early flights (0-20) focused primarily on LEO, ISS, and GTO missions

Mid-range flights (20-60) show increased diversity in orbit types

Later flights (60-90) show heavy use of VLEO and continued GTO missions

GTO (Geosynchronous Transfer Orbit): Consistent presence throughout all flight numbers, with a mix of successes and failures

ISS: Regular missions spread across the timeline, suggesting ongoing space station support

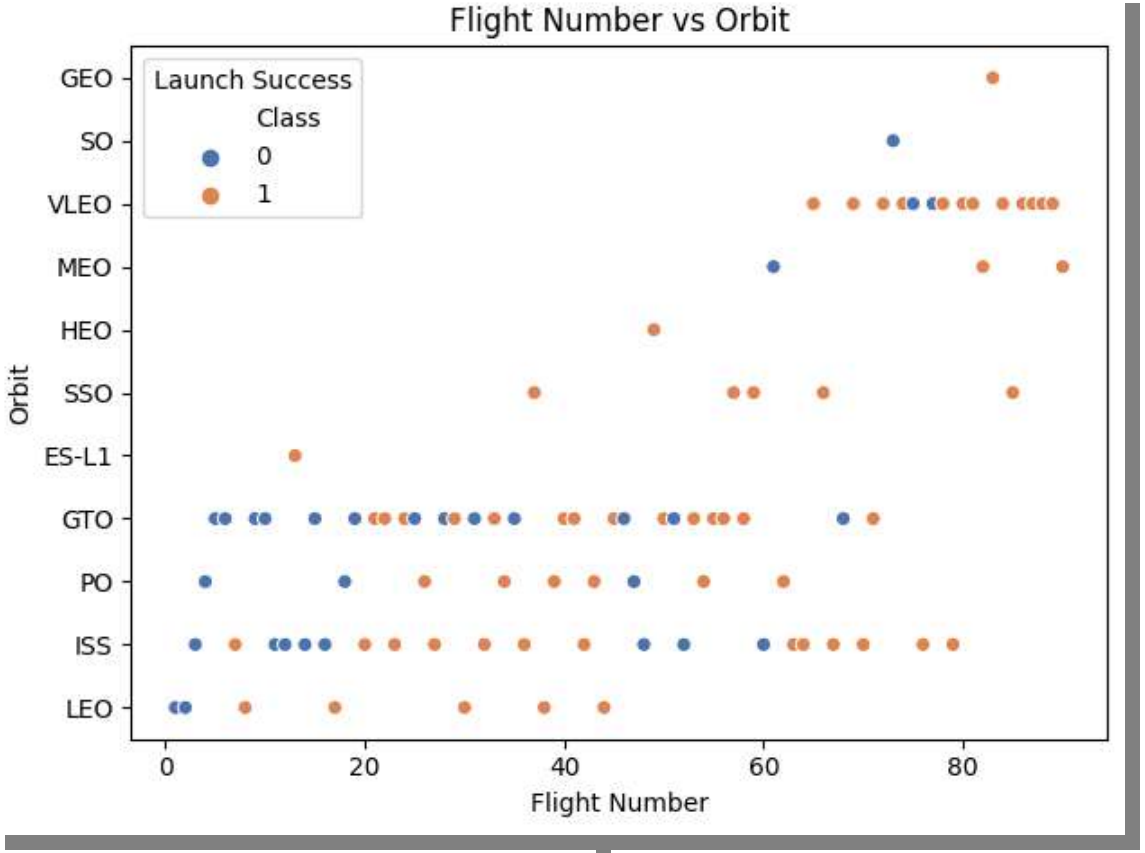
VLEO: Becomes very prominent in later flights (after flight #70), with mostly successful missions

LEO: More common in earlier flights, less frequent in later missions

SSO: Sporadic missions throughout, mostly successful

ES-L1: Single early mission (around flight #15)

GEO: Rare, with only a couple of missions late in the sequence




```
# 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.scatterplot(data=df, x='FlightNumber', y='Orbit', hue='Class', palette='deep')

plt.title('Flight Number vs Orbit')
plt.xlabel('Flight Number')
plt.ylabel('Orbit')
plt.legend(title='Launch Success')

plt.tight_layout()
plt.show()
[13]
```

Key Finding: A clear evolution from simpler, lower orbit missions to more diverse and technically challenging missions can be over time, with improving success rates.

Payload vs. Orbit Type

Payload Mass Distribution by Orbit:

GTO (Geosynchronous Transfer Orbit): Clustered around 3,000-8,000 kg
ISS missions: Mostly concentrated around 2,000-4,000 kg
VLEO (Very Low Earth Orbit): Heavier payloads, around 15,000 kg
LEO (Low Earth Orbit): Lighter payloads, generally under 5,000 kg
SSO (Sun-Synchronous Orbit): Relatively light payloads, around 2,000-4,000 kg
MEO (Medium Earth Orbit): Mid-range payloads around 4,000 kg
GEO (Geostationary Earth Orbit): One heavy payload around 6,000 kg

Heavier payloads (>10,000 kg) tend to show more successful launches

GTO missions show the most mixed success rate across different payload masses

Lighter payloads (<3,000 kg) show generally good success rates

Key Finding: Each orbit type seems to have a preferred payload mass range
Higher orbits (GEO, GTO) tend toward medium-weight payloads
Space station missions (ISS) cluster in a specific weight range
VLEO missions handle some of the heaviest payloads

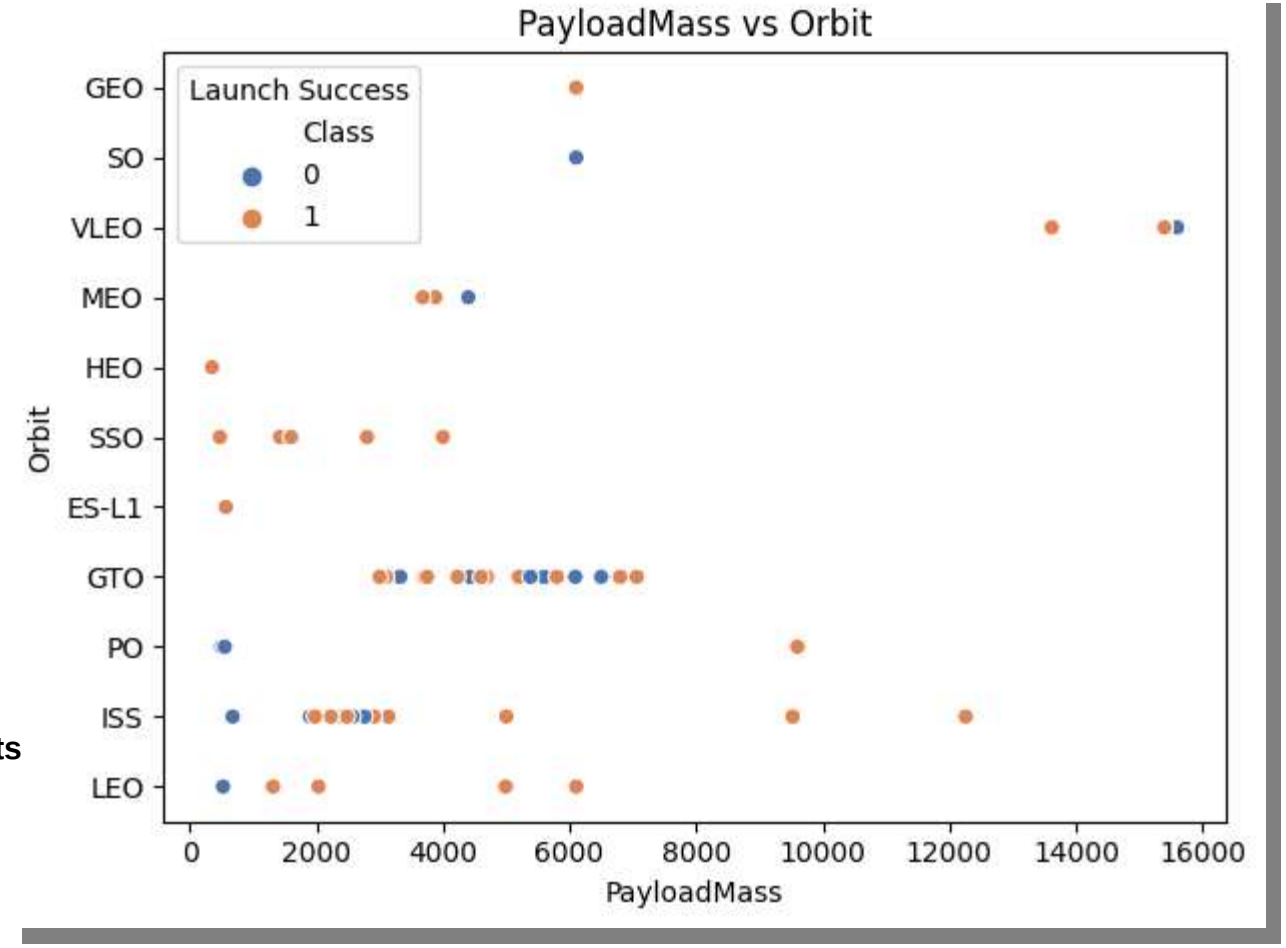
This suggests that different orbits have specific payload mass requirements and limitations

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit,

sns.scatterplot(data=df, x='PayloadMass', y='Orbit', hue='Class', palette='deep')

plt.title('PayloadMass vs Orbit')
plt.xlabel('PayloadMass')
plt.ylabel('Orbit')
plt.legend(title='Launch Success')

plt.tight_layout()
plt.show()
```



Launch Success Yearly Trend

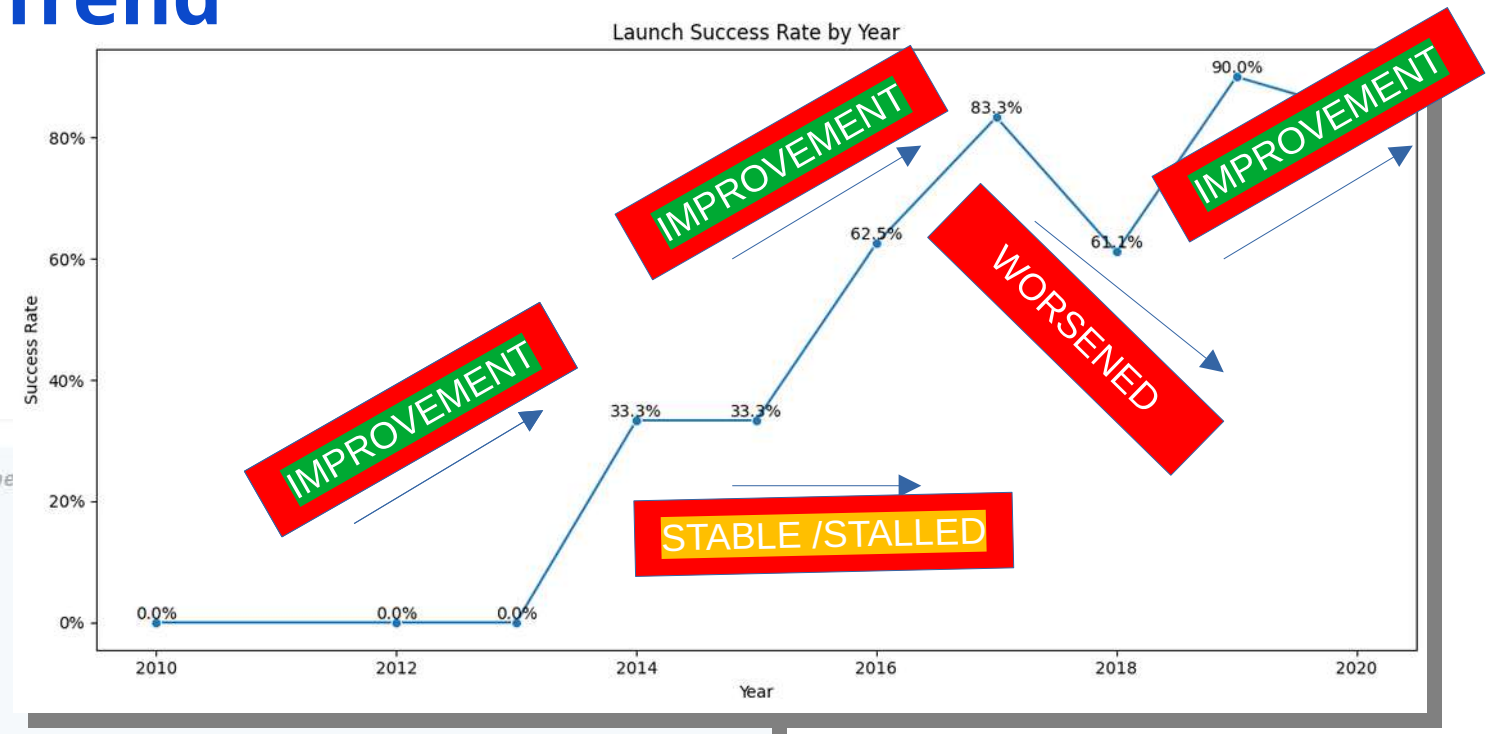
Clear upward trend in success rates over the decade

Most dramatic improvements occurred between 2015-2017

Maintained consistently high success rates (>80%) in recent years

Shows maturation of technology and operations over time

This pattern suggests SpaceX went through initial struggles, learned from failures, and successfully developed into a reliable launch provider with consistently high success rates by the end of the decade.




```
# Plot a line chart with x axis to be the extracted year and y axis to be

def extract_year(df):
    return pd.to_datetime(df['Date']).dt.year

years = extract_year(df)
yearly_success = df.groupby(years)['Class'].mean()

plt.figure(figsize=(12, 6))
sns.lineplot(x=yearly_success.index, y=yearly_success.values, marker='o')

plt.title('Launch Success Rate by Year')
plt.xlabel('Year')
plt.ylabel('Success Rate')

for x, y in zip(yearly_success.index, yearly_success.values):
    plt.text(x, y, f'{y:.1%}', ha='center', va='bottom')

plt.gca().yaxis.set_major_formatter(plt.FuncFormatter(lambda y, _: '{:.0%}'.format(y)))


plt.tight_layout()
plt.show()
```

All Launch Site Names

Task 1

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

```
cursor = con.cursor()
cursor.execute('SELECT DISTINCT "Launch_Site" FROM SPACEXTBL')
unique_sites = cursor.fetchall()
unique_sites
```

[23]

```
[('CCAFS LC-40',), ('VAFB SLC-4E',), ('KSC LC-39A',), ('CCAFS SLC-40',)]
```

cursor = con.cursor() - Creates a database cursor object to interact with the database
cursor.execute('SELECT DISTINCT "Launch_Site" FROM SPACEXTBL') - Executes an SQL query that:

Uses SELECT DISTINCT to get only unique values
Retrieves data from the "Launch_Site" column
From a table named "SPACEXTBL"

unique_sites = cursor.fetchall() - Fetches all the results from the query
The output shows the launch sites

Launch Site Names Begin with 'CCA'

Task 2

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

```
query = '''
SELECT *
FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5
'''

cca_sites = pd.read_sql(query, con)
cca_sites
```

[24]

SELECT * - Retrieves all columns from the table

FROM SPACEXTBL - Specifies the source table

WHERE "Launch_Site" LIKE 'CCA%' - Filters for launch sites that start with 'CCA'

The % is a wildcard character that matches any characters after 'CCA'

LIMIT 5 - Restricts the output to only 5 records

5 rows × 5 rows × 10 cols — Static Output

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	M
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	S
1	2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	S
2	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	S
3	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	S
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	S

The query is executed using pandas' read_sql function to load the results directly into a DataFrame for further analysis.

Total Payload Mass

Task 3

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

```
query = '''
SELECT SUM("PAYLOAD_MASS__KG_") as total_mass
FROM SPACEXTBL
WHERE Customer LIKE '%NASA (CRS)%'
'''

nasa_crs_payload = pd.read_sql(query, con)
nasa_crs_payload
```

[26]

1 row · 1 rows × 1 cols

	total_mass
0	48213

SELECT SUM("PAYLOAD_MASS__KG_") as total_mass - Adds up all payload masses and names this sum 'total_mass'

FROM SPACEXTBL - Specifies the source table

WHERE Customer LIKE '%NASA (CRS)%' - Filters for missions where the customer field contains "NASA (CRS)"

The % wildcards allow for any characters before or after "NASA (CRS)"

The result shows that NASA CRS missions carried a total payload mass of 48,213 kg.

The query uses pandas' read_sql to load the result into a DataFrame.

Average Payload Mass by F9 v1.1

Task 4

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

```
query = '''
SELECT AVG("PAYLOAD_MASS__KG_") as average_mass
FROM SPACEXTBL
WHERE "Booster_Version" = 'F9 v1.1'
'''

ave_payload = pd.read_sql(query, con)
ave_payload
```

[27]

1 row · 1 rows × 1 cols

	average_mass
0	2928.4

SELECT AVG("PAYLOAD_MASS__KG_") as average_mass - Calculates the mean of payload masses and names it 'average_mass'

FROM SPACEXTBL - Specifies the source table

WHERE "Booster_Version" = 'F9 v1.1' - Filters for missions that used the F9 v1.1 booster version specifically

The result shows that missions using the F9 v1.1 booster carried an average payload mass of 2,928.4 kg.

The query uses pandas' read_sql to load the result into a DataFrame.

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
query = '''
SELECT MIN(Date) as first_success_date
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (ground pad)'
'''

first_landing = pd.read_sql(query, con)
first_landing
```

[28]

1 row · 1 rows × 1 cols

	first_success_date
0	2015-12-22

SELECT MIN(Date) as first_success_date - Finds the earliest date and labels it 'first_success_date'

FROM SPACEXTBL - Specifies the source table

WHERE "Landing_Outcome" = 'Success (ground pad)' - Filters for missions where the landing was successful specifically on a ground pad

The result shows that the first successful ground pad landing was achieved on **December 22, 2015.** This was a significant milestone in SpaceX's development of reusable rockets.

The query uses pandas' read_sql to load the result into a DataFrame.

Successful Drone Ship Landing with Payload between 4000 and 6000

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

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

booster_selection = pd.read_sql(query, con)
booster_selection
```

[30]

4 rows 4 rows × 1 cols

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

SELECT DISTINCT "Booster_Version" - Retrieves unique booster versions without duplicates

FROM SPACEXTBL - Specifies the source table

Three conditions are combined with AND:

WHERE "Landing_Outcome" = 'Success (drone ship)' - Only successful drone ship landings

AND "PAYLOAD_MASS__KG_" > 4000 - Payload must be greater than 4000 kg

AND "PAYLOAD_MASS__KG_" < 6000 - Payload must be less than 6000 kg

The results show four different Falcon 9 (F9) booster versions that met these criteria:

F9 FT B1022, F9 FT B1026, F9 FT B1021.2 and F9 FT B1031.2

The query uses pandas' read_sql to load the results into a DataFrame.

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
mission_outcome_query = '''
SELECT Mission_Outcome, COUNT(*) as count
FROM SPACEXTBL
GROUP BY Mission_Outcome
'''

mission_outcome = pd.read_sql(mission_outcome_query, con)
mission_outcome
```

[32]

4 rows × 2 cols

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

SELECT Mission_Outcome, COUNT(*) as count - Selects the mission outcome and counts occurrences

FROM SPACEXTBL - Specifies the source table

GROUP BY Mission_Outcome - Groups the results by different mission outcomes

The results show:

1 "Failure (in flight)" mission

98 "Success" missions

1 additional "Success" entry

1 "Success (payload status unclear)" mission

Total: 101 missions, with the vast majority (99) being successful in some form, and only one clear failure.

The query uses pandas' read_sql to load the results into a DataFrame, presenting a summary of mission success rates.

Boosters Carried Maximum Payload

Task 8

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

```
booster_versions_query = '''
SELECT DISTINCT Booster_Version
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL
)
'''

booster_versions = pd.read_sql(booster_versions_query, con)
booster_versions
```

```
booster_versions
[33]
```

12 rows · 12 rows × 1 cols

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

Main query:

SELECT DISTINCT Booster_Version - Gets unique booster versions

FROM SPACEXTBL - From the main table

Subquery (inside the WHERE clause):

SELECT MAX(PAYLOAD_MASS__KG_) - Finds the maximum payload mass

FROM SPACEXTBL - From the same table

This subquery returns a single value that represents the heaviest payload

The WHERE clause:

Matches the payload mass to the maximum value found in the subquery

This effectively filters for only those boosters that carried the heaviest payload

2015 Launch Records

Task 9

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

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

```
year2015query = '''
SELECT
    substr(Date, 6, 2) as month,
    Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM SPACEXTBL
WHERE substr(Date, 0, 5) = '2015'
AND Landing_Outcome LIKE 'Failure (drone ship)'
'''

year2015query = pd.read_sql(year2015query, con)
year2015query
```

[34]

2 rows × 4 cols

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

Extract the month from dates in 2015 using substr

Show the failed drone ship landings in 2015

Results show two failures these were:

January (01): F9 v1.1 B1012 at CCAFS LC-40

April (04): F9 v1.1 B1015 at CCAFS LC-40

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

Task 10

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

```
landing_outcomes_query = '''
SELECT Landing_Outcome, COUNT(*) as count
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY count DESC
'''

landing_outcomes_query = pd.read_sql(landing_outcomes_query, con)
landing_outcomes_query
```

[35]

8 rows × 8 rows × 2 cols

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

Counts landing outcomes between June 2010 and March 2017

Groups them by outcome type

Orders results by count in descending order

Results show that:

No attempt: 10 cases

Success/Failure (drone ship): 5 each

Success (ground pad)/Controlled (ocean): 3 each

Uncontrolled (ocean)/Failure (parachute): 2 each

Precluded (drone ship): 1 case (I was not sure what this meant)


Section 3
Launch Sites
Proximities Analysis

Launch Sites

```
37 # Display the map
38 site_map
[10]
```

Global Map

Zoom Controls

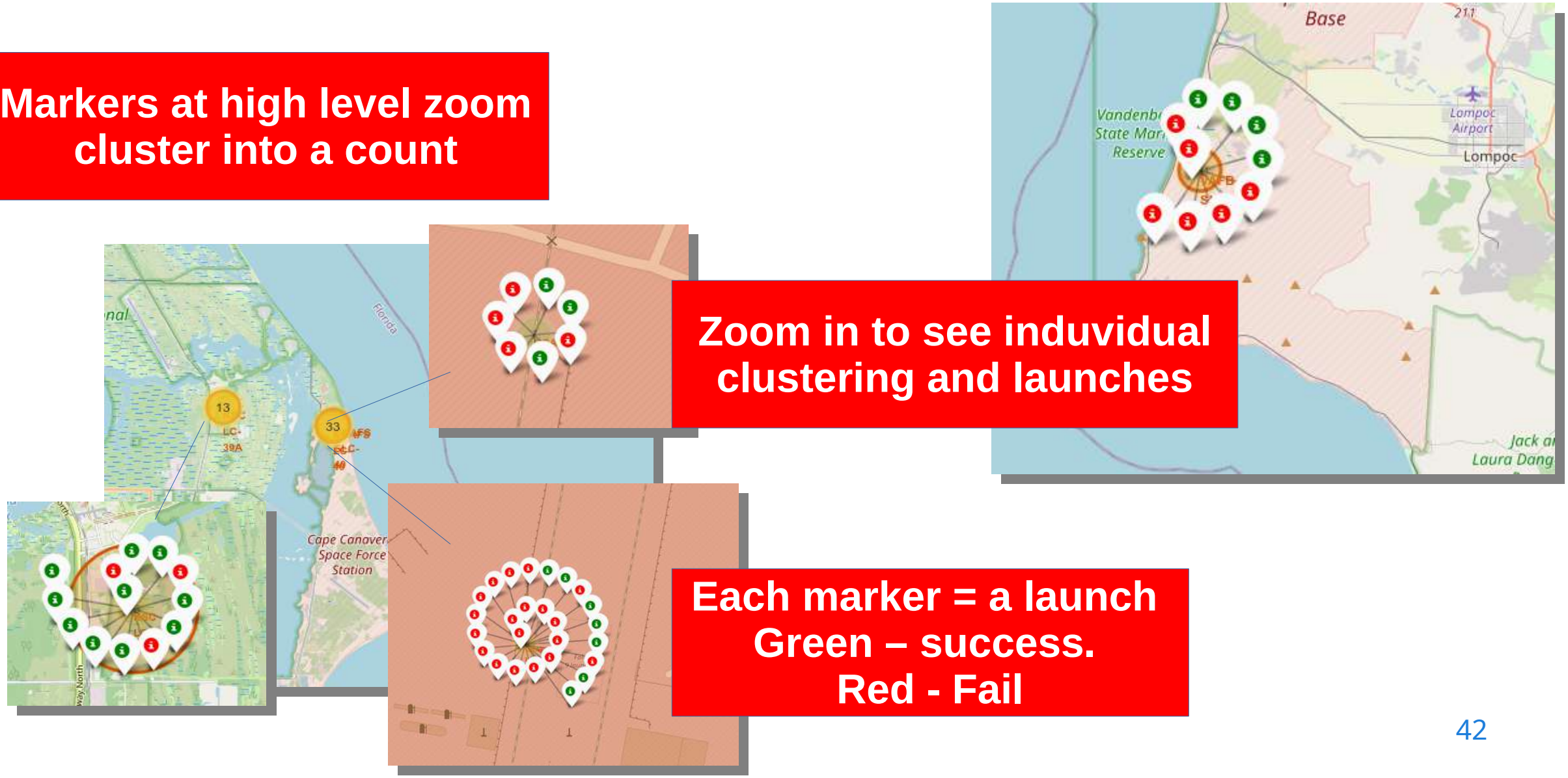
Map Pane (site_map)

Circle, Point and Label

US Sites Map

Markers, Clustered and Coloured

Markers at high level zoom cluster into a count

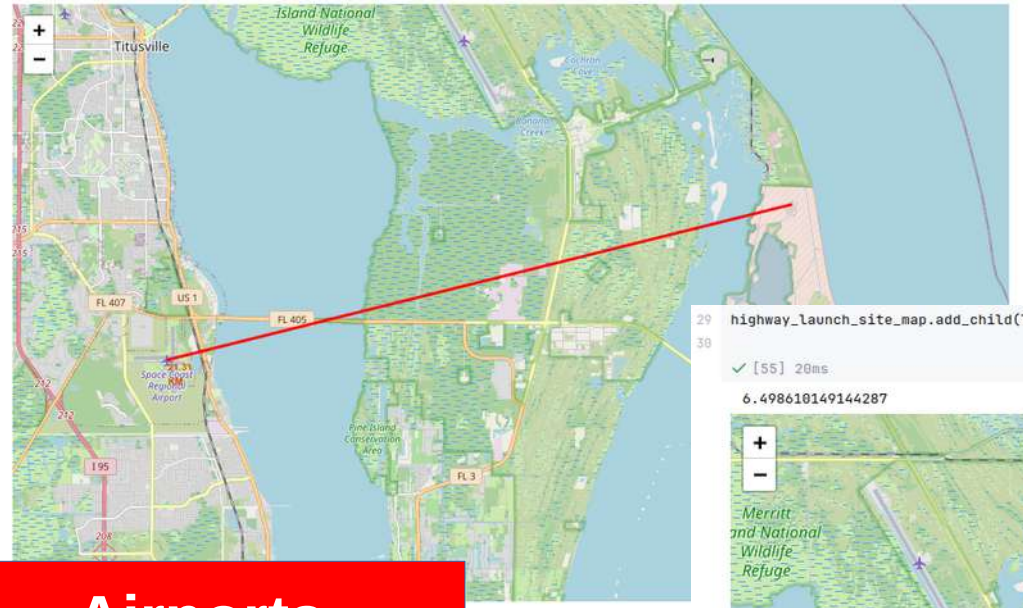


Zoom in to see induvidual clustering and launches

Each marker = a launch
Green – success.
Red - Fail

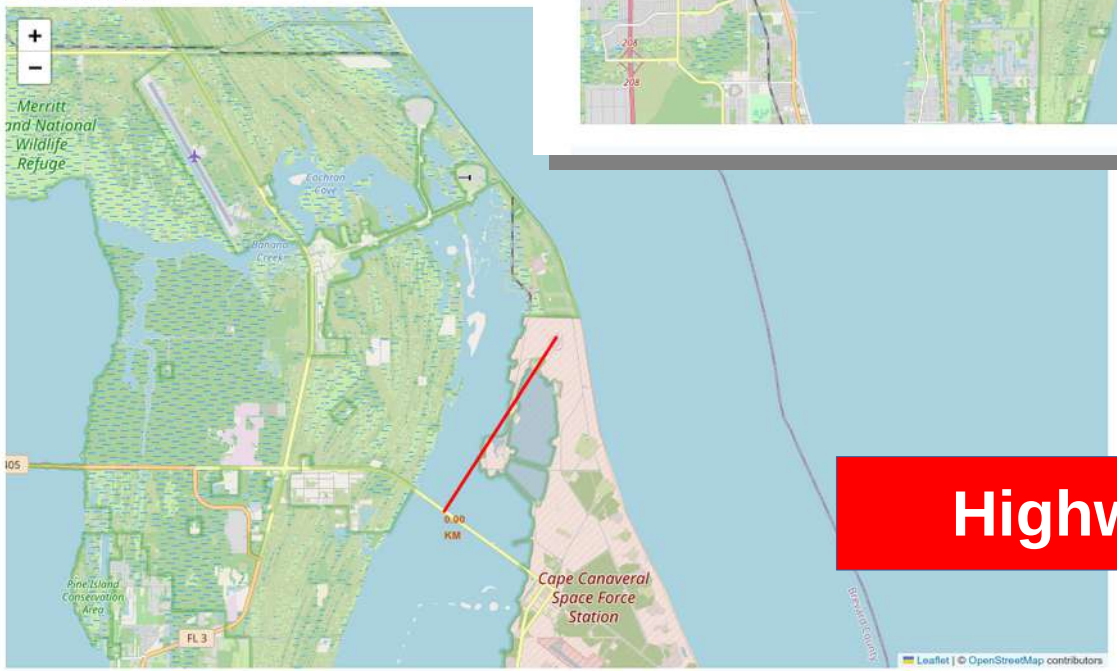
Launch Site Distance from …

```
airport_launch_site_map.add_child(lines)
[63] 20ms
21.309461410851824
```



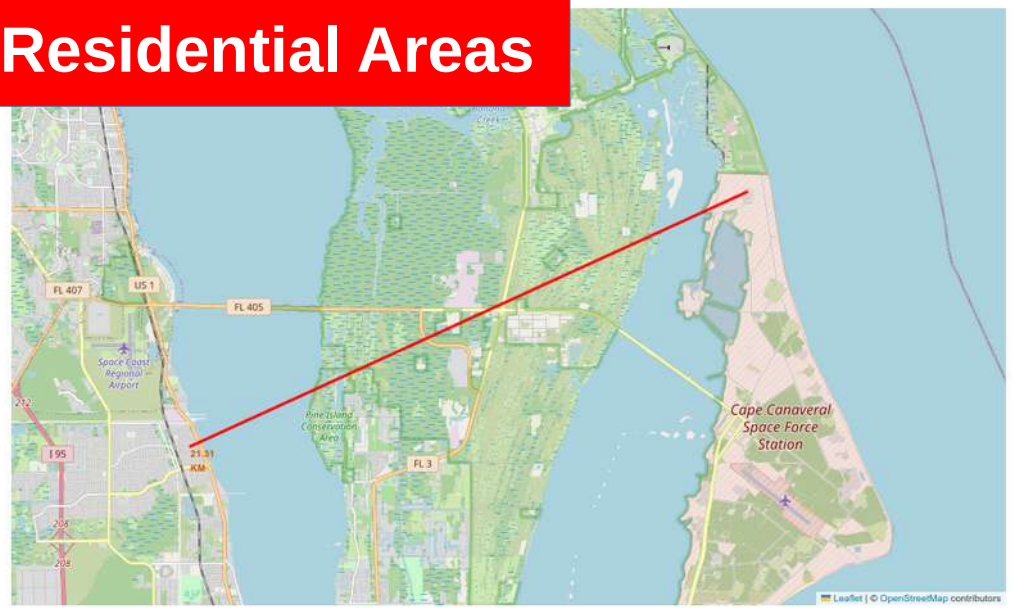
Airports

```
highway_launch_site_map.add_child(lines)
[55] 20ms
6.498610149144287
```

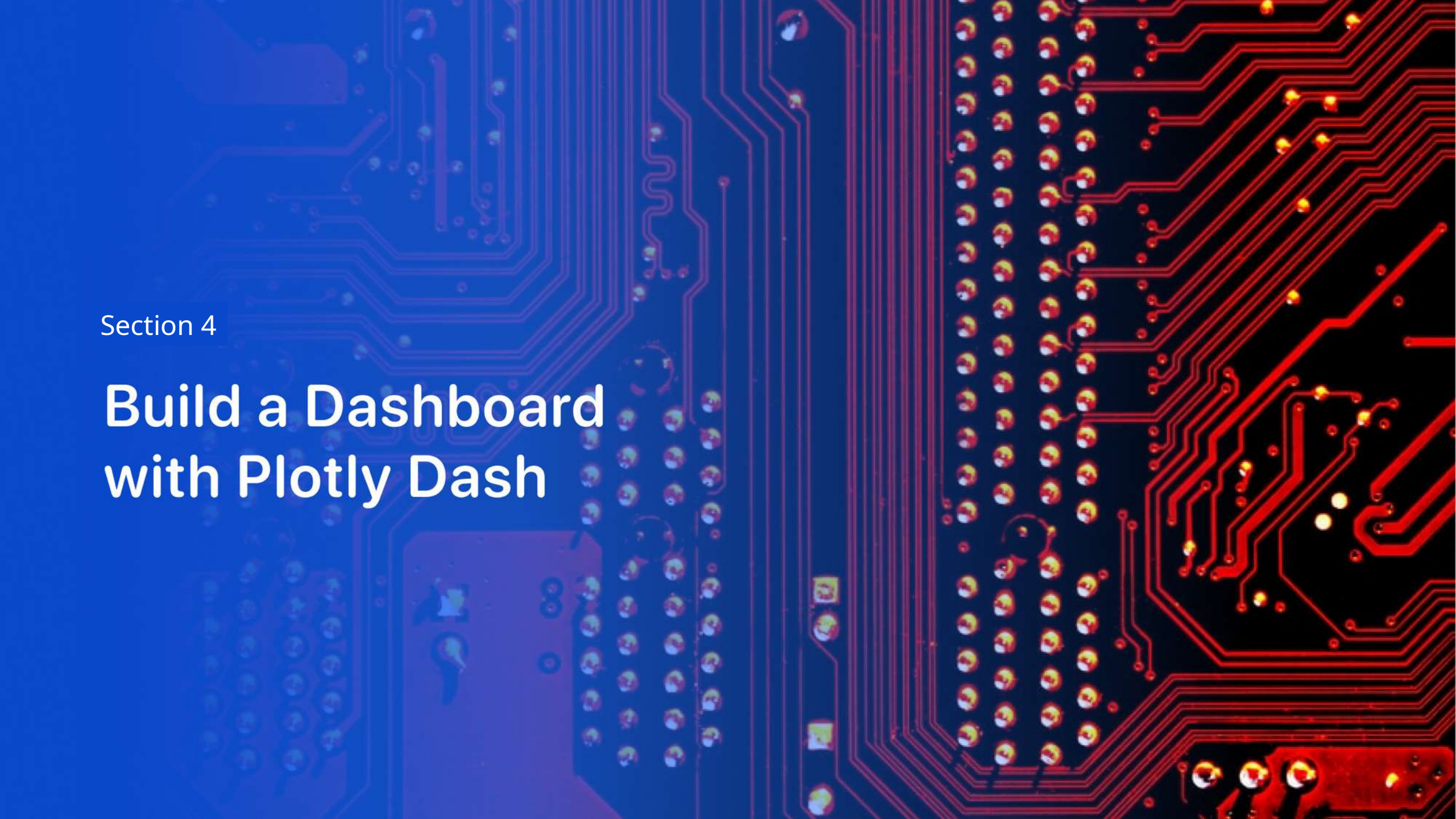


Highways

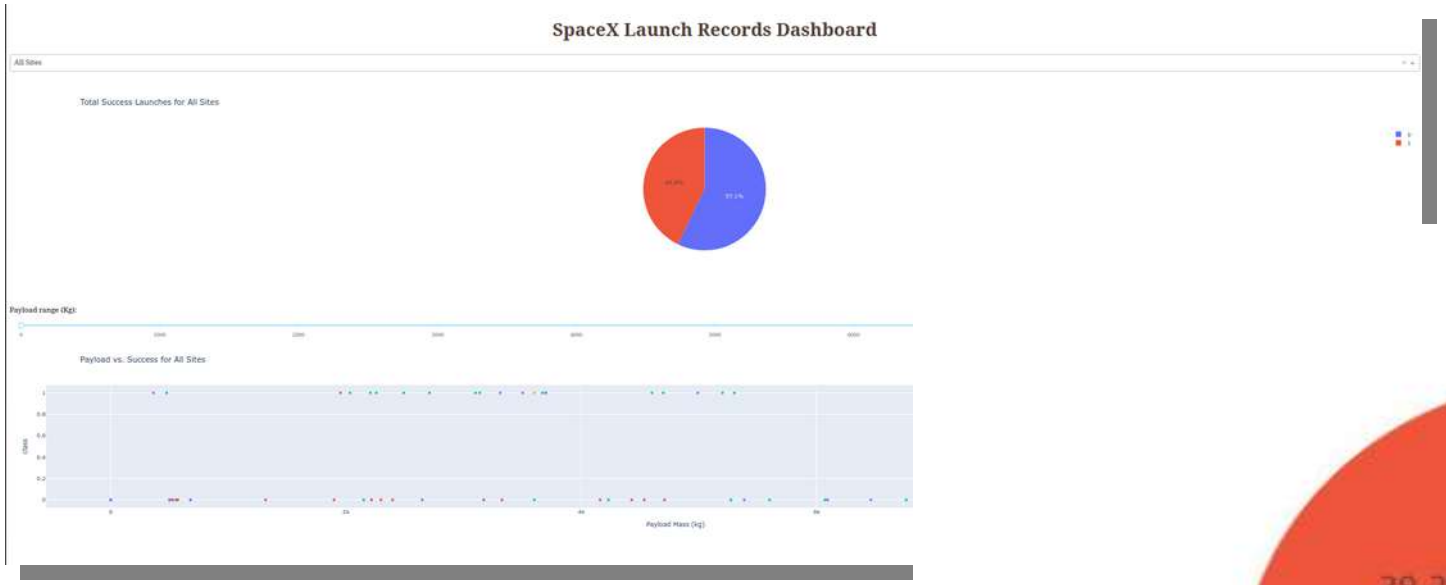
```
residential_launch_site_map.add_child(lines)
[59] 16ms
```

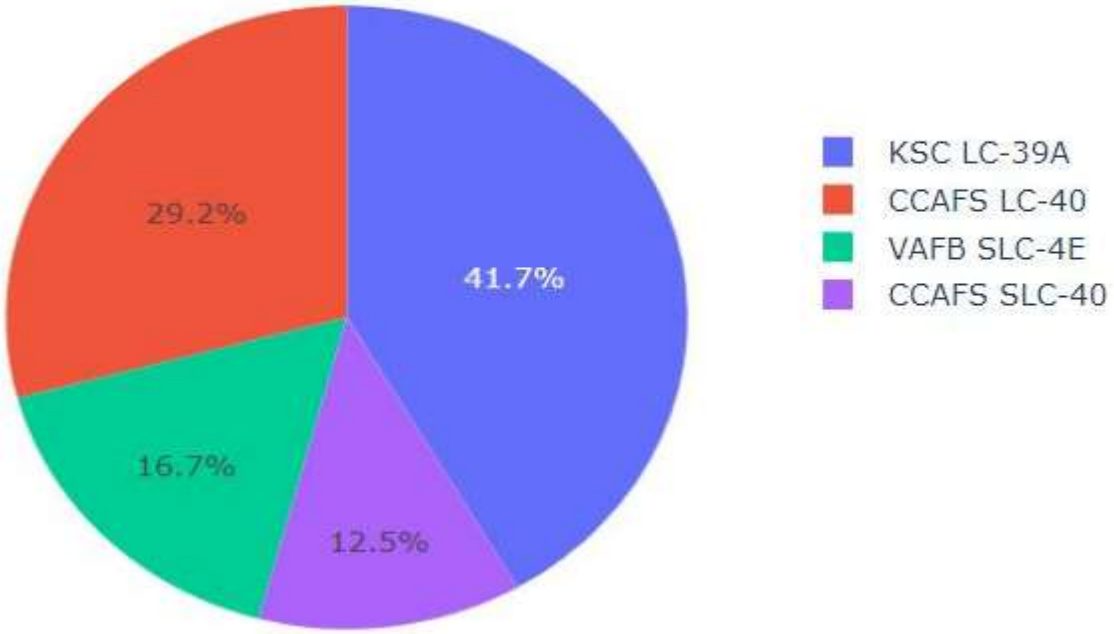


Residential Areas


Section 4
Build a Dashboard with Plotly Dash

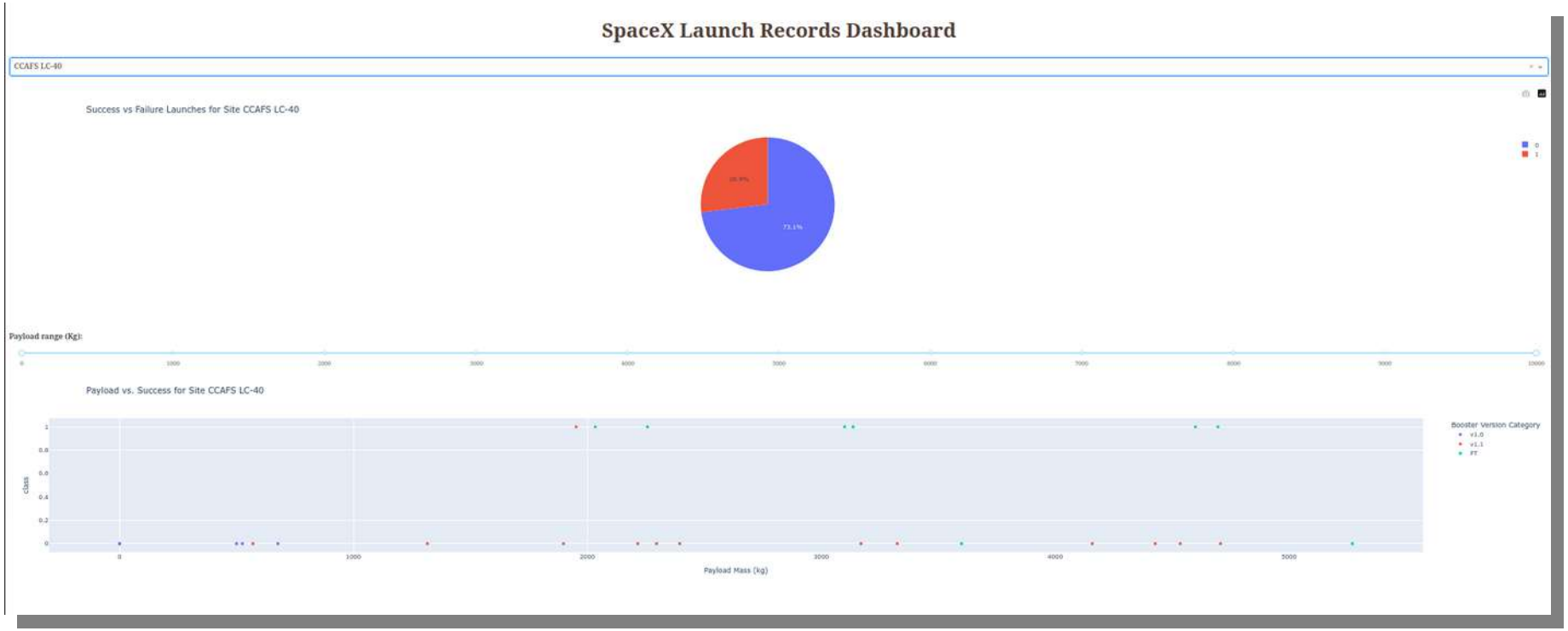
Launch Success Dashboard - All





Launch Site KSC LC-39A has the highest success rate at 41.7% of the launches across all launch sites

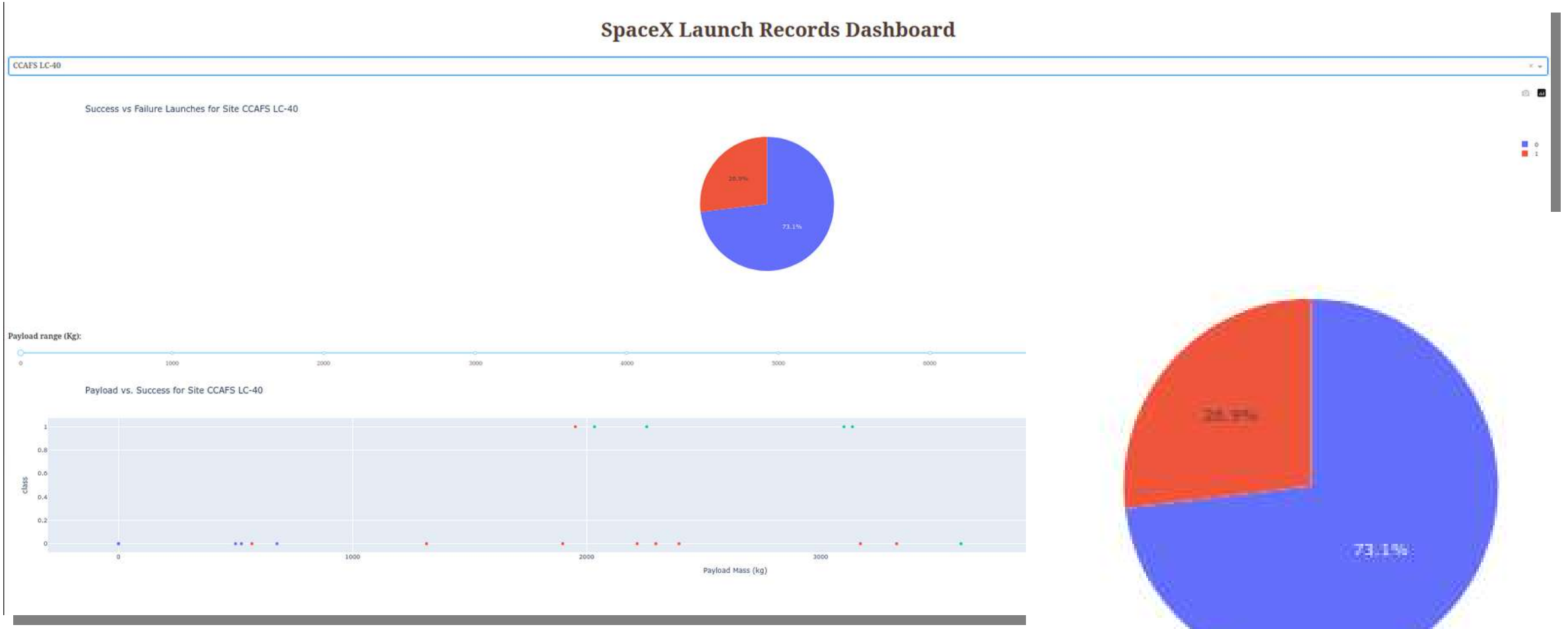
Highest Success Rate Dashboard



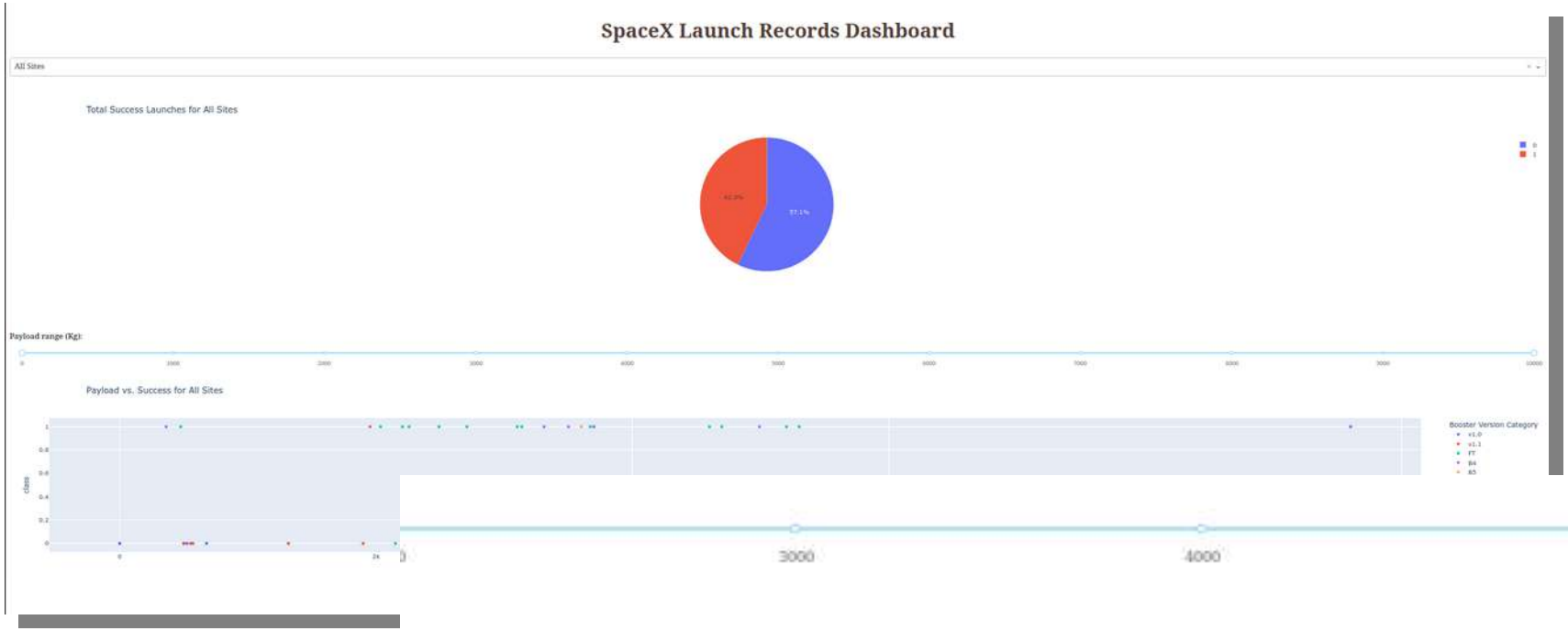
KSC LC-39A has the highest success rate at **76.9%,** compared to the other sites:

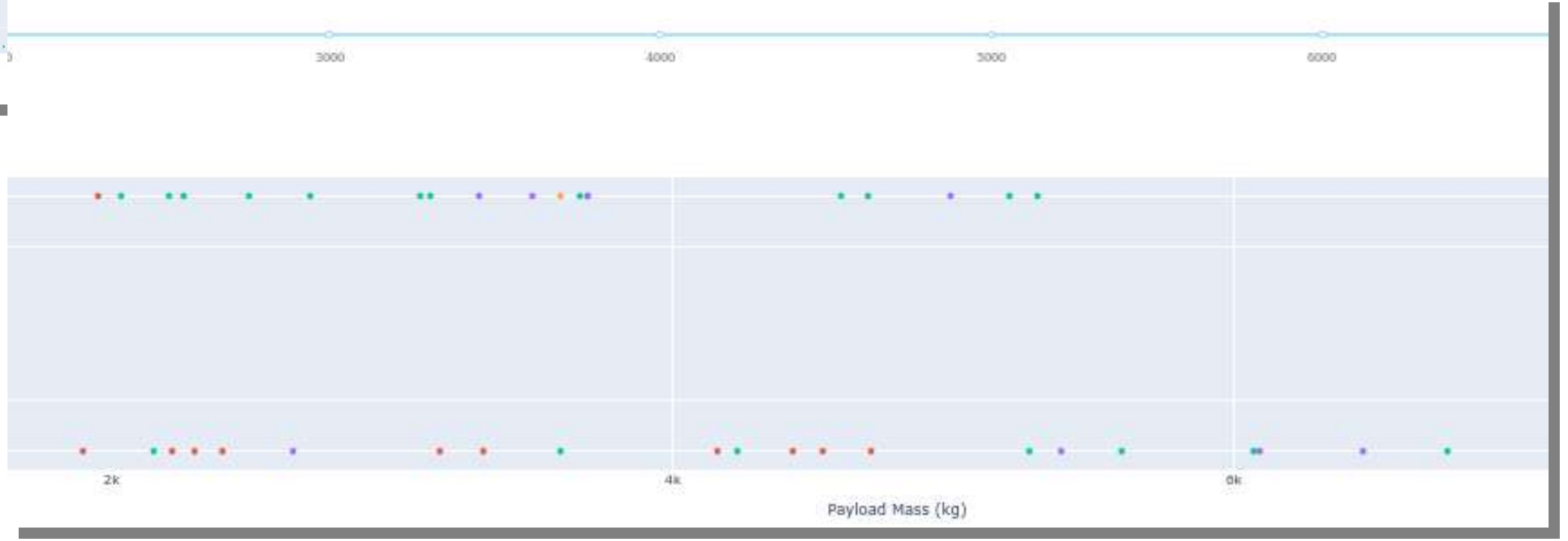
- 73.1% for CCAFS-LC-40
- 60% for VAFB-SLC-4E
- 57.1% for CCAFS-SLC-40

Highest Launch Success Ratio Dashboard



Pay Load vs Launch Outcome Dashboard





Section 5

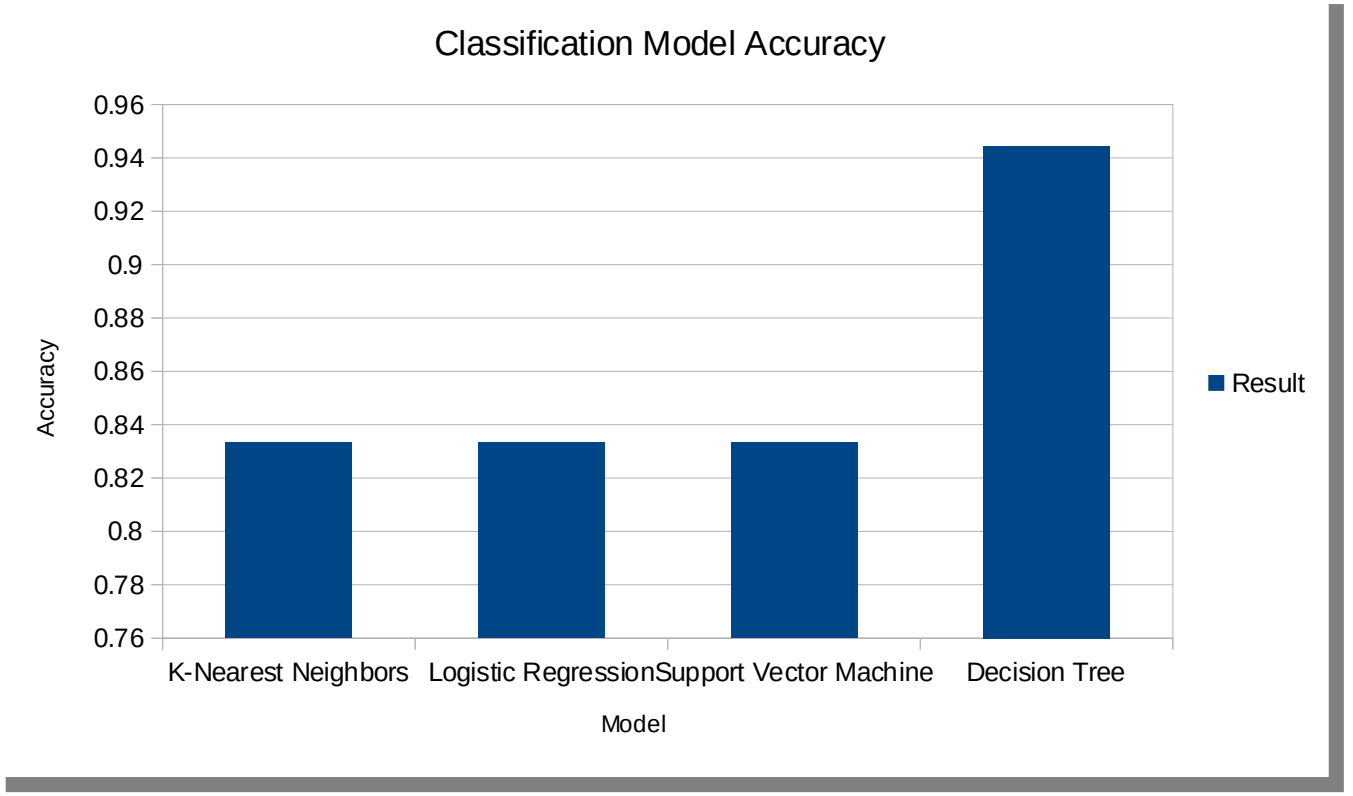
Predictive Analysis (Classification)

Classification Accuracy

Of the four models tested:

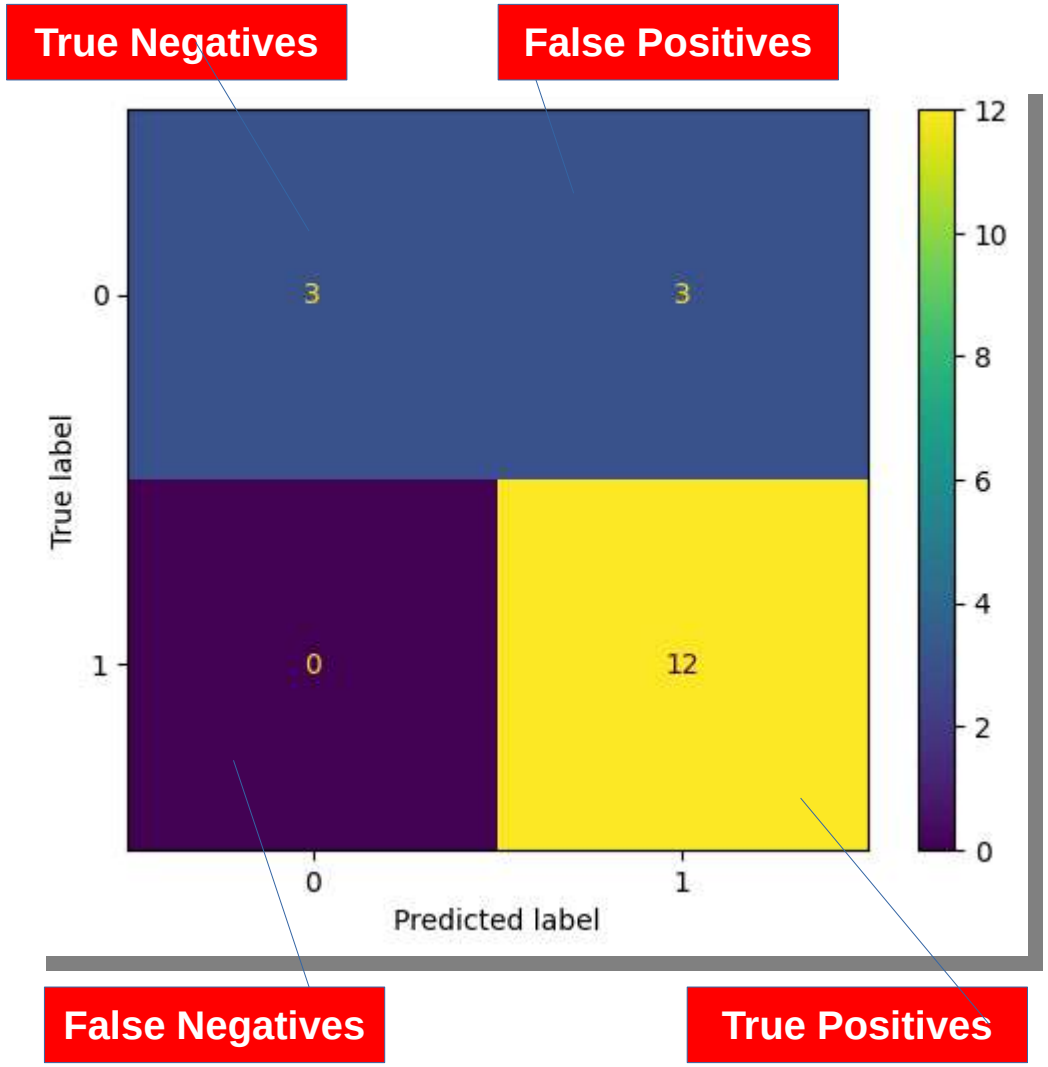
- K-Nearest Neighbors,
- Logistic Regression,
- Support Vector Machine and
- Decision Tree,

Decision Tree achieved the highest level of accuracy at 0.944 compared with 0.8333 across the other three.



Github link:
https://github.com/christophekey/Coursea_Data_Science_Capstone/tree/main/predictive_analytics

Confusion Matrix



The matrix shows two classes (0 and 1) and compares the true labels (y-axis) with the predicted labels (x-axis).

True Negatives (Top Left): 3 cases where the model correctly predicted class 0

False Positives (Top Right): 3 cases where the model predicted class 1 but the true label was 0

False Negatives (Bottom Left): 0 cases where the model predicted class 0 but the true label was 1

True Positives (Bottom Right): 12 cases where the model correctly predicted class 1

Performance metrics from this matrix:

Accuracy: (3 + 12)/(3 + 3 + 0 + 12) = 15/18 ≈ 83.3%
Precision for class 1: 12/(3 + 12) = 12/15 = 80%
Recall for class 1: 12/(0 + 12) = 12/12 = 100%
False Positive Rate: 3/(3 + 3) = 50%

The model appears to be performing well at identifying class 1 (100% recall) but has some difficulty with false positives, as it incorrectly classified 3 instances of class 0 as class 1. The overall accuracy is good at 83.3%. It should be noted this is based on a small sample

Conclusions

Factors that influence successful rocket launches:

Payload Mass: Smaller payloads correlate with higher success rates.

Orbit Selection: Certain orbits (VLEO, ES-L1, GEO, HEO, and SSO) demonstrate better success rates than others.

Launch Site Location: Sites are strategically positioned along coastlines for recovery purposes and away from populated areas for safety.

Historical Trend: Success rates have improved over time, with more recent launches showing better outcomes.

Prediction Model: Decision tree classifiers achieved the highest accuracy (87%) among tested algorithms for predicting landing outcomes.

This analysis suggests that optimising these factors - particularly payload size and orbit selection - while leveraging decision tree models for prediction, can help maximize mission success probability.

Thank you!