IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Dolev Zaiderman
10/04/2024

Outline

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

Executive Summary

- Summary of methodologies:

• Data Collection via API, Web Scraping

• Exploratory Data Analysis (EDA) with Data Visualization

• EDA with SQL

• Interactive Map with Folium

• Dashboards with Plotly Dash

• Predictive Analysis

- Summary of all results:

• Exploratory Data Analysis results

• Interactive maps and dashboard

• Predictive results

Introduction

- Project background and context

The objective of this project is to forecast the successful landing of the Falcon 9 first stage. According to SpaceX's website, the launch cost of the Falcon 9 rocket is $62 million, significantly lower than other providers whose costs can exceed $165 million per launch. This discrepancy in price is attributed to SpaceX's capability to reuse the first stage. By accurately predicting the landing success of the stage, we can ascertain the overall cost of a launch. Such insights are valuable for other companies aiming to rival SpaceX in the rocket launch market.

- Problems you want to find answers

1.What are the key attributes indicating a successful or unsuccessful landing?

2.How do different rocket variables influence the likelihood of a successful or failed landing?

3.What specific conditions are necessary for SpaceX to optimize its landing success rate?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scrapping from
- Perform data wrangling
  - Dropping unnecessary columns
  - One Hot Encoding for Classification models
- 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

Data Collection

•Datasets are collected from Rest SpaceX API and webscrapping Wikipedia

•The information obtained by the API are rocket, launches, payload information.

•The Space XREST API URL is api.spacexdata.com/v4/

•The information obtained by the webscrapping of Wikipedia are launches, landing, payload information.
URL is:
https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Data Collection – SpaceX API

Step 1: API Connection

```
In [6]:
spacex_url="https://api.spacexdata.com/v4/launches/past"
In [7]:
response = requests.get(spacex_url)
```

Step 2: Creating JSON File

```
data = response.json()
data = pd.json_normalize(data)
```

Step 3: Data Transformation

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
getBoosterVersion(data)
```

Step 4: Dictionary

```
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}
```

Step 5: Creating DF

```
df = pd.DataFrame.from_dict(launch_dict)
data_falcon9=df[df['BoosterVersion']!='Falcon 1']
```

Click Here For Code

Data Collection - Scraping

Step 1: Connect to URL via HTML

```
data = requests.get(static_url).text
```

Step 2: BeautifulSoup Object

```
soup = BeautifulSoup(data, 'html.parser')
```

Step 3: Find all method

```
html_tables = soup.find_all('table')
```

Step 4: Get columns names

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

Step 5: Get columns names

```
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'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Step 6: Add data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
```

Step 7: Converting to DataFrame

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

Click Here for Code

Data Wrangling

- Data wrangling is the process of cleaning, transforming, and preparing raw data for analysis. It involves steps such as acquiring data, cleaning errors, transforming formats, integrating multiple sources, enriching with additional information, validating quality, exploring insights, and documenting the process.

Step 1: Calculating numbers of each site

```
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Step 2: Occurrences of each obit

```
df['Orbit'].value_counts()
```

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

Step 3: Outcome of each occurrence

```
landing_outcomes=df['Outcome'].value_counts()
landing_outcomes
```

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

Step 4: Outcome Labeling

```
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

Step 5: Creating DF

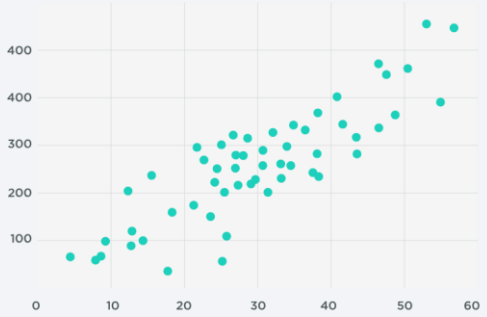
```
df['Class']=landing_class
```

[Click Here for Code]

EDA with Data Visualization

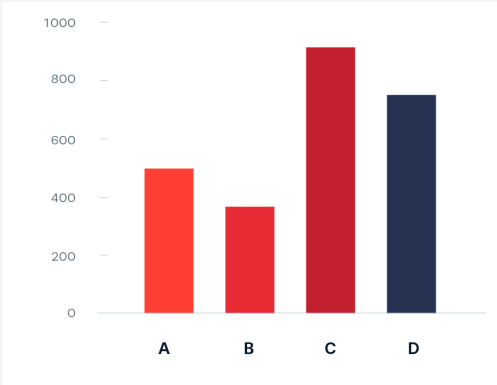
Scatter Graphs:

1. Scatter graphs are useful for displaying the relationship between two continuous variables.
2. They are effective in identifying patterns, trends, and correlations in data.
3. Useful for spotting outliers or clusters within the data.



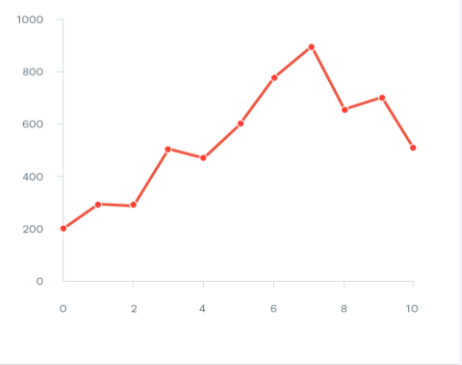
Bar Graphs:

1. Bar graphs are effective for comparing categorical data.
2. They make it easy to visualize and compare the frequency, distribution, or magnitude of different categories.
3. Useful for displaying discrete data or data that can be categorized into distinct groups.



Line Graphs:

1. Line graphs are ideal for displaying trends and changes over time.
2. They show how a variable changes in relation to another variable, typically time.
3. Effective in illustrating patterns, fluctuations, and trends in data.



EDA with SQL

Click Here for Code

- Show the unique launch site names in the space missions.
- Display five records where the launch sites start with ‘CCA’.
- Present the total payload mass carried by boosters launched by NASA (CRS).
- Show the average payload mass carried by booster version F9 v1.1.
- List the date of the first successful landing outcome on a ground pad.
- Enumerate the names of boosters that successfully landed on a drone ship with payload mass between 4000 and 6000.
- Provide the total counts of successful and failed mission outcomes.
- List the names of booster versions that carried maximum payload mass.
- Display records showing month names, failure landing outcomes on drone ships, booster versions, and launch site for the year 2015.
- Rank the count of successful landing outcomes between April 6, 2010, and March 20, 2017, in descending order.

Build an Interactive Map with Folium

- **Folium map object is a map centered on NASA Johnson Space Center at Houson, Texas:**
  1. Red circle at NASA Johnson Space Center's coordinate with label showing its name (folium.Circle, folium.map).
  2. Red circles at each launch site coordinates with label showing launch site name (folium.Circle, folium.map.Marker, folium.features.DivIcon).
  3. The grouping of points in a cluster to display multiple and different information for the same coordinates (folium.plugins).
  4. Markers to show successful and unsuccessful landings. Green for successful landing and Red for unsuccessful landing. (folium.map.Marker, folium).
  5. Markers to show distance between launch site to key locations (railway, highway, coastway, city) and plot a line between them . folium.map.Marker, folium.PolyLine, folium.features.DivIcon)

- **These elements are designed to enhance comprehension of the issue and the dataset. They enable straightforward visualization of all launch sites, their environs, and the tally of successful and unsuccessful landings.**

Click Here for Code

Build a Dashboard with Plotly Dash

•**Dashboard has dropdown,pie chart,rangeslider andscatter plot components:**

•Dropdownallows a user to choose the launch site or all launch sites (dash_core_components.Dropdown).

•Pie chart shows the total success and the total failure for the launch site chosen with the dropdown component(plotly.express.pie).

•Rangeslider allows a user to select a payload mass in a fixed range (dash_core_components.RangeSlider).

•Scatter chart showsthe relationship between two variables, in particular Success vs Payload Mass (plotly.express.scatter).

- **Dropdowns, pie charts, range sliders, and scatter plots in Plotly are essential for visualizing various data types and enabling interactive exploration. Dropdowns filter categories, pie charts show proportions, range sliders select data ranges, and scatter plots visualize variable relationships, enhancing data analysis.**

Click Here for Code

Predictive Analysis (Classification)

Step 1-Data preparation:

Dataset loading

Data normalization

Splitting data into training and test sets

Step 2-Model preparation:

Selection of machine learning algorithms

Setting parameters for each algorithm using GridSearchCV

Training GridSearchModel models with the training dataset

Step 3-Model evaluation:

Obtaining best hyperparameters for each model

Computing accuracy for each model with the test dataset

Plotting Confusion Matrix

Step 4-Model comparison:

Comparing models based on their accuracy

Choosing the model with the best accuracy

Click Here for Code

Results

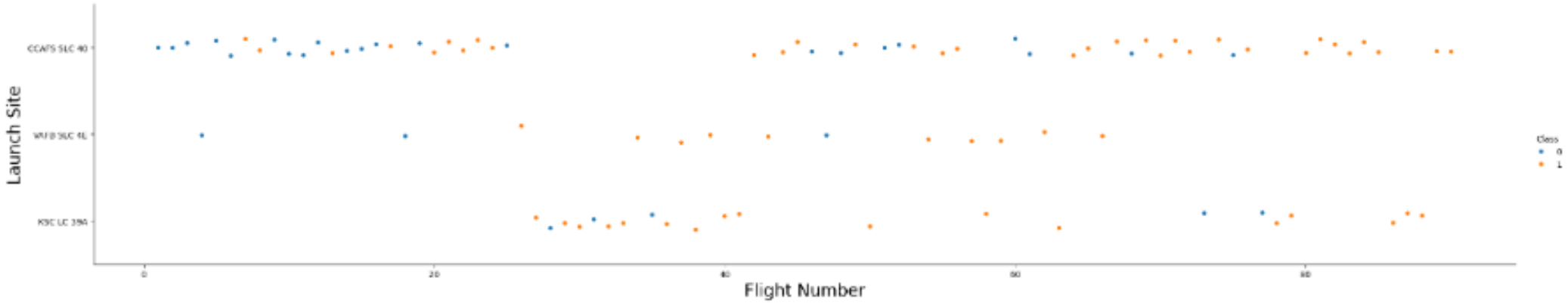
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

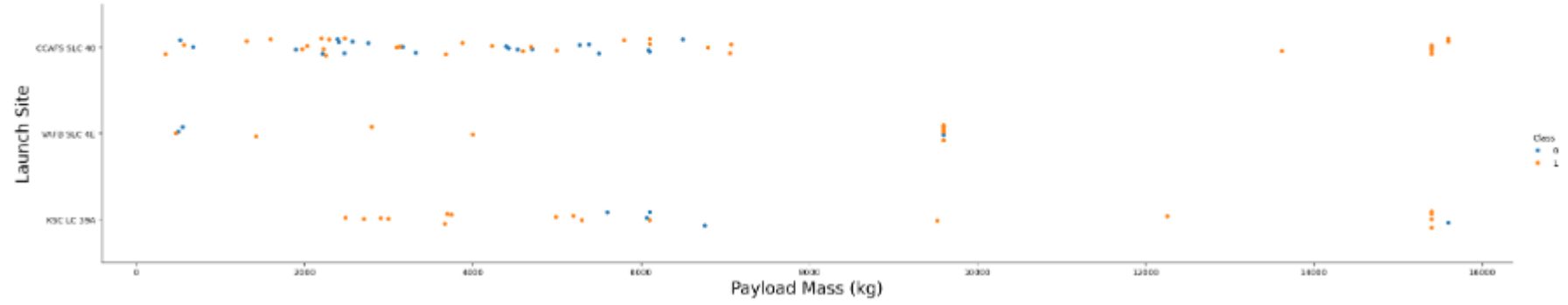
```
sns.catplot(y="LaunchSite",x="FlightNumber",hue="Class", data=df, aspect = 5)
5
plt.ylabel("Launch Site",fontsize=20)

plt.xlabel("Flight Number",fontsize=20)

plt.show()
```



We note that the success rate is on the rise for every site.

Payload vs. Launch Site

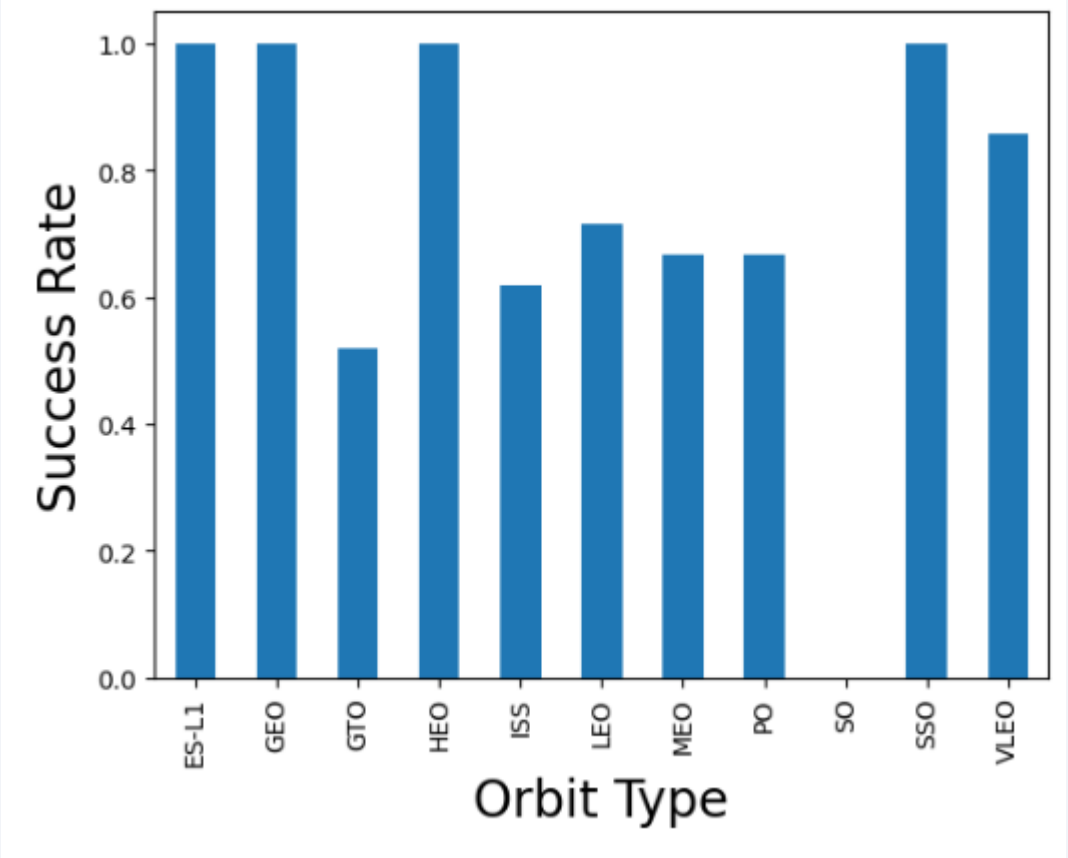
```
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



The choice of launch site may necessitate a heavier payload for a successful landing, but conversely, an excessively heavy payload could lead to a failed landing.

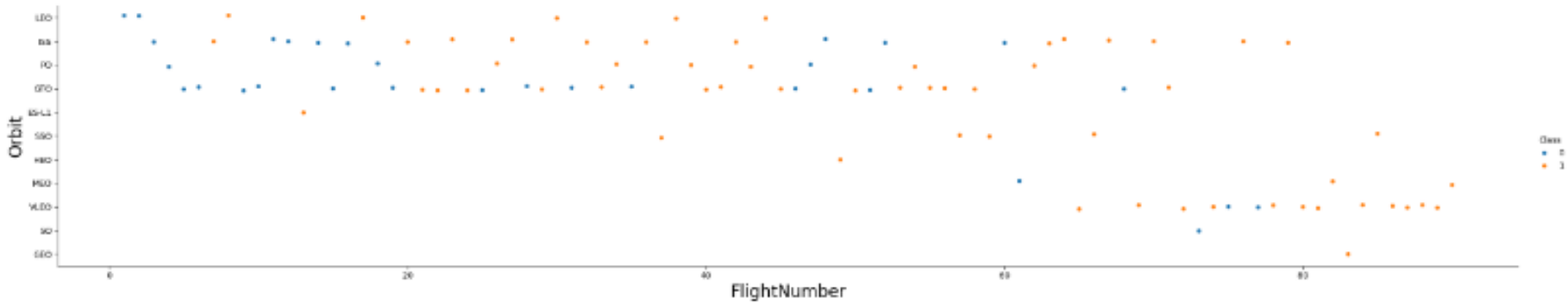
Success Rate vs. Orbit Type

Using this graph, we can visualize the success rates for various orbit types. It's evident that ES L1, GEO, HEO, and SSO exhibit the highest success rates.



Flight Number vs. Orbit Type

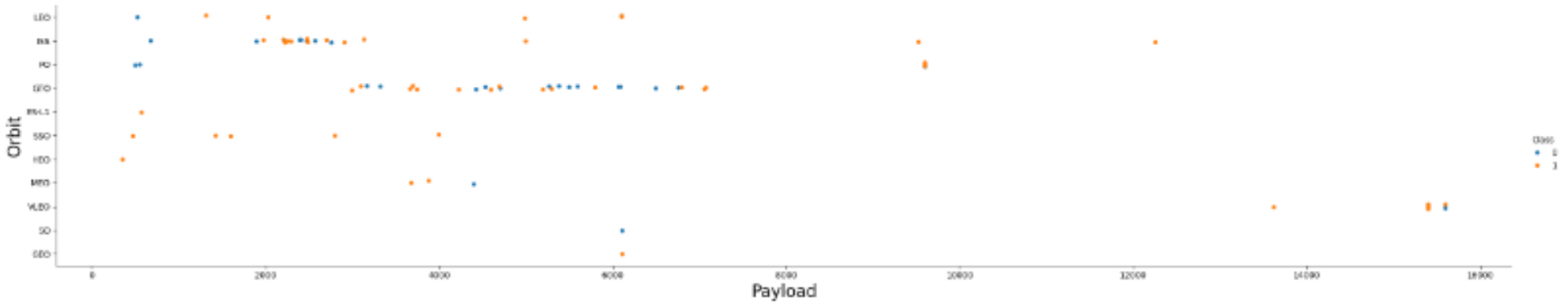
```
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



We observe a correlation between the success rate and the number of flights for the LEO orbit, where the success rate tends to increase with more flights. However, for orbits such as GTO, there appears to be no discernible relationship between the success rate and the number of flights. Nevertheless, it's reasonable to speculate that the high success rates observed in orbits like SSO or HEO may be attributed to insights gained from previous launches in other orbits.

Payload vs. Orbit Type

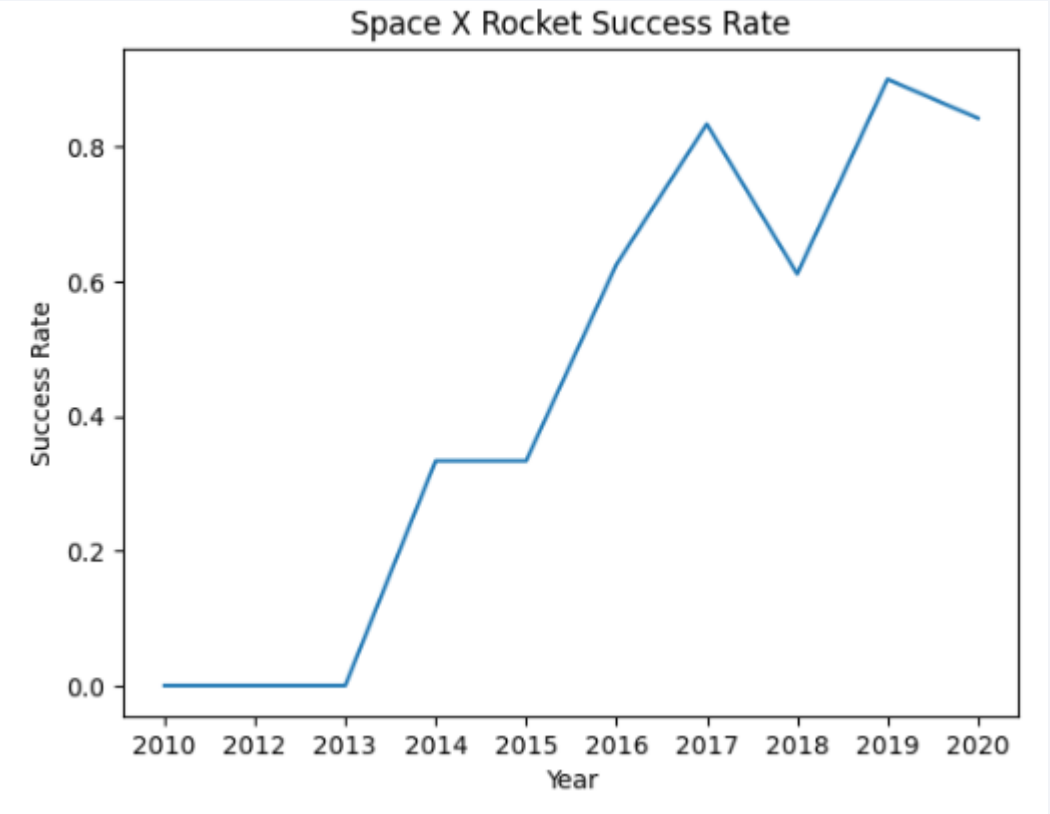
```
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



The weight of payloads significantly impacts the success rate of launches in specific orbits. For instance, heavier payloads enhance success rates for the LEO orbit, while reducing payload weight improves launch success in a GTO orbit.

Launch Success Yearly Trend

From 2013 onward, there has been a noticeable uptick in the success rate of SpaceX rocket launches.



All Launch Site Names

SQL Query:

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

Result:

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

Explanation:

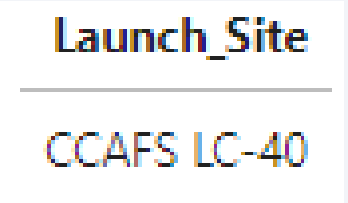
This SQL query retrieves unique values from the column "LAUNCH_SITE" in the table named "SPACEXTBL." It ensures that only distinct (unique) launch site names are returned, eliminating any duplicate entries.

Launch Site Names Begin with 'CCA'

SQL Query:

```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

Result:



Launch_Site
CCAFS LC-40

Explanation:

This SQL query selects the launch sites from the "SPACEXTBL" table where the launch site names start with 'CCA', limiting the output to the first five results.

Total Payload Mass

SQL Query:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

Result:

SUM(PAYLOAD_MASS__KG_)
45596

Explanation:

This SQL query sums the payload masses (in kilograms) from the "SPACEXTBL" table where the customer is 'NASA (CRS)'.

Average Payload Mass by F9 v1.1

SQL Query:

```
SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%'
```

Result:

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

Explanation:

This SQL query computes the average payload mass (in kilograms) from the "SPACEXTBL" table for records where the booster version starts with 'F9 v1.1'.

First Successful Ground Landing Date

SQL Query:

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';
```

Result:

MIN(Date)
2015-12-22

Explanation:

This SQL query finds the earliest date of successful ground pad landings recorded in the "SPACEXTBL" table.

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query:

```
SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND 4000 < PAYLOAD_MASS__KG_ < 6000;
```

Result:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Explanation:

This SQL query selects the booster versions from the "SPACEXTBL" table for successful drone ship landings with payload masses between 4000 and 6000 kilograms.

Total Number of Successful and Failure Mission Outcomes

SQL Query:

```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

Result:

Mission_Outcome	TOTAL_NUMBER
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Explanation:

This SQL query counts the occurrences of each unique mission outcome in the "SPACEXTBL" table and displays the results alongside the corresponding mission outcomes.

Boosters Carried Maximum Payload

SQL Query:

```
%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTBL \
WHERE "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

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

Explanation:

We utilized a subquery to filter data by retrieving only the maximum payload mass using the MAX function. The main query then utilizes the results of the subquery to return the unique booster versions alongside the maximum payload mass.

2015 Launch Records

SQL Query:

```
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL\
WHERE "LANDING_OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

Result:

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

Explanation:

This query retrieves the month, booster version, and launch site for unsuccessful landings that occurred in 2015. It uses the Substr function to extract the month or year from the date. Substr(DATE, 4, 2) extracts the month, while Substr(DATE, 7, 4) extracts the year.

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

SQL Query:

```
%%sql
SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING_OUTCOME
ORDER BY TOTAL_NUMBER DESC
```

Result:

Landing_Outcome	TOTAL_NUMBER
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:

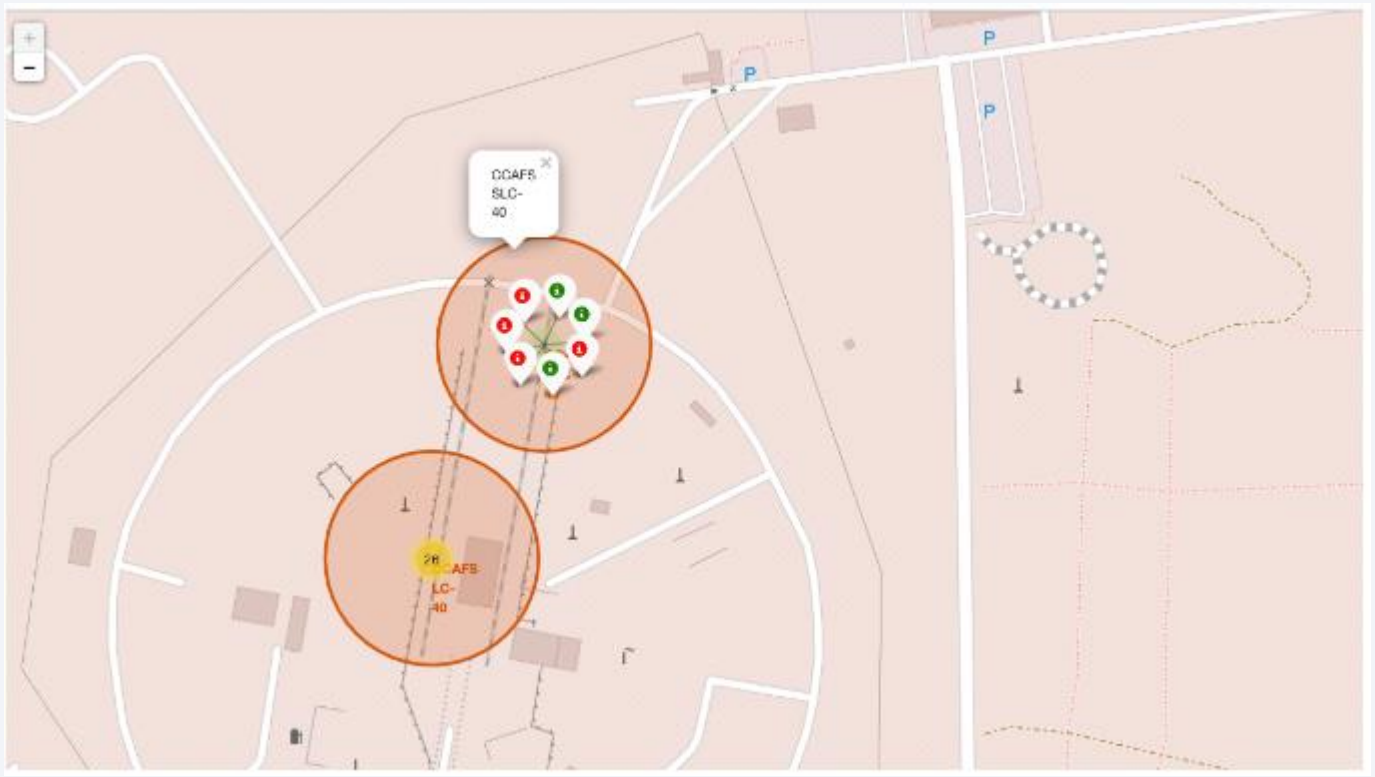
This SQL query counts the occurrences of each landing outcome in the "SPACEXTBL" table between June 4th, 2010, and March 20th, 2017. It then orders the results by the total number of occurrences in descending order.


Section 3
Launch Sites
Proximities Analysis

Folium Map- Launch main locations

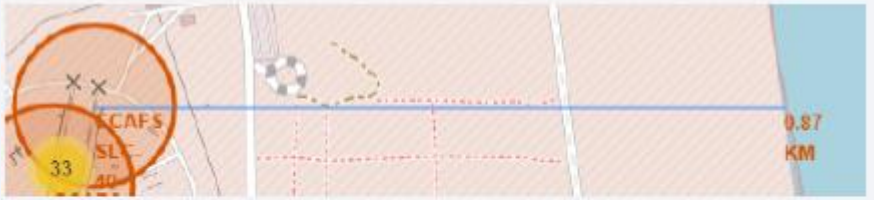
We see that Space X launch sites are located both in east and west coast of the United States.

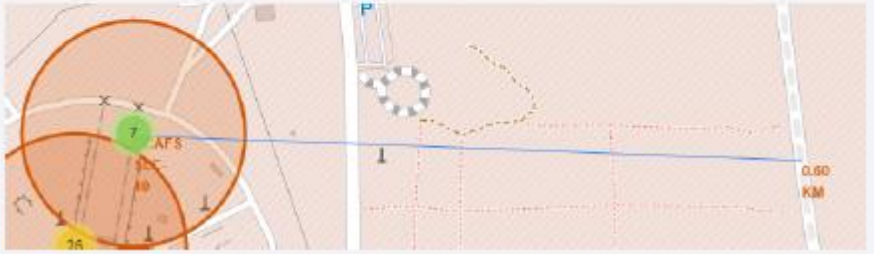
Folium Map- CCAFS SLC-40 Marks



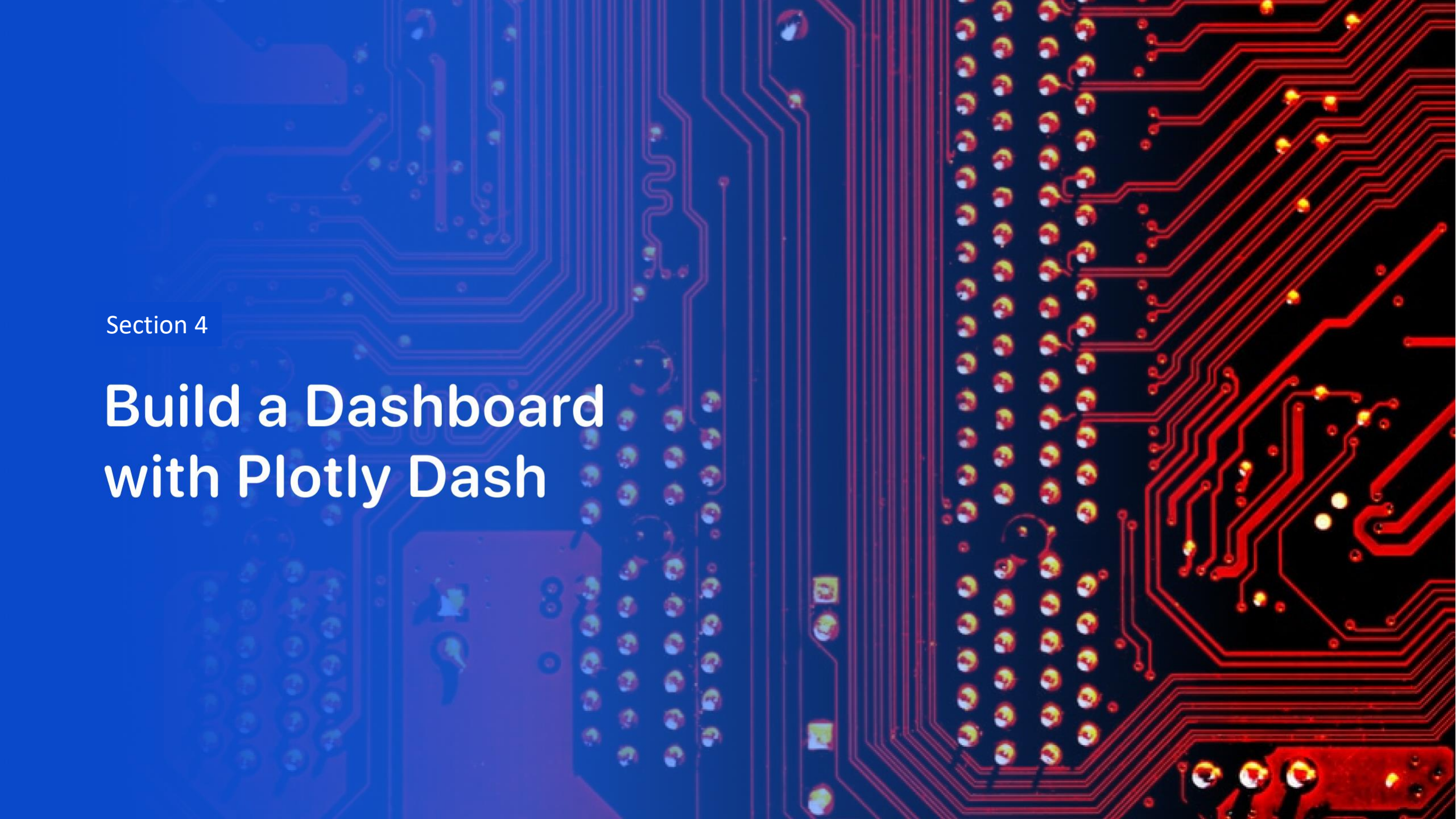
Green marker represents successful launches. Red marker represents unsuccessful launches.
We note that CCAFS SLC-40 has a lower launch success rate.

Folium Map- CCAFS SLC-40 Terrain

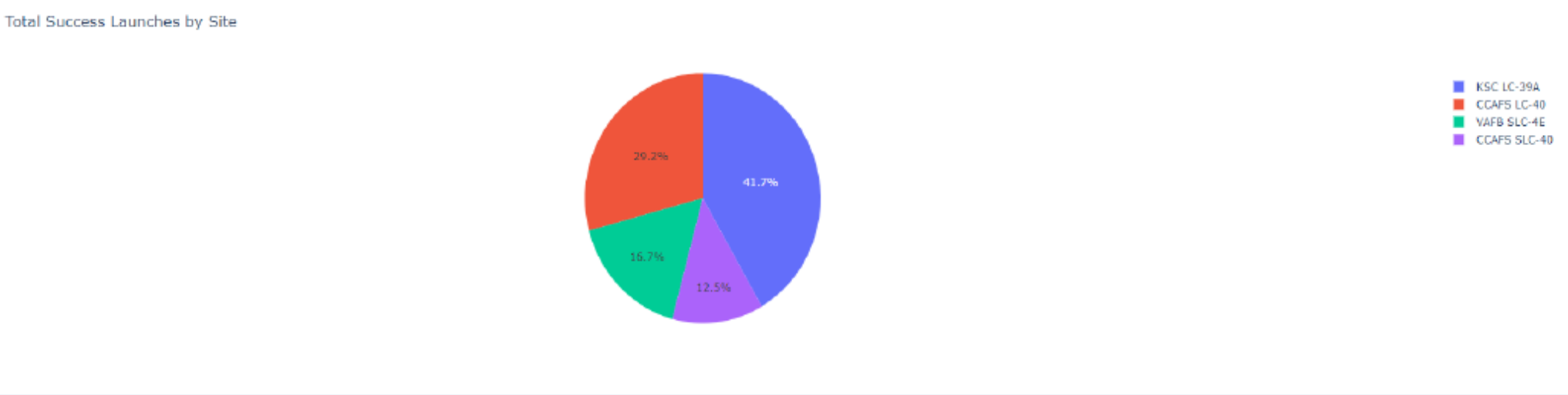




Is CCAFS SLC-40 in close proximity to railways ? Yes
Is CCAFS SLC-40 in close proximity to highways ? Yes
Is CCAFS SLC-40 in close proximity to coastline ? Yes
Do CCAFS SLC-40 keeps certain distance away from cities ?
No

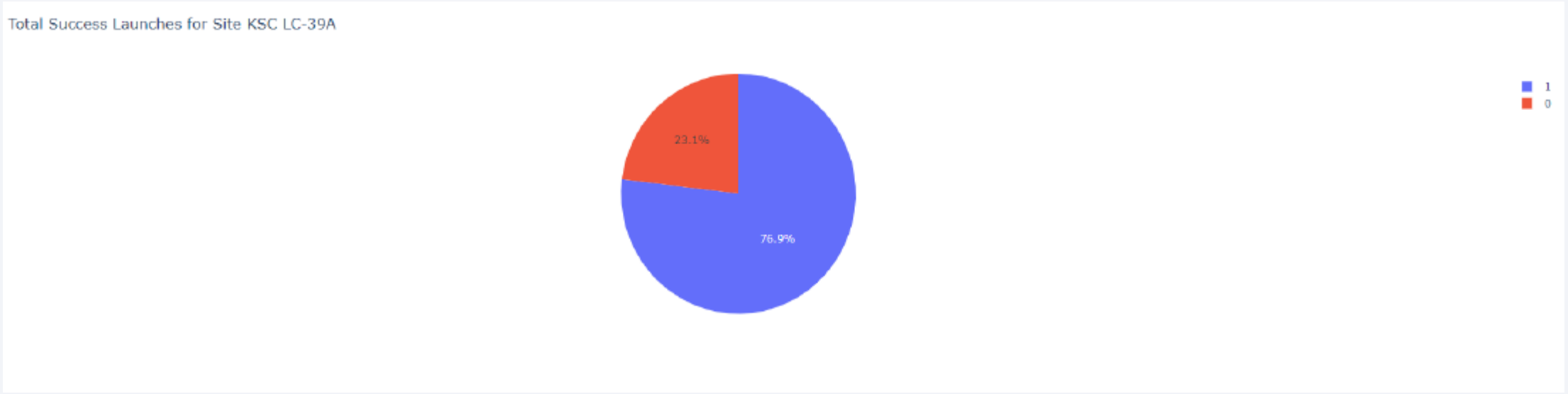

Section 4
Build a Dashboard with Plotly Dash

Dashboard – Success rate



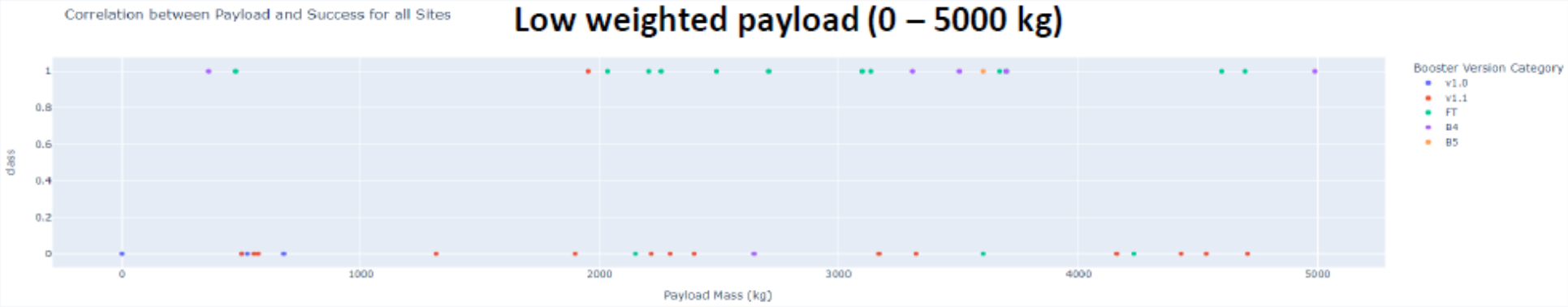
As a result we see that KSC LC39A has the best success rate of launches and VAFB SLC-4E has the least.

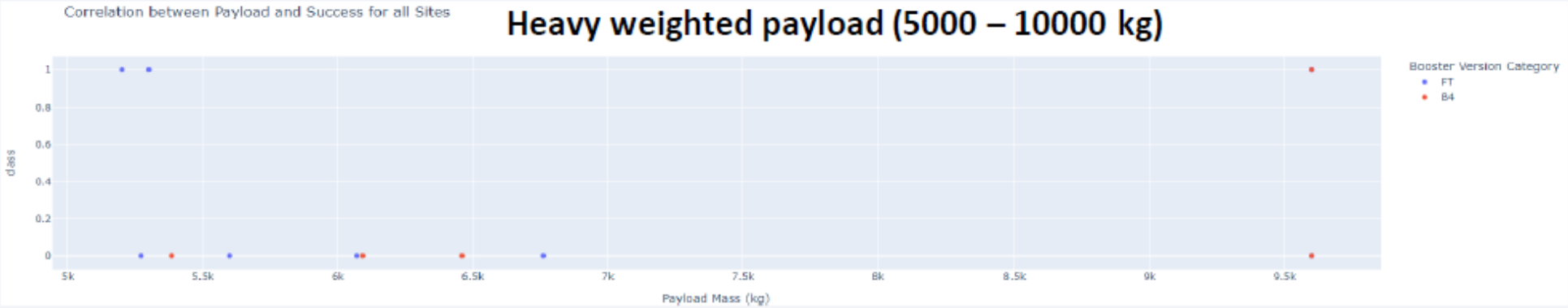
Dashboard- Site KSC LC-39A success distribution



As a result, we can see that KSC LC-39A has achieved a 76.9% success rate while getting a 23.1% failure rate.

Dashboard-Outcome plotted against payload mass for various sites with diverse payload masses selected.





Payloads with lower weight exhibit higher success rates compared to heavier payloads.

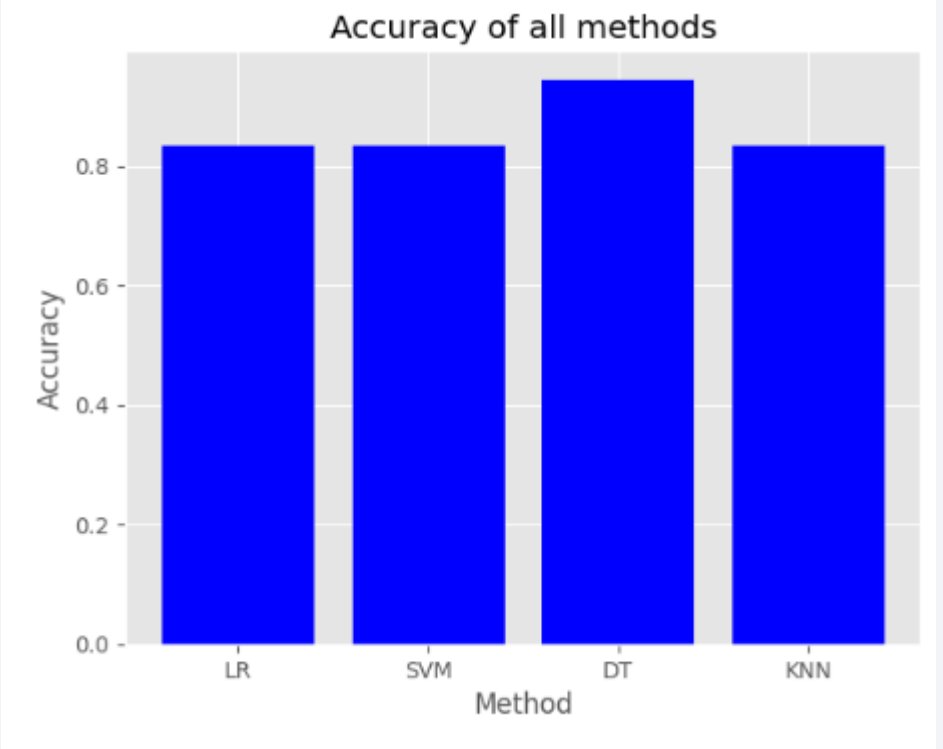
Section 5

Predictive Analysis (Classification)

Classification Accuracy

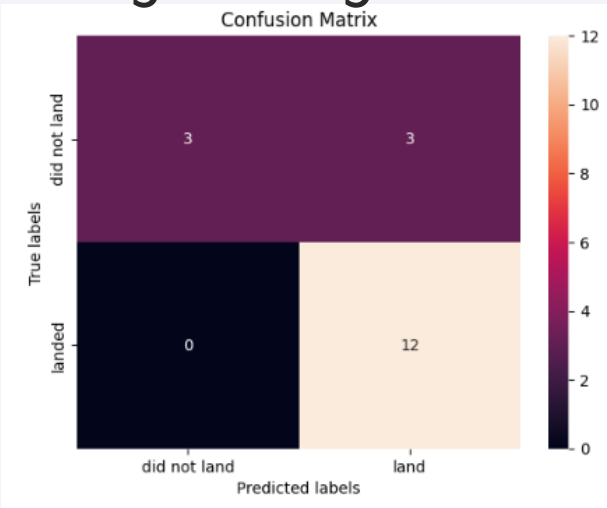
```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.8888888888888888
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```

The decision tree achieved higher accuracy in the confusion matrix compared to other methods, indicating its superior performance in correctly predicting instances across all classes.

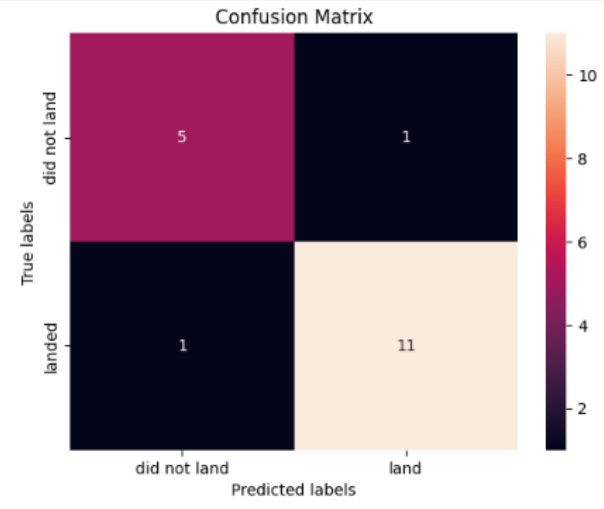


Confusion Matrix

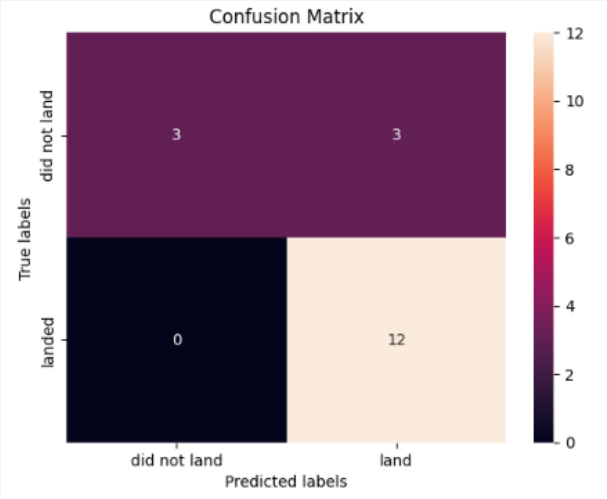
Logistic Regression



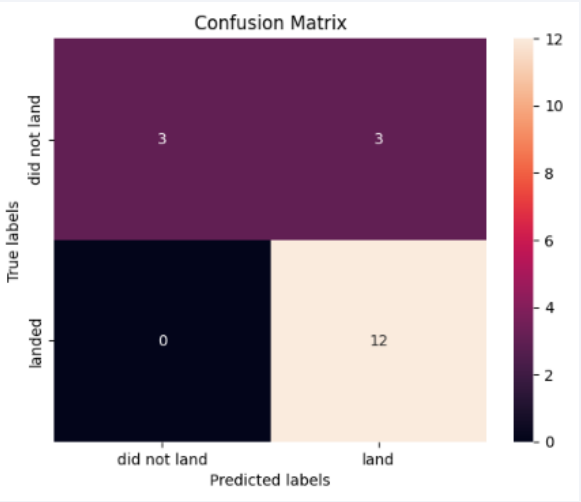
Decision Tree



SVM



KNN



The decision tree outperformed other methods in accuracy within the confusion matrix, demonstrating its superior ability to predict instances accurately across all classes.

Conclusions

- Mission success can be attributed to various factors, including the launch site, orbit type, and notably, the number of prior launches. It's reasonable to assume that knowledge gained between launches contributes to transitioning from launch failures to successes.
- Orbits with the highest success rates include GEO, HEO, SSO, and ES L1.
- Payload mass plays a significant role in mission success depending on the orbit. Some orbits require light or heavy payloads, though generally, lower-weight payloads tend to perform better than heavier ones.
- The reasons behind the varying success rates among launch sites, with KSC LC 39A being the most successful, remain unclear with the current data. Obtaining additional atmospheric or relevant data may provide insights into this phenomenon.
- Following analysis via the confusion matrix, the Decision Tree Algorithm emerges as the preferred model for this dataset.

Thank you!