IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tinashe Melissa Nyamurowa
24 August 2025

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection using REST API
  - Data Collection using Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis using SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics using Folium
  - Build interactive Dashboard using Plotly Dash
  - Machine Learning Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  In this capstone project, we aim to predict whether the Falcon 9 first stage will successfully land. SpaceX advertises Falcon 9 rocket launches on its website for a cost of $62 million, while other providers charge upwards of $165 million for each launch. Much of the savings for SpaceX comes from the ability to reuse the first stage. Consequently, if we can accurately determine whether the first stage will land, we can also assess the overall cost of a launch. This information could be valuable for an alternate company, Space Y, looking to bid against SpaceX for a rocket launch. In this lab, you will collect data and ensure it is in the correct format from an API. Below is an example of a successful launch.

- Problems you want to find answers
  - Our task is to determine the price of each SpaceX launch.
  - To achieve this, we will gather information about SpaceX and create dashboards for our team.
  - Additionally, we will assess whether SpaceX will reuse the first stage of the rocket.
  - Instead of relying on complex rocket science to predict the success of the first stage landing, we will train a machine learning model using public data to make predictions about SpaceX's reusability of the first stage.

Section 1

Methodology

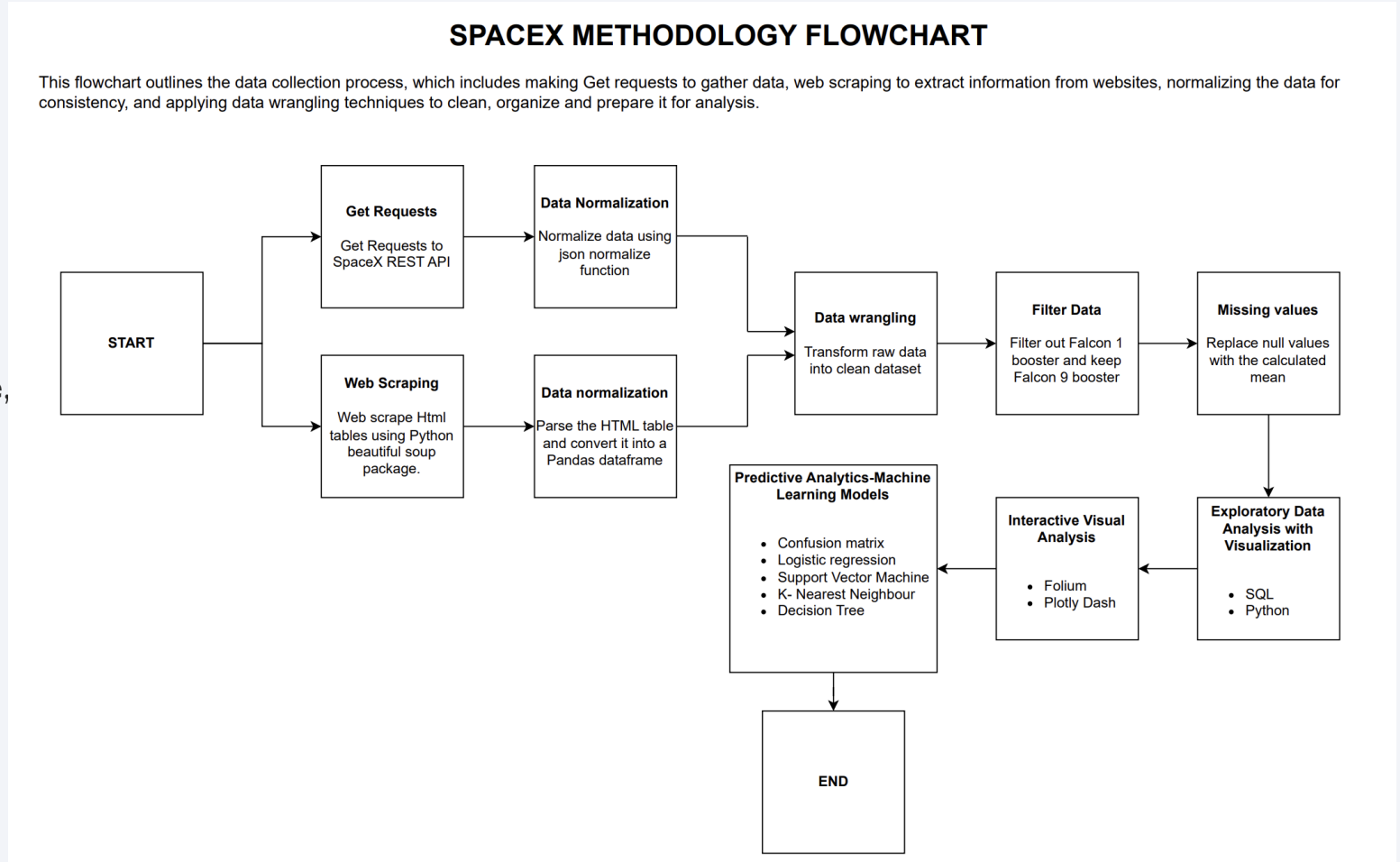
Methodology

Executive Summary

- Data collection methodology:
  - We collected the data using SpaceX REST API and Webscraping related Wiki pages
- Perform data wrangling
  - Using One-hot encoding was applied to categorical features
- 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

- The data was collected using two methods SpaceX Rest API and Webscraping.
- Get requests were performed to obtain launch data from SpaceX API.
- Then, the response was decoded in the form of a list of JSON objects by calling the .json() method.
- To normalize the structured json data into a flat table, we converted the JSON to a pandas dataframe using the json normalize function.
- Additionally, we used the Python BeautifulSoup package to web-scrape HTML tables that contain valuable Falcon 9 launch records. The data was parsed and converted to Pandas data frame for further visualization and analysis.
- Finally, the data was cleaned to provide meaning, filtered to remove Falcon 1 booster and we checked and replaced Null/missing values by calculating the mean of PayLoadMass and analyzed.



Data Collection – SpaceX API

- We used GET requests to obtain launch data from the SpaceX REST API.
- The data was normalized from a flat table into a Pandas DataFrame using the `json_normalize` function.
- We then transformed the raw data into a clean dataset by performing data wrangling with the API.
- After that, we filtered out the Falcon 1 boosters, as we only needed data for the Falcon 9 boosters.
- Finally, we addressed any null or missing values by calculating the mean of the relevant columns and replacing the null values with these means.
- GitHub URL of the completed SpaceX API calls notebook:

https://github.com/Melissa-Tynah/IBM-Applied-Data-Science-Capstone-Project./blob/main/Module%201.0_SpaceX-Data-Collection-API_TMN.ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
[9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API
```

We should see that the request was successfull with the 200 status response code

```
[10]: response=requests.get(static_json_url)
```

```
[11]: response.status_code
```

```
[11]: 200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
[12]: # Use json_normalize method to convert the json result into a dataframe

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

```
[23]: # Create a data from launch_dict

data_falcon9 = pd.DataFrame(launch_dict)
print(data_falcon9.head())
```

```
   FlightNumber        Date BoosterVersion  PayloadMass Orbit  \
0             1  2006-03-24       Falcon 1         20.0   LEO
1             2  2007-03-21       Falcon 1          NaN   LEO
2             4  2008-09-28       Falcon 1        165.0   LEO
3             5  2009-07-13       Falcon 1        200.0   LEO
4             6  2010-06-04       Falcon 9          NaN   LEO

       LaunchSite    Outcome  Flights  GridFins  Reused   Legs LandingPad  \
0  Kwajalein Atoll  None None        1     False   False  False       None
1  Kwajalein Atoll  None None        1     False   False  False       None
2  Kwajalein Atoll  None None        1     False   False  False       None
3  Kwajalein Atoll  None None        1     False   False  False       None
4     CCSFS SLC 40  None None        1     False   False  False       None

   Block  ReusedCount    Serial   Longitude   Latitude
0    NaN            0  Merlin1A  167.743129   9.047721
1    NaN            0  Merlin2A  167.743129   9.047721
2    NaN            0  Merlin2C  167.743129   9.047721
3    NaN            0  Merlin3C  167.743129   9.047721
4    1.0            0     B0003  -80.577366  28.561857
```

Show the summary of the dataframe

```
[24]: # Show the head of the dataframe

data.head()
```

Data Collection - Scraping

- We extracted the Falcon 9 launch HTML table from Wikipedia using BeautifulSoup. Then, we parsed the table and converted it into a Pandas DataFrame. The steps we followed included:

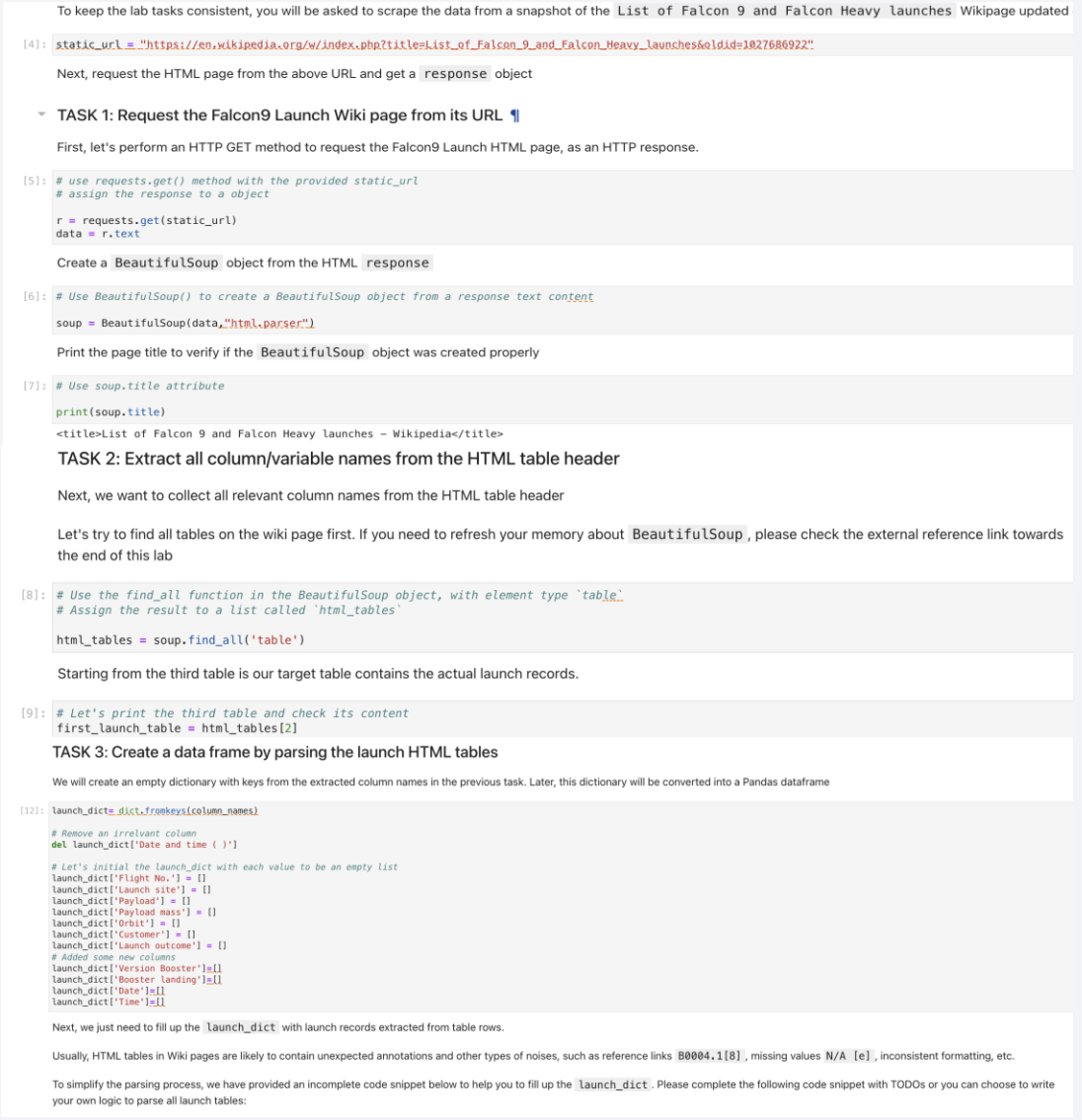
    i. Requesting the Falcon 9 launch Wikipedia page from its URL.

    ii. Creating a DataFrame by parsing the HTML table containing the launch information.

    iii. Extracting all column names from the HTML table header.

- GitHub URL of the completed SpaceX API calls notebook:

https://github.com/Melissa-Tynah/IBM-Applied-Data-Science-Capstone-Project./blob/100598121793d5c9e3abcd4818c04674a6170a93/Module%201.1_SpaceX_Webscraping_1_TMN.ipynb



To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated

[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 HTTP response.

[5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object

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

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

[6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

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

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

[7]:
```
# Use soup.title attribute

print(soup.title)
```

```
<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

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

[8]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`

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

Starting from the third table is our target table contains the actual launch records.

[9]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

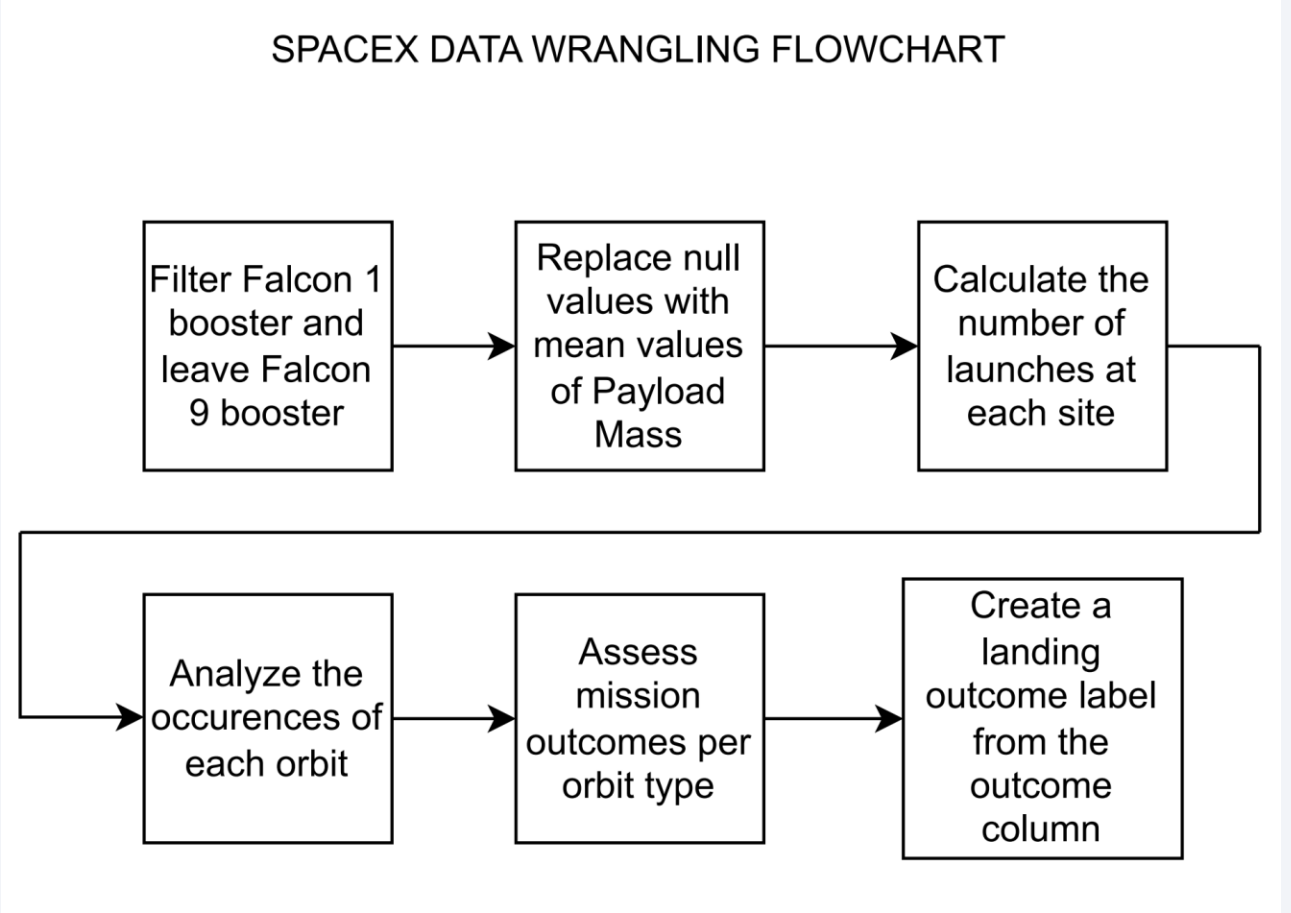
[12]:
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.

Usually, HTML tables in Wiki pages are likely to contain unexpected annotations and other types of noises, such as reference links `B0004.1[8]`, missing values `N/A [e]`, inconsistent formatting, etc.

To simplify the parsing process, we have provided an incomplete code snippet below to help you to fill up the `launch_dict`. Please complete the following code snippet with TODOs or you can choose to write your own logic to parse all launch tables:

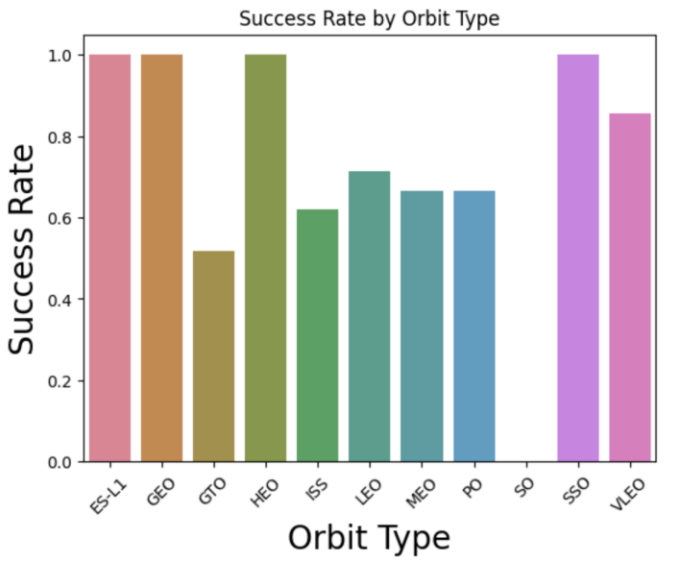
Data Wrangling

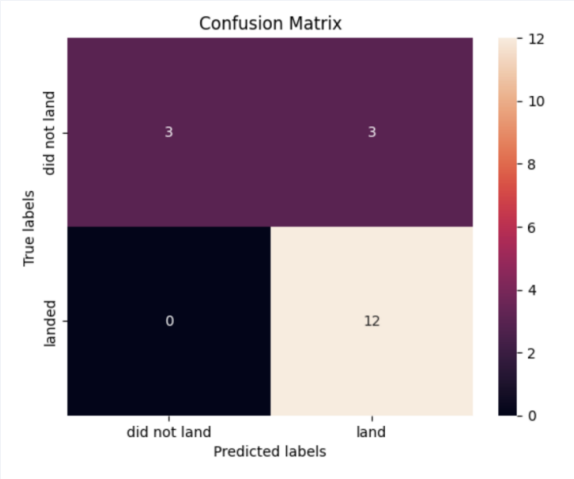


- The raw data was transformed into a clean dataset through data wrangling with an API. We filtered out the Falcon 1 booster, focusing only on the Falcon 9 booster, and addressed missing values by replacing them with the mean of relevant columns.

- The GitHub URL of the completed SpaceX API calls notebook: https://github.com/Melissa-Tynah/IBM-Applied-Data-Science-Capstone-Project./blob/100598121793d5c9e3abcd4818c04674a6170a93/Module%201.2_SpaceX_Data%20Wrangling_TMN.ipynb

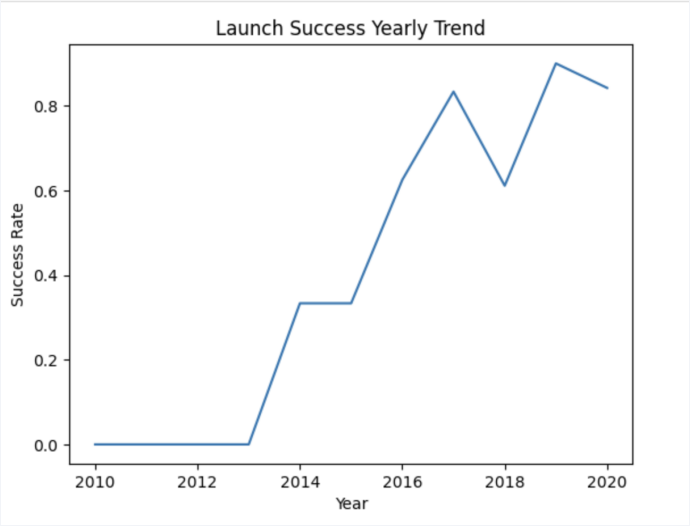
EDA with Data Visualization

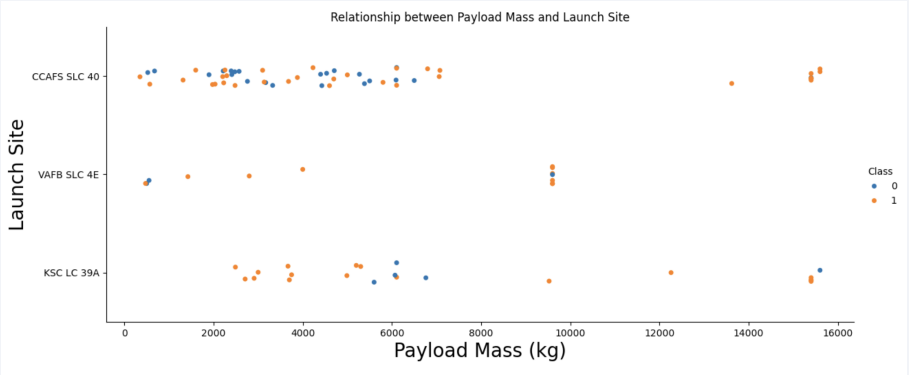


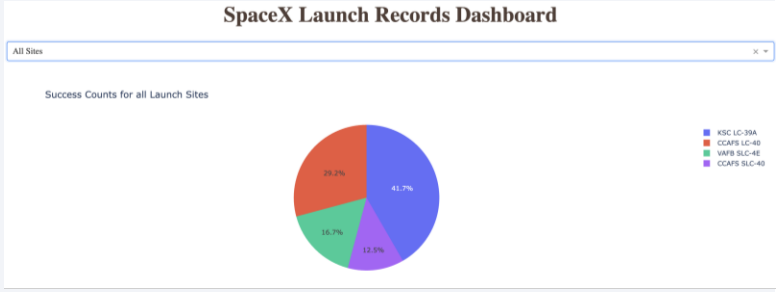


- We conducted exploratory analysis to visualize detailed launch records.
- Various plots, including catplots, bar charts, and line plots, were utilized to illustrate the relationships between different parameters, such as Launch Site, Flight Number, and Payload Mass, as well as to highlight yearly trends.
- The GitHub URL of the completed SpaceX API calls notebook:

https://github.com/Melissa-Tynah/IBM-Applied-Data-Science-Capstone-Project./blob/717702f81c164eae64498ddade1053596e349936/Module%202.1_SpaceX_EDA_Visualization_TMN.ipynb







EDA with SQL

- Firstly, we loaded the dataset into the corresponding table in a Db2 database.
- Using SQL, we performed the following queries to get insights from the data:
    i. Displayed unique launch site names.
    ii. Retrieved launch sites starting with 'CCA'.
    iii. Calculated NASA (CRS) boosters' total payload mass.
    iv. Identified the first successful ground pad landing date.
    v. Listed boosters with successful drone ship landings and specific payload mass.
    vi. Calculated total successful and failed mission outcomes.
    vii. Listed booster versions with maximum payload mass using subqueries.
    viii. Displayed failure landings in drone ships for 2015.
    ix. Ranked landing outcomes between 2010-06-04 and 2017-03-20.
- GitHub URL of the completed SpaceX API calls notebook: https://github.com/Melissa-Tynah/IBM-Applied-Data-Science-Capstone-Project./blob/100598121793d5c9e3abcd4818c04674a6170a93/Module%202.0_SpaceX_EDA_SQL_TMN.ipynb

Build an Interactive Map with Folium

- We created an interactive Folium map marking launch sites and their proximities using marker clusters and circles.
- Red markers indicate failures (class 0) and green markers represent successes (class 1).
- We calculated distances to nearby railways, highways, coastlines, and cities, using polylines to visualize these proximities.
- This analysis aimed to determine the proximity of launch sites to these features.

- GitHub URL of the completed SpaceX API calls notebook:

https://github.com/Melissa-Tynah/IBM-Applied-Data-Science-Capstone-Project./blob/main/Module%203.0_SpaceX_Visual%20analytics_Folium_TMN.ipynb

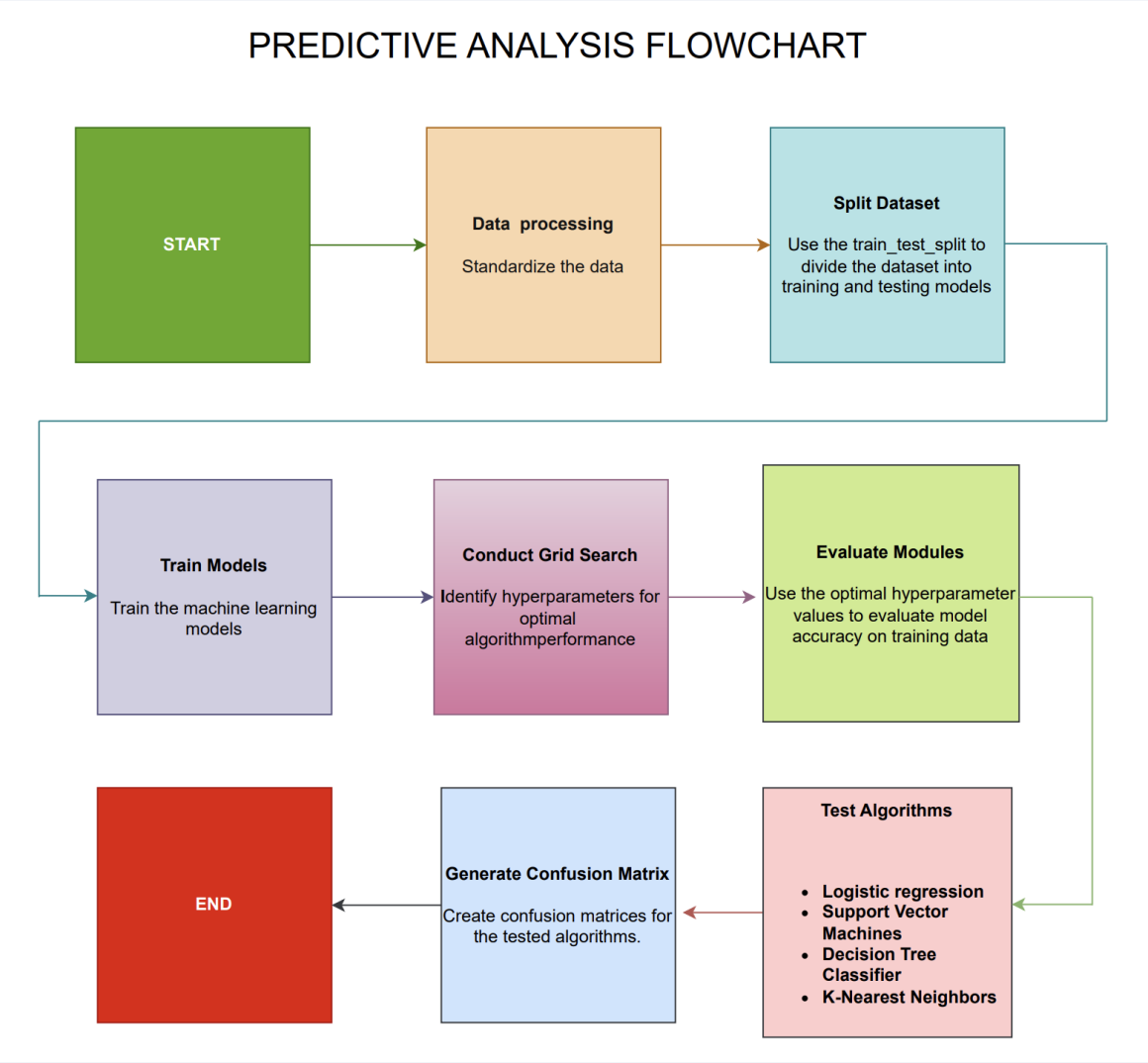
Build a Dashboard with Plotly Dash

- Using Plotly Dash, we developed an interactive dashboard.
- We created pie charts to display the sites with the highest number of successful launches.
- The scatter plot illustrated the payload ranges and F9 Booster versions, highlighting both the lowest and highest launch success rates, as well as identifying the F9 Booster version with the highest success rate.
- The GitHub URL of the completed SpaceX API calls notebook:

https://github.com/Melissa-Tynah/IBM-Applied-Data-Science-Capstone-Project./blob/717702f81c164eae64498ddade1053596e349936/Module%203.1_SpaceX_Dash_App_TMN.py

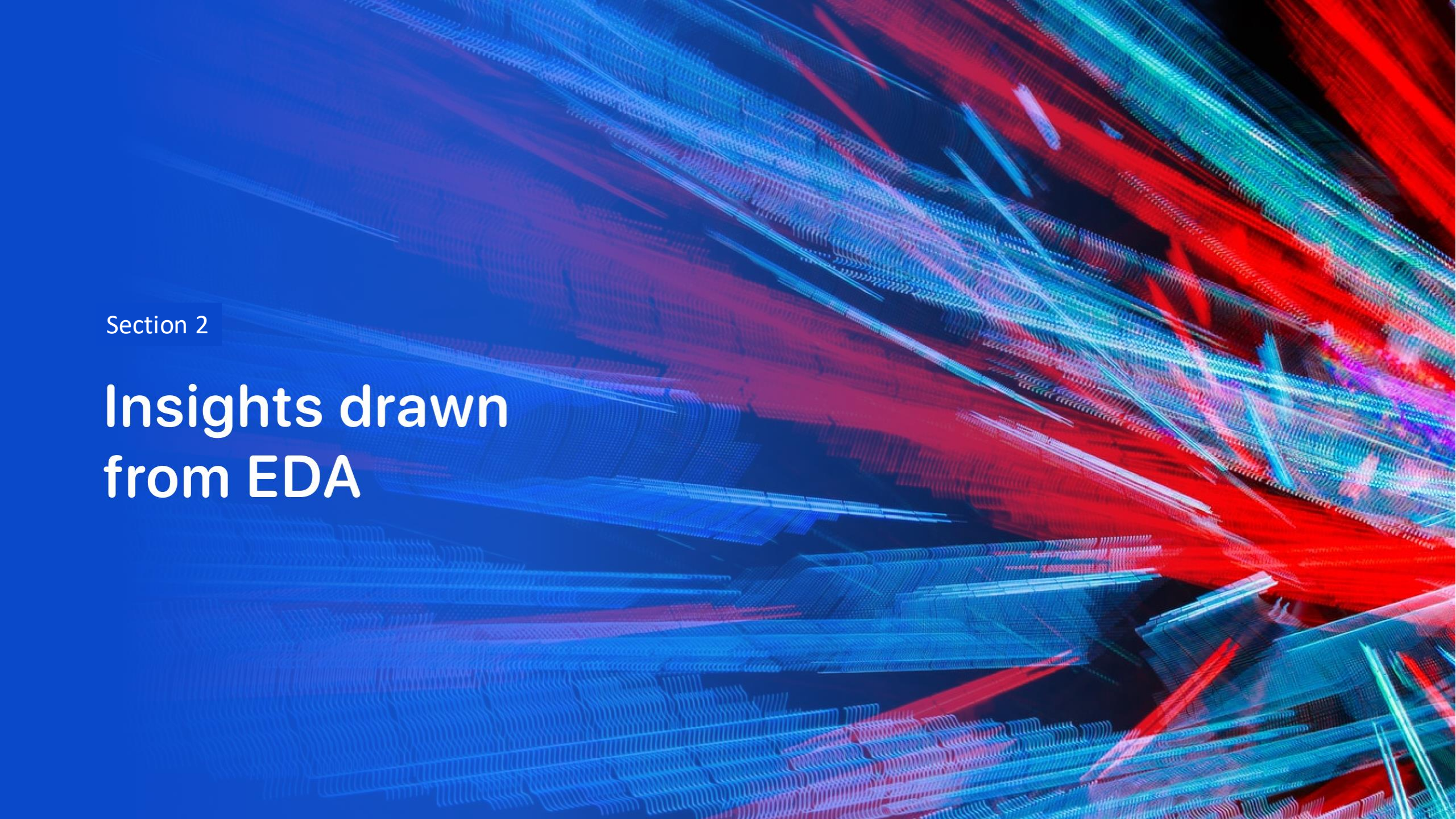
Predictive Analysis (Classification)

- We developed machine learning pipelines to predict the successful landing of the first stage of the Falcon 9 rocket.
- First, we processed the data to standardize it and then used Train_test_split to divide our dataset into training and testing sets.
- Next, we trained our models and conducted a Grid Search to identify the hyperparameters that yielded the best algorithm performance.
- With the optimal hyperparameter values in hand, we evaluated which model achieved the highest accuracy using the training data.
- Finally, we tested several algorithms, including Logistic Regression, Support Vector Machines, Decision Tree Classifier, and K-Nearest Neighbors, and generated the corresponding confusion matrix.
- GitHub URL of the completed SpaceX API calls notebook: https://github.com/Melissa-Tynah/IBM-Applied-Data-Science-Capstone-Project./blob/717702f81c164eae64498ddade1053596e349936/Module%204_SpaceX_Machine%20Learning%20Prediction_TMN%20.ipynb

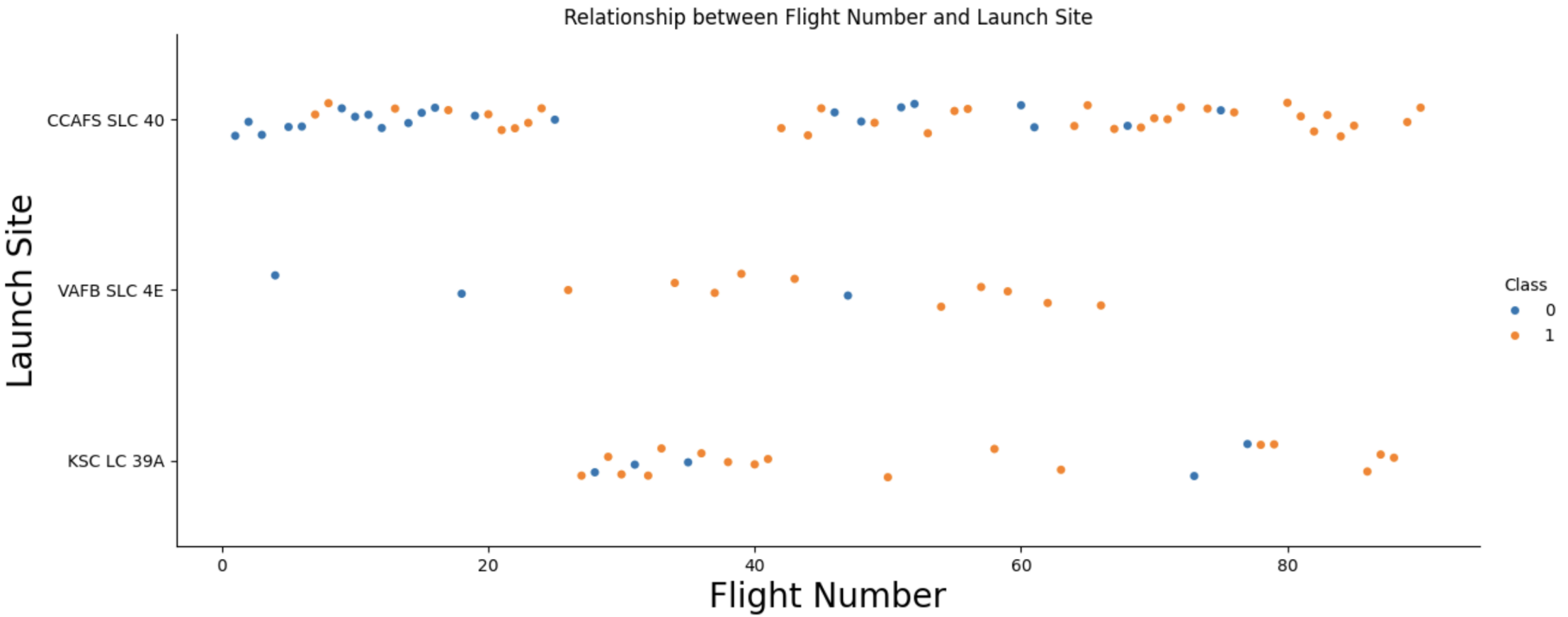


Results

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

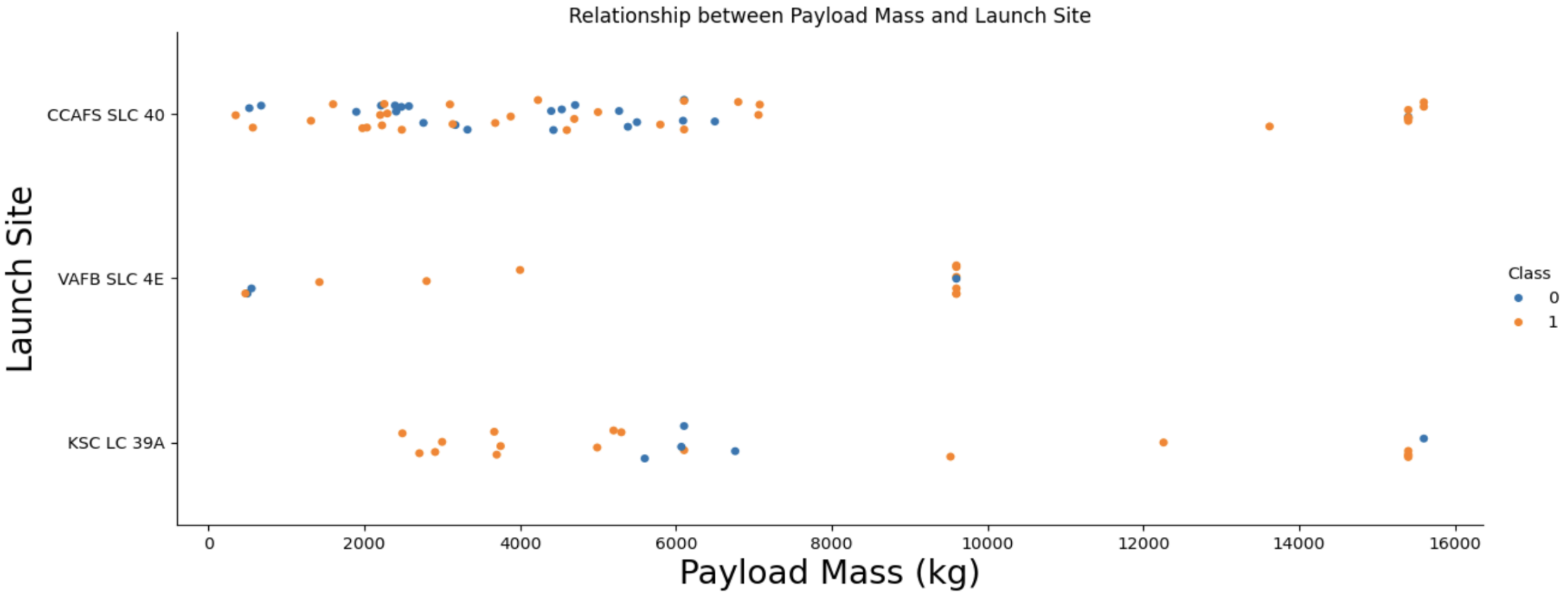

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



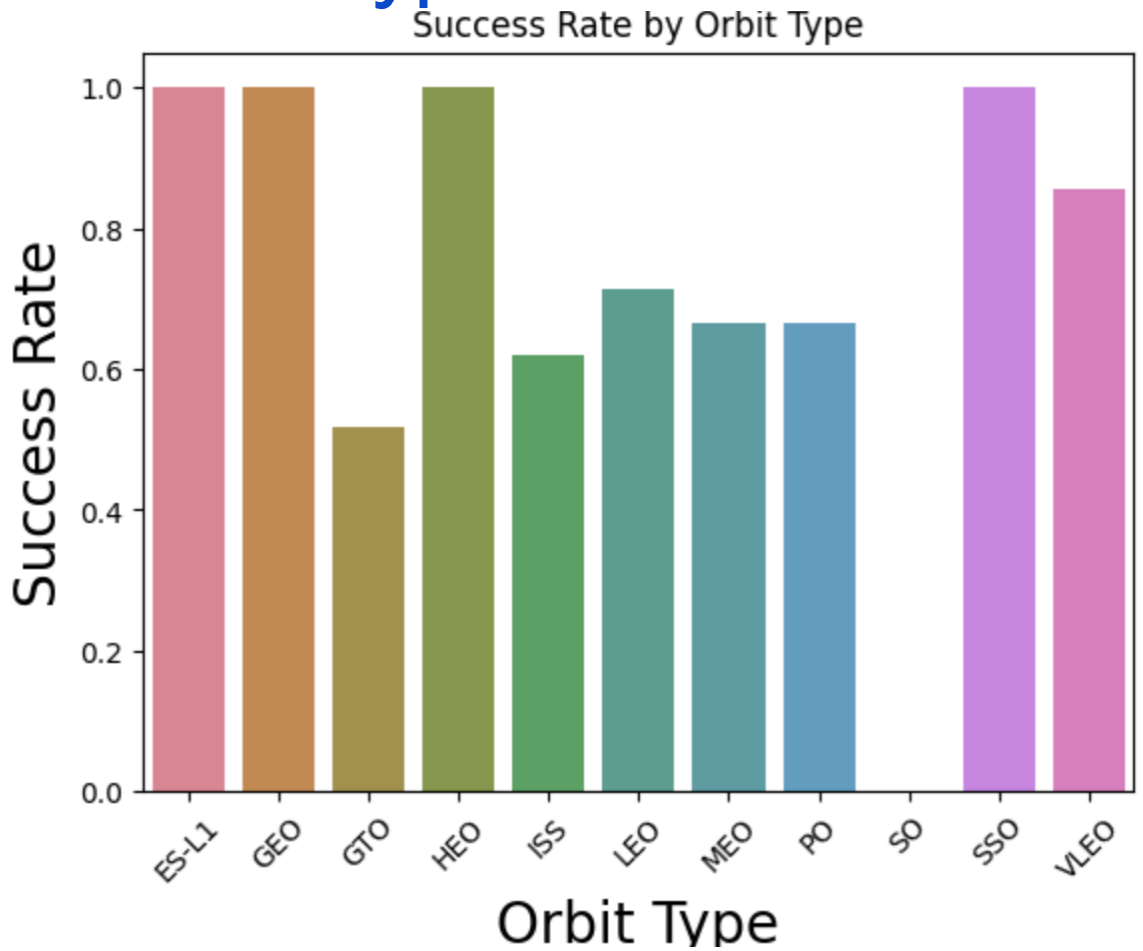
- Based on the plot, we observed that as the flight number increases, the launch success also tends to improve across all sites.
- The launch site showing the strongest correlation between flight numbers and success is CCAFS SLC 40.

Payload vs. Launch Site



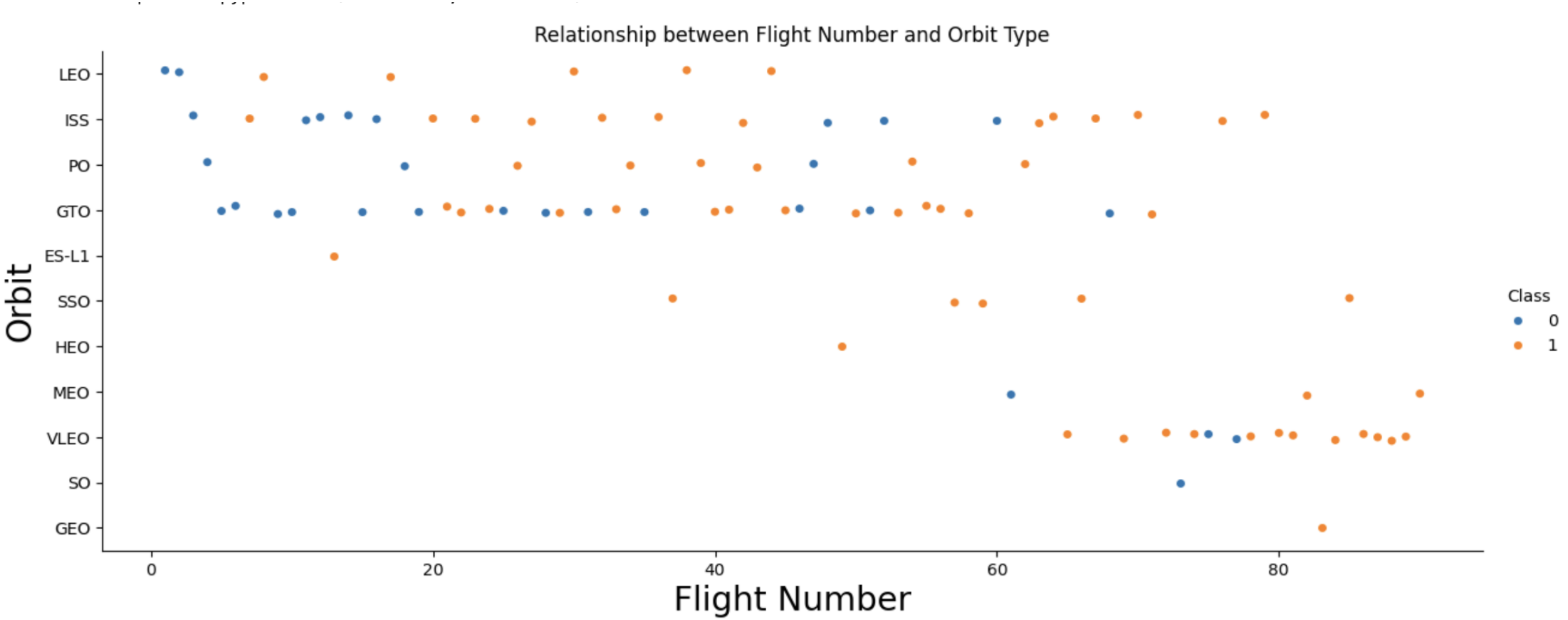
- The analysis of the plot revealed a strong positive relationship between payload mass and the launch site, particularly at CCAFS SLC 40.
- Notably, when the payload mass was 16,000 kg, there were successful launches at this site.
- In contrast, at lower payload masses ranging from 5,000 kg to 6,000 kg, both KSC LC 39A and CCAFS SLC 40 experienced the highest number of failed launches.

Success Rate vs. Orbit Type



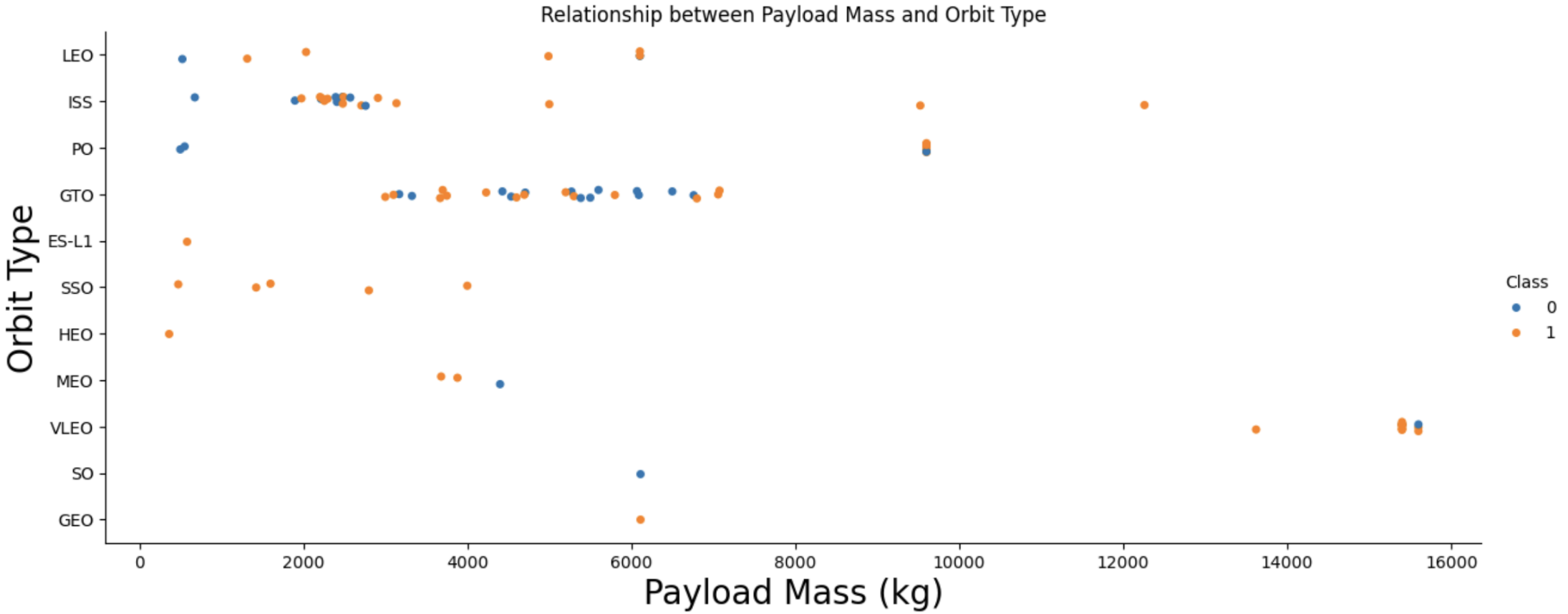
- The plot shows that ES-L1, GEO, HEO, SSO, and VLEO had the highest success rates, indicating that missions in these orbits are more likely to achieve their goals.
- This information could be crucial for future mission planning and resource allocation.

Flight Number vs. Orbit Type



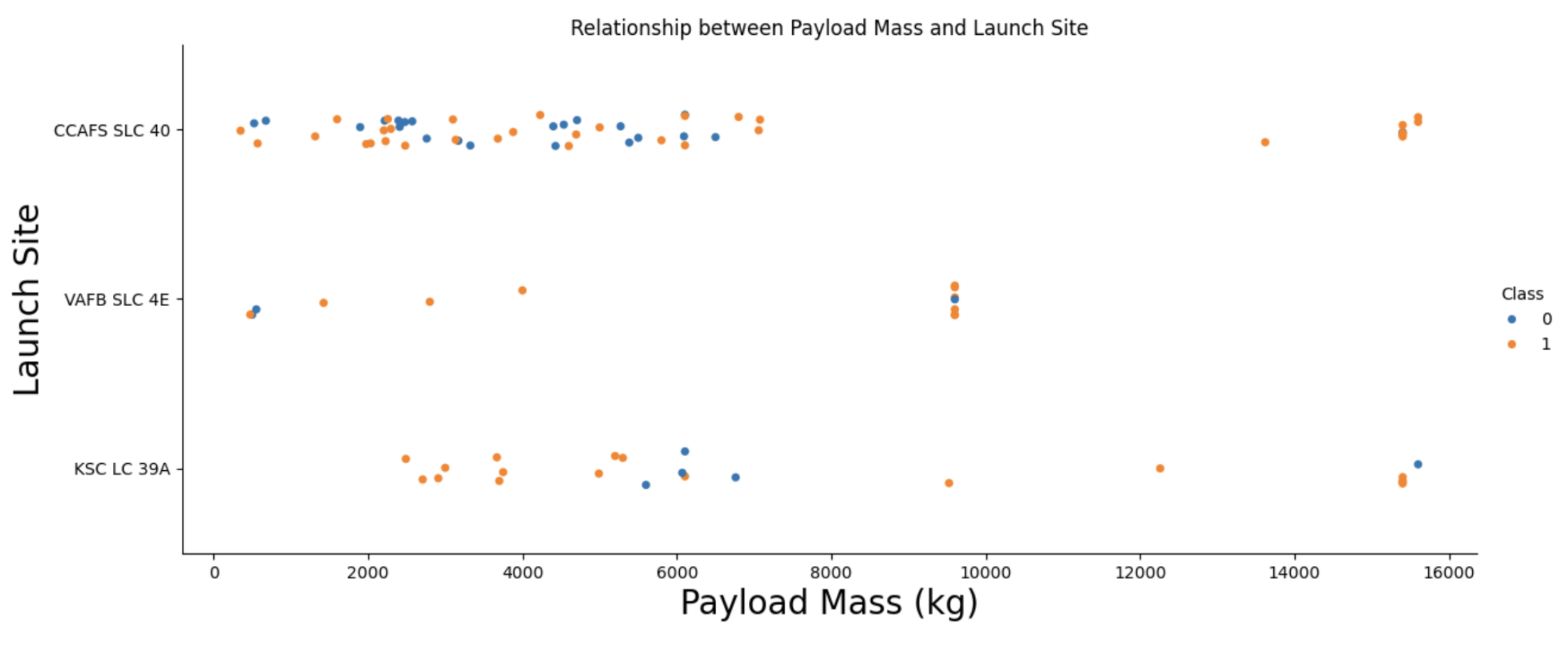
- The plot shows a strong positive correlation between Flight number and various orbit categories, including LEO, ISS, GTO, and VLEO.
- This indicates that as the Flight number increases, so does the activity in these orbits.

Payload vs. Orbit Type



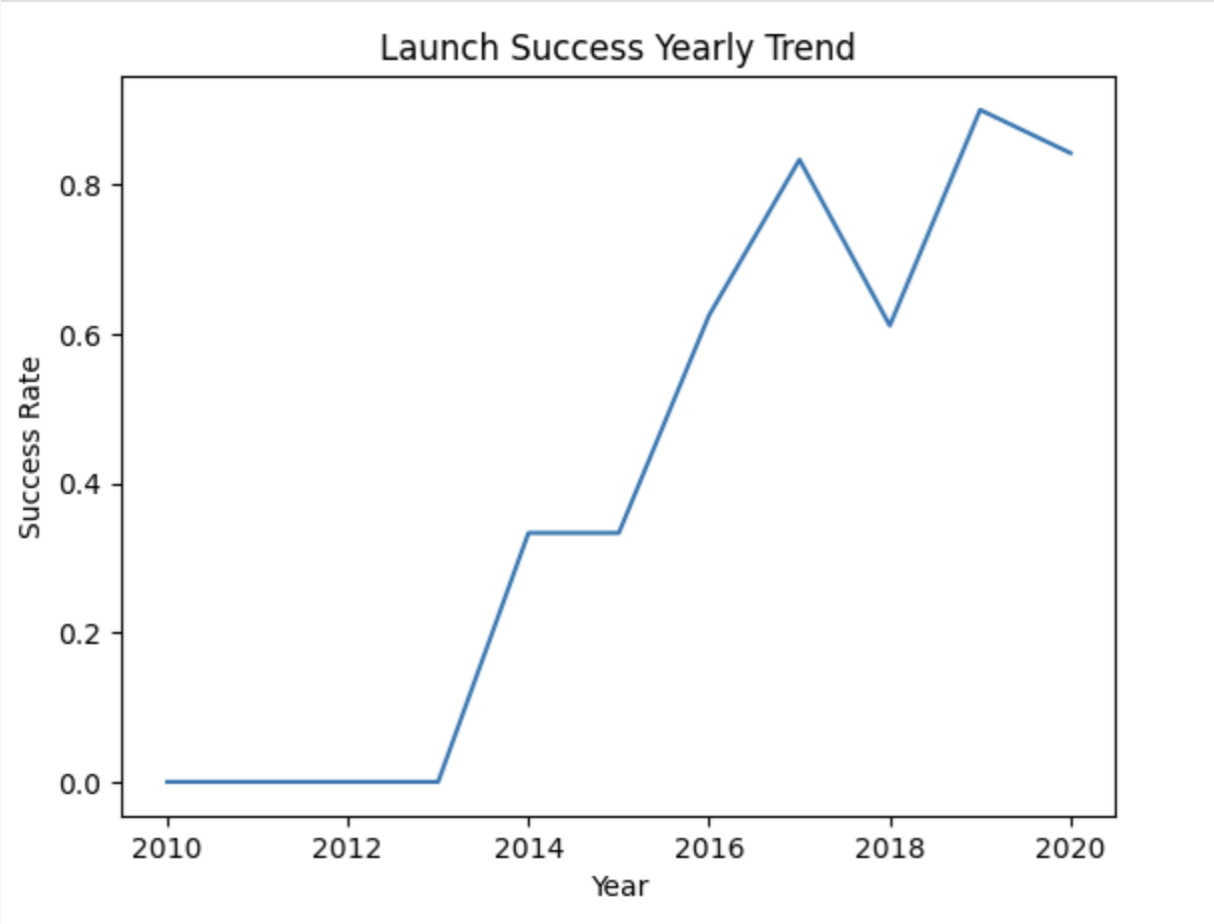
- Successful landings occur more frequently for heavy payloads in PO, LEO, VLEO, and ISS orbits.

Payload vs. Launch Site



- SpaceX's payload capacity varies by launch site.
- Cape Canaveral supports heavier payloads to geostationary orbits, while Vandenberg limits payloads for polar orbits.
- Local regulations and safety zones also impact maximum payloads.

Launch Success Yearly Trend



- Since 2013, SpaceX has steadily increased its launch success rate.
- Initially, the company had fewer missions, but by 2020, it was regularly conducting multiple successful launches each year, including crewed missions and satellite deployments, establishing itself as a leader in the aerospace industry.

All Launch Site Names

The names of unique launch sites in the space mission are:

i. CCAFS LC-40

Cape Canaveral Force Station Launch Complex 40

ii. VAFB SLC-4E

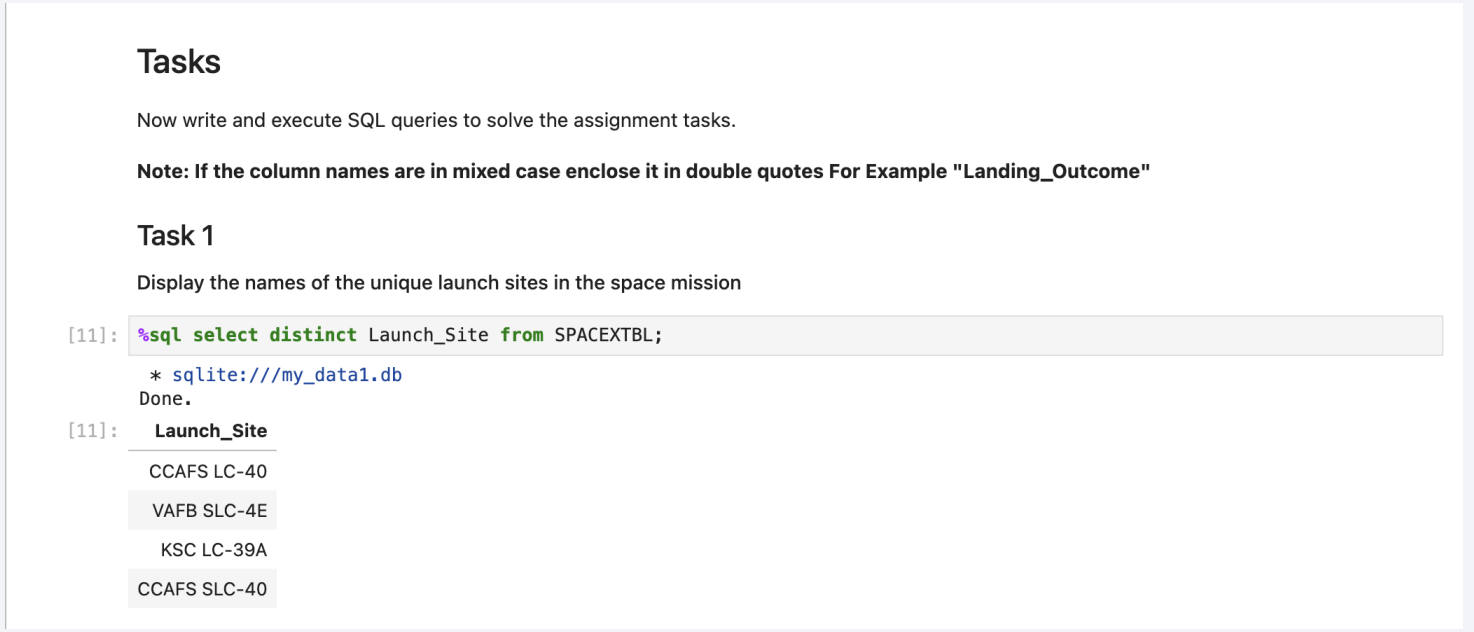
Vandenberg Air Force Base Space Launch Complex 4E

iii. KSC LC-39A

Kennedy Space Center Launch Complex 39A

iv. CCAFS SLC-40

Cape Canaveral Force Space Launch Complex 40



Tasks

Now write and execute SQL queries to solve the assignment tasks.

Note: If the column names are in mixed case enclose it in double quotes For Example "Landing_Outcome"

Task 1

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

```
[11]: %sql select distinct Launch_Site from SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

[11]:

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

Launch Site Names Begin with 'CCA'

Five records where launch sites begin with `CCA` are:

i. CCAFS LC-40

ii. CCAFS LC-40

iii. CCAFS LC-40

iv. CCAFS LC-40

v. CCAFS LC-40

Task 2

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

[12]:
```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5 ;
```

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

[12]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

Task 3

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

```
[13]: %sql select sum(PAYLOAD_MASS__KG_)from SPACEXTBL where Customer= 'NASA (CRS)';
```

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

[13]:

sum(PAYLOAD_MASS__KG_)
45596

- The total payload mass carried by boosters launched by NASA (CRS) was 45 596kg.

Average Payload Mass by F9 v1.1

Task 4

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

```
[14]: %sql select avg (PAYLOAD_MASS__KG_)from SPACEXTBL where Booster_Version='F9 v1.1';
```

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

[14]:

avg (PAYLOAD_MASS__KG_)
2928.4

- The average payload mass carried by boosters version F9 v1.1 was 2 928.4kg.

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

[15]:
```
%sql select min(Date) from SPACEXTBL where Landing_Outcome='Success (ground pad)';
```

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

[15]:

min(Date)
2015-12-22

- The first successful lading outcome in the ground pad was achieved on 22 December 2015.

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

```
[16]: %sql select distinct Booster_Version from SPACEXTBL where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_
```

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

[16]:

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

The names of boosters that have successfully landed on the drone ship, with a payload mass greater than 4000 kg but less than 6000 kg, are:

i. F9 FT B1022

ii. F9 FT B1026

iii. F9 FT B1021.2

iv. F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

[17]:
```
%sql select substr(Mission_Outcome,1,7) as Mission_Outcome, count(*) from SPACEXTBL group by 1;
```

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

[17]:

Mission_Outcome	count(*)
Failure	1
Success	100

There were 100 successful missions and 1 failure mission.

Boosters Carried Maximum Payload

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

[18]:
```
%sql SELECT DISTINCT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

[18]:

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

- We identified the booster that carried the maximum payload using a subquery in the WHERE clause along with the MAX() function.
- The booster versions are capable of carrying the maximum payload mass of 15,600 kg
- These versions are all designed to handle the specified payload capacity efficiently.

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.

```
%%sql • • •
```

FINAL ANSWER

[20]:
```
%%sql
SELECT
    CASE SUBSTR(DATE, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS Month_Name,
    LANDING_OUTCOME,
    BOOSTER_VERSION,
    LAUNCH_SITE
FROM SPACEXTBL
WHERE SUBSTR(DATE, 0, 5) = '2015'
  AND LANDING_OUTCOME = 'Failure (drone ship)';
```

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

[20]:

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

The failed landing attempts of the drone ship occurred in January and April 2015, originating from the CCAFS LC-40 launch site with booster versions F9 v1.1 B1012 and F9 v1.1 B1015.

2015 Launch Records

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.

[21]:
```
%sql select Landing_Outcome, count(*) from SPACEXTBL where Date between '2011-06-04' and '2017-03-20' group by Landing_Outcome order by 2 desc;
```

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

[21]:

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

From June 4, 2010, to March 20, 2017, there were a total of 8 successful landing outcomes (5 on the drone ship and 3 on the ground pad) and 5 failed outcomes from the drone ship.

Section 3

Launch Sites Proximities Analysis

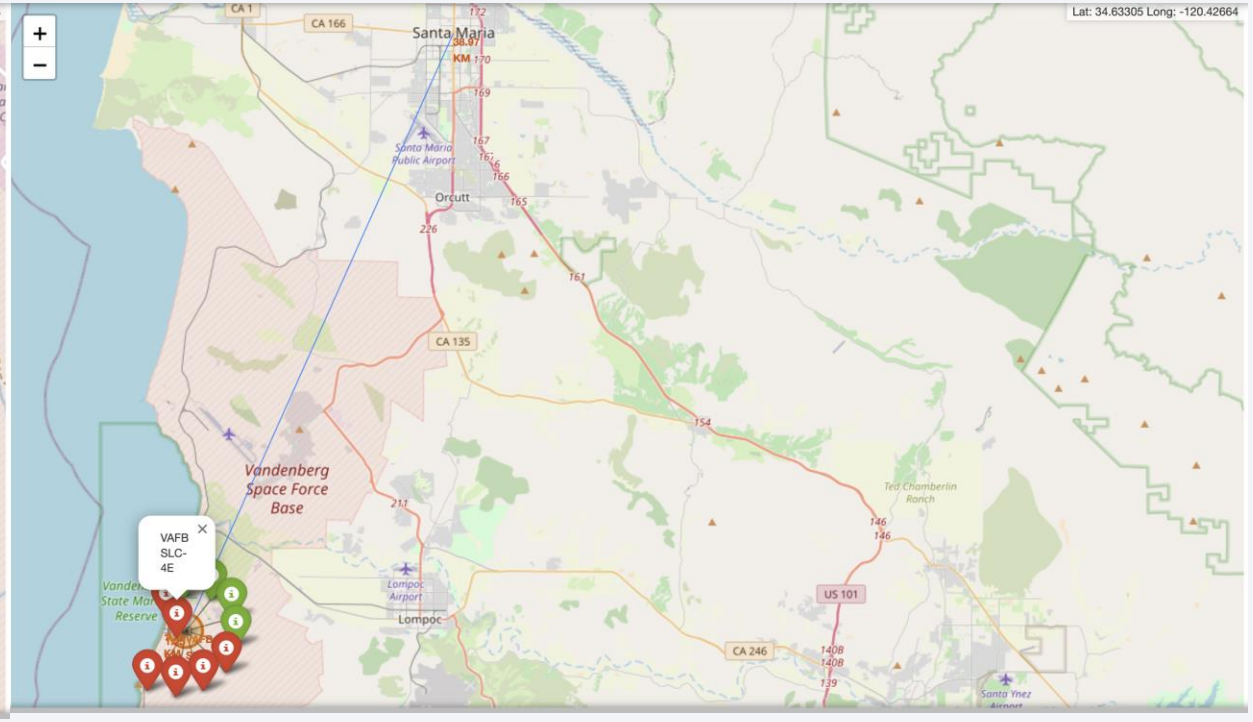
Launch sites Location Markers

VAFB SL-4E is the only launch site located in Western USA, specifically in California. The Kennedy Space Center Launch Complex 39A, Cape Canaveral Space Force Station Launch Complex 40, and Cape Canaveral Space Force Station Space Launch Complex 40 are all situated close to one another in Florida. Additionally, all these launch sites are positioned very near the coastline.

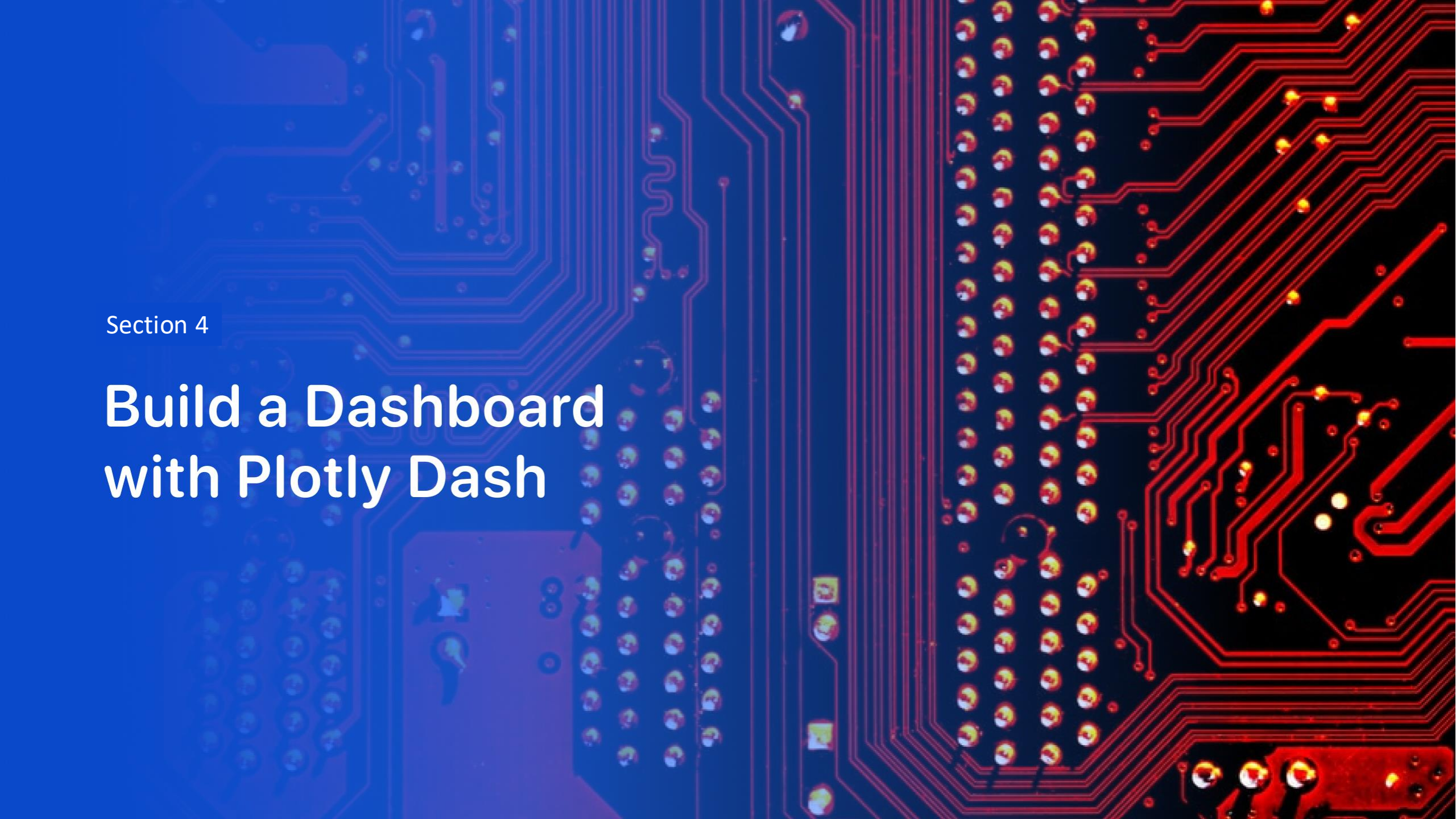
Success/Failed launches for each site on the map

- KSC LC 39A had the highest rocket launch success.
- The site with the least launch success rate was CCAFS SLC-40.
- The launch site CCAFS LC-40 had the most total launches.

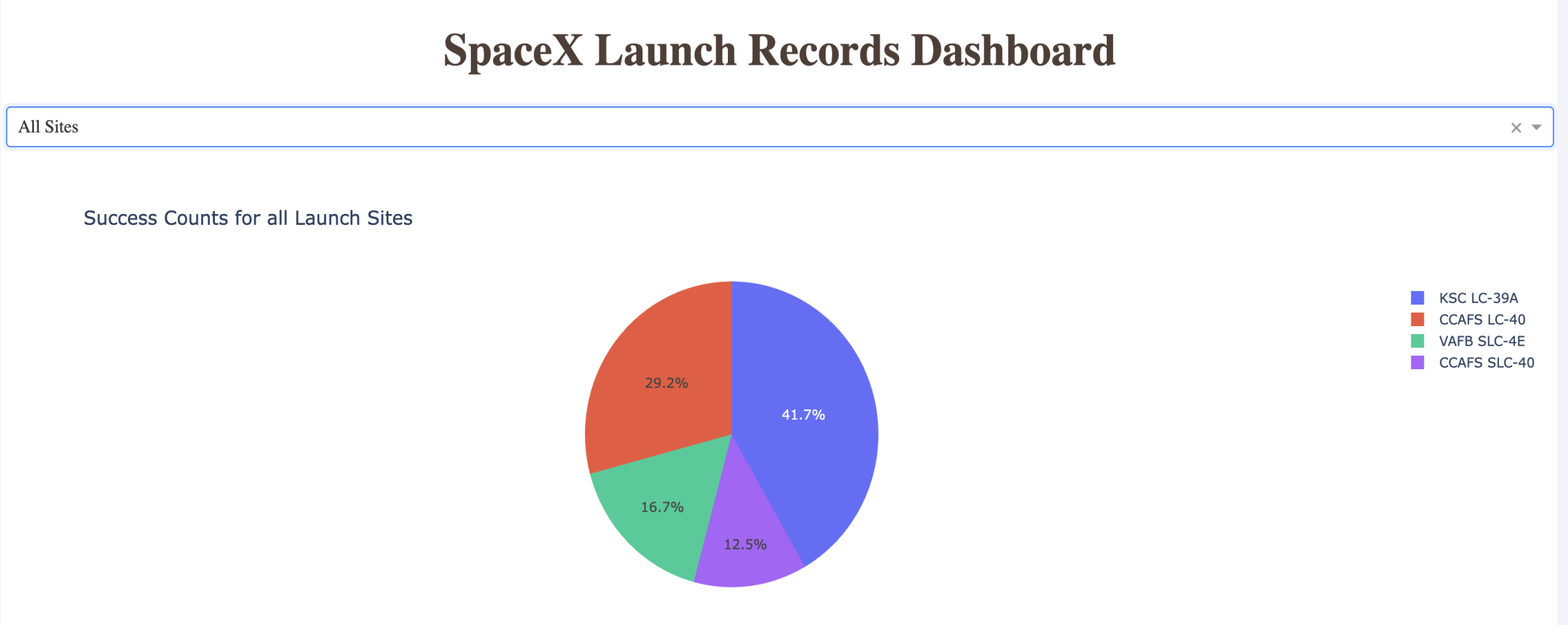
Launch sites Markers



- Vandenberg Space Launch complex 4 is very close to the highway (1.25km), railway (1.29km) and the city of Santa Maria (38.97km) as shown in the diagrams above.

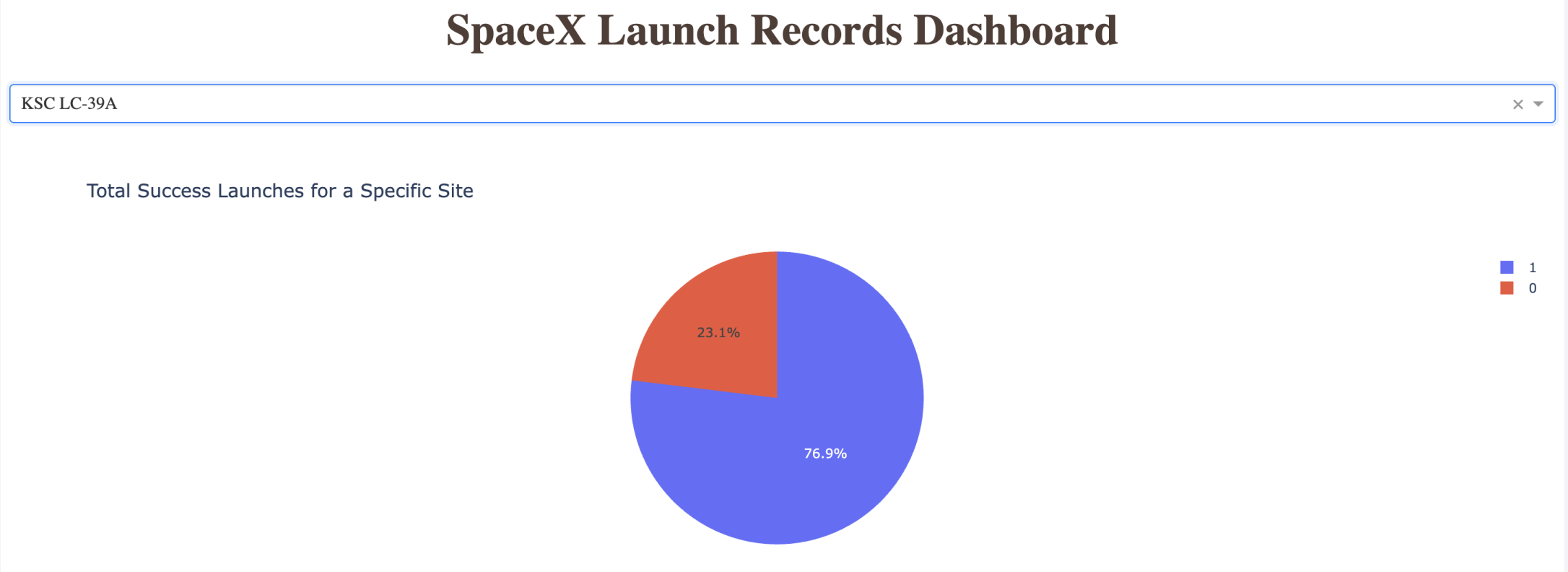

Section 4
Build a Dashboard with Plotly Dash

Pie Chart showing Launch Success Count for All Sites



- KSC LC-39A launch site achieved the highest success count, while VAFB SLC-4E had the lowest.

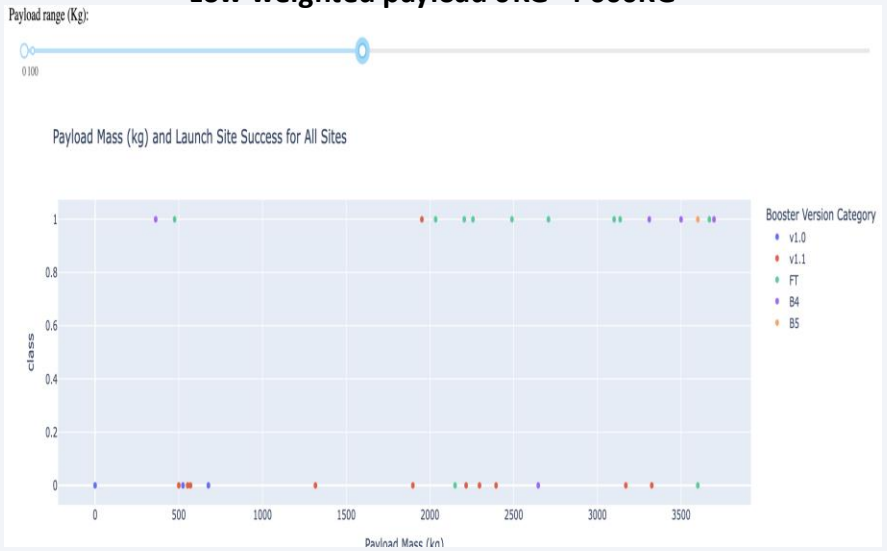
Specific Site with Highest Launch Success Ratio



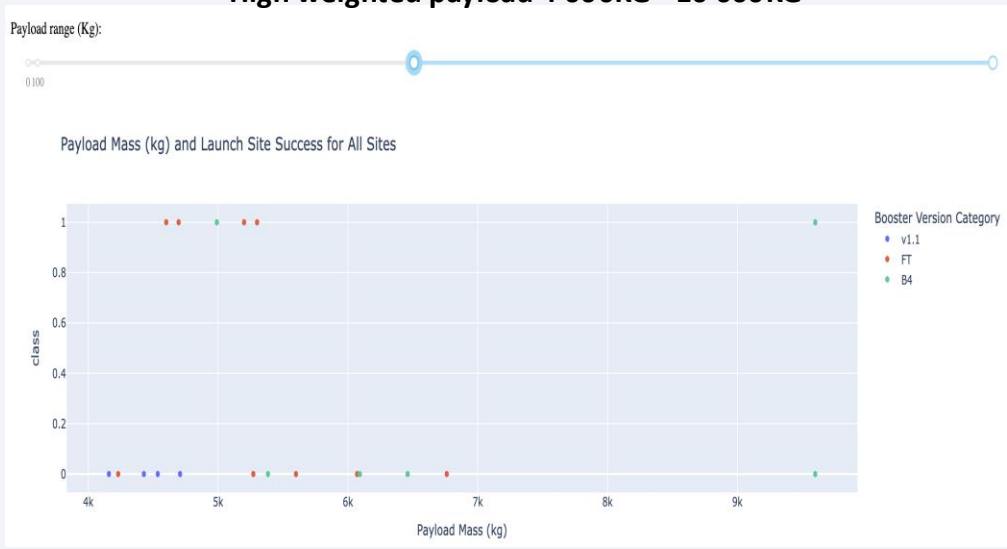
- The site with the highest launch success ratio was KSC LC-39A.

Payload vs Launch Sites Scatter Plot for All Sites

Low weighted payload 0KG- 4 000KG



High weighted payload 4 000KG - 10 000KG



The booster version category with the highest launch success rate was FT, while booster version v1.1 had the highest failed launch rate.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

TASK 12

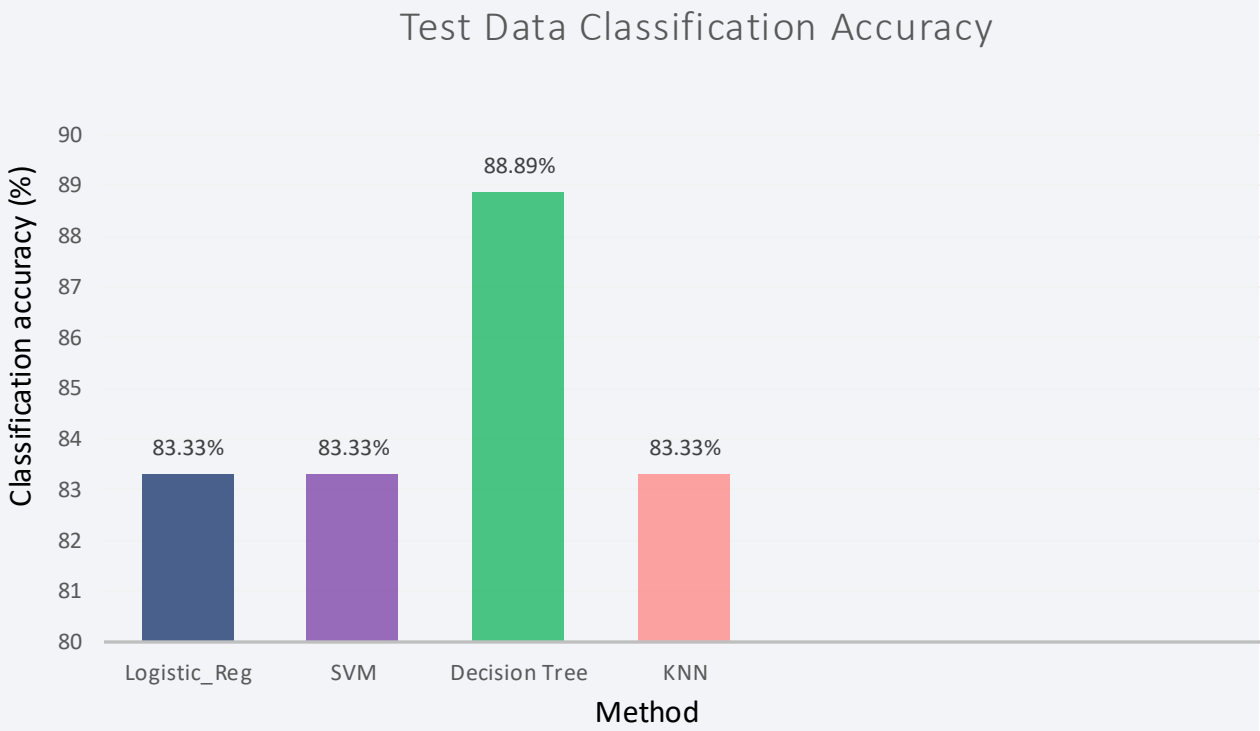
Find the method performs best:

```
[34]: Report = pd.DataFrame({'Method' : ['Test Data Accuracy']})

knn_accuracy=knn_cv.score(X_test, Y_test)
Decision_tree_accuracy=tree_cv.score(X_test, Y_test)
SVM_accuracy=svm_cv.score(X_test, Y_test)
Logistic_Regression=logreg_cv.score(X_test, Y_test)

Report['Logistic_Reg'] = [Logistic_Regression]
Report['SVM'] = [SVM_accuracy]
Report['Decision Tree'] = [Decision_tree_accuracy]
Report['KNN'] = [knn_accuracy]

Report.transpose()
```

[34]:

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.888889
KNN	0.833333

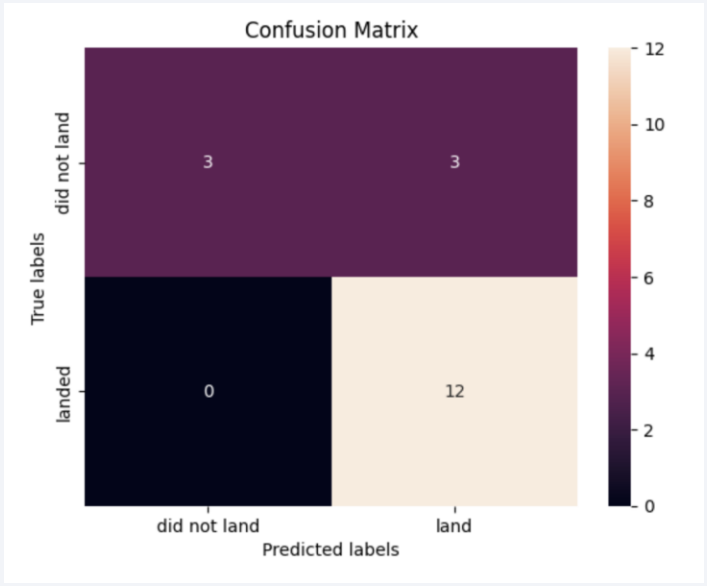
After comparing accuracy of above methods, they all preformed practically the same, except for tree which fit train data slightly better but test data worse.

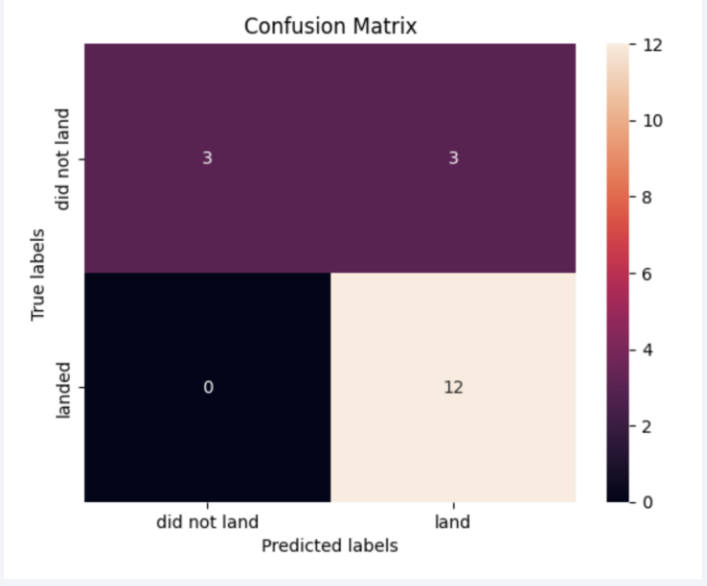


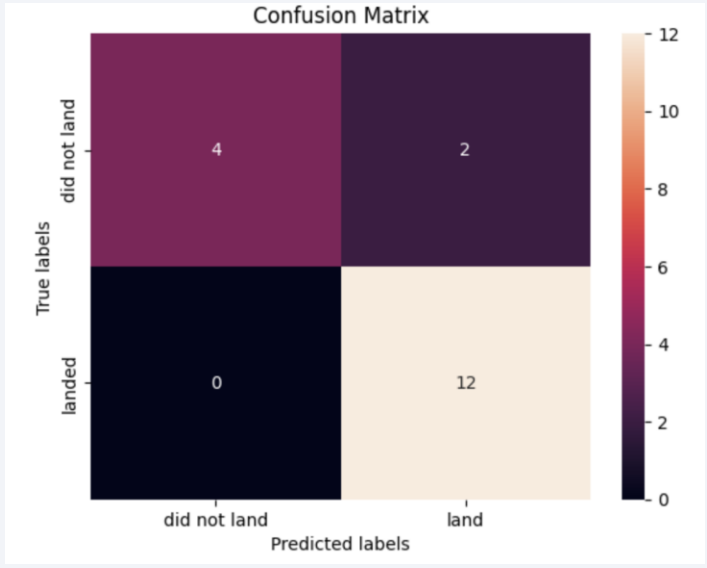
- The decision tree model achieved the highest classification accuracy(88.89%).
- In comparison, logistic regression, SVM, and KNN all performed equally, with each attaining an accuracy of 83.33%.

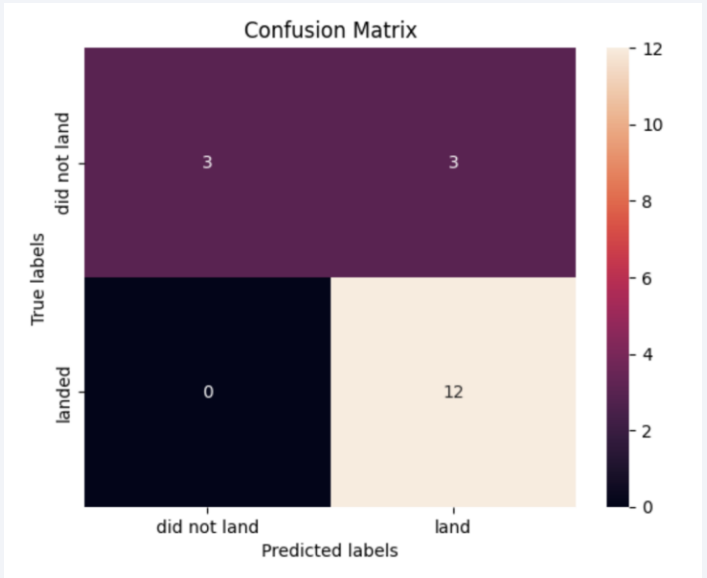
Confusion Matrix

- The analysis highlights a notable issue with false positives in all matrices, particularly in predicting the 1st stage booster landing.
- While the classifier performs well overall, it struggles with accurately identifying successful landings, as shown by incorrect predictions in the test samples.
- Improving this aspect could enhance the model's reliability and effectiveness.









Conclusions

- The analysis of SpaceX launch data reveals a clear correlation between the volume of flights conducted at a launch site and the associated success rates. Specifically, as the number of launches at a site increases, the likelihood of successful missions also rises. This trend has been particularly evident from 2013 to 2020, a period during which SpaceX significantly enhanced its launch success rates, reflecting improvements in technology, experience, and operational procedures.

- Among the various orbital destinations, orbits such as Earth-Sun L1 (ES-L1), Geostationary Orbit (GEO), Highly Elliptical Orbit (HEO), Sun-Synchronous Orbit (SSO), and Very Low Earth Orbit (VLEO) demonstrated notably high success rates. This suggests that certain orbits may be more favorable for achieving successful launches, potentially due to factors like mission complexity, payload characteristics, and launch vehicle compatibility.

- When evaluating launch sites, Kennedy Space Center's Launch Complex 39A (KSC LC-39A) stood out as the most successful in terms of the number of successful launches. This site has become synonymous with pivotal missions, often serving as a launching pad for significant projects, and highlighting its strategic importance in SpaceX's operations.

- To effectively analyze and predict launch outcomes, a decision tree classifier has proven to be the most adept machine learning algorithm for this task. This algorithm is particularly effective in discerning patterns and relationships within the data, helping to categorize the factors that contribute to successful launches and enabling more accurate forecasting for future missions.

- In summary, the findings illustrate the interplay between launch frequency and success rates, the prominence of certain orbits in achieving successes, the exceptional track record of specific launch sites, and the utility of machine learning in enhancing predictive analytics in the aerospace domain.

Appendix

- Notebooks to to recreate dataset, analysis, and models can be found in repository:

https://github.com/Melissa-Tynah/IBM-Applied-Data-Science-Capstone-Project./tree/main

- Acknowledgements

  I would like to express my heartfelt gratitude to Joseph Santarcangelo and all instructors/educators at IBM for their dedicated efforts in creating such an insightful course and high-quality materials.

  I am also thankful to Coursera for making this valuable learning opportunity accessible online. Your contributions are trulyappreciated!

Thank you!