IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Omar Ait Bouhmad
November 2nd, 2025

Outline

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

Executive Summary

- Summary of methodologies

The project collected SpaceX data using the SpaceX API and also by web scraping historical launch records. The data was cleaned and prepared for analysis. Exploratory data analysis was done using SQL, and the results were visualized with Python, Pandas, and Matplotlib. Launch sites were studied using interactive maps and charts with Folium and Plotly Dash. Finally, machine learning was used to predict rocket landing outcomes.

- Summary of all result

The EDA results included interactive visual analytics and dashboards to explore the data. In addition, predictive analysis was performed using classification techniques to make forecasts based on the data.

Introduction

- Project background and context

  SpaceX advertises Falcon 9 rocket launches on its website at a cost of 62 million dollars, while other providers charge over 165 million dollars. Much of this saving comes from SpaceX reusing the first stage. If we can predict whether the first stage will land successfully, we can estimate the launch cost. This information can help other companies decide if they want to compete with SpaceX for a rocket launch.

- Problems you want to find answers

In this capstone, we will use Falcon 9 launch data from the SpaceX website to predict whether the rocket's first stage will land successfully.

Section 1

Methodology

Methodology

Executive Summary

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

Data Collection

- Describe how data sets were collected.

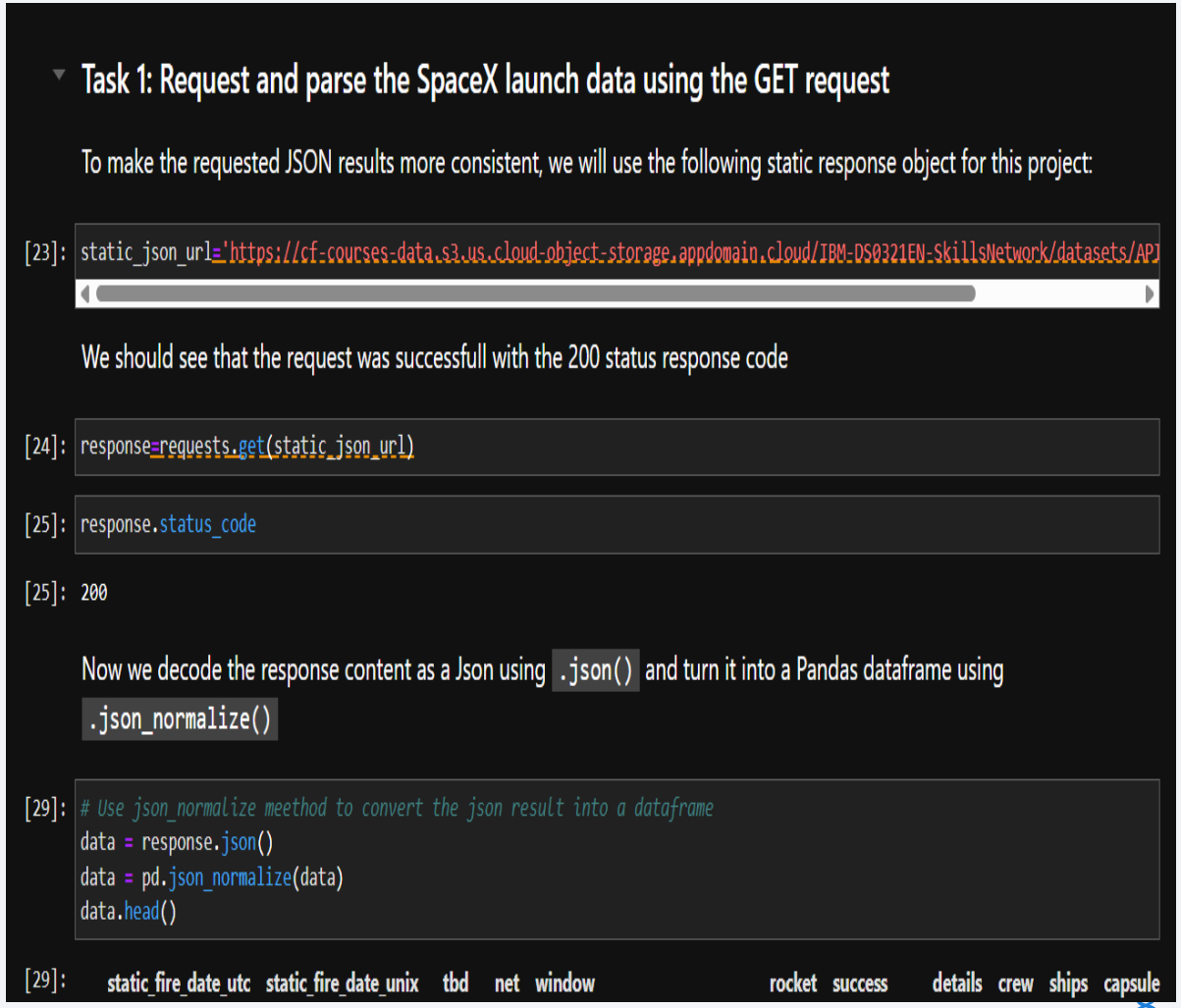
The data was first collected using the SpaceX API, which is a RESTful API, by sending GET requests. Helper functions were created to extract information from the API using launch identification numbers, and then the rocket launch data was requested from the API URL. The JSON response was decoded and converted into a Pandas DataFrame for easier use.

In addition, web scraping was used to collect historical Falcon 9 launch records from the Wikipedia page *List of Falcon 9 and Falcon Heavy launches*. Using the `BeautifulSoup` and `requests` libraries, the HTML table with launch records was extracted, parsed, and converted into a Pandas DataFrame.

Data Collection – SpaceX API

- Data was collected from the SpaceX API by sending a GET request. The response was decoded as JSON and then converted into a Pandas DataFrame for analysis.

- Here is the GitHub URL of the completed SpaceX API calls notebook
- https://github.com/om2044/Data-Science-capstone-SpaceX



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:

```
[23]: 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

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

[25]: response.status_code

[25]: 200
```

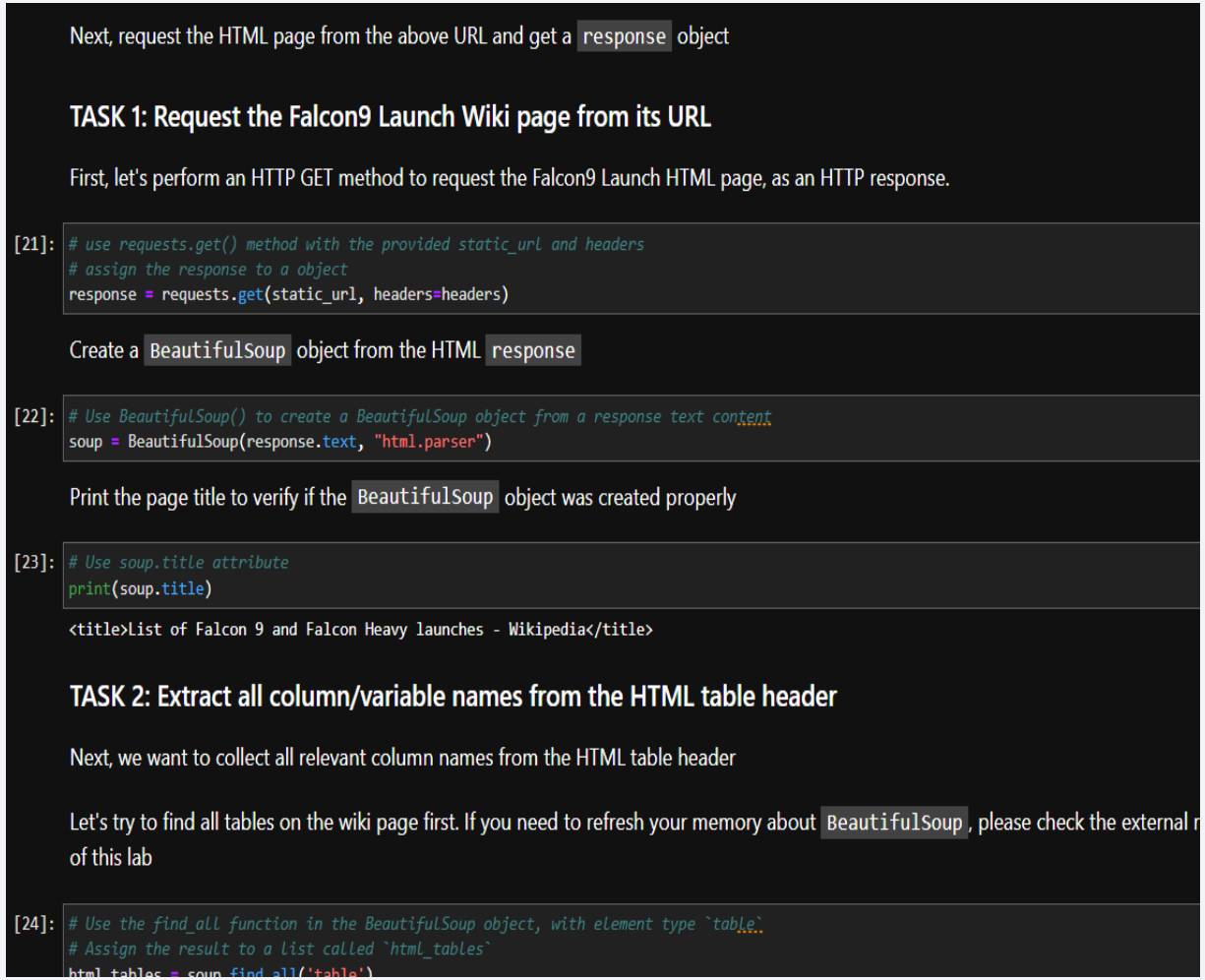
Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
[29]: # Use json_normalize meethod to convert the json result into a dataframe
      data = response.json()
      data = pd.json_normalize(data)
      data.head()

[29]:   static_fire_date_utc  static_fire_date_unix  tbd  net  window  rocket  success  details  crew  ships  capsule
```

Data Collection - Scraping

- Web scraping was used to collect historical Falcon 9 launch records from Wikipedia. Using `BeautifulSoup` and `requests`, the HTML table with launch data was extracted, parsed, and converted into a Pandas DataFrame.

- https://github.com/om2044/Data-Science-capstone-SpaceX



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.

[21]:
```
# use requests.get() method with the provided static_url and headers
# assign the response to a object
response = requests.get(static_url, headers=headers)
```

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

[22]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, "html.parser")
```

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

[23]:
```
# 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

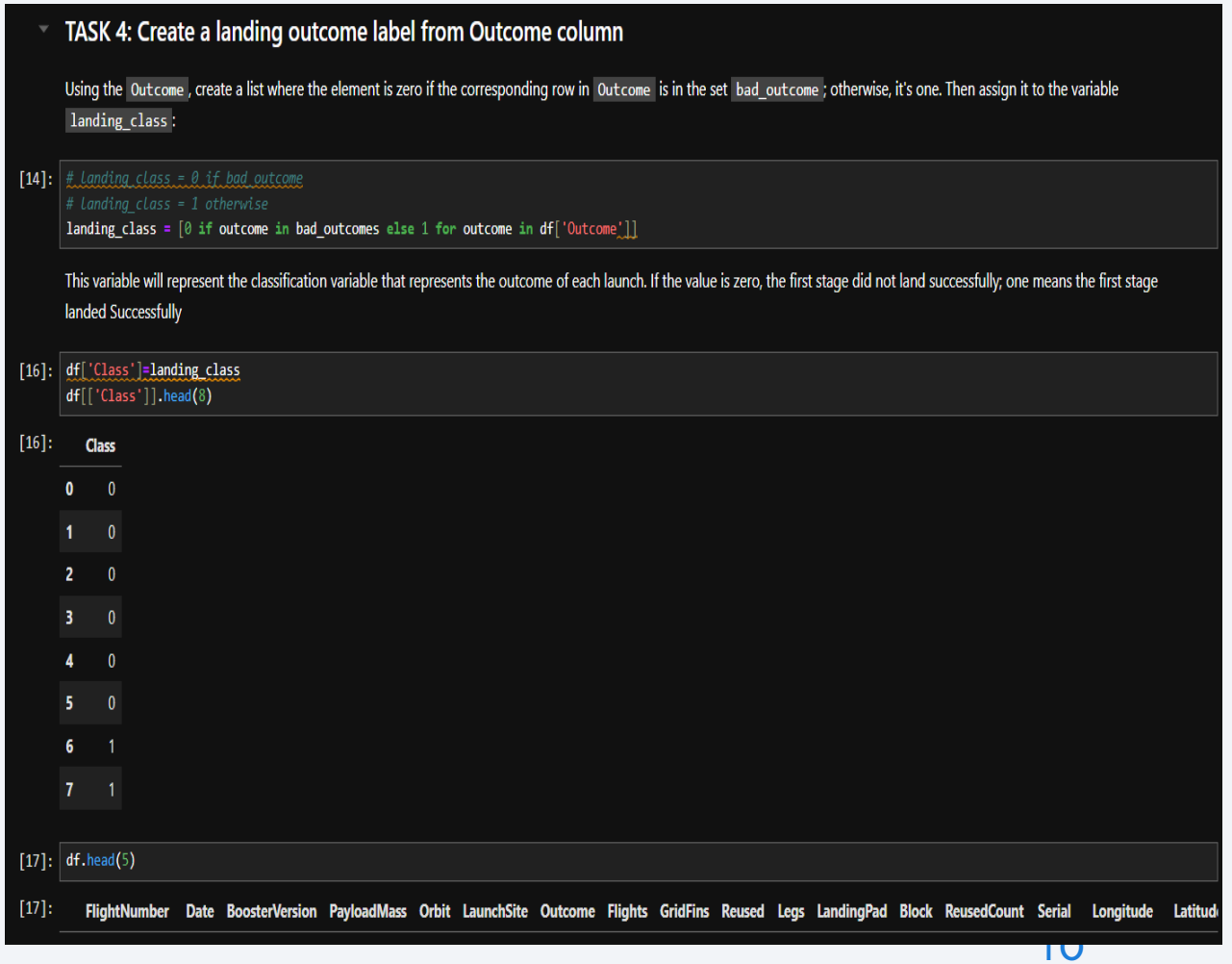
Next, we want to collect all relevant column names from the HTML table header

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 r
of this lab

[24]:
```
# 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')
```

Data Wrangling

After creating a Pandas DataFrame from the collected data, we filtered it using the `BoosterVersion` column to keep only Falcon 9 launches. Missing values in the `LandingPad` and `PayloadMass` columns were handled, replacing missing `PayloadMass` values with the column's mean. We also performed some exploratory data analysis (EDA) to find patterns in the data and decide on the label for training supervised models.



TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

[14]:
```
# Landing_class = 0 if bad_outcome
# Landing_class = 1 otherwise
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

[16]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

[16]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

[17]:
```
df.head(5)
```

[17]:

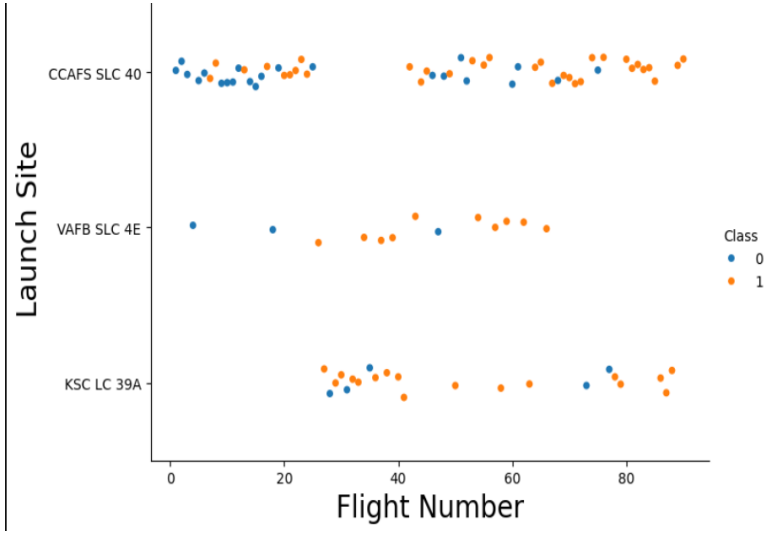
FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitud

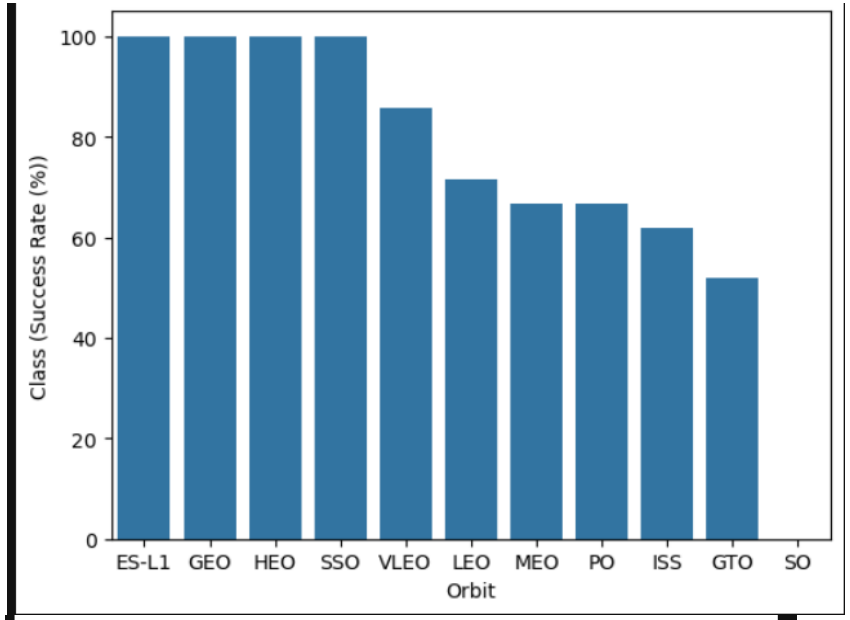
EDA with Data Visualization

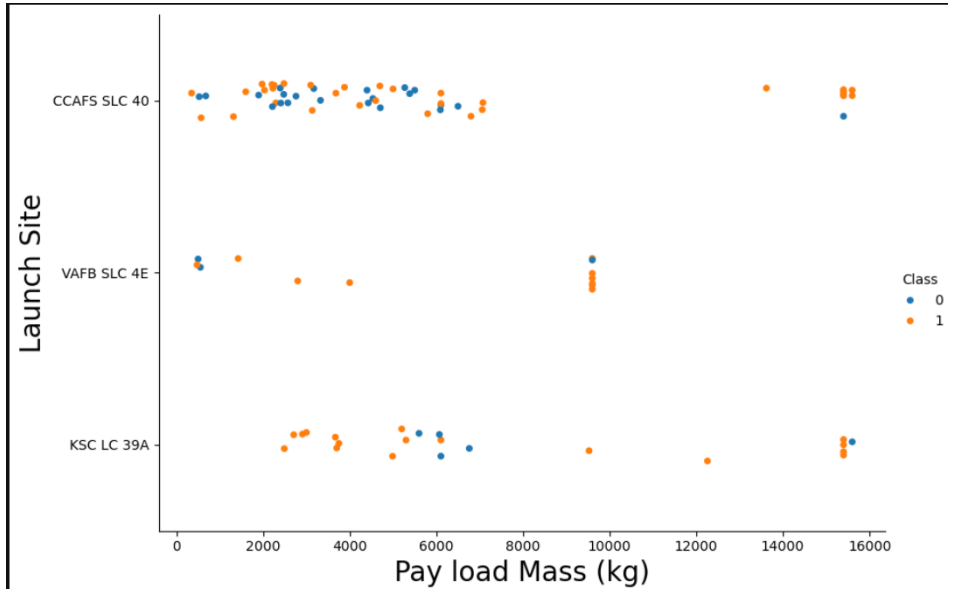
- Data analysis and feature engineering were performed using Pandas and Matplotlib, including:
- Data analysis and feature engineering were performed using Pandas and Matplotlib, including:
- Exploratory Data Analysis (EDA)
- Preparing data and feature engineering
- Scatter plots to show relationships between Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit type, and Payload and Orbit type
- Bar charts to show the success rate for each orbit type
- Line plots to show the yearly trend of launch success

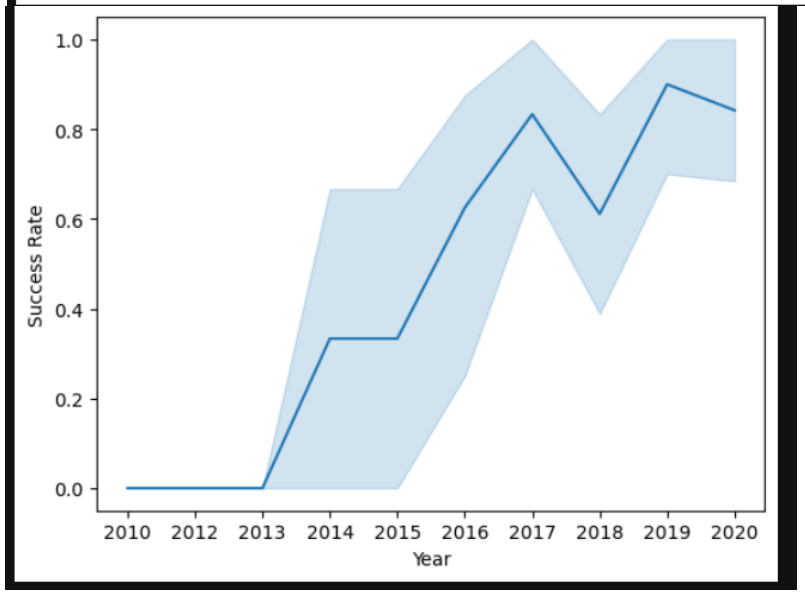
- https://github.com/om2044/Data-Science-capstone-SpaceX

EDA with Data Visualization (charts...)









EDA with SQL

Task 1

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

```
[10]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE ;
```

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

Task 2

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

```
]: %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

Task 3

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

```
3]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)';
```

Task 4

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

```
]: %sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

Task 5

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

Hint:Use min function

```
22]: %sql SELECT MIN(DATE) FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';
```

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

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

Build an Interactive Map with Folium

- A Folium map was created to mark all launch sites. Markers, circles, and lines were added to show the success or failure of launches at each site. A launch outcome column was also created, with failure = 0 and success = 1.

- https://github.com/om2044/Data-Science-capstone-SpaceX

Build a Dashboard with Plotly Dash

- An interactive dashboard was built using Plotly Dash. It included a drop-down menu to select the launch site and a callback function to display a success pie chart based on the selected site. A range slider was added to choose the payload range, and another callback function was used to show the success–payload scatter plot.

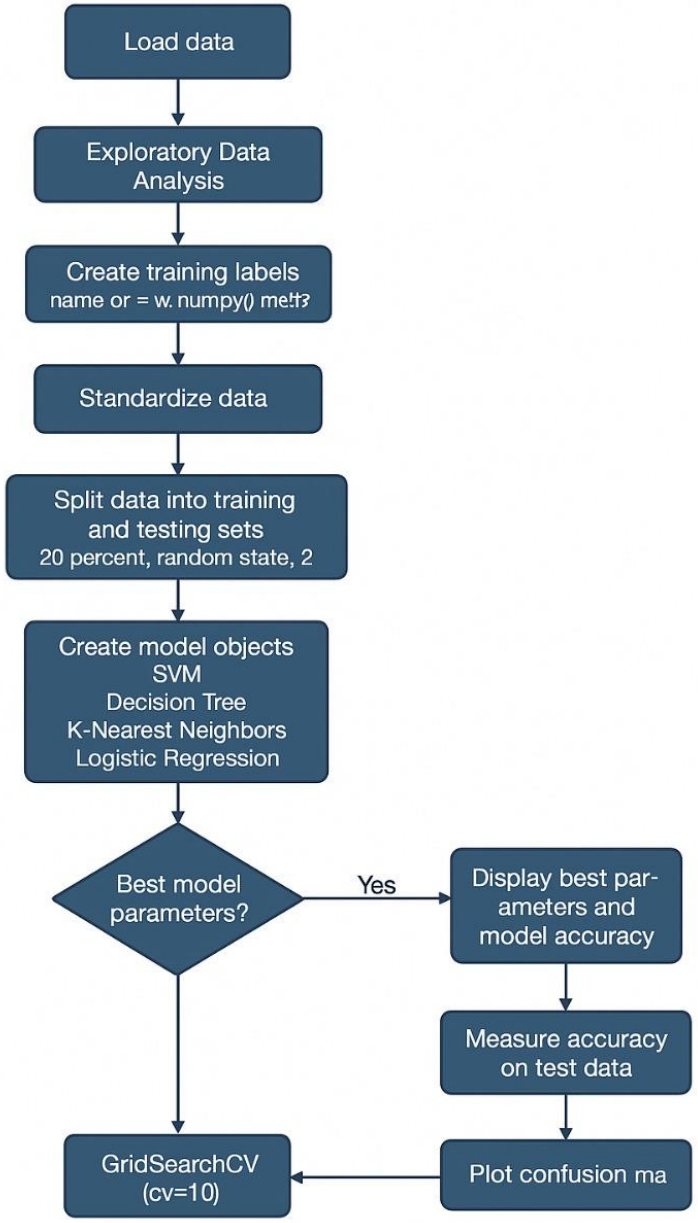
- https://github.com/om2044/Data-Science-capstone-SpaceX

Predictive Analysis (Classification)

- To build, evaluate, and improve the classification models, the data was first loaded into a Pandas DataFrame. Exploratory Data Analysis was done, and training labels were created by converting the **Class** column into a NumPy array using the `to_numpy()` method and storing it in the variable **Y**. The feature dataset (**X**) was standardized using the `StandardScaler()` function from Scikit-learn. The data was then split into training and testing sets using `train_test_split`, with 20% for testing and a random state of 2.
- To find the best-performing machine learning model among SVM, Decision Tree, K-Nearest Neighbors, and Logistic Regression, an object was created for each algorithm. A `GridSearchCV` object was then used with a set of parameters for each model, using 10-fold cross-validation (**cv=10**) to find the best hyperparameters. After training, the best parameters were displayed with `best_params_`, and model accuracy was checked with `best_score_`. Finally, the `score` method was used to measure accuracy on the test data, and a confusion matrix was plotted for each model to compare their predictions.

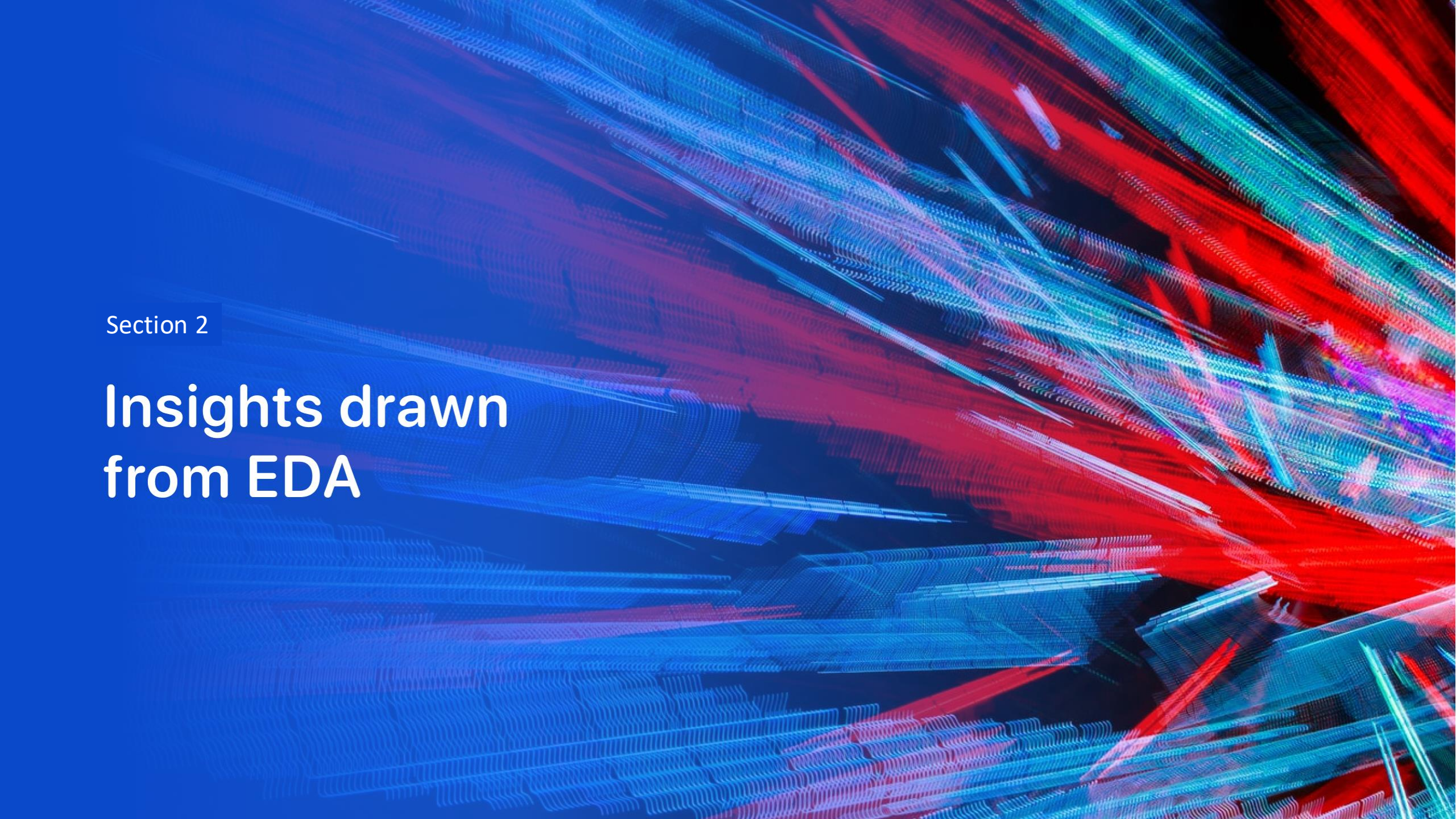
- https://github.com/om2044/Data-Science-capstone-SpaceX

Flowchart

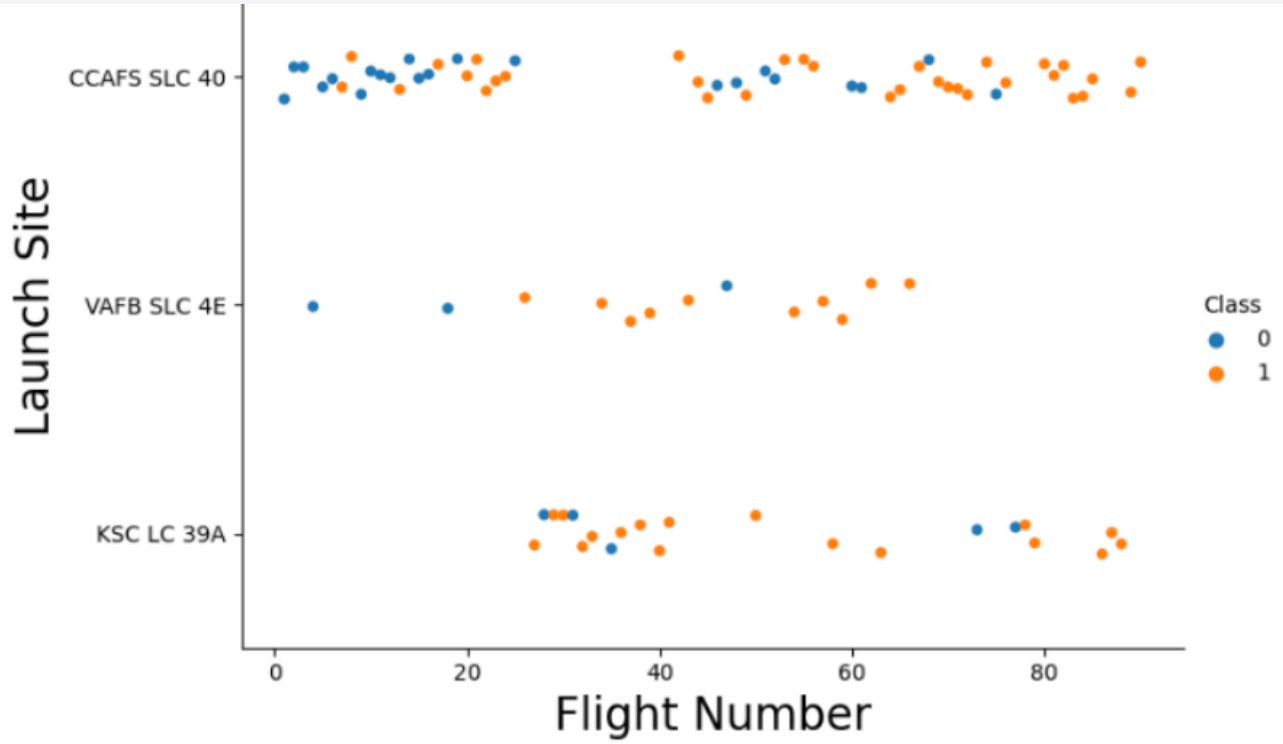

Load data
Exploratory Data Analysis
Create training labels
name or = w. numpy() me!t?
Standardize data
Split data into training and testing sets
20 percent, random state, 2
Create model objects
SVM
Decision Tree
K-Nearest Neighbors
Logistic Regression
Best model parameters?
Yes
Display best par-ameters and model accuracy
Measure accuracy on test data
Plot confusion ma
GridSearchCV (cv=10)

Results

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


Section 2
Insights drawn from EDA

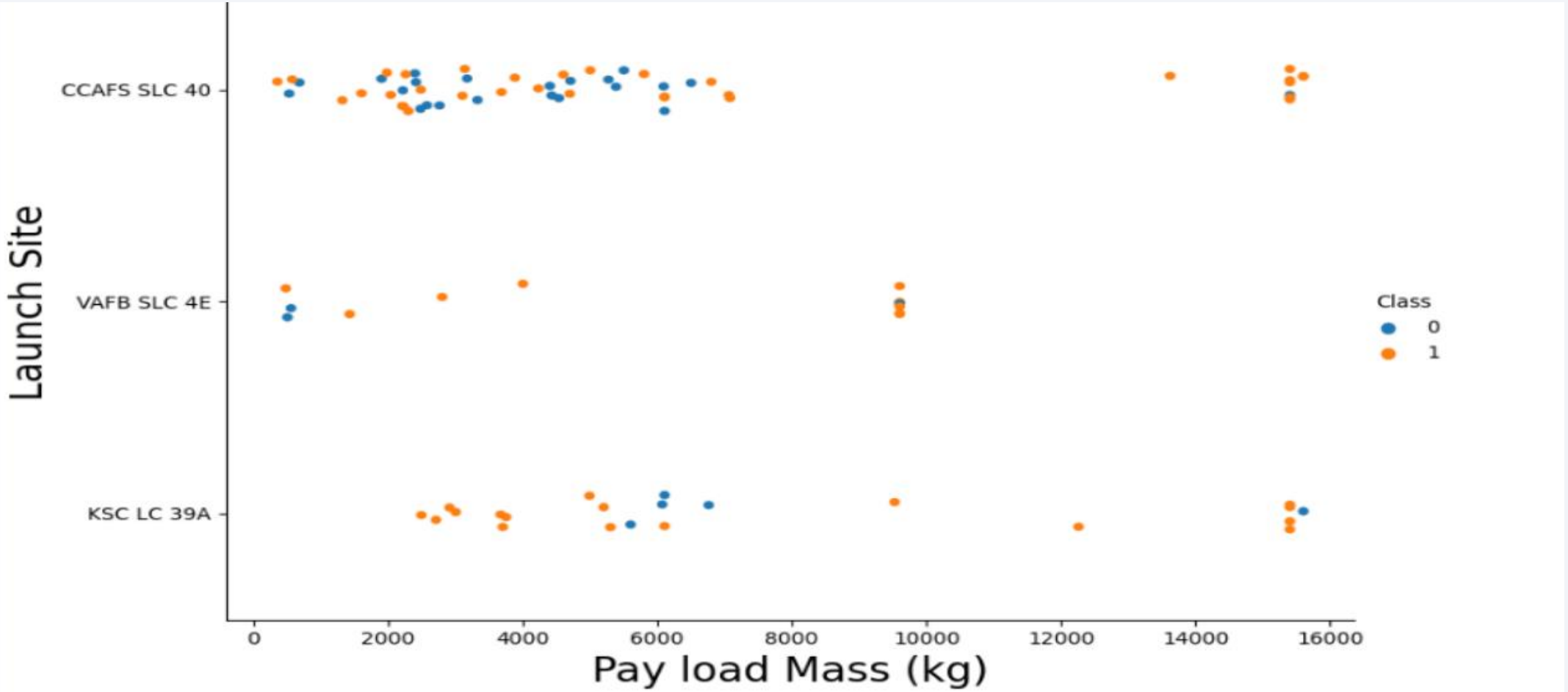
Flight Number vs. Launch Site



Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

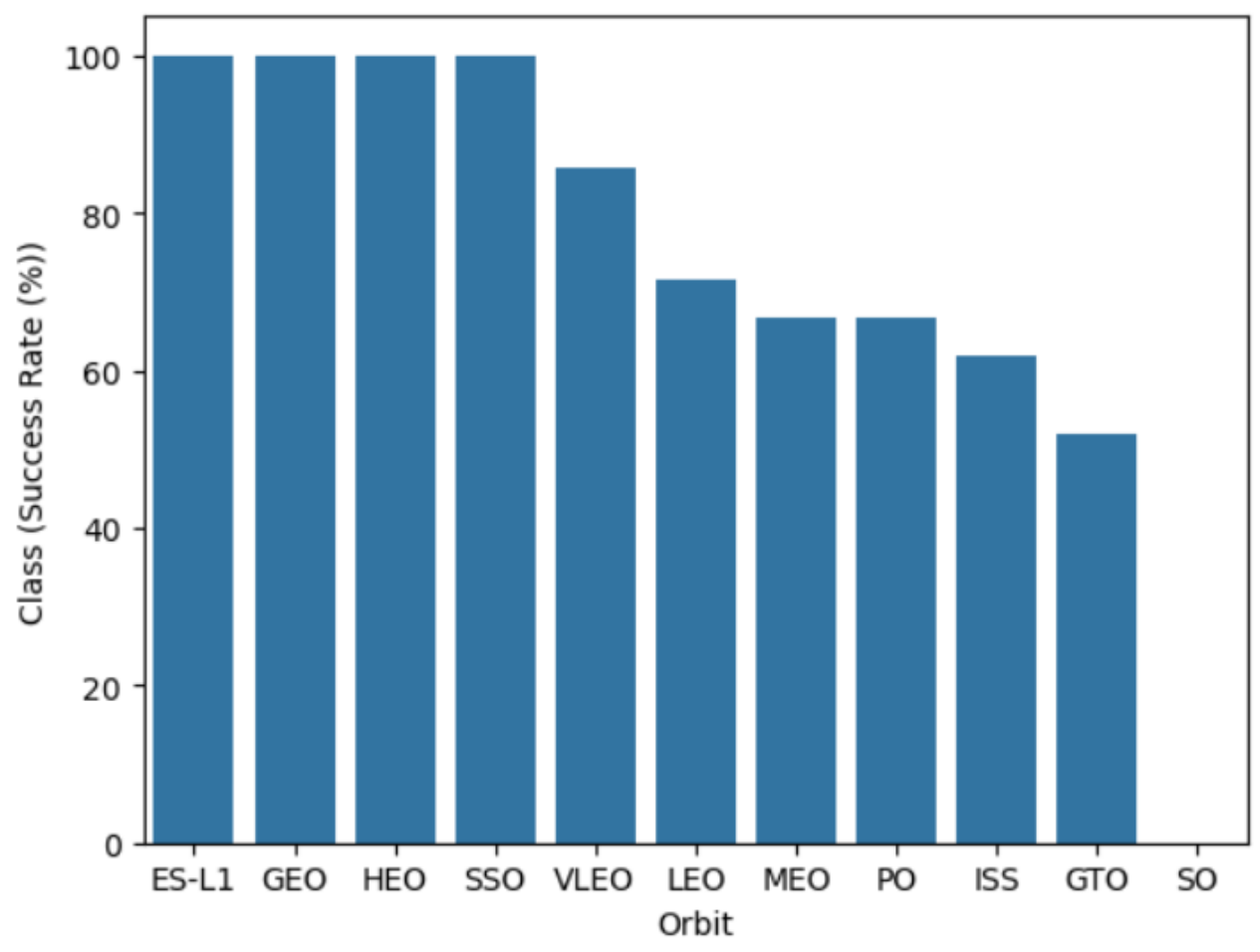
We can deduce that, as the flight number increases in each of the 3 launcg sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80th flight.

Payload vs. Launch Site



Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

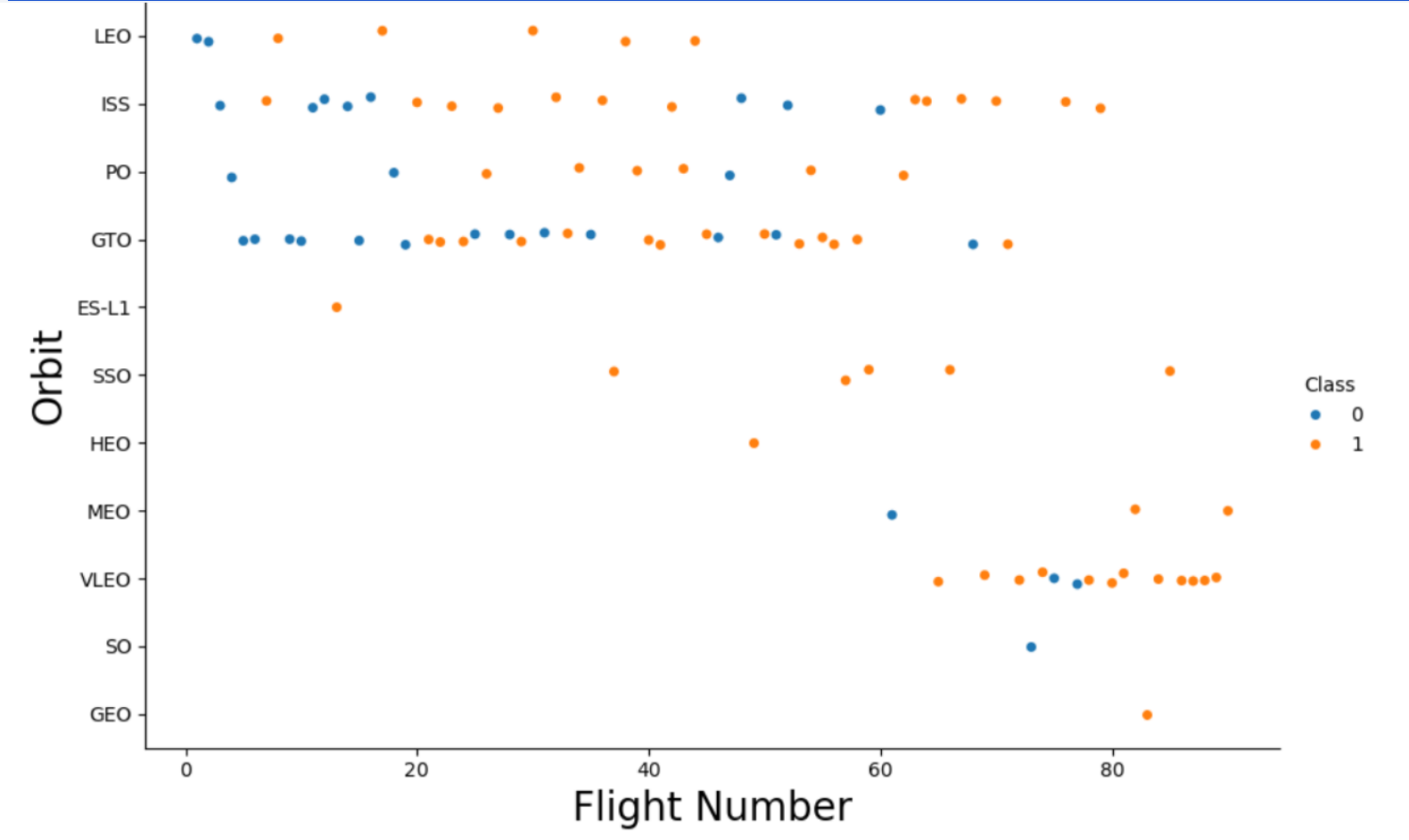
Success Rate vs. Orbit Type



Analyze the ploted bar chart try to find which orbits have high sucess rate.

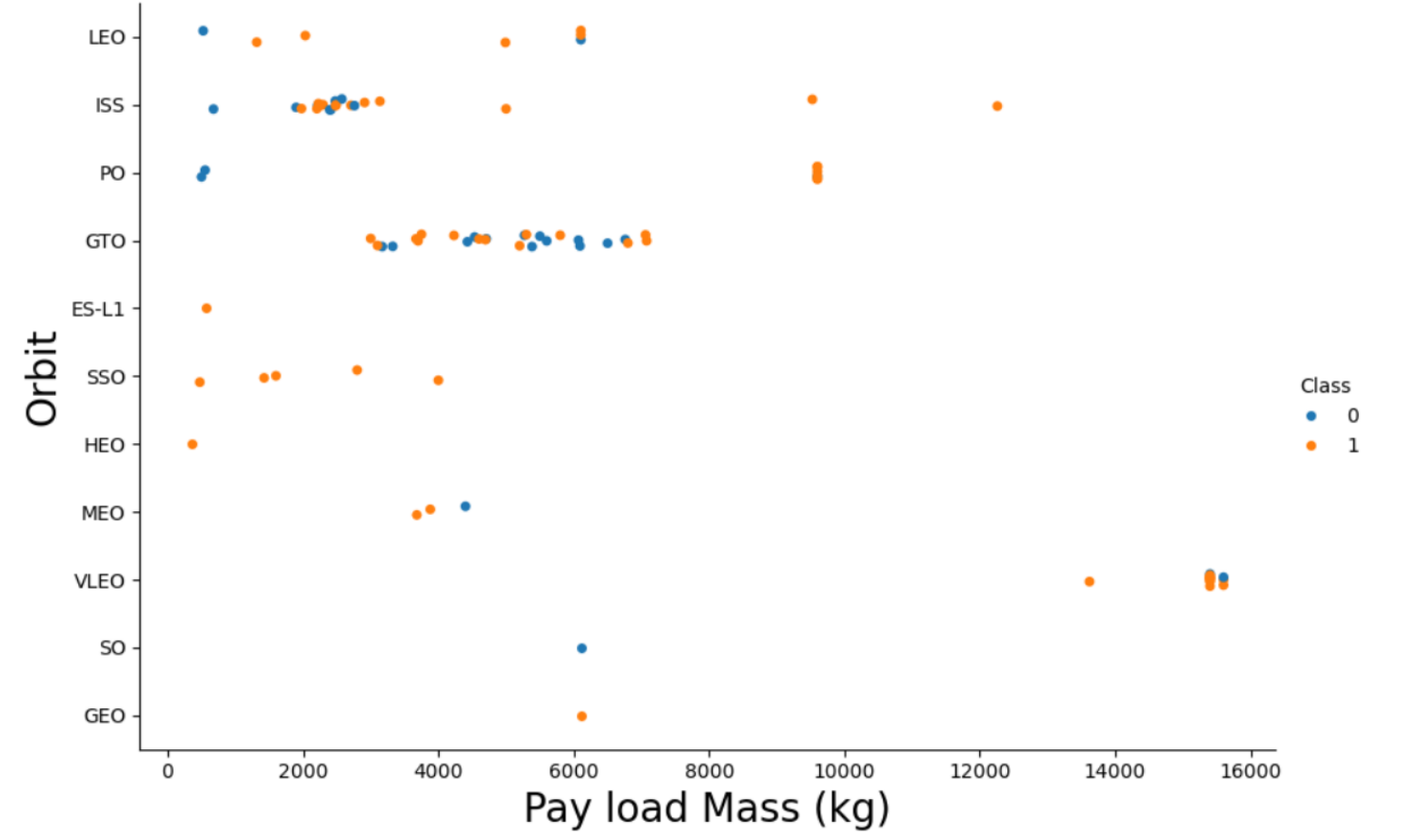
Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.

Flight Number vs. Orbit Type



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

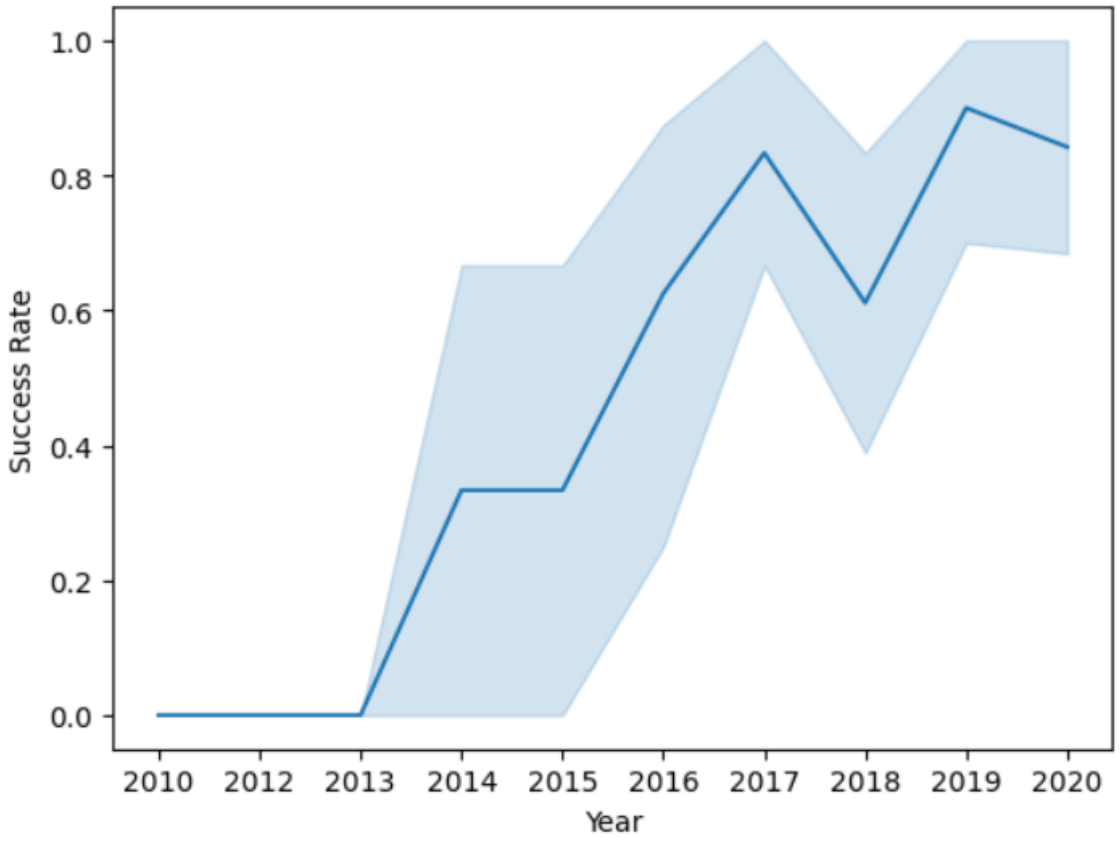
Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

Task 1

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

```
[10]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE ;
```

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

[10]:

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

Launch Site Names Begin with 'CCA'

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

```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

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

Total Payload Mass

Task 3 ¶

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

[13]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)';
```

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

[13]:

TotalPayloadMass
45596

Average Payload Mass by F9 v1.1

Task 4 ¶

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

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

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

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

MIN(DATE)
2015-12-22

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

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

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

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
]: %sql SELECT Mission_Outcome, COUNT(Mission_Outcome) as Total FROM SPACEXTBL GROUP BY Mission_Outcome;
```

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

]:

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

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.

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

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

]:

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

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 SELECT substr(Date,6,2) AS Month, Mission_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date,1,4)='2015' AND Landing_Outcome LIKE '%Failur
```

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

Month	Mission_Outcome	Booster_Version	Launch_Site
01	Success	F9 v1.1 B1012	CCAFS LC-40
04	Success	F9 v1.1 B1015	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.

```
7]: %sql SELECT Landing_Outcome, COUNT(*) AS Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Count DESC;
```

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

7]:

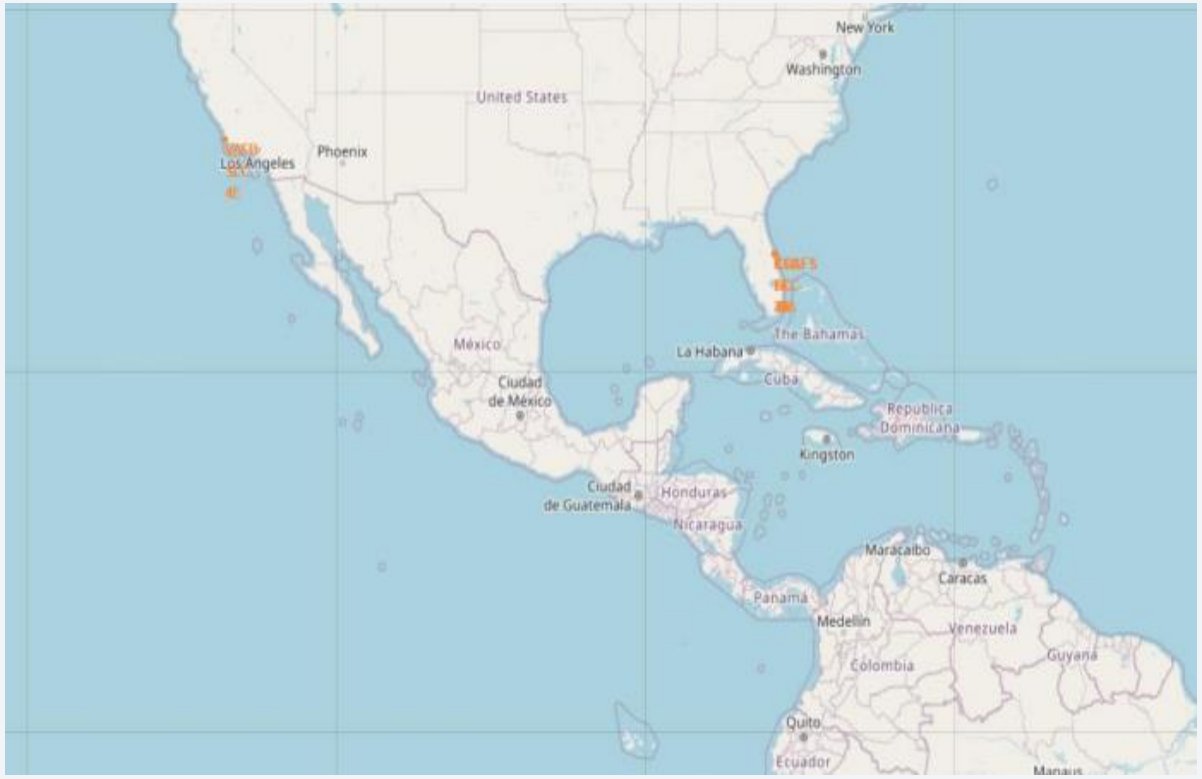
Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

Task 1: Identify all launch sites on the map

- Not all launch sites are in close proximity to the Equator.
- The launch site at Vandenberg Air Force Base (VAFB SLC-4E) is located at a latitude of 34.63, which is further from the Equator compared to the other sites in Florida.
- Yes, all launch sites are in close proximity to the coast.
- The Cape Canaveral sites (CCAFS LC-40 and CCAFS SLC-40) and Kennedy Space Center (KSC LC-39A) are near the coast in Florida.
- Vandenberg Air Force Base (VAFB SLC-4E) is also near the coast in California.



Task 2: Visualize launch outcomes for each site on the map

This enhanced visualization with clustered markers allows for better exploration and analysis of SpaceX launch data. The clustering makes it easier to manage a large number of markers and observe patterns that might be hidden in a less organized plot. By examining the marker colors and popup information, you can gain deeper insights into the characteristics and distribution of SpaceX launches.

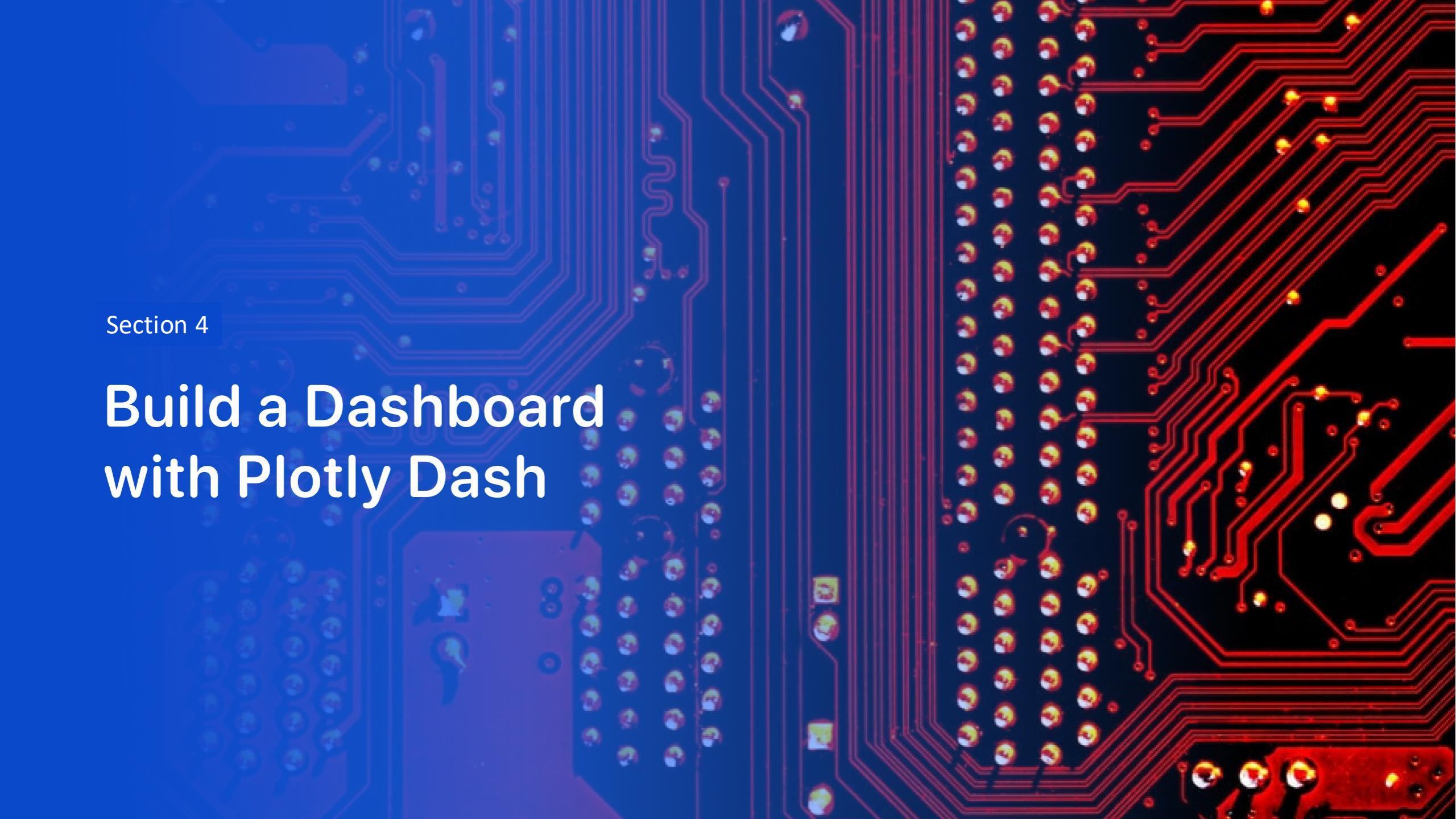
- For example, in the provided screenshot, out of 26 launch sites for CCAFS LC-40, there are 19 red markers and 7 green markers. This color-coding helps to quickly identify the success rate and other categorical distinctions of the launches from this specific site. The red markers might represent unsuccessful launches, while the green markers indicate successful ones, providing immediate visual feedback on the performance of launches at each site.

Task 3: Measure distances from a launch site to its surroundings

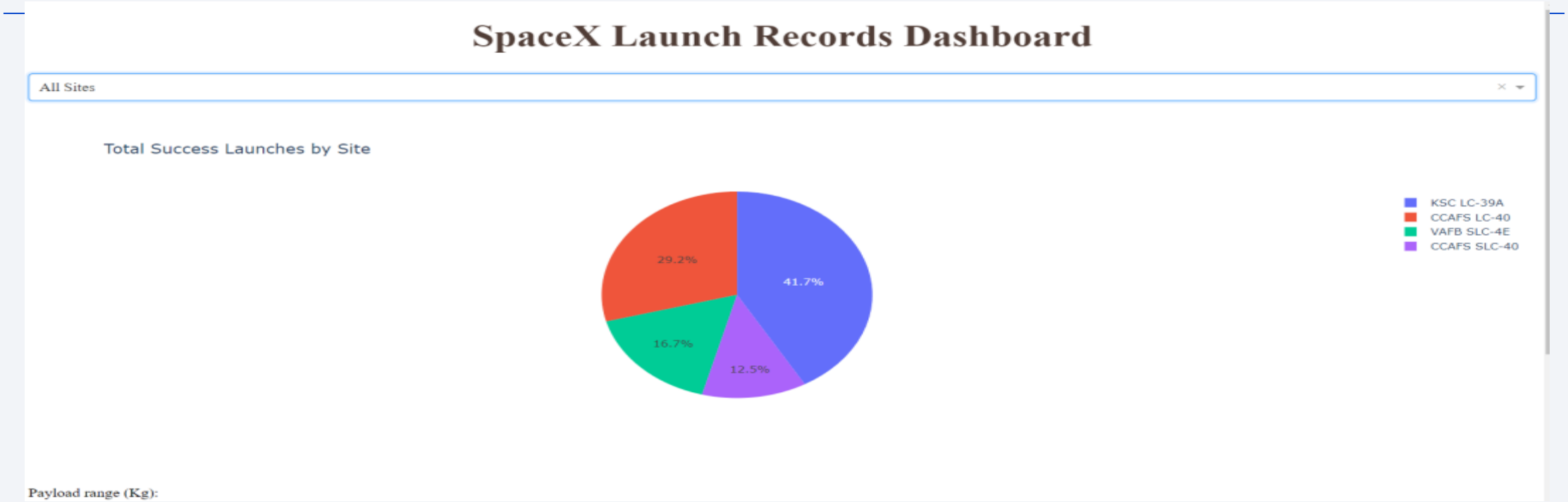
This plot provides a visual representation of the distance between the CCAFS SLC-40 launch site and the closest coastline. The calculated distance is approximately 0.51 kilometers, as indicated by the marker. The added PolyLine clearly shows the straight-line distance, highlighting the proximity of the launch site to the coast. This close proximity to the coastline is typical for launch sites to facilitate over-water flight paths and safe recovery operations, ensuring minimal risk to populated areas.

```
coastline_lat = 28.56367
coastline_lon = -80.57163

# Example launch site coordinates (replace with actual launch site coordinates)
launch_site_lat = launch_sites_df.loc[launch_sites_df['Launch Site'] == 'CCAFS SLC-40', 'Lat'].values[0]
launch_site_lon = launch_sites_df.loc[launch_sites_df['Launch Site'] == 'CCAFS SLC-40', 'Long'].values[0]

# Calculate distance using the calculate_distance function
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)

print(f"Distance from launch site to closest coastline: {distance_coastline} km")
```

```
Distance from launch site to closest coastline: 0.5097439631188213 km
```


Section 4
Build a Dashboard with Plotly Dash

Launch Success Count for all sites



SpaceX Launch Records Dashboard, uses a pie chart to show the **Total Success Launches by Site**. The most important finding is that the **KSC LC-39A** site (Kennedy Space Center) has been used for the most successful missions, representing the largest share at **41.7%**. The next busiest site is **CCAFS LC-40** (Cape Canaveral) with **29.2%**. This data shows that the majority of SpaceX's successful launches take place at the launch sites in Florida, making them the company's primary operational locations.

Pie chart for the launch site with highest launch success ratio

This pie chart highlights the launch site with the **highest launch success ratio**, which is **KSC LC-39A**. The key finding is that KSC LC-39A is highly reliable and effective, achieving a significant success rate of **76.9%** (Class 1 launches). This success rate is much higher than the unsuccessful rate (Class 0 launches), which stands at **23.1%**, emphasizing KSC LC-39A's strong performance as a primary launch location.

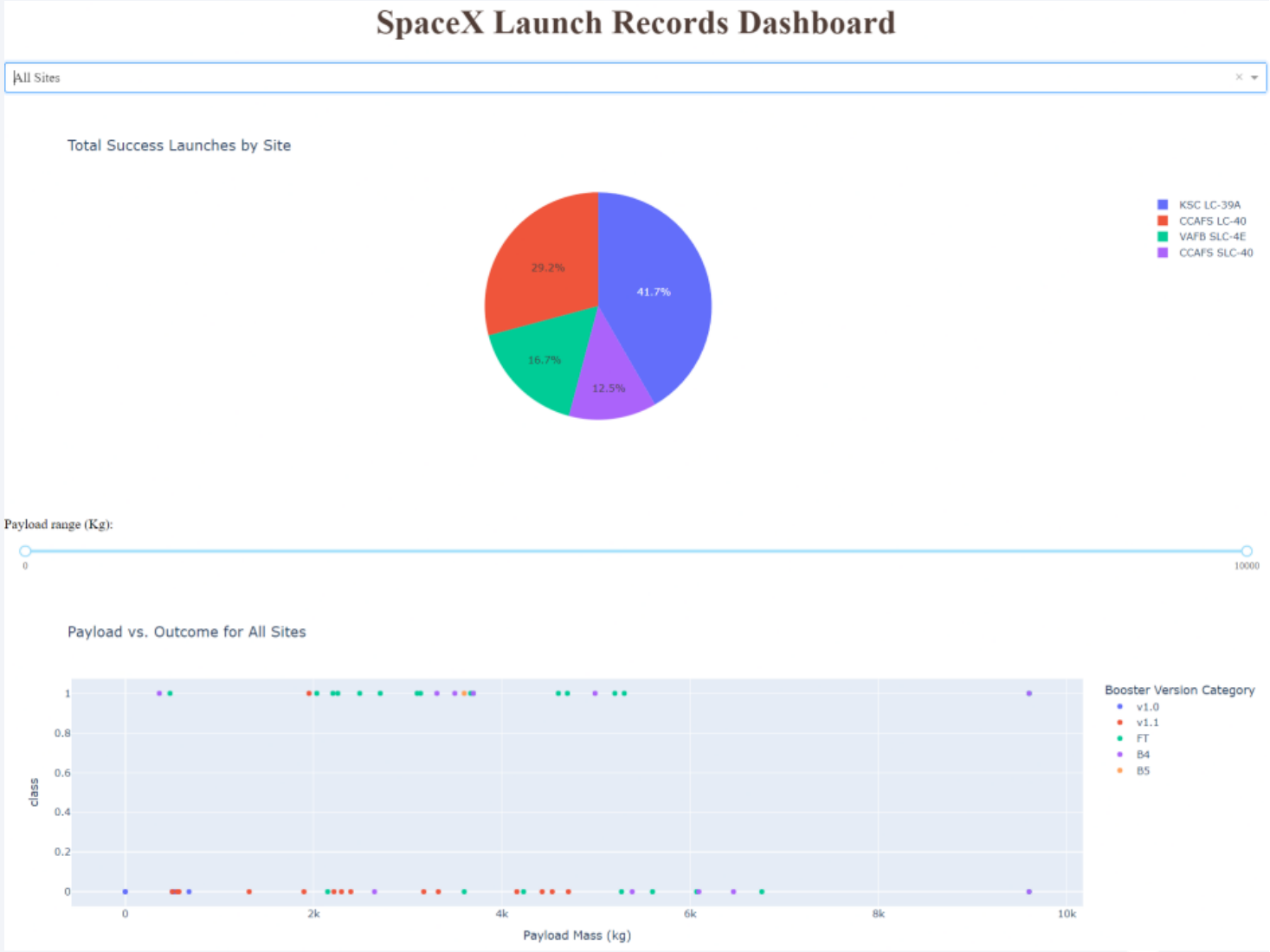
Insights from SpaceX Launch Data Dashboard

- Launch Site Reliability

The data shows that **CCAFS LC-40** (Cape Canaveral) has the **highest success rate at 43.7%**, meaning it is the most reliable launch site analyzed. Other sites, including KSC LC-39A, have lower success rates, which suggests launch success varies greatly depending on the location.

- Booster Performance

Regarding the rockets themselves, the **"FT" booster version** is used the most often and is very successful with different payload weights. The overall data suggests that simply having a heavier payload does not necessarily cause the success rate of the booster



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Based on the code results shown, the finding is that all four machine learning models, **Logistic Regression**, **SVM**, **Decision Tree**, and **KNN** performed **equally well** on the test data, each achieving an identical classification accuracy of **0.8333**. This result directly contradicts the paragraph's claim that the Decision Tree model was one of the best at 0.8333. Therefore, for this dataset and based only on accuracy, we cannot say one model is better than the others; **all models are equally suited** for the classification task.

TASK 12

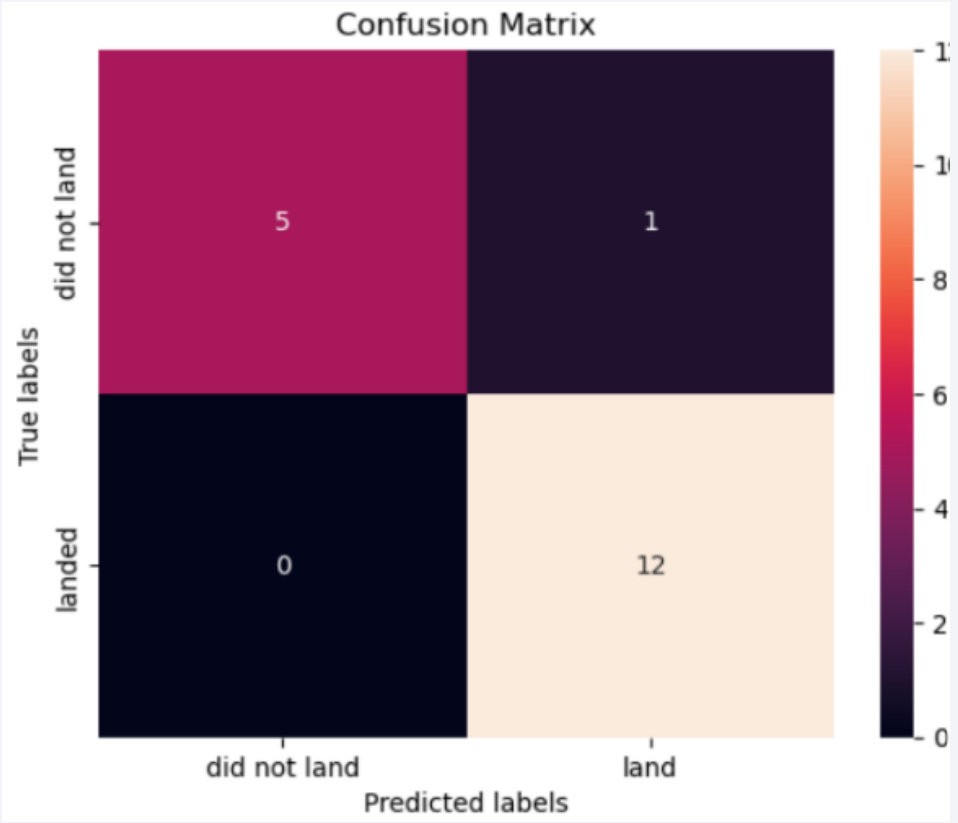
Find the method performs best:

[45]:
```
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_accuracy = logreg_cv.score(X_test, Y_test)

print("=== Test Data Accuracy Report ===")
print(f"Logistic Regression : {logistic_regression_accuracy:.4f}")
print(f"SVM                 : {svm_accuracy:.4f}")
print(f"Decision Tree       : {decision_tree_accuracy:.4f}")
print(f"KNN                 : {knn_accuracy:.4f}")
```

```
=== Test Data Accuracy Report ===
Logistic Regression : 0.8333
SVM                 : 0.8333
Decision Tree       : 0.8333
KNN                 : 0.8333
```

Confusion Matrix

- The model is performing very well, with a high overall accuracy of **83.33%** in predicting successful Falcon 9 first stage landings. A critical positive finding is the **absence of false negatives**, meaning the model never failed to predict an actual successful landing, which is vital for safety and planning in space operations. While there is a single **false positive** (predicting a success that didn't happen), this is considered a minor issue because over-preparing is much safer and easier to manage than under-preparing. Overall, the model is highly acceptable for real-world use as it is reliable and slightly favors predicting success, which is important for cost estimation and operational readiness.



Conclusions

- **Point 1: High Accuracy** The model is very accurate, scoring **83.33%**. It correctly identified many successful and unsuccessful landings, proving it works well for predicting if the Falcon 9 rocket's first stage will land successfully.

- **Point 2: No False Negatives** The model is perfect at identifying successful landings it **never missed a successful landing**. This is extremely important for safety and making sure everything is ready in space operations.

- **Point 3: Manageable False Positives** The model only made **one mistake by wrongly predicting a success** (1 false positive). This type of mistake is easier to deal with than missing an actual success, as it just means they over-prepared instead of being unprepared. This makes the model very practical.

- **Point 4: Balanced Performance** The model is reliable overall, with a small focus on predicting successful landings. This focus is ideal for the space industry because making sure a landing is successful is the **most important thing** for planning and estimating costs.

Thank you!