IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sara Sampaio
12 August 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection API with Web Scraping
  - Data Wrangling
  - Exploratory Analysis using SQL
  - Exploratory Data Analysis for Data Visualization
  - Interactive Visual Analytics and Dashboard
  - Predictive Analysis with Machine Learning

- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics
  - Predictive Analytics result

Introduction

- Project background and context
  - The commercial space age is making space travel more accessible, with companies like SpaceX leading the way. SpaceX’s success is largely due to its ability to reuse the first stage of its Falcon 9 rocket, significantly reducing launch costs. To compete, Space Y needs to analyze SpaceX's data to predict launch costs and the likelihood of reusing the first stage.
- Problems you want to find answers
  - Cost Prediction: Estimate the price of a Falcon 9 launch.
  - Reuse Prediction: Predict if the first stage will land and be reused.
  - Launch Success: Identify key factors for successful launches.
  - Competitive Analysis: Compare Space Y's potential with SpaceX's performance.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using SQL and data visualization
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis with Machine Learning (ML)
  - How to build, tune and evaluate classification models

Data Collection

- Data collection was done using get request to the SpaceX API
- The response content was decoded as a Json using .json() function call and turnned it into a pandas dataframe using .json_normalize()
- Data was cleaned, missing values were checked and filled where necessary
- Web scraping was performed for Falcon 9 launch records with BeautifulSoup.
- Main objective: extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- The get request to the SpaceX API to collect data was used, cleanned the requested data and did some basic data wrangling and formatting
- Link to the notebook: https://github.com/sararsampaio/testrepo/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

In [5]:
```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

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

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

In [10]:
```
response.status_code
```

Out[10]:
```
200
```

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

In [15]:
```
# Use json_normalize meethod to convert the json result into a dataframe
# Decode the JSON response content
json_data = response.json()

# Convert the JSON data into a Pandas DataFrame
data = pd.json_normalize(json_data)

# Display the first few rows of the DataFrame
print(data.head())
```

Data Collection - Scraping

- Web scrapping was applied to webscrap Falcon 9 launch records with BeautifulSoup
- The table was parsed and converted it into a pandas dataframe.
- Link to the notebook: https://github.com/sararsampaio/testrepo/blob/main/jupyter-labs-webscraping.ipynb

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

In [4]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)

# assign the response to a object
soup = BeautifulSoup(response.text, 'html.parser')
```

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

In [5]:
```
# 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

In [6]:
```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

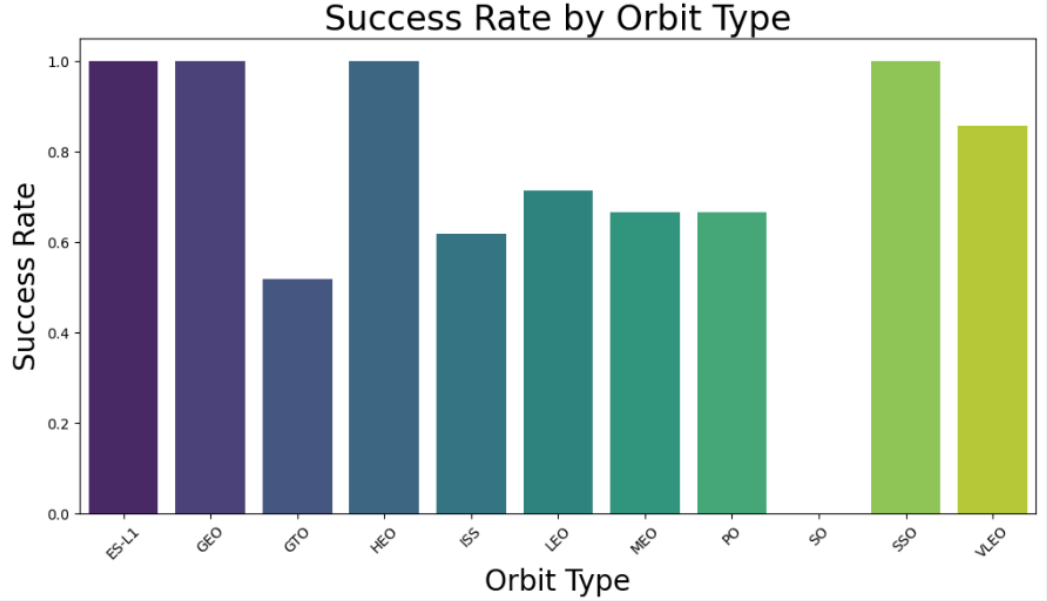
In [ ]:
```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

Data Wrangling

- Exploratory data analysis was performed and the training labels were determined
- The number of launches at each site and the number and occurrence of each orbits was calculated
- The landing outcome label was created from outcome column and the results were exported to csv
- Link to the notebook: https://github.com/sararsampaio/testrepo/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- The data was explored by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend
- Link to the notebook: https://github.com/sararsampaio/testrepo/blob/main/edadataviz.ipynb



EDA with SQL

- The SpaceX dataset was loaded and the connection was established with the database
- EDA was applied using SQL to get more information from the data

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.

In [32]:
```
%sql SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' (
```

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

Out[32]:

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

- The link to the notebook is: https://github.com/sararsampaio/testrepo/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- All launch sites were added to the map, as well markers, circles, lines to mark the success or failure of launches for each site on the folium map
- The launch outcomes (failure or success) were assigned to class 0 and 1 (0 for failure, and 1 for success)
- The launch sites that have relatively high success rate were assigned using color-labeled marker clusters
- The distances between a launch site to its proximities were calculated
- Link to notebook: https://github.com/sararsampaio/testrepo/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

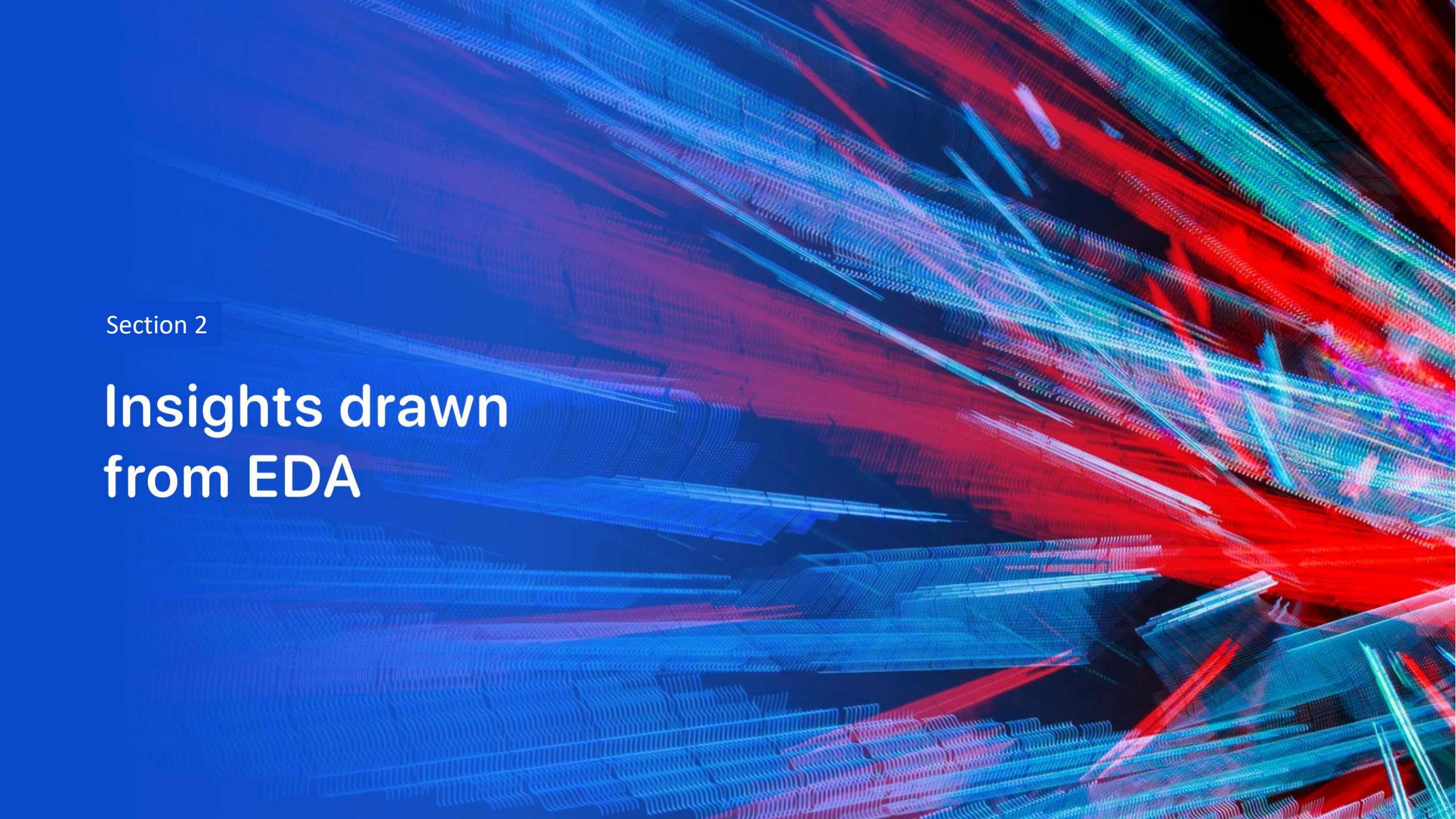
- An interactive dashboard with Plotly dash was built
- Pie charts were plotted showing the total launches by a certain sites
- Scatter plot graphs were plotted showing the relationship with Outcome and Payload Mass (Kg) for the different booster version
- Link to notebook: https://github.com/sararsampaio/testrepo/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Data was loaded using numpy and pandas, transformed, and then data was split into training and testing
- Different machine learning models were built and different hyperparameters were tunned using GridSearchCV
- Accuracy was the metric used for this model and the model was improved using feature engineering and algorithm tunning
- The best performing classification model was found
- Link to notebook: https://github.com/sararsampaio/testrepo/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

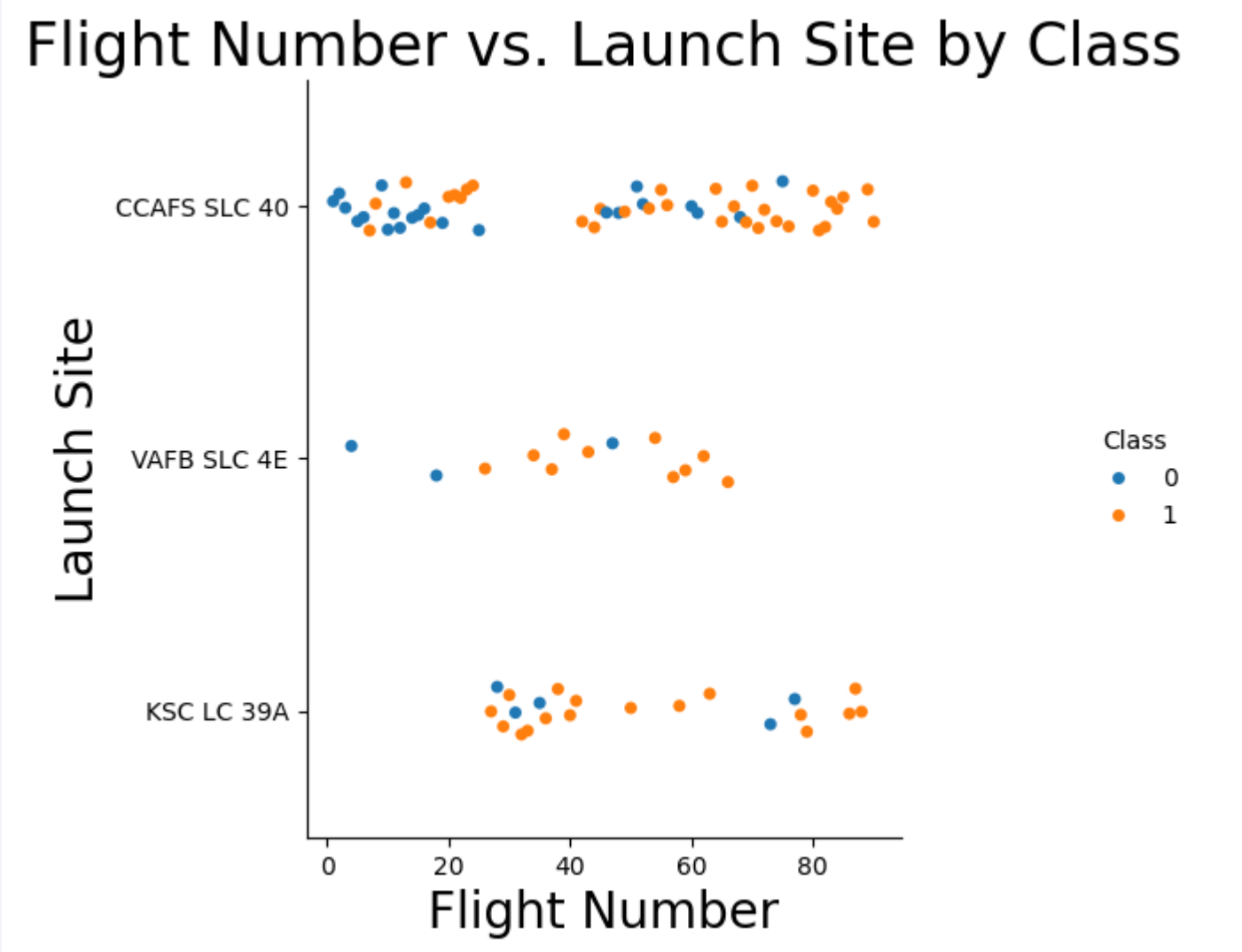
Results

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


Section 2
Insights drawn from EDA

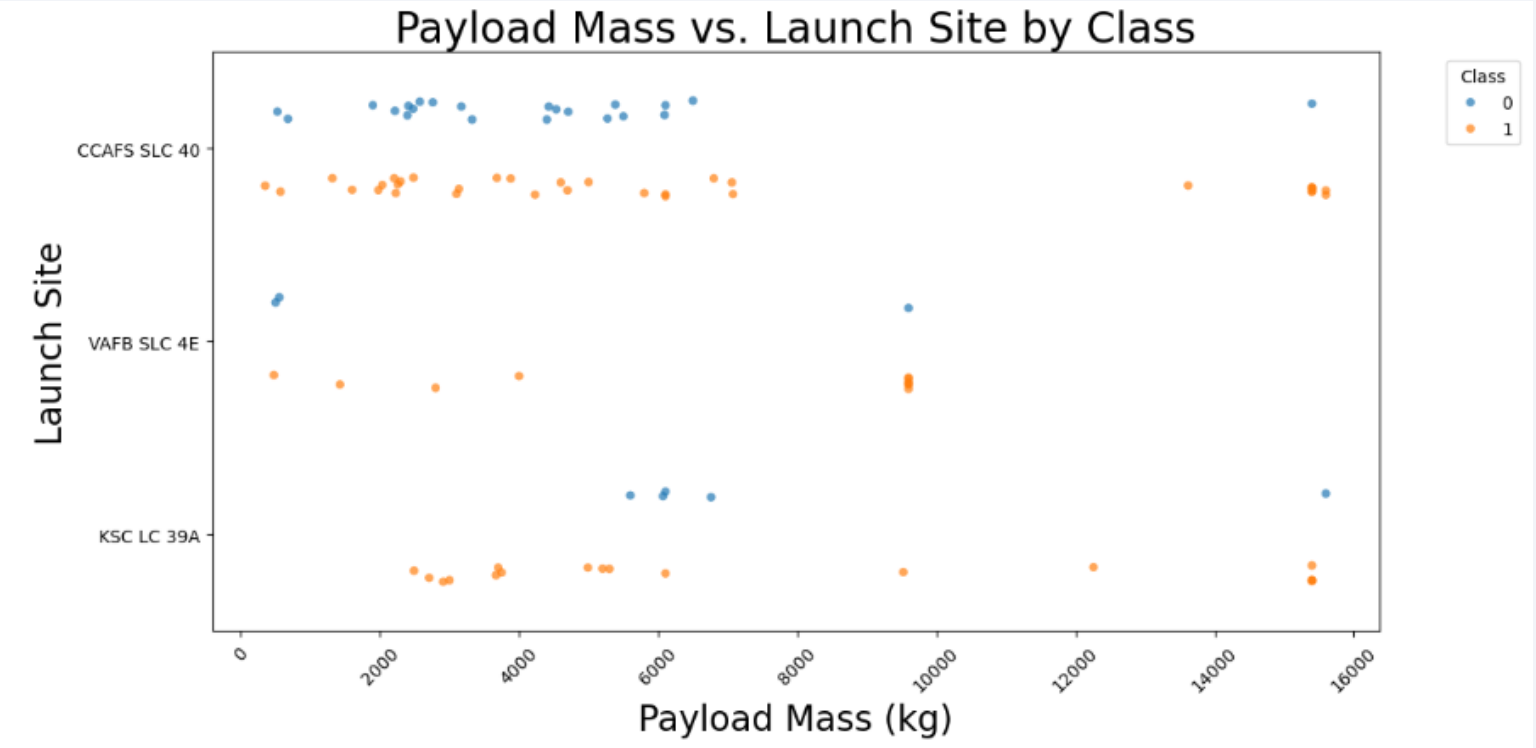
Flight Number vs. Launch Site

- The larger the flight numbers at a launch site, the greater the success rate at a launch site



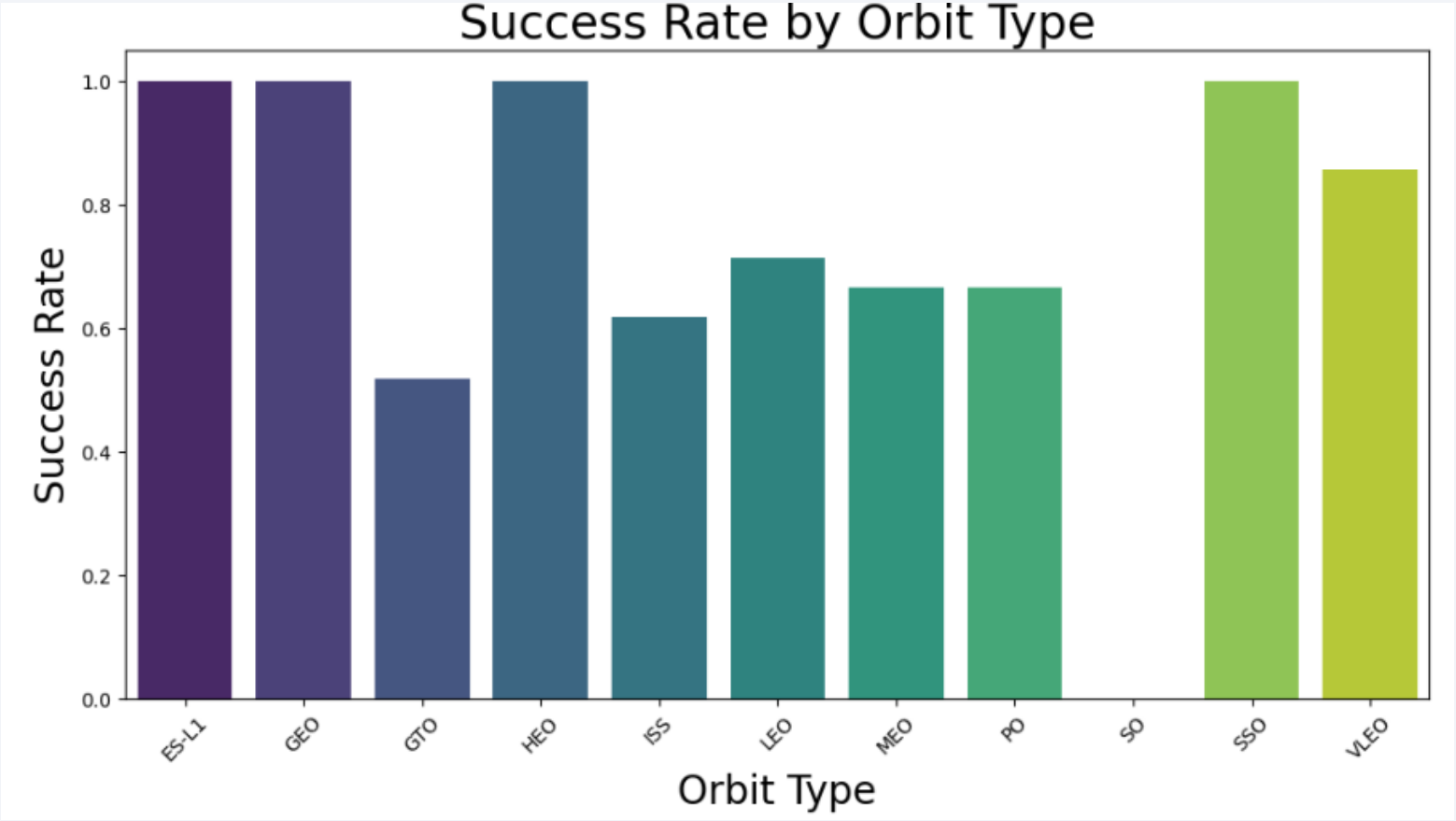
Payload vs. Launch Site

- The launch sites with higher payload mass are CCAF5 SLC 40 and KSC LC 39A



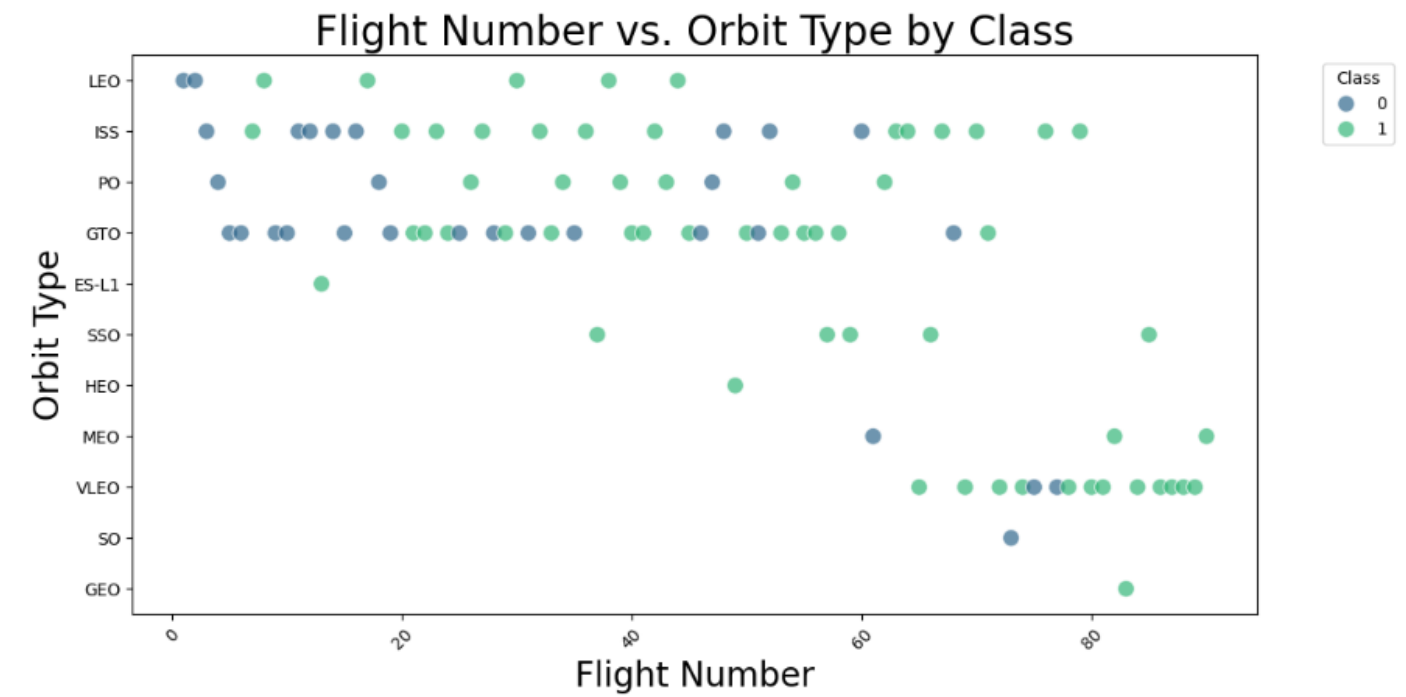
Success Rate vs. Orbit Type

- The ES-L1, GEO, HEO and SSO are the orbit types with 100% success rate



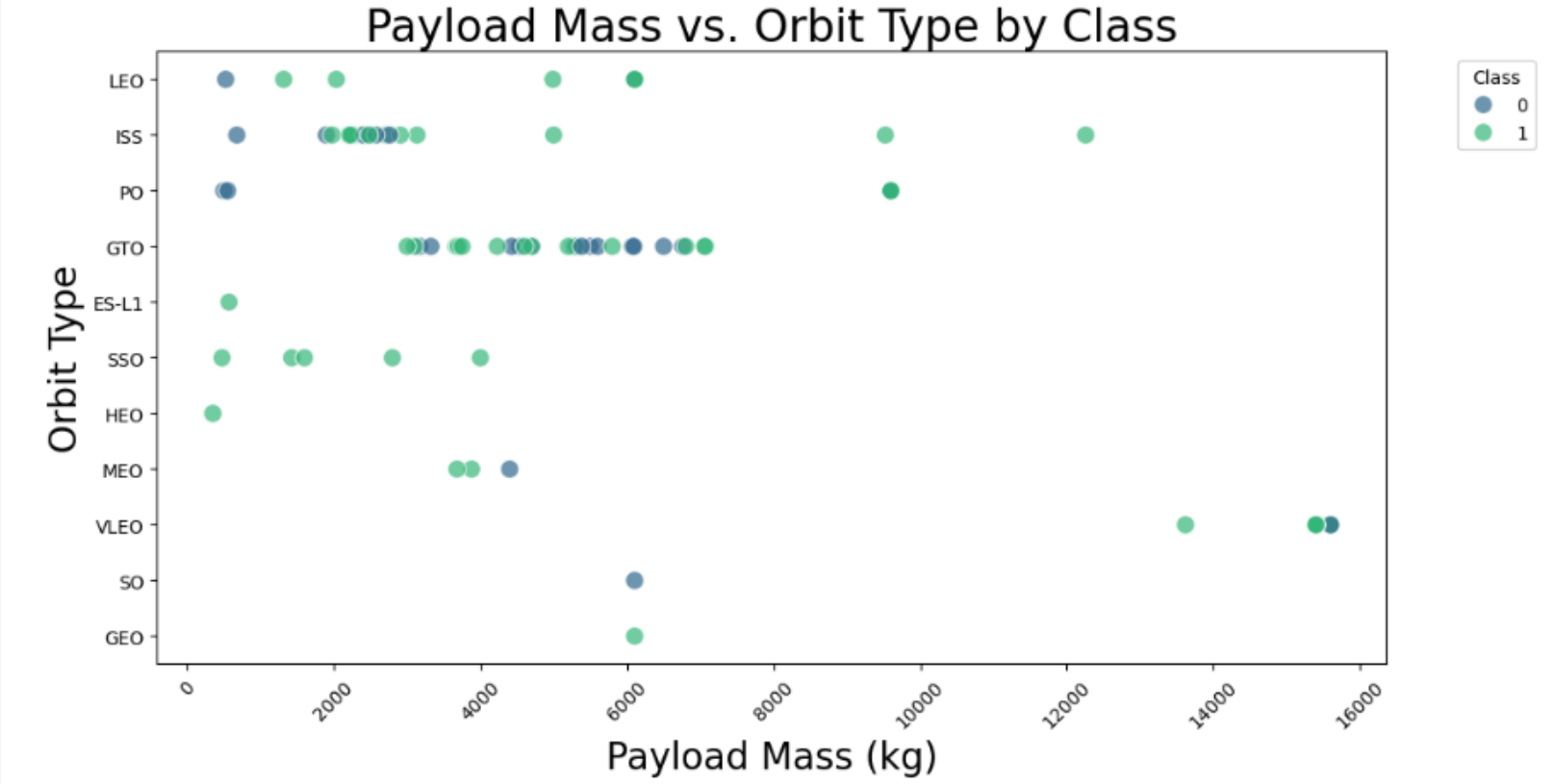
Flight Number vs. Orbit Type

- ISS and VLEO are the orbit types with higher flight numbers and seem to be the ones with higher success. On the other hand, the GTO orbit type appears to not have any relationship between flight number and success



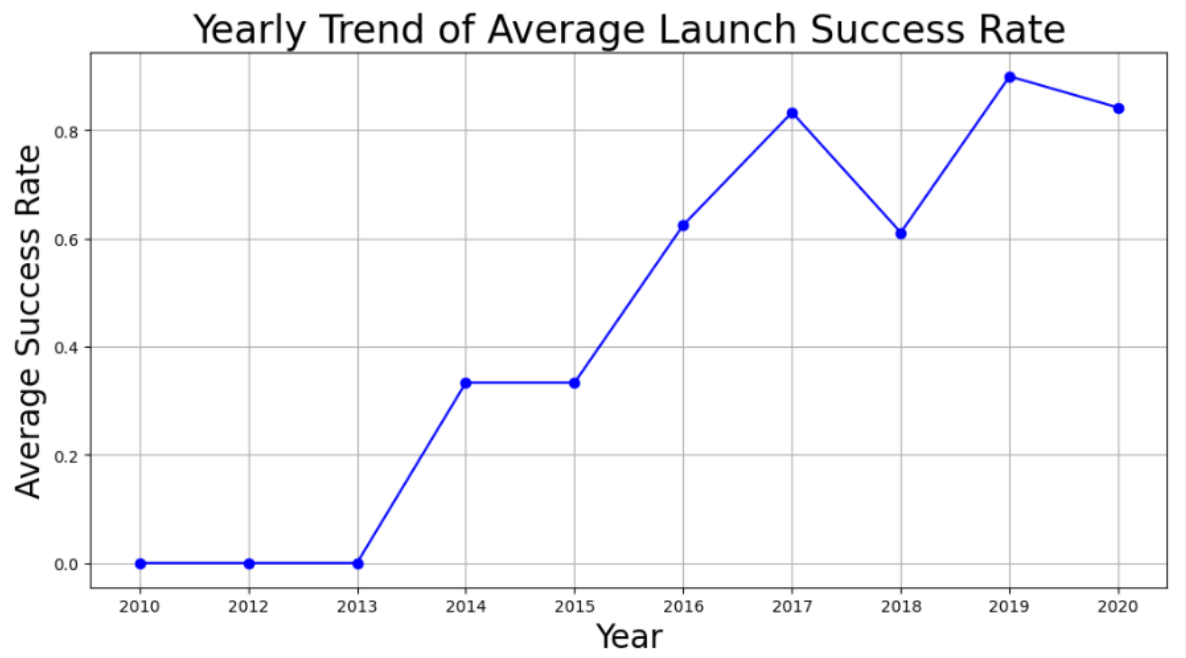
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for PO, 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

- The success rate since 2013 kept increasing till 2020



All Launch Site Names

- DISTINCT was used to show only unique launch sites from the SpaceX data.

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

In [20]:
```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE
```

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

Out[20]:

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

Launch Site Names Begin with 'CCA'

- The presented query was used to display 5 records where launch sites begin with `CCA`

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

In [21]:
```
# Query to 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.
```

Out[21]:

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

- The total payload mass carried by boosters from NASA (48213Kg) was calculated using the query below:

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

In [18]:
```
# Connect to the SQLite database
con = sqlite3.connect("my_data1.db")
cur = con.cursor()

%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass FROM SPACEXTABLE WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

Out[18]:

Total_Payload_Mass
48213

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 (2928.4 Kg) was calculated using the query below:

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

In [19]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';
```

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

Out[19]:

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

- The dates of the first successful landing outcome on ground pad was 22nd December 2015

```
In [22]: %sql SELECT MIN(Date) AS First_Successful_Landing FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';
 * sqlite:///my_data1.db
Done.
Out[22]: First_Successful_Landing
         2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- The WHERE clause was used to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [23]:
```
%sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

Out[23]:

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

Total Number of Successful and Failure Mission Outcomes

- COUNT was used to count all mission outcomes and GROUP BY to group them by Mission_Outcome. In total we have 1 failure and 100 successful missions

List the total number of successful and failure mission outcomes

In [24]:
```
%sql SELECT "Mission_Outcome", COUNT(*) AS Total FROM SPACEXTABLE GROUP BY "Mission_Outcome";
```

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

Out[24]:

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

Boosters Carried Maximum Payload

- The booster that have carried the maximum payload was determined using a subquery in the WHERE clause and the MAX() function

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

In [29]:
```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

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

Out[29]:

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

2015 Launch Records

- A combination of the WHERE clause, AND, and substr conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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.

In [31]:
```
%sql SELECT substr(Date, 6, 2) AS Month, "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Failure (drone ship)' AND substr(Date, 1, 4) = '2015';
```

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

Out[31]:

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

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

- There was a COUNT of landing outcomes from the data and the WHERE clause was used to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20. The GROUP BY clause was used to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order

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.

In [32]:
```
%sql SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY Outcome_Count DESC;
```

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

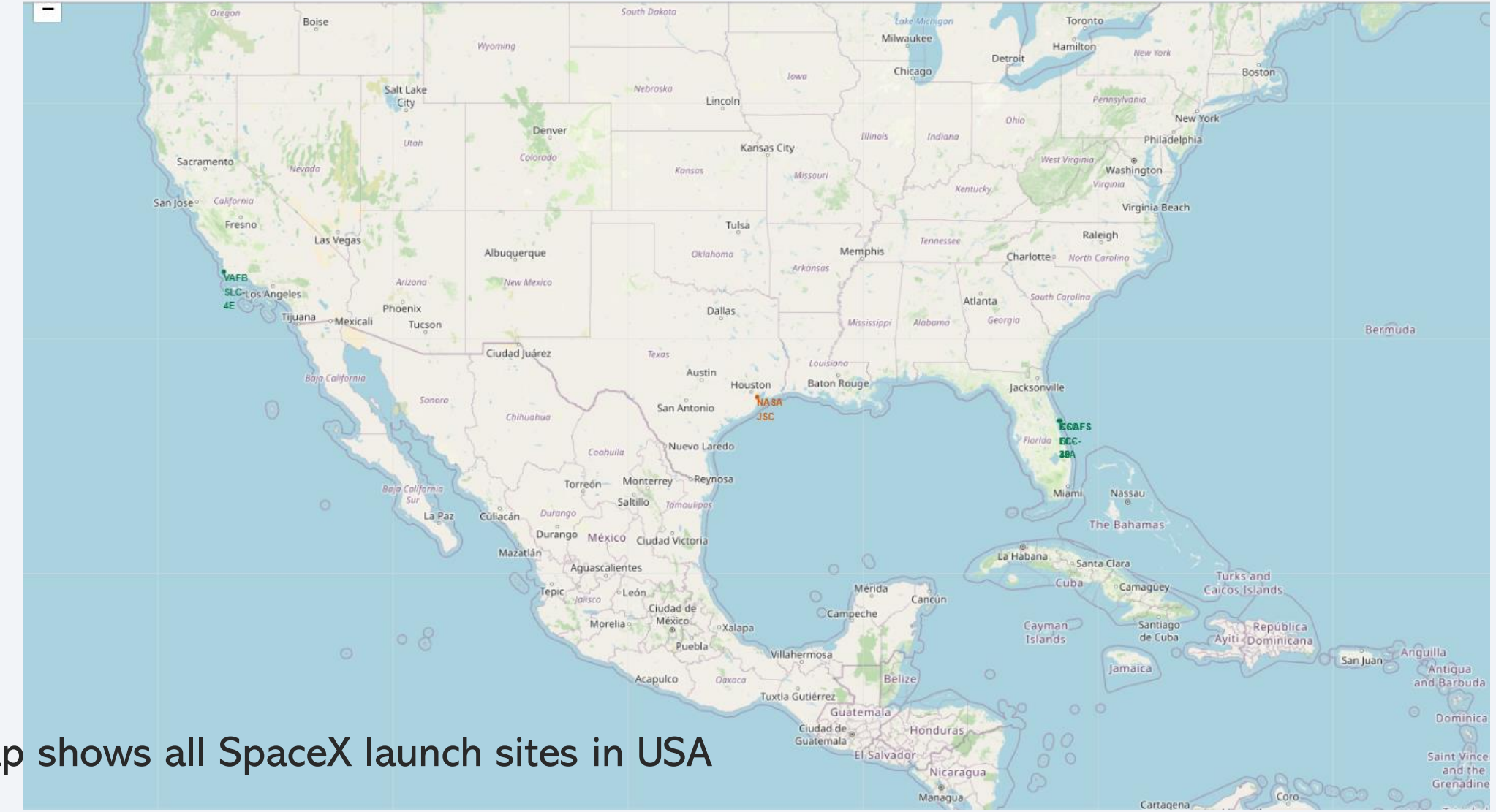
Out[32]:

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

Section 3

Launch Sites Proximities Analysis

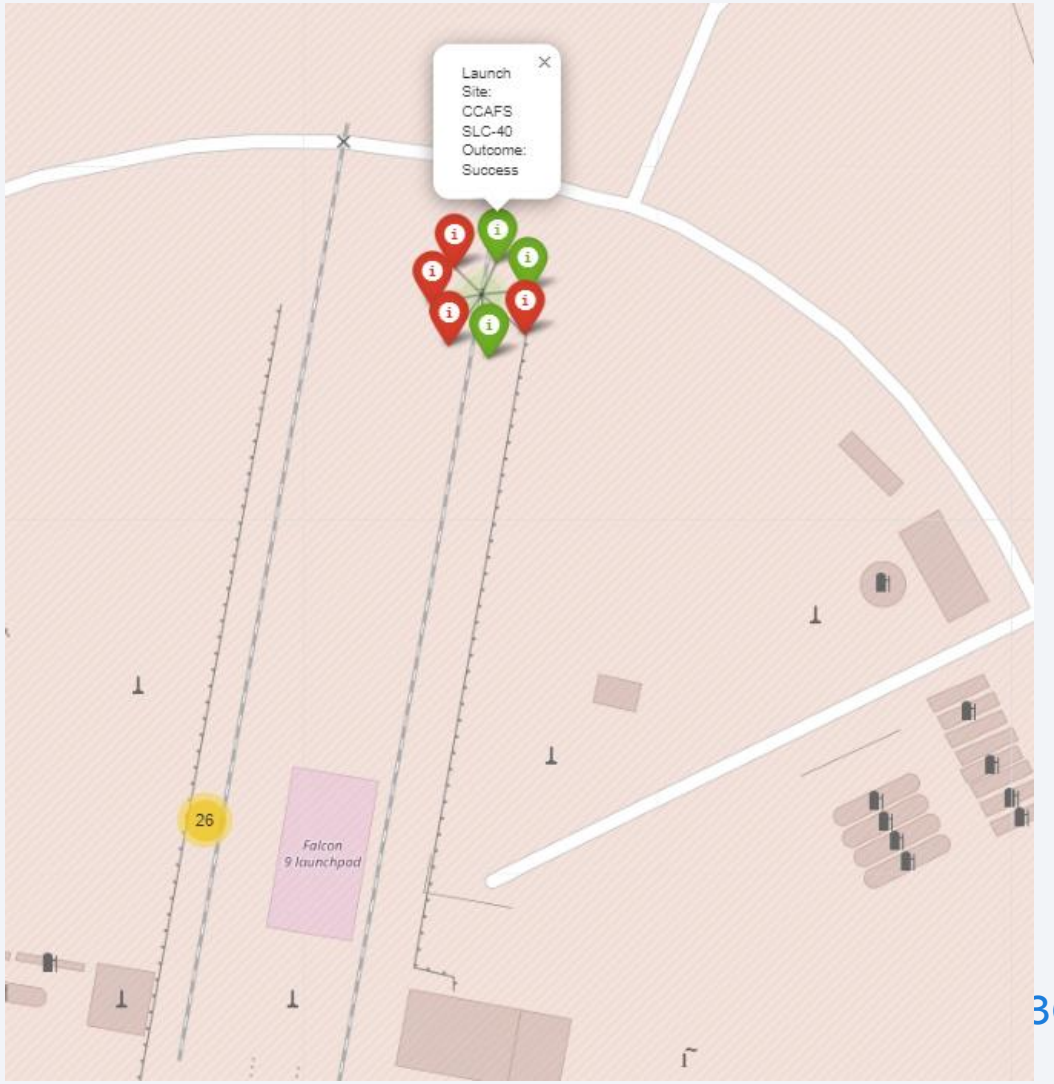
<Folium Map Screenshot 1>



- Map shows all SpaceX launch sites in USA

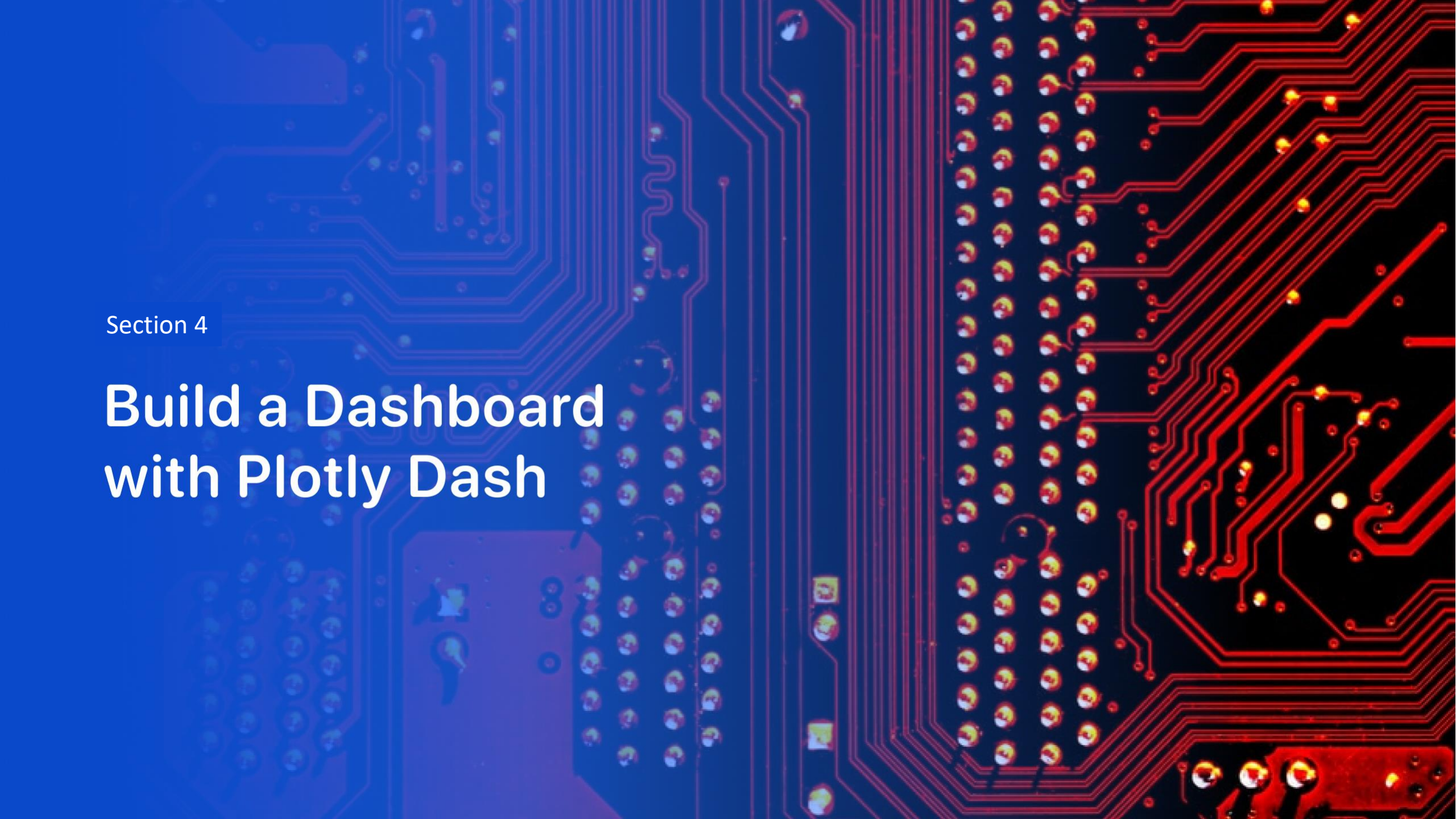
<Folium Map Screenshot 2>

- Mas showing Florida launch sites. Green markers are successful launches, and the red ones are failures.

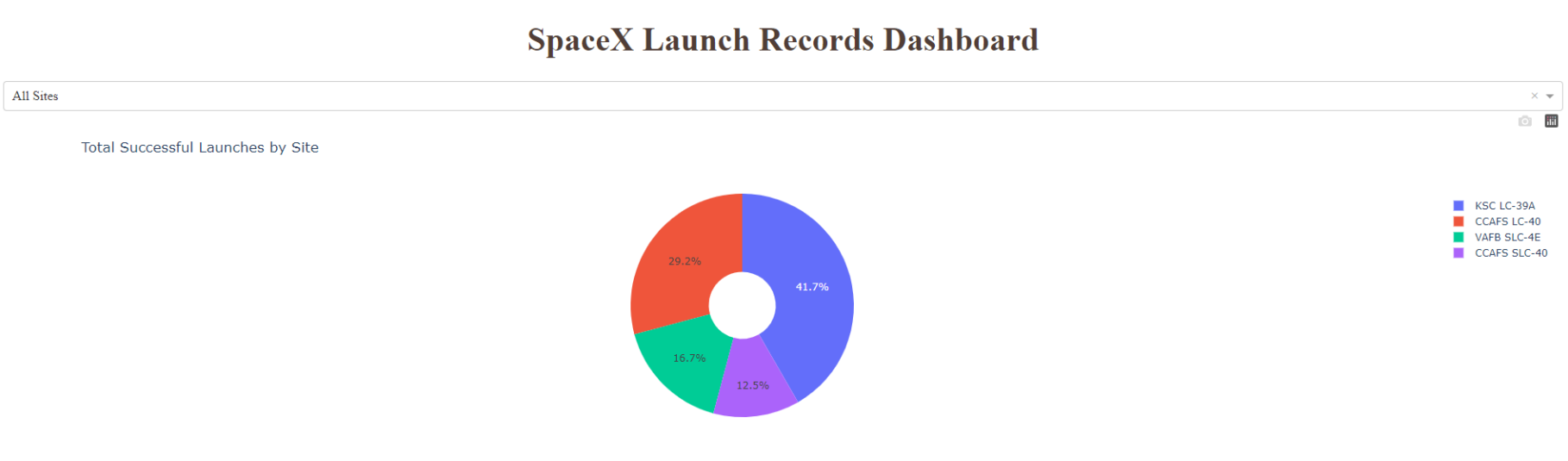


<Folium Map Screenshot 3>

- Map showing the lines between the launch site and each point of interest, in Florida. Red line means distance to city; green distance to railway; blue distance to highway

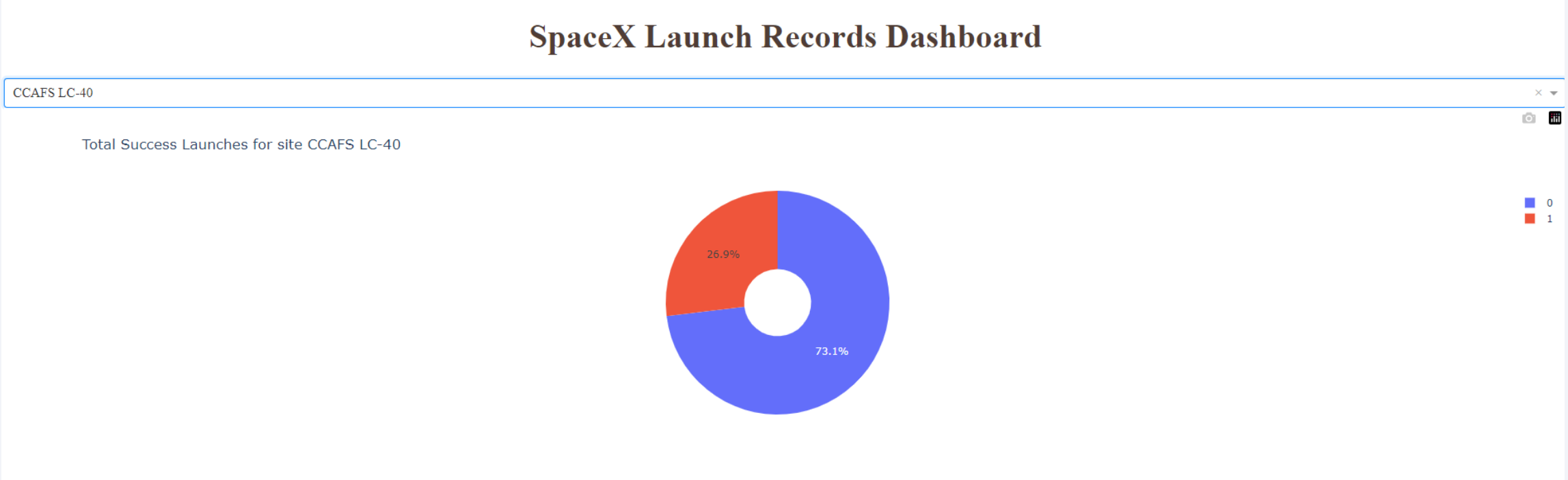

Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



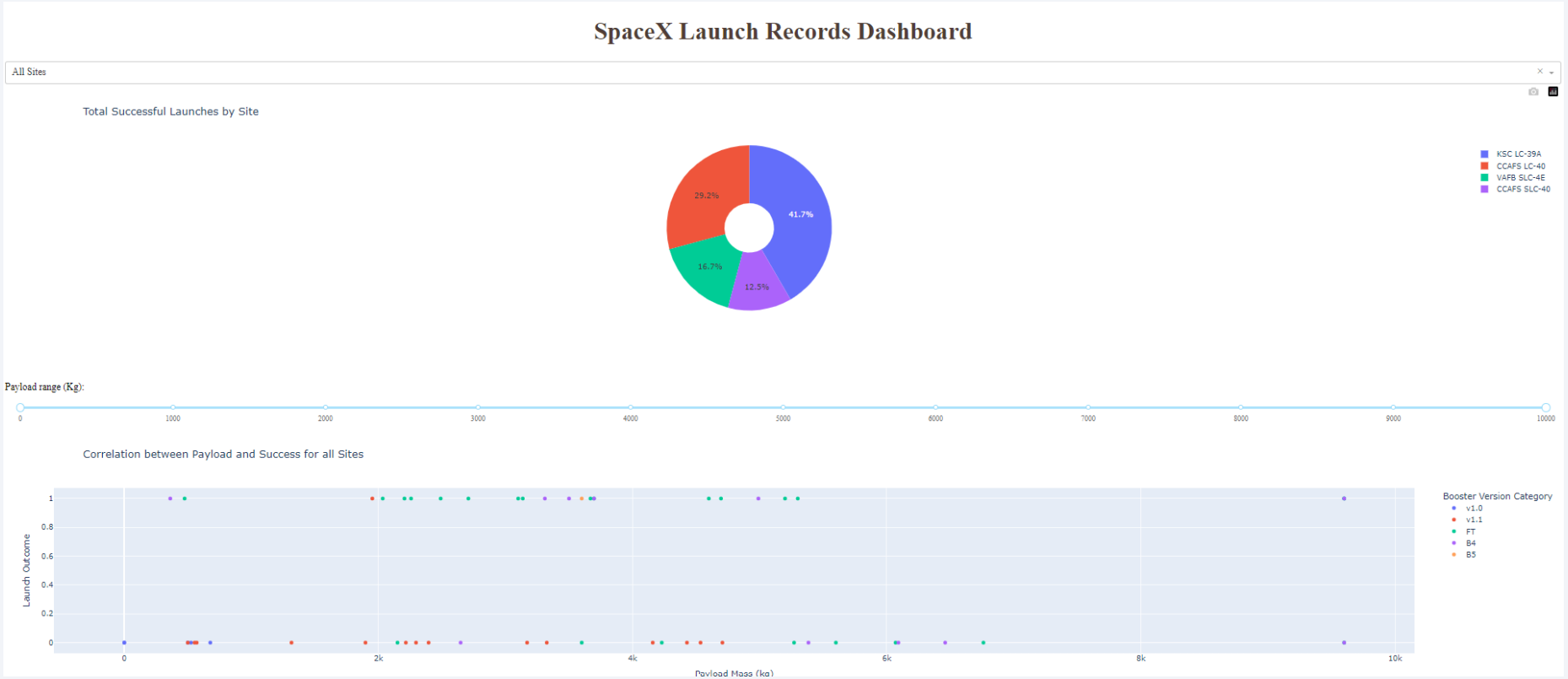
- The figure shows that KSC LC-39A has the most successful lunches from all sites

<Dashboard Screenshot 2>



- The figure shows that CCAFS LC-40 has a total success rate of 73.1% and 26.9% of failure rate

<Dashboard Screenshot 3>



- The figure shows the total successful lauches by site for all sites. FT is the booster version with highest successful launch outcomes and B4 the version with highest payload mass for both launch outcomes

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The best model with the highest classification accuracy is the decision tree model

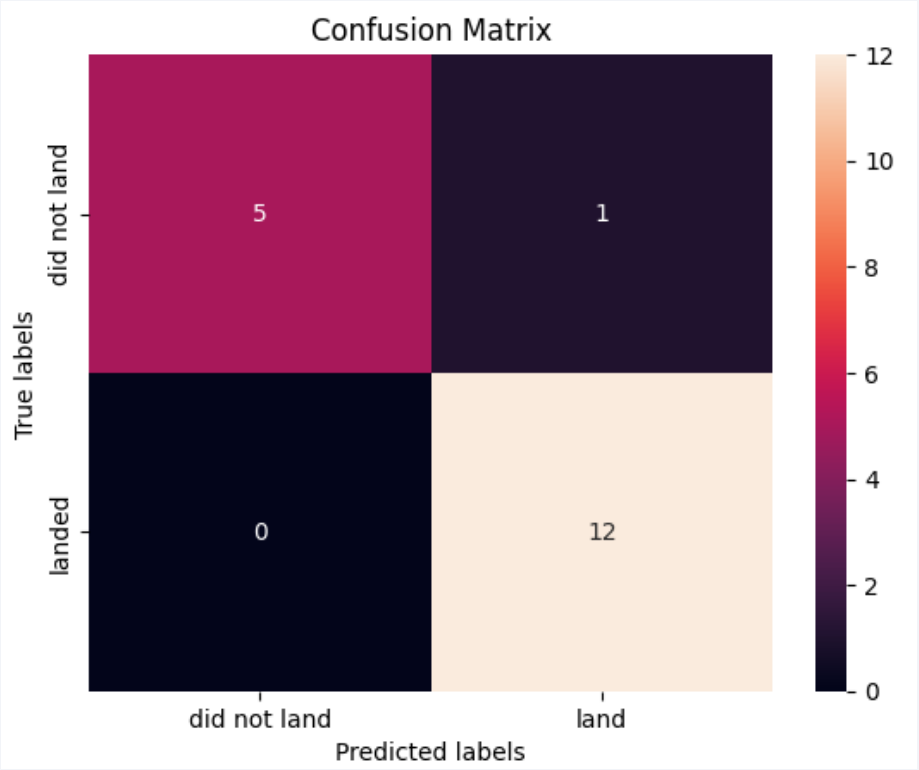
Find the method performs best:

In [43]:
```
# Example of printing all results
print("Logaccuracy :",logreg_cv.best_score_)
print("Test Accuracy: ", test_accuracy)
print("Tree est Accuracy: ", tree_test_accuracy) #best
print("Knn accuracy :",knn_cv.best_score_)
```

```
Logaccuracy : 0.8464285714285713
Test Accuracy:  0.8333333333333334
Tree est Accuracy:  0.9444444444444444
Knn accuracy : 0.8482142857142858
```

Confusion Matrix

- The figure shows the confusion matrix for the decision tree. This classifier can distinguish between the different classes.



Conclusions

- The higher the flight number at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 untill 2020.
- The ES-L1, GEO, HEO and SSO orbits had the most success rate (100%).
- KSC LC-39A had the most successful launches from all sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!