IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jack Feen
03/20/2025

Outline

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

Executive Summary

Methodology

- **Data Collection**: Falcon 9 first-stage landing data (2010–2020) was collected using a public API and publicly available sources like Wikipedia.
- **Data Wrangling**: Extracted and cleaned landing outcome data to serve as the dependent variable for machine learning models.
- **Exploratory Data Analysis**: Used SQL queries and data visualizations, including static plots and interactive maps, to uncover insights.
- **Interactive Analytics**: Created interactive visual analytics using Folium and dashboards to analyze the dataset dynamically.
- **Predictive Modeling**: Implemented machine learning models (Logistic Regression, SVM, Decision Tree, KNN) to predict landing outcomes.

Results

- **Exploratory Findings**: Analyzed launch details, including flight number, launch date, payload mass, orbit type, launch site, and mission outcomes.
- **Interactive Insights**: Developed interactive analytics with dashboards and visualizations.
- **Predictive Analysis**: Found that Logistic Regression, SVM, and KNN performed equally well for predicting Falcon 9 landing success.

Introduction

Project Background and Context

- SpaceX offers Falcon 9 rocket launches at a much lower cost than competitors charging due to reusable rocket technology.
- Predicting first-stage landings can help assess launch costs and provide valuable insights for competing rocket companies looking to bid against SpaceX.
- The goal is to develop a **machine learning pipeline** to predict whether a Falcon 9 first-stage landing will be successful.

Key Questions to Explore

- What factors determine if the Flacon-9 will land successfully?
- Which machine learning model provides the highest accuracy for predicting landing success?
- What conditions ensure the success of future launches?

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
  - Numpy and SKLearn were used to devolop and fit the models

Data Collection

- Data was collected via a request to the SpaceX API
- The information was read into our Jupyter notebook and turned into a dataframe using Pandas
- A Wikipedia Page was then web scraped using Beautiful Soup

Data Collection – SpaceX API

- https://github.com/JackFeen/Coursera/blob/main/Data%20Science/jupyter-labs-spacex-data-collection-api-v2%20(2).ipynb

Below we will define a series of helper functions that will help us use the API to extract information using identification numbers in the launch data.

From the `rocket` column we would like to learn the booster name.

In [3]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
            BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [4]:
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
            Longitude.append(response['longitude'])
            Latitude.append(response['latitude'])
            LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [5]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        if load:
            response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
            PayloadMass.append(response['mass_kg'])
            Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

In [6]:
```
# 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'])
```

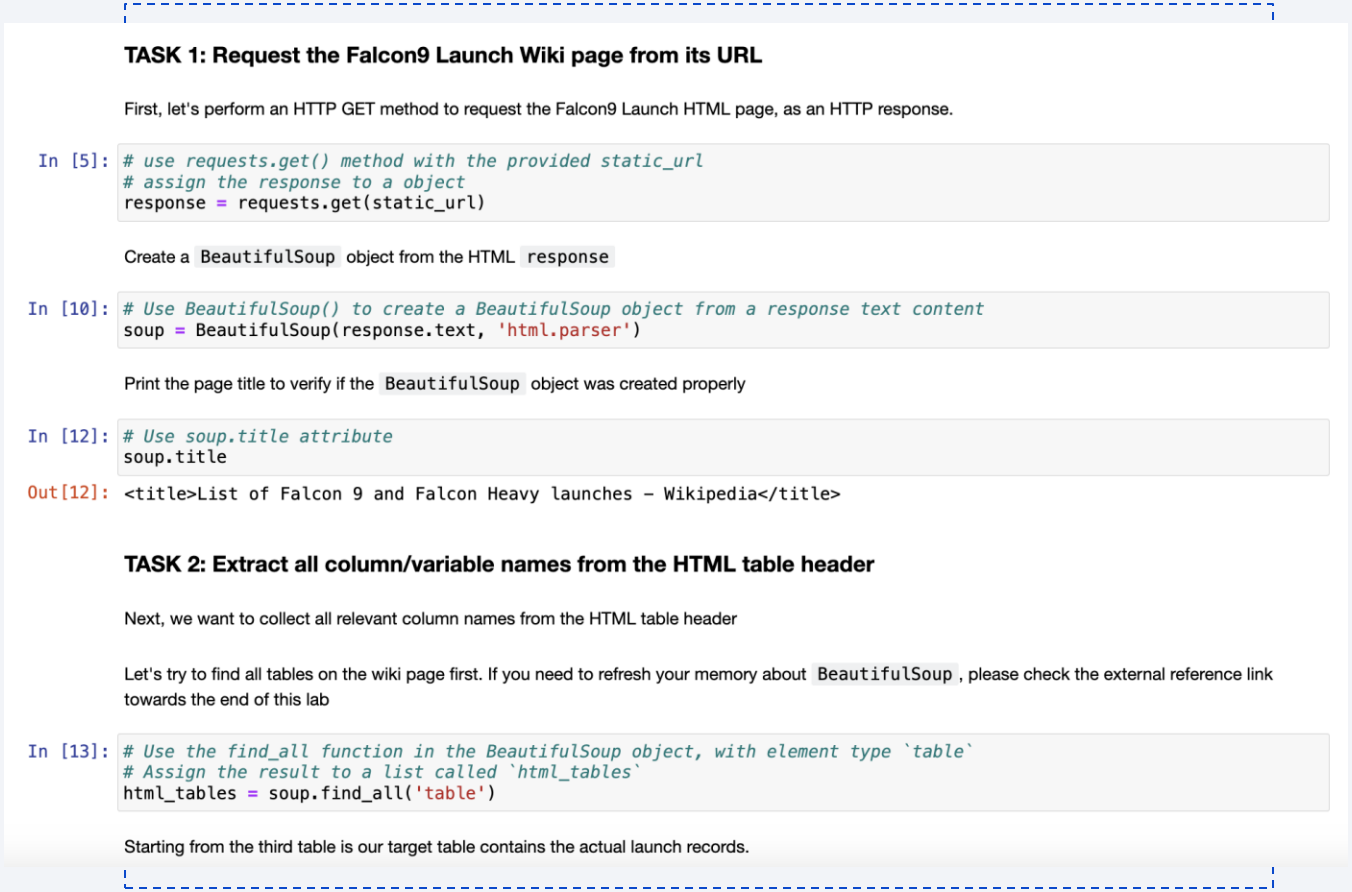
Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [7]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [8]:
```
response = requests.get(spacex_url)
```

Data Collection - Scraping

- https://github.com/JackFeen/Coursera/blob/main/Data%20Science/jupyter-labs-webscraping%20(1).ipynb



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.

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

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

In [10]:
```
# 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 [12]:
```
# Use soup.title attribute
soup.title
```

Out[12]:
```
<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

In [13]:
```
# 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.

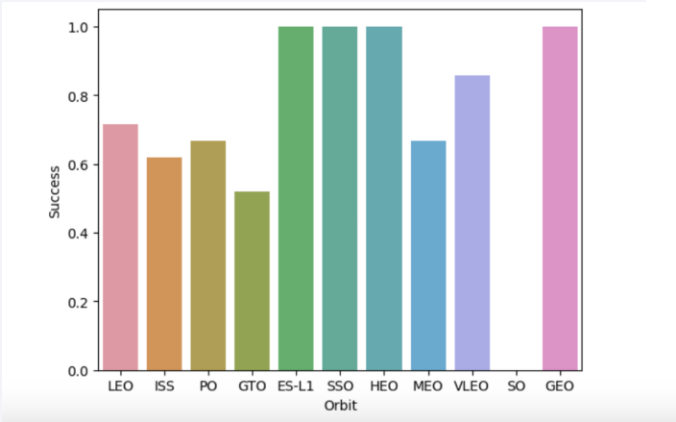
Data Wrangling

- Data was wrangled and processed, doing exploratory analysis on factors such as missing values
- Categorical column “bad outcomes” was used to create a new column detailing if a launch was successful
- Calculated the success rate with this info
- https://github.com/JackFeen/Coursera/blob/main/Data%20Science/labs-jupyter-spacex-Data%20wrangling-v2%20(1).ipynb

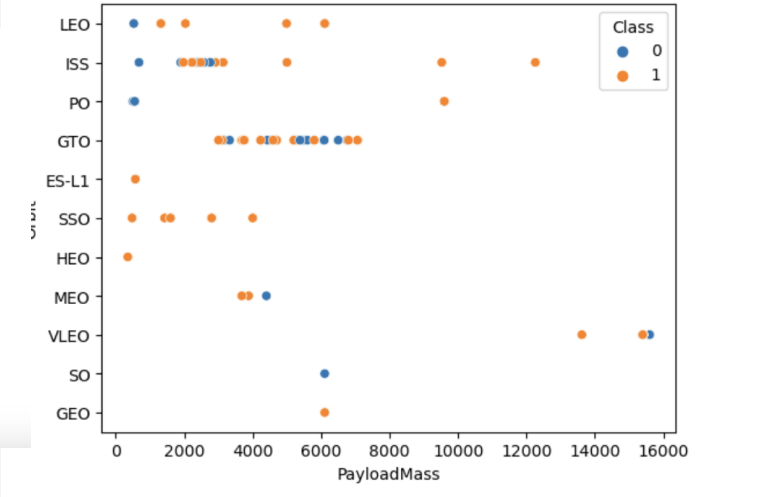
EDA with Data Visualization

- Here are a few charts we plotted, and why we plotted them
- https://github.com/JackFeen/Coursera/blob/main/Data%20Science/jupyter-labs-eda-dataviz-v2%20(1).ipynb

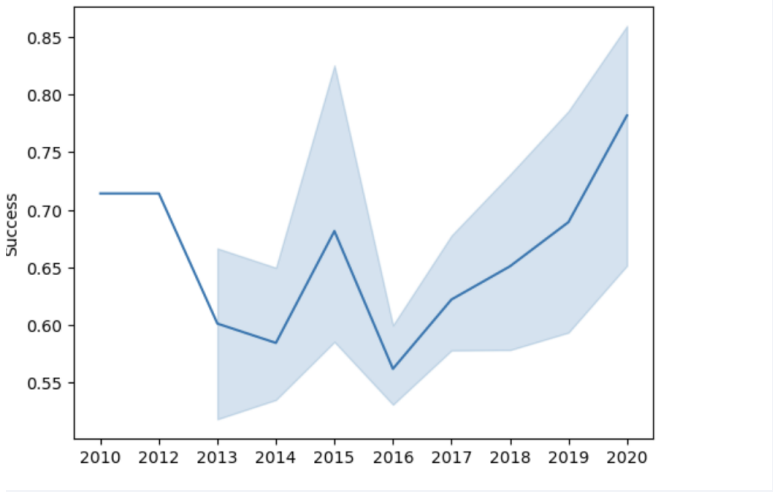
Shows which orbits have a high Success rate



Attempt to find relationship between flight number and orbit type



Analyze trends in success rate by year



EDA with SQL

- SQL queries were used to do the following
  - Create and drop tables
  - Find Unique launch sites
  - Find total payload mass of NASA boosters
  - Find average payload mass for specific boosters
  - List successful and failed mission outcomes
- https://github.com/JackFeen/Coursera/blob/main/Data%20Science/jupyter-labs-eda-sql-coursera_sqllite%20(1)%20(1).ipynb

Build an Interactive Map with Folium

- Added launch sites to a map, including their successes and failures by color to see trends in geolocation relating to successes
- Examined the distance of a launch site to its proximities
  - Are launch sites near coastlines?
  - Are launch sites near railroads?
- https://github.com/JackFeen/Coursera/blob/main/Data%20Science/lab-jupyter-launch-site-location-v2%20(2).ipynb

Build a Dashboard with Plotly Dash

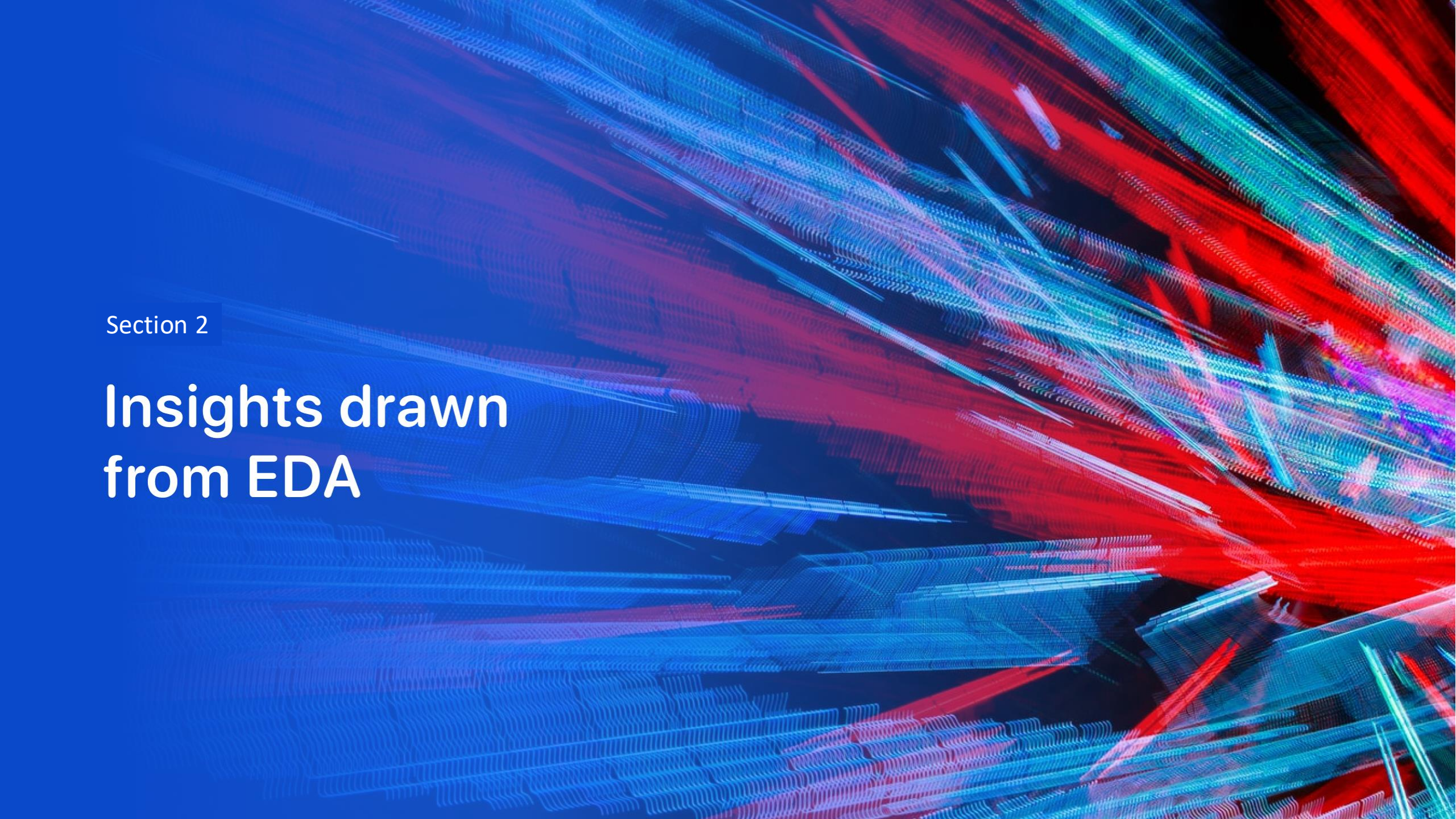
- The interactive plotly dashboard showcases a pie chart and scatterschart
- The pie chart shows the proportion of successful to failed launches
- The scatterplot showcases outcomes by landing mass and booster version
- https://github.com/JackFeen/Coursera/blob/main/Data%20Science/Plotly%20Dash%20App.py

Predictive Analysis (Classification)

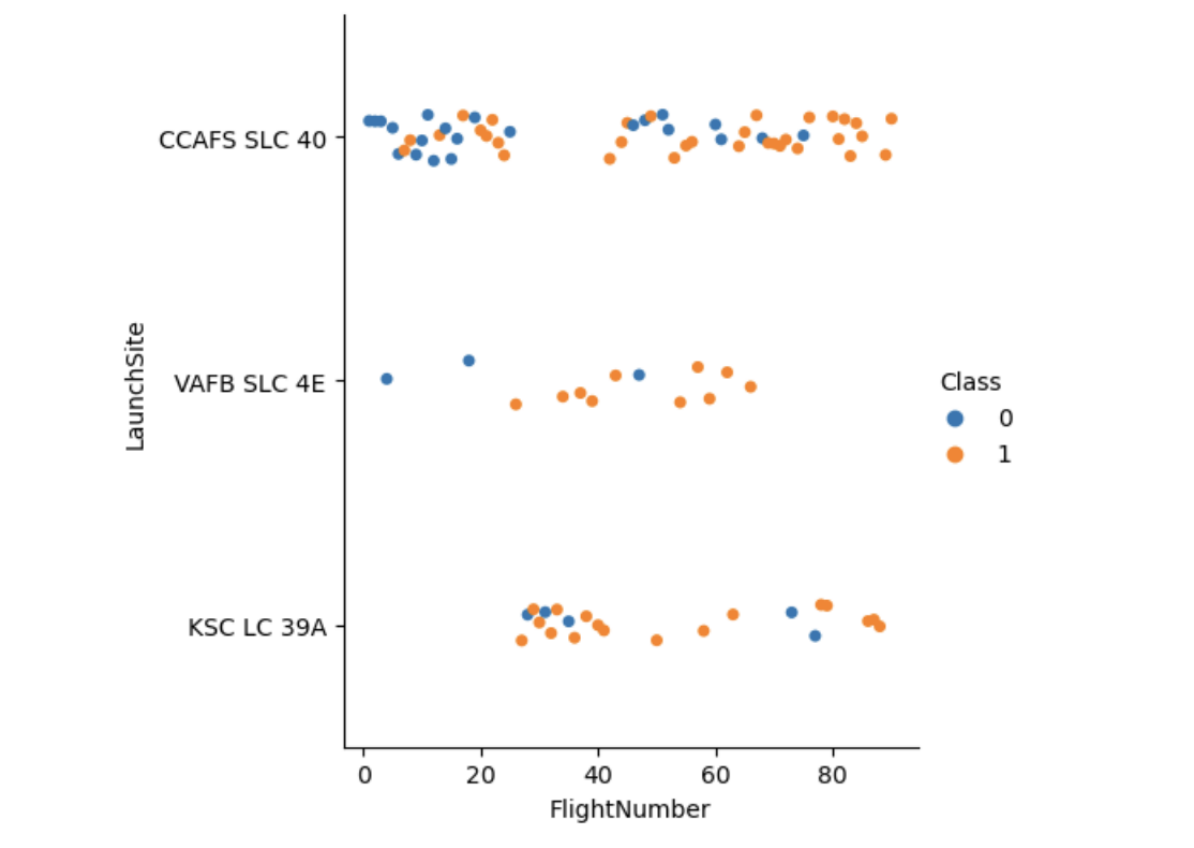
- The data was split into training and testing sets
- Several models were used to train the data, including logistic regression, descision tree, SVM, and KNN
- Grid Search CV was used to determine the best hyperparameters
- Models Were scored for accuracy
- https://github.com/JackFeen/Coursera/blob/main/Data%20Science/SpaceX-Machine-Learning-Prediction-Part-5-v1%20(1).ipynb

Results

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


Section 2
Insights drawn from EDA

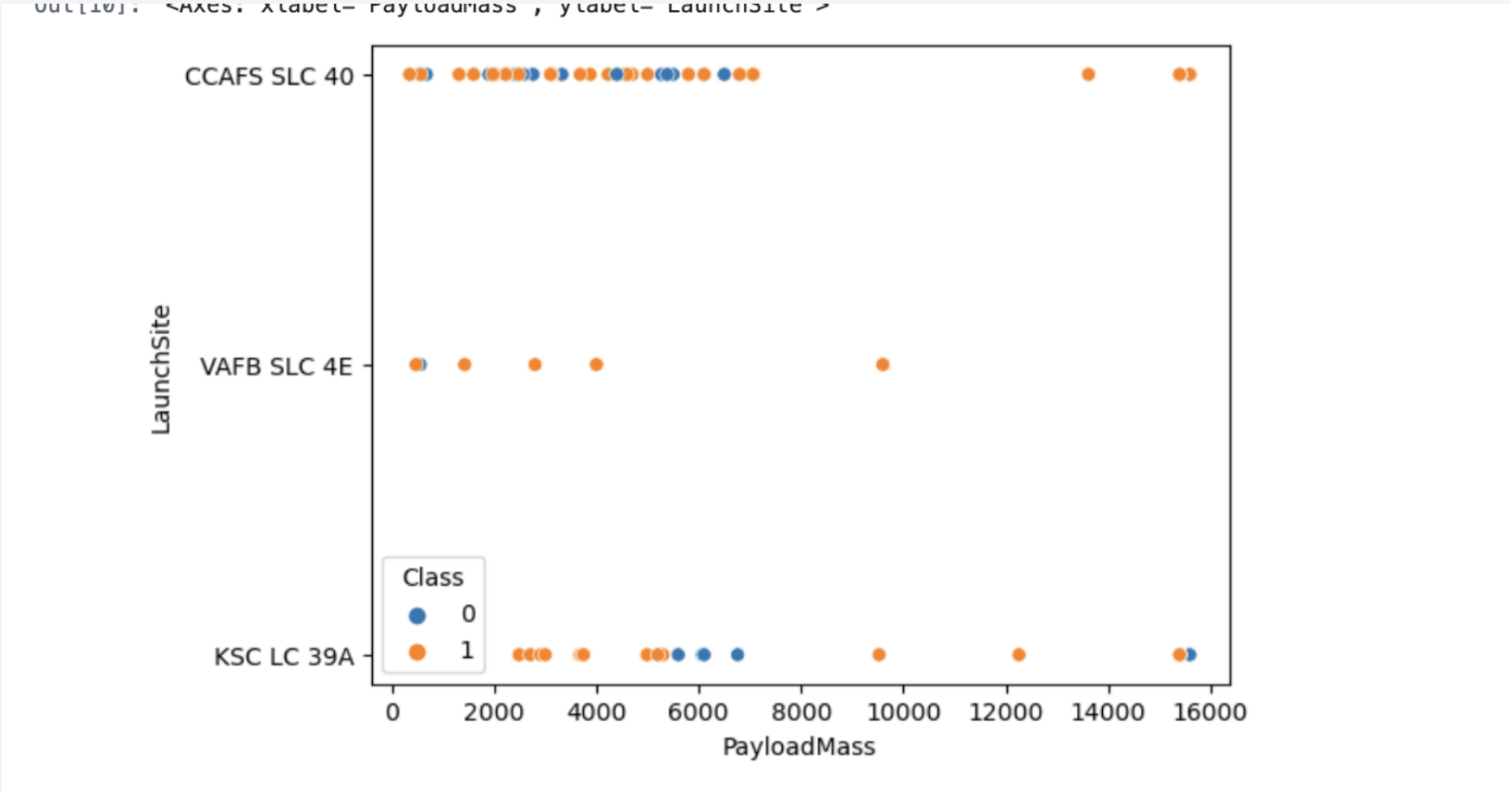
Flight Number vs. Launch Site



Greater Flight Numbers are associated with better outcomes

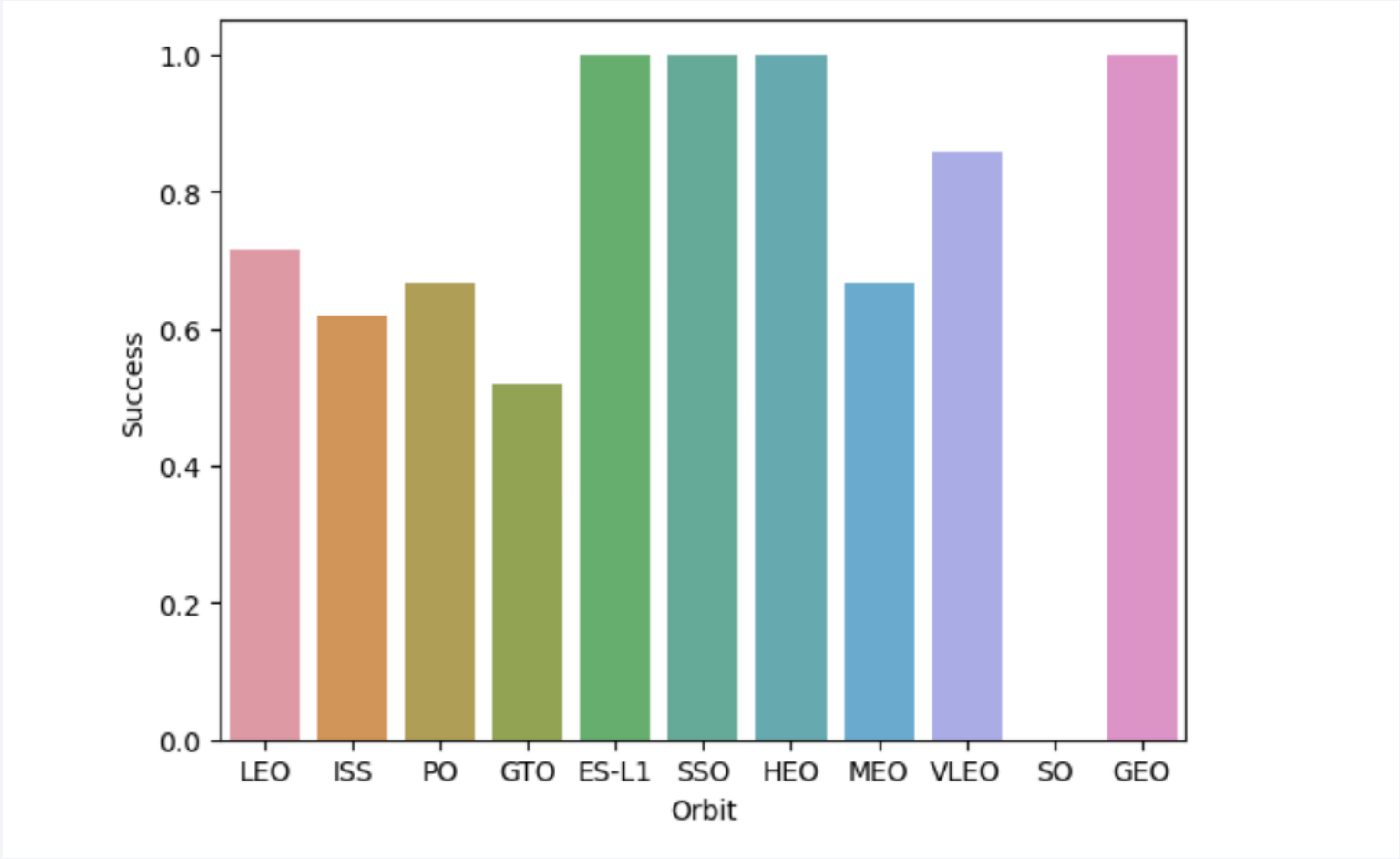
Payload vs. Launch Site

Higher payload masses are moderately associated with better flight results



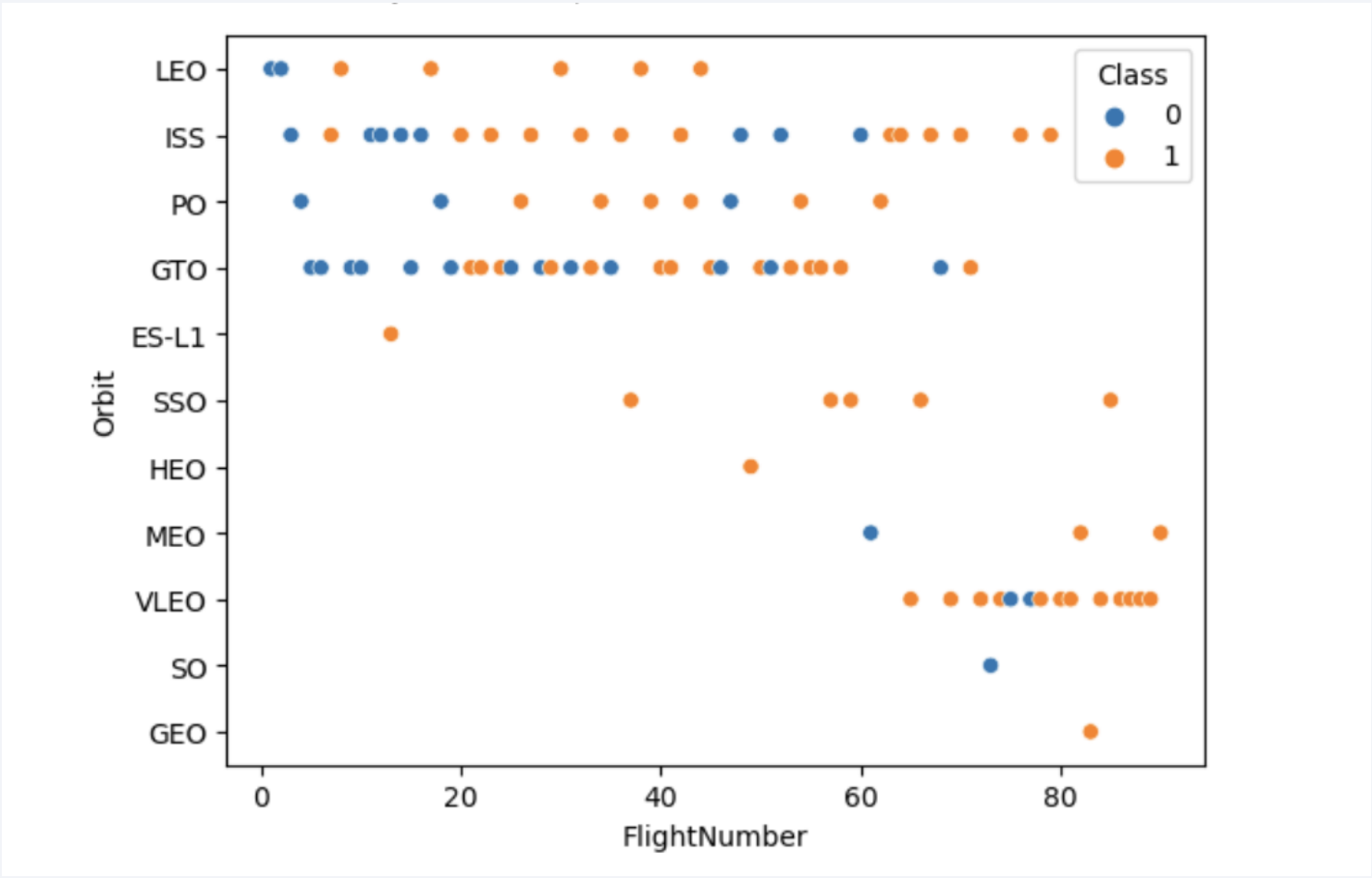
Success Rate vs. Orbit Type

- ES-L1, SSO, HEO, and GEO are the most successful rockets
- SO is the least successful



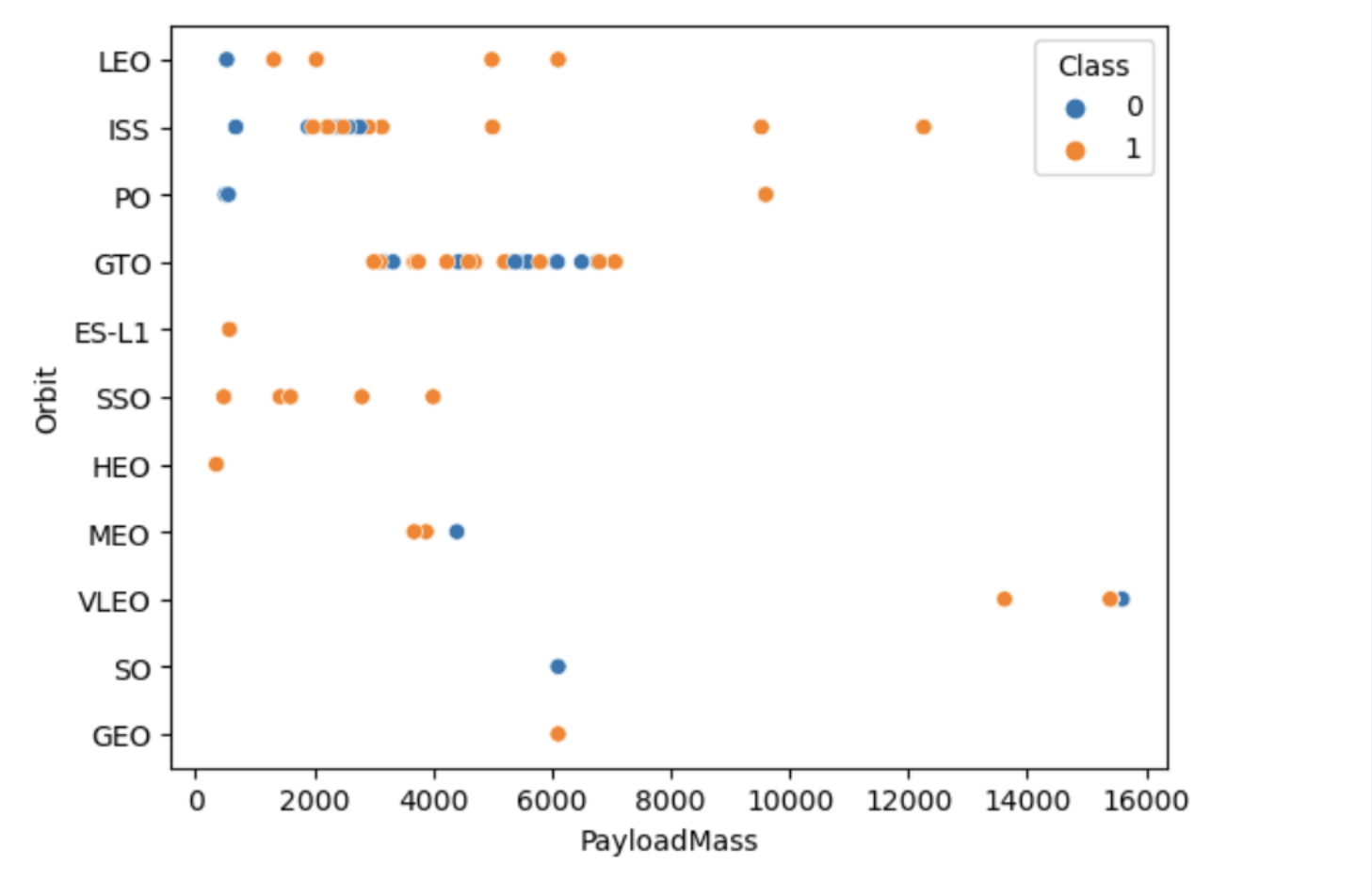
Flight Number vs. Orbit Type

- The more successful Orbits happen to be the ones with higher launch numbers



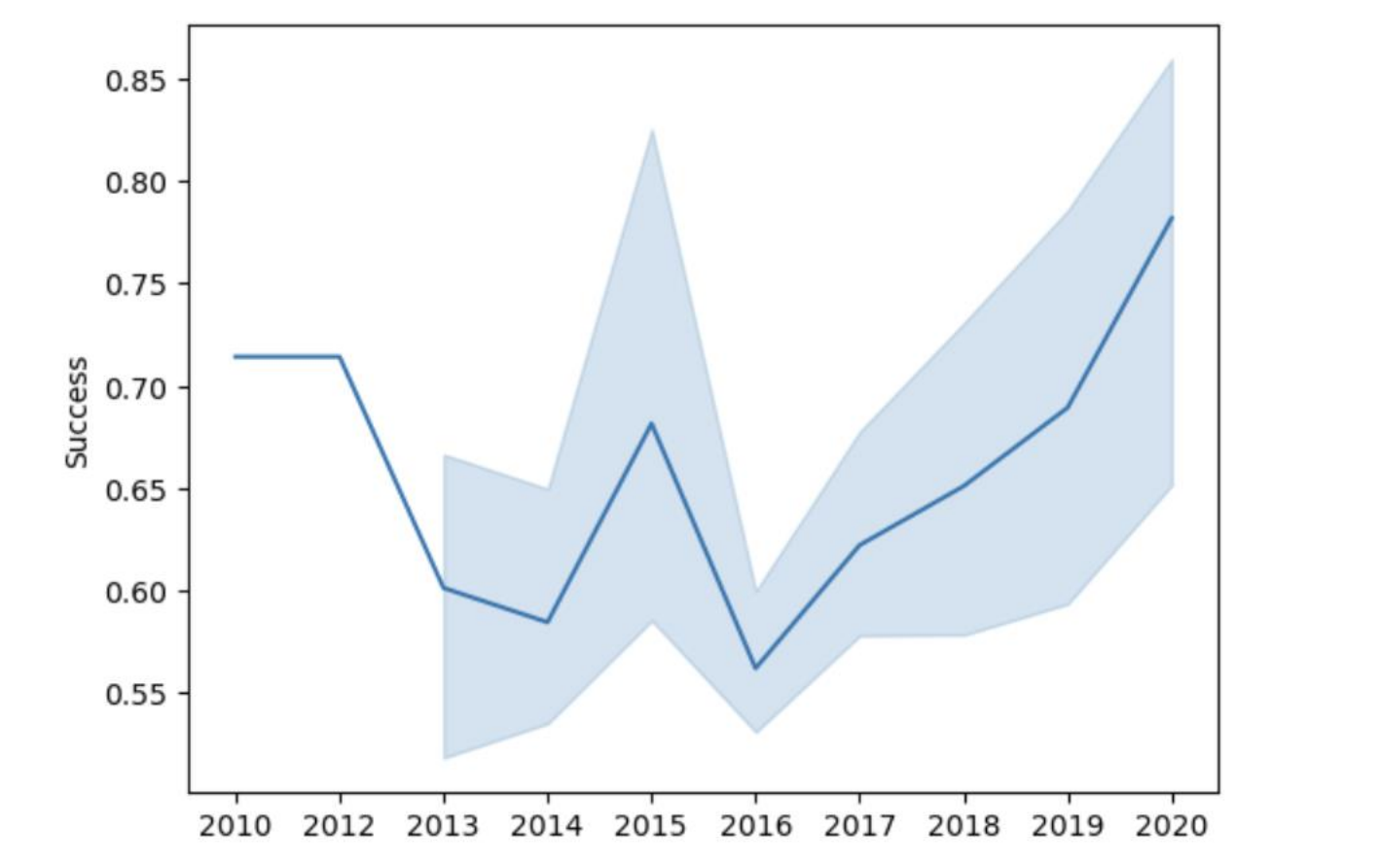
Payload vs. Orbit Type

- PO and ISS are better at doing heavy Payload Masses



Launch Success Yearly Trend

- The yearly trend is positive



All Launch Site Names

- Here are the 4 launch site names

```
n [15]: %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

ut[15]:

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

Launch Site Names Begin with 'CCA'

In [17]:
```
%sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE "CCA%" LIMIT 5;
```

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

Out[17]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Total Payload Mass

Task 3

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

In [20]:
```
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Customer" LIKE "%CRS%"
```

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

Out[20]:

SUM("PAYLOAD_MASS__KG_")
48213

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

Task 4

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

In [22]:
```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Booster_Version" == "F9 v1.1"
```

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

Out[22]:

AVG("PAYLOAD_MASS__KG_")
2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
In [24]: %sql SELECT MIN("Date") FROM SPACEXTABLE;
 * sqlite:///my_data1.db
Done.
Out[24]: MIN("Date")
         2010-06-04
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

In [25]:
```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

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

Out[25]:

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 B4 B1043.1
F9 FT B1032.2
F9 B4 B1040.2
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5B1054

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
In [27]: %sql SELECT COUNT(*), "Mission_Outcome" FROM SPACEXTABLE GROUP BY "Mission_Outcome"
 * sqlite:///my_data1.db
Done.
```

Out[27]:

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

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

In [39]:
```
%sql SELECT "Booster_Version" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM
```

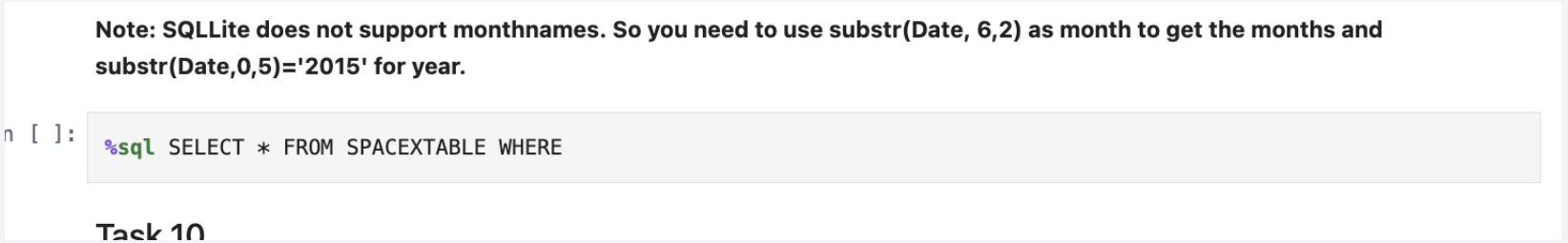
```
 * sqlite:///my_data1.db
Done.
```

Out[39]:

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

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015



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

- 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

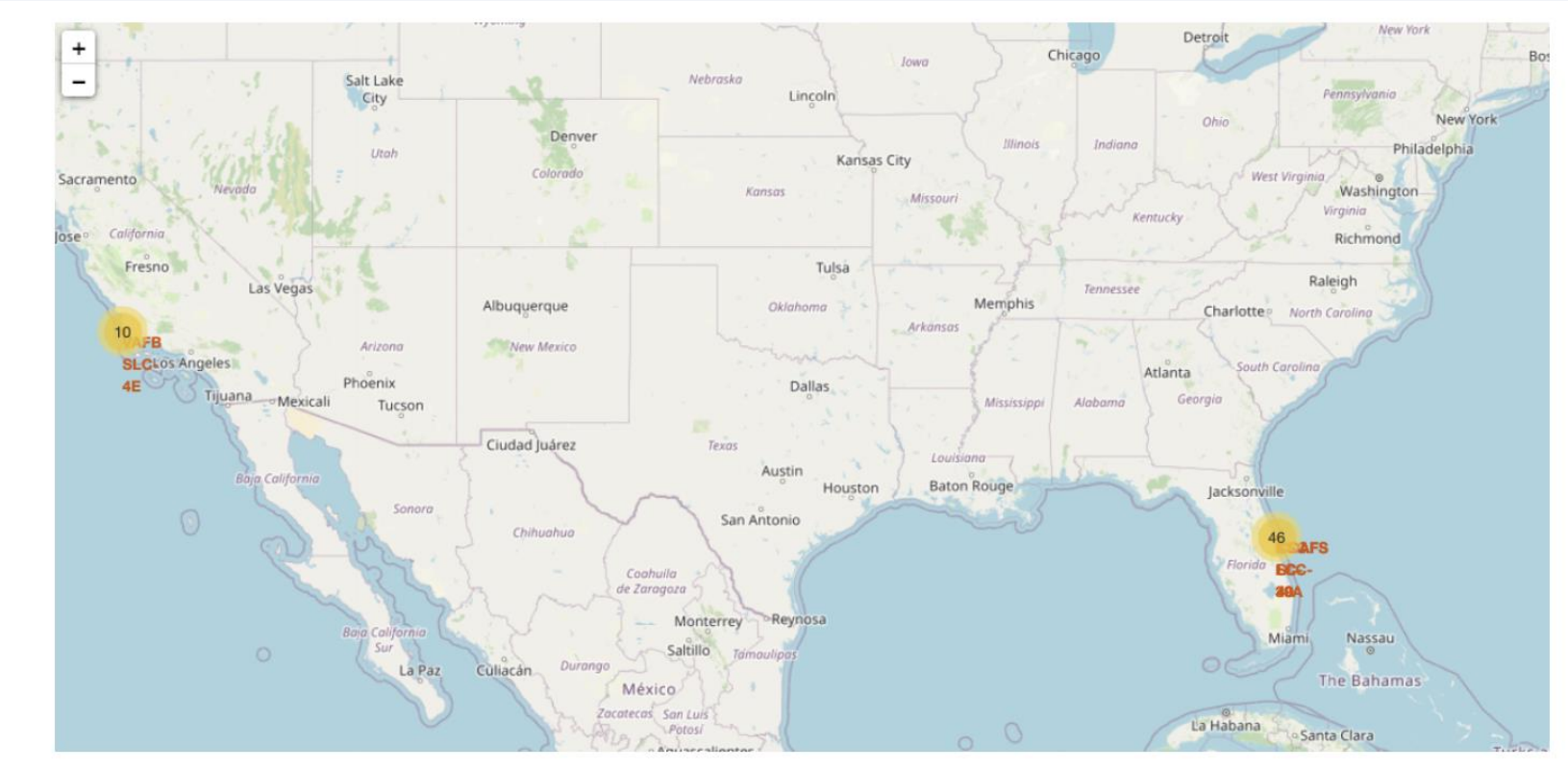
- Present your query result with a short explanation here

Section 3

Launch Sites Proximities Analysis

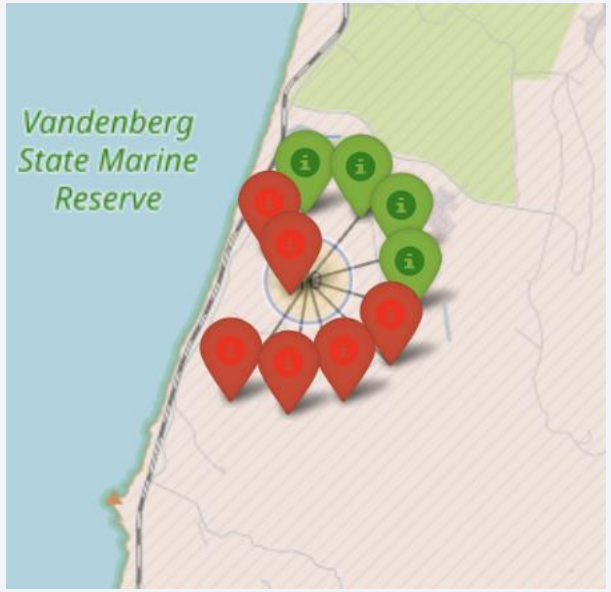
Falcon 9 Launch Site Locations

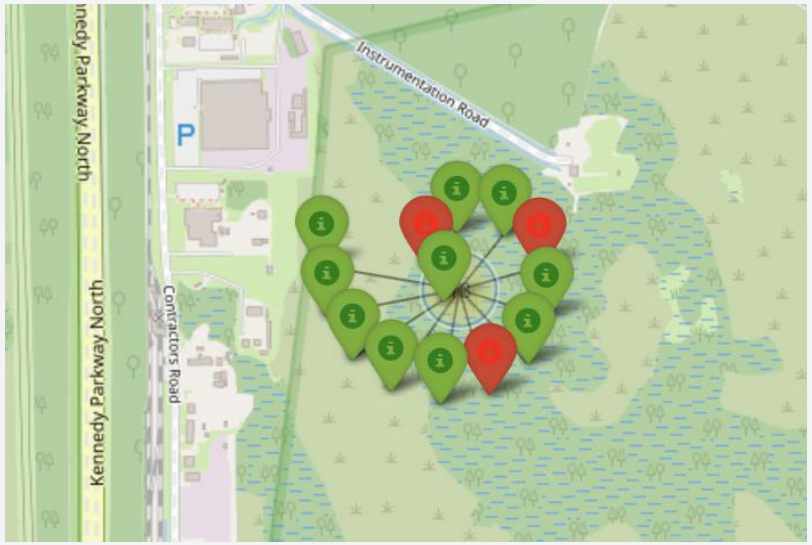
Locations of various falcon 9 launch sites

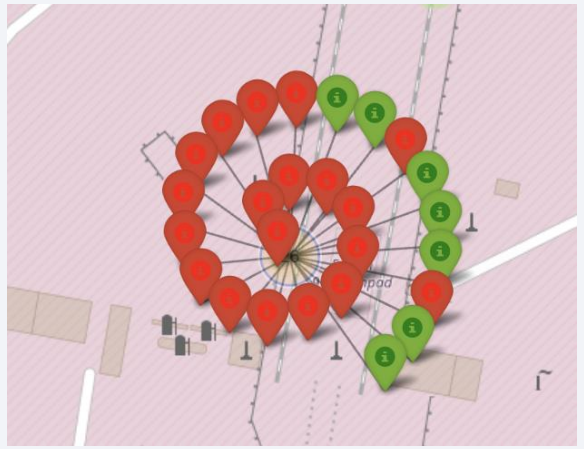


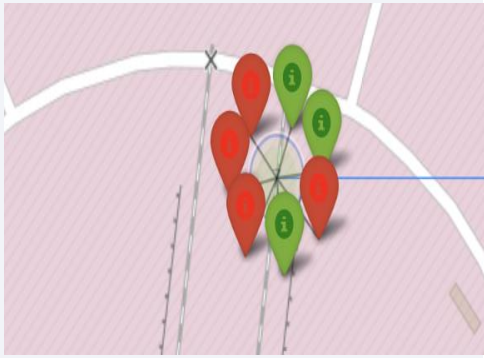
Markers for launch site successes and failures

The green and red indicators show successful and failed launches respectively



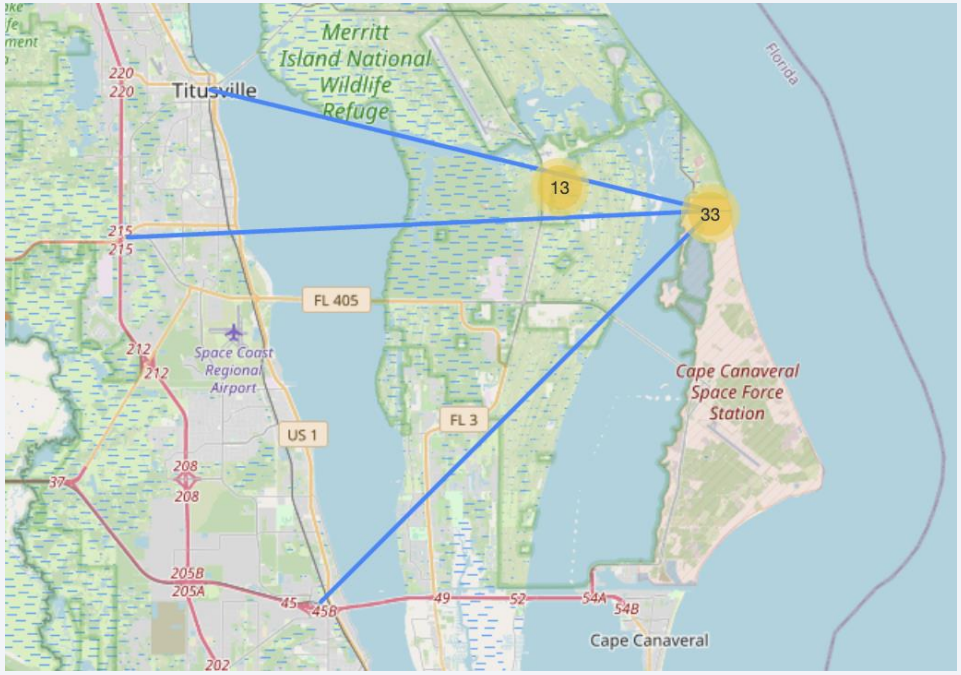


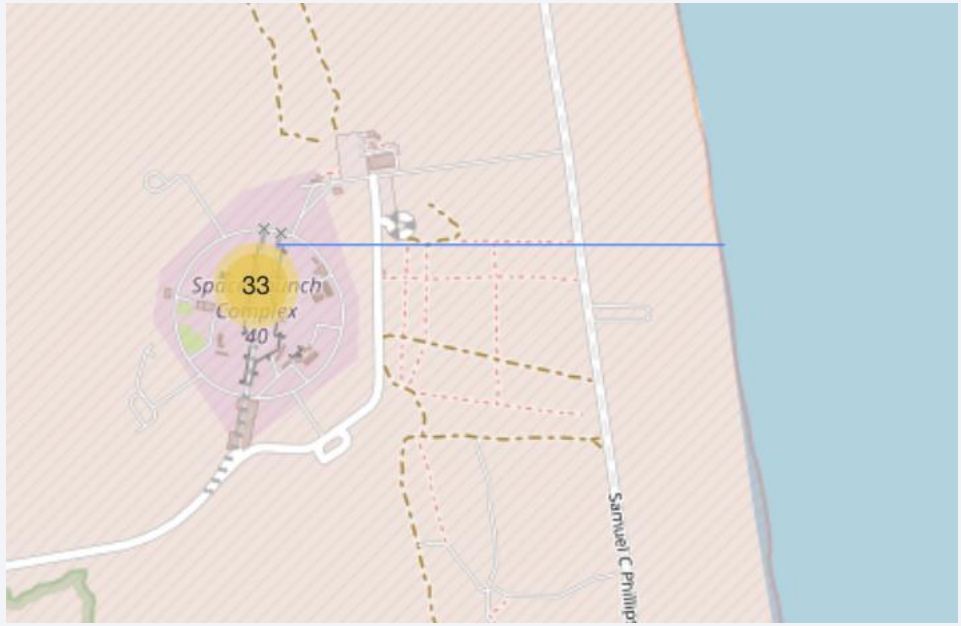


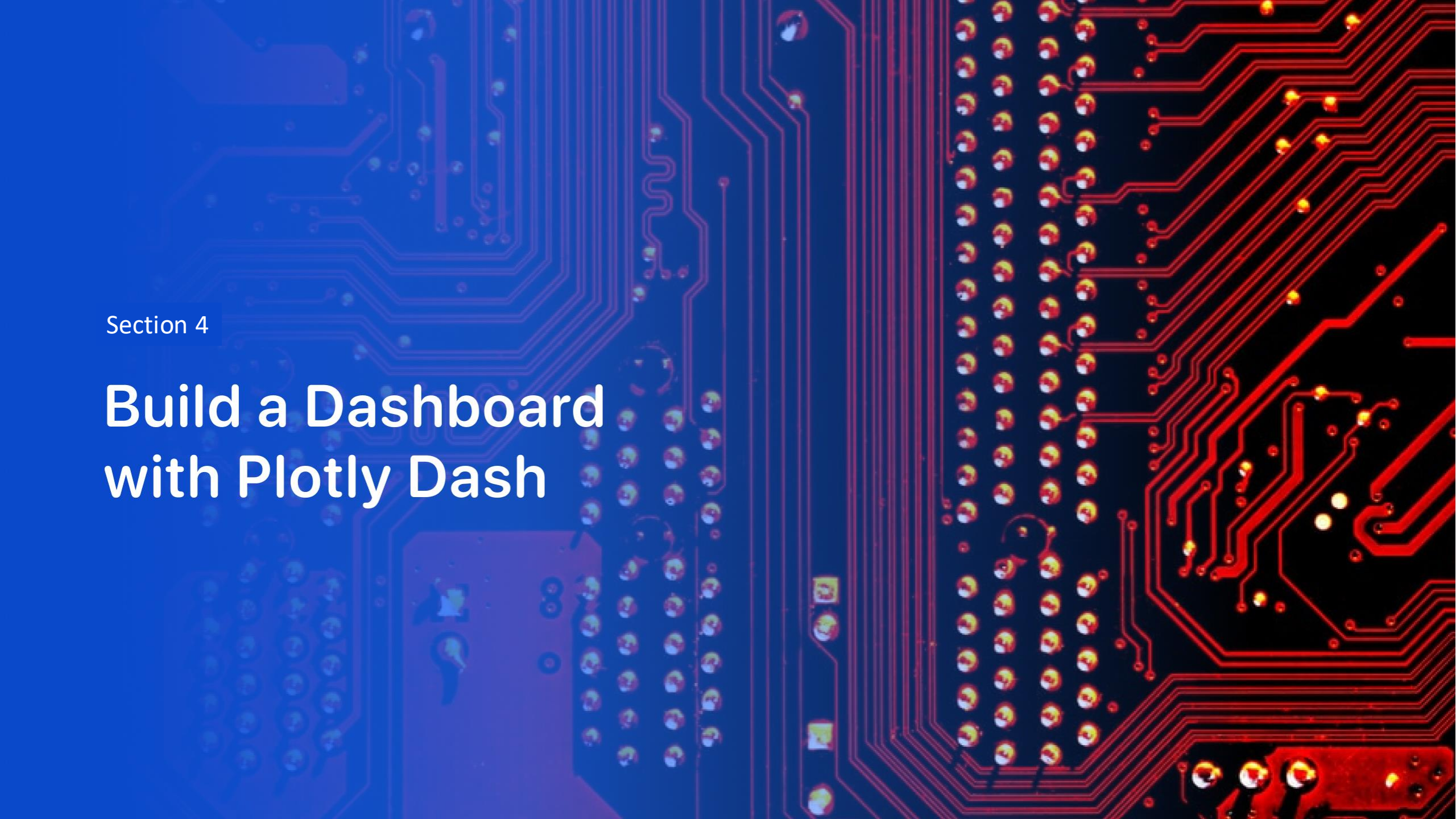


<Folium Map Screenshot 3>

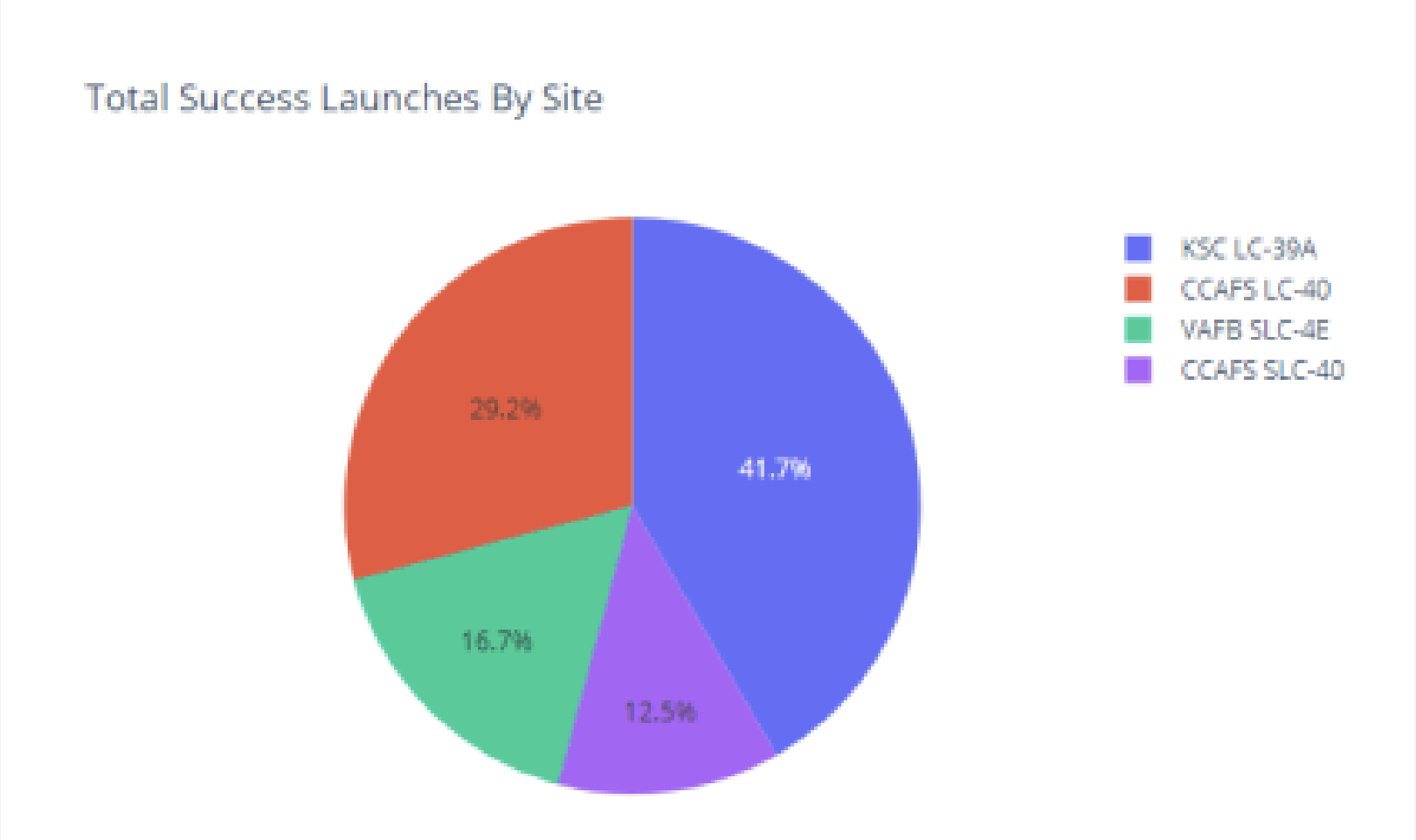
Launch sites appear close to coastlines, but not close to features such as railroads



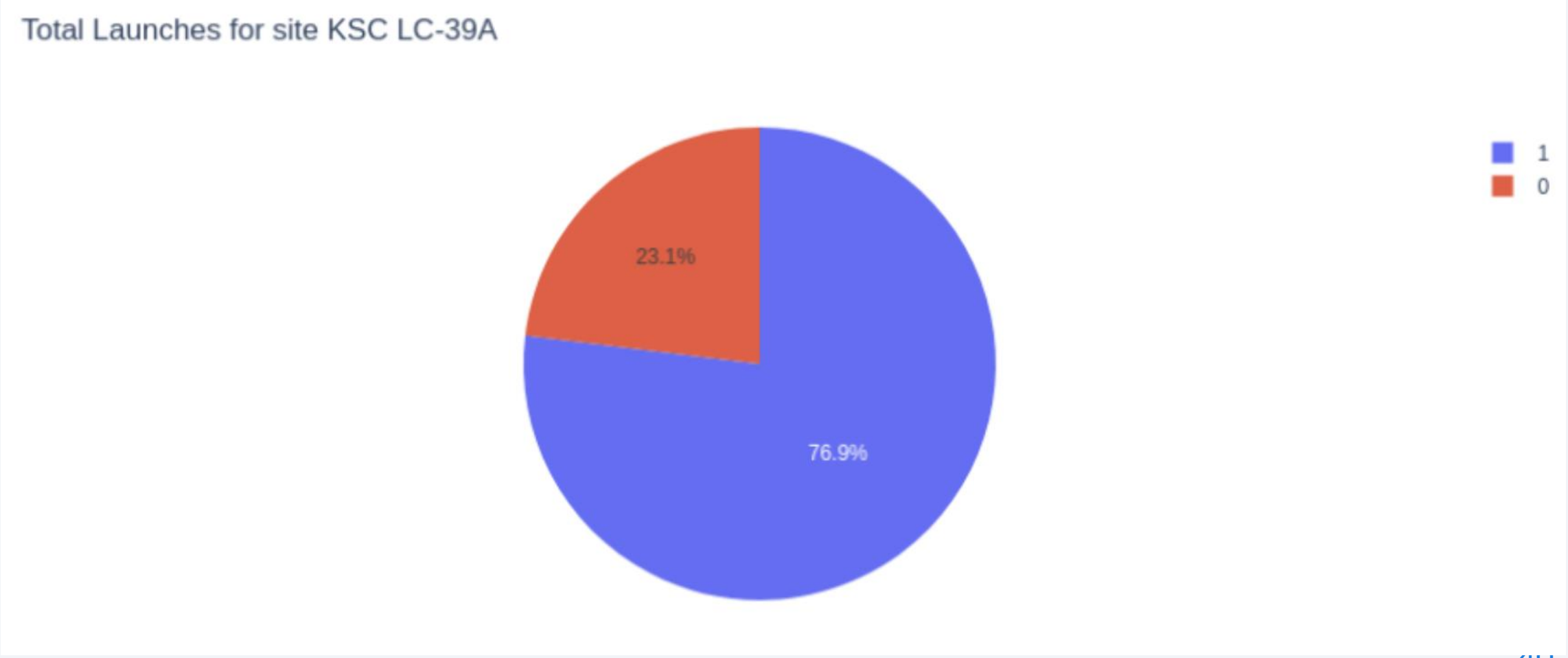



Section 4
Build a Dashboard with Plotly Dash

Launch successes for all sites



Most Successful launch site ratio



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The best model is the decision tree classifier, as seen by the output of this code

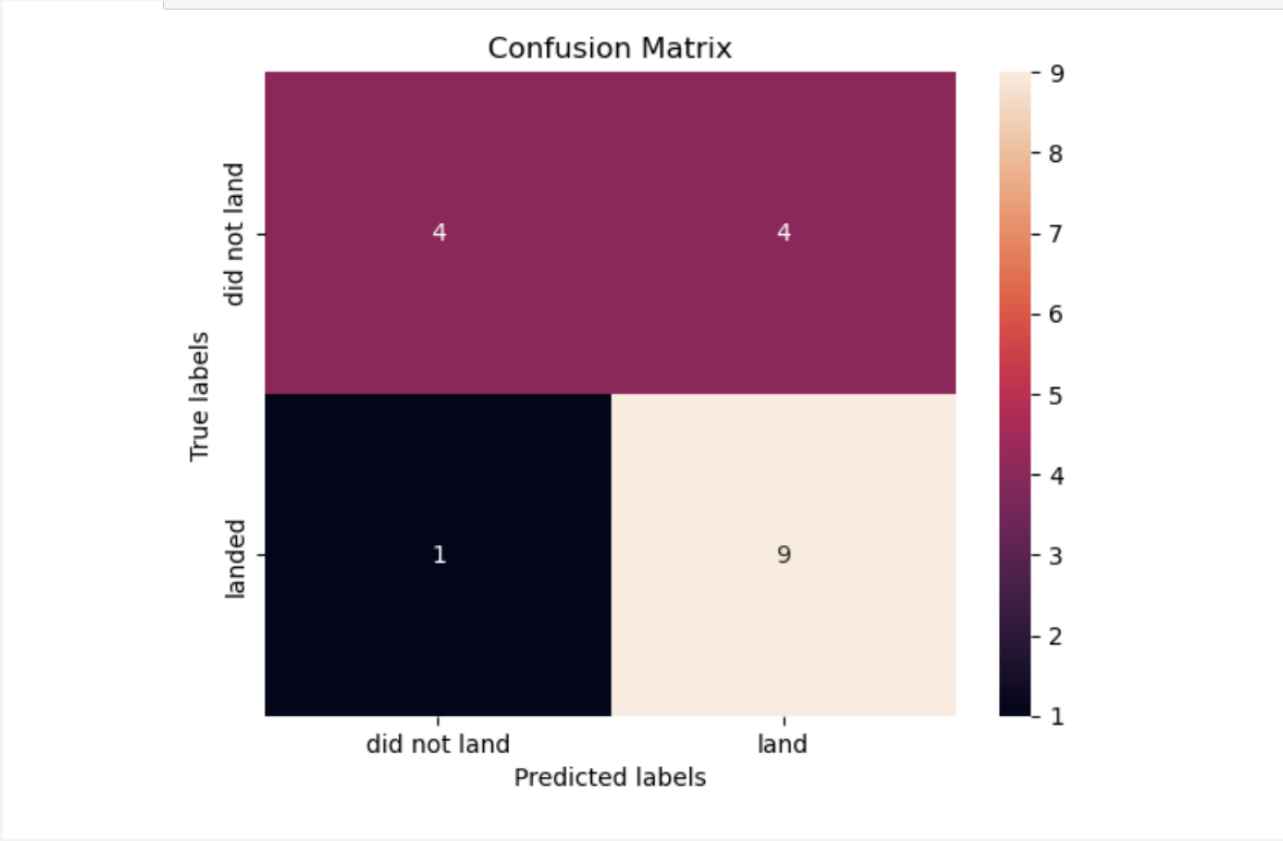
```
rank_dict = {'Model':['Logistic Regression', 'SVM', 'Decision Tree', 'KNN'], 'Score':[logistic_regression_score, svm
rank_df = pd.DataFrame(ranking_dict)

def status(score):
    if score == rank_df['Score'].max():
        return 'Max Score'
    else:
        return '0'

rank_df['Max Score'] = rank_df['Score'].apply(status)
rank_df
```

Confusion Matrix

- This decision tree tends to make the mistake of predicting that the rocket will land, when it actually does not. (False positive, Type I error)
- The model is great at predicting whether a rocket will not land, but a prediction that it lands is less certain



Conclusions

- A decision tree classifer model can be used to predict successful landings
- Falcon 9 outcomes have been steadily improving over time
- Larger payload masses are associated with higher success rates

Thank you!