IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<QIAN WANG>
<2023/04/03>

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- **Summary of all results**
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  Space X promotes Falcon 9 rocket launches for 62 million dollars while other suppliers charge upwards of 165 million dollars for each launch. A large portion of the savings is due to Space X's ability to reuse the first stage. Hence, if we can figure out whether the first stage will land, we can figure out how much a launch will cost. If another business wishes to submit a proposal for a rocket launch against space X, they can use this information. The project's objective is to build a pipeline for machine learning that can forecast if the initial stage will land successfully.

- Problems you want to find answers
  - Key factors for a sucessful landing
  - The interaction between factors that determine the successful landing.
  - Optimal operating conditions.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - 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

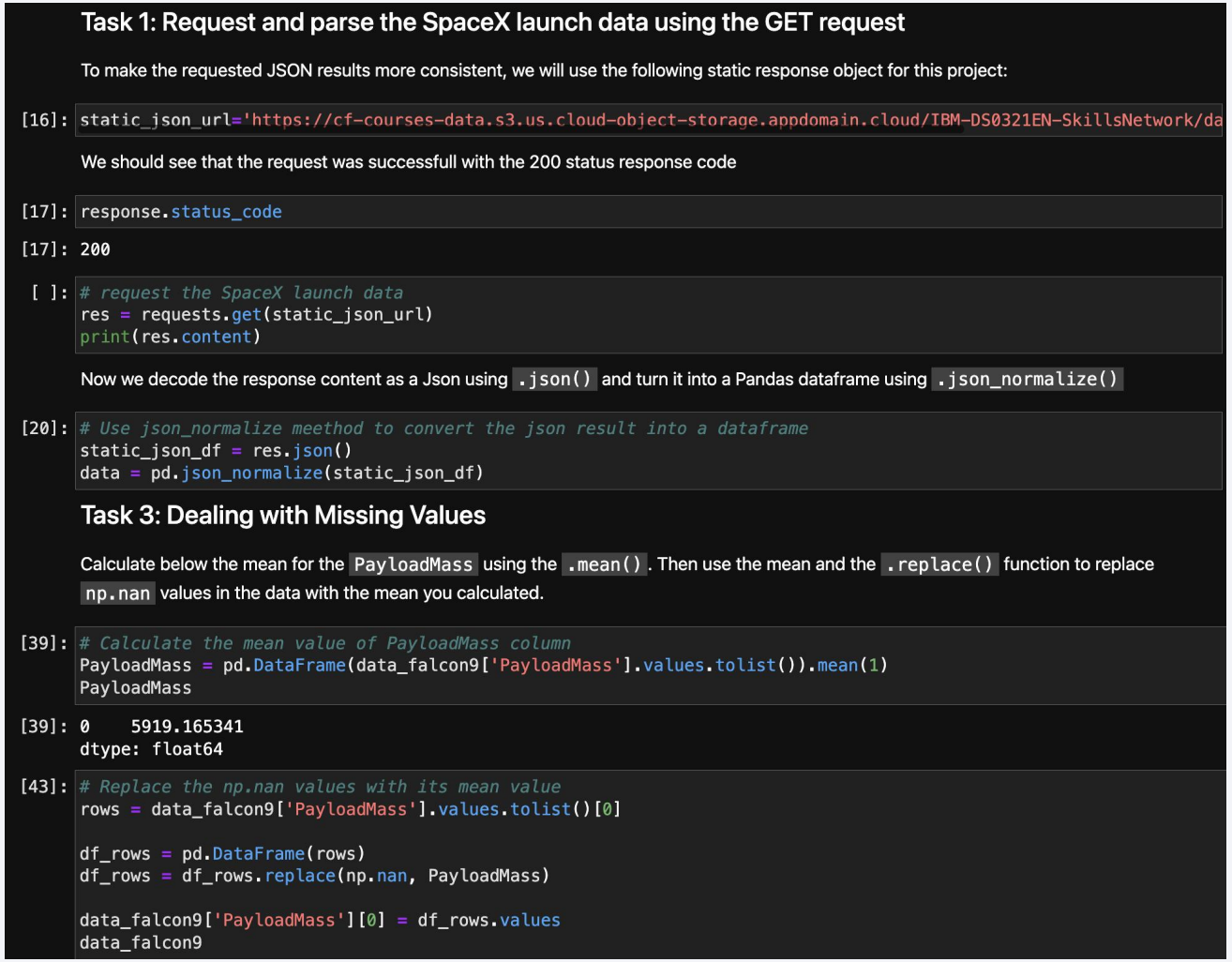
Data Collection

- The data was collected using various methods
    1. Using get request to the SpaceX API
    2. Decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
    3. Cleaned the data, checked for missing values and fill in missing values where necessary.
    4. Performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.

The objective is to retrieve the launch records as HTML table, parse the table and prepare it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- Get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is : https://github.com/ShirleyUna/IBM-Certificate-Final-Project-SpaceX/blob/main/Data_Cllection_API_SpaceX.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:

[16]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

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

[17]:
```
response.status_code
```

[17]:
```
200
```

[ ]:
```
# request the SpaceX launch data
res = requests.get(static_json_url)
print(res.content)
```

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

[20]:
```
# Use json_normalize meethod to convert the json result into a dataframe
static_json_df = res.json()
data = pd.json_normalize(static_json_df)
```

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

[39]:
```
# Calculate the mean value of PayloadMass column
PayloadMass = pd.DataFrame(data_falcon9['PayloadMass'].values.tolist()).mean(1)
PayloadMass
```

[39]:
```
0    5919.165341
dtype: float64
```

[43]:
```
# Replace the np.nan values with its mean value
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is: https://github.com/ShirleyUna/IBM-Certificate-Final-Project-SpaceX/blob/main/Web_Scrapping_SpaceX.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.

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

[5]: 200

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

[6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

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

[7]: <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]
print(first_launch_table)
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

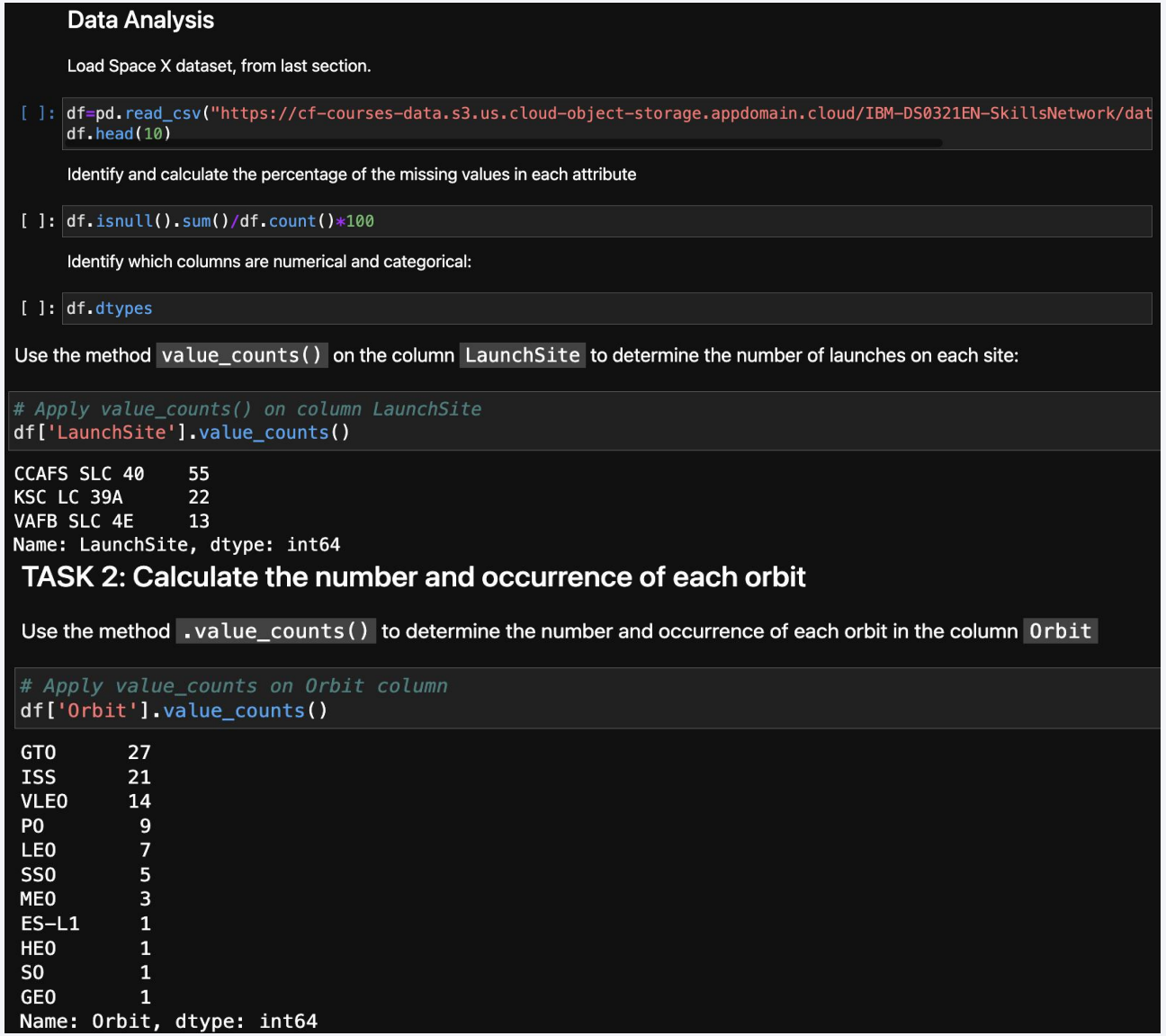
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

Data Wrangling

- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site, and the number and occurrence of each orbits
- Created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is: https://github.com/ShirleyUna/IBM-Certificate-Final-Project-SpaceX/blob/main/Data_Wrangling_SpaceX.ipynb



Data Analysis

Load Space X dataset, from last section.

```
[ ]: df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/dat
     df.head(10)
```

Identify and calculate the percentage of the missing values in each attribute

```
[ ]: df.isnull().sum()/df.count()*100
```

Identify which columns are numerical and categorical:

```
[ ]: df.dtypes
```

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

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

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

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

TASK 3: Calculate the number and occurence of mission outcome per orbit type

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes`.Then assign it to a variable landing_outcomes.

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

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

[9]:
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

[10]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

[10]:
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

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`:

[11]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
#landing_class = [x for x in bad_outcomes if df['Outcome'][x] ]

landing_class = []

for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

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

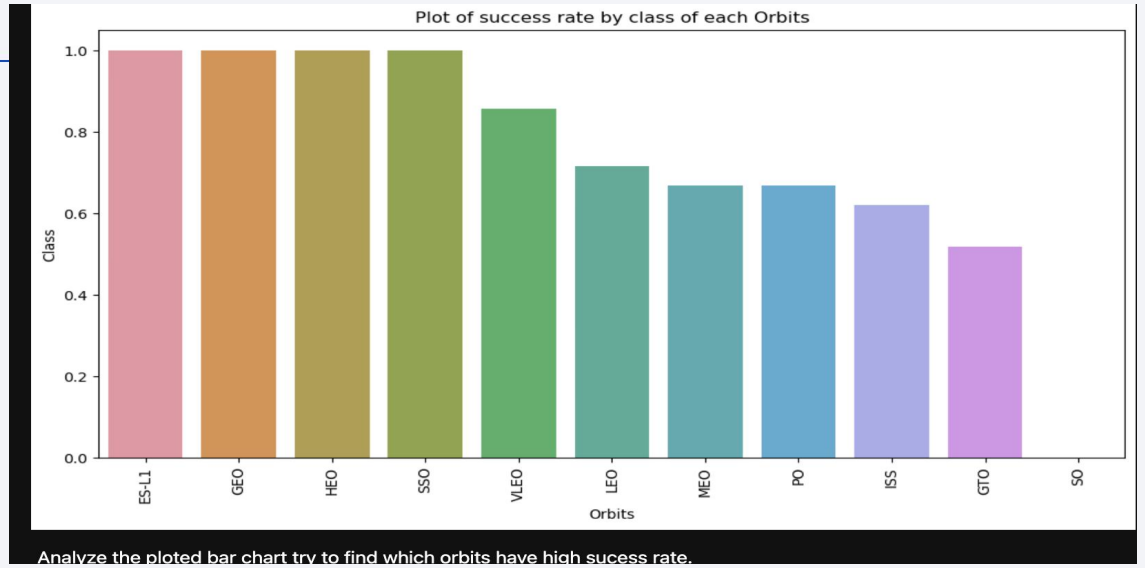
We can use the following line of code to determine the success rate:

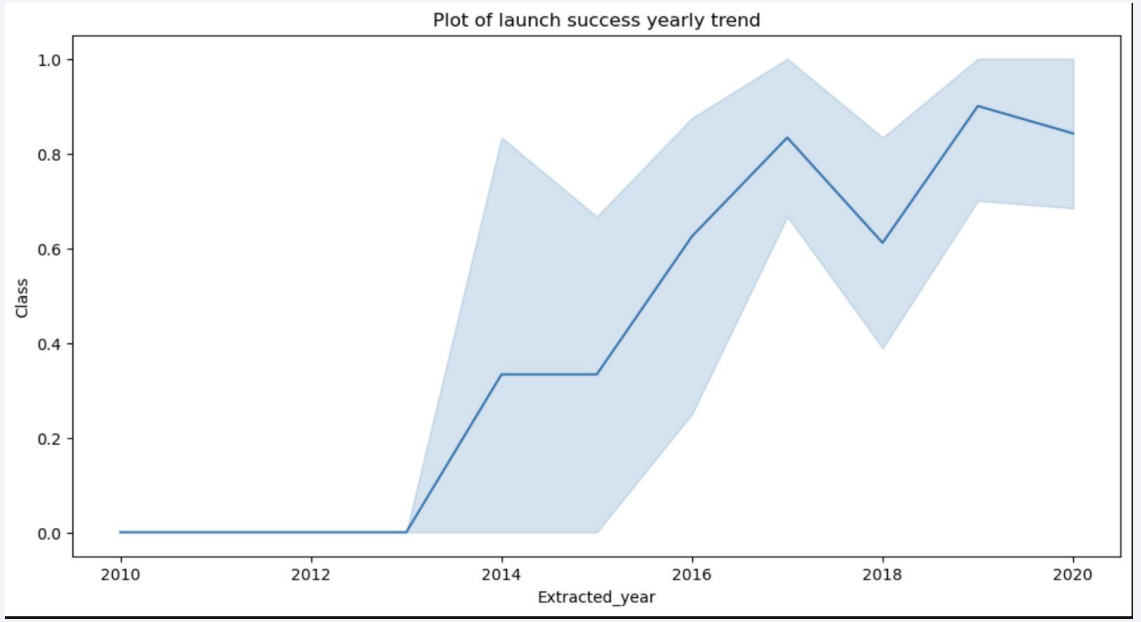
[14]:
```
df["Class"].mean()
```

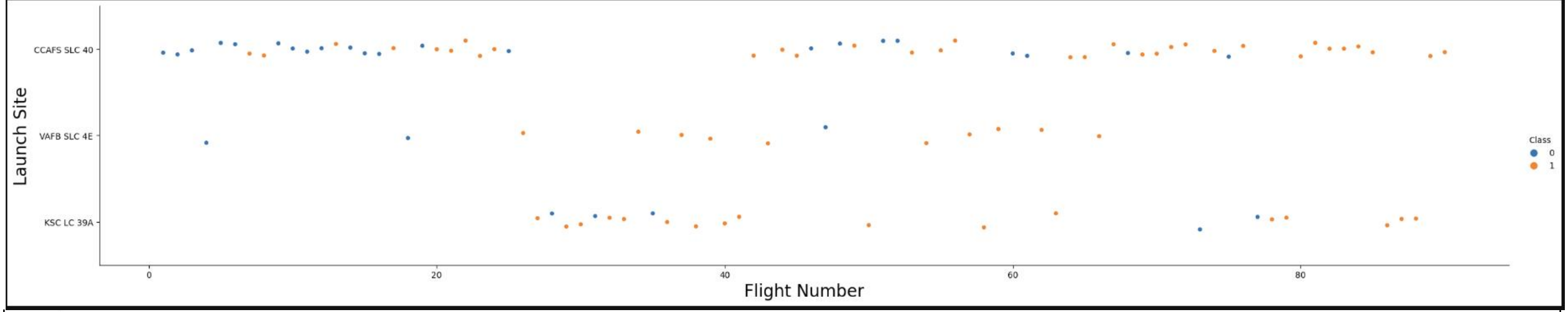
[14]:
```
0.6666666666666666
```

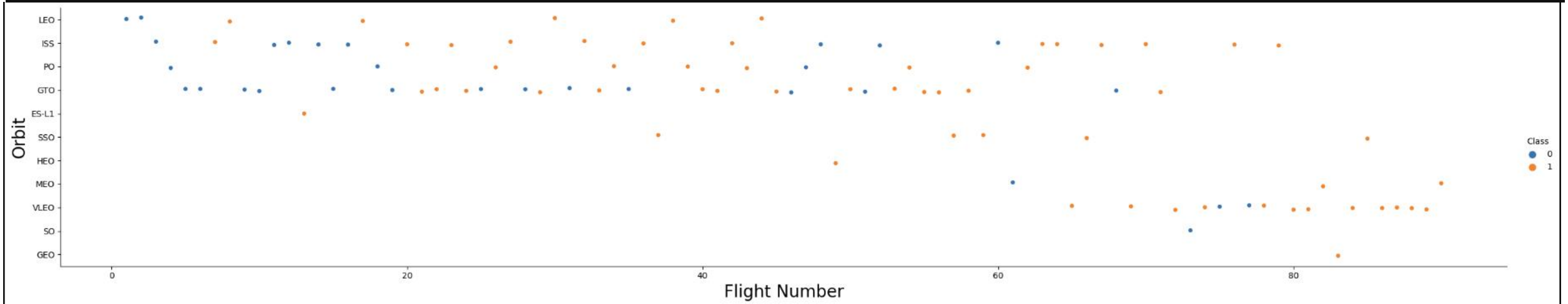
EDA with Data Visualization

- We examined the data 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. We choose these important factors to pick the most influential one that contributes to the final launch.
- The link to the notebook is: https://github.com/ShirleyUna/IBM-Certificate-Final-Project-SpaceX/blob/main/Data_Visualization_SpaceX.ipynb






Launch Site
CCAFS SLC 40
VAFB SLC 4E
KSC LC 39A
0
20
40
60
80
Flight Number
Class
0
1


Orbit
LEO
ISS
PO
GTO
ES-L1
SSO
HEO
MEO
VLEO
SO
GEO
0
20
40
60
80
Flight Number
Class
0
1


Orbit
LEO
ISS
PO
GTO
ES-L1
SSO
HEO
MEO
VLEO
SO
GEO
0
2000
4000
6000
8000
10000
12000
14000
16000
PayloadMass
Class
0
1

EDA with SQL

- We applied EDA with SQL and input queries to find out for instance:

- The names of unique launch sites in the space mission.

- The total payload mass carried by boosters launched by NASA (CRS)

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

- The total number of successful and failure mission outcomes

- The failed landing outcomes in drone ship, their booster version and launch site names.

- The link to the notebook is : https://github.com/ShirleyUna/IBM-Certificate-Final-Project-SpaceX/blob/main/Sql_SpaceX.ipynb

Build an Interactive Map with Folium

- We marked all launch sites on the map, and added distinct objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color–labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Does the launch site near railways, highways and coastlines?
  - Does launch site keep certain distance away from cities?

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/ShirleyUna/IBM-Certificate-Final-Project-SpaceX/blob/main/app.py

Predictive Analysis (Classification)

- We explored the data using numpy and pandas, transformed the data, split our data into training and testing.
- We tested different machine learning models and tune different hyperparameters using GridSearchCV.
- We tested accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We picked the best performing classification model.
- The link to the notebook is https://github.com/ShirleyUna/IBM-Certificate-Final-Project-SpaceX/blob/main/Machine_Learning_Prediction_SpaceX.ipynb

Results

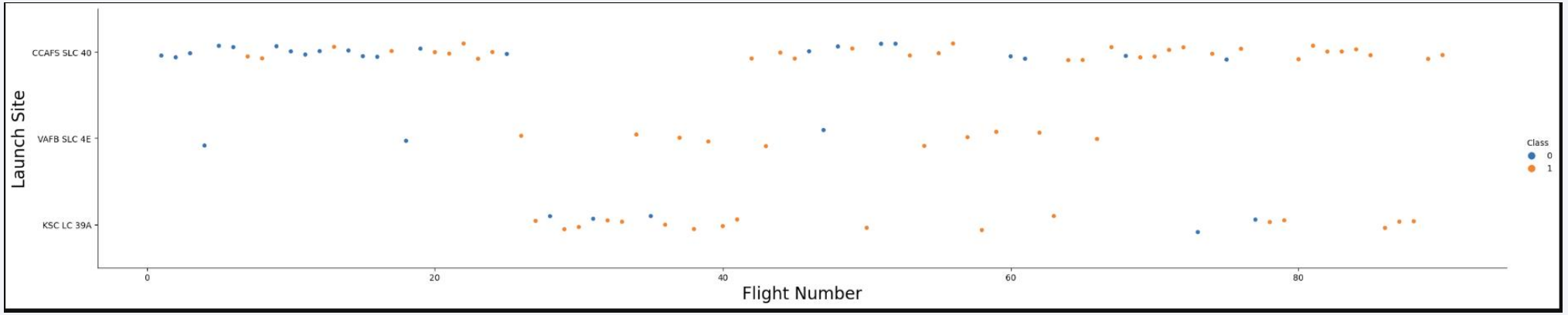
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

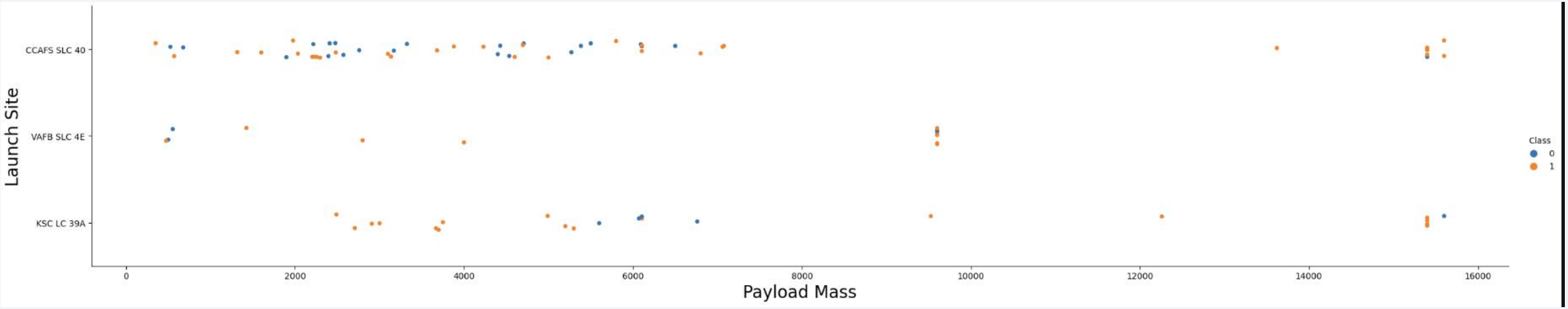
- Show a scatter plot of Flight Number vs. Launch Site



The plot indicates that the larger the flight amount at a launch site, the greater the success rate at a launch site.

Payload vs. Launch Site

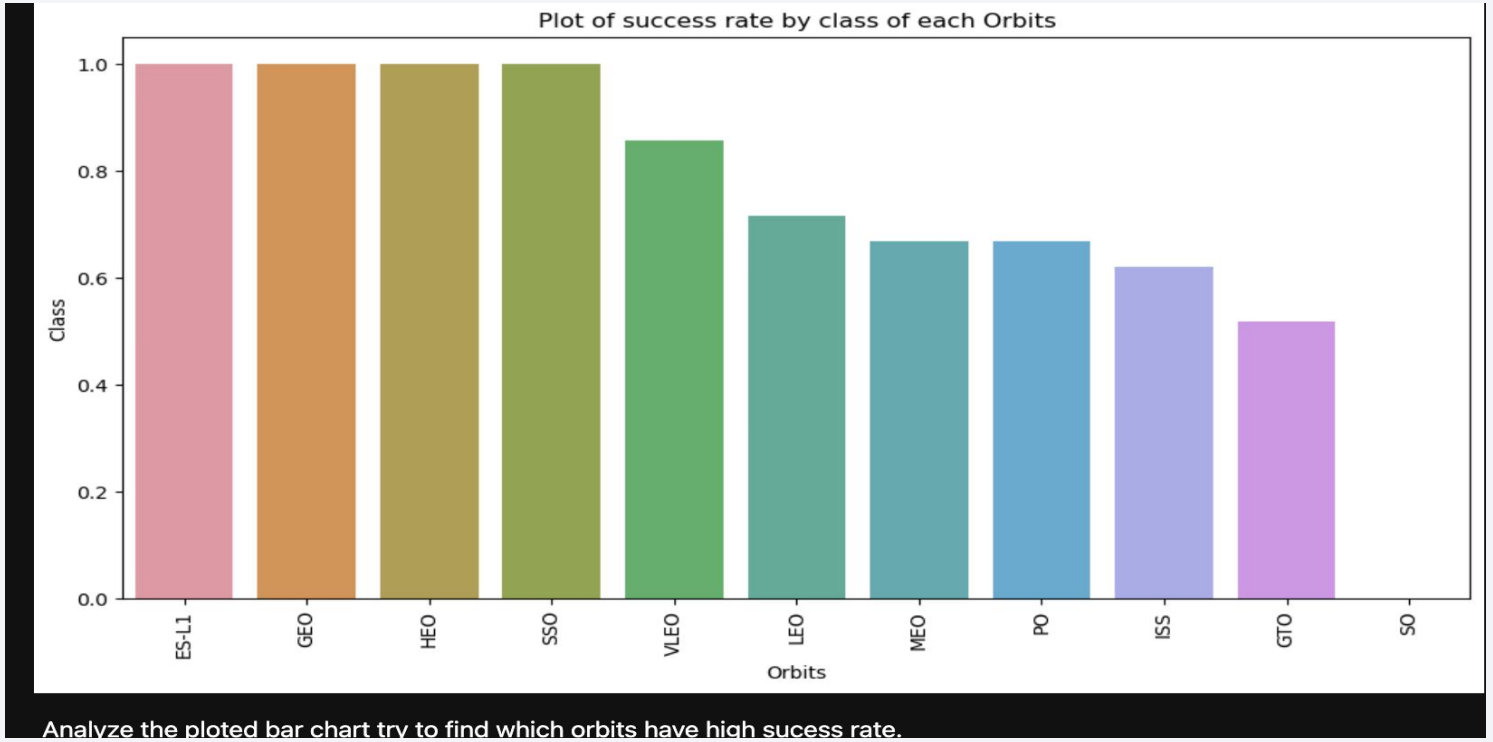
- Show a scatter plot of Payload vs. Launch Site



The plot indicates that the greater the payload mass forlaunch site CCAFS SLC 40 the higherthe success rate for the rocket.

Success Rate vs. Orbit Type

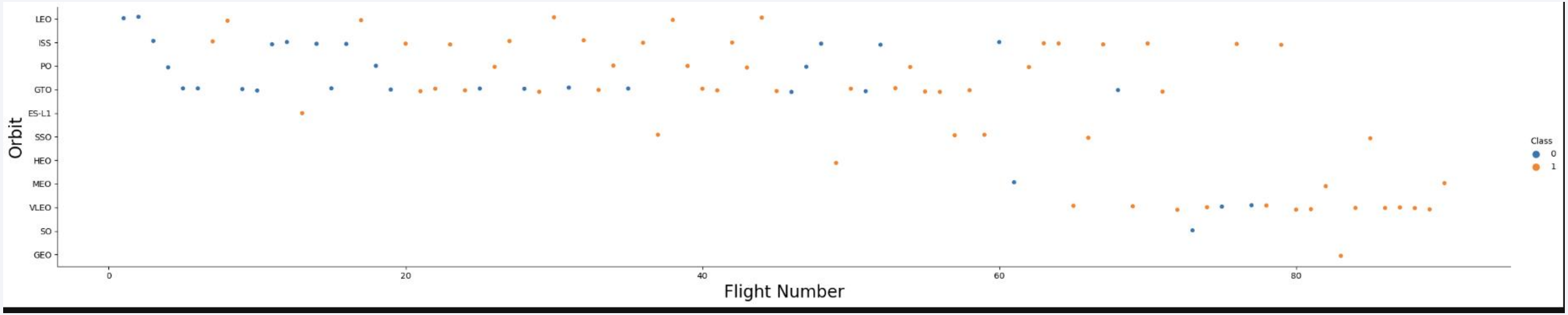
- Show a bar chart for the success rate of each orbit type



The plot indicates that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

Flight Number vs. Orbit Type

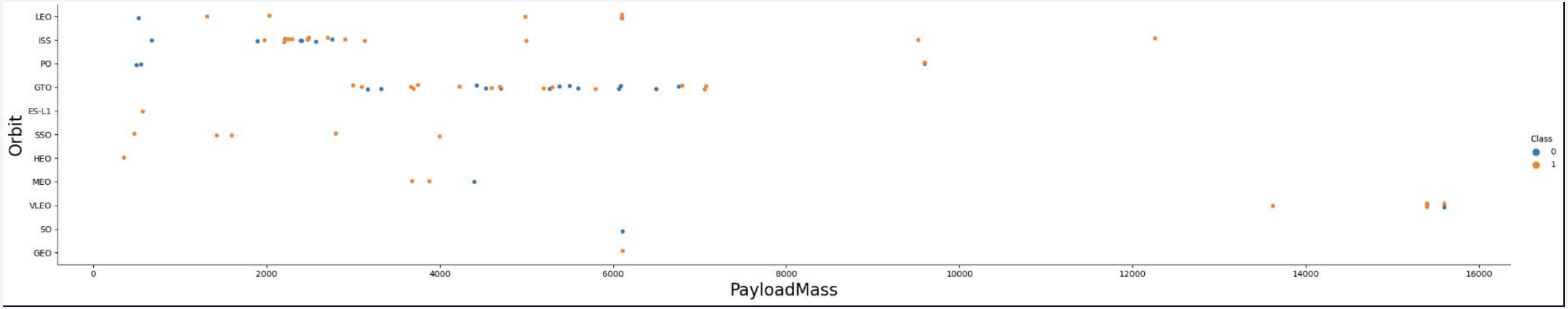
- Show a scatter point of Flight number vs. Orbit type



The plot indicates that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

Payload vs. Orbit Type

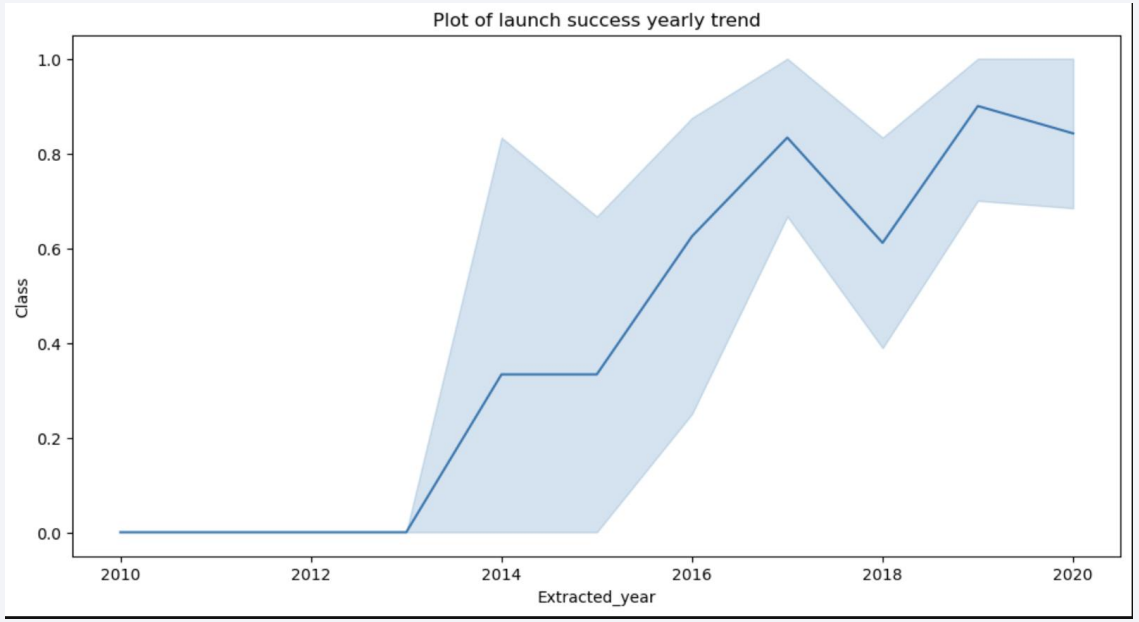
- Show a scatter point of payload vs. orbit type



We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate



The plot shows a continuously increasing success rate since 2013 till 2020.

All Launch Site Names

- Find the names of the unique launch sites

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

```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

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

```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

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

```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

	total_payloadmass
0	45596

The total payload carried by boosters from NASA as 45596 using the query.

Average Payload Mass by F9 v1.1

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

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

```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

	avg_payloadmass
0	2928.4

The average payload mass carried by booster version F9 v1.1 as 2928.4

First Successful Ground Landing Date

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

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

Hint:Use min function

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

	firstsuccessfull_landing_date
0	2015-12-22

The dates of the first successful landing outcome on ground pad was 2015-12-22.

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

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

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

List the total number of successful and failure mission outcomes

```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

	failureoutcome
0	1

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

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

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

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

```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

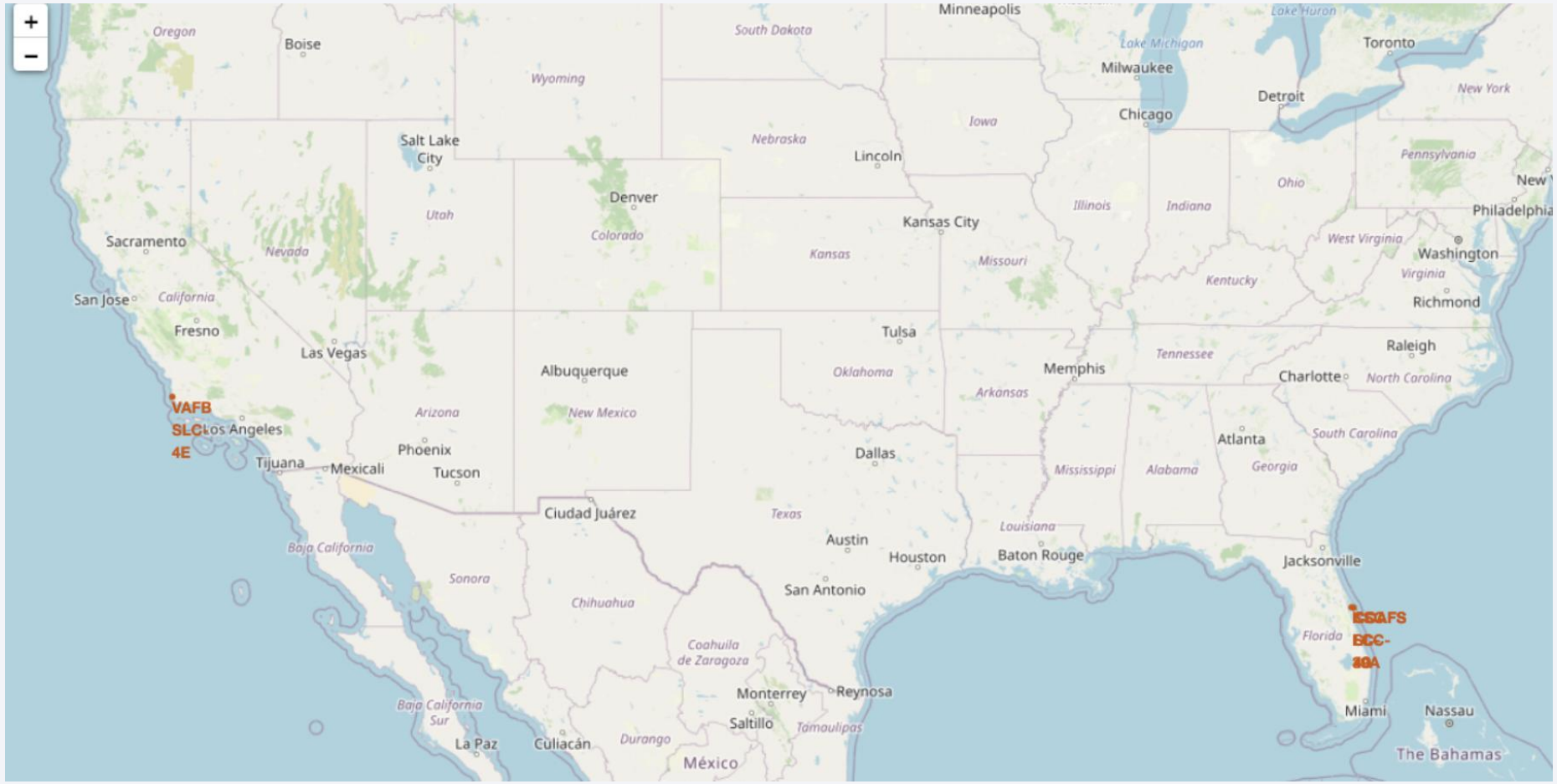
Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

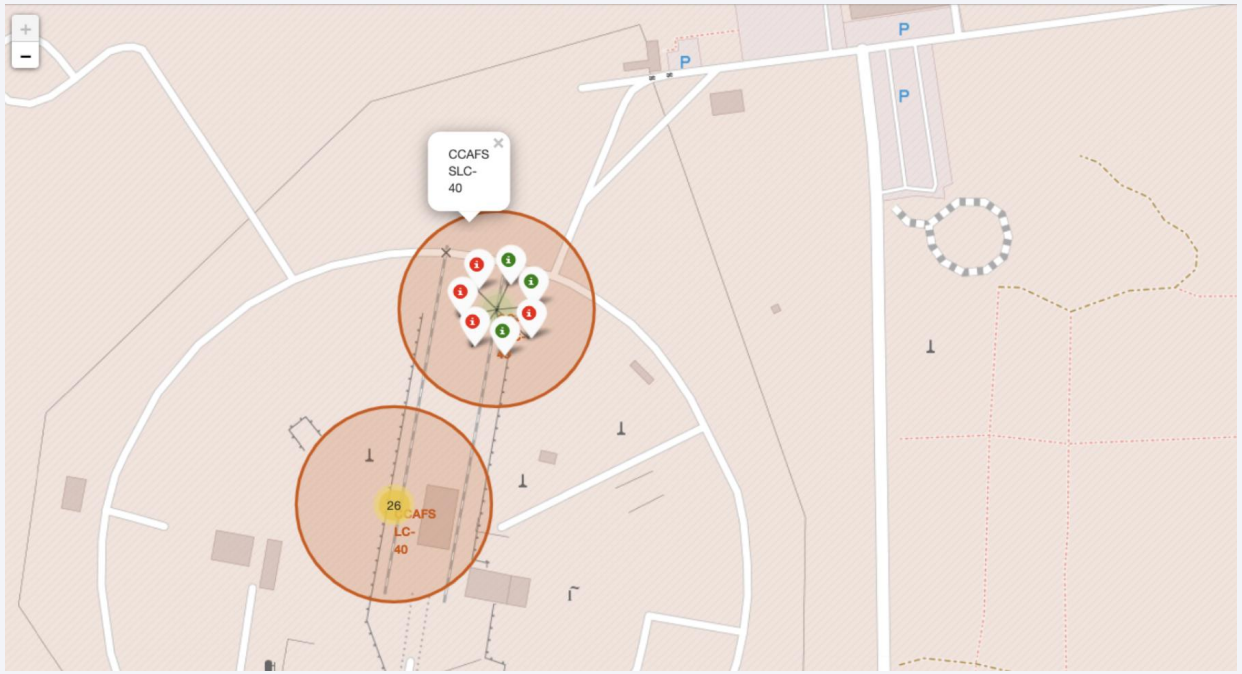

Section 3
Launch Sites
Proximities Analysis

<Folium Map Screenshot 1>



SpaceX has launch sites are in the United States of America coasts.
Florida and California

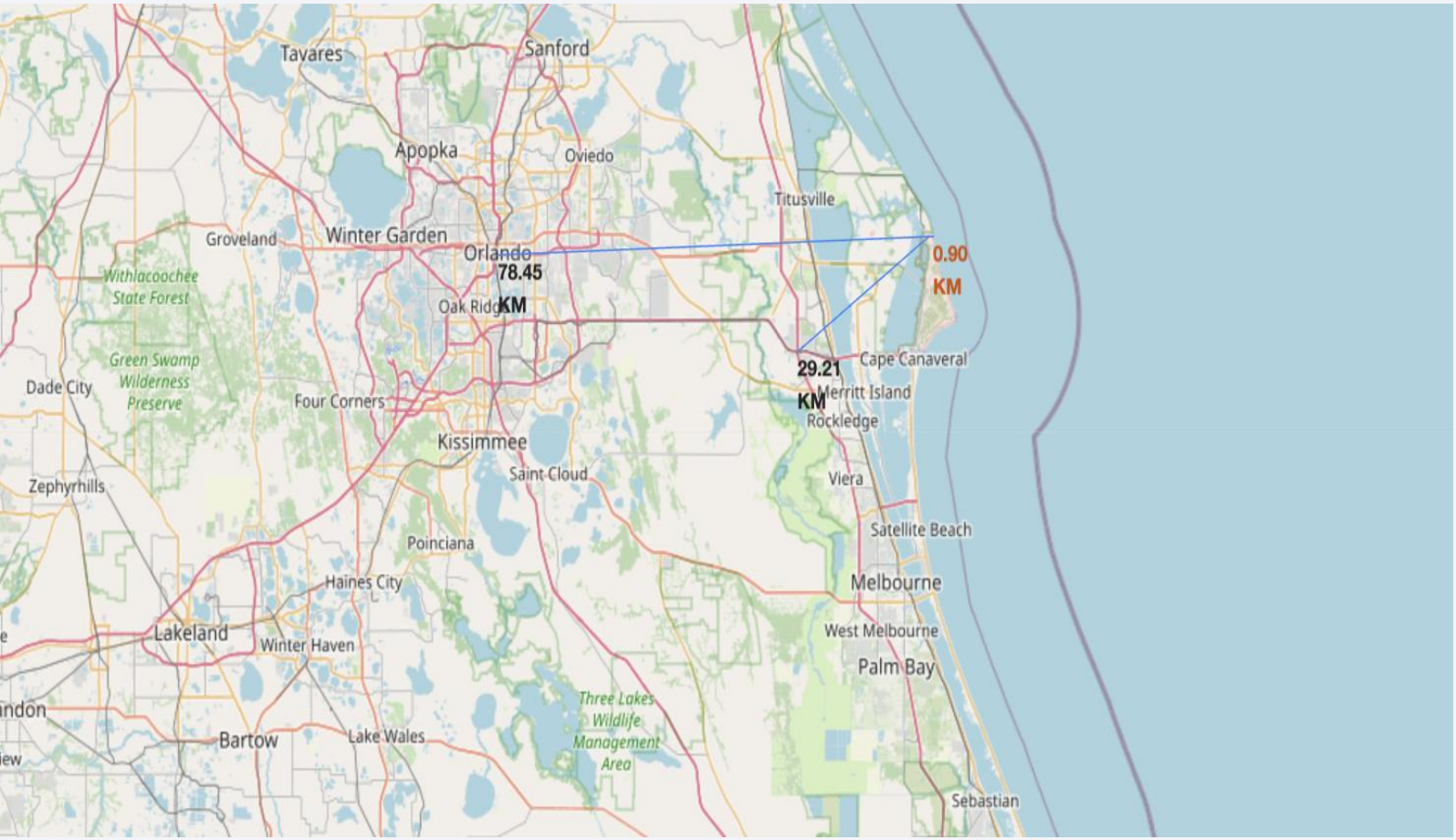
<Folium Map Screenshot 2>



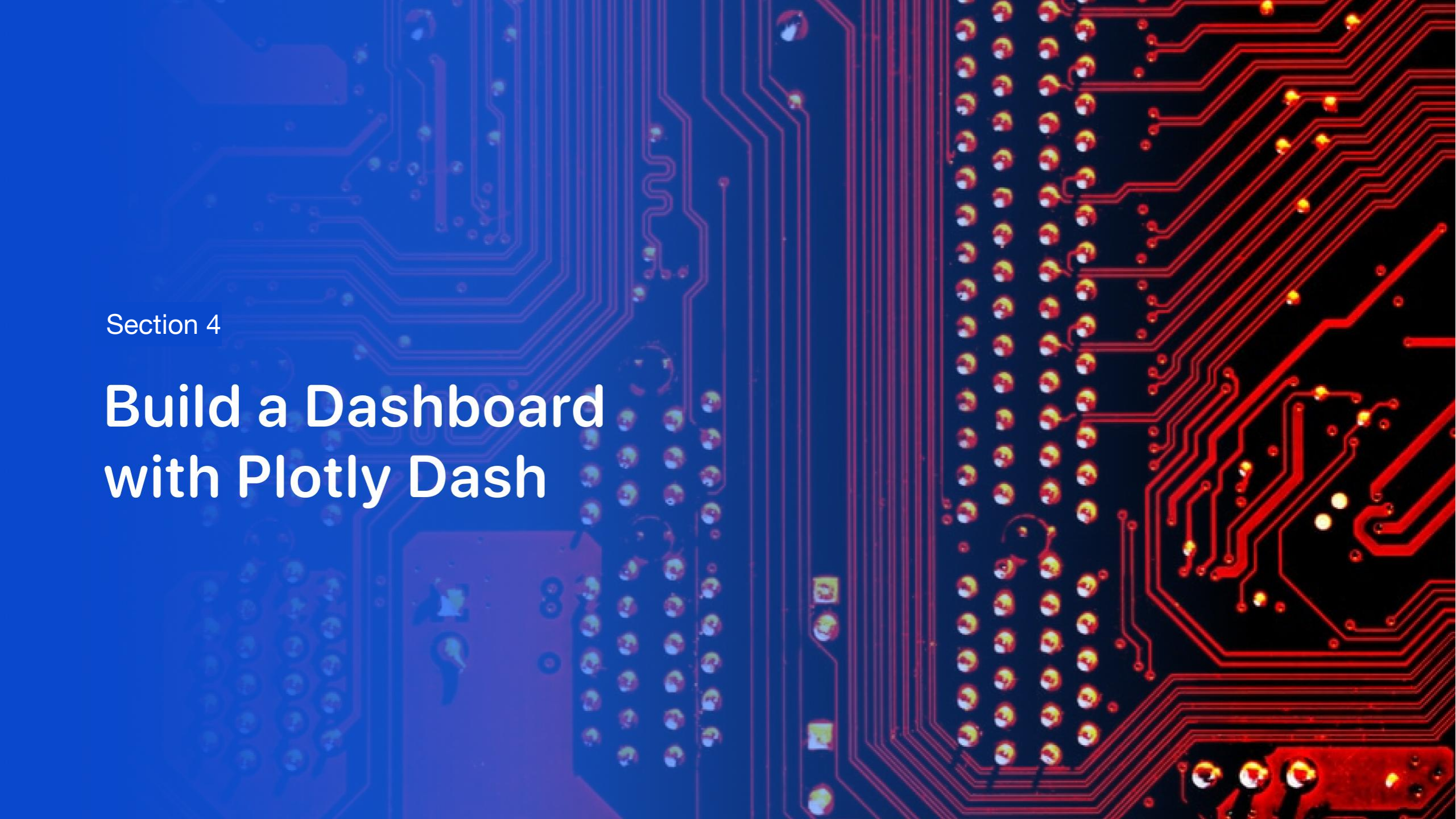
Florida

California

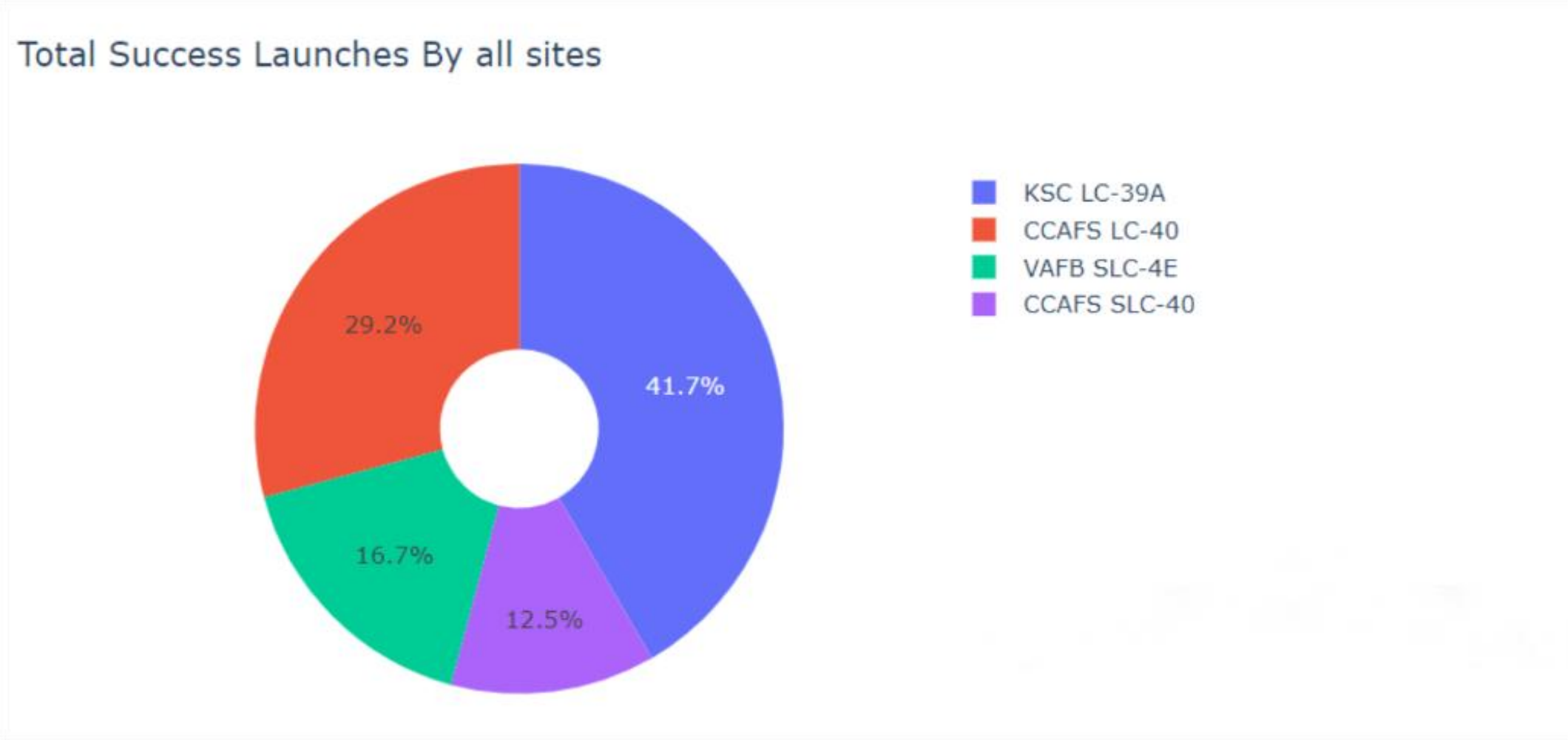
<Folium Map Screenshot 3>



* Are launch sites in close proximity to railways? NO
* Are launch sites in close proximity to highways? NO
* Are launch sites in close proximity to coastline? YES
* Do launch sites keep certain distance away from cities? YES

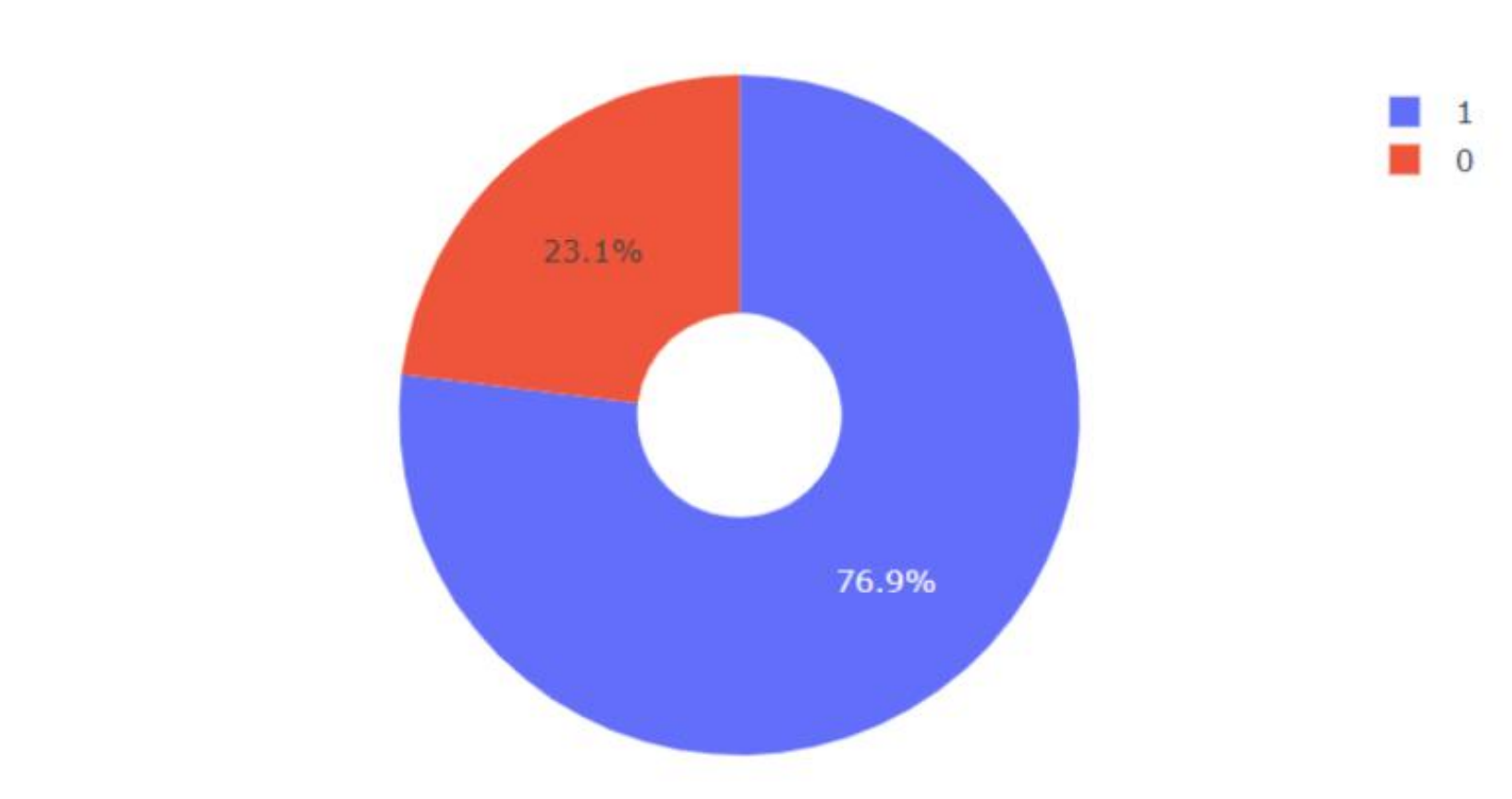

Section 4
Build a Dashboard with Plotly Dash

Pie chart : Success percentage for each launch site



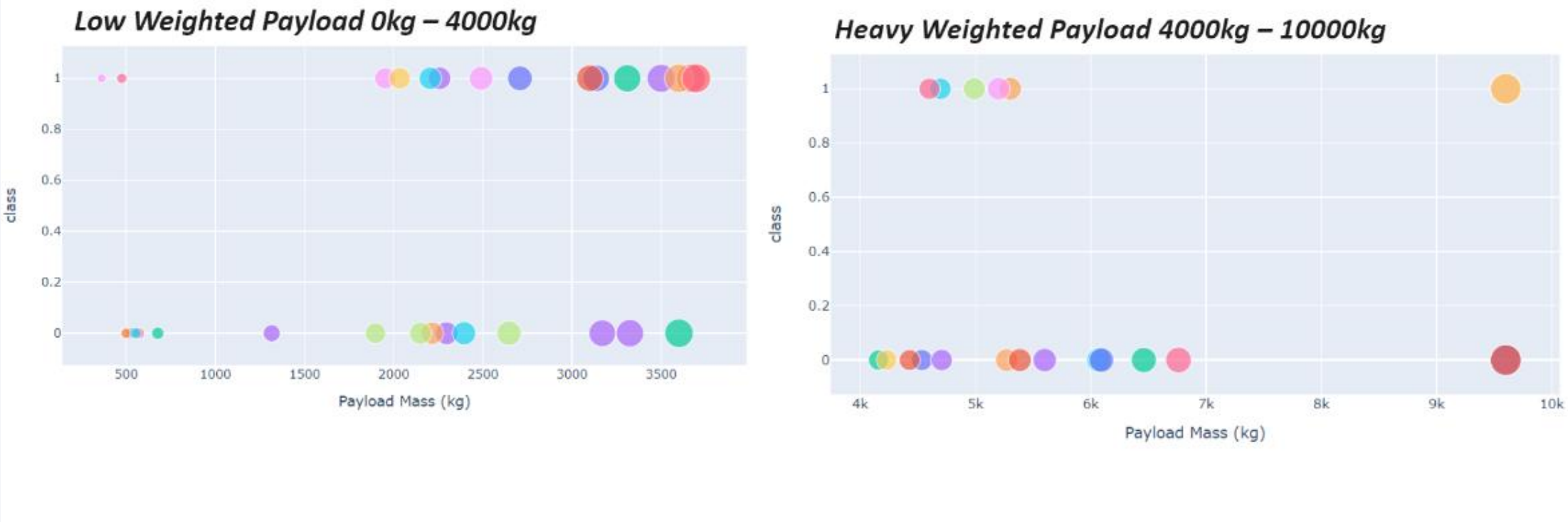
KSC LC-39A had the most successful launches among all the sites.

Pie chart: Launch success percentage for KSC LC-39A



KSC LC-39A has a 76.9% success rate with only 23.1% failure rate.

Scatter plot: Payload vs Launch Success



The success rates for low weighted payloads is higher than those with heavy weighted payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

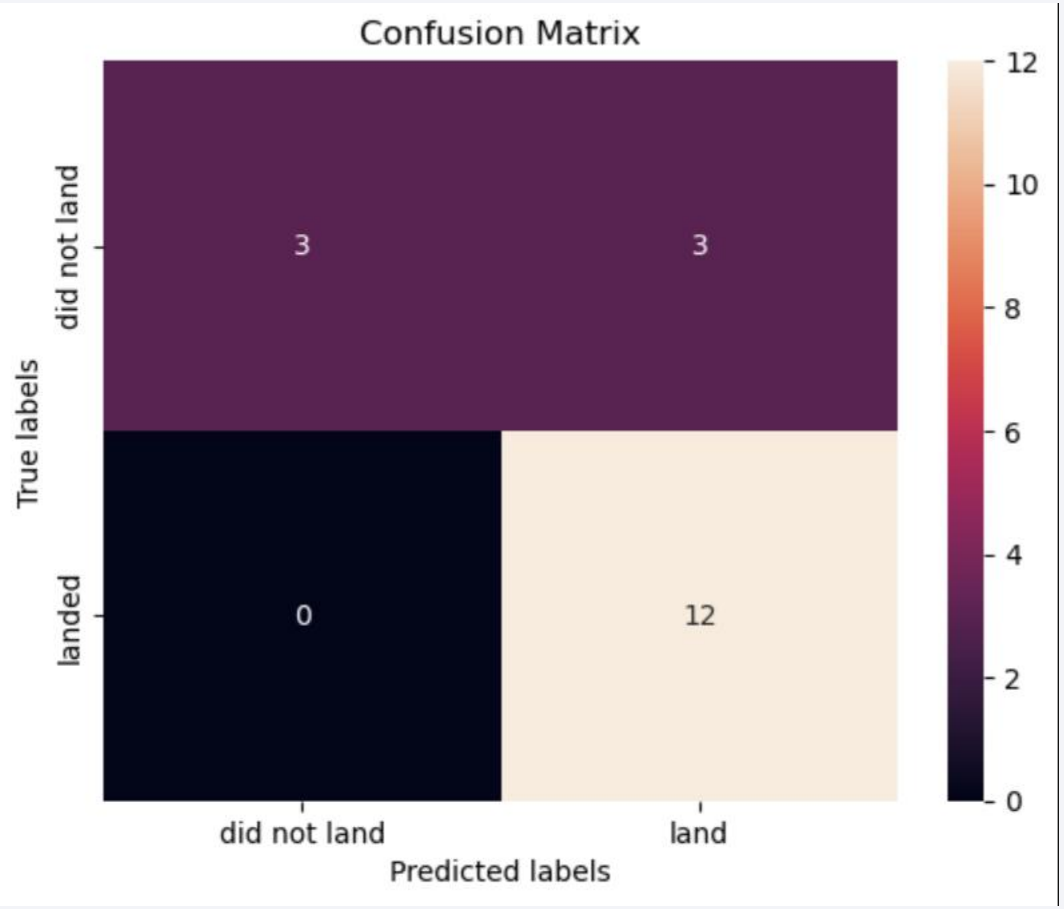
Find the method performs best:

```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split':
5, 'splitter': 'random'}
```

Confusion Matrix



- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.

Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.[Postive relationship]
- Launch success rate are increasing ever 2013 till 2020, so we look bright into the future of launch.
- Orbits ES–L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC–39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for predicting the launching success.

Thank you!