IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

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 advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

1. What factors determine if the rocket will land successfully?
2. The interaction amongst various features that determine the success rate of a successful landing.
3. What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical feature
- 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.
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- https://github.com/anhvw/proj-py-capstone

```
[6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
     response = requests.get(spacex_url)
     print(response.content)
```


```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6P
h45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":
{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.s
pace.com/2196-spacex-inaugural-falcon-1-rocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-
17T00:00:00.000Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":3
3,"altitude":null,"reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and loss of vehicle","crew":[],"ships":[],"capsules":
[],"payloads":["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flight_number":1,"name":"FalconSat","date_utc":"2006-03-24T22:30:0
0.000Z","date_unix":1143239400,"date_local":"2006-03-25T10:30:00+12:00","date_precision":"hour","upcoming":false,"cores":[{"core":"5e9e289df35918033d3
b2623","flight":1,"gridfins":false,"legs":false,"reused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"au
to_update":true,"tbd":false,"launch_library_id":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reused":false,"recovery_attempt":false,"recovere
d":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/f9/4a/ZboXReNb_o.png","large":"https://images2.imgbox.com/80/a2/bkWotCIS_o.
png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://ww
w.youtube.com/watch?v=Lk4zQ2wP-Nc","youtube_id":"Lk4zQ2wP-Nc","article":"https://www.space.com/3590-spacex-falcon-1-rocket-fails-reach-orbit.html","wi
kipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":null,"static_fire_date_unix":null,"net":false,"window":0,"rocket":"5e9d0d95ed
a69955f709d1eb","success":false,"failures":[{"time":301,"altitude":289,"reason":"harmonic oscillation leading to premature engine shutdown"}],"detail
s":"Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach or
bit, Failed to recover first stage","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b6b6c3bb0006eeb1e2"],"launchpad":"5e9e4502f5090995de566f8
6","flight_number":2,"name":"DemoSat","date_utc":"2007-03-21T01:10:00.000Z","date_unix":1174439400,"date_local":"2007-03-21T13:10:00+12:00","date_prec
```


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

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

```
[7]: response.status_code
```

```
[7]: 200
```

```
[9]: # Use json_normalize method to convert the json result into a dataframe
     response = requests.get(static_json_url).json()
     data = pd.json_normalize(response)
```

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

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

[46]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
      soup = BeautifulSoup(data, 'html5lib')

[47]: # 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

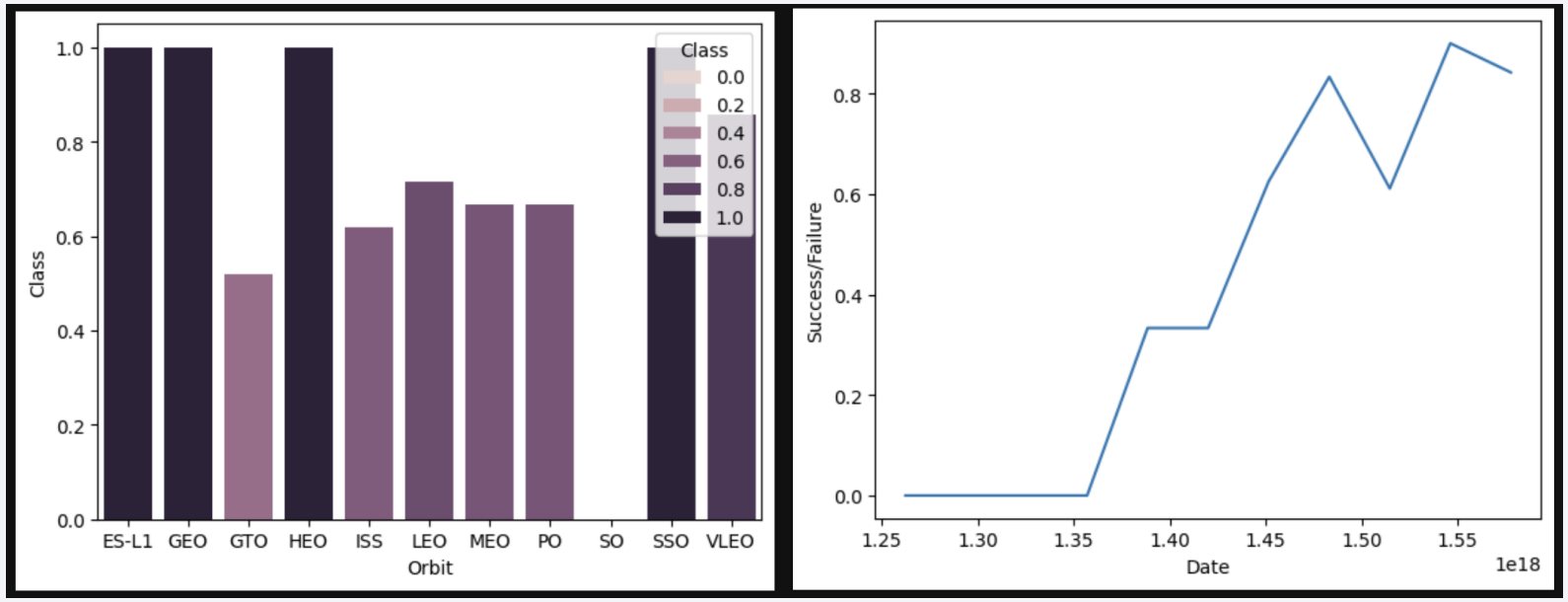
```
[48]: # Use the find_all function in the BeautifulSoup object, with element type `table`
      # Assign the result to a list called `html_tables`
      html_tables = soup.find_all('table')

[49]: # Let's print the third table and check its content
      first_launch_table = html_tables[2]
      print(first_launch_table)

[50]: 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
      for row in first_launch_table.find_all('th'):
          name = extract_column_from_header(row)
          if (name != None and len(name) > 0):
              column_names.append(name)

[51]: print(column_names)

      ['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

Data Wrangling

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.

EDA with Data Visualization

- We explored 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



EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

Build an Interactive Map with Folium

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote 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.

Build a Dashboard with Plotly Dash

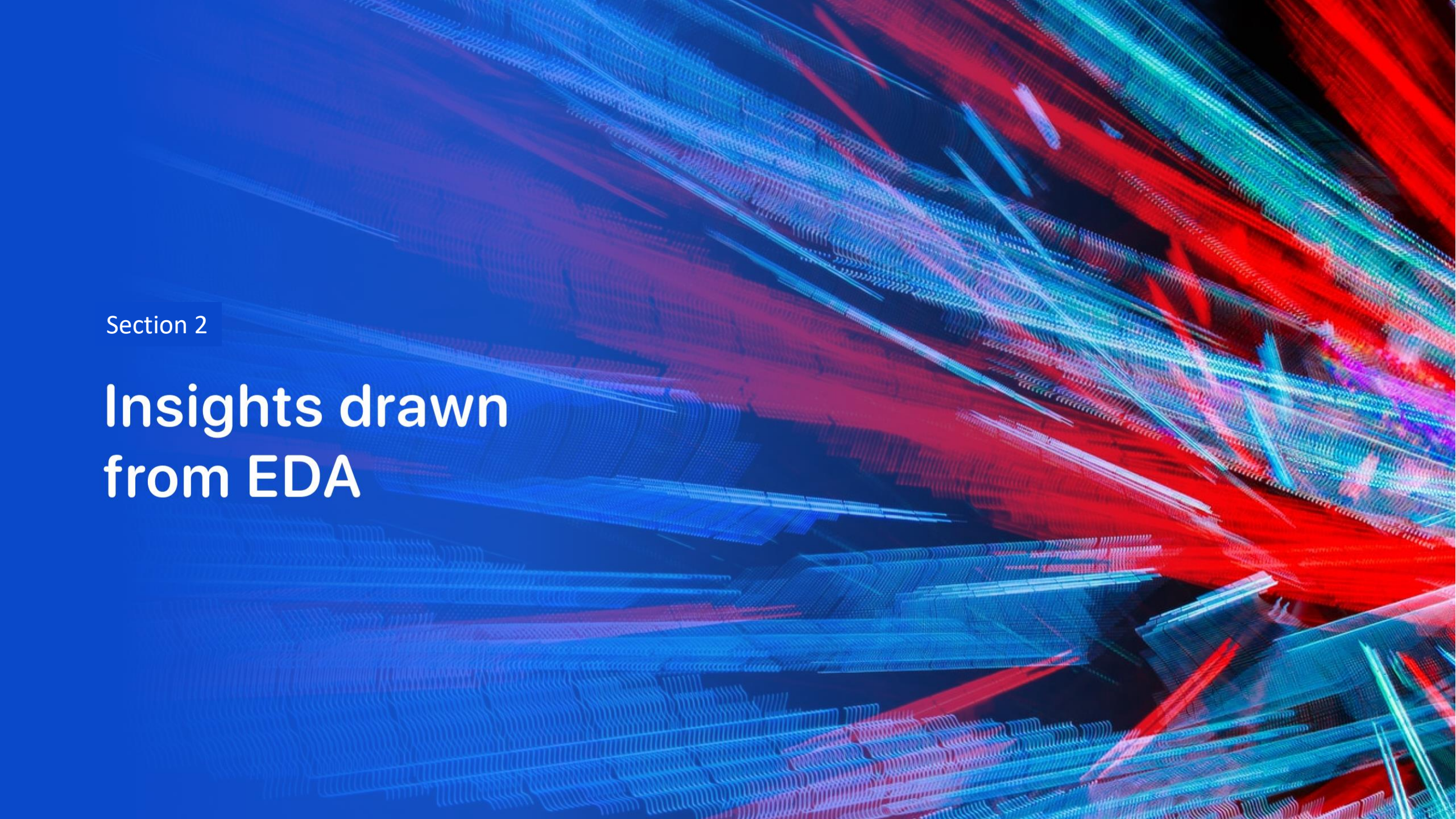
- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.

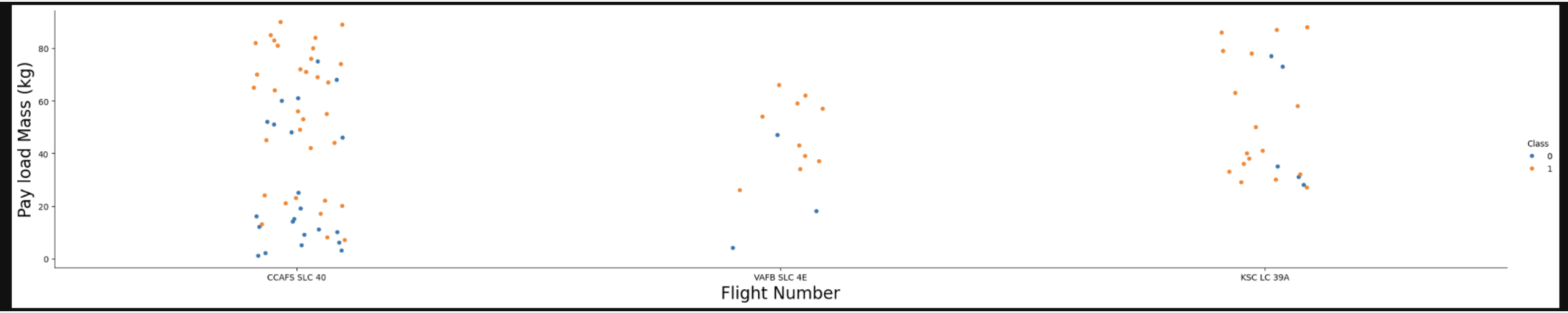
Results

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


Section 2
Insights drawn from EDA

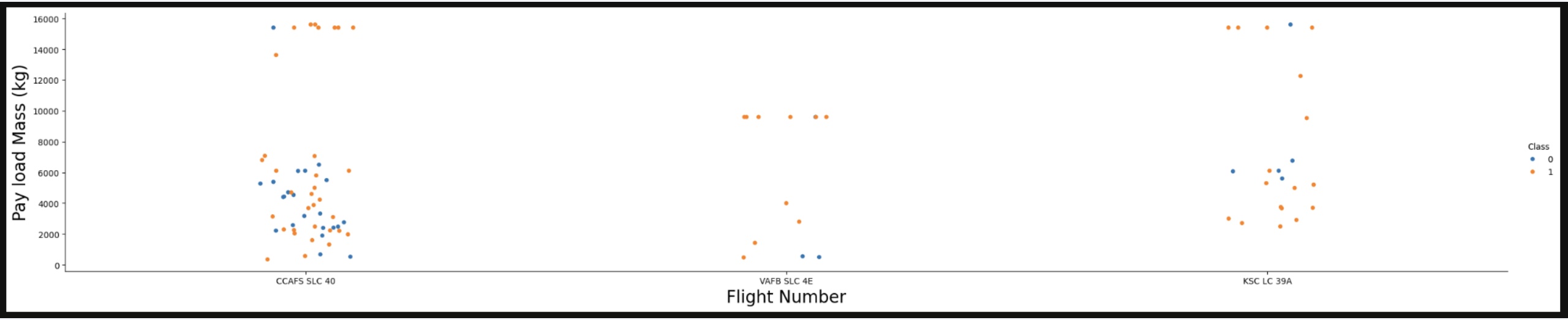
Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



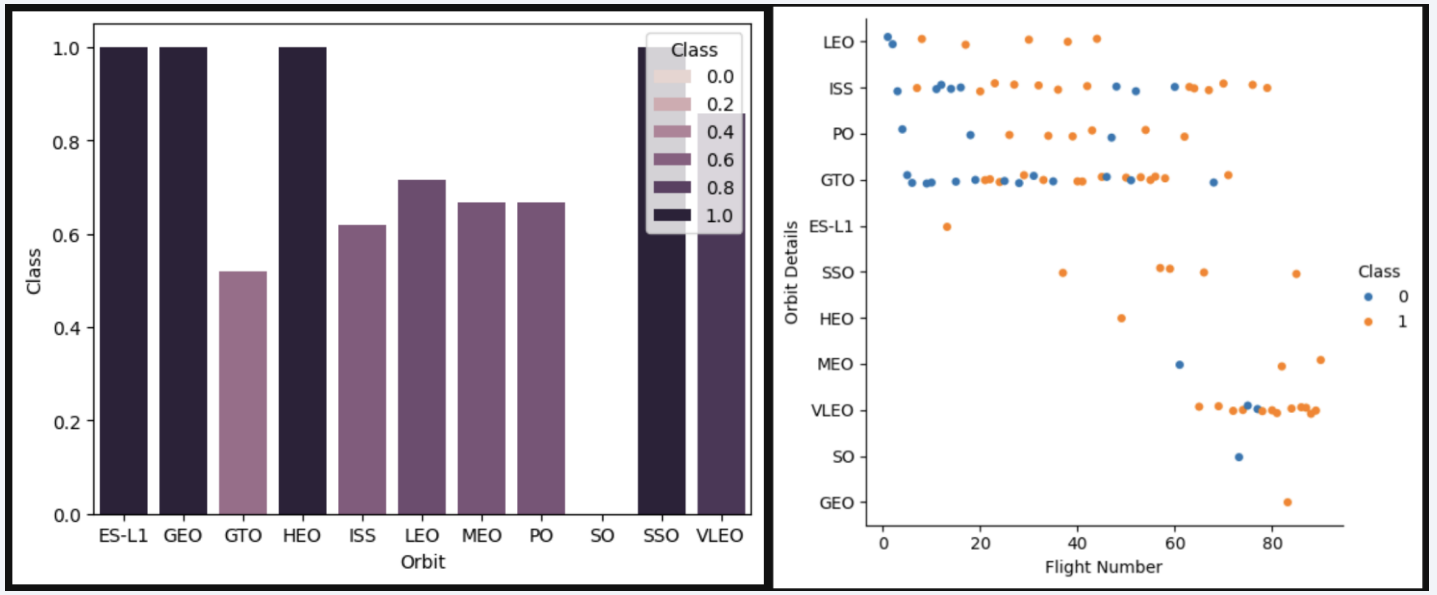
Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site



Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type
- Show the screenshot of the scatter plot with explanations



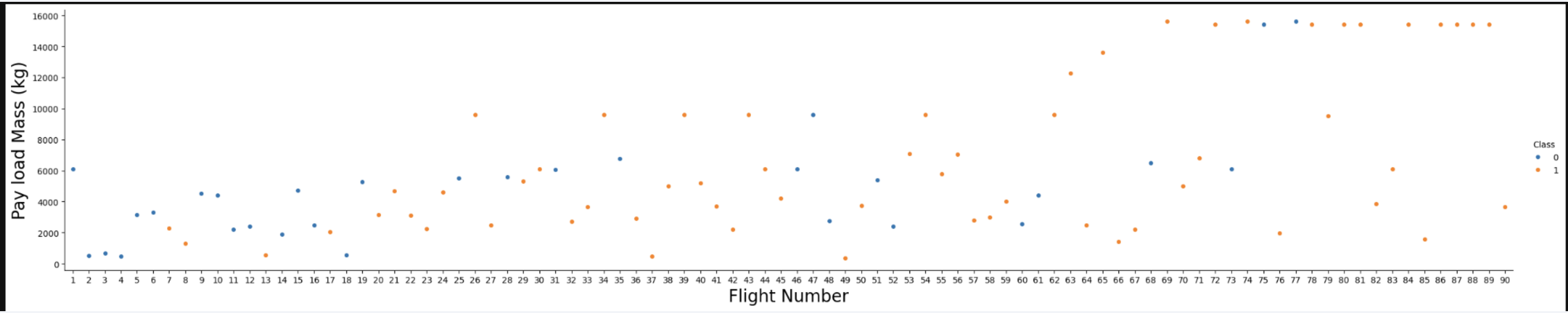
Flight Number vs. Orbit Type

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

- Show the screenshot of the scatter plot with explanations

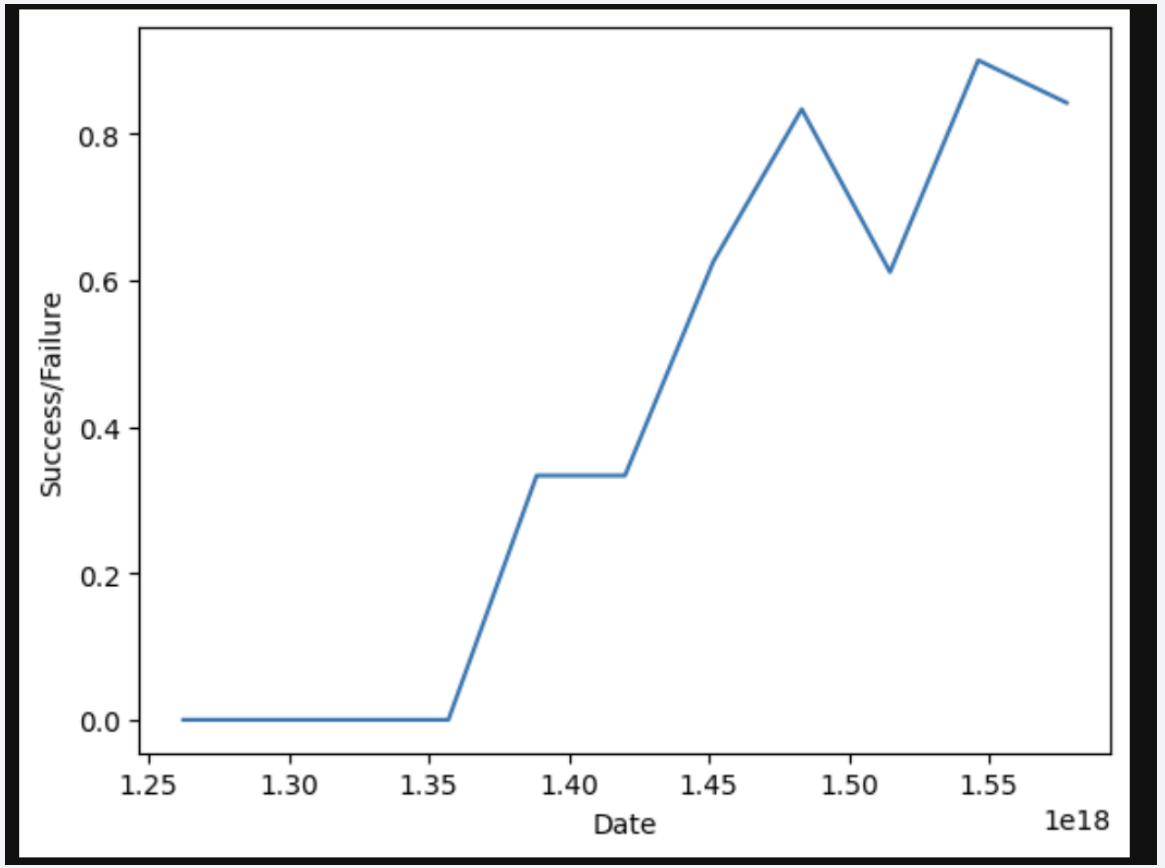
Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type



Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations



All Launch Site Names

- Find the names of the unique launch sites
- Present your query result with a short explanation here

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Present your query result with a short explanation here

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

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

```
[23]: #Task 3 Display the total payload mass carried by boosters launched by NASA (CRS)
      %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL;

       * sqlite:///my_data1.db
      Done.
[23]: sum(PAYLOAD_MASS__KG_)
      619967
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

```
[24]: #Task 4 Display average payload mass carried by booster version F9 v1.1
      %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL;

       * sqlite:///my_data1.db
      Done.
[24]: avg(PAYLOAD_MASS__KG_)
      ---------------------
            6138.287128712871
```

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here

```
[22]: #Task 5 List the date when the first succesful landing outcome in ground pad was acheived.
      %sql select min(DATE) from SPACEXTBL;
```

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

[22]:

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

```
[27]: #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 BOOSTER_VERSION from SPACEXTBL where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
       * sqlite:///my_data1.db
      Done.
[27]: Booster_Version
      F9 FT B1022
      F9 FT B1026
      F9 FT B1021.2
      F9 FT B1031.2
```

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

```
[31]: #Task 7 List the total number of successful and failure mission outcomes
      %sql select Mission_Outcome, count(Mission_Outcome) from SPACEXTBL GROUP BY Mission_Outcome;
```

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

[31]:

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here

```
[32]: #Task 8 List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
      %sql select BOOSTER_VERSION from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
       * sqlite:///my_data1.db
      Done.
[32]: 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

```
[37]: %sql SELECT substr(Date,6,2),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where substr(Date,0,5)='2015' and Mission_Outcome!='Success';
 * sqlite:///my_data1.db
Done.
```

[37]:

substr(Date,6,2)	Mission_Outcome	Booster_Version	Launch_Site
06	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40

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

```
[43]: %sql SELECT Landing_Outcome, count(Landing_Outcome) FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' group by Landing_Outcome ORDER BY cou
 * sqlite:///my_data1.db
Done.
```

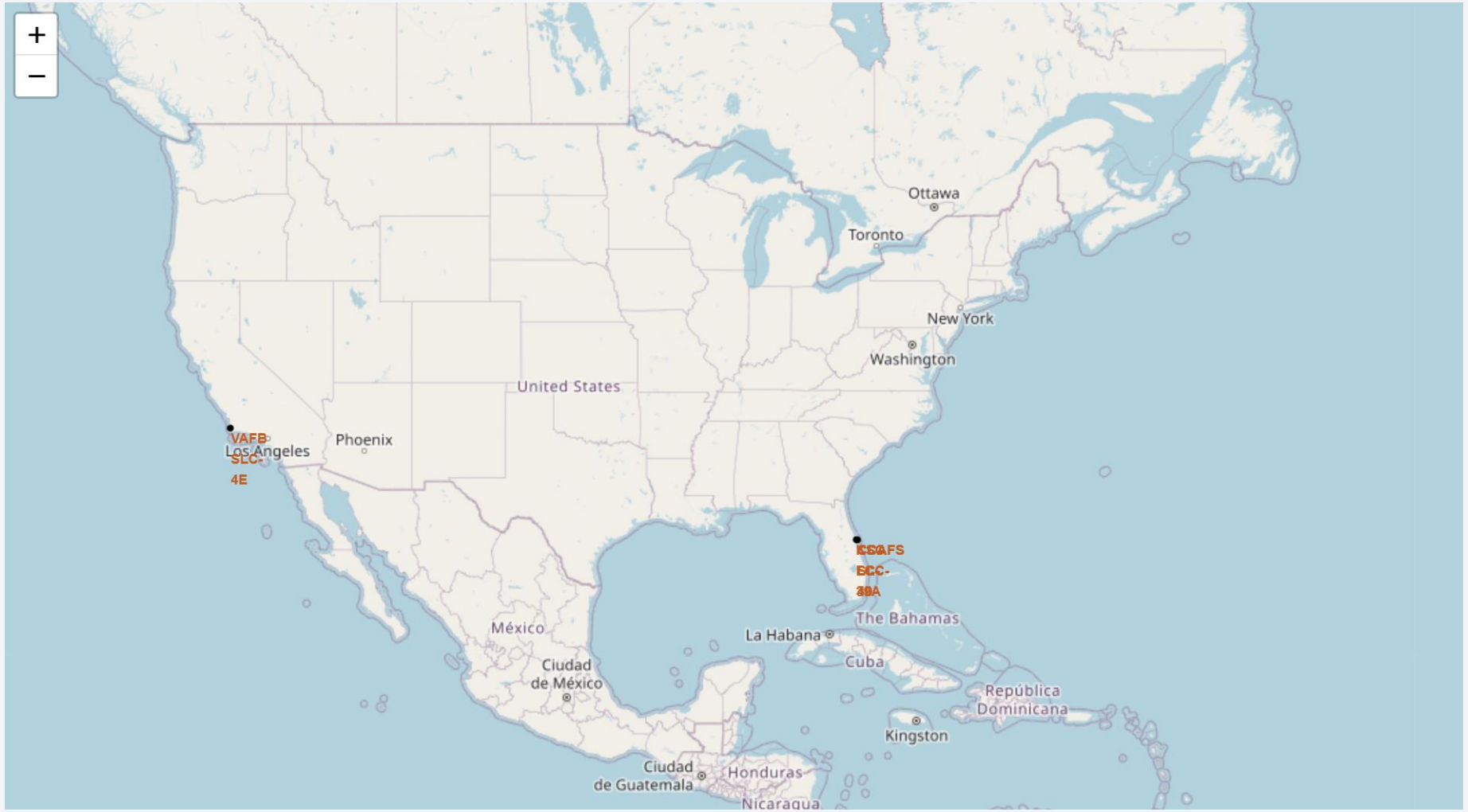
[43]:

Landing_Outcome	count(Landing_Outcome)
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

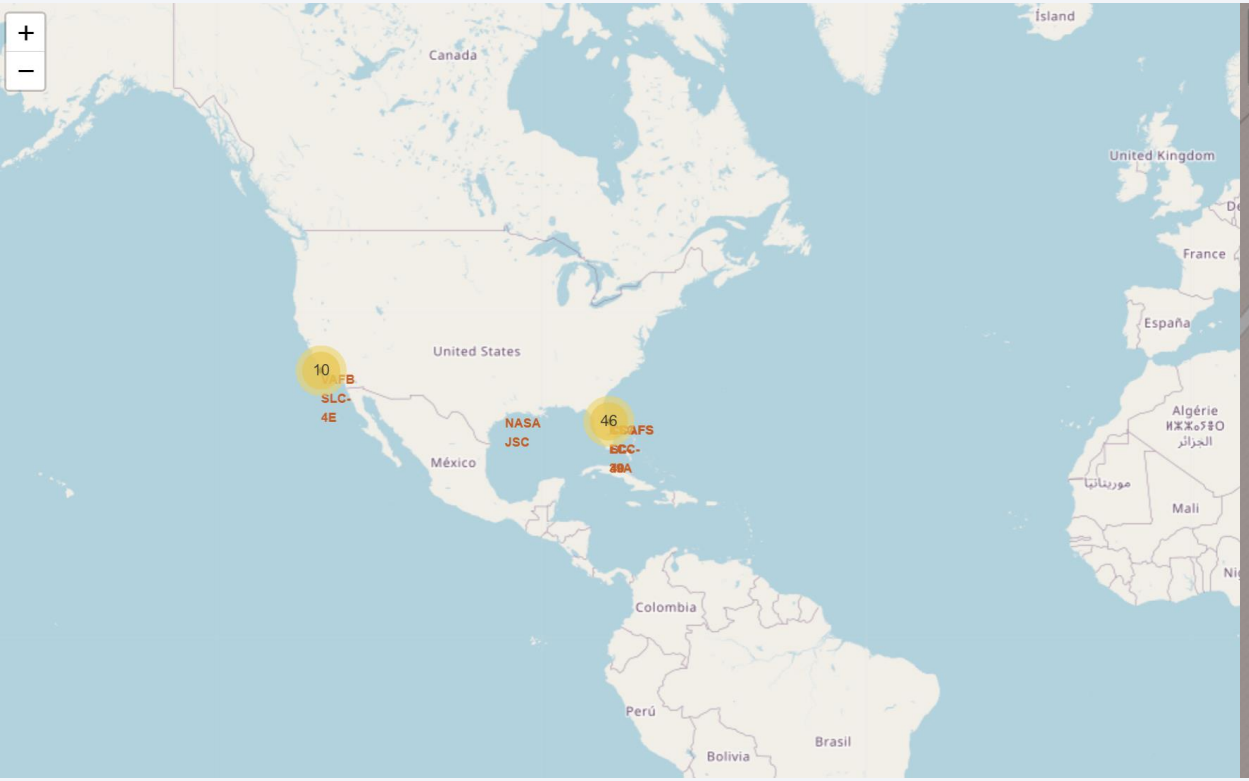
Section 3

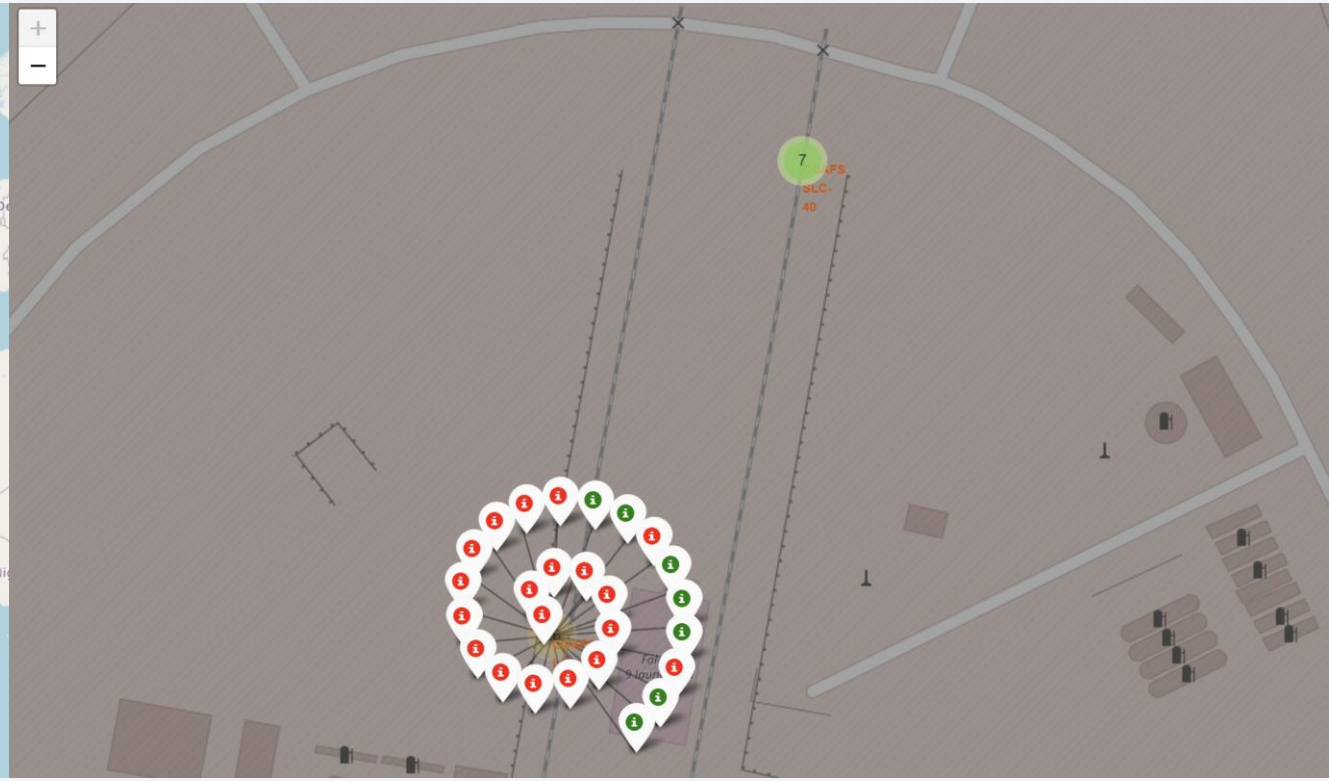
Launch Sites Proximities Analysis

All launch sites global map markers

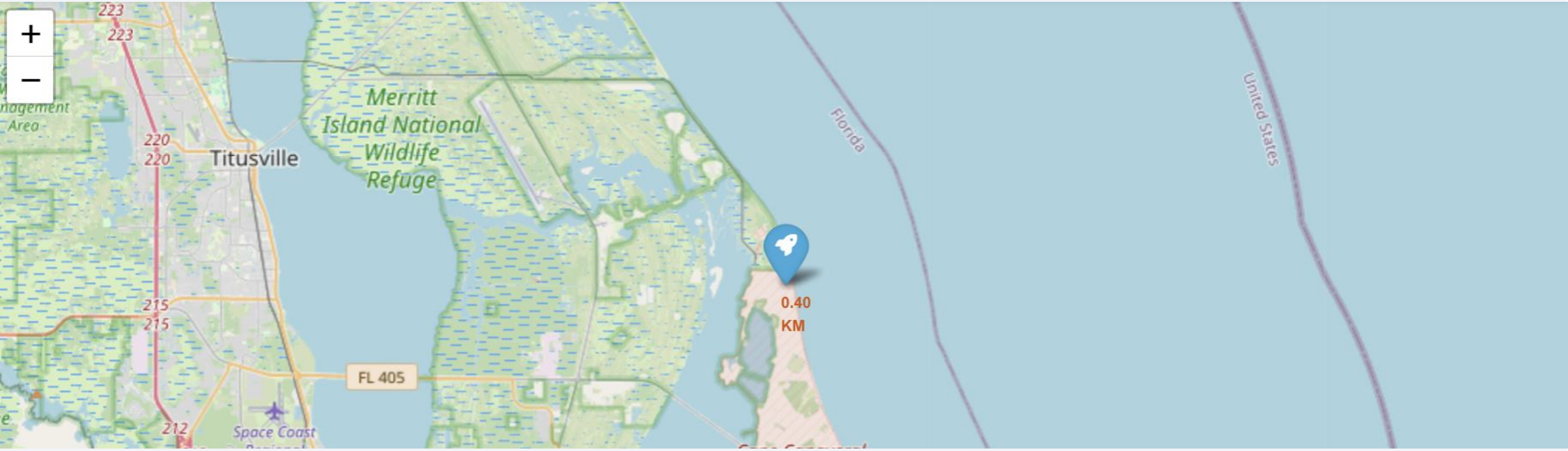


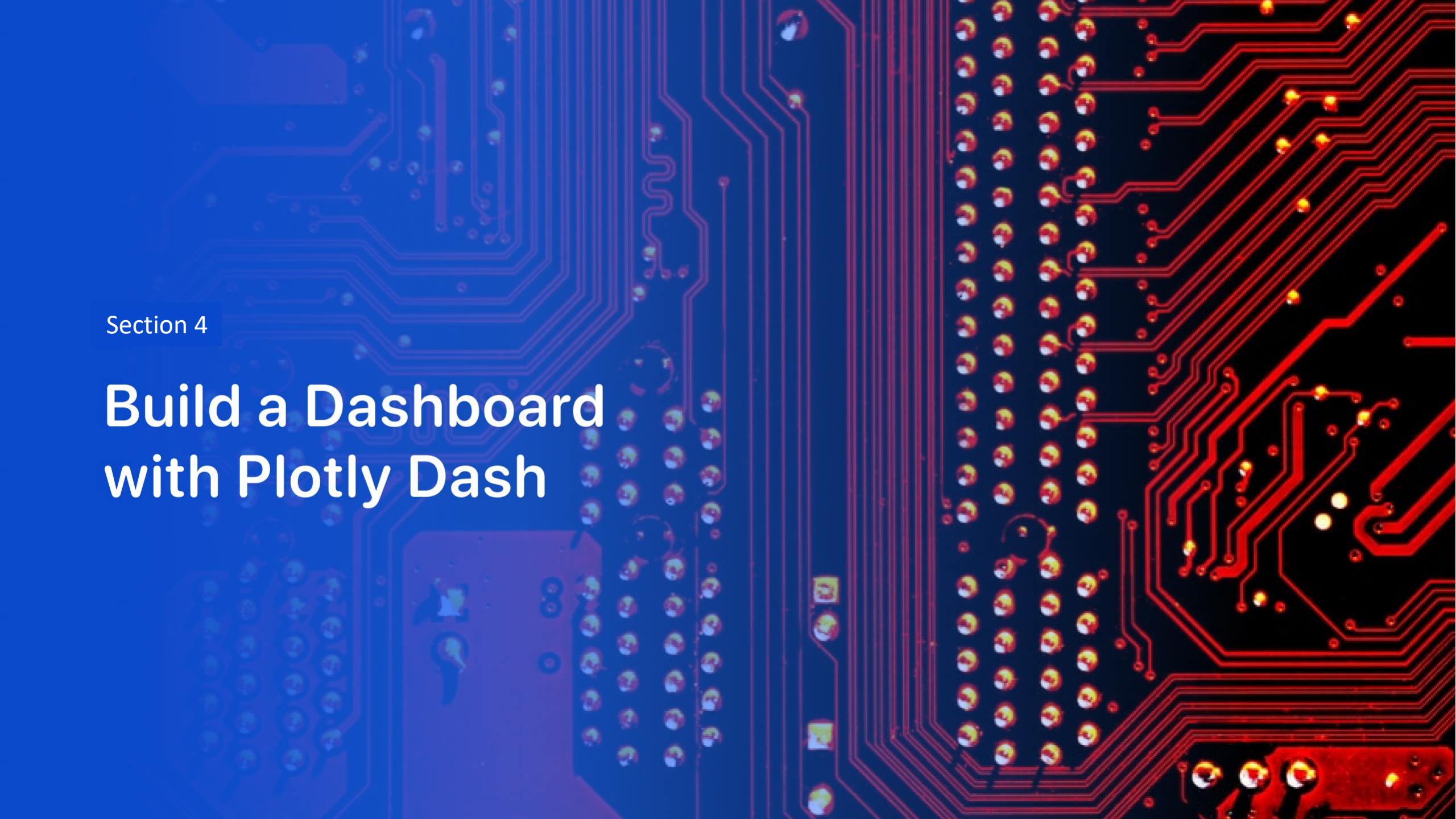
Markers showing launch sites with color labels





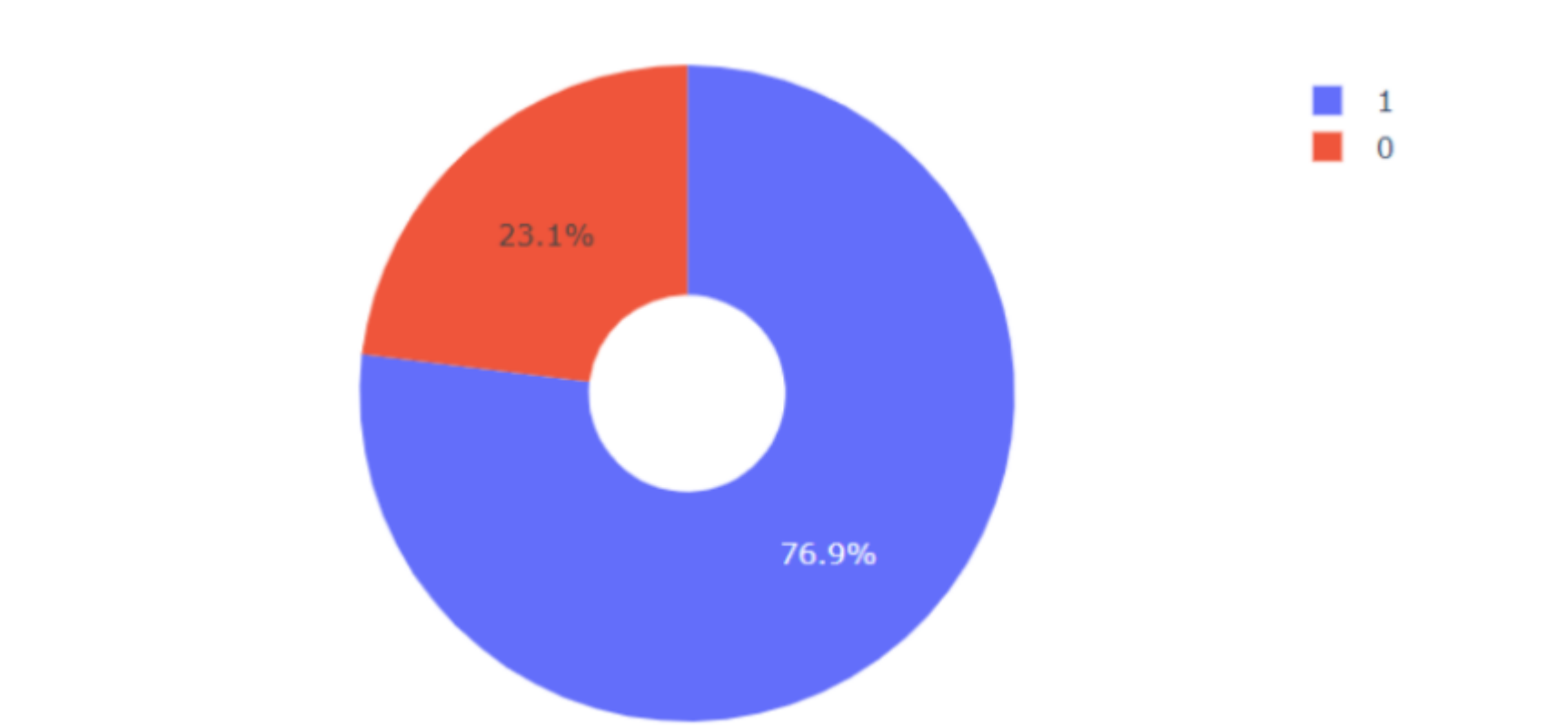
Launch Site distance to coastline




Section 4
Build a Dashboard with Plotly Dash

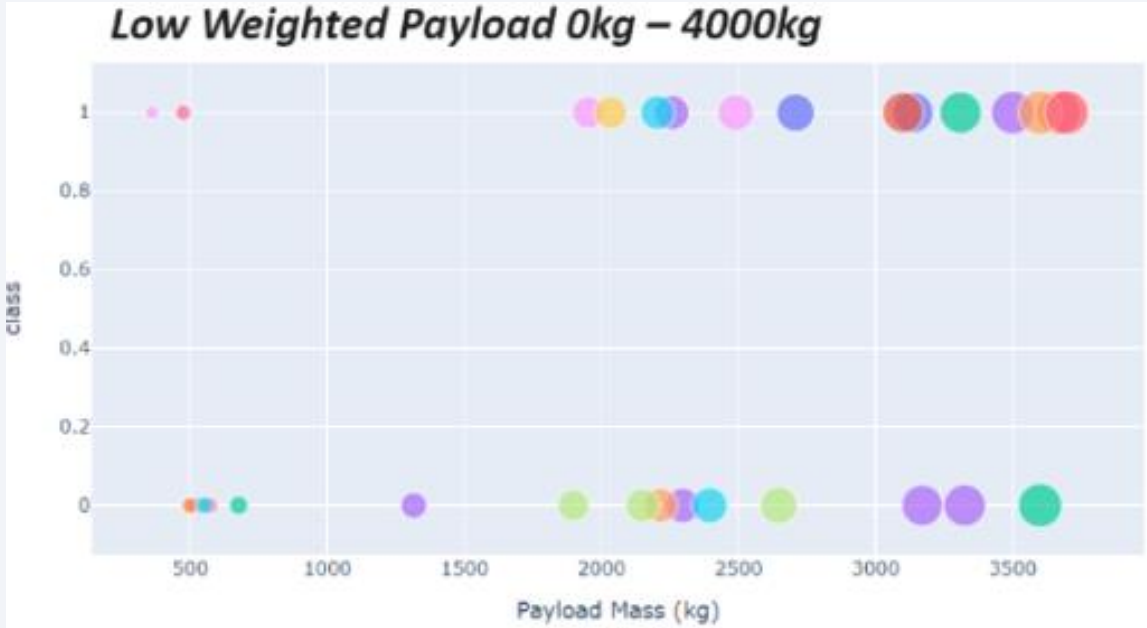
Pie chart showing the success percentage achieved by each launch site

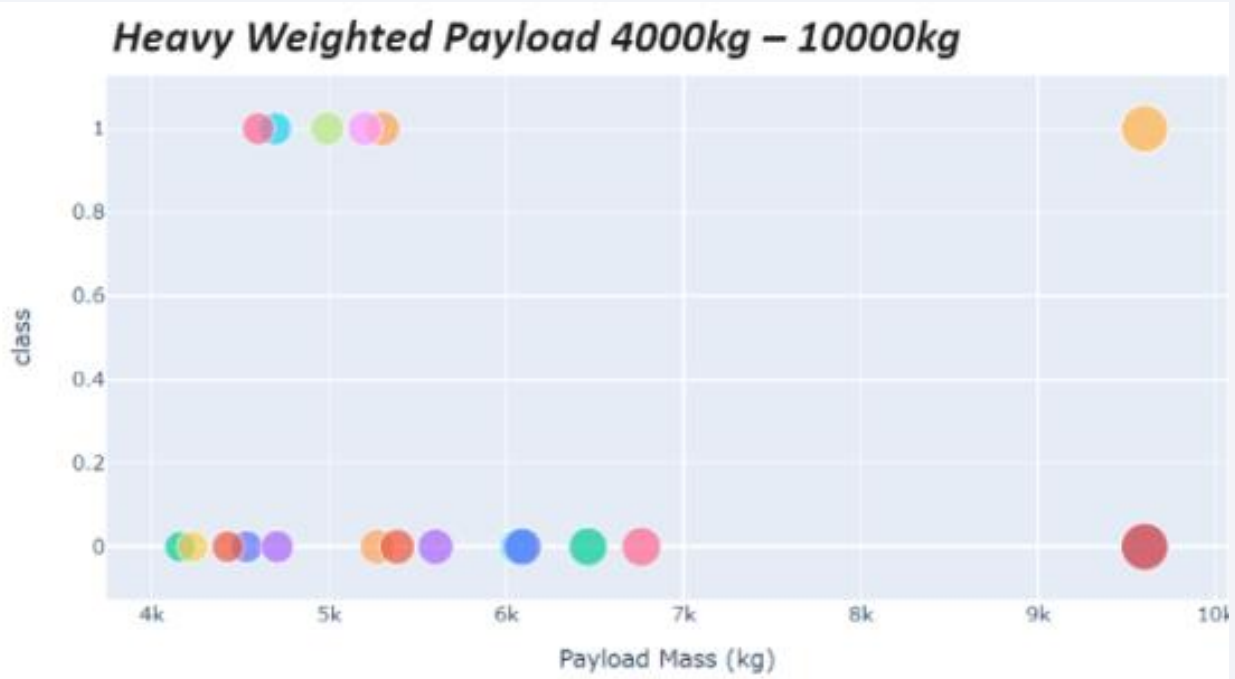
Pie chart showing the Launch site with the highest launch success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

TASK 8

Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.

[52]:
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

[53]:
```
tree_cv = GridSearchCV(tree,parameters,cv=10)
tree_cv.fit(X_train, Y_train)
```

```
 0.78928571 0.76607143 0.69464286 0.81785714 0.83392857 0.77678571
 0.69285714 0.66607143 0.75357143 0.69464286 0.80357143 0.79107143
 0.80178571 0.77857143 0.70535714 0.84821429 0.75714286 0.80535714
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
        nan        nan        nan        nan        nan        nan
 0.73571429 0.79285714 0.80535714 0.76607143 0.77678571 0.81785714
 0.74642857 0.75       0.75178571 0.83214286 0.76607143 0.77678571
 0.77678571 0.775      0.77678571 0.71964286 0.69464286 0.75178571]
  warnings.warn(
```

[53]:

GridSearchCV

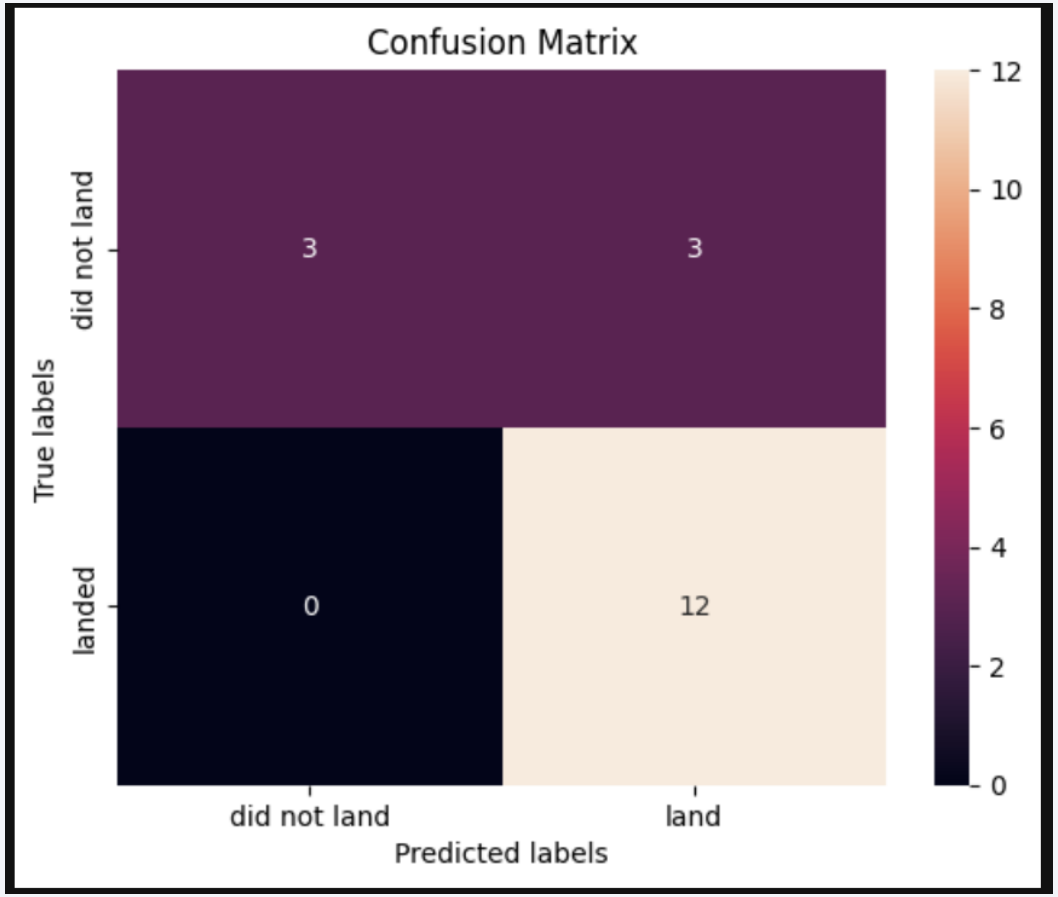
best_estimator_: DecisionTreeClassifier

DecisionTreeClassifier

[54]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 2,
'splitter': 'random'}
accuracy : 0.9035714285714287
```

Confusion Matrix



Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- 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 this task.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!