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
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - Exploratory data analysis results
  - Interactive data analysis
  - Predictive analysis results

Introduction

- Project background and context

  In this analysis we will focus on the study of the first stage of the SpaceX Falcon 9 Rocket in order to obtain conclusions that allow us to make cost projections as well as obtain insight on the implications in the area beyond the economic ones.

  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
  - What factors determine if the rocket will land successfully?
  - Predicting if the Falcon 9 first stage will land successfully
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

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

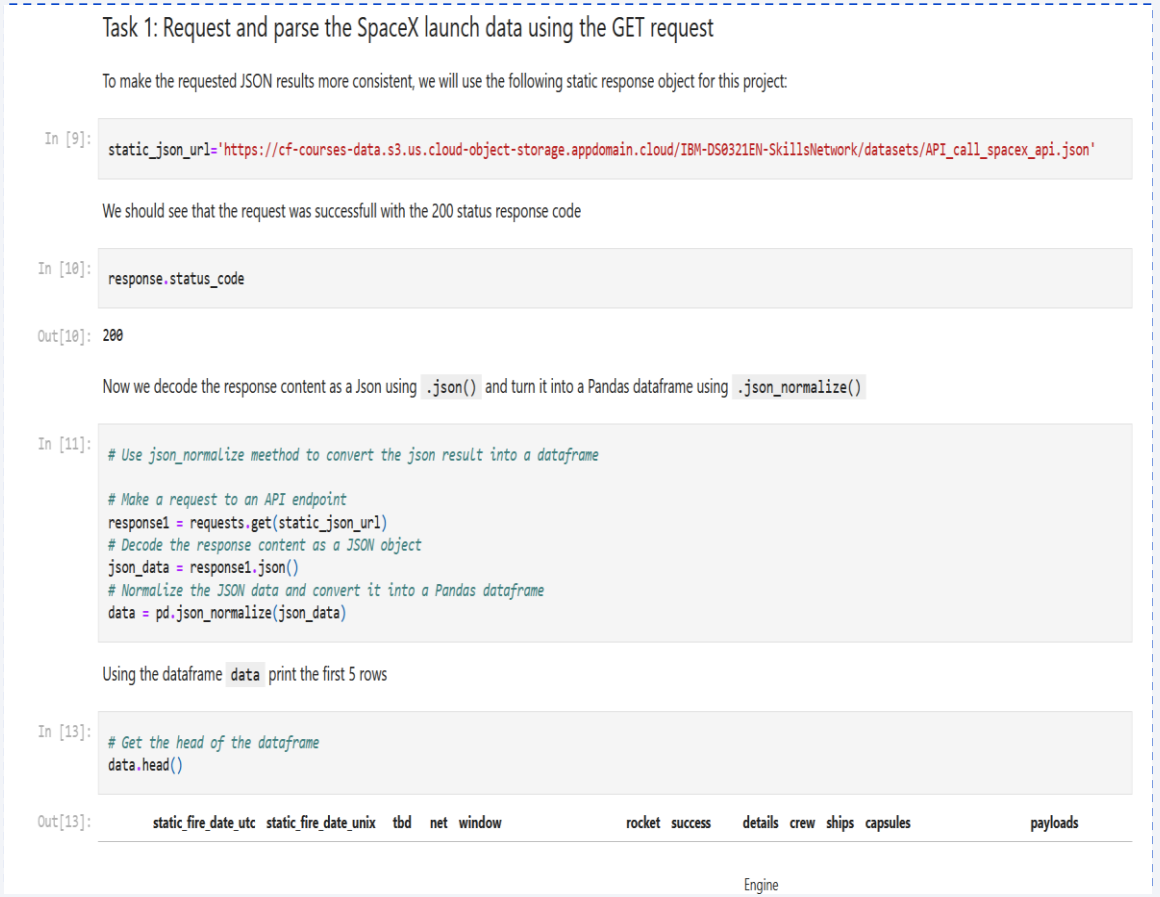
Data Collection

- Describe how data sets were collected:
  - Data collection was done using get request to the SpaceX API.
  - Use of static response object to make the JSON result more consistent
  - Convert the JSON result into a pandas dataframe using .json_normalize()
  - Filter the dataframe to keep only falcon9 launches
  - Check for the missing values and filled them up with its mean 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

- The get requests was used to collect data from the SpaceX API. Data was cleaned and formatted including son data wrangling.

- The link to the notebook is https://github.com/RodrigueFomena/Applied_Data_Science_Capstone/blob/main/data-collection-api-jupyter-labs-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:

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

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 [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe

# Make a request to an API endpoint
response1 = requests.get(static_json_url)
# Decode the response content as a JSON object
json_data = response1.json()
# Normalize the JSON data and convert it into a Pandas dataframe
data = pd.json_normalize(json_data)
```

Using the dataframe `data` print the first 5 rows

In [13]:
```
# Get the head of the dataframe
data.head()
```

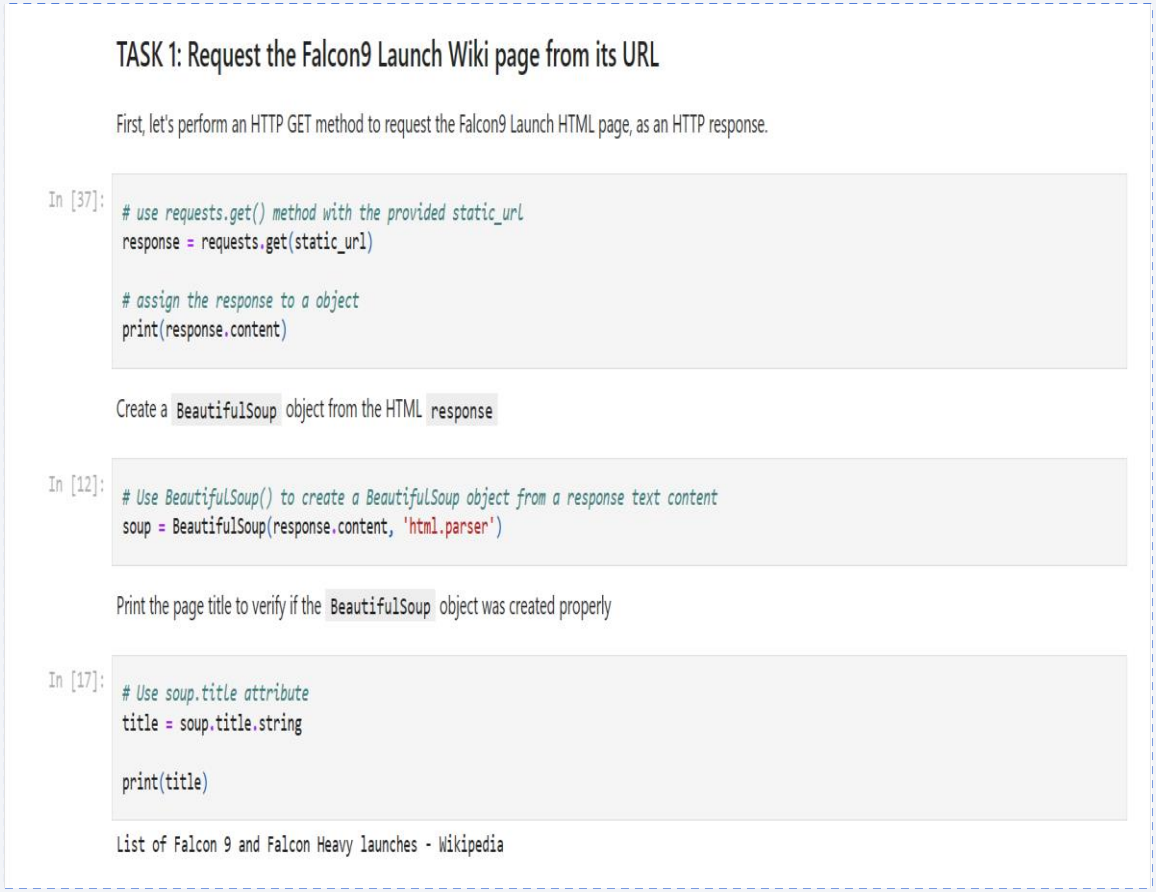
Out[13]:

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	payloads

Engine

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 [Applied_Data_Science_Capstone/data-collection-webscraping-jupyter-labs-spacex.ipynb at main : RodrigueFomena/Applied_Data_Science_Capstone (github.com)]



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 [37]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)

# assign the response to a object
print(response.content)
```

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

In [12]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, 'html.parser')
```

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

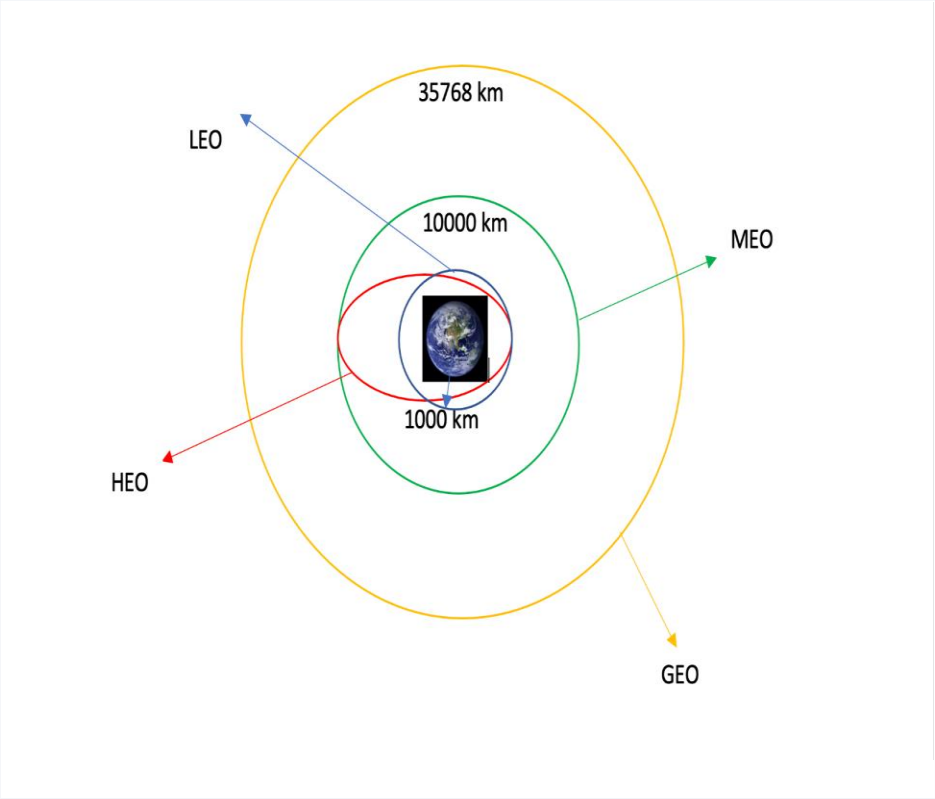
In [17]:
```
# Use soup.title attribute
title = soup.title.string

print(title)
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

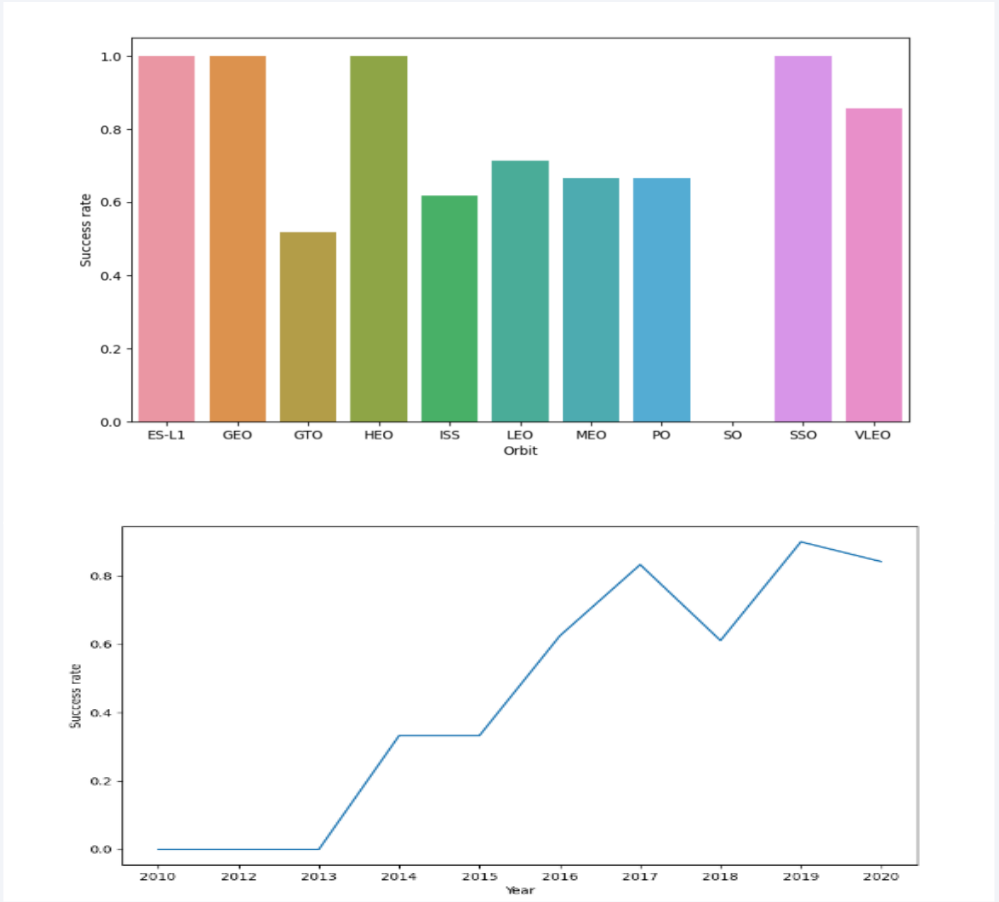
Data Wrangling

- We performed some Exploratory Data Analysis (EDA) to find some patterns in the data
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created a landing outcome label from the Outcome column and exported the results to csv
- We also determined the label for training supervised models
- The link to the notebook is Applied_Data_Science_Capstone/EDA-data-wrangling-jupyter-labs-spacex.ipynb at main · RodrigueFomena/Applied_Data_Science_Capstone (github.com)



EDA with Data Visualization

- We observed the relationship between several factors
- 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.
- The github link to the notebook is: [Applied_Data_Science_Capstone/EDA-dataviz.-jupyter-labs-spacex.ipynb at main · RodrigueFomena/Applied_Data_Science_Capstone (github.com)]



EDA with SQL

- 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.
- The github link to the notebook is [Applied_Data_Science_Capstone/EDA-with-sql-sqllite_db-jupyter-labs-spacex.ipynb at main · RodrigueFomena/Applied_Data_Science_Capstone (github.com)]

Build an Interactive Map with Folium

- 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.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- The github link to the notebook is: Applied_Data_Science_Capstone/Interactive-Visual-Analytics-with-Folium-labs-spacex.ipynb at main · RodrigueFomena/Applied_Data_Science_Capstone (github.com)

Build a Dashboard with Plotly Dash

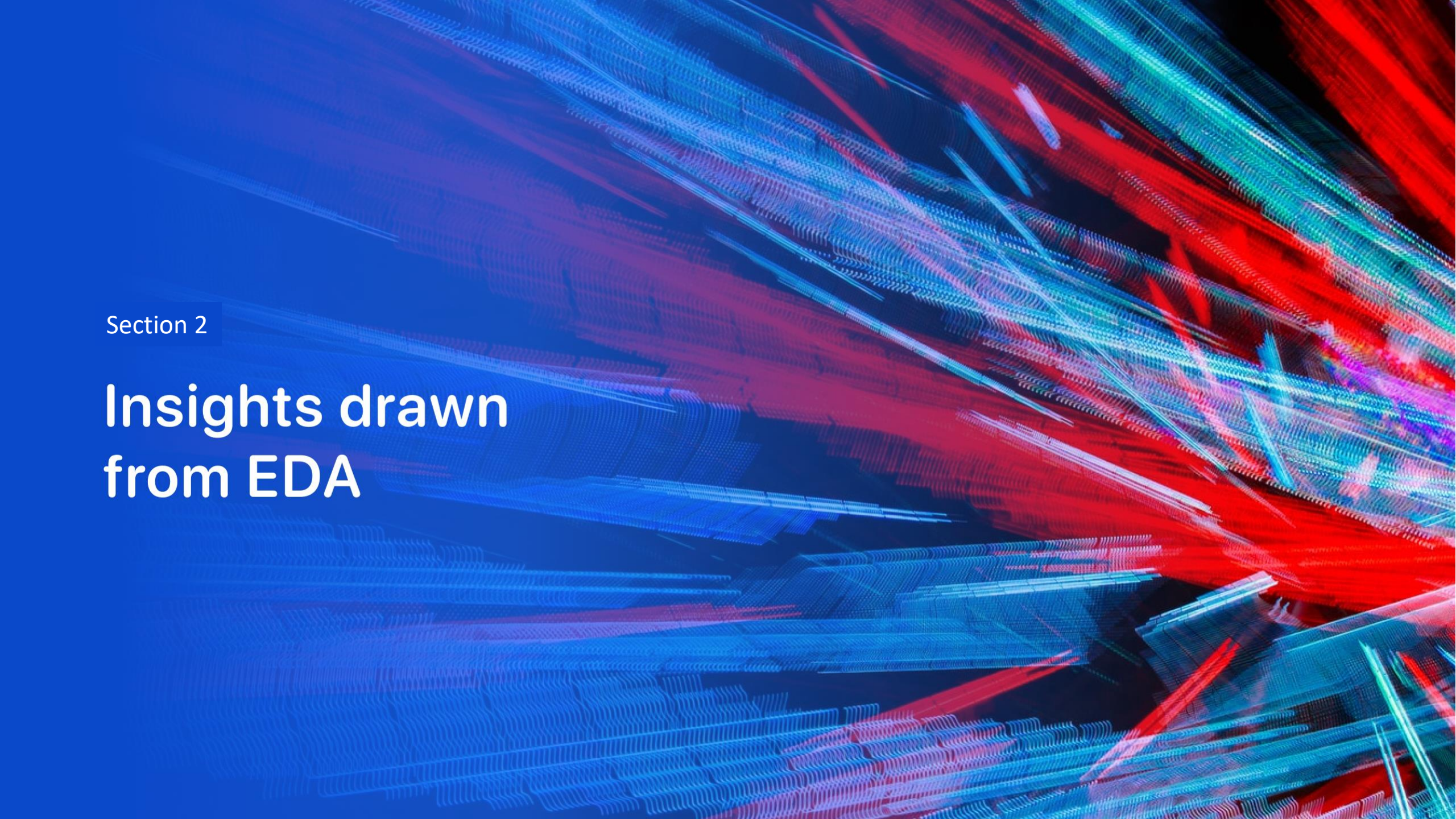
- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is Applied_Data_Science_Capstone/Interactive-Visual-Analytics-with-dash_app-jupyter-labs-spacex.py at main · RodrigueFomena/Applied_Data_Science_Capstone (github.com)

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.
- The link to the notebook is Applied_Data_Science_Capstone/Machine_Learning_Prediction-jupyter-labs-spacex.ipynb at main · RodrigueFomena/Applied_Data_Science_Capstone (github.com)

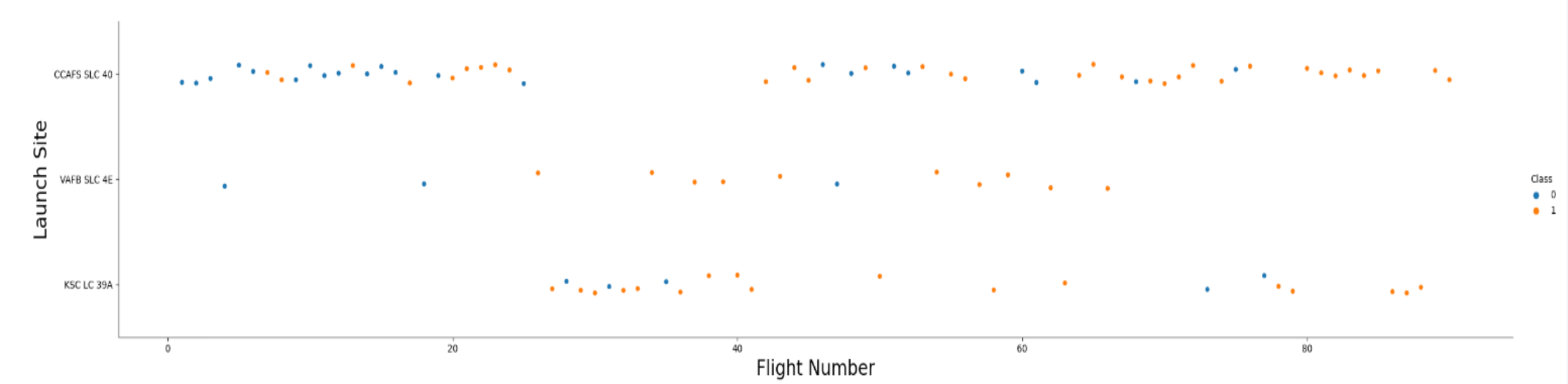
Results

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


Section 2
Insights drawn from EDA

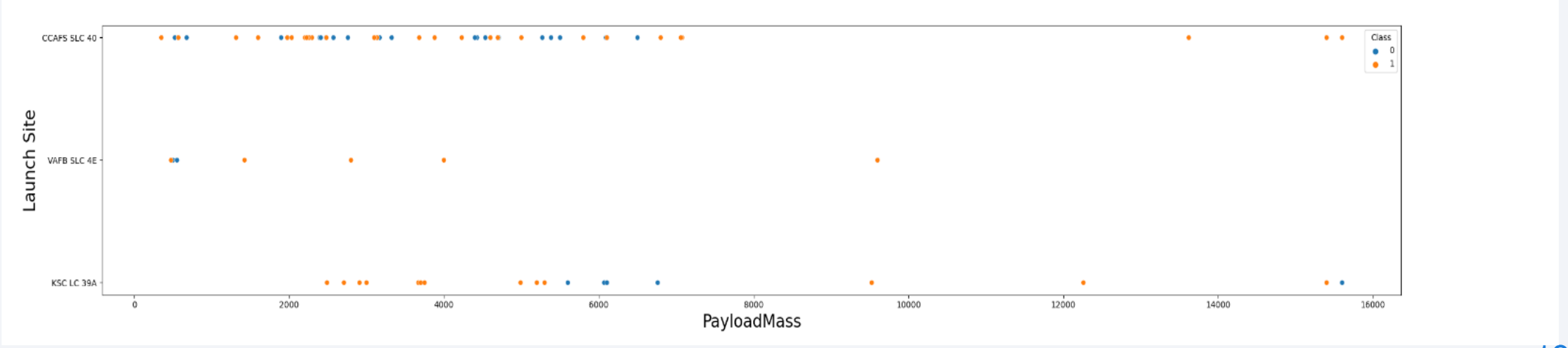
Flight Number vs. Launch Site

- We can observe from the plot below that:
  - For the LEO and ISS orbit the larger the flight amount at a launch site, the greater the success rate at a launch site.
  - For the GTO orbit, there is no clear relationship between the orbit type and number of flight.



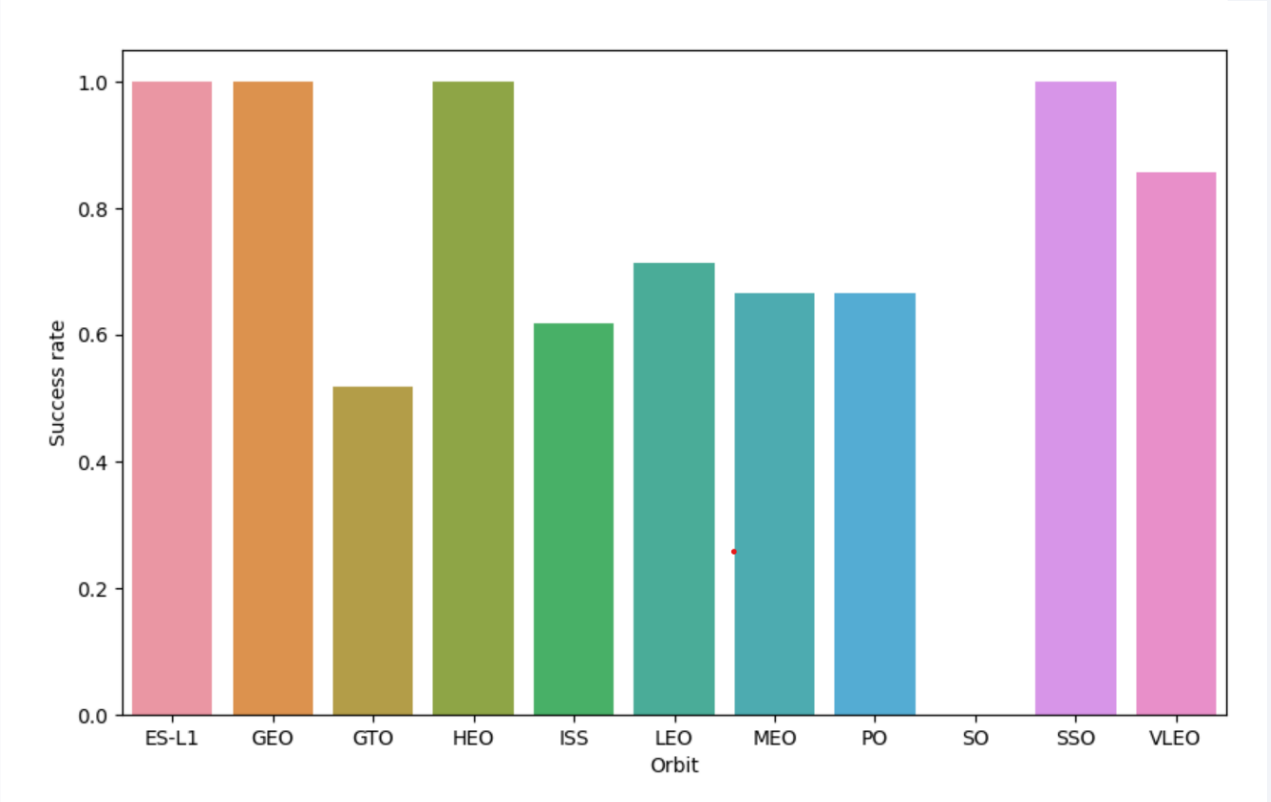
Payload vs. Launch Site

- From the plot below, we can observe that the higher the payload mass, the higher the number of successful launches mostly for the launch site VAFB SLC 4E and CCAF SLC 40.



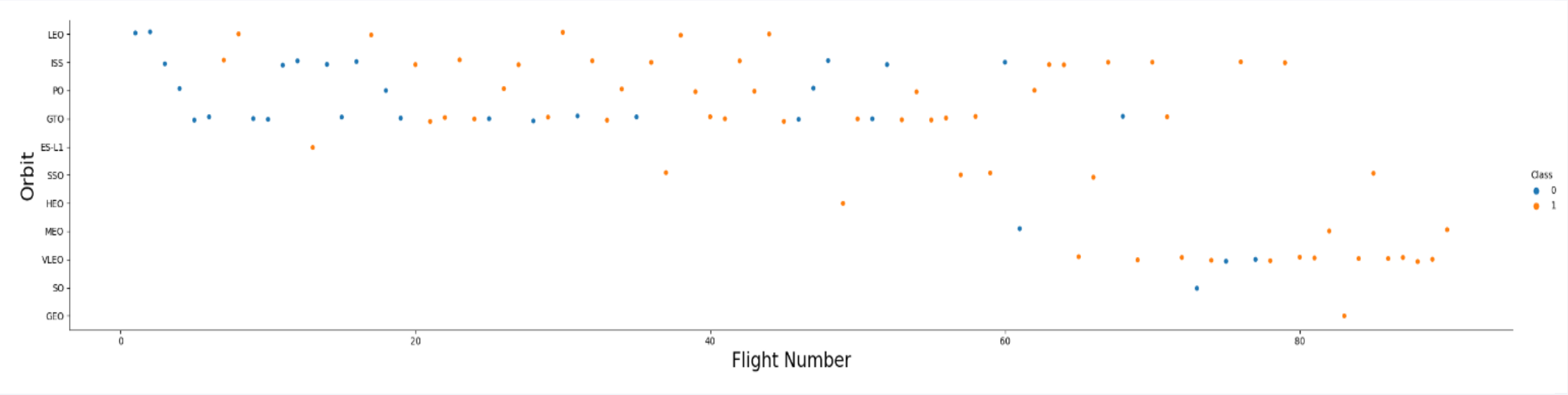
Success Rate vs. Orbit Type

- We can observe that ES-L1, GEO, HEO, SSO have the most success rate and VLEO comes at the second position.



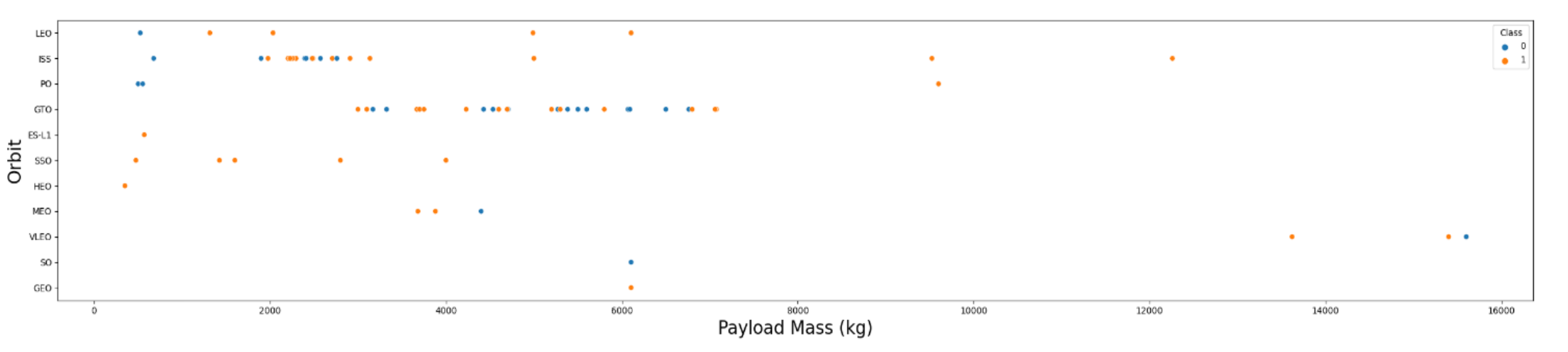
Flight Number vs. Orbit Type

- We can observe from this chart below that:
  - In the LEO and ISS orbit, succes rate is relative to the number of flight taken.
  - There is no observable relationship between the orbit type and number of flight in general.



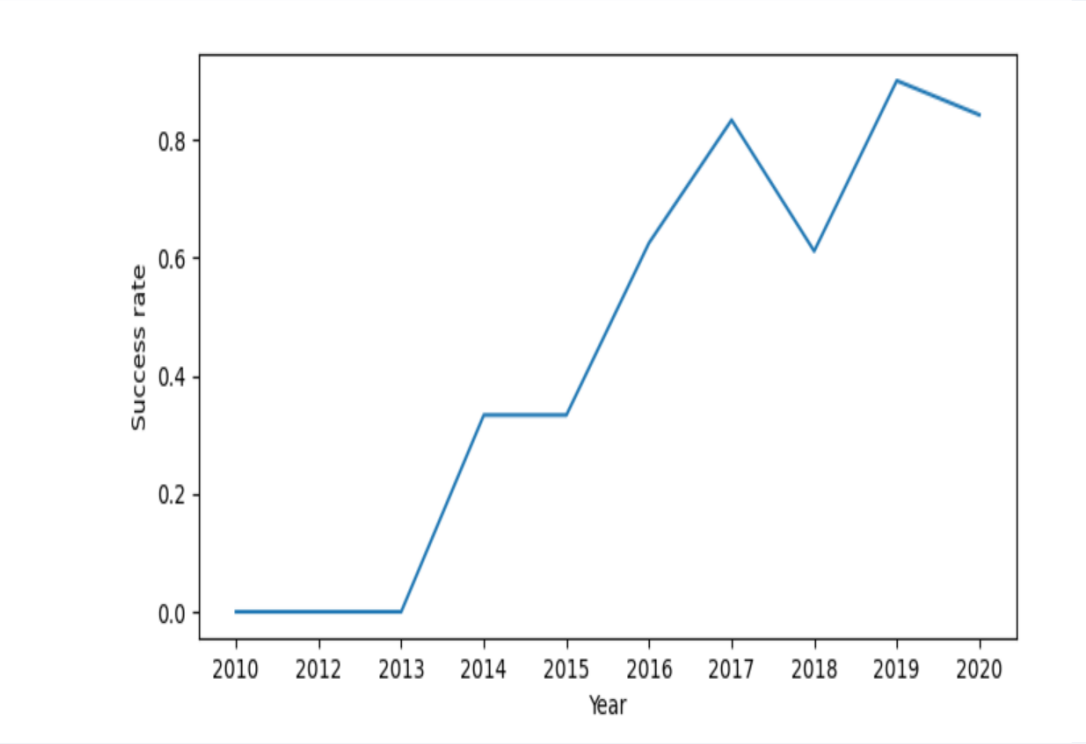
Payload vs. Orbit Type

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



Launch Success Yearly Trend

- we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- We applied the SLECT DISTINCT query to show only unique launch sites from the SpaceX data.

Task 1

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

In [56]:

```
cur.execute("SELECT DISTINCT launch_site FROM SPACEXTBL;")
result=cur.fetchall()
df1 = pd.DataFrame(result, columns=['launch_site'])
df1

#%%sql
#SELECT DISTINCT launch_site
#FROM SPACEXTBL;
```

Out[56]:

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

Launch Site Names Begin with 'CCA'

- We used the LIKE query to display 5 launch site names that begin with `CCA`

Task 2

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

In [55]:

```
cur.execute("SELECT * FROM SPACEXTBL WHERE launch_site LIKE 'CCA%'LIMIT 5;")
five_records = cur.fetchall()
df2 = pd.DataFrame(five_records, columns=[i[0] for i in cur.description])
df2
#for row in five_records:
    #print(row)
```

Out[55]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	08-12-2010	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	22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We used this query below to calculate the total payload carried by boosters from NASA and the result is 45596

Task 3

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

In [54]:
```
cur.execute("SELECT SUM(payload_mass__kg_) FROM SPACEXTBL WHERE customer = 'NASA (CRS)';")
cur.fetchall()
```

Out[54]:
```
[(45596,)]
```

Average Payload Mass by F9 v1.1

- We used the query below to calculate the average payload mass carried by booster version F9 v1.1 and the result is 2928.4

Task 4

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

In [53]:
```
cur.execute("SELECT AVG(payload_mass__kg_) FROM SPACEXTBL WHERE booster_version = 'F9 v1.1';")
cur.fetchall()
```

Out[53]:
```
[(2928.4,)]
```

First Successful Ground Landing Date

- We used the query below to return the date of the first successful landing outcome on ground

```
In [67]: cur.execute('SELECT MIN(Date) AS FirstSuccessfull_landing_date FROM SPACEXTBL WHERE "Landing _Outcome" LIKE "Success (ground pad)";')
         cur.fetchall()

Out[67]: [('01-05-2017',)]
```

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

[94]:
```
cur.execute('SELECT Booster_Version FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)"\
            AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;')
Success_Booster=cur.fetchall()
df6=pd.DataFrame(Success_Booster, columns=[i[0] for i in cur.description])
df6
```

[94]:

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

- We used this query to filter for boosters which have successfully landed on drone and to determine successful landing with payload mass greater than 4000 and less than 6000

Total Number of Successful and Failure Mission Outcomes

- We used these queries to filter the total number of failure and success mission outcomes.

Task 7

List the total number of successful and failure mission outcomes

In [75]:
```
#number of successful mission outcomes
cur.execute("SELECT COUNT(mission_outcome) AS Success_outcome FROM SPACEXTBL WHERE mission_outcome LIKE 'Success%'")
success=cur.fetchall()
df7A = pd.DataFrame(success, columns=[i[0] for i in cur.description])
print(df7A.to_string(index=False))
```

```
Success_outcome
            100
```

In [51]:
```
#number of failure mission outcomes
cur.execute("SELECT COUNT(mission_outcome) AS Failure_outcome FROM SPACEXTBL WHERE mission_outcome LIKE 'Failure%';")
success=cur.fetchall()
df7B = pd.DataFrame(success, columns=[i[0] for i in cur.description])
print(df7B.to_string(index=False))
```

```
Failure_outcome
              1
```

Boosters with Maximum Payload

- We used the query below to determine the booster that have carried the maximum payload.

```
cur.execute("SELECT Booster_Version, Payload_mass__kg_ FROM SPACEXTBL\
            WHERE Payload_mass__kg_ = (SELECT MAX(Payload_mass__kg_) FROM SPACEXTBL)\
            ORDER BY Booster_Version")
max_mass=cur.fetchall()
df8 = pd.DataFrame(max_mass, columns=[i[0] for i in cur.description])
df8
```

	Booster_Version	PAYLOAD_MASS__KG_
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

- We used these query to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
cur.execute('SELECT SUBSTR(Date, 4, 2) AS Month, "Landing _Outcome", Booster_Version, Launch_Site\
             FROM SPACEXTBL WHERE SUBSTR(Date,7,4)="2015" AND "Landing _Outcome" = "Failure (drone ship)";')
month_2015=cur.fetchall()
df9 = pd.DataFrame(month_2015, columns=[i[0] for i in cur.description])
df9
```

	Month	Landing _Outcome	Booster_Version	Launch_Site
0	01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
1	04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- We selected Landing outcomes and the count of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BEWEEN 04-06-2010 to 20-03-2010.
- We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

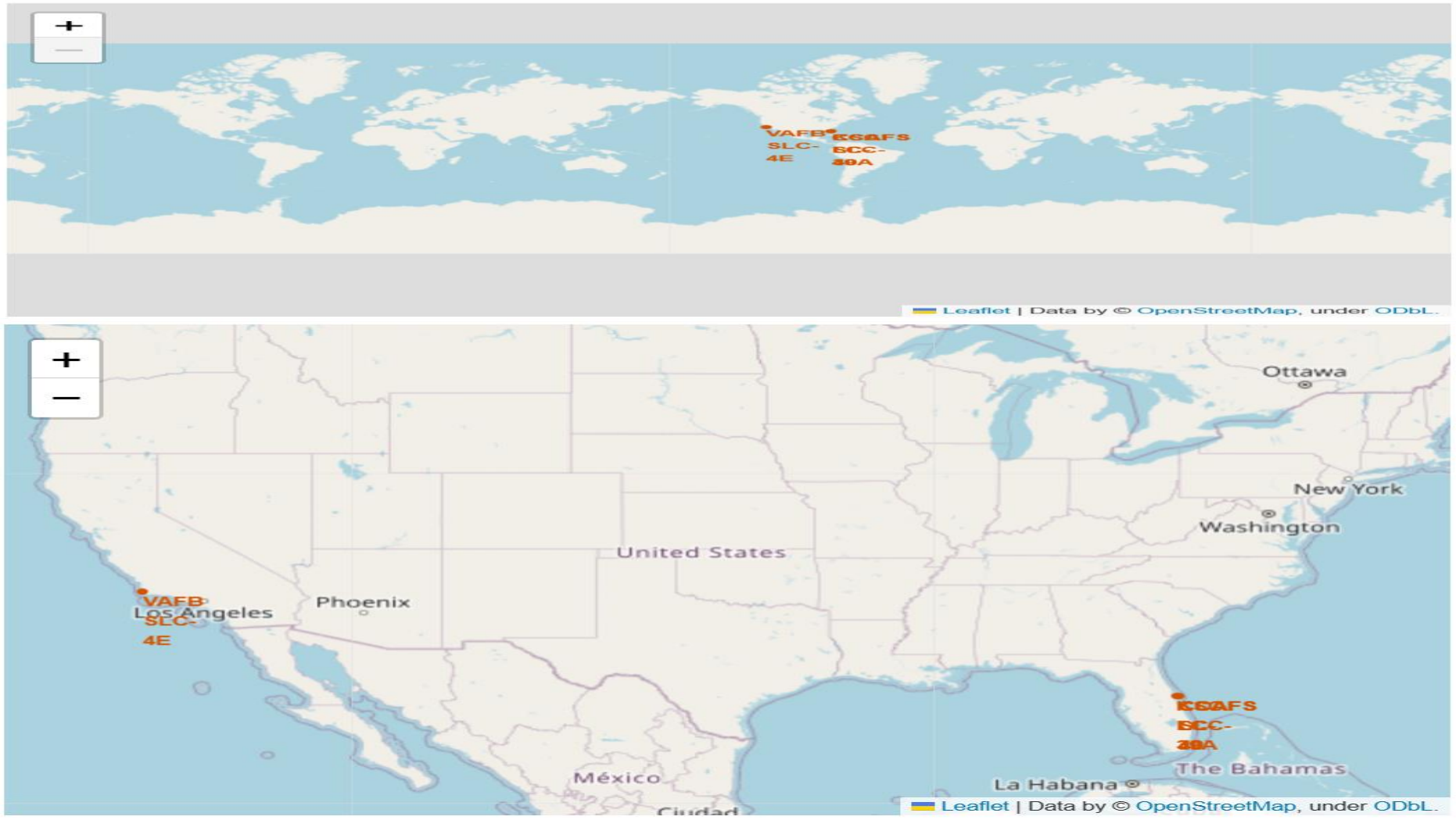
```
cur.execute('SELECT "Landing _Outcome", COUNT("Landing _Outcome") AS Count_Landing_Outcome\
            FROM SPACEXTBL WHERE DATE BETWEEN "04-06-2010" AND "20-03-2017"\
            GROUP BY "Landing _Outcome" ORDER BY COUNT("Landing _Outcome") DESC ;')
Rank=cur.fetchall()
df10 = pd.DataFrame(Rank, columns=[i[0] for i in cur.description])
df10
```

	Landing _Outcome	Count_Landing_Outcome
0	Success	20
1	No attempt	10
2	Success (drone ship)	8
3	Success (ground pad)	6
4	Failure (drone ship)	4
5	Failure	3
6	Controlled (ocean)	3
7	Failure (parachute)	2
8	No attempt	1

Section 3

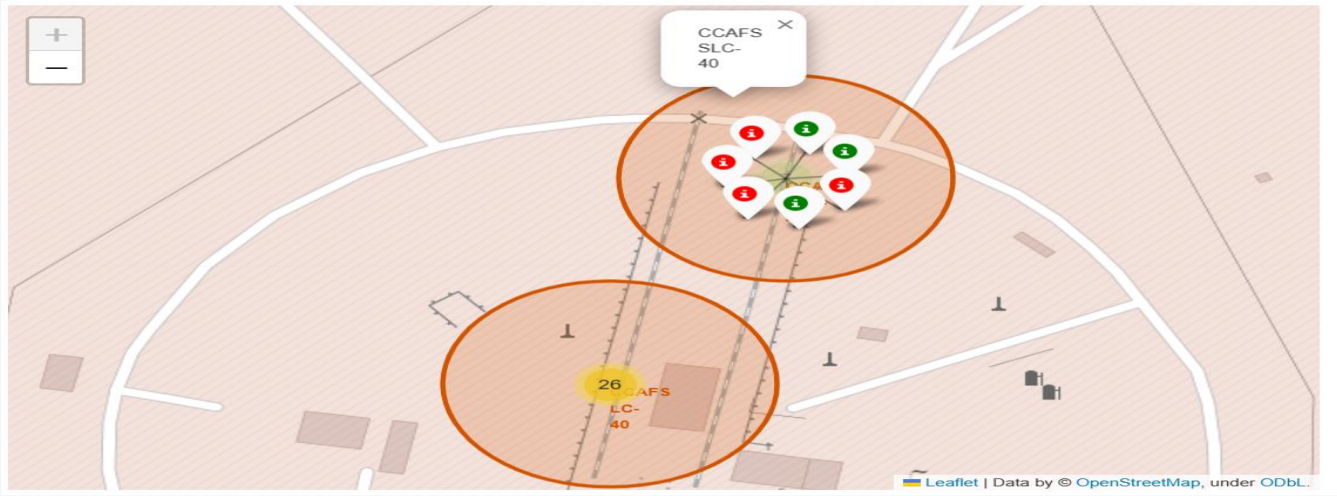
Launch Sites Proximities Analysis

All launch sites on the site map



- All launch sites are in the United States of America coasts and in proximity to the Equator line
- All launch sites in very close proximity to the coast

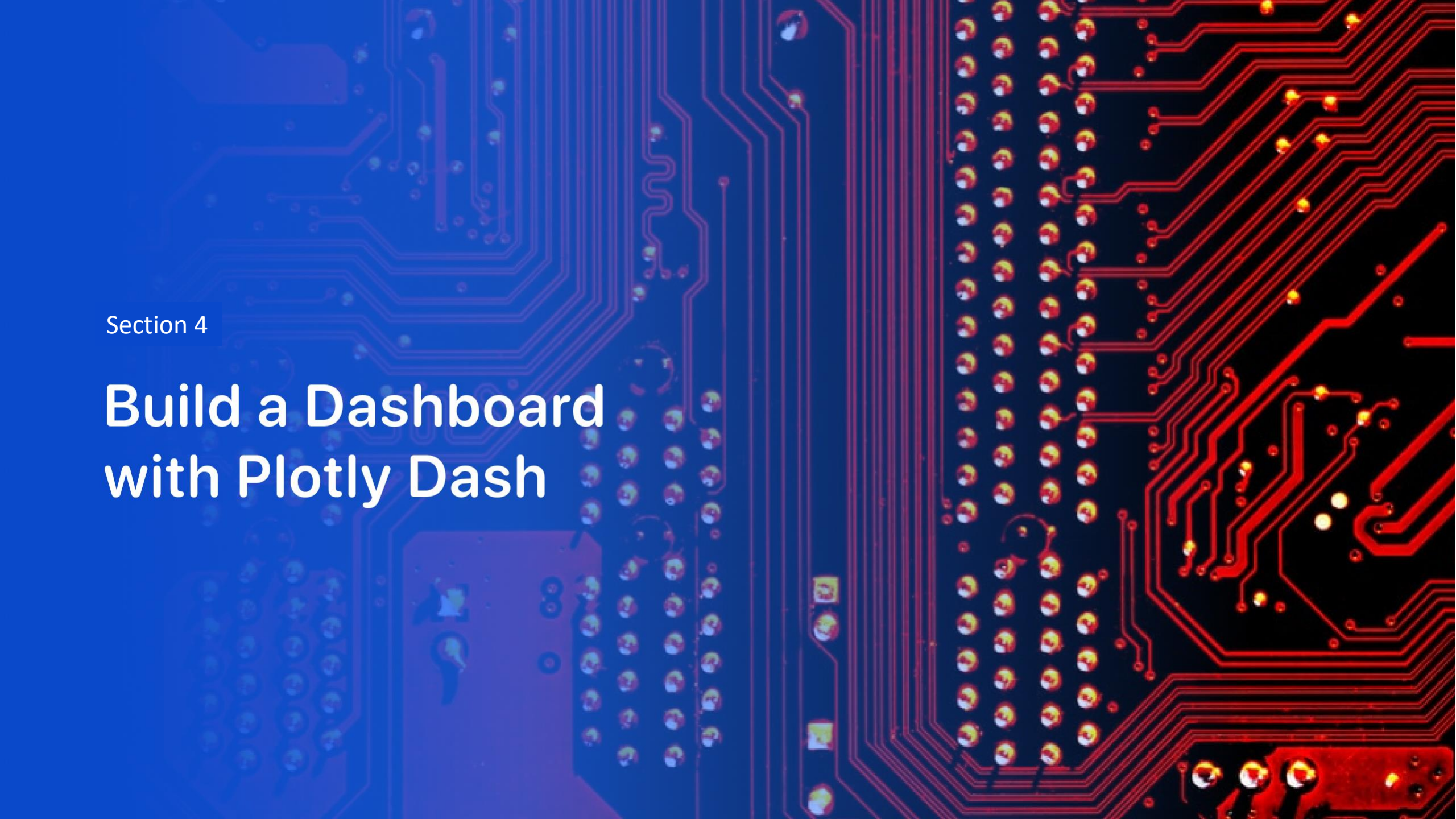
Mark of the success/failed launches for each site on the map



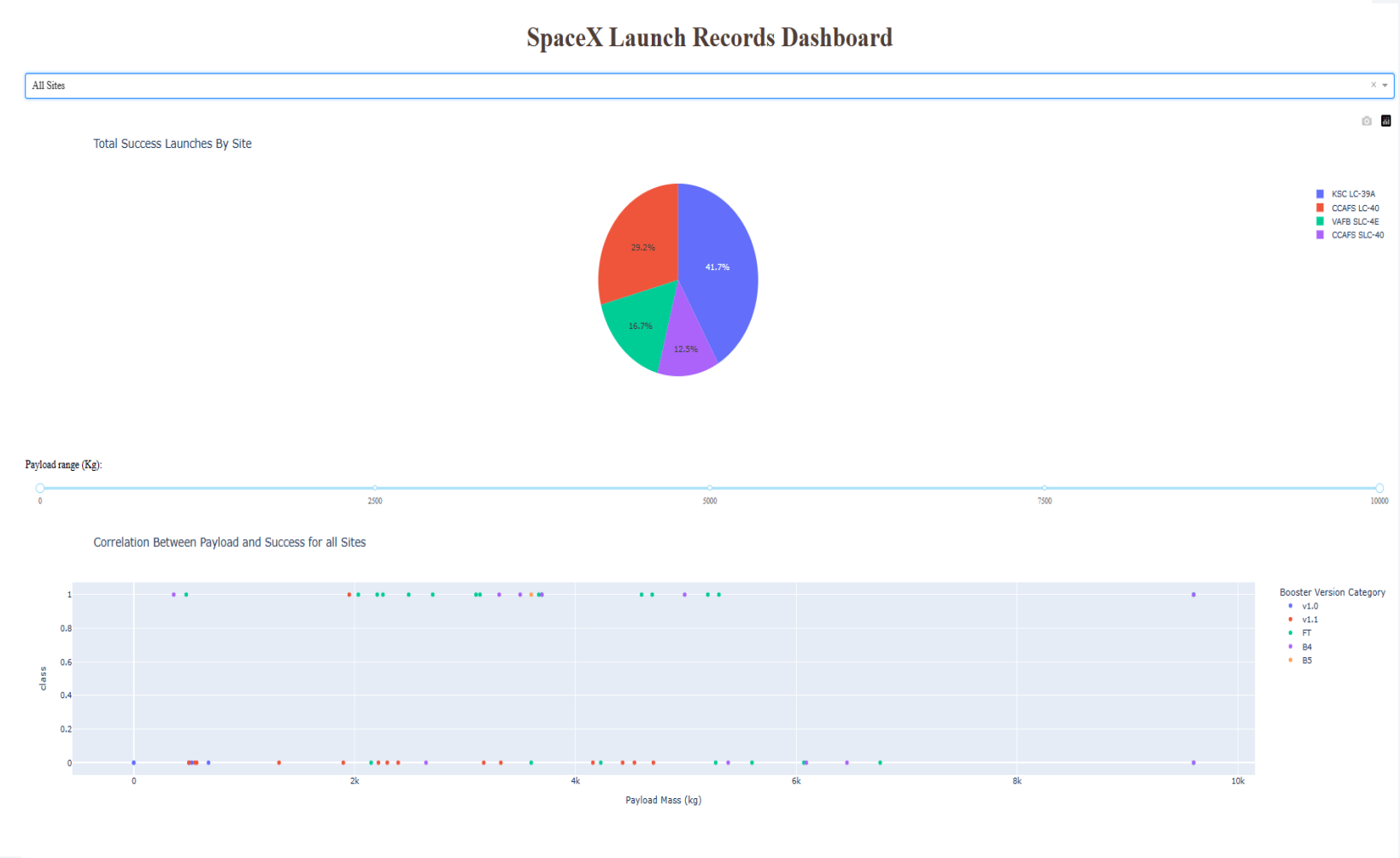
- The yellow circles represent the number of launches (success/failed)
- The green color Marker shows success launches and red color show failed launches

Launch site distance to coastline coordinates

- 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

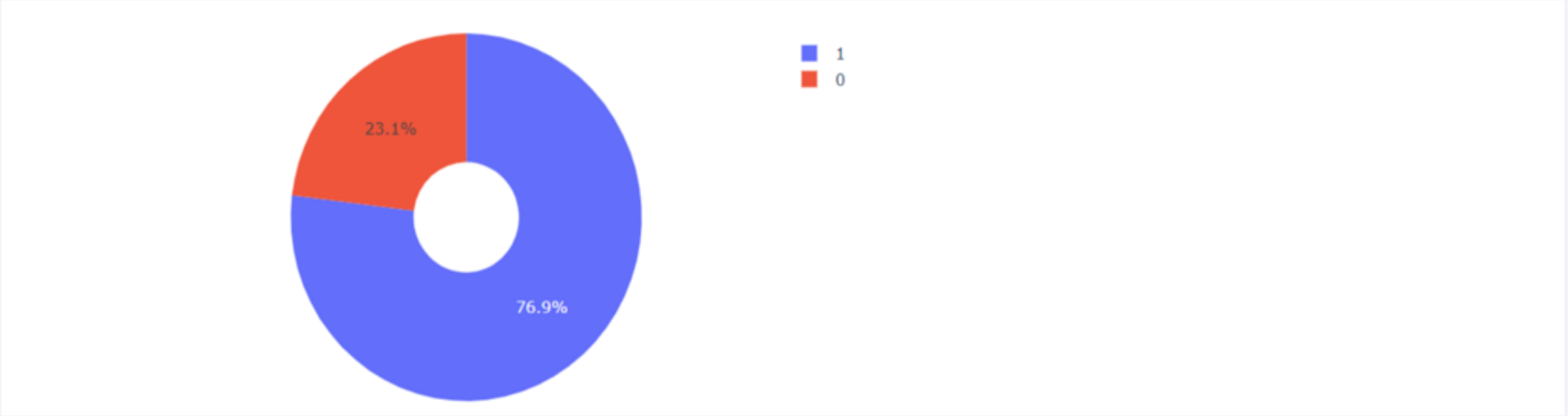
Dashboard Application showing the success percentage of each launch



- The KSC LC-39A has the most successful launches (41.7%)

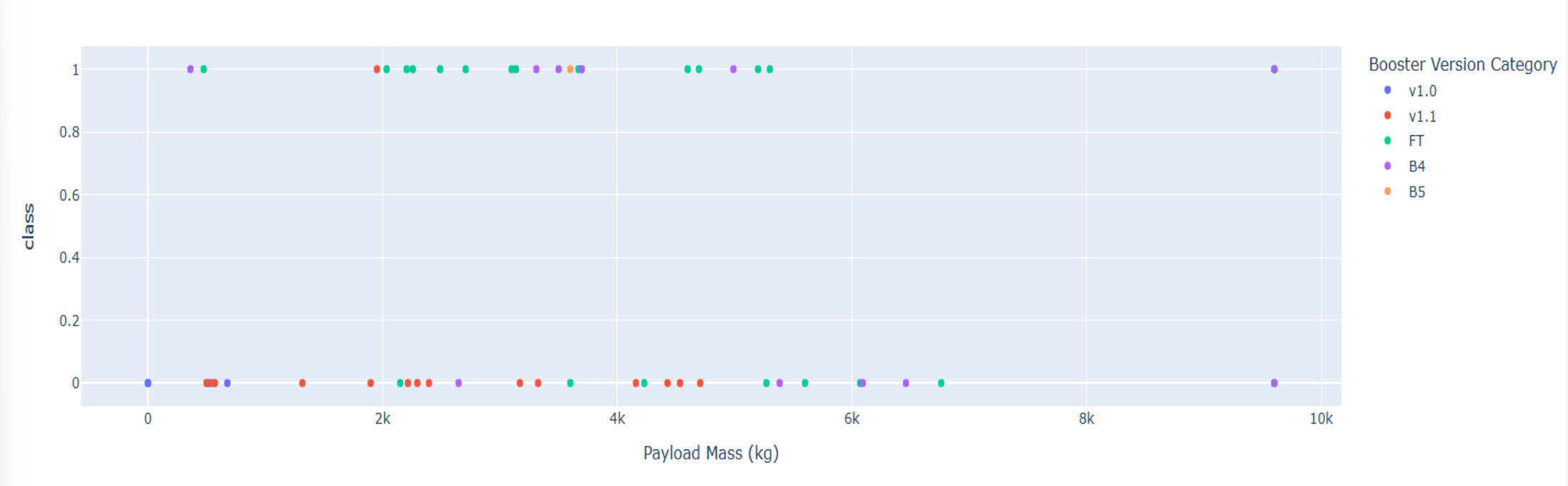
Dashboard for the Launch site (KSC LC-39A) with the highest launch success ratio

- This pie chart shows the highest launch site with 23.1% failure rate and 76.9% successful rate.



Correlation between the Payload Mass and the Success for all Sites

- This scatter plot shows the Launch Outcome (class) in function of the payload mass depending on the Booster Version Category



Section 5

Predictive Analysis (Classification)

Classification Accuracy

Find the method performs best:

In [37]:

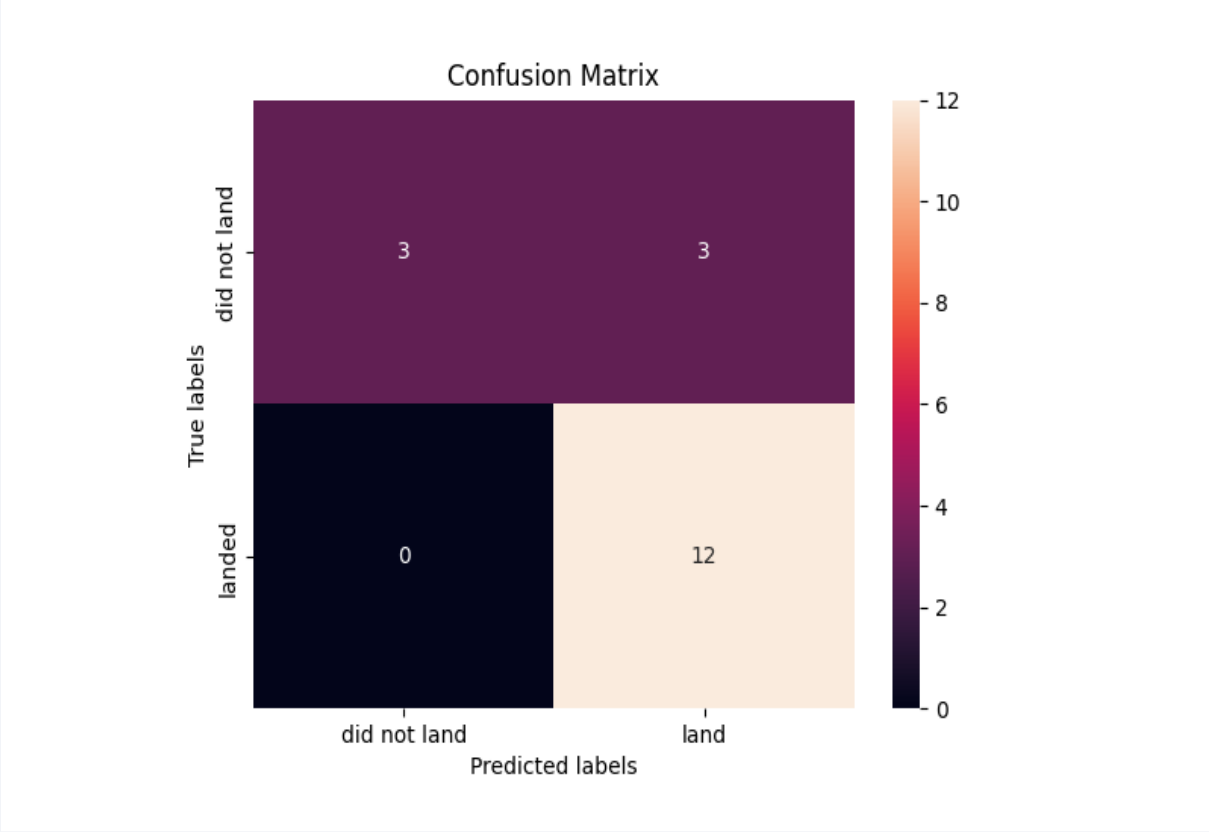
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.875
Best params is : {'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
```

- The Decision Tree Classifier is the model that performs the best with the highest score: 0.875

Confusion Matrix



- This heatmap represents the confusion matrix of the best performing model (Decision Tree Classifier) and we can see that there are:
  - no FN (False Negative)
  - 3 FP (False Positives) that represent what was not good predicted with the model

Conclusions

- It can be inferred that the success rate at a launch site is directly proportional to the number of flights launched.
- The launch success rate has been on the rise from 2013 to 2020.
- Orbits such as ES-L1, GEO, SSO, HEO, and VLEO exhibited the highest success rates.
- KSC LC-39A has had the highest number of successful launches among all the sites.
- Among the machine learning algorithms used for this task, the Decision tree classifier is the most effective.

Appendix

- See my Github directory for the entire applied data science capstone:
  RodrigueFomena/Applied_Data_Science_Capstone (github.com)

Thank you!