IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Carlos Olivella
February 8, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection through Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium and Plotly
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis Result
  - Interactive Analysis in Screenshots
  - Predictive Analytics Result

Introduction

- Project background and context
  - SpaceX advertises Falcon 9 rocket launches on its website, costing 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings because SpaceX 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 SpaceX for a rocket launch.
- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions need 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 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, and evaluate classification models

Data Collection

- Data Collection was done using various methods.
- 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 an HTML table, parse the table and convert it to a pandas data frame 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.

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_
```

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()`

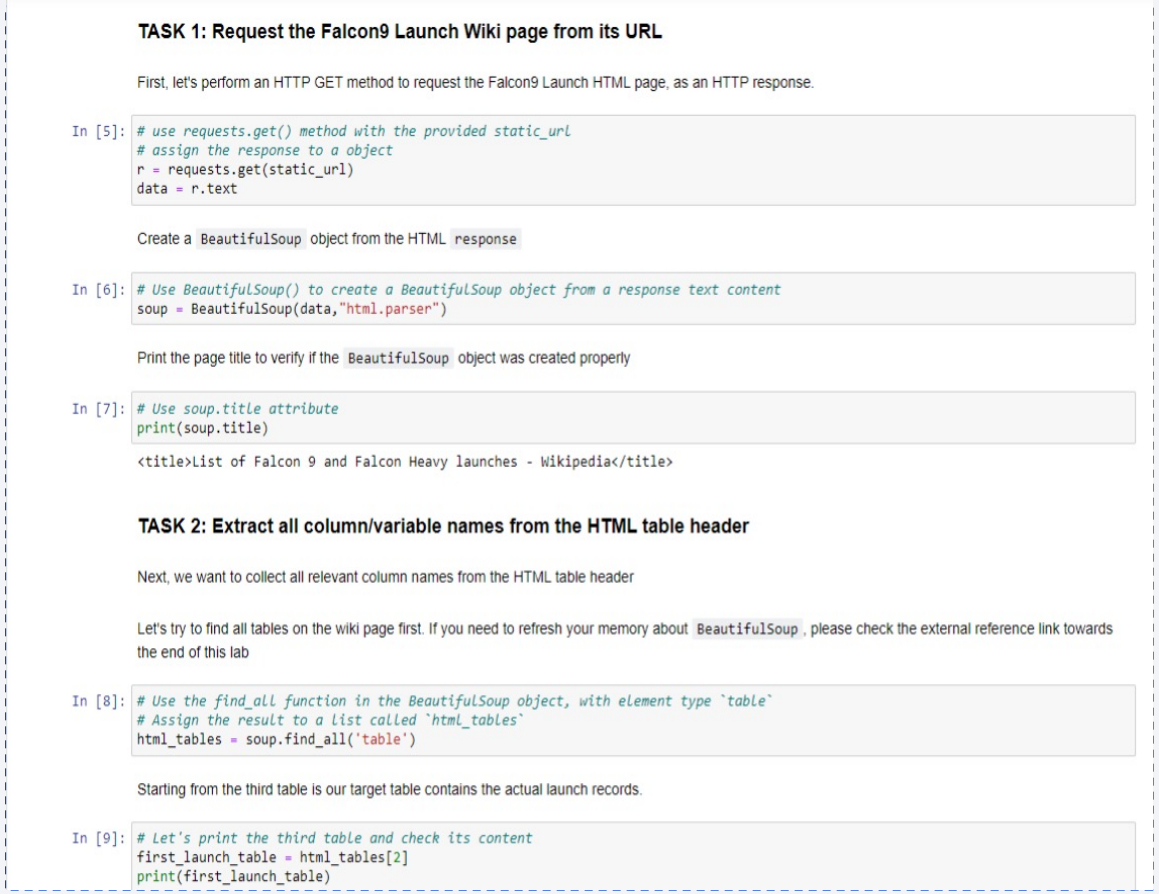
```
In [31]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())  # convert to flat table
         data.head()
```

Out[31]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3b
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}]	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 orbit, Failed to	[]	[]	[]	[5eb0e4b6b6c3b

Data Collection - Scraping

- We applied web scrapping to web scrap Falcon 9 launch records with BeautifulSoup.
- We parsed the table and converted it into a pandas data frame.
- The GitHub URL of the completed web scraping notebook.



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
r = requests.get(static_url)
data = r.text
```

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

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

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

In [7]:
```
# 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

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

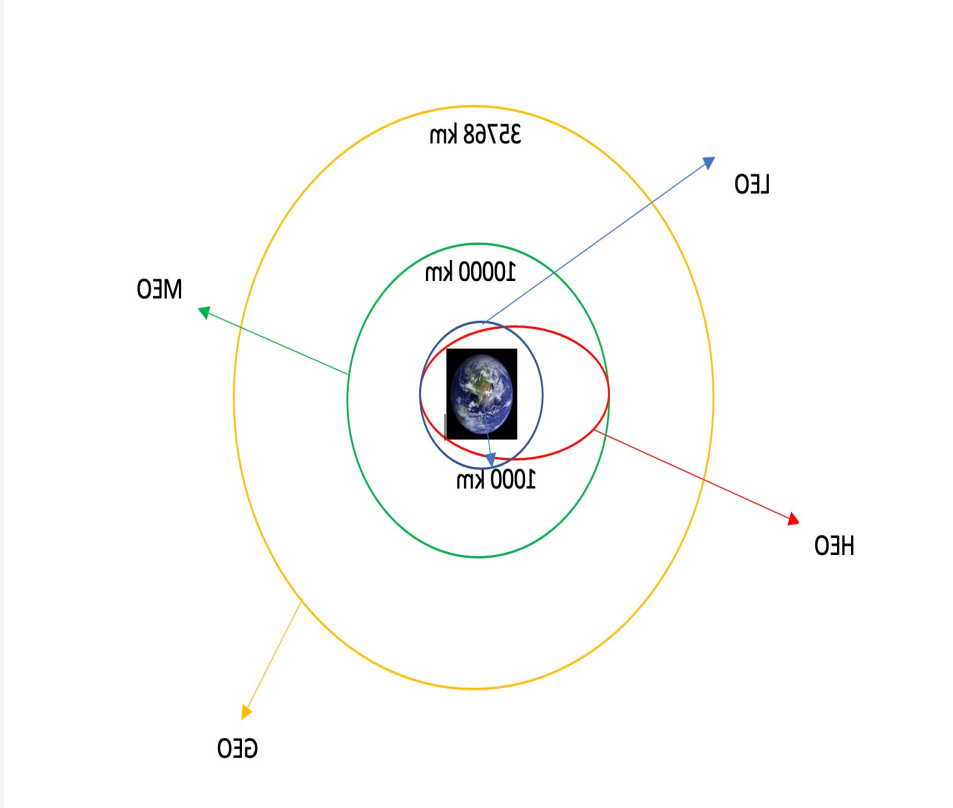
In [8]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

In [9]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

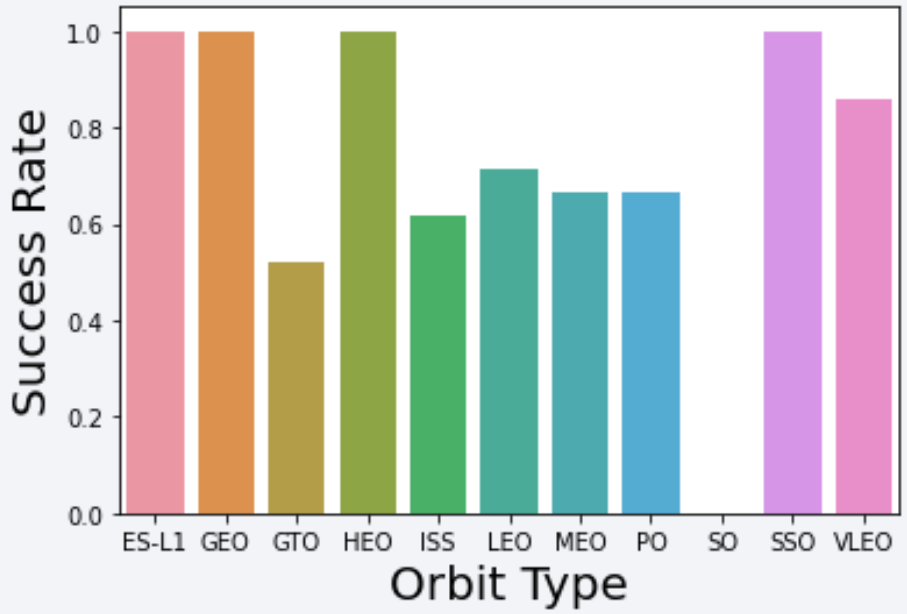
Data Wrangling

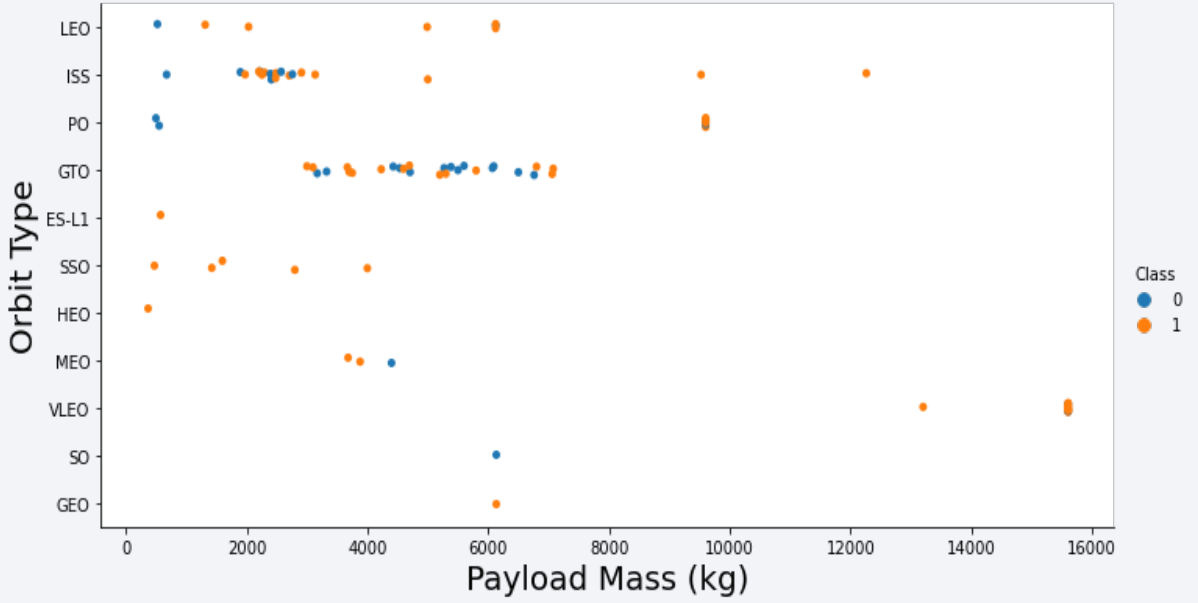
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbit.
- We created a landing outcome label from the outcome column and exported the results to CSV.
- The GitHub URL of the completed data wrangling related notebooks



EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, the success rate of each orbit type, flight number and orbit type, and the launch success yearly trend.
- The GitHub URL of the completed EDA with Data Visualization notebook.





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

EDA with SQL

- The GitHub URL of the completed EDA with SQL-related notebooks.

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

In [8]:
```
q = pd.read_sql("select * from spacexdata where Launch_Site like 'CCA%' limit 5", conn)
q
```

Out[8]:

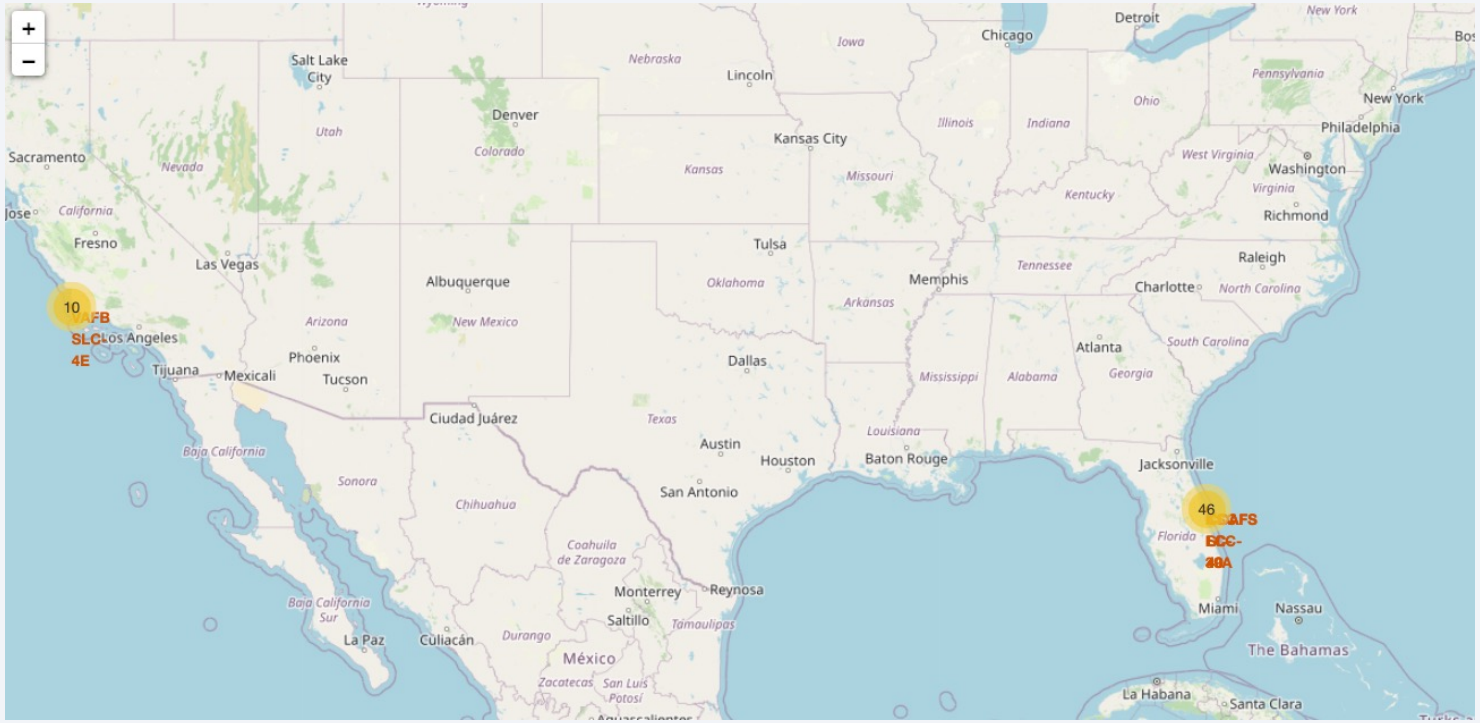
	index	Date	Time_(UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing__Outcome
0	0	2010-06-04 00:00:00	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	1	2010-12-08 00:00:00	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2	2012-05-22 00:00:00	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	3	2012-10-08 00:00:00	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	4	2013-03-01 00:00:00	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Build an Interactive Map with Folium

- We marked all launch sites and added map objects such as markers, circles, and 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 classes 0 and 1. i.e., 0 for failure, and 1 for success.
- Using the colored marker clusters, we identified which launch sites have a relatively high success rate.
- We calculated the distances between a launch site to its proximities.
- We answered some questions for instance:
  - Are launch sites near railways, highways, and coastlines?
  - Do launch sites keep a certain distance away from cities?

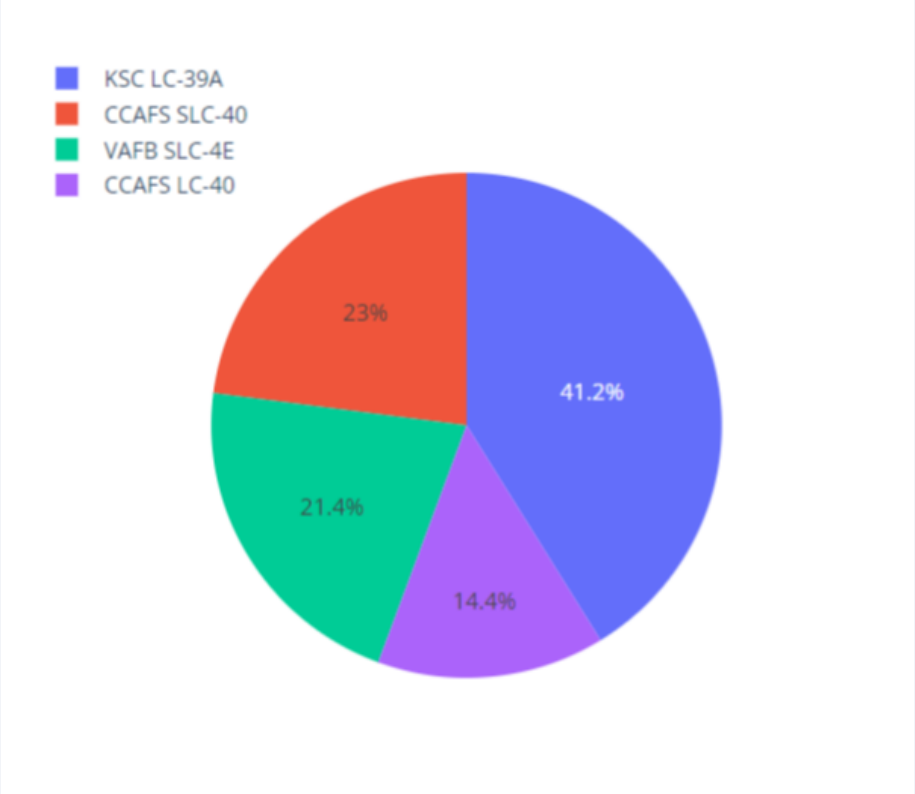
Build an Interactive Map with Folium

- The GitHub URL of the completed Interactive map with Folium-related notebooks.



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 site.
- We plotted a scatter graph showing the relationship between Outcome and Payload Mass (Kg) for the different booster versions.
- The GitHub URL of the completed Dashboard with Plotly-related notebooks

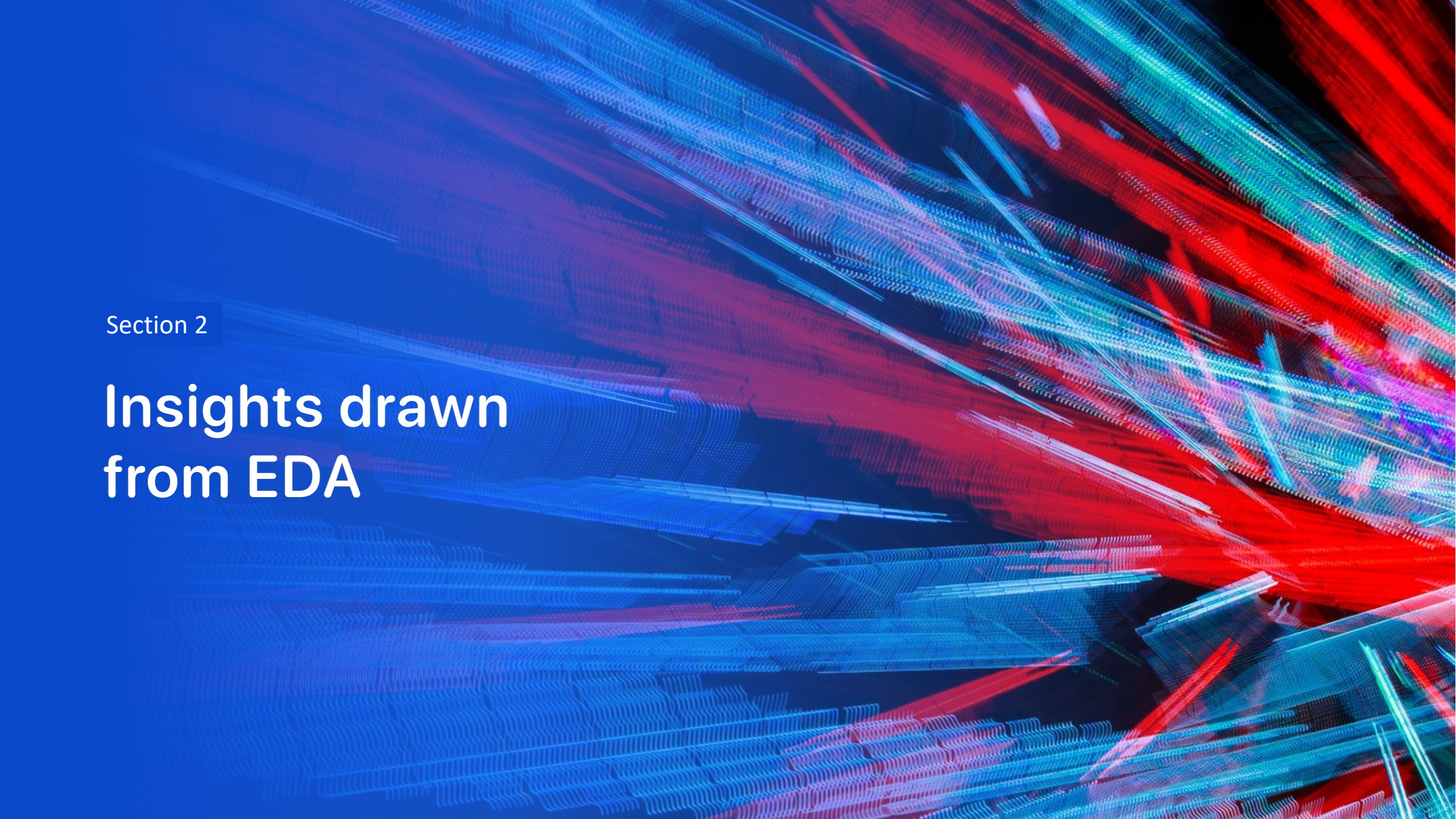


Predictive Analysis (Classification)

- We loaded the data using NumPy and pandas, transformed the data, and 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 and improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The GitHub URL of the completed best-performing Classification Model-related notebook.

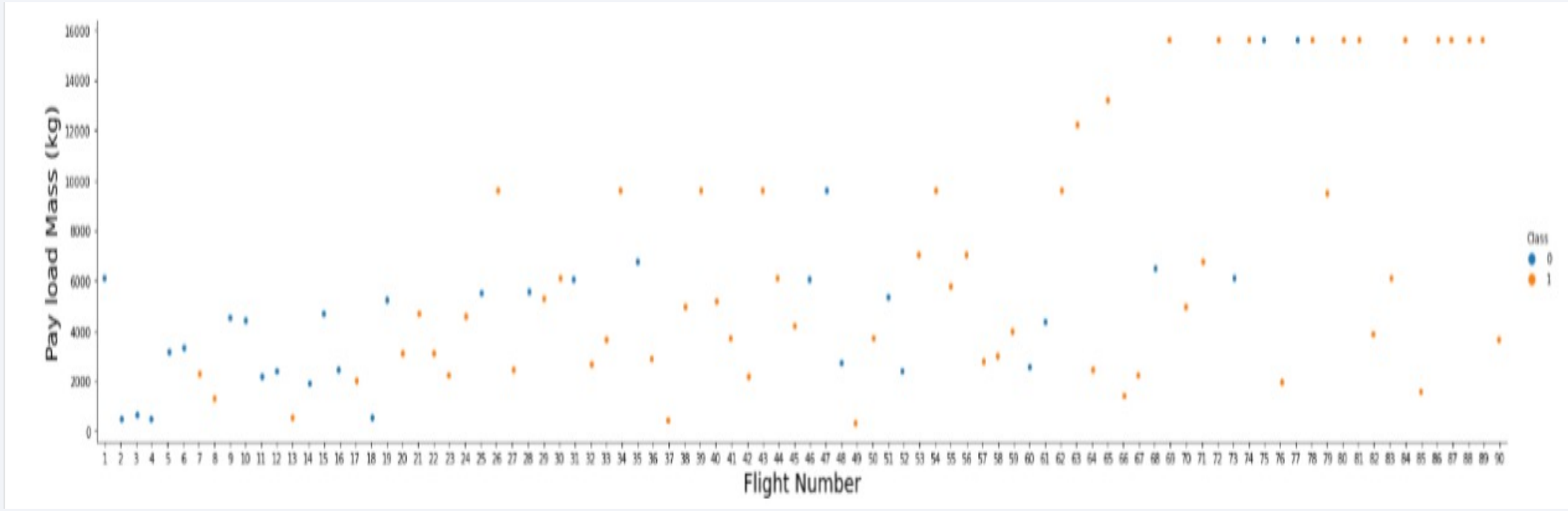
Results

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


Section 2
Insights drawn from EDA

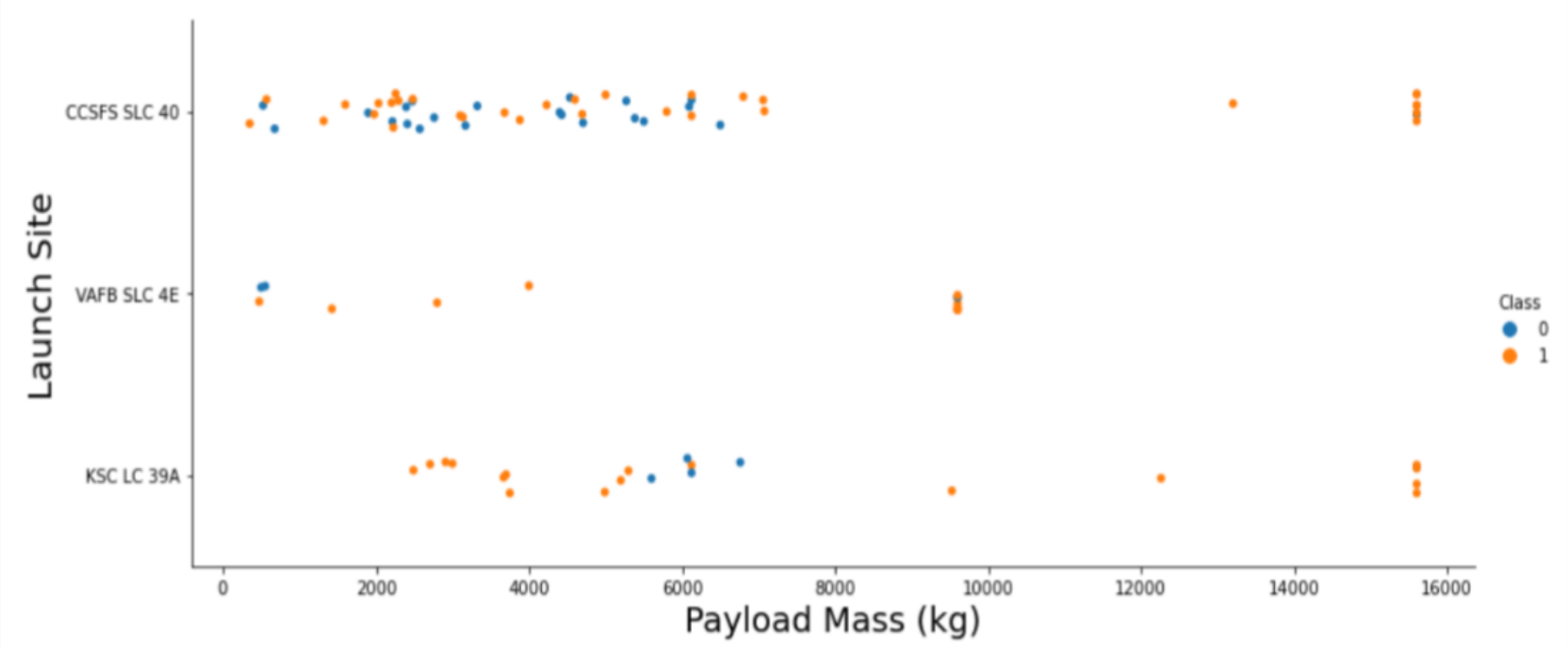
Flight Number vs. Launch Site

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



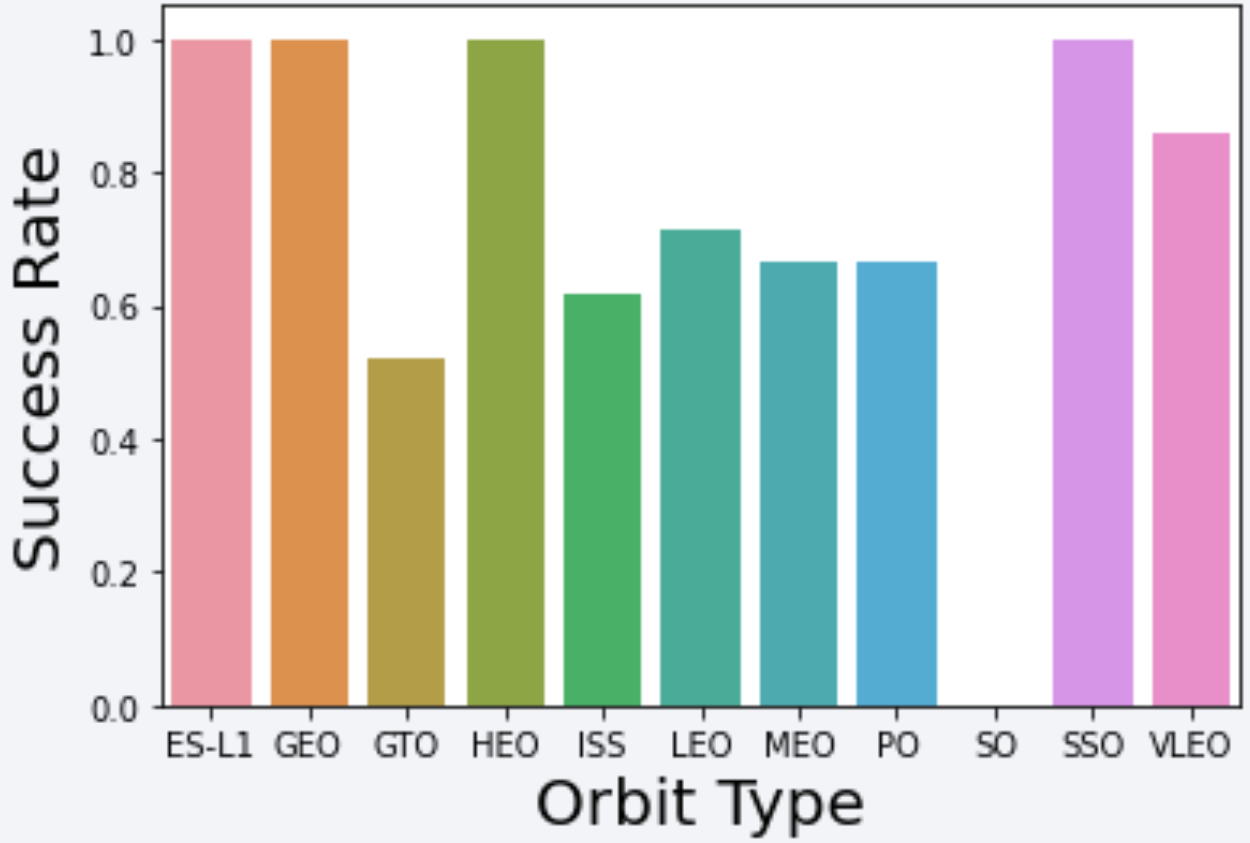
Payload vs. Launch Site

- Now if you observe Payload Vs. Launch Site scatters point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).



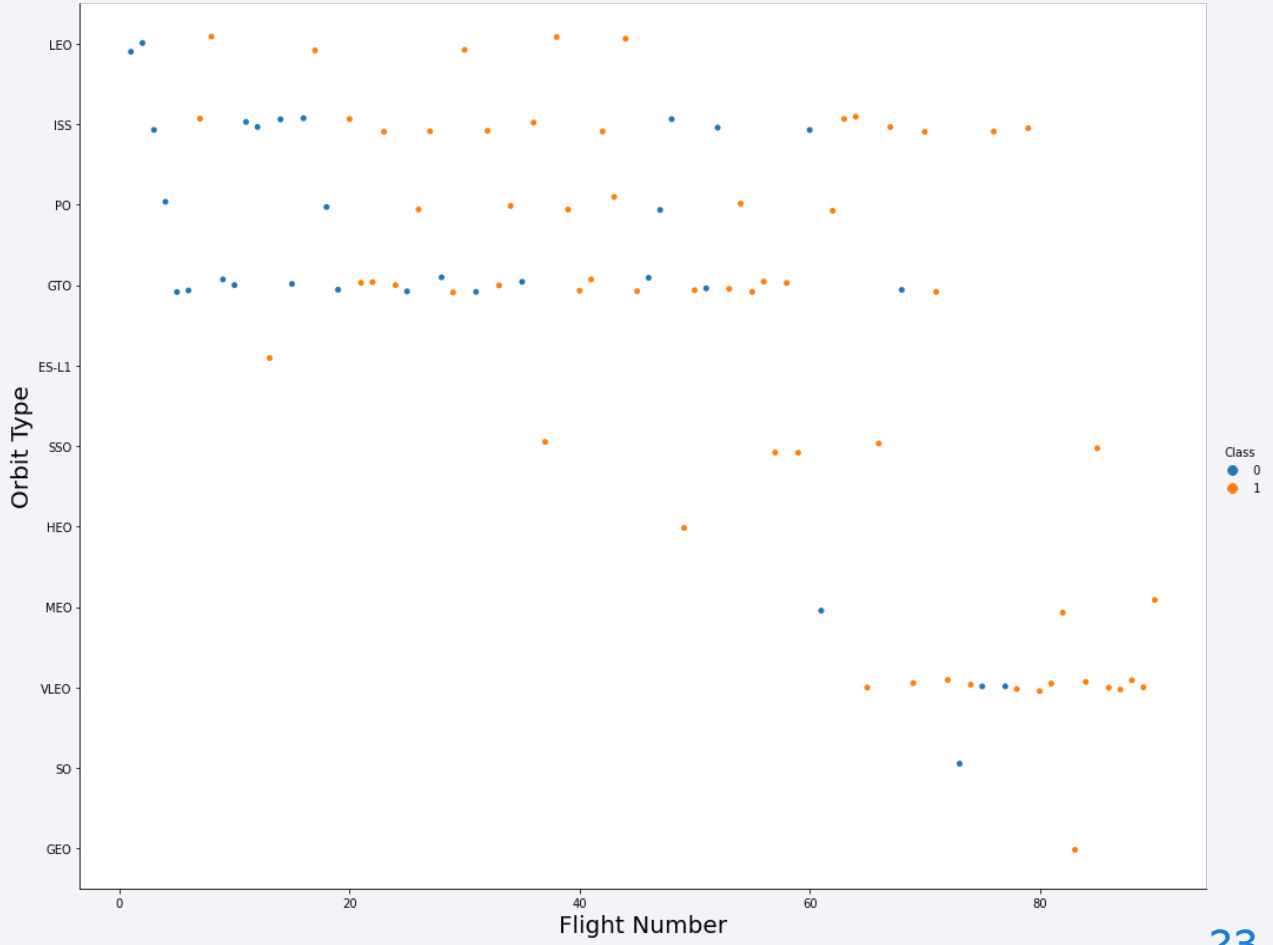
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, and VLEO had the most success rate



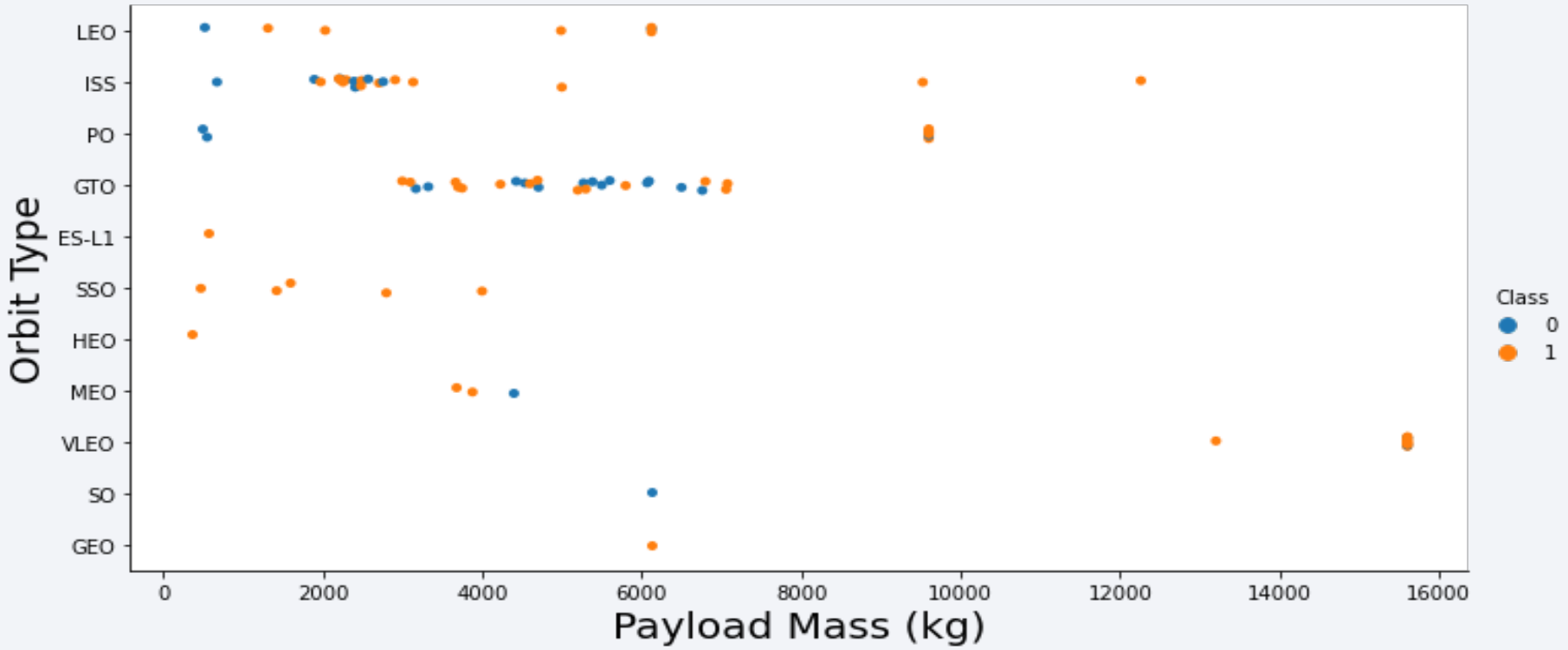
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe 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.



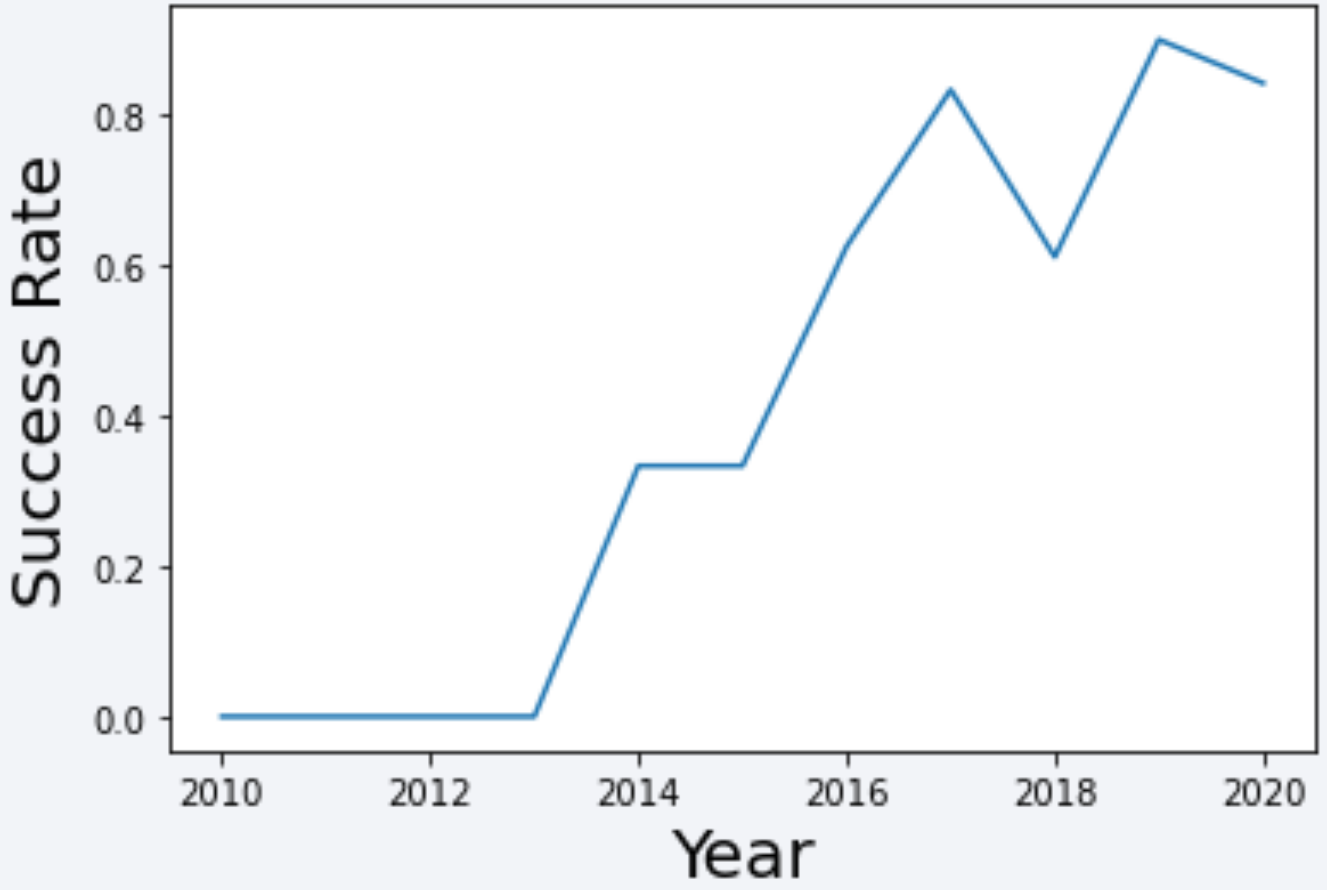
Payload vs. Orbit Type

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



Launch Success Yearly Trend

- From the plot, we can observe that the success rate since 2013 kept on increasing till 2020



All Launch Site Names

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

```
In [6]: q = pd.read_sql('select distinct Launch_Site from spacexdata', conn)
        q
```

Out[6]:

	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 query above to display 5 records where launch sites begin with `CCA`.

```
In [8]: q = pd.read_sql("select * from spacexdata where Launch_Site like 'CCA%' limit 5", conn)
        q
```

Out[8]:

	index	Date	Time_(UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing__Outcome
0	0	2010-06-04 00:00:00	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	1	2010-12-08 00:00:00	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	2	2012-05-22 00:00:00	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	3	2012-10-08 00:00:00	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	4	2013-03-01 00:00:00	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 calculated the total payload carried by boosters from NASA as 45596 using the query below

```
In [9]: q = pd.read_sql("select sum(PAYLOAD_MASS__KG_) from spacexdata where Customer='NASA (CRS)'", conn)
        q
```

Out[9]:

	sum(PAYLOAD_MASS__KG_)
0	45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

```
In [10]: q = pd.read_sql("select avg(PAYLOAD_MASS__KG_) from spacexdata where Booster_Version='F9 v1.1'", conn)
         q
```

Out[10]:

	avg(PAYLOAD_MASS__KG_)
0	2928.4

First Successful Ground Landing Date

- We observed that the date of the first successful landing outcome on the ground pad were 22nd December 2015.

```
q = pd.read_sql("select min(Date) from spacexdata where Landing__Outcome='Success (ground pad)'", conn)
q
```

	min(Date)
0	2015-12-22 00:00:00

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
q = pd.read_sql("select distinct Booster_Version from spacexdata where Landing__Outcome='Success (drone ship)' and PAYLOAD_M
q
```

	Booster_Version
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

- We used wildcards like '%' to filter for WHERE Mission Outcome was a success or a failure.

```
q = pd.read_sql("select substr(Mission_Outcome,1,7) as Mission_Outcome, count(*) from spacexdata  group by 1", conn
q
```

	Mission_Outcome	count(*)
0	Failure	1
1	Success	100

Boosters Carried Maximum Payload

- We determined the booster that has carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

```
q = pd.read_sql("select distinct Booster_Version from spacexdata where PAYLOAD_MASS__KG
q
```

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

2015 Launch Records

- We used a combination of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for the year 2015.

```
q = pd.read_sql("select distinct Landing__Outcome, Booster_Version, Launch_Site from spacexdata where Landing__Outcome='Failure (
q
```

	Landing__Outcome	Booster_Version	Launch_Site
0	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
1	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
2	Failure (drone ship)	F9 v1.1 B1017	VAFB SLC-4E
3	Failure (drone ship)	F9 FT B1020	CCAFS LC-40
4	Failure (drone ship)	F9 FT B1024	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 BETWEEN 2010-06-04 to 2010-03-20.
- 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

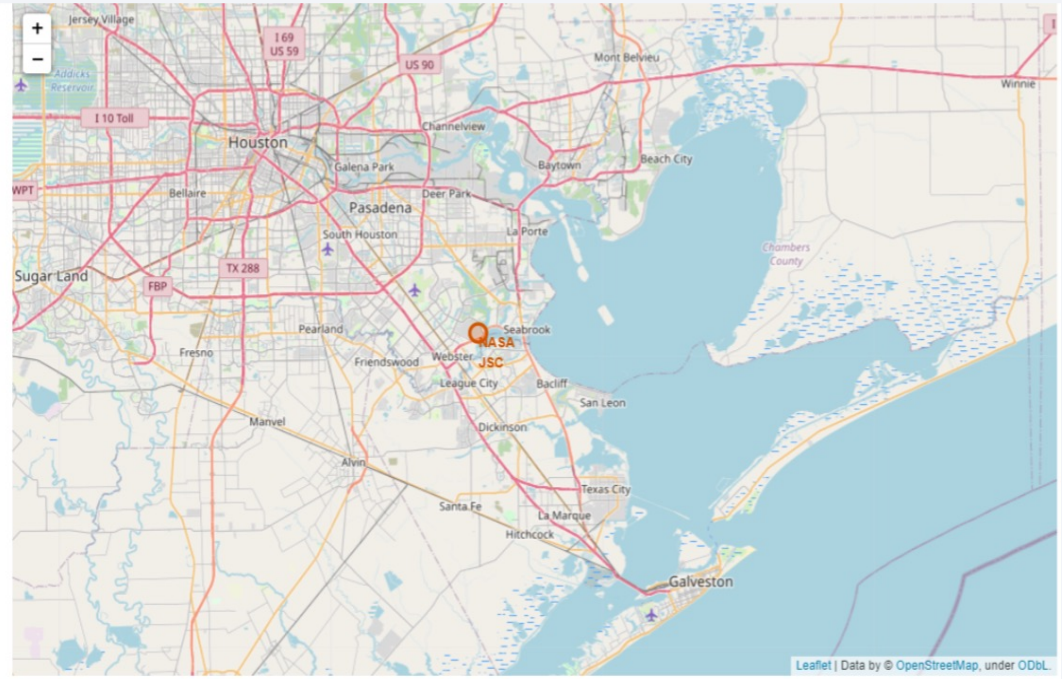
```
q = pd.read_sql("select Landing__Outcome, count(*) from spacexdata where Date between '2011-06-04' and '2017-03-20' group by La
q
```

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

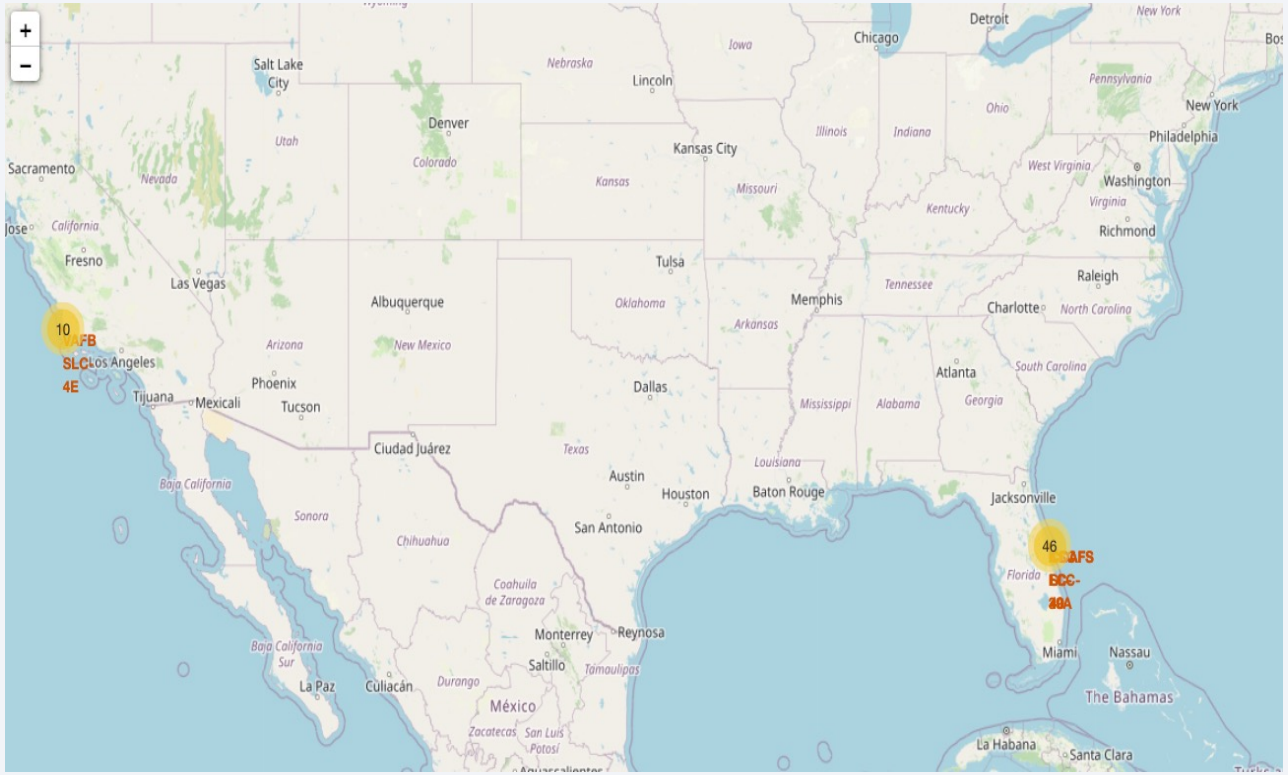

Section 3
Launch Sites
Proximities Analysis

All Launch Sites Global Map Markers

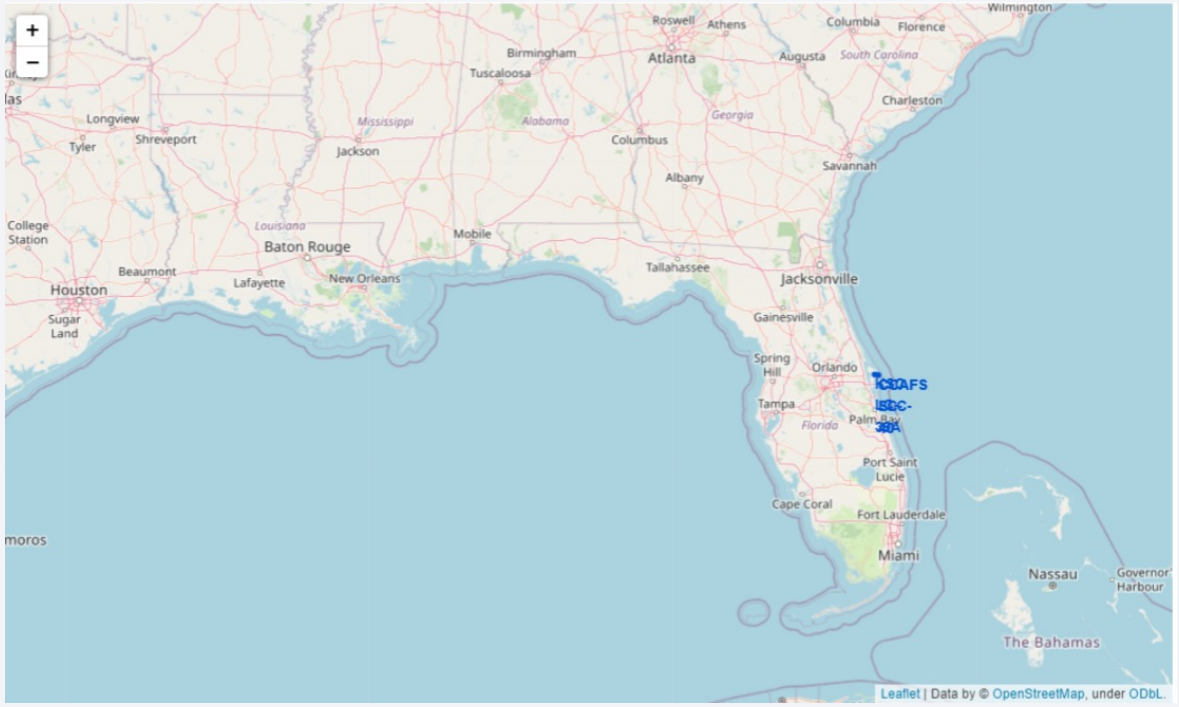
- We used folium. Circle to add a highlighted circle area with a text label on a specific coordinate
- We can see that the SpaceX launch sites are in Coastal regions of the United States of America.

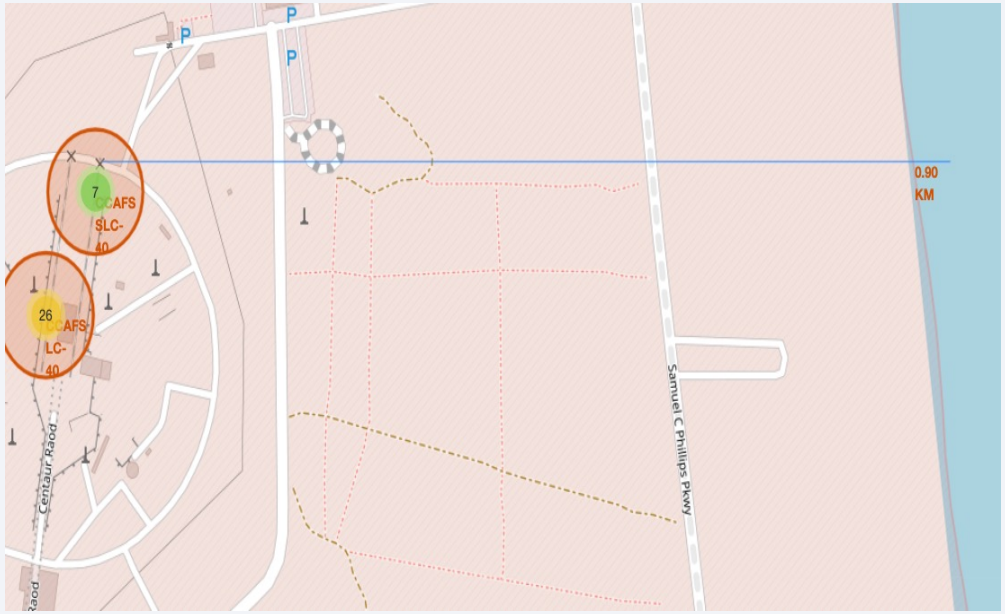


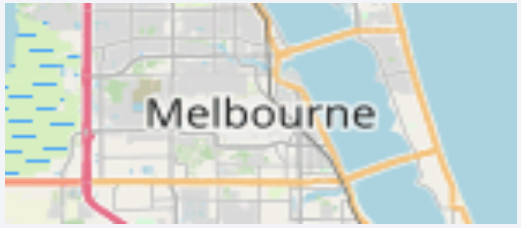
Markers showing launch sites with color labels

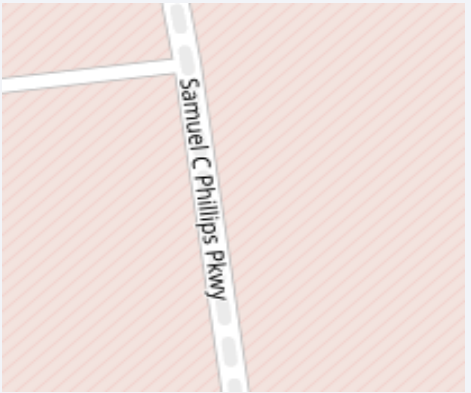


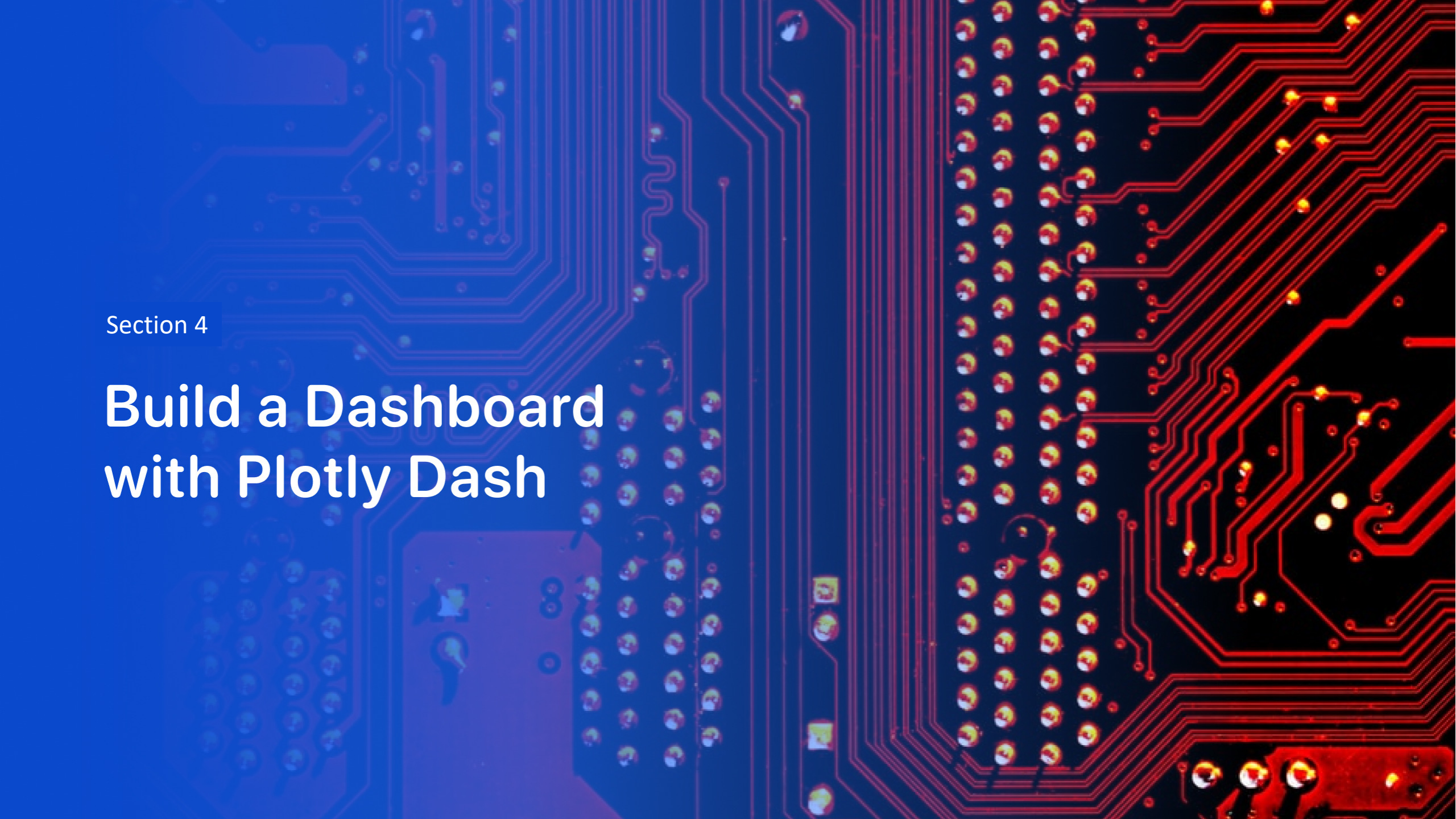
Launch Site distance to landmarks





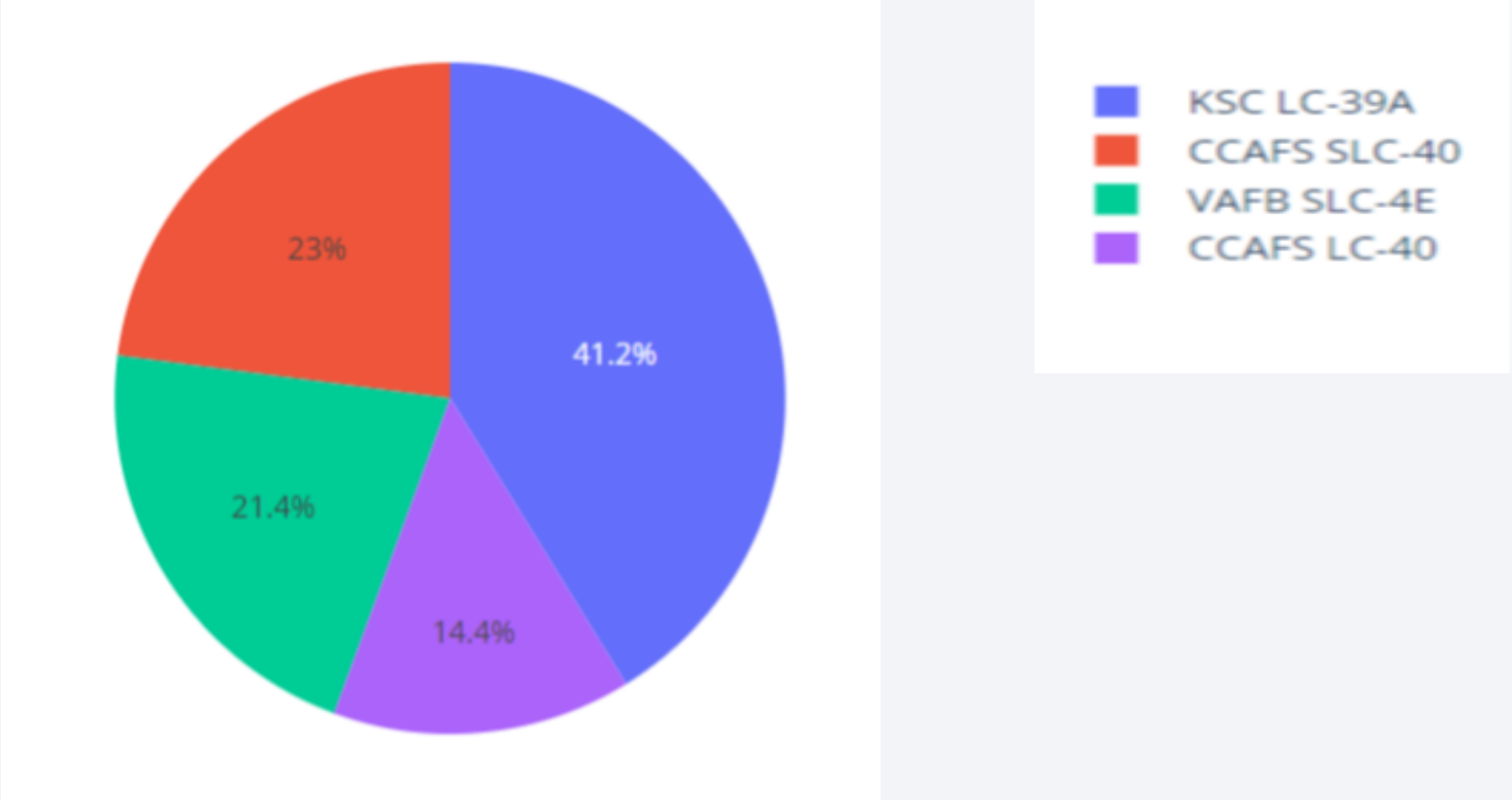





Section 4
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site

- We can see that KS LC-39A had the most successful launches from all sites.

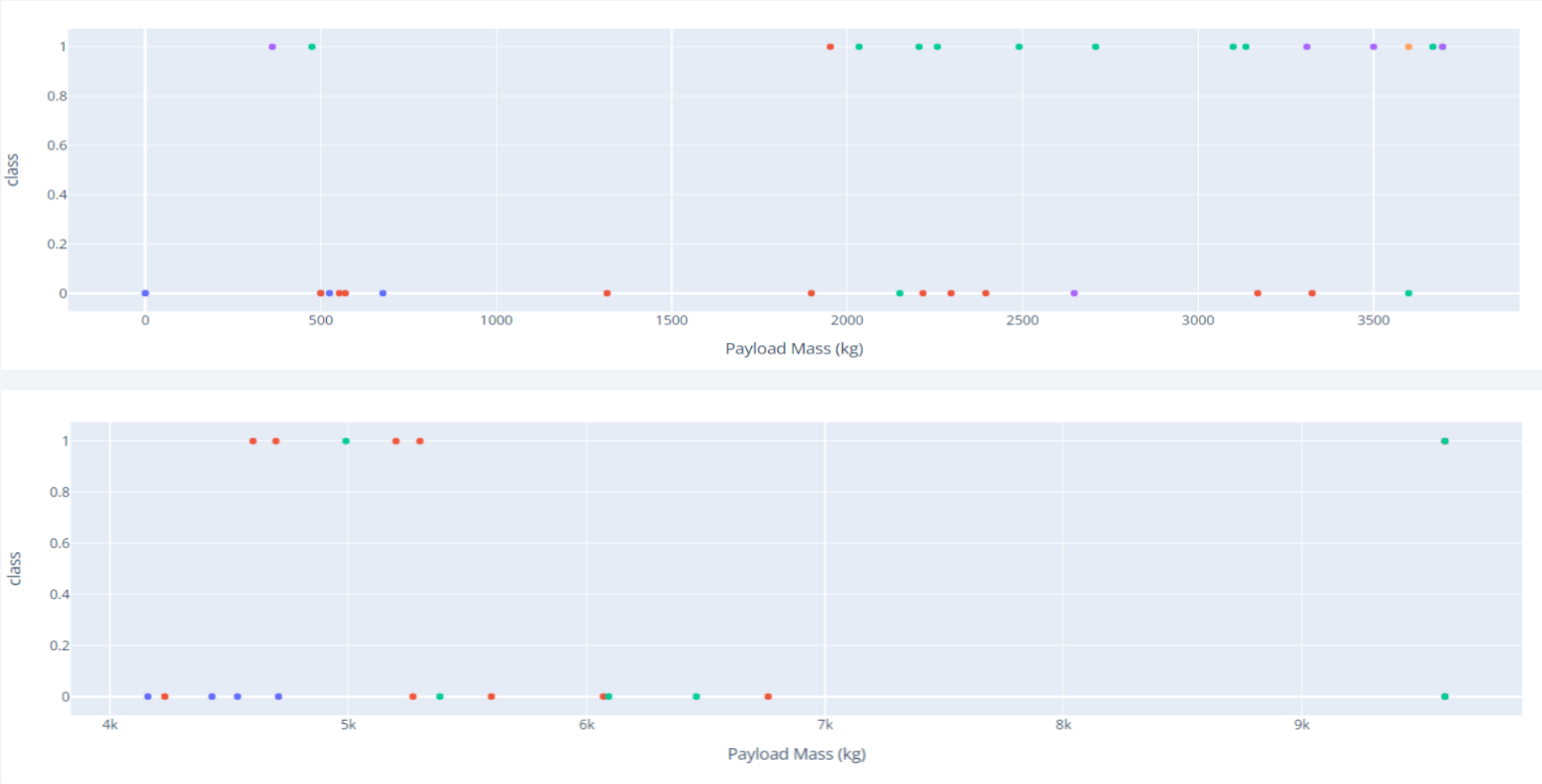


Pie chart showing the Launch site with the highest launch success ratio

- 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 payloads higher than heavily weighted payloads.



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The Logistic Regression and SVM are the models with the highest Classification Accuracy.

Scores on test data for each method

- Logistic Regression: 0.944
- SVM: 0.944
- Decision Tree: 0.888
- KNN: 0.888

Conclusion: Logistic Regression and SVM deliver the best performance on test data

Confusion Matrix

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

Conclusions

- We can conclude that:
  - The larger the flight amount at a launch site, the greater the success rate at a launch site.
  - Launch success rate started to increase from 2013 till 2020.
  - Orbits ES-L1, GEO, HEO, SSO, and VLEO had the most success rate.
  - KSC LC-39A had the most successful launches of any site.
  - The Logistic Regression classifier is the best machine learning algorithm for this task.

Thank you!