IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Alberto Pintos Martín
09/10/2022

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

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.

  Our company is competing against the enterprise SpaceX, that has a rocket model called Falcon 9 with a cost of 62 million dollars, while other companies rocket launches cost up to 165 million dollars. This is due to the reutilization of the first stage of the rocket. So, in this project, we will analyze the key factors for the success of the first stage of the rocket reutilization, in order to compete with the company SpaceX.

- Problems you want to find answers.

  - Factors that contribute to a successful landing of the rocket first stage.
  - The correlation between several characteristics that determine the success rate of a successful landing.
  - Main conditions to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

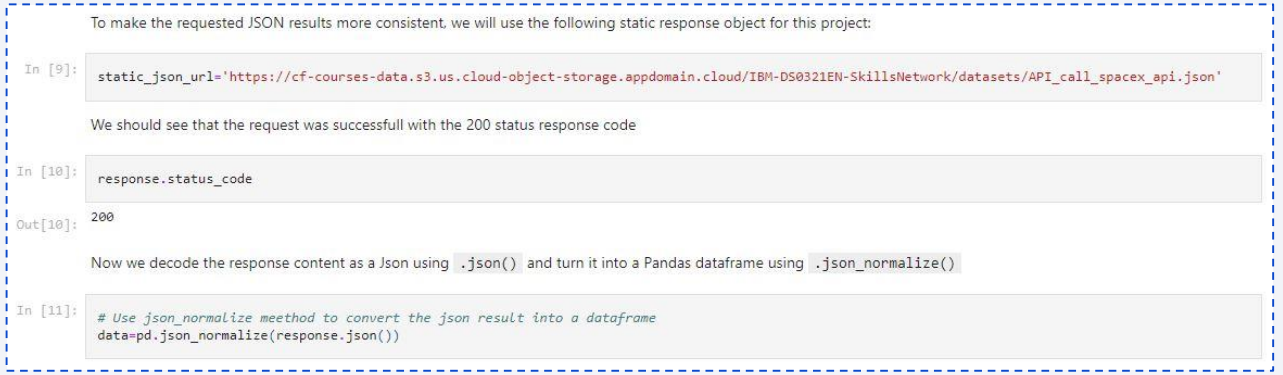
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling.
  - One-hot encoding was applied to categorical features.
- Perform exploratory data analysis (EDA) using visualization and SQL.
- Perform interactive visual analytics using Folium and Plotly Dash.
- Perform predictive analysis using classification models.
  - How to build, tune, evaluate classification models.

Data Collection

- Data was collected using two methods:
  - First, we have collected data using get request to the SpaceX API.
  - Then, we stored the data from a .json file into a pandas dataframe through the method .json_normalize().
  - In addition, we cleaned the data and check for missing values.
  - On the other hand, we have also gathered data using web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - This time, we extracted the launch records as HTML table, parse the table and convert it to a pandas dataframe.

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.
- The link to the notebook is https://github.com/APintos2/test repo/blob/master/Data%20Collection%20API%20lab.ipynb



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'
```

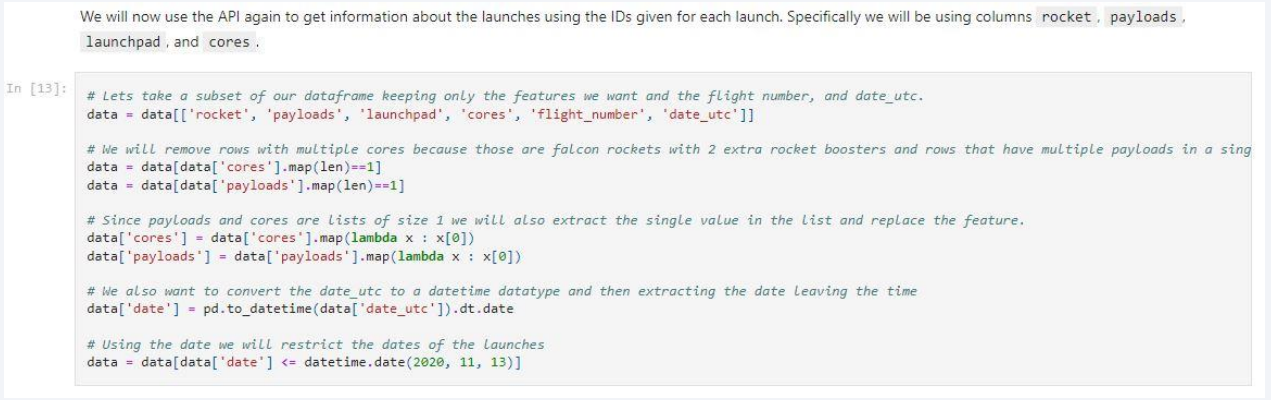
We should see that the request was successfull with the 200 status response code

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
data=pd.json_normalize(response.json())
```

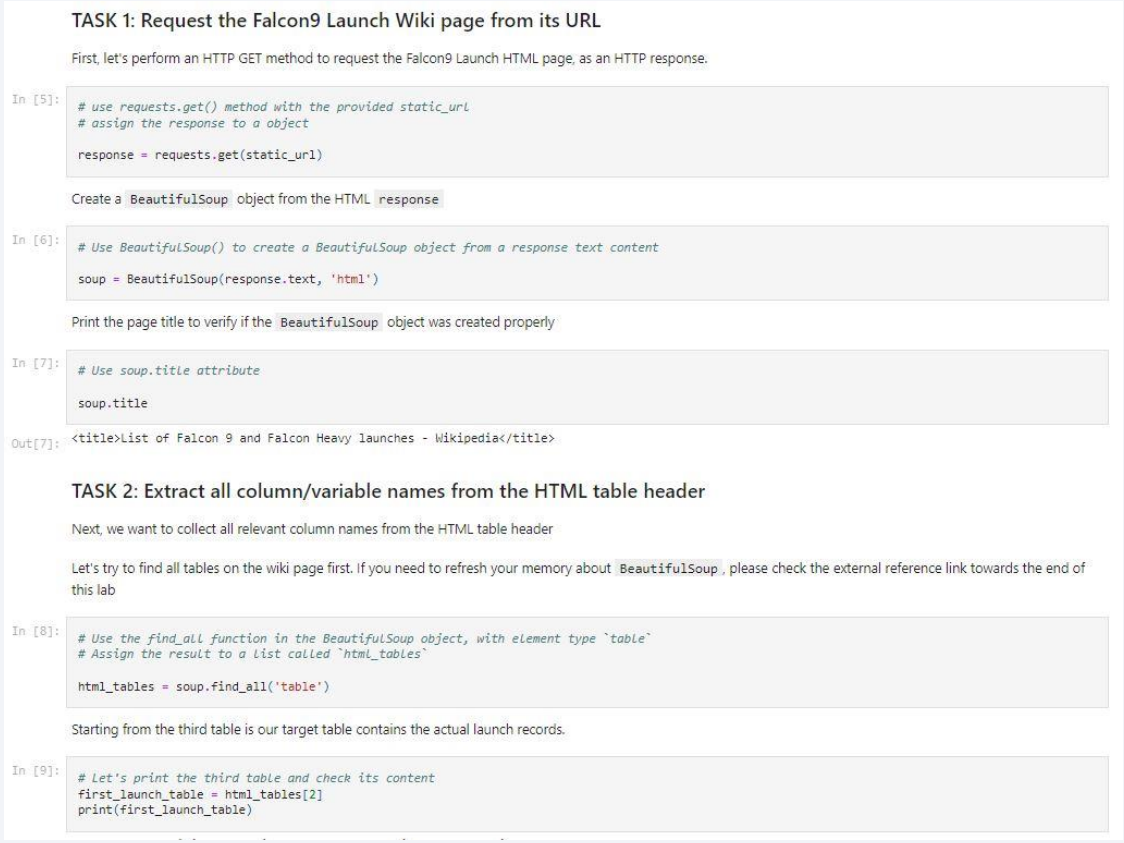


We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

In [13]:
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a sing
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

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/APintos2/testrepo/blob/master/Data%20Collection%20with%20Web%20Scraping%20lab.ipynb



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

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

response = requests.get(static_url)
```

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

soup = BeautifulSoup(response.text, 'html')
```

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

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

soup.title
```

Out[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

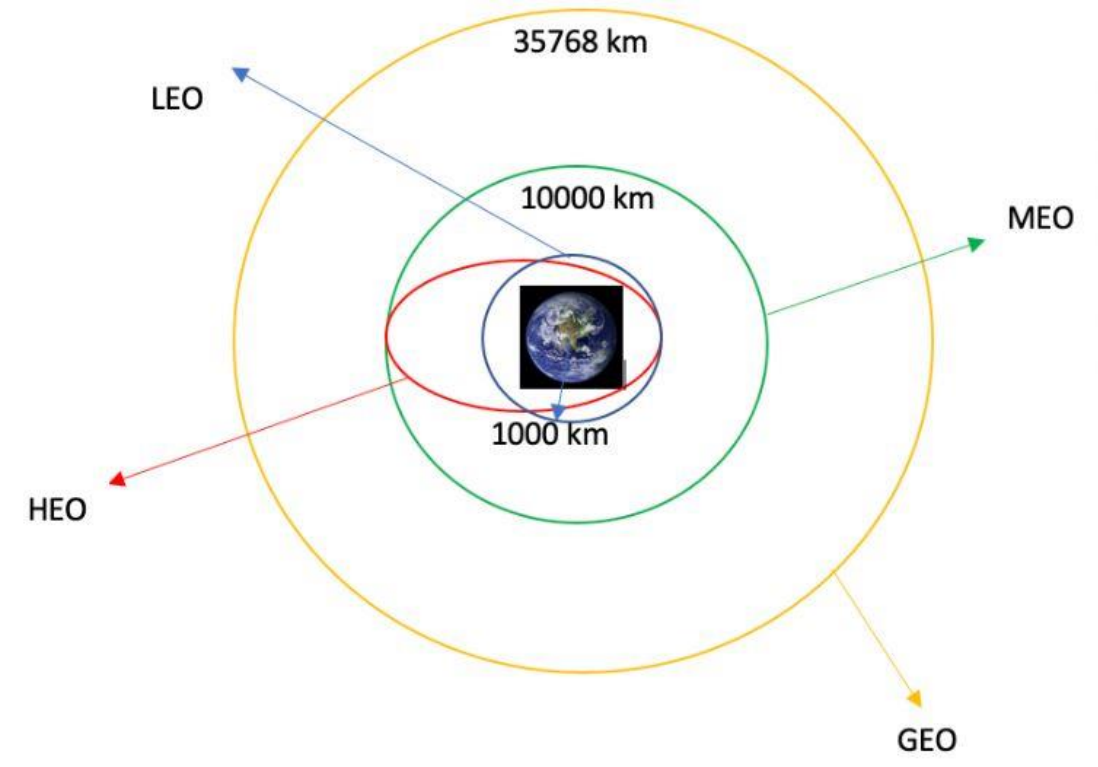
In [8]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`

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)
```

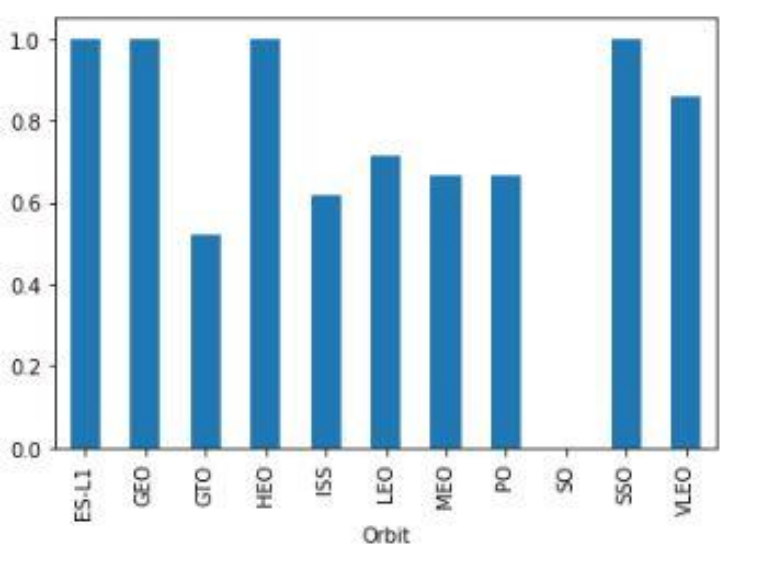
Data Wrangling

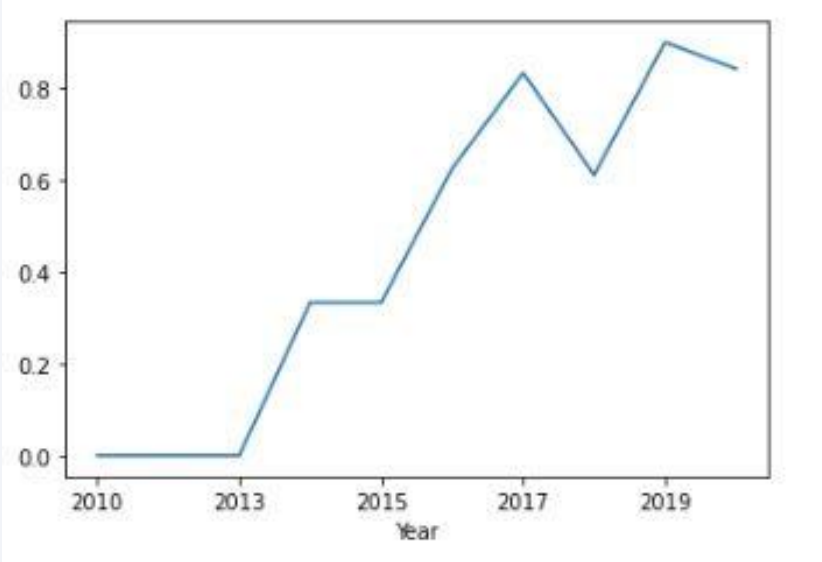
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/APintos2/testrepo/blob/master/Data%20Wrangling.ipynb



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.
- The link to the notebook is https://github.com/APintos2/testrepo/blob/master/EDA%20with%20Visualization%20Lab.ipynb





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 link to the notebook is https://github.com/APintos2/testrepo/blob/master/EDA%20with%20SQL%20lab.ipynb

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.
- Link to the notebook is https://github.com/APintos2/testrepo/blob/master/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

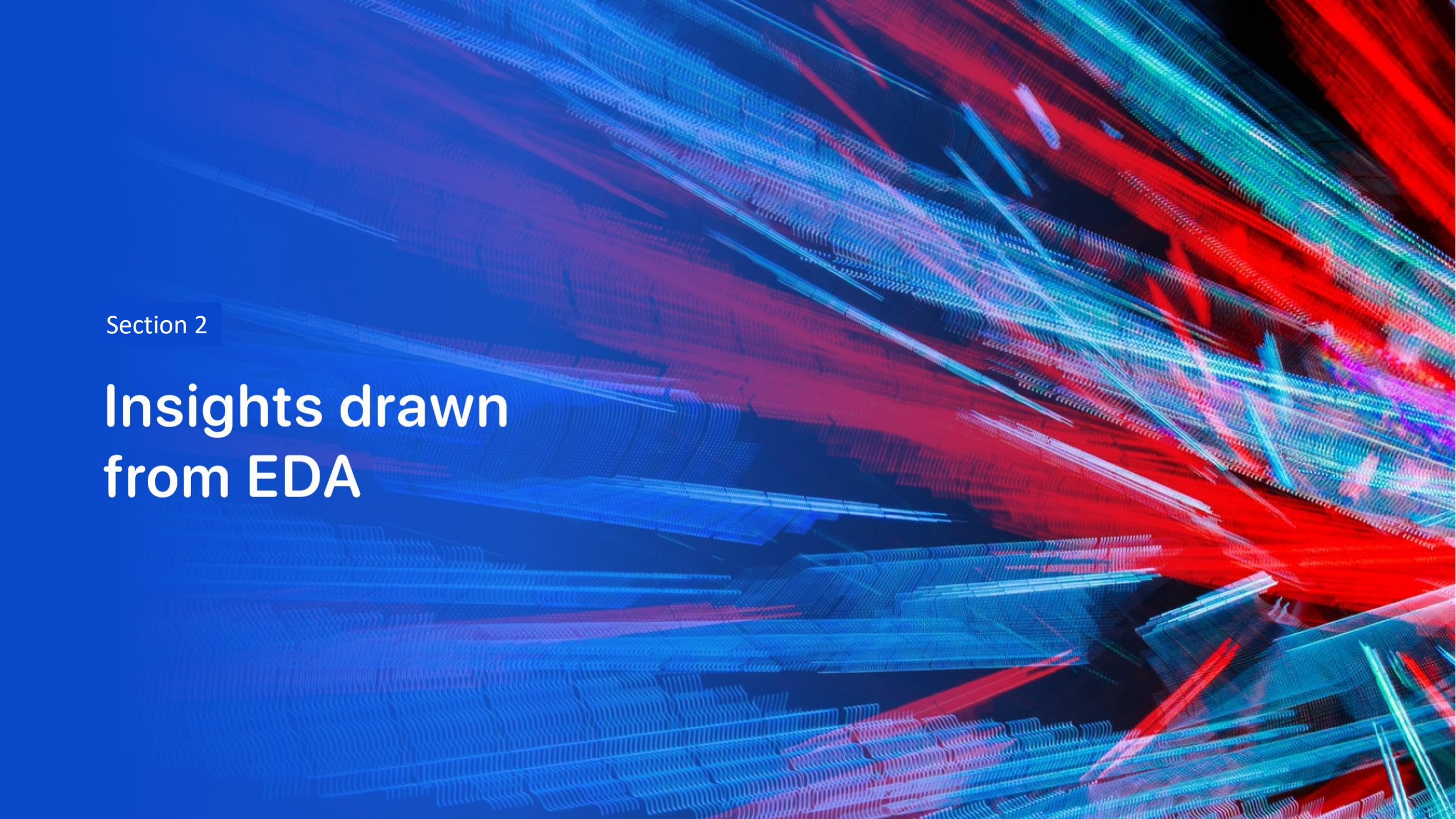
- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/APintos2/testrepo/blob/master/spacex_dash_app.py

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 https://github.com/APintos2/testrepo/blob/master/SpaceX_Machine%20Learning%20Prediction_Part_5%20(1).ipynb

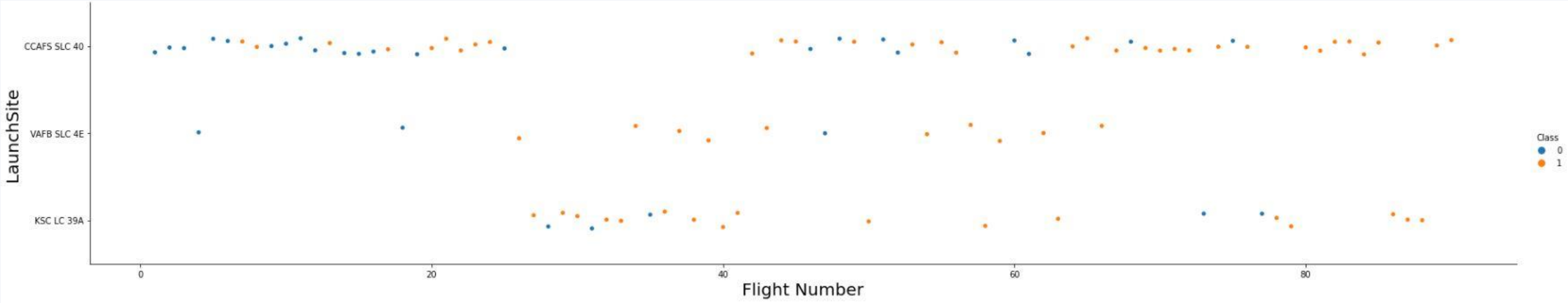
Results

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


Section 2
Insights drawn from EDA

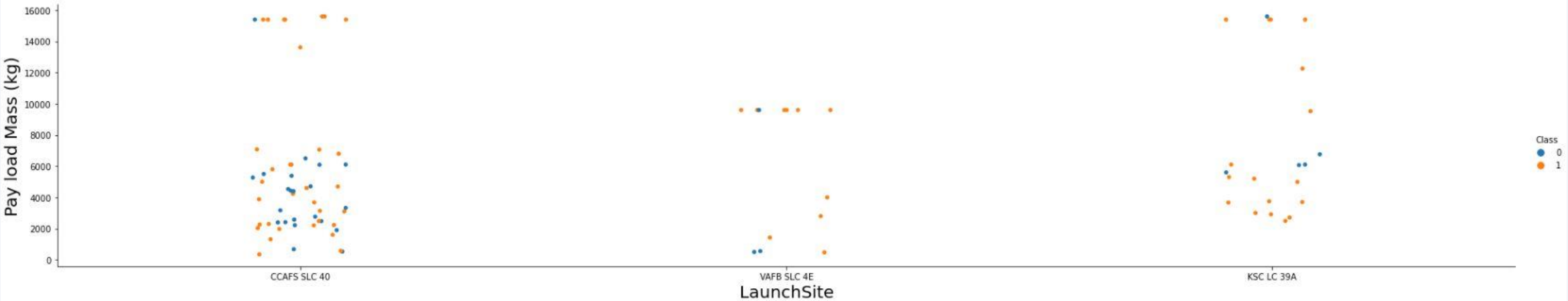
Flight Number vs. Launch Site

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



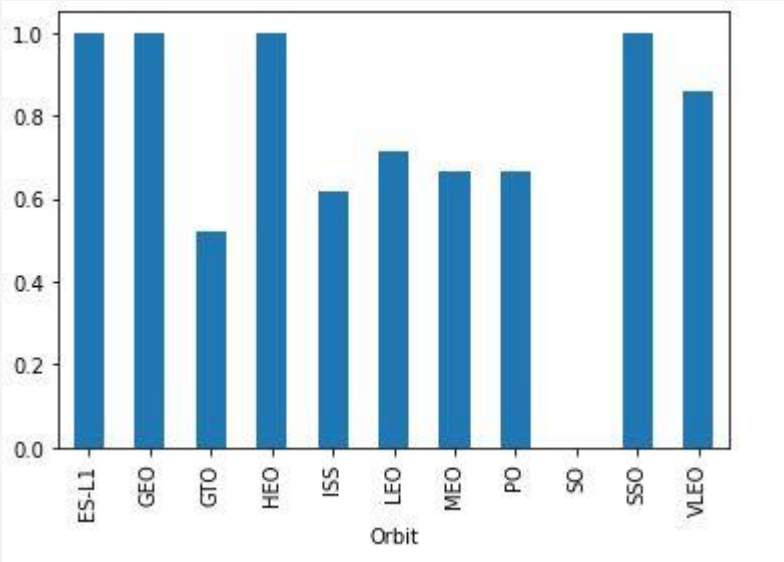
Payload vs. Launch Site

- We can see that the greater the payload mass for the CCAFS SLC 40, the greater is the success rate.



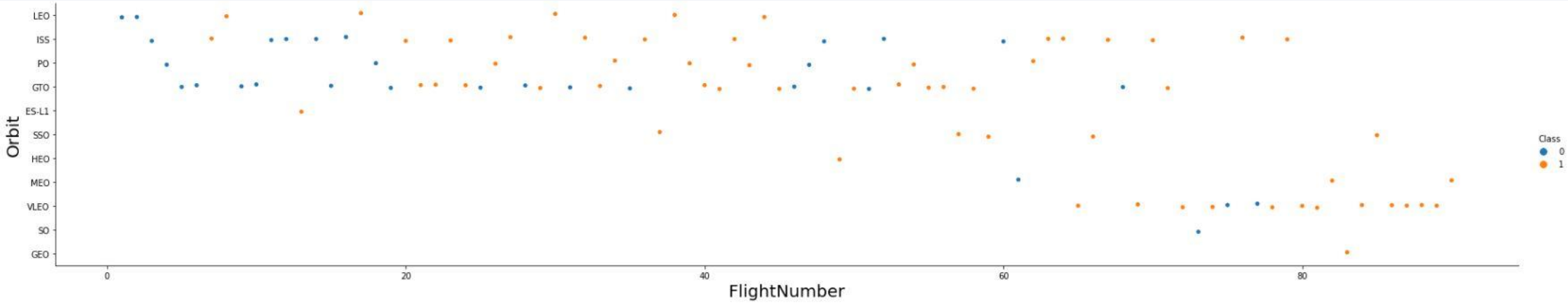
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had a success rate of a 100%.



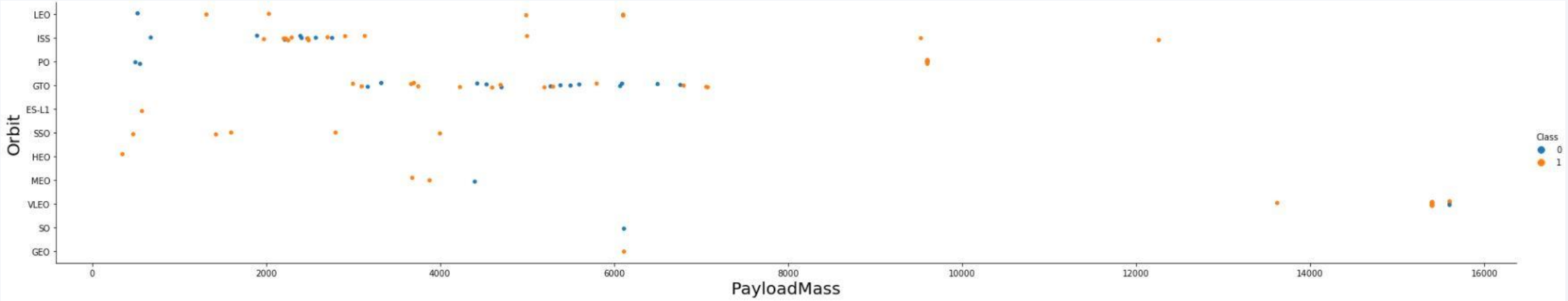
Flight Number vs. Orbit Type

- 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.



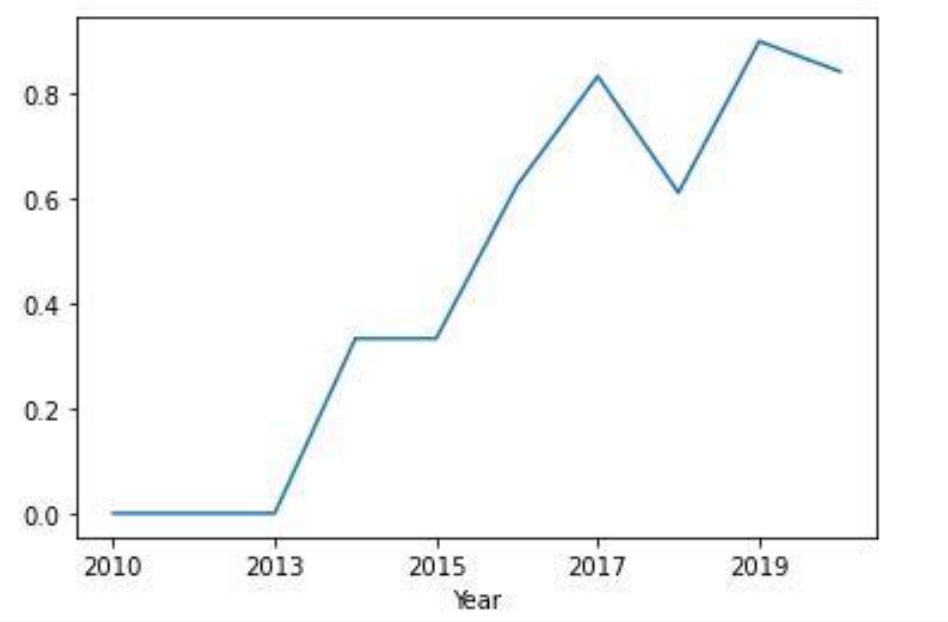
Payload vs. Orbit Type

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



Launch Success Yearly Trend

- In a scale of success from 0 to 1, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- I used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

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

```
[16]: sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

[16]:

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

Launch Site Names Begin with 'CCA'

- We used the query below to display 5 records where launch sites begin with ‘CCA’:

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

```
[17]: sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

[17]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
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
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
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 calculated the total payload carried by boosters from NASA using the query below:

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

```
[18]: sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD FROM SPACEXTBL WHERE PAYLOAD LIKE '%CRS%';
```

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

[18]:

TOTAL_PAYLOAD
111268

Average Payload Mass by F9 v1.1

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

Task 4

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

```
[19]: sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
 * sqlite:///my_data1.db
Done.
```

[19]:

AVG_PAYLOAD
2928.4

First Successful Ground Landing Date

- I couldn’t find results related to Landing Outcome because the column couldn’t be found. Anyway I think the code to do so would be:

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

Hint:Use min function

[73]:
```
sql SELECT MIN(DATE) AS FIRST_SUCCESS_GP FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING_OUTCOME
[SQL: SELECT MIN(DATE) AS FIRST_SUCCESS_GP FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- I couldn’t find results related to Landing Outcome because the column name couldn’t be found. Anyway I think the code to do so would be:

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

```
[25]: sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND LANDING_OUTCOME = 'Success (drone ship)';
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING_OUTCOME
[SQL: SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND LANDING_OUTCOME = 'Success (drone ship)';]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Total Number of Successful and Failure Mission Outcomes

- We used Count() to get the number of failures and successes in MissionOutcome.

List the total number of successful and failure mission outcomes

```
[28]: sql SELECT MISSION_OUTCOME, COUNT(*) AS QTY FROM SPACEXTBL GROUP BY MISSION_OUTCOME ORDER BY MISSION_OUTCOME;
```

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

[28]:

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

Boosters Carried Maximum Payload

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

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

```
[29]: sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL) ORDER BY BOOSTER_VERSION;
 * sqlite:///my_data1.db
Done.
```

[29]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

- I couldn’t find results related to Landing Outcome because the column couldn’t be found. Anyway I think the code to do so would be:

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.

[33]:
```
sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Failure (drone ship)' AND DATE_PART('YEAR', DATE) = 2015;
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING_OUTCOME
[SQL: SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Failure (drone ship)' AND DATE_PART('YEAR', DATE) = 2015;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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

- I couldn’t find results related to Landing Outcome because the column couldn’t be found. Anyway I think the code to do so would be:

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

```
[42]: sql SELECT LANDING_OUTCOME, COUNT(*) AS QTY FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY QTY DESC;
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING_OUTCOME
[SQL: SELECT LANDING_OUTCOME, COUNT(*) AS QTY FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY QTY DESC;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Section 3

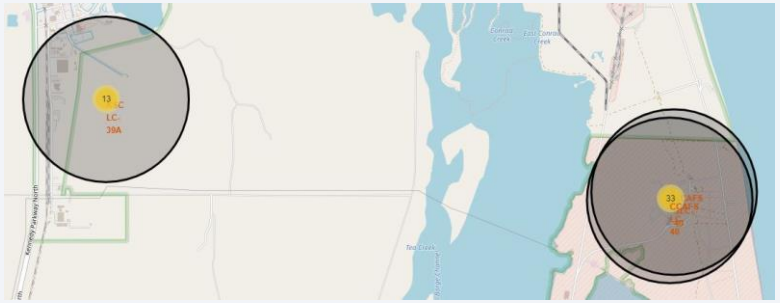
Launch Sites Proximities Analysis

All launch sites global map markers

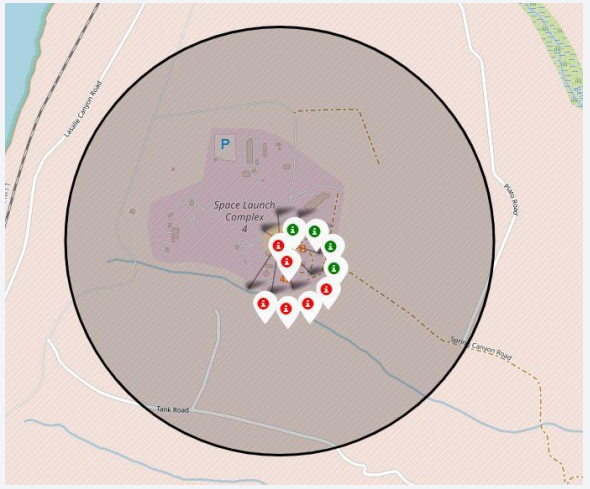
- We see that SpaceX launches are located in the USA, in Florida and California.

Markers showing launch sites with color labels

- Green show successful launches.
- Red show unsuccessful launches.



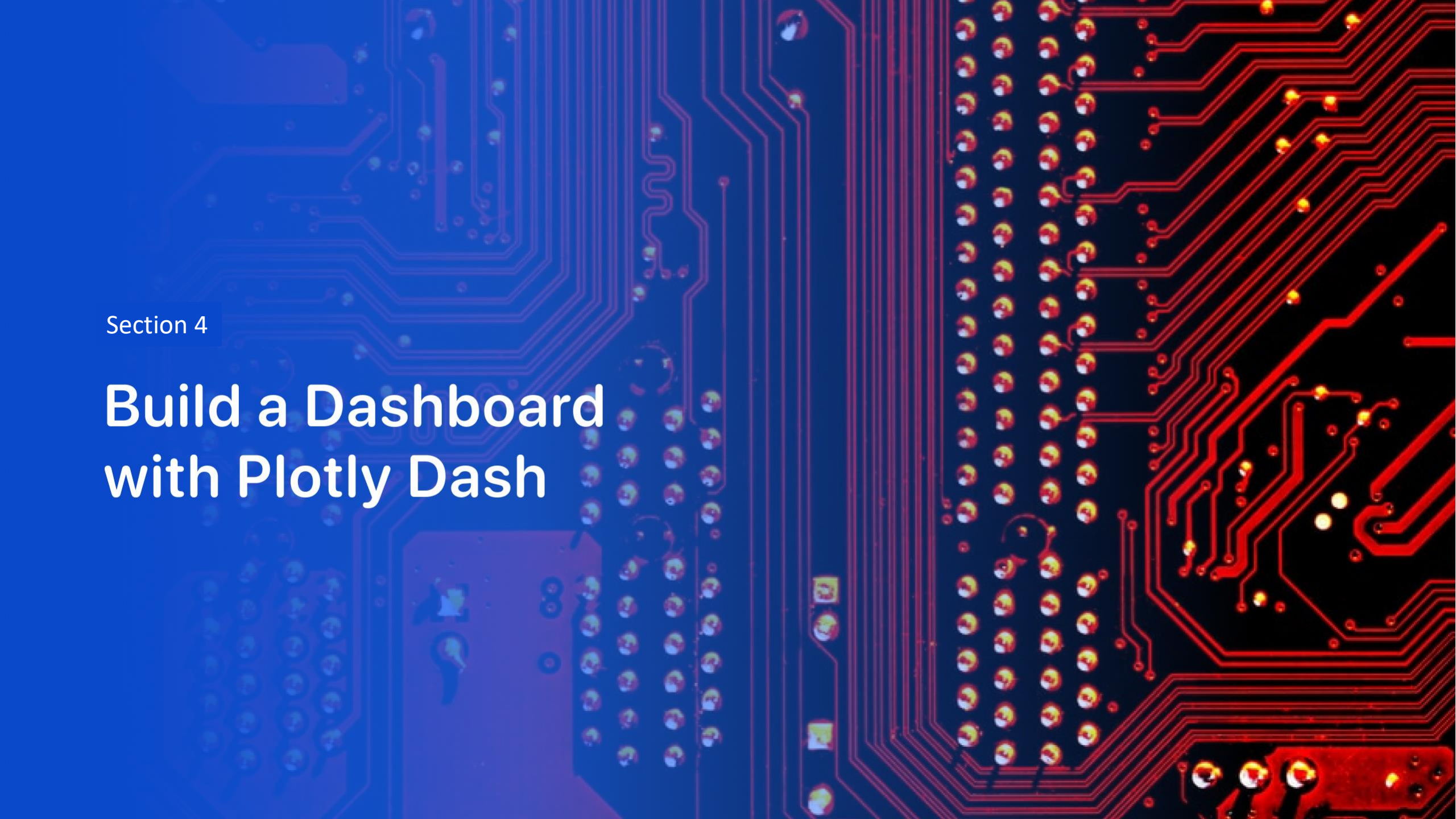
- Florida Launch Sites.



- California Launch Site.

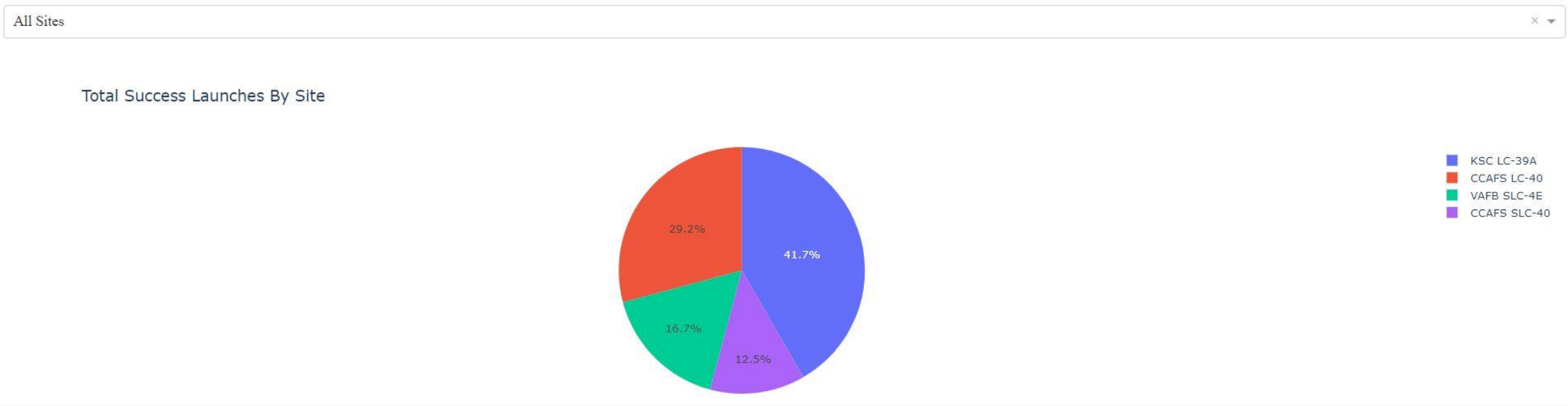
Launch Site distance to landmarks

- Replace <Folium map screenshot 3> title with an appropriate title
- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed
- Explain the important elements and findings on the screenshot


Section 4
Build a Dashboard with Plotly Dash

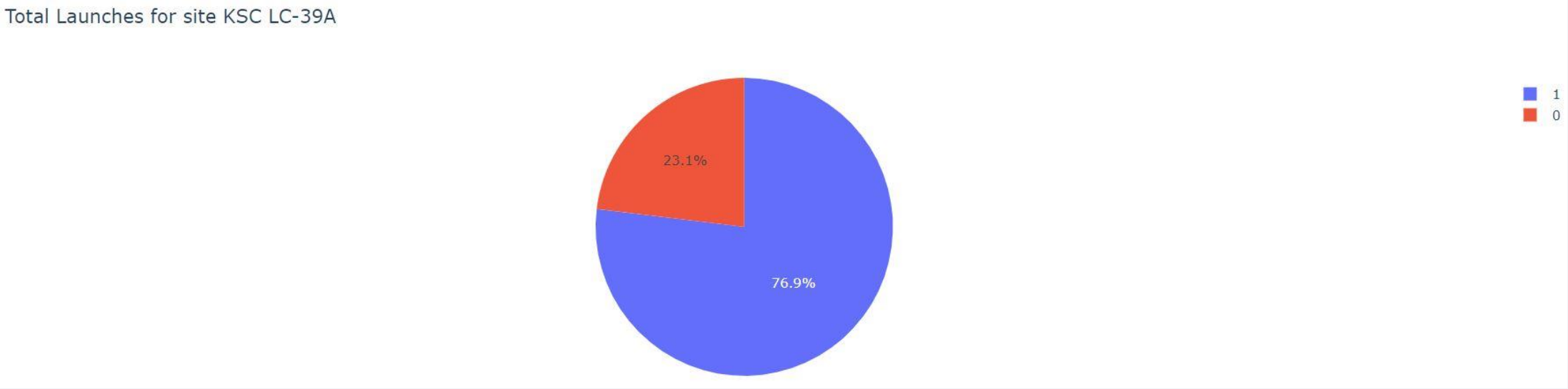
Pie chart showing the success percentage achieved by each launch site

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



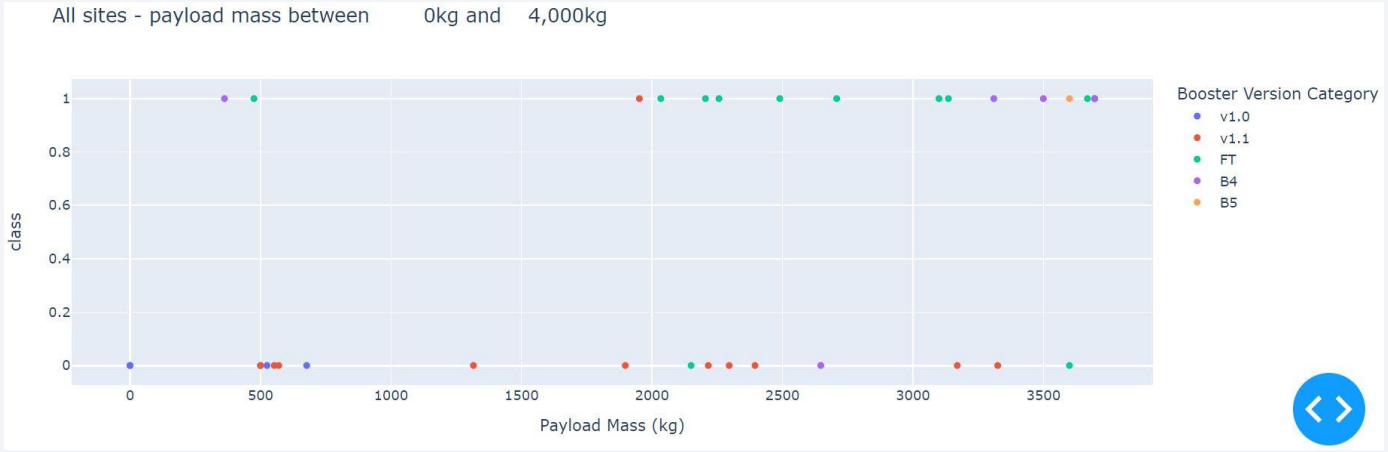
Pie chart showing the Launch site with the highest launch success ratio

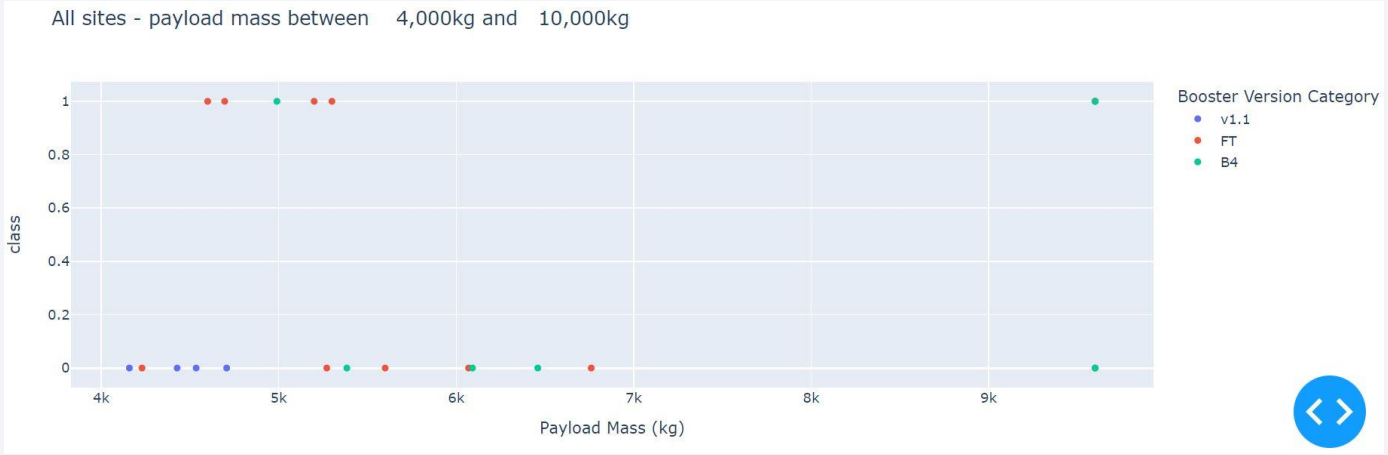
- We see that KSC LC-39A achieved a 76.9% success rate and a 23.1% failure rate:



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

- The success rate is higher for low weighted Payloads:



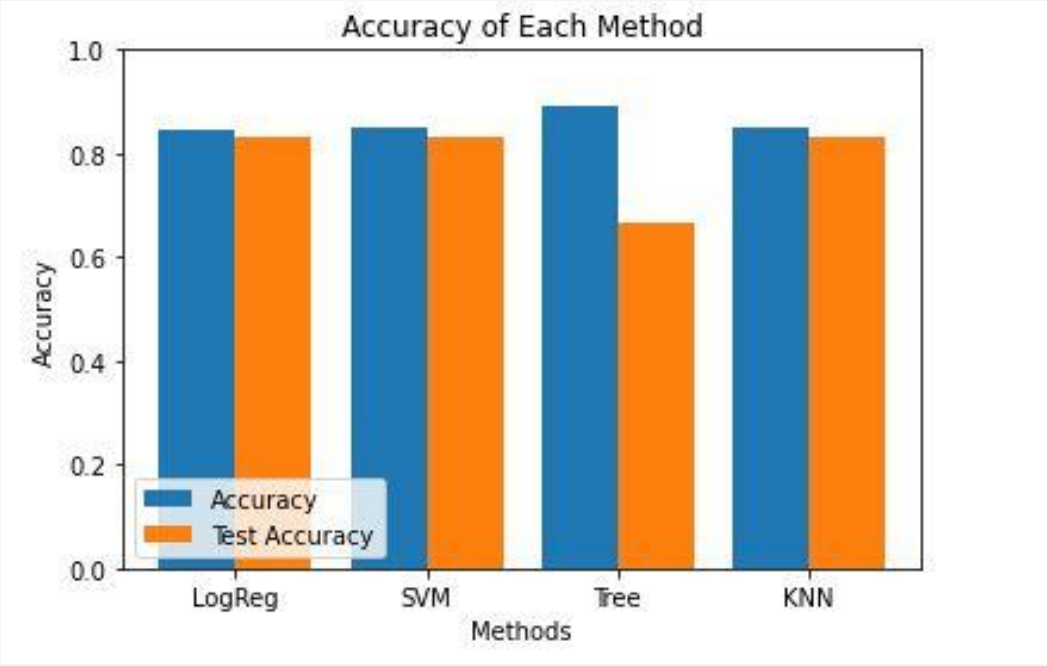


Section 5

Predictive Analysis (Classification)

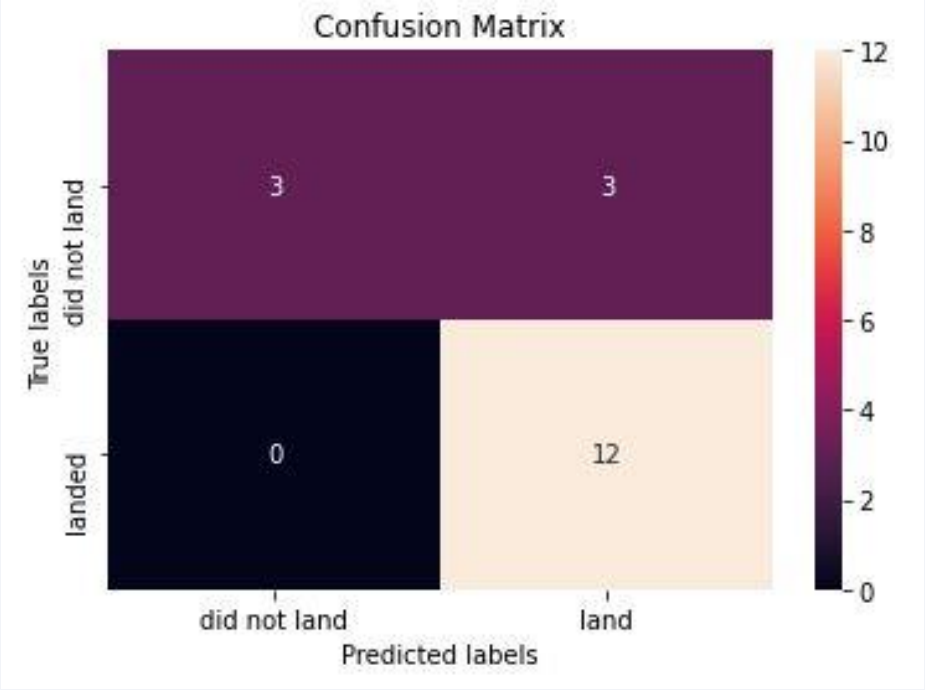
Classification Accuracy

- We can see in the bar chart that the method with the highest accuracy is the Tree method.



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

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 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.

Thank you!