IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nguyễn Thanh Tùng
Thursday, May 4, 2023

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context
- Problems you want to find answers

Section 1

Methodology

Methodology

Executive Summary

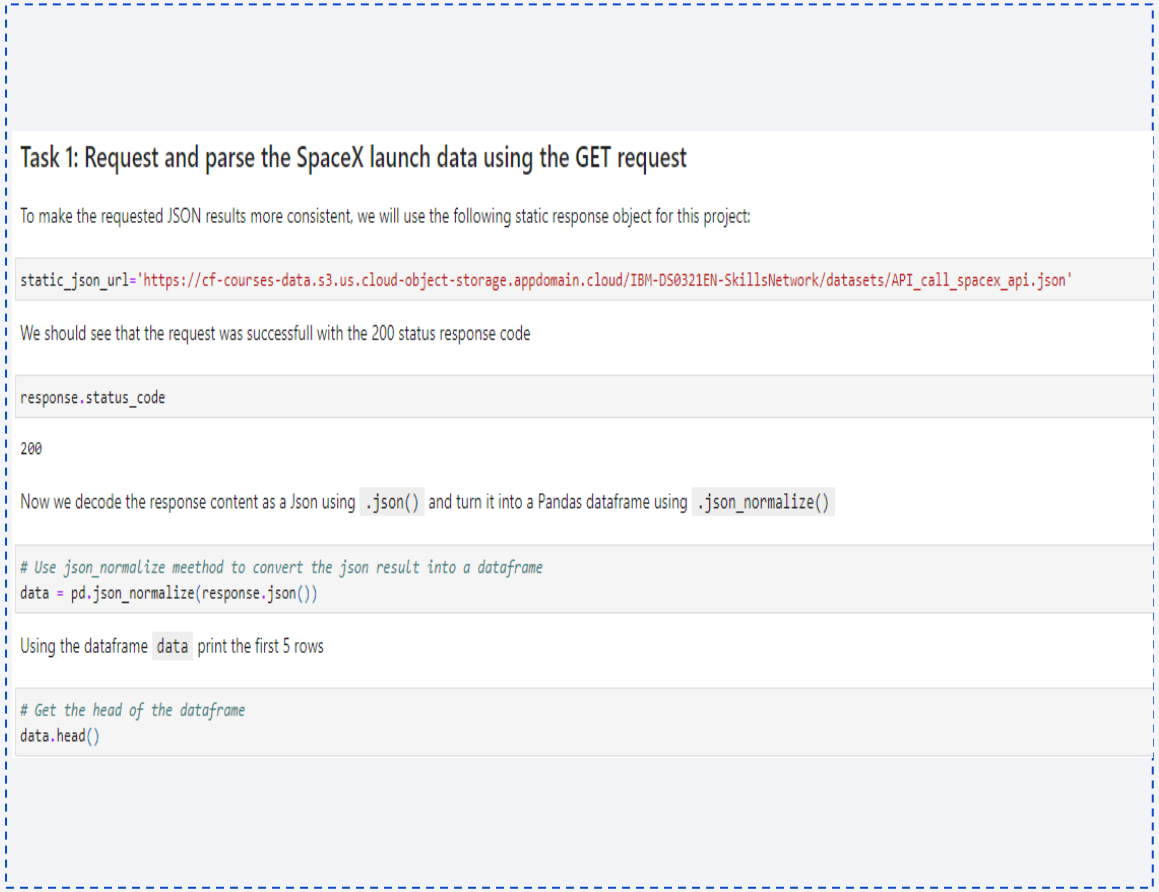
- Data collection methodology:
  - The libraries used are requests, pandas, numpy, and datetime. Helper functions are defined to extract information from the SpaceX API, including booster name, launch site name and coordinates, payload mass and orbit, spacecraft information, and landing outcome. The data is collected from the SpaceX API.
- 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.
- You need to present your data collection process use key phrases and flowcharts

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/Tunggbe/SpaceY/blob/main/SpaceY/Week%201/Collecting%20the%20data.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:

```
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

```
response.status_code
```

```
200
```

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

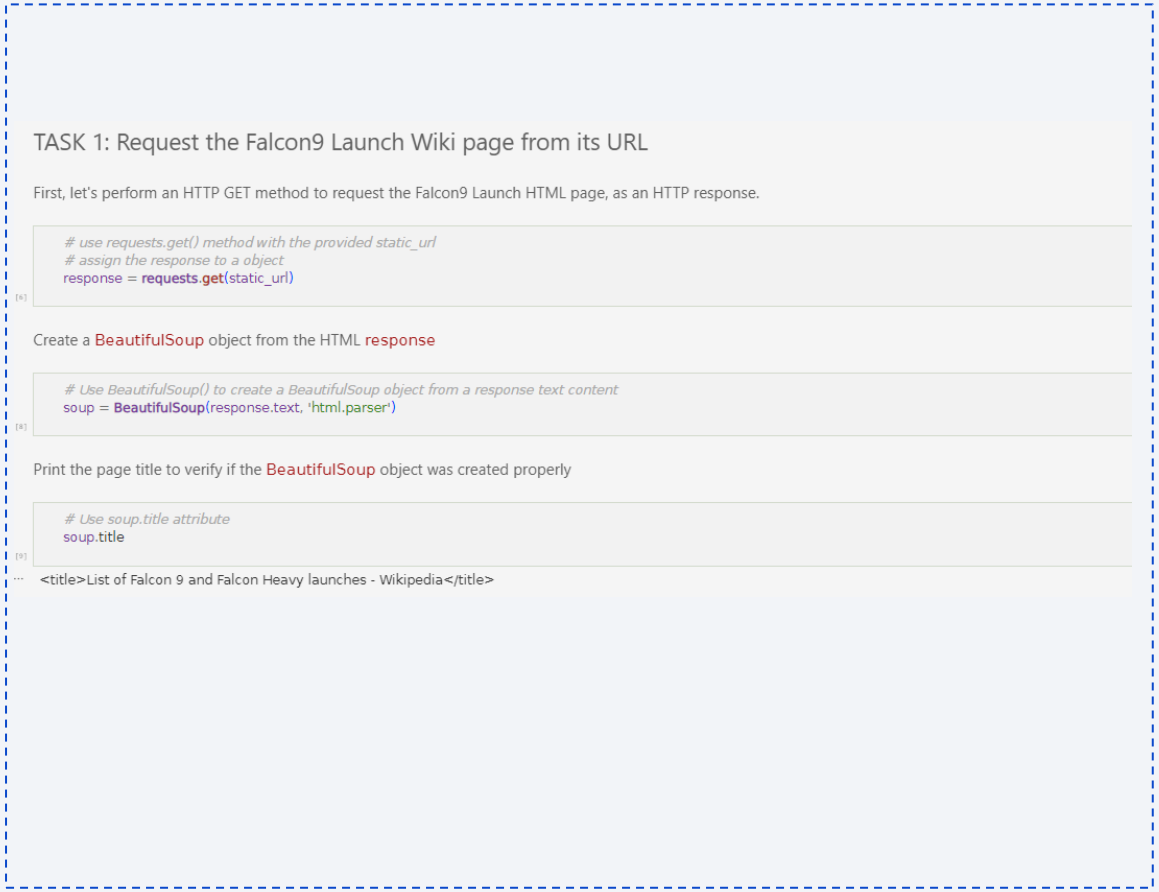
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

Data Collection - Scraping

- We applied web scrapping to webscraping Falcon 9 launch records with BeautifulShoup
- https://github.com/Tunggbe/SpaceY/blob/main/SpaceY/Week%201/Collecting%20the%20data.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.

```
# 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

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

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

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data Wrangling

- We performed exploratory data analysis and determined the training labels
- We calculated the number of launches at each site, and number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv
- https://github.com/Tunggbe/SpaceY/blob/main/SpaceY/Week%201/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
- https://github.com/Tunggbe/SpaceY/blob/main/SpaceY/Week%202/the%20EDA%20with%20Visualization%20lab.ipynb

EDA with SQL

- Selected all columns from the "SPACEXTBL" table.
- Selected distinct values in the "LAUNCH_SITE" column from the "SPACEXTBL" table.
- Selected the "LAUNCH_SITE" column from the "SPACEXTBL" table where the "Launch_Site" column contains the string "CCA".
- Calculated the sum of the "PAYLOAD_MASS__KG_" column in the "SPACEXTBL" table.
- Calculated the average of the "PAYLOAD_MASS__KG_" column in the "SPACEXTBL" table where the "Booster_Version" column contains the string "F9 v1.1".
- Selected the "BOOSTER_VERSION" column from the "SPACEXTBL" table where the "Landing Outcome" column contains the string "Success (drone ship)" and the "PAYLOAD_MASS__KG" column is between 4000 and 6000.
- Counted the number of occurrences of each distinct value in the "Mission_Outcome" column from the "SPACEXTBL" table.
- Selected the "Payload" and "PAYLOAD_MASS__KG_" columns from the "SPACEXTBL" table where the "PAYLOAD_MASS__KG_" column has the maximum value.
- Selected the "Booster_Version" and "Launch_site" columns from the "SPACEXTBL" table where the "Landing _Outcome" column contains the string "Failure (drone ship)" and the year in the "Date" column is 2015.
- Selected the "Date", "Booster_Version", and "Landing _Outcome" columns from the "SP”
- https://github.com/Tunggbe/SpaceY/blob/main/SpaceY/Week%202/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We marked all launch sites and added map objects such as markers, circles, and lines to indicate the success or failure of launches for each site on the Folium map.
- We assigned the launch outcomes (failure or success) to class 0 and 1, i.e., 0 for failure and 1 for success.
- Using color-labeled marker clusters, we identified which launch sites have a relatively high success rate.
- We also calculated the distances between launch sites and their proximities, answering questions such as:
  - Are launch sites near railways, highways, and coastlines?
  - Do launch sites keep a certain distance away from cities?
- https://github.com/Tunggbe/SpaceY/blob/main/SpaceY/Week%203/spacex_dash_app.py

Build a Dashboard with Plotly Dash

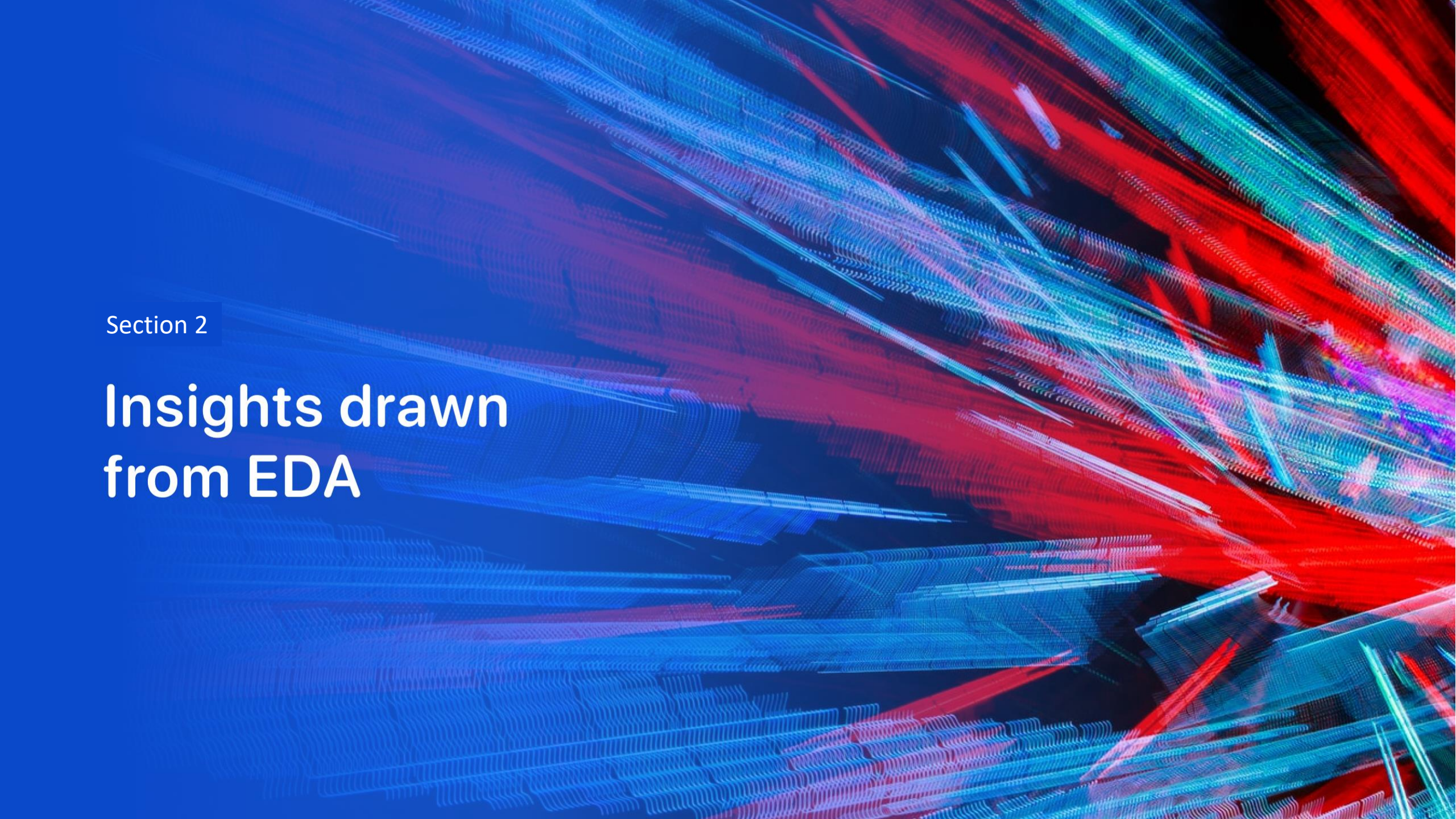
- We bullt an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and PayloadMass (Kg) for the different booster version.
- https://github.com/Tunggbe/SpaceY/blob/main/SpaceY/Week%203/spacex_dash_app.py

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.
- https://github.com/Tunggbe/SpaceY/blob/main/SpaceY/Week%204/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

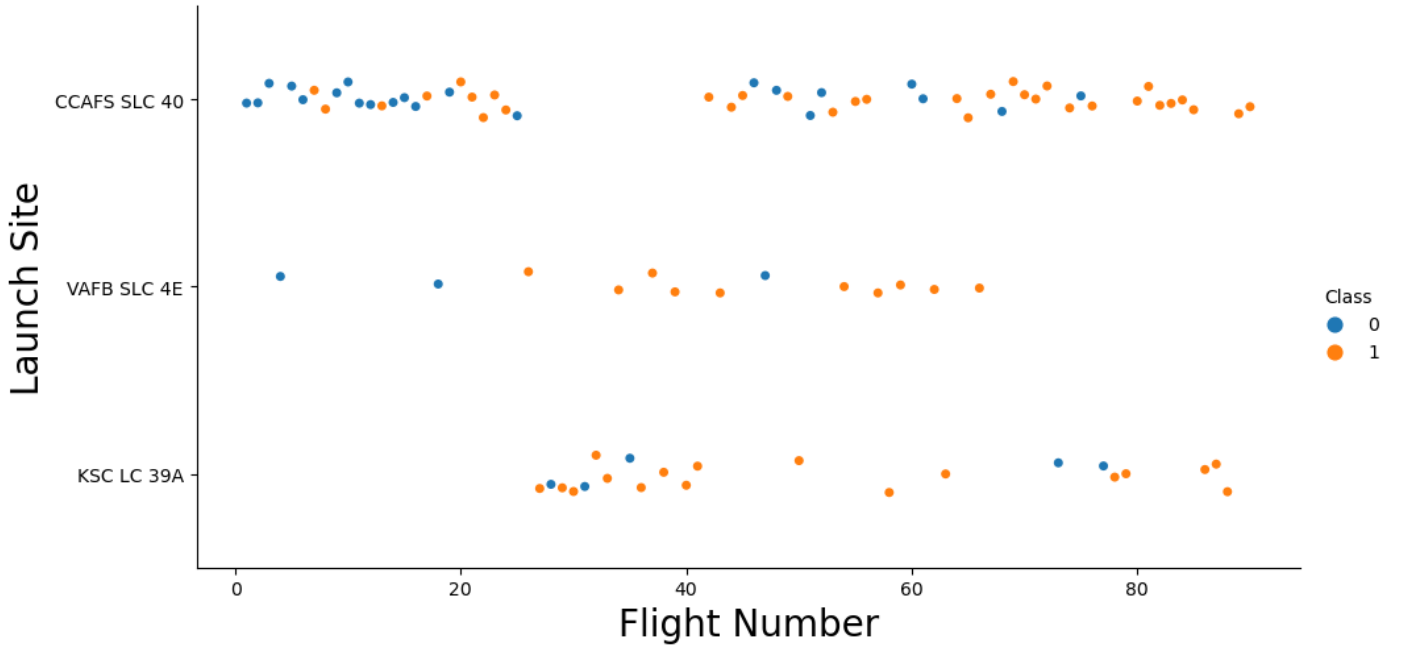
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


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.

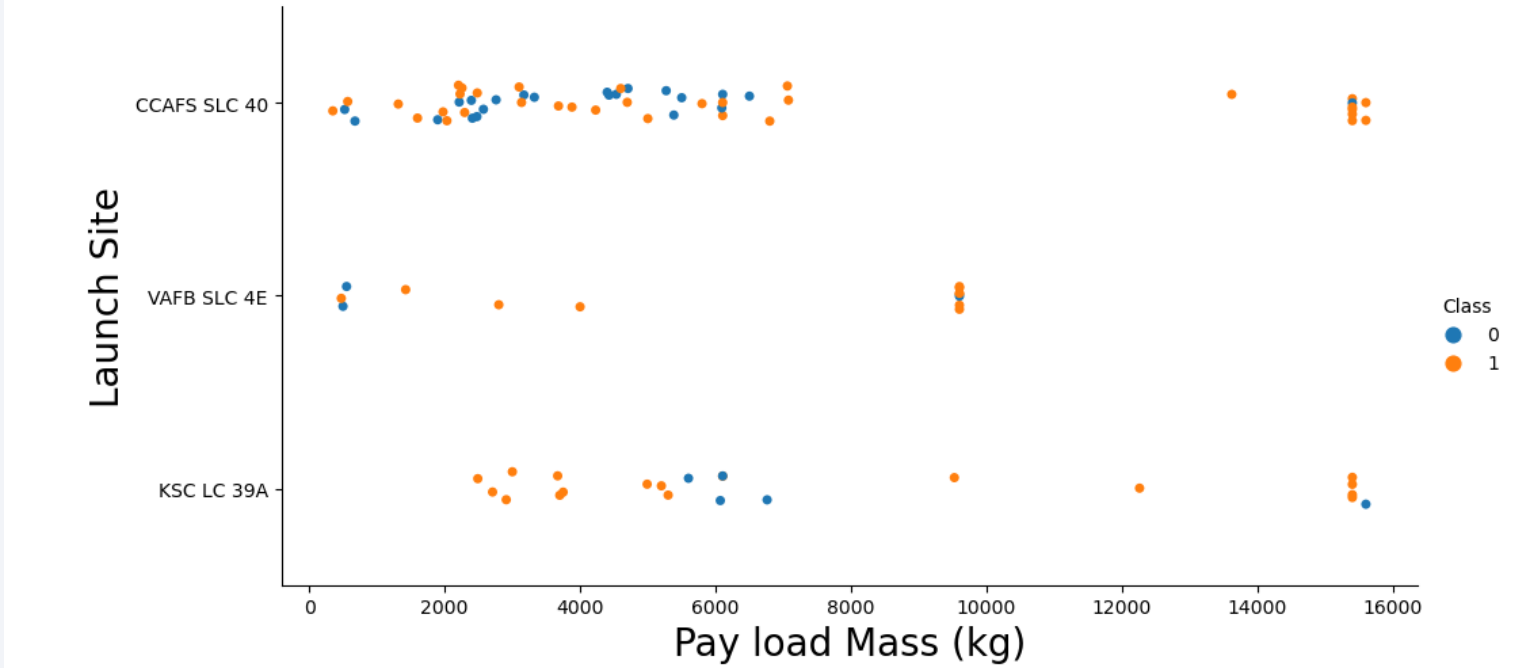
In [6]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(x="FlightNumber", y="LaunchSite", hue="Class", data=df, aspect = 2)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Payload vs. Launch Site

- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.

In [7]:
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df, aspect = 2)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

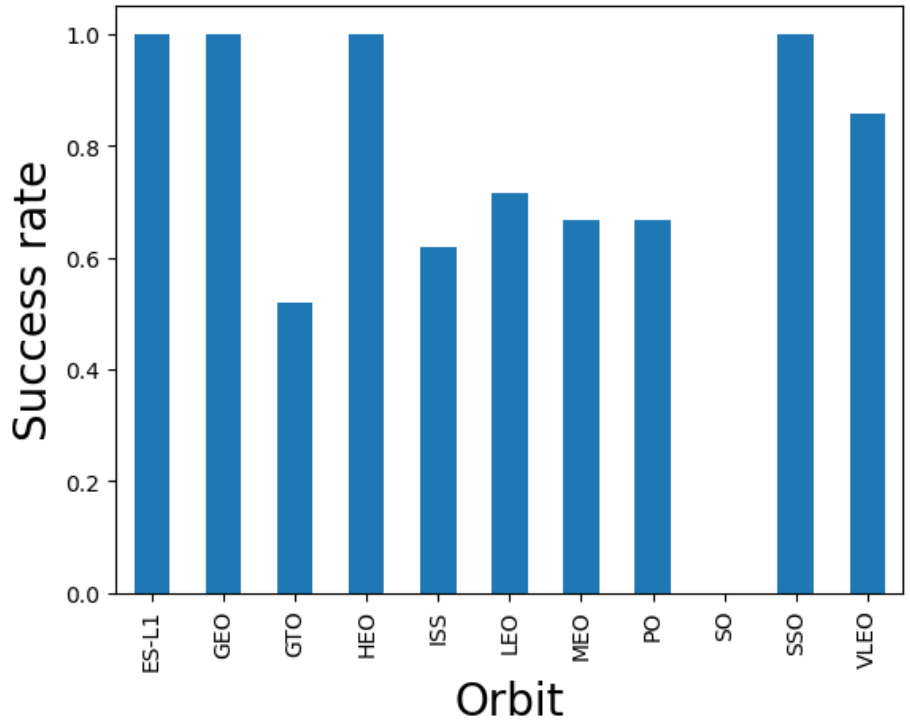


Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, and SSO had the highest success rates, while SO had the lowest.

In [13]:

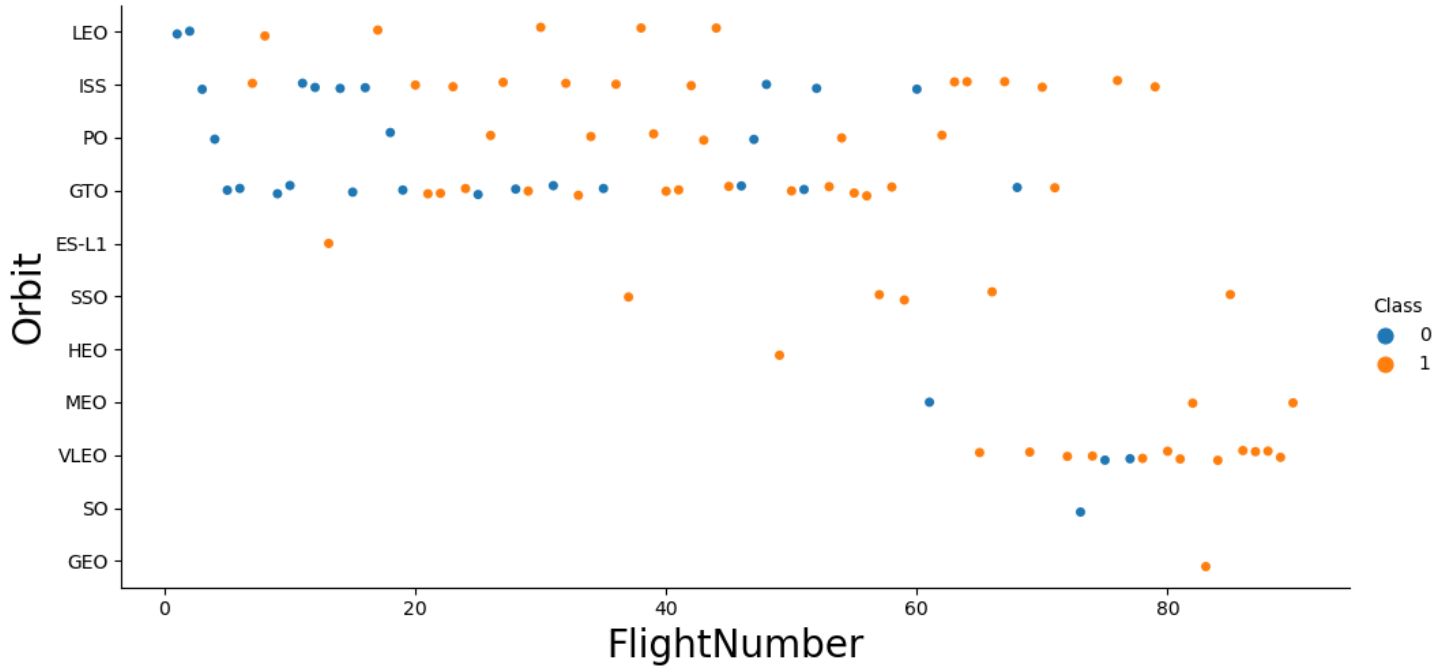
```
# HINT use groupby method on Orbit column and get the mean of Class column
df.groupby(["Orbit"]).mean()["Class"].plot(kind='bar')
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Success rate",fontsize=20)
plt.show()
```



Flight Number vs. Orbit Type

- The plot below displays the relationship between Flight Number and Orbit type. We can see that for LEO orbits, the success rate appears to increase with the number of flights, while for GTO orbits, there seems to be no correlation between the number of flights and success rate.

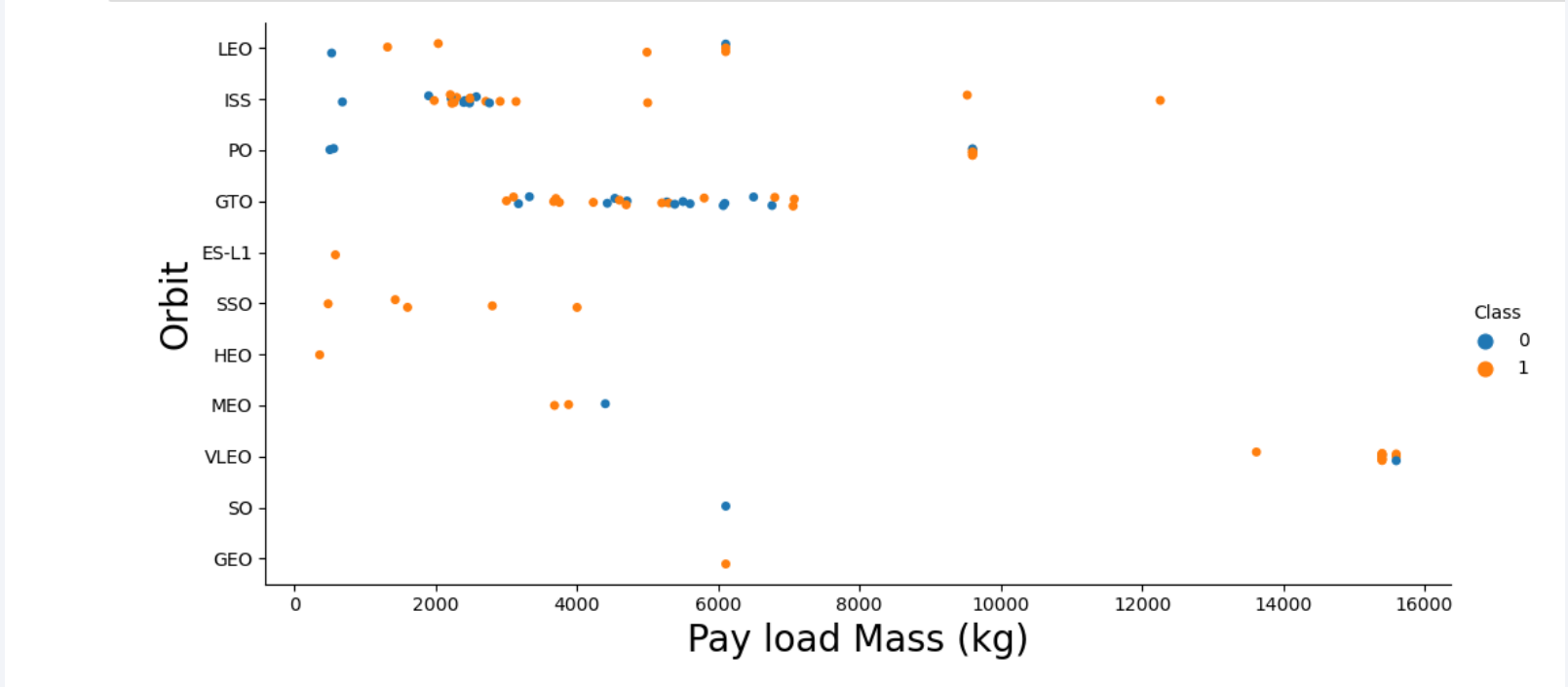
In [14]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="FlightNumber", y="Orbit", hue="Class", data=df, aspect = 2)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Payload vs. Orbit Type

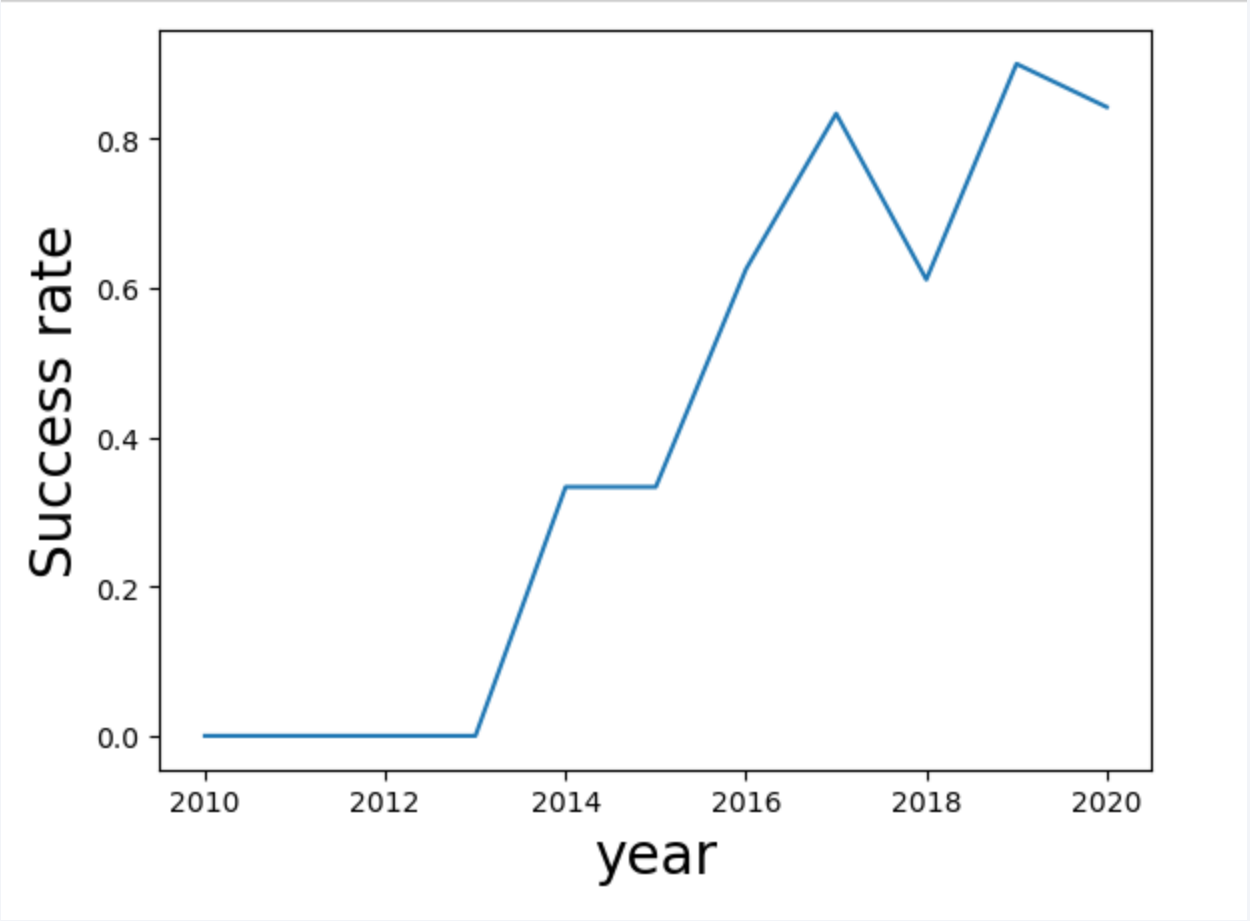
- The following plot displays the relationship between Flight Number and Orbit Type. It can be observed that in the LEO orbit, the success rate is positively correlated with the number of flights, while in the GTO orbit, there is no apparent relationship between the two variables.

```
In [14]: # Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="PayloadMass", y="Orbit", hue="Class", data=df, aspect = 2)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

- We can observe that for heavy payloads, successful landings are more likely to occur in the PO, LEO, and ISS orbits.



All Launch Site Names

- We used the key word DISTINCT to show only unique launch sites from the SpaceX

In [7]:
```
%sql select DISTINCT Launch_Site from SPACEXTBL
```

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

Out[7]:

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

Launch Site Names Begin with 'CCA'

- We utilized the query above to show 5 records with launch sites starting with 'CCA'.

```
[9]: %sql select * from SPACEXTBL where "Launch_Site" like '%CCA%' limit 5
```

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

[9]:

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 as 45596using the query below

```
[11]: %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer like 'NASA (CRS)'
 * sqlite:///my_data1.db
Done.
[11]: sum(PAYLOAD_MASS__KG_)
                       45596
```

Average Payload Mass by F9 v1.1

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

```
[11]: %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1'
 * sqlite:///my_data1.db
Done.
```

[11]:

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

```
[41]: %sql UPDATE SPACEXTBL SET Date = substr(Date, 7, 4) || '-' || substr(Date, 4, 2) || '-' || substr(Date, 1, 2);

 * sqlite:///my_data1.db
101 rows affected.
[41]: []
```

```
[46]: %sql select min(Date), "Landing _Outcome"  from SPACEXTBL where "Landing _Outcome" like '%Success%';

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

[46]:

min(Date)	Landing _Outcome
2015-12-22	Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

```
[25]: %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.
```

[25]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like '%' to filter for WHERE Mission_Outcome was a success or a failure.

```
[27]: %sql select "Mission_Outcome", count("Mission_Outcome") from SPACEXTBL group by MISSION_OUTCOME;
 * sqlite:///my_data1.db
Done.
```

[27]:

Mission_Outcome	count("Mission_Outcome")
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.

```
[30]: %sql select "Payload", PAYLOAD_MASS__KG_ from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
 * sqlite:///my_data1.db
Done.
```

[30]:

Payload	PAYLOAD_MASS__KG_
Starlink 1 v1.0, SpaceX CRS-19	15600
Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
Starlink 3 v1.0, Starlink 4 v1.0	15600
Starlink 4 v1.0, SpaceX CRS-20	15600
Starlink 5 v1.0, Starlink 6 v1.0	15600
Starlink 6 v1.0, Crew Dragon Demo-2	15600
Starlink 7 v1.0, Starlink 8 v1.0	15600
Starlink 11 v1.0, Starlink 12 v1.0	15600
Starlink 12 v1.0, Starlink 13 v1.0	15600
Starlink 13 v1.0, Starlink 14 v1.0	15600
Starlink 14 v1.0, GPS III-04	15600
Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

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

```
[34]: %sql select substr(Date, 4, 2) as mounth, Booster_Version, Launch_site from SPACEXTBL where substr(Date,7,4) = '2015' and "Landing _Outcome" ='Failure (drone ship)'
 * sqlite:///my_data1.db
Done.
```

[34]:

mounth	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	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 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.

[51]:
```
%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 count("Landing _Outcome") desc;
```

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

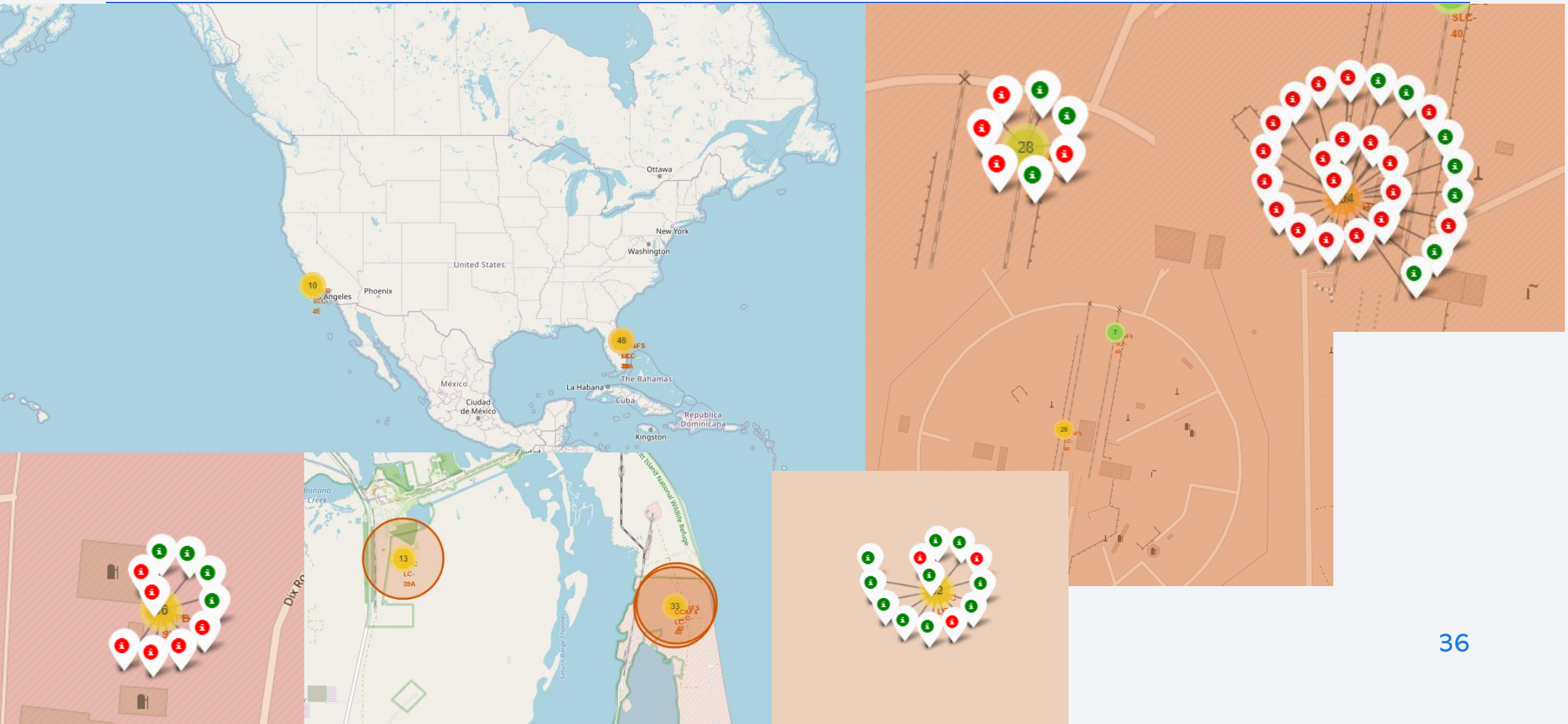
[51]:

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

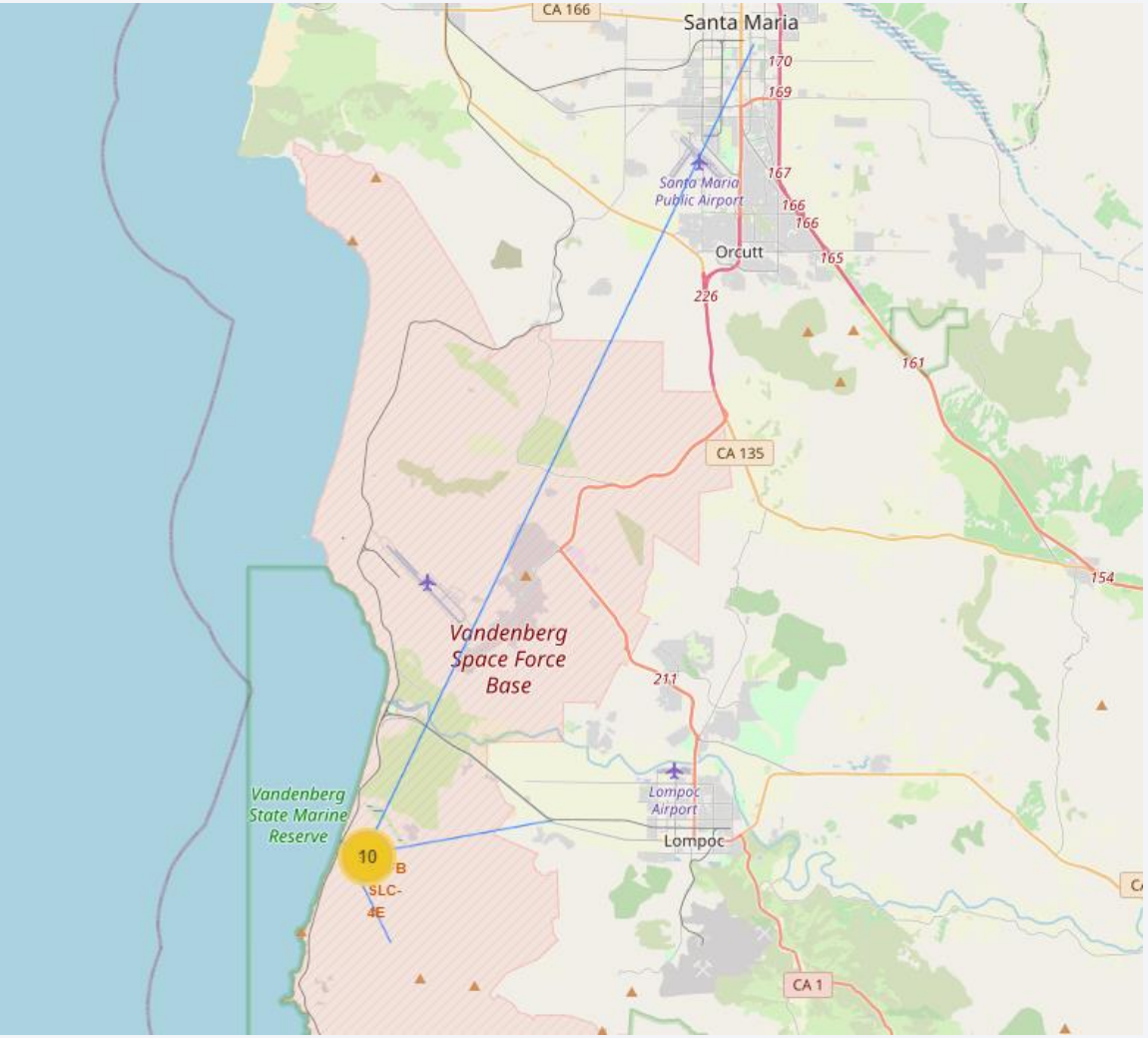

Section 3
Launch Sites
Proximities Analysis

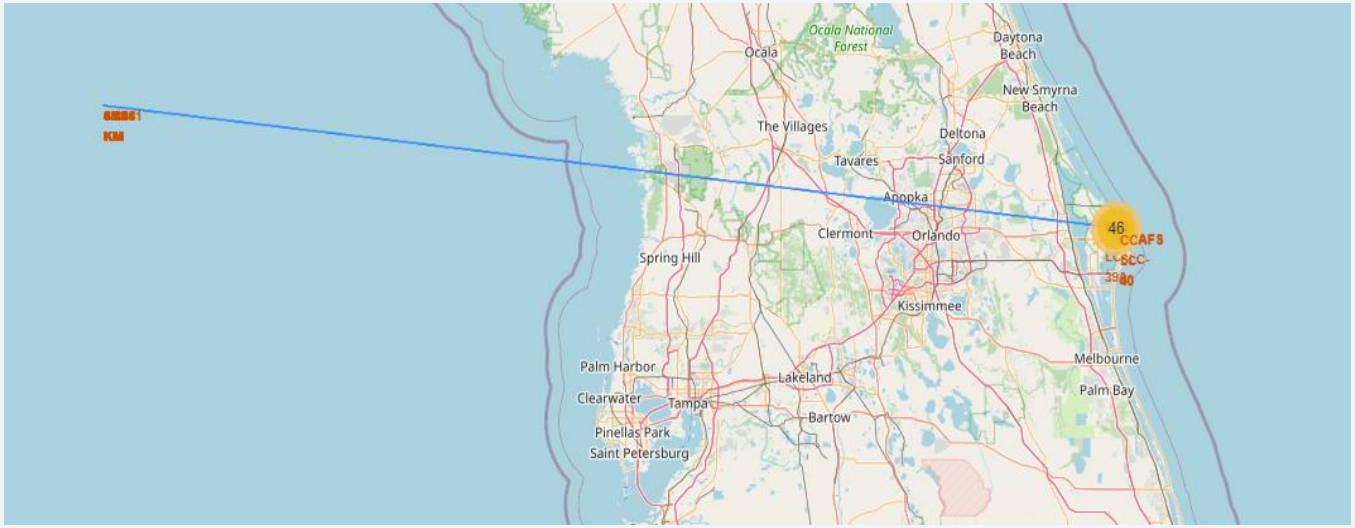
All launch sites global map markers

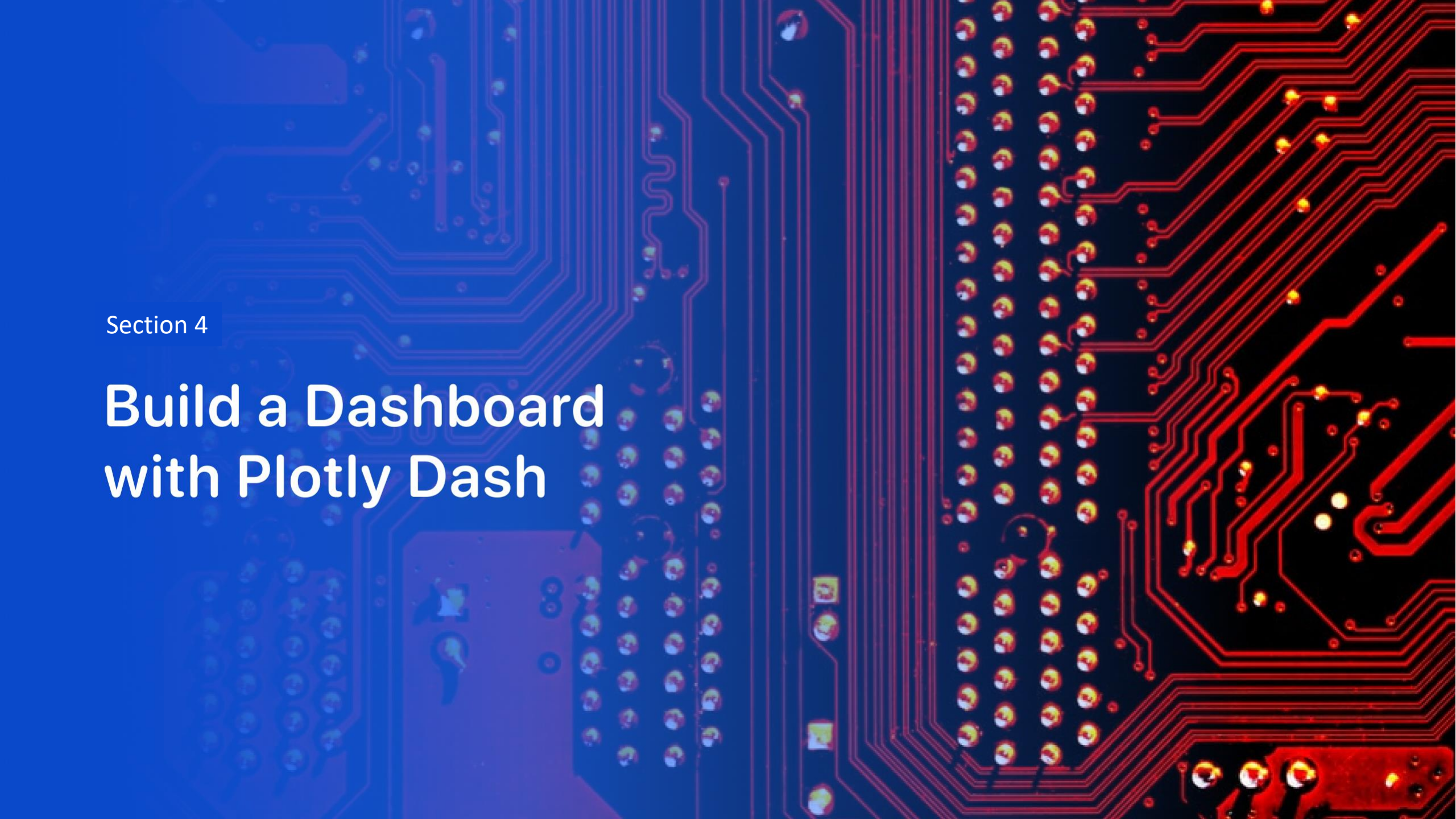
Markers showing launch sites with color labels



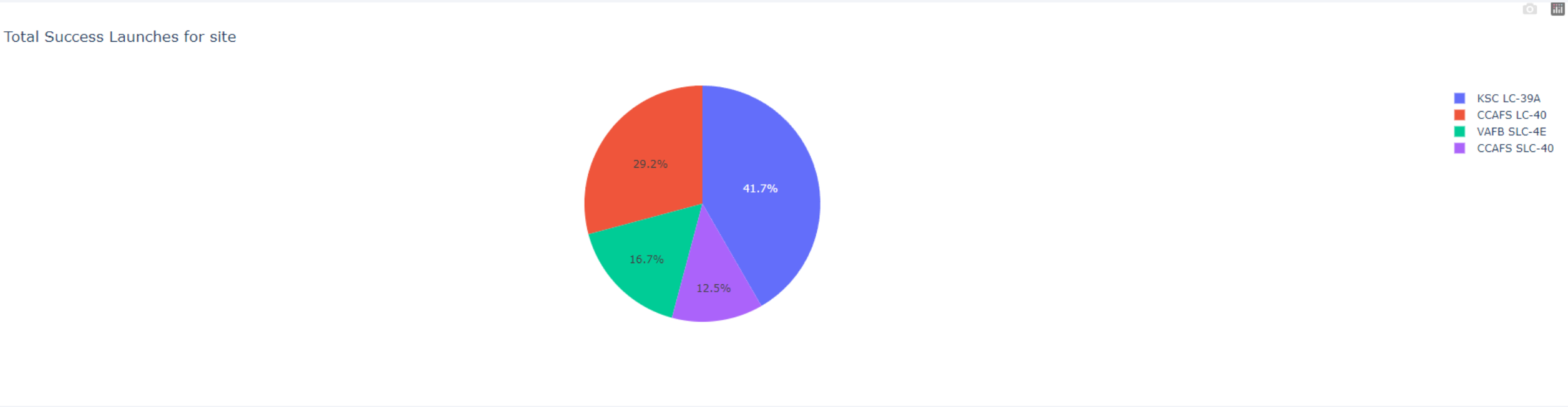
Launch Site distance to landmarks





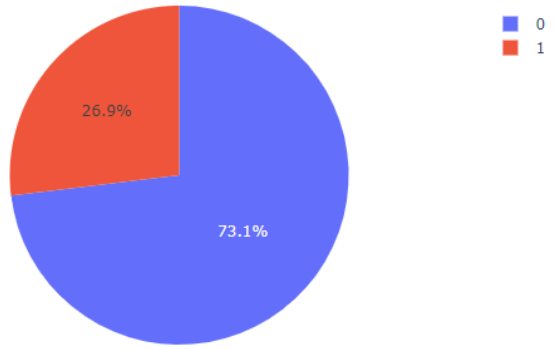

Section 4
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



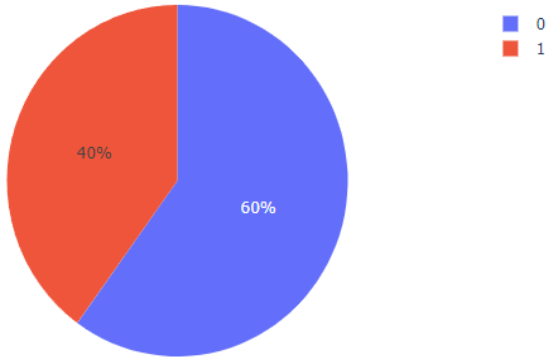
CCAFS LC-40

Total Success Launches for site CCAFS LC-40



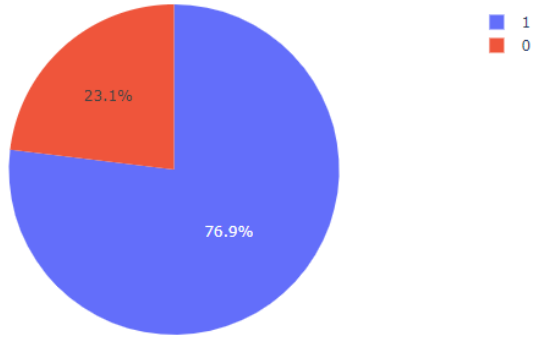
VAFB SLC-4E

Total Success Launches for site VAFB SLC-4E



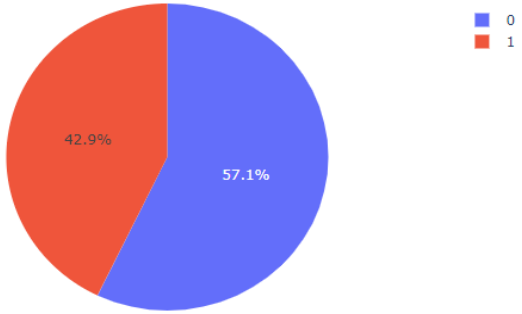
KSC LC-39A

Total Success Launches for site KSC LC-39A

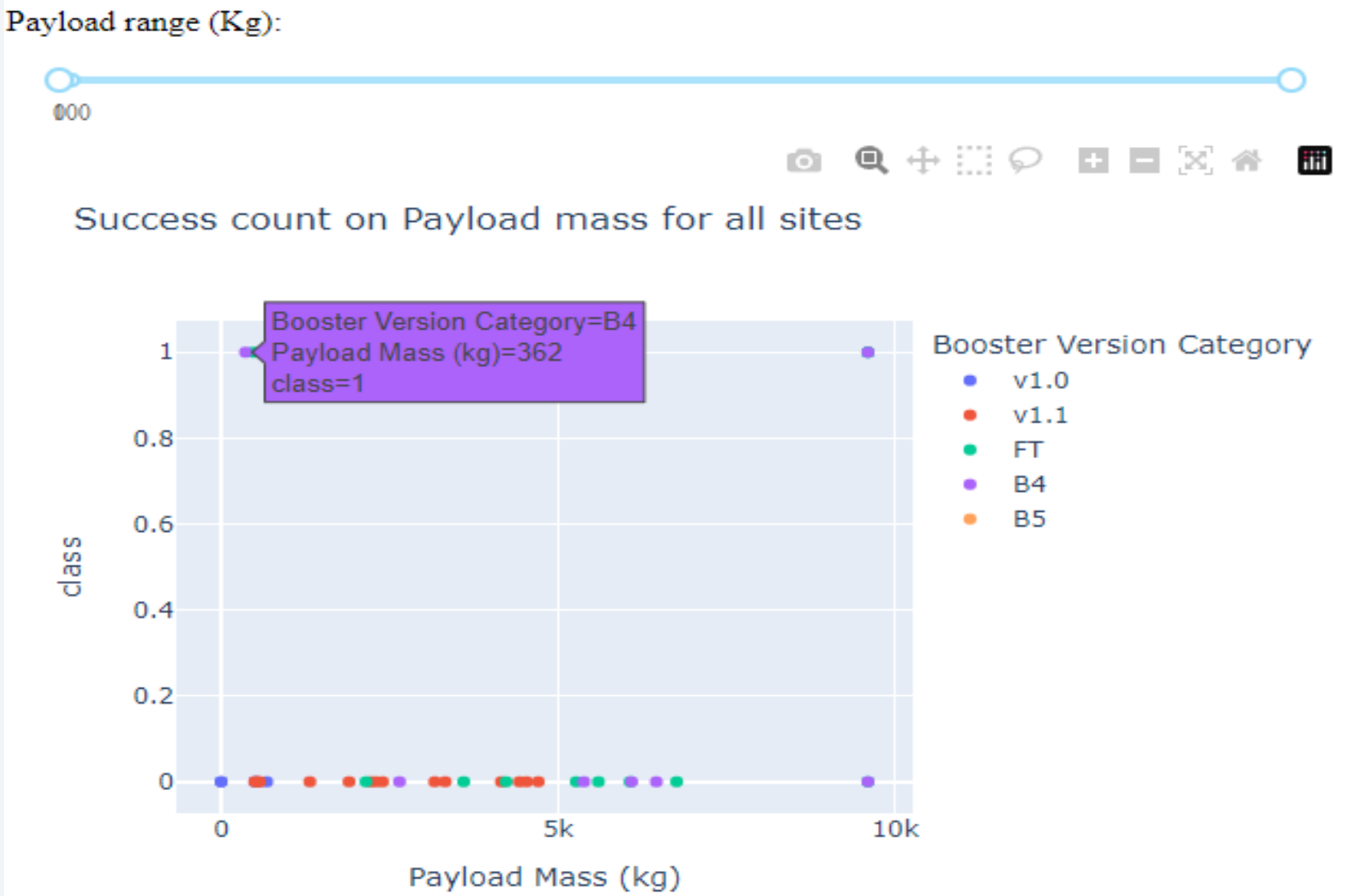


CCAFS SLC-40

Total Success Launches for site CCAFS SLC-40



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



Section 5

Predictive Analysis (Classification)

Classification Accuracy

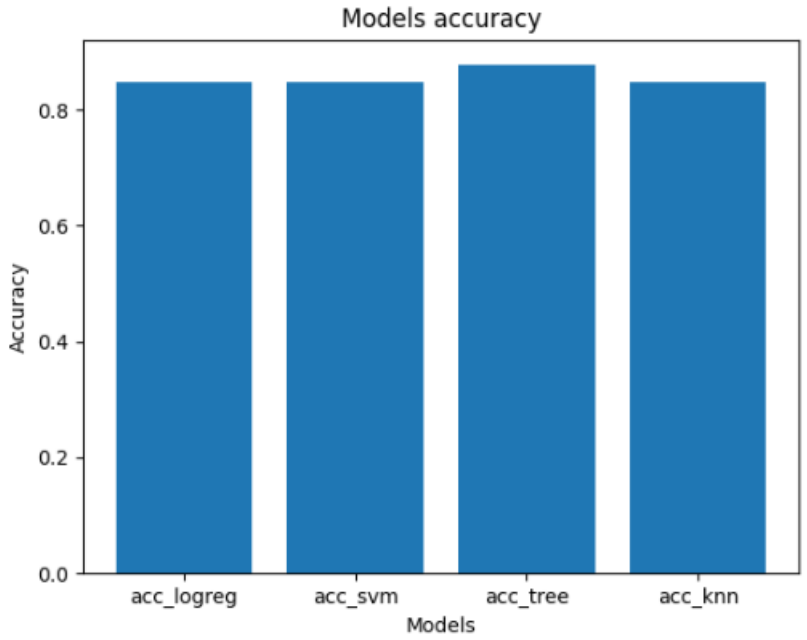
[44]:
```
models = {'acc_logreg':logreg_cv.best_score_,'acc_svm':svm_cv.best_score_,'acc_tree':tree_cv.best_score_,'acc_knn':knn_cv.best_score_}
keys_with_best_value = []
best = max(models.values())
for key, value in models.items():
    if value == best:
        keys_with_best_value.append(key)
print('best model is', keys_with_best_value[0],"with accuracy is", best)
```

```
best model is acc_tree with accuracy is 0.8767857142857143
```

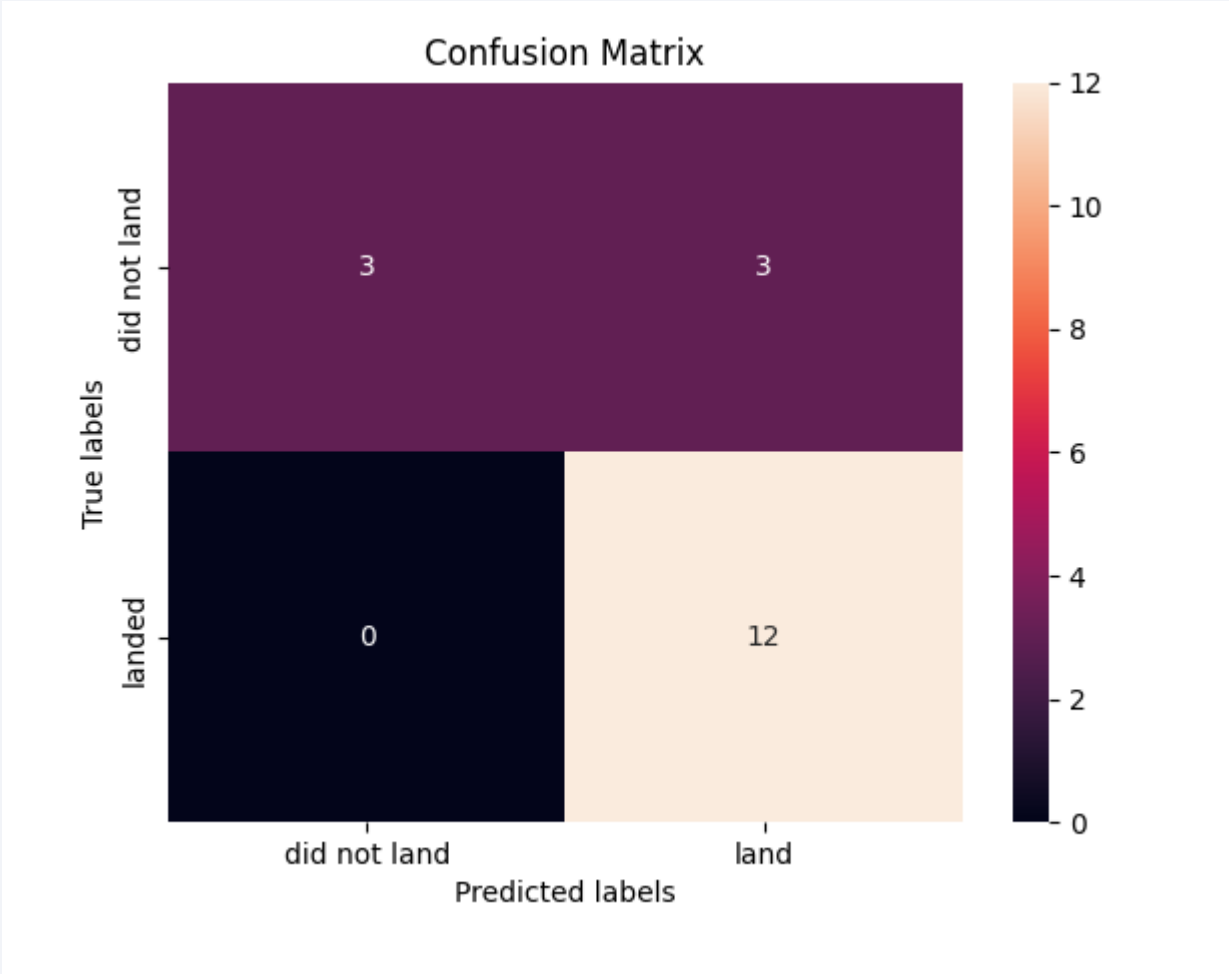
[45]:
```
plt.bar(models.keys(), models.values())

plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Models accuracy')


plt.show()
```



Confusion Matrix



- The confusion matrix for the decision tree classifier indicates that the classifier is capable of distinguishing between the various classes. However, the primary issue lies in the false positives, where unsuccessful landings are incorrectly identified as successful landings by the classifier.

Conclusions

- We can draw the following conclusions:
- • The higher the number of flights from a launch site, the higher the success rate at that site.
- • The launch success rate has been on the rise from 2013 to 2020.
- • Orbits ES-L1, GEO, HEO, SSO, and VLEO have the highest success rates.
- • KSC LC-39A has the most successful launches of any launch site.
- • The decision tree classifier is the best machine learning algorithm for this task.

Thank you!