IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Riddhima Bhalla
09/29/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection with SpaceX API
  - Data collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) using SQL
  - EDA using Data Visualization
  - Interactive Visual Analytics with Folium and Dashboard
  - •Predictive Analysis with Machine Learning
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- PROJECT BACKGROUND AND CONTEXT

  Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars, while other providers cost upward of 165 million dollars each. Much of the savings are because Space X can reuse its 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. The objective of this project is to create a machine learning pipeline to predict if the first stage will land successfully.

- PROBLEMS WE WANT TO FIND ANSWERS TO :

  - 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 needs to be in place to ensure a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was obtained using SpaceX API and web-scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was done on 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
  - Logistic Regression, SVM, Decision Tree, and KNN models were used
  - Hyperparameters were tuned using GridSearch, cross-validation
  - Accuracy on test data was calculated using score method and confusion matrix was plotted.

Data Collection

- **The following steps were involved in Data Collection:**

SPACEX API -

1. Data was collected by sending a GET request to SpaceX API.
2. The data was decoded as json using .json() and converted to a DataFrame using .json_normalize().
3. The data was then cleaned, missing values were identified, and nan values were also accommodated for.

WEB SCRAPING -

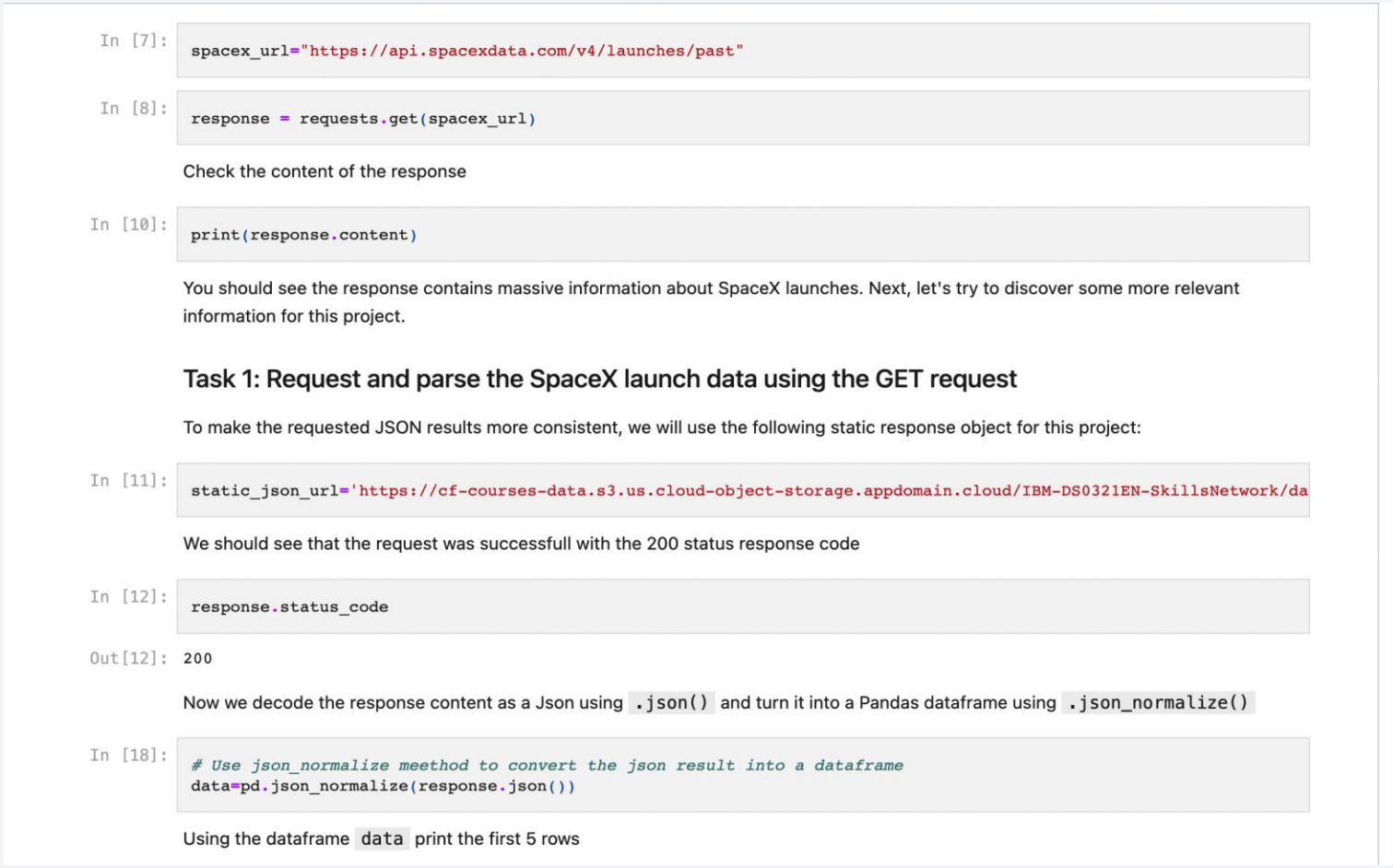
1. Falcon 9 launch records were also collected using web-scraping from Wikipedia.
2. The HTTP GET request was sent to Falcon 9 launch HTML page .
3. The data was then parsed and stored as a DataFrame using BeautifulSoup.

Data Collection – SpaceX API

Data was collected by sending a GET request to SpaceX API. The data was decoded as json using .json() and converted to a DataFrame using .json_normalize().

- GitHub URL-

https://github.com/riddhima2b/SPACEX/blob/main/-spacex-DATA-COLLECTION-.ipynb



In [7]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [8]:
```
response = requests.get(spacex_url)
```

Check the content of the response

In [10]:
```
print(response.content)
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

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 [11]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

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

In [12]:
```
response.status_code
```

Out[12]:
```
200
```

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

In [18]:
```
# 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

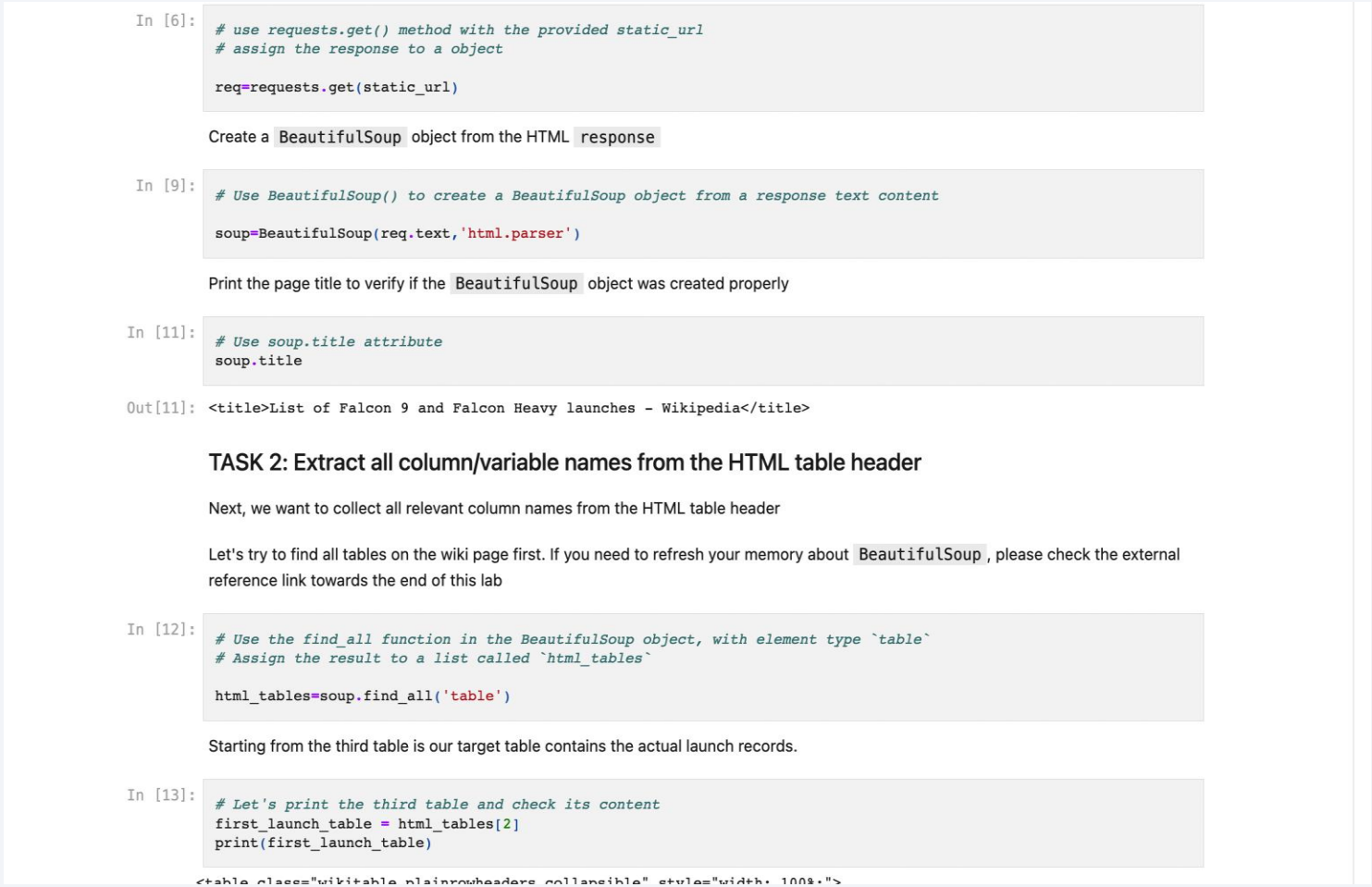
Data Collection - Scraping

Falcon 9 launch records were also collected using web-scraping from Wikipedia.

The HTTP GET request was sent to Falcon 9 launch HTML page. The data was then parsed and stored as a DataFrame using BeautifulSoup.

- GitHub URL-

https://github.com/riddhima2b/SPACEX/blob/main/WEBSCRAPING.ipynb



In [6]:
```
# use requests.get() method with the provided static_url
# assign the response to a object

req=requests.get(static_url)
```

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

In [9]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

soup=BeautifulSoup(req.text,'html.parser')
```

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

In [11]:
```
# Use soup.title attribute
soup.title
```

Out[11]:
```
<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 [12]:
```
# 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.

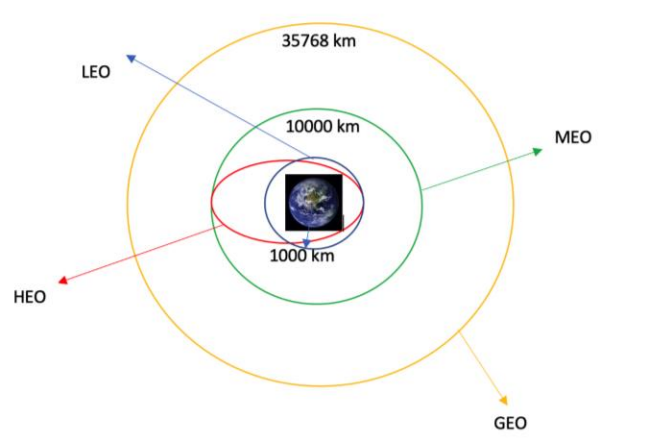
In [13]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
```

Data Wrangling

- We calculated the number of launches at each site, the number and occurrence of each orbits. Then, created landing outcome label from outcome column and exported the results to csv.
- GitHub -

https://github.com/riddhima2b/SPACEX/blob/main/spacex-DATA%20WRANGLING_jupyterlite.jupyterlite.ipynb



In [30]:
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

In [31]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Out[31]:
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

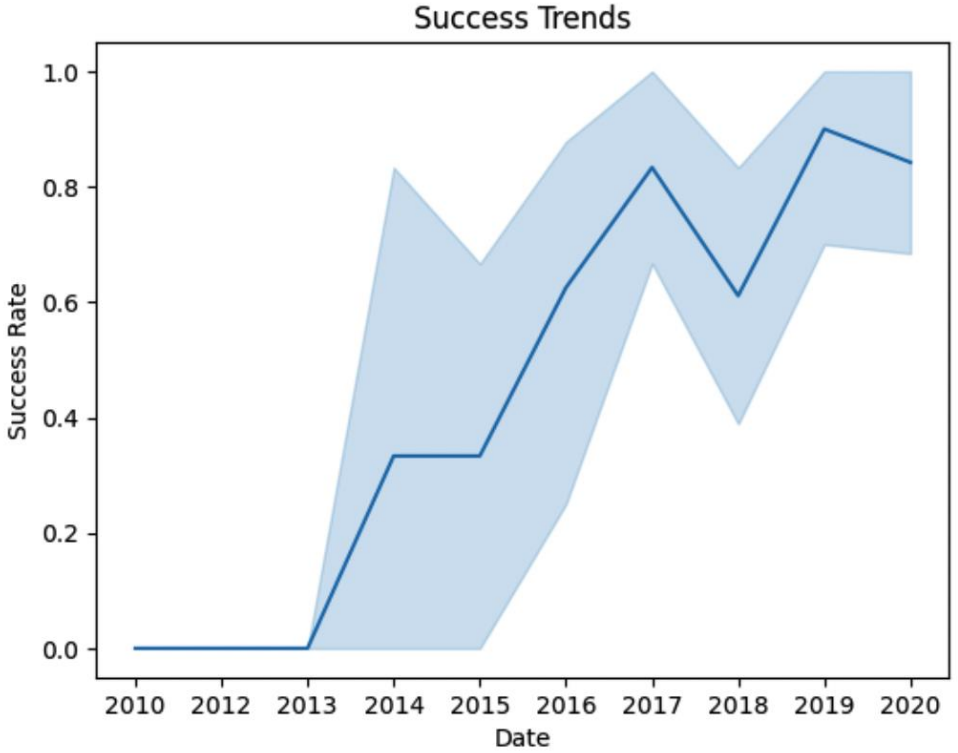
In [33]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class=[]
for i in df['Outcome']:
    if i in bad_outcomes:
        landing_class+=[0]
    else:
        landing_class+=[1]
```

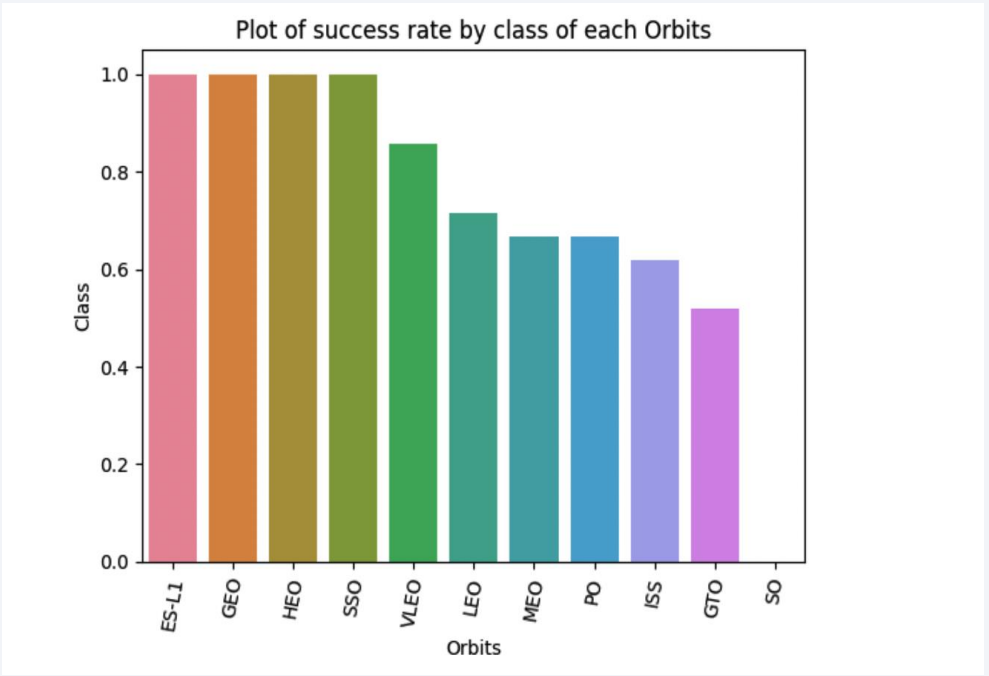
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [34]:

EDA with Data Visualization

- We were able to visualize relationships between FlightNumber and PayloadMass, FlightNumber vs Launch Site, Launch site vs Orbits, yearly success trends, etc. We plotted bar charts, line plots and scatter plots.





GitHub Link-
https://github.com/riddhima2b/SPACEX/blob/main/EDA%20DATAVIZ.ipynb

EDA with SQL

- We used SQLite for various queries -
  - Display the total payload mass carried by boosters launched by NASA (CRS)
  - Display average payload mass carried by booster version F9 v1.1
  - List the date when the first succesful landing outcome in ground pad was acheived.
  - List the total number of successful and failure mission outcomes

And many more..

- GitHub URL - https://github.com/riddhima2b/SPACEX/blob/main/eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We marked launch sites, added map objects (markers) 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 were able to identify the launch sites with a relatively high success rate.
- We calculated the distances between the launch site and its proximities like railways, highways, and cities

- GitHub URL-

https://github.com/riddhima2b/SPACEX/blob/main/LAunch%20site%20and%20Folium.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.
- Added an option to select the launch sites based on name.
- GitHub URL- https://github.com/riddhima2b/SPACEX/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

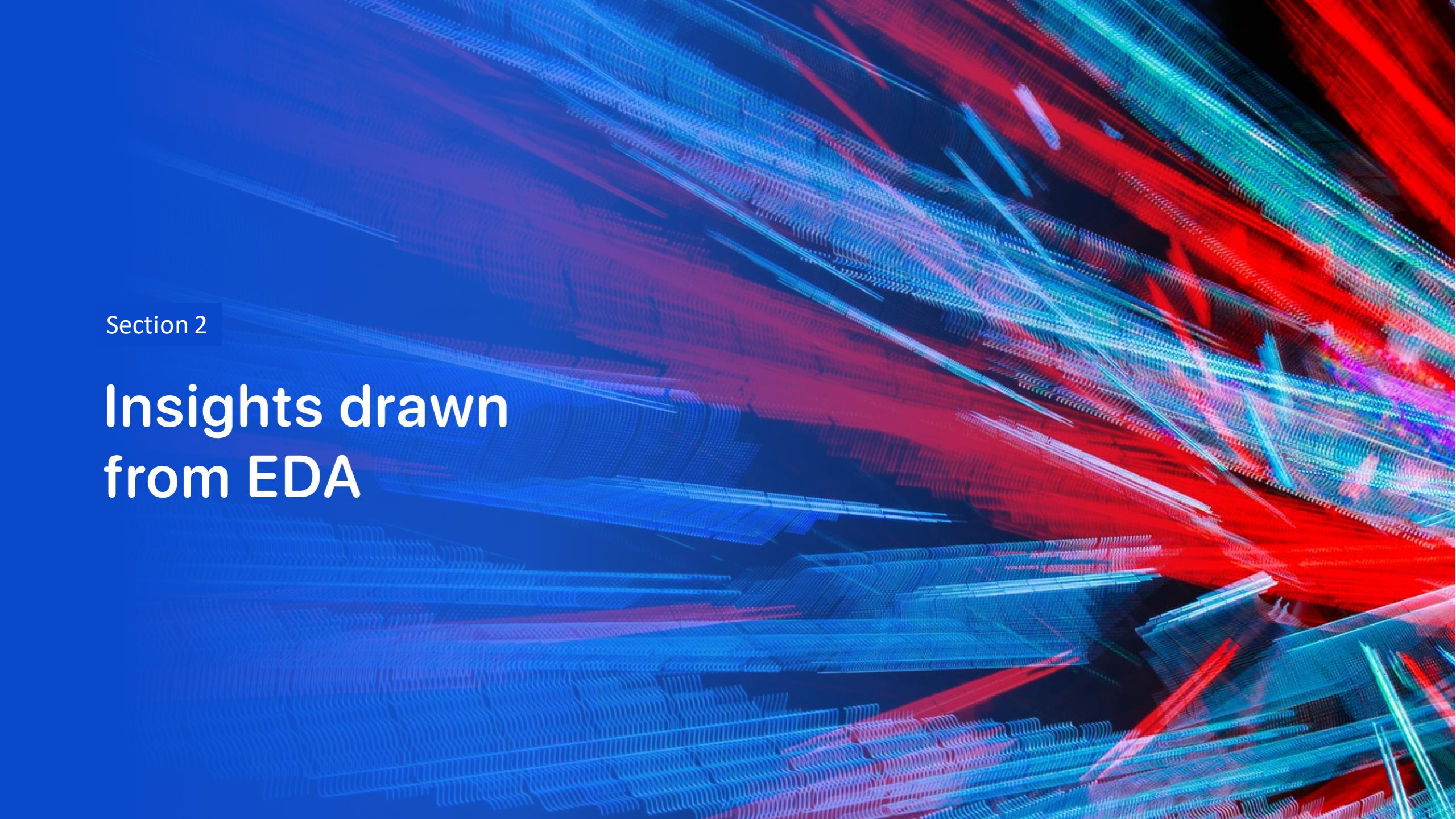
- We split the dataset into training and testing set and then, built machine learning models and tuned different hyperparameters using GridSearchCV.
- We were able to find the best performing classification model.
- GitHub
  URL https://github.com/riddhima2b/SPACEX/blob/main/Machine%20learning.ipynb

Results

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

Section 2

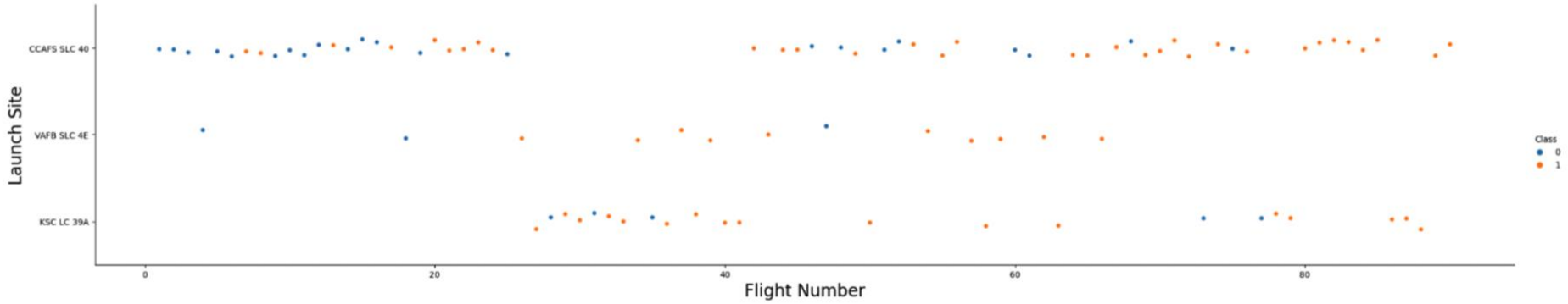
Insights drawn from EDA



Flight Number vs. Launch Site

In [28]:

```
# 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
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```



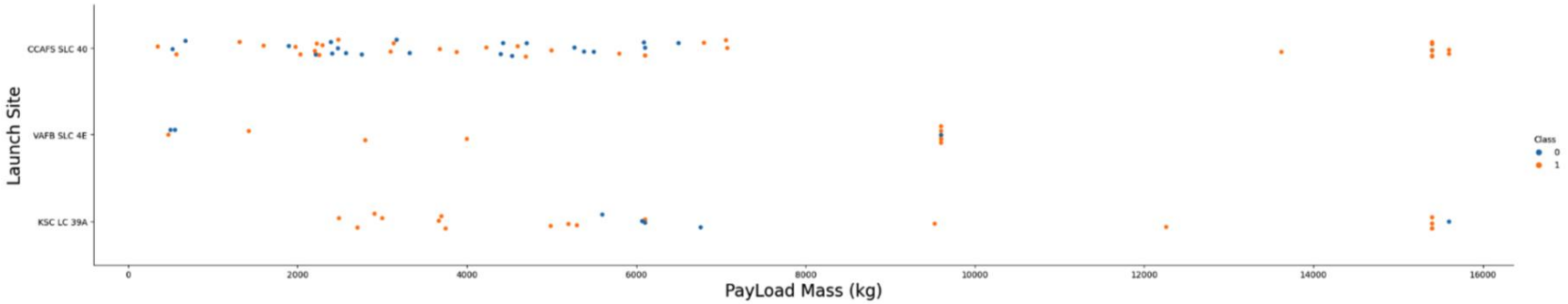
Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

- From the above-mentioned scatter plot of Flight Number vs. Launch Site, we see that greater the number of flights at a launch site, greater is the success rate at the launch site.

Payload vs. Launch Site

In [27]:

```
# 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

sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayLoad Mass (kg)",fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

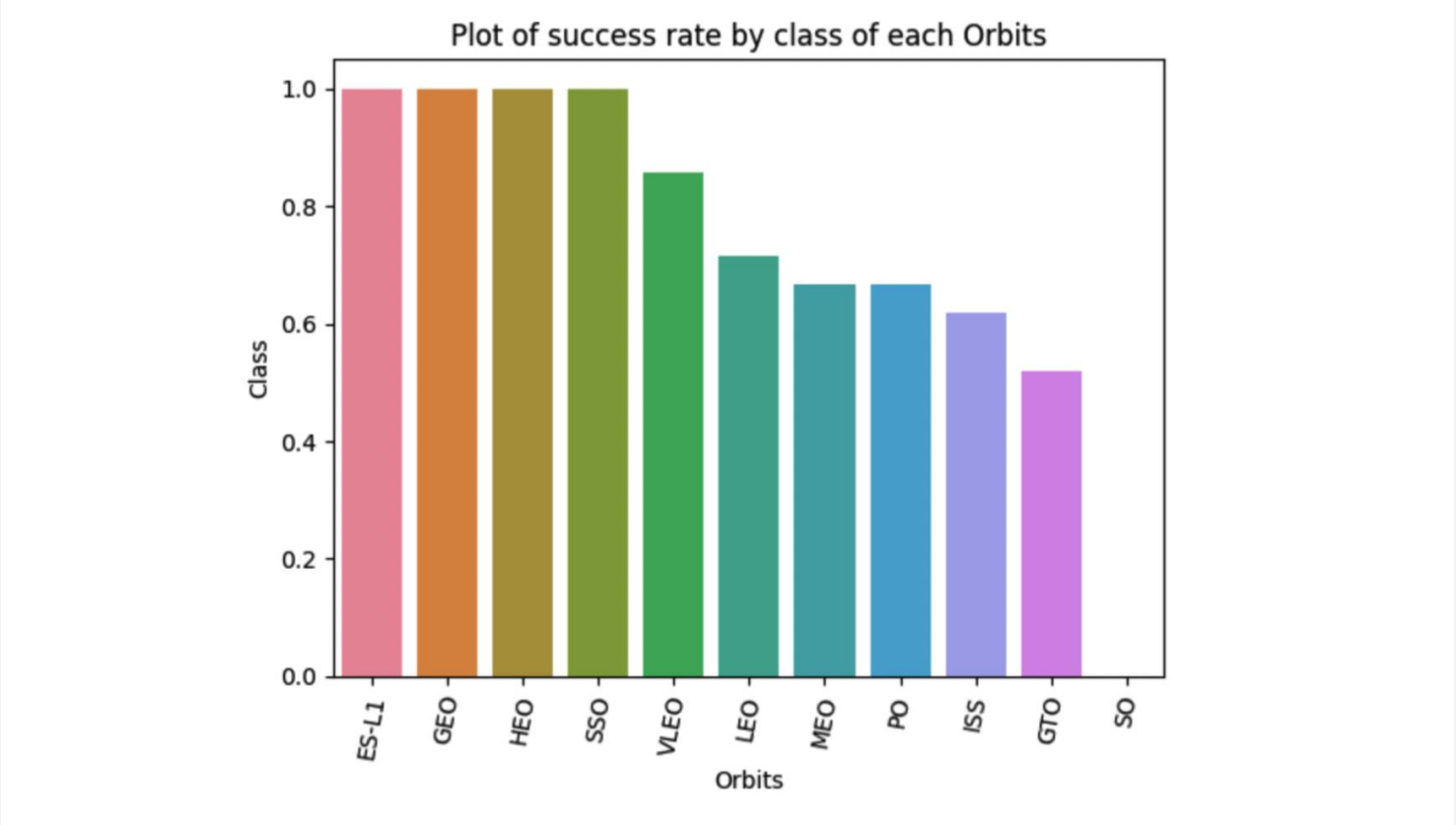


Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

- For VAFB-SLC launch site there are no rockets launched for payload greater than 10,000 kg.
- For KDC LC 39A launch site there are no rockets launched for payload less than 2,000 kg

Success Rate vs. Orbit Type

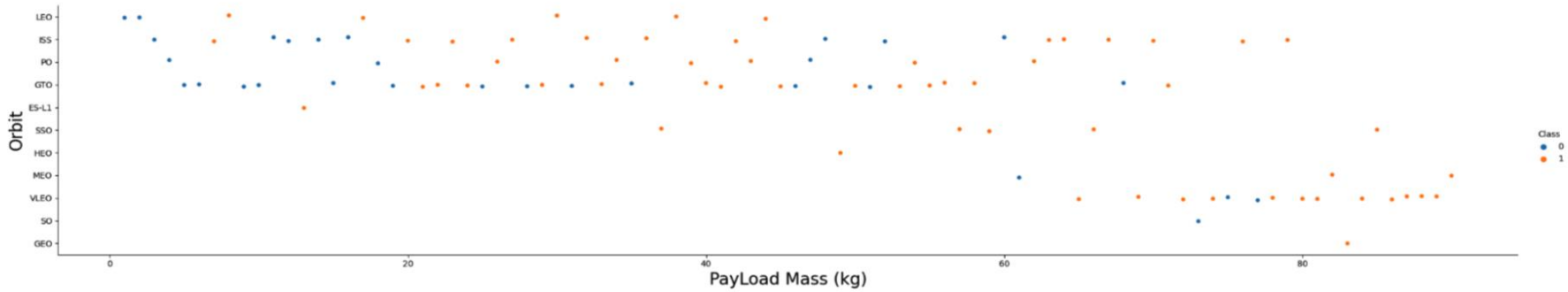
- We can see that ES-L1, GEO, HEO and SSO have the highest success rate.



Flight Number vs. Orbit Type

In [36]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class v
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("PayLoad Mass (kg)",fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.show()
```

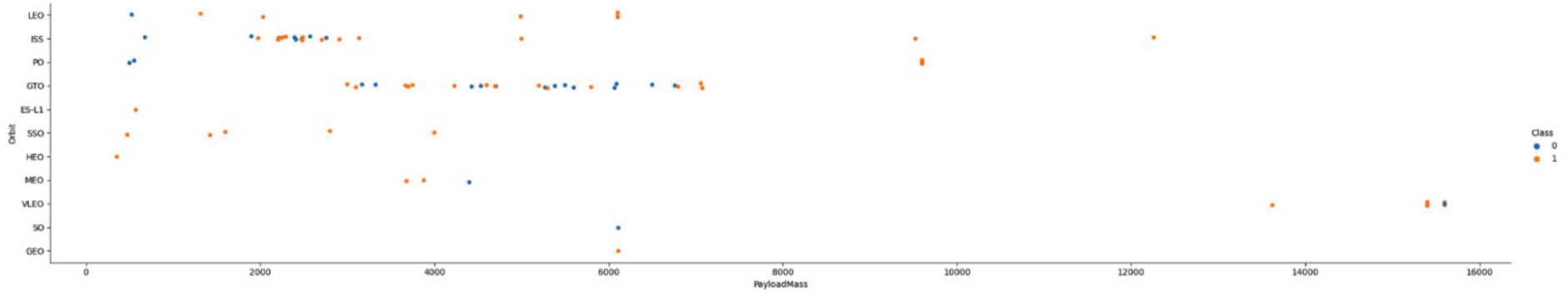


- We see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type

In [35]:

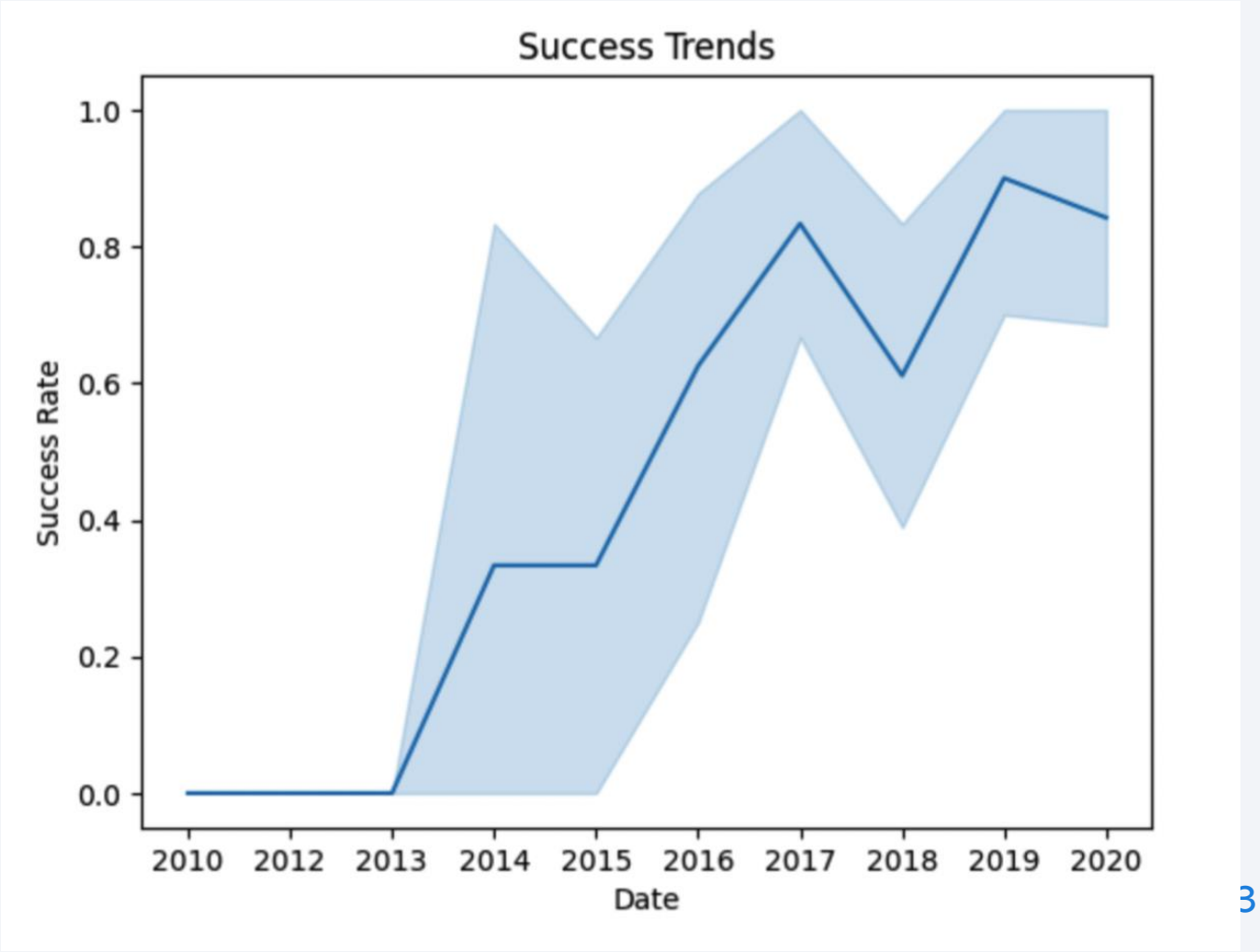
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value

sns.catplot(y="Orbit",x="PayloadMass",hue="Class",data=df,aspect=5)
plt.xlabel("PayloadMass")
plt.ylabel("Orbit")
plt.show()
```



- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend

- We can observe that the success rate since 2013 kept increasing till 2020.



All Launch Site Names

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

```
In [32]: %sql select distinct (Launch_site) from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

Out[32]:

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

Launch Site Names Begin with 'CCA'

Task 2

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

In [44]:
```
%sql select * from SPACEXTABLE where Launch_site like 'CCA%' LIMIT 5
```

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

Out[44]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (I
2010-08-12	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 (I
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

LIKE keyword was used.

Total Payload Mass

- We used sum(payloadmass_kg) to find total payload mass carried with a condition that booster should be from NASA.

Task 3

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

```
In [46]: %sql SELECT sum(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE 'NASA (CRS)'
```

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

Out[46]:

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

Task 4

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

In [48]:
```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version like 'F9 v1.1'
```

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

Out[48]:

avg(PAYLOAD_MASS__KG_)
2928.4

We used avg() function here.

First Successful Ground Landing Date

- The date when the first successful landing outcome in ground pad was achieved can be retrieved with min() function.

Task 5

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

Hint:Use min function

In [50]:
```
%sql select min(Date) from SPACEXTABLE where Landing_outcome like 'Success (ground pad)'
```

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

Out[50]:

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Query used - '%sql select Booster_Version from SPACEXTABLE where ((Landing_outcome like 'Success (ground pad)' and (PAYLOAD_MASS__KG_ 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

In [51]:
```
%sql select Booster_Version from SPACEXTABLE where ((Landing_outcome like 'Success (ground pad)' and (PAYLOAD_MAS
```

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

Out[51]:

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [58]:
```
%sql select count(Mission_Outcome)FROM SPACEXTABLE where Mission_Outcome like 'Success%'
```

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

Out[58]:

count(Mission_Outcome)
100

In [59]:
```
%sql select count(Mission_Outcome)FROM SPACEXTABLE where Mission_Outcome like 'Failure%'
```

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

Out[59]:

count(Mission_Outcome)
1

Count() function used.

Boosters Carried Maximum Payload

- **Query used -** **'%sql** select Booster_Version,PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_= (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)'
-

Task 8

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

In [62]:
```
%sql select Booster_Version,PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_= (select max(PAYLOAD_MASS_
```

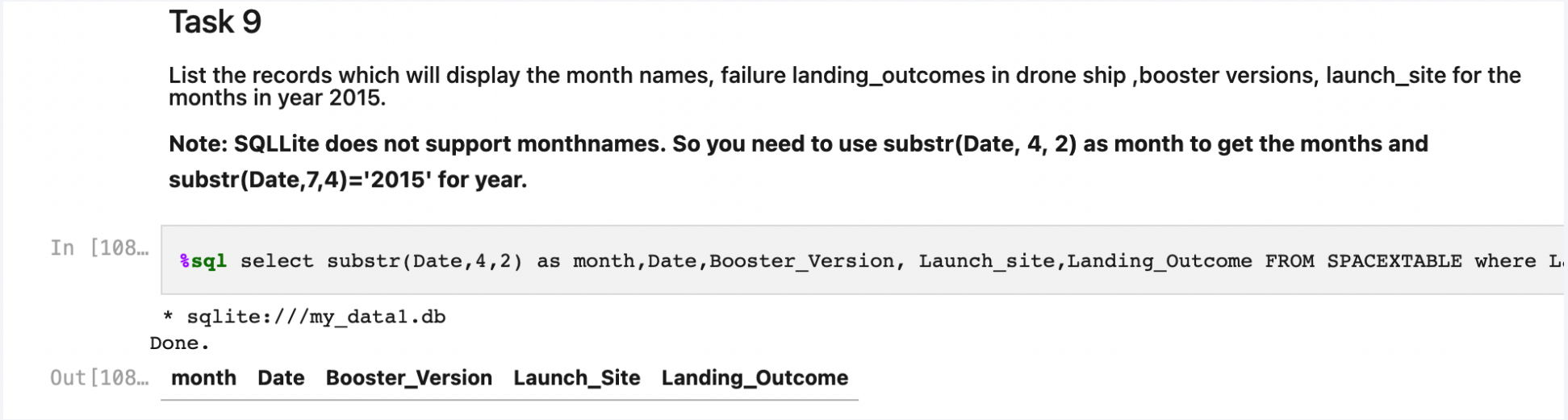
```
 * sqlite:///my_data1.db
Done.
```

Out[62]:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- Query used here - ' **%sql** select substr(Date,4,2) as month,Date,Booster_Version, Launch_site,Landing_Outcome FROM SPACEXTABLE where Landing_Outcome like 'Failure (drone ship)' and substr(Date,7,4)='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.

In [108…

```
%sql select substr(Date,4,2) as month,Date,Booster_Version, Launch_site,Landing_Outcome FROM SPACEXTABLE where L
```

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

Out[108…

month	Date	Booster_Version	Launch_Site	Landing_Outcome

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

- **Query used - %sql** select Landing_Outcome, count(Landing_Outcome) from SPACEXTABLE where Date between '2010-06-04' and '2017-03-20' Group by Landing_Outcome order by count(Landing_Outcome) desc
-

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

In [114…

```
%sql select Landing_Outcome, count(Landing_Outcome) from SPACEXTABLE where Date between '2010-06-04' and '2017-03
```

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

Out[114…

Landing_Outcome	count(Landing_Outcome)
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

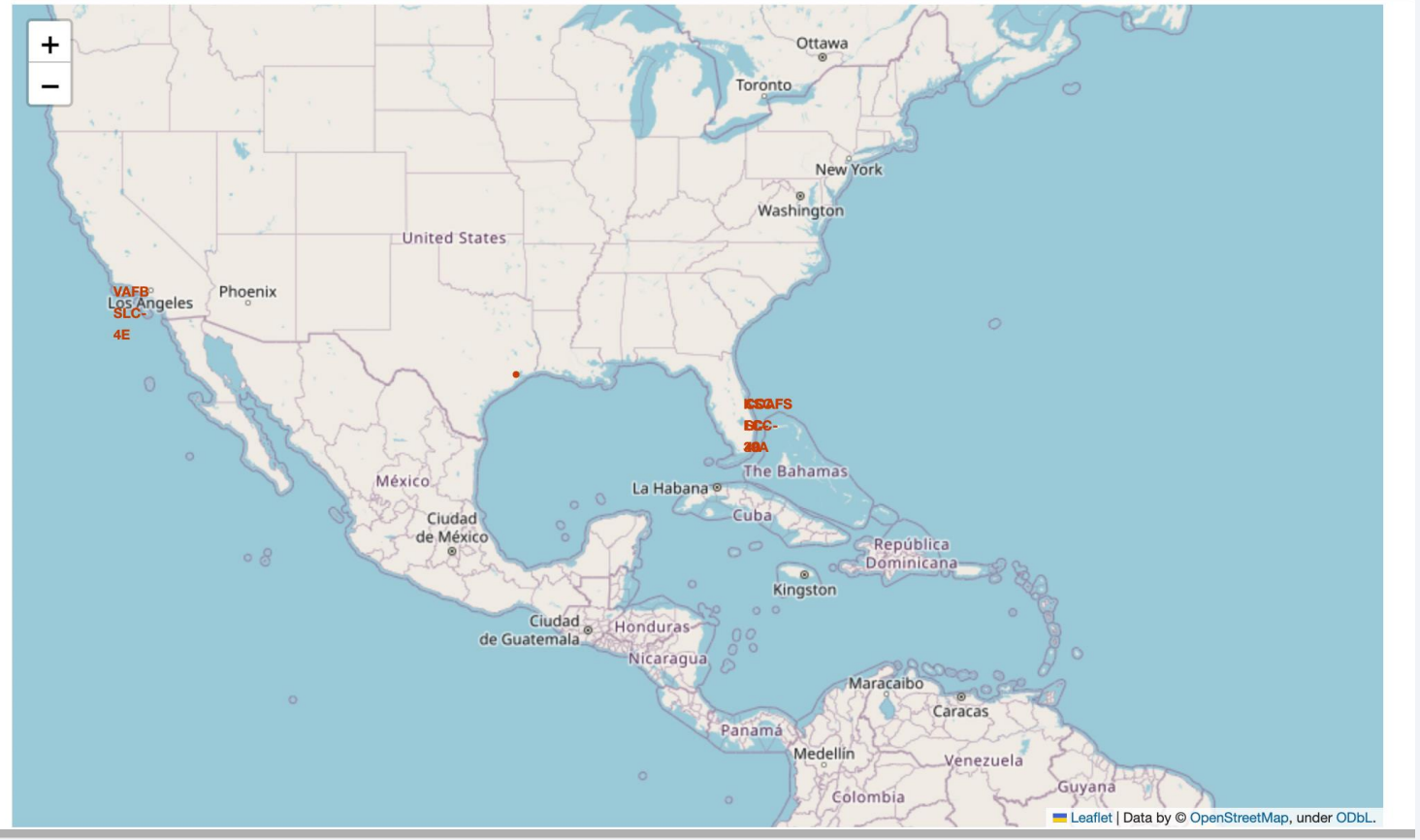

Section 3
Launch Sites
Proximities Analysis

ALL LAUNCH SITES ON THE MAP

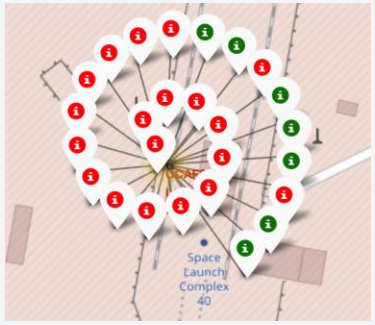
- The SpaceX Launch Sites are on the coasts of Florida and California in the USA.

```
site_map
```

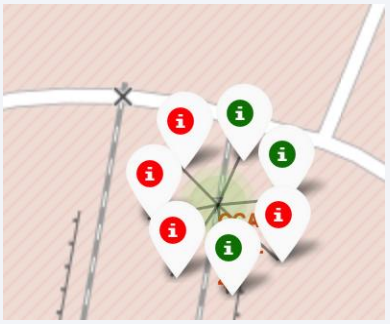
8]:



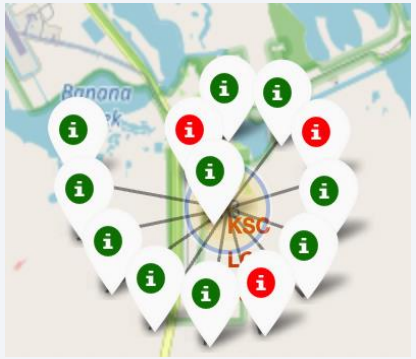
SUCCESS/FAILED LAUNCHES FOR EACH SITE ON THE MAP



CCAFS LC 40



CCAFS SLC 40



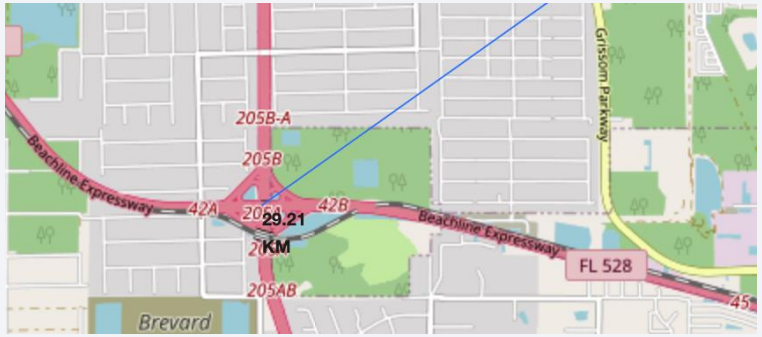
KSC LC 39A

Red Markers – Failure;
Green Markers - Success

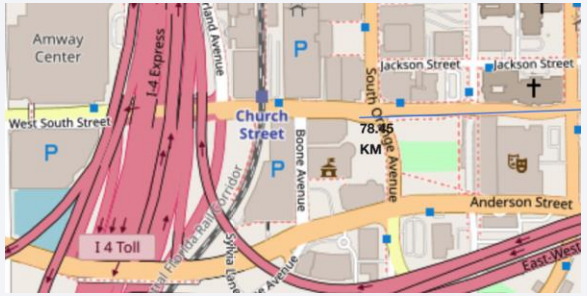
Launch Site Proximity From Landmarks

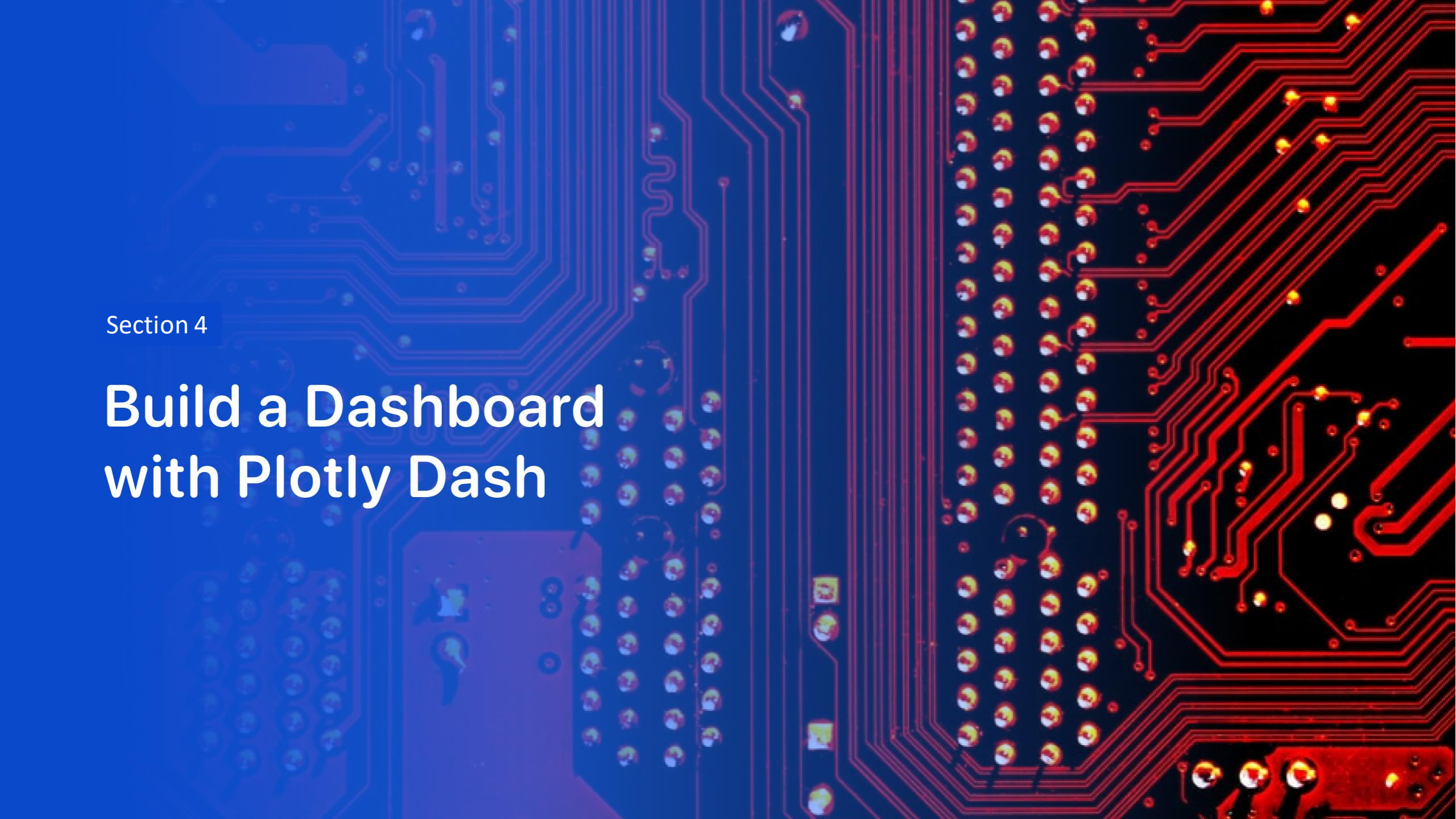
- Is distance to coastline close proximity? Yes, 0.9 km

- Are the highways close? No. (29.1km)

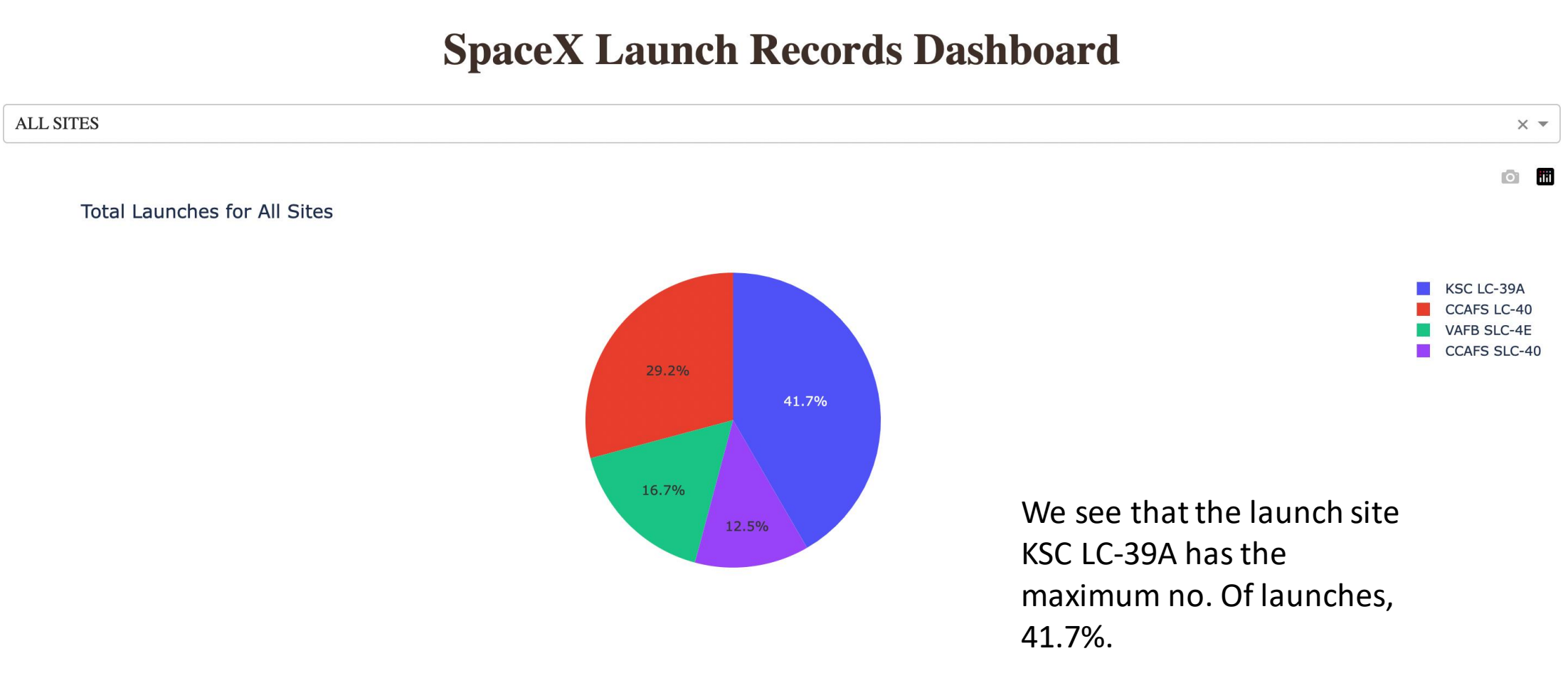


- Is the city close? No, 78.45 kms away.



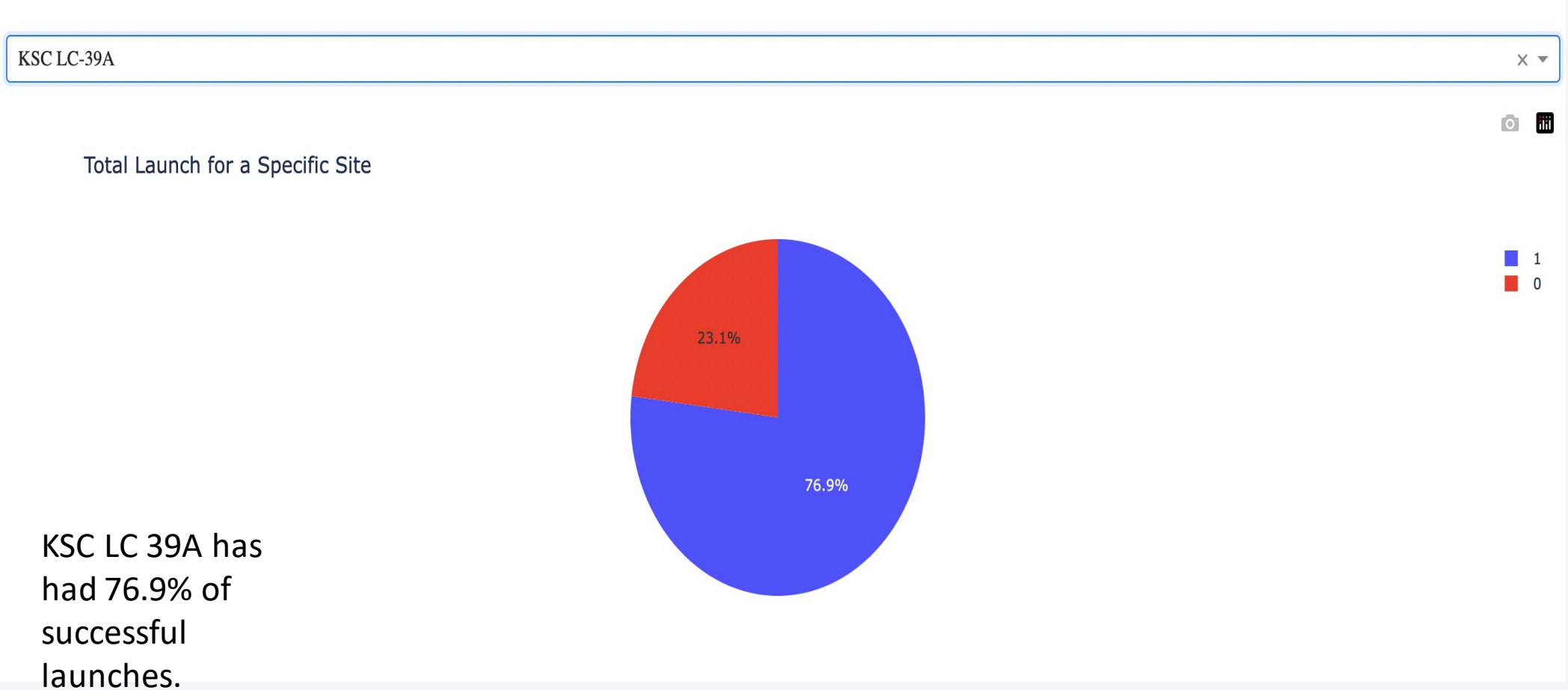

Section 4
Build a Dashboard with Plotly Dash

LAUNCH SUCCESS COUNT FOR ALL SITES



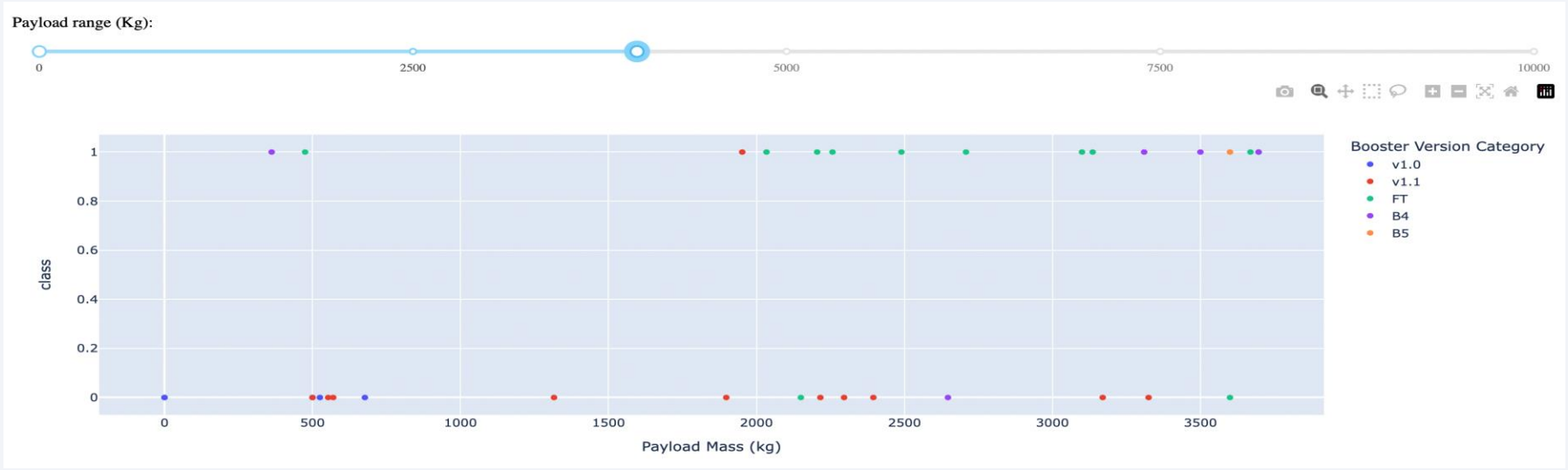
We see that the launch site KSC LC-39A has the maximum no. Of launches, 41.7%.

LAUNCH SITE WITH HIGHEST LAUNCH SUCCESS RATIO

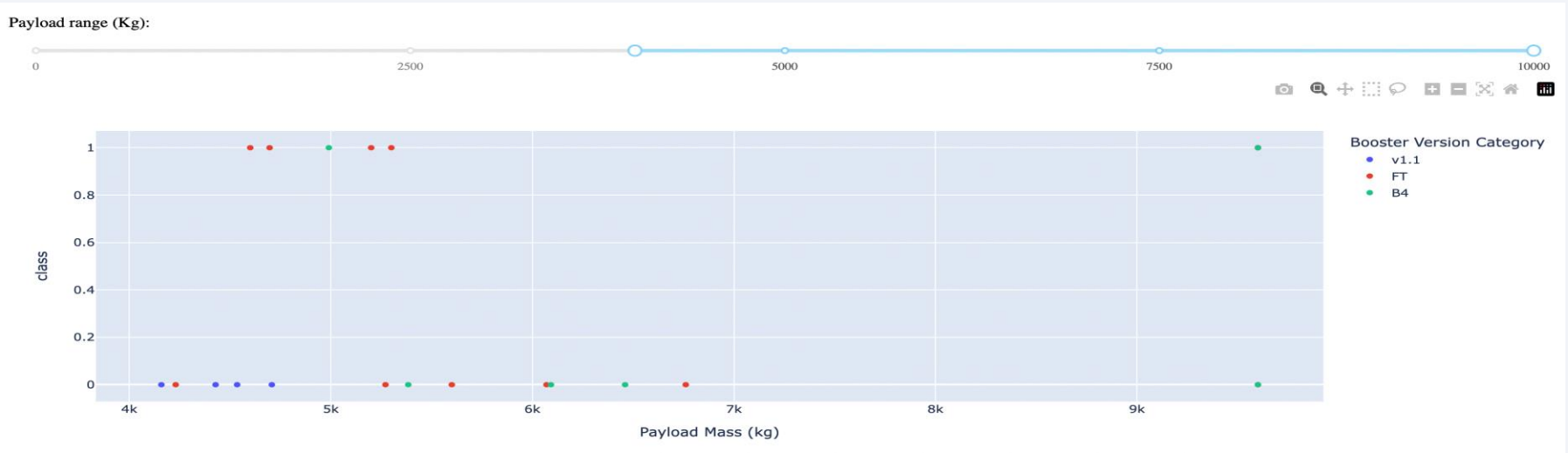


KSC LC 39A has had 76.9% of successful launches.

Payload vs. Launch Outcome



Low payload range :
less than 3500kg



High payload range :
b/w 3500kg and 10000kg

Success rate is less than
'low payload range'

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Find which model has the highest classification accuracy

TASK 12

Find the method performs best:

In [56]:

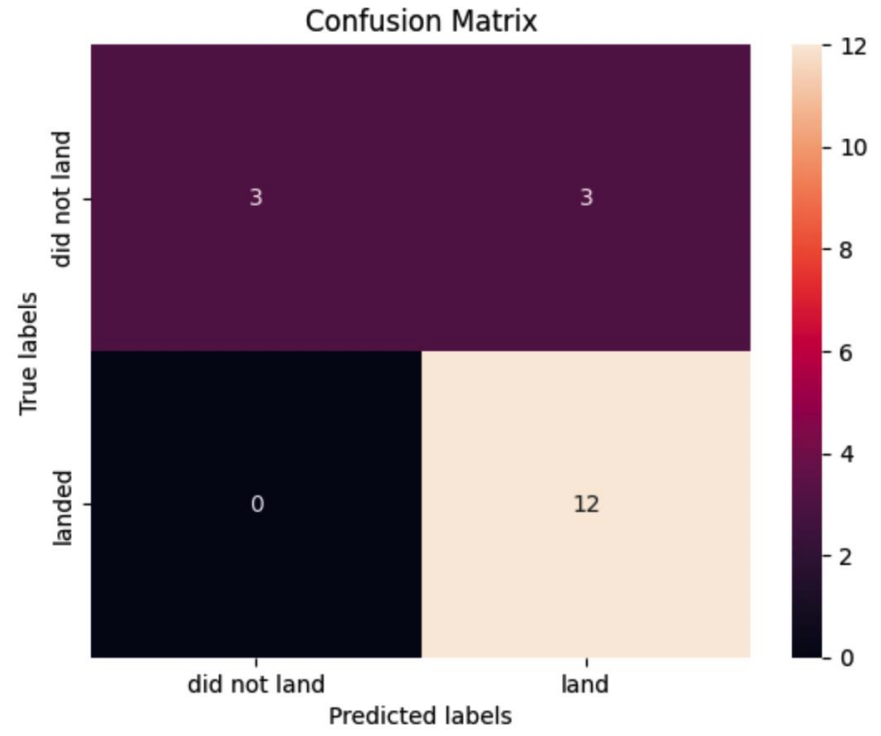
```
models = {'KNeighbors':knn_cv.best_score_,
          'DecisionTree':tree_cv.best_score_,
          'LogisticRegression':logreg_cv.best_score_,
          'SupportVector': svm_cv.best_score_}

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 0.8767857142857143
Best params is : {'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_
split': 2, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the Decision Tree Classifier shows that the classifiers gives a high number of false positives, i.e., the rocket did not actually land successfully but the classifiers predict that it landed successfully.

```
In [49]: yhat = knn_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Conclusions

We can conclude that -

- The greater the number of flights at a launch site, greater is the success rate at the launch site.
- ES-L1, GEO, HEO, and SSO orbits have the highest success rate.
- For heavier payloads, success rate increases for LEO, ISS and Polar orbits.
- Launch success rate increased from 2013 to 2020 with minor fluctuations throughout the period.
- Launch Site KDC LC-39 A has the Greatest Launch Success Rate
- The Decision Tree Classifier is the best machine learning algorithm to predict that SpaceX Falcon 9 rocket's first stage will land successfully.

Thank you!