IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ron Minihan
December, 2024

Outline

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

Executive Summary

- Summary of methodologies

  Data Collection using http api and webscraping
  Data Cleaning and Preparation (Data wrangling)
  Exploratory Data Analysis (EDA) using SQL
  Exploratory Data Analysis using Graphs
  Visual Analytics using Folium
  Visual Analytics using Dashboard - Python Dash
  Statistical Modeling - Machine Learning

- Summary of all results

  EDA
  Visual Analytics
  Modeling

Introduction

- Project Rationale

  SpaceX is a commercial space company that has been successfully launching payloads into orbit. Their cost per launch is advertised at 62 million per launch compared to the 160 million dollars of other competitors. The reason SpaceX is much more cost effective is simple: They reuse the 1rst stage. In order for Space Y to be competitive, the company must also reuse the first stage.

- Desired Project Outcome / Problems to Solve

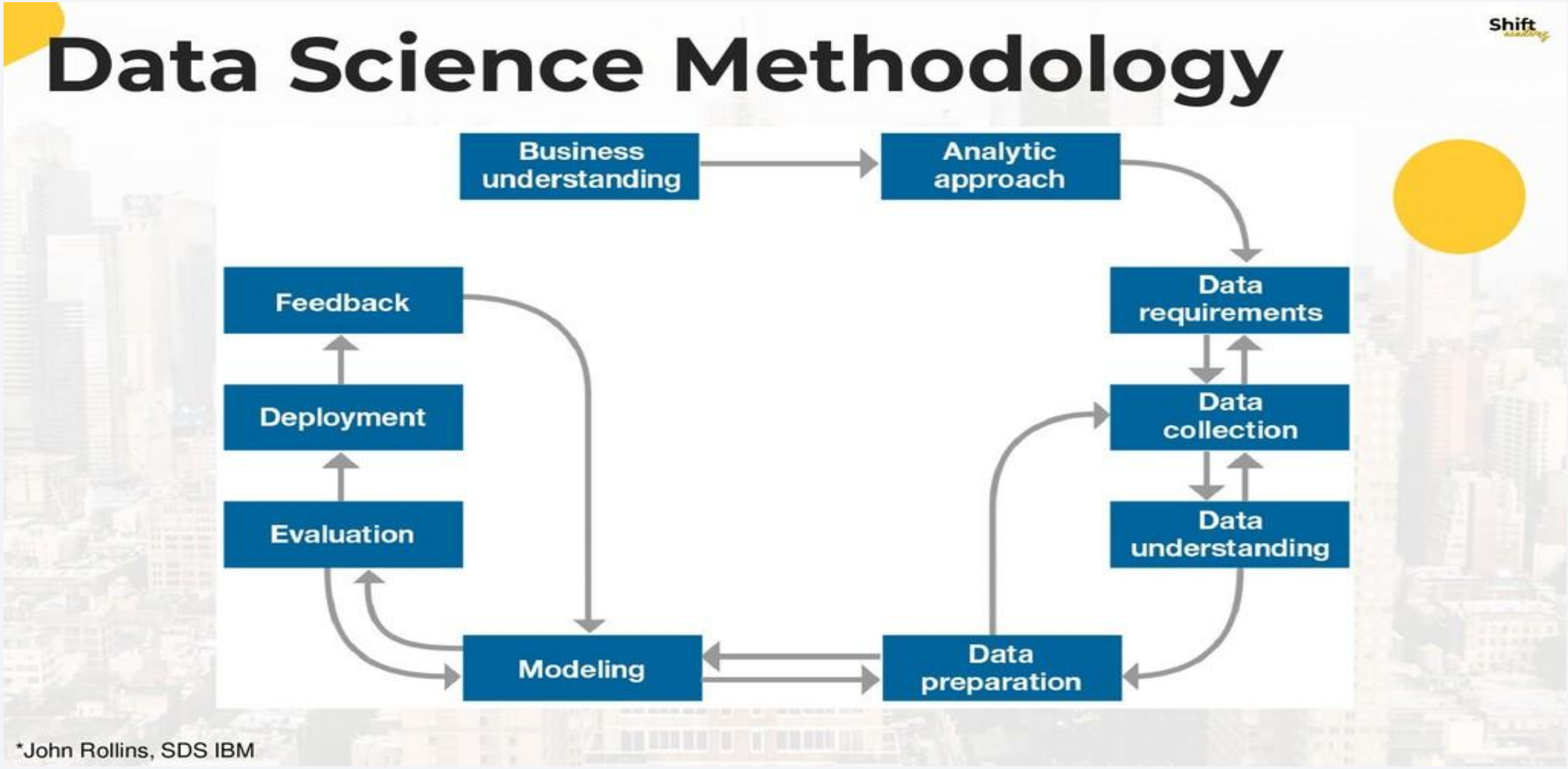
  By studying SpaceX Launch data, we will determine the factors leading to first stage reuse in SpaceX launches.

  -

Section 1

Methodology

Methodology



Methodology

The step-wise process followed was from IBM, as shown on the previous slide. This helped guide the following activities:

Data collection

- Data was retrieved from SpaceX using HTTP.
- Webscraping was also used to harvest data from a webpage

Data wrangling / Data preparation

- **Fill nulls with mean values**
- **Use one hot encoding to better analyze categorical variables**

Data Understanding

- Exploratory data analysis (EDA) using visualization and SQL
- visual analytics using Folium and Plotly Dash

Modeling

- Perform predictive analysis using classification models

Data Collection – SpaceX API

- Using an HTTP GET request, we gather the data from https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json
- We then convert the json object transferred from the REST api to a pandas dataframe for analysis.
  pd.json_normalize(json)

The notebook containing the code to gather the spacex data can be found here:
https://github.com/Coursera68/SpaceX/blob/main/jupyter-labs-spacex-data-collection-api-v2.ipynb

```
response = requests.get(static_json_url)
response.status_code
print("Response is of type ", type(response))
json = response.json()
print("Json is of type ", type(json))
```

```
Response is of type  <class 'requests.models.Response'>
Json is of type  <class 'list'>
```

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head(5)
```

Data Collection - Scraping

- The data was also harvested from wikipedia by using beautifulsoup, giving us the ability to get data from an html table.

- The url to the notebook can be found here:https://github.com/Coursera68/SpaceX/blob/main/jupyter-labs-webscraping.ipynb

```
# 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')
print("html_tables is of type :", type(html_tables))
```

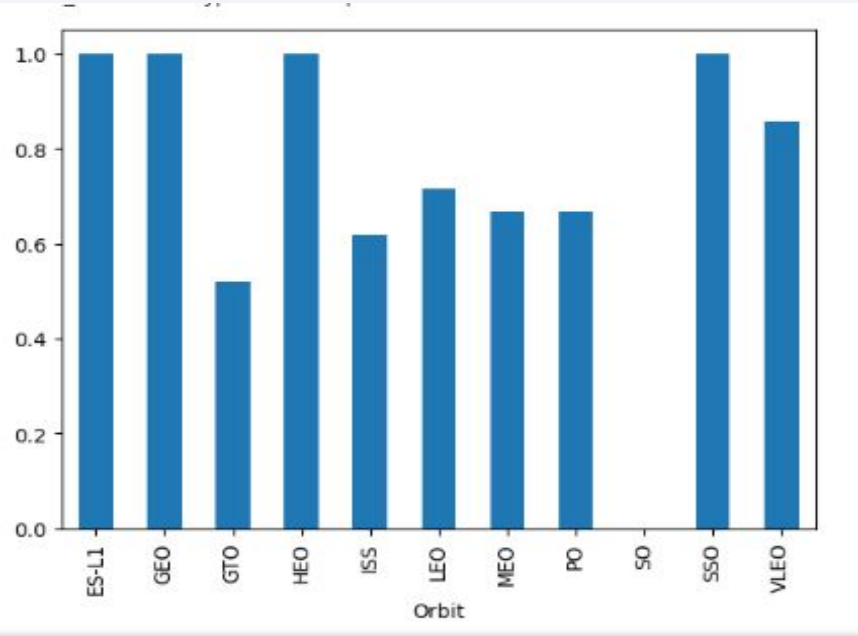
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
```

Data Wrangling

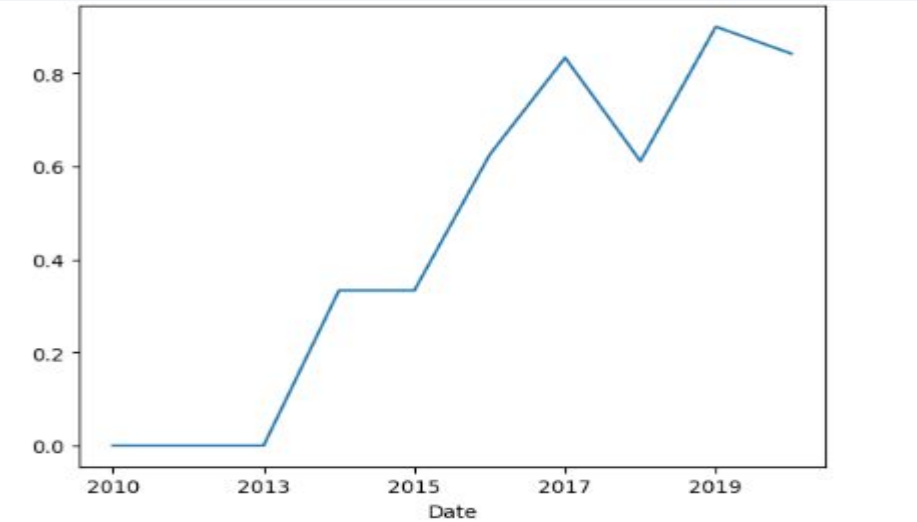
- In this effort, we started to understand our data. We took an earlier csv file from data collection and converted to a pandas dataframe.
- Using the value_counts() method on the outcome column of the dataframe, we got a series of the landing outcomes
- Using this series, generated a list of bad outcomes.
- Finally, we were able to create a new binary variable showing whether the landing was a success or failure. (1 or 0) This new variable is key to our mission: determining success landings,thus reusing the first stage.
- The link to this notebook is located at: https://github.com/Coursera68/SpaceX/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- Data visualization is another great way to understand the data and the relationships between different data attributes. Here is an example of successes based on the type of orbit.



- The below graph shows the rising success rate over the launch years.



- The complete notebook can be found here:
  https://github.com/Coursera68/SpaceX/blob/main/edadataviz.ipynb

EDA with SQL

- Here are some sql queries used to better understand the data
  - boosters carrying the maximum payload:

    %sql select booster_version from spacextable where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextable)

- Landing outcomes in descending order of their total occurrences

  %sql select landing_outcome, count(*) from spacextable group by landing_outcome order by count(*) desc

- The average payload weight for the various launches:

  %sql select avg(PAYLOAD_MASS__KG_) from spacextable

- Here is a url to the notebook:

https://github.com/Coursera68/SpaceX/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Folium allows you to display an interactive map to visualize where the launches occur. The map contains a “marker” showing the launch sites, obviously near coastal areas to facilitate water landings and avoid population centers.

- The map contains marker clusters of each launch sites. The markers are color codes to show a successful or failed launch.

The notebook containing the code for the folium maps can be found here:
https://github.com/Coursera68/SpaceX/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- In this dashboard, we added two main interactive graphs. One graph is a pie chart that shows the percentage of successful flights for each location. When you drill down to a specific site, you see success vs failur for that site.
-
- We also added a scatterplot with a range finder to be able to understand success rates at various payloads.
- We added the chosen graphs to our dashboard to answer the following questions :
  - Which site has the largest successful launches?
  - Which site has the highest launch success rate?
  - Which payload range(s) has the highest launch success rate?
  - Which payload range(s) has the lowest launch success rate?
  - Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate?
- The code is found here: (Can copy out of wordpad to get rid of formatting)

  https://github.com/Coursera68/SpaceX/blob/main/Dashboard_Code.rtf

Predictive Analysis (Classification)

- We first split the data into training and test data sets to evaluate model performance.
- We applied StandardScalar to the independent variables to ensure they all have the same scale so there magnitude does not skew the outcome.
- We used different classification models, (Decision Tree Classifier, Support Vector Machine, Logistic Regression, K Nearest Neighbor) to predict the successful landings.
-
- Finally, we looked at metrics to see which model performs the best, accuracy score. We can also see best hyperparameters, and the confusion matrix.

- Here is the url to the notebook for the analysis effort:

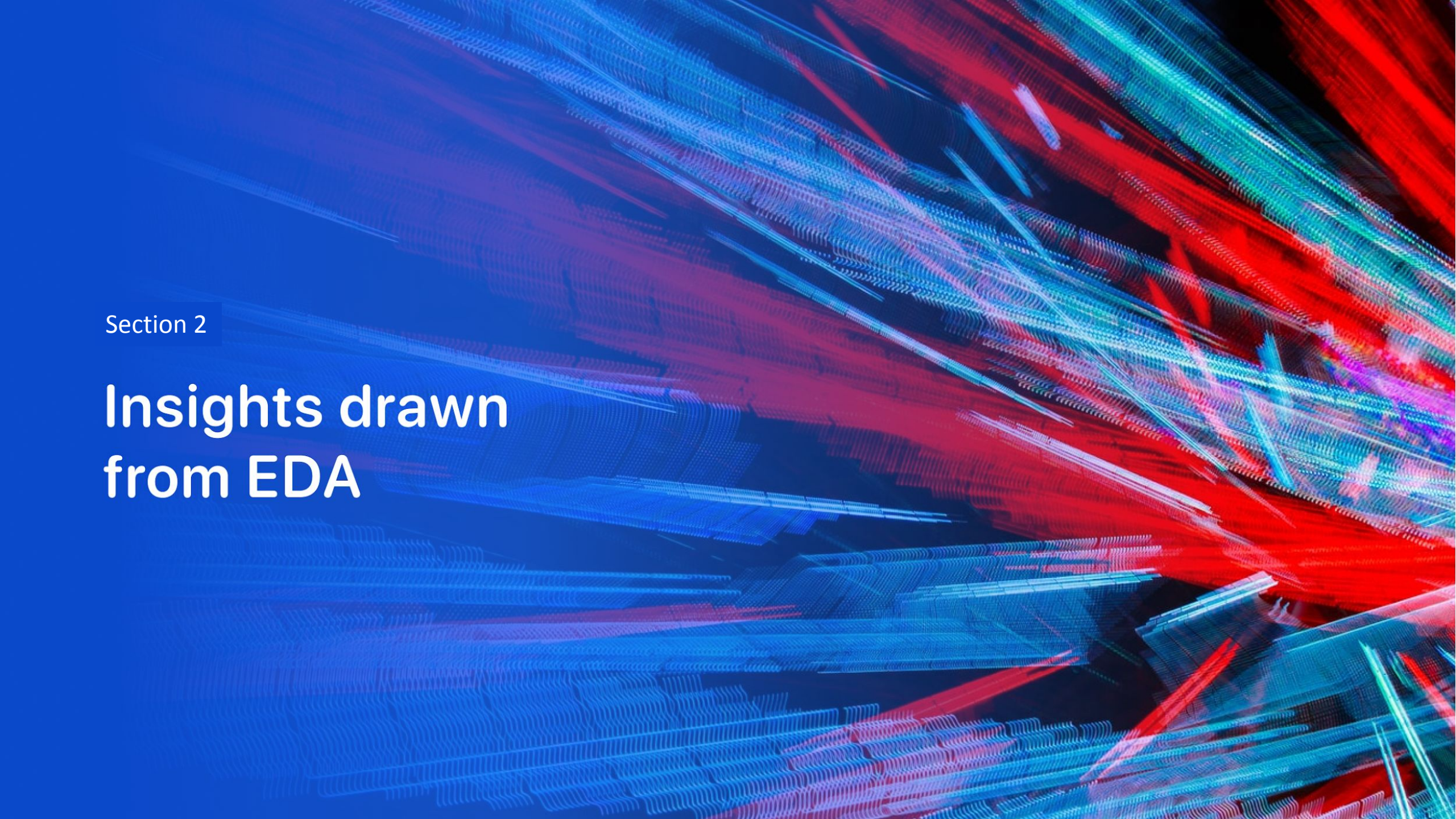
  https://github.com/Coursera68/SpaceX/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

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

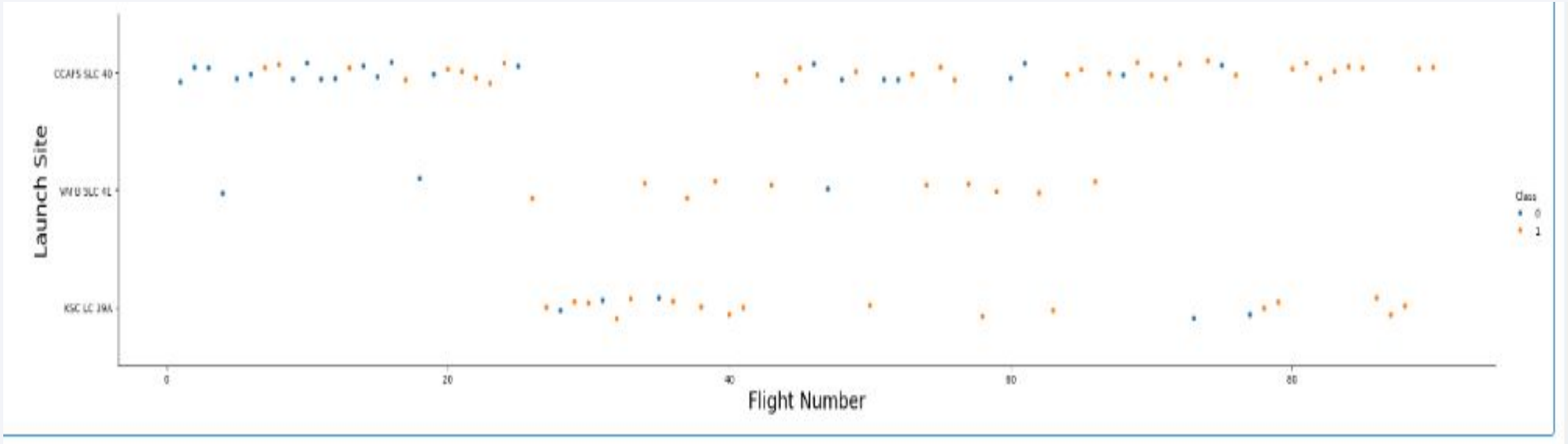
Section 2

Insights drawn from EDA



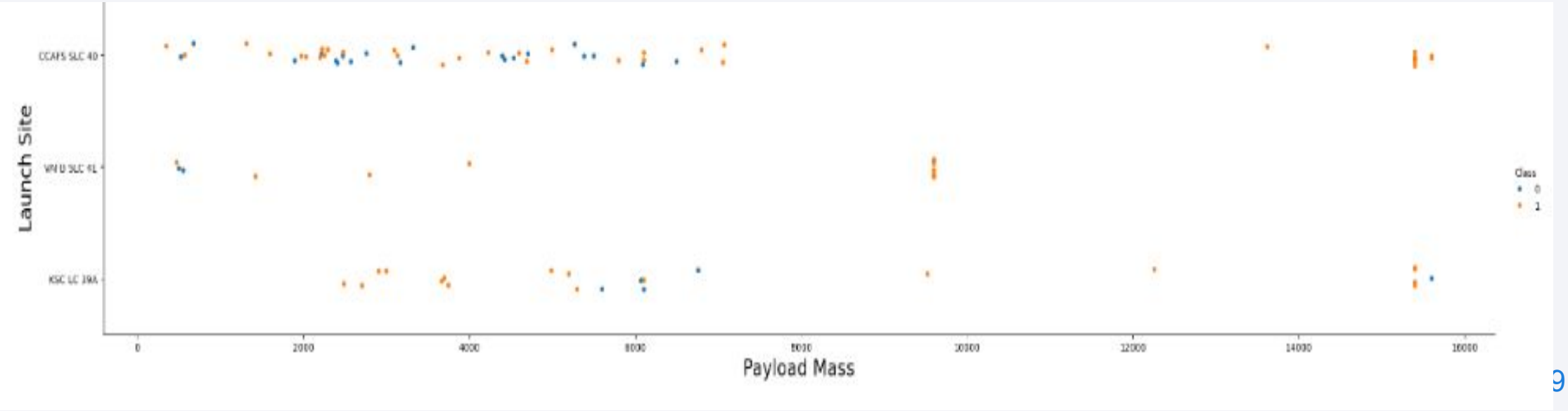
Flight Number vs. Launch Site

Looking at the chart of flight number and launch site, we can derive some insights. You can see the success of the launches improved as the flight number (hence time) went on. You can also see site CCAFS SLC 40 was clearly a favored launch site, known as the Cape Canaveral launch complex. Clearly the experience of all involved at the site is an asset.



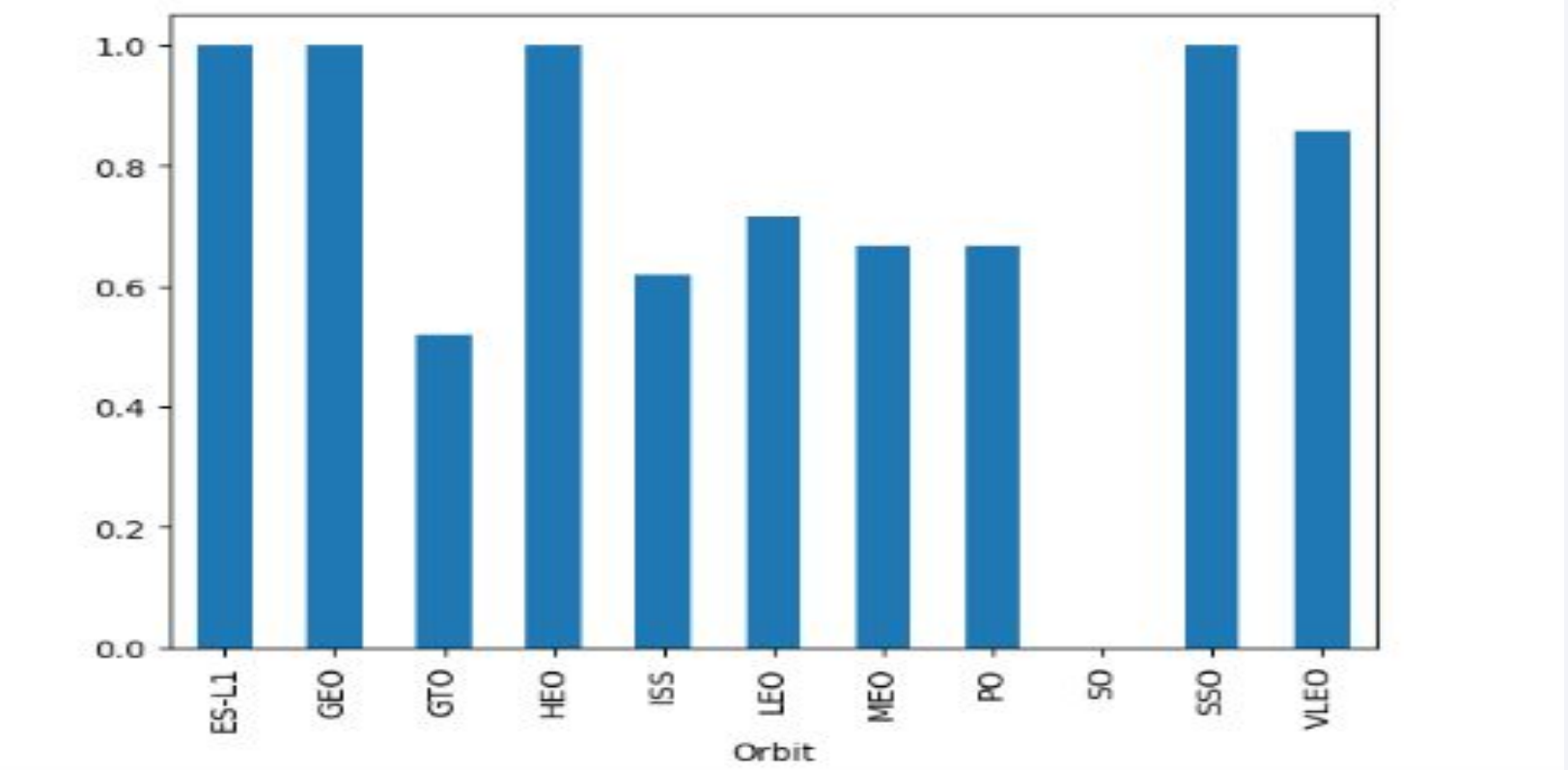
Payload vs. Launch Site

Here we see the sites with the largest payloads were launched from CCAFS SLC 40 and KSC LC39A, the Cape Canaveral Launch Complex and the Kennedy Space Center. This data may yet again say, Experience matters. The Florida launch centers may have needed infrastructure to support larger missions.



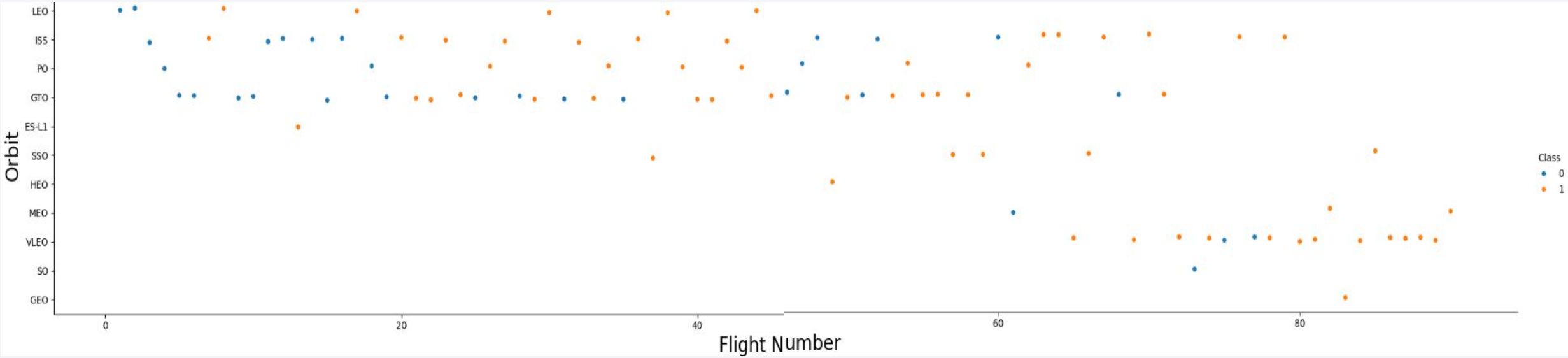
Success Rate vs. Orbit Type

From the graph, we see that four orbits had the highest success rates, indicating that choice of orbits may be a success factor.



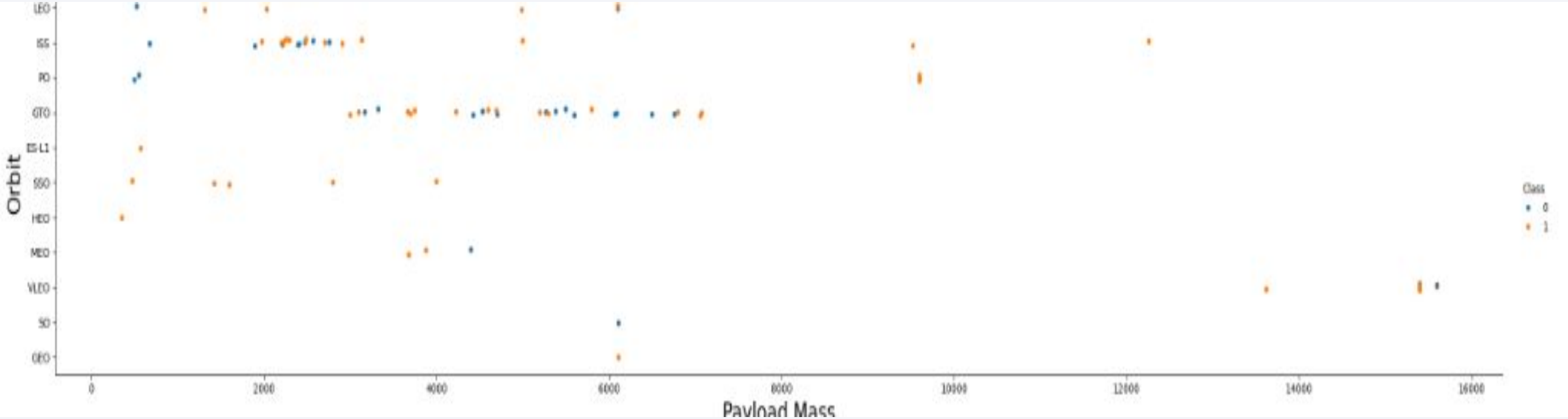
Flight Number vs. Orbit Type

From the graph we can see that four orbits are favored throughout all the launch sequences. In later flights, Very Low Earth orbit for some reason also became a popular choice.



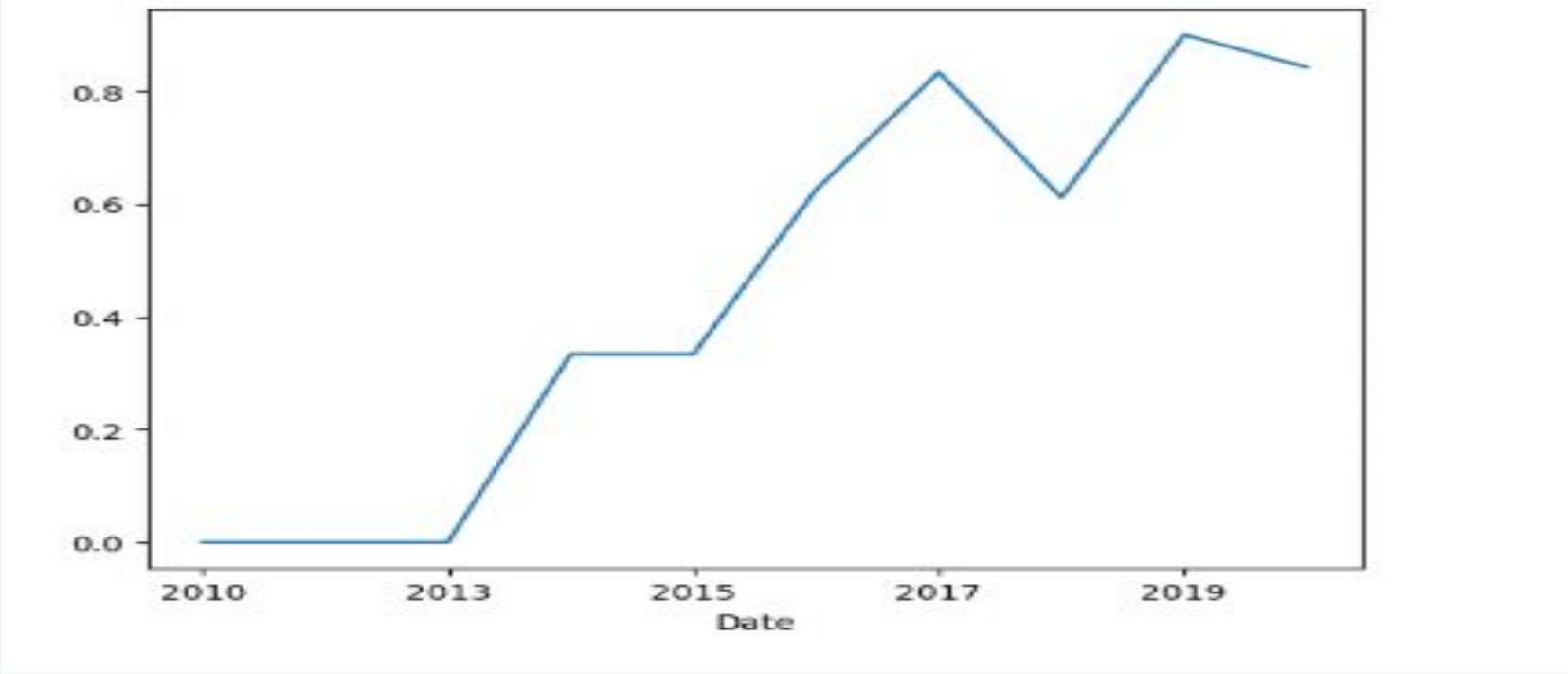
Payload vs. Orbit Type

- From the data, the highest payloads are in the ISS (International Space Station) and Very low earth orbits. Possibly cheaper to get most mass to lower orbits.



Launch Success Yearly Trend

- The yearly trend of launches shows successful launches are generally increasing over time.



All Launch Site Names

- Here is a query showing the launch sites. The distinct keyword only shows the unique launch sites.

```
In [12]: %sql select distinct(launch_site) from spacextable
 * sqlite:///my_data1.db
Done.
```

Out[12]:

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

Launch Site Names Begin with 'CCA'

Here’s an example of only launch records from the Cape Canaveral Launch complex, in case you wanted to study those specific launches.

In [22]:
```
%sql select * from spacextable where launch_site like 'CCA%' limit 7
```

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

Out[22]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parach
2010-12-08	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 (parach
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No atte
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No atte
2013-	15:10:00	F9 v1.0 B0007	CCAFS LC-	SpaceX	677	LEO	NASA	Success	No atte

Total Payload Mass

In this query, we show the payload mass delivered by the customer Nasa (CRS), a company that resupplies the International Space station.

```
%sql select customer, sum(PAYLOAD_MASS__KG_) from spacextable where customer = 'NASA (CRS)'
```

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

Customer	sum(PAYLOAD_MASS__KG_)
NASA (CRS)	45596

Average Payload Mass by F9 v1.1

In this query,we use an aggregate function - avg, to display the average payload carried by booster F9 v.1.1

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

In [26]:
```
%sql select avg(PAYLOAD_MASS__KG_) from spacextable where Booster_Version = 'F9 v1.1'
```

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

Out[26]:

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

Here we show the date of the first ground landing function by selecting the minimum landing date from the dataset of ground landings.

```
In [30]: %sql select min(date) from spacextable where landing_outcome like('%ground pad)')
 * sqlite:///my_data1.db
Done.
Out[30]: min(date)
         2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

By using the correct WHERE clause, we can filter the data to show only booster versions landing successfully on drone ship and having a weight between 4000 to 6000 kilograms.

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

In [37]:
```
%sql select booster_version from spacextable where landing_outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 400
```

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

Out[37]:

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

Total Number of Successful and Failure Mission Outcomes

By using the group by function, we can aggregate the count of various mission outcomes.

In [47]:
```
%sql select mission_outcome, count(*) from spacextable group by mission_outcome
```

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

Out[47]:

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

Boosters Carried Maximum Payload

Here we use a subquery to find the booster versions carrying the maximum payload. Obviously, the “F9 B5” family of boosters are quite a work horse.

In [48]:
```
%sql select booster_version from spacextable where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextable)
```

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

Out[48]:

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

2015 Launch Records

This query was supposed to show the 2015 failed landing outcomes… However, I started getting a system error when I go back to fix this problem. Error is: Keyerror: Default Put in a support ticket, but gave up.

```
In [49]: %sql select substr(Date,6,2), landing_outcome from spacextable
         where substr(Date,0,5) = '2015' and landing_outcome like ('Failure%')

  Cell In[49], line 2
    where substr(Date,0,5) = '2015' and landing_outcome like ('Failure%')
          ^
SyntaxError: invalid syntax
```

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

Here are the totals of landing outcomes obtained by using the group by function.

```
In [53]: %sql select landing_outcome, count(*) from spacextable group by landing_outcome order by count(*) desc
 * sqlite:///my_data1.db
Done.
```

Out[53]:

Landing_Outcome	count(*)
Success	38
No attempt	21
Success (drone ship)	14
Success (ground pad)	9
Failure (drone ship)	5
Controlled (ocean)	5
Failure	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1
No attempt	1

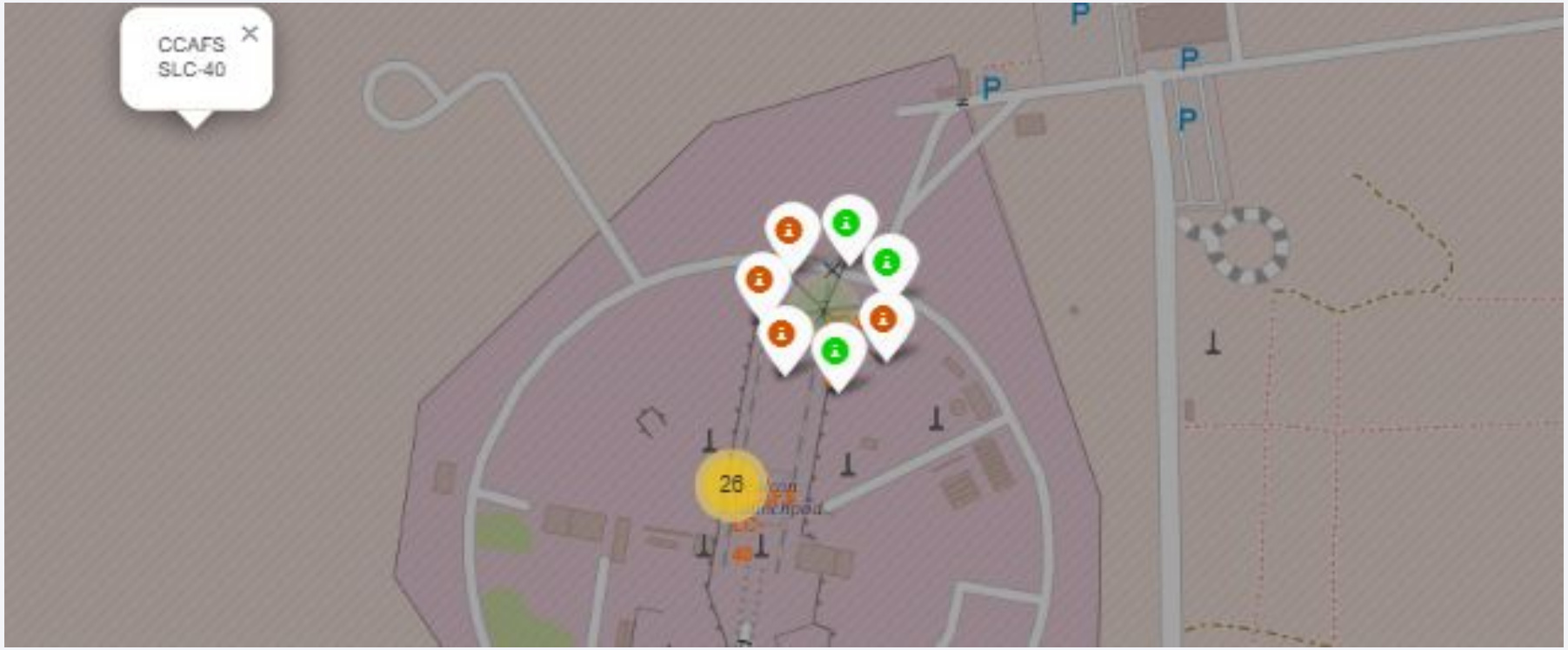

Section 3
Launch Sites
Proximities Analysis

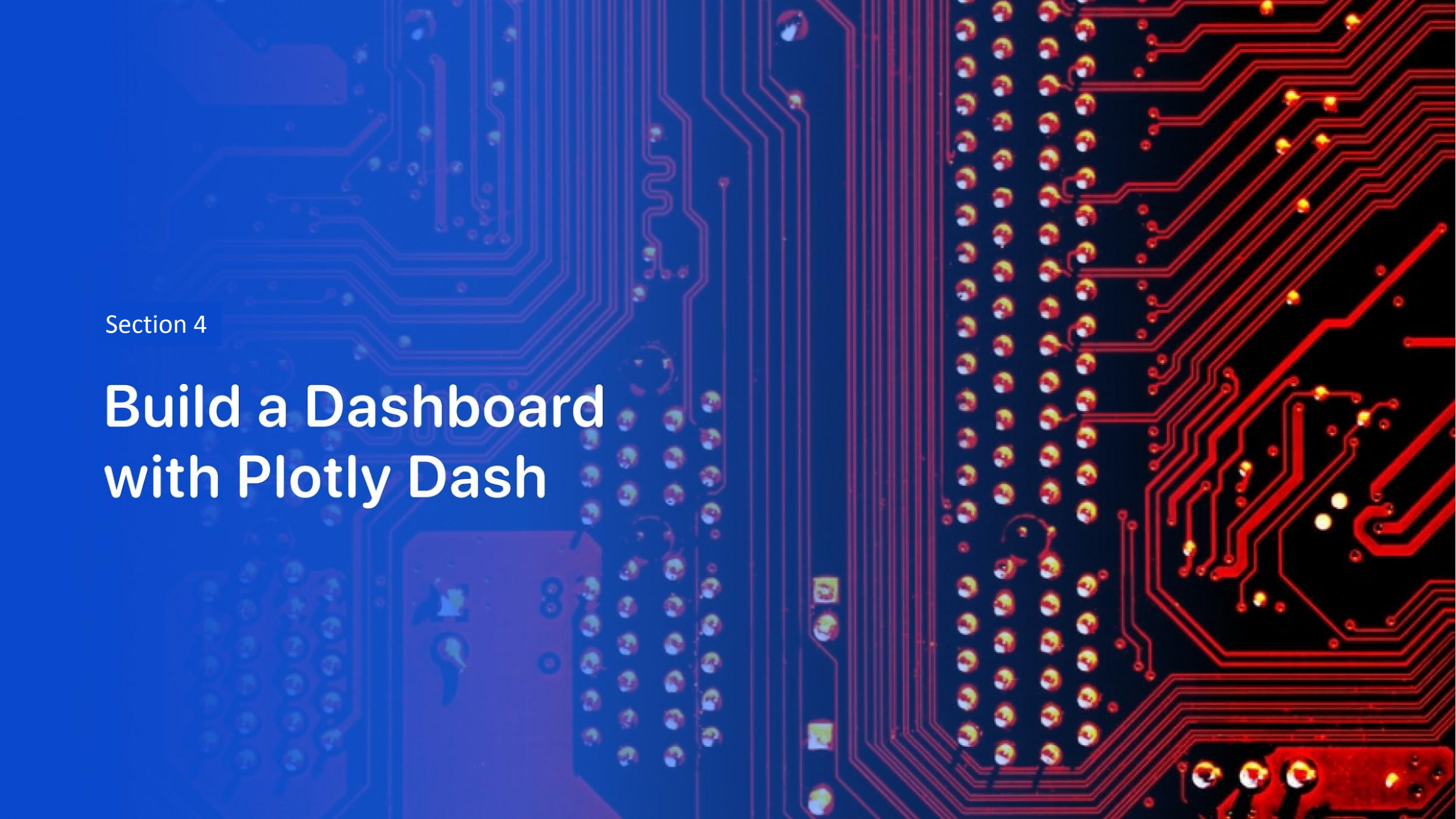
Launch Sites in the United States

Here we can see the SpaceX launch sites. Located on the coast of the United States in California and Florida. Being near the ocean is obviously necessary for water landings and to avoid population centers.

Launch Sites - Folium Capabilities

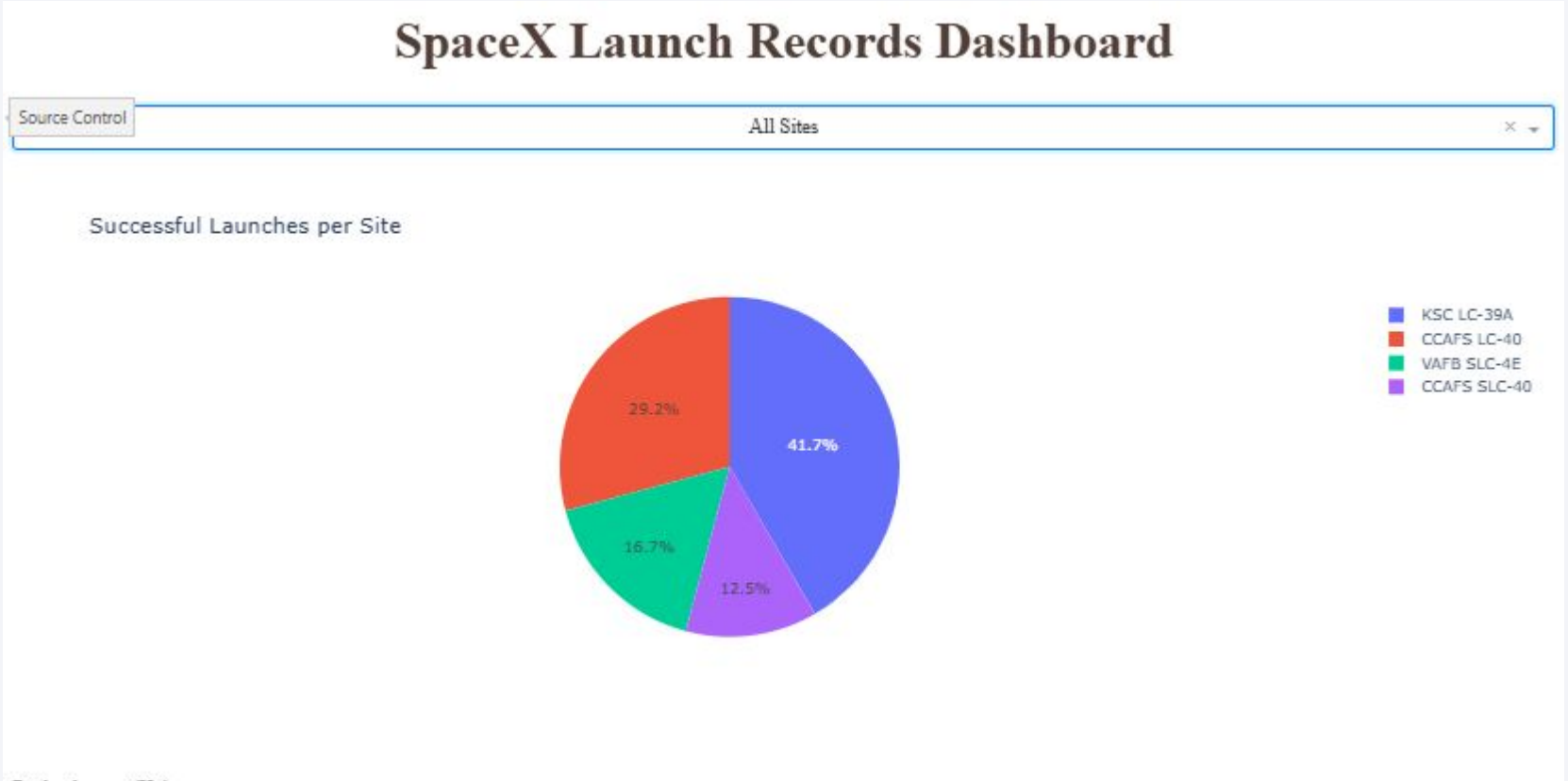
Here is a feature of folium that shows the ability to gain further understanding by expanding or “drilling down” into the map. Here is a particular launch site with marker cluster showing the successful (green) and failed (red) launches.




Section 4
Build a Dashboard with Plotly Dash

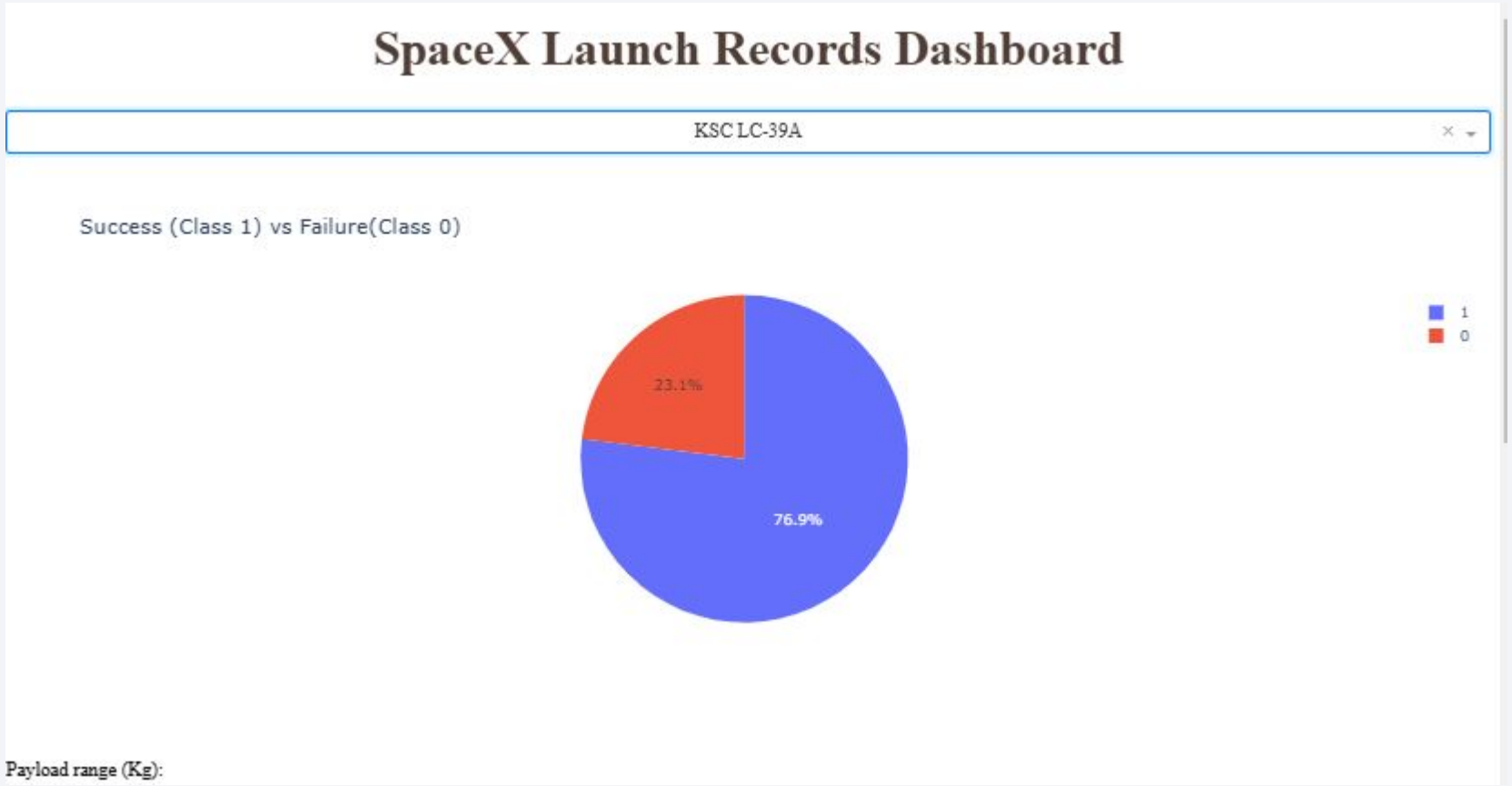
Dashboard with Ploty Dash - All Launches

In this dash board, we can see see the successful launch sites per site.
Perhaps certain sites have more favorable features than others.



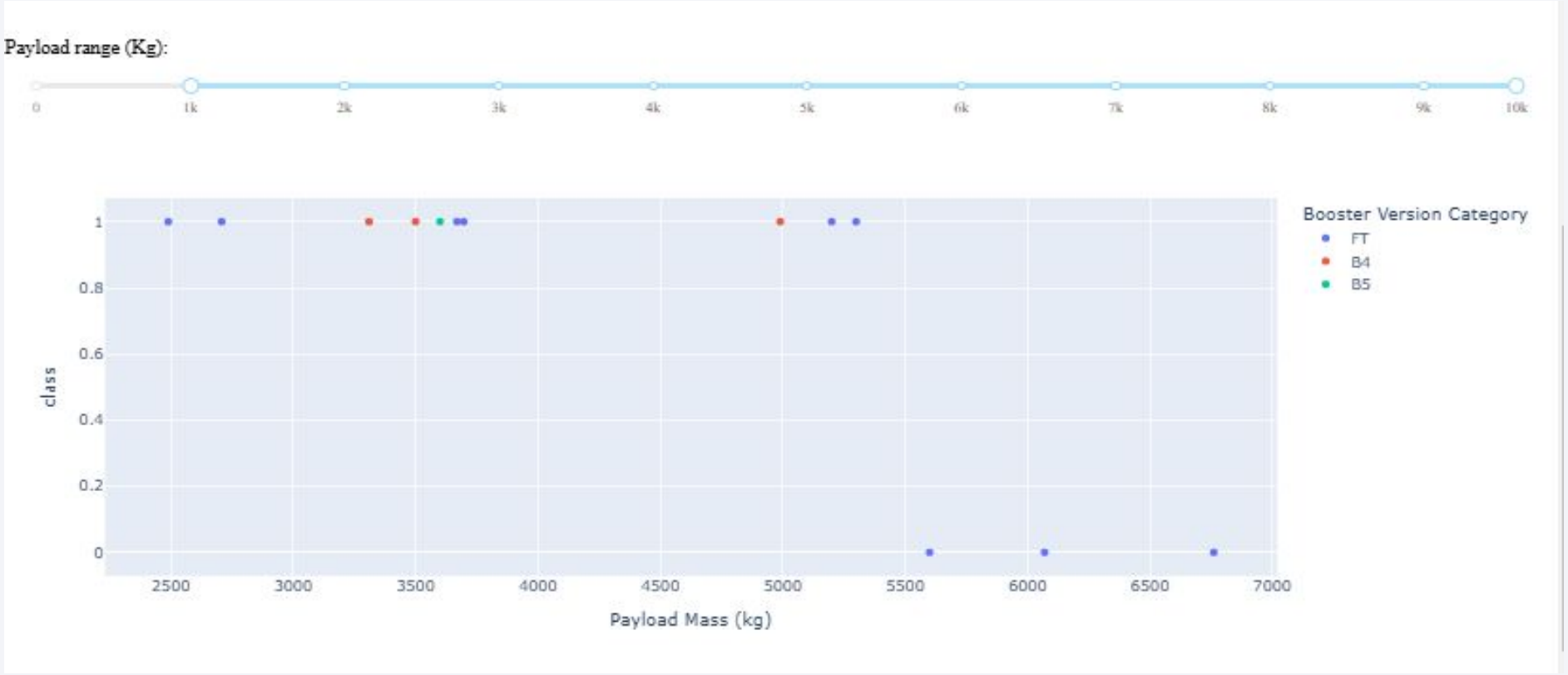
Dashboard with Ploty Dash - Site Launches

Here we can see the success and failures of the launches at a particular site.



Dashboard with Ploty Dash - Range Slider

- This dashboard utilizes a range slider. By customizing the range of payload masses, you can see the corresponding boosters used in those launches. From this data, we see the “FT” category of boosters was the heavy lifter..



Section 5

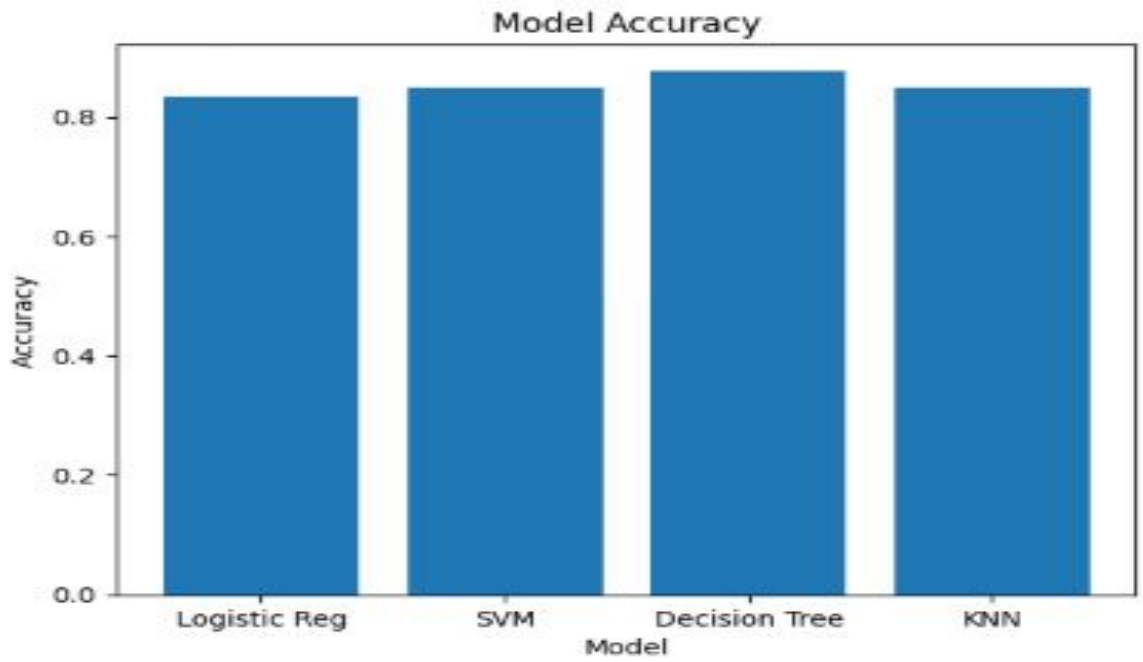
Predictive Analysis (Classification)

Classification Accuracy

- Here is a bar chart showing the model accuracy scores for the models.

```
[36]: #show a bar chart of the models accuracy

plt.bar(models.keys(),models.values())
plt.title('Model Accuracy')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.show()
```

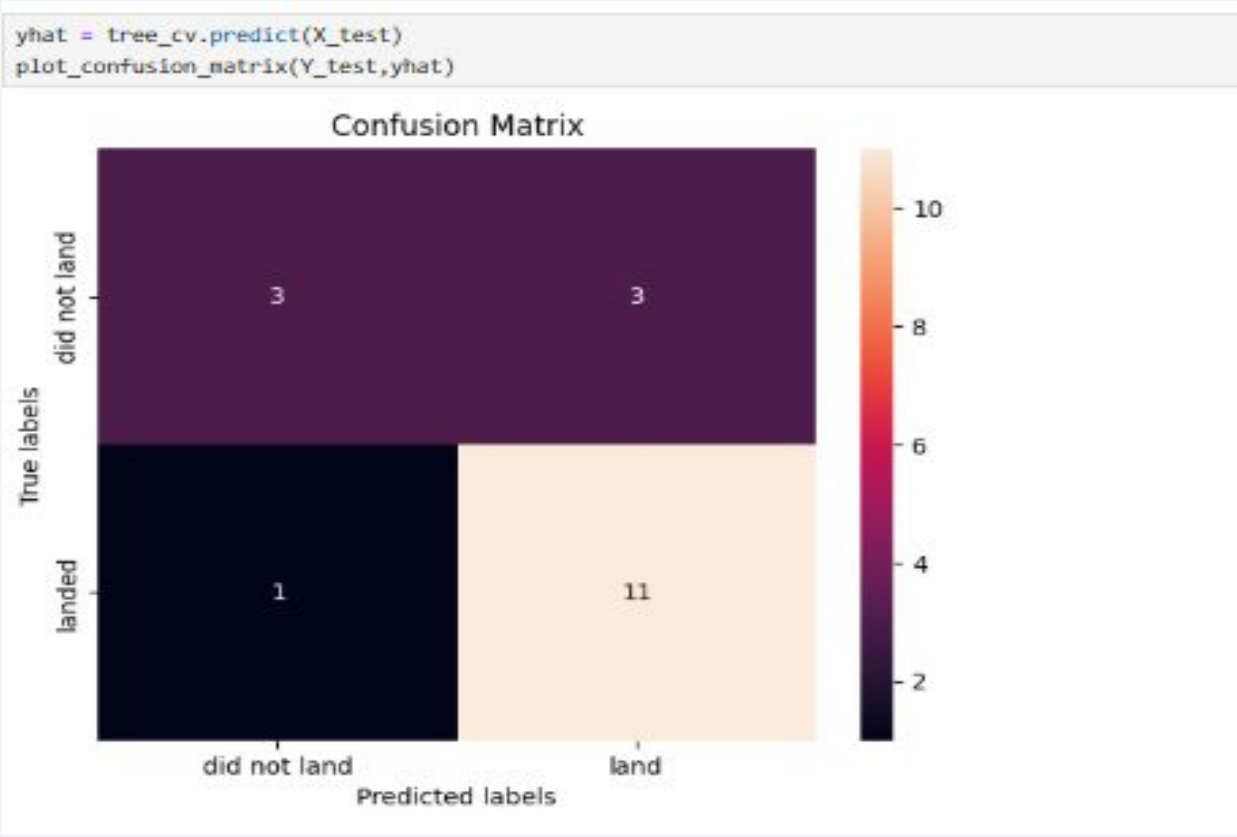


As you can see, the decision tree classifier model has the best accuracy with a score of .875

Confusion Matrix - Decision Tree Model

The decision tree confusion matrix shows why the model has the best accuracy. Basically, what actually happened vs what the model predicted was better than the other models.

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- The most promising conclusion to be had is gained from looking at the chart showing the growing success of launches from 2013 to 2020. No obvious road blocks preventing success.
- We need to pay attention to booster versions, as some appear more successful and can carry more payload. Should be an area of detailed study.
- The east coast launch complexes appear to be the most used / successful. Utilize experience (Nasa) and take advantage of great facilities.
- Some orbits are more successful to others. Should be an area of more detailed study.

Thank you!