IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Amogha Subramanya
11/02/2024

Outline

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

Executive Summary

- This report aims to predict if the first stage of the SpaceX Falcon 9 rocket will land or not for the launch of payload.
- This information can be used to predict the cost of launch of a given payload; information that can be key to competing companies placing a bid for a rocket launch.
- The dataset required to predict the outcome is collected from SpaceX API and web scraping; data is then cleaned to facilitate building of models to predict the outcome of launch of a future payload of given characteristics and intended orbit.
- The prediction is a ‘classification’ problem, hence logistic regression, decision tree and SVM models are trained and tested to make predictions about future launches.
- All three models were able to predict with equal accuracy.

Introduction

- SpaceX can offer rocket launches at a significantly lower cost than competitors as they are able to land and reuse the Falcon 9 rocket.
- Here are some examples the Falcon 9 landing back after a launch:

Introduction

- However, depending on the payload characteristics and the orbit it is being sent to – the Falcon 9 may not be able to return and make a successful landing.
- Most unsuccessful landings are planned and performed in the ocean. Below are some examples:

- Knowing if SpaceX will be able to land the stage back successfully or not can be used to predict the cost of a launch.
- Information that can be used by competitors while placing a bid for rocket launches.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Required data collected with SpaceX API and web scraping Wikipedia with Beautiful Soup.
- Perform data wrangling
  - Collected data was cleaned and missing data were handled to make it complete.
  - Prepared data for training by one hot encoding the categorical variables to allow for training of ML models
- 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

Methodology

- Perform predictive analysis using classification models
  - Being a classification problem – Logistic Regression, Decision Trees and Support Vector Machine (SVM) ML models are chosen.
  - The data is split into training and testing samples – the models are trained using the training data and we verify their accuracy of predictions using the testing data.
  - The best performing model is than re-trained using all the data.

Data Collection

Dataset about past launches are collected utilizing the SpaceX API.

The ‘requests’ library in python is used to get this information in a json format from the SpaceX API which is then normalized and stored in a pandas Dataframe.

We get information about the past launches:

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
data = pd.json_normalize(response.json())
data.head(3)
```

✓ 0.3s Python

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	sh
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	
							[{'time': 301, 'altitude':	Successful first stage burn and transition to second stage, maximum		

- A lot of information that we need are ids, next we make separate API calls to get

Data Collection – SpaceX API

- From the rocket id we would like to learn the booster name.

```
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

- From the payload id we would like to learn the mass of the payload and the orbit that it is going to.

```
def getPayloadData(data):
    for load in data['payloads']:
      if load:
       response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
       PayloadMass.append(response['mass_kg'])
       Orbit.append(response['orbit'])
```

- From the launchpad id we would like to know the name of the launch site being used, the longitude, and the latitude.

```
def getLaunchSite(data):
    for x in data['launchpad']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
            Longitude.append(response['longitude'])
            Latitude.append(response['latitude'])
            LaunchSite.append(response['name'])
```

- From cores id we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, whether the core is reused, whether legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

```
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Data Collection - Scraping

- Data is collected from a table available on Wikipedia
- We make use of python library BeautifulSoup to accomplish this
- The data is extracted from the html table and is then saved to a dictionary
- The dictionary is converted to a dataframe and saved to a csv file.

Get html data with requests library

Parse html data with BeautifulSoup

Iterate through each row of the table

Iterate through each element in the row and store cleaned data to a dictionary

Convert dictionary to a pandas DataFrame

Save to .csv file

Data Wrangling

- The ‘Outcome’ column has 8 categorical labels describing the mission outcome.
- There are four labels signifying a successful mission and four labels representing a failed mission
- These eight labels are converted to labels 0 or 1 (signifying failure or success) and is then appended under new column ‘Class’ of the dataframe.
- GitHub Repository: https://github.com/AmoMTL/SpaceXLandingPrediction

EDA with Data Visualization

- Plotted a scatter plot between Payload and the Launch Site used to see if there is a relationship between the two.
- Created a Bar Chart to check the relationship between the success rate and orbit type of the launch.
- Plotted a scatter plot between Flight Number and Orbit Type to see if there is a relationship between the two on mission outcome.
- Plotted a scatter plot between the Payload and Orbit type to see if there is a relationship between the two on mission outcome.
- Plotted the launch success yearly trend to see how mission outcome has varied over the years.
- GitHub Repository: https://github.com/AmoMTL/SpaceXLandingPrediction

EDA with SQL

- SQL query to display the unique launch sites in the space mission.
- SQL query to display the total payload mass launched by customer ‘NASA (CRS)’
- SQL query to find the average playload mass carried by booster version ‘F9 v1.1’.
- SQL query to find the date of the first successful landing outcome
- SQL query to find the booster which have successful landing on drone ship for a payload mass between 4000-6000kg
- SQL query to find the total number of successful and unsuccessful mission outcomes.

EDA with SQL

- SQL query to find the booster versions that have carried the maximum payload mass.
- SQL query to 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.¶
- SQL query to 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.
- GitHub Repository: https://github.com/AmoMTL/SpaceXLandingPrediction

Build an Interactive Map with Folium

- Marked the Nasa Johnson Space Center and all the SpaceX launch sites on a map – allowing us to determine their proximity to the equator and other features such as highways, train tracks etc.
- Marked the success and failed launches for each launch site on the map by using a marker cluster.
- Determined the launch sites with the highest successful launches – 'KSC LC-39A'
- GitHub Repository: https://github.com/AmoMTL/SpaceXLandingPrediction

Build a Dashboard with Plotly Dash

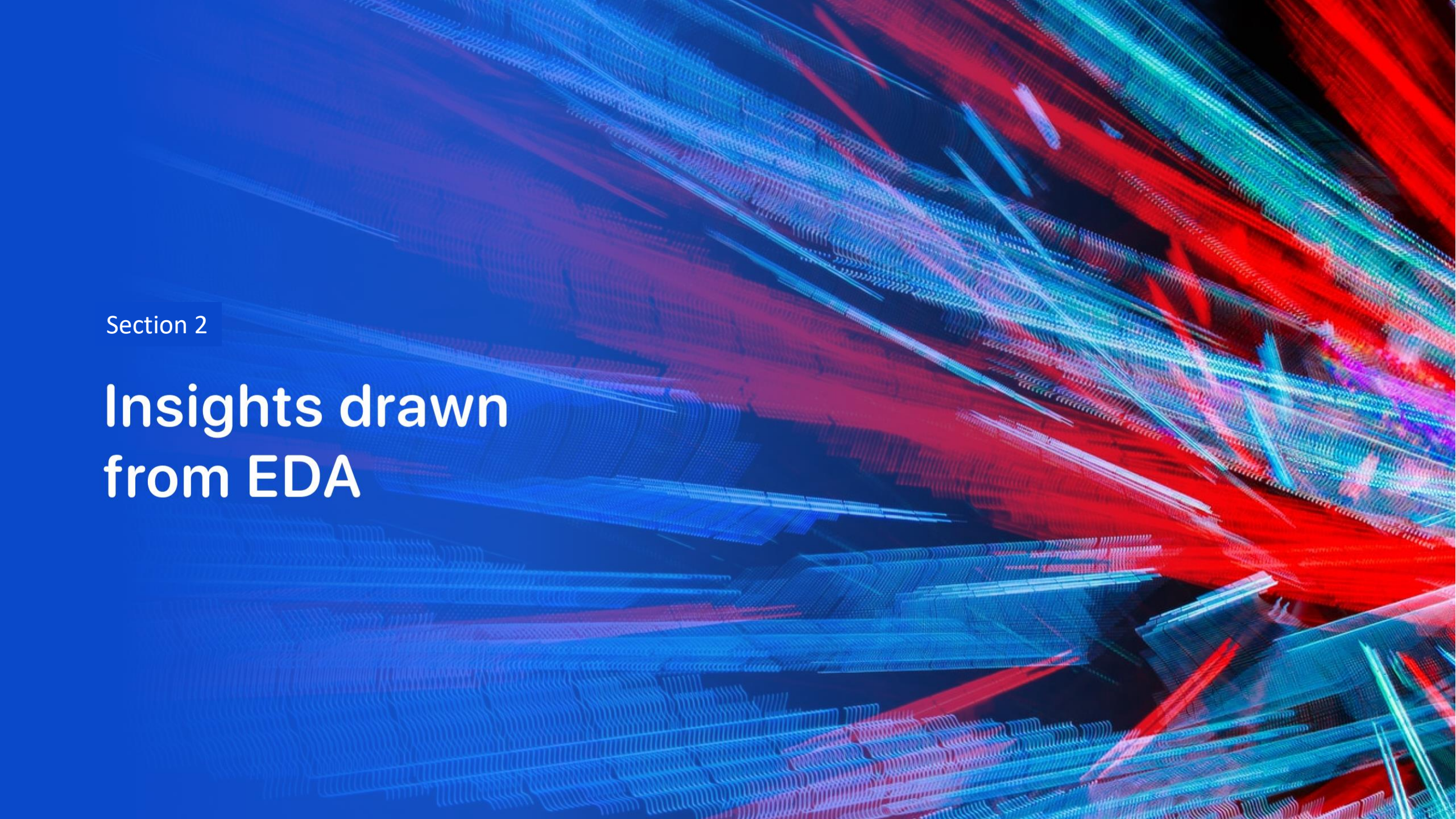
- Created a dash app to display the following graphs for a selected launch site:
  - Pie chart showing successful and unsuccessful mission outcomes
  - Scatter plot between payload mass and the mission outcome for different booster version
- The user is able to select the range of the payload through a slider and the particular launch site for which to see the graphs through a dropdown menu.
- The dash allowed to easily visualize the data for different payload and launch site conditions, without having to write code for each scenario
- GitHub Repository: https://github.com/AmoMTL/SpaceXLandingPrediction

Predictive Analysis (Classification)

- The dataset was split into a training and testing subset – the models would be trained on 80% of the data and the accuracy of the predictions would then be compared with the 20% of the test data.
- Being a classification problem, the ML models chosen to predict the outcome are – Logistic Regression, Decision Trees and SVM. The models are trained using the sklearn library on python.
- The model with the highest accuracy is chosen and then retrained to include all of the data (training and testing data)
- GitHub Repository: https://github.com/AmoMTL/SpaceXLandingPrediction

Results

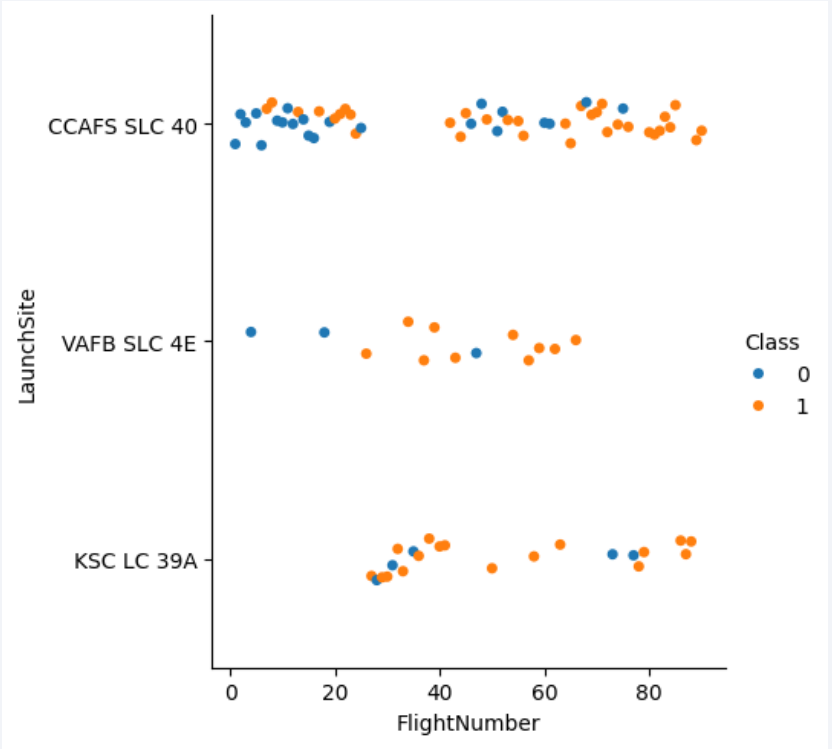
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

- Scatter plot of Flight Number vs. Launch Site

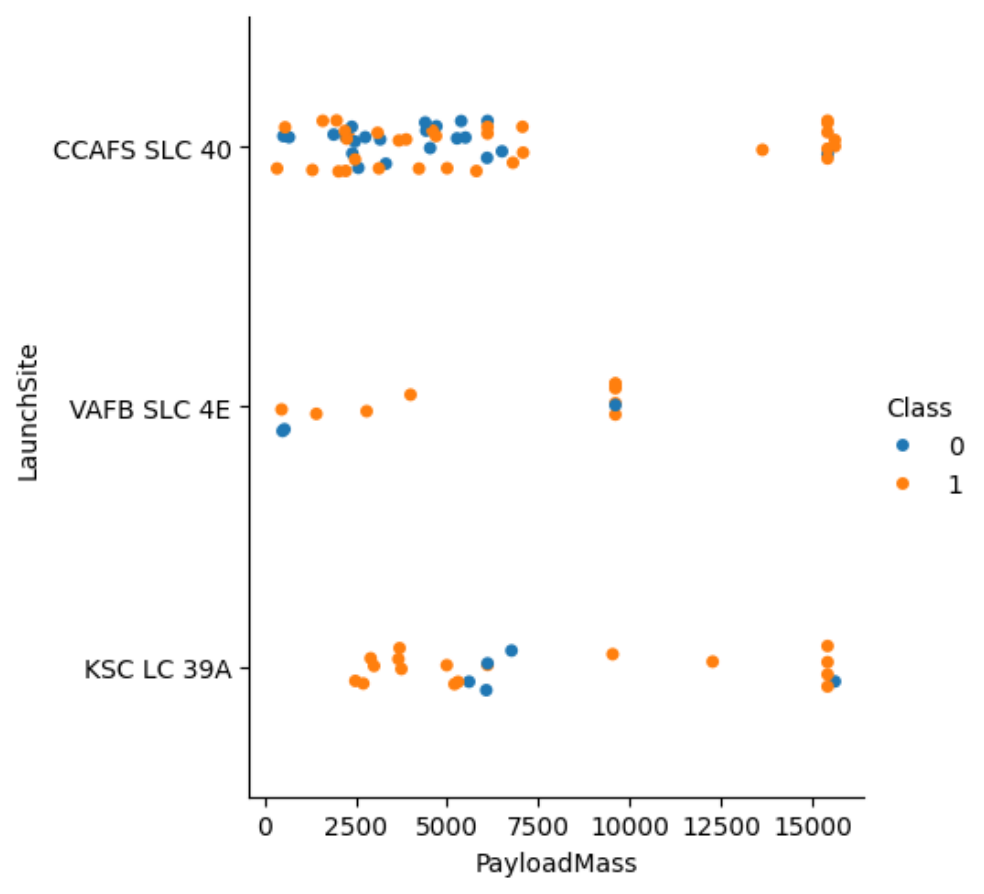
- We see that the most used sites are the CCAFS SLC 40 and KSC LC 39A.
- CCAFS SLC 40 has had a large number of unsuccessful missions.



Payload vs. Launch Site

- Scatter plot of Payload vs. Launch Site

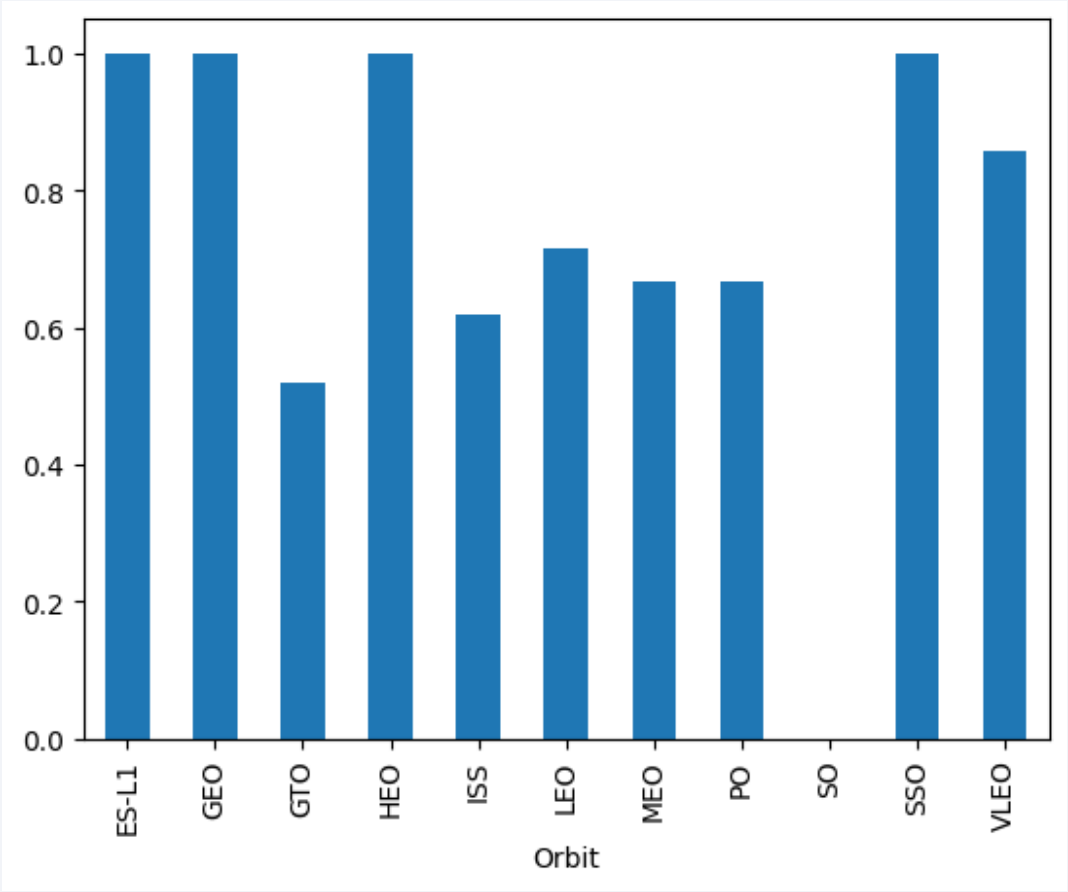
- There are no heavy payload launches from launchsite VAFB SLC 4E.
- The most number of unsuccessful missions were of payload mass less than 7500 kg



Success Rate vs. Orbit Type

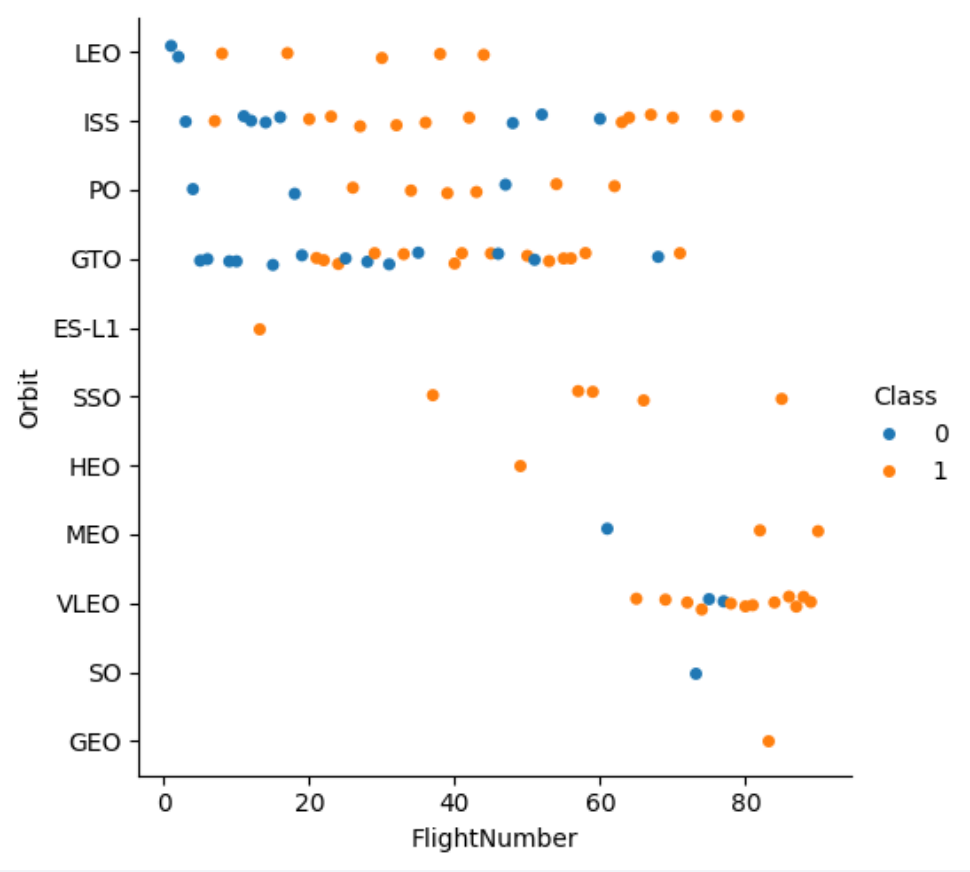
- Bar chart for the success rate of each orbit type

- ES-L1, GEO, HEO and SSO orbits have the highest successful missions.
- GTO orbits has the least number of successful missions.



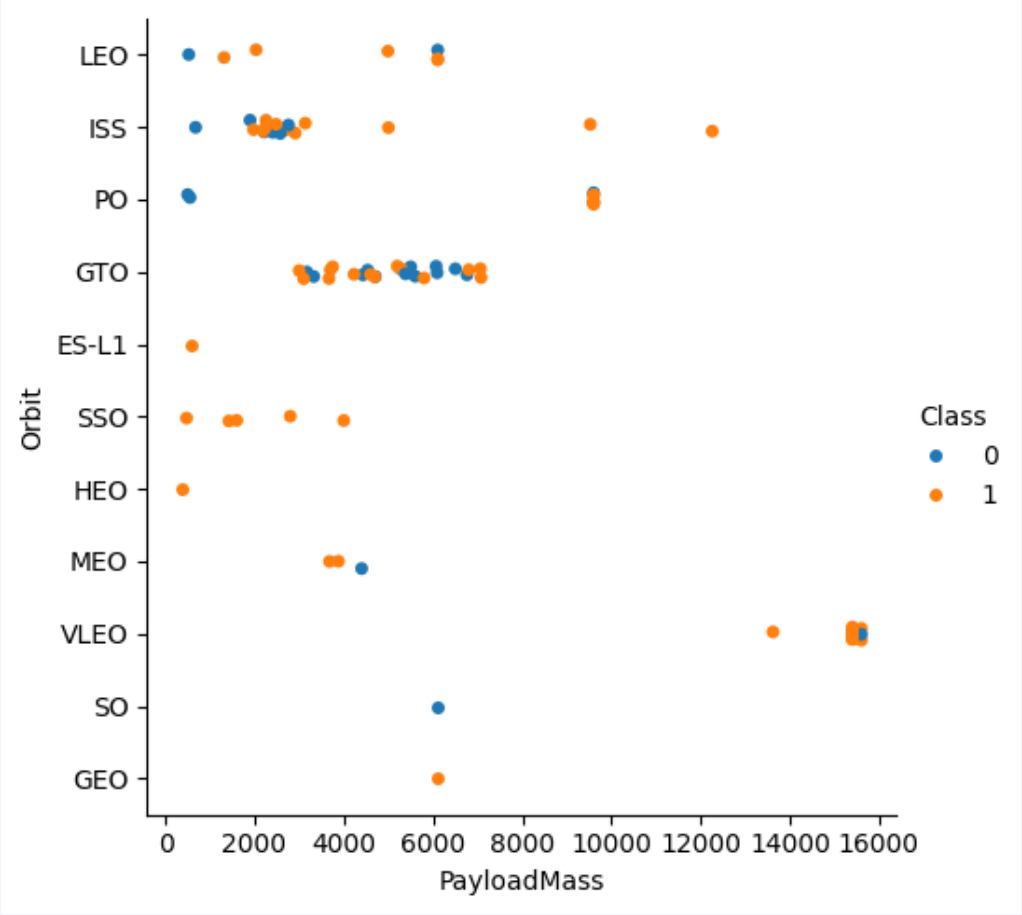
Flight Number vs. Orbit Type

- Scatter plot of Flight number vs. Orbit type
- 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.



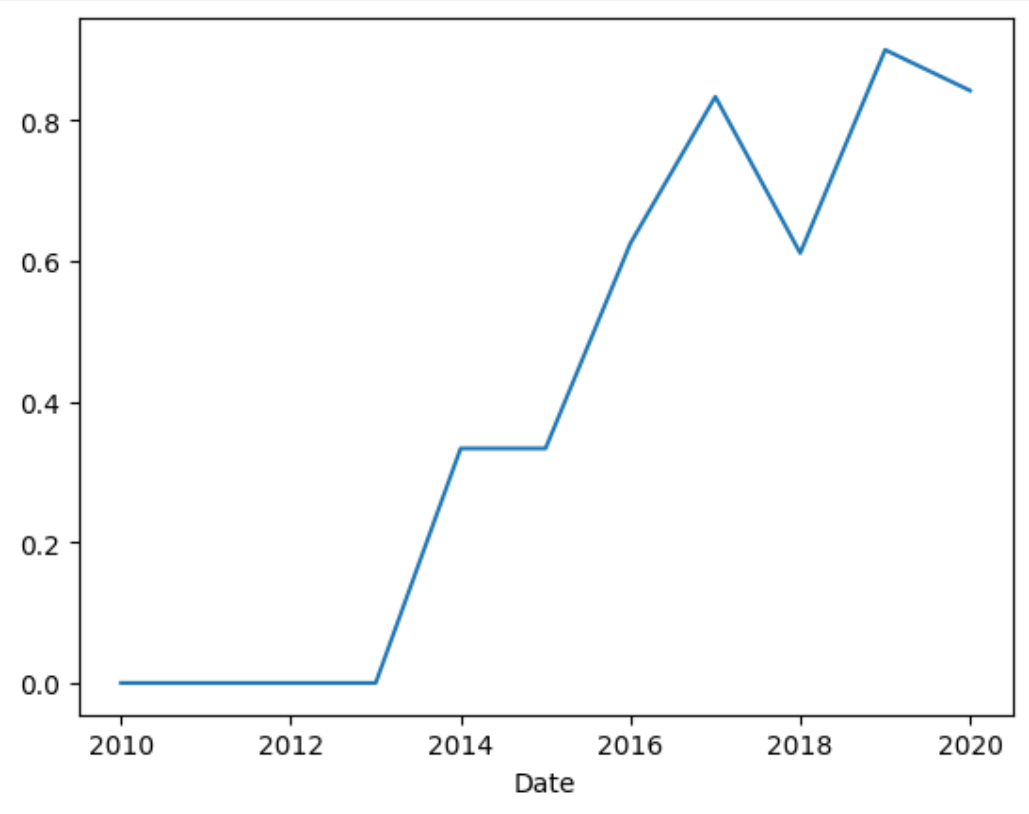
Payload vs. Orbit Type

- Scatter plot of payload vs. orbit type
- Heavy payloads the successful landing rate are more for Polar, LEO and ISS.
- For GTO we cannot distinguish this well as both positive landing rate and negative landing are both there here.



Launch Success Yearly Trend

- Line chart of yearly average success rate
- The success rate from 2013 kept increasing till 2017 (stable in 2014) and after 2015 it increased again.
- There is a dip in 2018.



All Launch Site Names

```
%%sql
SELECT DISTINCT Launch_Site FROM SPACEXTABLE
```

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

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

Launch Site Names Begin with 'CCA'

```
%%sql
SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	M
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	
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	
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	

Total Payload Mass

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE '%NASA%';
```

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

SUM(PAYLOAD_MASS__KG_)
107010

Average Payload Mass by F9 v1.1

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_MASS_KG FROM SPACEXTABLE WHERE Booster_Version LIKE '%F9 v1.1%'
```

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

AVG_PAYLOAD_MASS_KG
2534.6666666666665

First Successful Ground Landing Date

```
%%sql
SELECT MIN(Date) AS First_Ground_Pad_Landing FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (ground pad)';
```

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

First_Ground_Pad_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (drone ship)' AND PAYLOAD_MASS__KG_>4000 AND PAYLOAD_MASS__KG_<6000;
```

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

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

Total Number of Successful and Failure Mission Outcomes

```
success = %sql SELECT count(Landing_Outcome) AS SUCCESS FROM spacextable WHERE Landing_Outcome L
fail = %sql SELECT COUNT(Landing_Outcome) AS FAILURE FROM SPACEXTABLE WHERE Landing_Outcome LIKE
success = success[0]['SUCCESS']
fail = fail[0]['FAILURE']
print(success,fail)
```

```
 * sqlite:///my_data1.db
Done.
 * sqlite:///my_data1.db
Done.
61 10
```

Boosters Carried Maximum Payload

```
%%sql
SELECT  DISTINCT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_=(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

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

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

```
%%sql

 SELECT CASE SUBSTR(Date,6,2)
        WHEN '01' THEN 'JAN'
        WHEN '02' THEN 'FEB'
        WHEN '03' THEN 'MAR'
        WHEN '04' THEN 'APR'
        WHEN '05' THEN 'MAY'
        WHEN '06' THEN 'JUN'
        WHEN '07' THEN 'JUL'
        WHEN '08' THEN 'AUG'
        WHEN '09' THEN 'SEP'
        WHEN '10' THEN 'OCT'
        WHEN '11' THEN 'NOV'
        WHEN '12' THEN 'DEC'
    END AS MONTH, Landing_Outcome, Booster_Version, Launch_Site
    FROM SPACEXTABLE
    WHERE SUBSTR(Date,1,4)='2015' AND Landing_Outcome LIKE 'Failure (drone ship)'
```

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

MONTH	Landing_Outcome	Booster_Version	Launch_Site
JAN	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
APR	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

```
%%sql
SELECT Landing_Outcome, COUNT(Landing_Outcome) AS OCC FROM SPACEXTABLE WHERE Date>='2010-06-0
```

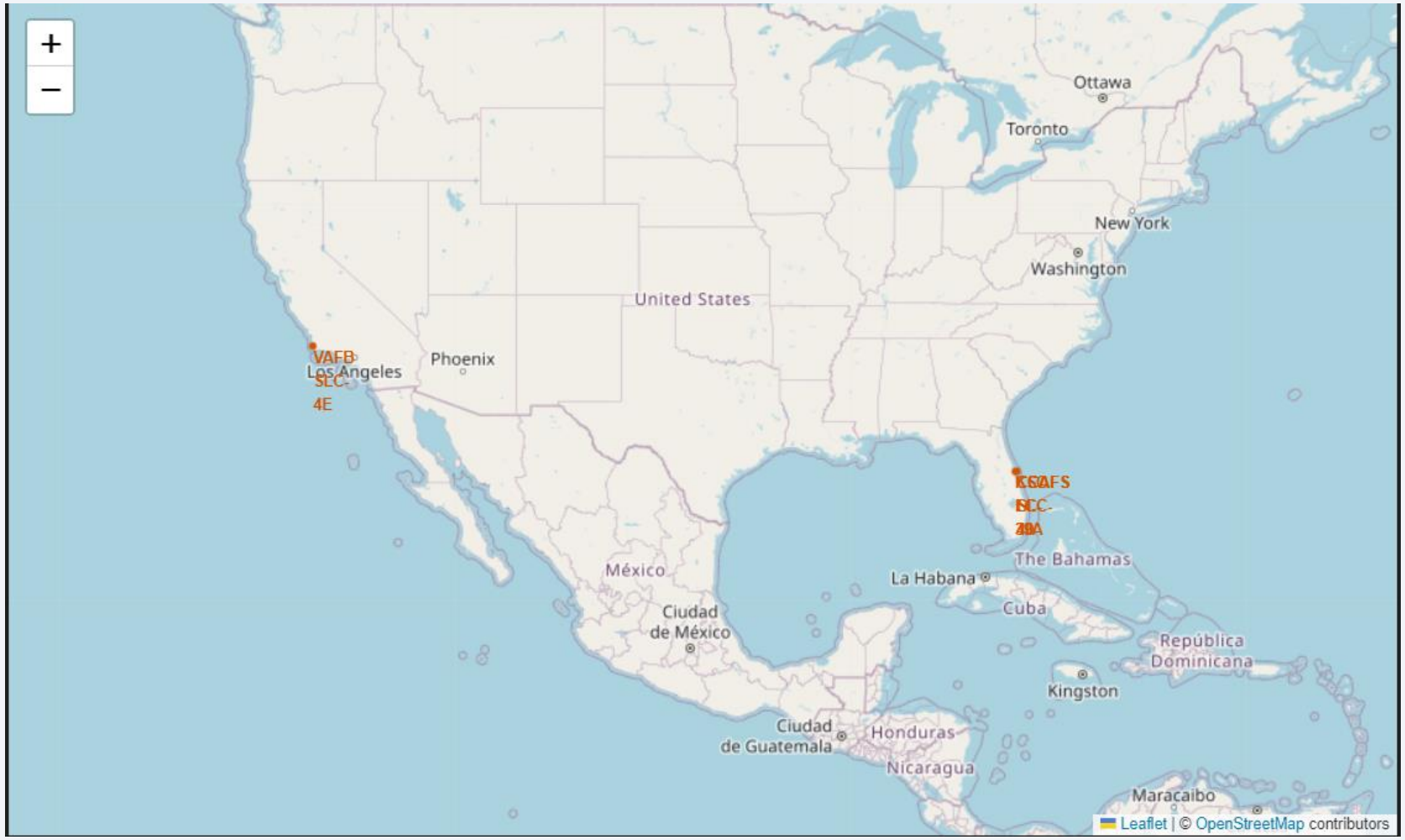
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	OCC
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

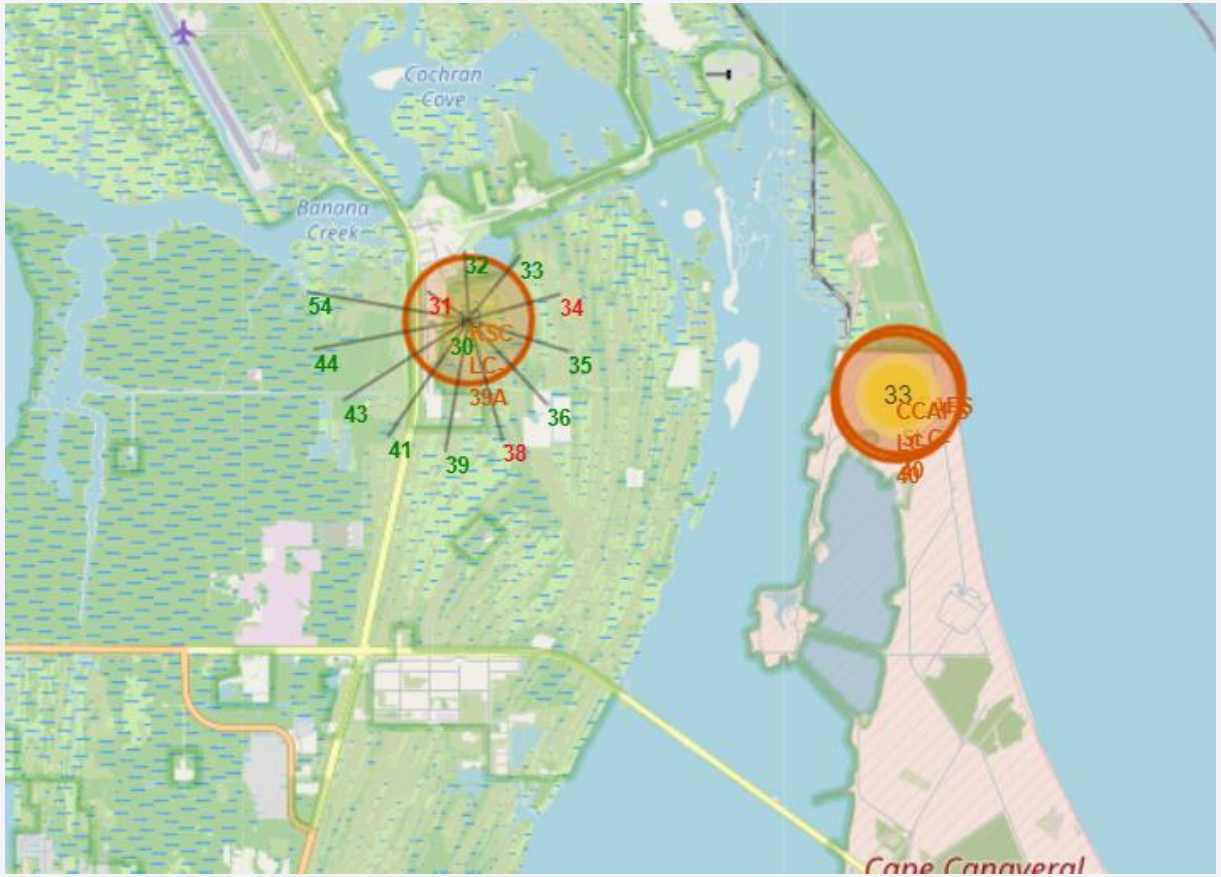
Launch Sites Proximities Analysis

All SpaceX Launch Sites

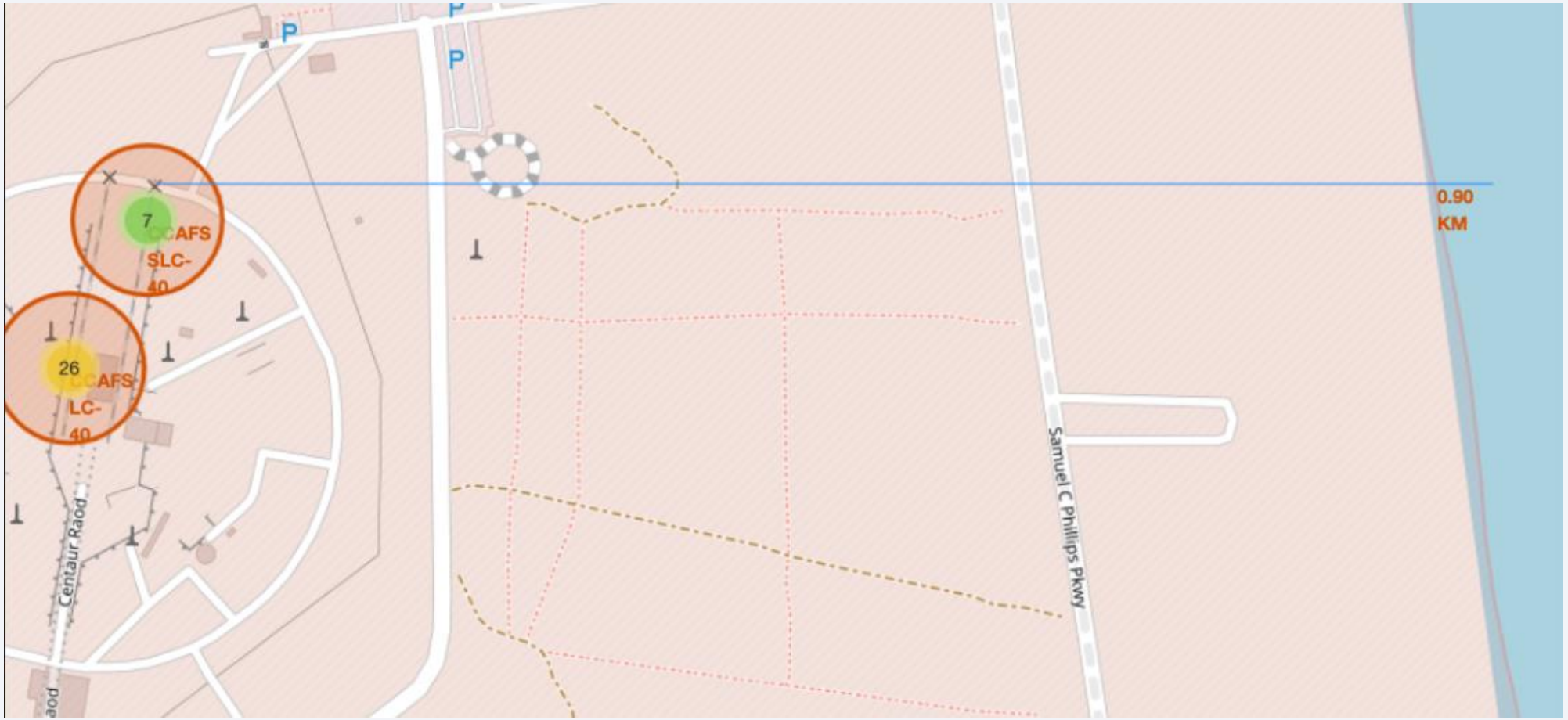


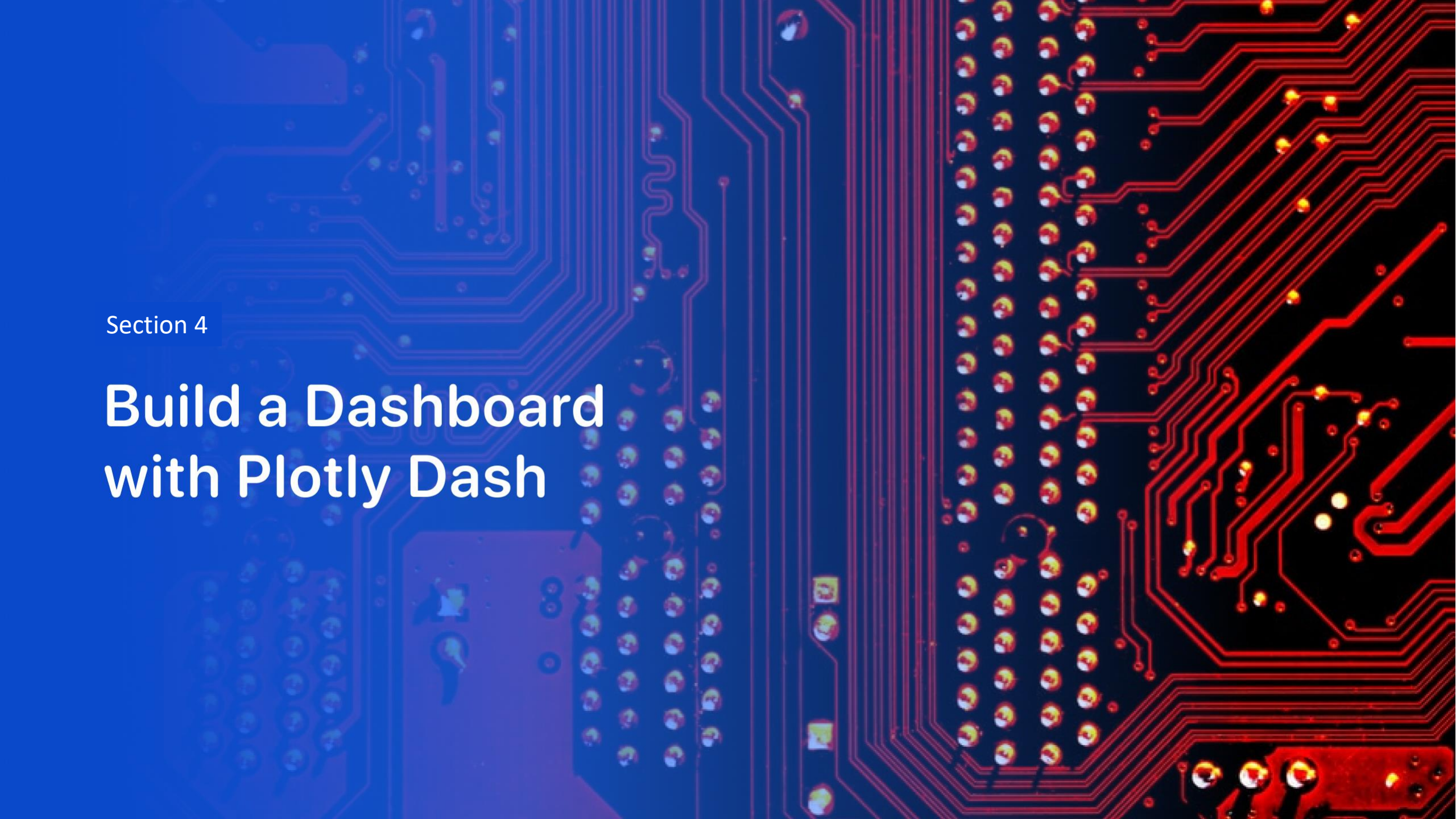
Launch Sites and their mission outcomes

- KSC LC-39A Has the most number of successful mission outcomes (i.e. landing of the the falcon 9 rocket)

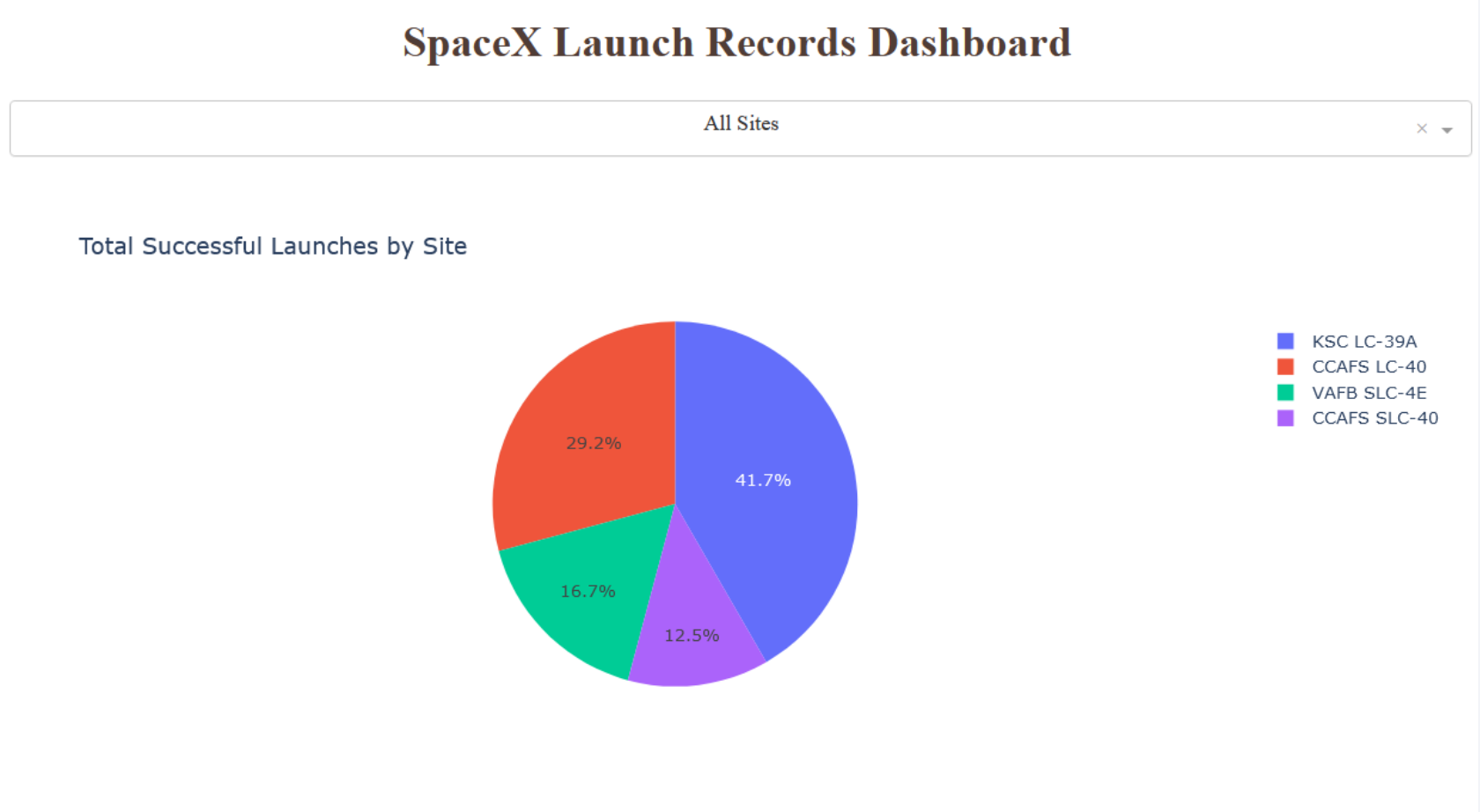


Proximity of Launch site to the ocean

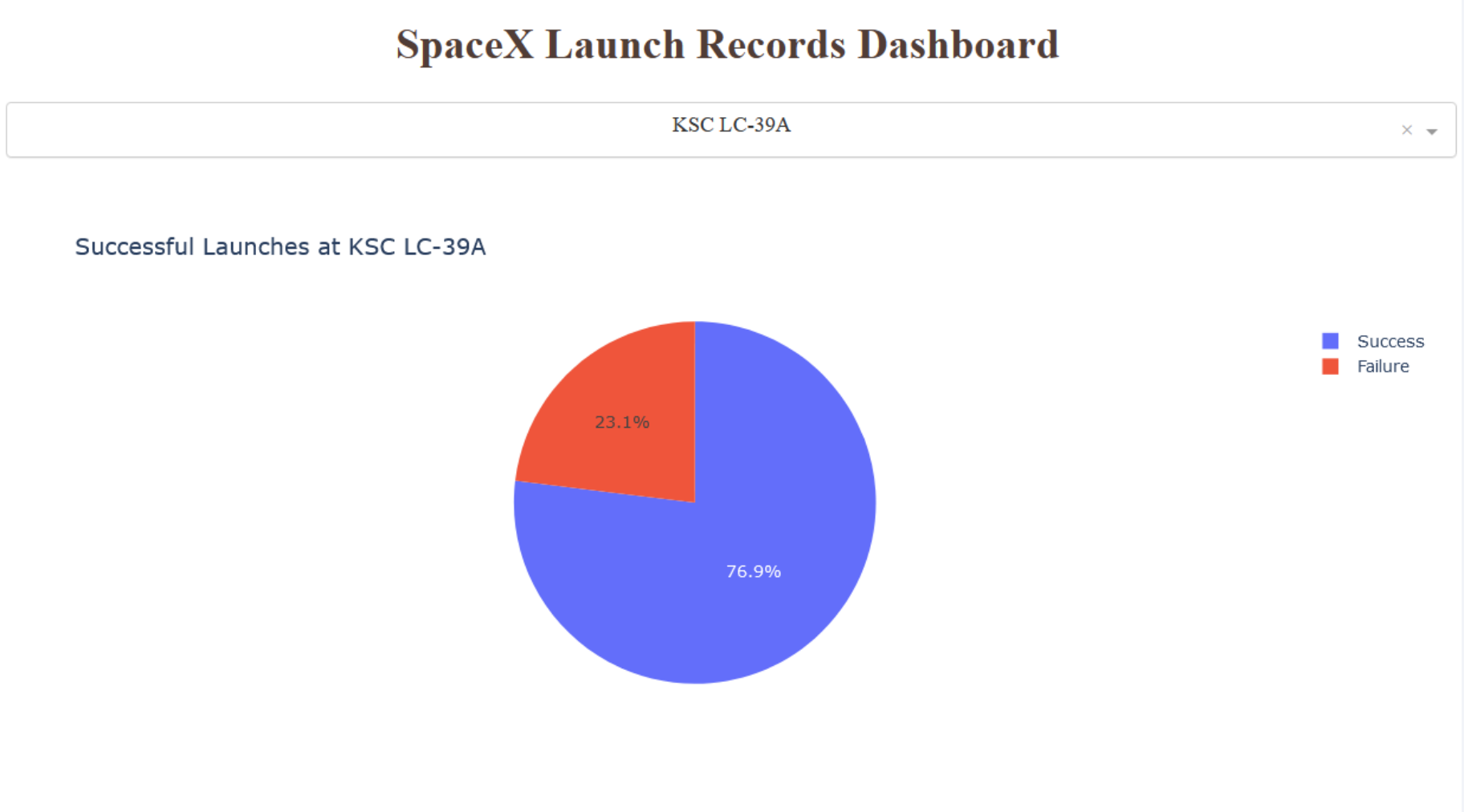



Section 4
Build a Dashboard with Plotly Dash

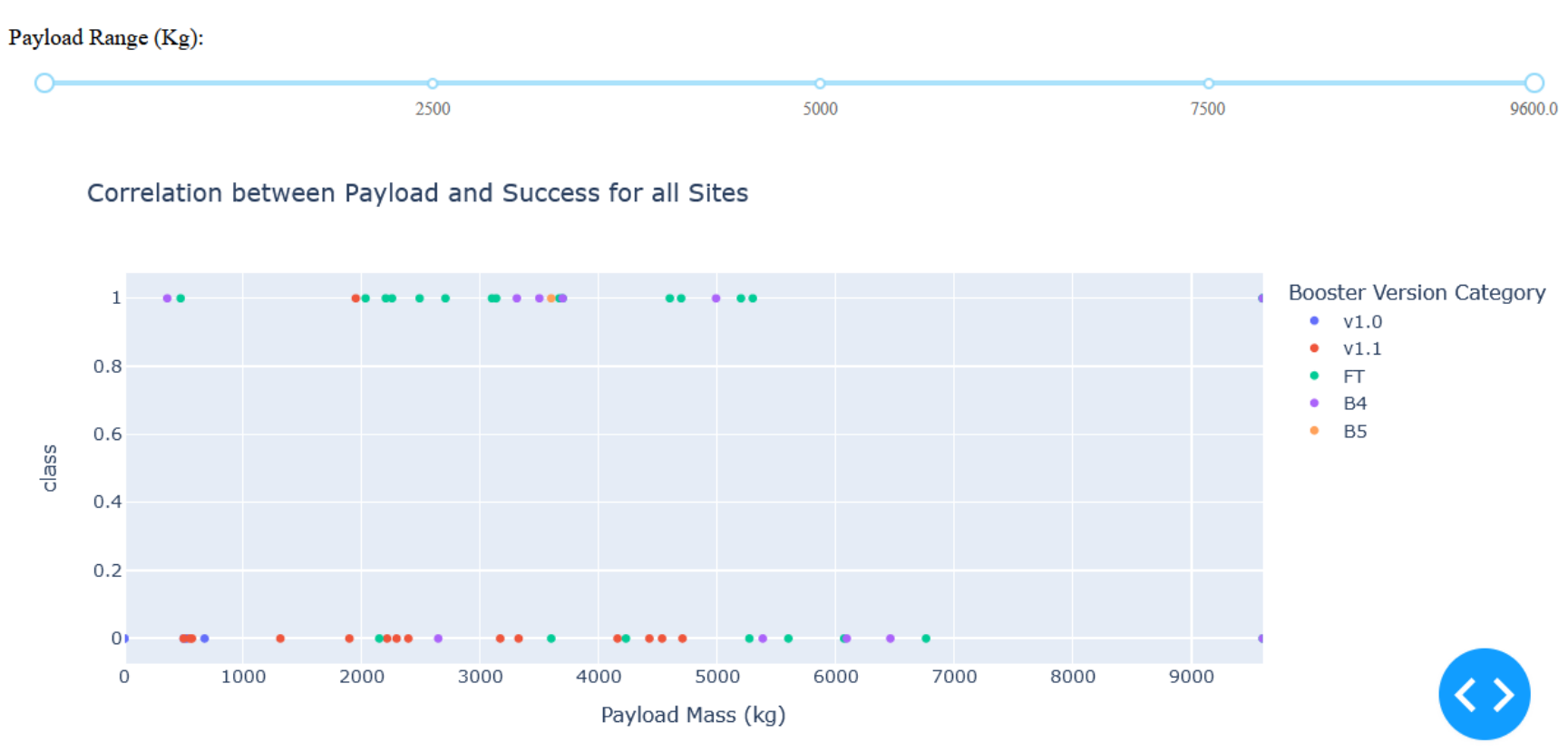
SpaceX Launch Records Dashboard



Launch site with the Highest Mission Success rate



Correlation between Payload and Success for all Sites



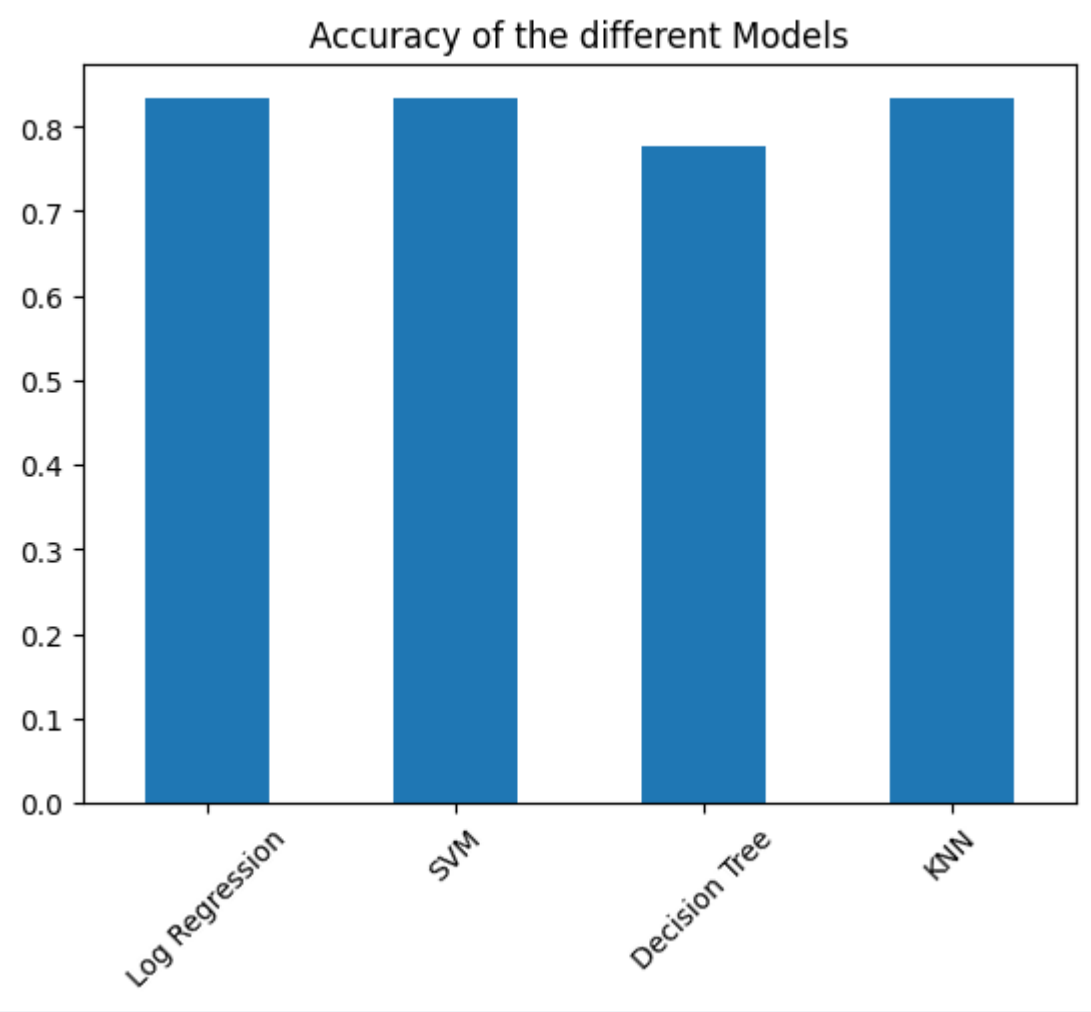
- Payloads between 2000 kg and 5500 kg seem to have the highest success rate.
- Payloads above 5500 kg seem to have a very low success rate.

Section 5

Predictive Analysis (Classification)

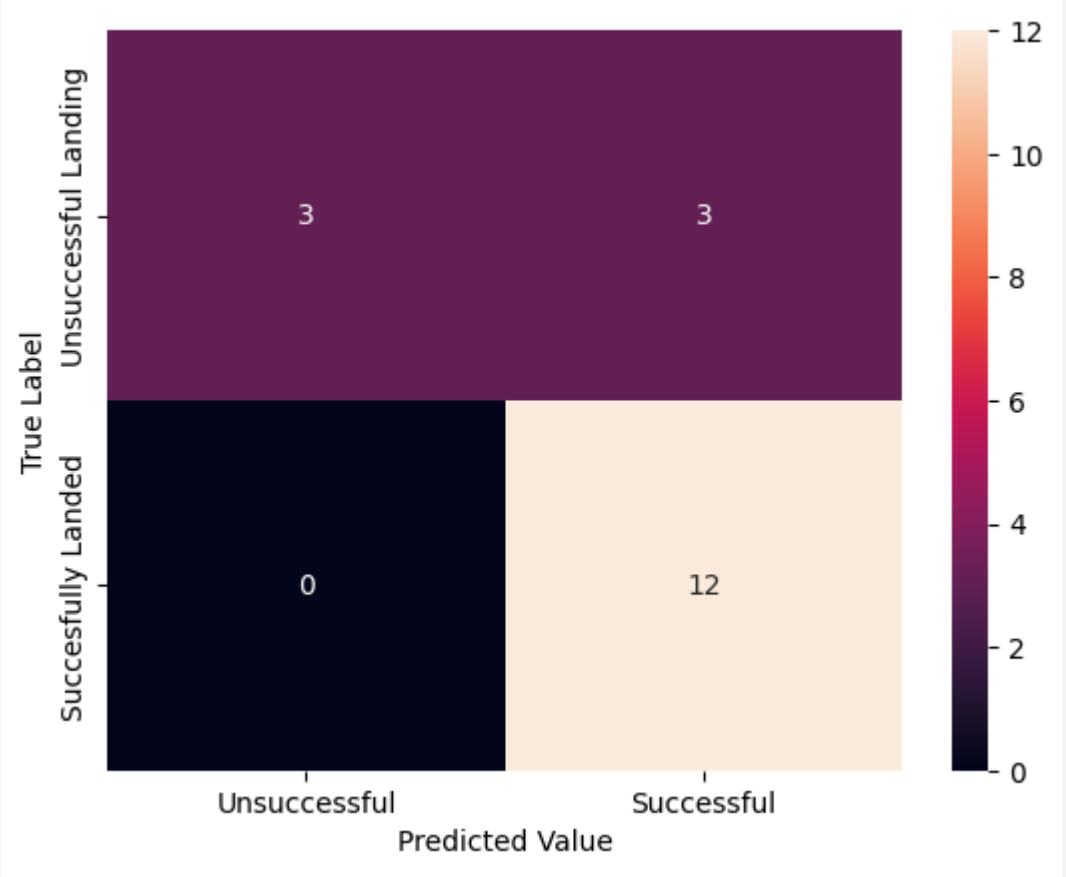
Classification Accuracy

- Log Regression, SVM and KNN have the highest accuracy.
- Decision Tree performs the least best among the models
- We can use wither Log regression, SVM or KNN to predict future mission outcomes.



Confusion Matrix of the Logistic Regression Model

- It correctly predicted all of the successful landings i.e., 12 of 12.
- It falsely predicted 3 landing that were unsuccessful as successful.
- There are 3 False Positives.



Conclusions

- Logistic Regression, SVM and KNN models are suitable to predict the landing outcome of future missions, having an accuracy score of 83.33% on the test data.
- Decision Trees algorithm had the least accuracy score with 77.78%
- The models correctly predict the successful landings, but have a tendency to generate False Positives, i.e. predict a mission will be successful even though it may not.
- Overall the accuracy of the model is high and we are able to determine with confidence if a rocket will land or not given the mission characteristics.

Appendix

- GitHub Repository: https://github.com/AmoMTL/SpaceXLandingPrediction

Thank you!