Winning Space Race with Data Science

Carlos Velazquez
Nov. 26 2021

OUTLINE

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

Executive Summary

- **Summary of Methodologies**
  - Data collection
  - Data Wrangling and Analysis
  - Folium Interactive Map
  - Dashboard with Plotly Dash
  - Predictive Analysis for models
- **Summary of Results**
  - Exploratory Data Analysis
  - Interactive Visualization
  - Predictive Analysis

Introduction

- **Project Background and Context:**

- The commercial space age has arrived, and companies are competing to bring affordable space travels for everyone. One company sticks out of the bunch and that is SpaceX. Advertised at 62 million dollars per launch, SpaceX's rocket launches are relatively inexpensive, while other companies advertise prices upwards of 165 million dollars each. The savings are due to SpaceX being able to reuse the first stage. Therefore, determining the first stages success we allow us to determine the cost of a launch

- **Problems we want to find answers:**

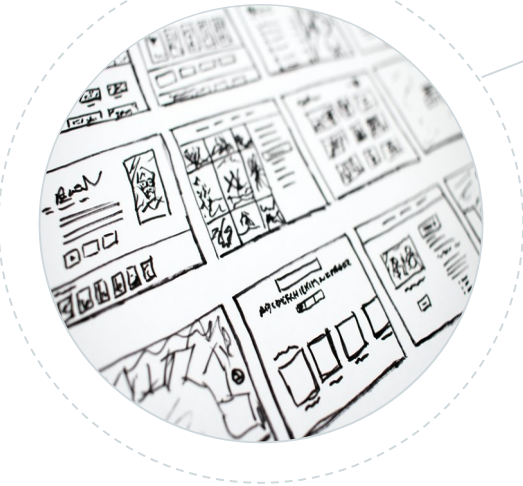
  - Mission parameters that effect landing outcome
  - Determine the price of each launch

Section 1

Methodology

Methodology

- Data Collection Methodology:
  - SpaceX REST API
  - Web Scraping
- Perform Data Wrangling:
  - Pandas
  - Numpy
- Perform Exploratory Data Analysis using Visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models



Data Collection

When collecting data you must have a goal or purpose for the kind of data your planning on gathering. You must know the methods and procedure that you'll be using to collect, store, and process the information.

SpaceX API:

- Request rocket launch data from SpaceX API
- Apply helper functions on data
- Construct dataset for creation of DataFrame
- Filter data and export

Web Scraping:

- Request HTML object
- Create BeautifulSoup object
- Extract names from HTML table header
- Create dataframe by parsing HTML tables
- Convert to pandas DataFrame
- Dataframe to csv

Data Collection – SpaceX API

- Request rocket launch data from SpaceX API

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

- Helper functions to extract information

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

- Construct dataset for creation of a pandas DataFrame

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']), 'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass, 'Orbit':Orbit,
'LaunchSite':LaunchSite, 'Outcome':Outcome,
'Flights':Flights, 'GridFins':GridFins,
'Reused':Reused, 'Legs':Legs,
'LandingPad':LandingPad, 'Block':Block,
'ReusedCount':ReusedCount, 'Serial':Serial,
'Longitude': Longitude, 'Latitude': Latitude}

df = pd.DataFrame(launch_dict)
```

- Filter the DataFrame and export to file

```
data_falcon9 = data.drop(data[data['BoosterVersion'] == 'Falcon 1'].index)

data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9

data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

GitHub URL

Data Collection – Web Scraping

Request HTML Response

```
HTML_response = requests.get(static_url)
```

Create BeautifulSoup Object

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

Extract names from the HTML table header

```
html_tables = soup.find_all('table')
first_launch_table = html_table[2]
columns_names = []
columns = frist_launch_table.find_all('th')
for row in range(len(columns)):
    name = extract_column_from_header(columns[row])
    if name != None and len(name) > 0:
        column_names.append(name)
```

Create data frame by parsing HTML tables

```
extracted_row = 0
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowhe
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        row=rows.find_all('td')
        if flag:
            extracted_row += 1
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
```

Convert to pandas DataFrame

```
df = pd.DataFrame.from_dict(launch_dict, orient='index')
df = df.transpose()
```

Export out

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

GitHub URL

Data Wrangling

Data Wrangling involves transforming and mapping data fro the aim of making the data more accessible for our analytics

1. Calculate the number of launches on each site

```
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

2. Calculate the number and occurences of each orbit

```
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
HEO       1
GEO       1
ES-L1     1
SO        1
Name: Orbit, dtype: int64
```

3. Create a landing outcome label from outcome column

```
landing_class = np.where(df['Outcome'].isin(bad_outcomes), 0, 1)
landing_class
```

```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```

4. Export to csv

```
df.to_csv("dataset_part\_2.csv", index=False)
```

GitHub URL

EDA with Data Visualization

- Scatter chart
  - Scatter plots helps us visualize the correlation between variables and how they affect the launch outcome
- Bar chart
  - Bar charts help interpret relationships between variables. We can visually check if there are any relationship between success rate and orbit type
- Line chart
  - Line charts helps us visually see trends over time. We can observe that the success rate since 2013 kept increasing till 2020

GitHub URL

EDA with SQL

SQL is a computer language for storing, manipulating and retrieving data stored in a relational database. With SQL we create statements that are used for queries. We query the database which in this case is IBM Db2, analyzing the data forr insights.

SQL Queries performed:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- 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 successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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 URL

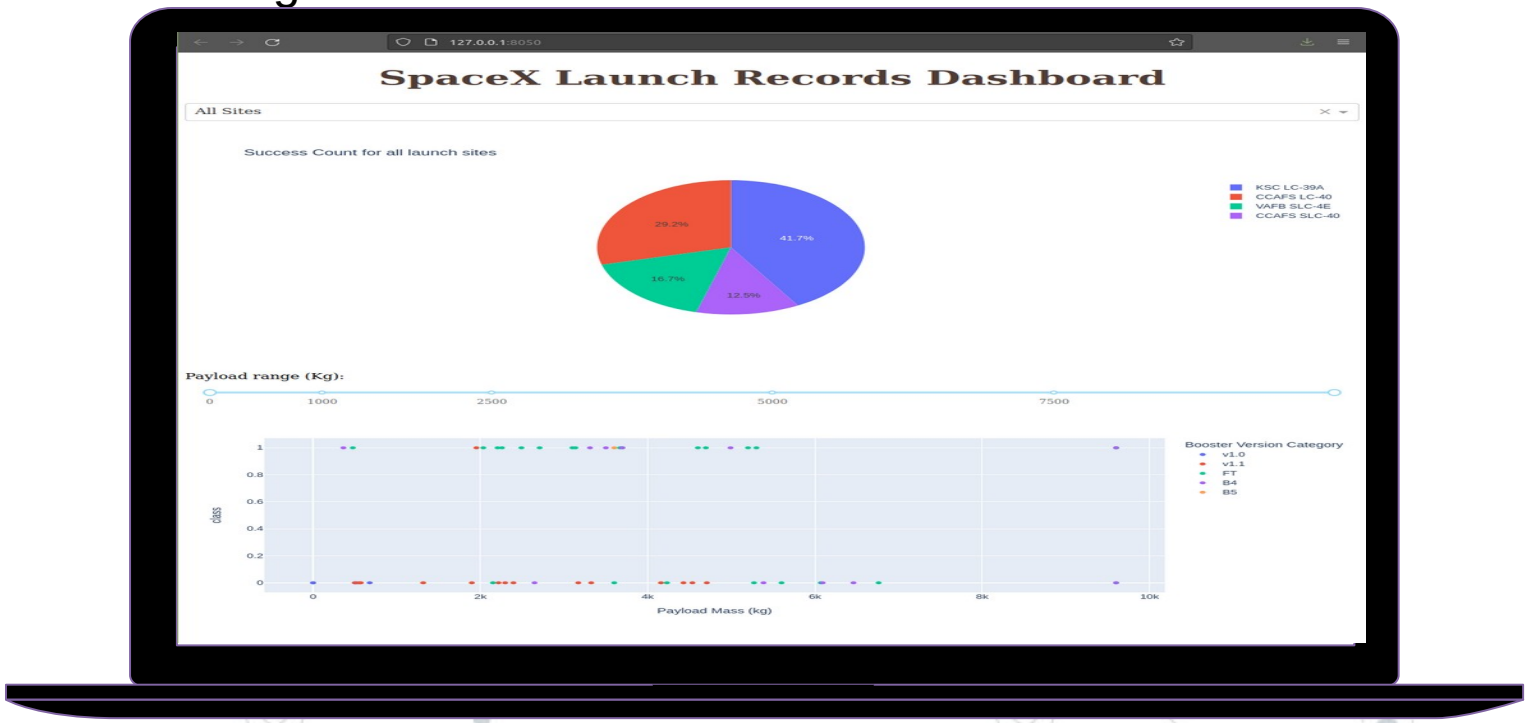
Building an Interactive Map with Folium

After manipulating the data we start building our interactive map using Folium. We want to look for launch success rate dependencies. We create markers and circle objects from our data for marking and highlighting areas of interest. For example we highlight each launch site with a circle object and mark the location name with a marker object. We can also create line objects and measure the distance between points. When creating multiple objects with the same coordinate it's best to simplify the display with marker clustering.

GitHub URL

Build a Dashboard with Plotly Dash

For this dashboard application we added input components such as a **Drop-down** list and a **Range-Slider**. The input components allow us to interact with a pie and a scatter chart. Through interacting with the charts we're able to get insight on which sites has the highest launch success rates and which payload ranges have the highest and lowest success rates.



GitHub URL

Predictive Analysis (Classification)

1. Building the model:

- Import libraries and load the data
- Create a Numpy array from column “Class” and assign to Y
- Standardize and transform the data in X
- Split the data in X and Y into training and test data
- Set test_size and random_state parameters
- Create machine learning objects
- Create GridSearchCV object

2. Evaluate model:

- Calculate the accuracy on the test data
- Plot Confusion Matrix

3. Improve model:

- Fit object with best parameters

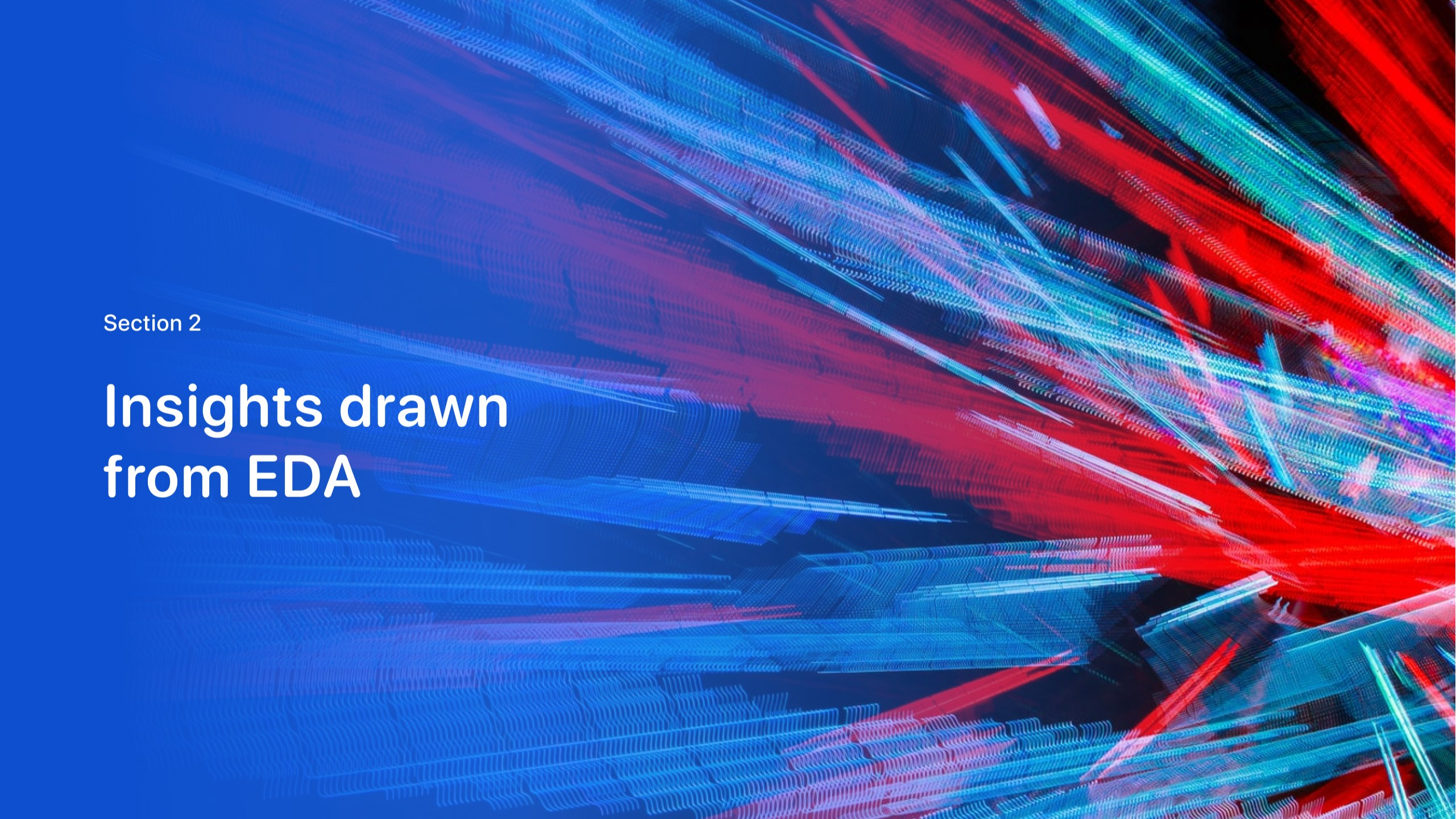
4. Find best performing classification model:

- Model with highest accuracy score

GitHub URL

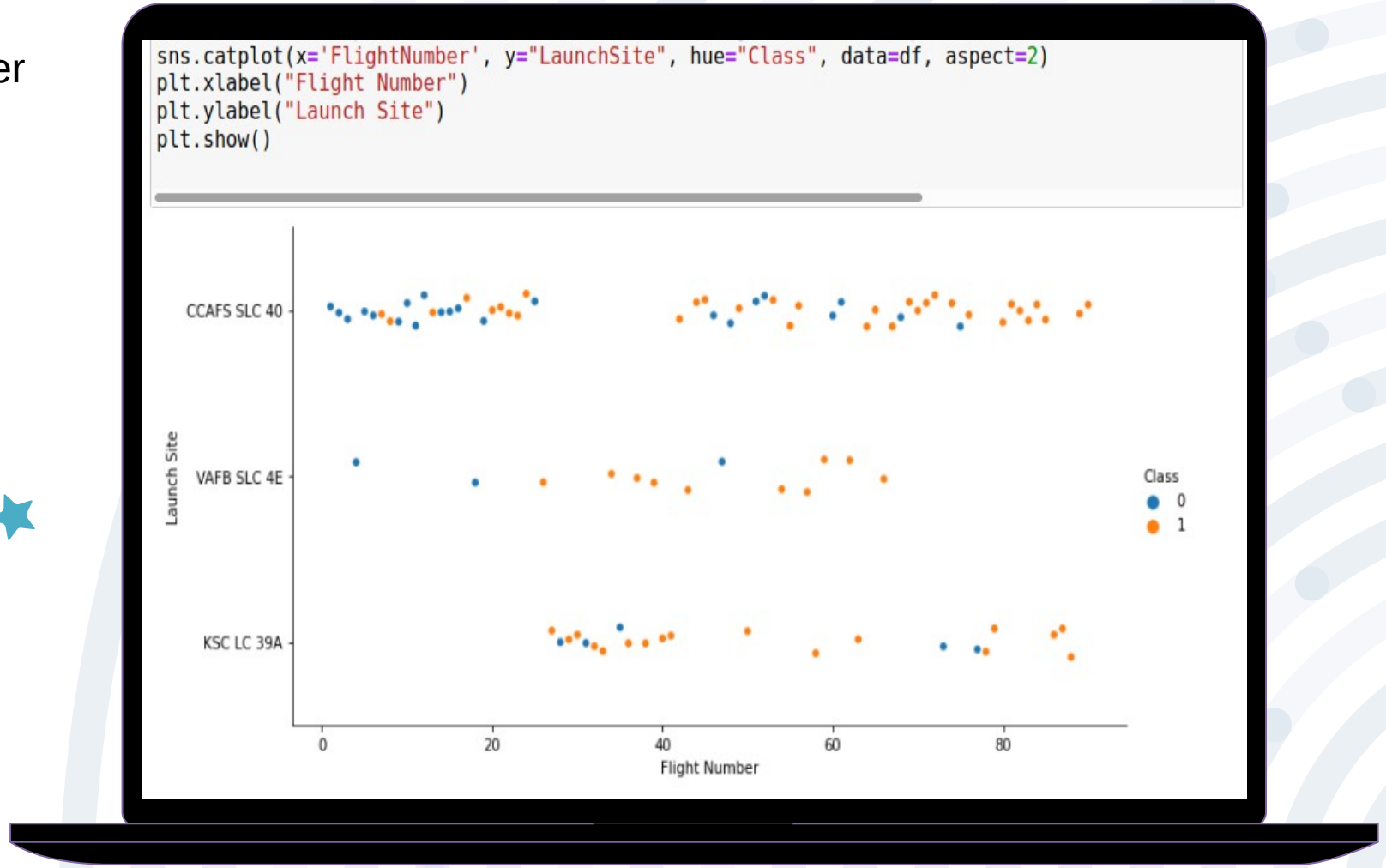
Results

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

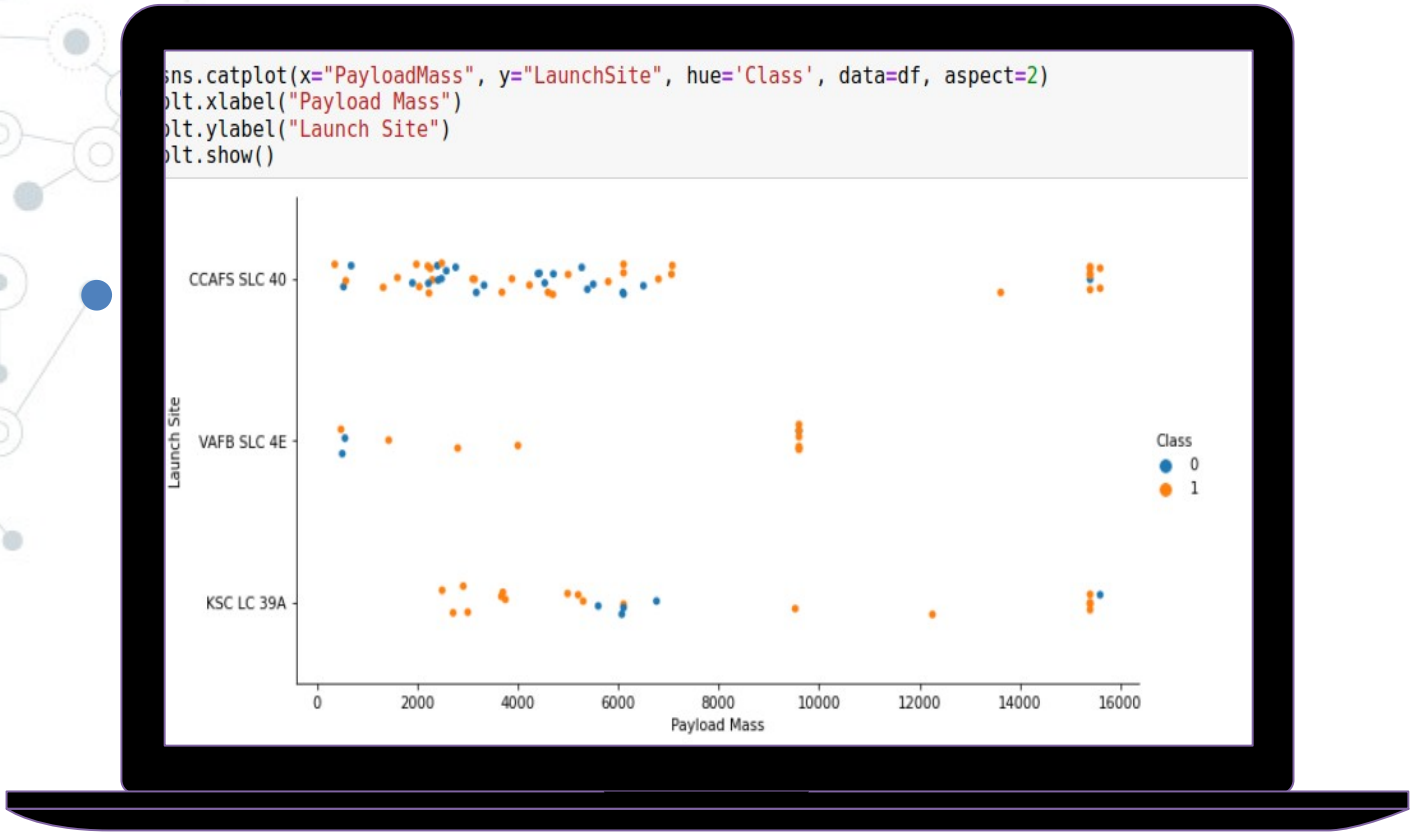

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

The greater the Flight Number per Launch Site, the higher the success rate gets



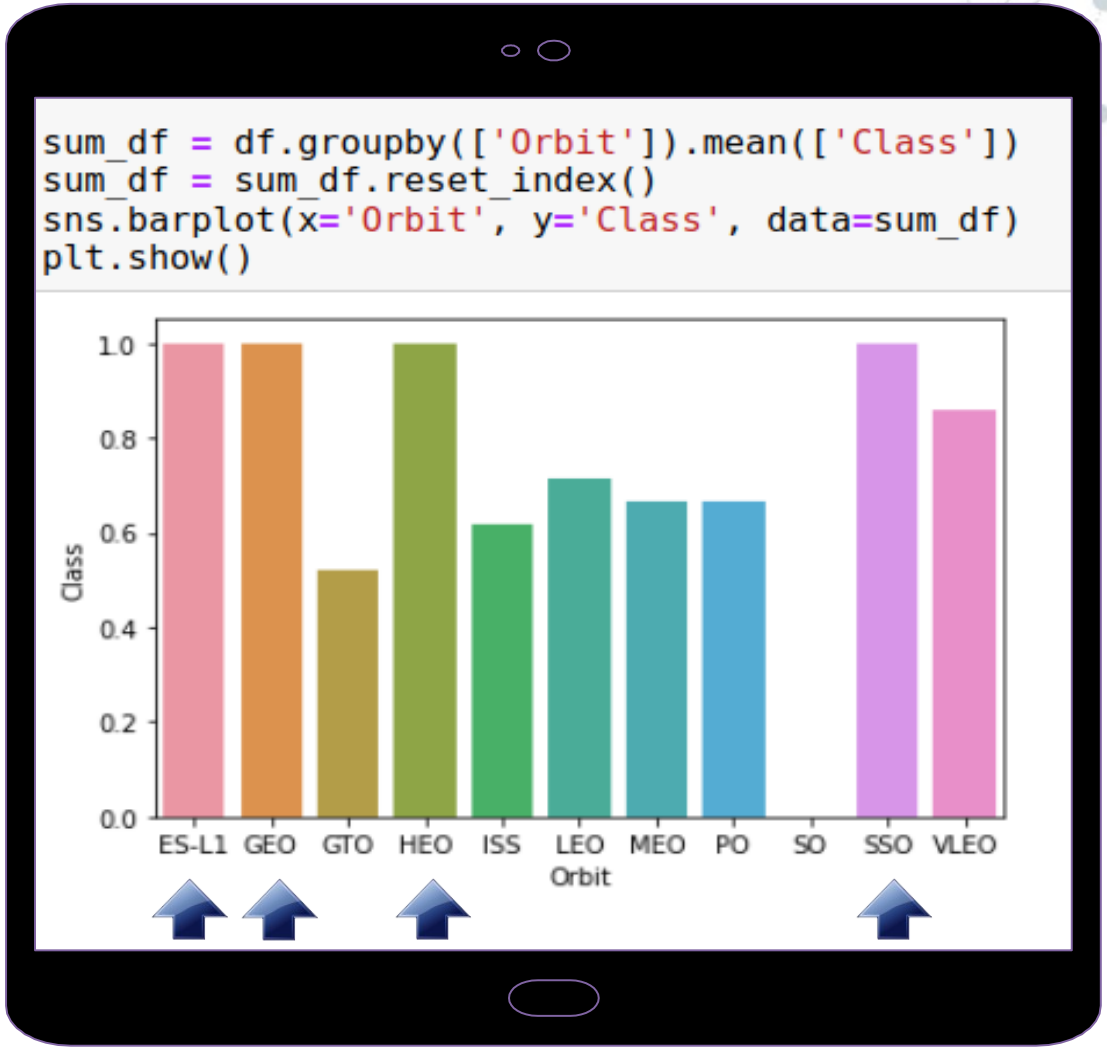
Payload vs. Launch Site



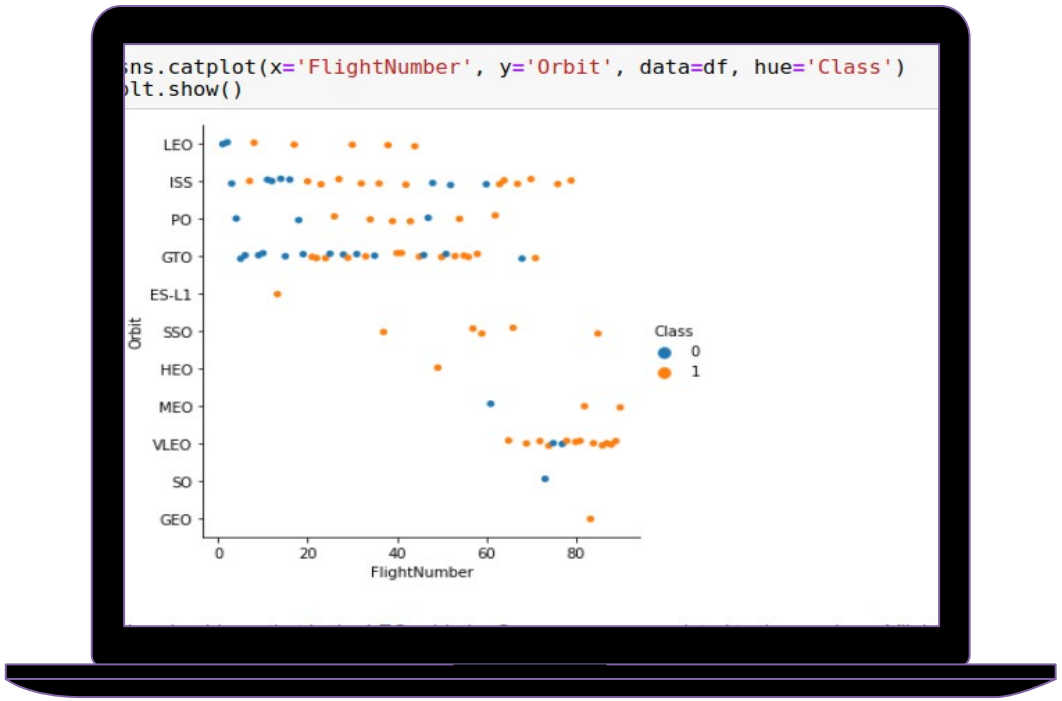
From the chart we can gather that Launch Site “**VAFB SLC 4E**” doesn’t contain rocket launches on payloads greater then 10000.

Success Rate vs. Orbit Type

From the bar chart we're able to see the Orbit Types with the highest success rate: "ES-L1","GEO","HEO","SSO"



Flight Number vs. Orbit Type

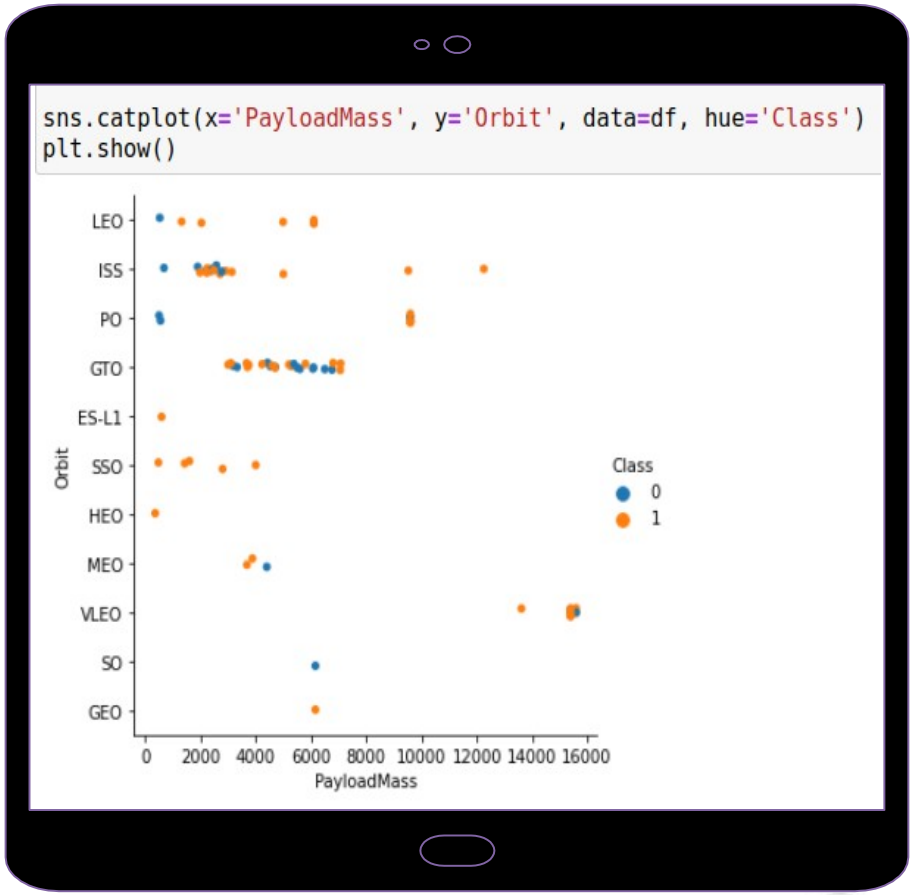


We can see some positive correlation between Flight Number and Orbit type “LEO” .
We can also see that there is no correlation for Orbit type “GTO” and Flight Number.

Payload vs. Orbit Type

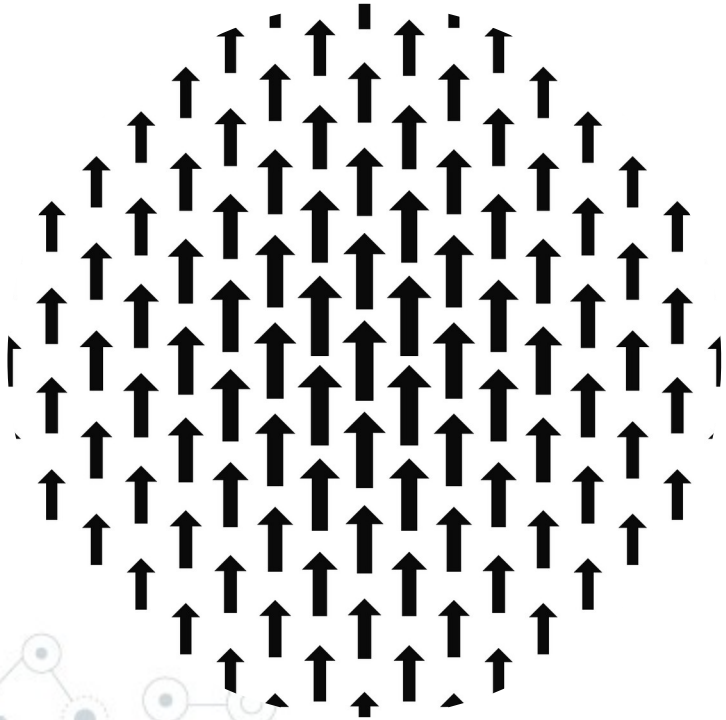
We can observe Orbit types “Po”,”LEO” and “ISS” have a higher positive landing rate with heavy Pay loads

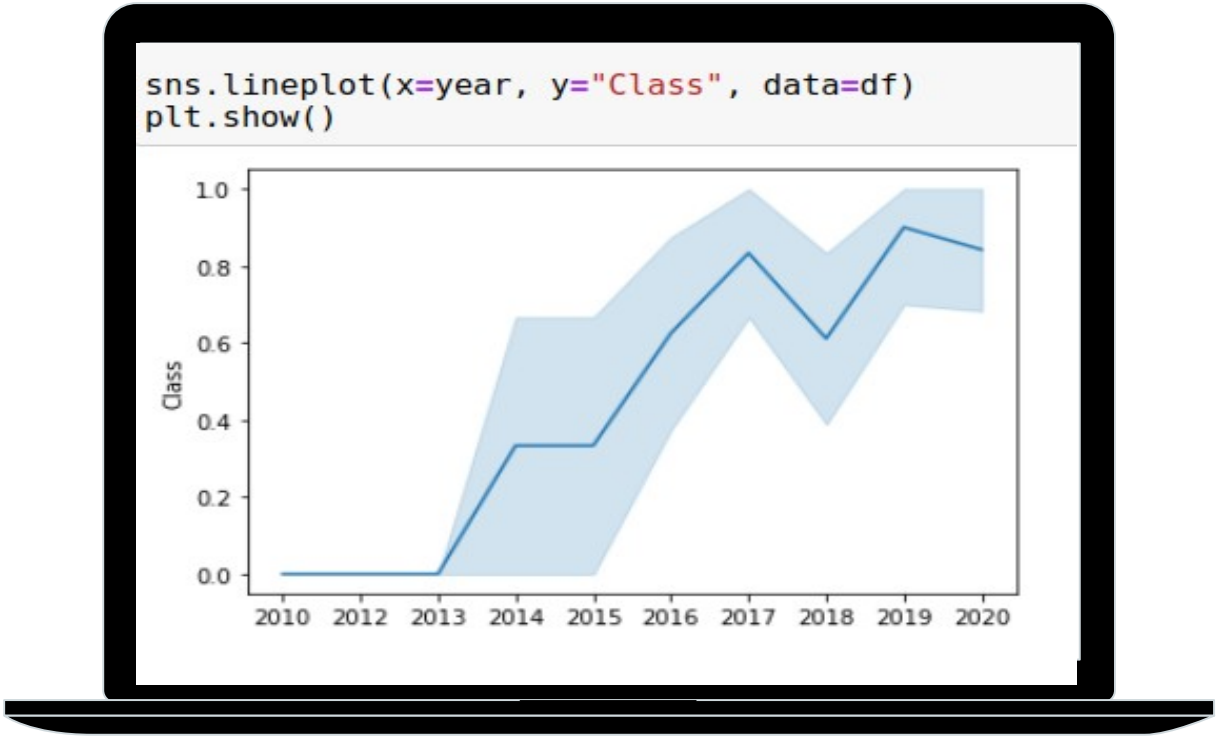
However we can’t positively correlate or negatively correlate the data for Orbit type “GTO”



Launch Success Yearly Trend

We observe the success rate start increasing in 2013 till 2020





All Launch Site Names

SQL Query:

```
%sql SELECT DISTINCT launch_site FROM SPACEXDATASET;
```

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

We display the names of the launch sites using “SELECT DISTINCT” statement to return only distinct values.

Launch Site Names Begin with ‘CCA’

SQL Query:

```
%sql SELECT * FROM SPACEXDATASET WHERE (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

We use the “**WHERE**” clause and the “**LIKE**” operator with the **“%**” wildcard at the end of the value I want to be found.

DATE	Time (UTC)	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	Landing _Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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 (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

SQL Query:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass" FROM SPACEXDATASET WHERE (CUSTOMER) LIKE 'NASA%';
```

We use the “**SUM**” function to return the “Total Payload Mass” and the “**WHERE**” clause with the “**LIKE**” operator to find all the boosters from “NASA”

Total Payload Mass
99980

Average Payload Mass by F9 v1.1

SQL Query:

```
%%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster version F9 v1.1" FROM SPACEXDATASET
WHERE (BOOSTER_VERSION) LIKE 'F9 v1.1%';
```

We use the “**AVG**” function to return the average payload mass carried and the “**WHERE**” clause followed by the “**LIKE**” operator to specify booster version F9 v1.1

Average Payload Mass by Booster version F9 v1.1
2534

First Successful Ground Landing Date

SQL Query:

```
%%sql SELECT MIN(DATE) AS "First Ground Pad Landing" FROM SPACEXDATASET
WHERE "Landing _Outcome" LIKE 'Success (ground pad)%';
```

We use the “**MIN**” function to return the smallest value of “DATE” and the “**WHERE**” clause followed by the “**LIKE**” operator to select when the first successful landing outcome in ground pad was achieved

First Ground Pad Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query:

```
%%sql SELECT booster_version FROM SPACEXDATASET
WHERE "Landing _Outcome" LIKE 'Success (drone ship)%'
AND payload_mass__kg_ BETWEEN 4000 AND 6000;
```

We select the booster versions using the “**WHERE**” clause and the “**LIKE**” operator to to list the booster version that have successful drone ship landing. We also use the “**BETWEEN**” operator to select the values between 4000 and 6000.

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

Total Number of Successful and Failure Mission Outcomes

SQL Query:

```
%sql SELECT COUNT(mission_outcome) AS Total, mission_outcome FROM SPACEXDATASET GROUP BY mission_outcome;
```

We use the “**COUNT**” function and the “**GROUP BY**” statement to calculate the total number of successful and failed mission outcomes

total	mission_outcome
1	Failure (in flight)
99	Success
1	Success (payload status unclear)

Boosters Carried Maximum Payload

SQL Query:

```
%%sql SELECT booster_version, payload_mass__kg_
FROM SPACEXDATASET
WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_)
                           FROM SPACEXDATASET);
```

We use a **subquery** nested inside our query. Our subquery is using the "**MAX**" function to return the booster version carrying only maximum payload

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

SQL Query:

```
%%sql SELECT booster_version, launch_site
FROM SPACEXDATASET
WHERE DATE LIKE '2015%' AND "Landing _Outcome" LIKE 'Failure (drone ship)%';
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

We query the booster version and launch site using the “**WHERE**” clause and the “**LIKE**” operator twice to return the failed landing outcomes in drone ship for the year 2015

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

SQL Query:

```
%%sql SELECT "Landing _Outcome", COUNT("Landing _Outcome") as Amount
FROM SPACEXDATASET
WHERE DATE BETWEEN '2010-06-04' and '2017-03-20'
GROUP BY "Landing _Outcome"
ORDER BY Amount DESC;
```

We rank the landing outcomes between dates using a “**WHERE**” clause followed by a “**BETWEEN**” operator and the dates we want. We use the “**GROUP BY**” statement to group the “**COUNT**” of Landing Outcomes. We rank them in descending order by using the “**ORDER BY**” keyword.

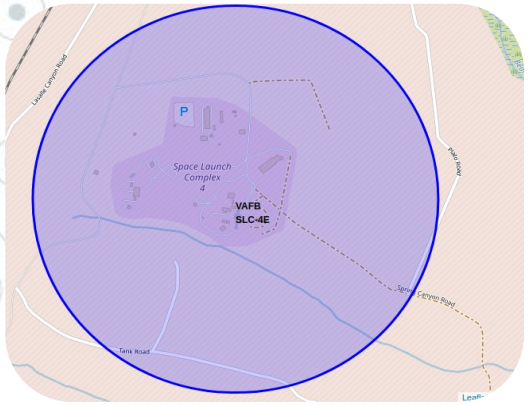
Landing _Outcome	amount
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

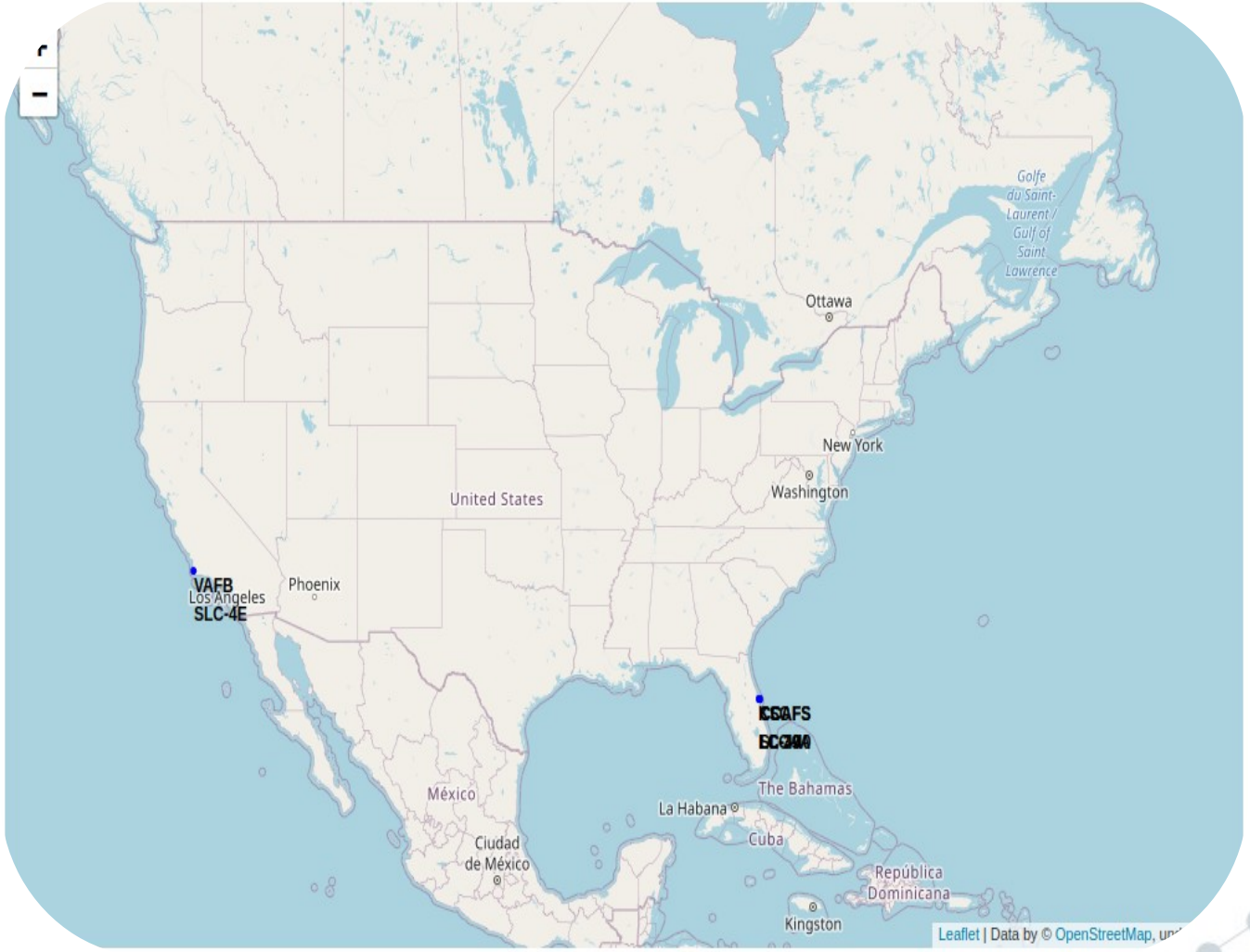
Section 4

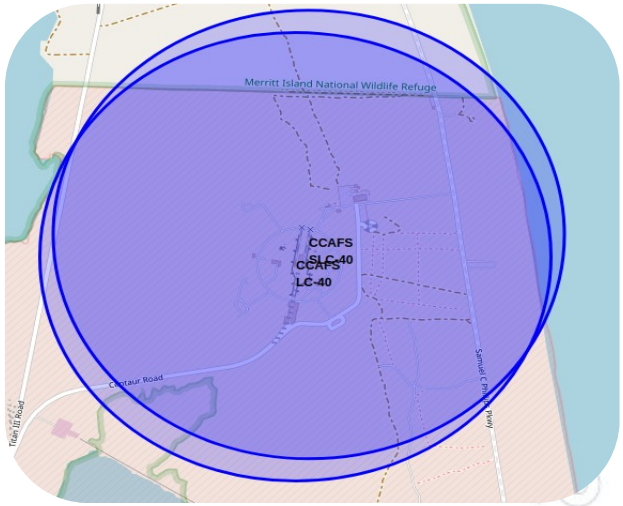
Launch Sites Proximities Analysis

Folium: All Launch Sites

We can observe that all the launch sites are located near a coast

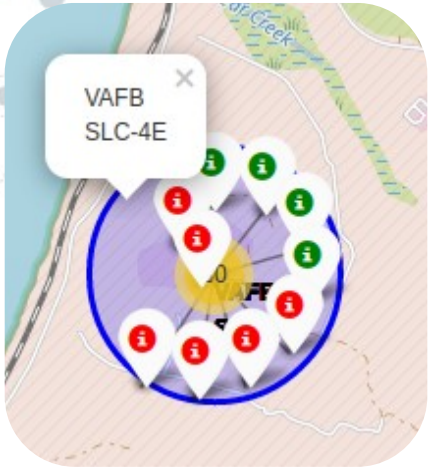


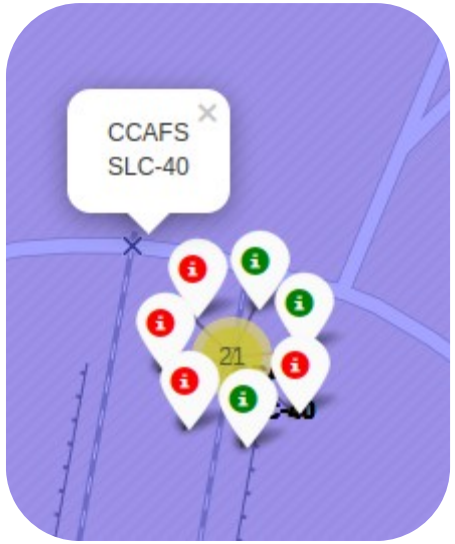


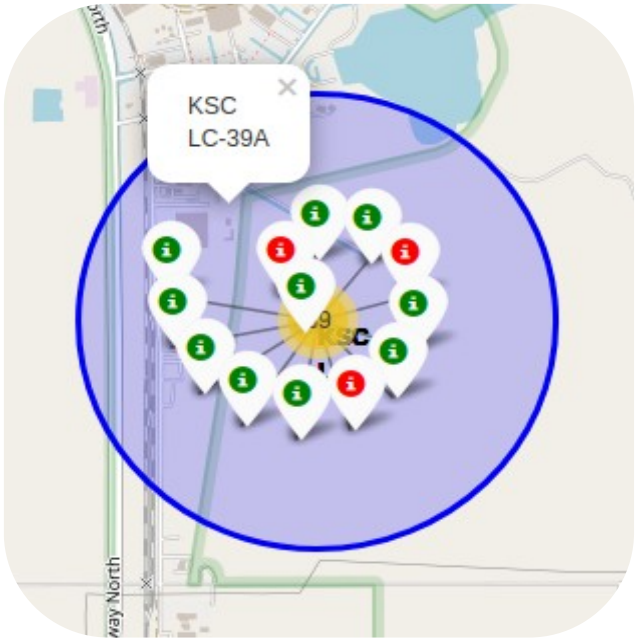


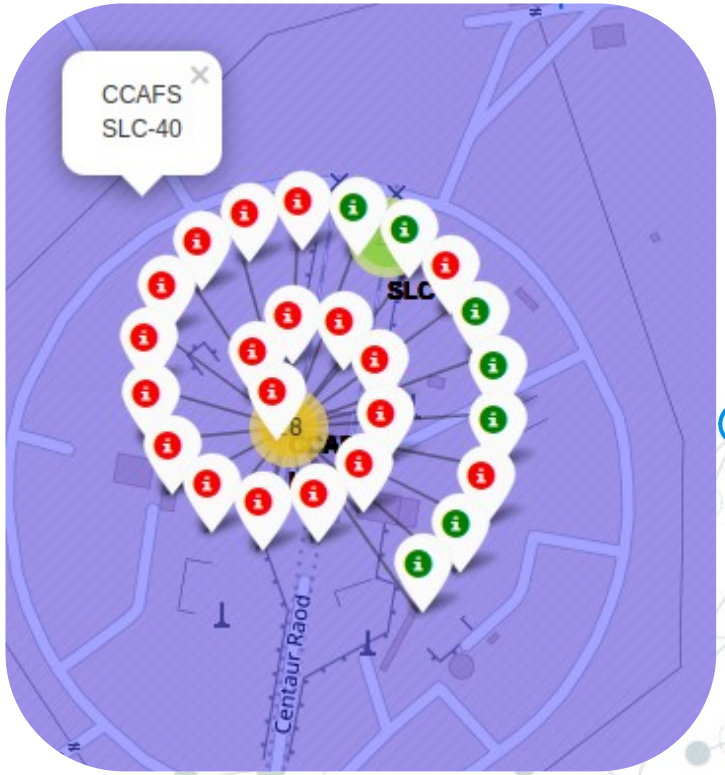
Folium: Mark the successful and failed launches

To determine which sites have high success rate we enhance the map by adding the launch outcomes for each site. **Green marker** shows successful launches while the **red markers** show failed launches.

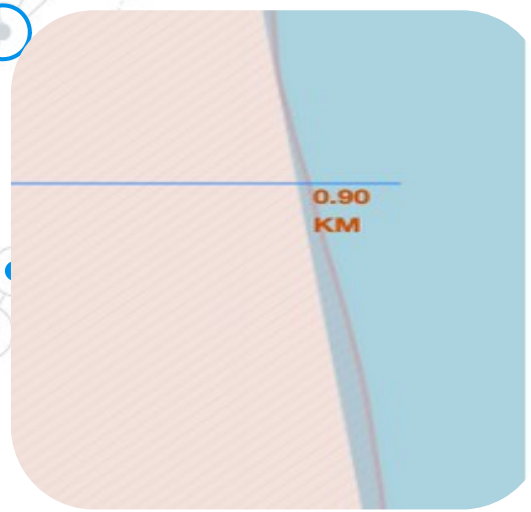


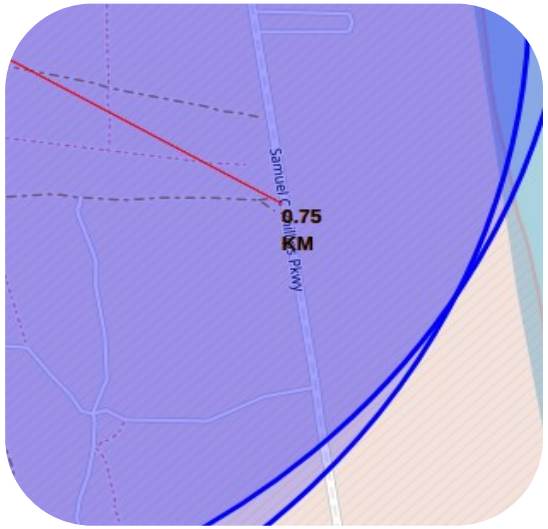






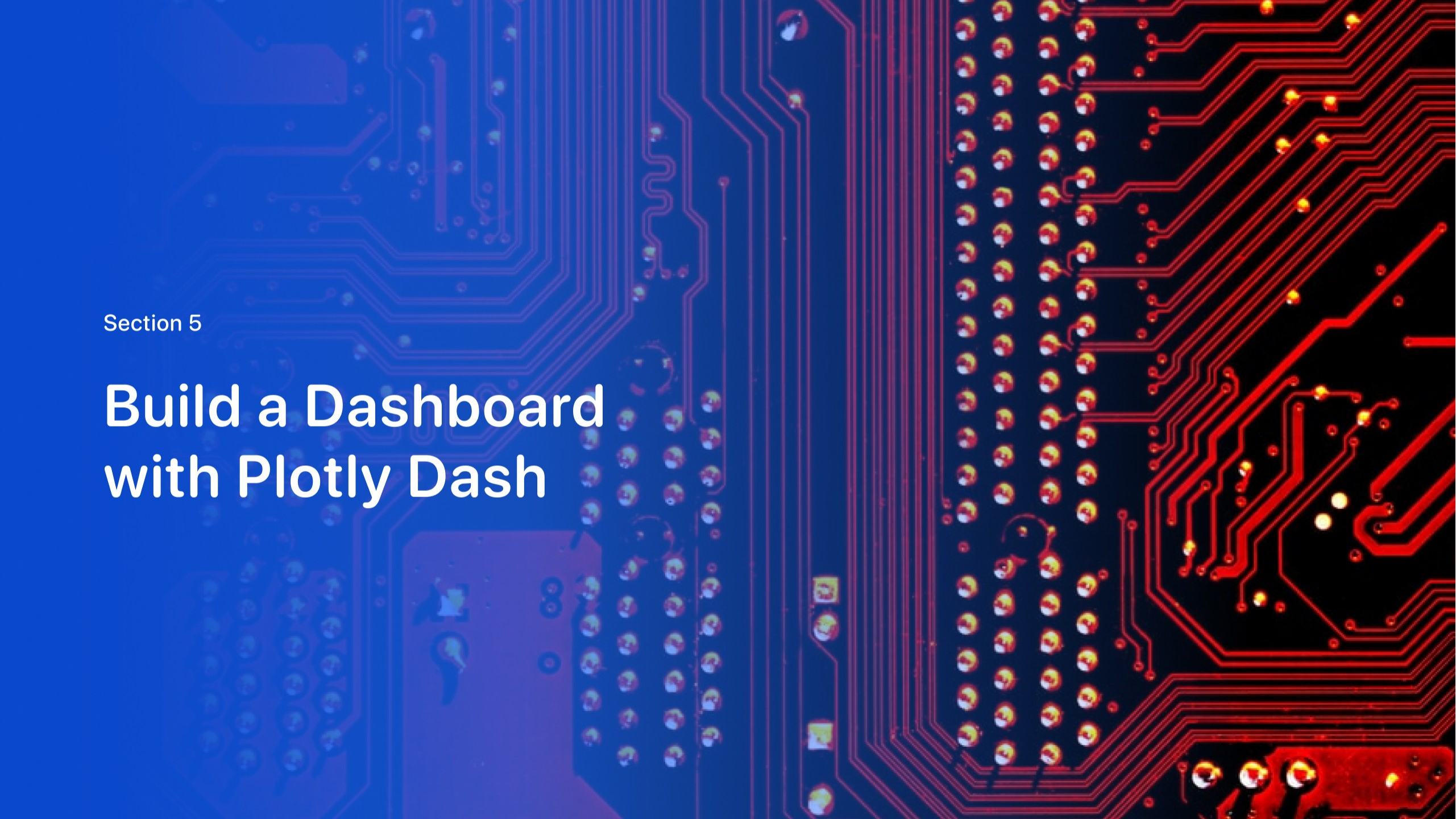
Folium: Railway, Highway, Coastline map proximity

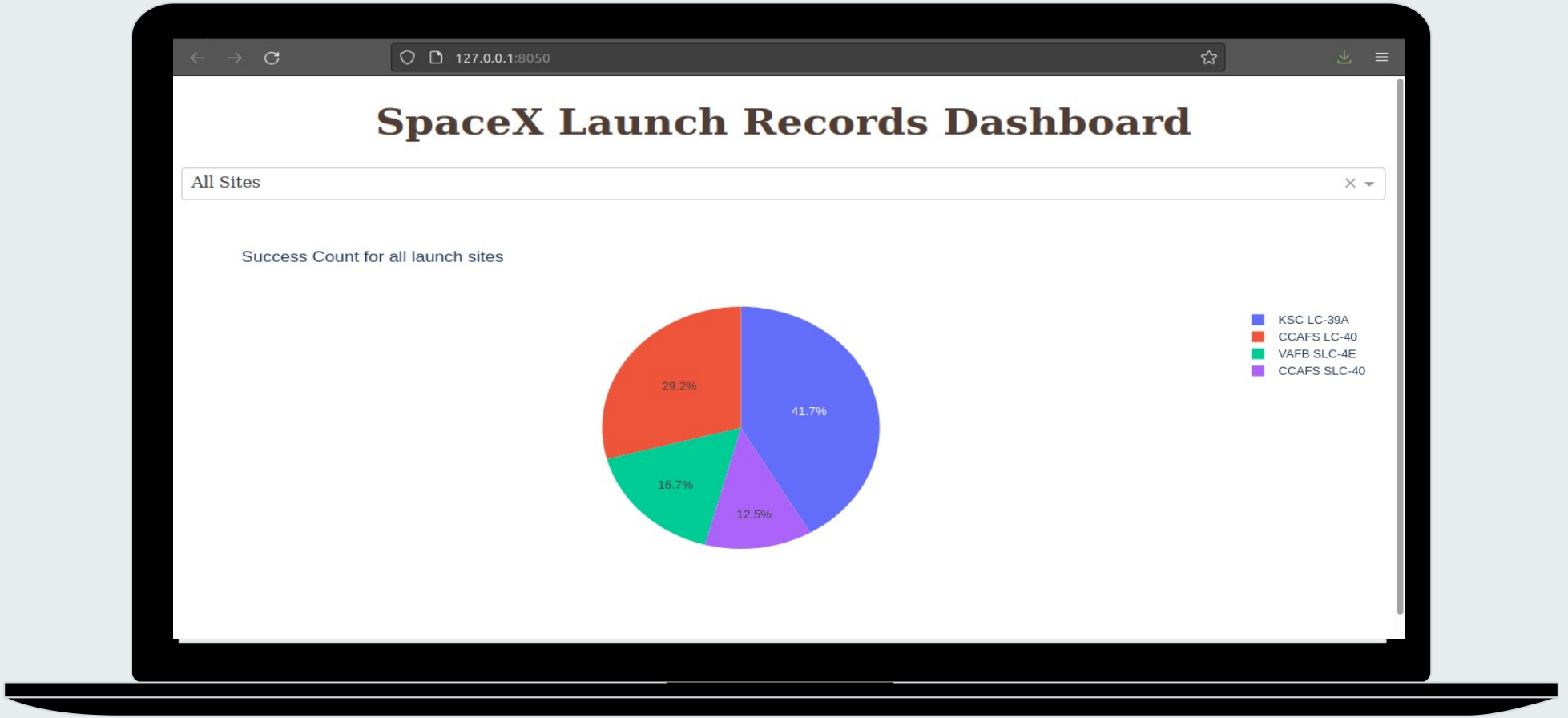




After plotting the distance lines to the proximity we can determine if the launch sites are in close proximity:

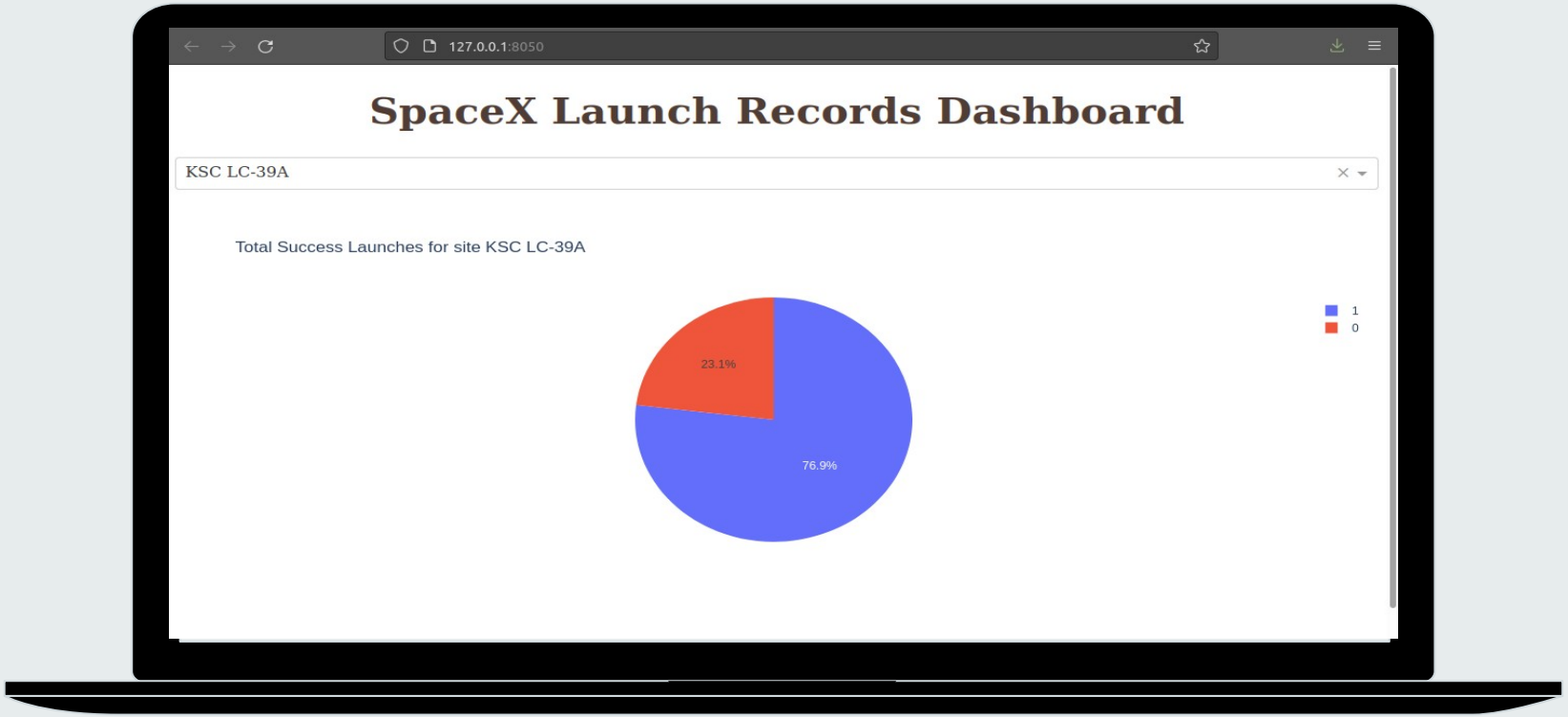
- Are launch sites in close proximity to railway? 1.49 KM
- Are launch sites in close proximity to highways? 0.75 KM
- Are launch sites in close proximity to coastline? 0.90 KM
- Do launch sites keep certain distance away from cities? 18.18 KM


Section 5
Build a Dashboard with Plotly Dash



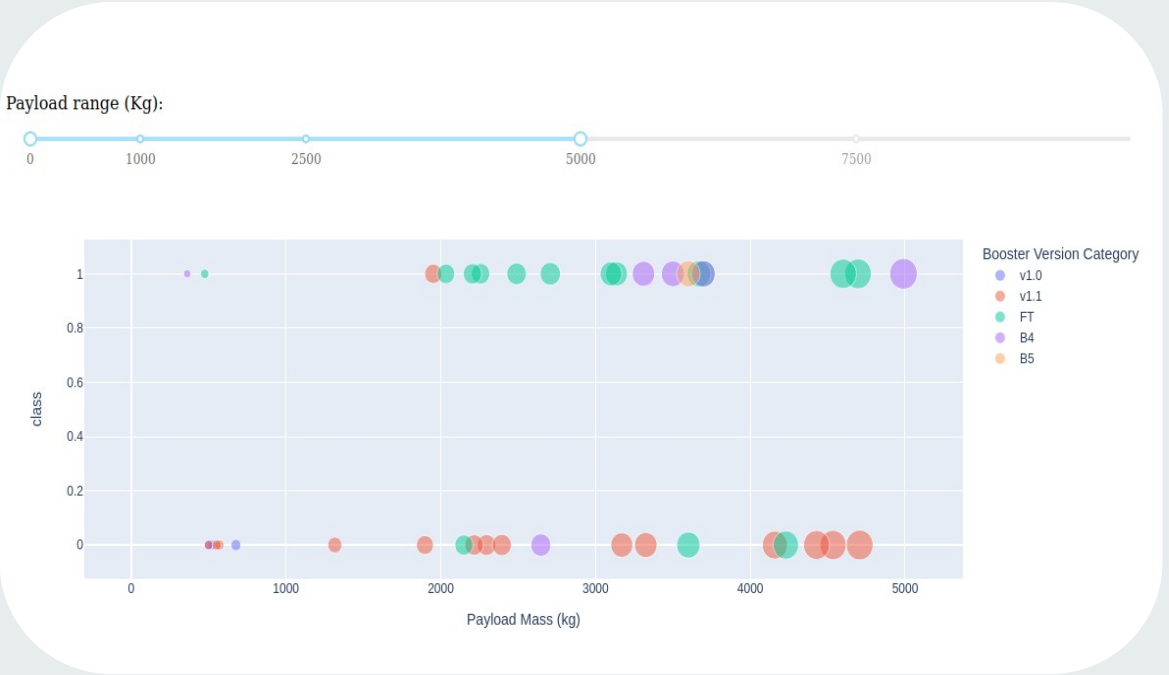
Dashboard – Success Count for all launch sites

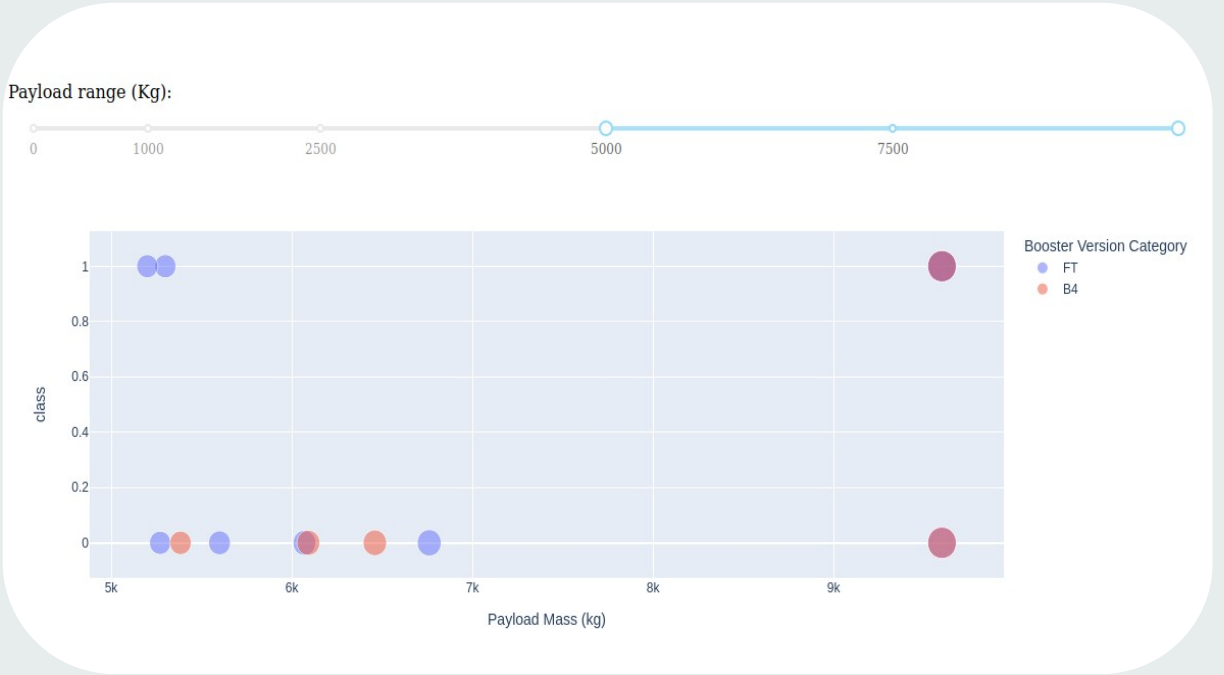
We can determine the launch site “KSC LC-39A” has had the most success



Dashboard – Highest Launch Success Ratio

At 76.9% success rate “KSC LC-39A” has the highest launch success ration





Dashboard – Payload vs Launch Outcome

The **range slider** allows us to observe the correlation between payload range.

We can determine that the success rate is lower the heavier the payload.

Section 6

Predictive Analysis (Classification)

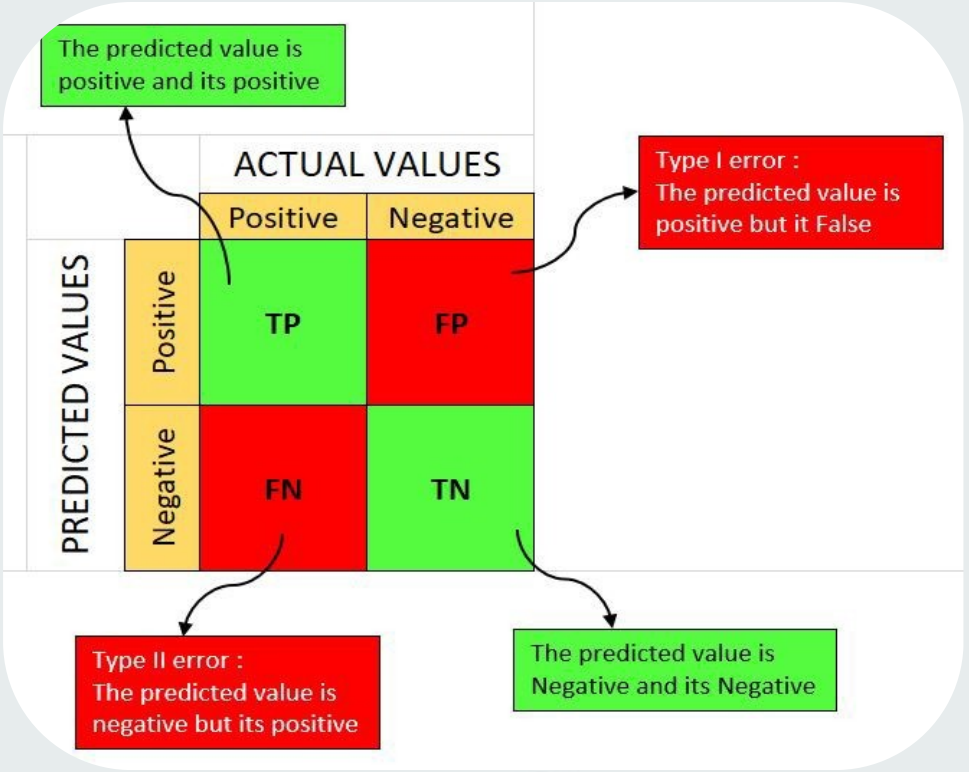
Classification Accuracy

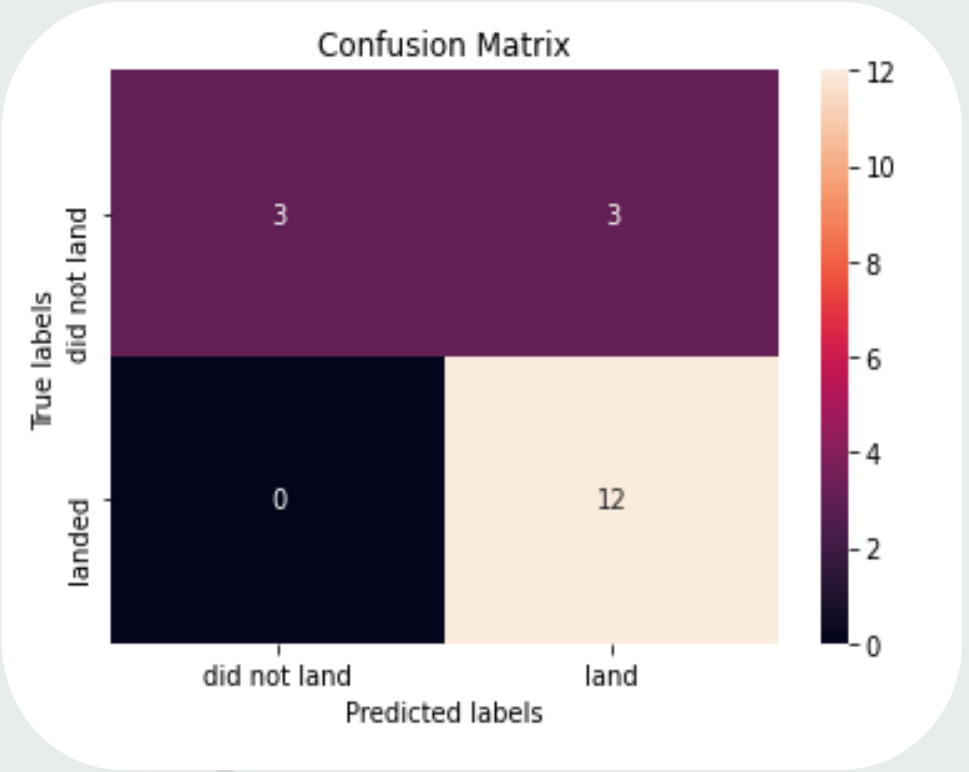
After plotting our bar chart we can deduce that the **Decision Tree** has a slight lead in accuracy over the others

Algorithm	Best Parameters	Validation Accuracy Score	Test Data Accuracy
Logistic Regression	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}	.8464	.8333
Support Vector Machine	{'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}	.8482	.8333
Decision Tree	{'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'random'}	.8892	.9444
K Nearest Neighbor	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}	.8482	.8333

Confusion Matrix

Decision Tree is our best performing model. After plotting the confusion matrix we can see that the major problem is false positive





Conclusion

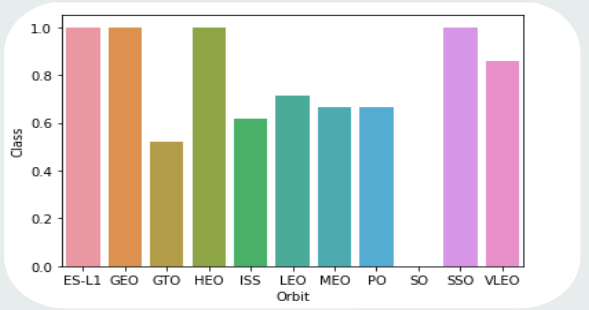
1) Orbits with highest success rates:
   - “ES-L1”,”GEO”,”HEO”,”SSO”

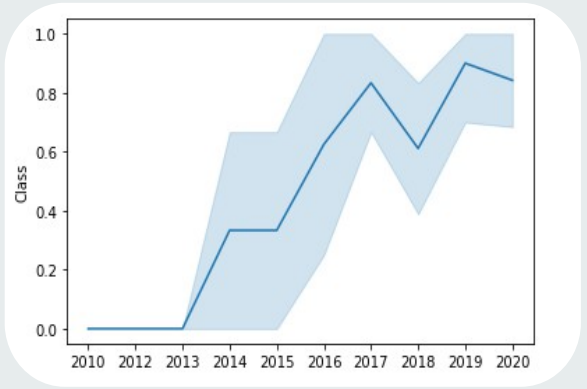
2) Yearly trends seems to indicate the more years under its belt the higher the success rate

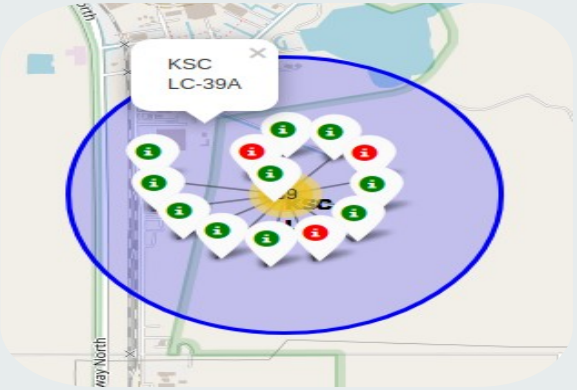
3) We determined launch site “KSC LC-39A” has the highest success rates among launch sites

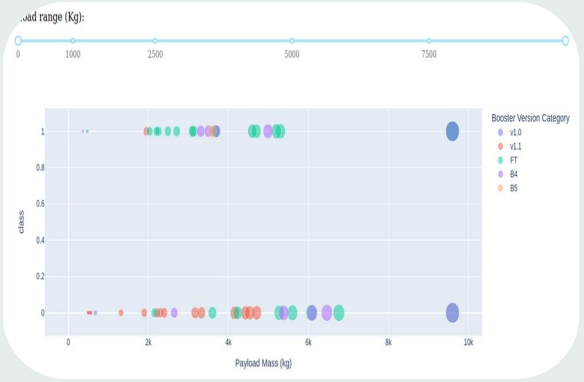
4) We determined that a lighter payload directly correlates with a higher success rate

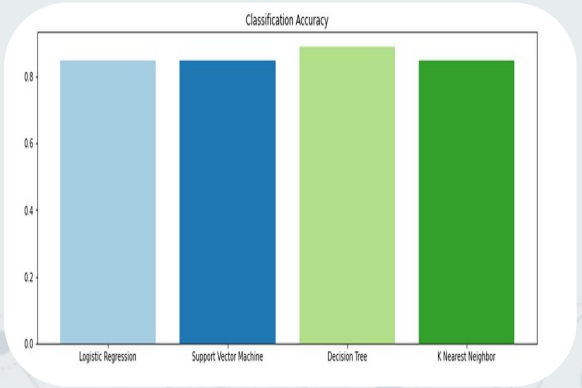
5) Best machine learning model:
   - Decision Tree Classifier











Appendix

- API
- List of Falcon 9 and Falcon Heavy Launches
- BeautifulSoup
- Folium
- Dash
- Scikit Learn

Thank you!