IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Davneet Kaur
12/16/2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API, Web Scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - EDA with SQL
  - Interactive Map with Folium
  - Dashboards with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive maps and dashboard
  - Predictive results

Introduction

- Project background and context
  - SpaceX is the most successful commercial space exploration company that is making space travel affordable. The company advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each. Much of these savings come from SpaceX's reuse of the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. Based on public information and machine learning models, we are going to predict if SpaceX will reuse the first stage.
- Problems you want to find answers
  - How do variables such as payload mass, launch site, number of flights, and orbits affect the success of the first stage landing?
  - Does the rate of successful landings increase over the years?
  - What is the best algorithm that can be used for binary classification in this case?

Section 1

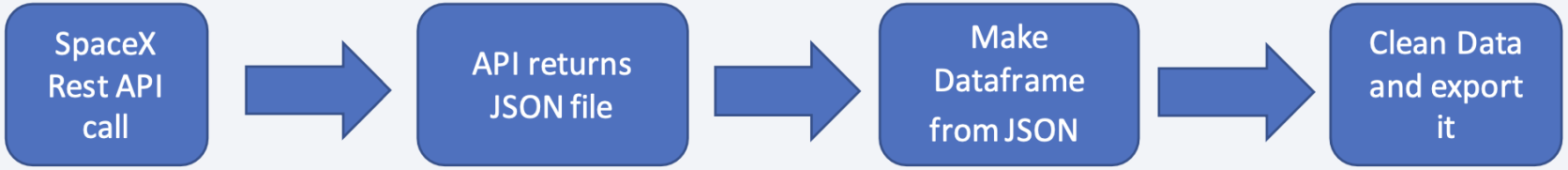
Methodology

Methodology

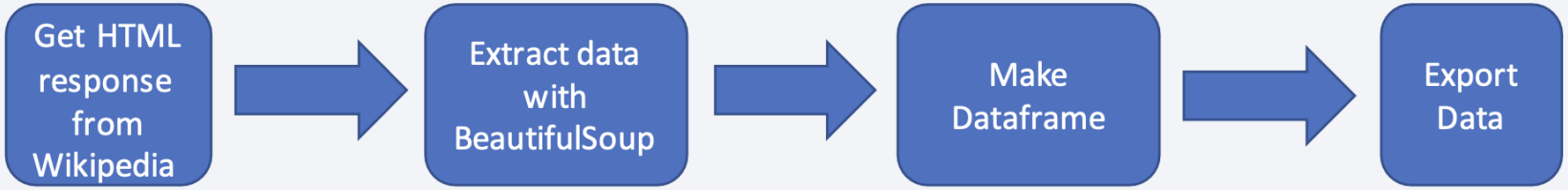
- Data collection methodology:
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Filtering the data
  - Dealing with missing values
  - Using One Hot Encoding to prepare the data to a binary classification
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Building, tuning and evaluation of classification models to ensure the best results

Data Collection

- Datasets are collected from Rest SpaceX API and webscrapping Wikipedia
  - The information obtained by the API are rocket, launches, payload information.
    - The Space X REST API URL is api.spacexdata.com/v4/



  - The information obtained by the webscrapping of Wikipedia are launches, landing, payload information.
    - URL is https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922



Data Collection – SpaceX API

1. Getting Response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Convert Response to JSON File

```
data = response.json()
data = pd.json_normalize(data)
```

3. Transform data

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

4. Create dictionary with data

```
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}
```

5. Create dataframe

```
data = pd.DataFrame.from_dict(launch_dict)
```

6. Filter dataframe

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

7. Export to file

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

[Github Link to Code]

Data Collection - Scraping

1. Getting Response from HTML

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

2. Create BeautifulSoup Object

```
soup = BeautifulSoup(response.text, "html5lib")
```

3. Find all tables

```
html_tables = soup.findAll('table')
```

4. Get column names

```
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0 :
        column_names.append(name)
```

5. Create dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Add data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is a
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.stri
                flag=flight_number.isdigit()
```

See notebook for the rest of code

7. Create dataframe from dictionary

```
df=pd.DataFrame(launch_dict)
```

8. Export to file

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

[Github Link to Code]

Data Wrangling

- In the dataset, there are several cases where the booster did not land successully.
  - True Ocean, True RTLS, True ASDS means the mission has been successful.
  - False Ocean, False RTLS, False ASDS means the mission was a failure.
- We need to transform string variables into categorical variables where 1 means the mission has been successful and 0 means the mission was a failure.

1. Calculate launches number for 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 occurence of each orbit

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

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

3. Calculate number and occurrence of mission outcome per orbit type

```
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
None ASDS       2
False Ocean     2
False RTLS      1
Name: Outcome, dtype: int64
```

4. Create landing outcome label from Outcome column

```
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
df['Class']=landing_class
```

5. Export to file

```
df.to_csv("dataset_part_2.csv", index=False)
```

[Github Link to Code]

EDA with Data Visualization

- Scatter Graphs
  - Flight Number vs. Payload Mass
  - Flight Number vs. Launch Site
  - Payload vs. Launch Site
  - Orbit vs. Flight Number
  - Payload vs. Orbit Type
  - Orbit vs. Payload Mass

Scatter plots show relationship between variables. This relationship is called the correlation.

- Bar Graph
  - Success rate vs. Orbit

Bar graphs show the relationship between numeric and categoric variables.

- Line Graph
  - Success rate vs. Year

Line graphs show data variables and their trends. Line graphs can help to show global behavior and make prediction for unseen data.

Github Link to Code

EDA with SQL

- We performed SQL queries to gather and understand the data. We display:
  - The names of the unique launch sites in the space mission.
  - 5 records where launch sites begin with the string 'CCA'
  - The total payload mass carried by boosters launched by NASA (CRS).
  - The average payload mass carried by booster version F9 v1.1.
  - The list of dates when the first successful landing outcome in ground pad was achieved.
  - The names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
  - The total number of successful and failed mission outcomes.
  - The names of the booster versions which have carried the maximum payload mass.
  - The records which will display the month, names, landing outcomes in drone ship, booster versions, launch site for the months in year 2015.
  - A ranking of the counts of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Github Link to Code

Build an Interactive Map with Folium

- Folium map object is a map centered on NASA Johnson Space Center at Houson, Texas
  - Red circle at NASA Johnson Space Center's coordinate with label showing its name (folium.Circle, folium.map.Marker).
  - Red circles at each launch site coordinates with label showing launch site name (folium.Circle, folium.map.Marker, folium.features.DivIcon).
  - The grouping of points in a cluster to display multiple and different information for the same coordinates (folium.plugins.MarkerCluster).
  - Markers to show successful and unsuccessful landings. Green for successful landing and Red for unsuccessful landing (folium.map.Marker, folium.Icon).
  - Markers to show distance between launch site to key locations (railway, highway, coastway, city) and plot a line between them (folium.map.Marker, folium.PolyLine, folium.features.DivIcon)
- These objects are created in order to understand better the problem and the data. We can show easily all launch sites, their surroundings and the number of successful and unsuccessful landings.

Github Link to Code

Build a Dashboard with Plotly Dash

- Dashboard has dropdown, pie chart, rangeslider and scatter plot components
  - Dropdown allows a user to choose the launch site or all launch sites (dash core components Dropdown).
  - Pie chart shows the total success and the total failure for the launch site chosen with the dropdown component (plotly.express.pie) .
  - Rangeslider allows a user to select a payload mass in a fixed range (dash core components RangeSlider) .
  - Scatter chart shows the relationship between two variables, in particular Success vs Payload Mass (plotly.express.scatter) .

Github Link to Code

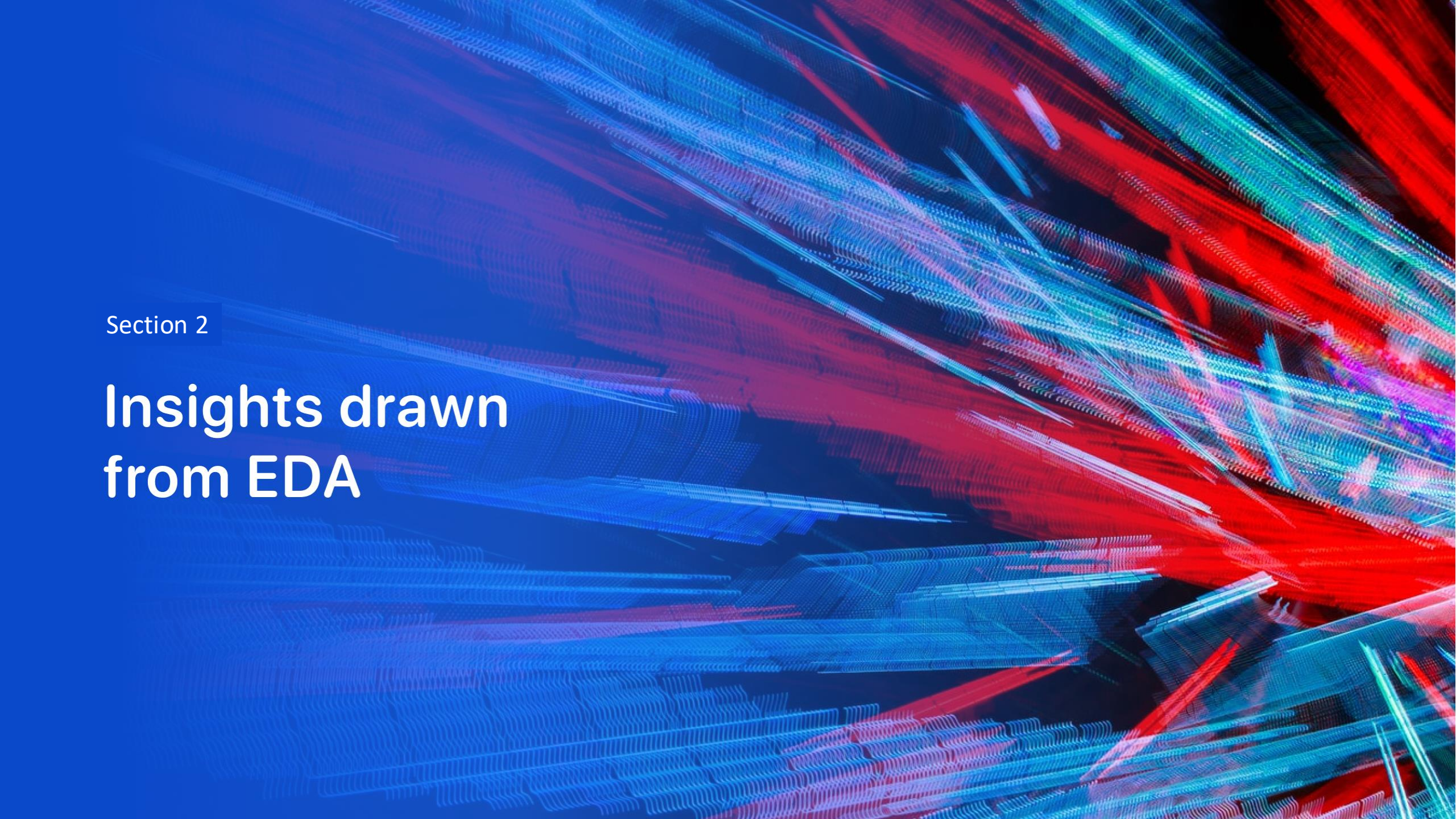
Predictive Analysis (Classification)

- Data preparation
  - Load dataset
  - Normalize data
  - Split data into training and test sets.
- Model preparation
  - Selection of machine learning algorithms
  - Set parameters for each algorithm to GridSearchCV
  - Training GridSearchModel models with training dataset
- Model evaluation
  - Get best hyperparameters for each type of model
  - Compute accuracy for each model with test dataset
  - Plot Confusion Matrix
- Model comparison
  - Comparison of models according to their accuracy
  - The model with the best accuracy will be chosen (see Notebook for result)

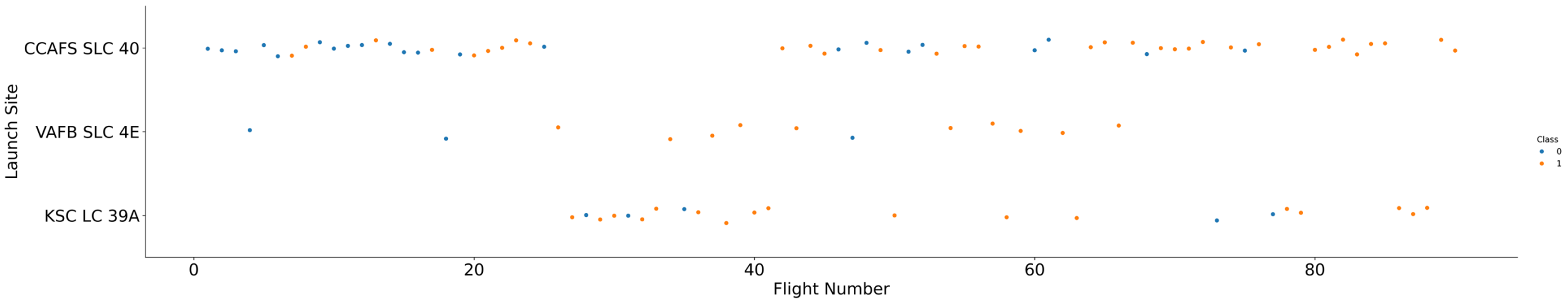
Github Link to Code

Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

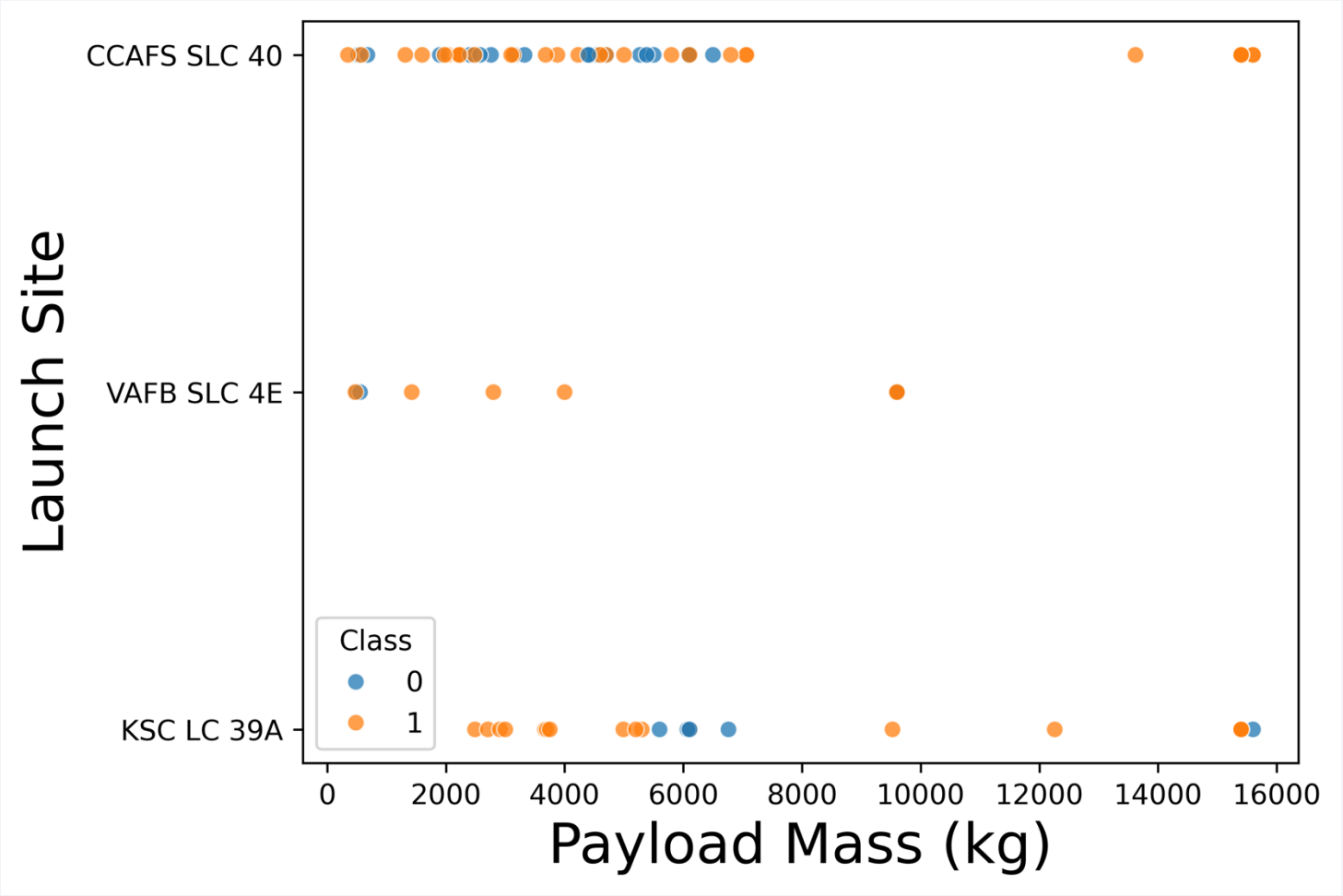



```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

As we follow the x-axis, the frequency of orange points increases relative to the frequency of blue points for each launch site. Thus, the success rate of each launch site is increasing.

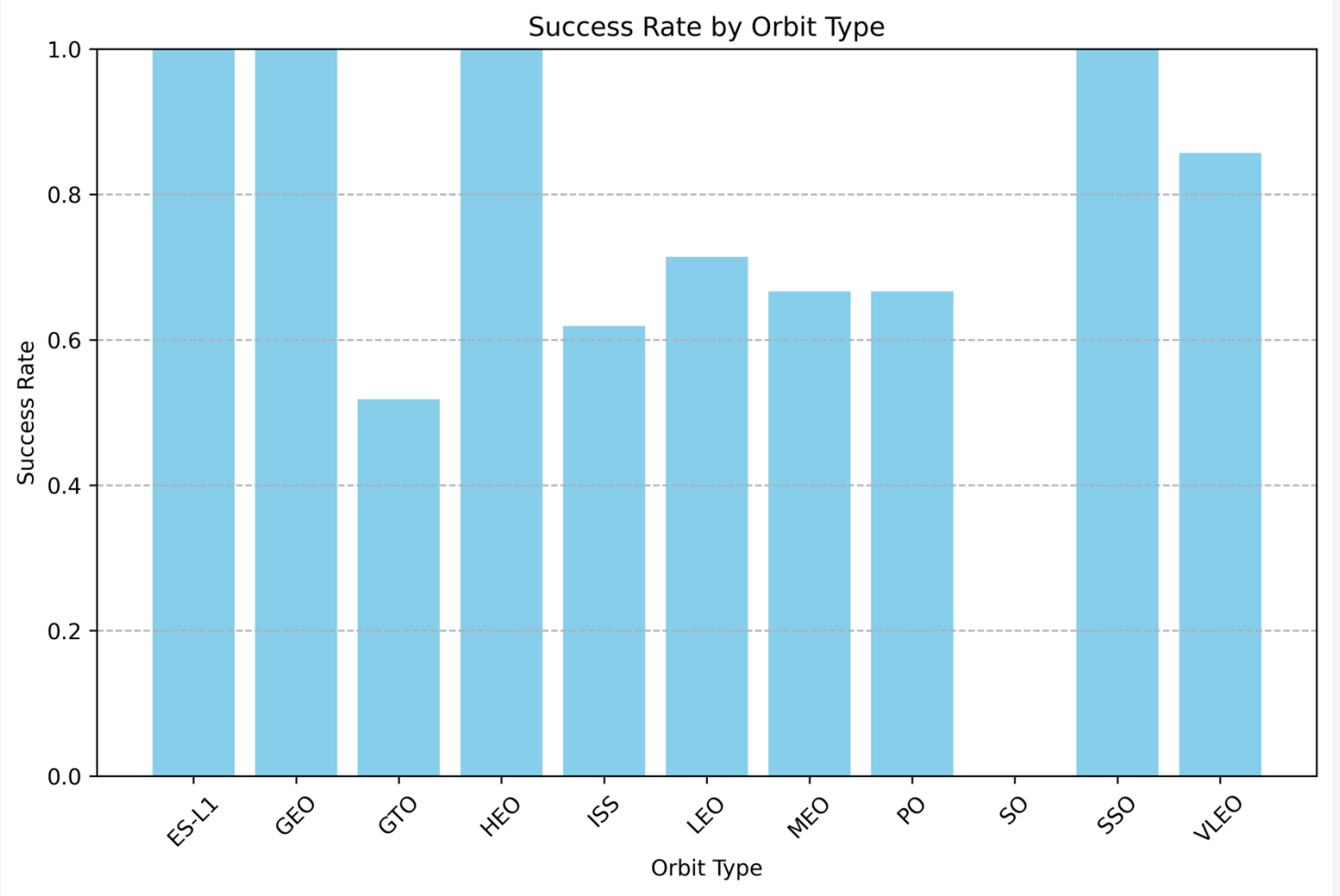
Payload vs. Launch Site

- For the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).
- There might be a correlation between playload mass and launch success.

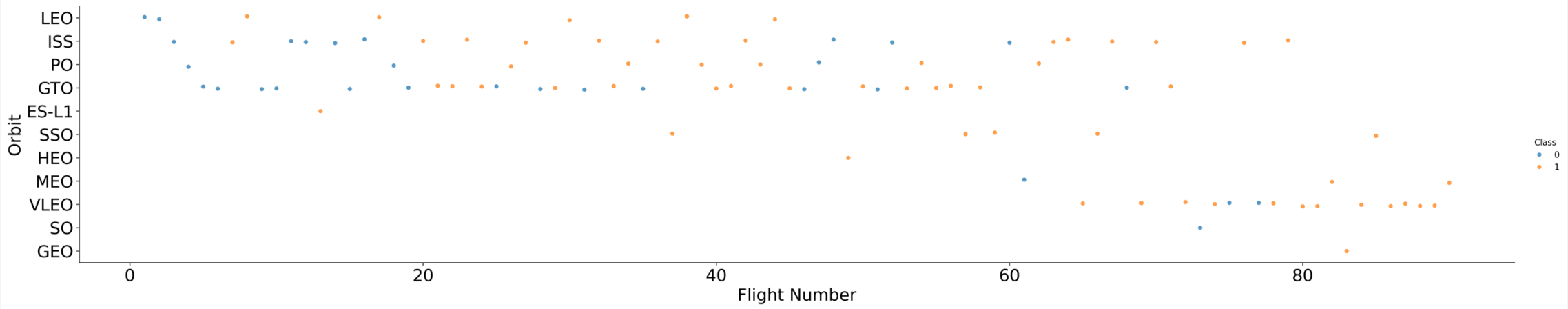


Success Rate vs. Orbit Type

- With this plot, we can see success rate for different orbit types. We note that ES-L1, GEO, HEO, SSO have the best success rate.

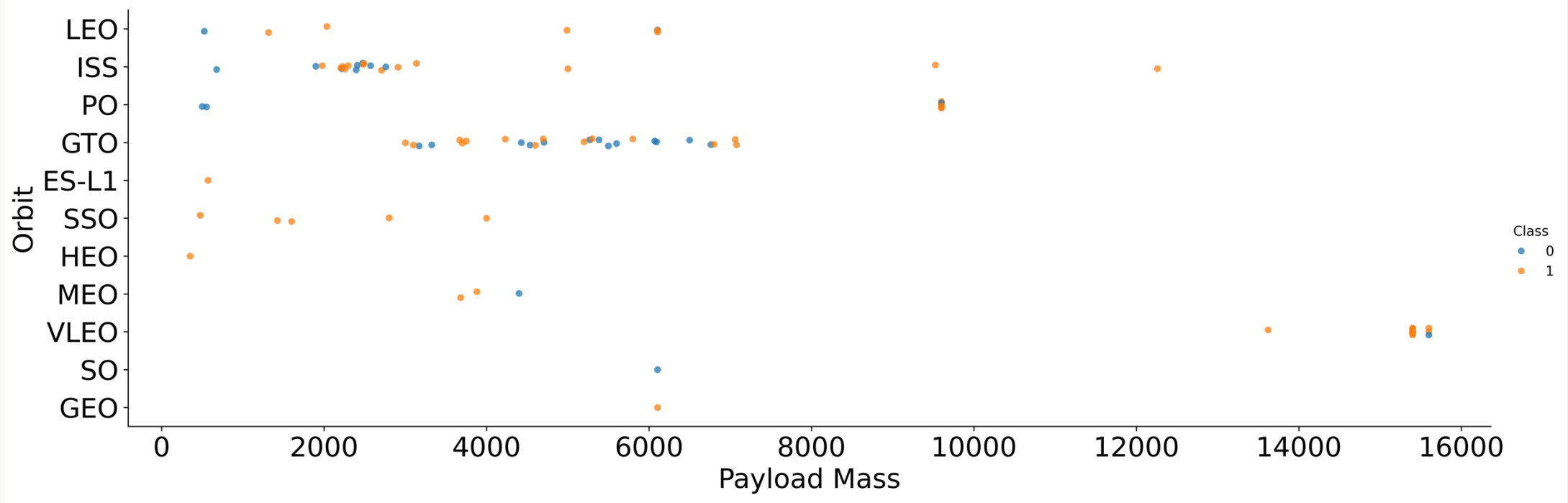


Flight Number vs. Orbit Type



We notice that the first few flights are all failures. The first many flights all belong to 4 categories of orbits. Once the success rate of flights for those orbits increases, other orbit types are also attempted. There seems to be a positive learning rate with flight number as the success rate increases with time.

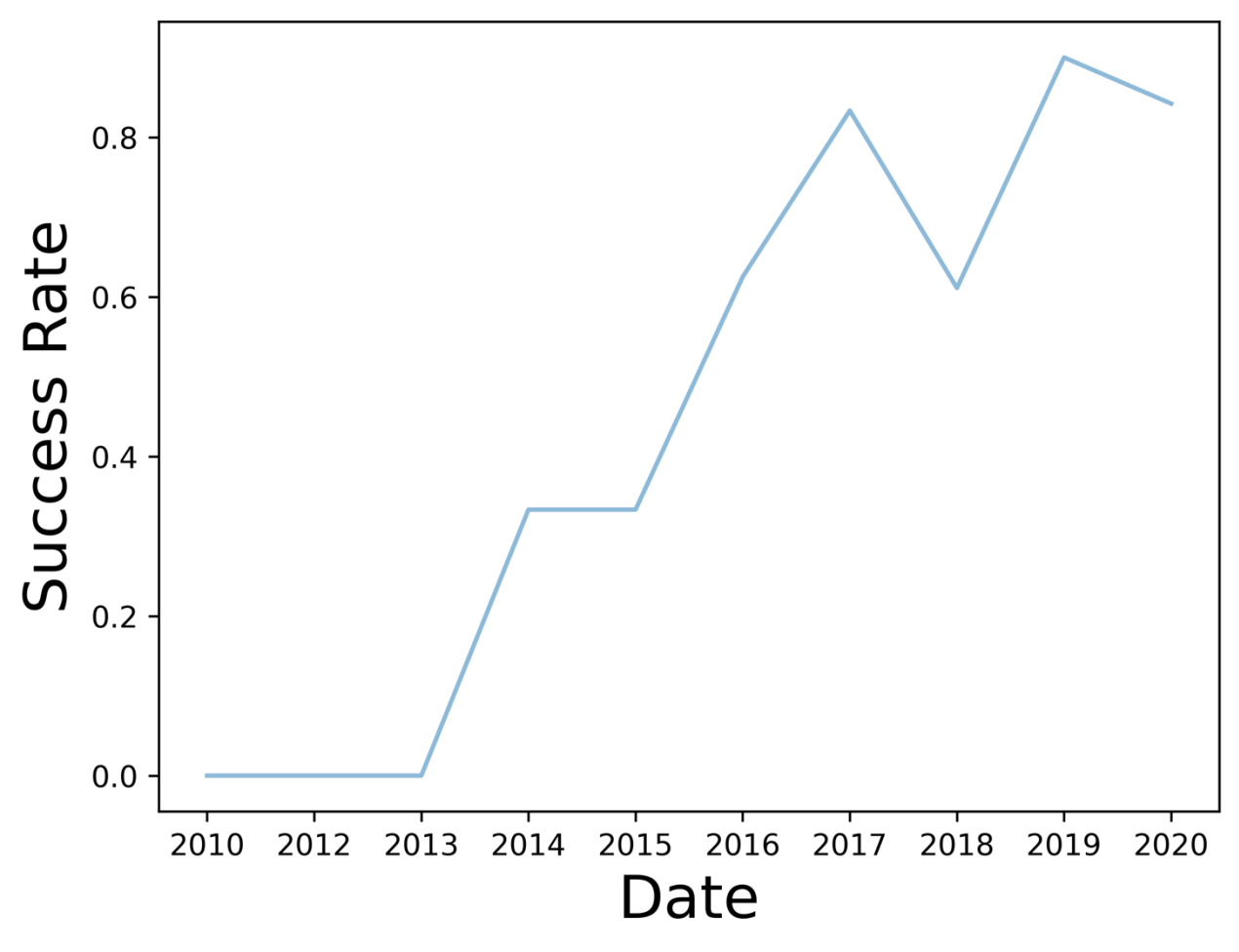
Payload vs. Orbit Type



Certain orbits have higher payload masses, such as VLEO. This may have to do with the requirements of Newton's Law's of Physics.

Launch Success Yearly Trend

- The success rate of SpaceX launches has been increasing since 2013. There was a lag period of 2-years before SpaceX had a significant number of launch successes.



All Launch Site Names

The use of DISTINCT in the query allows to remove duplicate LAUNCH_SITE.

```
%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'

The WHERE clause followed by LIKE clause filters launch sites that contain the substring CCA. LIMIT 5 shows 5 records from filtering.

Task 2

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

```
[11]: %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

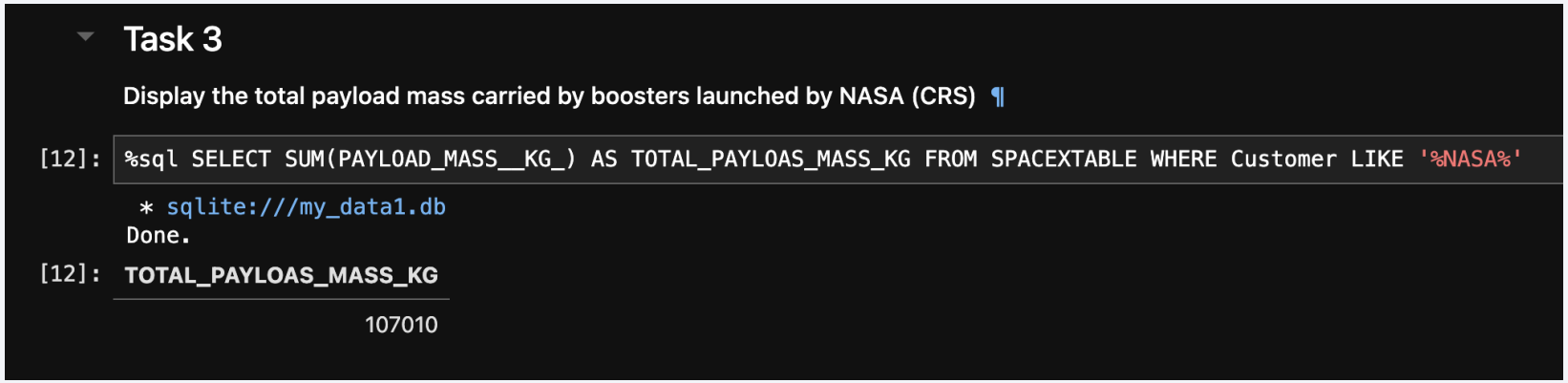
```
 * sqlite:///my_data1.db
Done.
```

[11]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

This query returns the sum of all payload masses where the customer is NASA (CRS).



Average Payload Mass by F9 v1.1

This query returns the average of all payload masses where the booster version contains the substring F9 v1.1.

Task 4

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

```
[13]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVE_PAYLOAS_MASS_KG FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%'
 * sqlite:///my_data1.db
Done.
[13]: AVE_PAYLOAS_MASS_KG
      2534.6666666666665
```

First Successful Ground Landing Date

With this query, we select the oldest successful landing. The WHERE clause filters dataset in order to keep only records where landing was successful. With the MIN function, we select the record with the oldest date.

Task 5

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

Hint:Use min function

```
%sql SELECT Date, Landing_Outcome FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success%' LIMIT 1
```

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

Date	Landing_Outcome
2015-12-22	Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT DISTINCT(Booster_Version) FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%' /
AND PAYLOAD_MASS__KG_> 4000 AND PAYLOAD_MASS__KG_<6000
```

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

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 B4 B1043.1
F9 FT B1032.2
F9 B4 B1040.2
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5B1054
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

This query returns the booster version where landing was successful and payload mass is between 4000 and 6000 kg. The WHERE and AND clauses filter the dataset.

Total Number of Successful and Failure Mission Outcomes

The first query counts the successful mission. The second query counts the unsuccessful mission. The WHERE clause followed by LIKE clause filters mission outcome. The COUNT function counts records filtered.

```
%sql SELECT COUNT(Mission_Outcome) FROM SPACEXTABLE WHERE Mission_Outcome LIKE '%Success%'
```

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

COUNT(Mission_Outcome)
100

```
%sql SELECT COUNT(Mission_Outcome) FROM SPACEXTABLE WHERE Mission_Outcome LIKE '%Failure%'
```

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

COUNT(Mission_Outcome)
1

Boosters Carried Maximum Payload

I used a subquery to filter data by returning only the heaviest payload mass with MAX function. The main query uses subquery results and returns unique booster version (SELECT DISTINCT) with the heaviest payload mass.

```
%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

- This query returns month, booster version, launch site where landing was unsuccessful and landing date took place in 2015.

```
%sql SELECT substr(Date,6,2) AS Month, Booster_Version, Landing_Outcome, Launch_Site  FROM SPACEXTABLE WHERE substr(Date,0,5)='2015'
 * sqlite:///my_data1.db
Done.
```

Month	Booster_Version	Landing_Outcome	Launch_Site
01	F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
02	F9 v1.1 B1013	Controlled (ocean)	CCAFS LC-40
03	F9 v1.1 B1014	No attempt	CCAFS LC-40
04	F9 v1.1 B1015	Failure (drone ship)	CCAFS LC-40
04	F9 v1.1 B1016	No attempt	CCAFS LC-40
06	F9 v1.1 B1018	Precluded (drone ship)	CCAFS LC-40
12	F9 FT B1019	Success (ground pad)	CCAFS LC-40

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

This query returns landing outcomes and their count where mission was successful and date is between 04/06/2010 and 20/03/2017. The GROUP BY clause groups results by landing outcome and ORDER BY COUNT DESC shows results in decreasing order.

```
%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) AS Landing_Count FROM SPACEXTABLE GROUP BY Landing_Outcome \
HAVING Date > '2010-06-04' AND DATE< '2017-03-20' ORDER BY Landing_Count DESC
```

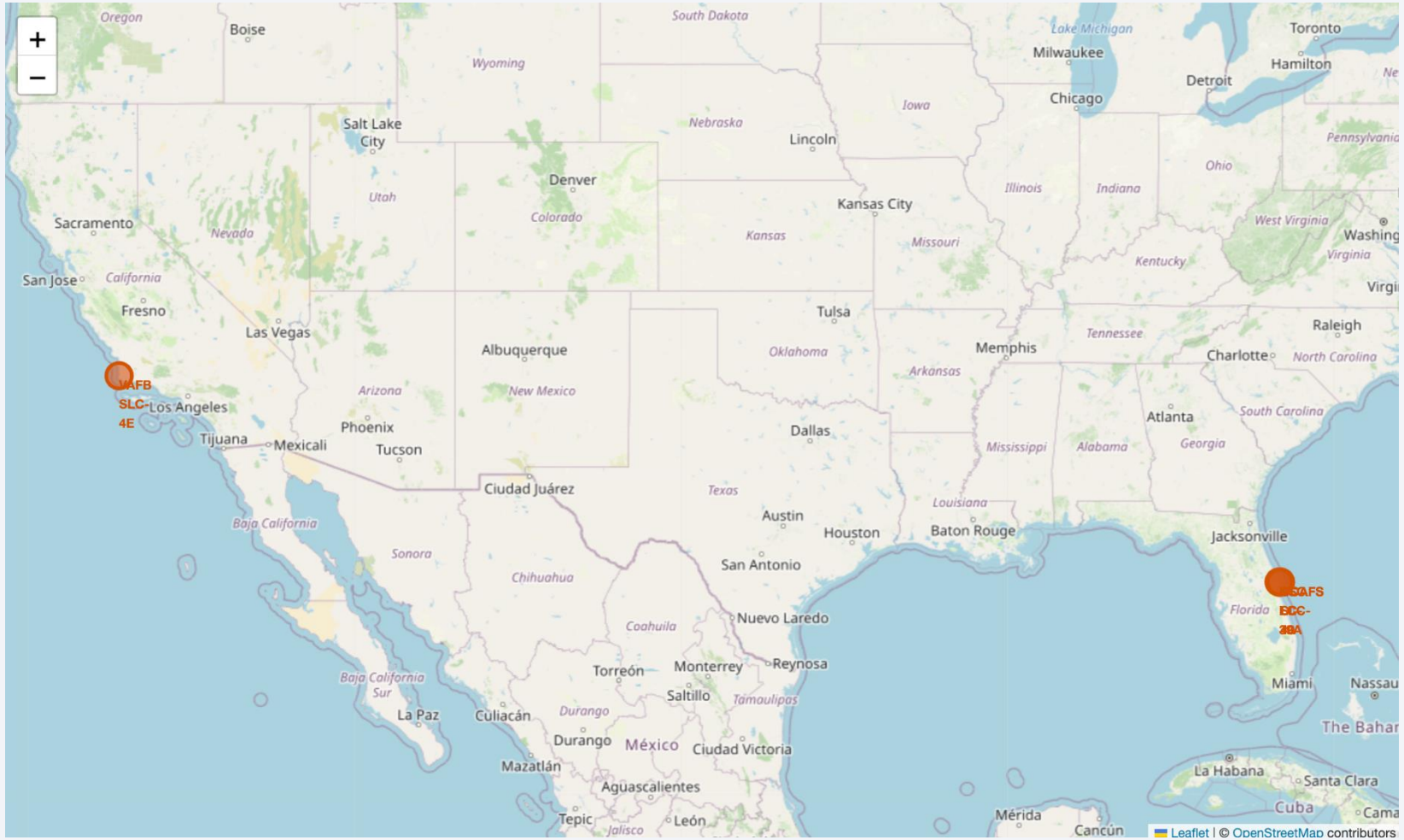
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	Landing_Count
No attempt	21
Success (drone ship)	14
Success (ground pad)	9
Failure (drone ship)	5
Controlled (ocean)	5
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

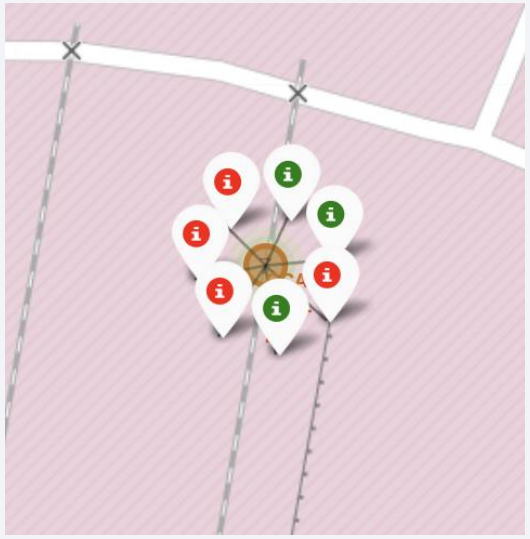
<Folium Map Screenshot 1>

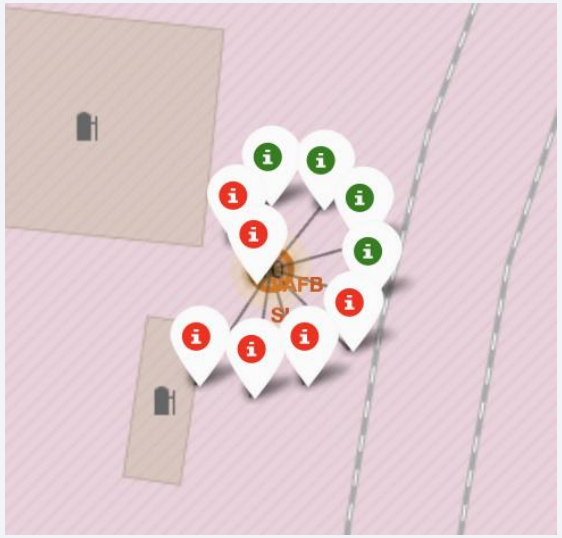


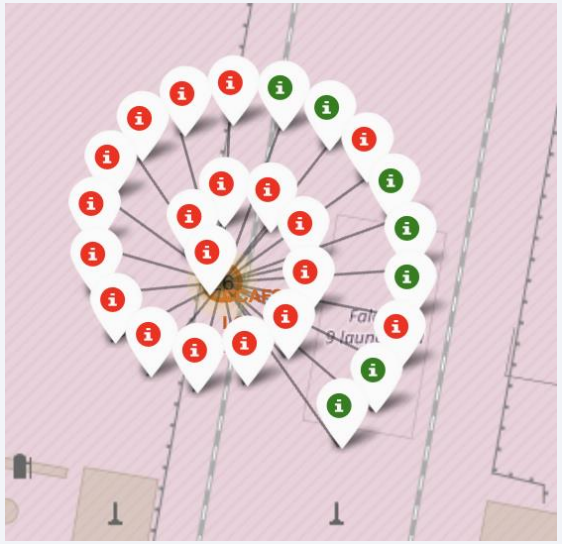
We see that Space X launch sites are located on the coast of the United States

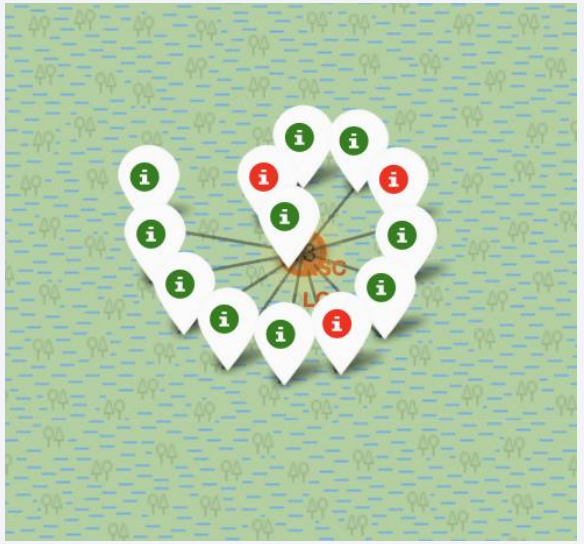
<Folium Map Screenshot 2>

- Green marker represents successful launches.
- Red marker represents unsuccessful launches.
- KSC LC-39A has a higher launch success rate (rightmost).

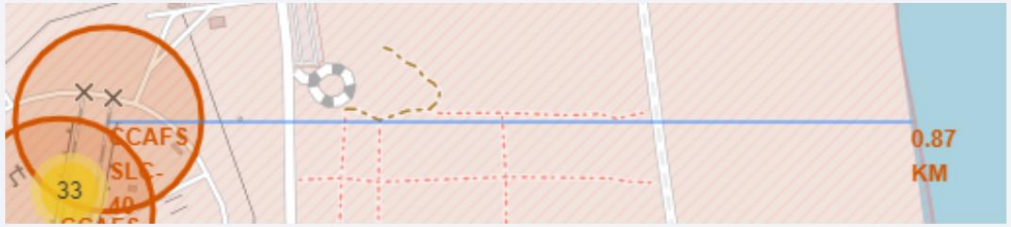


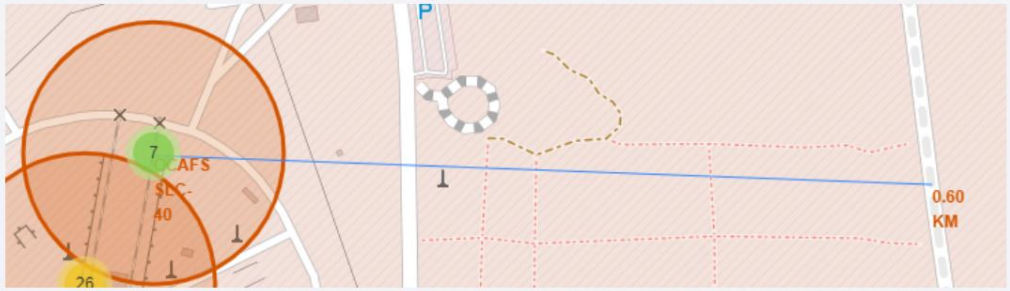




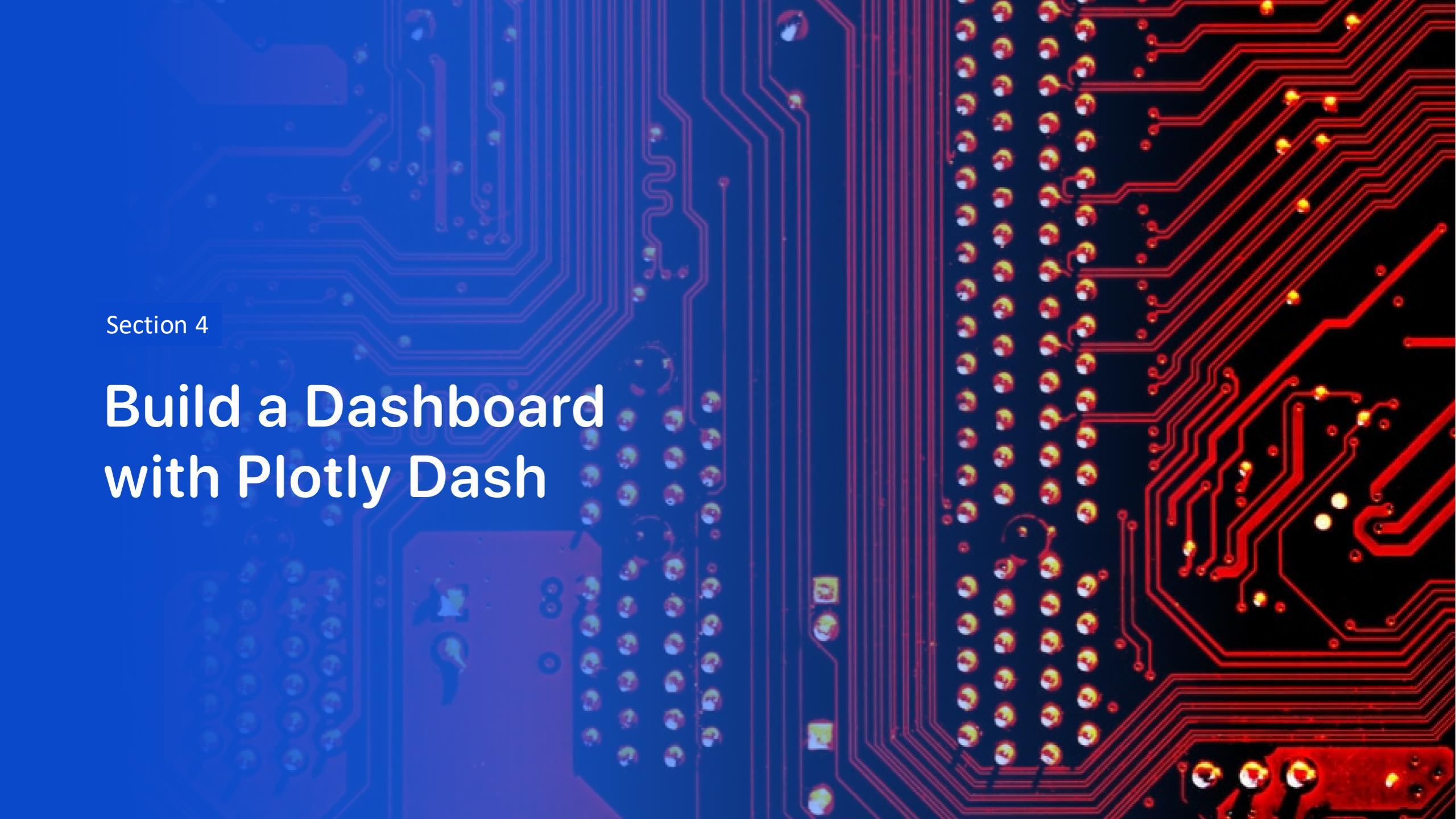


<Folium Map Screenshot 3>



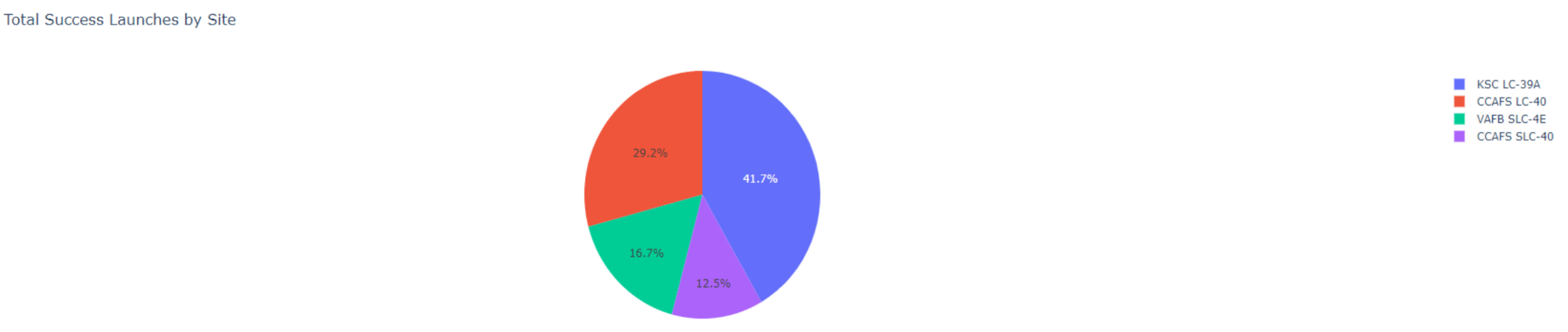


- Is CCAFS SLC-40 in close proximity to railways ? Yes
- Is CCAFS SLC-40 in close proximity to highways ? Yes
- Is CCAFS SLC-40 in close proximity to coastline ? Yes
- Do CCAFS SLC-40 keeps certain distance away from cities ? No


Section 4
Build a Dashboard with Plotly Dash

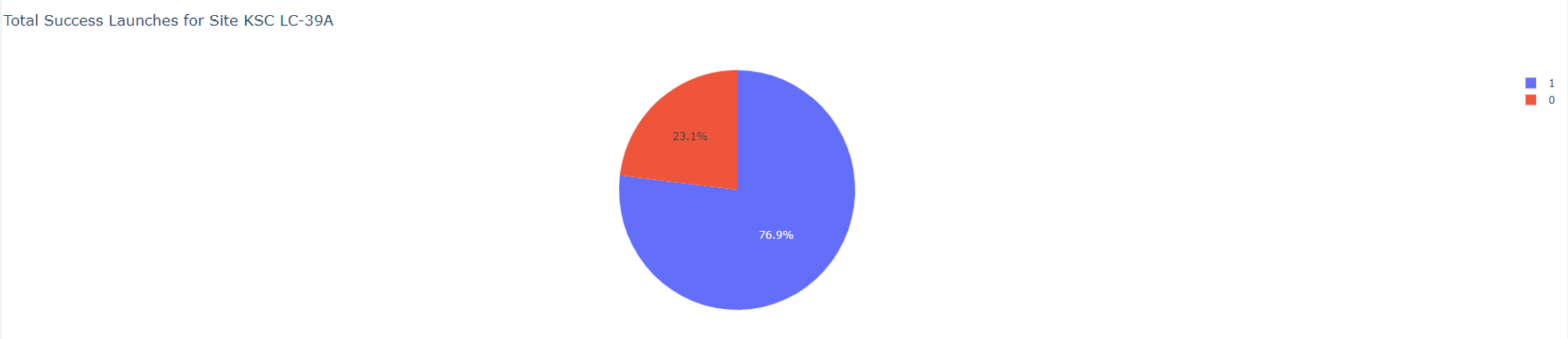
<Dashboard Screenshot 1>

We see that KSC LC-39A has the best success rate of launches.

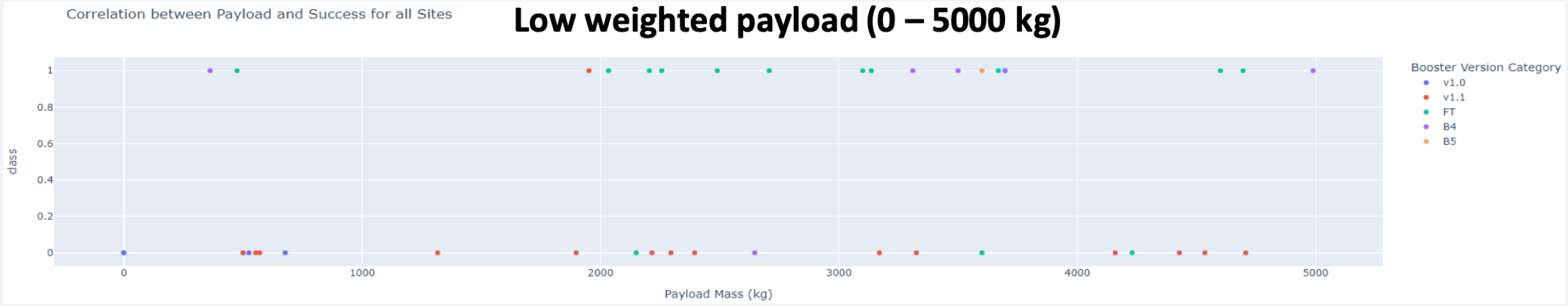


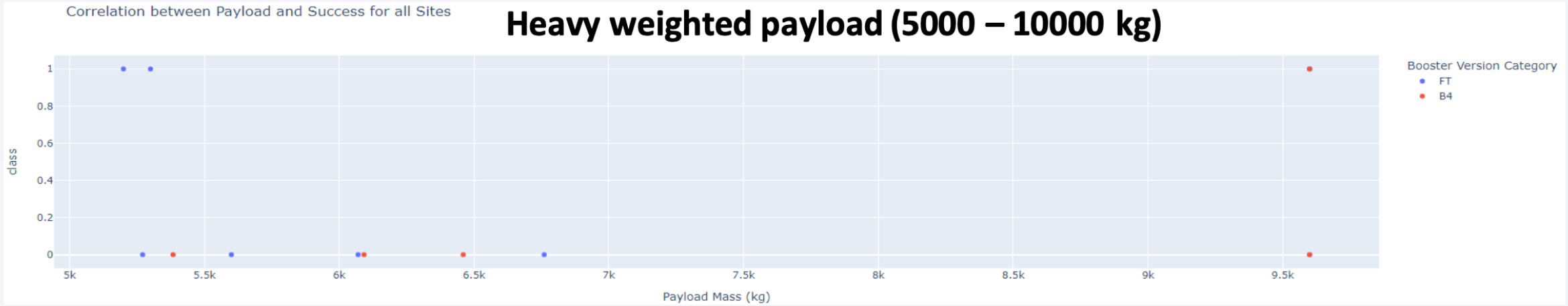
<Dashboard Screenshot 2>

We see that KSC LC-39A has achieved a 76.9% success rate while getting a 23.1% failure rate.



<Dashboard Screenshot 3>





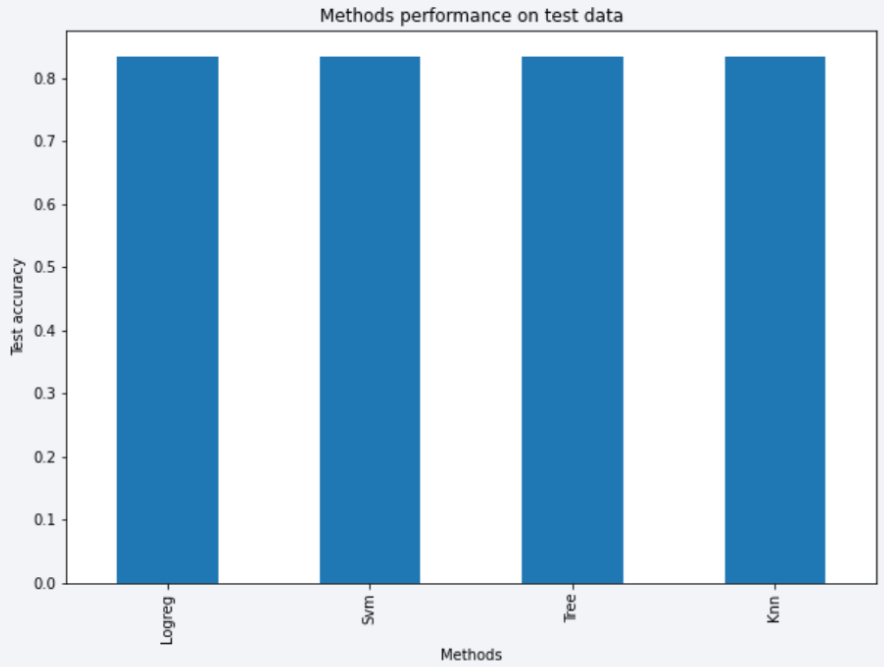
Low weighted payloads have a better success rate than the heavy weighted payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

	Accuracy Train	Accuracy Test
Tree	0.876786	0.833333
Knn	0.848214	0.833333
Svm	0.848214	0.833333
Logreg	0.846429	0.833333



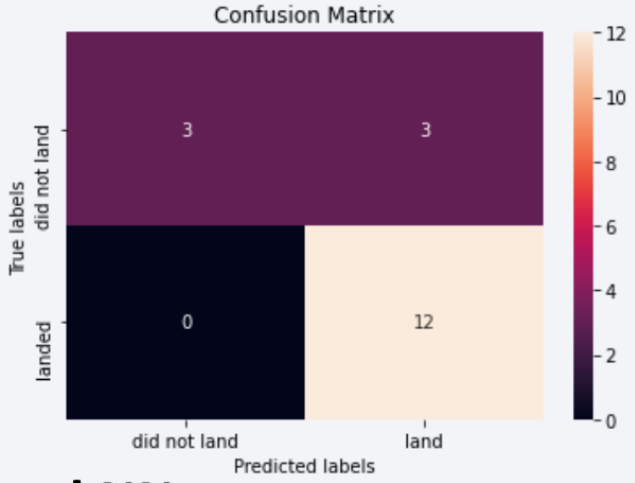
Decision tree best parameters

```
tuned hyperparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf':
4, 'min_samples_split': 2, 'splitter': 'random'}
```

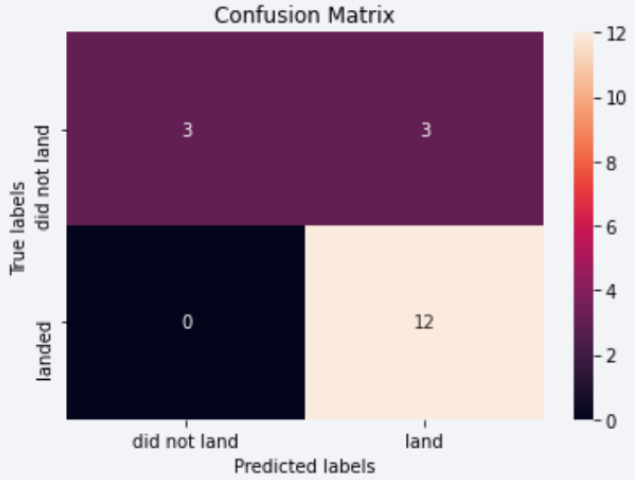
For accuracy test, all methods performed similarly. We could get more test data to decide between them. However, based on the training accuracy, we pick the decision tree.

Confusion Matrix

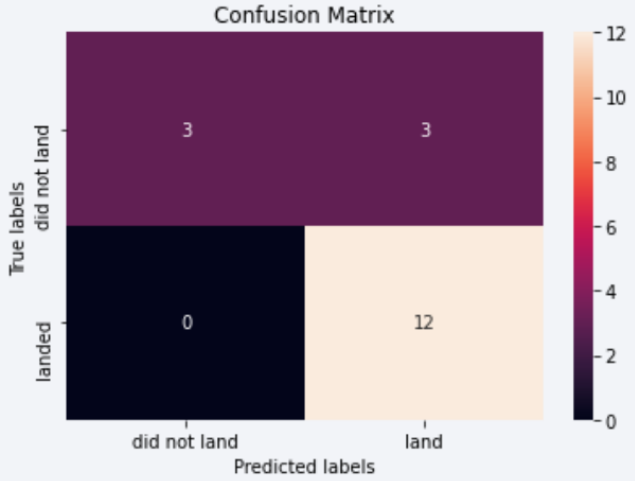
Logistic regression



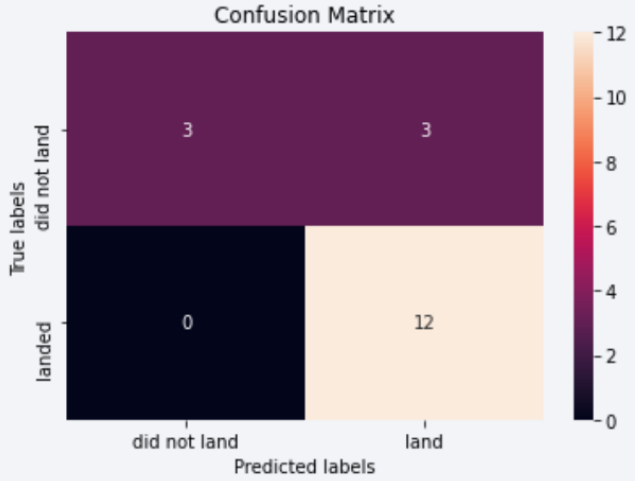
Decision Tree



kNN



SVM



As the test accuracy are all equal, the confusion matrices are also identical. The main issue with these models is the high number of false positives.

Conclusions

- The success of a mission is correlated with several factors such as the launch site, the orbit and especially the number of previous launches. We can assume that the gain in knowledge with every flight number allowed SpaceX to increase their success rate with time.
- The orbits with the best success rates are GEO, HEO, SSO, ES-L1.
- Depending on the orbits, the payload mass can be a criterion to take into account for the success of a mission. Some orbits require a light or heavy payload mass.
- KSC LC-39A is the best performing launch site. This may be because it's a launch site that opened later, when the company's launch success rate is better.
- Decision Tree Algorithm is chosen as the best model because it has the best train accuracy., even though the test accuracy between all the models is similar.

Thank you!