IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nikhil Rao
5th October 2022

Outline

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
  The aim of this project is to predict if the Falcon 9 first stage will successfully land. SpaceX says on its website that the Falcon 9 rocket launch cost 62 million dollars. Other providers cost upward of 165 million dollars each. The price difference is explained by the fact that SpaceX can reuse the first stage. By determining if the stage will land, we can determine the cost of a launch. This information is interesting for another company if it wants to compete with SpaceX for a rocket launch.
- Problems you want to find answers

  What are the main characteristics of a successful or failed landing ? • What are the effects of each relationship of the rocket variables on the success or failure of a landing ? • What are the conditions which will allow SpaceX to achieve the best landing success rate ?

Section 1

Methodology

Methodology

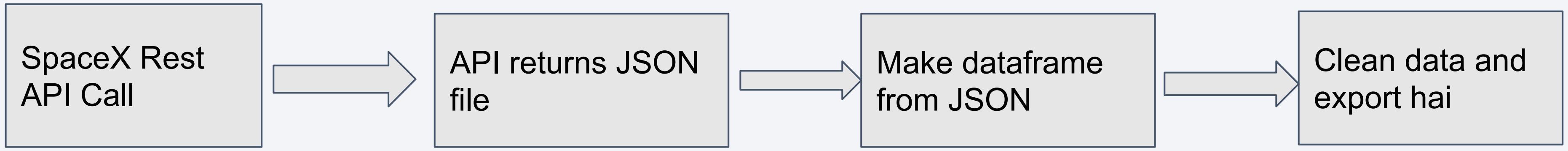
Executive Summary

- Data collection methodology:
  - SpaceX REST API • Web Scraping (Source: Wikipedia)
- Perform data wrangling
  - Dropping unnecessary columns • One Hot Encoding for classification 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

Data Collection

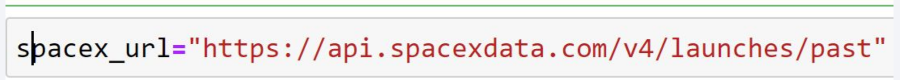
- Describe how data sets were collected.

  The information obtained by the API are rocket, launches, payload information. The information obtained by the webscraping Wikipedia URLs are launches, landing, payload information.

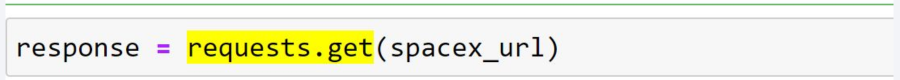


Data Collection – SpaceX API

- Getting Response from API

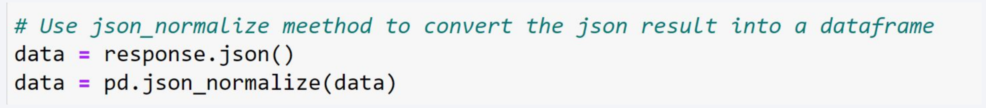


```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```



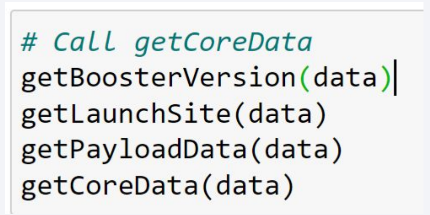
```
response = requests.get(spacex_url)
```

- Convert Response to JSON File



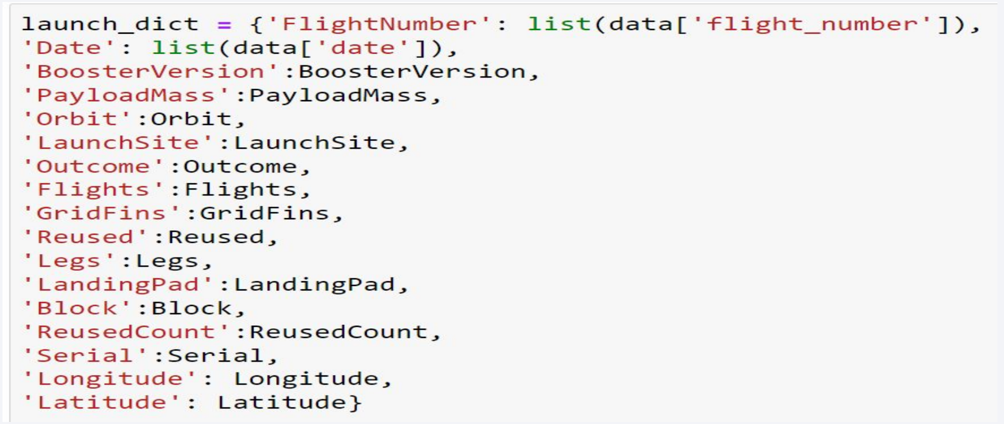
```
# Use json_normalize meethod to convert the json result into a dataframe
data = response.json()
data = pd.json_normalize(data)
```

- Transform data



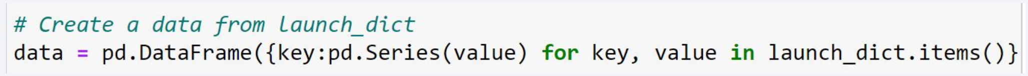
```
# Call getCoreData
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

- Create dictionary with data

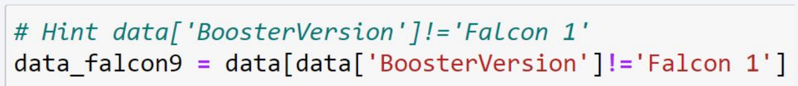


```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,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

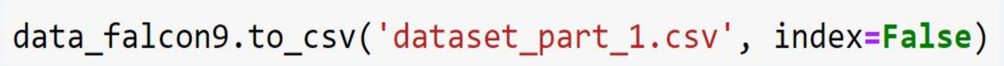
- Create and filter dataframe and export to file



```
# Create a data from launch_dict
data = pd.DataFrame({key:pd.Series(value) for key, value in launch_dict.items()})
```



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



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

Data Collection - Scraping

- Getting Response from HTML

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

Create BeautifulSoup Object

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

Find all tables

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

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)
```

Create dataframe from dictionary

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

Creating 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']=[]
```

Add data to keys, export to file

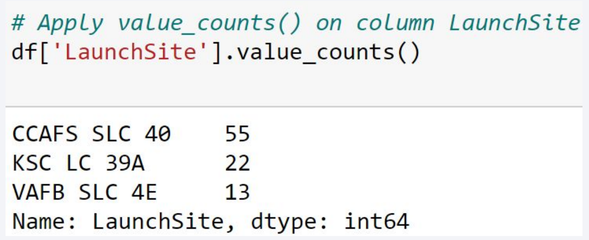
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowhea
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launc
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
```

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

Data Wrangling

- In the dataset, there are several cases where the booster did not land successfully. 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.

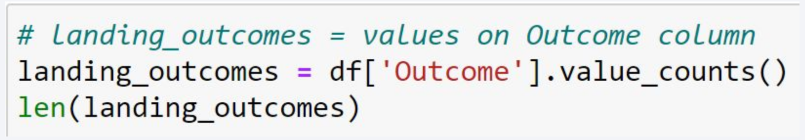
Calculate launches number for each site



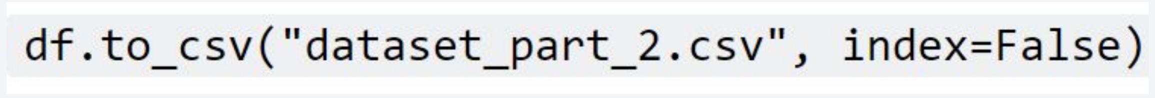
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

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

. Calculate number and occurrence of mission outcome per orbit

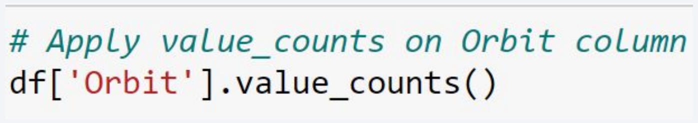


```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
len(landing_outcomes)
```



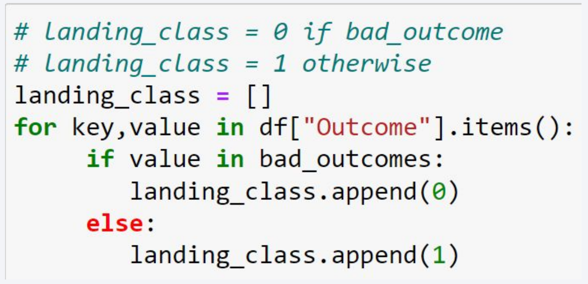
```
df.to_csv("dataset_part_2.csv", index=False)
```

Calculate the number and occurence of each orbit



```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

Create landing outcome label from Outcome column, export file

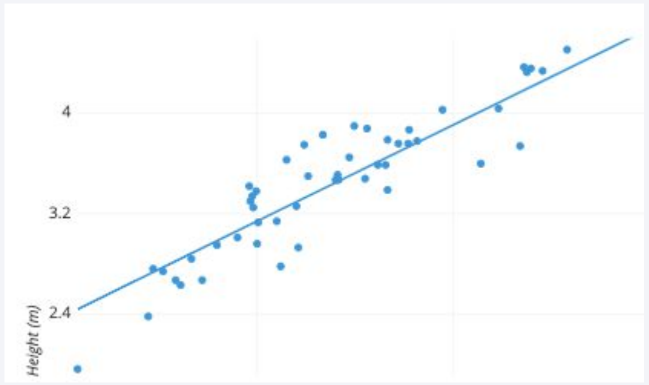


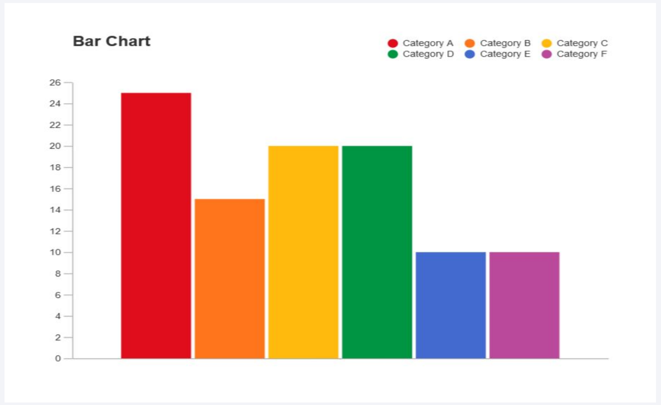
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key,value in df["Outcome"].items():
     if value in bad_outcomes:
        landing_class.append(0)
     else:
        landing_class.append(1)
```

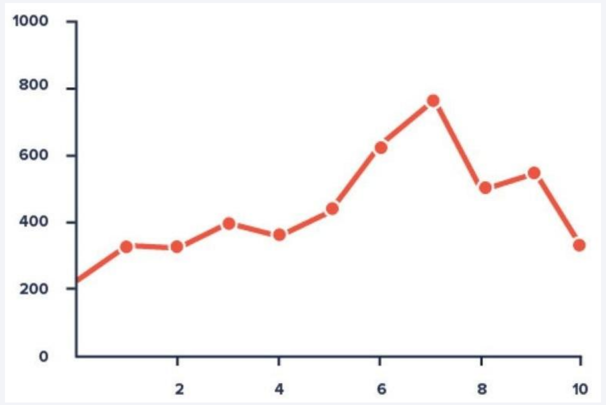
EDA with Data Visualization

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 correlation between variables.







Bar Graph • Success rate vs. Orbit

Bar graphs show the relationship between

numeric and categorical variables

Line Graph • Success rate vs. Year

Line graphs show data variables and their trends.

and show global behavior and make prediction

EDA with SQL

- We performed SQL queries to gather and understand data from dataset: Displaying 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. 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. Rank the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

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.

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)

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)

Results

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

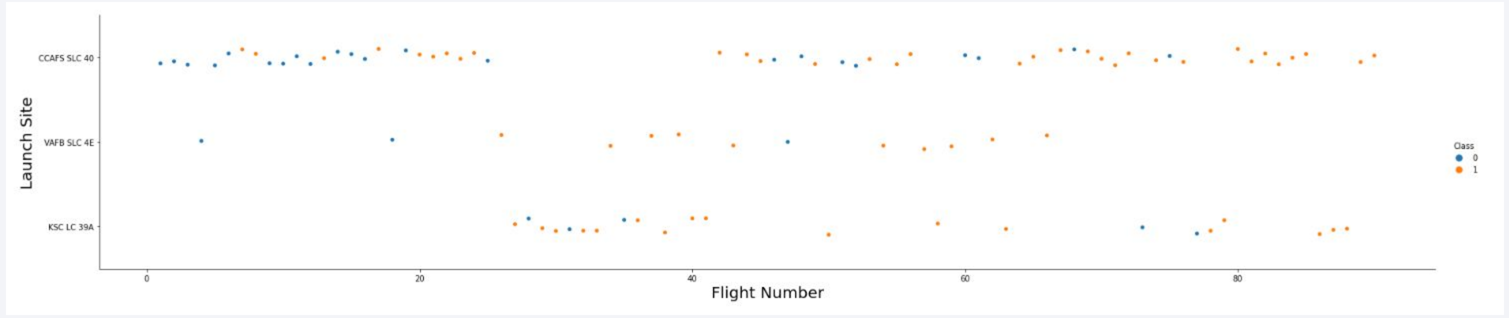
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site

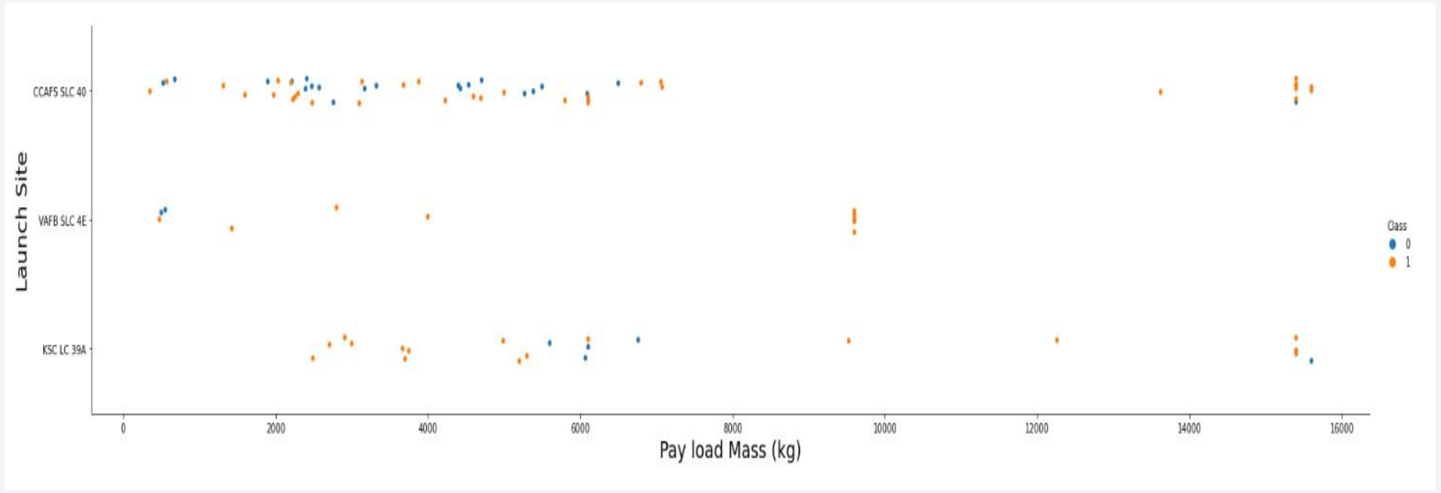
- Show the screenshot of the scatter plot with explanations



Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

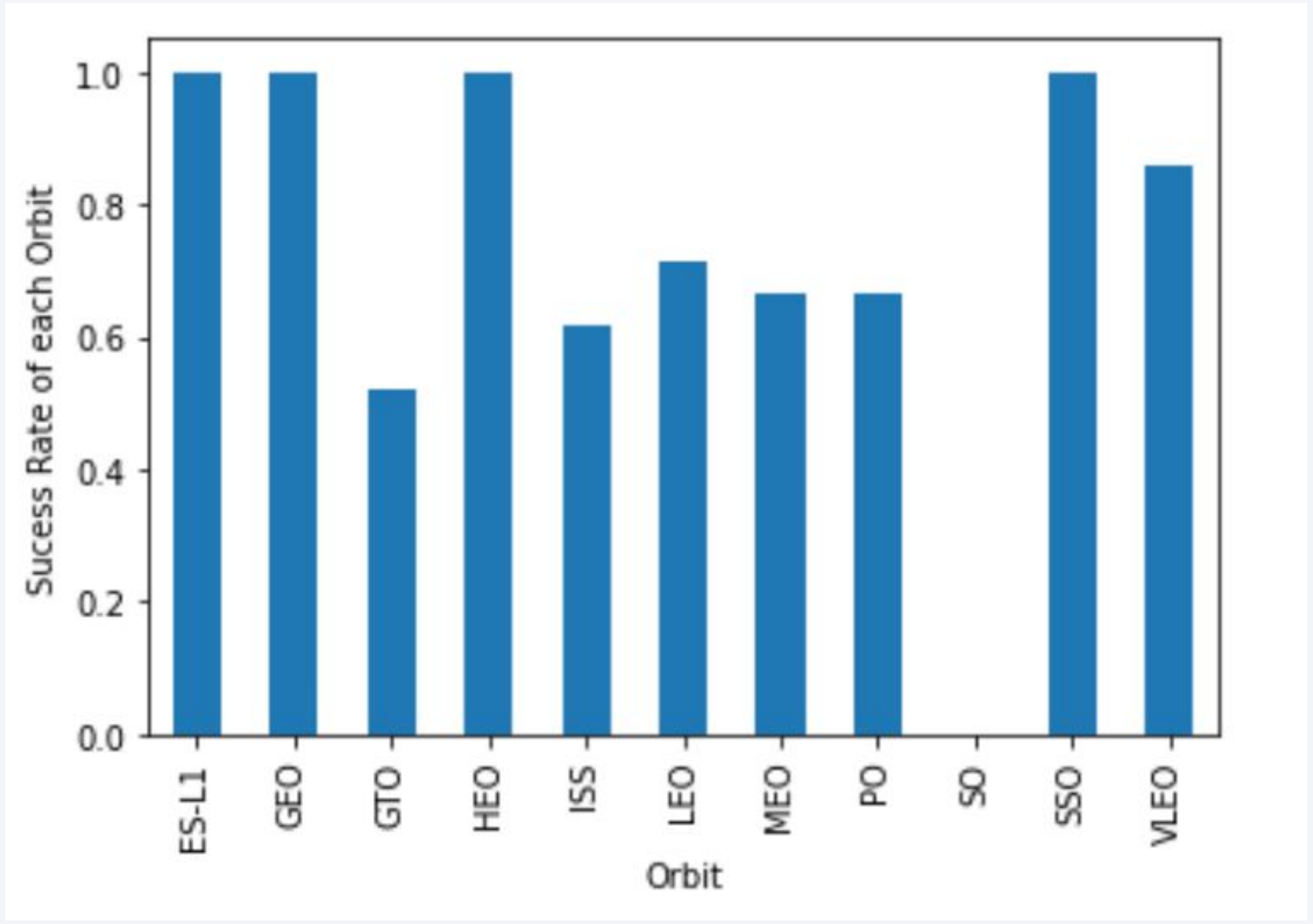
- Show the screenshot of the scatter plot with explanations



Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

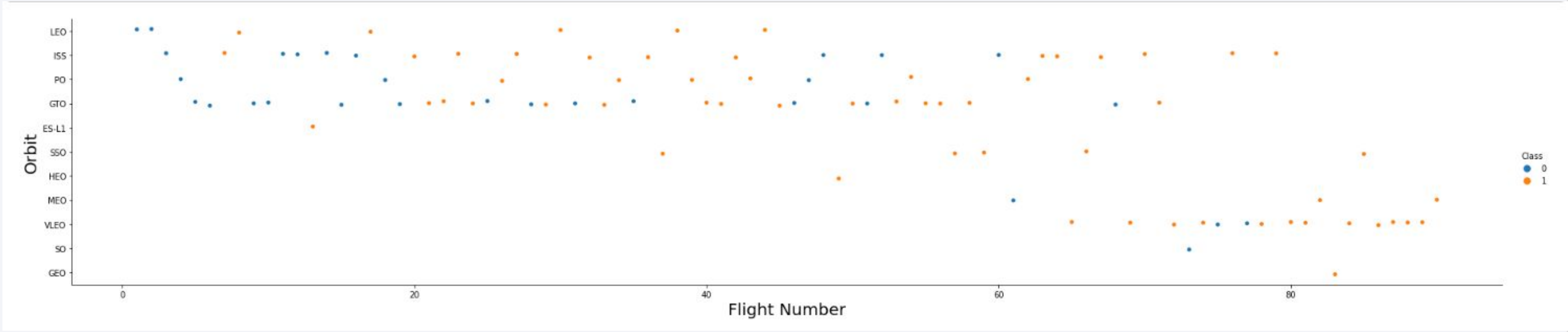
- Show the screenshot of the scatter plot with explanations



Flight Number vs. Orbit Type

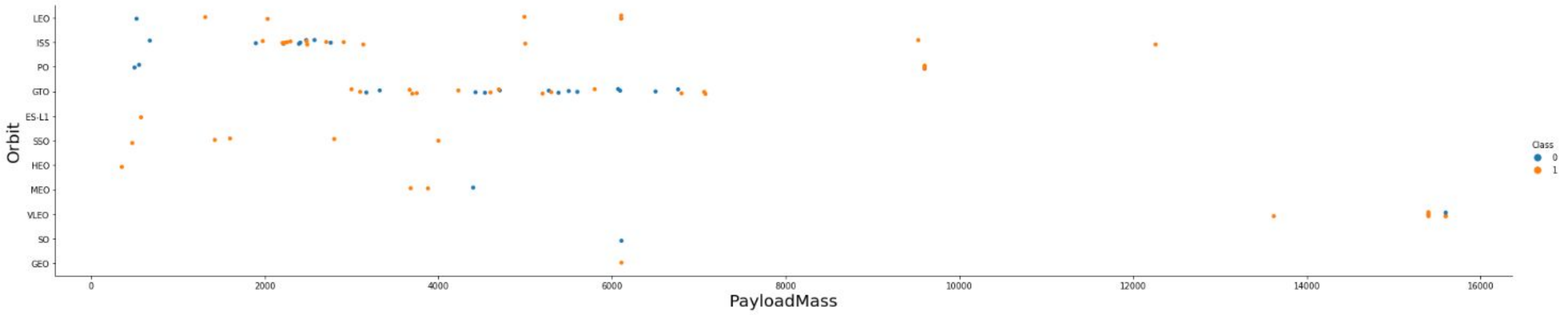
Show a scatter point of Flight number vs. Orbit type

Show the screenshot of the scatter plot with explanations



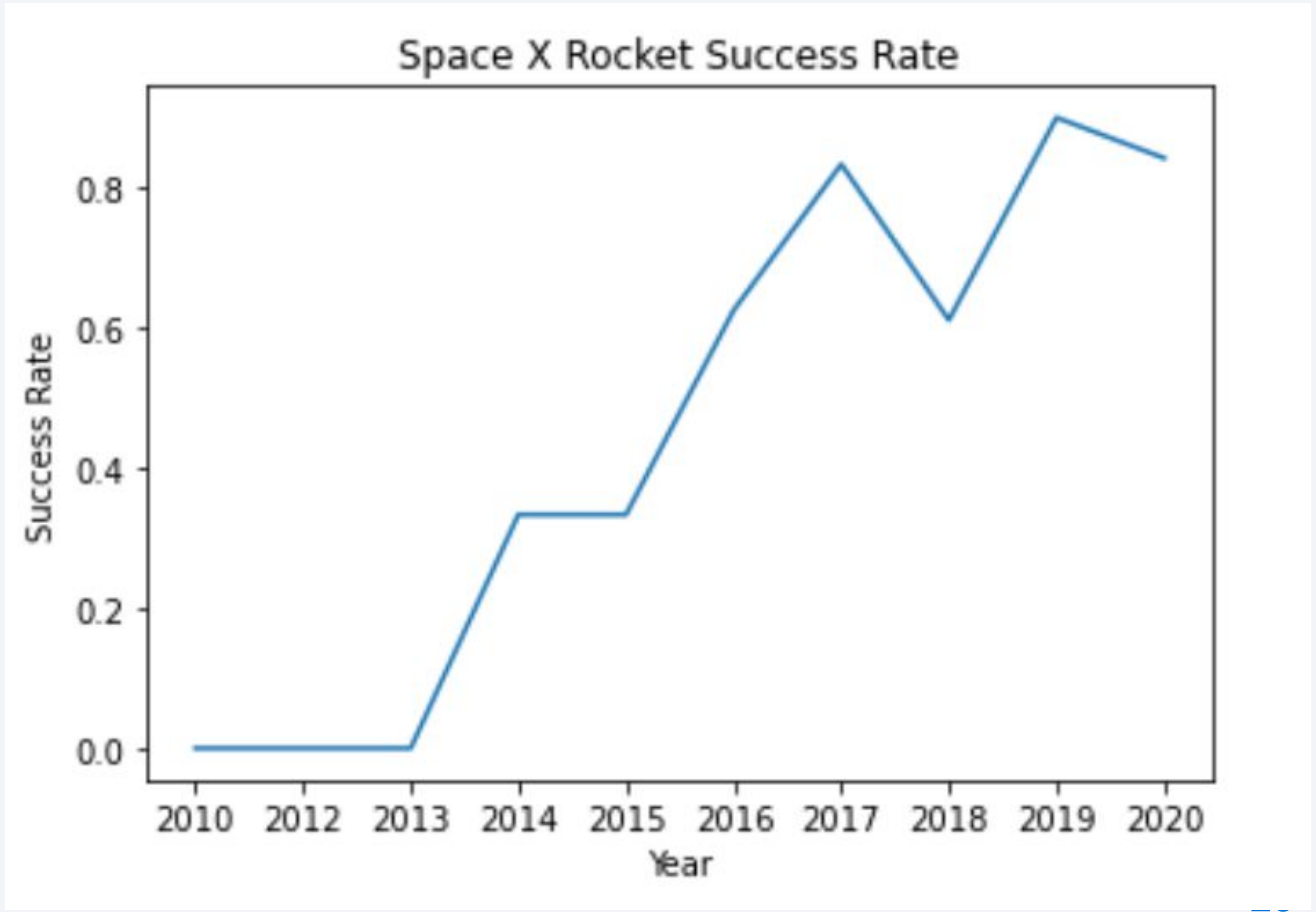
Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type
- Show the screenshot of the scatter plot with explanations



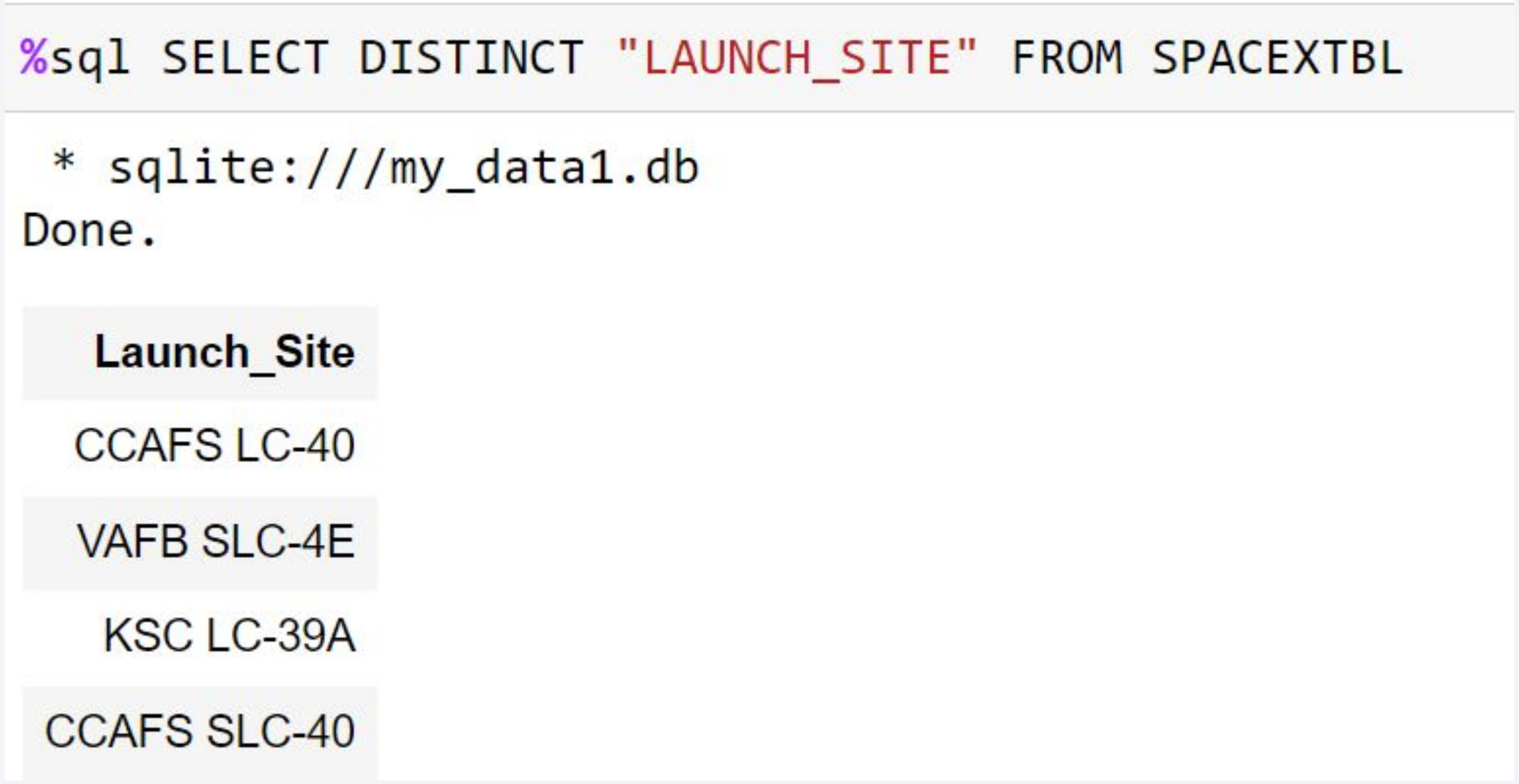
Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations



All Launch Site Names

- Find the names of the unique launch sites
- Explanation The use of DISTINCT in the query allows to remove duplicate LAUNCH_SITE.



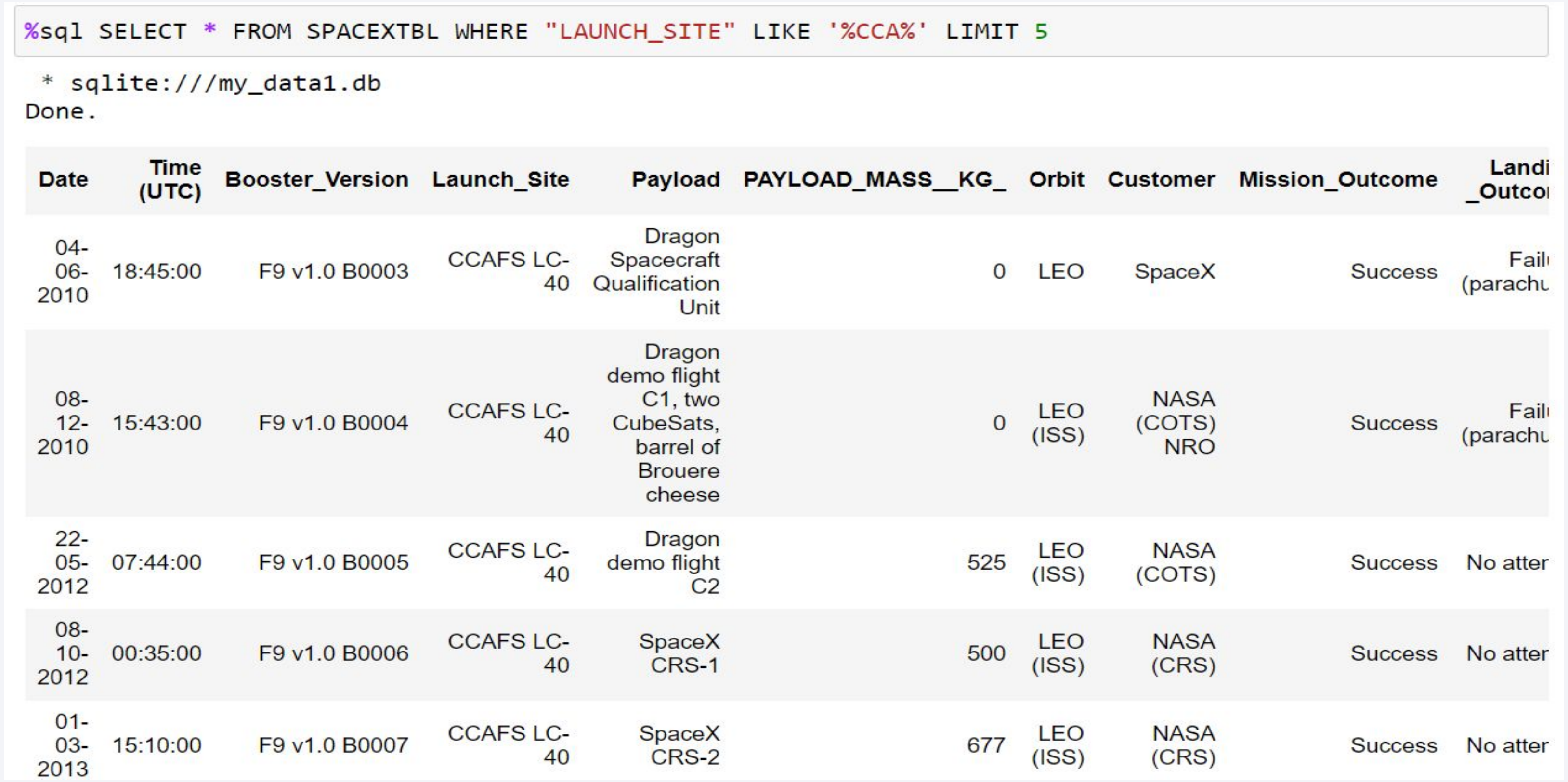
```
%sql SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTBL
```

```
 * 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'

- Find 5 records where launch sites 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.



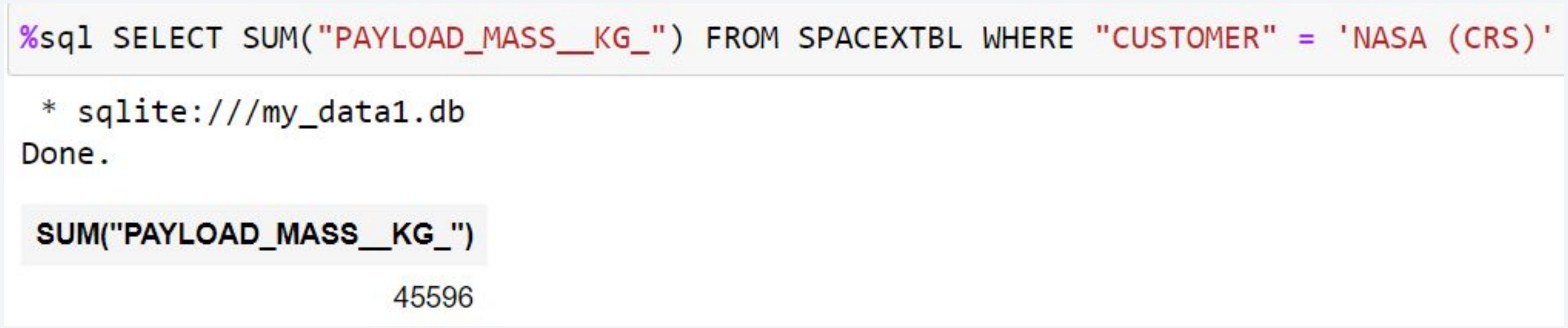
```
%sql SELECT * FROM SPACEXTBL WHERE "LAUNCH_SITE" LIKE '%CCA%' LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landi _Outcoı
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Faili (parachu
08-12-2010	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	Faili (parachu
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No atter
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No atter
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No atter

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- This query returns the sum of all payload masses where the customer is NASA (CRS)



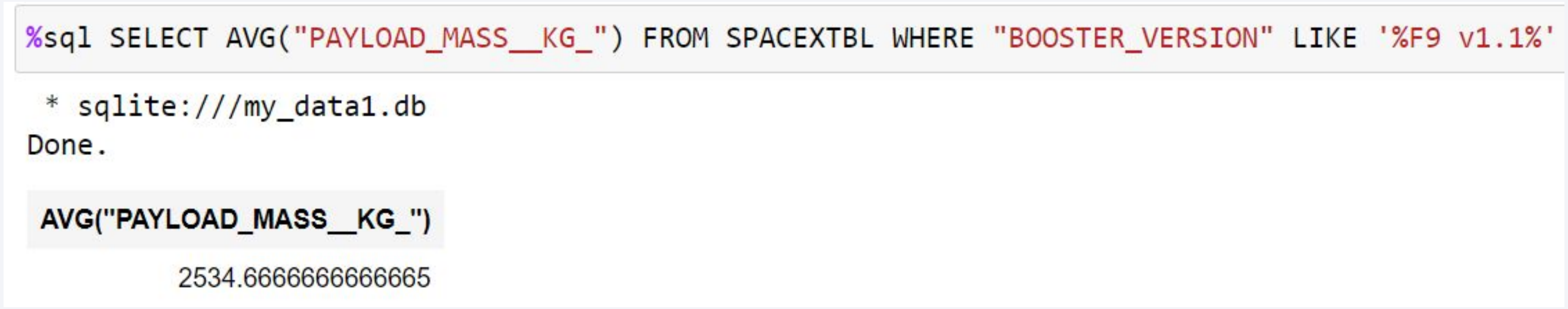
```
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "CUSTOMER" = 'NASA (CRS)'

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

SUM("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- This query returns the average of all payload masses where the booster version contains the substring F9 v1.1



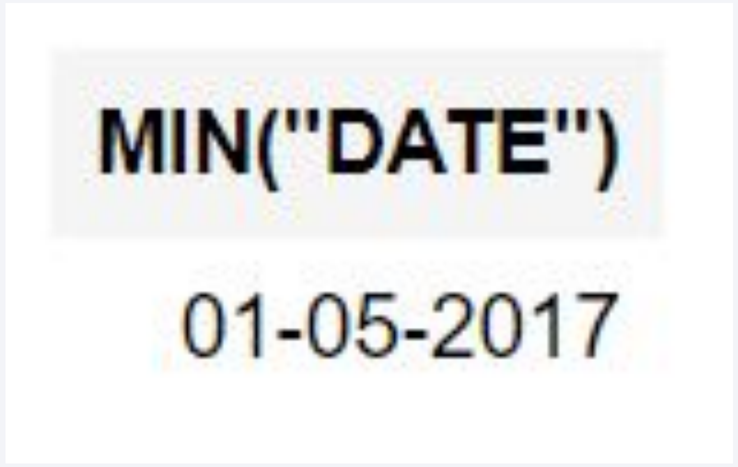
```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%'
```

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

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

First Successful Ground Landing Date

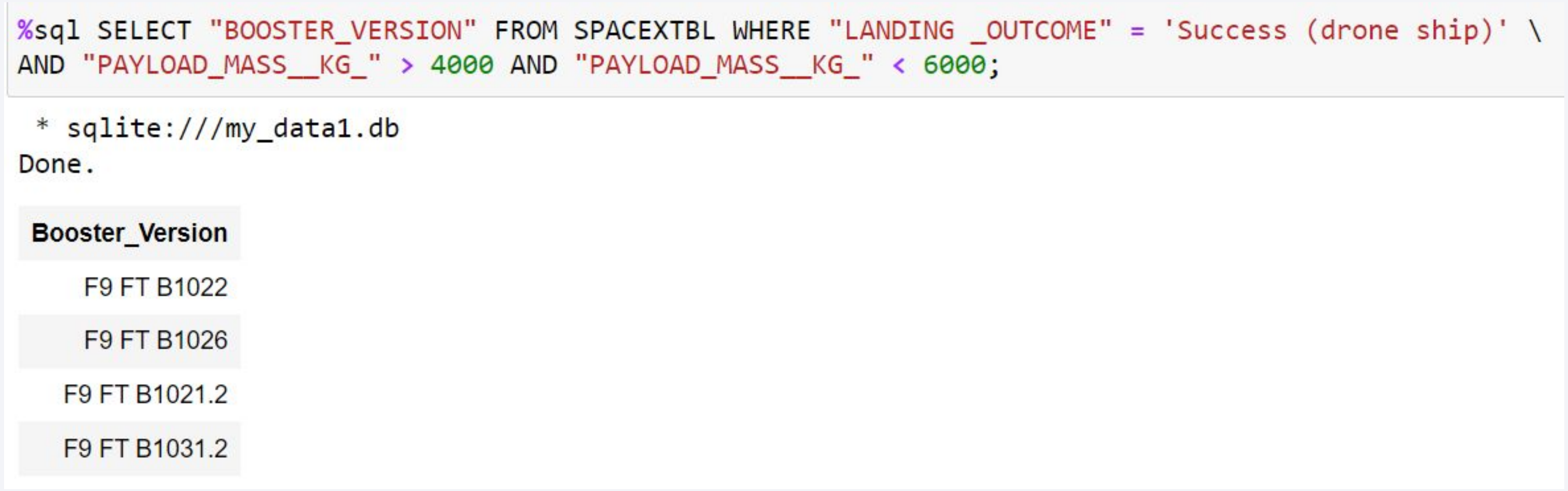
- Find the dates of the first successful landing outcome on ground pad
- With this query, we select the oldest successful landing. The WHERE clause filters dataset in order to keep only records where landing was successful. With the MIN function, we select the record with the oldest date.



```
MIN("DATE")
01-05-2017
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- 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.



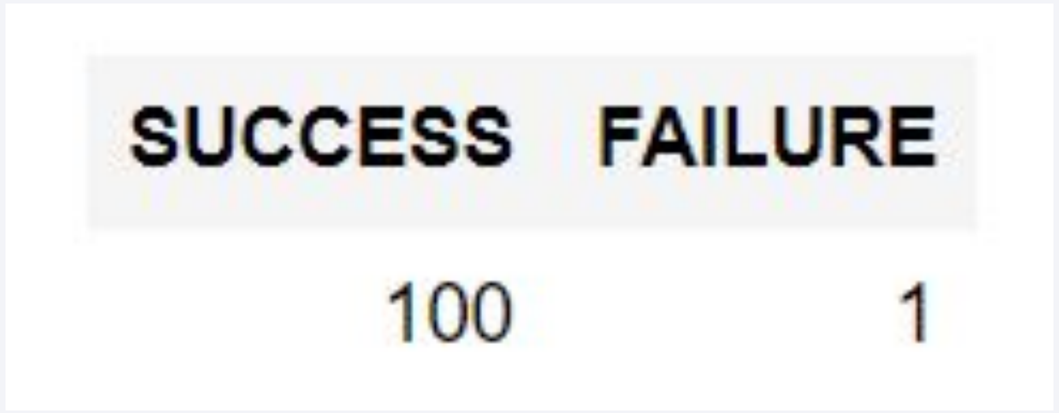
```
%sql SELECT "BOOSTER_VERSION" FROM SPACEXTBL WHERE "LANDING _OUTCOME" = 'Success (drone ship)' \
AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

```
 * 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

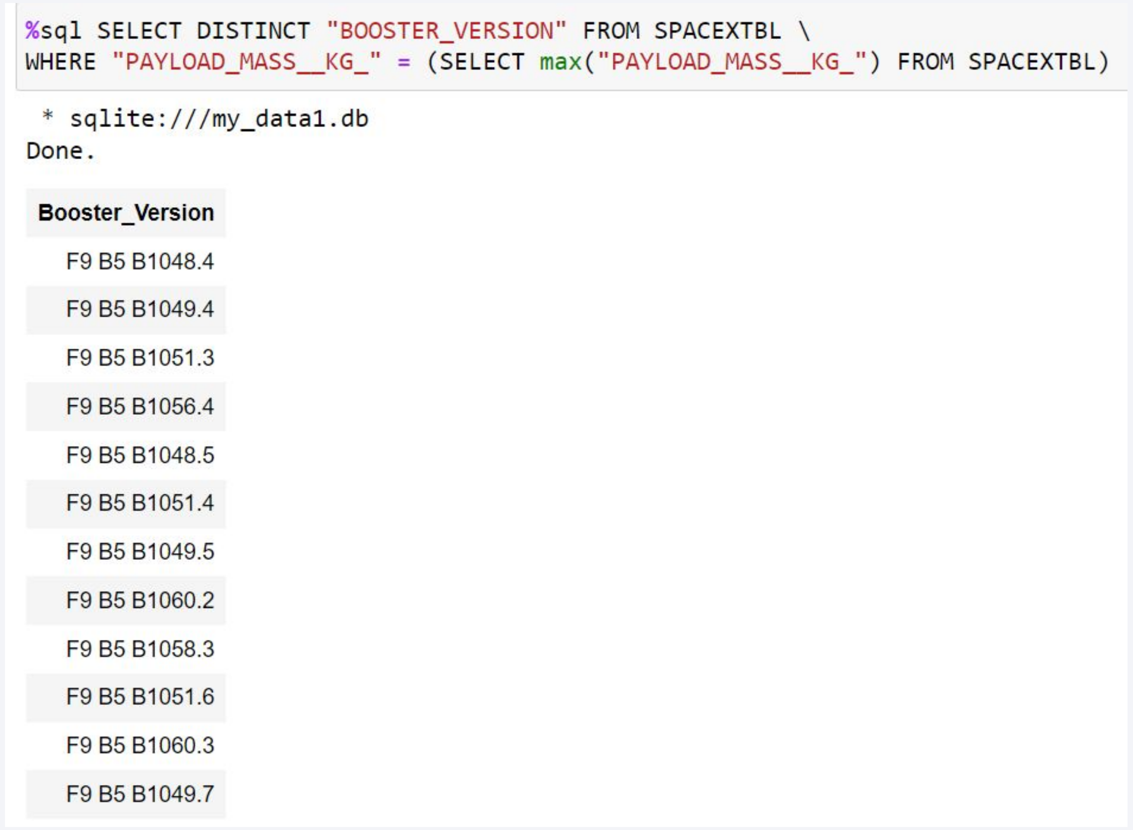
- Calculate the total number of successful and failure mission outcomes
- With the first SELECT, we show the subqueries that return results. The first subquery counts the successful mission. The second subquery counts the unsuccessful mission. The WHERE clause followed by LIKE clause filters mission outcome. The COUNT function counts records filtered.



SUCCESS	FAILURE
100	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- We 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.



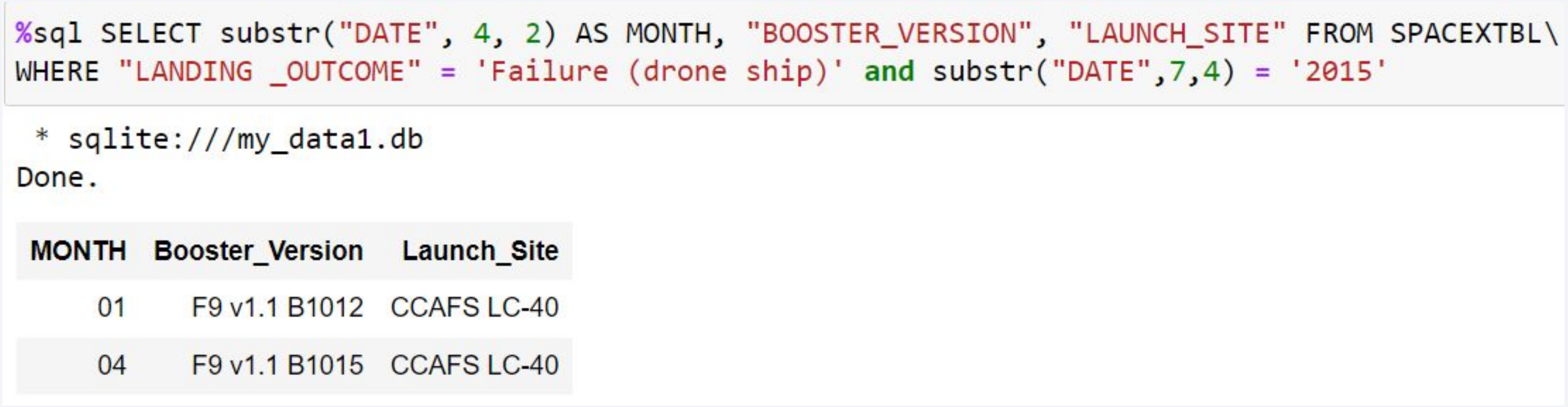
```
%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTBL \
WHERE "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

```
 * 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

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015.
- This query returns month, booster version, launch site where landing was unsuccessful and landing date took place in 2015. Substr function process



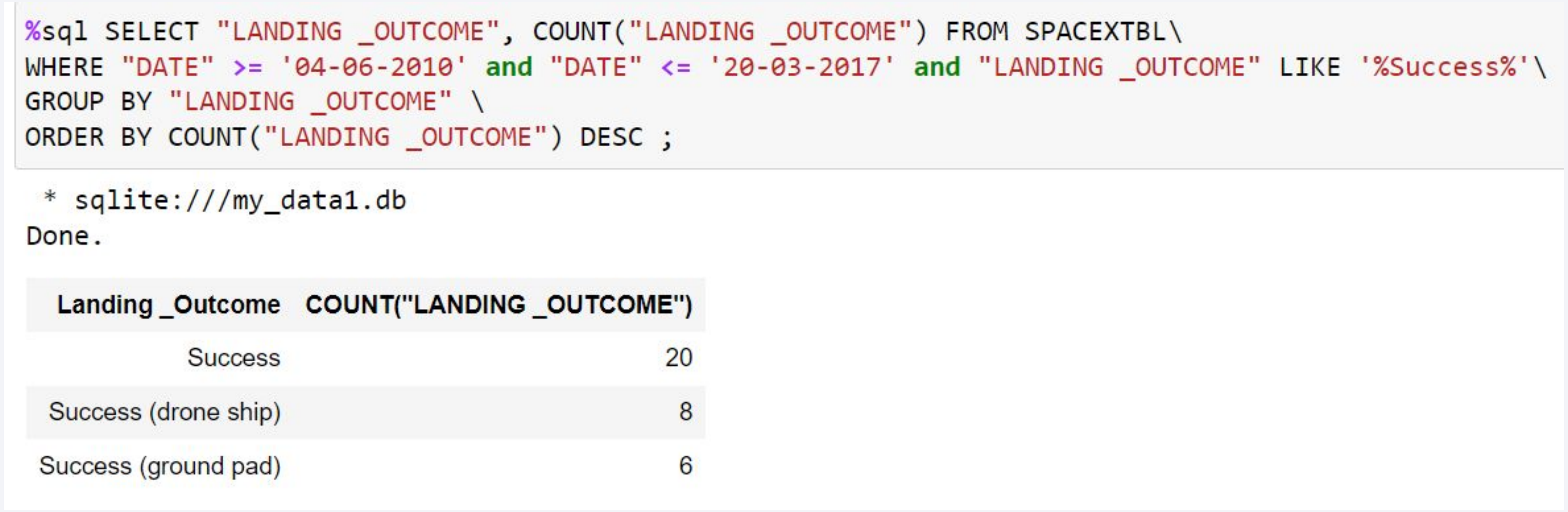
```
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL\
WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

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

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

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

- 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
- 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") FROM SPACEXTBL\
WHERE "DATE" >= '04-06-2010' and "DATE" <= '20-03-2017' and "LANDING _OUTCOME" LIKE '%Success%'\
GROUP BY "LANDING _OUTCOME" \
ORDER BY COUNT("LANDING _OUTCOME") DESC ;
```

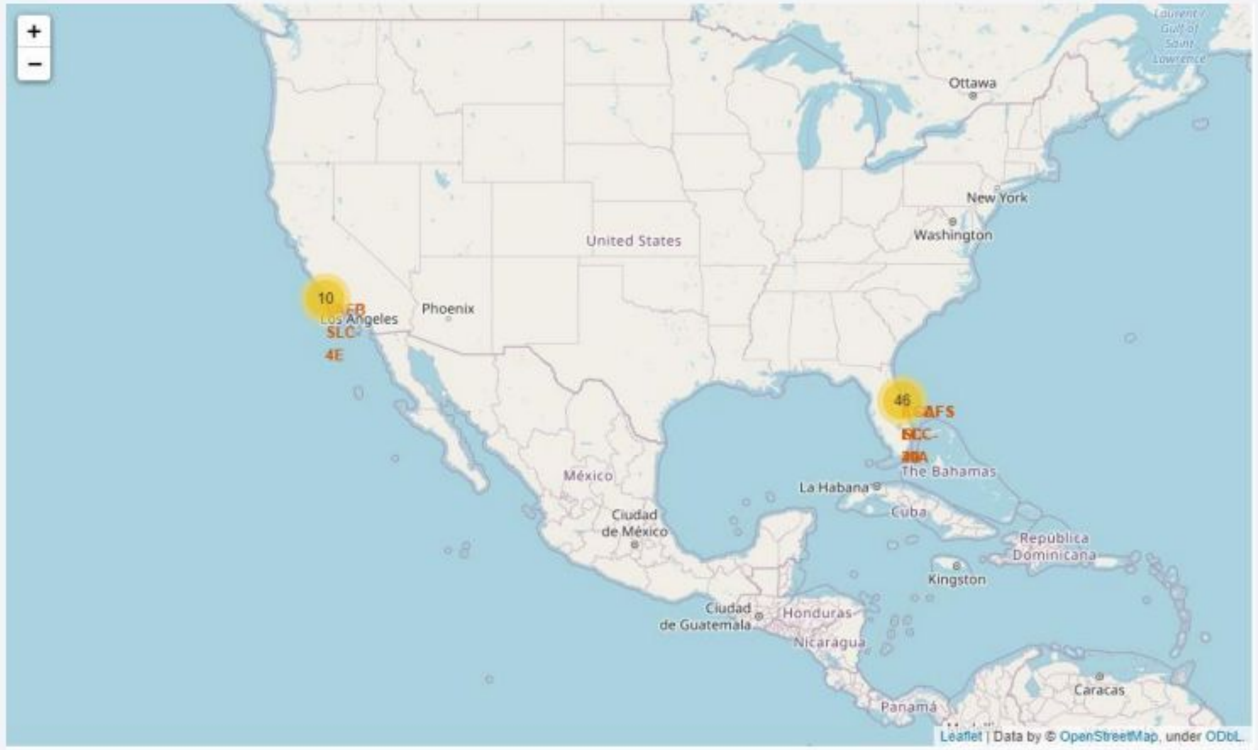
```
 * sqlite:///my_data1.db
Done.
```

Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

Section 3

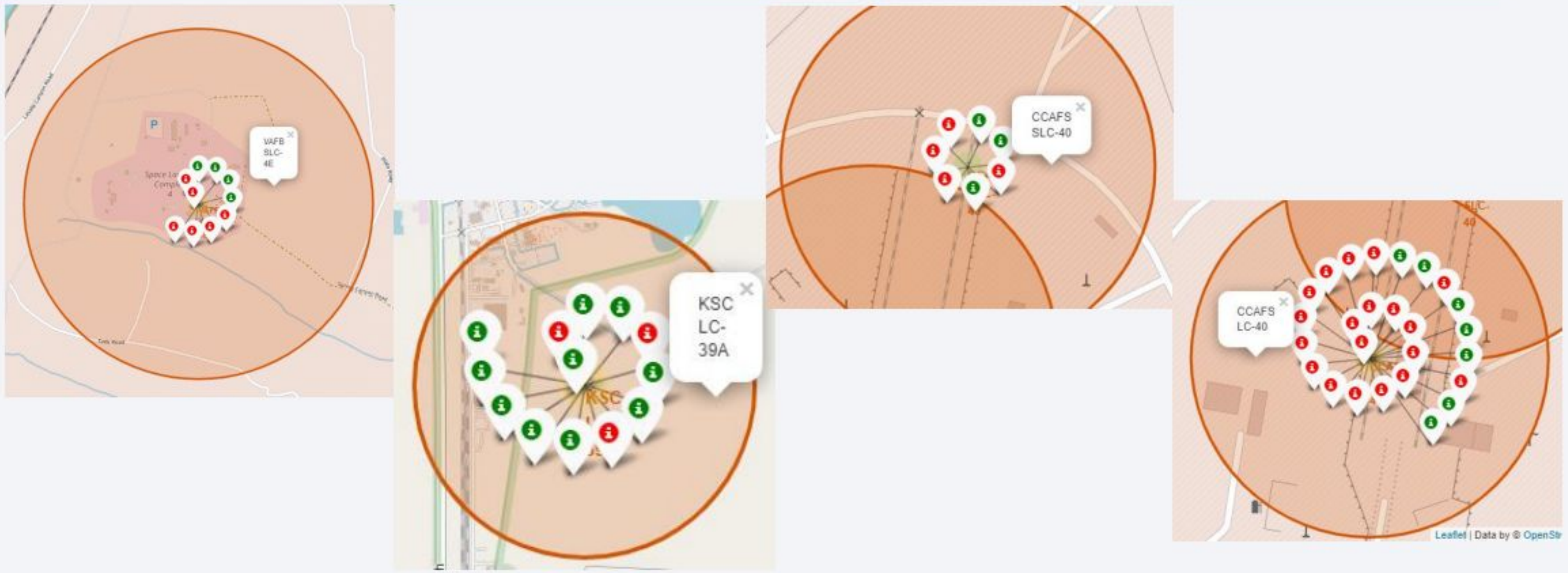
Launch Sites Proximities Analysis

Folium map – Ground stations



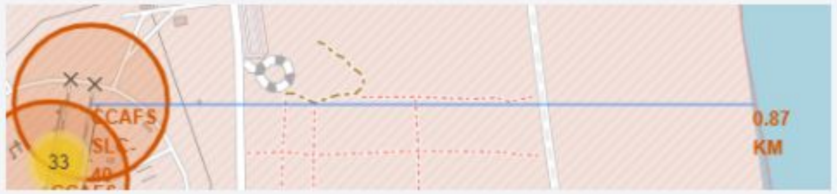
We see that SpaceX launch sites are located on the coast of the United States

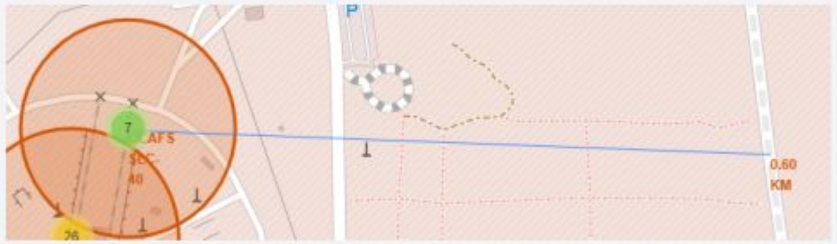
Folium Map 2 - Color Labeled Markers

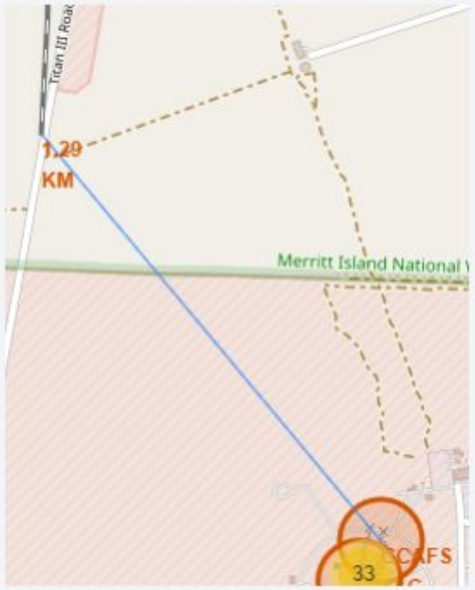


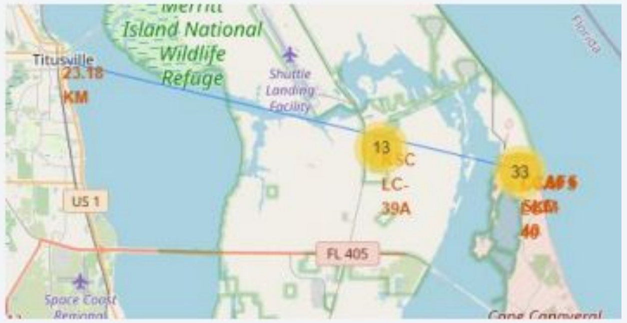
Green marker represents successful launches. Red marker represents unsuccessful launches. We note that KSC LC-39A has a higher launch success rate.

Folium Map 3 - Distances between CCAFS SLC-40 and its proximities







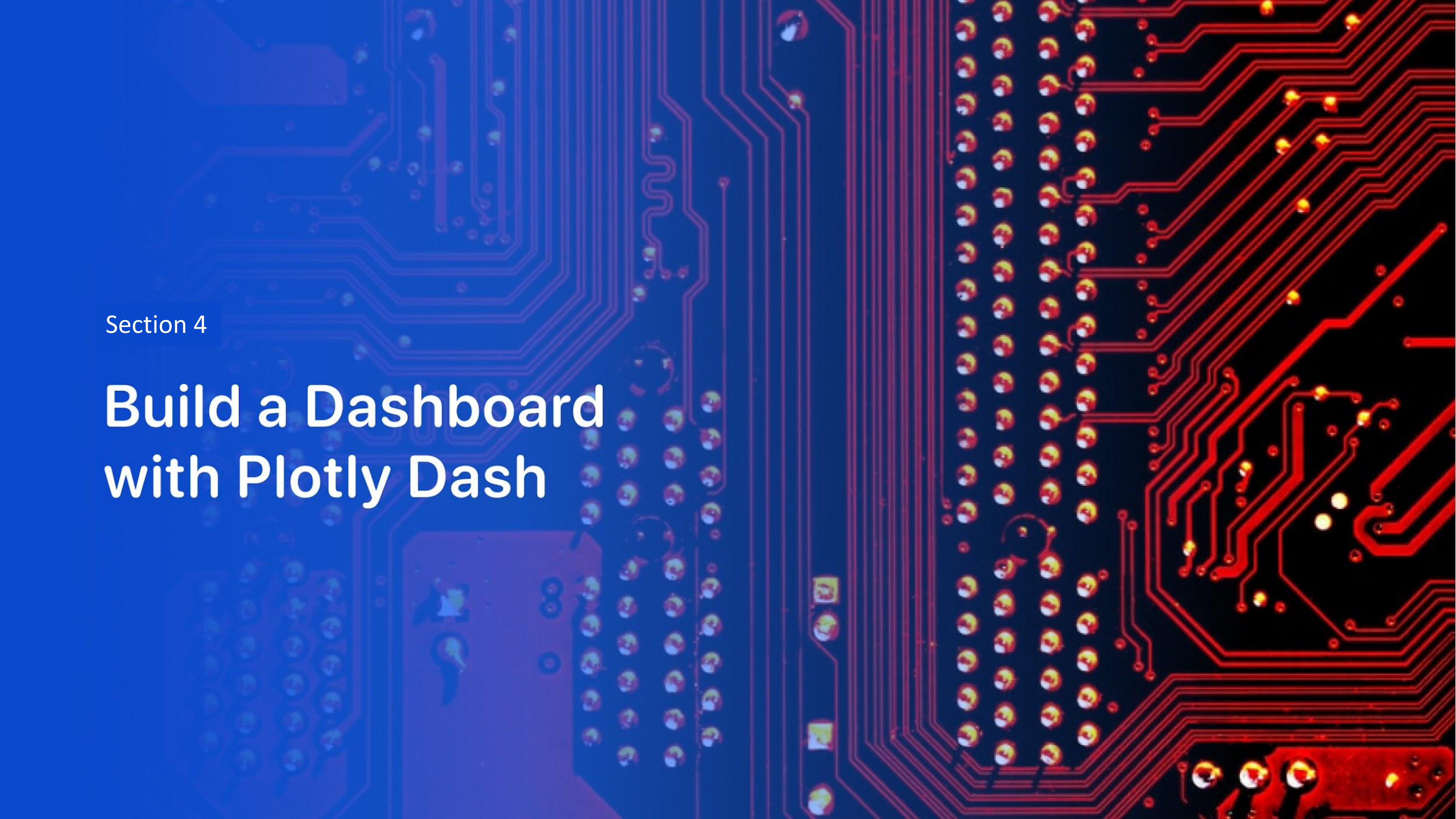


Is CCAFS SLC-40 in close proximity to railways ? Yes

Is CCAFS SLC-40 in close proximity to highways ? Yes

Is CCAFS SLC-40 in close proximity to coastline ? Yes

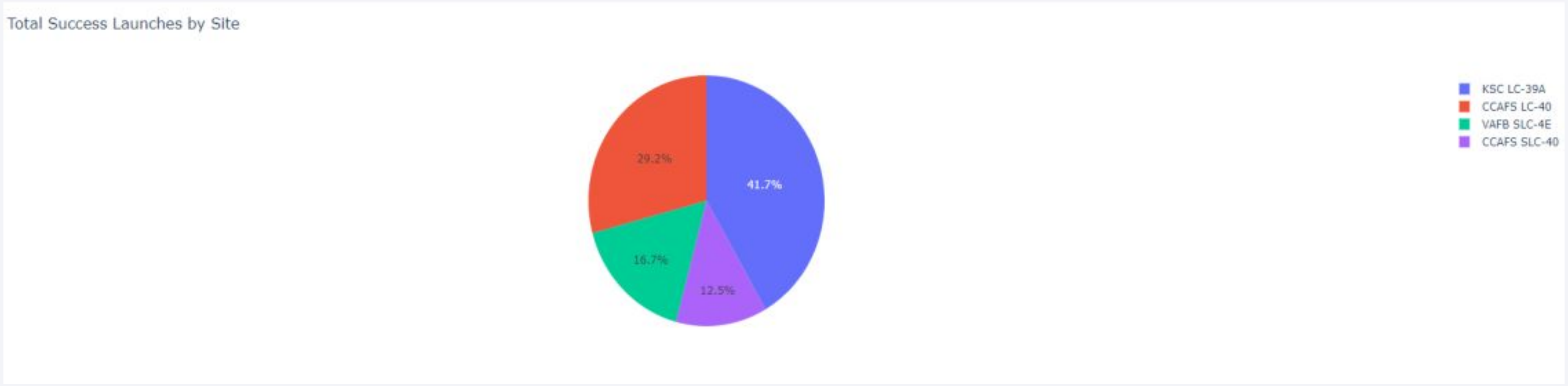
Do CCAFS SLC-40 keeps certain distance away from cities ? No



Section 4

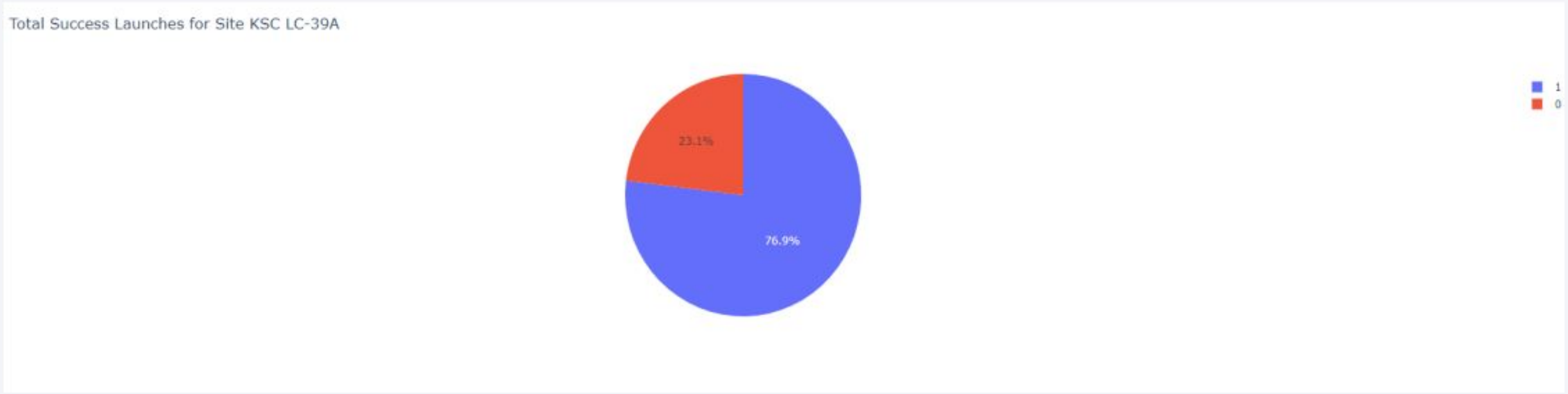
Build a Dashboard with Plotly Dash

Dashboard - Total success by Site



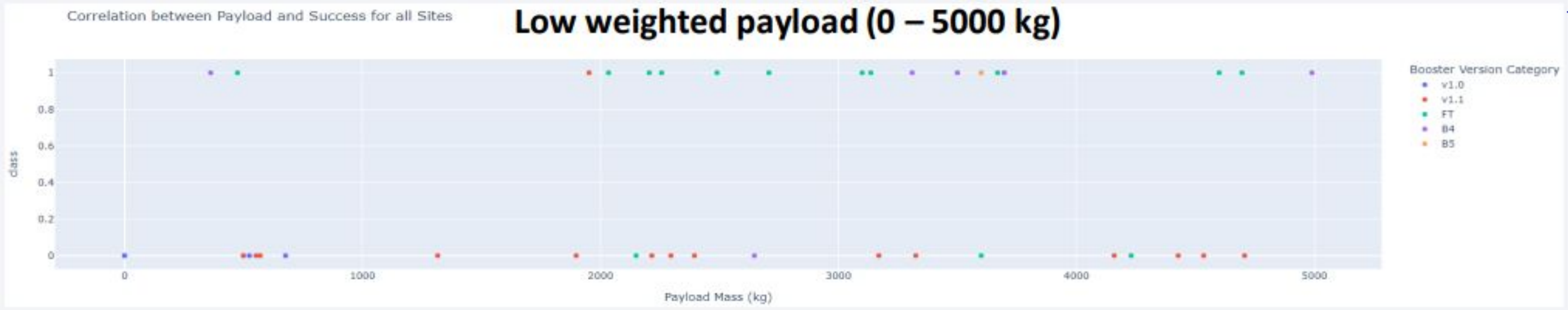
We see that KSC LC-39A has the best success rate of launches.

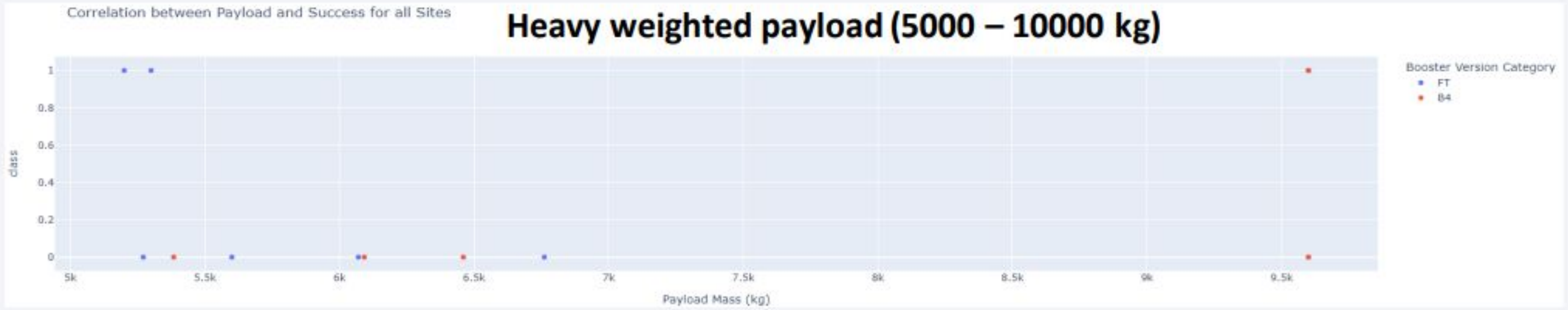
Dashboard 2 - Total success launches for Site KSC LC-39A



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

Dashboard 3 - Payload mass vs Outcome for all sites with different payload mass selected



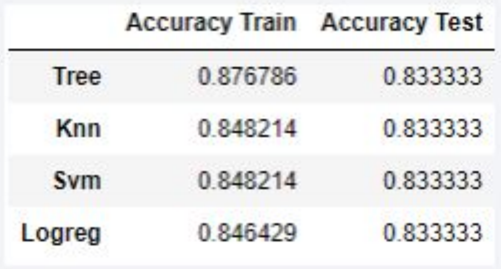


Low weighted payloads have a better success rate than the heavy weighted payloads.

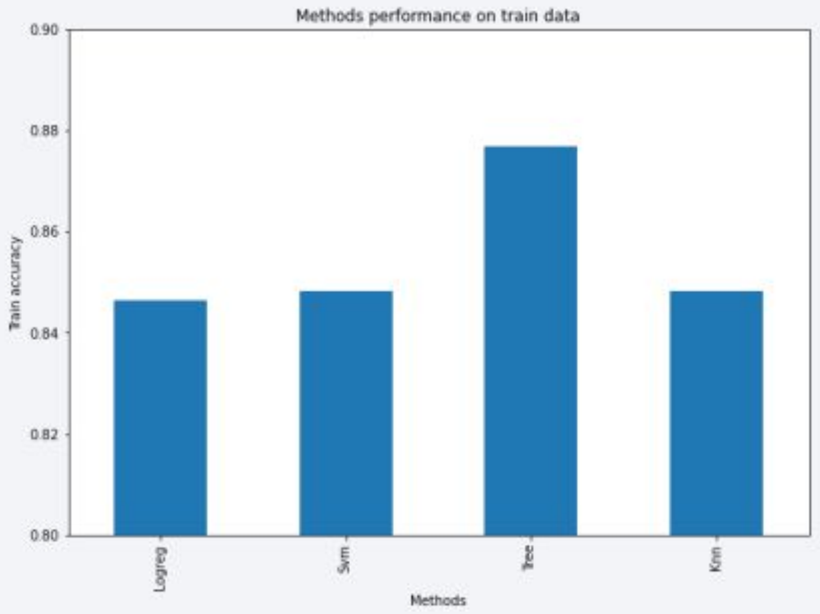
Section 5

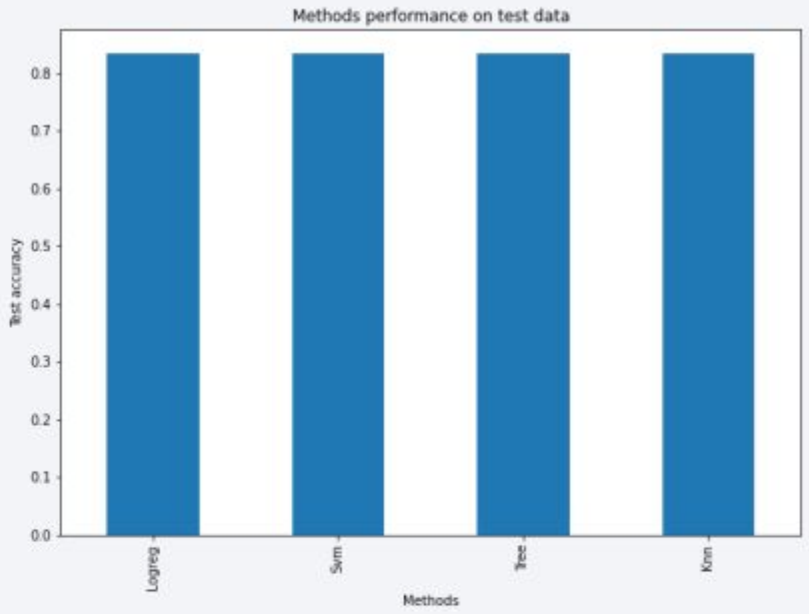
Predictive Analysis (Classification)

Classification Accuracy

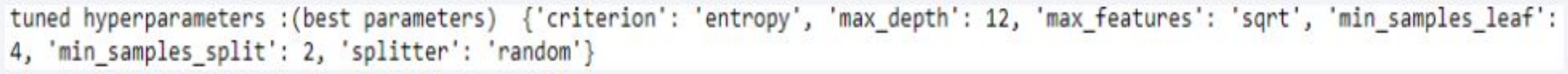


	Accuracy Train	Accuracy Test
Tree	0.876786	0.833333
Knn	0.848214	0.833333
Svm	0.848214	0.833333
Logreg	0.846429	0.833333





Decision tree best parameters



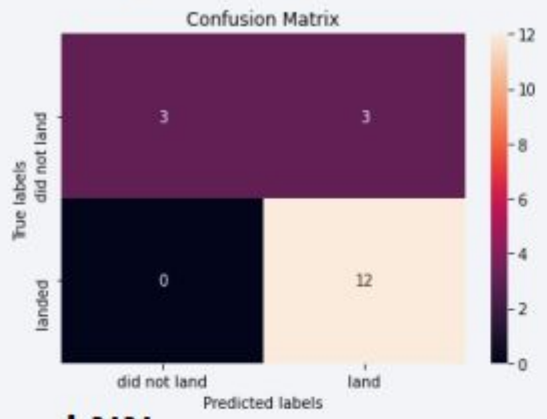
```
tuned hyperparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf':
4, 'min_samples_split': 2, 'splitter': 'random'}
```

For accuracy test, all methods performed similar. We could get more test data to decide between them. But if we really need to choose one right now, we would take the decision tree.

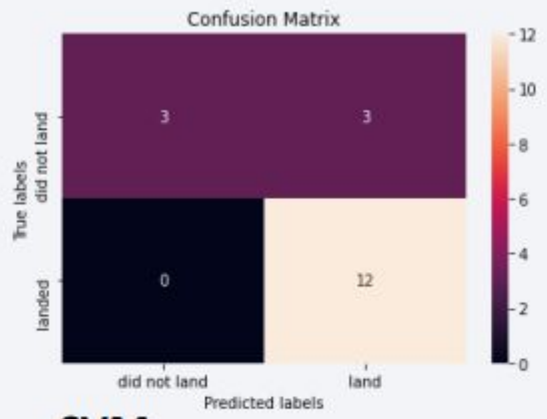
Confusion Matrix

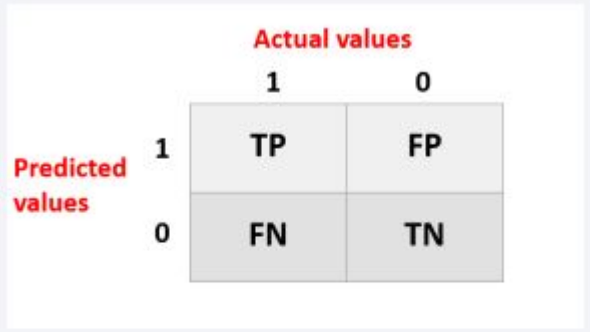
As the test accuracy are all equal, the confusion matrices are also identical. The main problem of these models are false positives.

Logistic regression



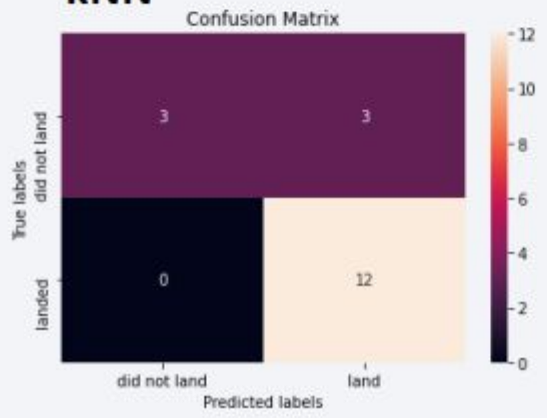
Decision Tree



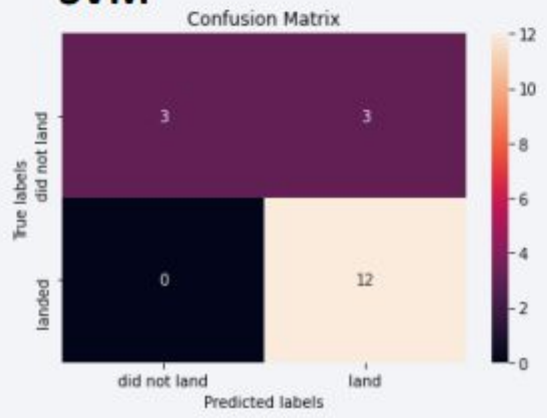


		Actual values	
		1	0
Predicted values	1	TP	FP
	0	FN	TN

kNN



SVM



Conclusions

- The success of a mission can be explained by several factors such as the launch site, the orbit and especially the number of previous launches. Indeed, we can assume that there has been a gain in knowledge between launches that allowed to go from a launch failure to a success.

• 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. But generally low weighted payloads perform better than the heavy weighted payloads.

• With the current data, we cannot explain why some launch sites are better than others (KSC LC-39A is the best launch site). To get an answer to this problem, we could obtain atmospheric or other relevant data.

• For this dataset, we choose the Decision Tree Algorithm as the best model even if the test accuracy between all the models used is identical. We choose Decision Tree Algorithm because it has a better train accuracy.

Thank you!