IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

GABRIELA BERRIOS
MAY 31 2022

Outline

Executive Summary

- Summary of methodologies
  - DATA COLLECTION USING API
  - DATA COLLECTION WITH WEBSCRAPPING
  - DATA WRANGLING
  - DATA ANALYSIS WITH SQL
  - DATA ANALYSIS WITH DATA VISUALIZATION USING DASH
  - VISUAL ANALYTICS USING FOLIUM
  - MACHINE LEARNING PREDICTION

- Summary of the results
  - BEST MODEL PREDICTION AND PREDICTIVE RESULT
  - MAPS , CHARTS COMPARISON,
  - INTERACTIVE VISUALIZATION

Introduction

- Project background and context:
  - **Falcon 9** is a two stage to orbit (TSTO) medium lift launch vehicle (MLV) that is designed and manufactured by SpaceX. Unlike most rockets in service, which are expendable launch systems, Falcon 9 is partially reusable, with the first stage capable of reentering the atmosphere and landing vertically after separating from the second stage. This feat was achieved for the first time on flight 20 in December 2015. The goal of this project is to create a machine learning pipeline to predict if the first state will land successfully.
- Problems you want to find answers
  - What factors can affect to make a launch successful such as weight, location, orbit. Etc.

Section 1

Methodology

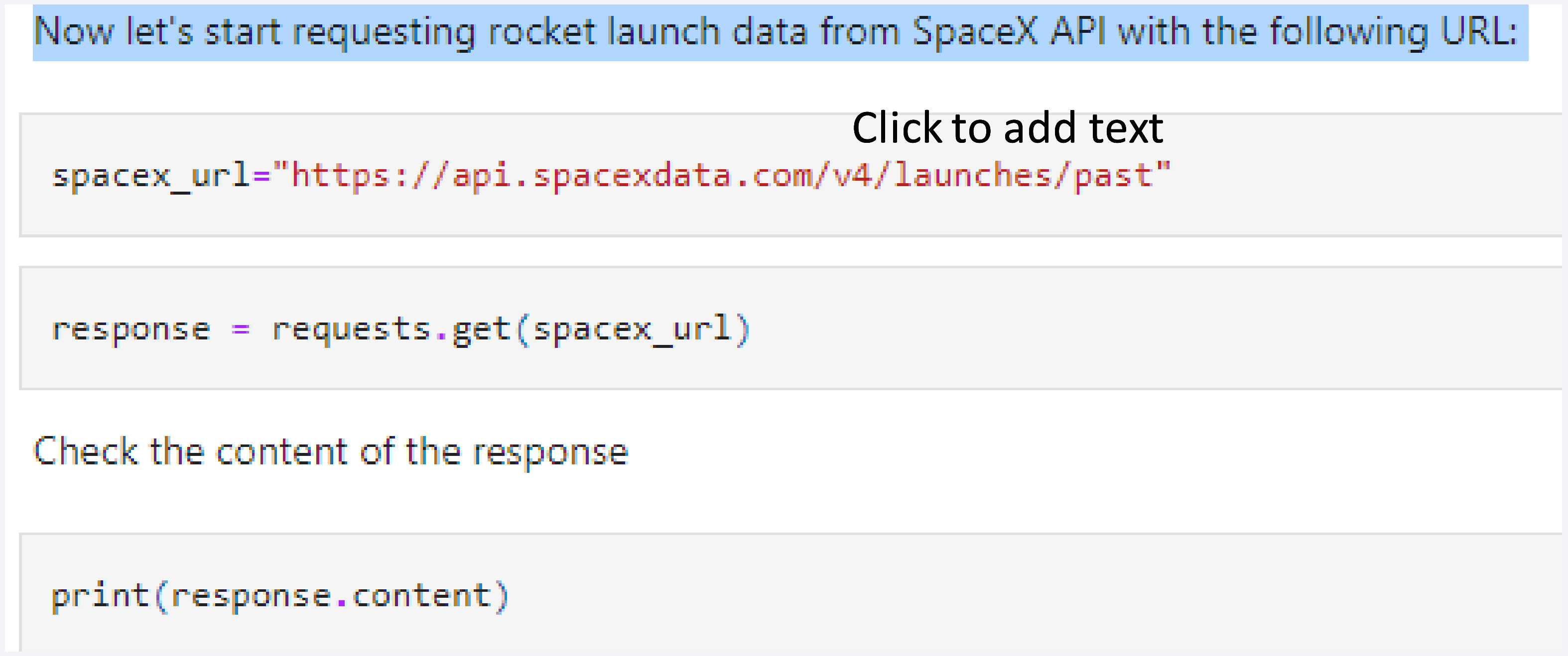
Methodology

Executive Summary

- Data collection methodology:
  - SPACEX API AND WEBSCRAPPING USING WIKIPEDIA
- Perform data wrangling
  - Describe how data was processed
- 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

- The data was collected using Spacex API
- You need to present your data collection process use key phrases and flowcharts


Now let's start requesting rocket launch data from SpaceX API with the following URL:

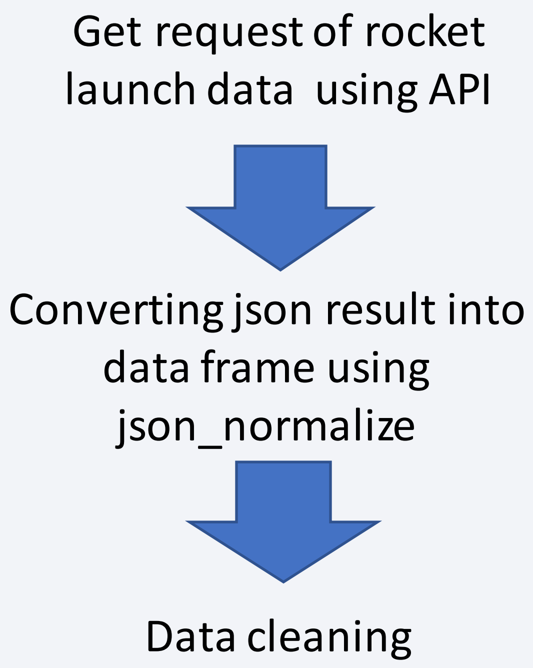
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

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

Check the content of the response

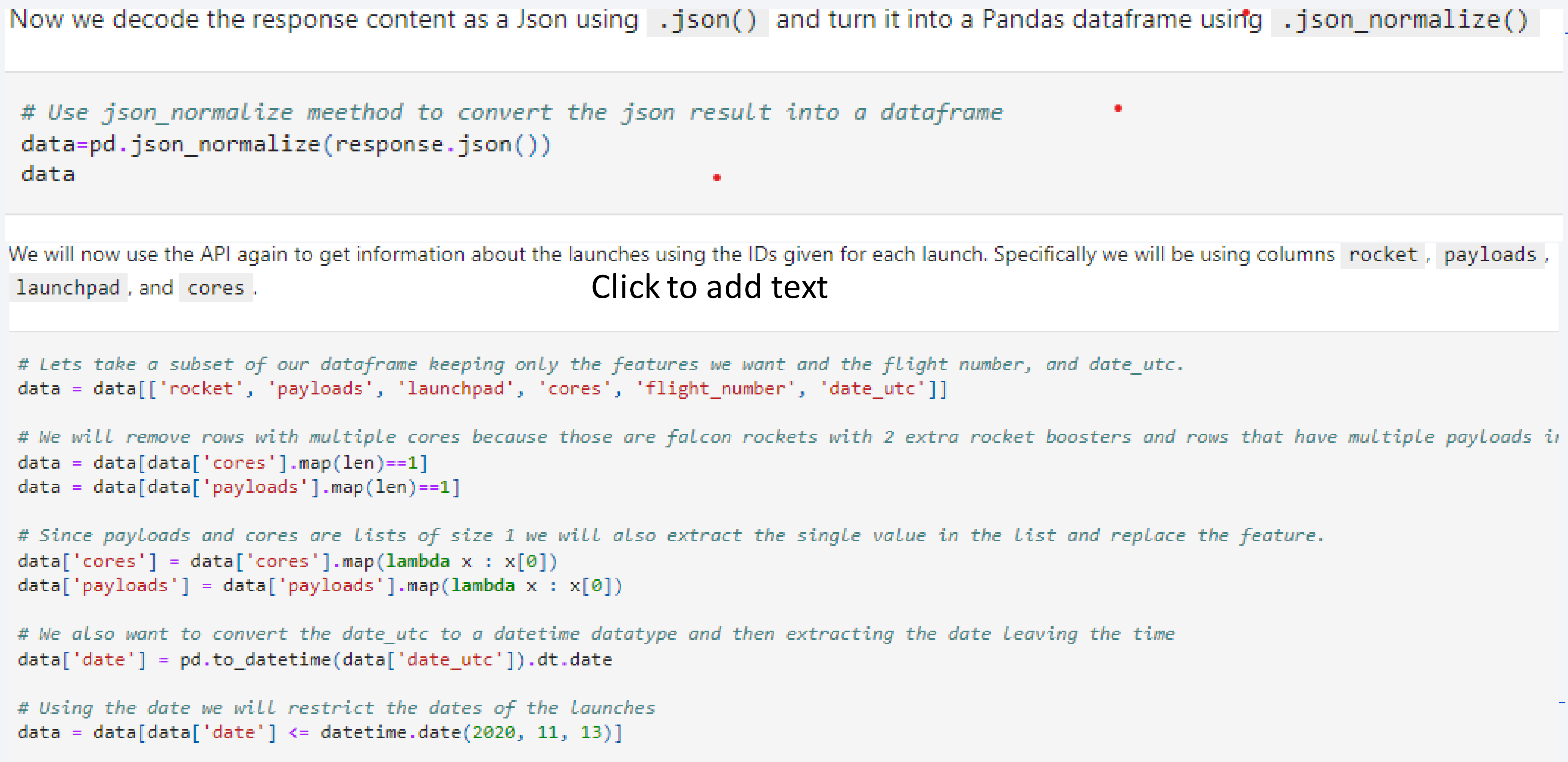
```
print(response.content)
```

Click to add text


Get request of rocket launch data using API
Converting json result into data frame using json_normalize
Data cleaning

https://github.com/gabrielaberrios1/projectos/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection – SpaceX API



Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

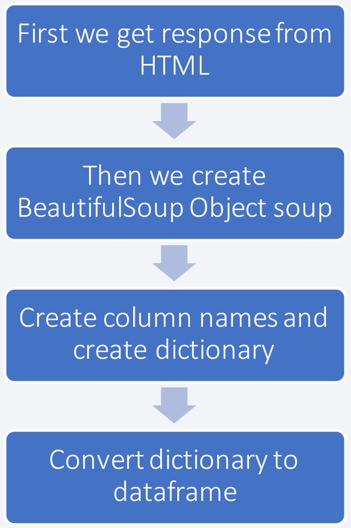
```
# Use json_normalize meethod to convert the json result into a dataframe
data=pd.json_normalize(response.json())
data
```

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

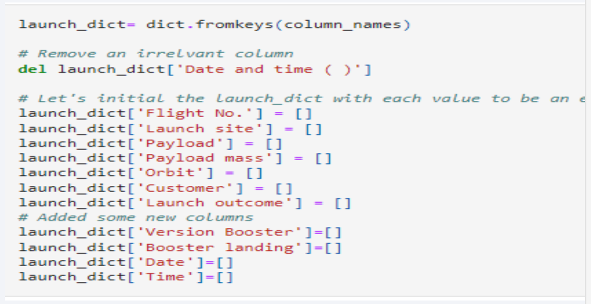
Click to add text

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads i
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

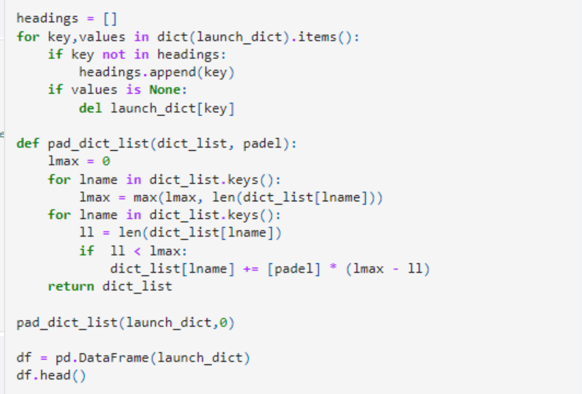
Data Collection - Scraping


First we get response from HTML
Then we create BeautifulSoup Object soup
Create column names and create dictionary
Convert dictionary to dataframe

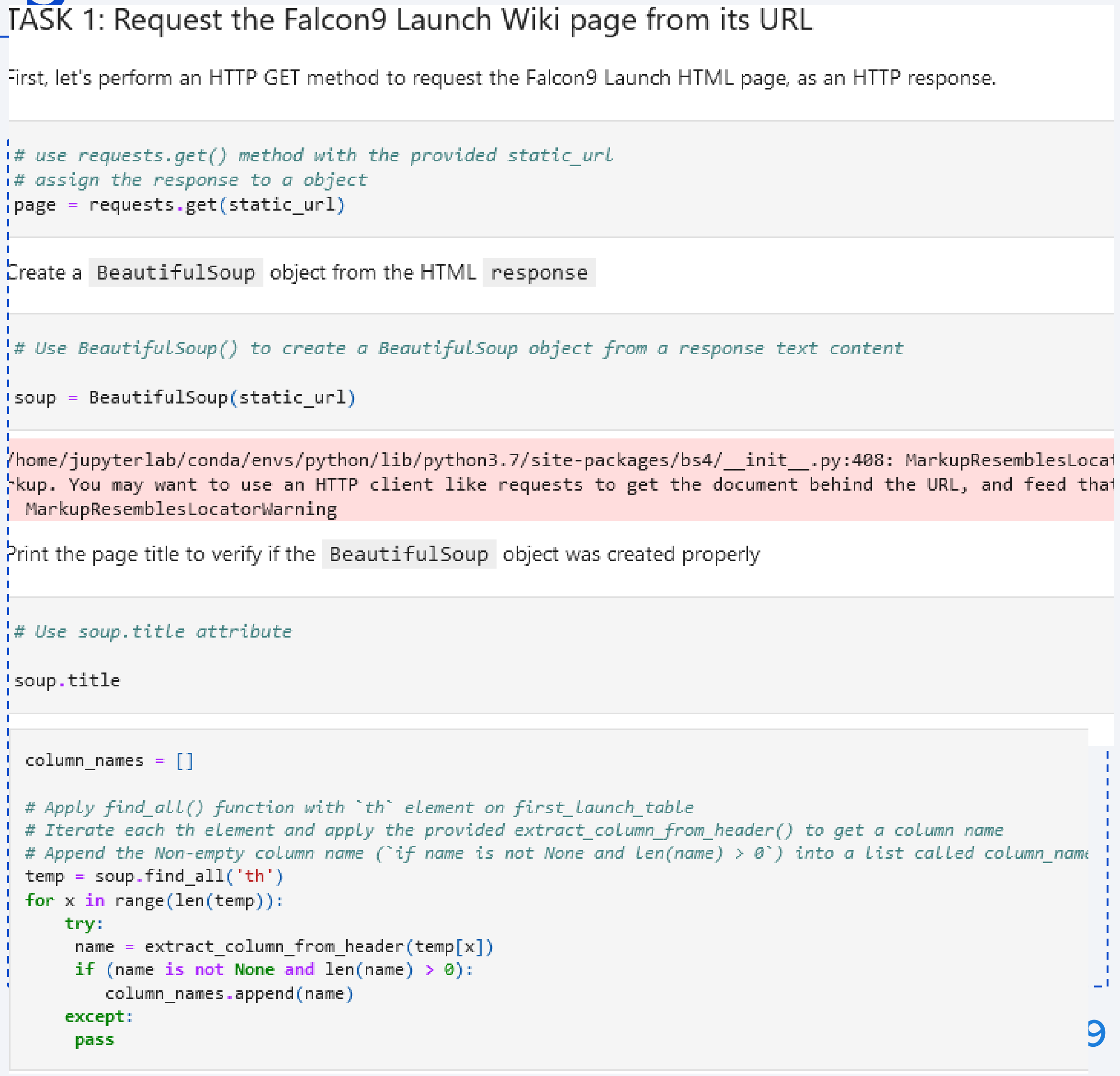
- https://github.com/gabrielaberrios1/projectos/blob/main/jupyter-labs-webscraping%20(1).ipynb



```
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 e
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']=[]
```



```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

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



TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
page = requests.get(static_url)
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

soup = BeautifulSoup(static_url)
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/bs4/__init__.py:408: MarkupResemblesLocat
rkup. You may want to use an HTTP client like requests to get the document behind the URL, and feed that
  MarkupResemblesLocatorWarning
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute

soup.title
```

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_name
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

Data Wrangling

In this method we need to perform exploratory data analysis EDA on the dataset and convert the successful and unsuccessful outcomes into training label 1 and 0 respectively.

Calculate the number of launches on each site

↓

Calculate the number of occurence

↓

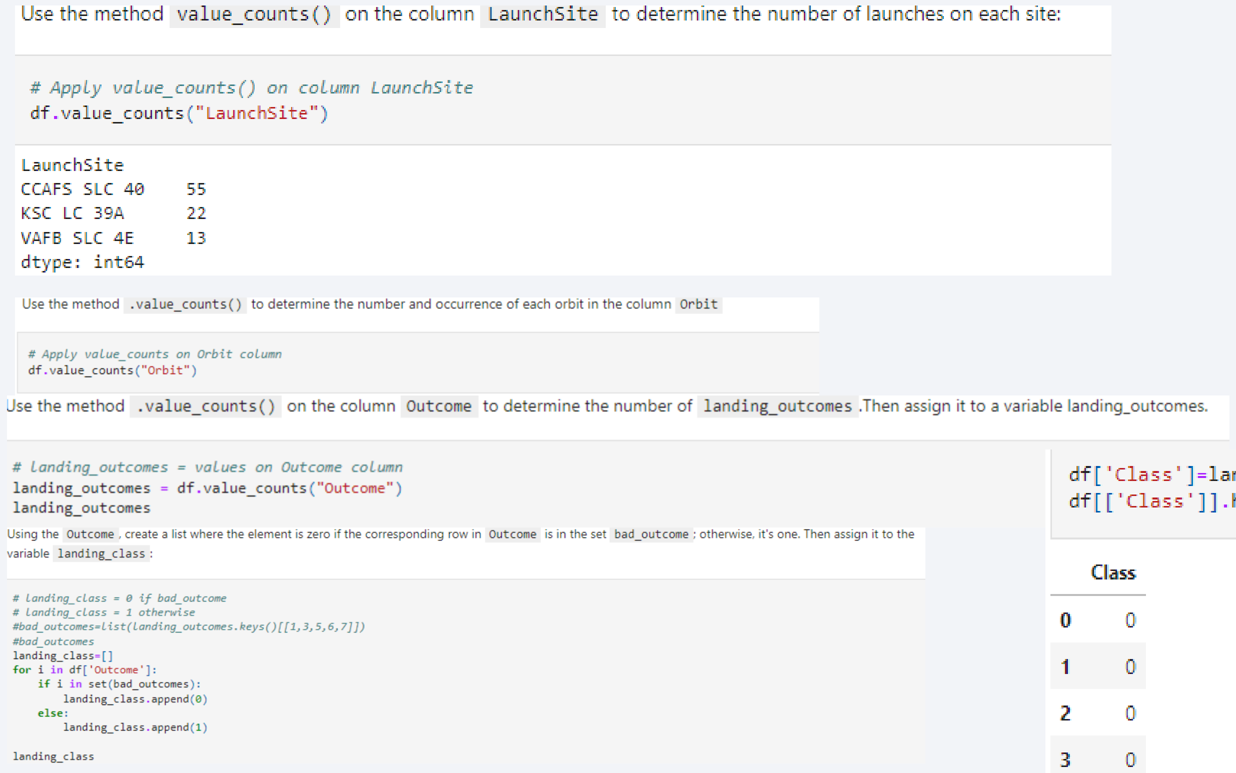
Calculate # and occurrence of mission outcome per orbit type

↓

Create landing outcome label from outcome column

↓

Convert outcome launch to class values 0,1.



Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

```
# Apply value_counts() on column LaunchSite
df.value_counts("LaunchSite")
```

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

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

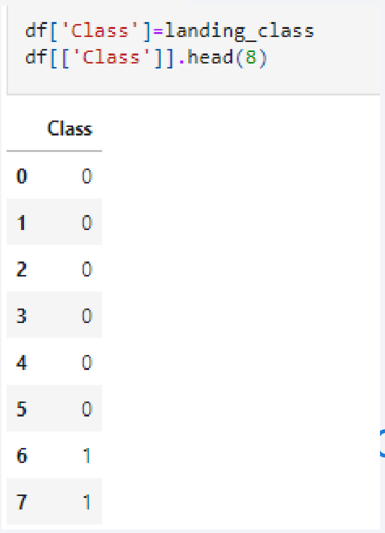
```
# Apply value_counts on Orbit column
df.value_counts("Orbit")
```

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

```
# landing_outcomes = values on Outcome column
landing_outcomes = df.value_counts("Outcome")
landing_outcomes
```

Using the `Outcome` , create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome` ; otherwise, it's one. Then assign it to the variable `landing_class` :

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
#bad_outcomes=list(landing_outcomes.keys()[[1,3,5,6,7]])
#bad_outcomes
landing_class=[]
for i in df['Outcome']:
    if i in set(bad_outcomes):
        landing_class.append(0)
    else:
        landing_class.append(1)

landing_class
```

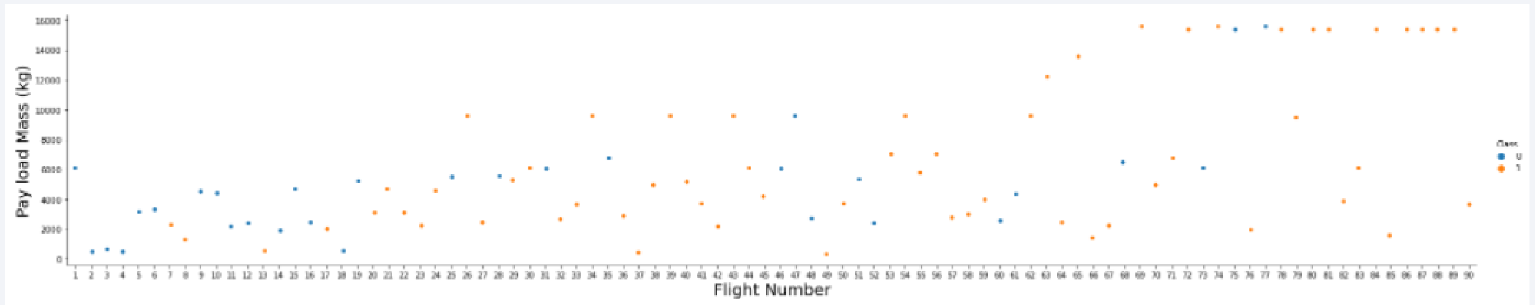


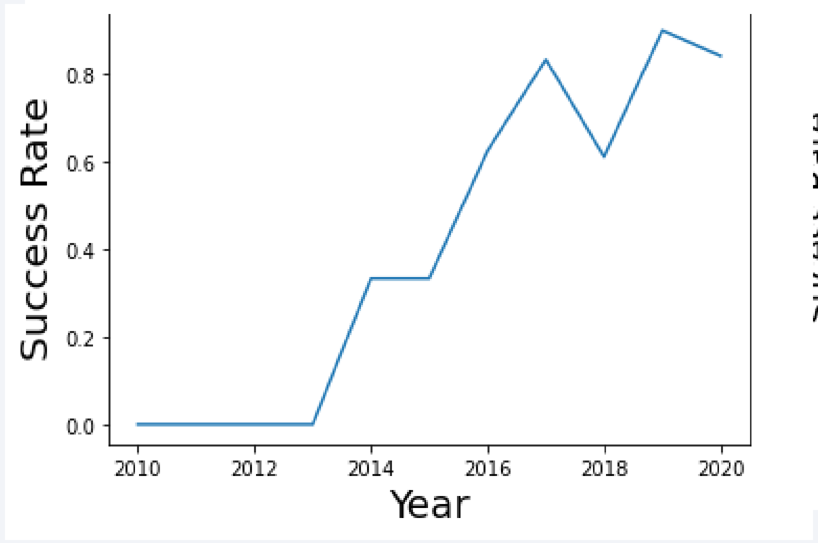
```
df['Class']=landing_class
df[['Class']].head(8)
```

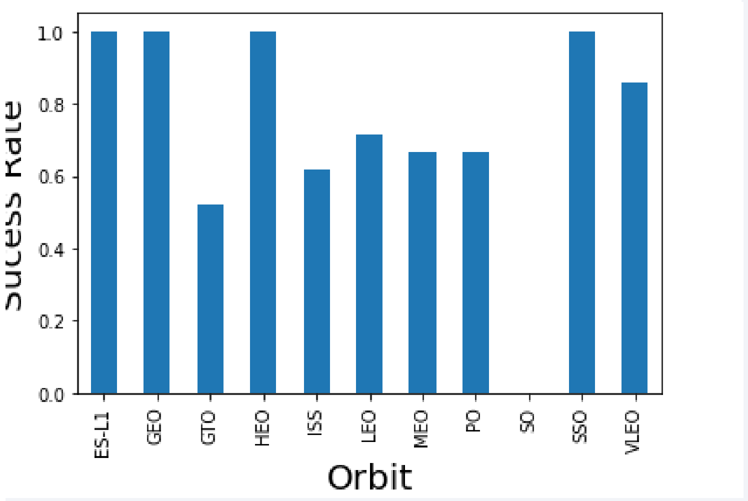
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

https://github.com/gabrielaberrios1/projectos/blob/main/labs-jupyter-spacex-Data%20wrangling%20(1).ipynb

EDA with Data visualization







WE CREATED DIFFERENT TO CHARTS TO COMPARED DIFFERENT FACTORS

We created different charts to compared different factors, such as orbit, flight number, payload mass, launch site.

We also plotted success rate years from 2010-2020. And visualized orbits success rate with a bar chart

https://github.com/gabrielaberrios1/projectos/blob/main/jupyter-labs-eda-dataviz%20(1)(1).ipynb

EDA with SQL

- 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 using 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
- https://github.com/gabrielaberrios1/projectos/blob/main/jupyter-labs-eda-sql-coursera%20(5).ipynb

Build an Interactive Map with Folium

- I created `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate, the Nasa Johnson Space center.
- I created folium.marker to create a pointer or marker on the same coordinates.
- Then I created the same objects on the different launch sites to mark the success/failed launches for each site and assigned the launch outcomes to classes 0 and 1 with green and red.
- I calculated the distance between different launch sites and the coast and other different locations to find trends around launch sites.
- https://github.com/gabrielaberrios1/projectos/blob/main/lab_jupyter_launch_site_location%20(1).ipynb

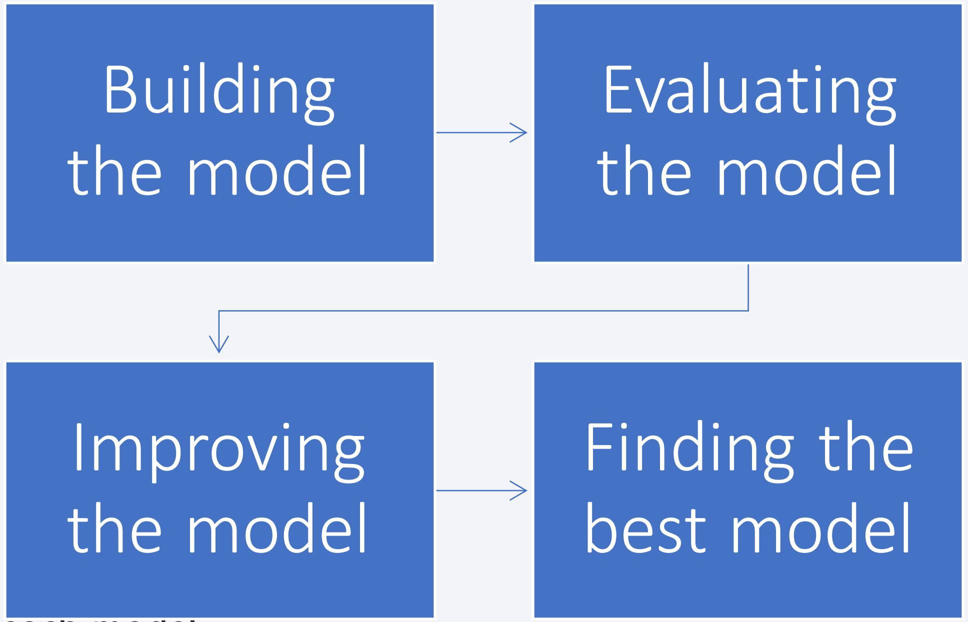
Build a Dashboard with Plotly Dash

- Graphs created-
  - Pie Chart shows the total launches by site-location
  - Scatter Graph : Outcome VS Payload Mass(KG) for different Booster Versions
- I wanted to check all graph and see that locations had more successful launches.
- Also how the payload mass could affect how succesful a launch could be
- https://github.com/gabrielaberrios1/projectos/blob/main/spacex_dash_app.py
- https://gblberrios-8050.theiadocker-2-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved,

  and found the best performing classification model
- I loaded the data set using Numpy and Pandas
- Transformed the Data
  Split the data into training set and data set
- Set parameters and classification model to GridSearchCV
- Fit the data set into GridSearchCV for each model



- After finding accuracy we found tuned hyper parameters for each model
- Plot created: Confusion Matrix for each model
- To Find the best performing had the highest accuracy.
- https://github.com/gabrielaberrios1/projectos/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5%20lab.ipynb

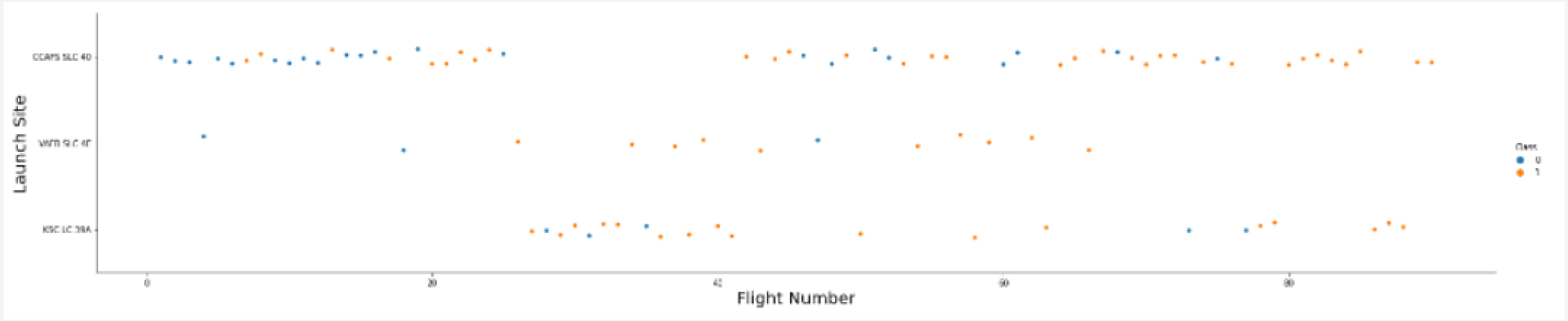
Results

- Exploratory data analysis results
- Using the payload charts we see that the lowest had better launch success.
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

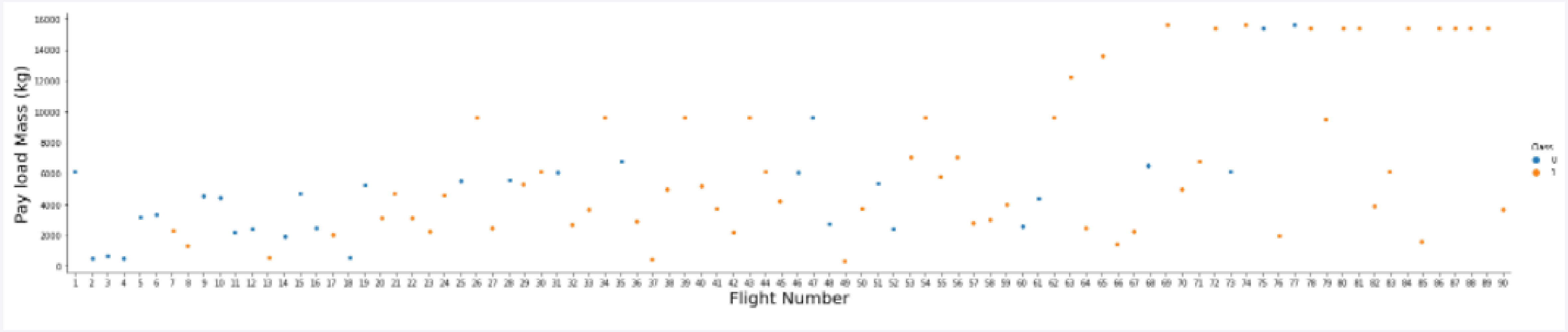
Insights drawn from EDA

Flight Number vs. Launch Site



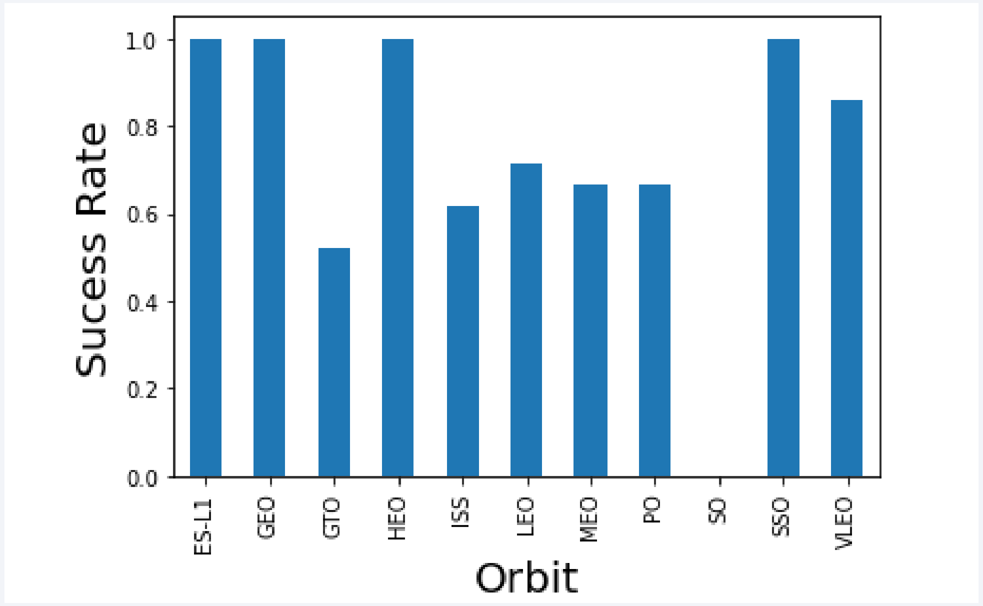
We see that the flight number factor affected the success rate. The higher the flight number increased the succes rate.

Payload vs. Launch Site



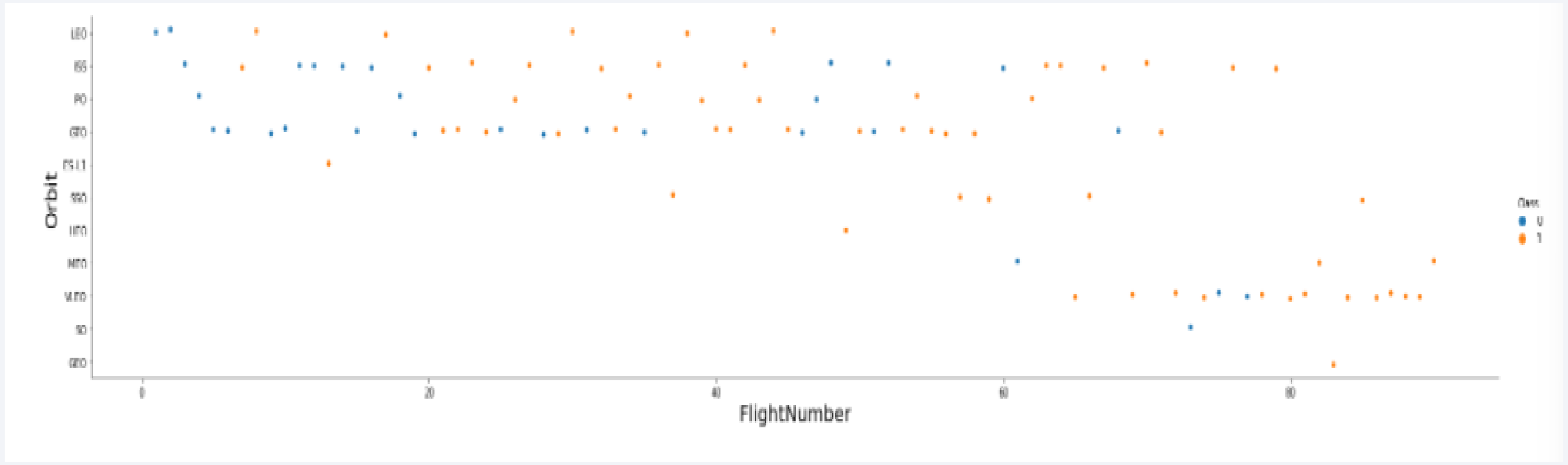
- Here we see that lower the payload mas the more succesful launches.

Success Rate vs. Orbit Type



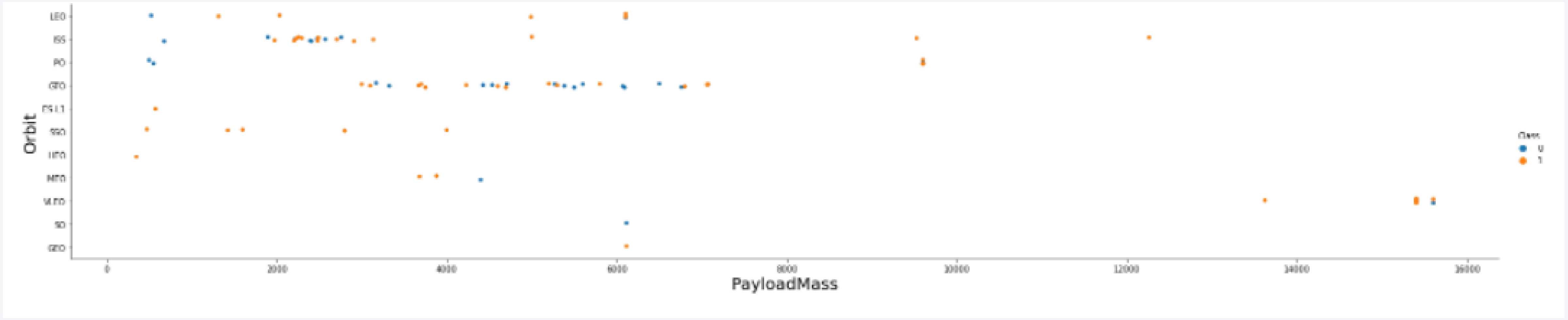
- The orbits with highest success rate are ES-L1, GEO, HEO, SSO.

Flight Number vs. Orbit Type



- The LEO orbit has the highest success rate.

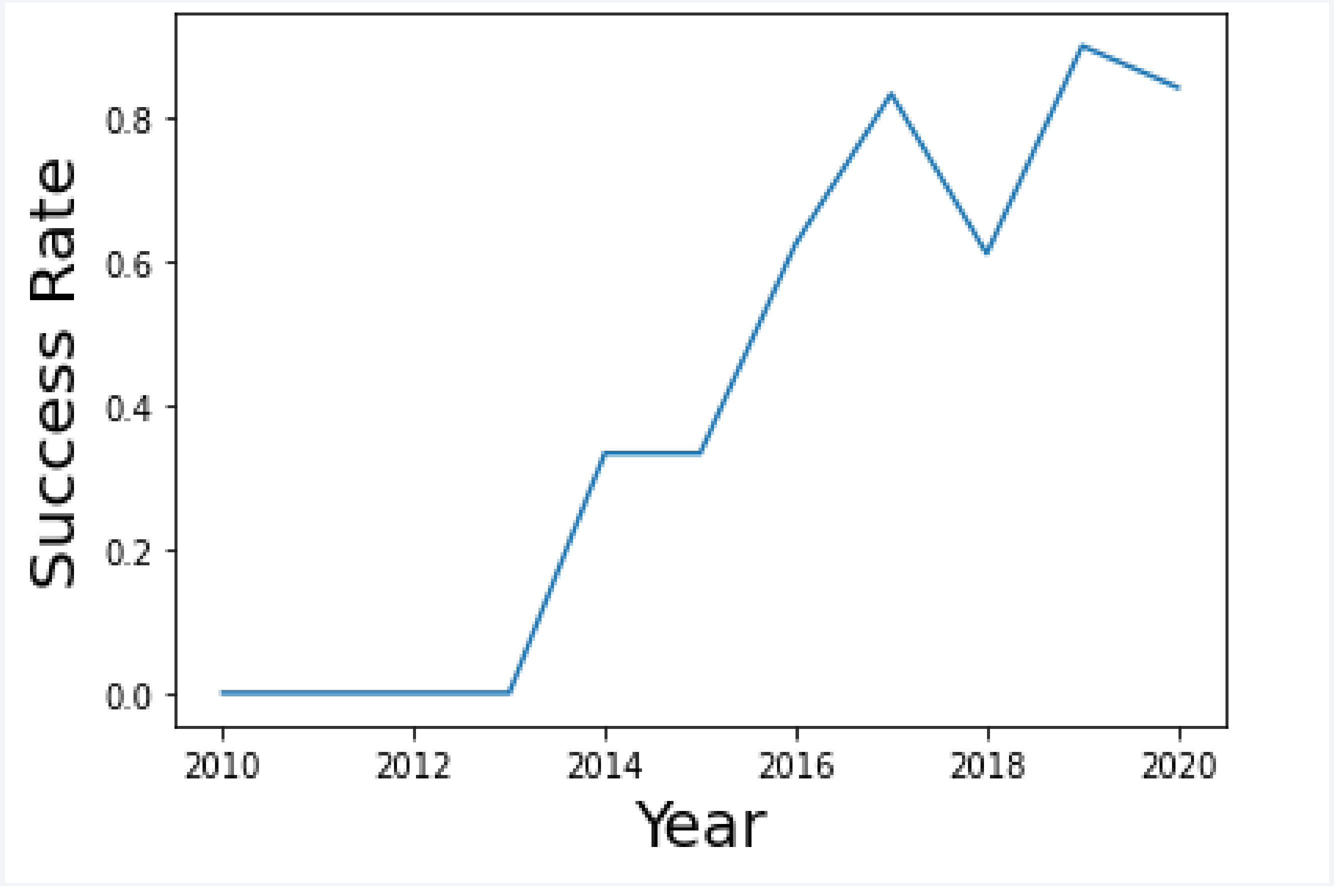
Payload vs. Orbit Type



- The highest success rate was achieved by LEO orbit but mostly had PayloadMass less than 6000kg.

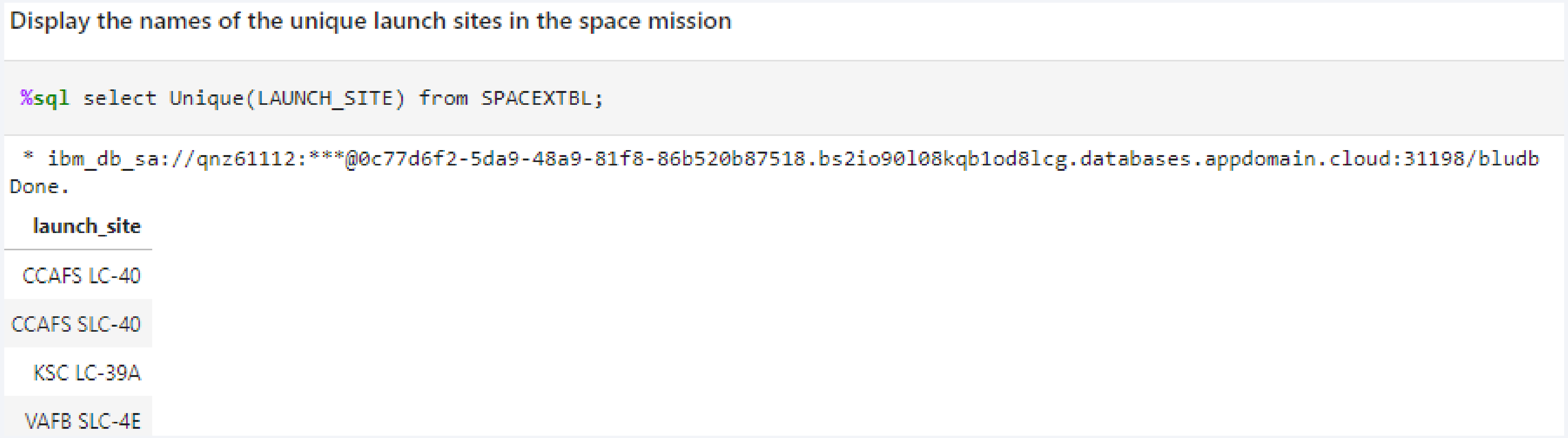
Launch Success Yearly Trend

- There is an increase of success rate after 2015 year to 2020.



All Launch Site Names

The keyword unique displays all the launch sites from spacex data.



Display the names of the unique launch sites in the space mission

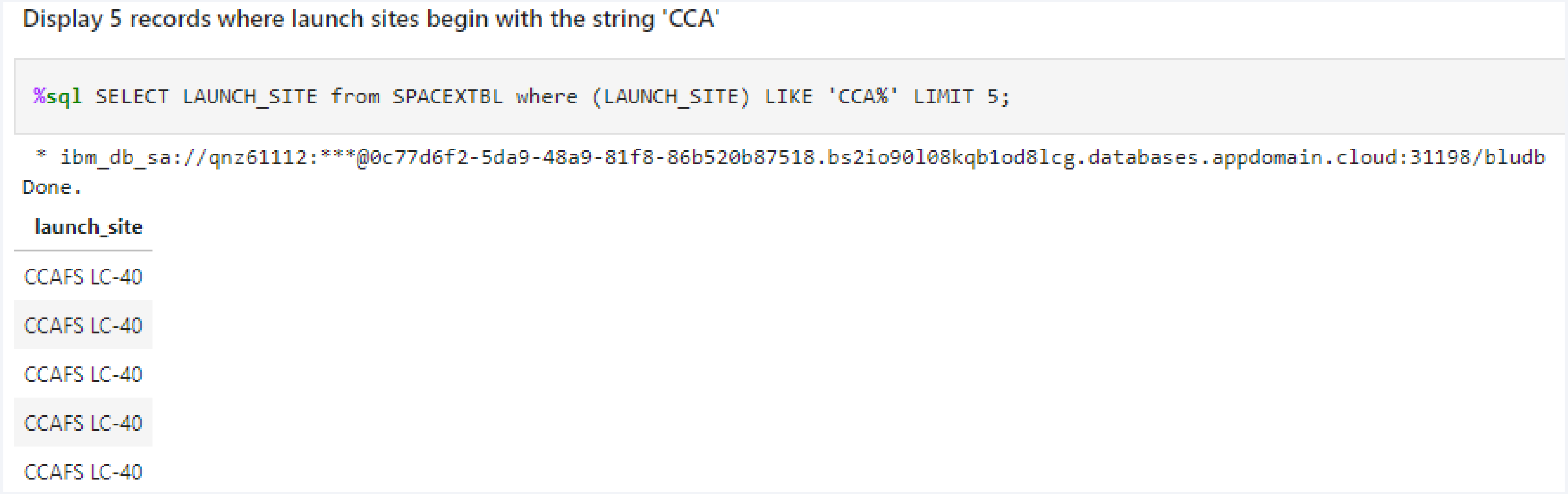
```
%sql select Unique(LAUNCH_SITE) from SPACEXTBL;
```

```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

- Here is display the launches that begin with CCA using "CCA%"



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

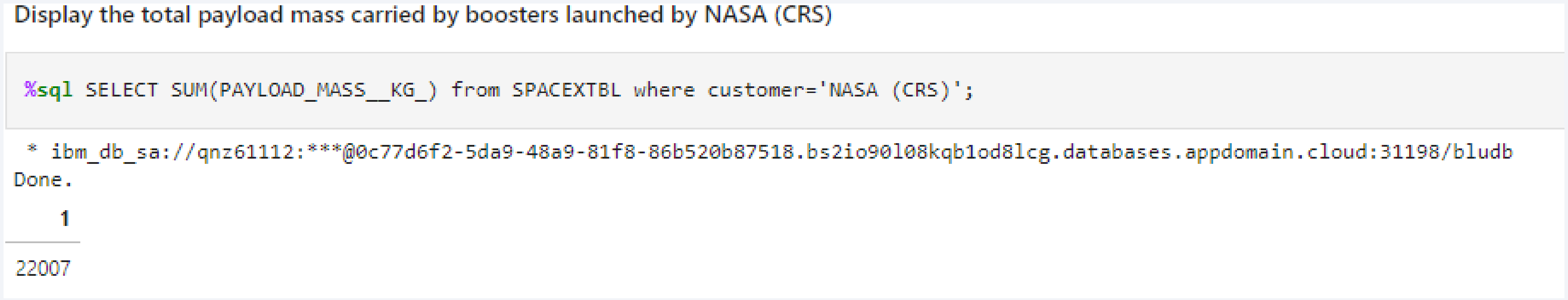
```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- We need the use the keyword sum order to summarized all the payload mass from that column.



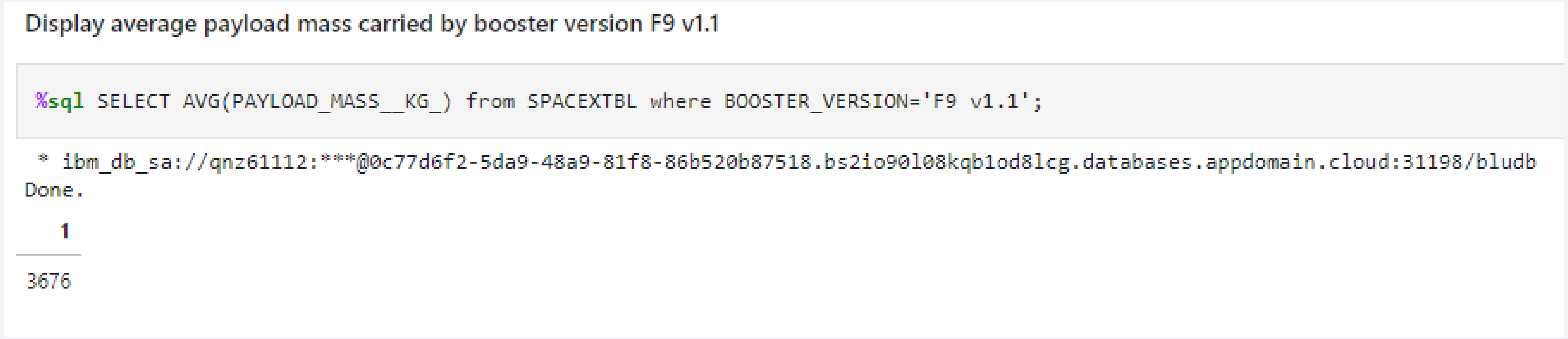
Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) from SPACEXTBL where customer='NASA (CRS)';
```

```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
    1
22007
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- To calculate with use the keyword avg from the column we used avg.



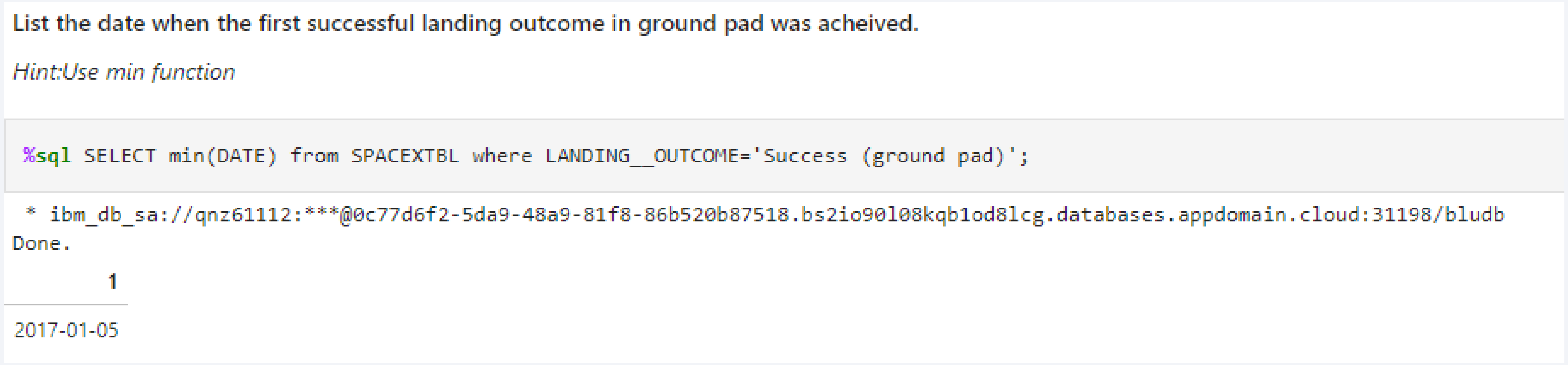
Display average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION='F9 v1.1';
```

```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
   1
3676
```

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here



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

Hint:Use min function

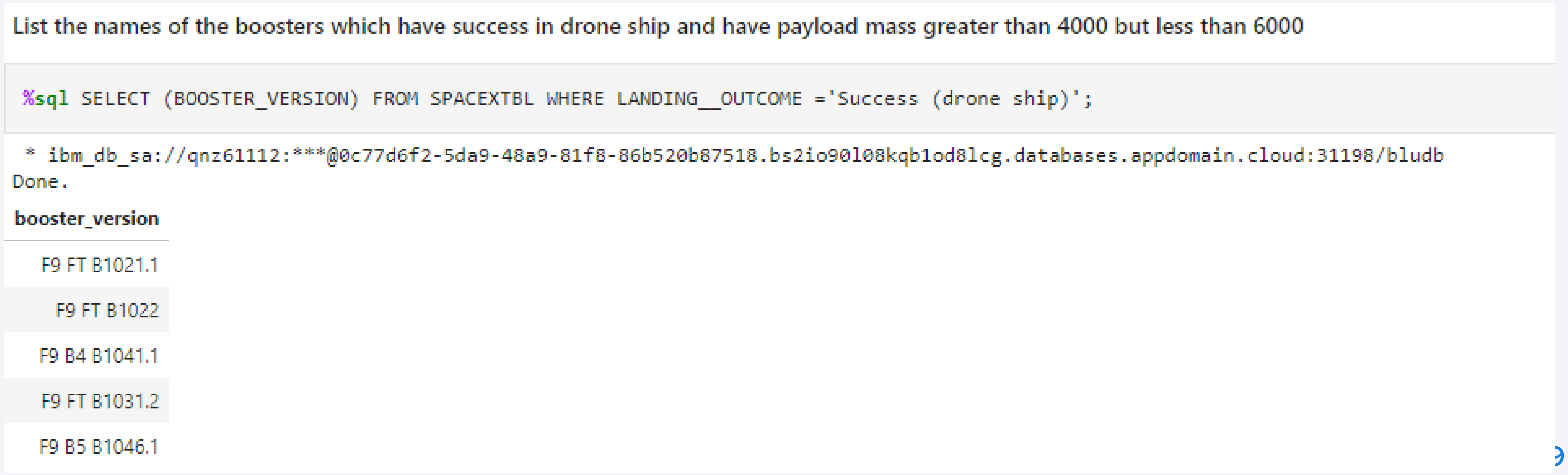
```
%sql SELECT min(DATE) from SPACEXTBL where LANDING__OUTCOME='Success (ground pad)';
```

```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

1
2017-01-05

Successful Drone Ship Landing with Payload between 4000 and 6000

- To find the booster version that have success but with a limit payload mass we need to use where keyword.



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

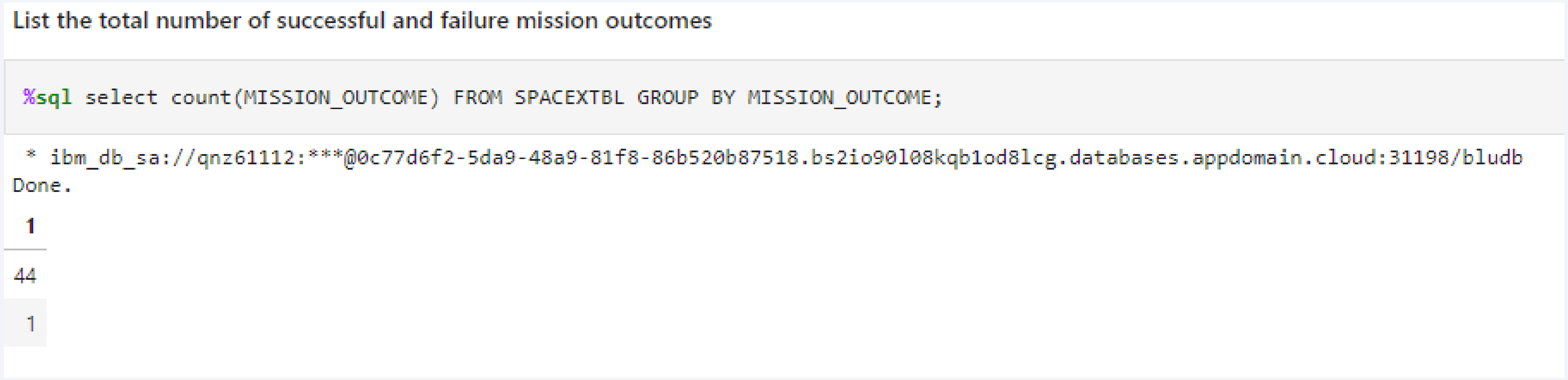
```
%sql SELECT (BOOSTER_VERSION) FROM SPACEXTBL WHERE LANDING__OUTCOME ='Success (drone ship)';
```

```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

booster_version
F9 FT B1021.1
F9 FT B1022
F9 B4 B1041.1
F9 FT B1031.2
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

- Here we need to the group by and count keyword to find the number of successful and failure mission outcomes.



List the total number of successful and failure mission outcomes

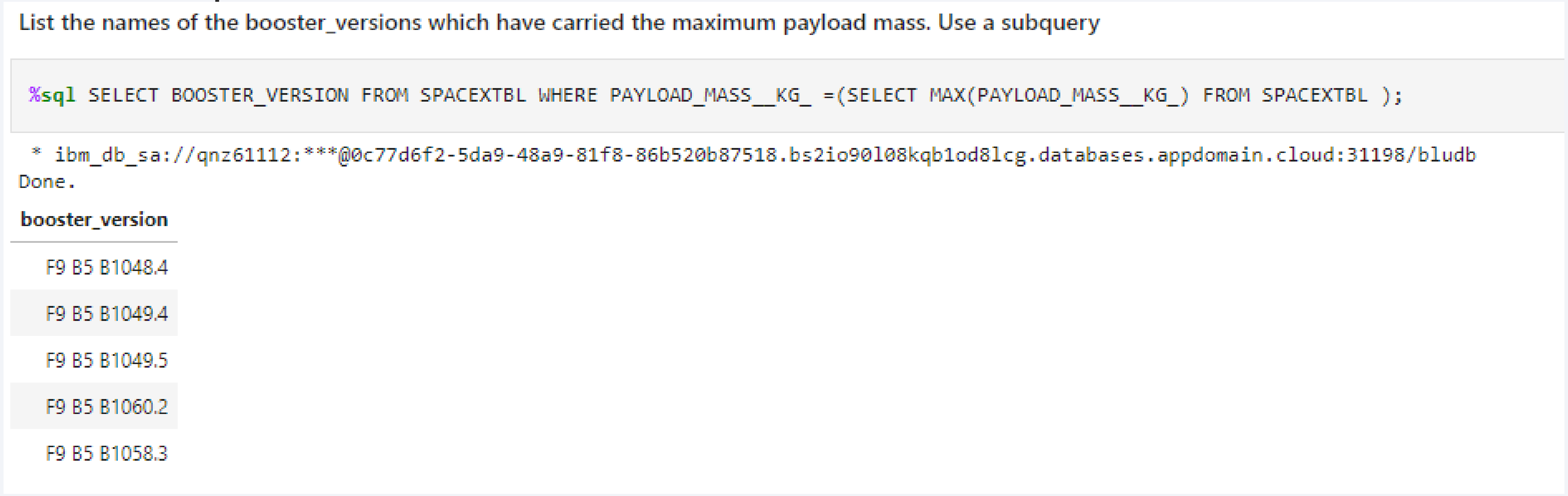
```
%sql select count(MISSION_OUTCOME) FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

1
44
1

Boosters Carried Maximum Payload

- To list the names of the booster_version which have carried the maximum payload mass we used subquery selecting the maximun payload_mass_kg from the spacextbl.


List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

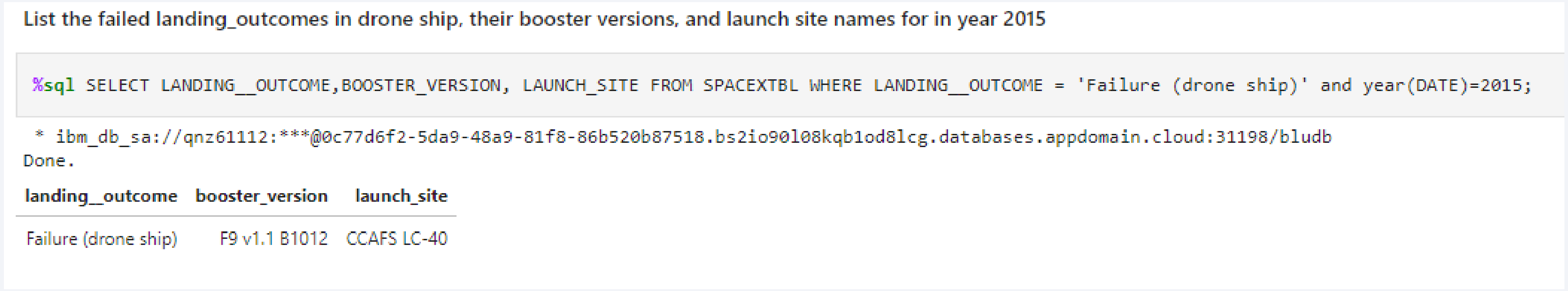
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL );
```

```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

booster_version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3

2015 Launch Records

- To list the failed landing_outcome we used where keyword and specified the year date.



List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

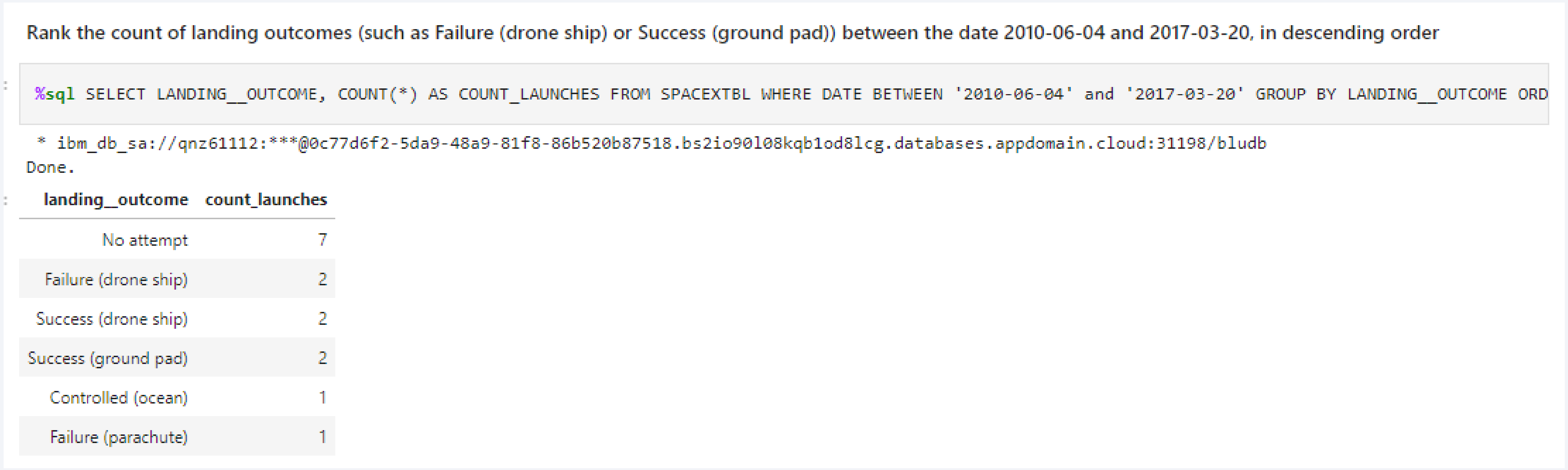
```
%sql SELECT LANDING__OUTCOME,BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Failure (drone ship)' and year(DATE)=2015;
```

```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	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
- Here we used between keyword and also group by and count(*).



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

```
%sql SELECT LANDING__OUTCOME, COUNT(*) AS COUNT_LAUNCHES FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' and '2017-03-20' GROUP BY LANDING__OUTCOME ORD
```

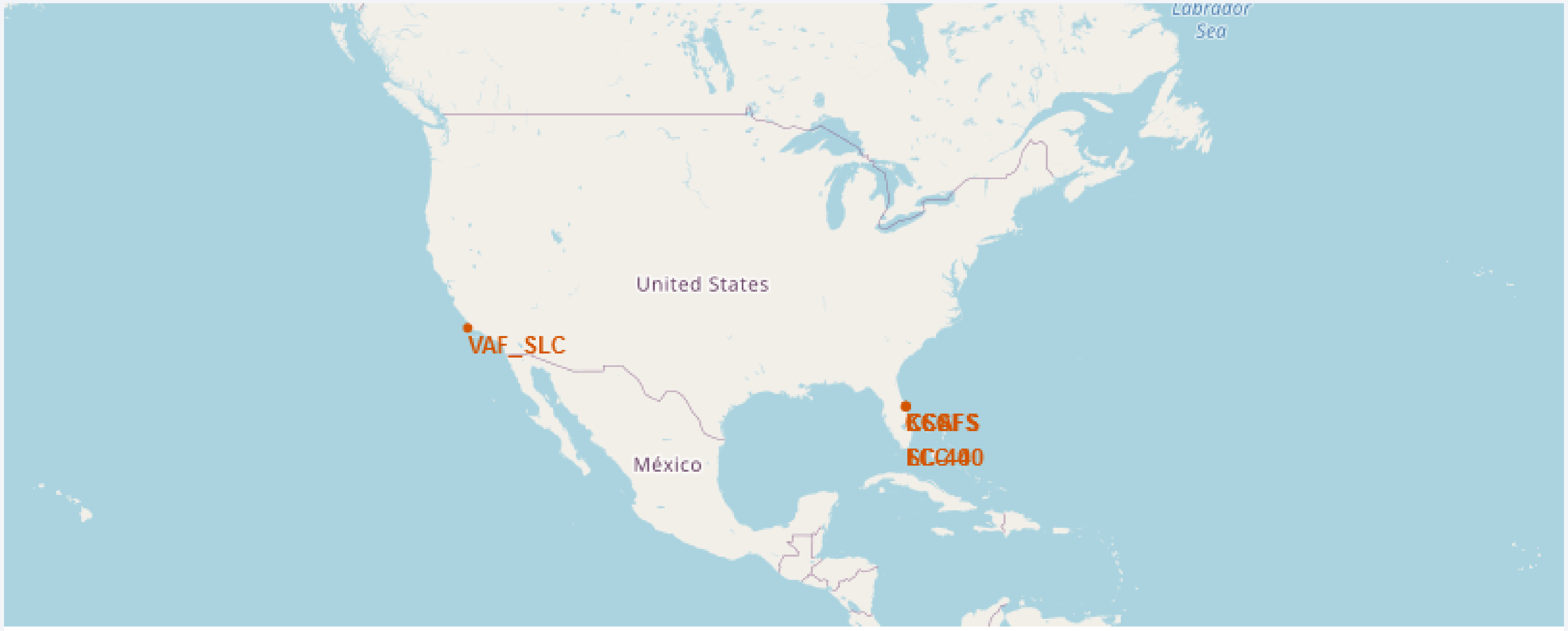
```
 * ibm_db_sa://qnz61112:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

landing__outcome	count_launches
No attempt	7
Failure (drone ship)	2
Success (drone ship)	2
Success (ground pad)	2
Controlled (ocean)	1
Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

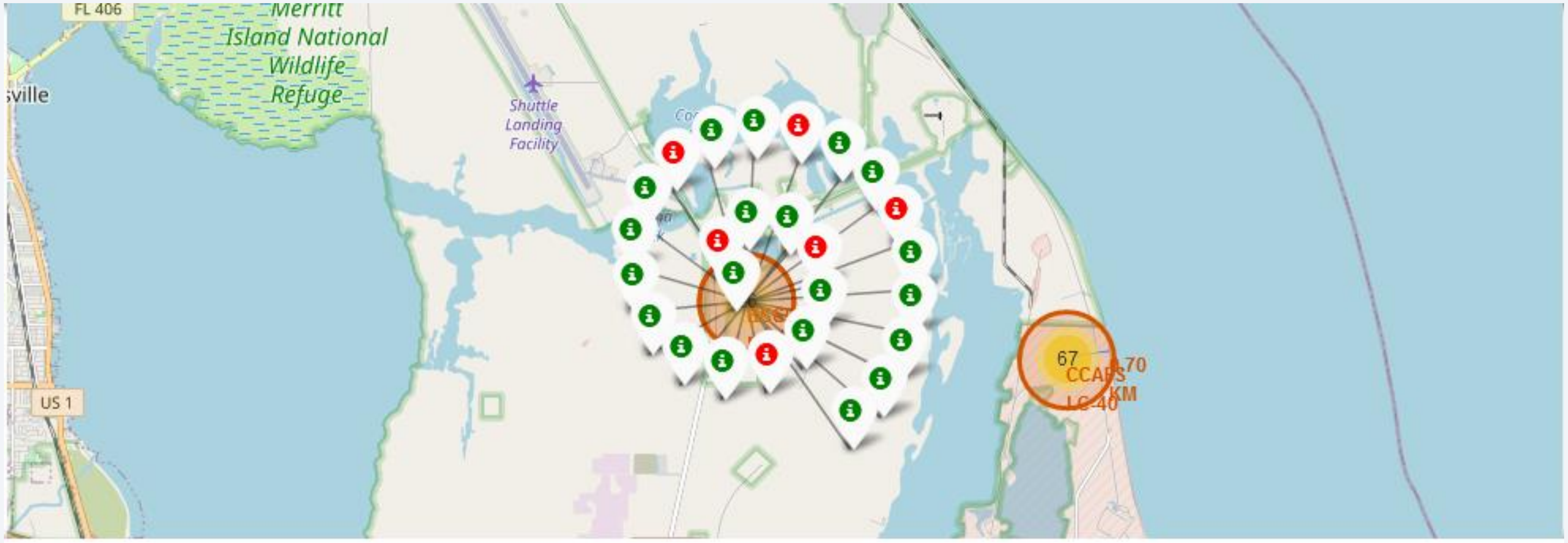
LaunchSites locations markers



- These markers are located in Florida and California, both close the coast.

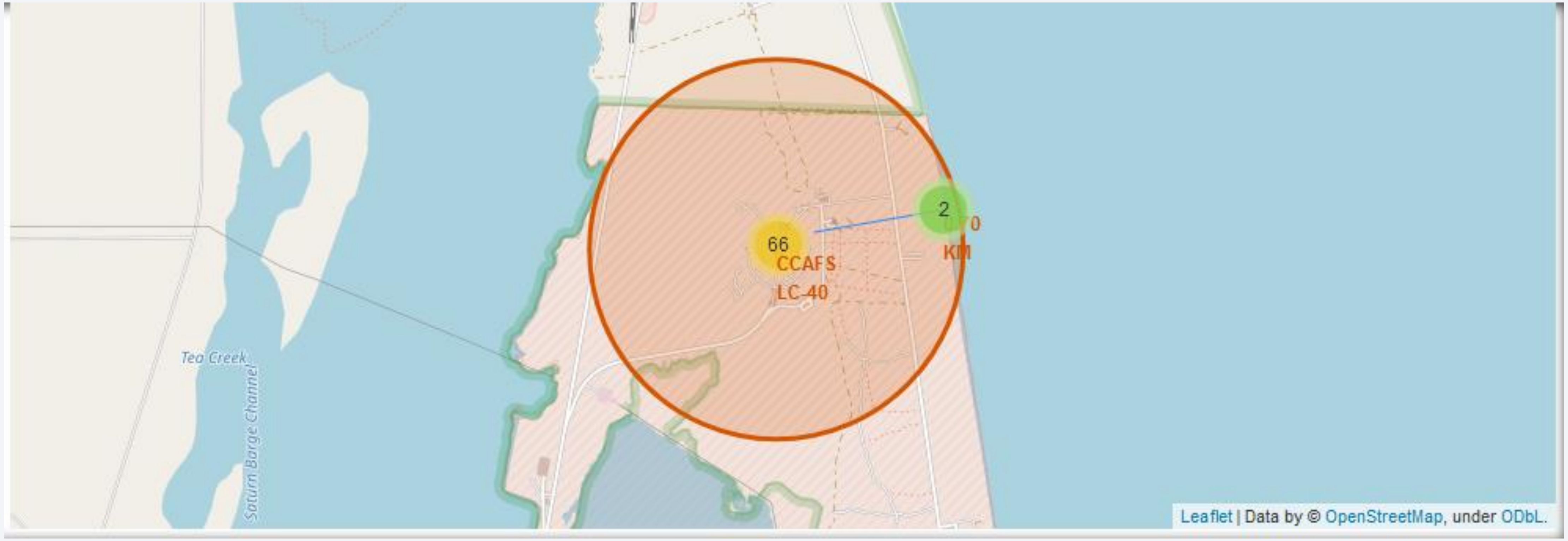
Color labeled launch outcomes

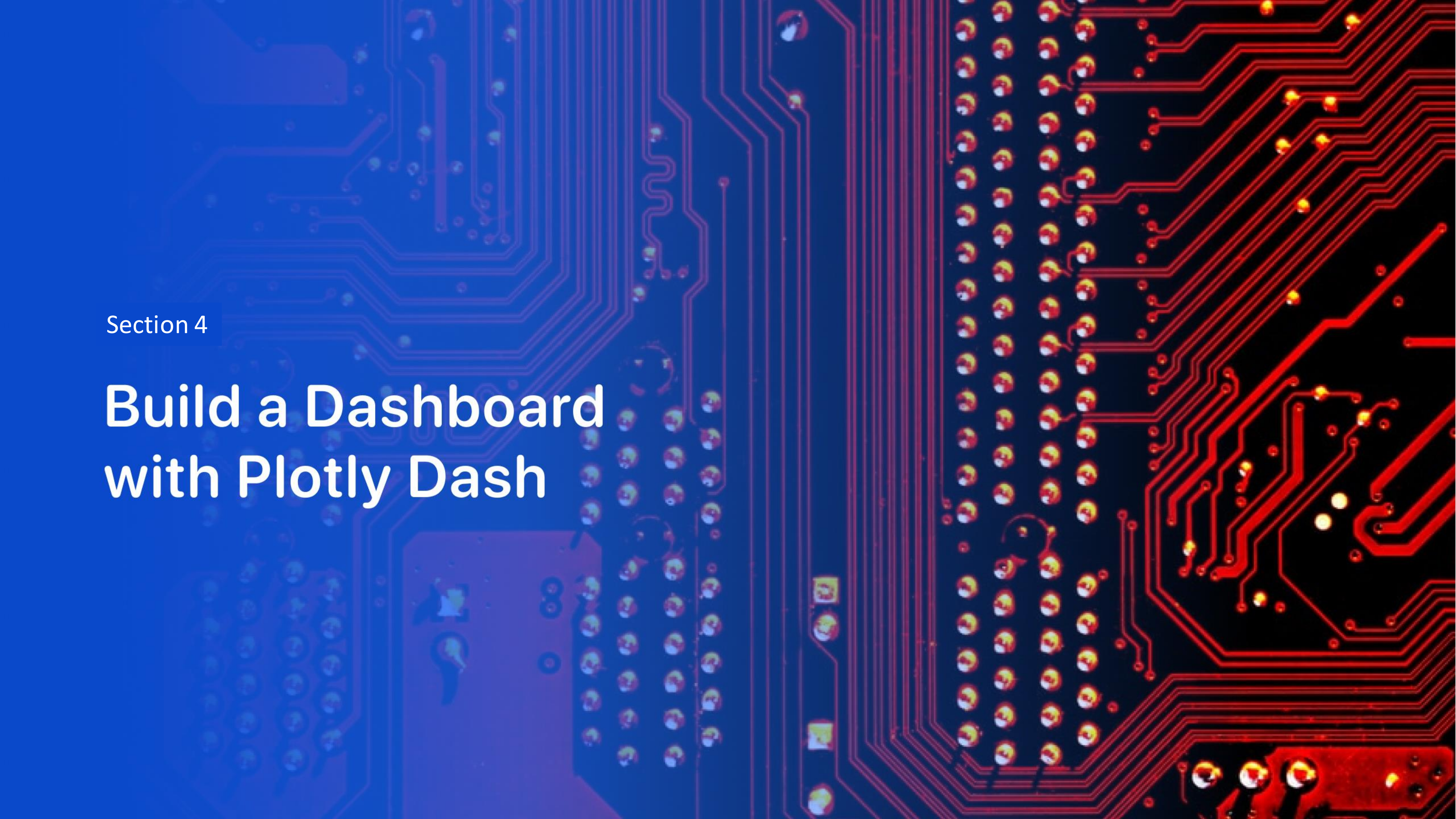
- Here we are able to see both colors green and red that represents if a launch was successful or not.



Distance marker map

- The distance between the lunch site CCAFS and the coastline is 0.70 km. We see the importance of being close to the coastline for security reasons.



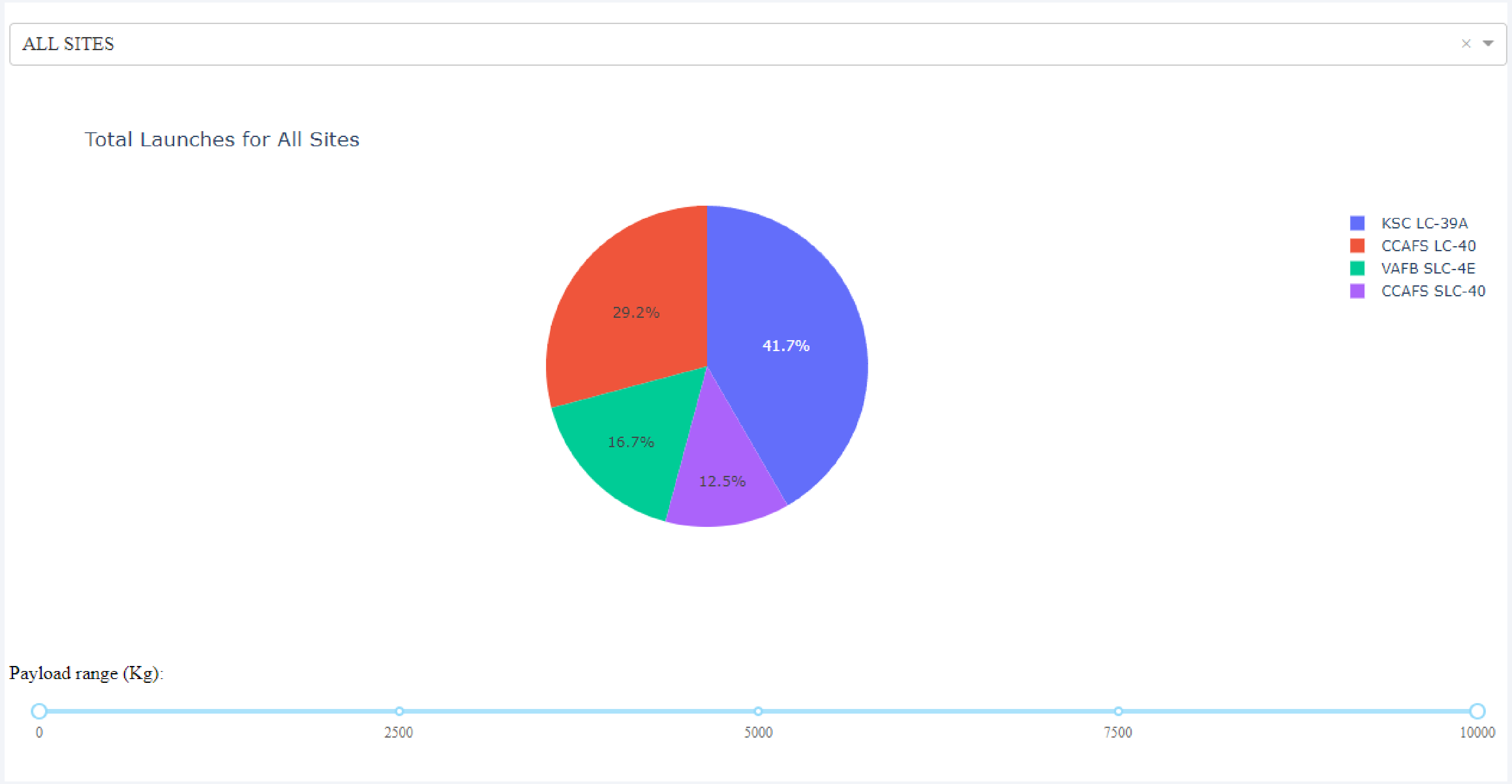


Section 4

Build a Dashboard with Plotly Dash

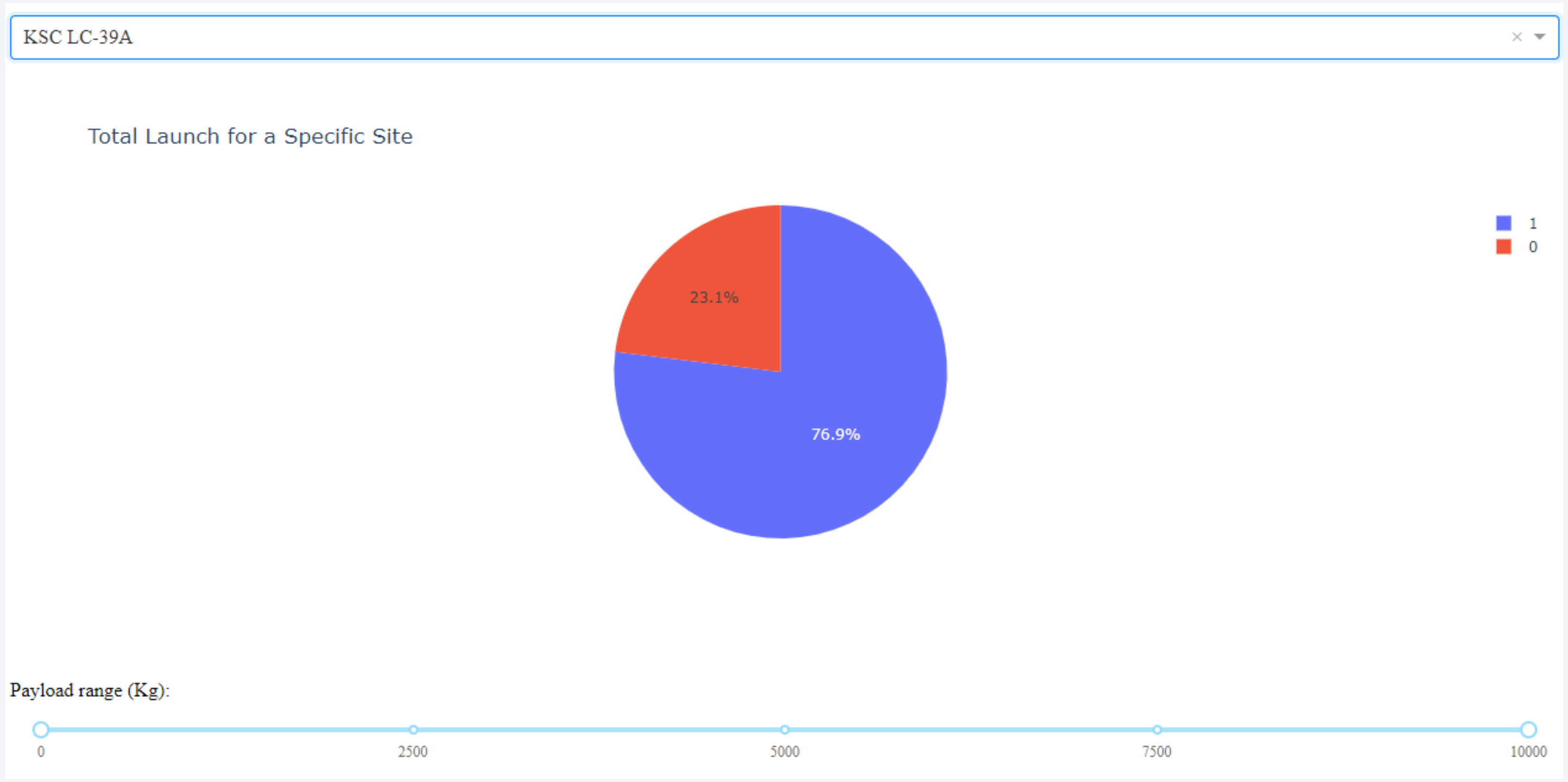
SpaceX Launch for all sites

- We see here that KSC-LC-39A had the most successful launches.



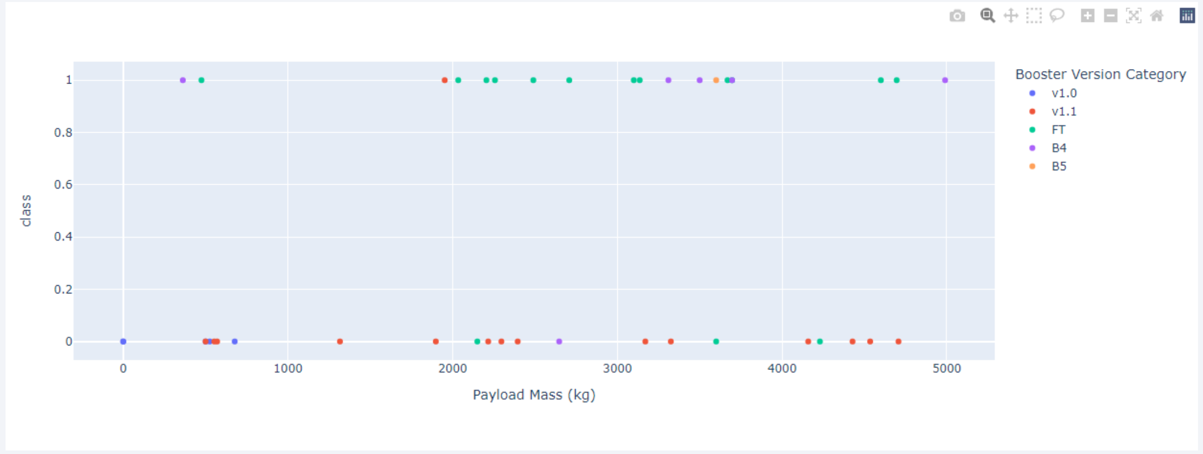
KSC LC-39A Launches

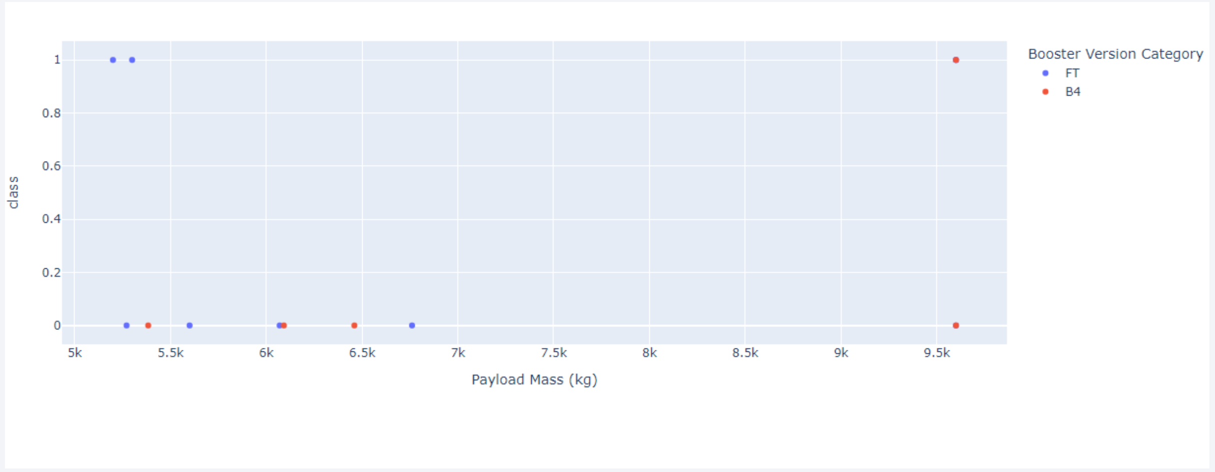
- KSC LC—39A had success rate of 76.9% make it the best location for launch-site



Scatter plot of Payload vs Launch Outcome for all sites

- The first plots represents payloads from 0kg-5000kg and the second plot payloads over 5000kg-10000kg
- The success rate for lower payloads is higher than the heavier payloads



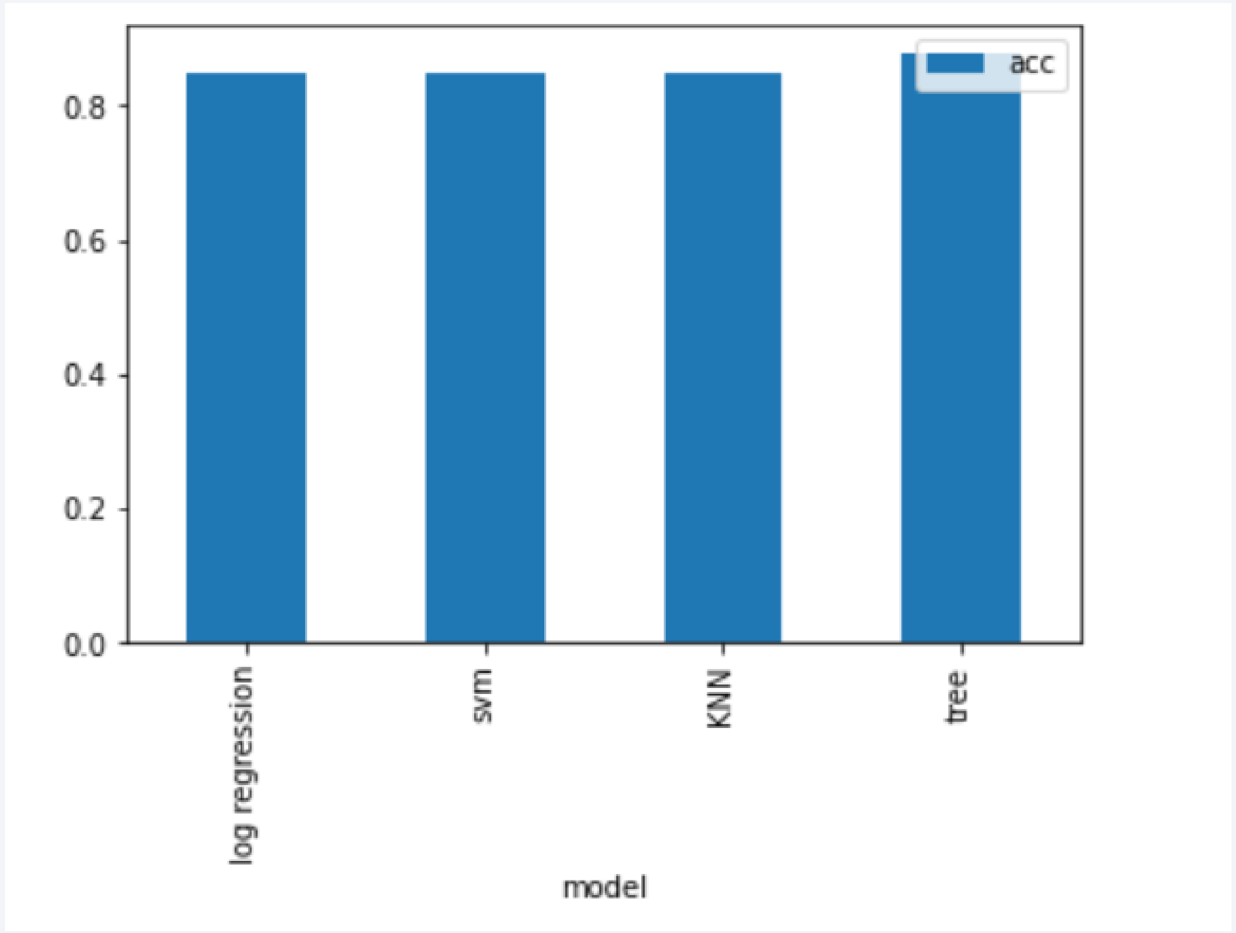


Section 5

Predictive Analysis (Classification)

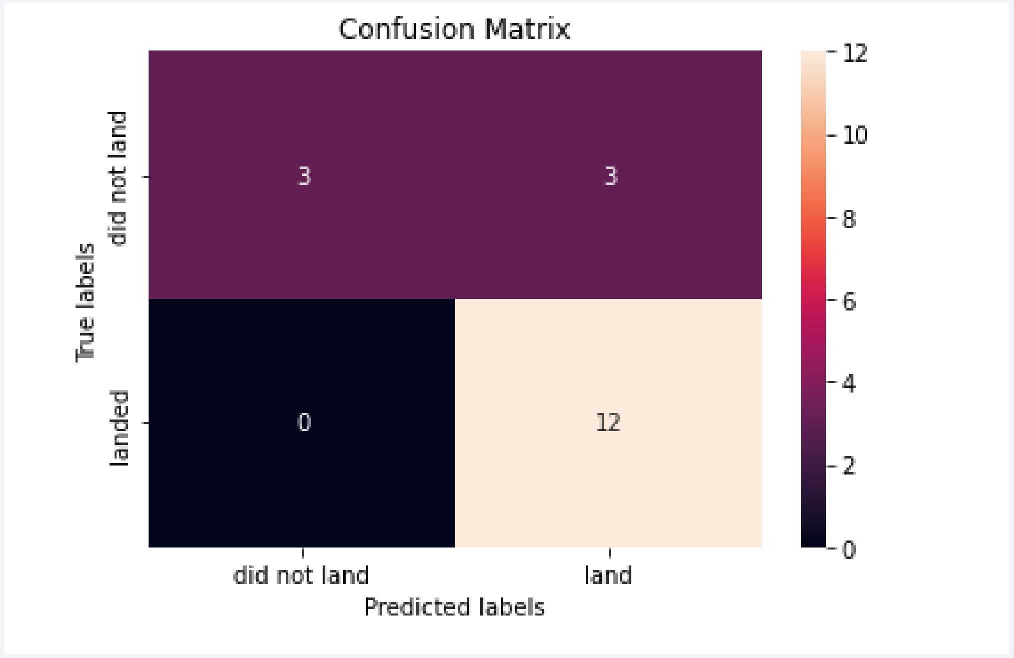
Classification Accuracy

- Here we can see the bar chart for different model accuracy values.
- The tree model has the highest classification accuracy in this project



Confusion Matrix

- Best performing model was tree model



Conclusions

- We conclude that the lower the PayloadMass the more successful launches.
- Tere is an increase of launch success rate from 2013-2020 .
- The decision tree model has the highest success rate.
- KSC-LC-39A had the most successful launches.

Thank you!