IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

3rd May 2023

Outline

Executive Summary

- Summary of methodologies
  - Data collection using an API and web scraping
  - Exploratory Data Analysis using Python and SQL
  - Creation of an interactive geographical map using Folium
  - Creation of an interactive dashboard using Plotly Dash
  - Predictive analysis with machine learning
- Summary of all results
  - Exploratory data analysis results
  - Interactive dashboard and maps
  - Predictive (machine learning) results

Introduction

- Project background and context
  - SpaceX charges $62 million for a Falcon 9 launch
  - Other existing providers charge $165 million for as launch
  - Price difference because SpaceX can reuse the first stage
  - The prediction of whether the stage will land determines the cost of a launch
- We want to answer the following questions:
  1. What are the main characteristics of a successful or failed landing?
  2. What are the effects of each the rocket variables on the success or failure of a landing?
  3. What are the conditions which will allow SpaceX to achieve the best landing success rate?

Section 1

Methodology

Methodology

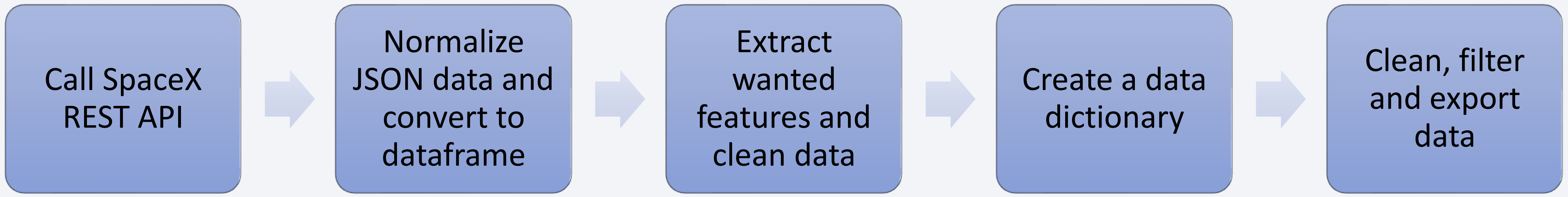
Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web scraping from Wikipedia
- Perform data wrangling
  - Drop columns that are not needed
  - Use one hot encoding for the 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

These data analysis processes were carried out using the Python programming language and documented in a series of Jupyter notebooks that contain the code and results of each step.
These notebooks are stored on GitHub and can be accessed using the links in this presentation, or follow this link to access the full list of project notebooks: Project link

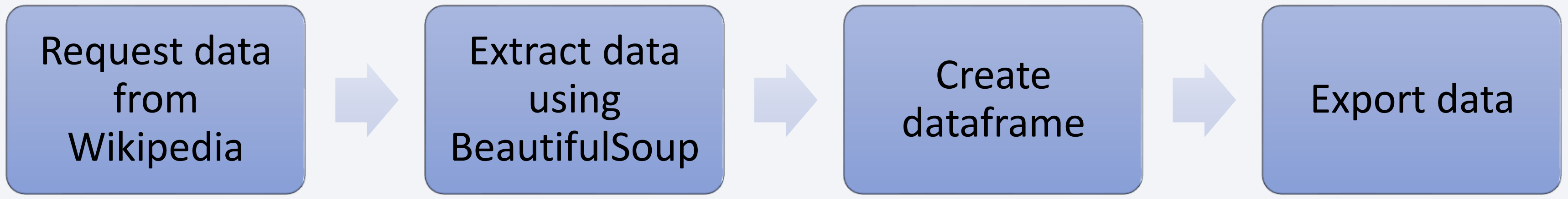
Data Collection: Overview

- Data sets were collected using SpaceX REST API and by web scraping Wikipedia
- SpaceX REST API (https://github.com/r-spacex/SpaceX-API)


Call SpaceX REST API
Normalize JSON data and convert to dataframe
Extract wanted features and clean data
Create a data dictionary
Clean, filter and export data

- Web scraping Wikipedia page

(https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922)



Data Collection – SpaceX API part 1

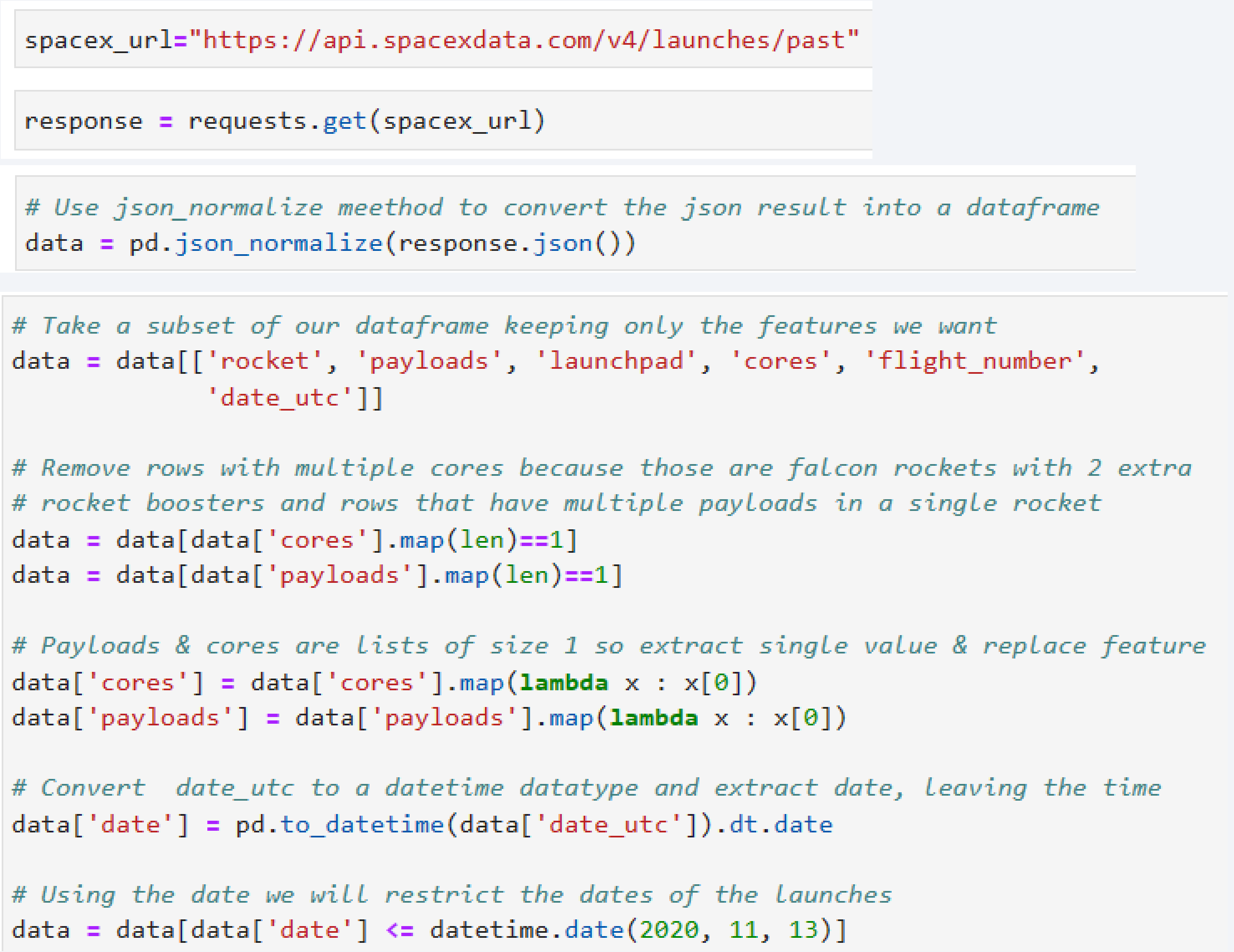
Use SpaceX API to request data

Normalize data and convert to a dataframe

Extract wanted features and clean the data

Link to code

continues



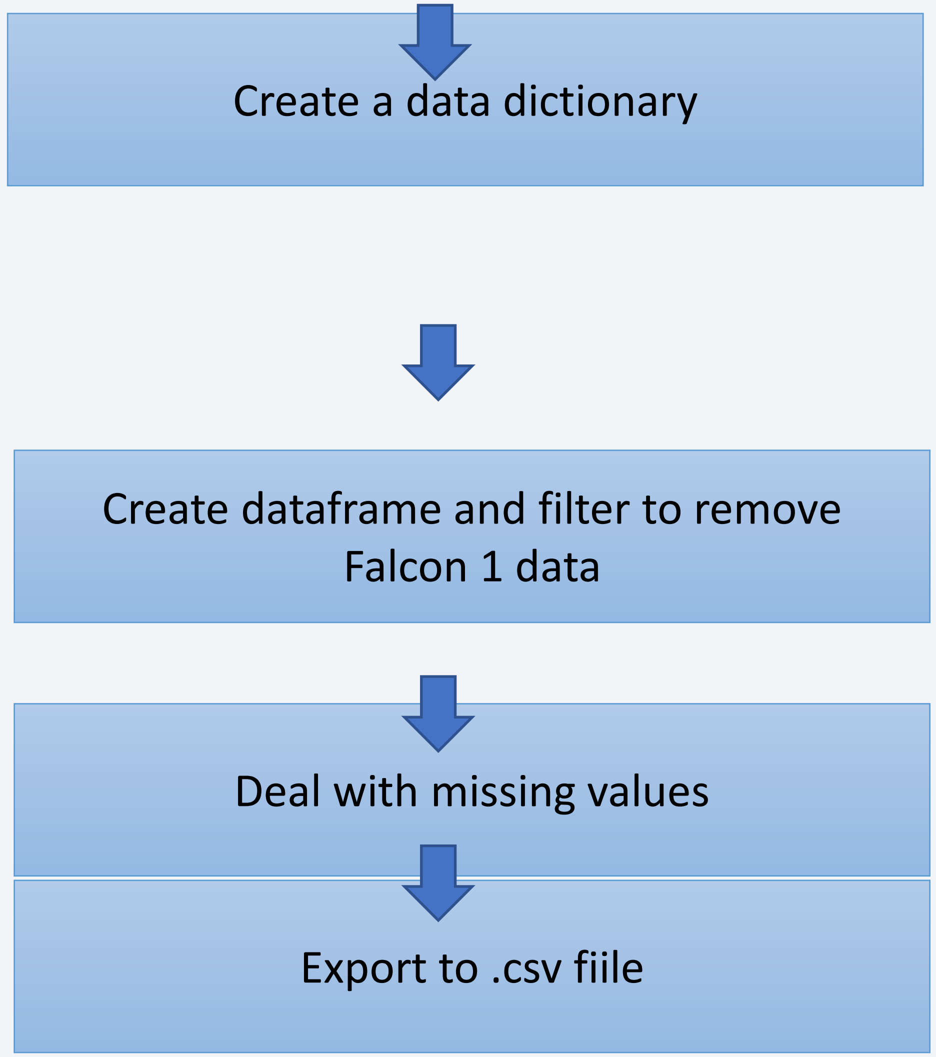
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

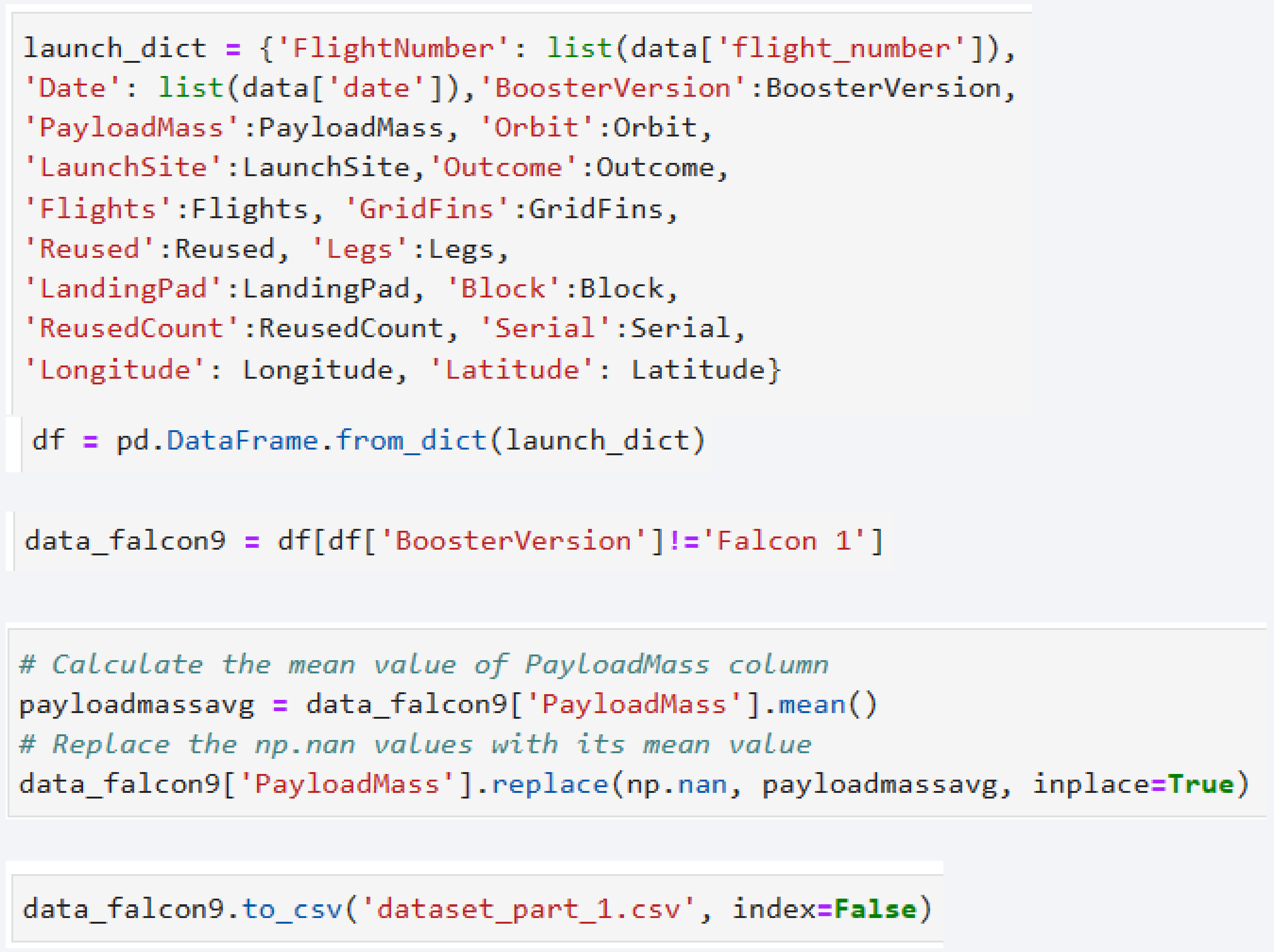
```
response = requests.get(spacex_url)
```

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

```
# Take a subset of our dataframe keeping only the features we want
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number',
             'date_utc']]

# Remove rows with multiple cores because those are falcon rockets with 2 extra
# rocket boosters and rows that have multiple payloads in a single rocket
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Payloads & cores are lists of size 1 so extract single value & replace feature
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# Convert  date_utc to a datetime datatype and extract 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 – SpaceX API part 2





```
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.from_dict(launch_dict)
```

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

```
# Calculate the mean value of PayloadMass column
payloadmassavg = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, payloadmassavg, inplace=True)
```

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

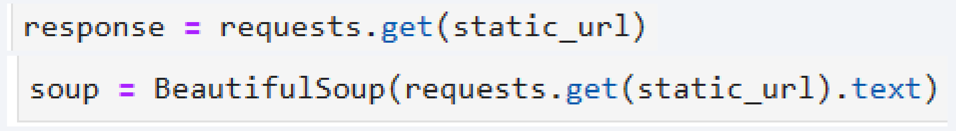
Data Collection – Web scraping part 1

Request Falcon 9 Wiki page and use it to create a BeautifulSoup object

Extract column names from the HTML table header

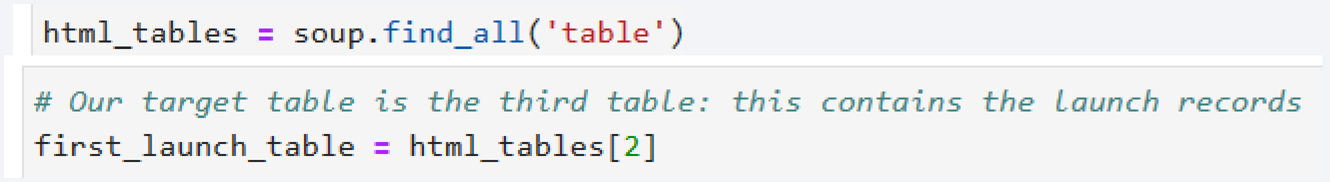
Get all the column names

continues



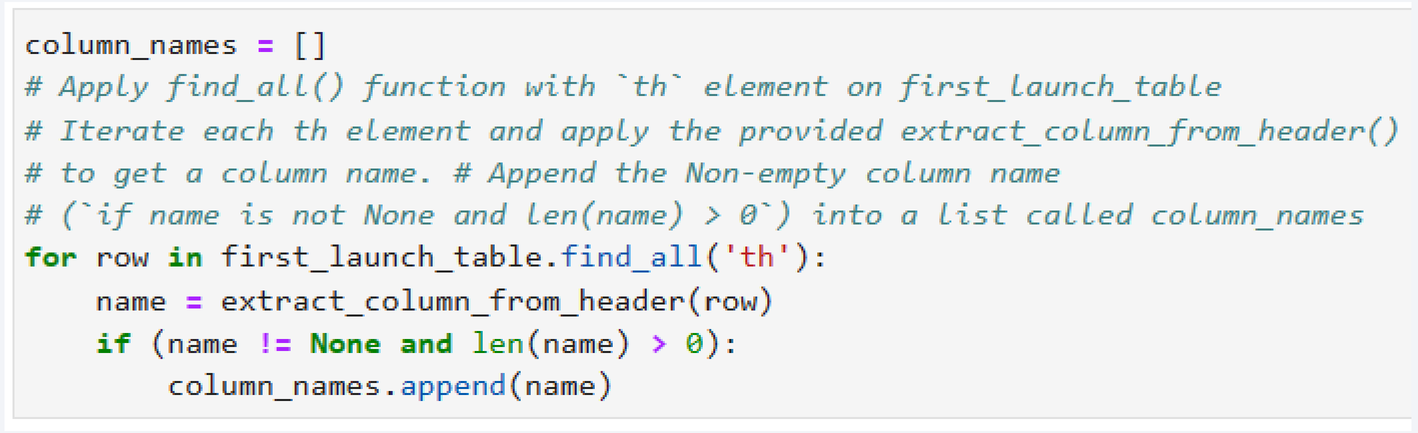
```
response = requests.get(static_url)
```

```
soup = BeautifulSoup(requests.get(static_url).text)
```



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

```
# Our target table is the third table: this contains the launch records
first_launch_table = html_tables[2]
```

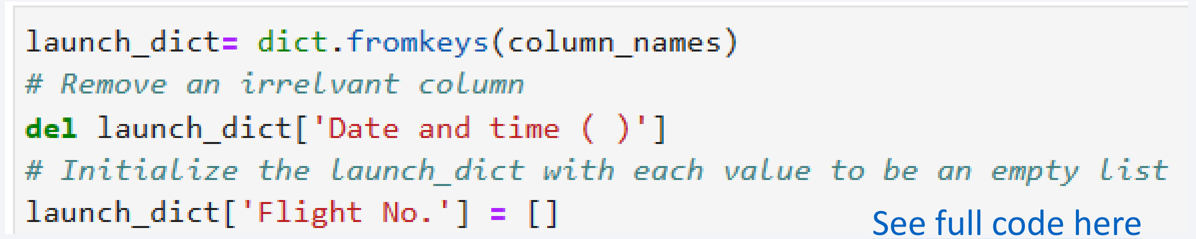


```
column_names = []
# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header()
# to get a column name. # Append the Non-empty column name
# (`if name is not None and len(name) > 0`) into a list called column_names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

Link to code

Data Collection – Web scraping part 2

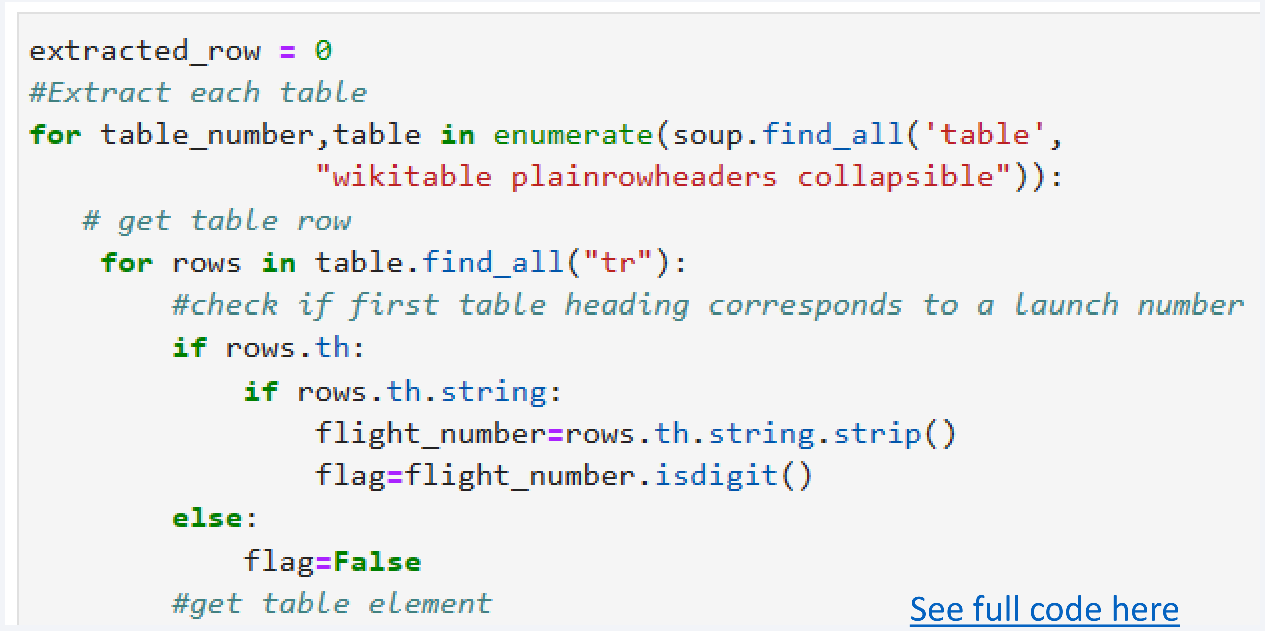
Create a data dictionary



```
launch_dict= dict.fromkeys(column_names)
# Remove an irrelvant column
del launch_dict['Date and time ( )']
# Initialize the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
```

See full code here

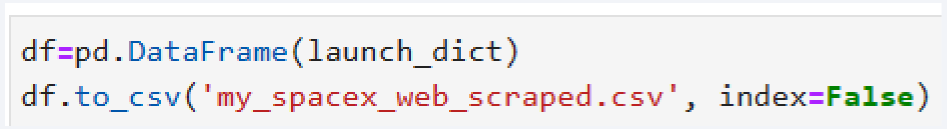
Add data to keys



```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',
                        "wikitable plainrowheaders collapsible")):
    # get table row
     for rows in table.find_all("tr"):
         #check if first table heading corresponds to a launch number
         if rows.th:
             if rows.th.string:
                 flight_number=rows.th.string.strip()
                 flag=flight_number.isdigit()
         else:
             flag=False
         #get table element
```

See full code here

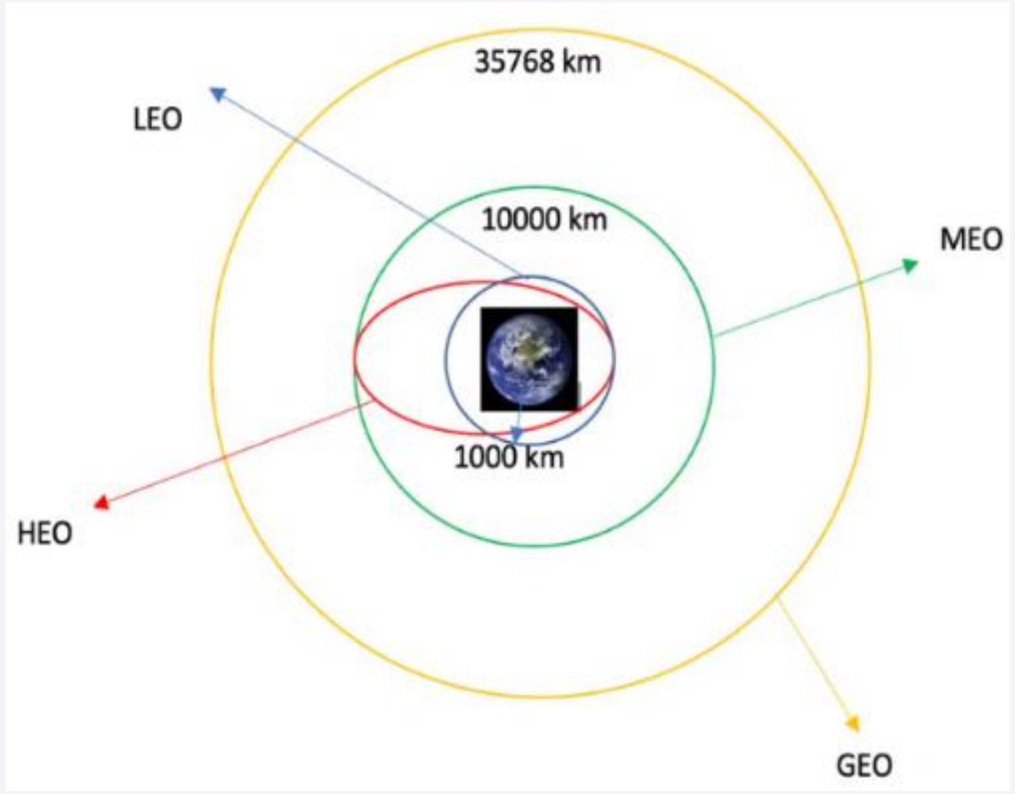
Create a dataframe and xxport to .csv fiile



```
df=pd.DataFrame(launch_dict)
df.to_csv('my_spacex_web_scraped.csv', index=False)
```

Data Wrangling: Overview

- Data wrangling is the step where data is cleaned, unified and summarized prior to visualization.
- The data contains several SpaceX launch sites. Each launch aims at a particular orbit, some of which are shown in the graphic on the right.
- First we calculate the number of launches from each site, and the mission outcome per orbit type.
- Next we create a landing outcome label for success versus failure. This will make statistical analysis and machine learning easier in later stages of the project.
- Finally we export the data to a .csv file.



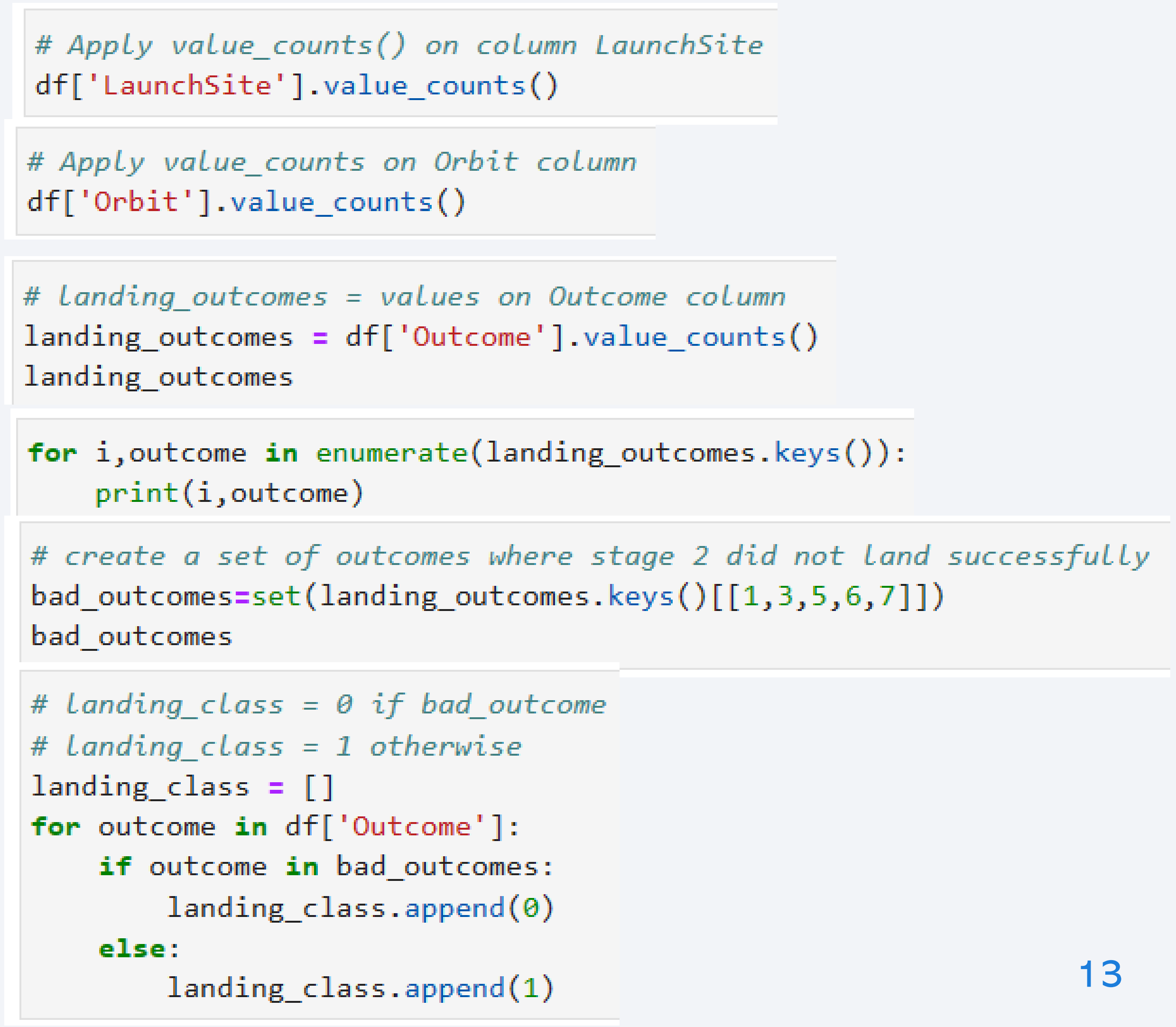
Data Wrangling

Calculate the number of launches at each site

Calculate the number and occurrence of each orbit

Calculate the number and occurrence of mission outcome per orbit type

Create a landing outcome label from the outcome column



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

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

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

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
# create a set of outcomes where stage 2 did not land successfully
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

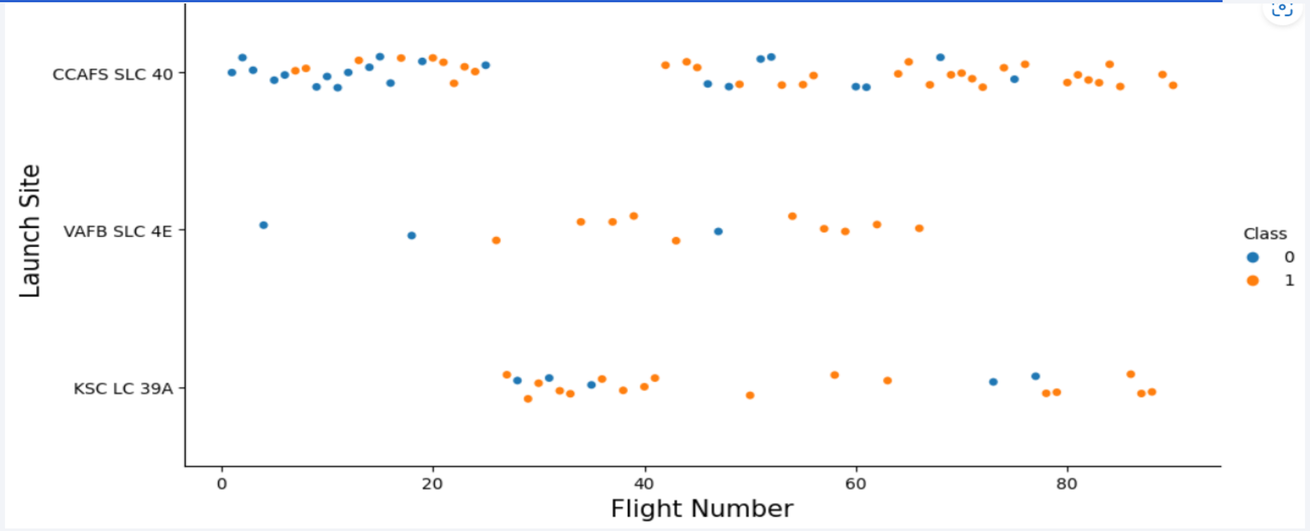
[Link to code]

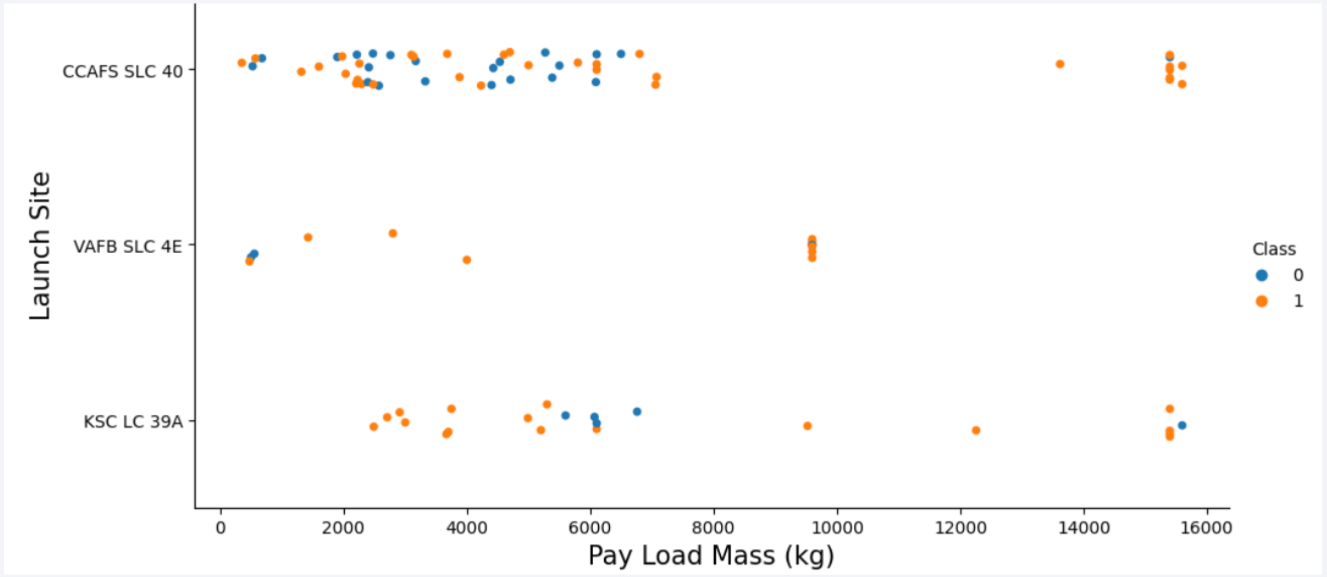
EDA with Data Visualization part 1

- Data visualization with graphs allows the relationship between different variables, and different types of variable, to be assessed quickly, in order to focus further work such as machine learning on where it might produce the most useful information.
- Three types of graph were used to carry out exploratory data analysis by visualizing the data:
  - Scatter graphs show the dependency of different attributes on each other so are a useful way to explore possible factors that affect the success of the landing outcomes.
  - Bar graphs show the relationship between categorical and numerical variables
  - Line graphs are used to show trends over time

EDA with Data Visualization part 2

- Initially, the relationship between the following variables was investigated using scatter graphs:
  - Payload mass and flight number
  - Flight number and launch site
  - Payload mass and launch site
  - Flight number and orbit type
  - Payload mass and orbit type
- In these plots, 0 (blue) is a fail and 1 (orange) is a success



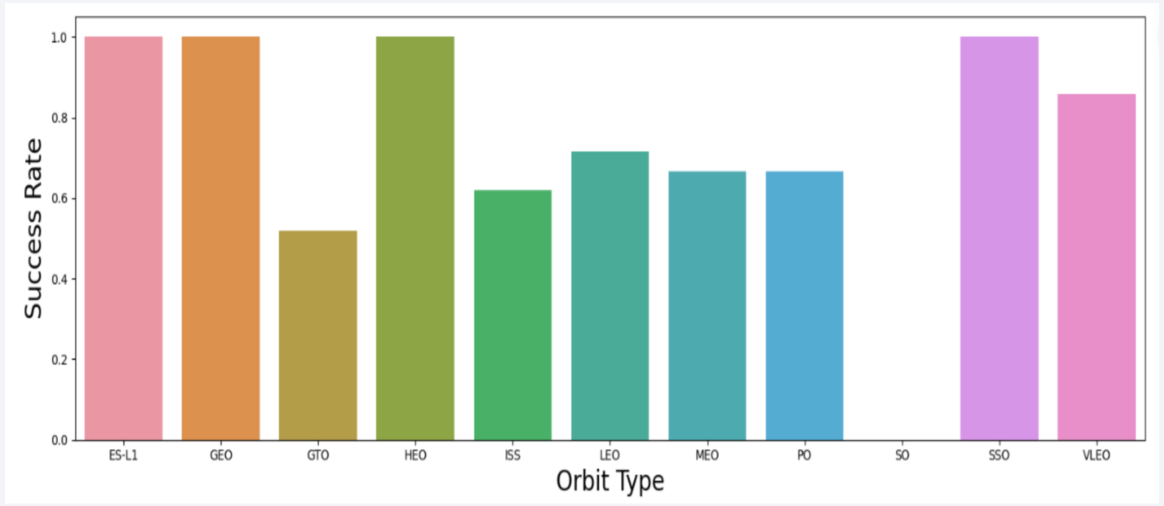


Link to code

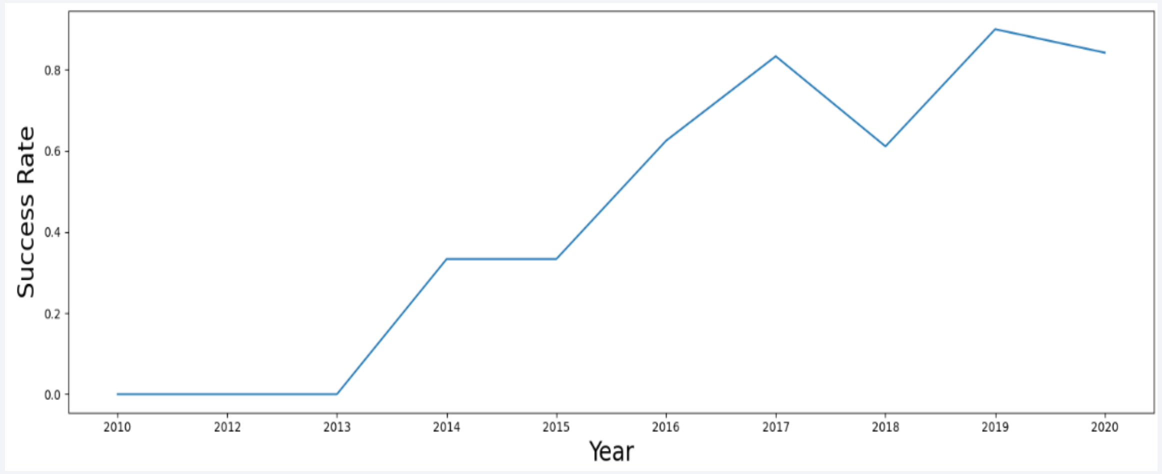
EDA with Data Visualization part 3

- The scatter plots provide useful information about where there are relationships between variables. These can the be further explored using bar charts and line graphs.

Success rate of each orbit type



Success rate over time



Link to code

EDA with SQL

- We performed SQL queries to explore and understand the following data:
- The names of the unique launch sites in the space mission.
- 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 date when the first successful landing outcome in ground pad was achieved.
- The names of the boosters which have had success with a drone ship and have payload mass greater than 4000 but less than 6000.
- The total number of successful and failure mission outcomes.
- The names of the booster versions which have carried the maximum payload mass.
- Records which display the month names, faiilure landing outcomes with the drone ship, booster versions, and launchsite for the months in year 2015.
- Rank the count of successful landiing_outcomes between the date 04-06-2010 and 20-03-2017in descending order.

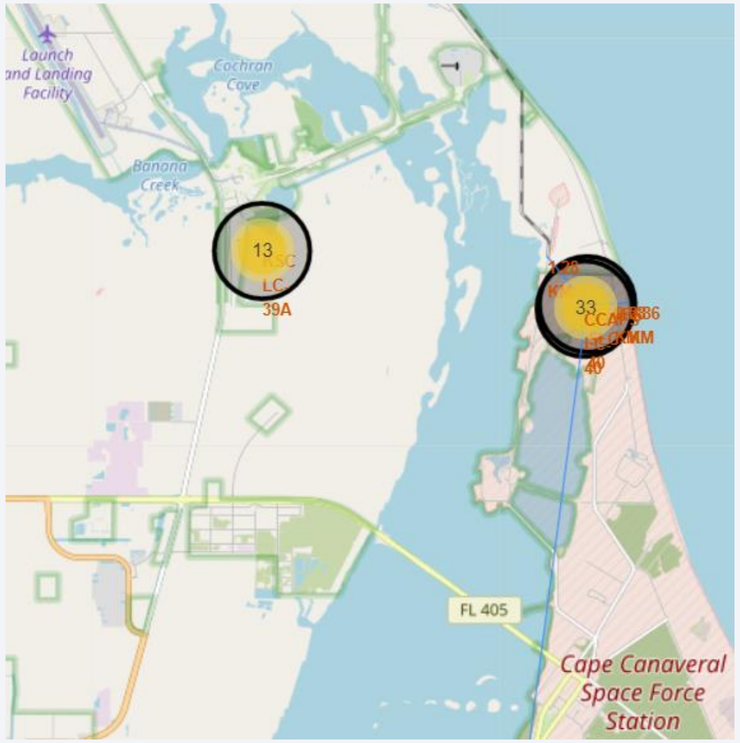
Link to code

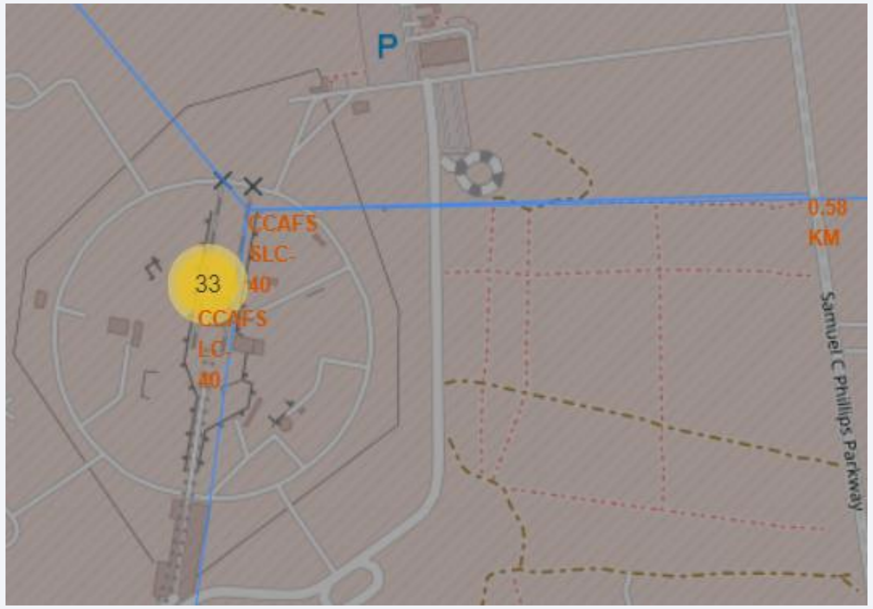
Build an Interactive Map with Folium

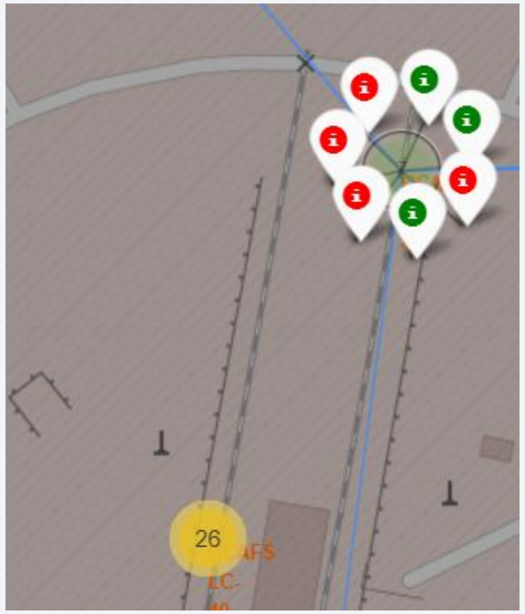
- The data were next visualized by building an interactive geographical map using Folium, a Python library for map building.
- The latitude and longitude for each site were added using a circle marker, labelled with the name of the site.
- We then used the designation 0 for a failed outcome and marked it in red, and 1 for a successful outcome and marked it in green
- Haversine's formula was used to calculate the distance of the launch sites to landmarks such as railways, highways, coastlines and nearby cities.
- Explain why you added those objects
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

Build an Interactive Map with Folium

- Selected screenshots of parts of the interactive map where the next image is obtained by clicking on a circle marker:


Launch and Landing Facility
Cochran Cove
Banana Creek
13
KSC LC 39A
33
CCAFS
FL 405
Cape Canaveral Space Force Station

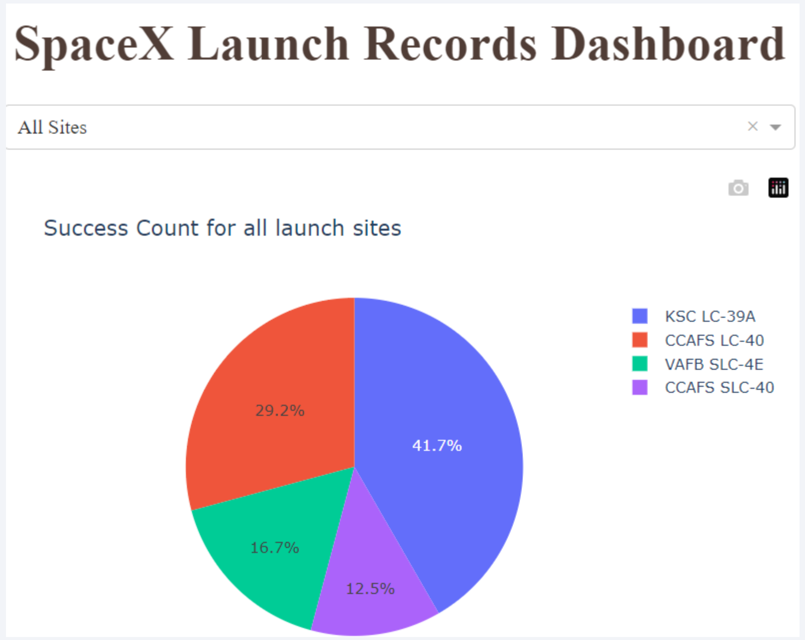


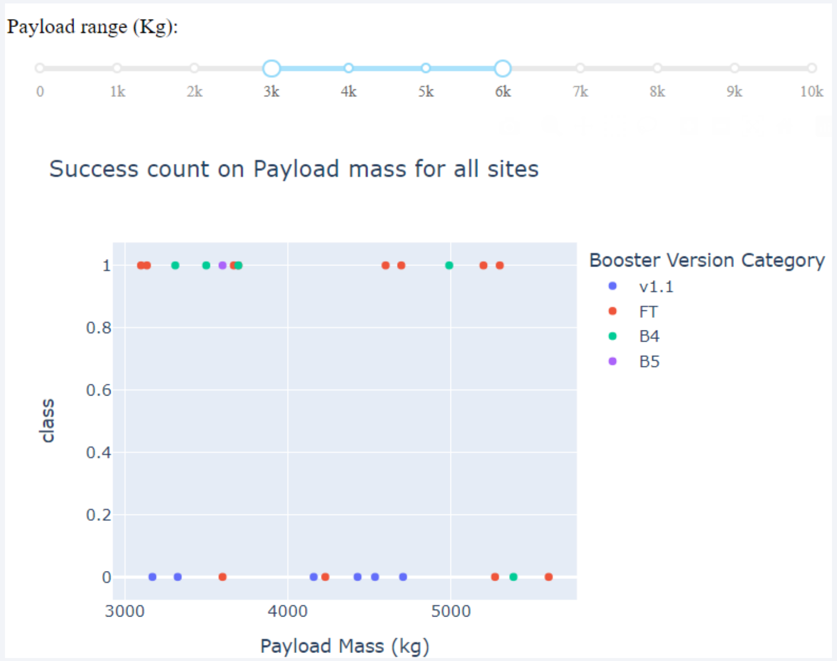


Link to code

Build a Dashboard with Plotly Dash

- We built an interactive dashboard using Plotly Dash. This allows the user to “play” with the data in an interactive way.
- We used pie charts to show the number of successful and failed launches at each site.
- We plotted scatter graphs which show the relationship between outcome and payload for the different booster versions.





[Link to code]()

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- You need present your model development process using key phrases and flowchart
- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

Build the model

- Load the data, which has been cleaned in previous steps of the project
- Transform the data and split into test and train datasets
- Choose a ML algorithm to use
- Set the parameters of the algorithm

Evaluate the model

- Check accuracy
- Choose best hyperparameters for each model type
- Plot a confusion matrix

Improve the model

- Use feature engineering and algorithm tuning

Find the best model

- The model with the best accuracy score is the best model

Link to code

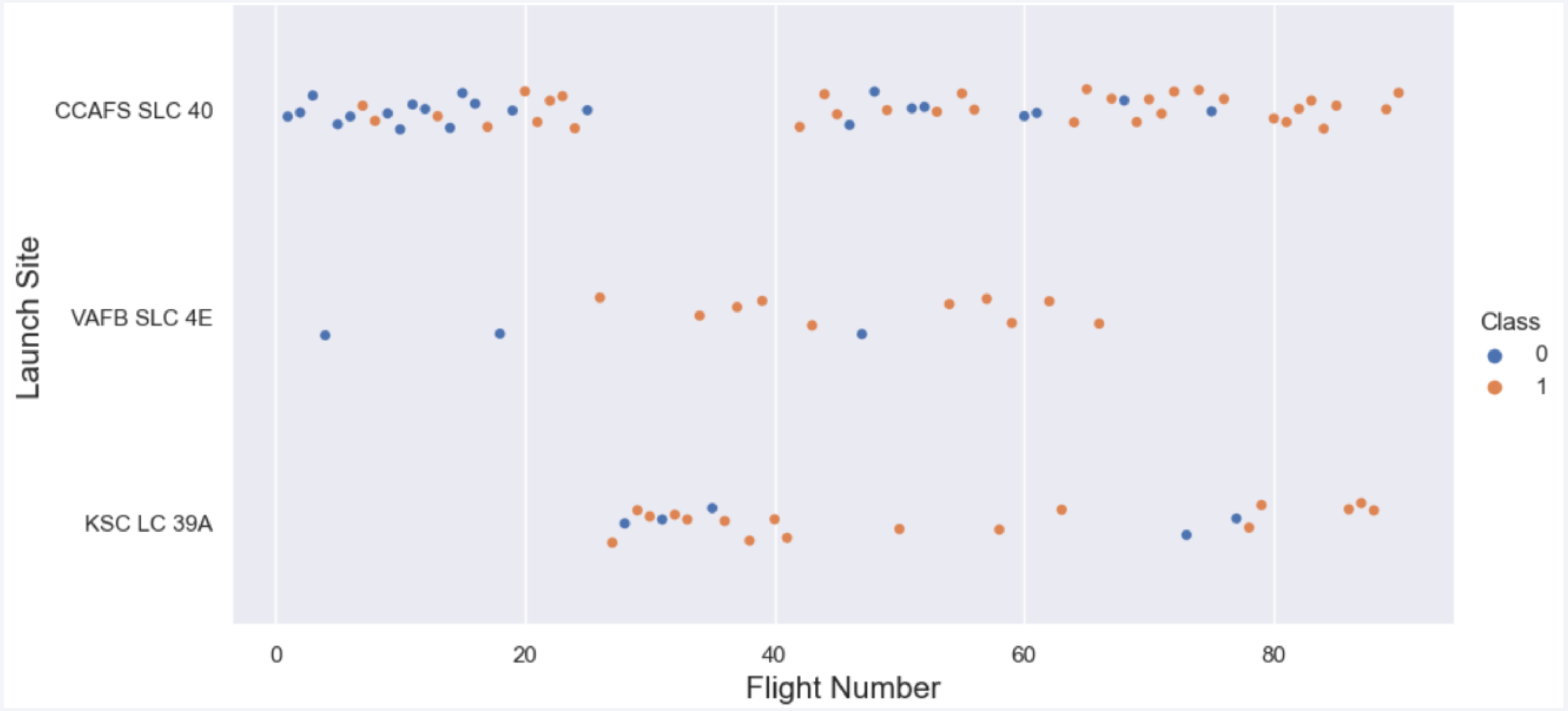
Results

- The results are presented in three sections:
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Section 2

Insights drawn from EDA

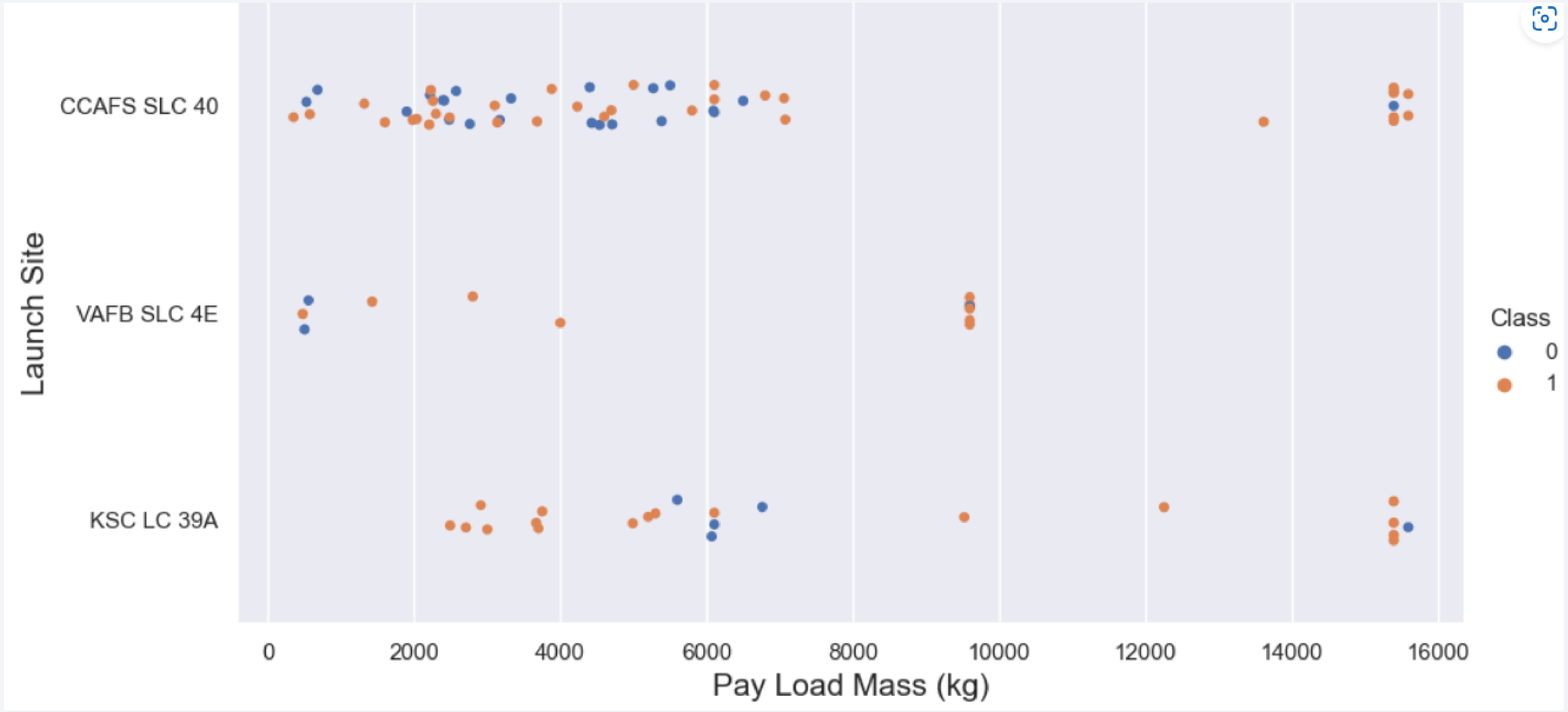
Flight Number vs. Launch Site



0 = failure, 1 = success

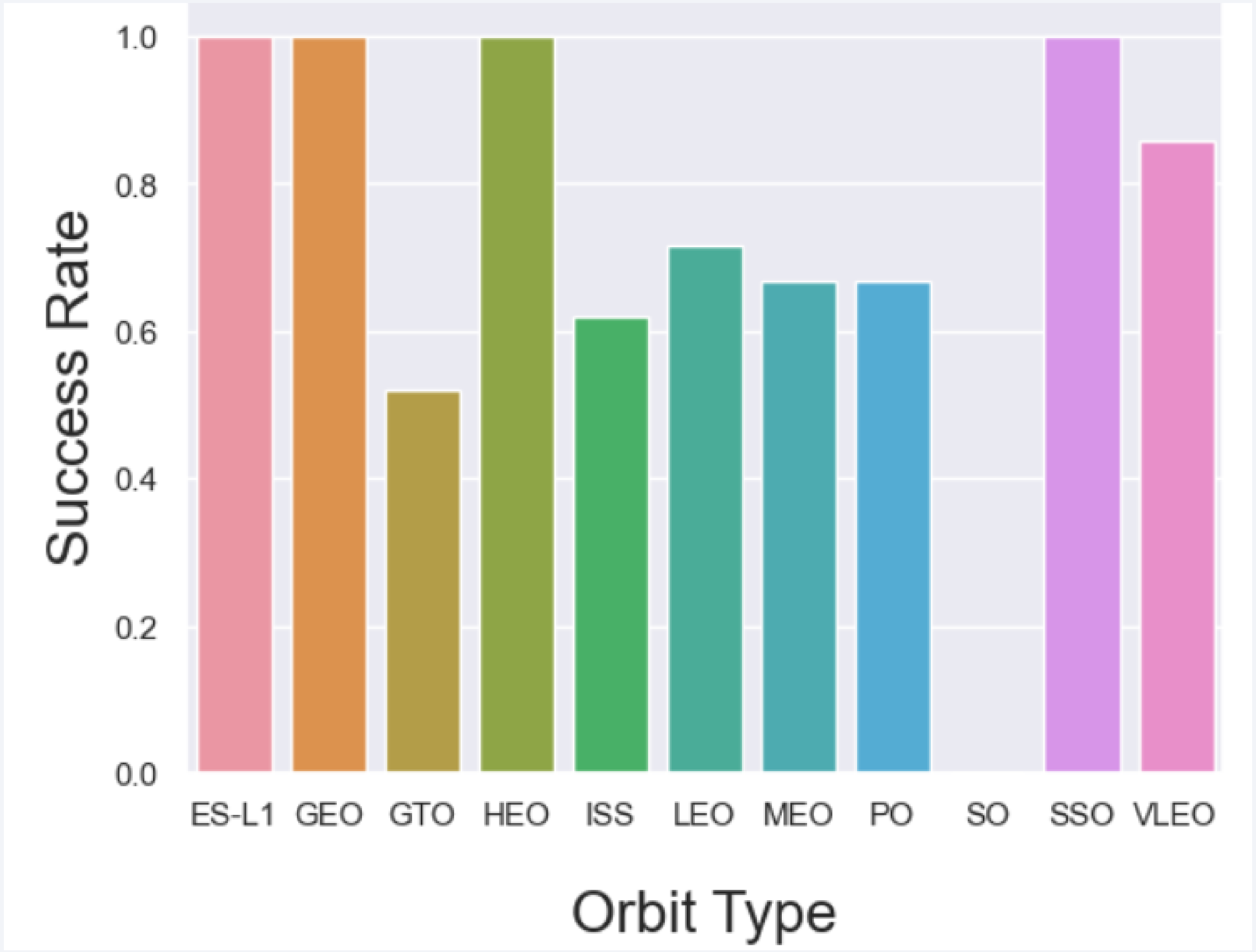
At each site, success is increasing over time

Payload vs. Launch Site



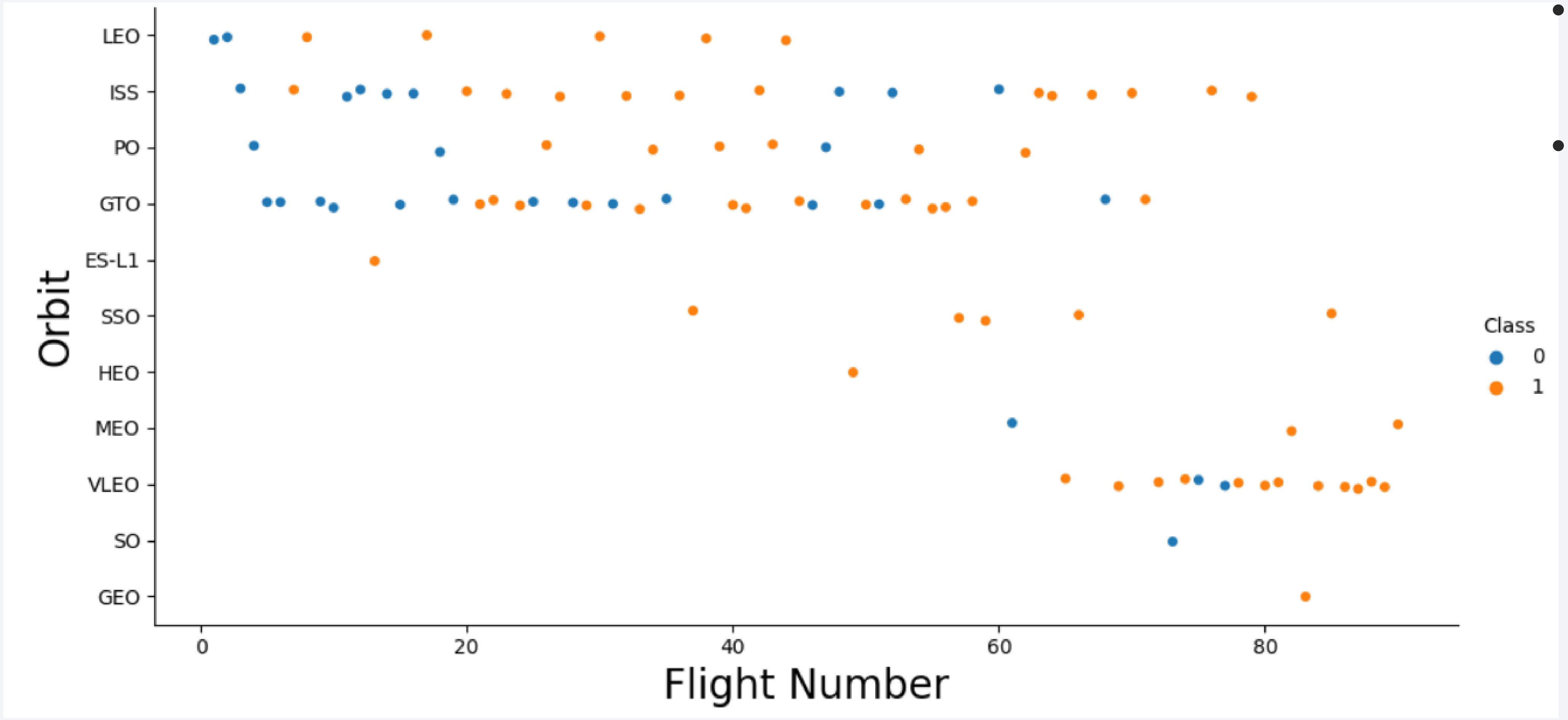
- 0 = failure, 1 = success
- These results suggest that higher payloads have a greater success rate

Success Rate vs. Orbit Type



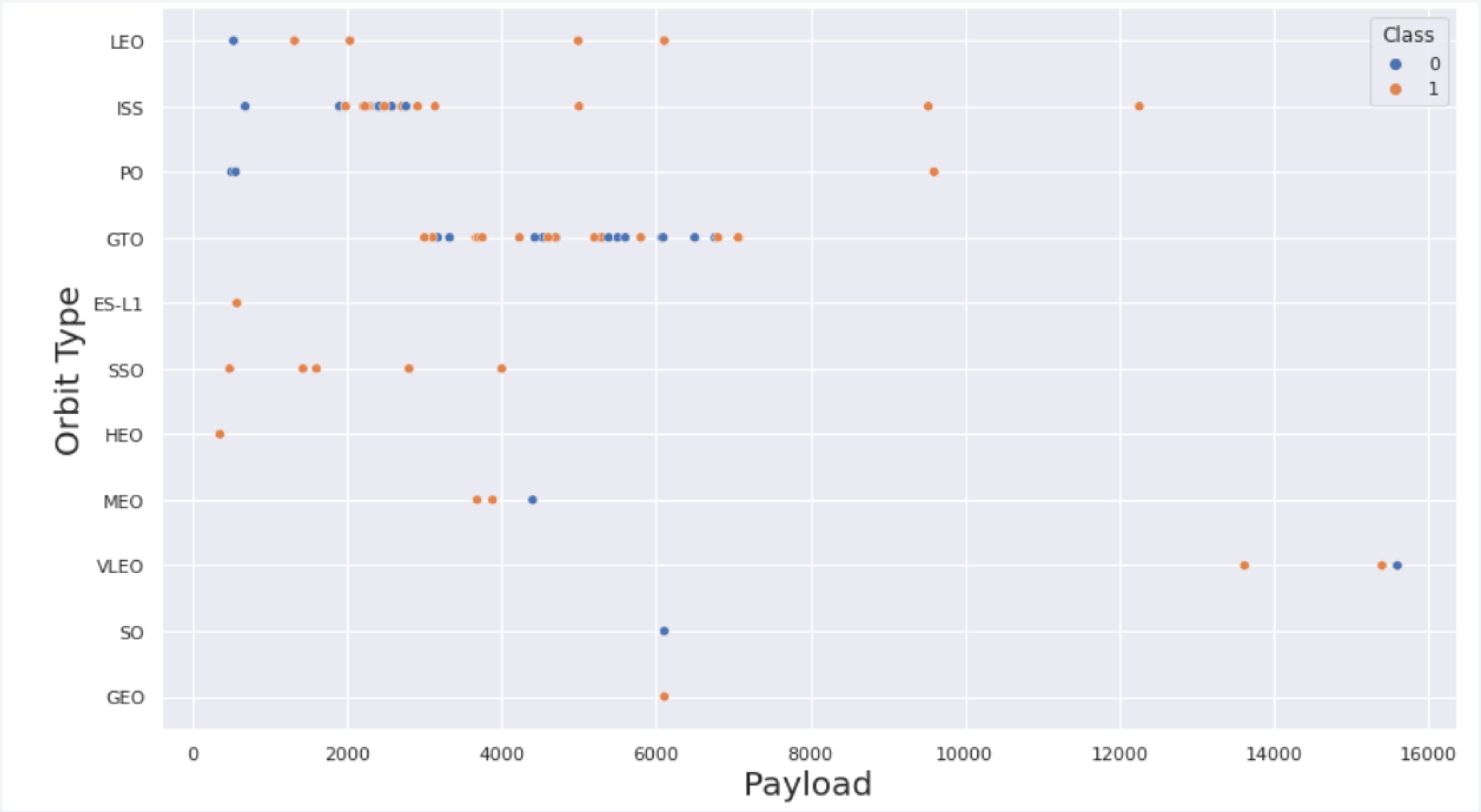
- ES-L1, GEO, HEO and SSO have a 100% success rate
- However, ES-L1, GEO SO and HEO have only one launch each, so these results should be treated with caution

Flight Number vs. Orbit Type



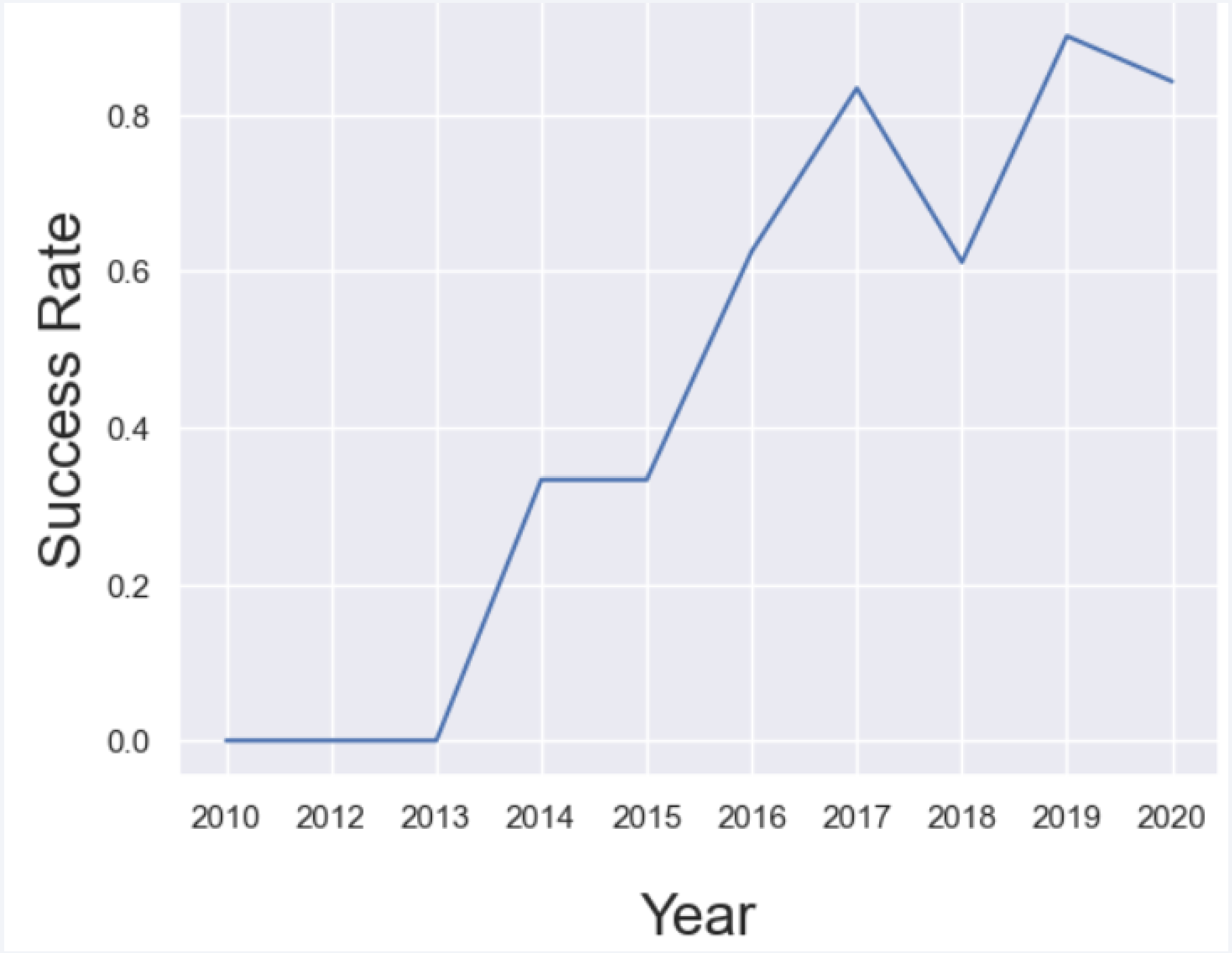
- 0 = failure, 1 = success
- Success rate has increased over time for all orbit types, with the exception of GTO

Payload vs. Orbit Type



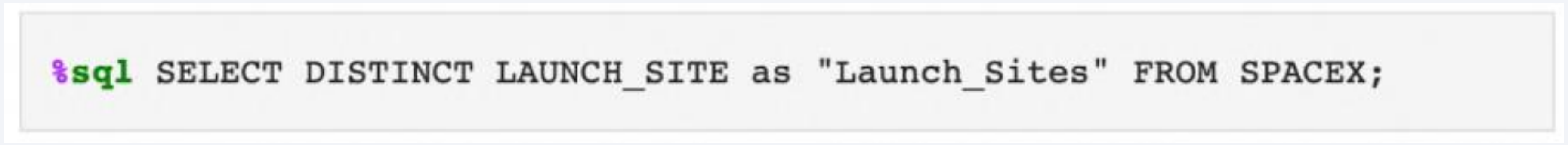
- 0 = failure, 1 = success
- Heavier payloads are associated with greater success rate for LEO, ISS and PO

Launch Success Yearly Trend

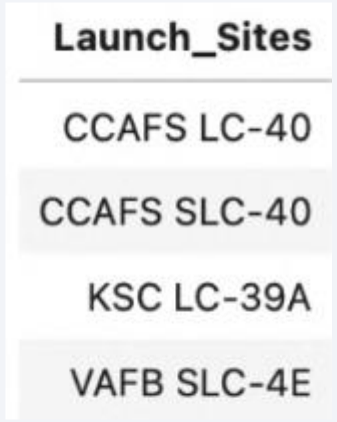


- Success rate rose steadily from 2013 to 2017.
- A dip in 2018 was followed by another rise, with a slight reduction in 2020

All Launch Site Names



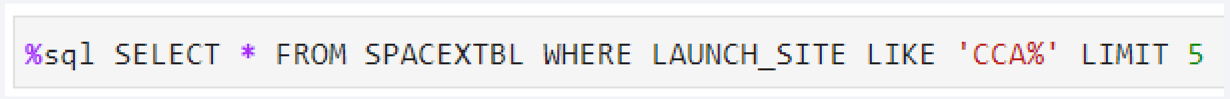
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```



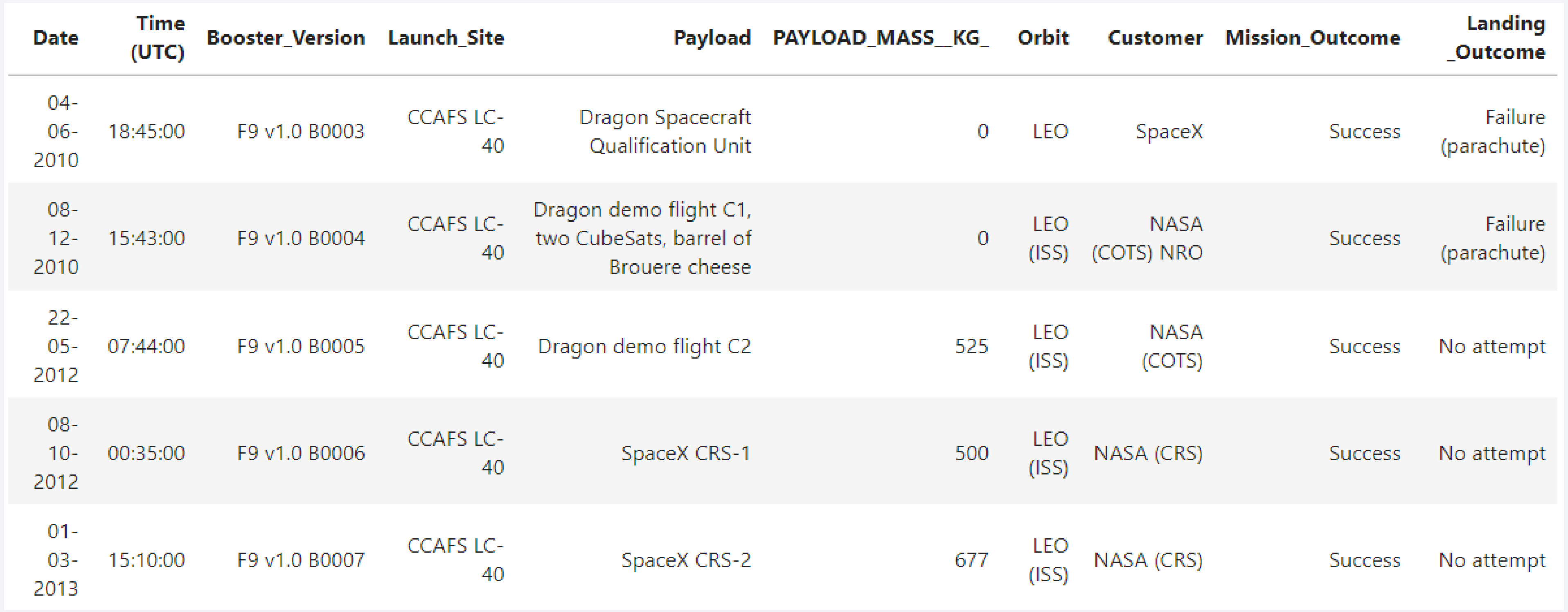
Launch_Sites
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

- This SQL query was used to find launch site names.
- The keyword “DISTINCT“ with the parameter “FROM SPACEX” restricted this to SpaceX launches only.

Launch Site Names Begin with 'CCA'



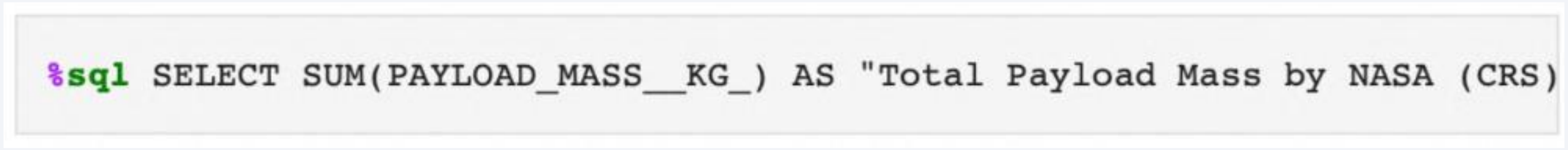
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```



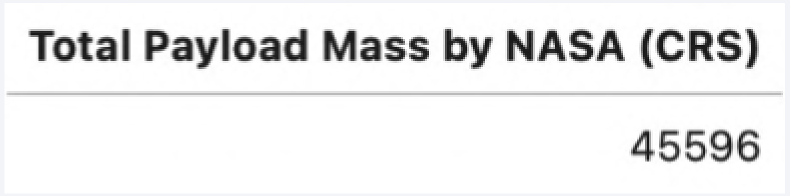
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- The WHERE clause followed by a LIKE clause filters launch sites containing the substring CCA. LIMIT 5 shows 5 records.

Total Payload Mass



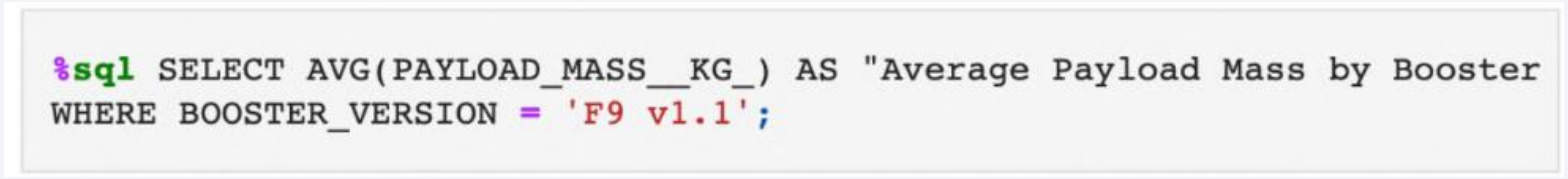
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)
```



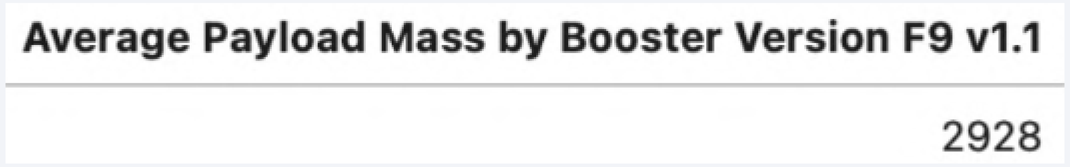
Total Payload Mass by NASA (CRS)
45596

- This query calculates the total payload mass carried by NASA

Average Payload Mass by F9 v1.1



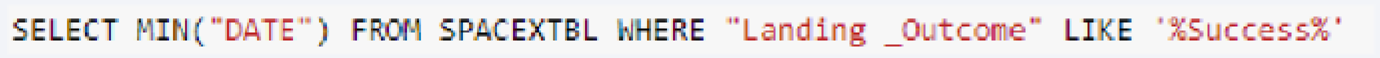
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster
WHERE BOOSTER_VERSION = 'F9 v1.1';
```



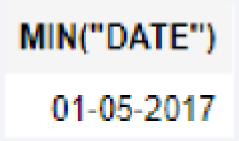
Average Payload Mass by Booster Version F9 v1.1
2928

- The average payload mass by booster for F9 v1.1 was 2928

First Successful Ground Landing Date



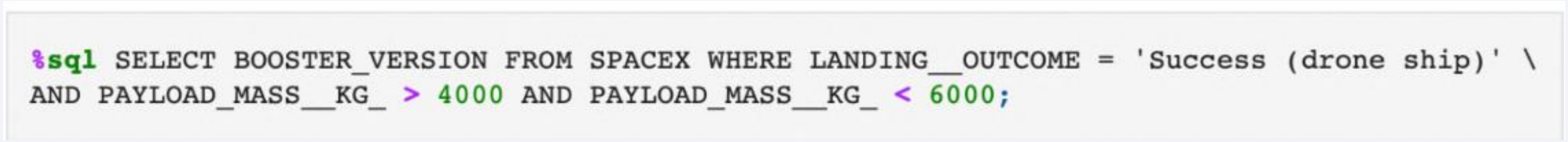
```
SELECT MIN("DATE") FROM SPACEXTBL WHERE "Landing _Outcome" LIKE '%Success%'
```



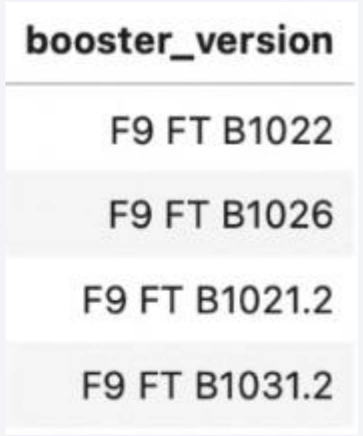
MIN("DATE")
01-05-2017

- The is query returns the earliest date in the table that was successful
- The date was 01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000



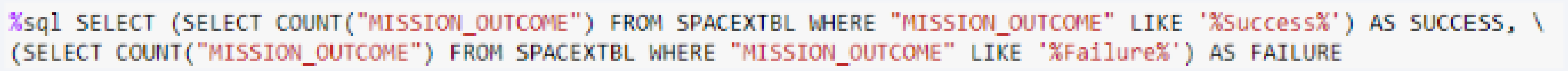
```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```



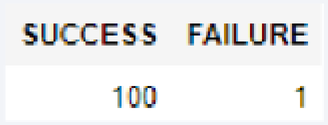
booster_version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

- This query filters the results for boosters that have successfully landed on the drone ship with a payload between 4000 and 6000 kg

Total Number of Successful and Failure Mission Outcomes



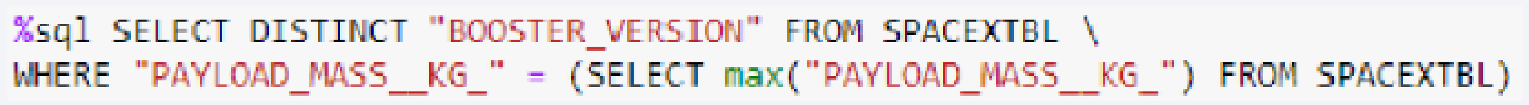
```
%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS SUCCESS, \
(SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE
```



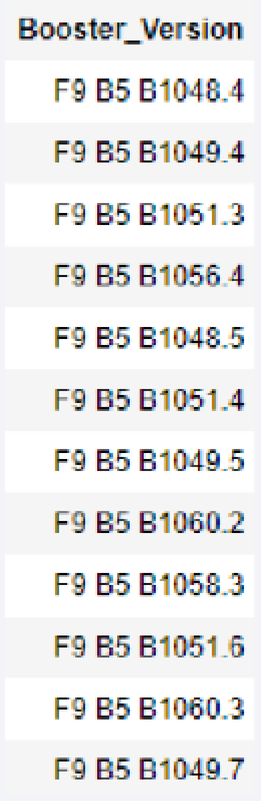
SUCCESS	FAILURE
100	1

- The first of the SELECT statements shows subqueries that return results.
- The first subquery calculates successful missions, the second unsuccessful ones.

Boosters Carried Maximum Payload



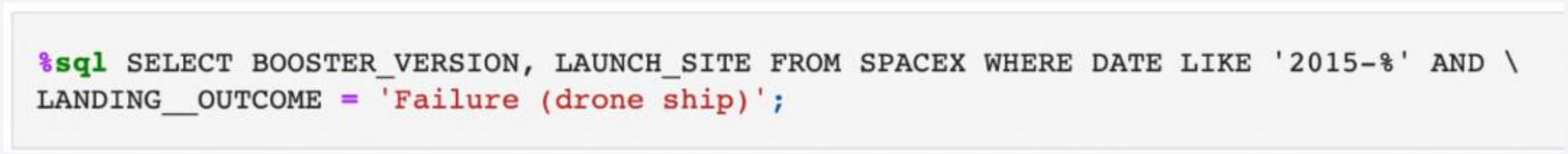
```
%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTBL \
WHERE "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```



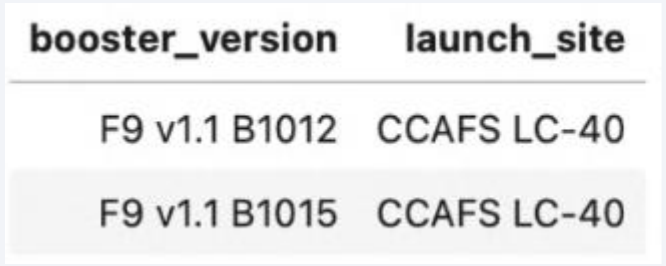
Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

- This is the use of a subquery to filter data only having the heaviest (MAX) payload.

2015 Launch Records



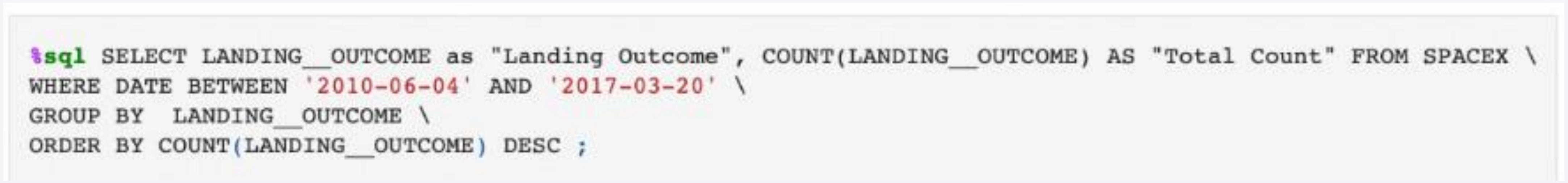
```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```



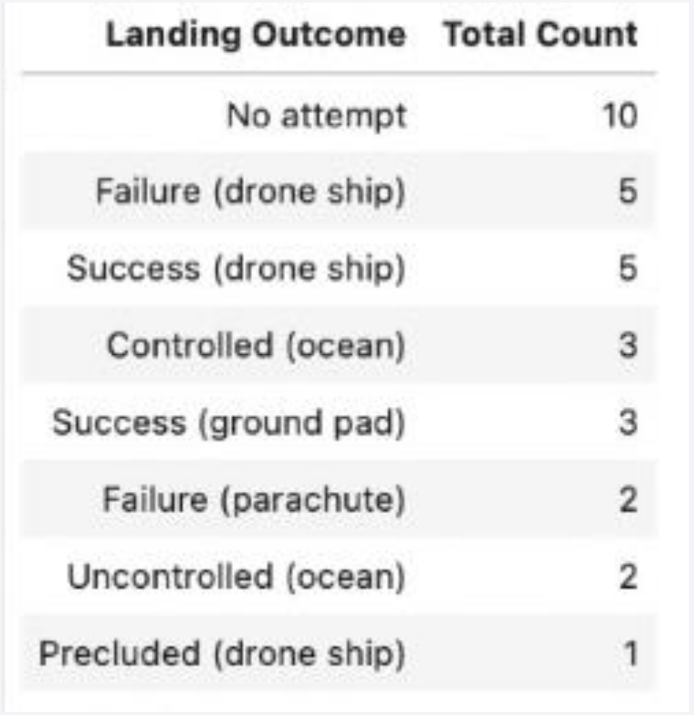
booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

- This query uses WHERE, LIKE and AND to filter for unsuccessful landings in 2015.

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



```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```



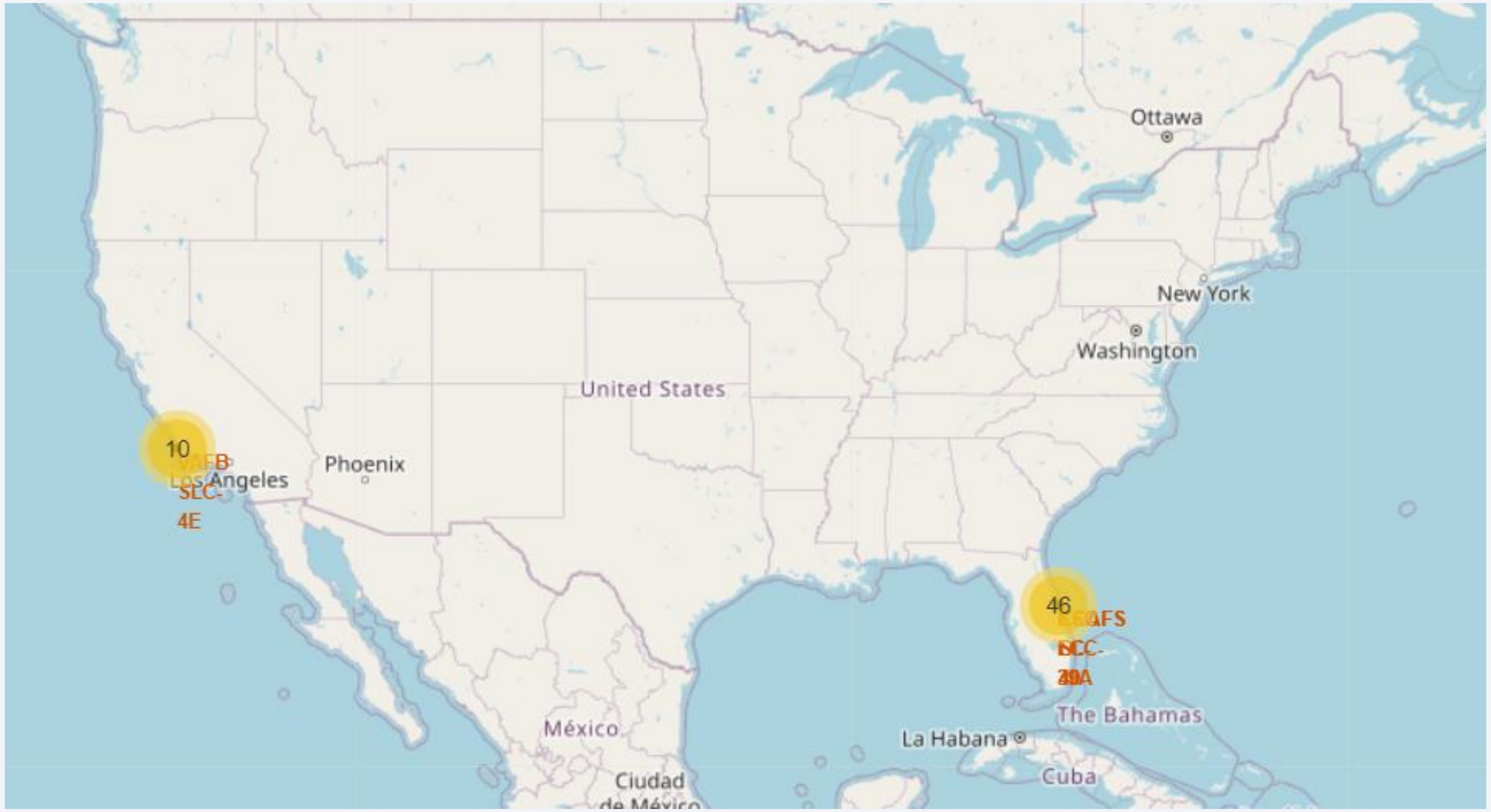
Landing Outcome	Total Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

- This query uses COUNT to count the number of outcome types between the two dates. ORDER BY (LANDING_OUTCOME) DESC sorts the GROUP BY landing outcomes into descending order

Section 3

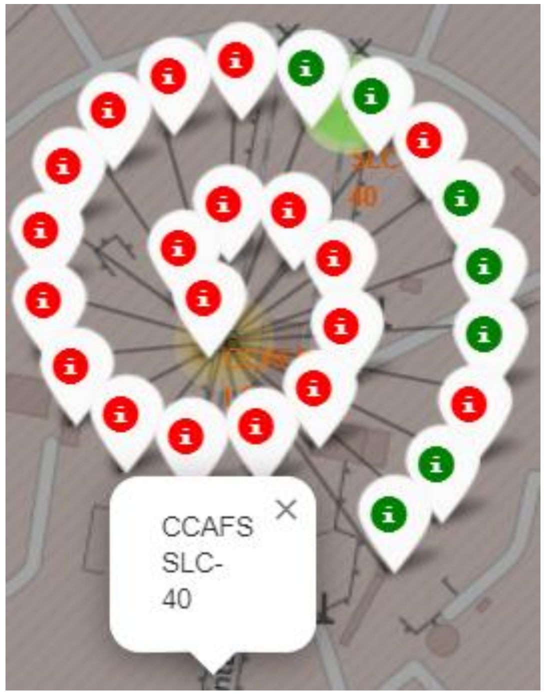
Launch Sites Proximities Analysis

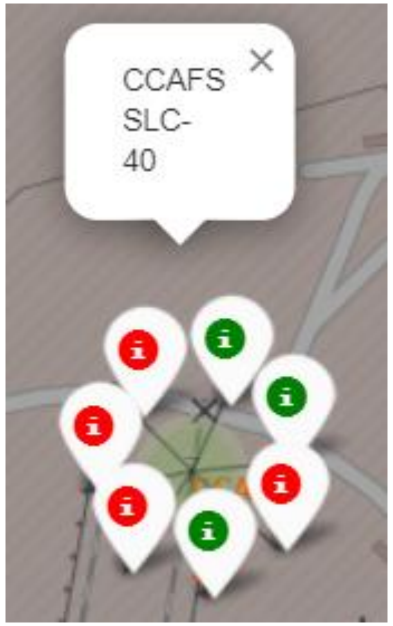
Folium Map: Launch Sites

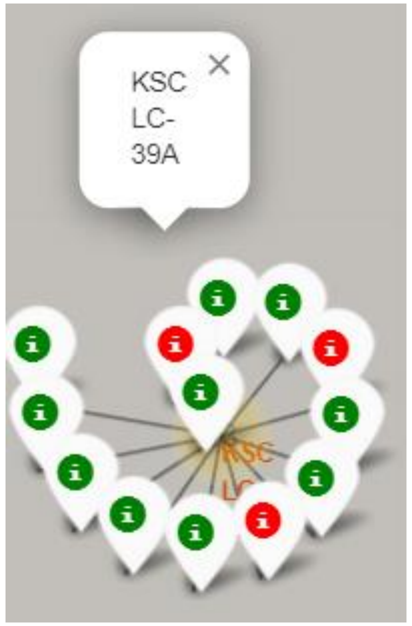


- All the SpaceX launch sites are located on the coast of the USA.
- There are two sites in Florida and one in California

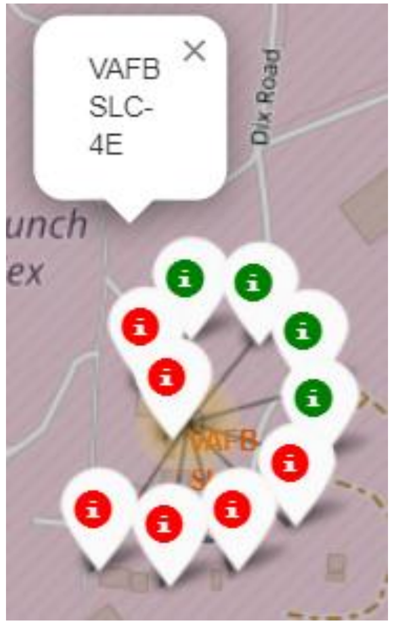
Folium Map: Launch Sites with Success Markers







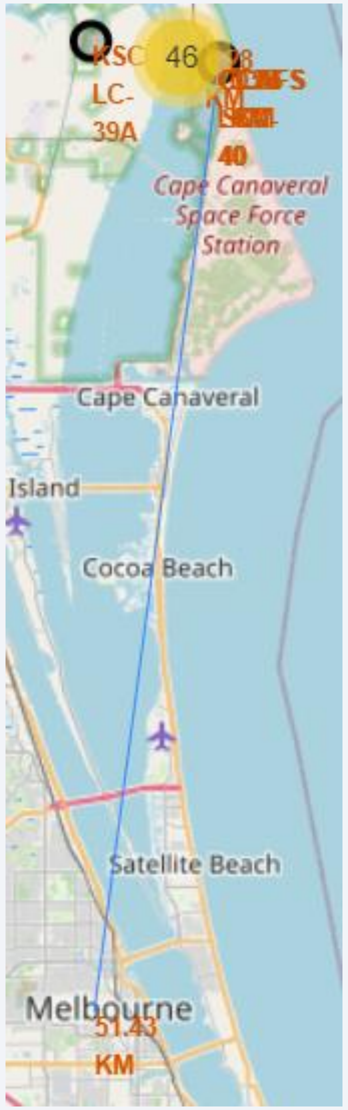
Florida sites

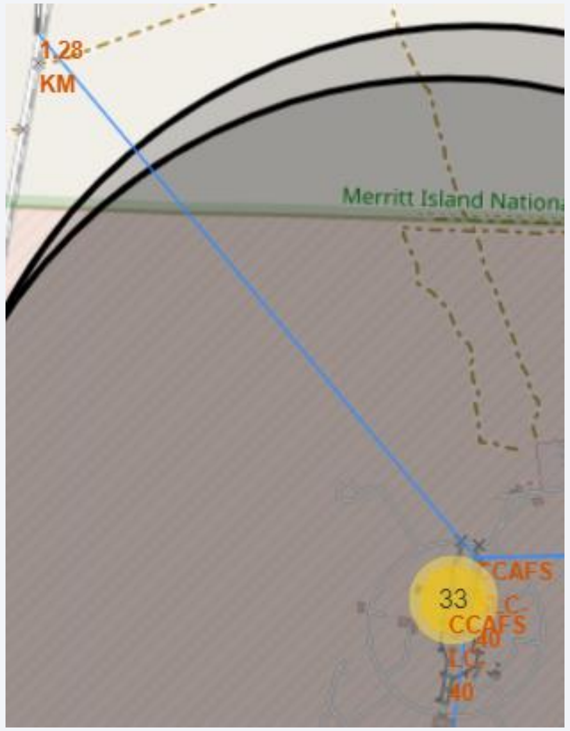


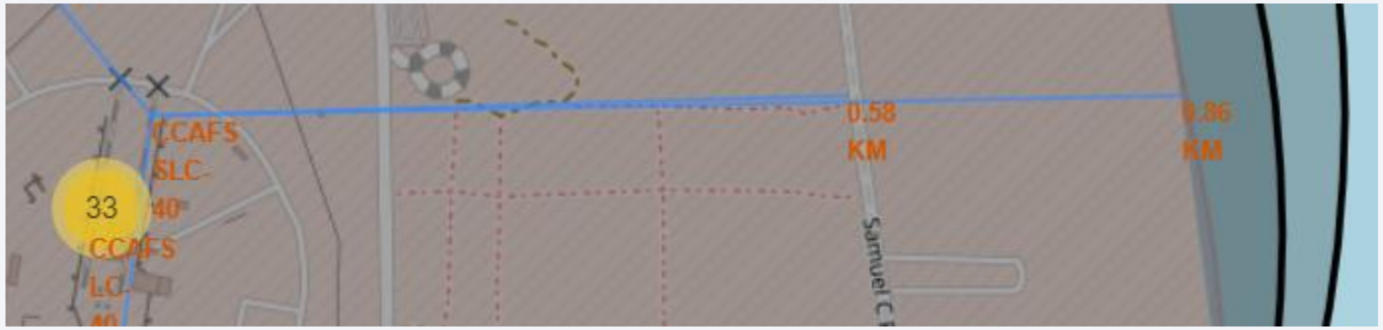
California site

- The green marker represents successful launches; the red marker represents unsuccessful launches

Folium Map: Proximity of CCAFS SLC-40 Site



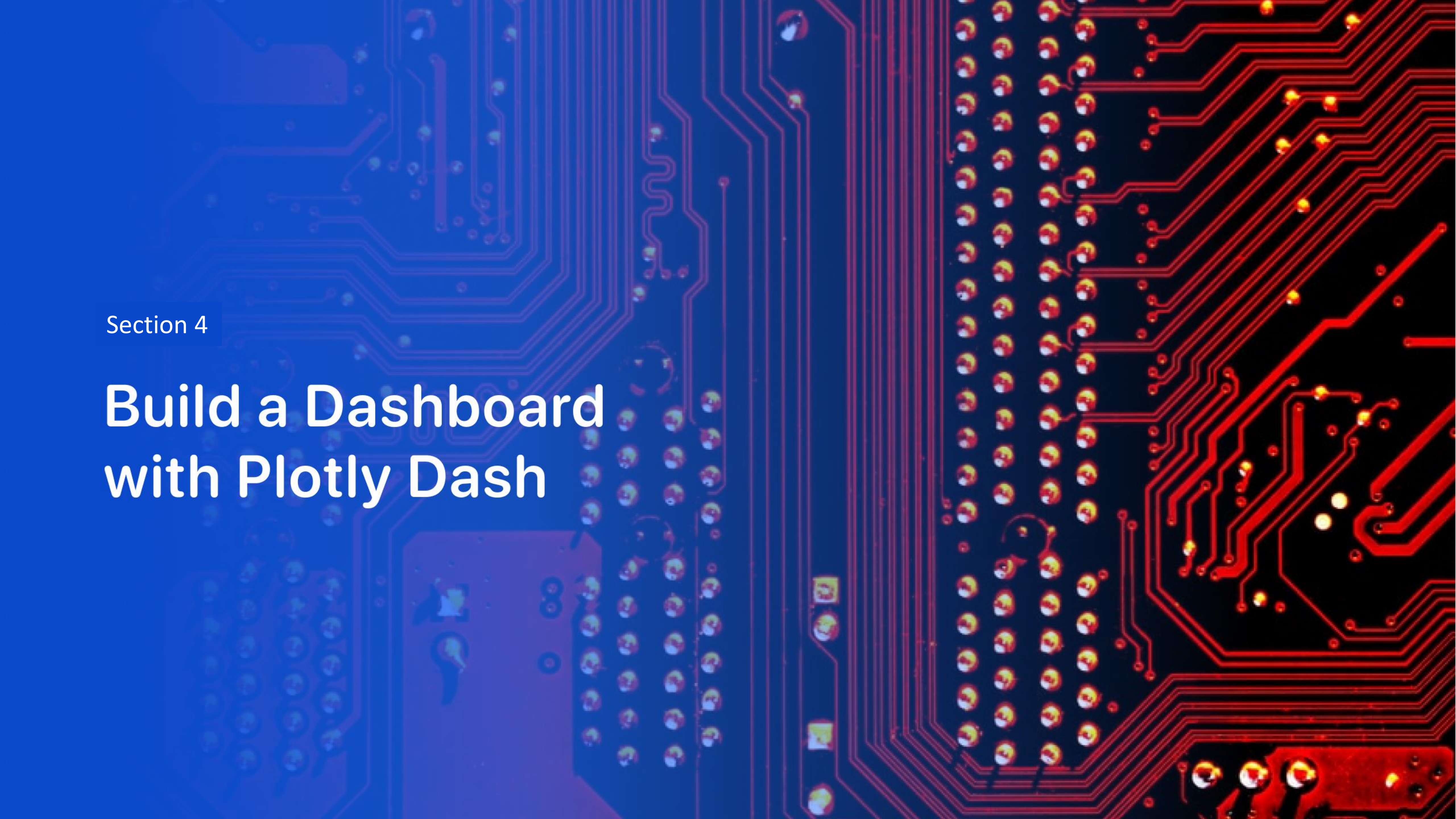




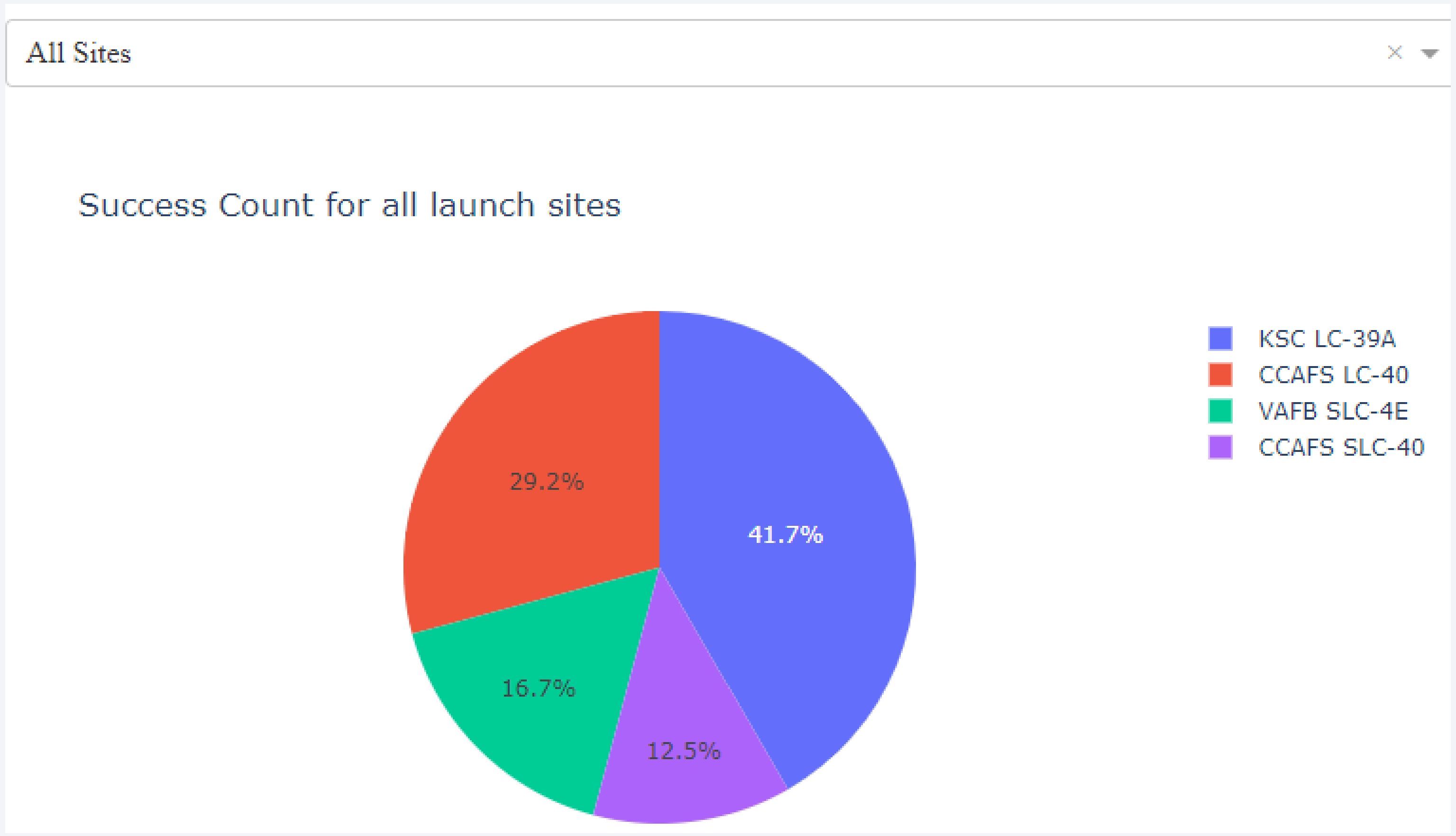
- The Folium map allows calculation of proximity to railways, highways, coastline, cities and other geographical features.
- The CCAFS SLC-40 launch site is 51.43 km from the nearest city (Melbourne) but only 1.28 km from the nearest railway line, and 0.58 km from the nearest highway, and 0.86 km from the nearest coast.

Section 4

Build a Dashboard with Plotly Dash

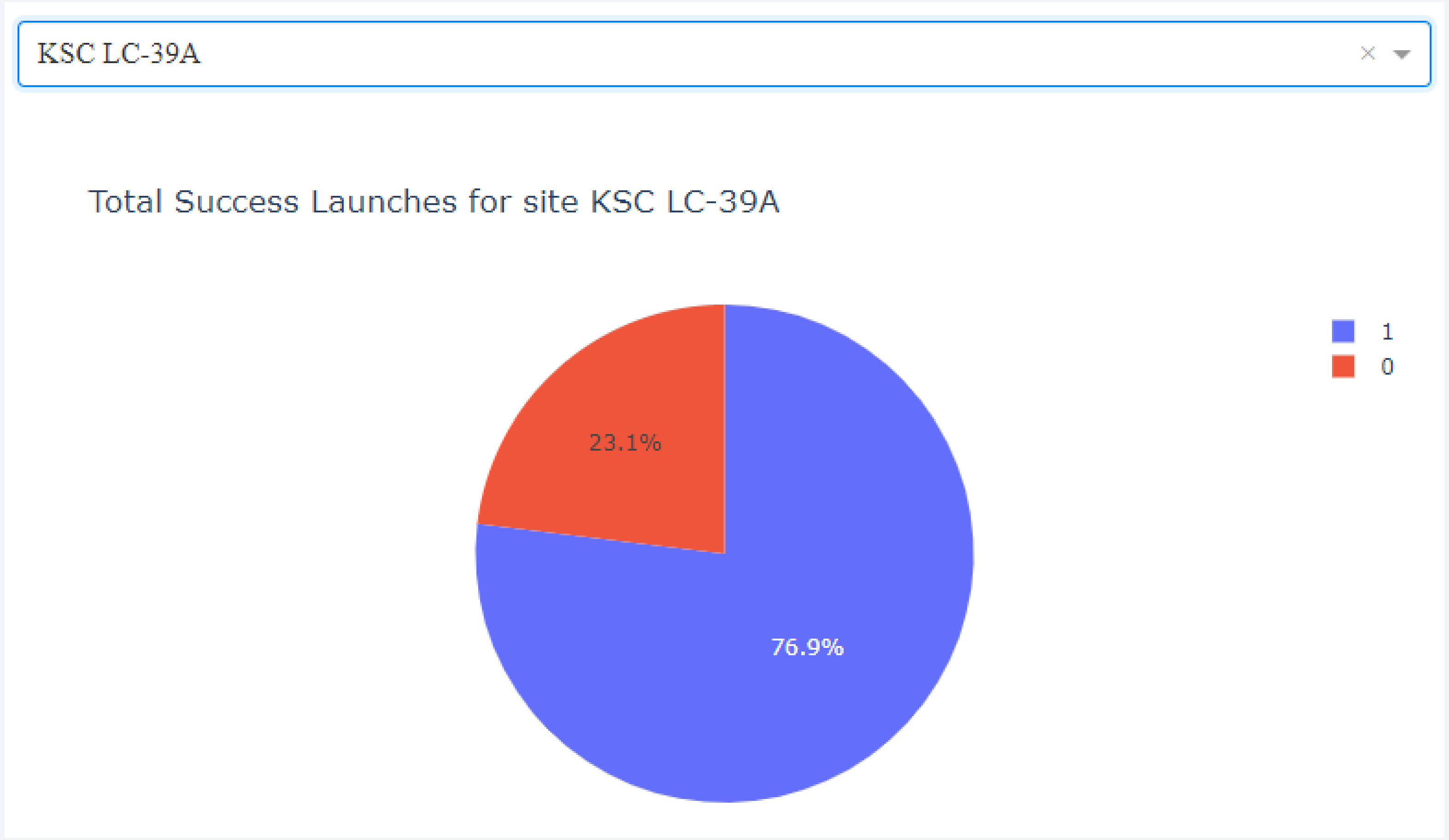


Plotly Dashboard: Successful Launches



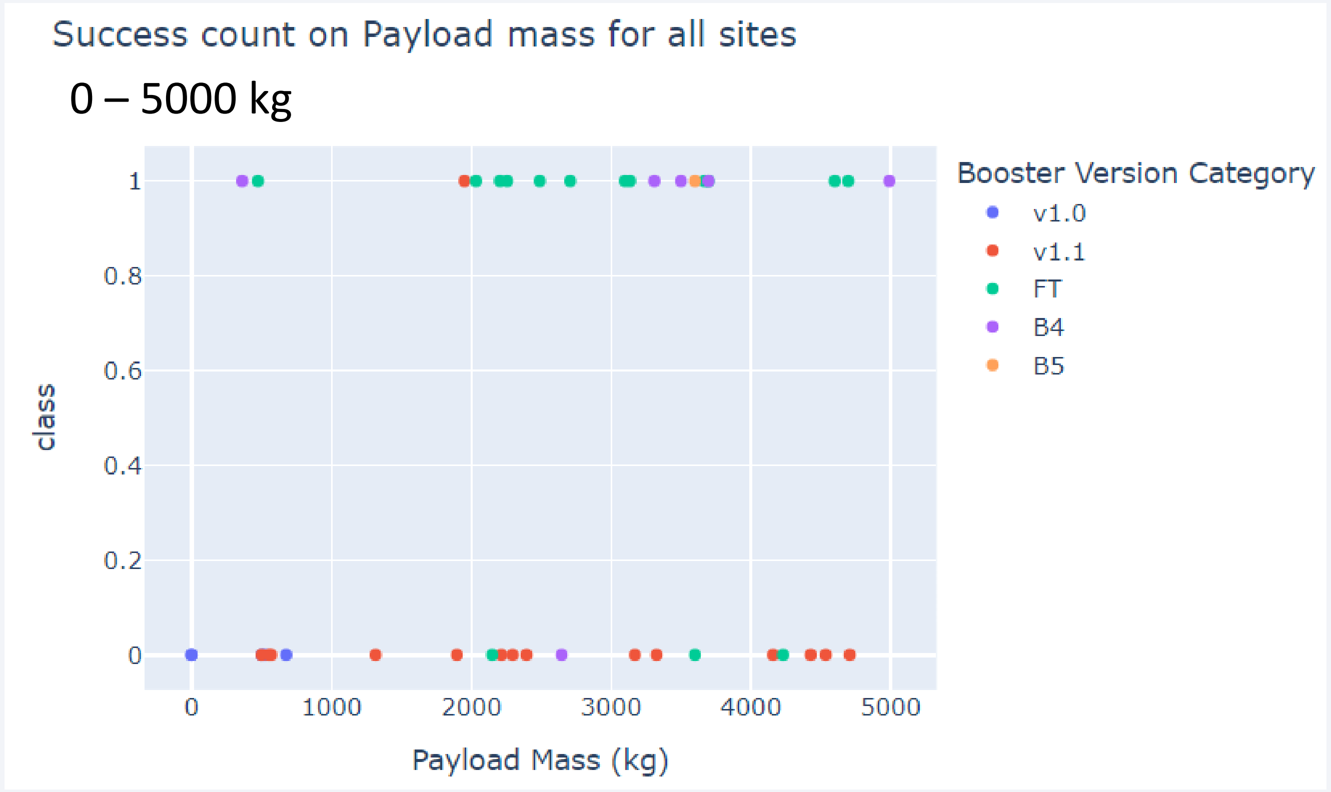
- The dashboard shows that KSC LC39-A had the greatest proportion of successful launches and CCAFS SLC-40 the lowest.

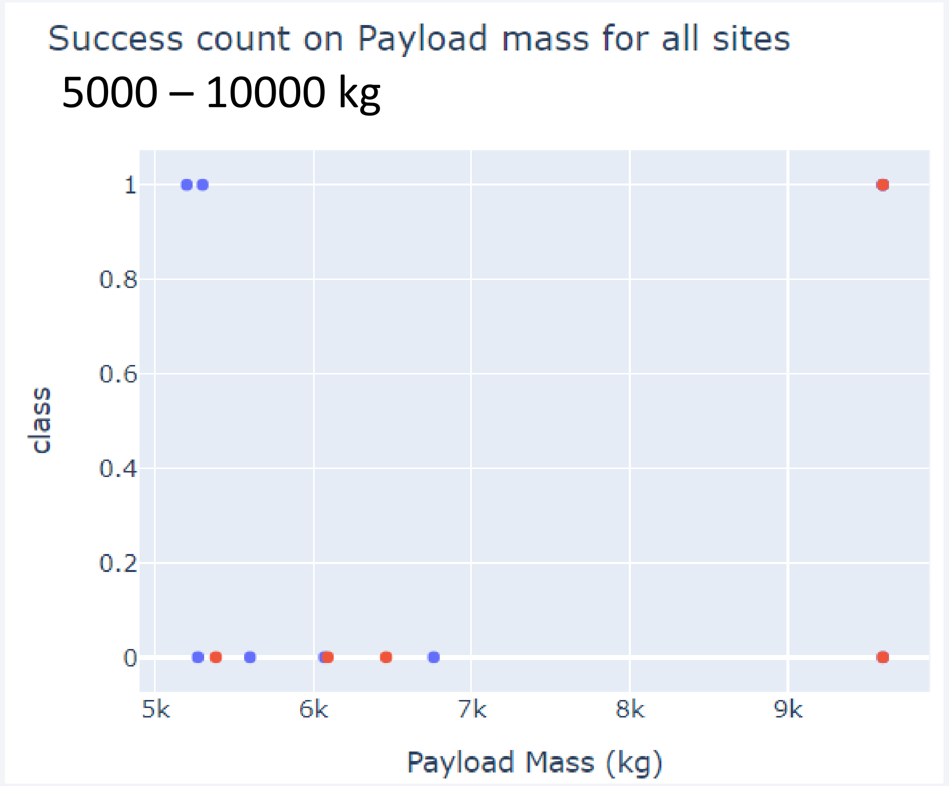
KSC LC-39A Success versus Failure



- KSC LC-39A had a success rate of 76.9% versus a failure rate of 23.1%

Payload versus Outcome Scatter Plots



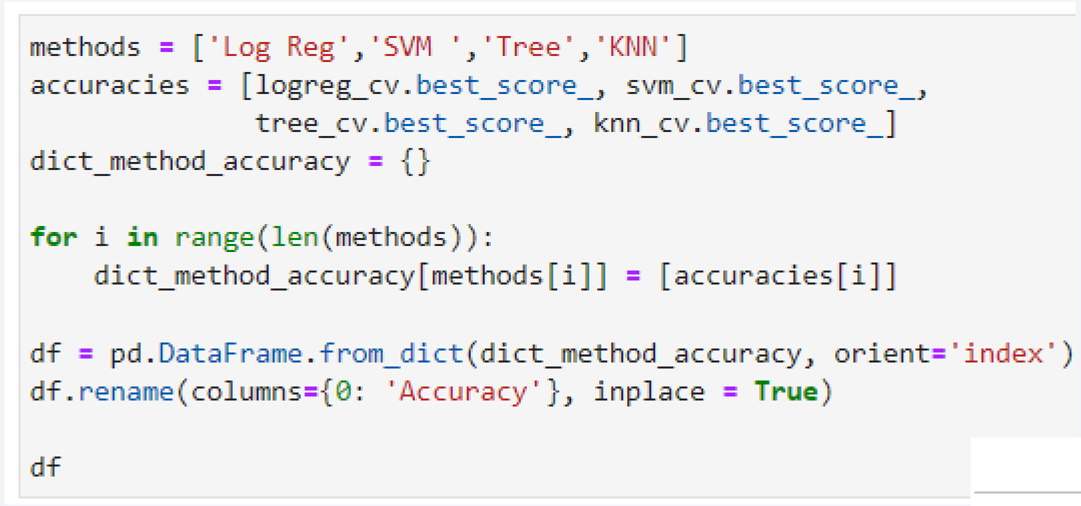


- The success rate for low (< 5000 kg) payloads is greater than for high (> 5000 kg) payloads

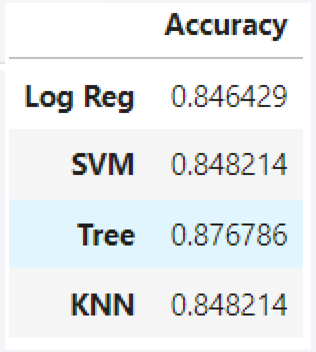
Section 5

Predictive Analysis (Classification)

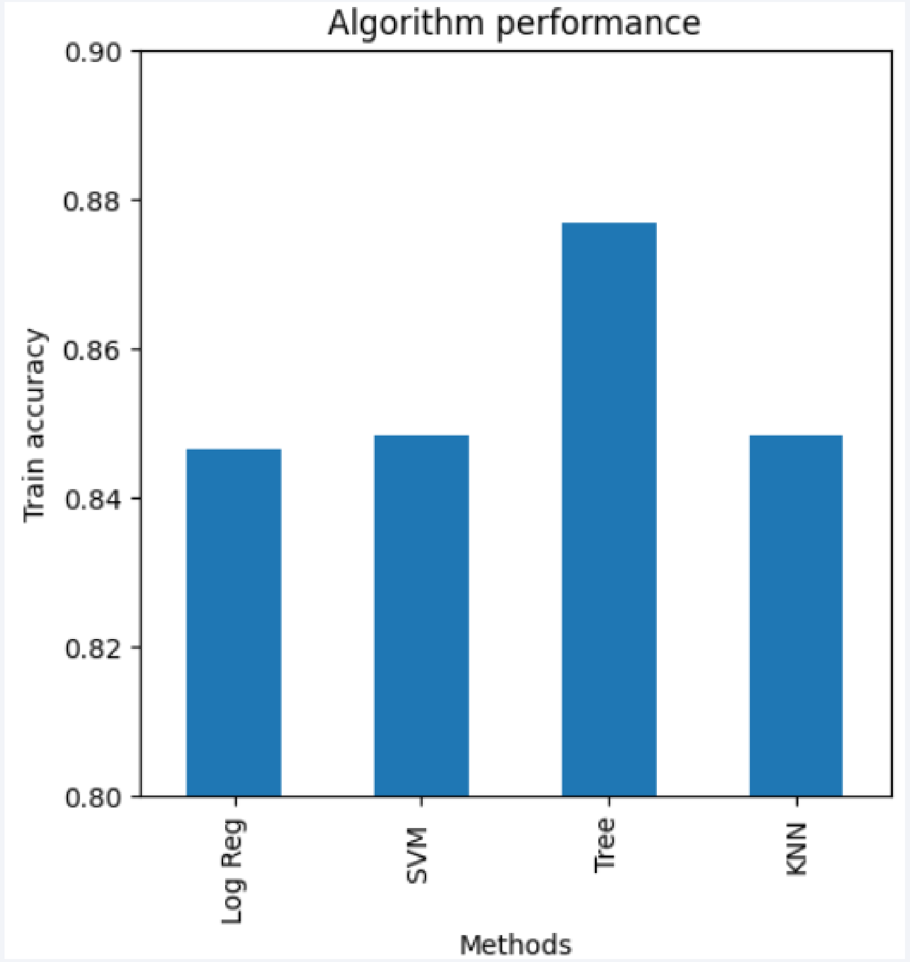
Classification Accuracy



```
methods = ['Log Reg','SVM ','Tree','KNN']
accuracies = [logreg_cv.best_score_, svm_cv.best_score_,
              tree_cv.best_score_, knn_cv.best_score_]
dict_method_accuracy = {}

for i in range(len(methods)):
    dict_method_accuracy[methods[i]] = [accuracies[i]]

df = pd.DataFrame.from_dict(dict_method_accuracy, orient='index')
df.rename(columns={0: 'Accuracy'}, inplace = True)

df
```



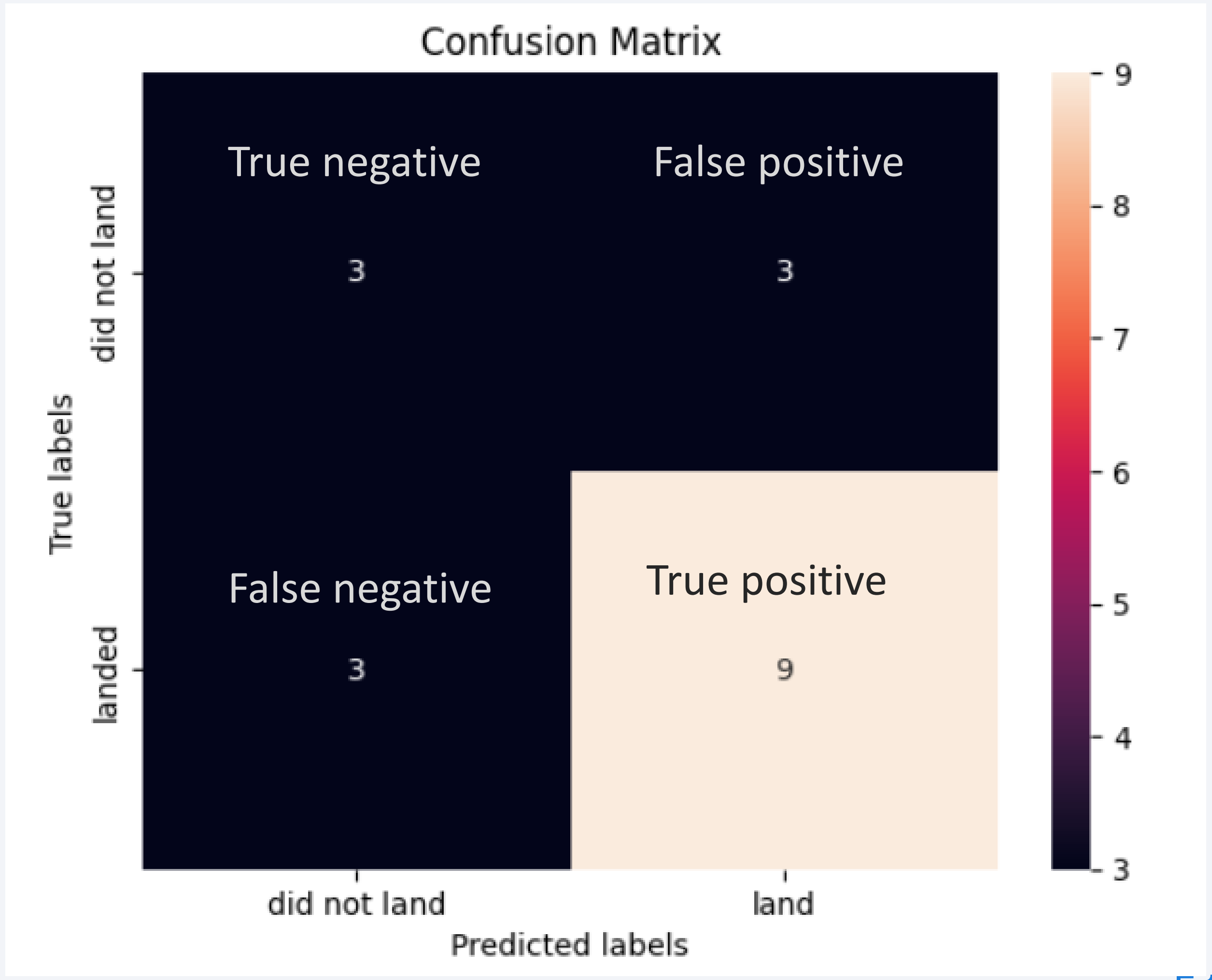
	Accuracy
Log Reg	0.846429
SVM	0.848214
Tree	0.876786
KNN	0.848214



- Using this code we can see that all four algorithms have a similar accuracy. However, the Decision Tree method has the highest accuracy by a narrow margin.

Confusion Matrix

- The confusion matrix visualizes the relationship between true and false negatives and positives.
- The confusion matrices were identical for all algorithms including the most accurate one, the decision tree.
- The main problem here is the false positives, that is, where a failed landing is classified as successful



Conclusions

- Whether a mission will be successful or not depends on factors such as launch site, type of orbit, number of previous launches and mass of the payload.
- The success rate of SpaceX missions has increased from the year 2013
- The orbit types with the highest success rates are GEO, HEO, SSO and ES-L1.
- Lower payloads have a higher success rate than higher payloads.
- KSC LC-39A is the site with the highest proportion of successful missions. To find the reasons for this we would need to consider weather or other atmospheric data.
- The decision tree classfier was the most accurate model at predicting the success or failure of a mission.

Appendix

- All Python code, Jupyter notebooks and datasets for this project can be found on my GitHub site for this project: https://github.com/erinyes1/app_ds_capstone

Thank you!