IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Huy Nguyen Tuan
22 August 2024

Outline

Executive Summary

- Summary of methodologies
  - SpaceX Data Collection using SpaceX API
  - SpaceX Data Collection with Web Scraping
  - SpaceX Data Wrangling
  - SpaceX Exploratory Data Analysis using SQL
  - SpaceX EDA DataViz Using Python Pandas and Matplotlib
  - Space-X Launch Sites Analysis with Folium-Interactive Visual Analytics and Ploty Dash
  - SpaceX Machine Learning Landing Prediction
- Summary of all results
  - EDA results
  - Interactive Visual Analytics and Dashboards
  - Predictive Analysis(Classification)

Introduction

- Project background and context

SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers

In this capstone, we will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website.

Section 1

Methodology

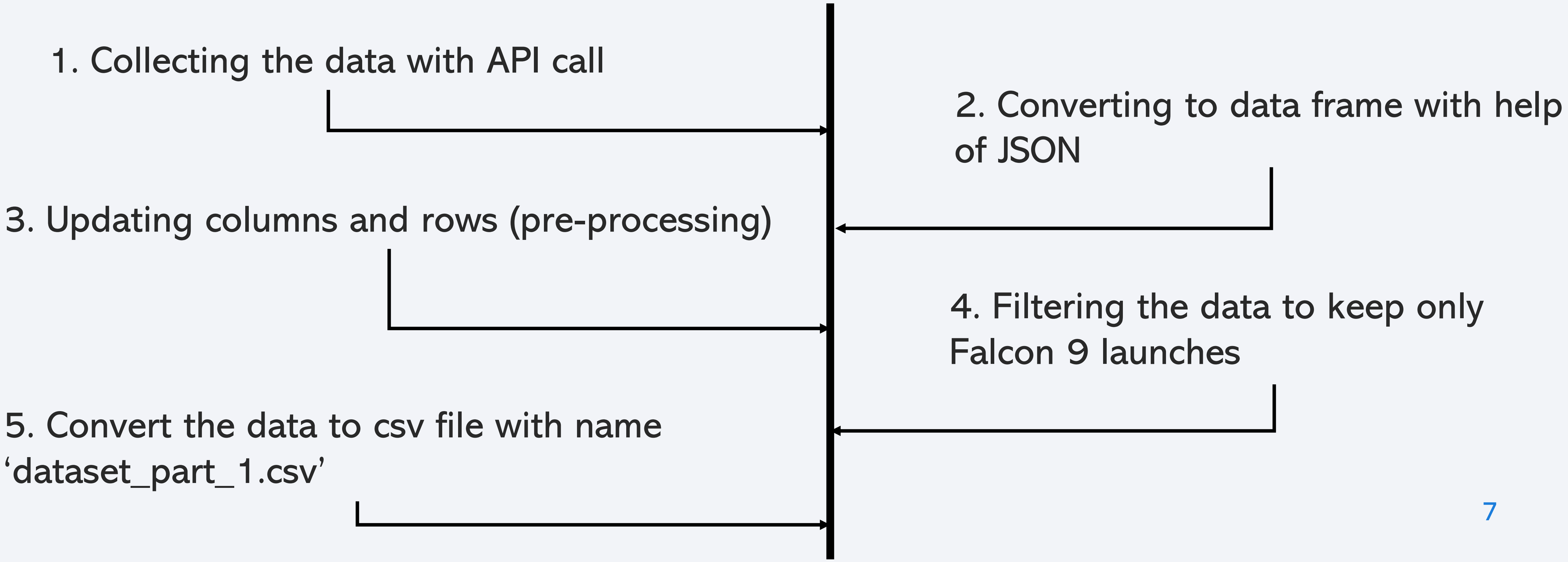
Methodology

Executive Summary

- Data collection methodology:
  - With Rest API and Web Scraping
- Perform data wrangling
  - Data were transformed and one hot encoded to be apply later on the ML models
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Discovering new pattern in the data with visualization techniques such as scatter plots
- Perform interactive visual analytics using Folium and Plotly Dash
  - Dash and Folium were used to achieve this goal
- Perform predictive analysis using classification models
  - Classification machine learning models were built to achieve this goal

Data Collection

- Data sets were collected using the APO call from several websites, I collected rocket, launchpad, payloads and cores data from https://api.spacexdata.com/v4 website.



1. Collecting the data with API call

2. Converting to data frame with help of JSON

3. Updating columns and rows (pre-processing)

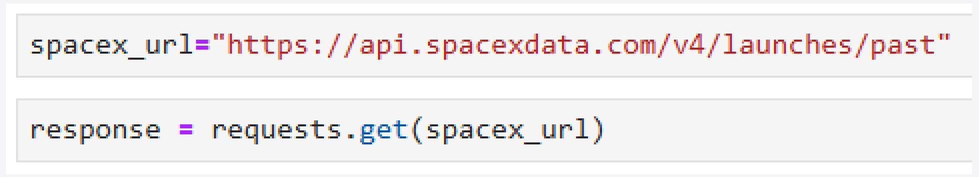
4. Filtering the data to keep only Falcon 9 launches

5. Convert the data to csv file with name 'dataset_part_1.csv'

Github repo: [DS_Coursera/jupyter-labs-spacex-data-collection-api.ipynb at master · huynguyentuank22/DS_Coursera (github.com)]

Data Collection – SpaceX API

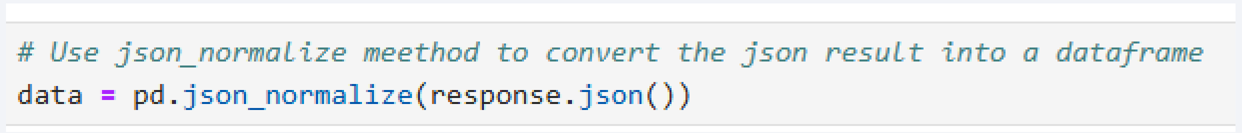
1. Collecting the data with API call



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

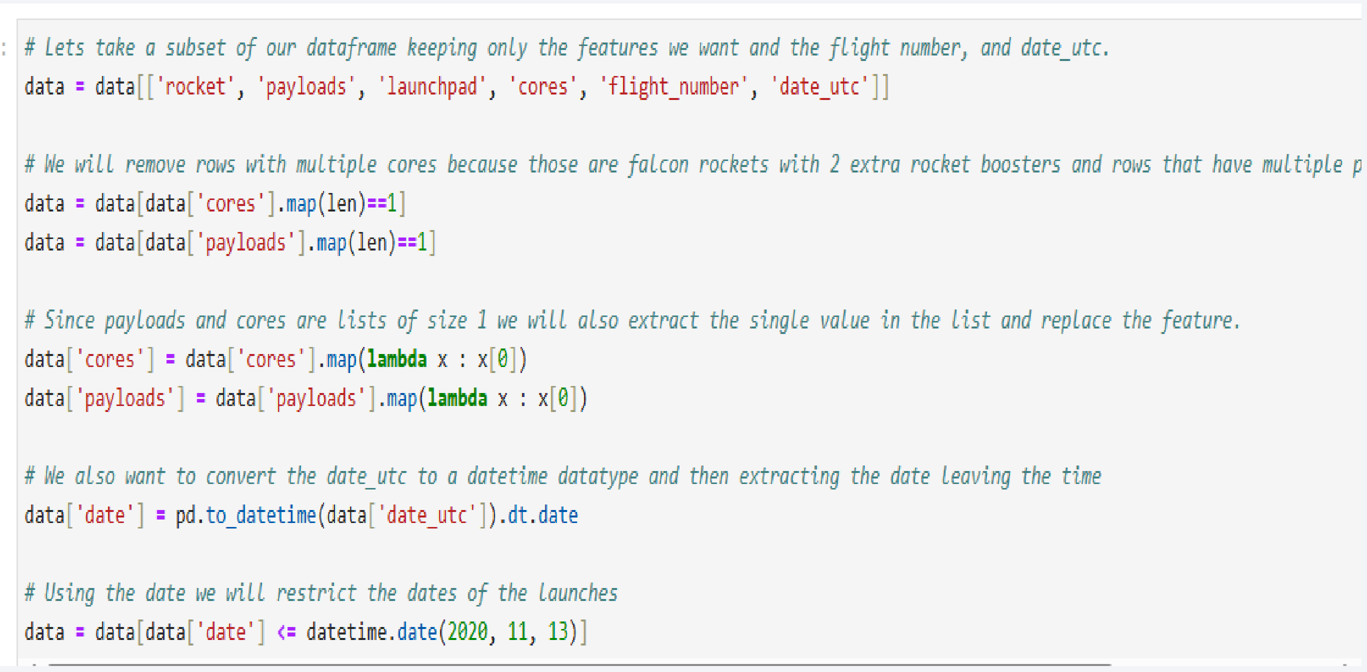
```
response = requests.get(spacex_url)
```

2. Converting to data frame with help of JSON



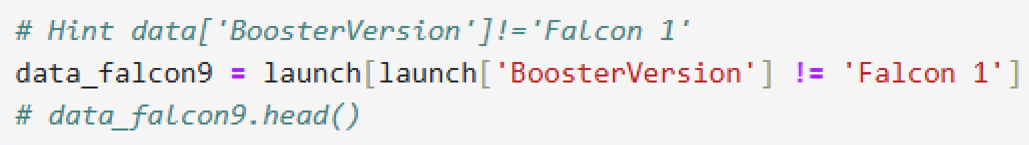
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

3. Updating columns and rows (pre-processing)



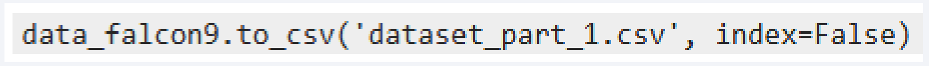
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple p
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)]
```

4. Filtering the data to keep only Falcon 9 launches



```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch[launch['BoosterVersion'] != 'Falcon 1']
# data_falcon9.head()
```

5. Convert the data to csv file with name 'dataset_part_1.csv'

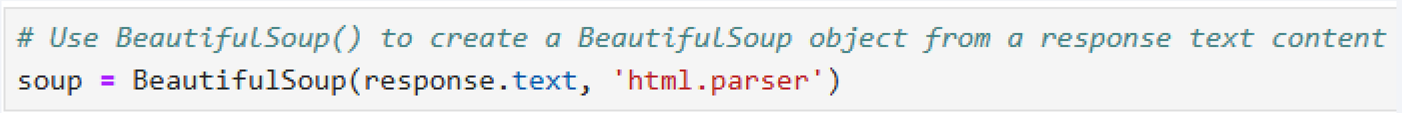


```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

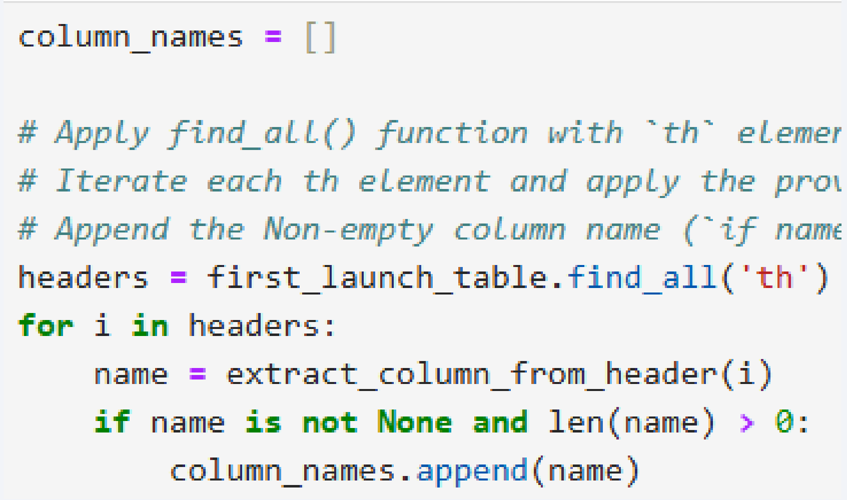
Github repo: [DS_Coursera/jupyter-labs-webscraping.ipynb at master · huynguyentuank22/DS_Coursera (github.com)]

1. Creating the BeautifulSoup



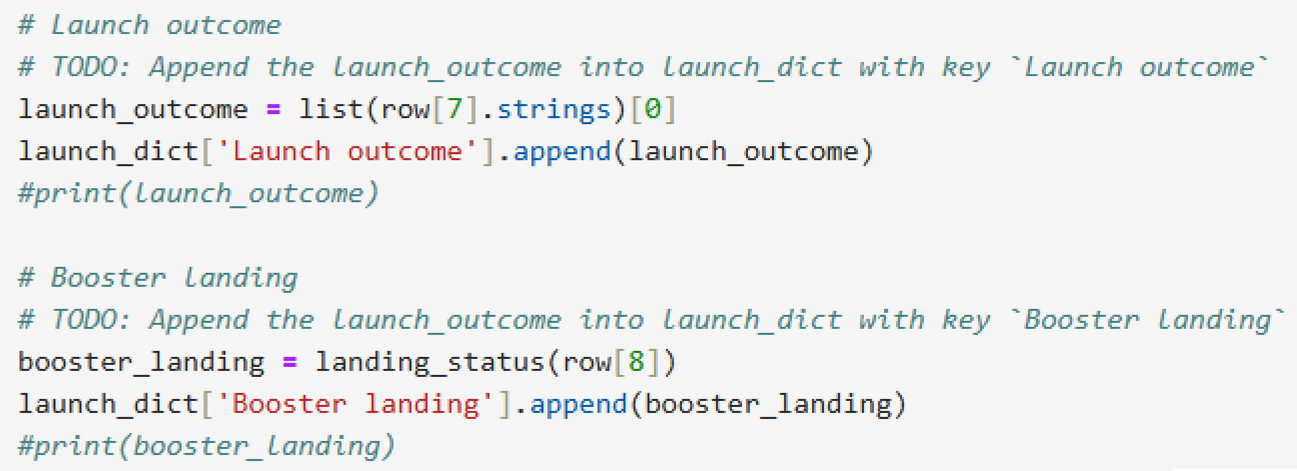
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

2. Getting column names



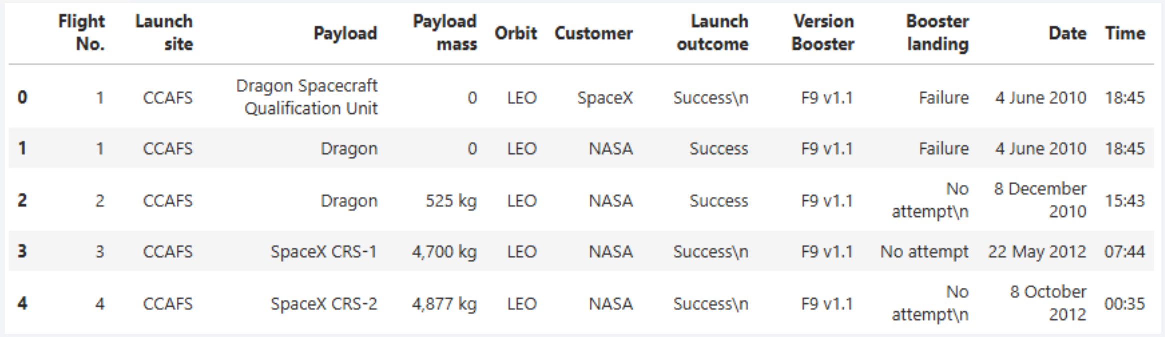
```
column_names = []

# Apply find_all() function with `th` elemen
# Iterate each th element and apply the prov
# Append the Non-empty column name (`if name
headers = first_launch_table.find_all('th')
for i in headers:
    name = extract_column_from_header(i)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

3. Creating the launch_dict



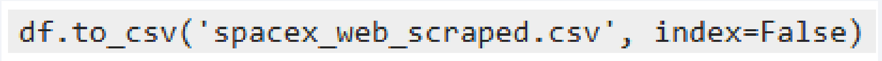
```
# Launch outcome
# TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
launch_outcome = list(row[7].strings)[0]
launch_dict['Launch outcome'].append(launch_outcome)
#print(launch_outcome)

# Booster landing
# TODO: Append the launch_outcome into launch_dict with key `Booster landing`
booster_landing = landing_status(row[8])
launch_dict['Booster landing'].append(booster_landing)
#print(booster_landing)
```

4. Converting the final dataframe



	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.1	Failure	4 June 2010	18:45
1	1	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.1	Failure	4 June 2010	18:45
2	2	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.1	No attempt\n	8 December 2010	15:43
3	3	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.1	No attempt	22 May 2012	07:44
4	4	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.1	No attempt\n	8 October 2012	00:35

5. Convert the data to csv file with name 'spacex_web_scraped.csv'

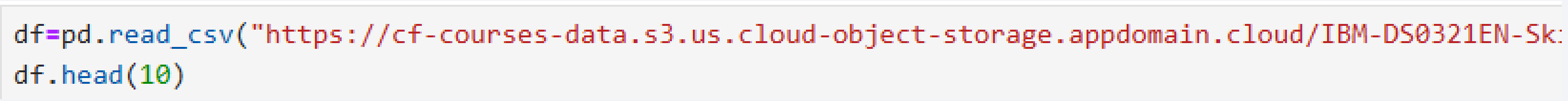


```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Github repo: DS_Coursera/labs-jupyter-spacex-Data wrangling.ipynb at master · huynguyentuank22/DS_Coursera (github.com)

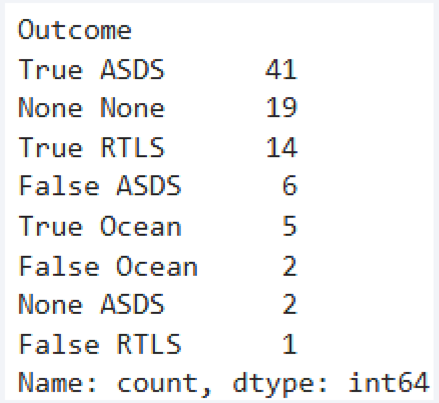
Data Wrangling

1. Loading the dataset



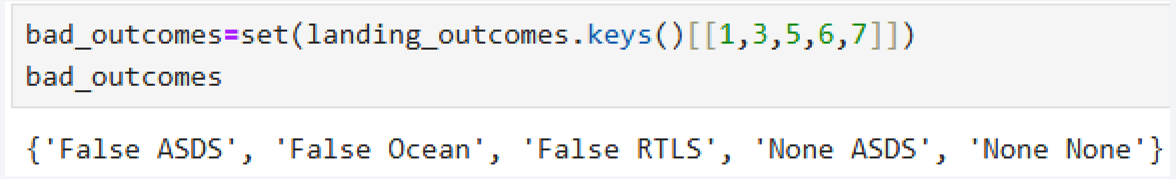
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
df.head(10)
```

2. Creating landing outcomes



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

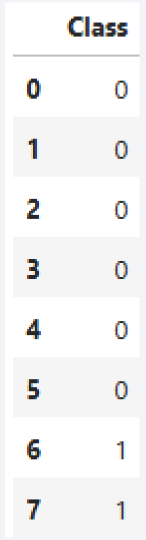
3. Finding the bad outcomes



```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

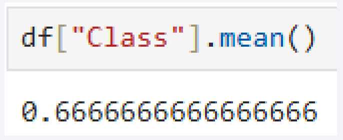
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

4. Presenting outcomes as 0 and 1



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

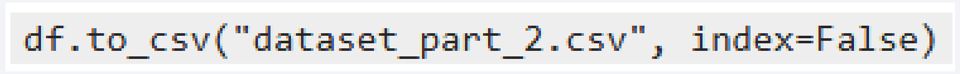
5. Determining the success outcome



```
df["Class"].mean()
```

```
0.6666666666666666
```

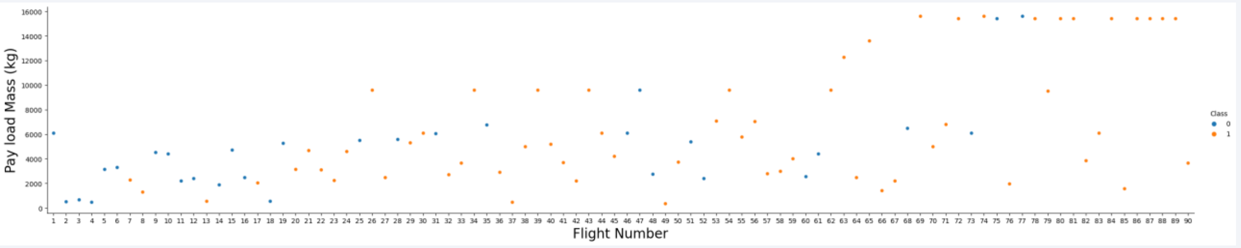
6. Convert the data to csv file with name ‘spacex_part_2.csv’



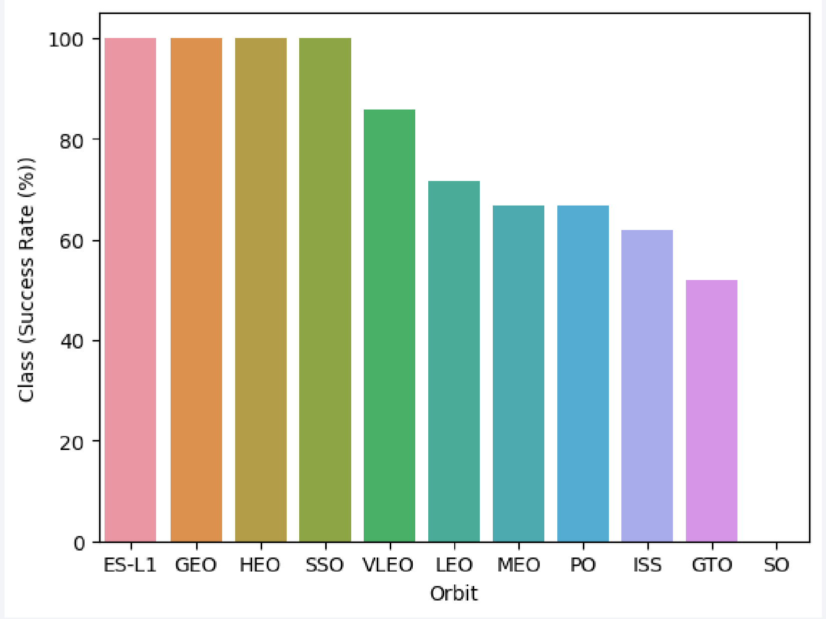
```
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

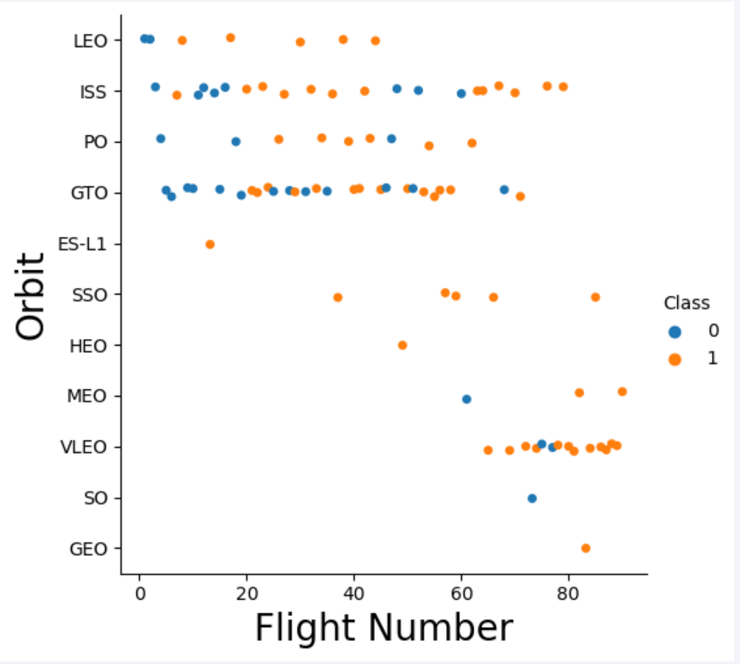
Github repo: [DS_Coursera/jupyter-labs-eda-dataviz.ipynb at master · huynguyentuank22/DS_Coursera (github.com)]



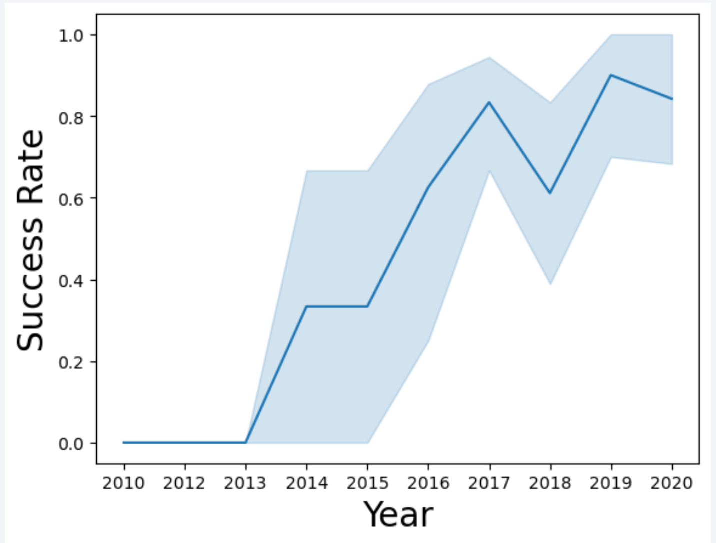
Categorial plot between Flight number and Payload mass (kg)



Bar chart between Orbit and Success rate of each orbit



Scatter plot between Orbit and Flight number



Line chart between Year and Success rate

EDA with SQL

Github repo: [DS_Coursera/jupyter-labs-eda-sql-coursera_sqllite.ipynb at master · huynguyentuank22/DS_Coursera (github.com)]

- The following SQL queries were performed for EDA
  - 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 succesful landing outcome in ground pad was acheived.
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - List the total number of successful and failure mission outcomes
  - List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
  - List the 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 landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

Build an Interactive Map with Folium

- folium.Marker() was used to create marks on the map.
- folium.Circle() was used to create circles above markers on the map.
- folium.Icon() was used to create an icon on the map.
- folium.PolyLine() was used to create polynomial line between the points.
- folium.plugins.AntPath() was used to create animated line between the points.
- makerCluster() was used to simplify the maps which contain several markers with identical coordination.

Github repo: DS_Coursera/lab_jupyter_launch_site_location.ipynb at master · huynguyentuank22/DS_Coursera (github.com)

Build a Dashboard with Plotly Dash

- Dash and html components were used as they are the most important thing and almost everything depends on them, such as graphs, tables, dropdowns, etc.
- Pandas was used to simplifying the work by creating dataframe.
- Plotly was used to plot the graphs.
- Pie chart and scatter chart were used to for plotting purposed.
- Rangeslider was used for payload mass range selection.
- Dropdown was used to launch sites.

Github repo: DS_Coursera/spacex_dash_app.py at master · huynguyentuank22/DS_Coursera (github.com)

Predictive Analysis (Classification)

1. Builing the model

Create column for the class

Standardize the data

Split the data into train and test sets

Build GridSearchCV model and fit the data

2. Evaluating the model

Calculating the accuracies

Calculating the confusion matrixes

Plot the results

3. Updating columns and rows (pre-processing)

Find the best hyperparameters for the models

Find the best model with highest accuracy

Confirm the optimal model

Results

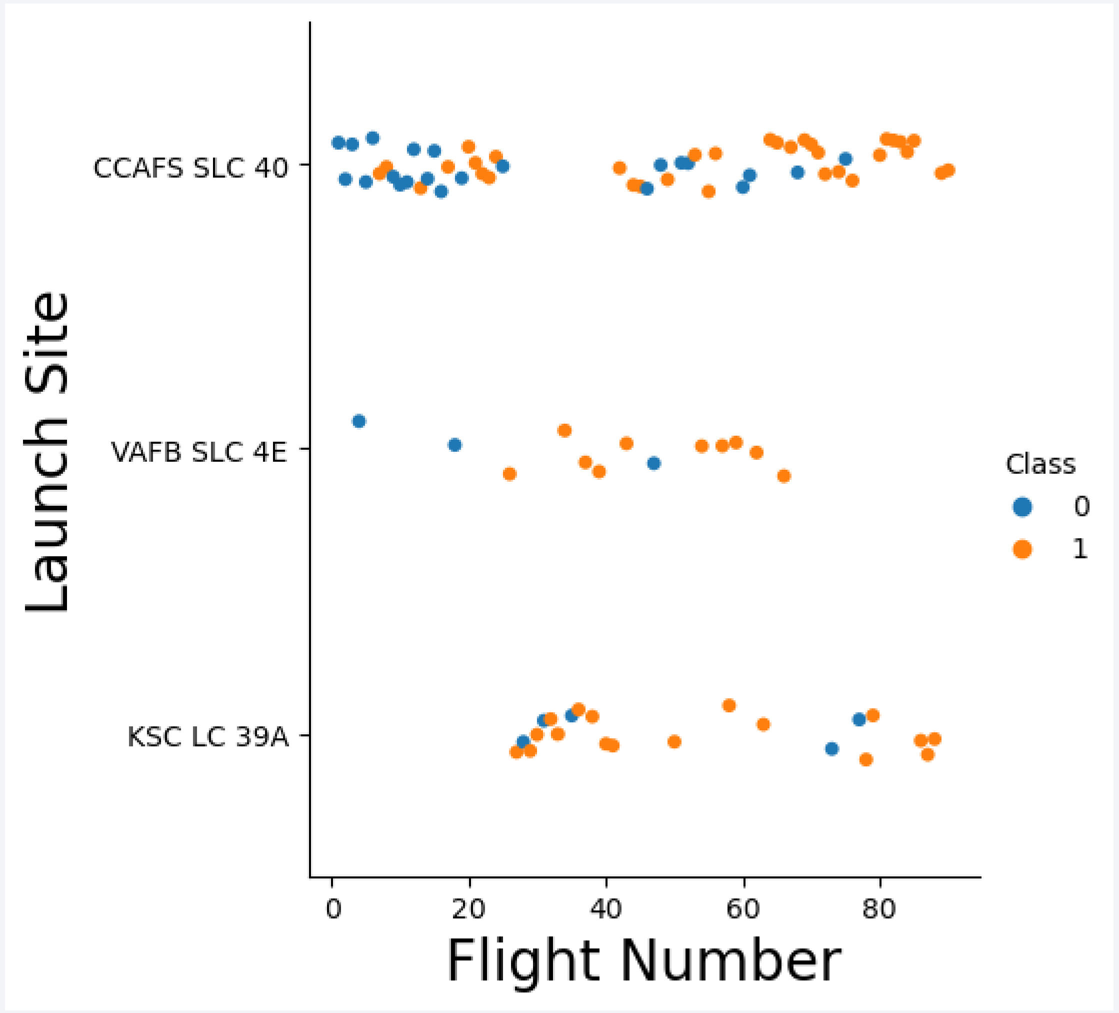
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

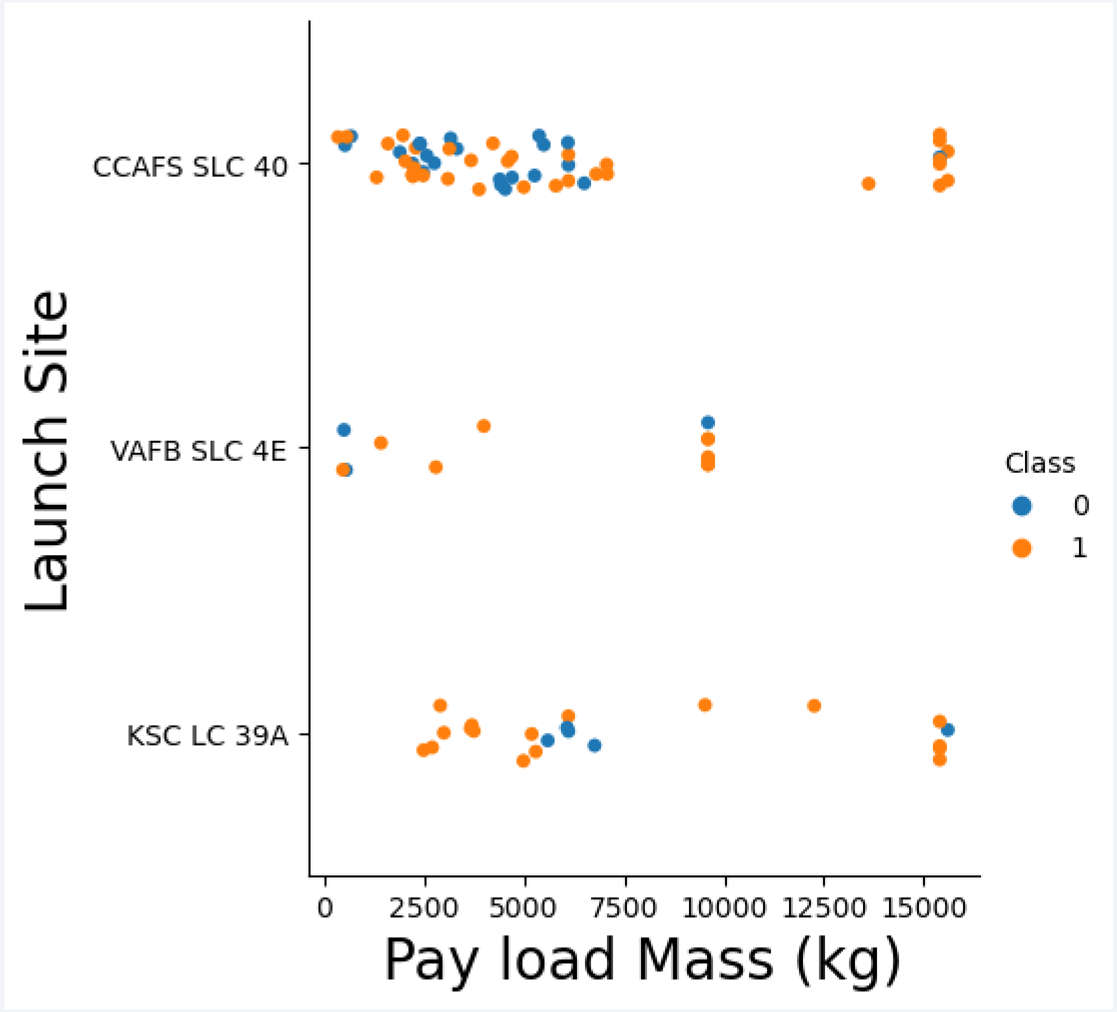
Flight Number vs. Launch Site

- With the increase of flight number, the success rate is increasing as well in the launch sites



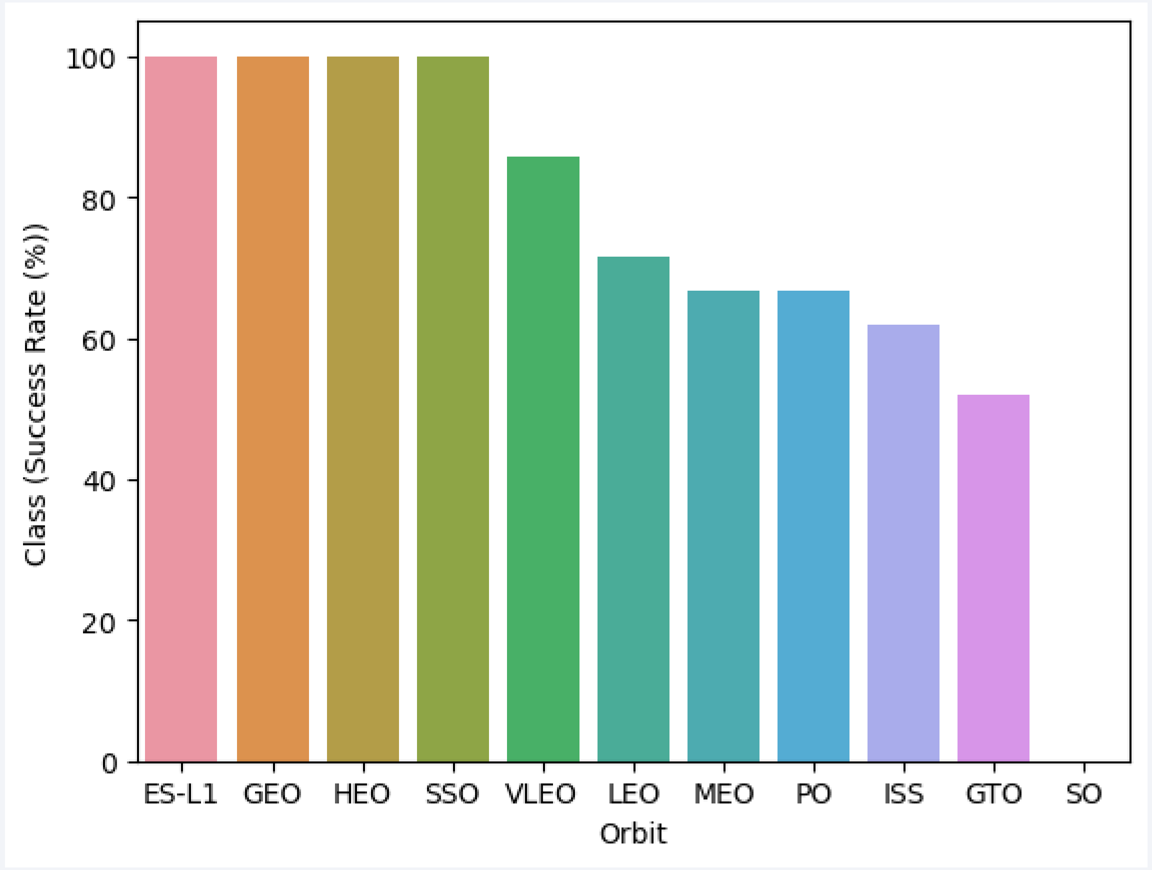
Payload vs. Launch Site

- With the increase of Payload Mass, the success rate is increasing as well in the launch sites



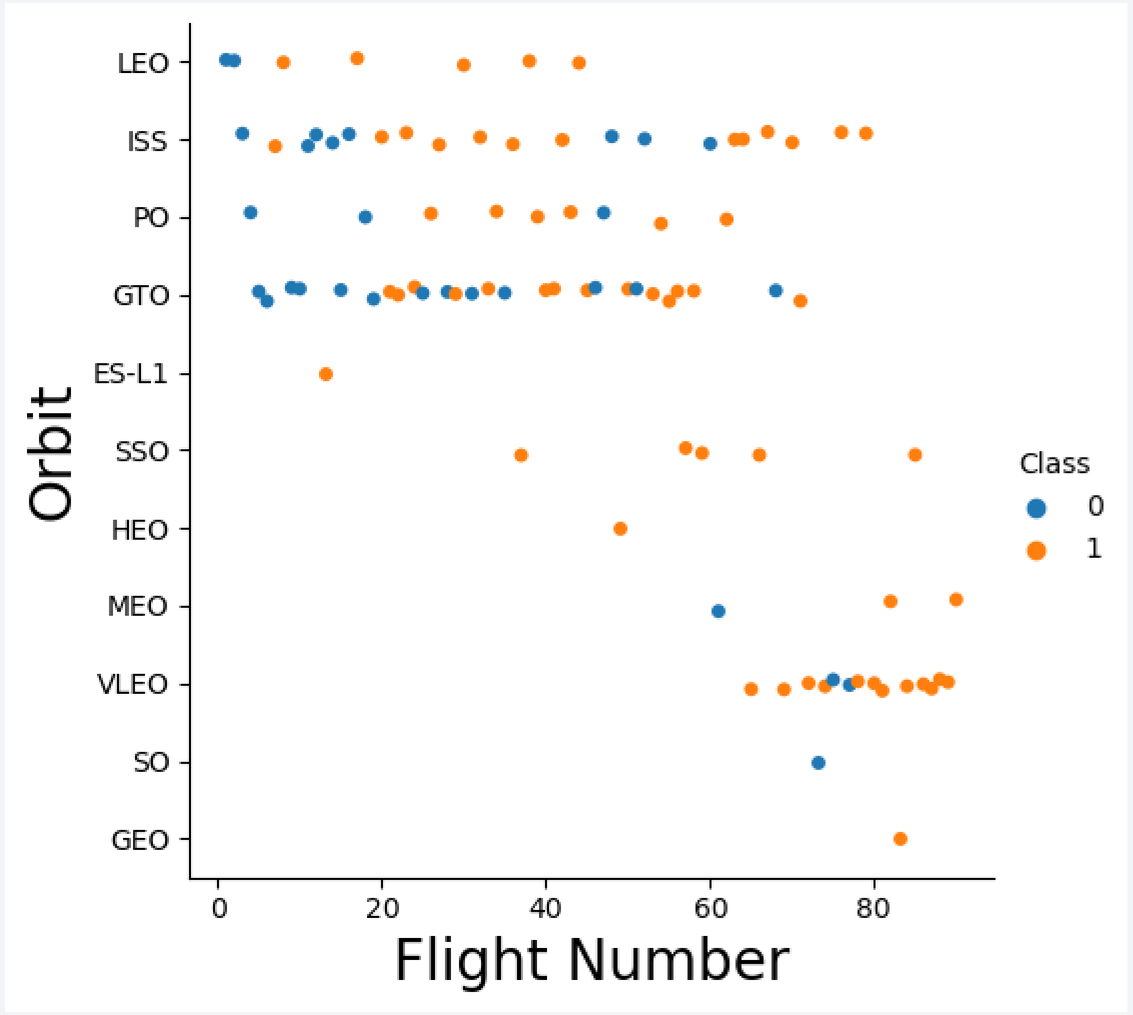
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO have a success rate of 100%
- SO has a success rate of 0%



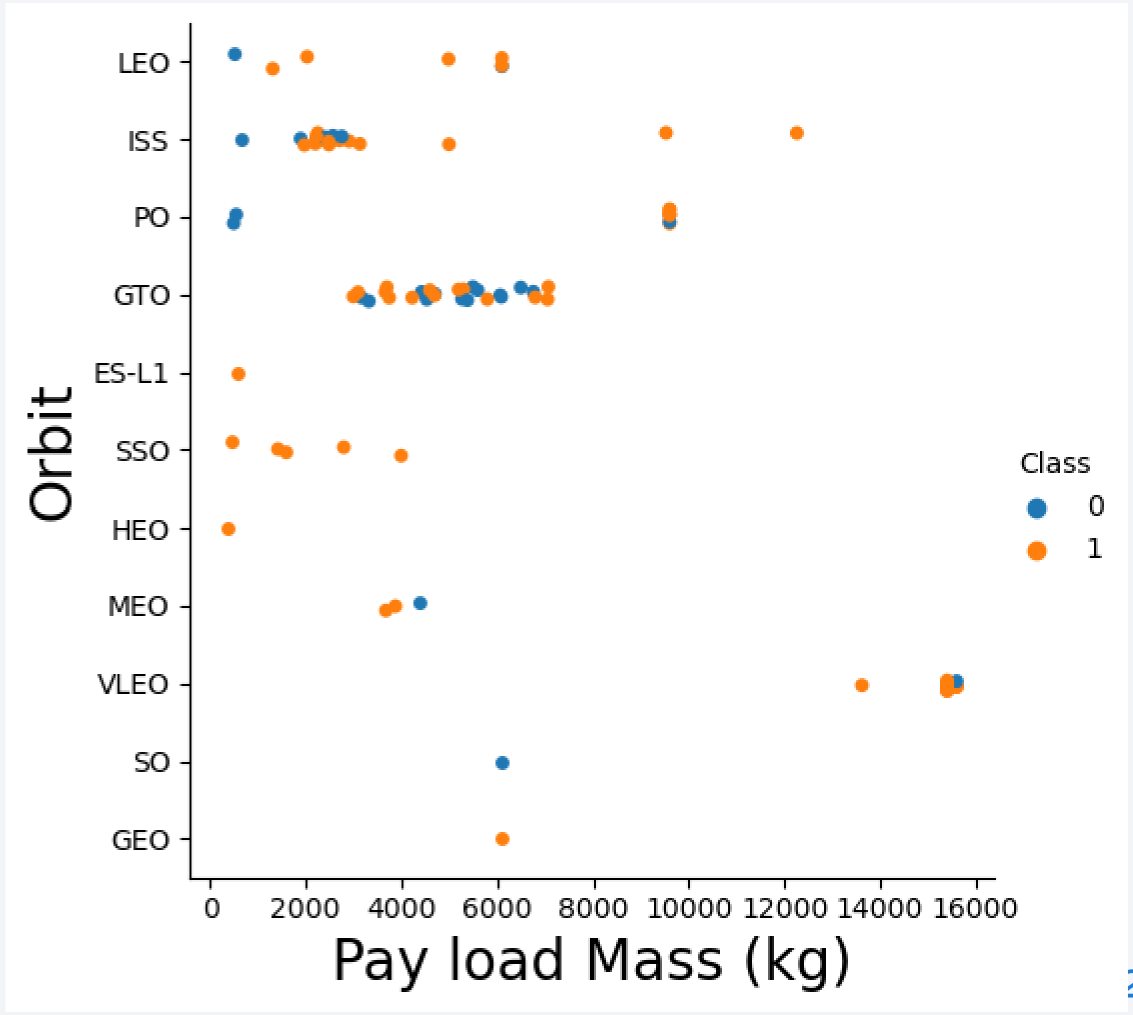
Flight Number vs. Orbit Type

- It's hard to tell anything here, but we can say there is no actual relationship between flight number and GTO



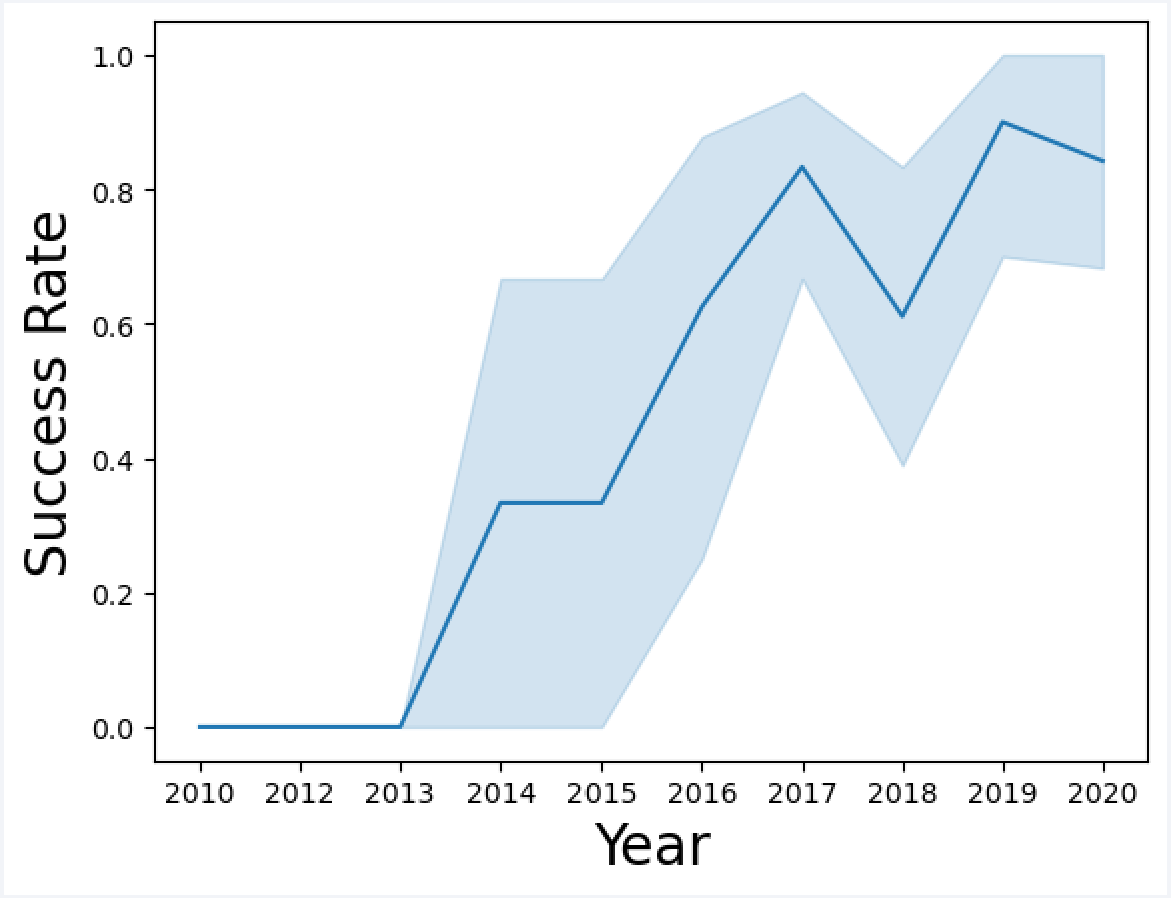
Payload vs. Orbit Type

- First thing to see in how the Payload Mass between 2000 and 3000 is affecting ISS.
- Similarly, Payload Mass between 3000 and 7000 is affecting GTO.

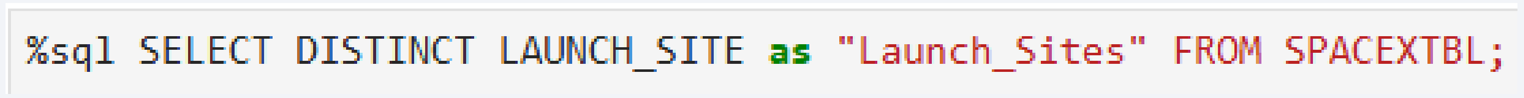


Launch Success Yearly Trend

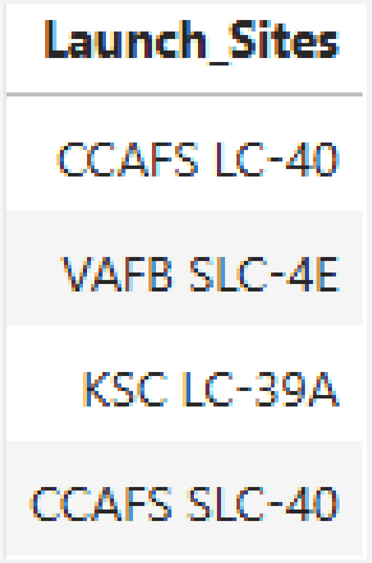
- Since the year 2013, there was a massive increase in success rate. However, it dropped little in 2018 but later it got stronger than before.



All Launch Site Names



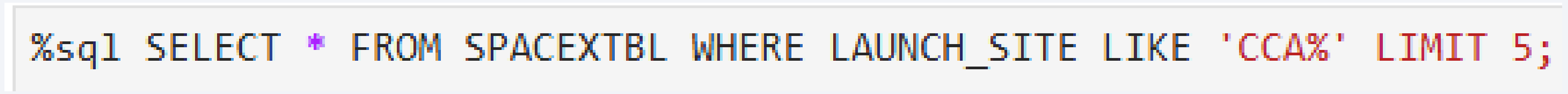
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```



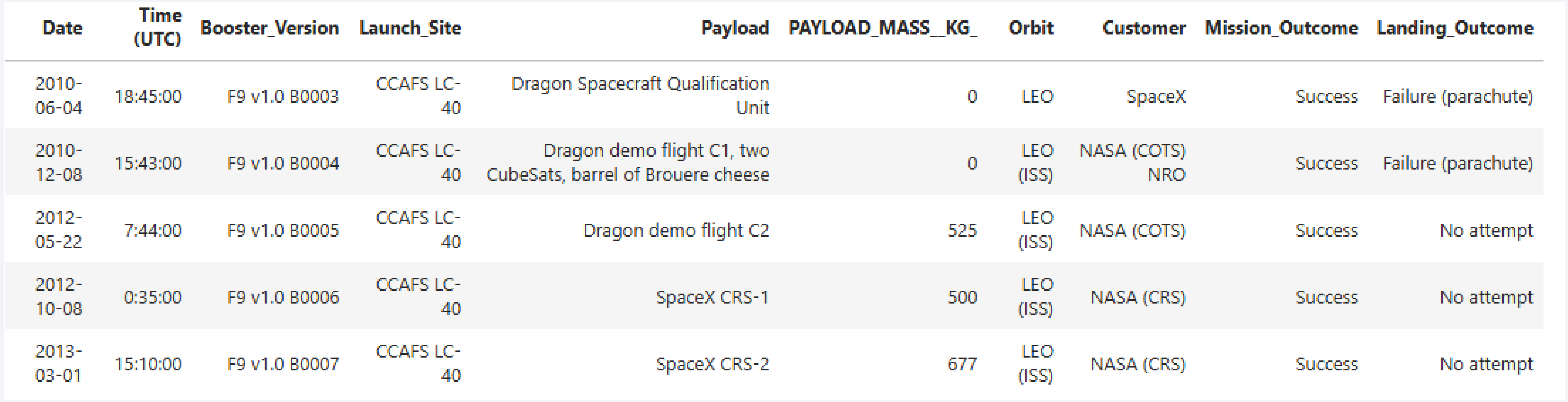
Launch_Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

- We can get the unique values by using "DISTINCT"

Launch Site Names Begin with 'CCA'



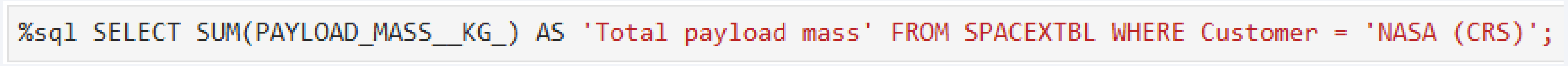
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```



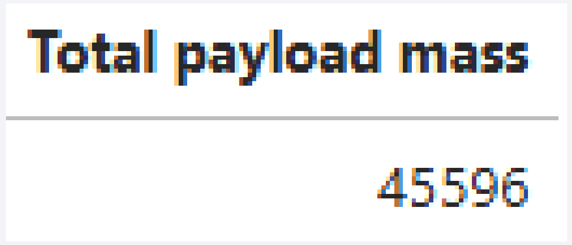
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- We can get only 5 rows by using "LIMIT"

Total Payload Mass



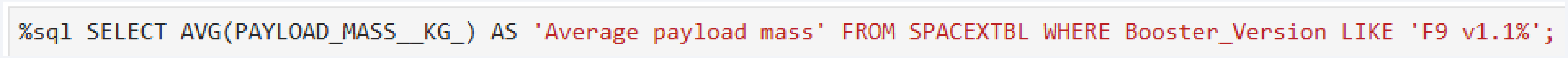
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS 'Total payload mass' FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```



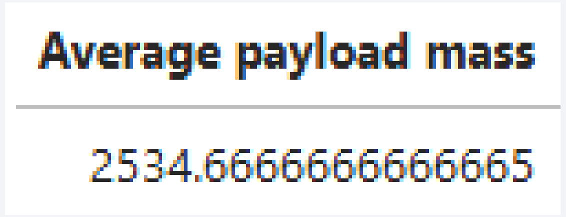
Total payload mass
45596

- We can get the sum of all values by using “SUM”

Average Payload Mass by F9 v1.1



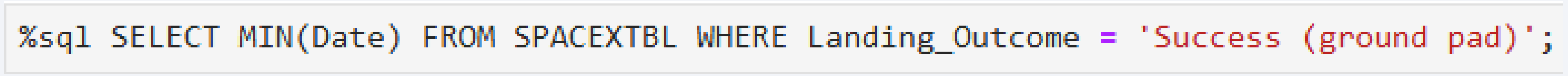
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS 'Average payload mass' FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1%';
```



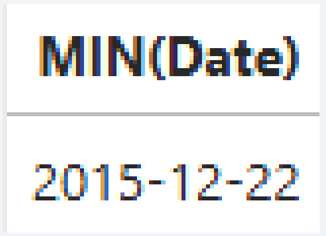
Average payload mass
2534.6666666666665

- We can get the average of all values by using “AVG”

First Successful Ground Landing Date



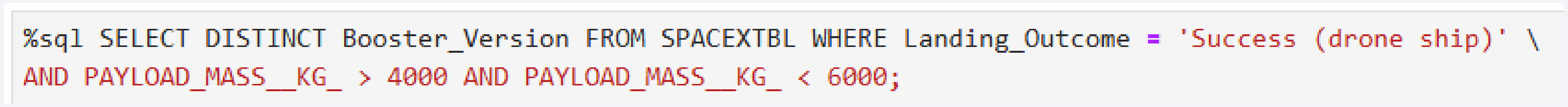
```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';
```



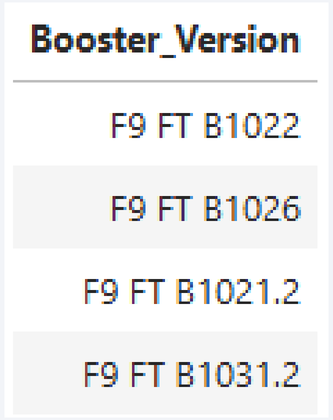
MIN(Date)
2015-12-22

- We can get the first successful data by using “MIN”, because the first date is same with the minimum date

Successful Drone Ship Landing with Payload between 4000 and 6000



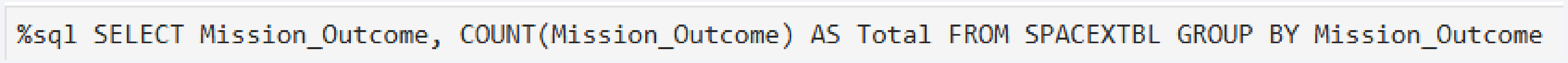
```
%sql SELECT DISTINCT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```



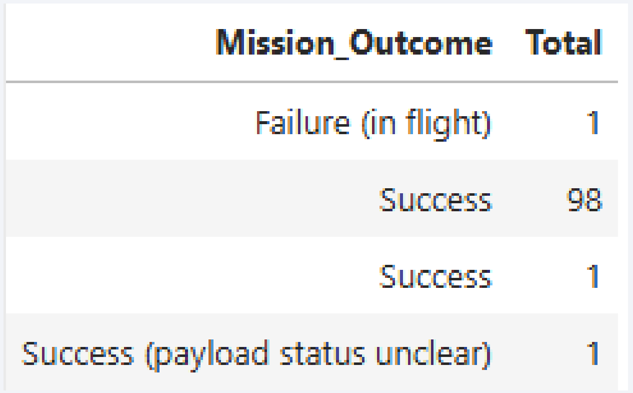
Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

- The payload mass data was taken between 4000 and 6000 only, and the landing outcome was determined to be “success drone ship”

Total Number of Successful and Failure Mission Outcomes



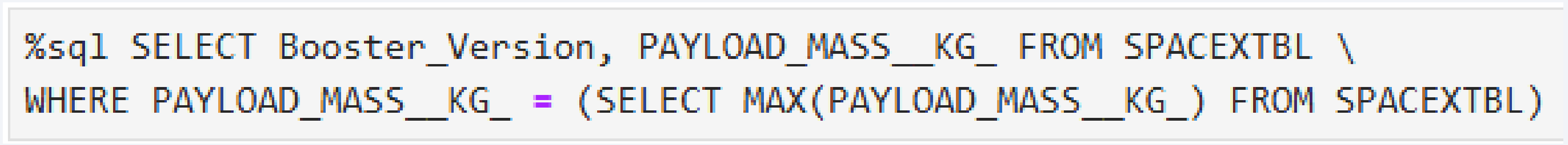
```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) AS Total FROM SPACEXTBL GROUP BY Mission_Outcome
```



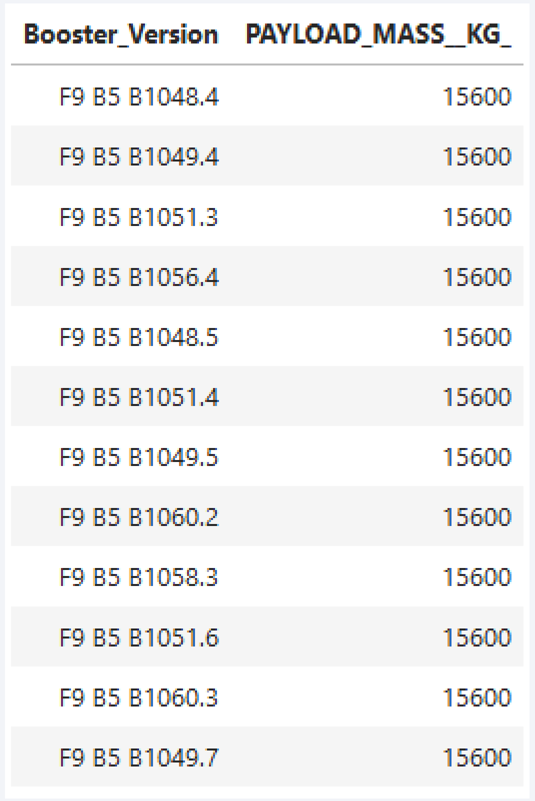
Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- We can get to total number of successful and failure mission by using "COUNT" and "GROUP BY"

Boosters Carried Maximum Payload



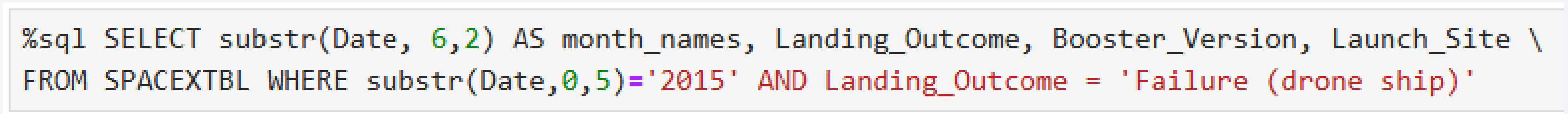
```
%sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```



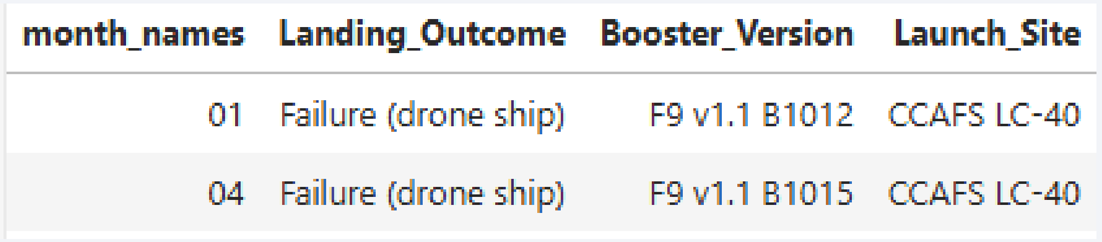
Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

- We can get the maximum payload masses by using “MAX”

2015 Launch Records



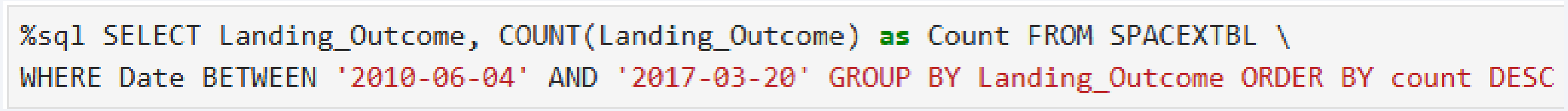
```
%sql SELECT substr(Date, 6,2) AS month_names, Landing_Outcome, Booster_Version, Launch_Site \
FROM SPACEXTBL WHERE substr(Date,0,5)='2015' AND Landing_Outcome = 'Failure (drone ship)'
```



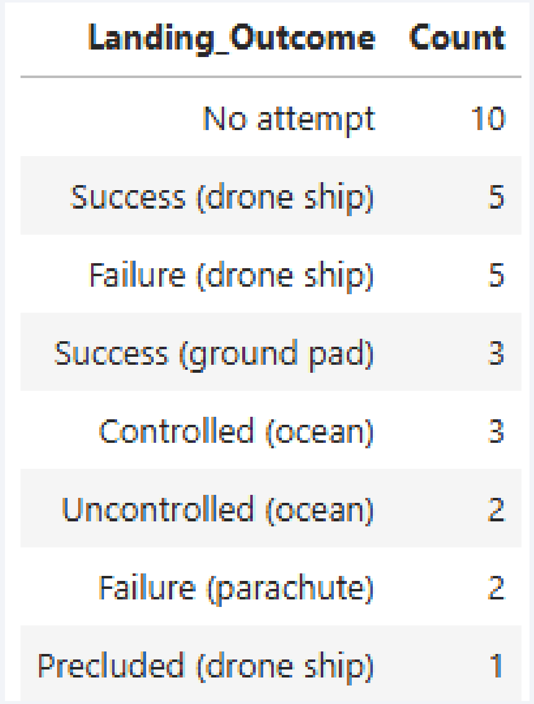
month_names	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- We can get the months by using month(DATE) and in the WHERE function we assigned the year value to “2015”

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



```
%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) as Count FROM SPACEXTBL \
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY count DESC
```



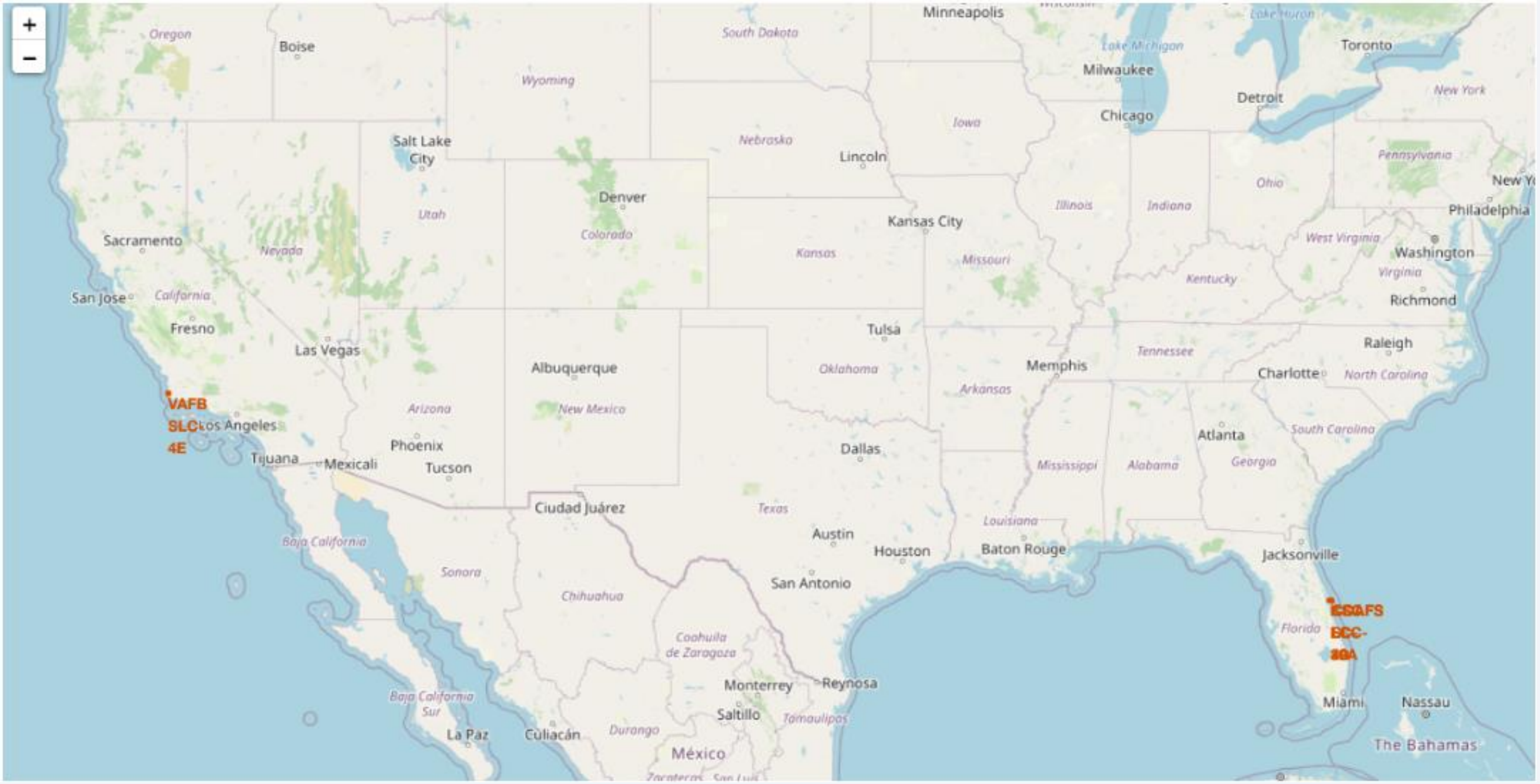
Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

- By using "ORDER" we can order the values in descending order, and with "COUNT" we can count all numbers as we did previously

Section 3

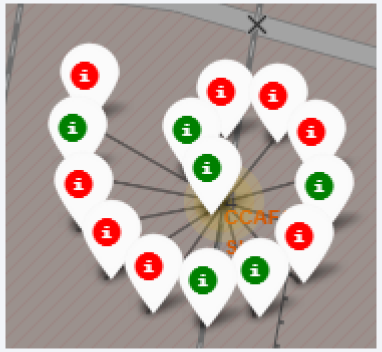
Launch Sites Proximities Analysis

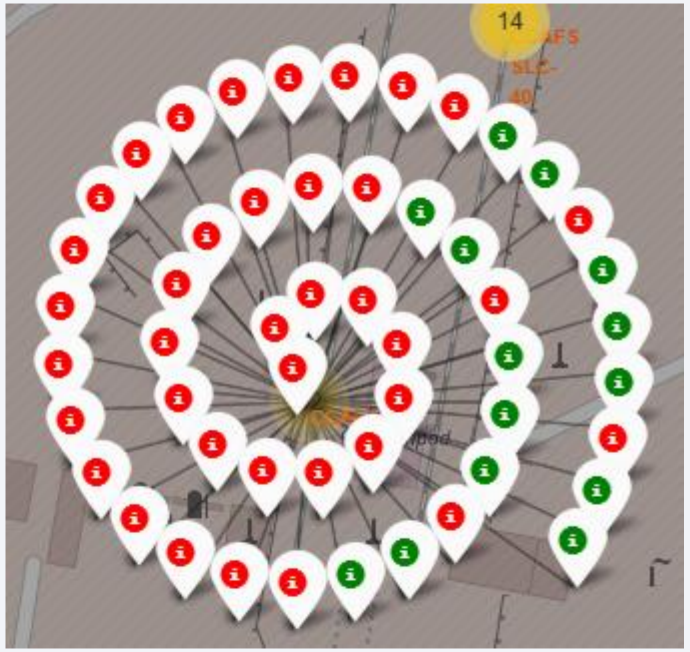
All Launch Sites Location Markers



- All the launches are near USA, Florida, and California

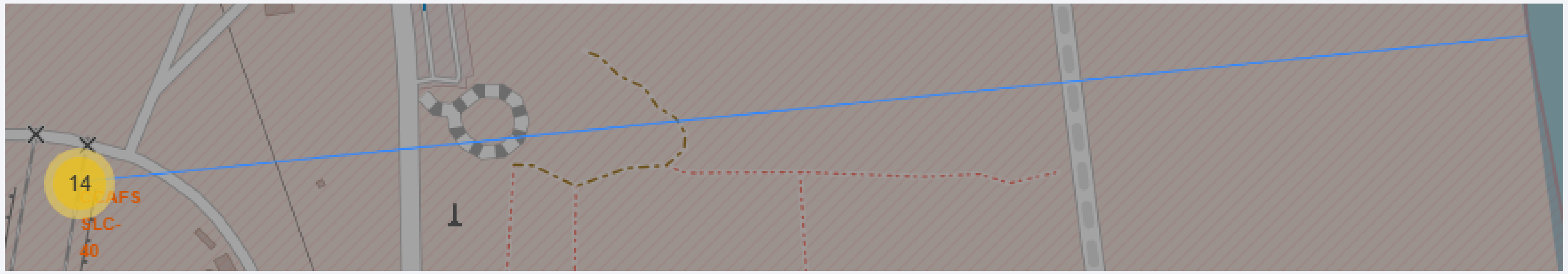
Color-labeled Launch Outcomes



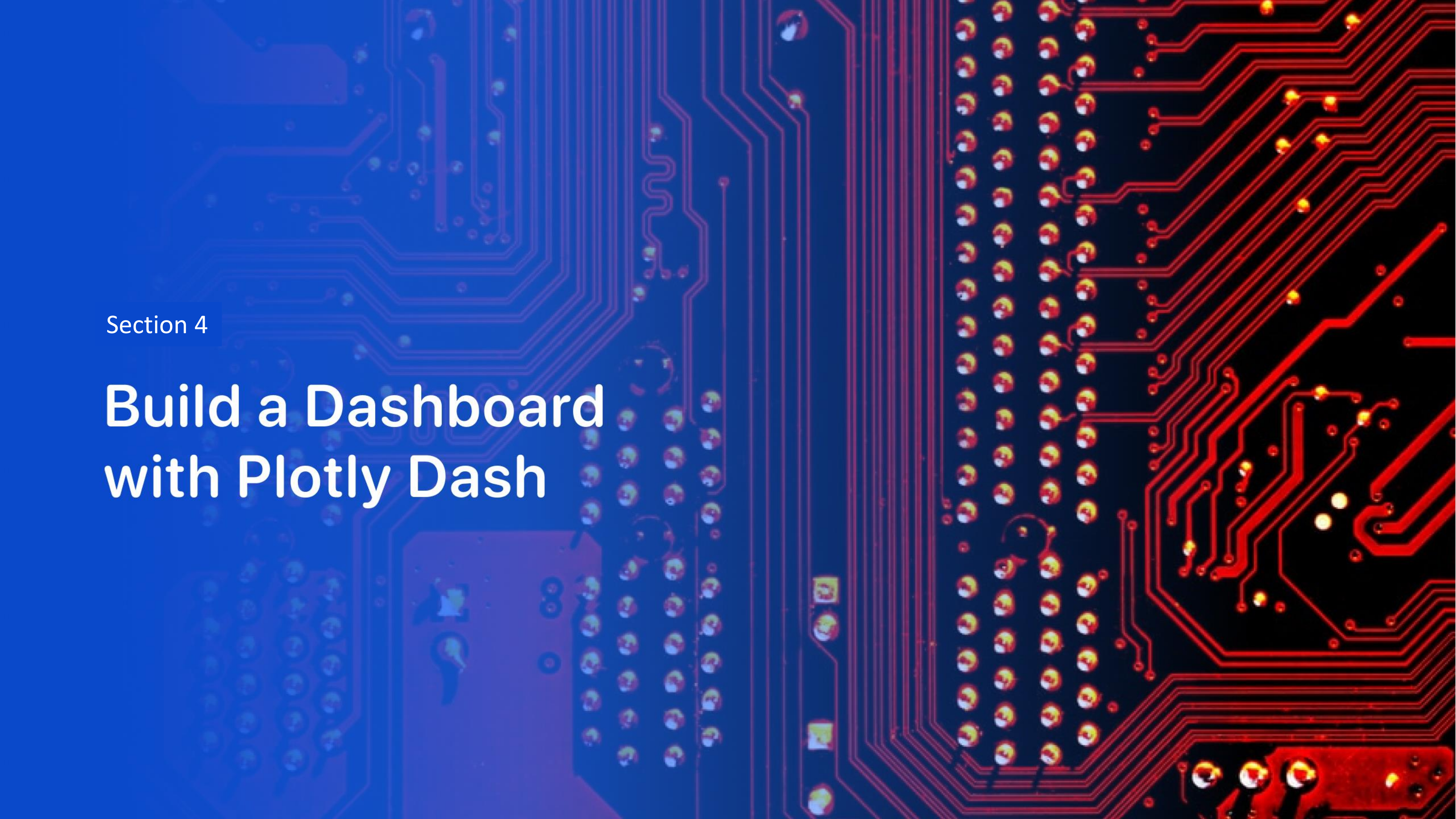


- Green means Successful
- Red means Failure

Launch Sites to its Proximities



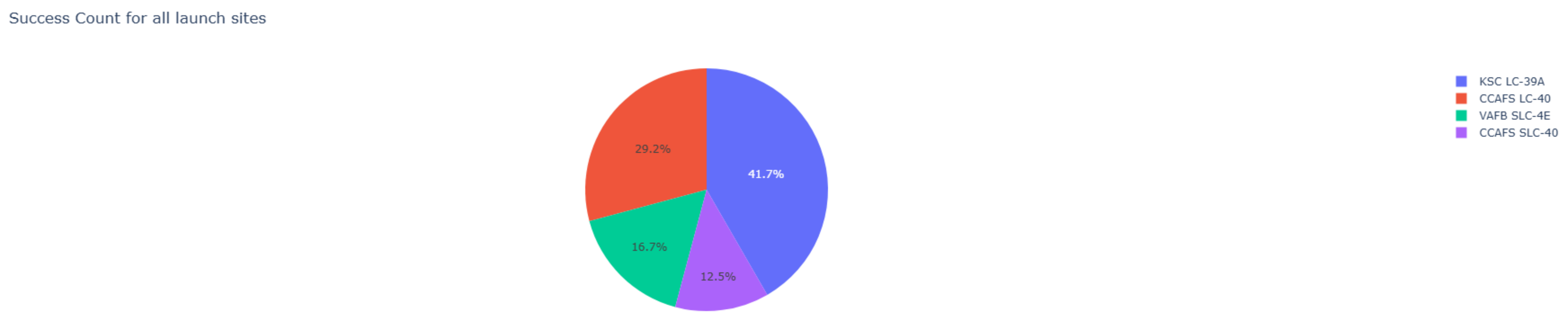
- The distance from launch sites to its proximite



Section 4

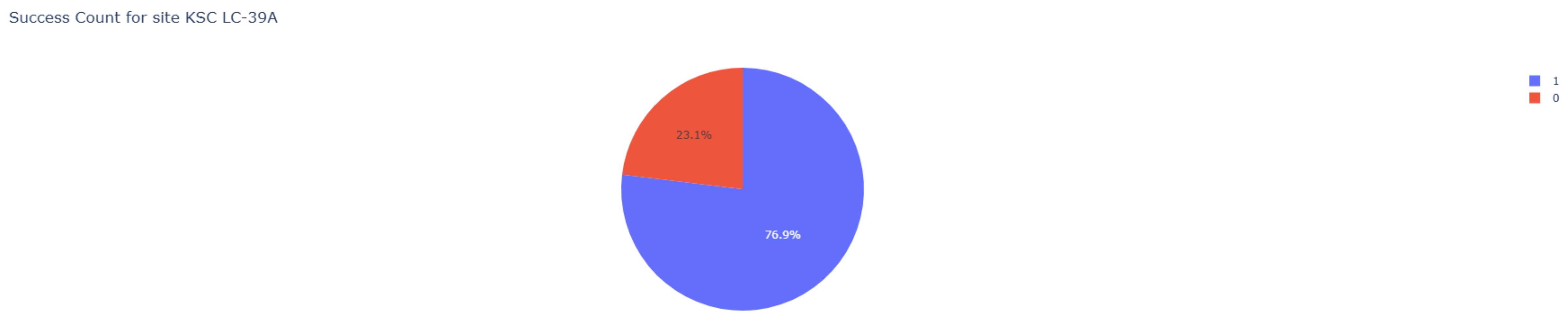
Build a Dashboard with Plotly Dash

Launch Success Count



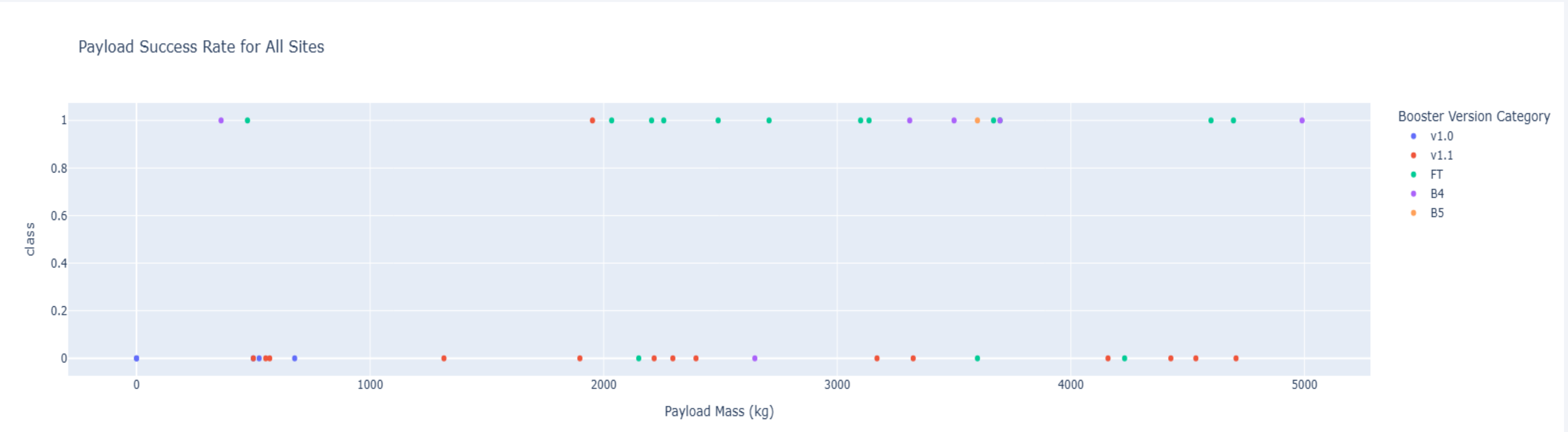
- KSC LC-39A has the highest success score with 41.7%
- CCAFS LC-40 comes next with 29.2%
- Finally, VAFB SLC-4E and CCAFS SLC-40 with 16.7% and 12.5% respectively

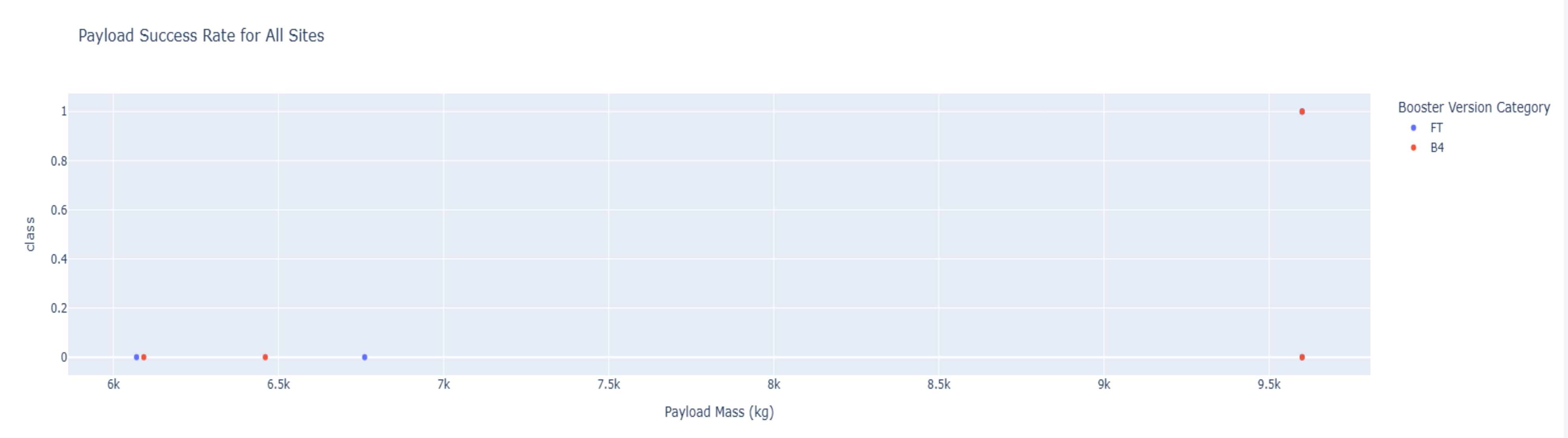
Launch Site with Highest Score



- KSC LC-39A has the highest score with 76.9% with payload range of 2000kg – 10000kg, and FT booster version has the highest score

Payload vs Launch Outcome





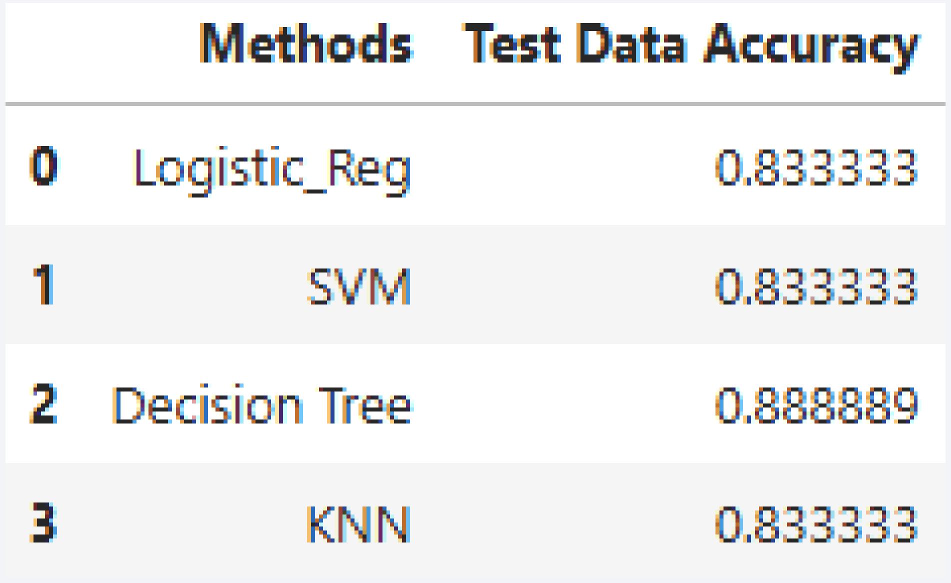
- Payload 0kg – 5000kg (first half)

- Payload 6000kg – 10000kg (second half)

Section 5

Predictive Analysis (Classification)

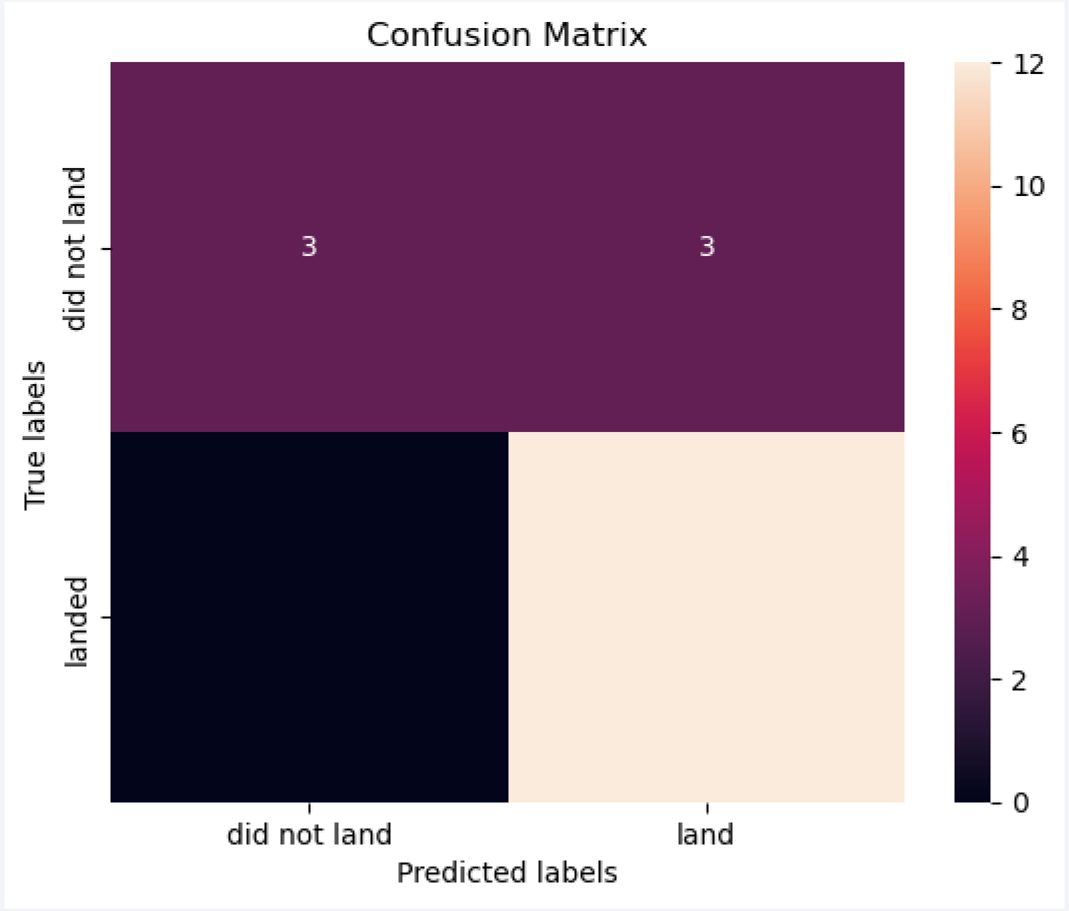
Classification Accuracy



	Methods	Test Data Accuracy
0	Logistic_Reg	0.833333
1	SVM	0.833333
2	Decision Tree	0.888889
3	KNN	0.833333

- Decision Tree has the highest accuracy with almost 0.89, then comes the remaining models with almost same accuracy of 0.83

Confusion Matrix



Conclusions

- We found the site with highest score which was KSC LC-39A
- The payload of 0kg to 5000kg was more diverse than 6000kg to 10000kg
- Decision Tree was the optimal model with accuracy of almost 89%
- We calculated the launch sites distance to its proximities

Appendix

- All code can be found on my Github
- Github repo: https://github.com/huynguyentuank22/DS_Coursera.git

Thank you!