IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Joey Kim>
<04/28/2023>

Outline

Executive Summary

- **Summary of methodologies**
  - Data Collection using API & Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis(EDA) via SQL and Data Visualization
  - Map Visualization with Folium
  - Dashboard by Plotly Dash
  - Predictive Analysis – Testing and Prediction
- **Summary of all results**
  - Exploratory Data Analysis(EDA)
  - Interactive Maps
  - Prediction

Introduction

- Project background and context
  - The prediction on successful landing of Falcon 9 first stage can determine the cost of rocket launch as the majority of enormous cost reduction of Falcon 9 Rocket Launch comes conditionally from having its first stage landing successfully. This information can be a cost estimator for SPACEX as well as for other potential competitors in the industry.
- Problems you want to find answers
  - I want to discover what variables contribute to a successful landing of Falcon 9's first stage in order to achieve the claimed low cost of Falcon 9 Rocket Launch

Section 1

Methodology

Methodology

- Executive Summary
- Data collection methodology:
  - SPACE X REST API & Wikipedia Web Scraping
- Perform data wrangling
  - Identification of counts of variables, variable types, binary-conversion, column changes
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Using charts and graphs to present patterns and findings of variables
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Gathering, converting, segregating, and testing data into different models

Data Collection

Rest API

Import necessary libraries -> send request API to retrieve 'rocket launch data'-> conversion of data (json -> dict -> dataframe) -> export data as csv

URL: https://api.spacexdata.com/v4/launches/past

Web Scraping

Import necessary libraries and functions -> send HTTP request to retrieve text data -> conversion of data (txt -> dict -> dataframe) -> export data as csv

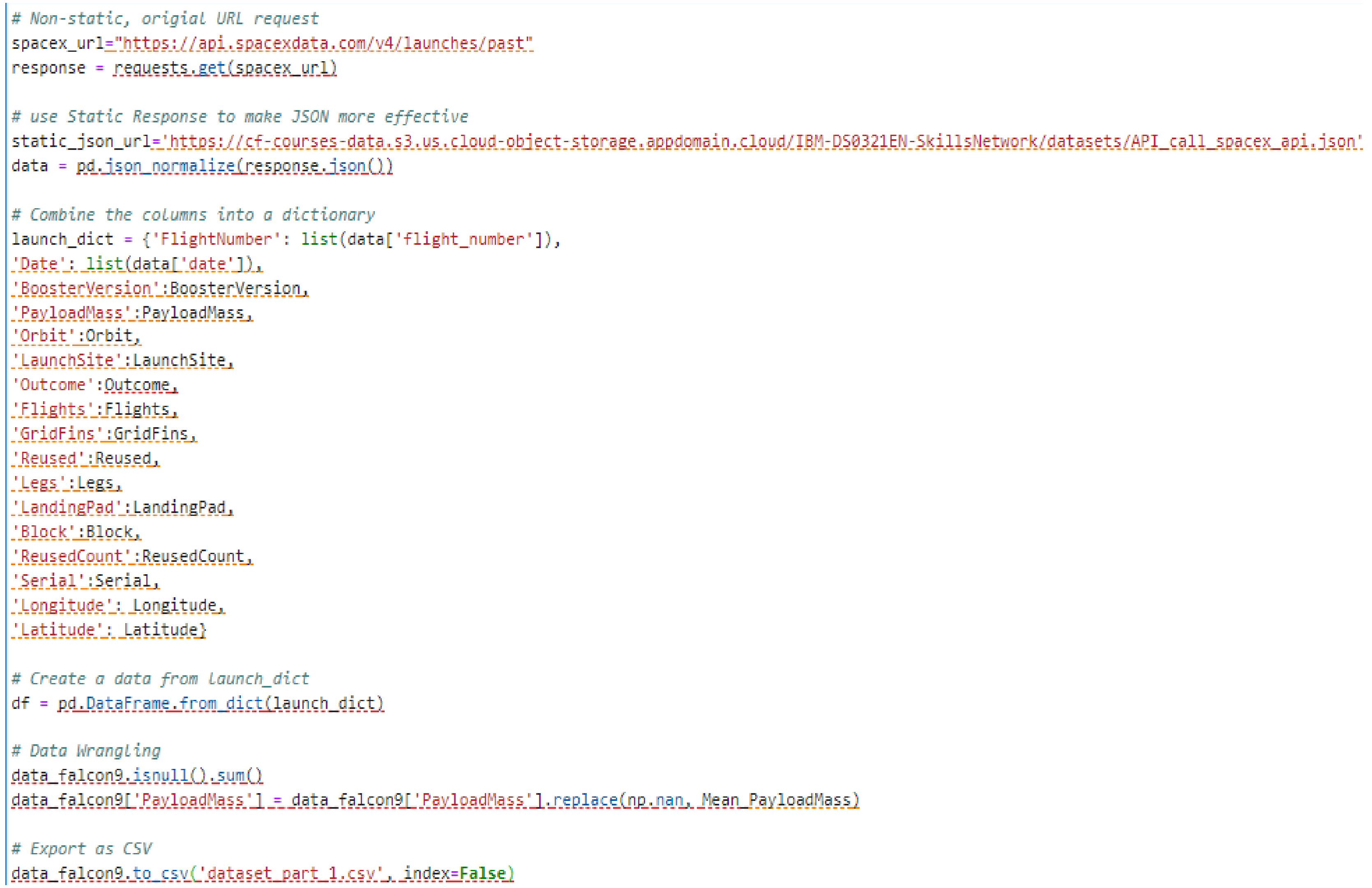
URL: https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Data Collection – SpaceX API

- Please follow the flow chart on the right and refer to '#' comments for details

- Github URL:

https://github.com/jksb1122/Peer-graded-Assignment-Submit-Your-Work-and-Grade-Your-Peers.git

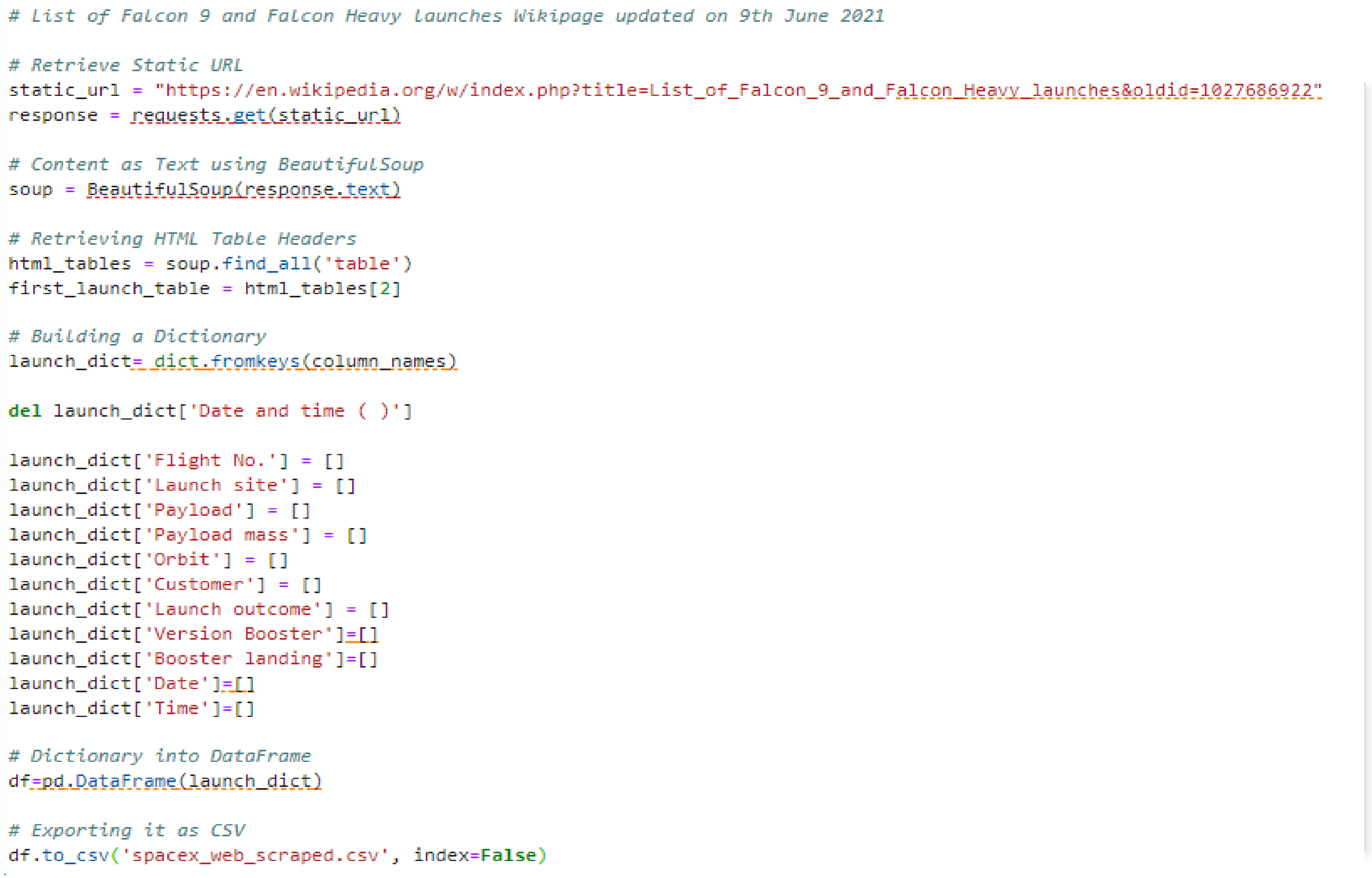


```
# Non-static, origial URL request
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)

# use Static Response to make JSON more effective
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
data = pd.json_normalize(response.json())

# Combine the columns into a dictionary
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 a data from launch_dict
df = pd.DataFrame.from_dict(launch_dict)

# Data Wrangling
data_falcon9.isnull().sum()
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, Mean_PayloadMass)

# Export as CSV
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

- **Please follow the flow chart on the right and refer to '#' comments for details**

- Github URL: https://github.com/jksb1122/Peer-graded-Assignment-Submit-Your-Work-and-Grade-Your-Peers.git

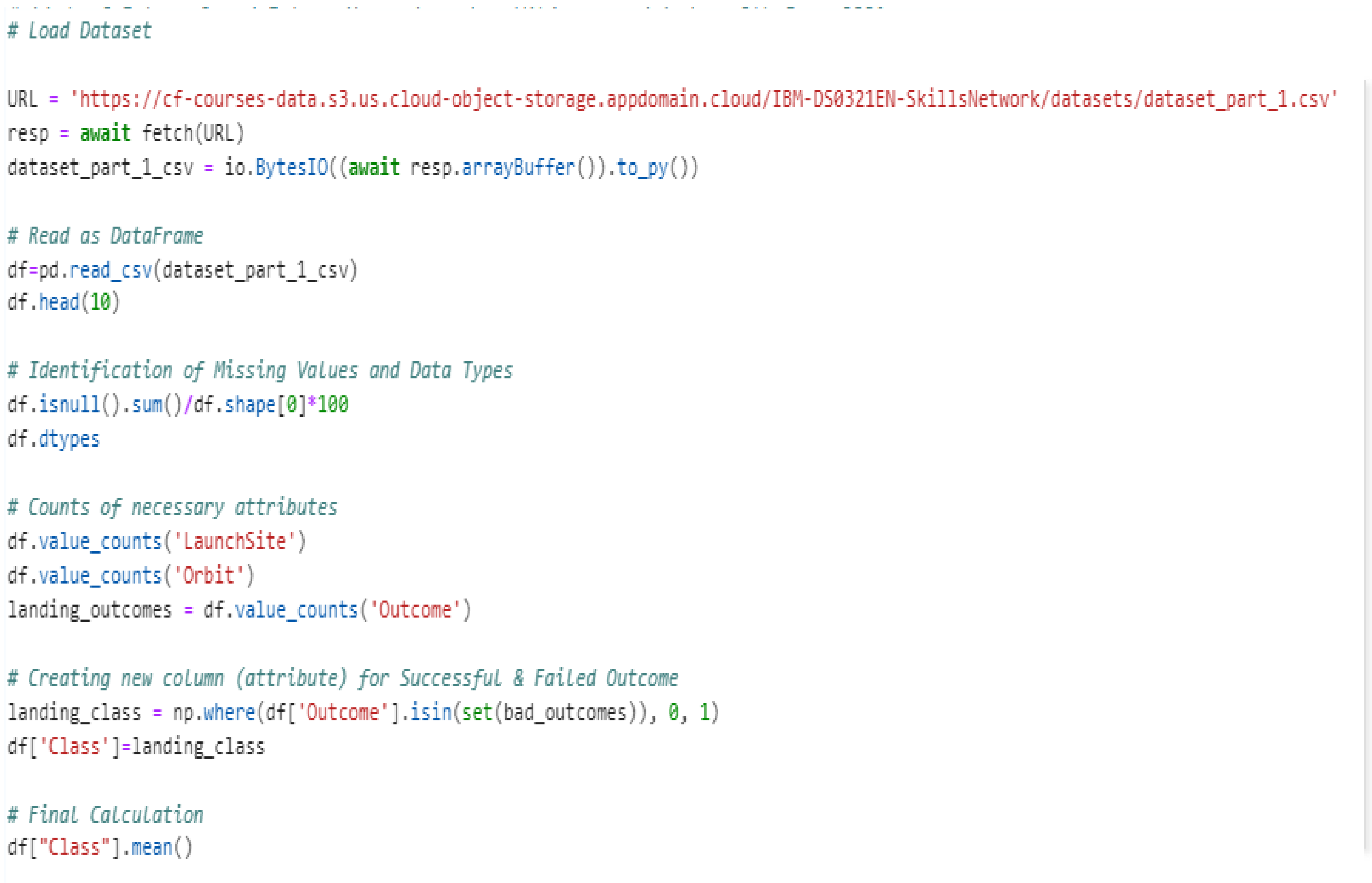


```
# List of Falcon 9 and Falcon Heavy launches Wikipage updated on 9th June 2021

# Retrieve Static URL
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
response = requests.get(static_url)

# Content as Text using BeautifulSoup
soup = BeautifulSoup(response.text)

# Retrieving HTML Table Headers
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]

# Building a Dictionary
launch_dict= dict.fromkeys(column_names)

del launch_dict['Date and time ( )']

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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]

# Dictionary into DataFrame
df=pd.DataFrame(launch_dict)

# Exporting it as CSV
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- Please follow the flow chart on the right and refer to '#' comments for details

- Github URL: https://github.com/jksb1122/Peer-graded-Assignment-Submit-Your-Work-and-Grade-Your-Peers.git



```
# Load Dataset

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv'
resp = await fetch(URL)
dataset_part_1_csv = io.BytesIO((await resp.arrayBuffer()).to_py())

# Read as DataFrame
df=pd.read_csv(dataset_part_1_csv)
df.head(10)

# Identification of Missing Values and Data Types
df.isnull().sum()/df.shape[0]*100
df.dtypes

# Counts of necessary attributes
df.value_counts('LaunchSite')
df.value_counts('Orbit')
landing_outcomes = df.value_counts('Outcome')

# Creating new column (attribute) for Successful & Failed Outcome
landing_class = np.where(df['Outcome'].isin(set(bad_outcomes)), 0, 1)
df['Class']=landing_class

# Final Calculation
df["Class"].mean()
```

EDA with Data Visualization

- Summary of Visualization
  - Scatter Plot (with hue as 'Class'): Scatter plots are used for below to identify any relationship between the two variables
    - Flight Number vs Pay Load Mass (KG)
    - Flight Number vs Launch Site
    - Flight Number vs Orbit Detail
    - Pay Load Mass vs Outcome (orbit type)
  - Bar Chart: Bar charts are used to compare two different groups and their relationships to one another
    - Orbit Mean vs Class
  - Line Graph: Line Graphs are used to get continuous trend or pattern over the years
    - Year vs Average Success Rate
- Github URL: https://github.com/jksb1122/Peer-graded-Assignment-Submit-Your-Work-and-Grade-Your-Peers.git

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
  - 1st: Installed SQL extension to Jupyter Notebook & Connected to SQLite DB
  - 2nd: set the DataFrame using Pandas for csv file from below location

    (https://cf-courses-data.s3.us.doud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/data/Spacex.csv)
  - 3rd: Wrote multiple 'select' queries to conditionally display different output tables
  - 4th: In addition to using conditions, aggregate functions and date parameters displayed adjusted(calculated) output tables
  - Conclusion (in detail):
    - Displayed distinct Launch Sites
    - Only 2 Launch Sites had string "CCA"
    - Total payload mass carried by boosters launched by NASA (CRS) is 45596KG
    - First successful landing outcome in ground pad was achieved: 2017/01/05
    - Total number of mission outcome is 100
    - The Max Payload Mass carried is 15600KG
    - There were two droneship failure in year 2015 both by the same Launch Site
    - There are 34 Successful Outcome between the date 04-06-2010 and 20-03-2017
- Github URL: https://github.com/jksb1122/Peer-graded-Assignment-Submit-Your-Work-and-Grade-Your-Peers.git

Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
  - Initially, 4 Markers and 4 Circles were created for 4 different launch sites, reflecting their corresponding latitudes and longitudes to specifically locate each launch site on the map
  - Then, Marker Cluster was created, for simplicity, to represent all markers for each and every launch record from 4 launch sites, with classification class 1 or 0 representing success or fail (green or red), respectively
  - In addition, a mouse position was added to the map in order to obtain necessary coordinates in case if we wanted to calculate specific distances to proximities or draw lines from specific launch site
  - Lastly, a line was drawn from CCAFS SLC-40 to the nearest coastline as well as the distance to the coastline. Additionally, the distances to near proximities (city, railroad, highway) were calculated as well
- Explain why you added those objects
  - Markers and Circles help viewers easily identify where to locate on the map without chance of missing to locate
  - Marker Cluster, on the other hand, helps to reduce the confusion of overlapped markers sharing same latitude and longitude.
  - A mouse position functions provides users geolocation information at their discretion in order to properly record the latitude and longitude of wanted place on the map in case further analysis/calculation is required using those coordinates
  - Lastly, added lines or calculated distances help viewers better understand what is nearby and how far is each launch site is from desired proximities
- GitHub URL: https://github.com/jksb1122/Peer-graded-Assignment-Submit-Your-Work-and-Grade-Your-Peers.git

Build a Dashboard with Plotly Dash

- **Dashboard has been built having below tools to effectively visualize data and for the use of analysis in finding patterns and trends**
  - **Dropdown List:** consists of 4 different Launch Site locations, with an option to select all, works as a basic categorical input value for Pie Chart and Scatter Plot on the dashboard
  - **Pie Chart:** Effectively visualize actual % (or numbers) of success and failure for all or each Launch Site selected in the dropdown list. This allows users to grasp both the overall statistics and particular information on a certain location.
  - **Scatter Plot**: allows direct visualization of relationship or trend, if any, between Payload Mass and Outcome (success/fail). In addition, Booster Version color category helps viewers better identify and consider possible factor, Booster Version, that may have an impact on outcome results along with Payload Mass
  - **Payload Mass Range Slider:** provides flexibility in retrieving data from scatter plot by adjusting mass range flexibly at user's discretion and needs
  - **In conclusion,** all of above plots and interactions were added to provide easy visualization for users to understand data (Outcome in relation to Launch Sites, Payload Mass, Booster Version) as well as for users to easily adjust variable types and ranges to better obtain desired information
- **Github URL:** https://github.com/jksb1122/Peer-graded-Assignment-Submit-Your-Work-and-Grade-Your-Peers.git

Predictive Analysis (Classification)

- Initial Steps
  - Retrieve the datasets from URLs and convert into Pandas data frame & transform data into Numpy array
  - Split arrayed data into train and test sets
  - Apply train data sets to multiple models – logistic regression, support vector machine (SVM), decision tree, K-nearest neighbor (KNN)
  - Derive algorithms, parameters, and accuracies from running train data

  Github URL:

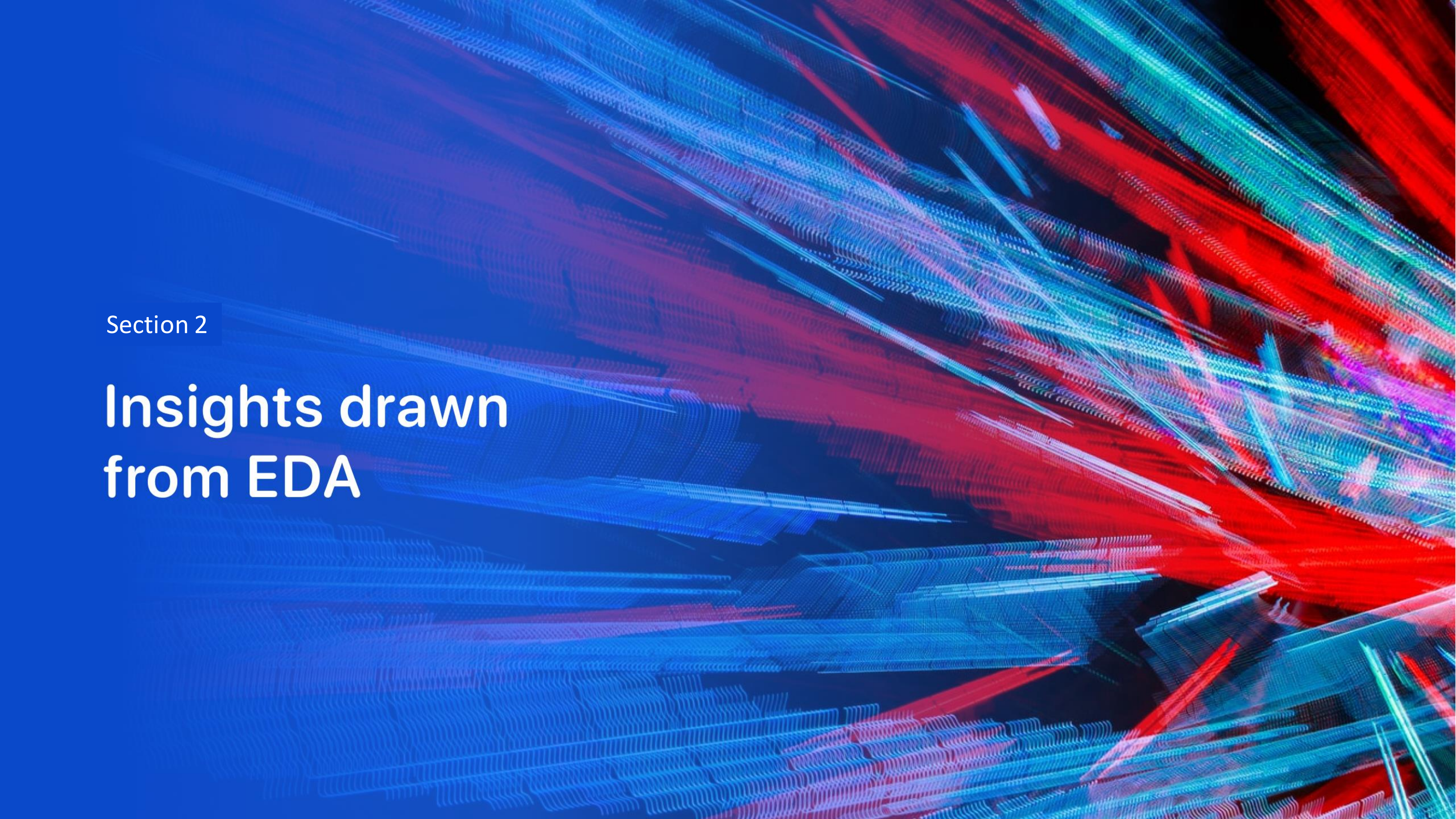
  https://github.com/jksb1122/Peer-graded-Assignment-Submit-Your-Work-and-Grade-Your-Peers.git

- Evaluation & Testing
  - Run each model using Test Data Sets and obtain accuracy scores
  - Generating a confusion matrix to make a prediction using Test Data Sets for each

- Identifying the best model
  - Compare accuracy scores obtained in Evaluation phase from each model
  - Conclusion: each model represents basically the same accuracy of 83.3%, but Decision Tree does have slightly better accuracy, 88.9%; therefore, the decision tree works the best with the given dataset

Results

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

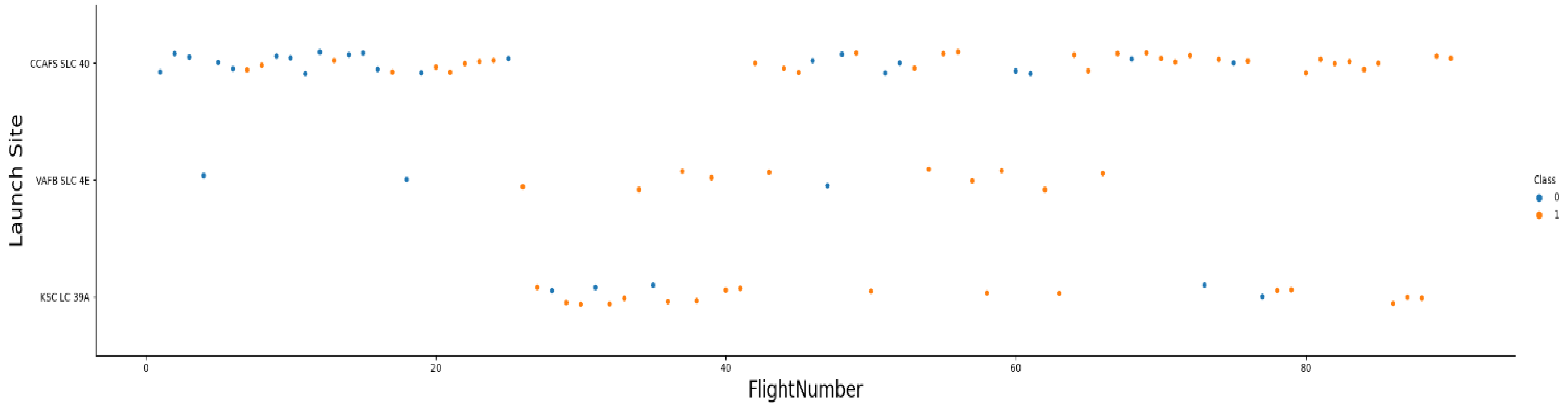


Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

```
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

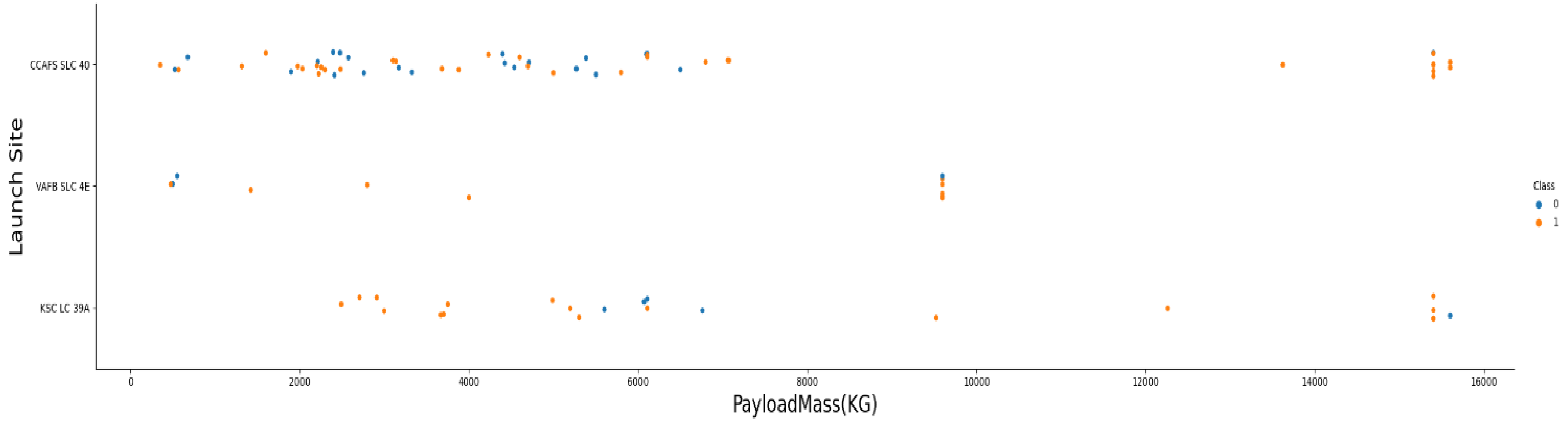


This graph shows as Flight Number increases, the first stage is more likely to land successfully (identified by class 1). And depending on each launch site, there a difference in success rate of landing the first stage successfully.

Payload vs. Launch Site

- This graph shows each lunch site has different ranges of PayloadMass when launching, but most importantly this graph represents there are substantially high chances of having the first stage landing with higher PayloadMass

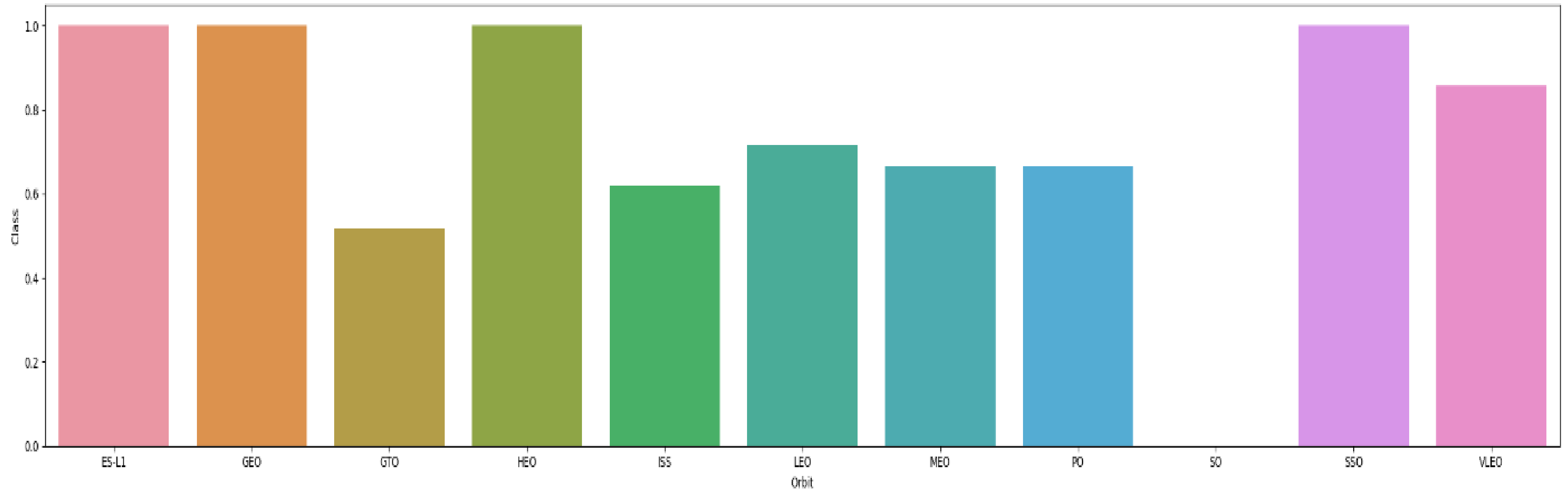
```
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("PayloadMass(KG)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type

- According to the graph, orbit ES1L1, GEO, HEO, SSO has the best success rate while GTO having the lowest success rate

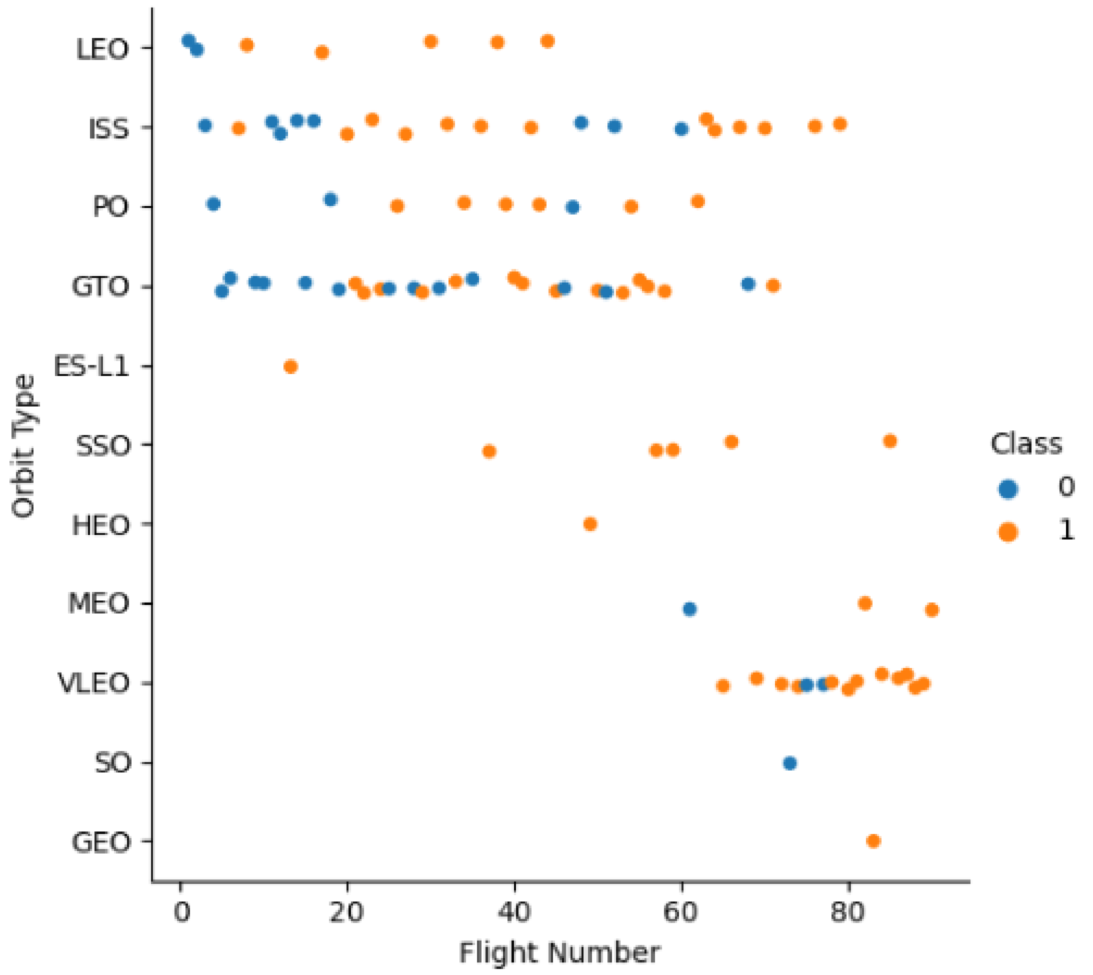
```
suc = df.groupby('Orbit').mean()
suc.reset_index(inplace=True)
sns.barplot(x='Orbit',y="Class",data=suc)
plt.show()
```



Flight Number vs. Orbit Type

- As per the graph, there is a positive relationship between Flight Numbers and Success Rate for basically all orbit types, especially LEO, ISS, PO, GTO – the ones having the most launches. The graph shows enough to conclude it is more likely to succeed in safe first stage landing regardless of orbit types as time progress although there still exists some variations among orbit types

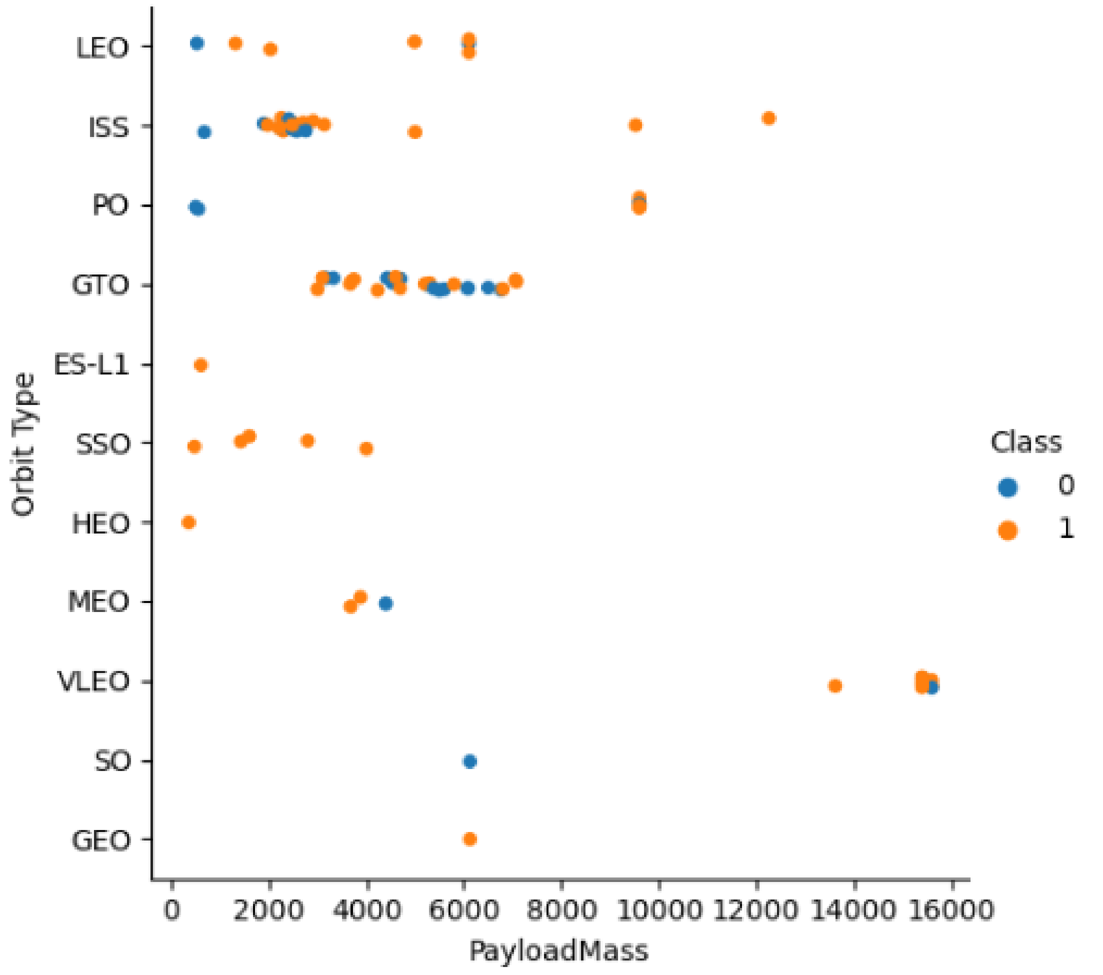
```
sns.catplot(x='FlightNumber',y='Orbit',data=df,hue='Class')
plt.xlabel('Flight Number')
plt.ylabel('Orbit Type')
plt.show()
```



Payload vs. Orbit Type

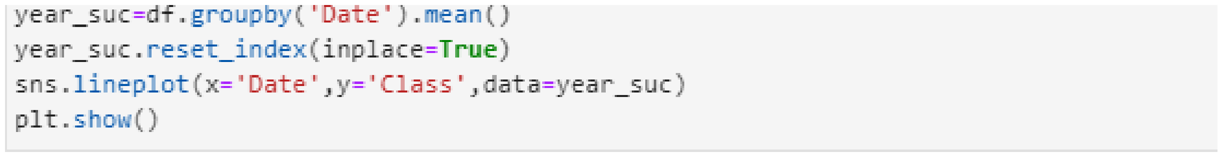
- It is hard to conclude or find any positive relationship between PayloadMass vs overall orbit types. However, it clearly shows for some orbits like LEO, ISS, PO do have higher success rate as PayloadMass gets larger

```
sns.catplot(x='PayloadMass',y='Orbit',data=df,hue='Class')
plt.xlabel('PayloadMass')
plt.ylabel('Orbit Type')
plt.show()
```

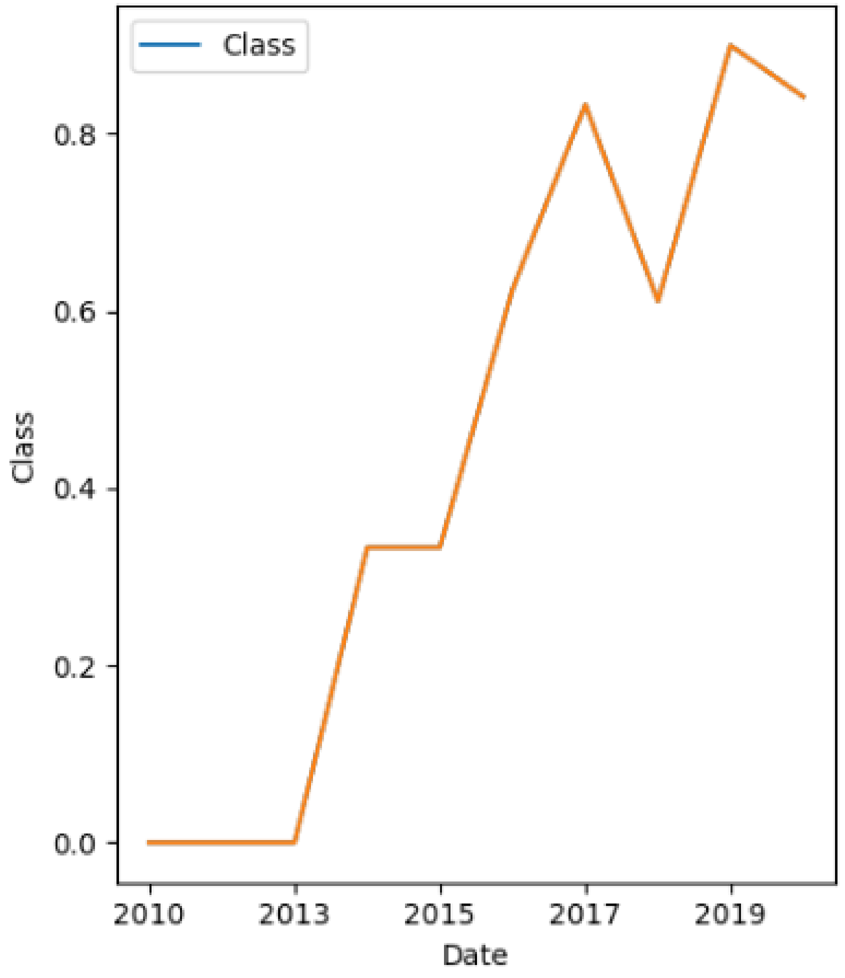


Launch Success Yearly Trend

- Though there are ups and downs, there clearly exists a positive relationship between Date (time) and Class (success Rate). We cannot specify the reasons for such improve, yet we can conclude that the success rate over time is improving as per the graph



```
year_suc=df.groupby('Date').mean()
year_suc.reset_index(inplace=True)
sns.lineplot(x='Date',y='Class',data=year_suc)
plt.show()
```

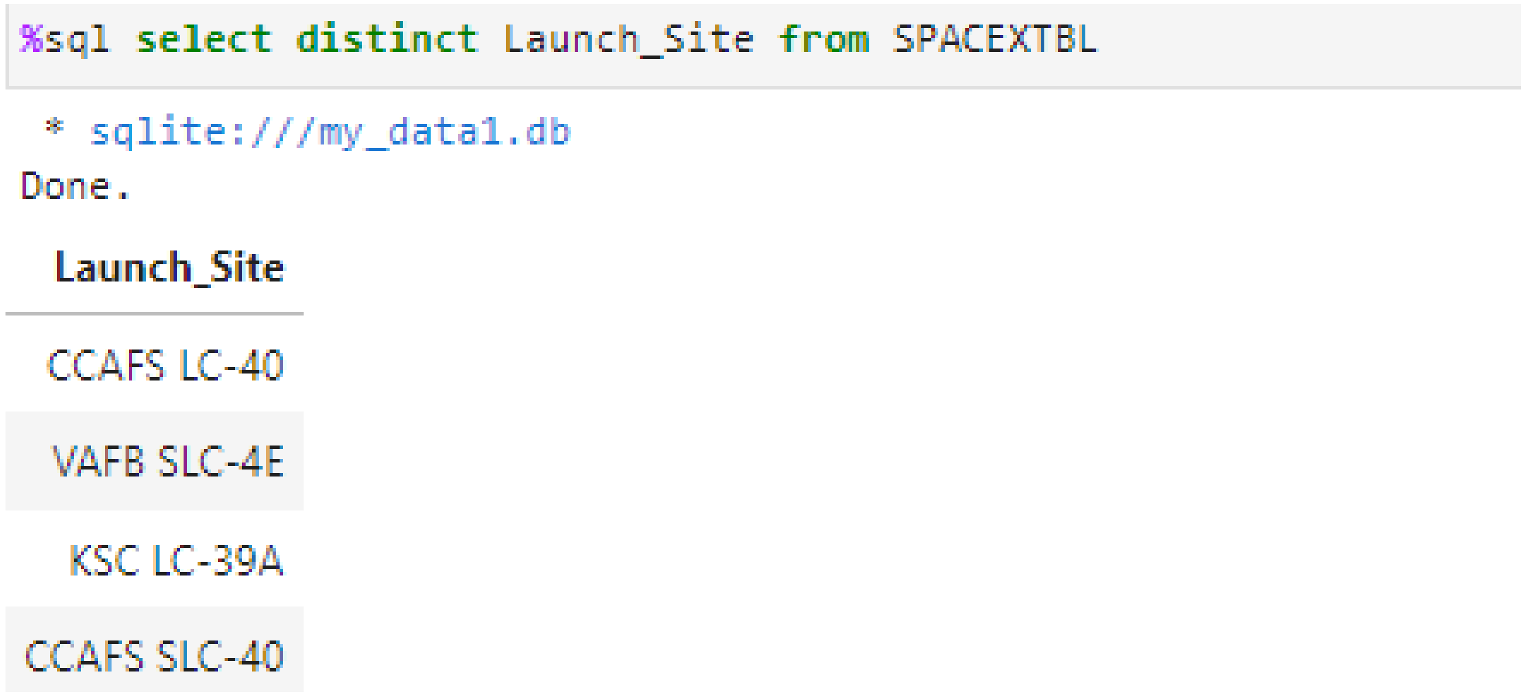


All Launch Site Names

- The four unique launch site names are as follow:

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

- The query in the image retrieve distinct values under the column "Launch_Site" from the table "SPACEXTBL"

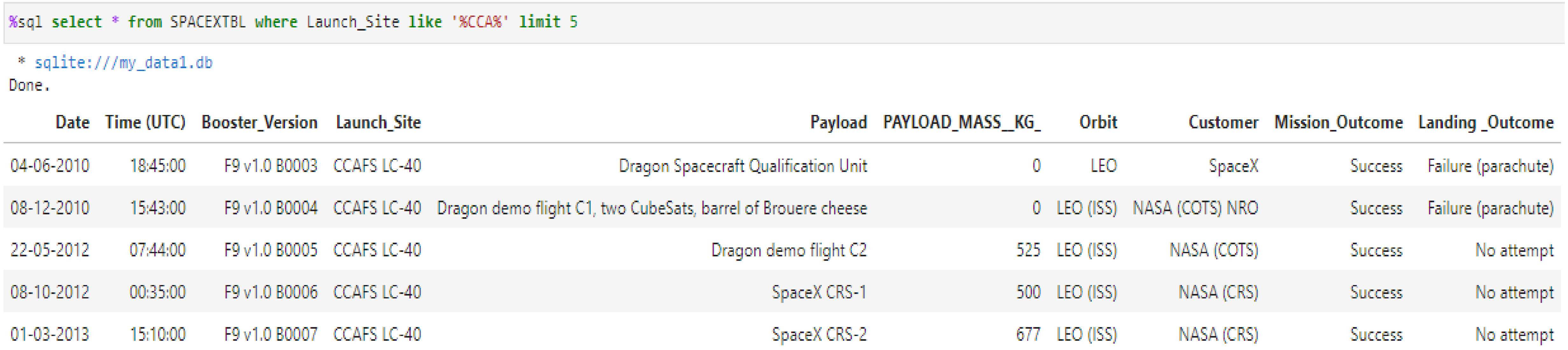


```
%sql select distinct Launch_Site from SPACEXTBL

 * 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'

- The written query in the image is to present 5 row information from SPACEXTBL with conditional setting to filter only for Launch_Sites having 'CCA' as part of their value (string)



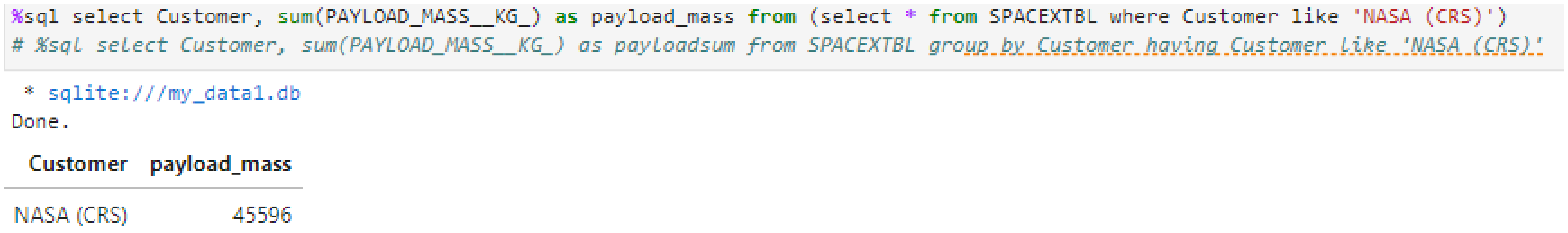
```
%sql select * from SPACEXTBL where Launch_Site like '%CCA%' limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	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

Total Payload Mass

- This query is to display Customer Name and its total sum of payload mass from the edited limited table where only NASA (CRS) is remaining. Therefore, the query result displays information only for NASA (CRS)

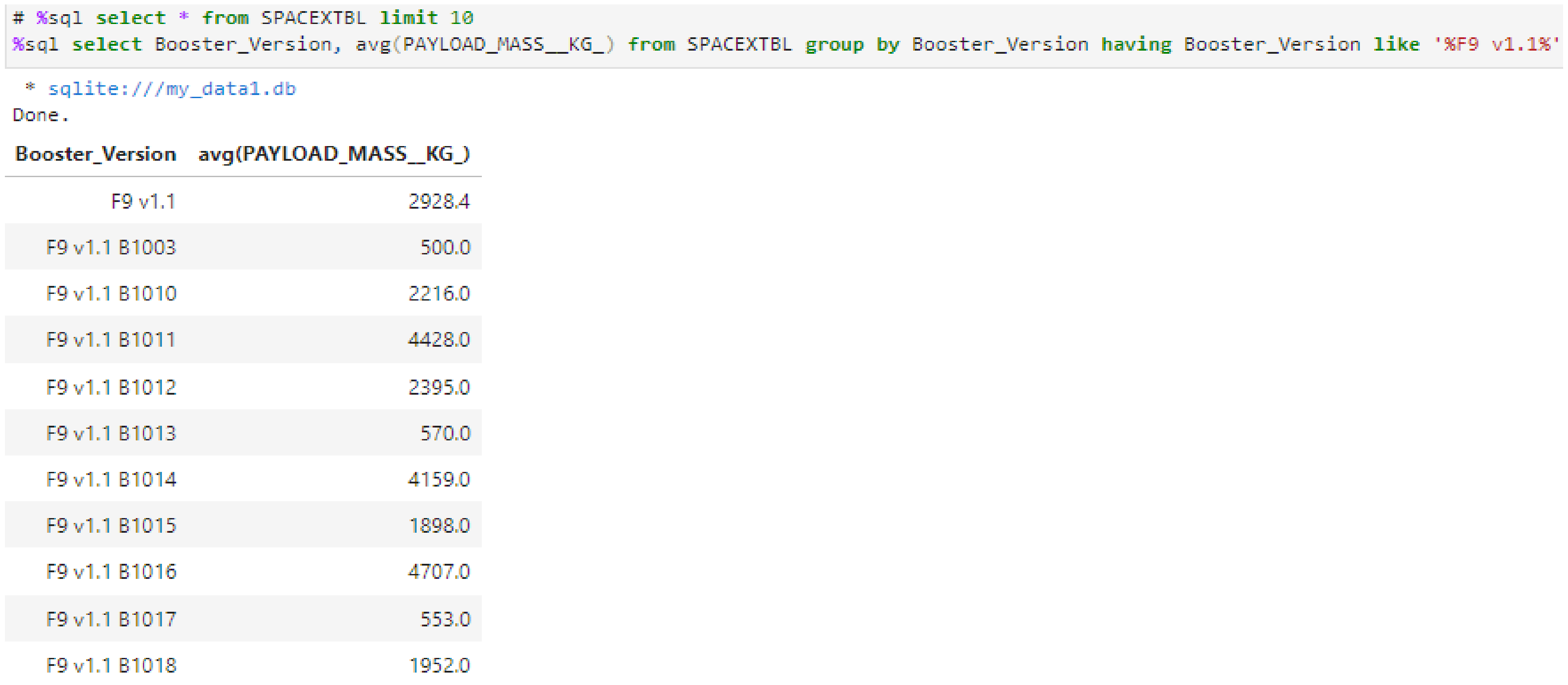


```
%sql select Customer, sum(PAYLOAD_MASS__KG_) as payload_mass from (select * from SPACEXTBL where Customer like 'NASA (CRS)')
# %sql select Customer, sum(PAYLOAD_MASS__KG_) as payloadsum from SPACEXTBL group by Customer having Customer like 'NASA (CRS)'

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

Customer	payload_mass
NASA (CRS)	45596

Average Payload Mass by F9 v1.1

- This query displays average payload mass calculated for each booster version whose name contain F9 v1.1. To find the direct answer for AVG(KG) of F9v1.1, please refer to the first row of the data, which is 2928.4kg



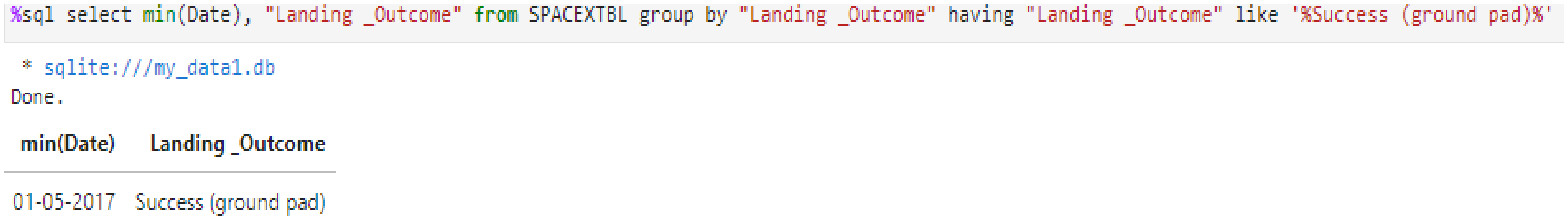
```
# %sql select * from SPACEXTBL limit 10
%sql select Booster_Version, avg(PAYLOAD_MASS__KG_) from SPACEXTBL group by Booster_Version having Booster_Version like '%F9 v1.1%'
```

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

Booster_Version	avg(PAYLOAD_MASS__KG_)
F9 v1.1	2928.4
F9 v1.1 B1003	500.0
F9 v1.1 B1010	2216.0
F9 v1.1 B1011	4428.0
F9 v1.1 B1012	2395.0
F9 v1.1 B1013	570.0
F9 v1.1 B1014	4159.0
F9 v1.1 B1015	1898.0
F9 v1.1 B1016	4707.0
F9 v1.1 B1017	553.0
F9 v1.1 B1018	1952.0

First Successful Ground Landing Date

- The earliest date of first successful landing outcome on ground pad is 01/05/2017
- The aggregate function min(date) returns the earliest date for Landing_Outcome attrubite whose condition is limited to having string containing "Success (ground pad)"



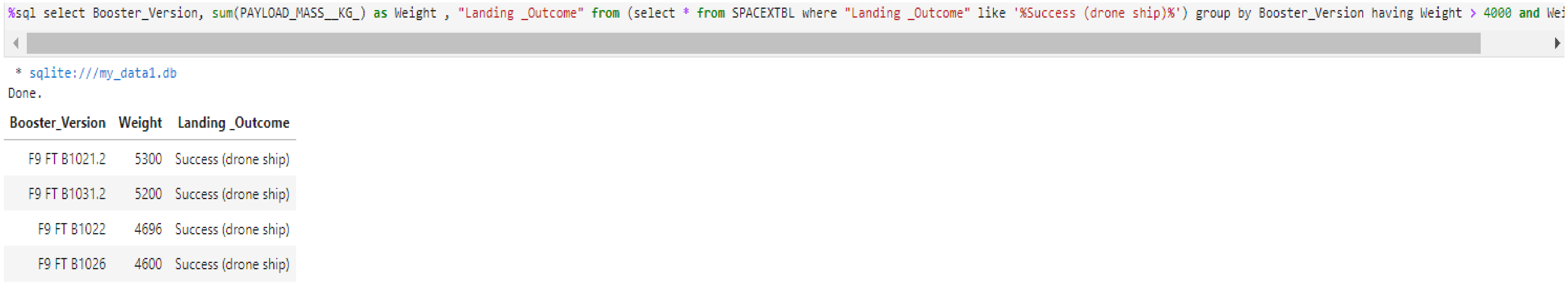
```
%sql select min(Date), "Landing _Outcome" from SPACEXTBL group by "Landing _Outcome" having "Landing _Outcome" like '%Success (ground pad)%'
```

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

min(Date)	Landing _Outcome
01-05-2017	Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

- The booster versions – F9 FT B10212.2,F9 FT B1031.2,F9 FT B1022, F9 FT B1026 have successful drone ship landing outcome for the weight range 4000 ~ 6000
- The query calculates aggregated sum for each booster version with successful landing outcome using drone ship and with wight between 4000 ~ 6000



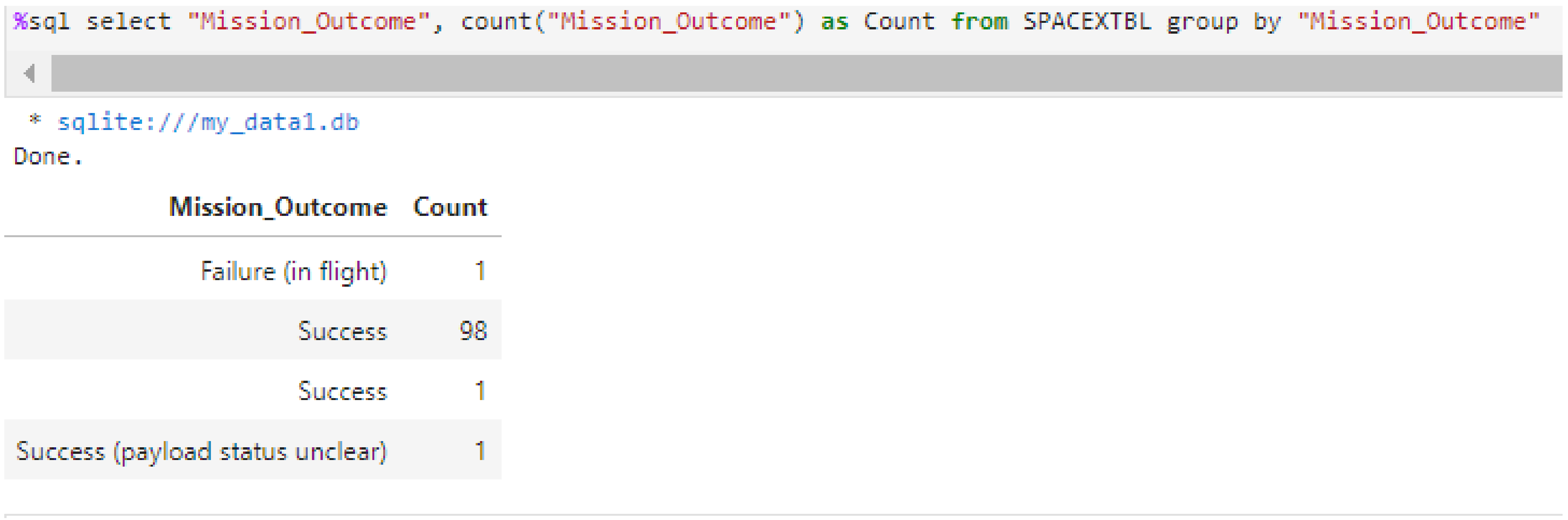
```
%sql select Booster_Version, sum(PAYLOAD_MASS__KG_) as Weight , "Landing _Outcome" from (select * from SPACEXTBL where "Landing _Outcome" like '%Success (drone ship)%') group by Booster_Version having Weight > 4000 and Wei
```

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

Booster_Version	Weight	Landing _Outcome
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- There are 99 successful outcomes and 1 failed outcome
- The query is to group each identical values in "Mission_Outcome" column then present the counts values of those each identical values



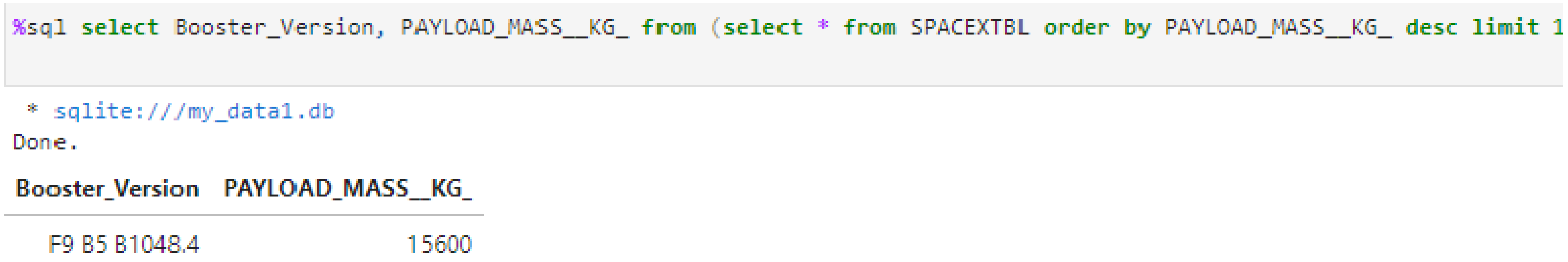
```
%sql select "Mission_Outcome", count("Mission_Outcome") as Count from SPACEXTBL group by "Mission_Outcome"
```

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

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

Boosters Carried Maximum Payload

- The booster named 'F9 B5 B1048.4' carried the most weight (15,600kg)
- The query displays Booster Version name and its mass from the table where there's only 1 value, the maximum, due to sorting condition and limiting to the first value



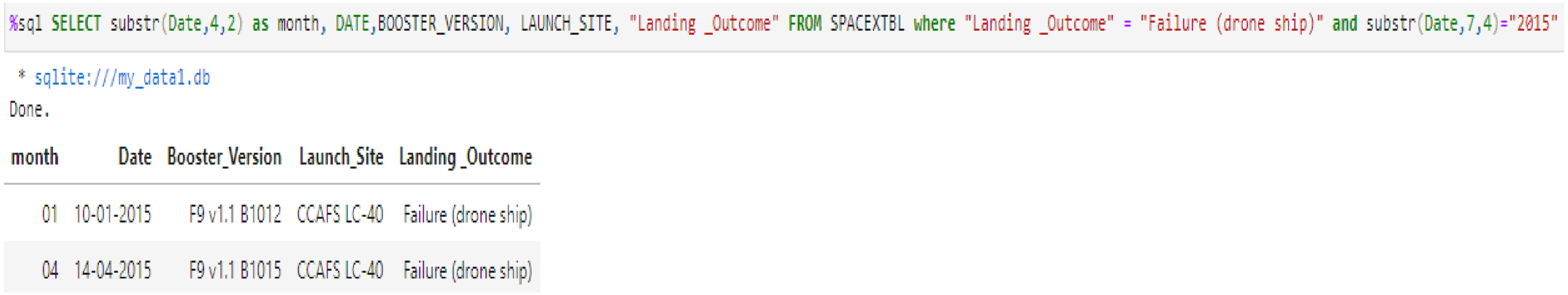
```
%sql select Booster_Version, PAYLOAD_MASS__KG_ from (select * from SPACEXTBL order by PAYLOAD_MASS__KG_ desc limit 1
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600

2015 Launch Records

- Booster Version F9 v1.1 B1012 and F9 V1.1 B1015 at Launch Site CCAFS LC-40 have landing outcome as fail (drone ship) in year 2015
- As SQLite doesn't support monthnames, substr function was used to derive corresponding month number and limiting the data condition to 2015 along with another condition to set the data only to filter for landing outcome = failure (drone ship)



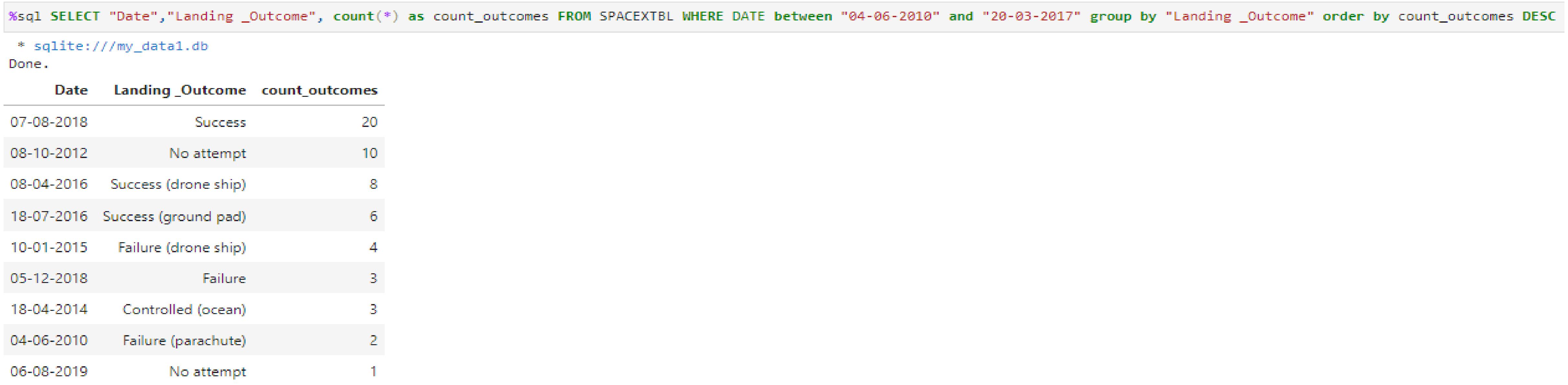
```
%sql SELECT substr(Date,4,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, "Landing _Outcome" FROM SPACEXTBL where "Landing _Outcome" = "Failure (drone ship)" and substr(Date,7,4)="2015"
```

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

month	Date	Booster_Version	Launch_Site	Landing _Outcome
01	10-01-2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	14-04-2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- The ranks are Success, No attempt, Success (drone ship), Success (ground pad), Failure (drone ship), Failure, Controlled (ocean), Failure (parachute), N
- The Date, Landing_Outcome, and Count of Landing Outcomes are displayed once conditioned for filter to present values between set date range, 04-06-2010 ~ 20-03-2017, and sorted from big to small



```
%sql SELECT "Date","Landing _Outcome", count(*) as count_outcomes FROM SPACEXTBL WHERE DATE between "04-06-2010" and "20-03-2017" group by "Landing _Outcome" order by count_outcomes DESC
```

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

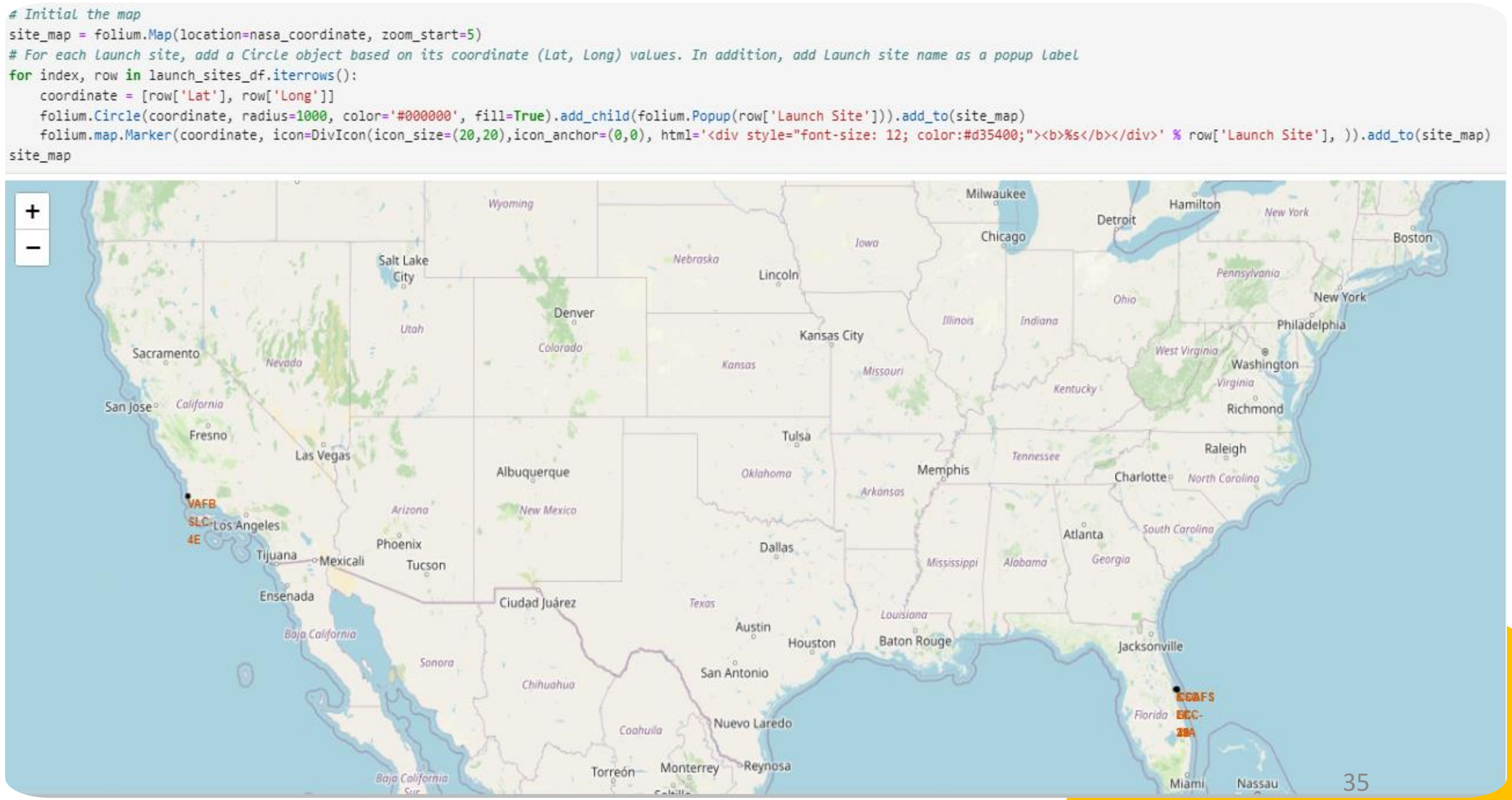
Date	Landing _Outcome	count_outcomes
07-08-2018	Success	20
08-10-2012	No attempt	10
08-04-2016	Success (drone ship)	8
18-07-2016	Success (ground pad)	6
10-01-2015	Failure (drone ship)	4
05-12-2018	Failure	3
18-04-2014	Controlled (ocean)	3
04-06-2010	Failure (parachute)	2
06-08-2019	No attempt	1

Section 3

Launch Sites Proximities Analysis

Locations of all 4 launch sites of SpaceX

- A circle and a marker is added to identify each location on the map for all 4 locations of launch sites using their coordinates

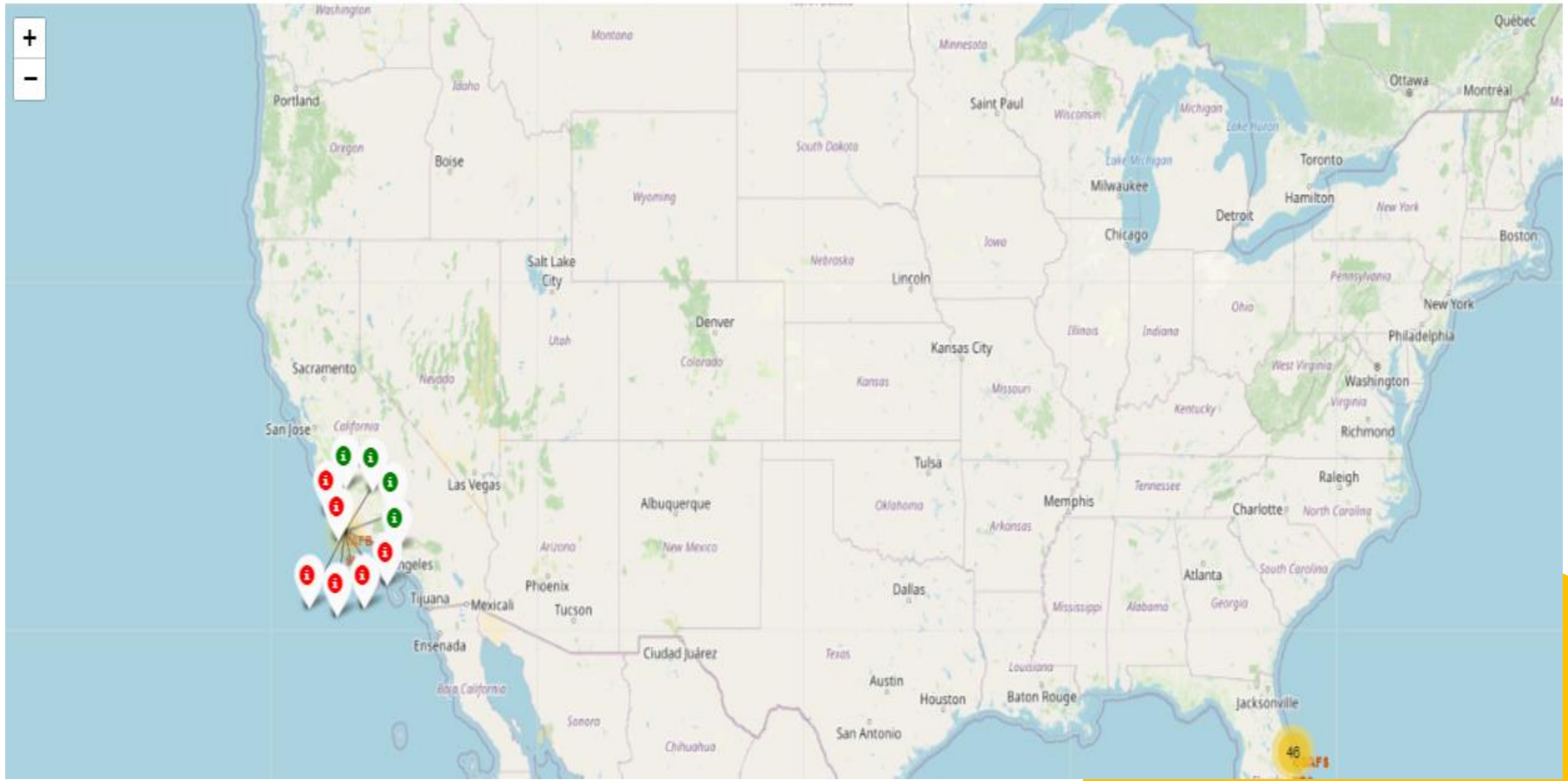


```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label
for index, row in launch_sites_df.iterrows():
    coordinate = [row['Lat'], row['Long']]
    folium.Circle(coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup(row['Launch Site'])).add_to(site_map)
    folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % row['Launch Site'], )).add_to(site_map)
site_map
```

Success or Failure of each launch site

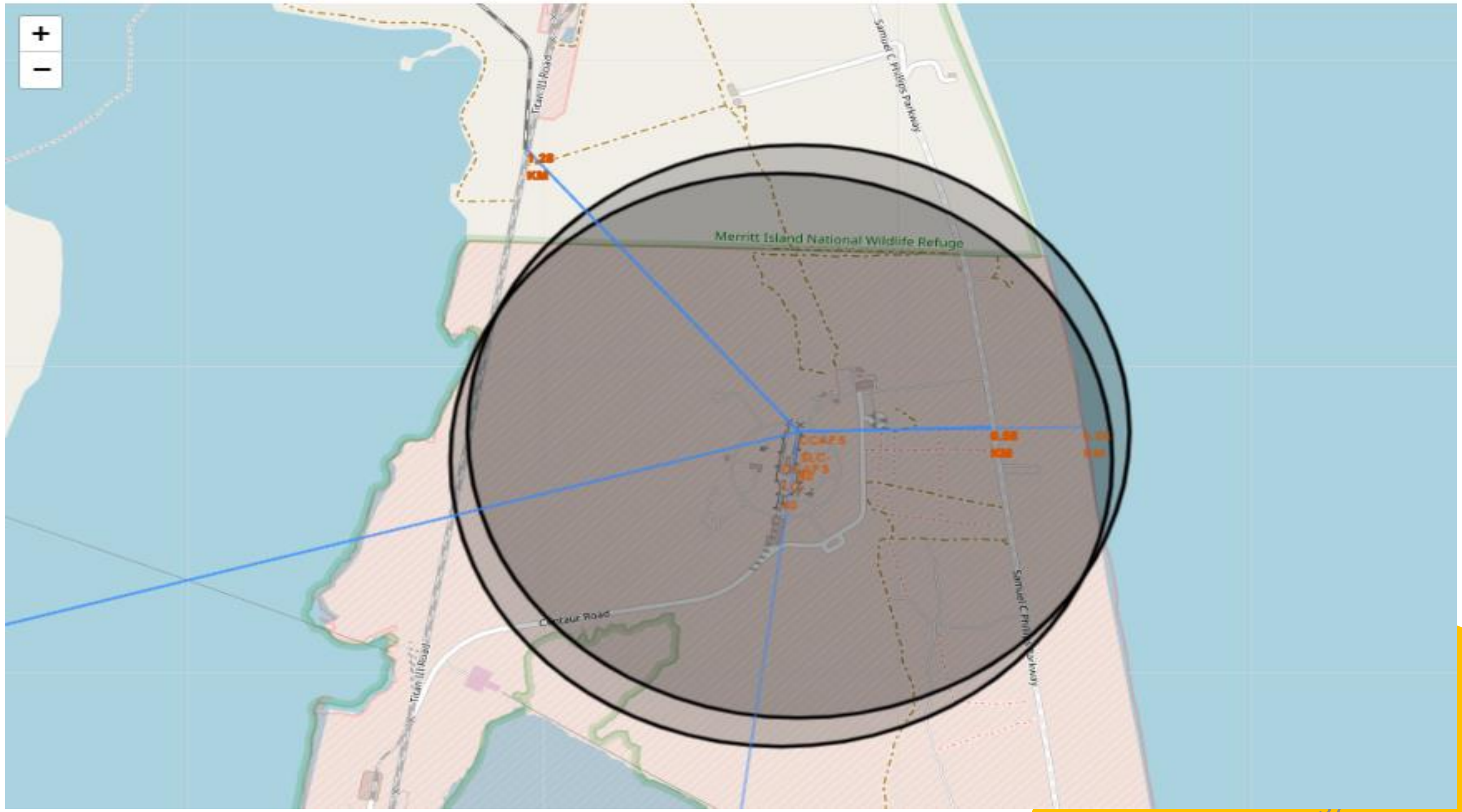
- By clicking on each launch site location, a marker-cluster will pop using two color indicators, green as success and red as failure, to easily visualize cases of success/failure

```
for index, row in spacex_df.iterrows():
    coordinate = [row['Lat'], row['Long']]
    folium.Marker(coordinate, icon=folium.Icon(color='white', icon_color=row['marker_color'])).add_to(marker_cluster)
site_map
```

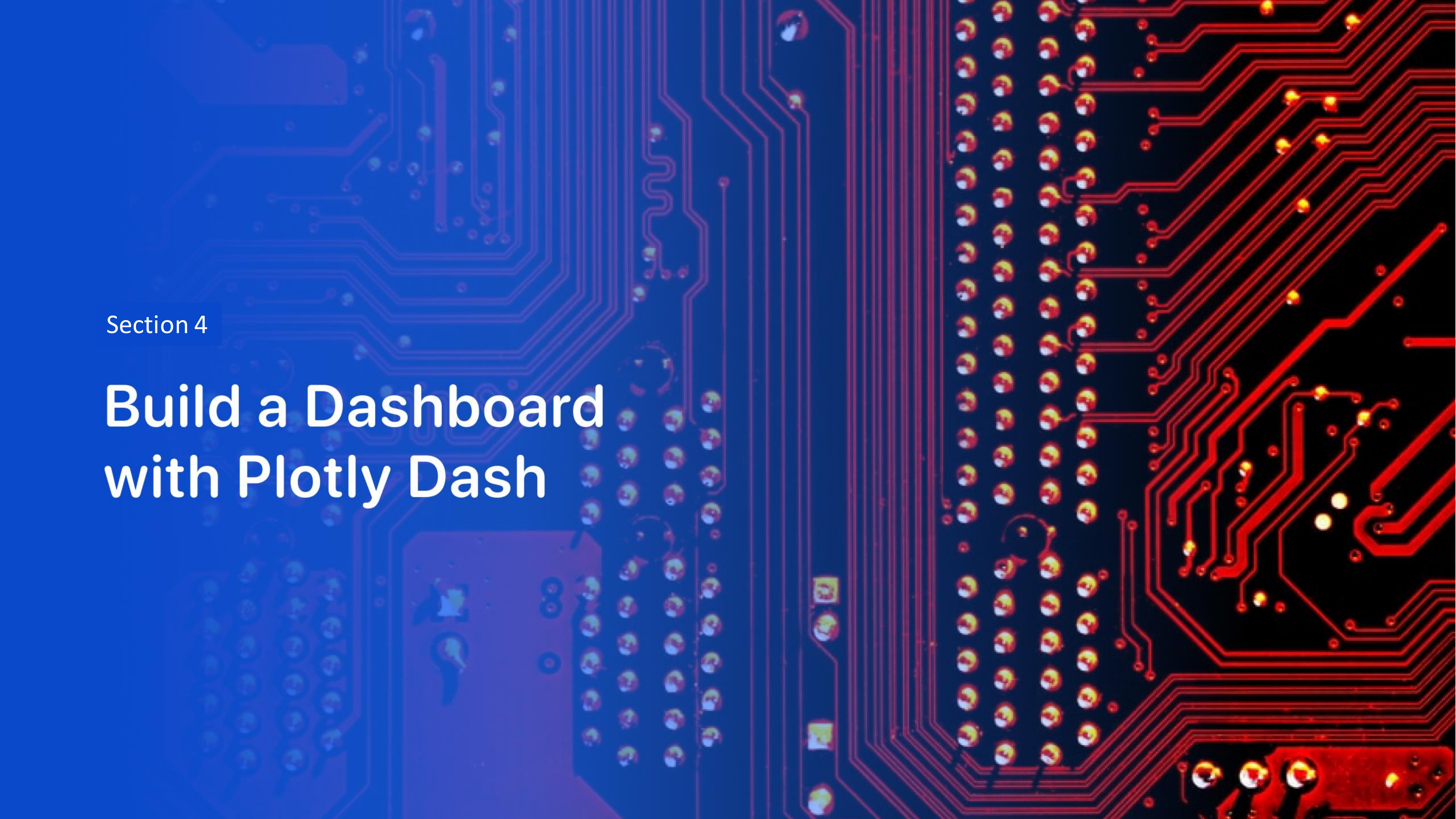


Locations and Distances of near proximities of CCAFS SLC-40 launch site

- Using the a latitude and a longitude of the object, launch site in this case, and of the target, railroad for example as specified on the image right, a line can be drawn and the distance calculated so that it can help views identify more detailed geological information about the launch sites of SpaceX

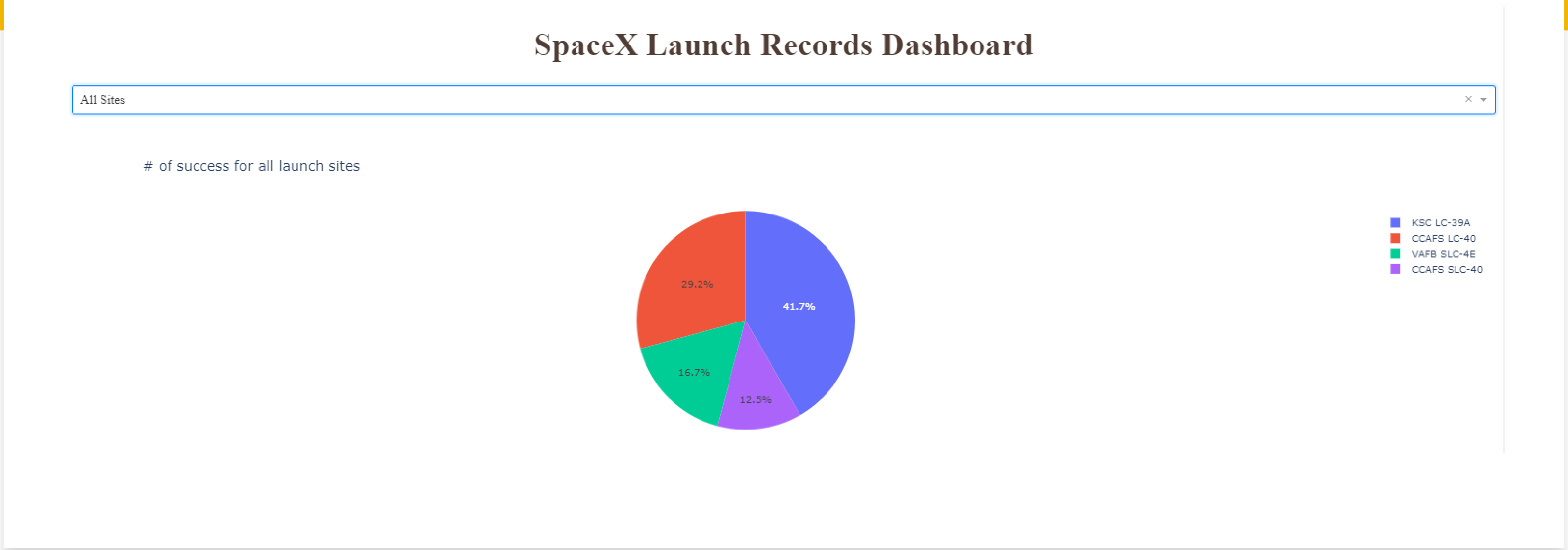


Titan III Road
Samuel C Phillips Parkway
KM
Merritt Island National Wildlife Refuge
CCAFS
SLC-
KM
Centaur Road
Titan III Road
Samuel C Phillips Parkway



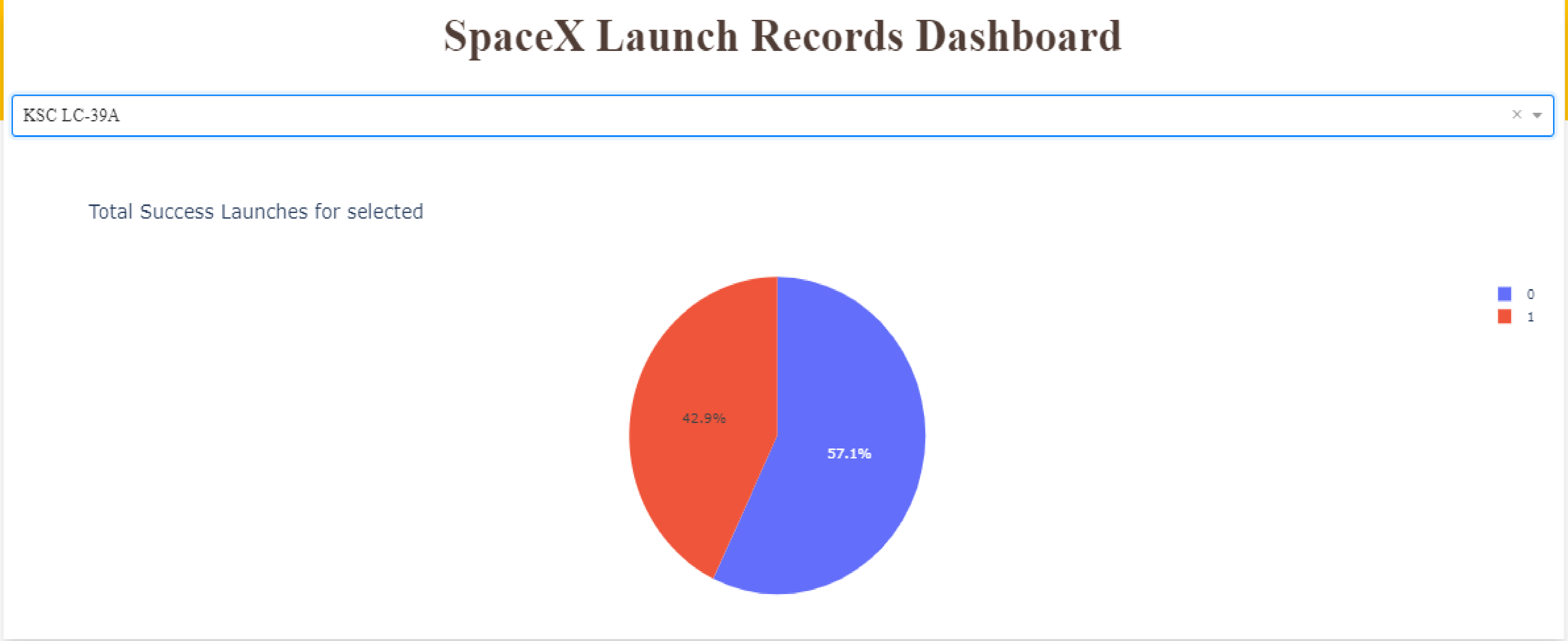
Section 4

Build a Dashboard with Plotly Dash



The Most Successful Launch Site

- The graph clearly indicates Blue has the biggest share of the pie. That is, blue launch site, KSC LC-39A, has the most successful launches among all 4 sites, having 41.7% of all successful launches in total launches

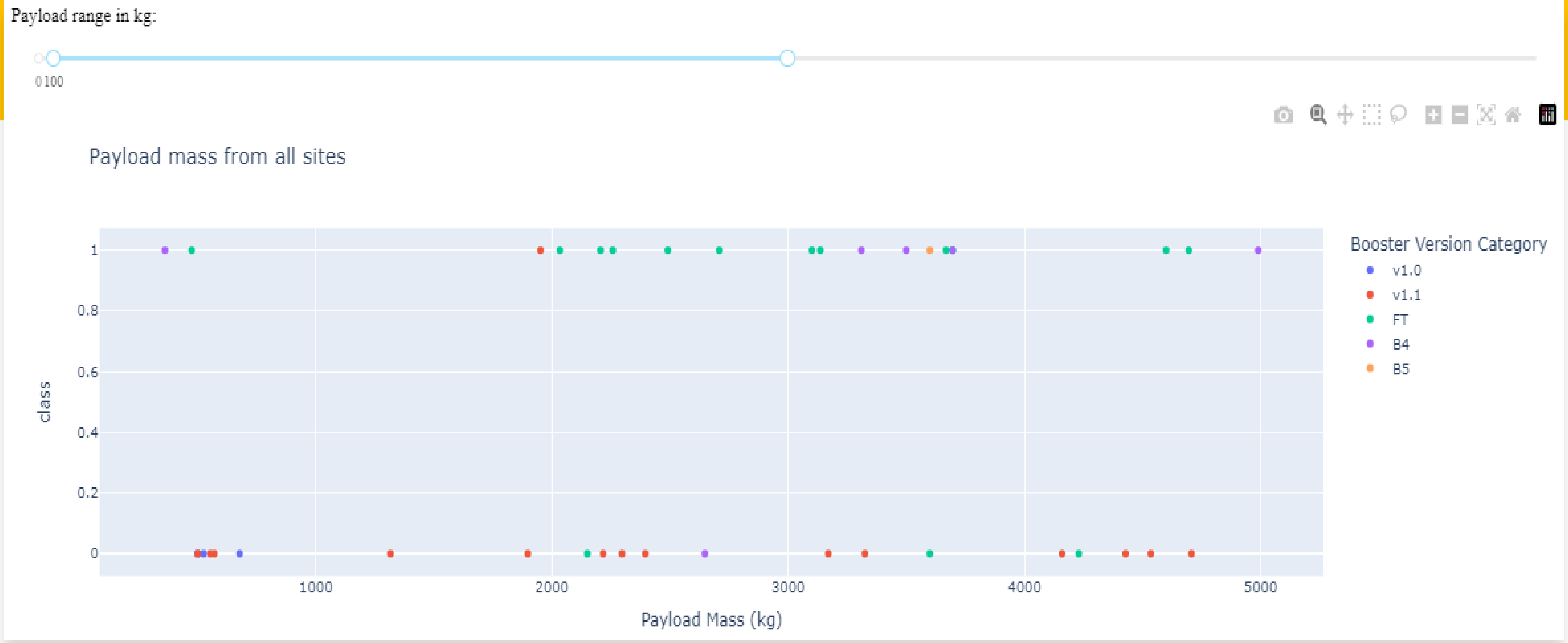

SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for selected

Most Successful Launch Site in Detail

- Although KSC LC-39A launch site is the most successful among all sites, its success rate of 57.1% is not dramatically higher than its failure rate of 42.9%. Therefore, there may be a necessity to examine more data in determining what site is the best



Success Rate for low range payload mass (0 ~ 5000kg range)

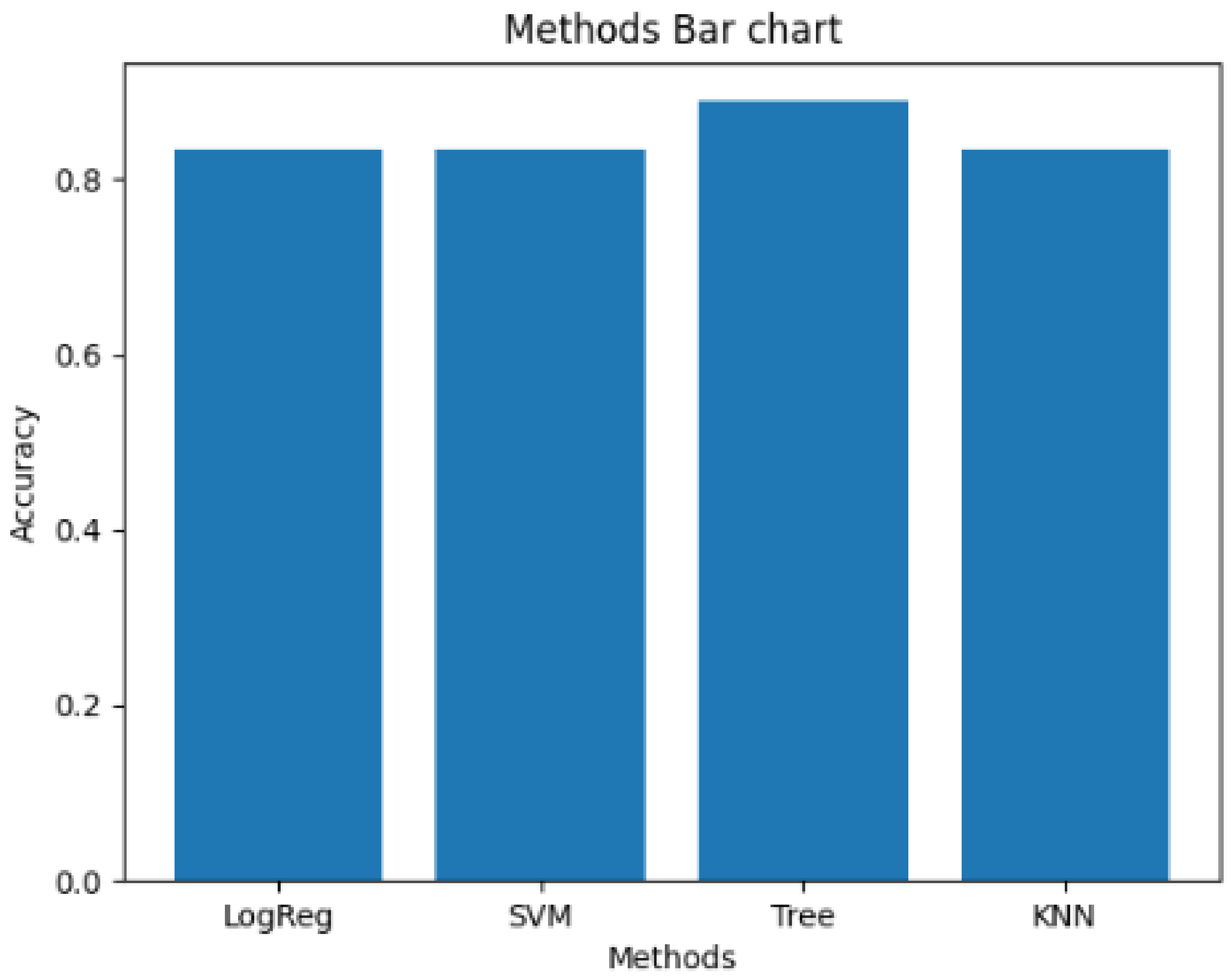
- According to the graph, weight does not seem to have a strong relationship with success or fail during low payload mass range. However, the color code representing Booster Version Category does indicate that Booster FT has the highest success rate while Booster V1.1 has the lowest success, or highest fail, rate.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

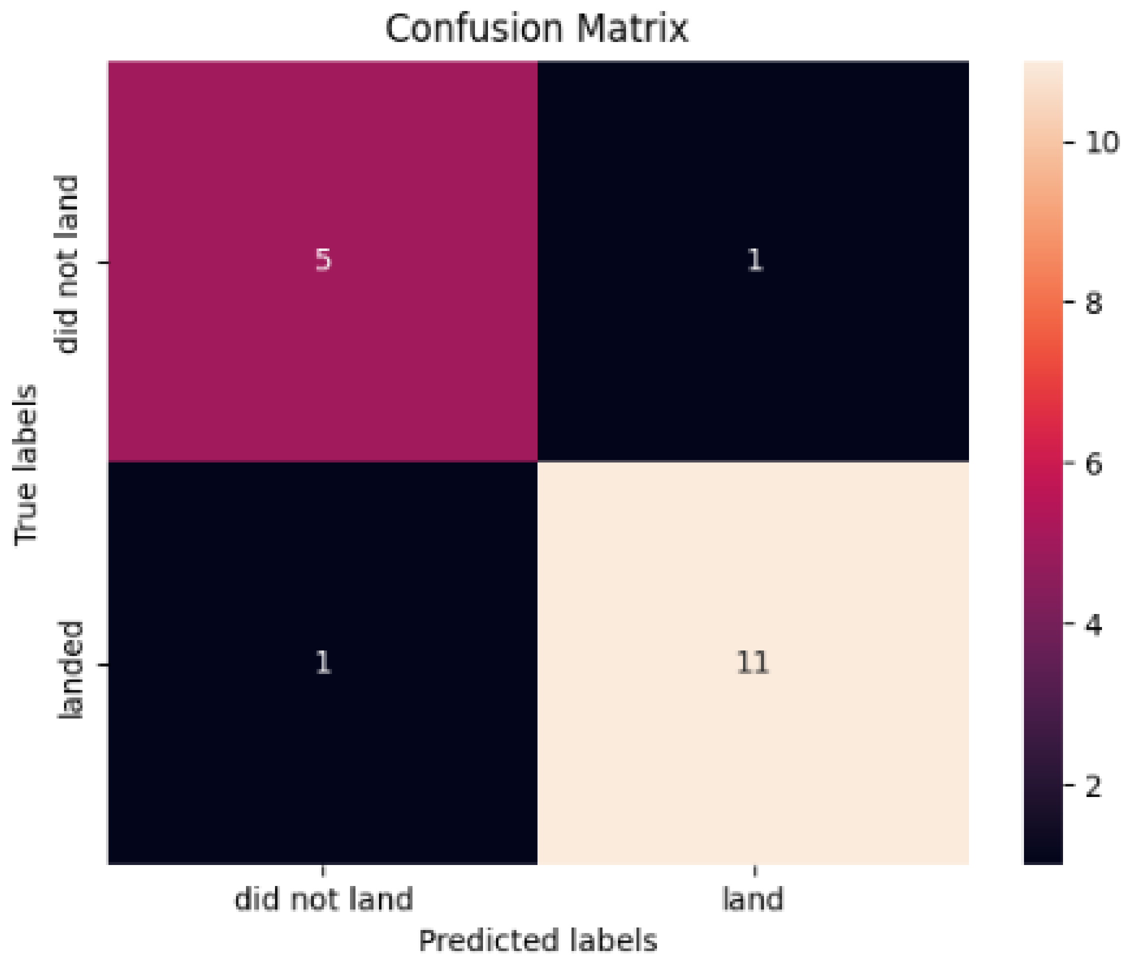
- The Tree Method has the highest classification accuracy rate according to the Bar Chart provided



Confusion Matrix

- The Tree Method is the best performing among all methods as per its Confusion Matrix. The tree has the highest number for true negative and true positives, which is 16, while other methods are having only 15

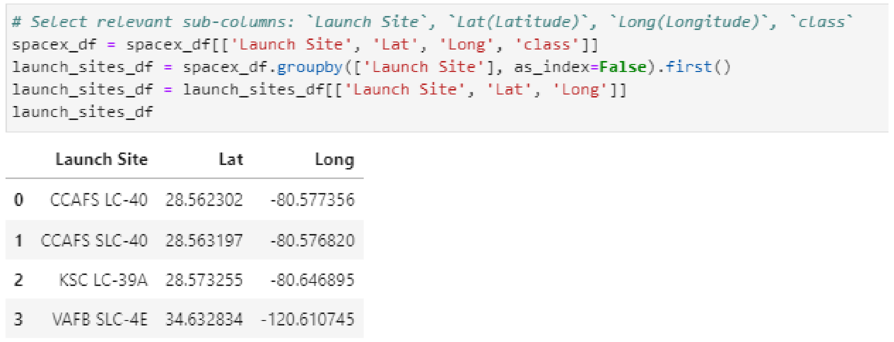
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- **Based on the collected data and analysis performed previously, we can conclude that:**
  - Overall, the success rate of landing first stage will increase as time passes with having more launches
  - It is even more likely to have successful first stage landing if Payload Mass is heavier
  - In terms of Orbits, ES1L1, GEO, HEO, SSO has the best success rate
  - Among 4 available launch sites, KSC LC-39A performs the best, taking up 41.7% of the total success with individual success rate of 57.1%
  - The tree classification method works best in prediction, providing better confusion maxtrix result and accuracy in comparason to other 3 motheds - SVM, LogReg, KNN

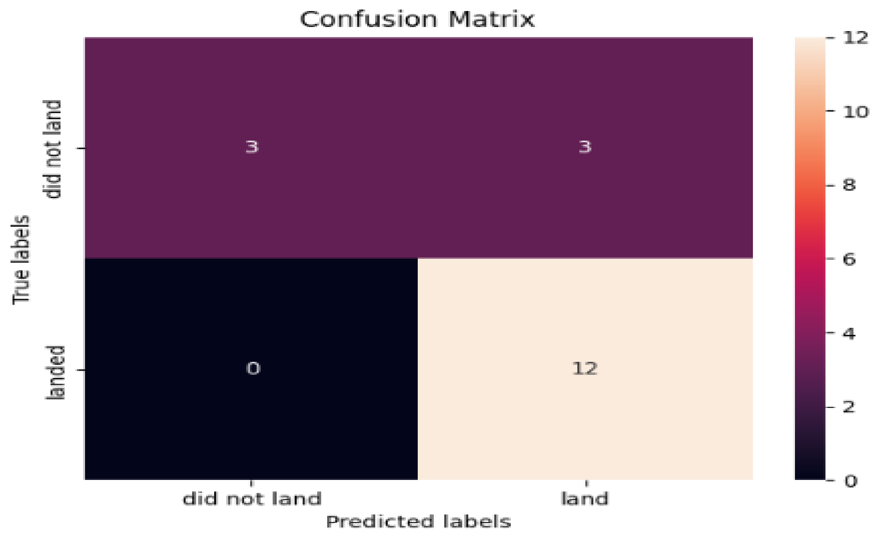
Appendix



```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

Coordinates used for Folium Map (slide 35)



Confusion Matrix of LogReg, SVM, KNN – all same (slide 44)

Thank you!