IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Williams A. Owusu
April 6,2023

Outline

Executive Summary

- Summary of methodologies
  - ✓ Collection of the data
  - ✓ Cleaning of the data
  - ✓ Exploratory Data Analysis with data visualization
  - ✓ Exploratory data analysis with SQL
  - ✓ Interactive map with Folium
  - ✓ Dashboard with Plotly Dash
  - ✓ Predictive analysis
- Summary of all results
  - ✓ Results of the exploratory data analysis
  - ✓ Screenshots of the outcome of the analysis
  - ✓ Results of the Predictive analysis

Introduction

- Project background and context

Space X advertises Falcon 9 rocket launches on its website for 62 million dollars; other providers charge up to 165 million dollars each; much of the savings is due to Space X's ability to reuse the first stage. As a result, if we can predict whether the first stage will land, we can estimate the cost of a launch. This data can be used if another company wants to compete with Space X for a rocket launch. I therefore created a machine learning pipeline to predict if the Falcon 9 first stage will land successfully.

- Problems wealth answering:
  - ✓ What factors courses the rocket to land successfully?
  - ✓ What conditions must the SpaceX company put in place to ensure the successful landing of the Falcon 9 rocket ?

Section 1

Methodology

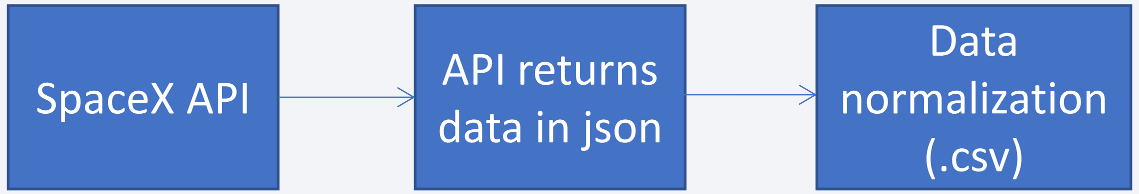
Methodology

Executive Summary

- Data collection methodology:
  - ✓SpaceX API and Web Scrapping from wiki page
- Perform data wrangling/Cleaning
  - ✓One Hot Encoding used fro data transformation
  - ✓Dropped unwanted columns
- Perform exploratory data analysis (EDA) using visualization and SQL
  - ✓Graphs: scatter, line, and bar graphs
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection Description

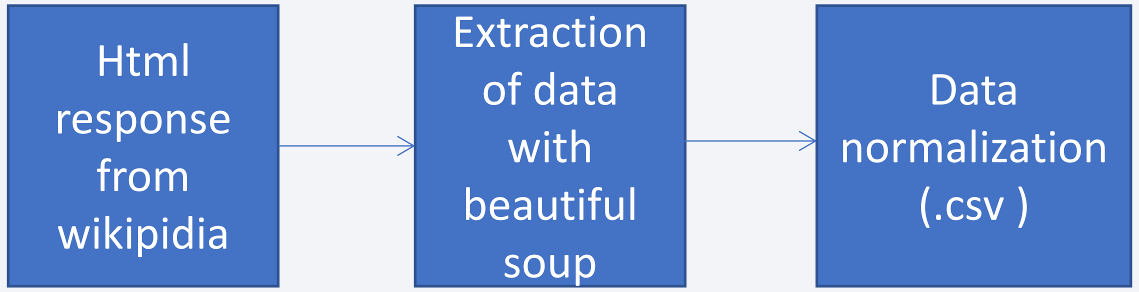
SpaceX API :



Data used was gathered from the SpaceX REST API. Information obtained include, landing outcomes, payload mass, rocket used, and, launch specifications.

Data was also obtained from Wikipedia through web scrapping using beautiful soup.

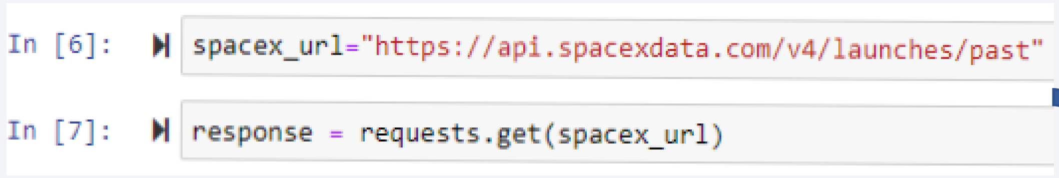
Web Scrapping :


Html response from wikipidia
Extraction of data with beautiful soup
Data normalization (.csv)

Data Collection – SpaceX API

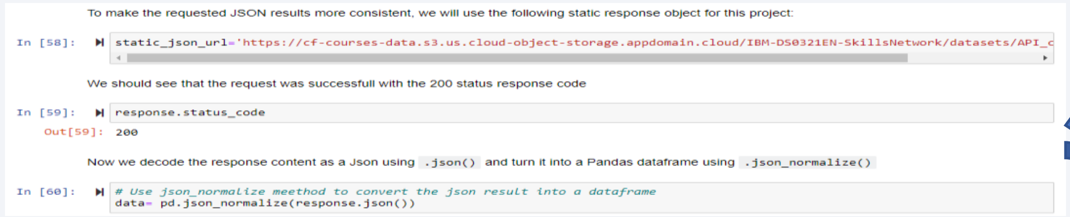
STEP BY STEP PROCEDURE:

1. Requesting SpaceX API Response



```
In [6]:  spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]:  response = requests.get(spacex_url)
```

2. Convert the Response to .json file



To make the requested JSON results more consistent, we will use the following static response object for this project:

```
In [58]:  static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_c
```

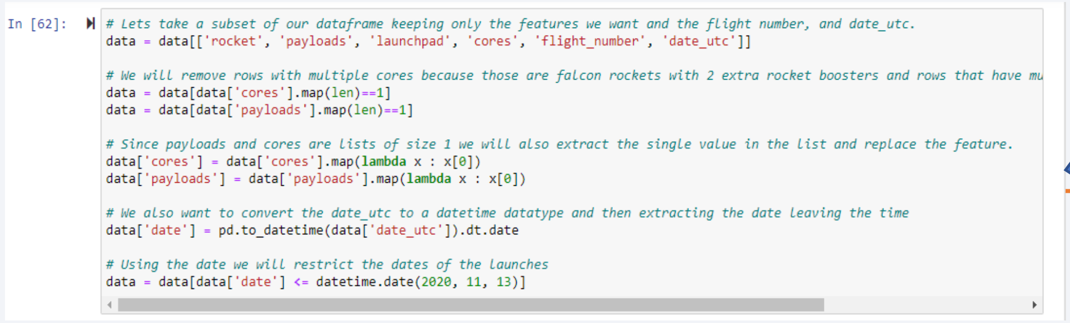
We should see that the request was successfull with the 200 status response code

```
In [59]:  response.status_code
Out[59]:  200
```

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

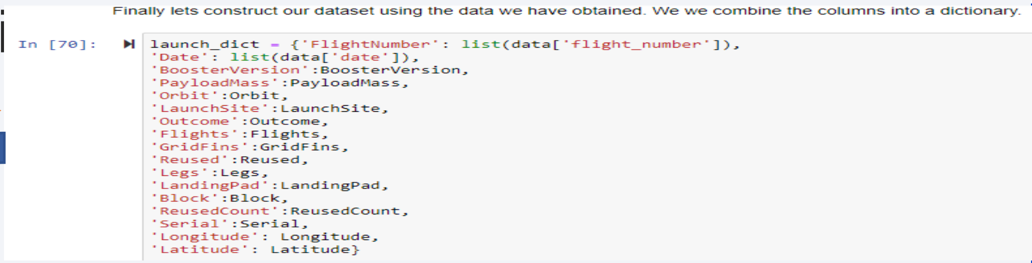
```
In [60]:  # Use json_normalize meethod to convert the json result into a dataframe
          data= pd.json_normalize(response.json())
```

3. Get Information About Launches using API



```
In [62]:  # 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 mu
          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)]
```

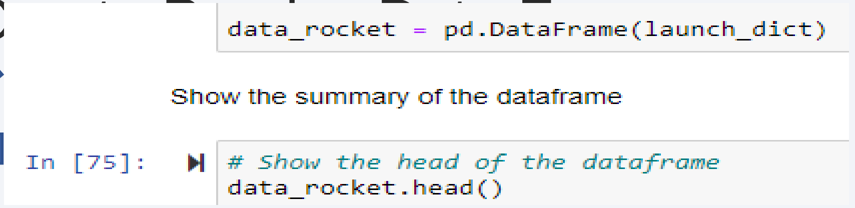
4. Construct Dataset and assign to dicti



Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
In [70]:  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}
```

5. C

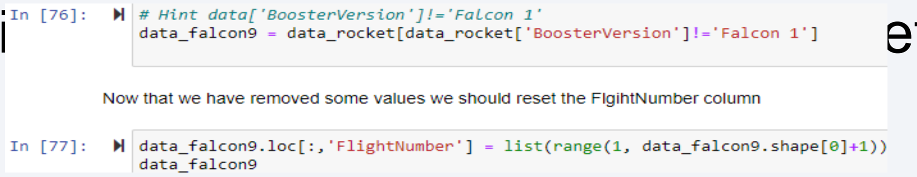


```
data_rocket = pd.DataFrame(launch_dict)
```

Show the summary of the dataframe

```
In [75]:  # Show the head of the dataframe
          data_rocket.head()
```

6. Fi et



```
In [76]:  # Hint data['BoosterVersion']!='Falcon 1'
          data_falcon9 = data_rocket[data_rocket['BoosterVersion']!='Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

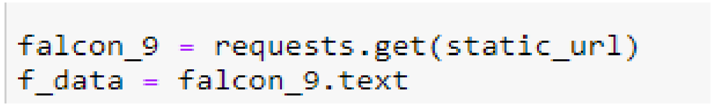
```
In [77]:  data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
          data_falcon9
```

Github URL: Click here

Data Collection - Scraping

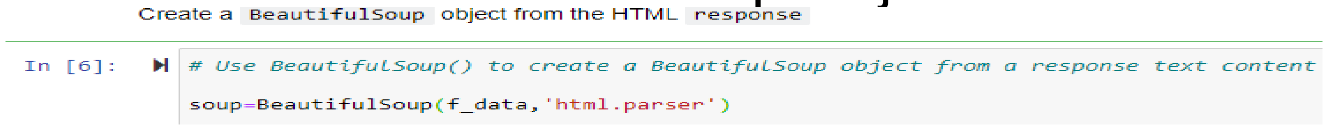
Procedure:

1.Request response from HTML page



```
falcon_9 = requests.get(static_url)
f_data = falcon_9.text
```

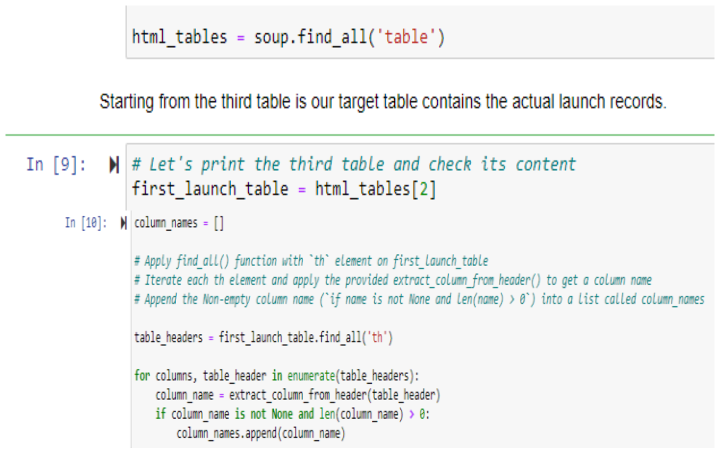
2. Create BeautifulSoup Object



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

```
In [6]:  # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
         soup=BeautifulSoup(f_data,'html.parser')
```

3. Extract Column names

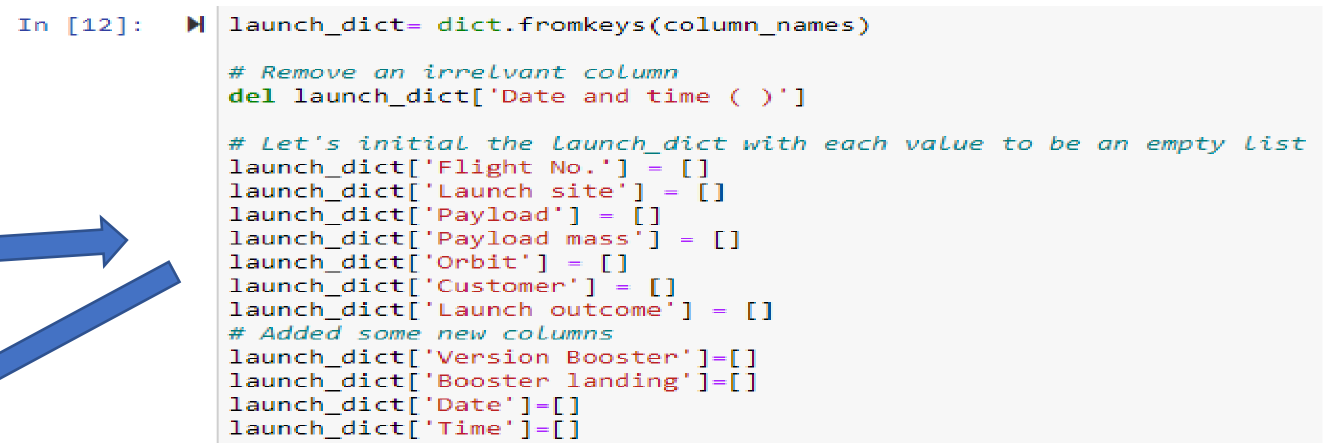


```
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

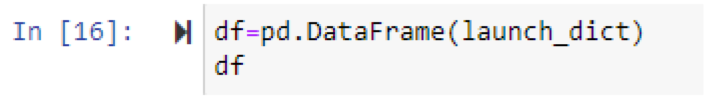
```
In [9]:  # Let's print the third table and check its content
         first_launch_table = html_tables[2]

In [10]: 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

         table_headers = first_launch_table.find_all('th')

         for columns, table_header in enumerate(table_headers):
             column_name = extract_column_from_header(table_header)
             if column_name is not None and len(column_name) > 0:
                 column_names.append(column_name)
```

4. Create Dictionary From extracted columns



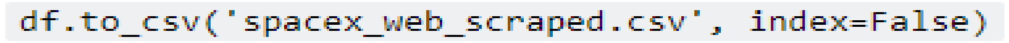
```
In [12]:  launch_dict= dict.fromkeys(column_names)

          # Remove an irrelvant column
          del launch_dict['Date and time ( )']

          # Let's initial the launch_dict with each value to be an empty list
          launch_dict['Flight No.'] = []
          launch_dict['Launch site'] = []
          launch_dict['Payload'] = []
          launch_dict['Payload mass'] = []
          launch_dict['Orbit'] = []
          launch_dict['Customer'] = []
          launch_dict['Launch outcome'] = []
          # Added some new columns
          launch_dict['Version Booster']=[]
          launch_dict['Booster landing']=[]
          launch_dict['Date']=[]
          launch_dict['Time']=[]
```

5. Create Dataframe



```
In [16]:  df=pd.DataFrame(launch_dict)
          df
```

6.save to .csv file



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

7.Github URL: Click here

Data Wrangling

1. EDA is performed on the dataset and the training labels are determined.

2. The number of launches for each site is calculated.

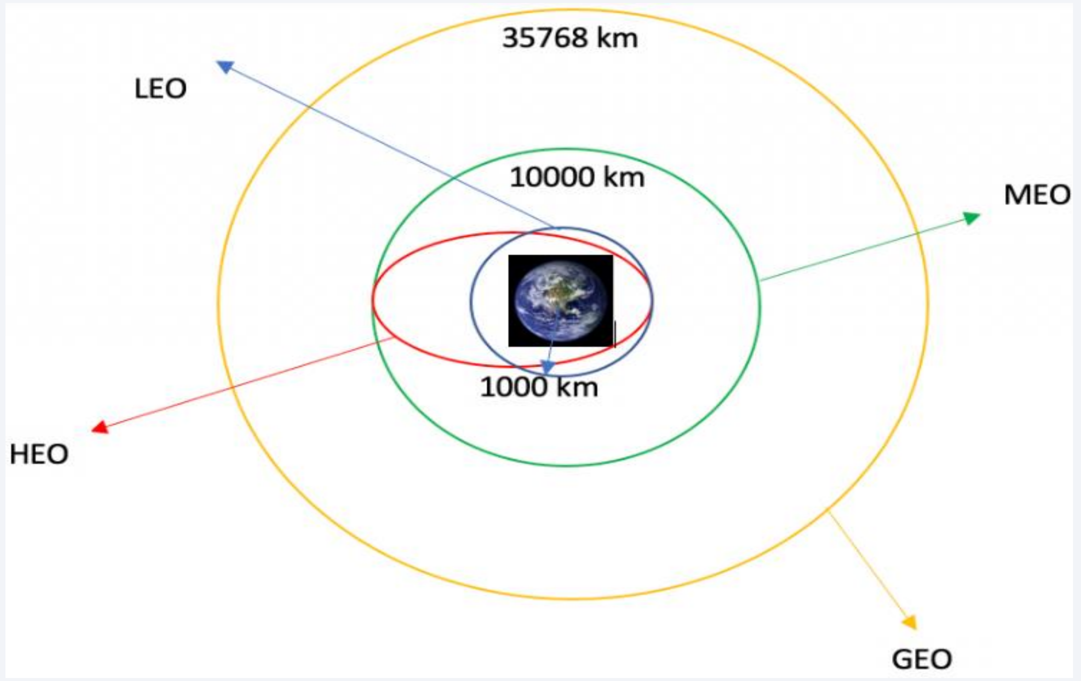
3. The number and occurrence of each orbit are determined.

4. The number and occurrence of mission outcome per orbit are calculated.

5. Landing outcome label is created, where bad-outcome is 0, otherwise it is 1.

6. Results are exported to .csv file.

Depending on the purpose of a rocket launch, each is launched to a dedicated orbit. Below is a plot showing some of the orbit types.



GitHub URL: [Click here]

EDA with Data Visualization

- The data was analyzed visually to show the relationship between some variables in the dataset.
- Scatter plots was drawn for the following:
  - ✓ Launch Site vrs Flight Number
  - ✓ Launch Site vrs Payload Mass (kg)
  - ✓ Orbit Vrs Flight number
  - ✓ Orbit vrs Payload Mass (kg)
- Scatter plots displays the relationship between two variables.

- GitHub URL: Click here
- Line graph was drown for :
  - ✓ Success Rate vrs Year
- Line graphs are used to track the treads of a variables over a period of time.
- A bar graph was also drown for :
  - ✓ Success Rate vrs Orbit.
- Bar graphs are used to display the comparisons among discrete categories at a glance

EDA with SQL

EDA was performed on the data using SQL in Jupyter Notebook.

Queries were used to find answers to some questions. Such as:

- The names of the unique launch sites in the space mission.
- The total payload mass carried by boosters launched by NASA (CRS).
- average payload mass carried by booster version F9 v1.1.
- The total number of successful and failure mission outcomes
- The names of the booster_versions which have carried the maximum payload mass.

- Github URL: Click here

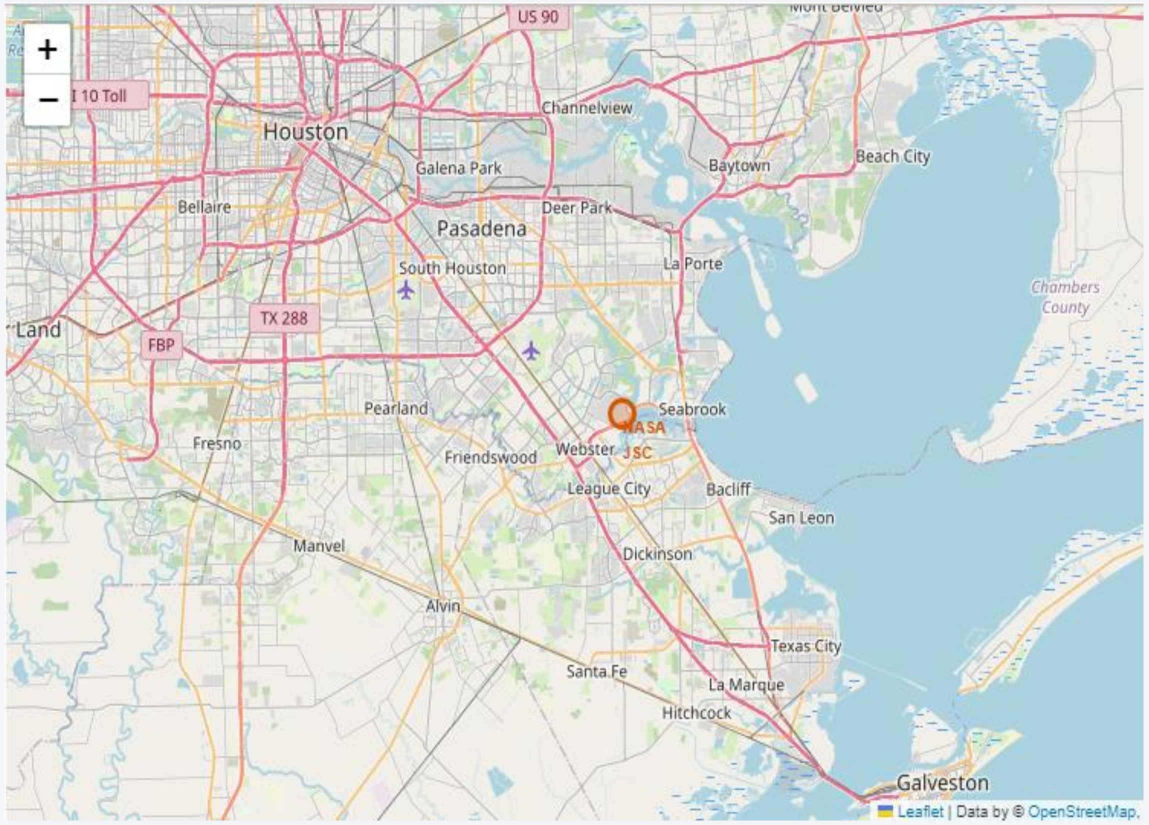
Build an Interactive Map with Folium

1. Folium map object is created with the initial location at NASA Johnson Space Center at Houston, Texas.

2. Map objects such as markers, and, circles, are created and added to each launch site on the map to highlight the area under consideration.

3. Launch Outcomes(successes, failures) in the dataframe is assign to classes 1 and 0 with the colors Green and Red markers on the map in a Marker Cluster object.

4. The distance between a launch site to its proximities are determined.



GitHub URL :Click here

nbviewer URL: Click here

Build a Dashboard with Plotly Dash

The Dashboard is built with Plotly Dash

Two graphs are plotted:

1. Pie Chart:
   - ✓ Displays the total launches by all the launch sites or certain launch sites.

2. Scatter Plot:
   - ✓ Shows the correlation between the Outcome and the Payload Mass (kg) for the different Booster Versions

GitHub URL :Click here

Predictive Analysis (Classification)

- The data is loaded using numpy and pandas
- The data is transformed
- The data is split into training and testing.
- Different machine learning models are built
- Different hyper-parameters are tuned using GridSearchCV
- Accuracy Score is used as the metric for the assessment of the model
- Feature-engineering and algorithm tuning are used for model improvement.
- The best performing classification model is determined.

GitHub URL: Click here

Results

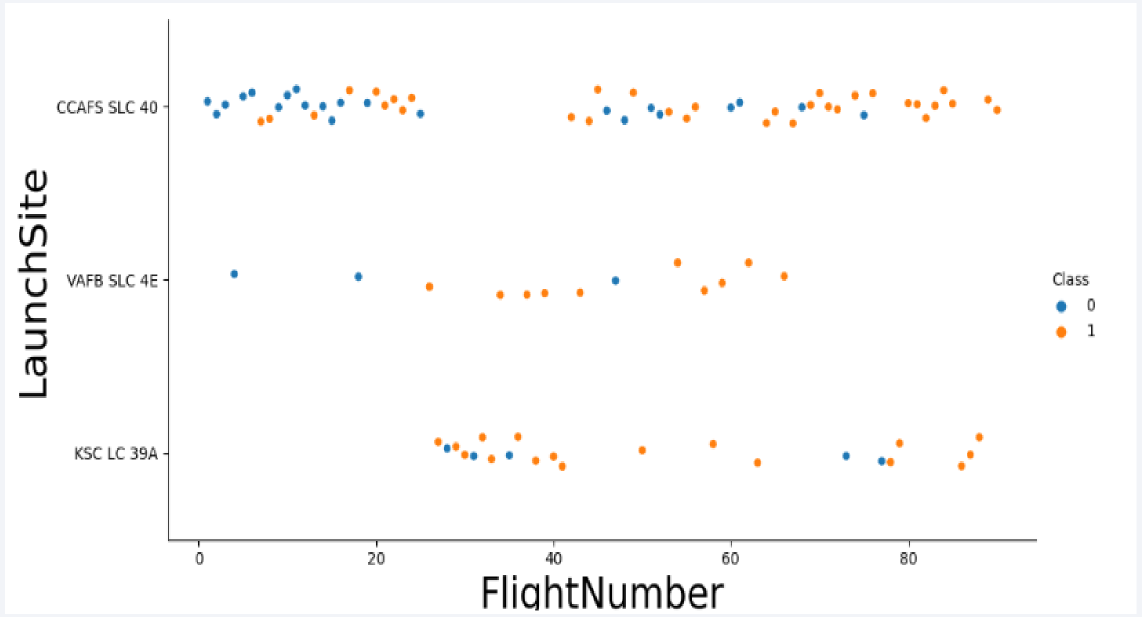
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

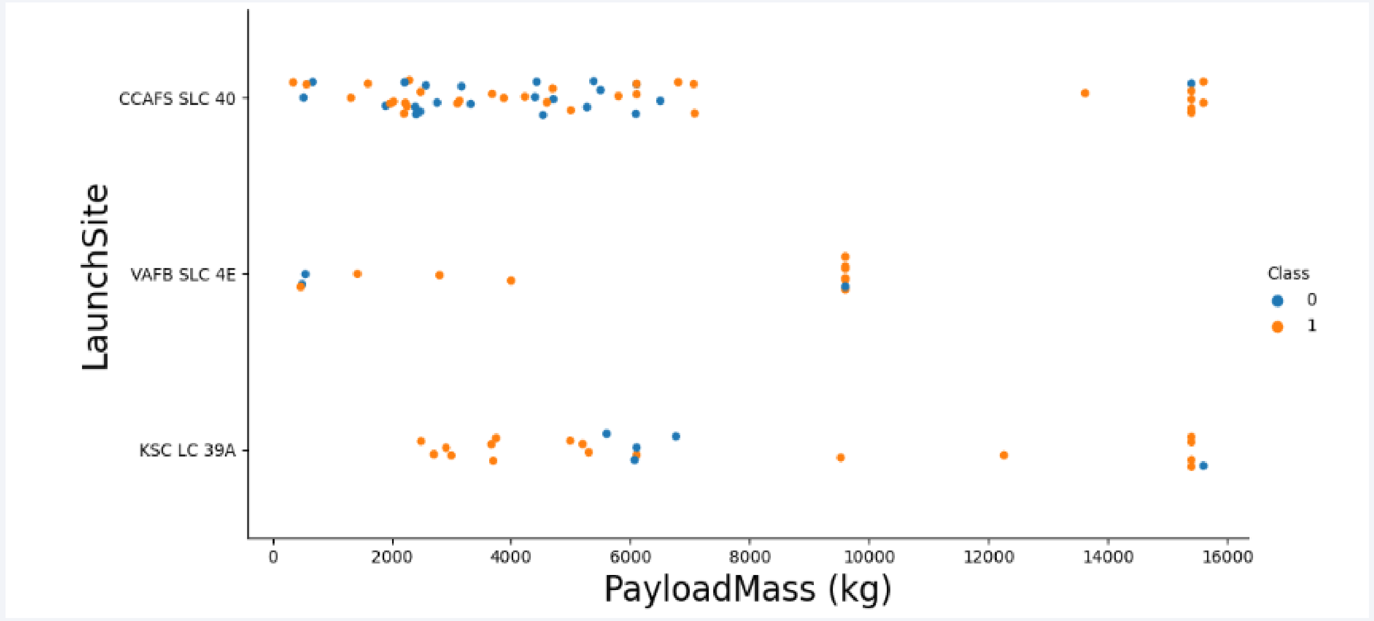
- Flight Number vs. Launch Site



- From the plot, it can be seen that launch sites with higher flight numbers have higher success launch rate.

Payload vs. Launch Site

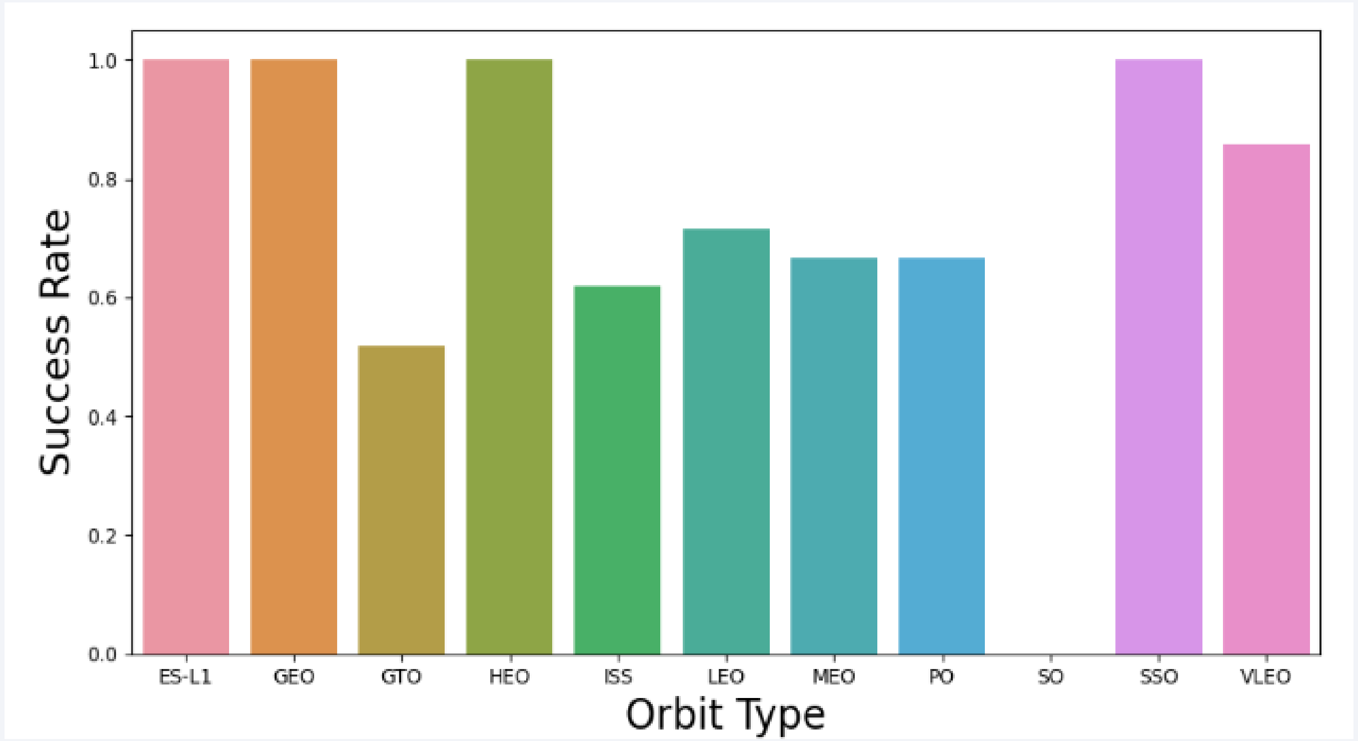
Scatter plot of Payload vs. Launch Site



From the plot, there is no clear correlation between the launch site and the payload mass(kg) in terms of the outcome of a rocket launch. However, launch site VAFB SLC 4E shows a higher success rate with increasing payload Mass (kg), even though, rocket launched never exceeded a Payload Mass (kg) of 10000.

Success Rate vs. Orbit Type

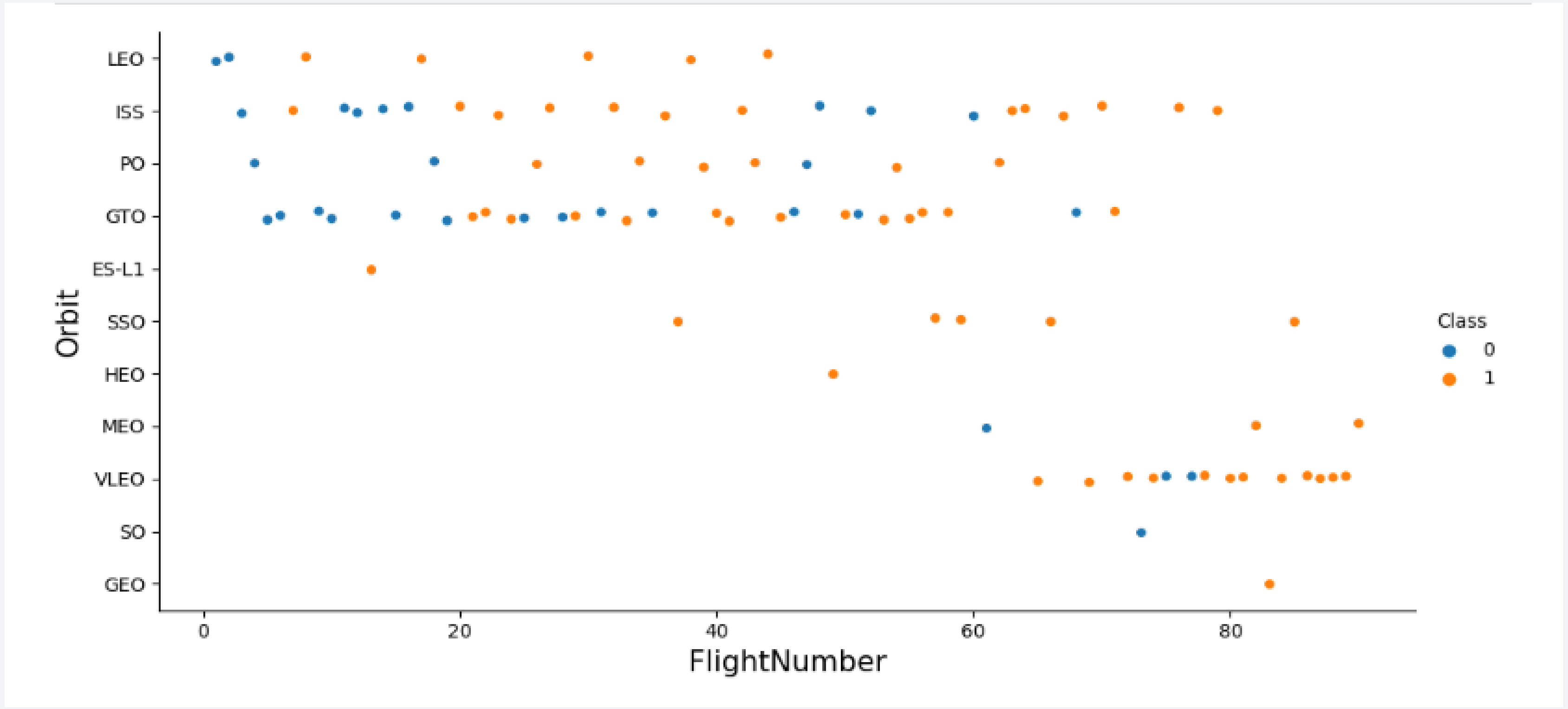
Bar chart for the success rate of each orbit type



As shown on the graph, ES-L1,GEO,HEO, and, SSO orbits had the most success rate.

Flight Number vs. Orbit Type

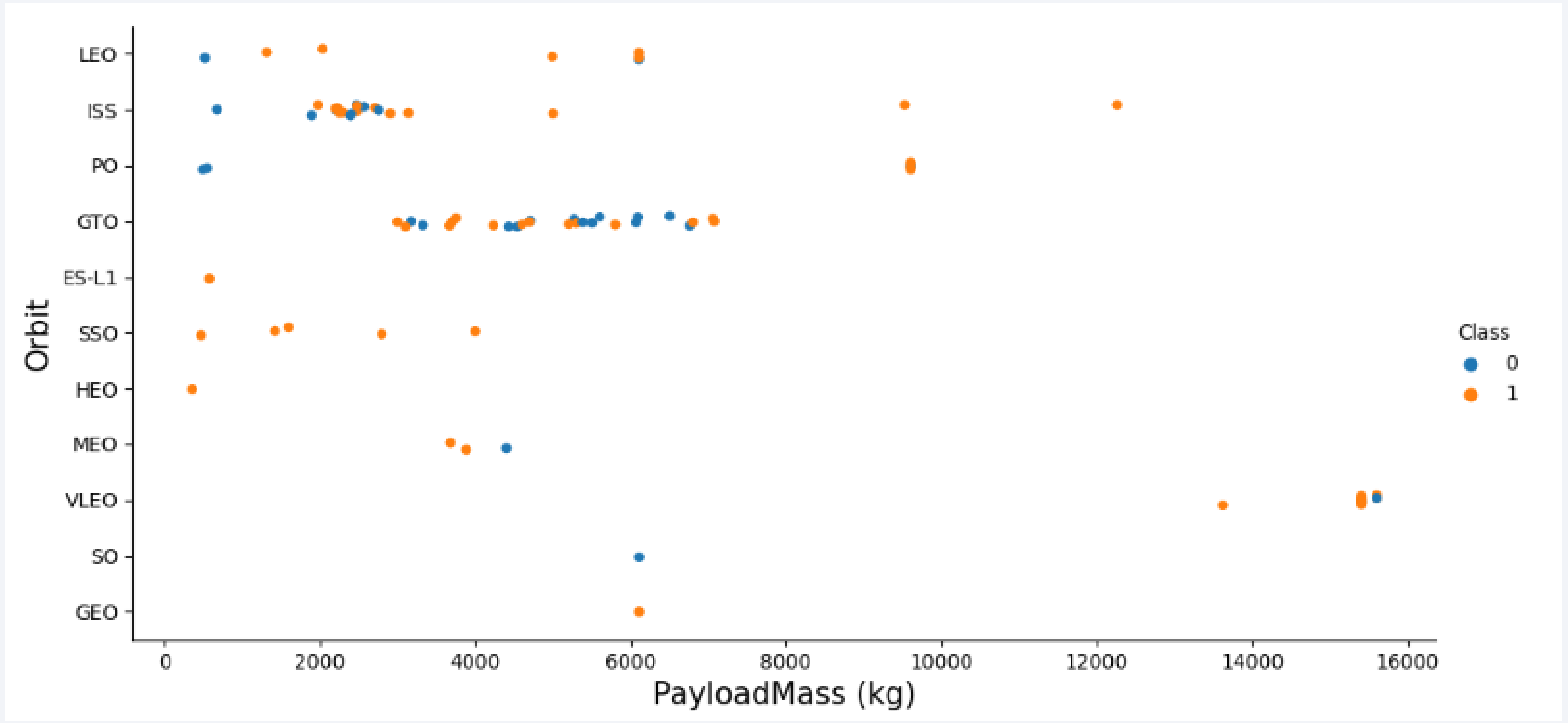
Scatter plot of Flight number vs. Orbit type



It can be observe that the LEO orbit correlates directly with the flight numbers. But, GTO orbit does not relate to the flight number.

Payload vs. Orbit Type

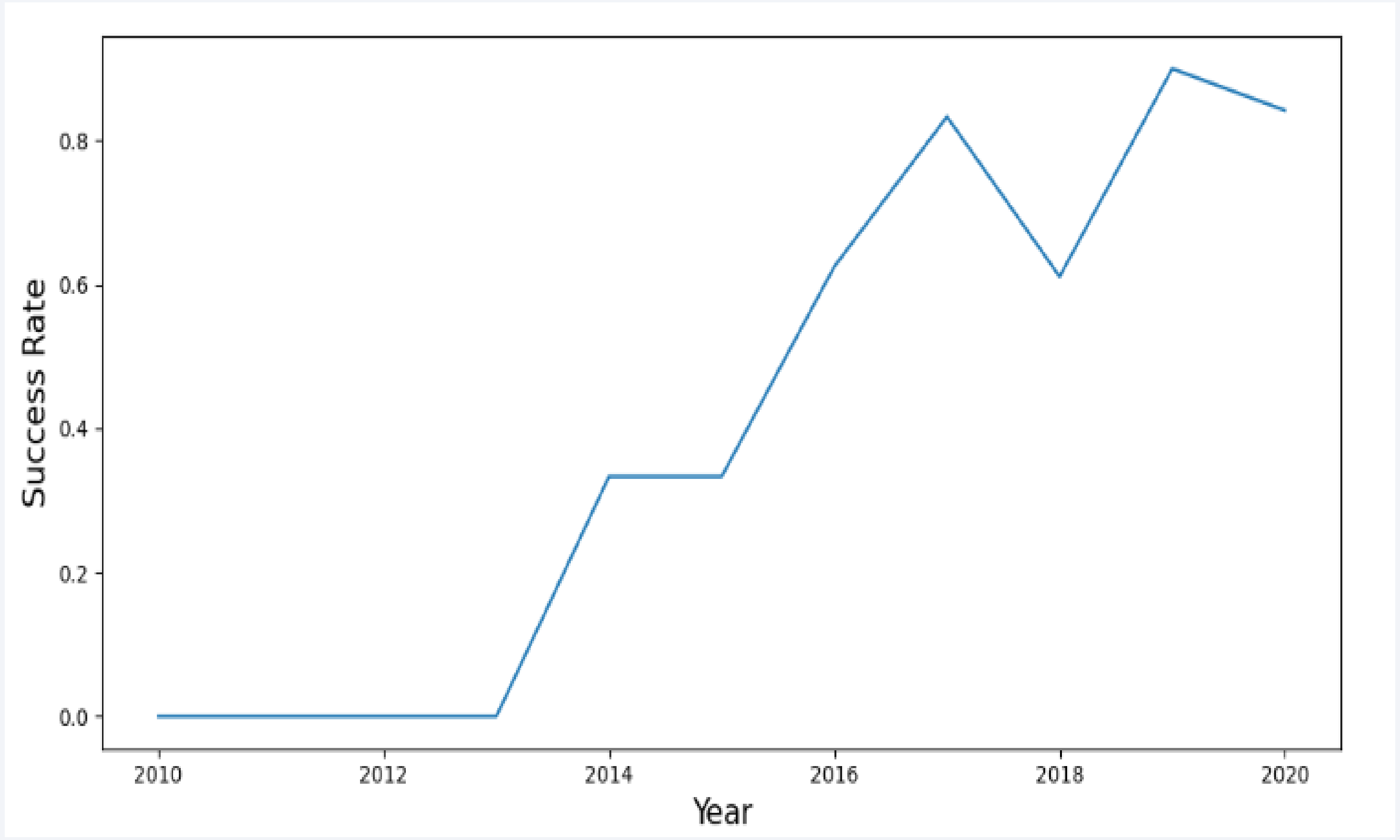
Scatter plot of payload vs. orbit type



From the graph, heavy Payload Mass from 4000 kg correlates positively with LEO, ISS, and, PO, orbits. However, heavy Payload Mass (kg) gives negative correlation with the GTO orbit

Launch Success Yearly Trend

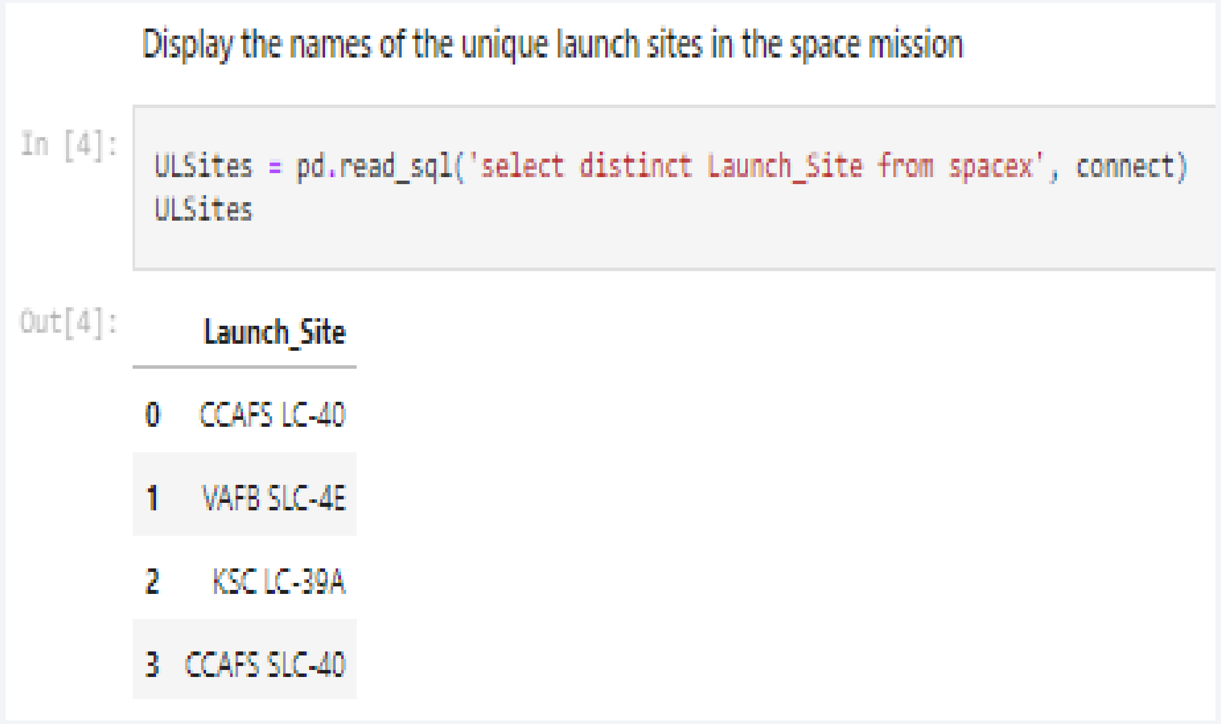
Line chart of yearly average success rate



The graph shows that the success rate increased from 2013 until 2020

All Launch Site Names

The names of the unique launch sites



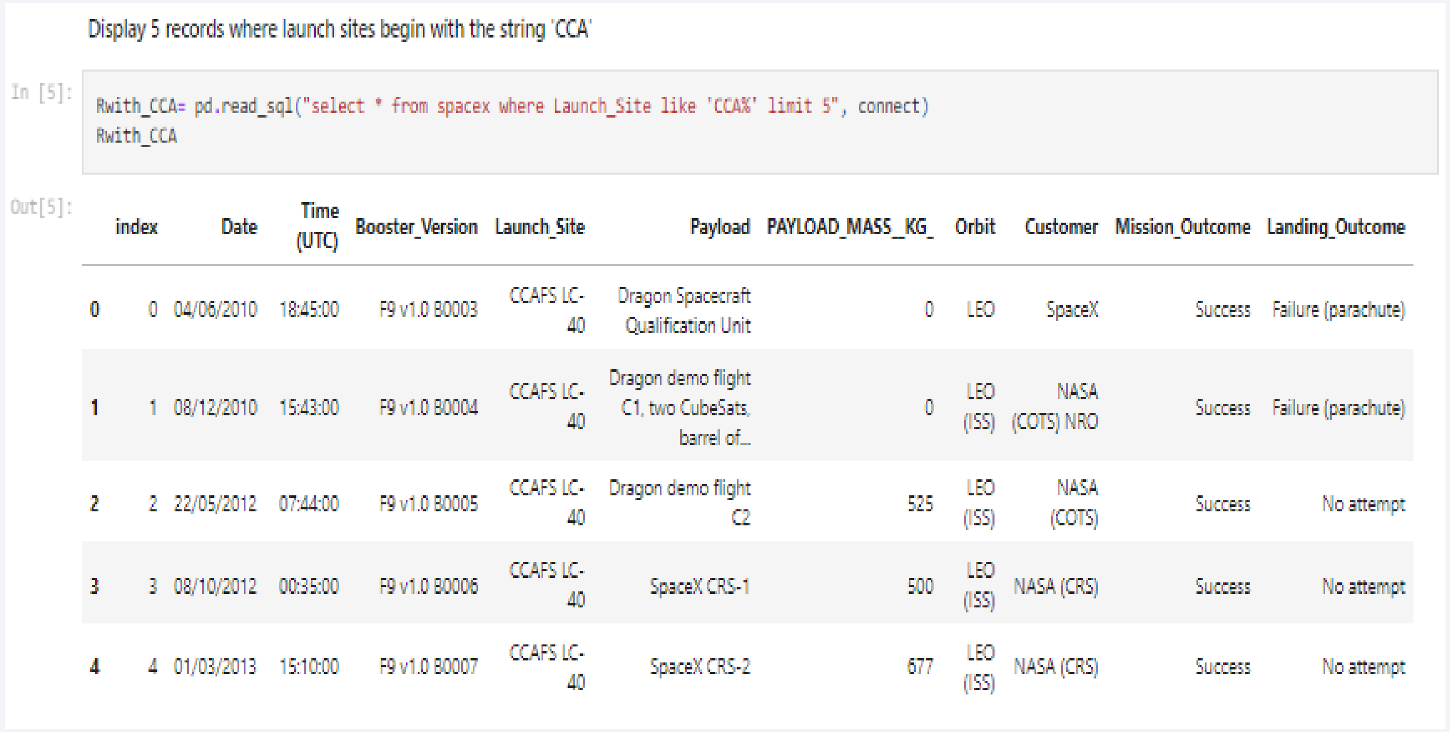
Display the names of the unique launch sites in the space mission

```
In [4]: ULSites = pd.read_sql('select distinct Launch_Site from spacex', connect)
        ULSites

Out[4]:
     Launch_Site
0    CCAFS LC-40
1    VAFB SLC-4E
2     KSC LC-39A
3   CCAFS SLC-40
```

The word ‘distinct’ is used to display only the unique launch sites in the Spacex dataset

Launch Site Names Begin with 'CCA'

5 records where launch sites begin with `CCA`



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

```
In [5]: Rwith_CCA= pd.read_sql("select * from spacex where Launch_Site like 'CCA%' limit 5", connect)
        Rwith_CCA
```

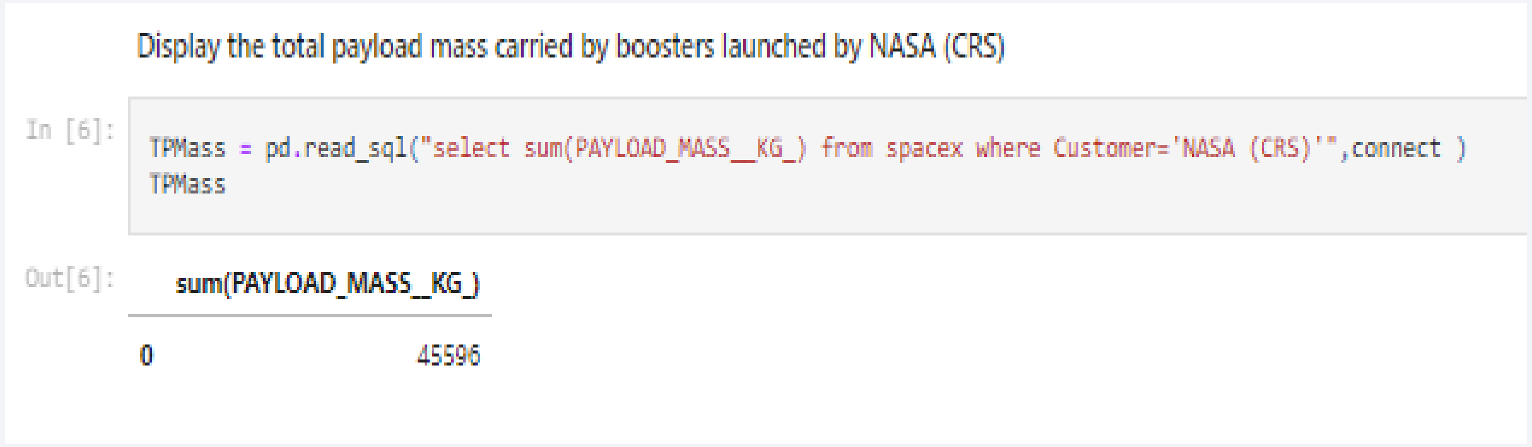
Out[5]:

	index	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	0	04/06/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	1	08/12/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2	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
3	3	08/10/2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	4	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 above code is used to generate the launch sites beginning with ‘CCA’. The ‘limit 5’ in the code is used to ensure only 5 records are displayed. The percentage sign behind ‘CCA%’ suggest that the name of the launch site must begin with CCA

Total Payload Mass

The total payload carried by boosters from NASA



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

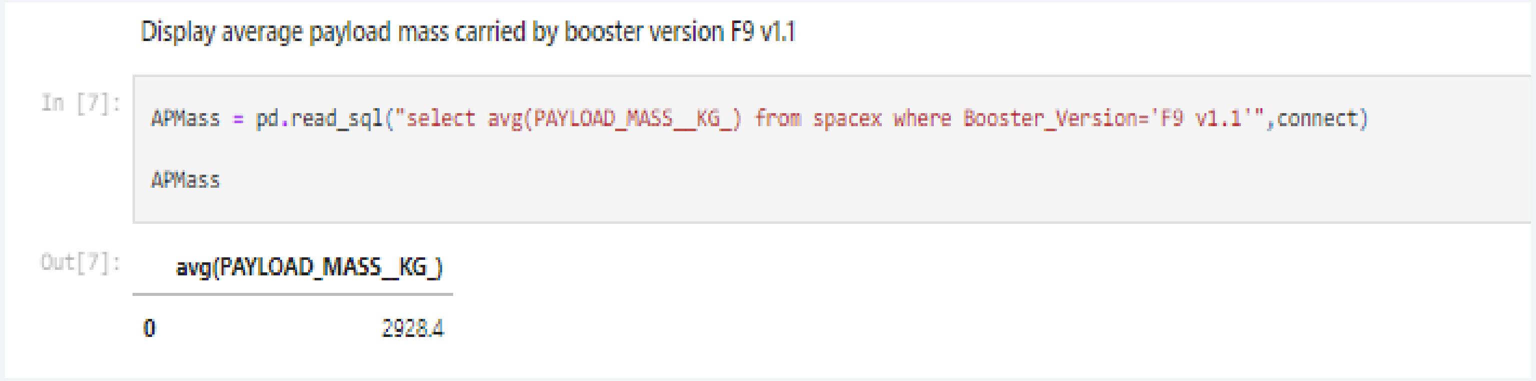
```
In [6]:  TPMass = pd.read_sql("select sum(PAYLOAD_MASS__KG_) from spacex where Customer='NASA (CRS)'",connect )
         TPMass
```

```
Out[6]:    sum(PAYLOAD_MASS__KG_)
        0                   45596
```

The total payload mass (kg) is calculated from the above query. The value is 45596 kg.

Average Payload Mass by F9 v1.1

The average payload mass carried by booster version F9 v1.1is determined as follows



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

```
In [7]: APMass = pd.read_sql("select avg(PAYLOAD_MASS__KG_) from spacex where Booster_Version='F9 v1.1'",connect)

        APMass
```

```
Out[7]:    avg(PAYLOAD_MASS__KG_)
        0                  2928.4
```

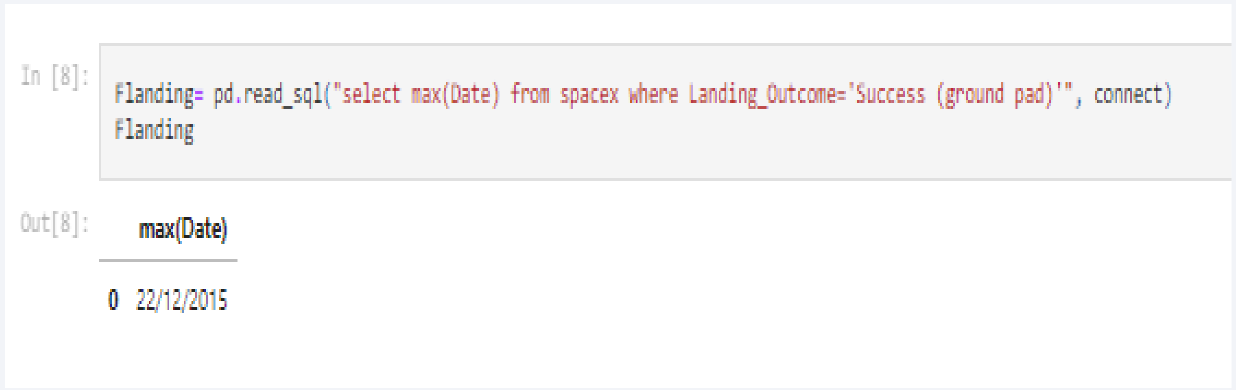
The average Payload Mass is 2928.4 kg

The ‘avg’ in the query is used to calculate the average value of the payload mass (kg)

The ‘where’ is used to filter the Booster Version column.

First Successful Ground Landing Date

The dates of the first successful landing outcome on ground pad is determined using the query

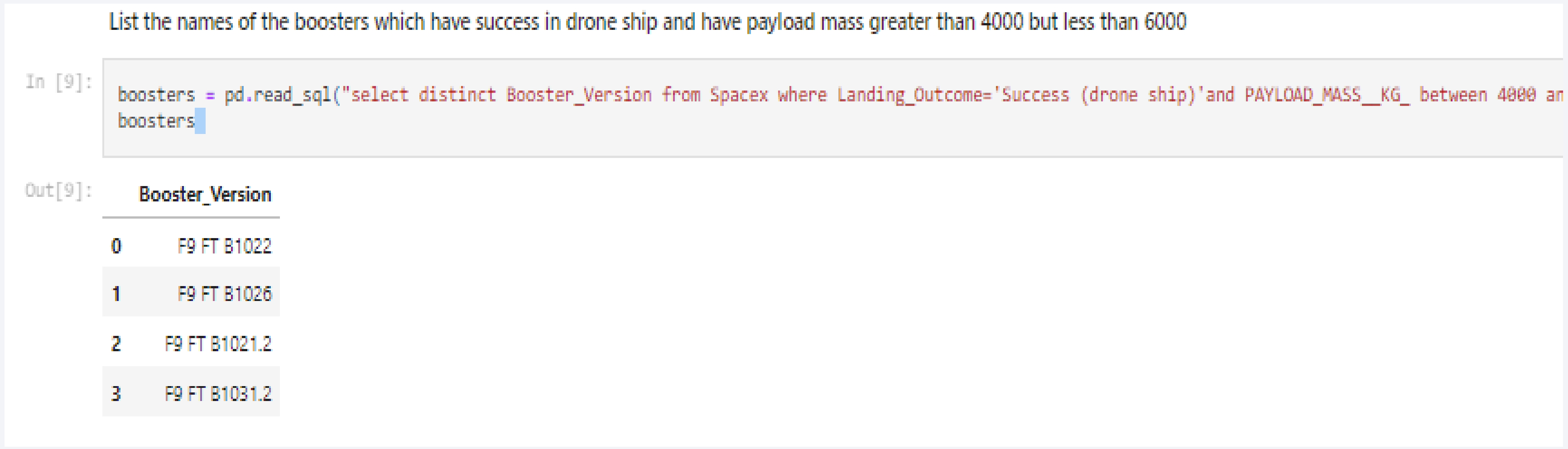


```
In [8]: Flanding= pd.read_sql("select max(Date) from spacex where Landing_Outcome='Success (ground pad)'", connect)
        Flanding

Out[8]:    max(Date)
        0  22/12/2015
```

The first successful landing occurs on 22/12/2015

Successful Drone Ship Landing with Payload between 4000 and 6000

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



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

```
In [9]:  boosters = pd.read_sql("select distinct Booster_Version from Spacex where Landing_Outcome='Success (drone ship)'and PAYLOAD_MASS__KG_ between 4000 an
         boosters
```

Out[9]:

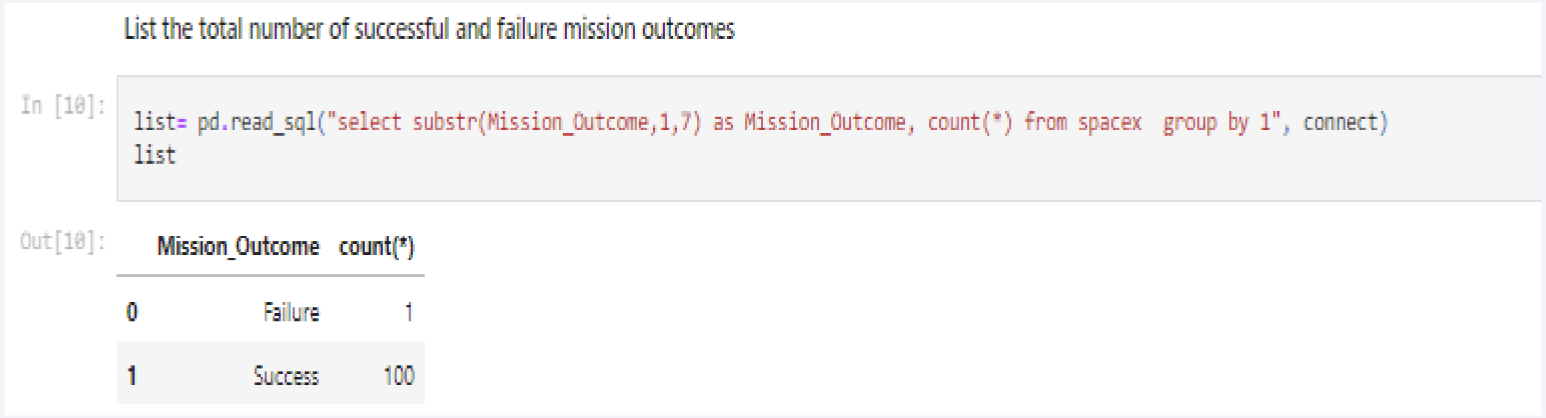
	Booster_Version
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

WHERE clause is used to filter for boosters which have successfully landed on drone ship.

AND condition is used to determine the successful landing with payload mass greater than 4000 but less than6000

Total Number of Successful and Failure Mission Outcomes

The total number of successful and failure mission outcomes



List the total number of successful and failure mission outcomes

```
In [10]:  list= pd.read_sql("select substr(Mission_Outcome,1,7) as Mission_Outcome, count(*) from spacex  group by 1", connect)
          list
```

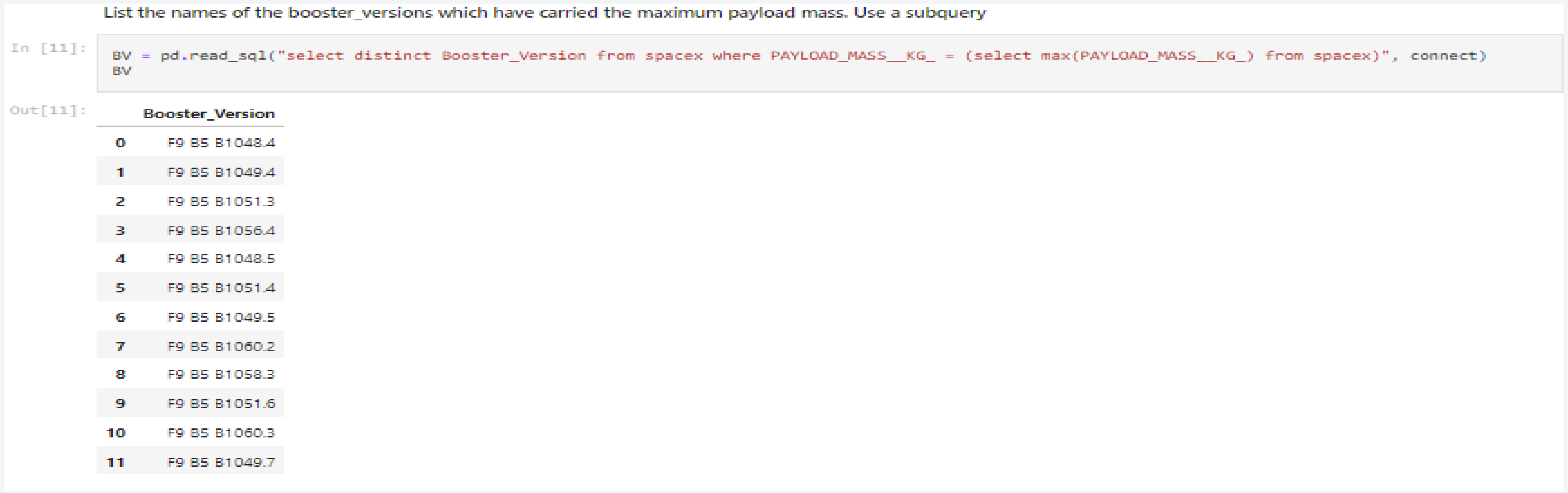
```
Out[10]:
     Mission_Outcome   count(*)
0            Failure          1
1            Success        100
```

The substr() function is used to extract the successful and failed outcomes from the mission outcome column of the SpaceX dataframe.

‘1’ in the sunstr() functions specifies the starting position and ‘7’ represents the number of characters to extract

Count(*) is used to count all the failed and successful outcomes and ‘group by ‘ is used to group the outcome into failure and success

Boosters Carried Maximum Payload



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

```
In [11]: BV = pd.read_sql("select distinct Booster_Version from spacex where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacex)", connect)
         BV
```

Out[11]:

	Booster_Version
0	F9 B5 B1048.4
1	F9 B5 B1049.4
2	F9 B5 B1051.3
3	F9 B5 B1056.4
4	F9 B5 B1048.5
5	F9 B5 B1051.4
6	F9 B5 B1049.5
7	F9 B5 B1060.2
8	F9 B5 B1058.3
9	F9 B5 B1051.6
10	F9 B5 B1060.3
11	F9 B5 B1049.7

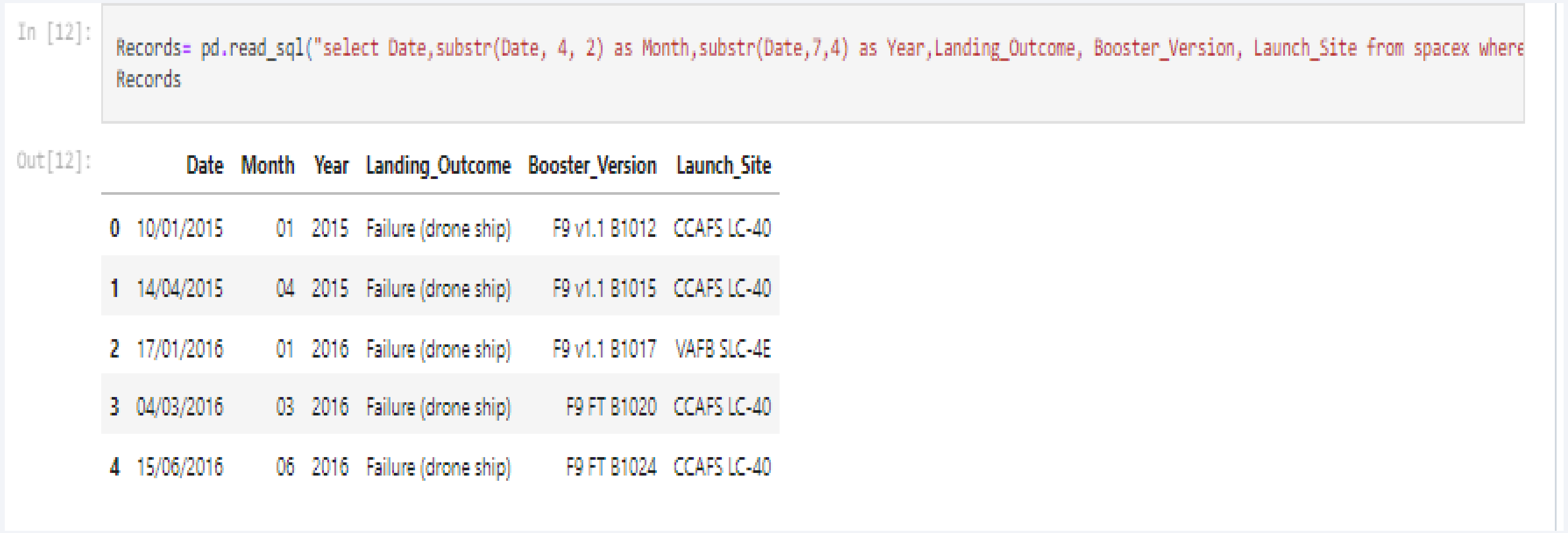
The above query is used to obtain the booster versions which carried the maximum payload mass.

Distinct in the query is used to show unique values of the booster version column in the Spacex data set.

‘Max’ is used to select the maximum payload mass

2015 Launch Records

Below is the query for showing 2015 launch records



```
In [12]: Records= pd.read_sql("select Date,substr(Date, 4, 2) as Month,substr(Date,7,4) as Year,Landing_Outcome, Booster_Version, Launch_Site from spacex where
         Records
```

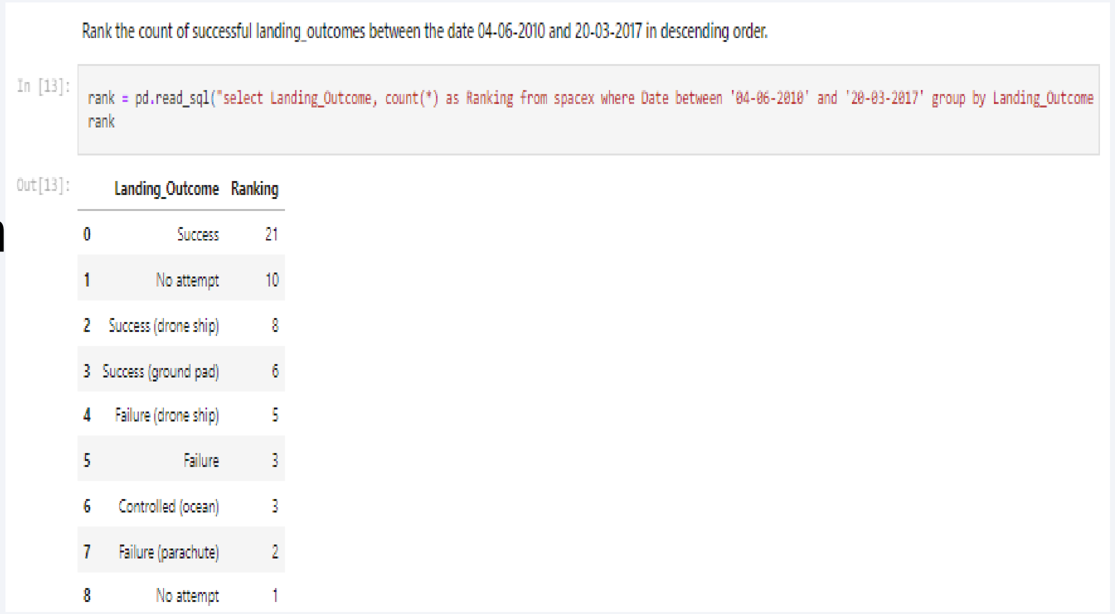
```
Out[12]:
         Date  Month  Year  Landing_Outcome      Booster_Version  Launch_Site
0  10/01/2015     01  2015  Failure (drone ship)  F9 v1.1 B1012  CCAFS LC-40
1  14/04/2015     04  2015  Failure (drone ship)  F9 v1.1 B1015  CCAFS LC-40
2  17/01/2016     01  2016  Failure (drone ship)  F9 v1.1 B1017  VAFB SLC-4E
3  04/03/2016     03  2016  Failure (drone ship)     F9 FT B1020  CCAFS LC-40
4  15/06/2016     06  2016  Failure (drone ship)     F9 FT B1024  CCAFS LC-40
```

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

Ranking of the landing outcomes is as shown below

The **GROUP By** clause is used to group the landing outcomes

The **ORDER BY** clause is used to order th grouped landing outcome in descending order.

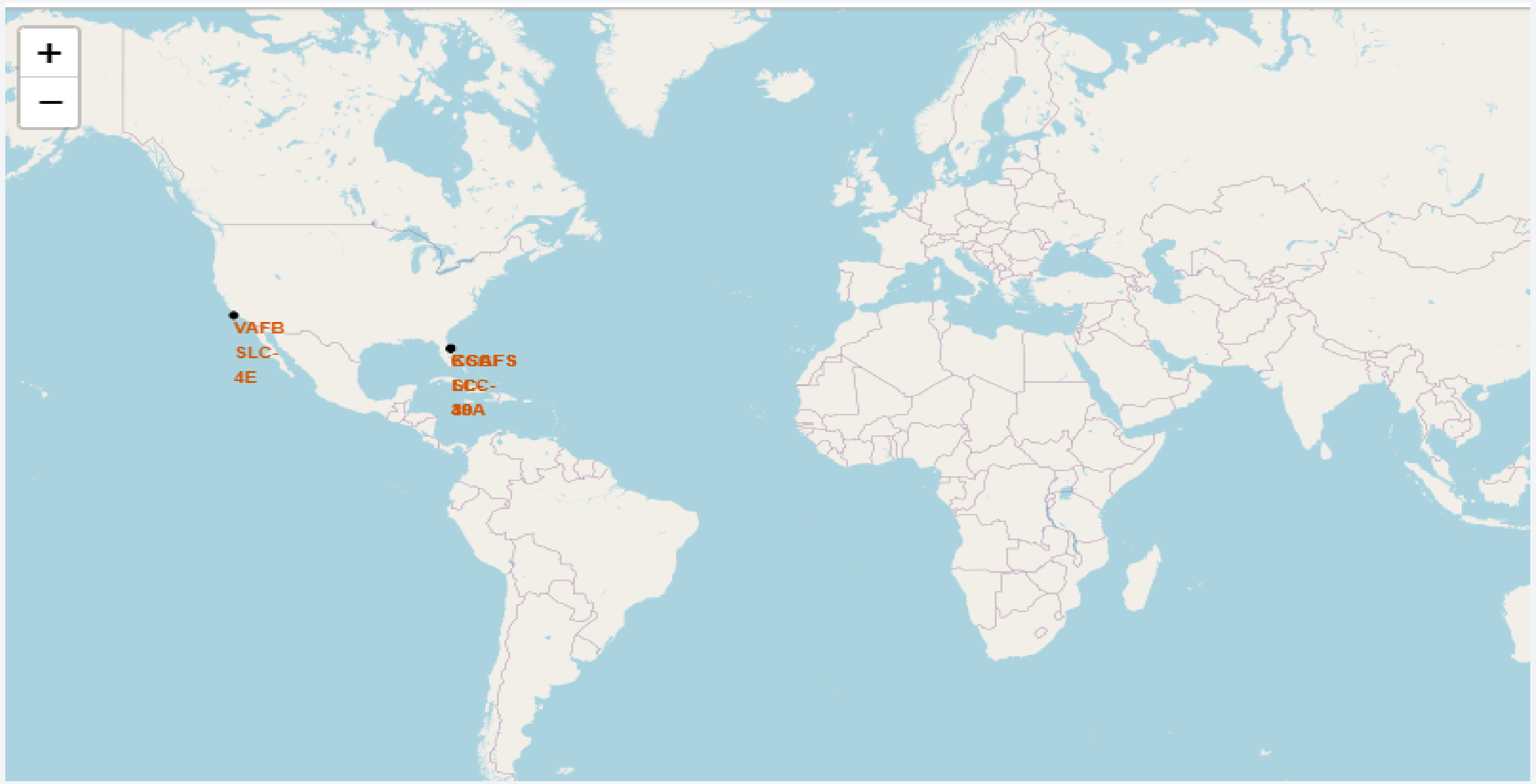

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
In [13]: rank = pd.read_sql("select Landing_Outcome, count(*) as Ranking from spacex where Date between '04-06-2010' and '20-03-2017' group by Landing_Outcome
         rank

Out[13]:
     Landing_Outcome  Ranking
0            Success       21
1         No attempt       10
2  Success (drone ship)     8
3  Success (ground pad)     6
4  Failure (drone ship)     5
5            Failure        3
6   Controlled (ocean)      3
7  Failure (parachute)      2
8         No attempt        1
```

Section 3

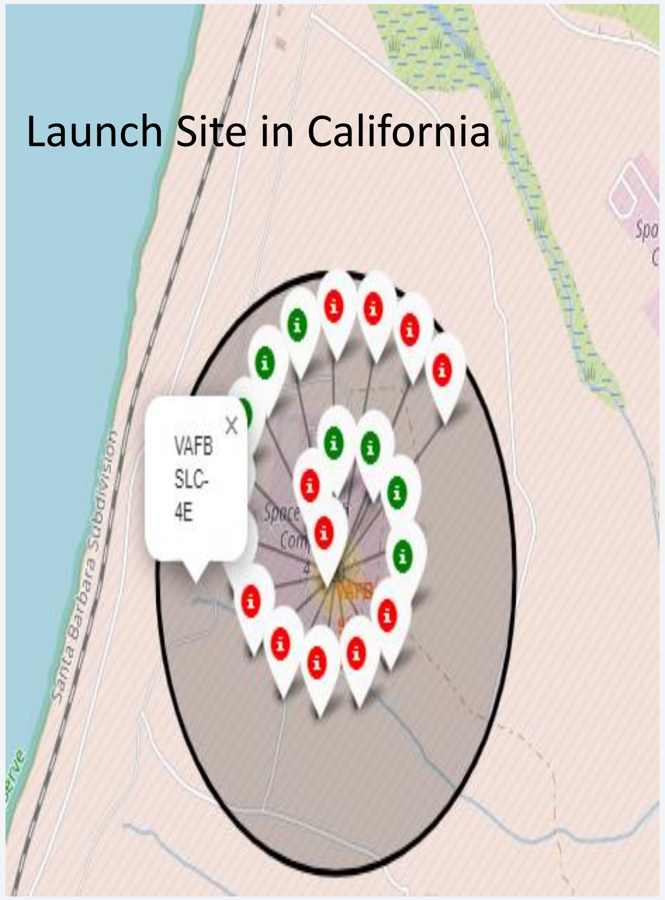
Launch Sites Proximities Analysis

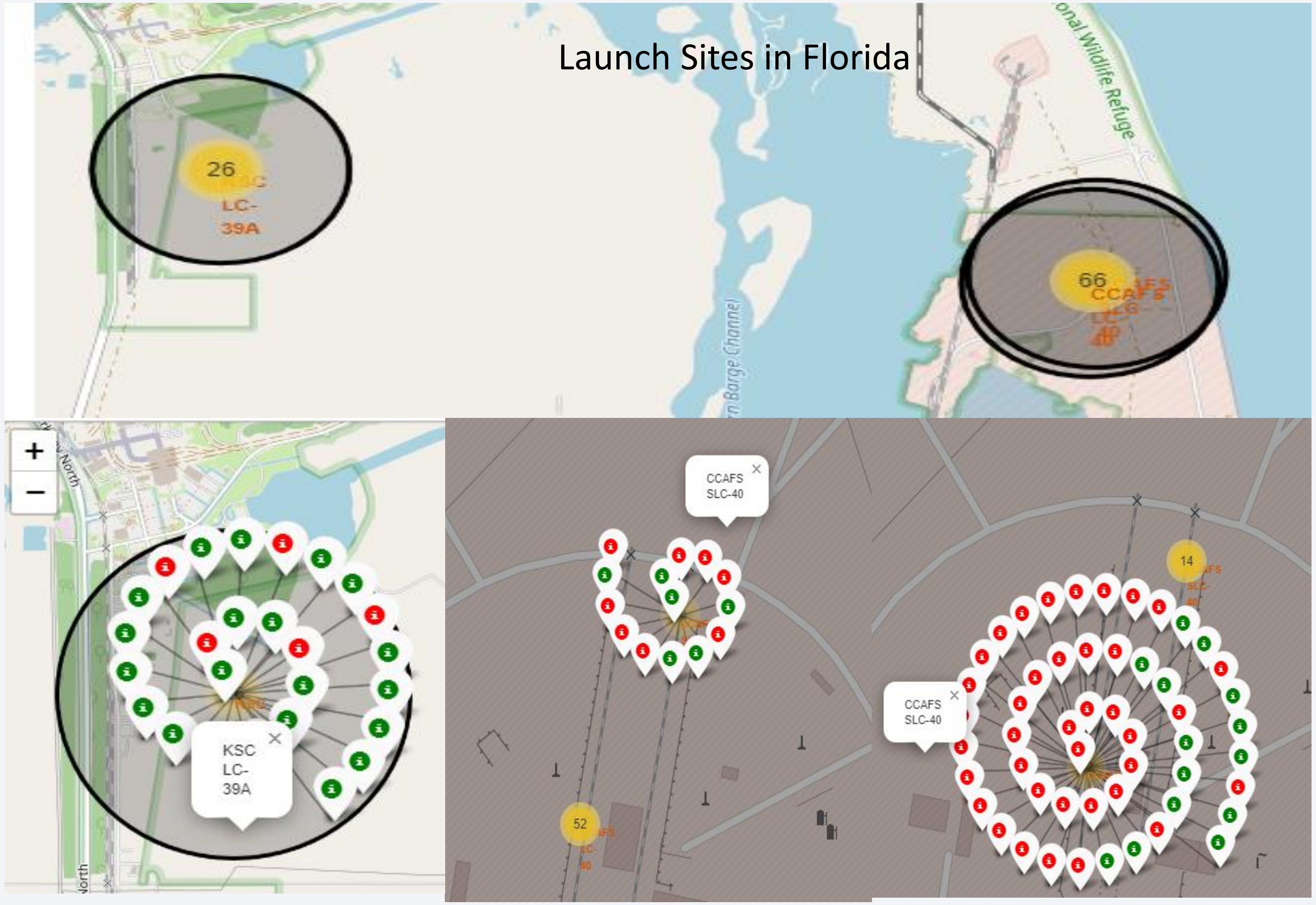
Global Map Markers for All Launch Sites



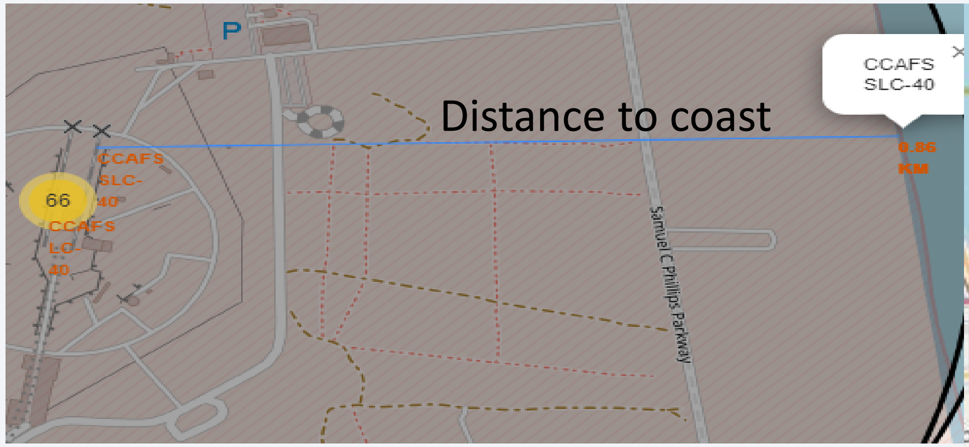
From the folium map, the launch sites are located at Florida and California in the United states.

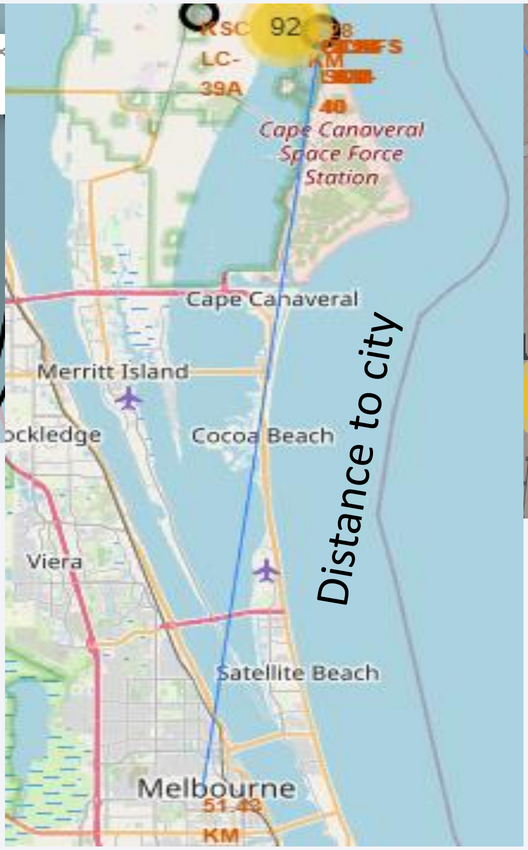
Launch Sites with Colored-Labeled Markers

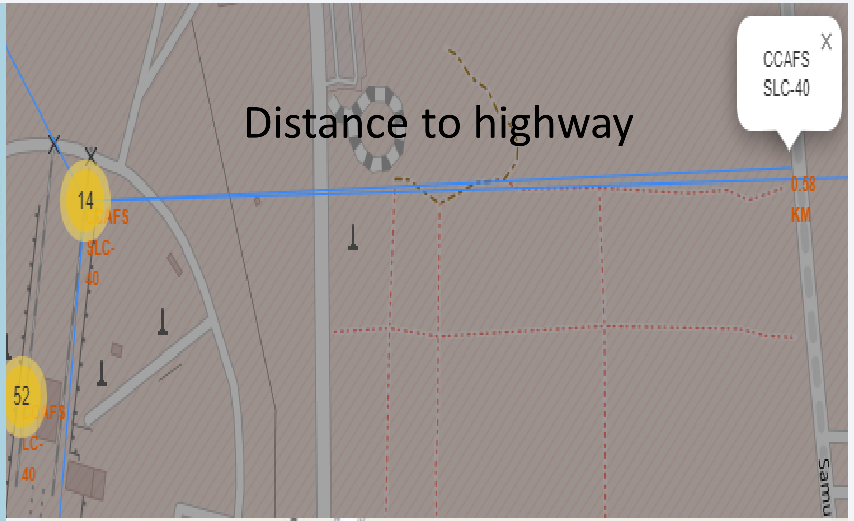

Launch Site in California

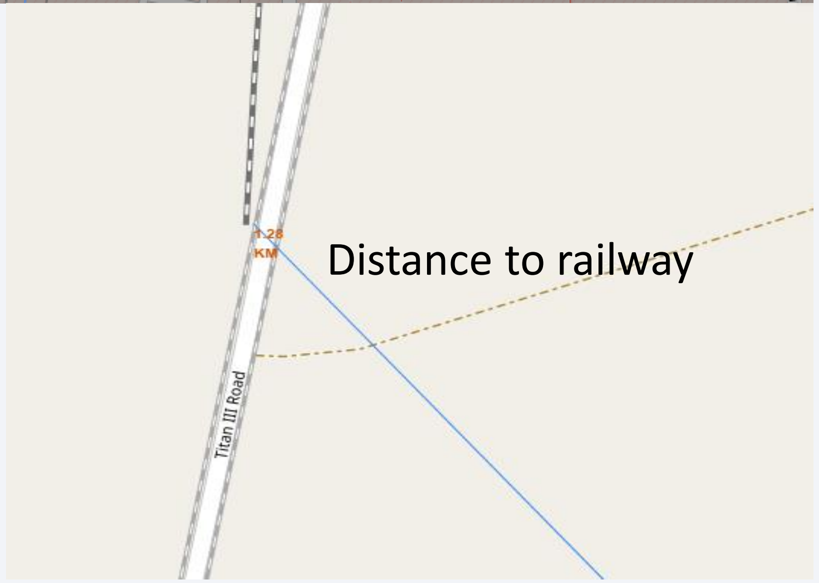

Launch Sites in Florida

Distance Between Launch Site and Landmarks









Are launch sites in close proximity to coastline? YES

Are launch sites in close proximity to railways? NO

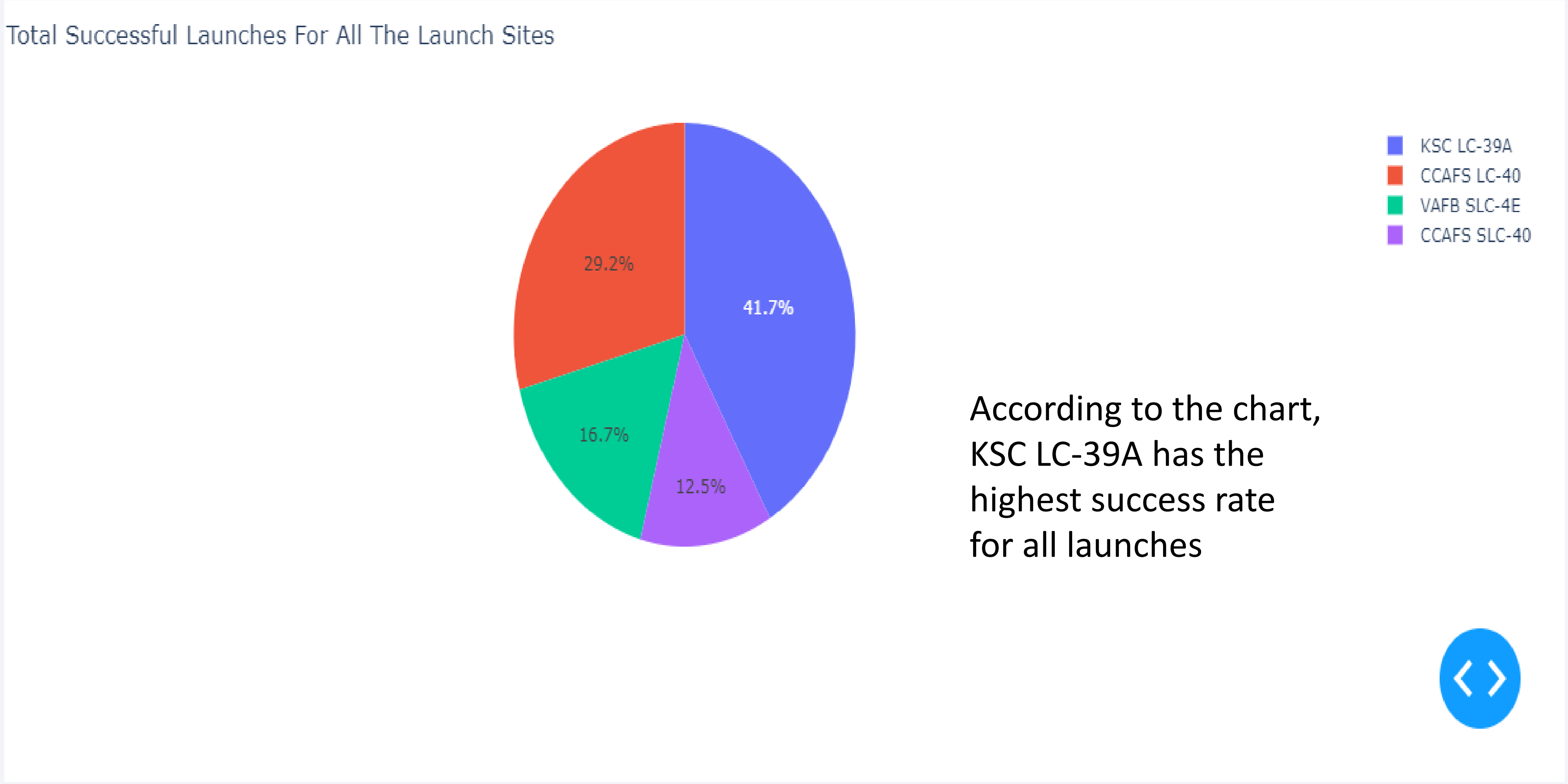
Are launch sites in close proximity to highways? YES, closer than the coastline

Do launch sites keep certain distance from cities? YES

Section 4

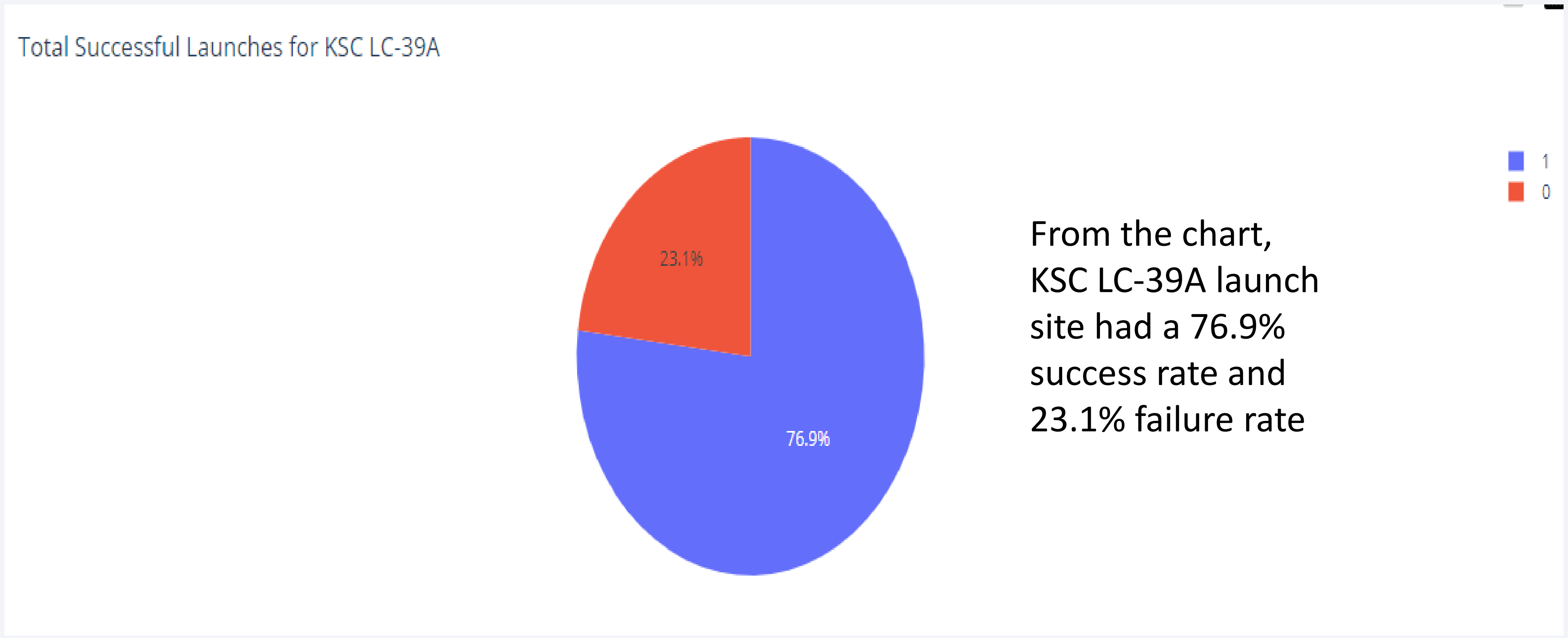
Build a Dashboard with Plotly Dash

Pie Chart Showing Total Successful Launches for all Site



According to the chart, KSC LC-39A has the highest success rate for all launches

Pie Chart For The Highest Success Ratio Launch Site

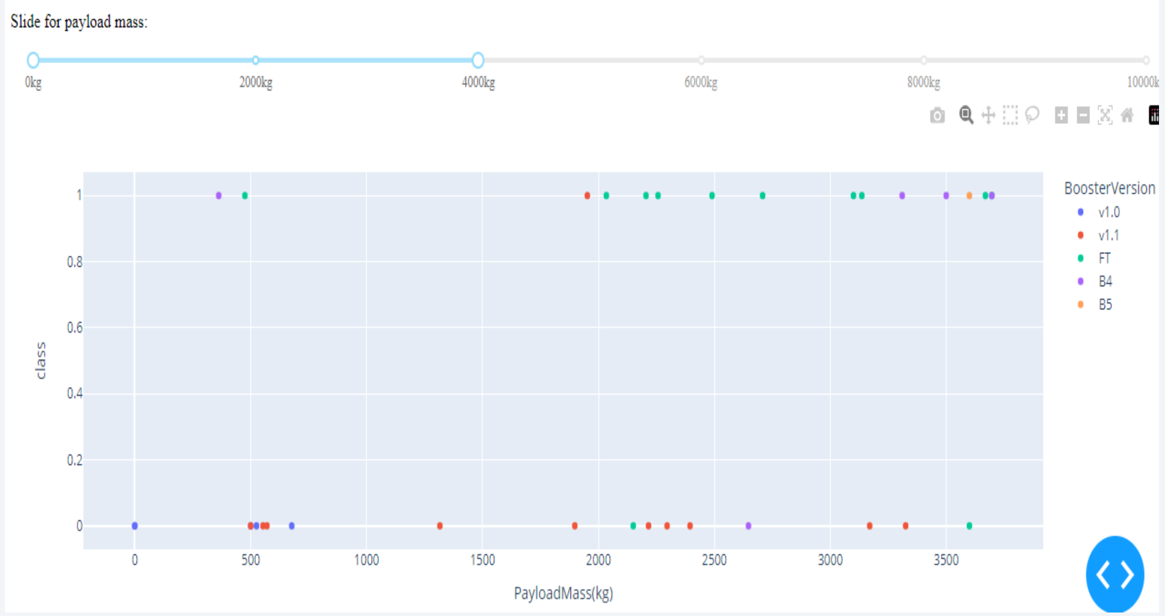


From the chart, KSC LC-39A launch site had a 76.9% success rate and 23.1% failure rate

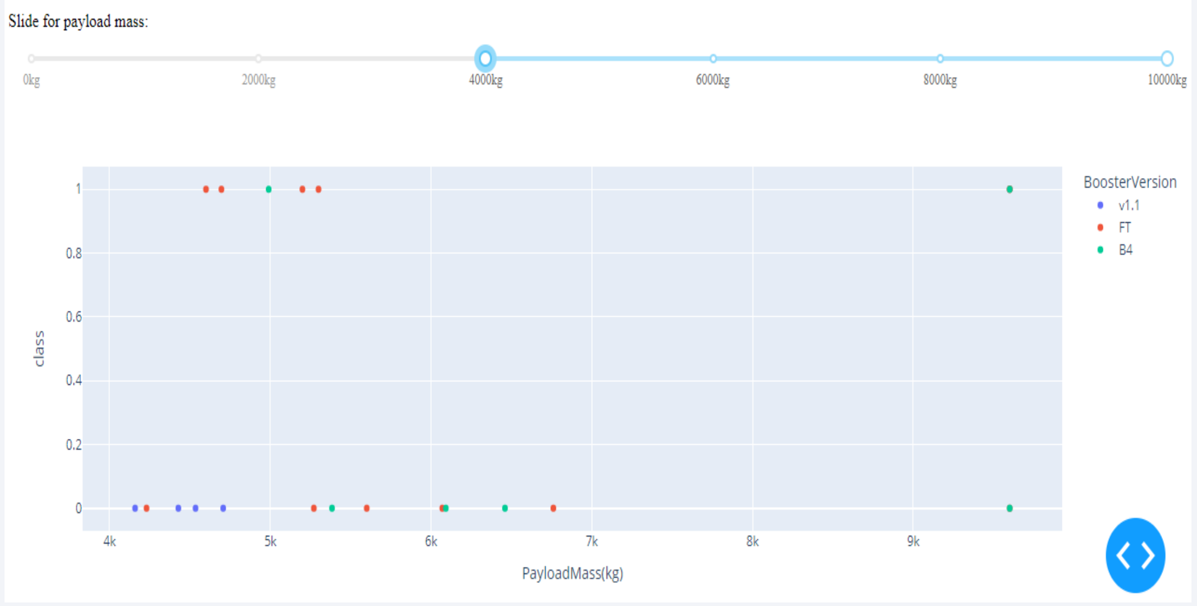
Scatter Plot For Payload vs. launch Outcome

Plot For All Sites, With Different Payload Selected In The Range Slider

Low Payload Mass (0kg – 4000kg)



Heavy Payload Mass (4000kg -10000kg)



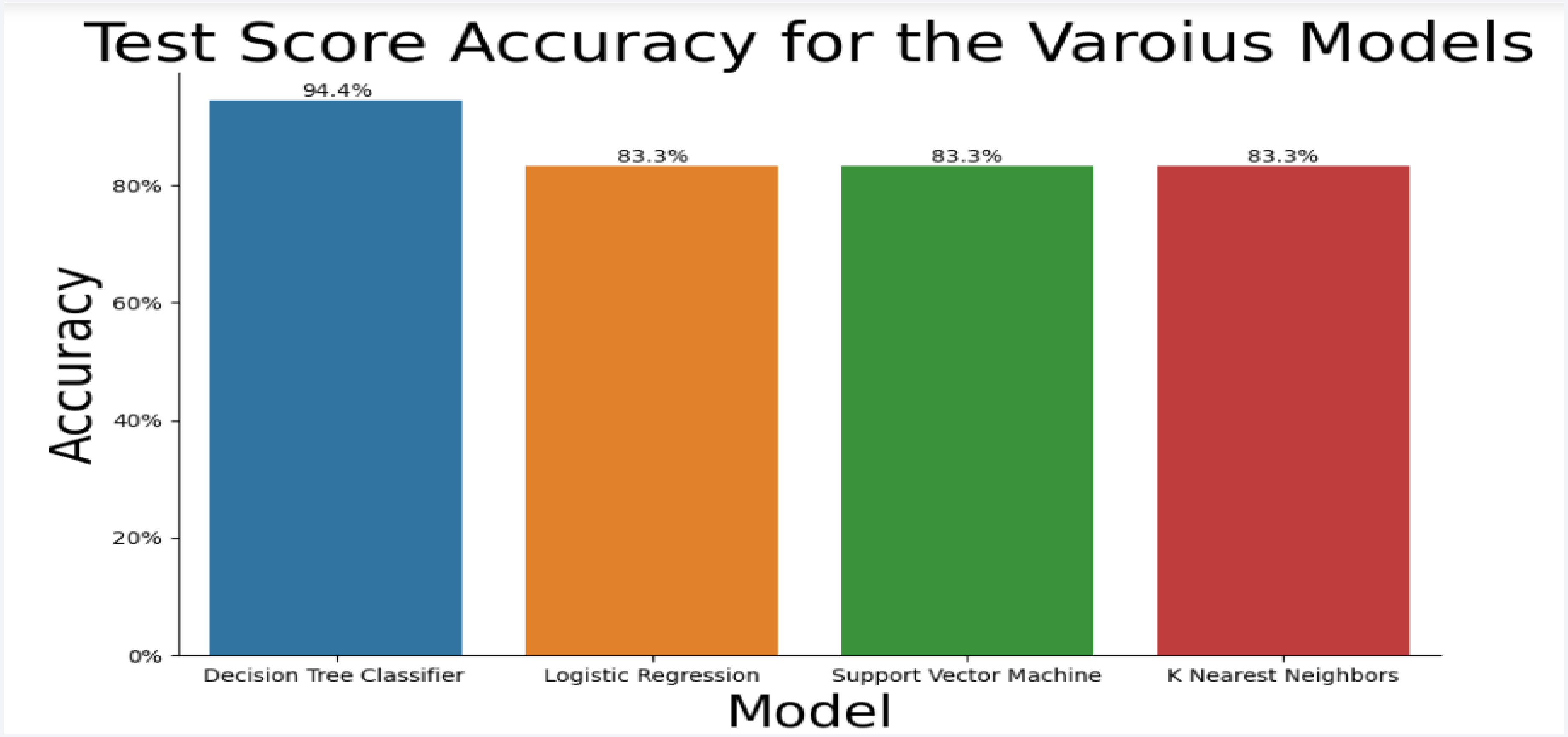
It can be observed from the two plots that, rockets with low payload mass (kg) had a higher success rate than rockets with heavy payload mass (kg).

Section 5

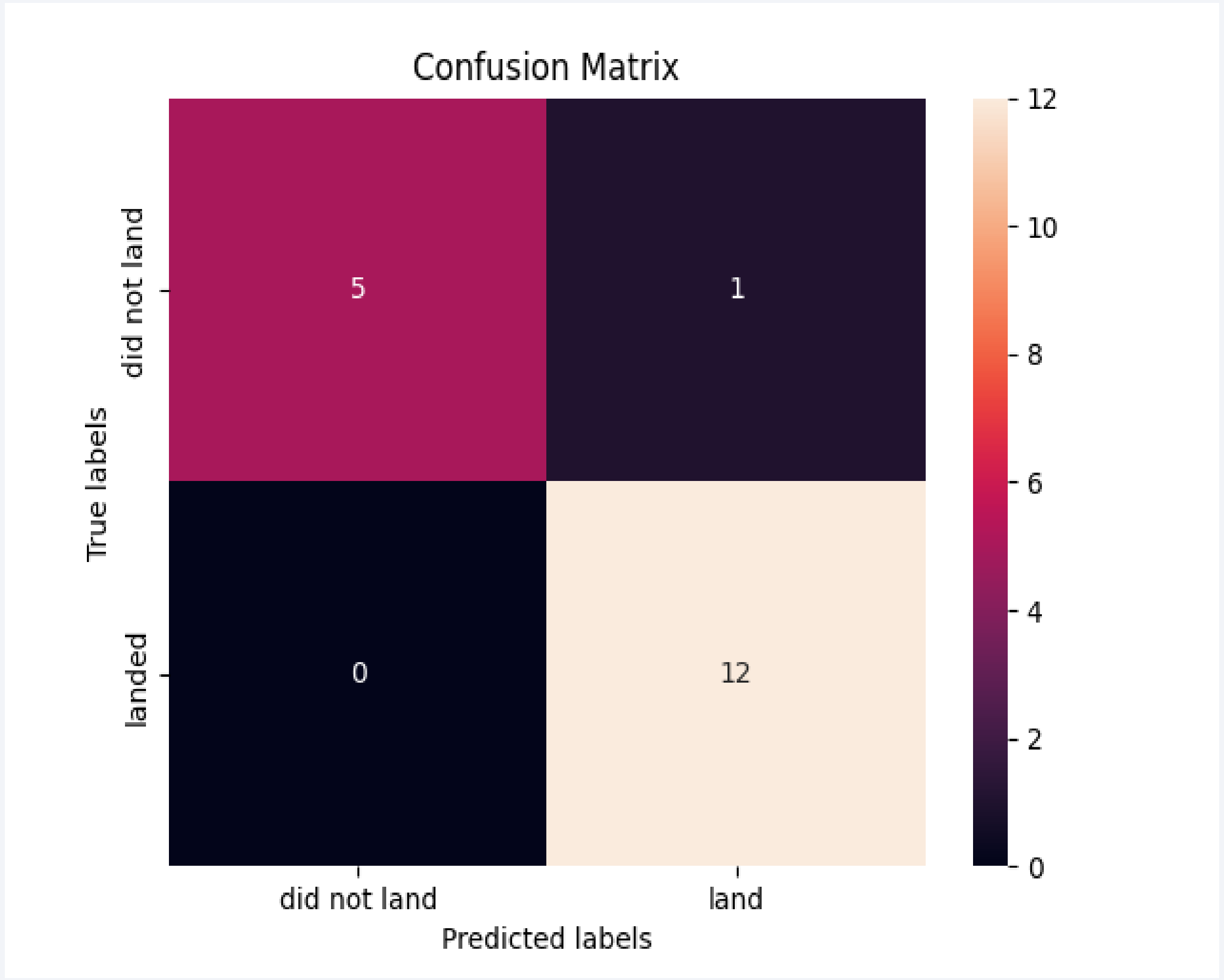
Predictive Analysis (Classification)

Classification Accuracy

From the bar graph, the Decision Tree Classifier has the best accuracy score of 94.4%



Confusion Matrix



The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes.

The major problem is that unsuccessful landing is marked as successful landing (false positives).

Conclusions

It can be concluded that:

1. Rockets with low Payload Mass (kg) performed better than rockets with heavy Payload Mass (kg).
2. The greater the flight number at a launch site, the higher the rate of success.
3. The Decision tree classifier is the best machine learning algorithm for this project
4. Successful launch rate increased from 2013 untill 2020.
5. The ES-L1, GEO, SSO,HEO,and, VLEO orbits had the most successful launch rate.
6. The KSC LC-39A site had the most successful launch rate of all the launch sites.

Thank you!