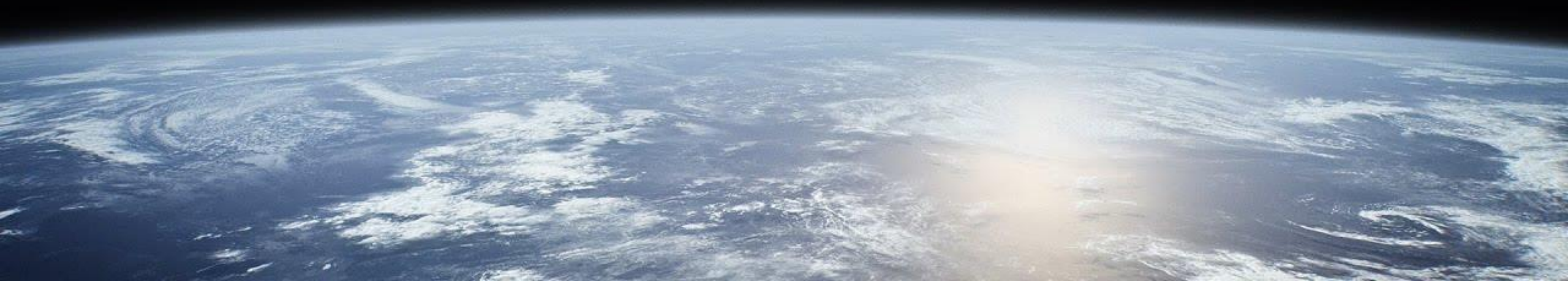
IBM Developer SKILLS NETWORK

Ong Chung Yau

IBM DATA Science Capstone Project

SPACEX



Outline

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - EDA with Data Visualization
  - EDA with SQL
  - Web Dashboard with Ploty Dash
  - Predictive Analysis
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context

This project is to predict Falcon 9 first stage landing outcome. Falcon 9 cost 62 million USD; while other providers cost more than 165 million USD. The reason of Falcon 9 cost lower than it competitors is because of reusable rocket. In conclusion, if we can predict the landing result, we are able to calculate the cost of a launch.

- Problems to be answer
  - What attributes affect rocket landing result.
  - Relationship between attributes to determine the rate of landing result.
  - Conditions for rocket to get best results and ensure the best rocket success landing rate.

Methodology

1. Data Collection
    - SpaceX API
    - Web Scrapping
2. Data Wrangling
    - Clean unnecessary data
    - Transform data for Machine Learning
3. EDA with Data Visualization and SQL
    - Show relationship between different attributes
    - Show pattern of data
4. Interactive Visual Analytics
    - Folium
    - Ploty Dash
5. Predictive Analysis
    - Build
    - Tune
    - Evaluate

1. Data Collection

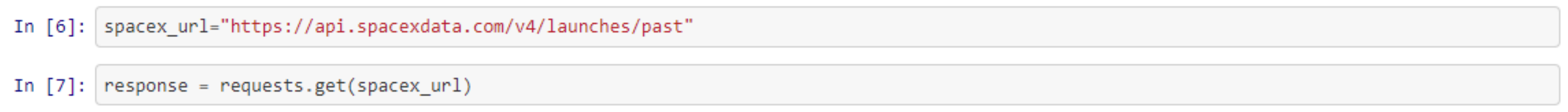
- SpaceX API
  - This API will give us data regarding on SpaceX launches in JSON format.
- Web Scrapping from Wikipedia
  - This will give us data regarding on SpaceX launches in HTML format.

The goal of this data collection is to use this data to predict whether SpaceX are able to land the rocket.

1.1 Data Collection via SpaceX API

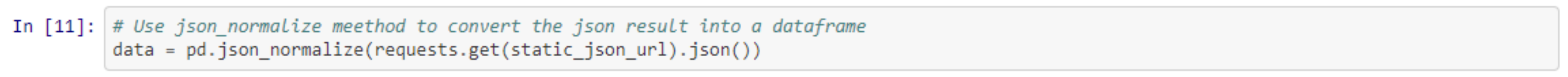
- [IBM Watson Studio URL]
- [Github URL]

1. Get Response



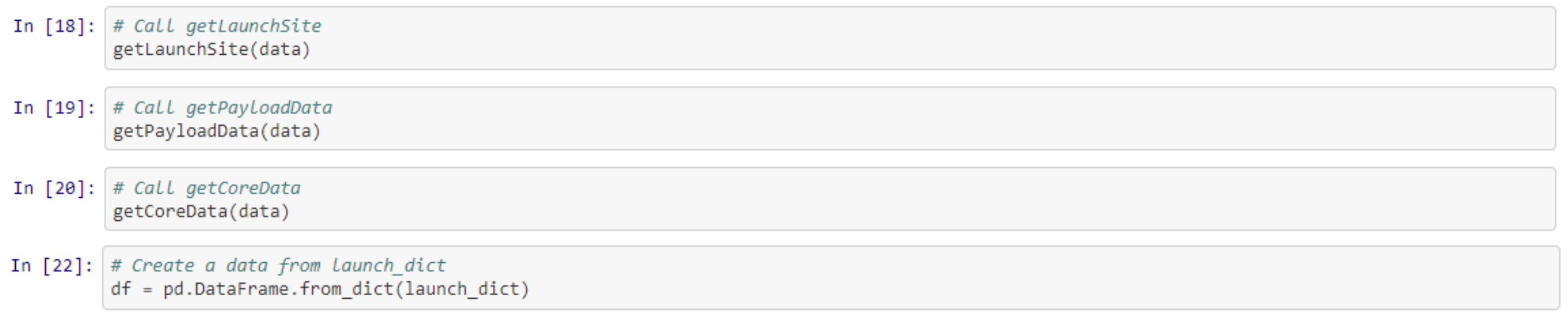
```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

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

2. Convert JSON into Pandas dataframe



```
In [11]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(requests.get(static_json_url).json())
```

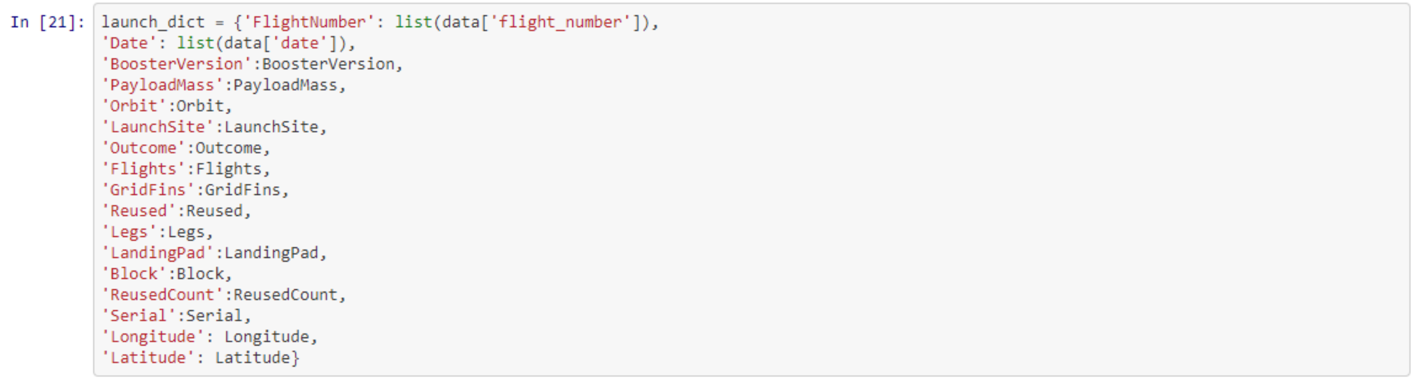
3. Clean data



```
In [18]: # Call getLaunchSite
         getLaunchSite(data)

In [19]: # Call getPayloadData
         getPayloadData(data)

In [20]: # Call getCoreData
         getCoreData(data)

In [22]: # Create a data from launch_dict
         df = pd.DataFrame.from_dict(launch_dict)
```

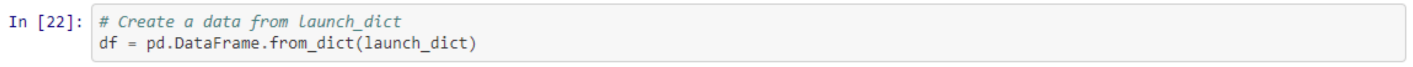
1.1 Data Collection via SpaceX API

4 Combine clean data into a dictionary



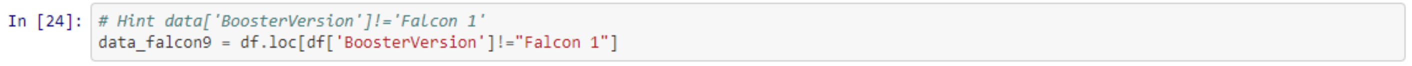
```
In [21]: launch_dict = {'FlightNumber': list(data['flight_number']),
         'Date': list(data['date']),
         'BoosterVersion':BoosterVersion,
         'PayloadMass':PayloadMass,
         'Orbit':Orbit,
         'LaunchSite':LaunchSite,
         'Outcome':Outcome,
         'Flights':Flights,
         'GridFins':GridFins,
         'Reused':Reused,
         'Legs':Legs,
         'LandingPad':LandingPad,
         'Block':Block,
         'ReusedCount':ReusedCount,
         'Serial':Serial,
         'Longitude': Longitude,
         'Latitude': Latitude}
```

5 Convert dictionary into dataframe

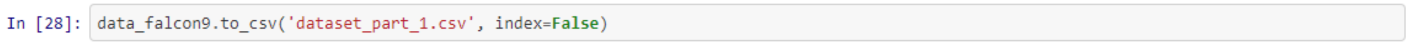


```
In [22]: # Create a data from launch_dict
         df = pd.DataFrame.from_dict(launch_dict)
```

6 Filter and export data



```
In [24]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```



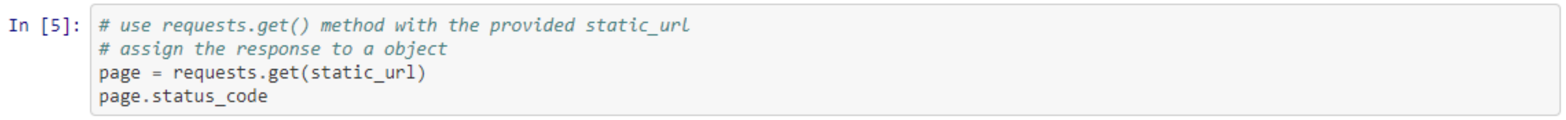
```
In [28]: data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

- IBM Watson Studio URL
- Github URL

1.2 Data Collection via Web Scrapping

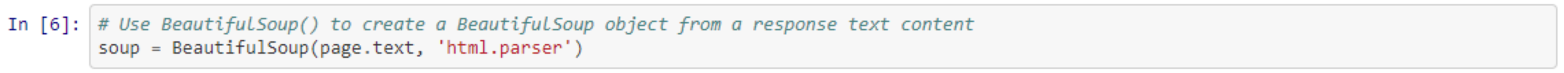
- IBM Watson Studio URL
- Github URL

1. Get Response



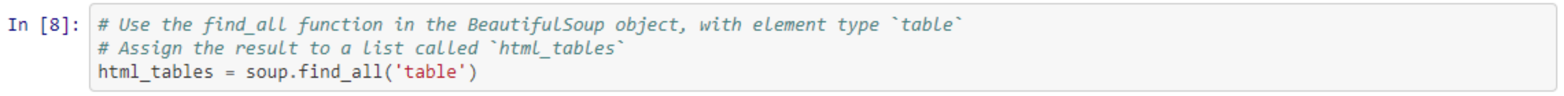
```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        page = requests.get(static_url)
        page.status_code
```

2. Create a BeautifulSoup Object



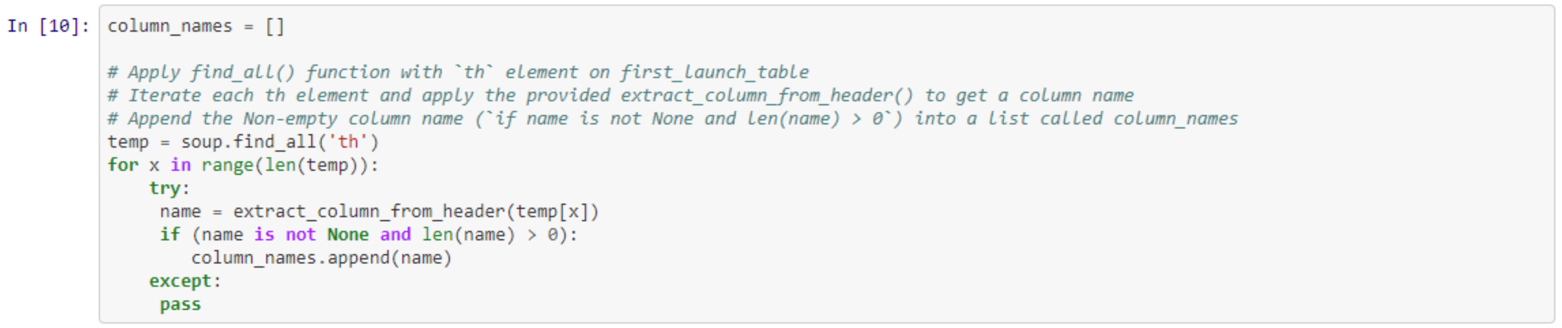
```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(page.text, 'html.parser')
```

3. Get HTML ‘table’ tag



```
In [8]: # Use the find_all function in the BeautifulSoup object, with element type `table`
        # Assign the result to a list called `html_tables`
        html_tables = soup.find_all('table')
```

4. Get column name

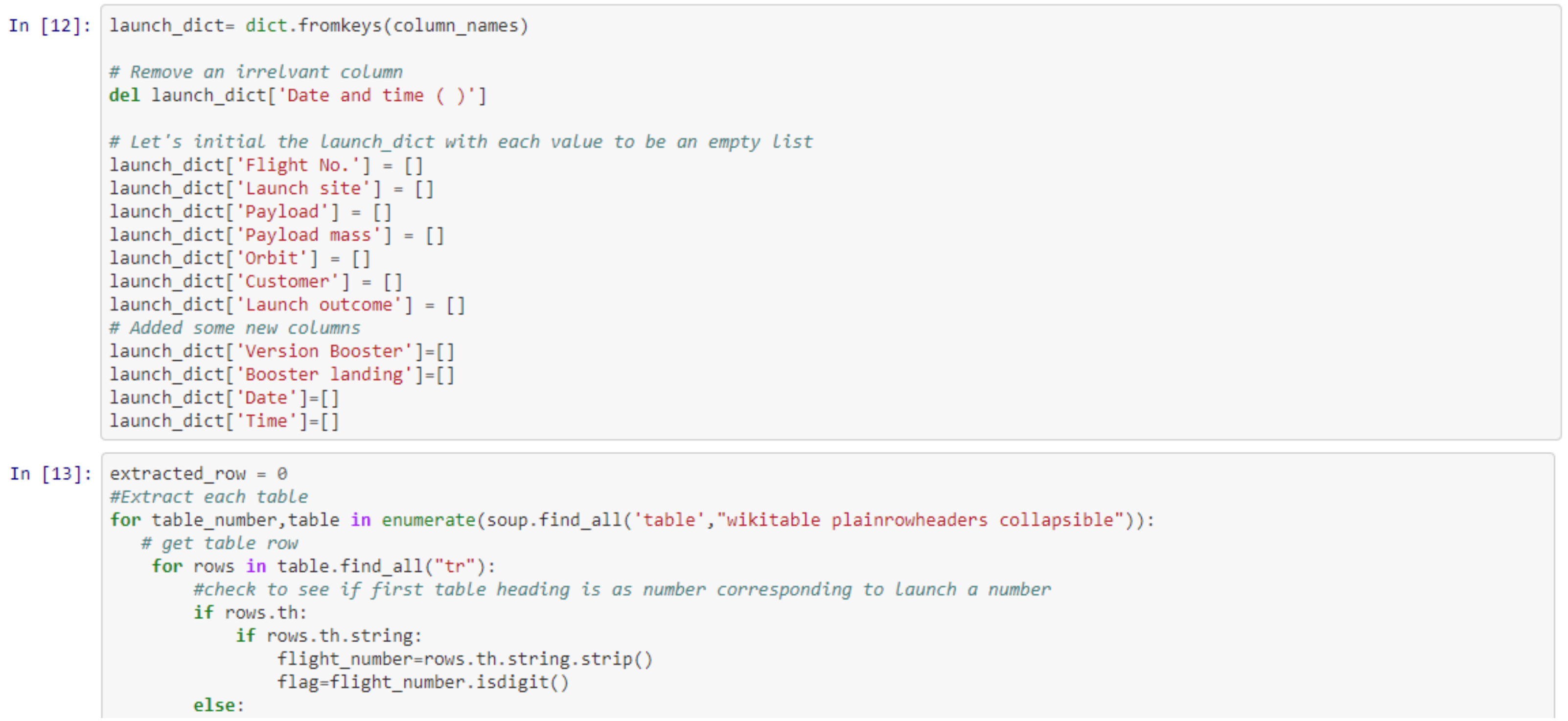


```
In [10]: column_names = []

         # 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
         temp = soup.find_all('th')
         for x in range(len(temp)):
             try:
              name = extract_column_from_header(temp[x])
              if (name is not None and len(name) > 0):
                 column_names.append(name)
             except:
              pass
```

1.2 Data Collection via Web Scrapping

5. Combine data into dictionary

- IBM Watson Studio URL
- Github URL

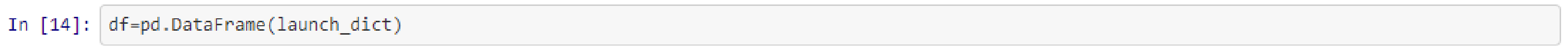


```
In [12]: launch_dict= dict.fromkeys(column_names)

         # 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']=[]
```

```
In [13]: extracted_row = 0
         #Extract each table
         for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
            # get table row
             for rows in table.find_all("tr"):
                 #check to see if first table heading is as number corresponding to launch a number
                 if rows.th:
                     if rows.th.string:
                         flight_number=rows.th.string.strip()
                         flag=flight_number.isdigit()
                 else:
```

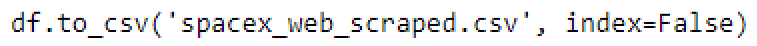
1.2 Data Collection via Web Scrapping

6. Convert dictionary into Pandas Dataframe



```
In [14]: df=pd.DataFrame(launch_dict)
```

7. Export data into CSV file



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

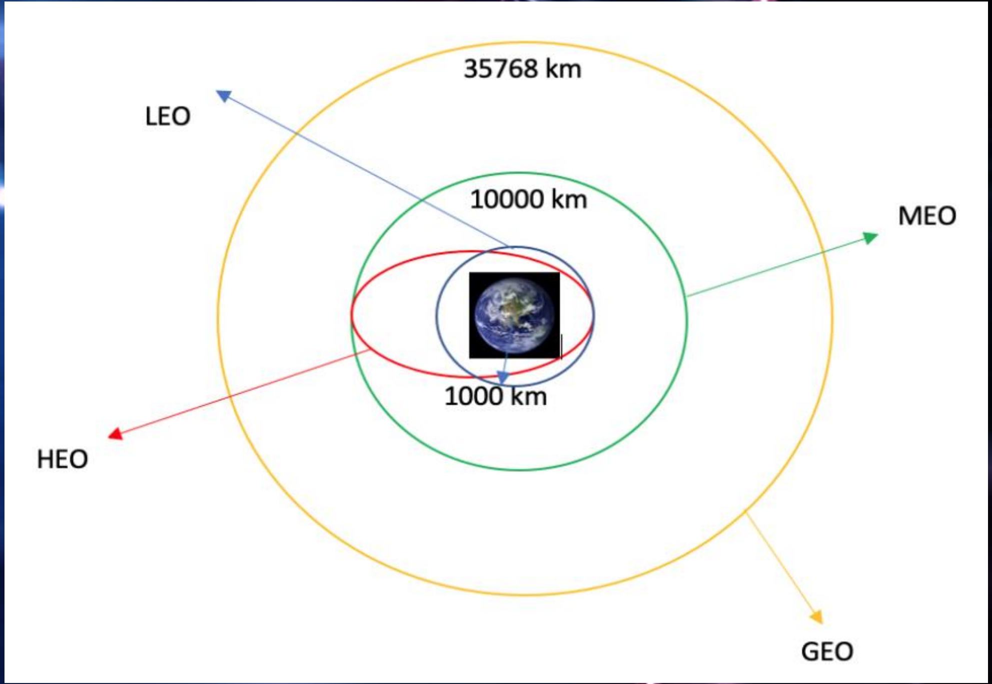
- IBM Watson Studio URL
- Github URL

2. Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.

We will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful. We will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Each launch aims to an dedicated orbit, and here are some common orbit types:



Task 1: Calculate the number of launches on each site

Task 2: Calculate the number and occurrence of each orbit

Task 3: Calculate the number and occurrence of mission outcome per orbit type

Task 4: Create a landing outcome label from Outcome column

Task 5: Export data into CSV file

- IBM Watson Studio URL
- Github RL

3.1 EDA with Data Visualization

- Scatter Graphs:
  - Flight Number VS. Payload Mass
  - Flight Number VS. Launch Site
  - Payload VS. Launch Site
  - Orbit VS. Flight Number
  - Payload VS. Orbit Type

Scatter plots show how much one variable is affected by another. The relationship between two variable is called correlation. Scatter plots usually consist of a large body of data.

- Bar Graph:
  - Mean VS. Orbit

A bar diagram make it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar graph can also show big changes in data over time.

- Line Graph
  - Success Rate VS. Year

Line graphs are useful in that they show data variable and trends very clearly and can help to make predictions about the results of data not yet recorded.

- IBM Watson Studio URL
- Github URL

3.2 EDA With SQL

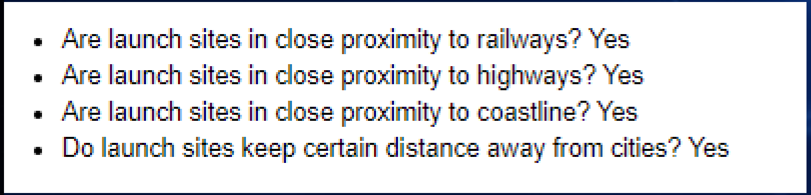
Performed SQL queries to gather information about the dataset

1. **Display the names of the unique launch sites in the space mission**
2. **Display 5 records where launch sites begin with the string 'CCA'**
3. **Display the total payload mass carried by boosters launched by NASA (CRS)**
4. **Display average payload mass carried by booster version F9 v1.1**
5. **List the date when the first successful landing outcome in ground pad was acheived**
6. **List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000**
7. **List the total number of successful and failure mission outcomes**
8. **List the names of the booster_versions which have carried the maximum payload mass.**
9. **List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015**
10. **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**

- IBM Watson Studio URL
- Github URL

4.1 Build an Interactive Map with Folium

- **Visualize launch data in an interactive map.** Using the coordinate (Latitude and Longitude) of each launch site and add circle marker around each launch site with launch site name as a label.
- **Project launch_outcomes (success and fail) in an interactive map.** Assigning classes 0 and 1 with Green and Red markers on the map in a Marker Cluster.
- **Calculate distance of Launch Site to different various of landmarks.** This is to find various trend about what is around the launch site



- Are launch sites in close proximity to railways? Yes
- Are launch sites in close proximity to highways? Yes
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

- IBM Watson Studio URL
- Github URL

4.2 Build a Dashboard with Plotly Dash

- **Pie Chart**
  - Showing the total success of all launch sites
  - Showing the total success and fail of each launch site
  - Allow us to identify which site has the largest successful launches
  - Allow us to identify which site has the highest launch success rate

- **Scatter Chart**
  - Showing the relationship of launch outcome and Payload Mass (kg)
  - Allow us to identify which payload range(s) has the highest launch success rate
  - Allow us to identify which payload range(s) has the lowest launch success rate
  - Allow us to identify which F9 Booster version has the highest launch success rate

- Github URL

5 Predictive Analysis (Classification)

- **Building Model**
    - Load dataset into dataframe
    - Transform Data
    - Split dataset into training and test dataset
    - Decide which type of machine learning algorithms to be use
    - Set our parameters and algorithms into GridSearchCV
    - Fit datasets into the GridSearchCV objects and train our model
- Evaluate Model
    - Check Accuracy of each model
    - Tune hyperparameters of each model
    - Plot Confusion Matrix
- Improve Model
    - Feature Engineering
    - Algorithm Tuning
- Identify Suitable Classification Model
    - Model with the highest accuracy will be selected as suitable model

- IBM Watson Studio URL
- Github URL

Results

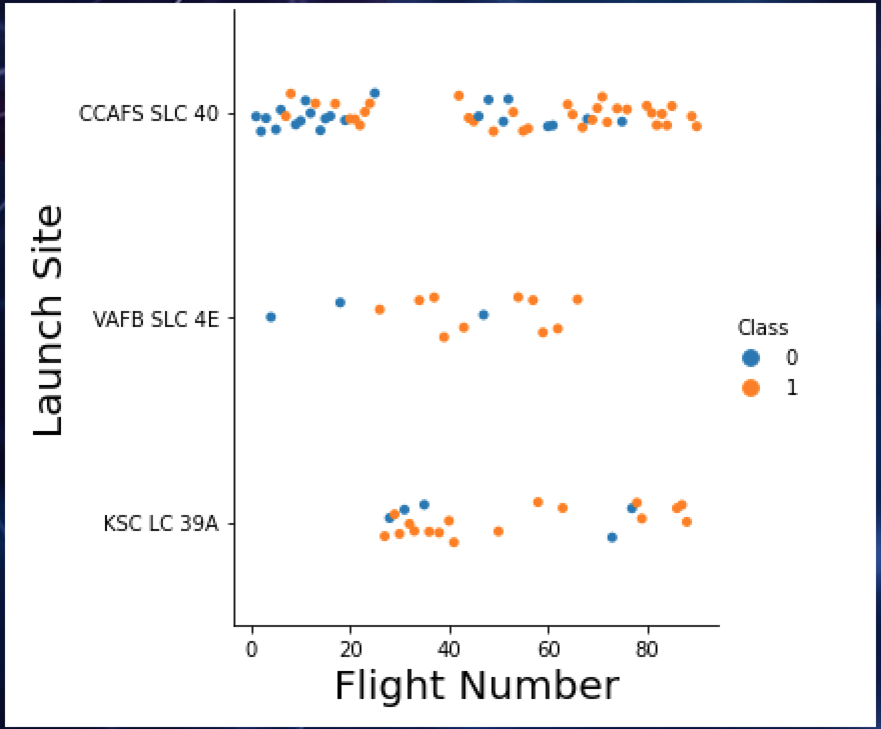
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Results

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

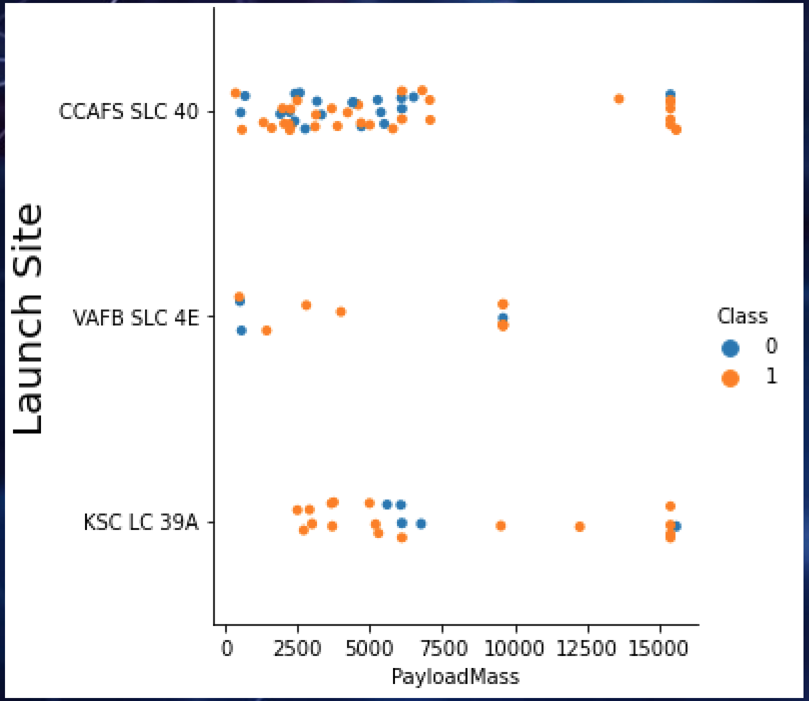
Insights drawn from EDA

Flight Number vs. Launch Site



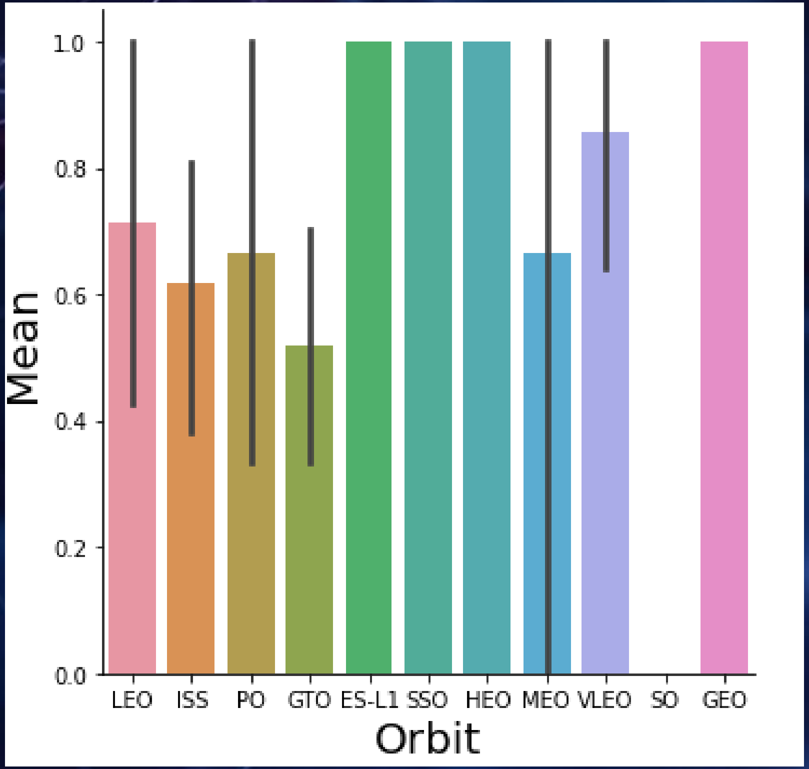
Explanation: The higher amount of flight number carrry out the higher the success rate of a launch outcome.

Payload vs. Launch Site



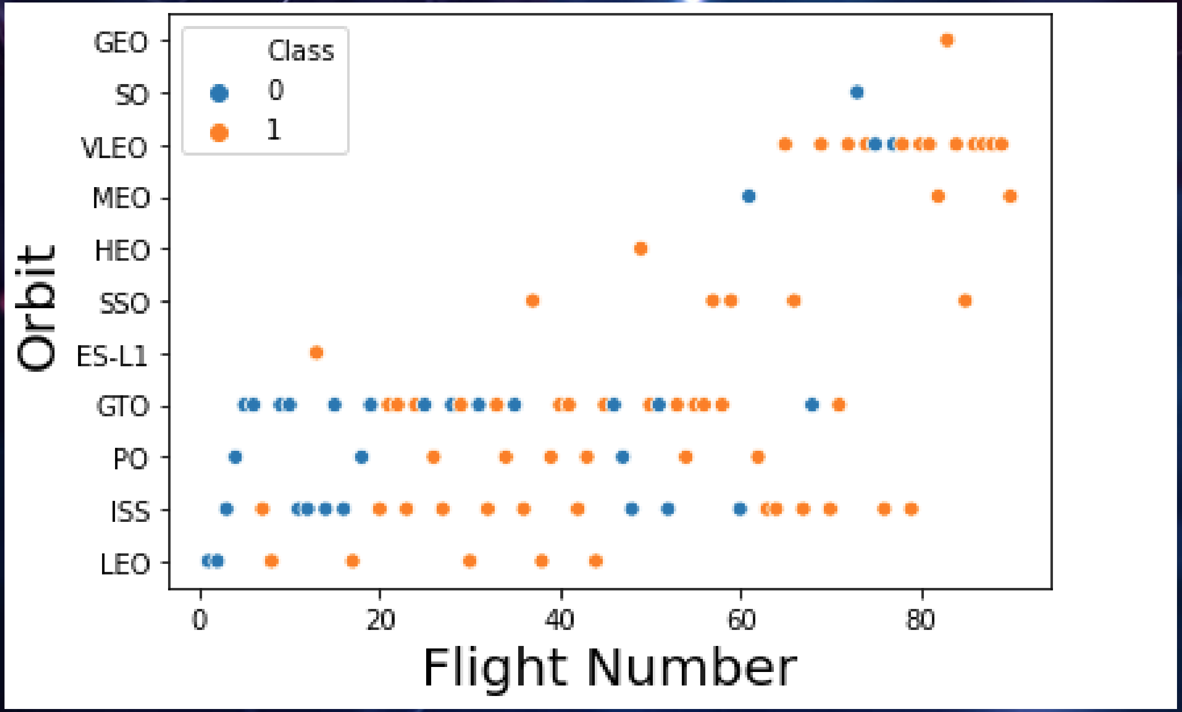
Explanation: The greater the payload mass the success rate of the launch outcome. There is no clear pattern to be found using Launch Site and Payload Mass to determine launch outcome.

Success Rate vs. Orbit Type



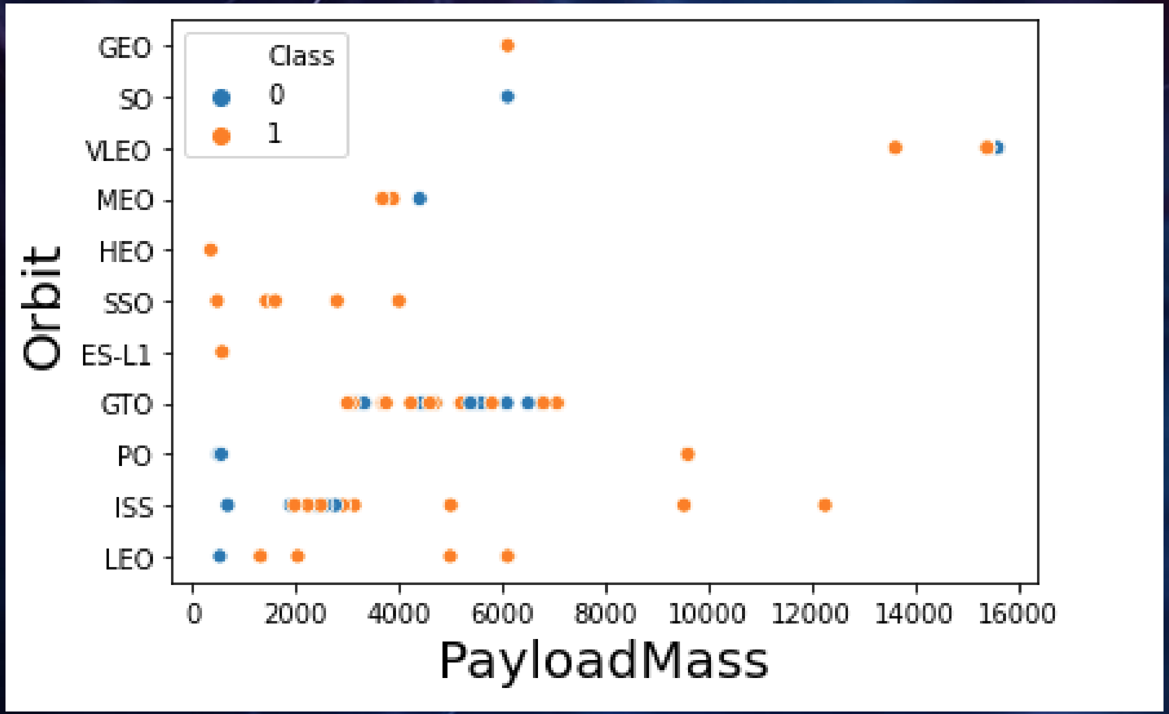
Explanation: Orbit type ES-L1, SSO, HEO and GEO have the best success rate.

Flight Number vs. Orbit Type



You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

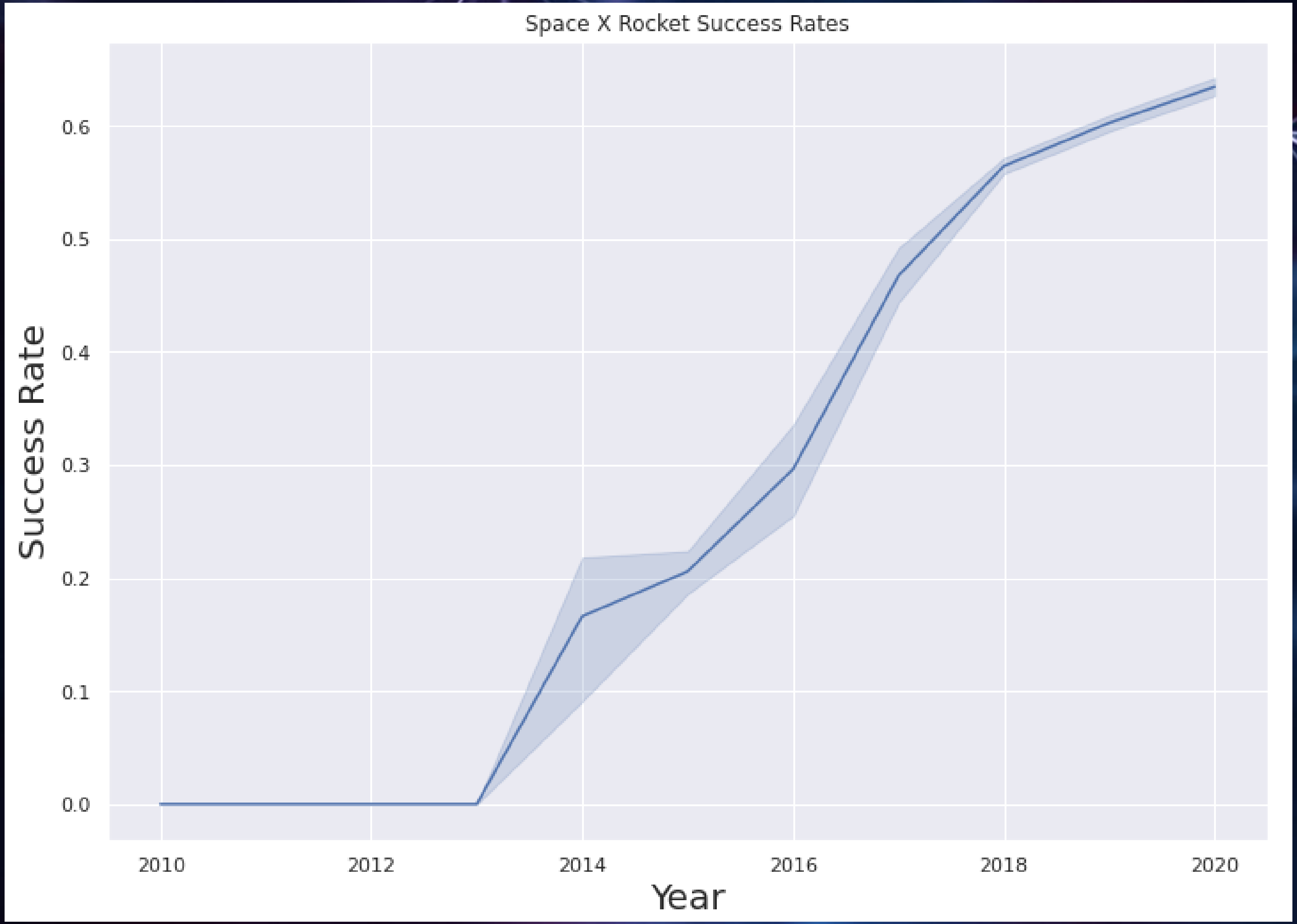
Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

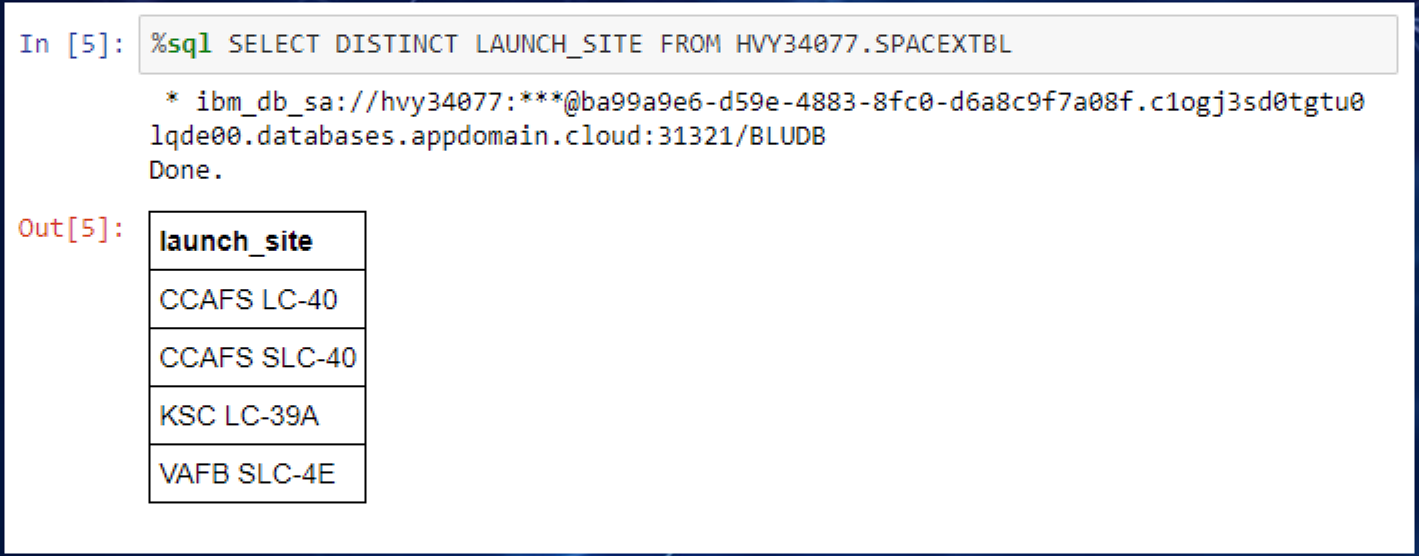
Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

EDA with SQL

All Launch Site Names



```
In [5]: %sql SELECT DISTINCT LAUNCH_SITE FROM HVY34077.SPACEXTBL

 * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0
lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

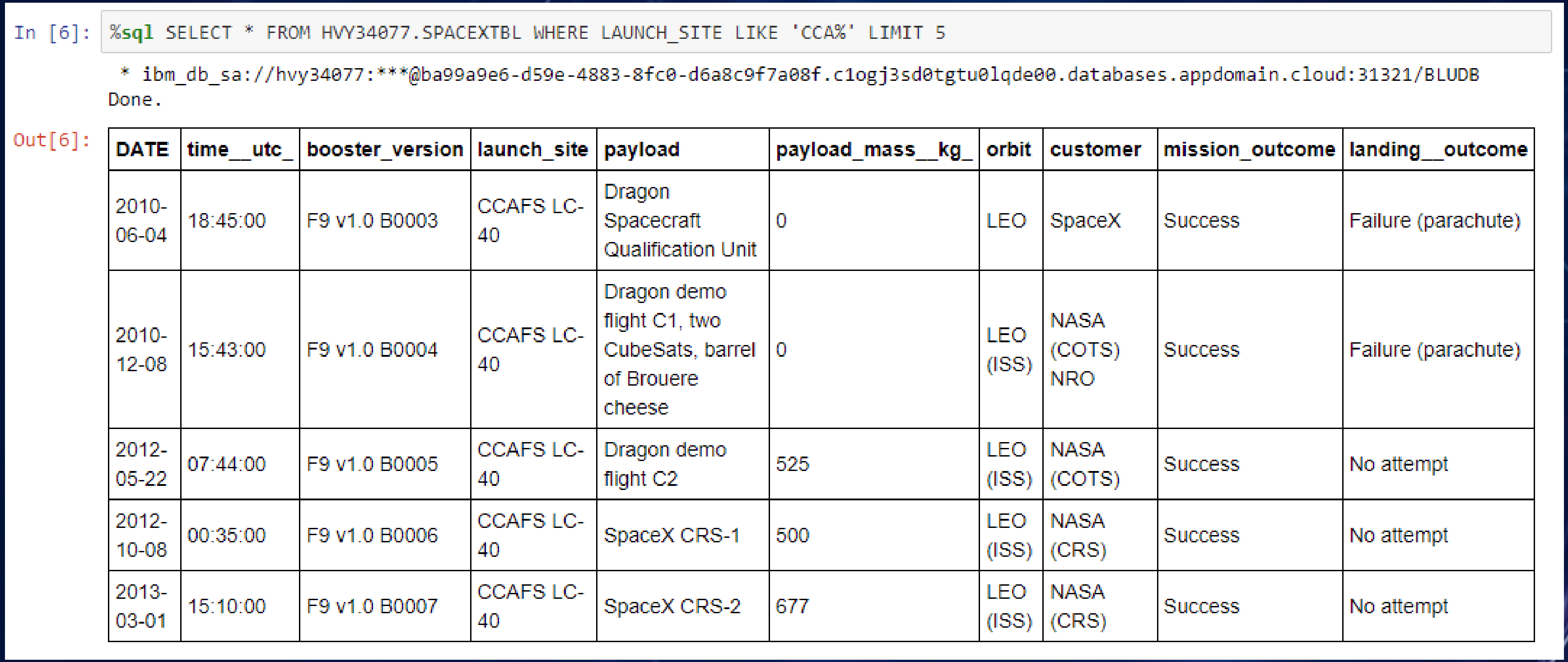
Out[5]:

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

Explanation:

1. **DISTINCT** in query will only show unique values in the Launch_Site column.

Launch Site Names Begin with ‘CCA’



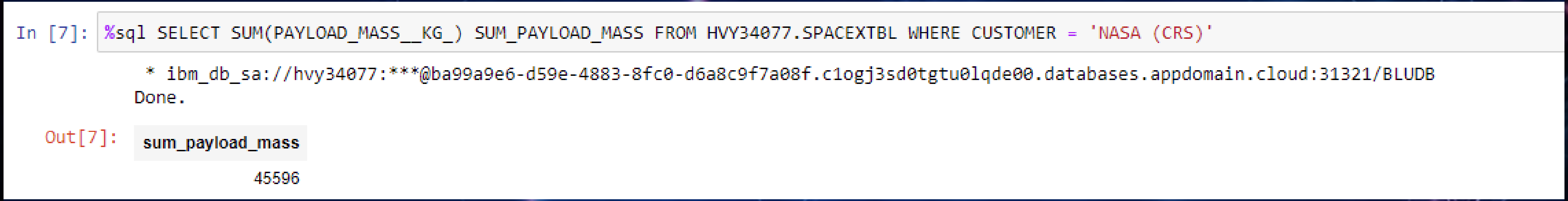
```
In [6]: %sql SELECT * FROM HVY34077.SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5

 * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.

Out[6]:
```

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

Explanation:

1. **LIMIT 5** will only show 5 records
2. **LIKE** keyword has a wild card with words ’CCA%’ , % represent can be combine with any characters

Total Payload Mass



```
In [7]: %sql SELECT SUM(PAYLOAD_MASS__KG_) SUM_PAYLOAD_MASS FROM HVY34077.SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)'
```

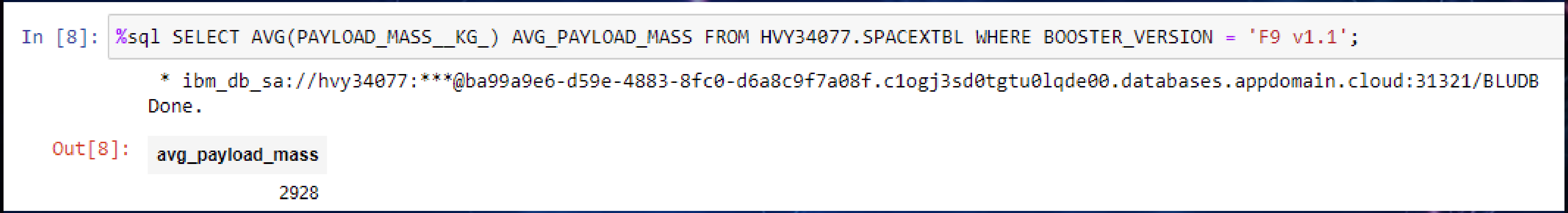
```
 * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

```
Out[7]: sum_payload_mass
                   45596
```

Explanation:

1. **SUM** will add up the total in the column PAYLOAD_MASS__KG_
2. **WHERE** filters out the dataset in CUSTOMER column with data ‘NASA (CRS)’

Average Payload Mass by F9 v1.1

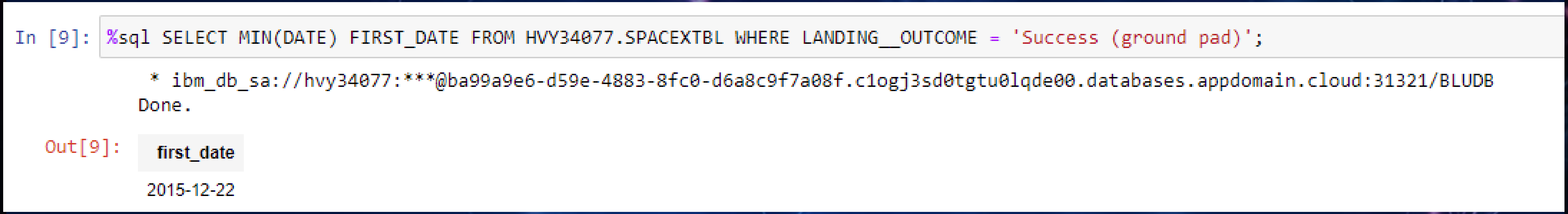


```
In [8]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AVG_PAYLOAD_MASS FROM HVY34077.SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';

         * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
        Done.

Out[8]: avg_payload_mass
                    2928
```

Explanation:

1. **AVG** will calculate the average in the column PAYLOAD_MASS__KG_
2. **WHERE** filters out the dataset in BOOSTER_VERSION column with data ‘F9 v1.1’

First Successful Ground Landing Date

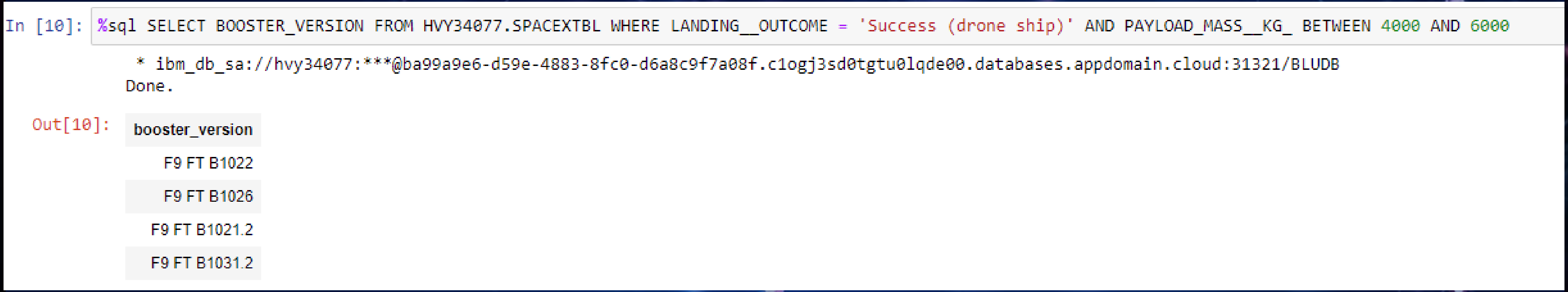


```
In [9]: %sql SELECT MIN(DATE) FIRST_DATE FROM HVY34077.SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)';

 * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.

Out[9]:
first_date
2015-12-22
```

Explanation:

1. **MIN** will find the minimum value in the column DATE
2. **WHERE** filters out the dataset in LANDING__OUTCOME column with data ‘Success (ground pad)’

Successful Drone Ship Landing with Payload between 4000 and 6000

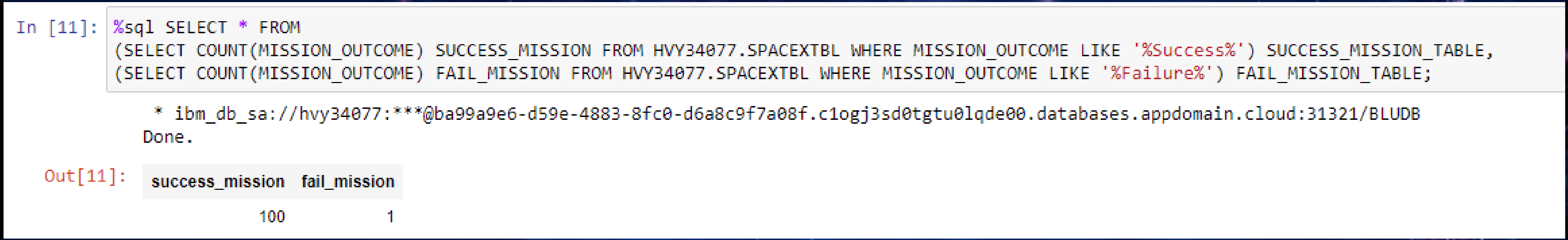


```
In [10]: %sql SELECT BOOSTER_VERSION FROM HVY34077.SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000

          * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
         Done.

Out[10]: booster_version
              F9 FT B1022
              F9 FT B1026
            F9 FT B1021.2
            F9 FT B1031.2
```

Explanation:

1. **WHERE** filters out the dataset in LANDING__OUTCOME column with data 'Success (drone ship)'
2. **BETWEEN** get value within 4000 and 6000 in PAYLOAD_MASS__KG_ column

Total Number of Successful and Failure Mission Outcomes



```
In [11]: %sql SELECT * FROM
         (SELECT COUNT(MISSION_OUTCOME) SUCCESS_MISSION FROM HVY34077.SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Success%') SUCCESS_MISSION_TABLE,
         (SELECT COUNT(MISSION_OUTCOME) FAIL_MISSION FROM HVY34077.SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Failure%') FAIL_MISSION_TABLE;

          * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
         Done.
```

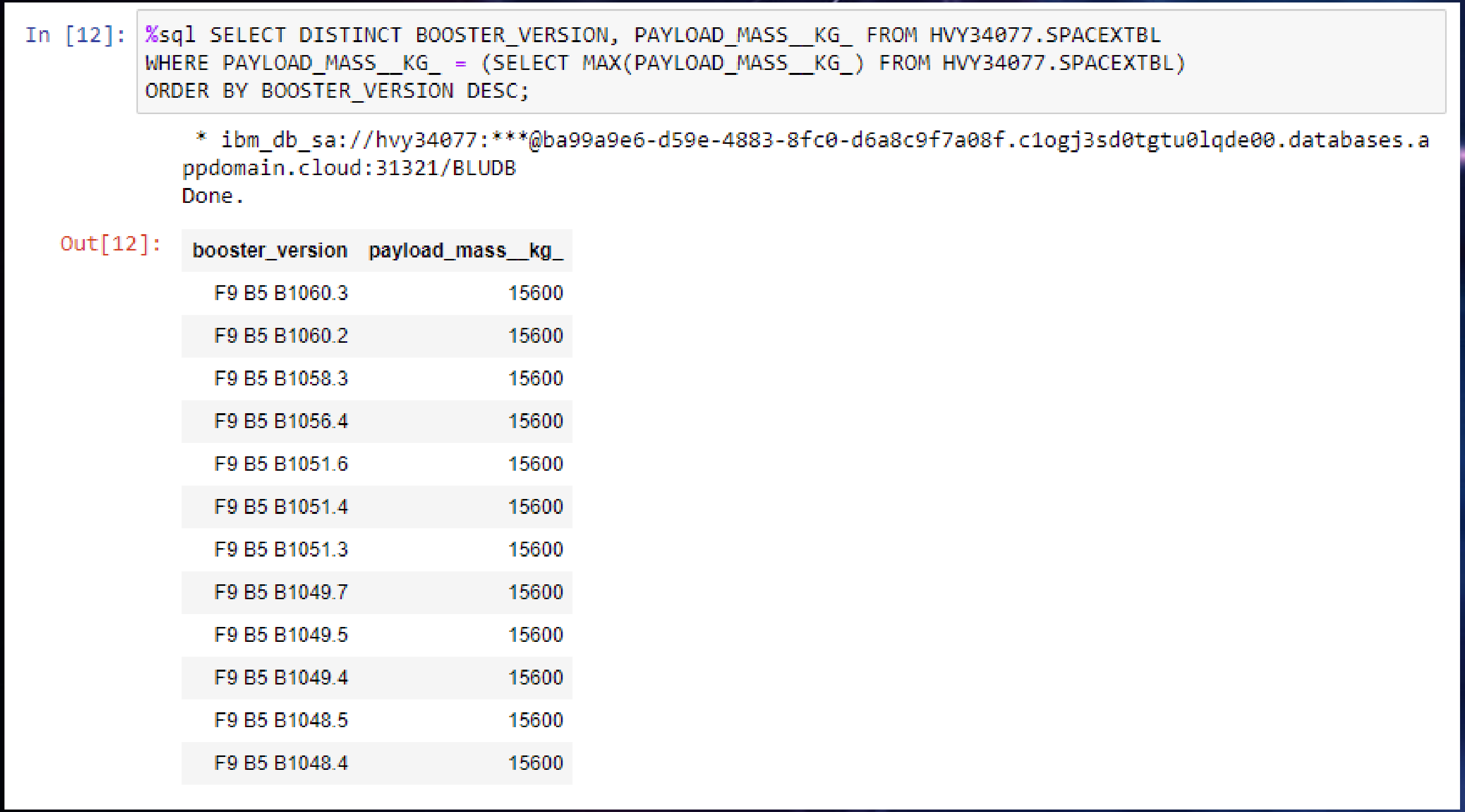
Out[11]:

success_mission	fail_mission
100	1

Explanation:

1. **LIKE** keyword has a wild card with words '%Success%' , % represent can be combine with any characters

Boosters Carried Maximum Payload



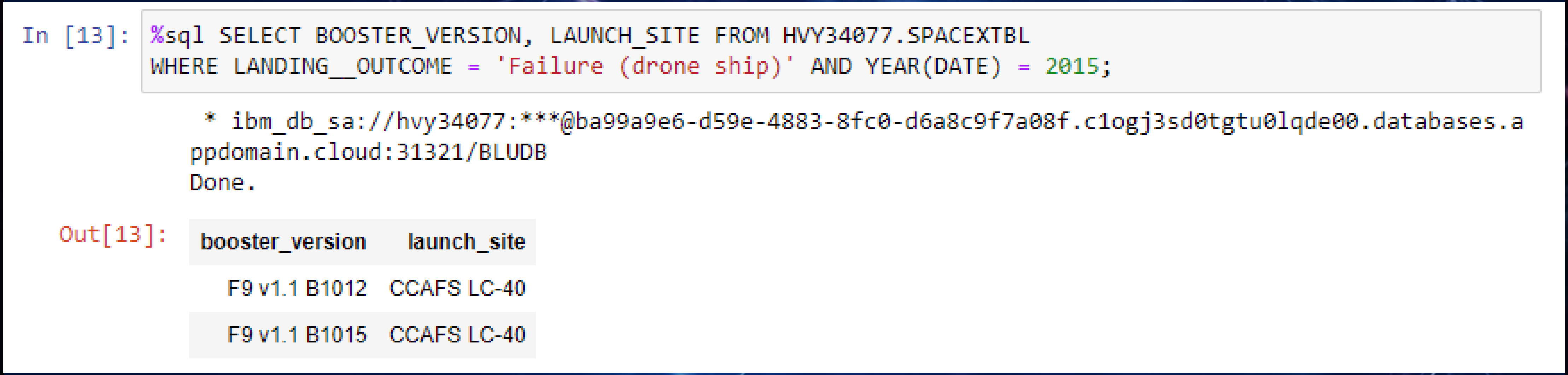
```
In [12]: %sql SELECT DISTINCT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM HVY34077.SPACEXTBL
         WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM HVY34077.SPACEXTBL)
         ORDER BY BOOSTER_VERSION DESC;

          * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.a
         ppdomain.cloud:31321/BLUDB
         Done.

Out[12]:
```

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

Explanation:

1. **DISTINCT** in query will only show unique values in the BOOSTER_VERSION, PAYLOAD_MASS__KG_ column.
2. **WHERE** filter out the dataset in PAYLOAD_MASS__KG_ column with data MAX(PAYLOAD_MASS__KG_)

2015 Launch Records



```
In [13]: %sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM HVY34077.SPACEXTBL
         WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND YEAR(DATE) = 2015;

 * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.a
ppdomain.cloud:31321/BLUDB
Done.
```

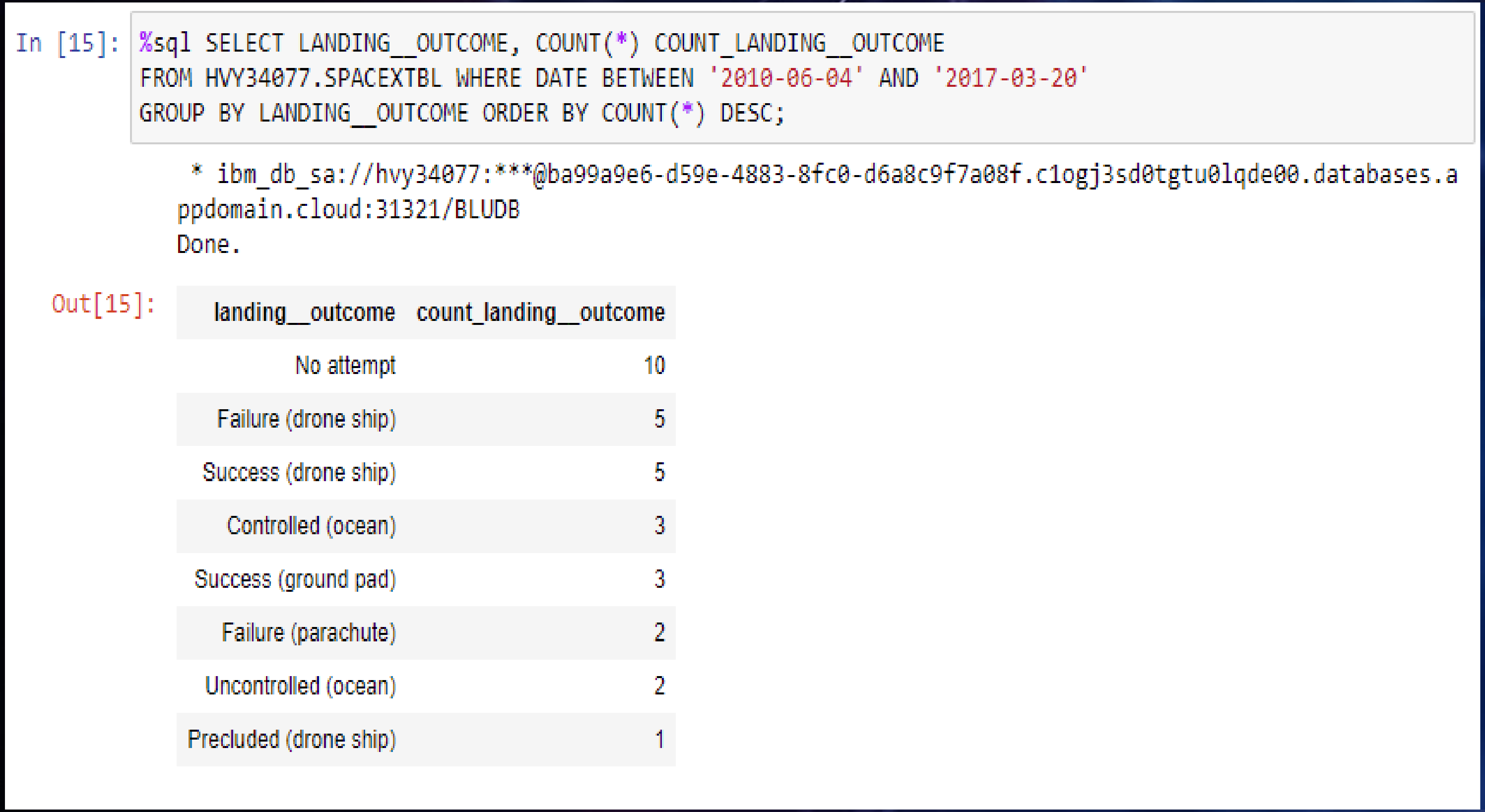
Out[13]:

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

Explanation:

1. **WHERE** filters out the dataset in LANDING__OUTCOME column with data ‘Failure (drone ship)’
2. **YEAR** will only read the year data of DATE column with data 2015

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



```
In [15]: %sql SELECT LANDING__OUTCOME, COUNT(*) COUNT_LANDING__OUTCOME
         FROM HVY34077.SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
         GROUP BY LANDING__OUTCOME ORDER BY COUNT(*) DESC;

 * ibm_db_sa://hvy34077:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.a
ppdomain.cloud:31321/BLUDB
Done.
```

Out[15]:

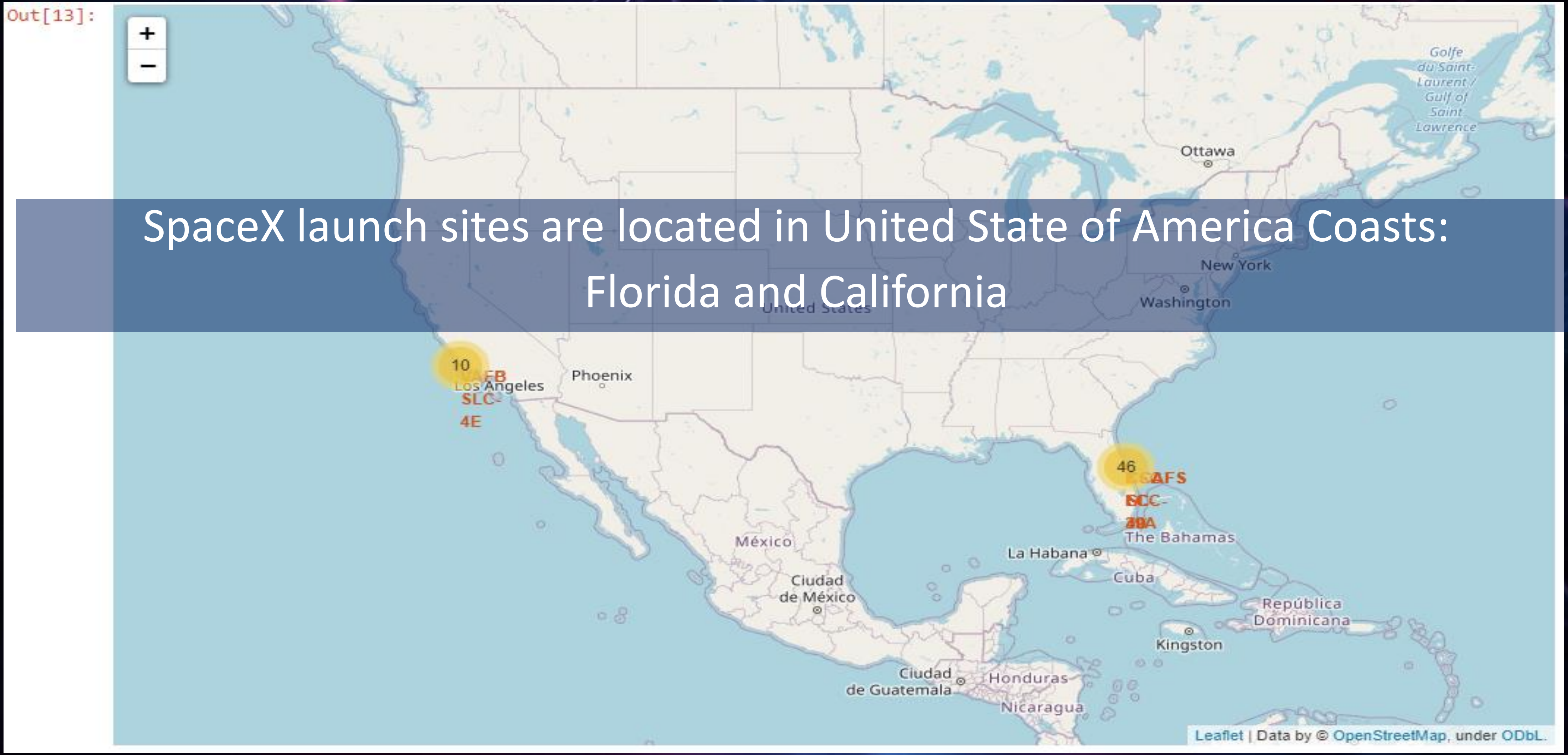
landing__outcome	count_landing__outcome
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Explanation:

1. **COUNT** return number of rows that matches a specified criteria
2. **BETWEEN** get value within 2010-06-04 and 2017-03-20 in DATE column
3. **GROUP BY**, group rows that have the same values in LANDING__OUTCOME into summary rows
4. **ORDER BY DESC** sort the record in descending order

Launch Sites Proximities Analysis

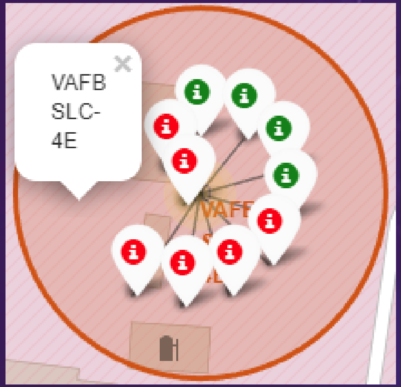
All launch sites' location on a global map



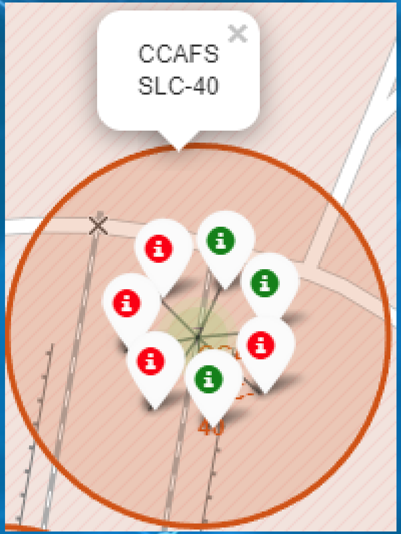
SpaceX launch sites are located in United State of America Coasts: Florida and California

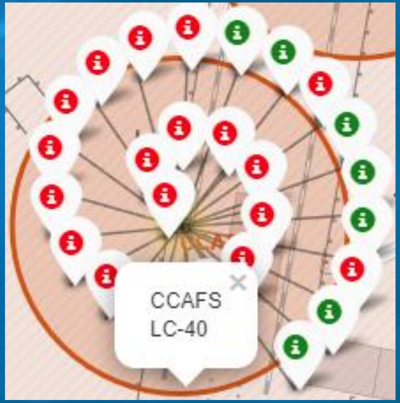
Color-labeled launch outcomes on the map

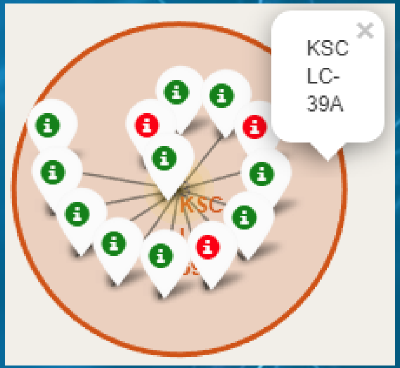
California Launch Site



Florida Launch Site



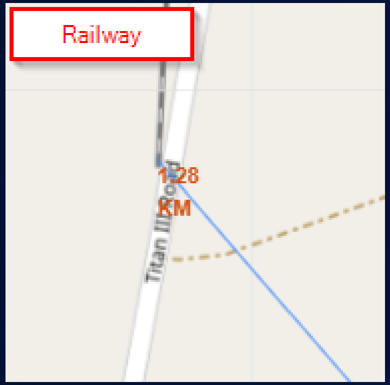


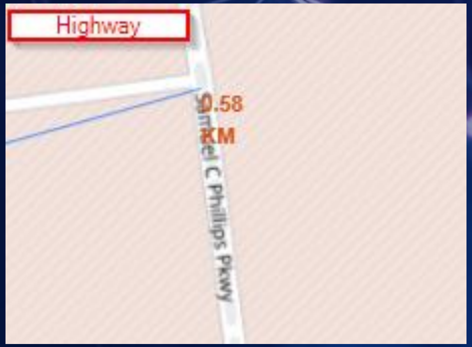


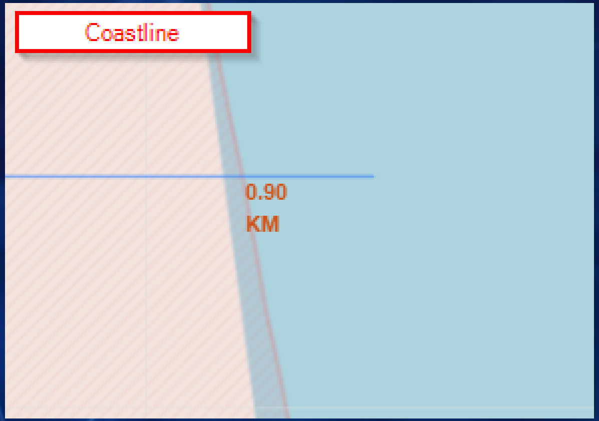
Green Marker shows successful launches

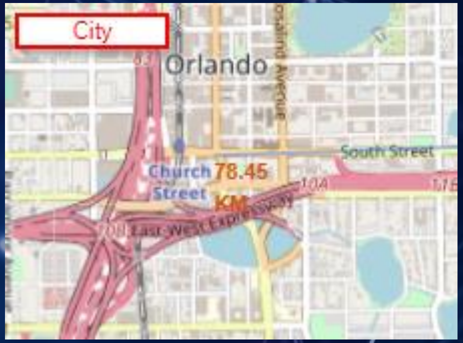
Red Marker shows failure launches

Color-labeled launch outcomes on the map







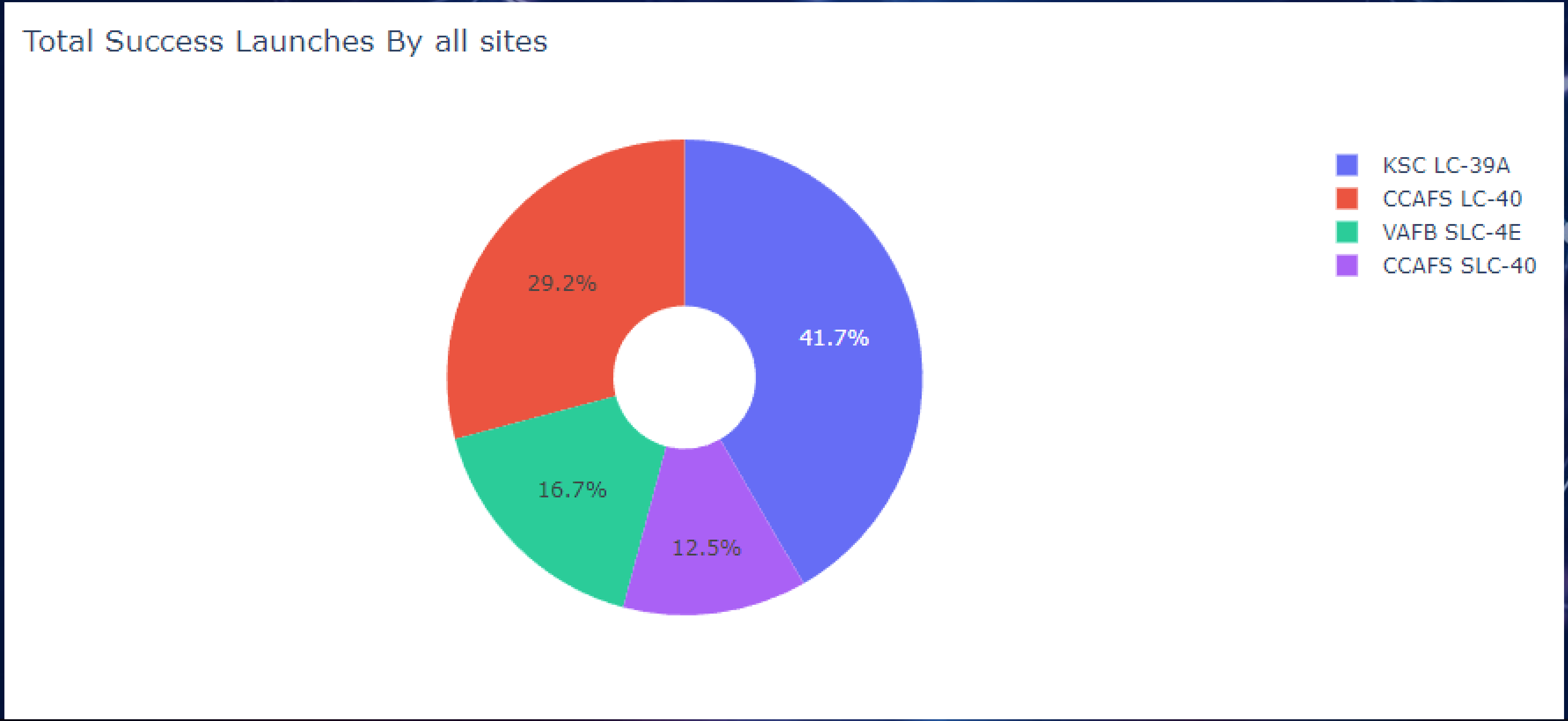


- Are launch sites in close proximity to railways? Yes
- Are launch sites in close proximity to highways? Yes
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Build a Dashboard with Plotly Dash

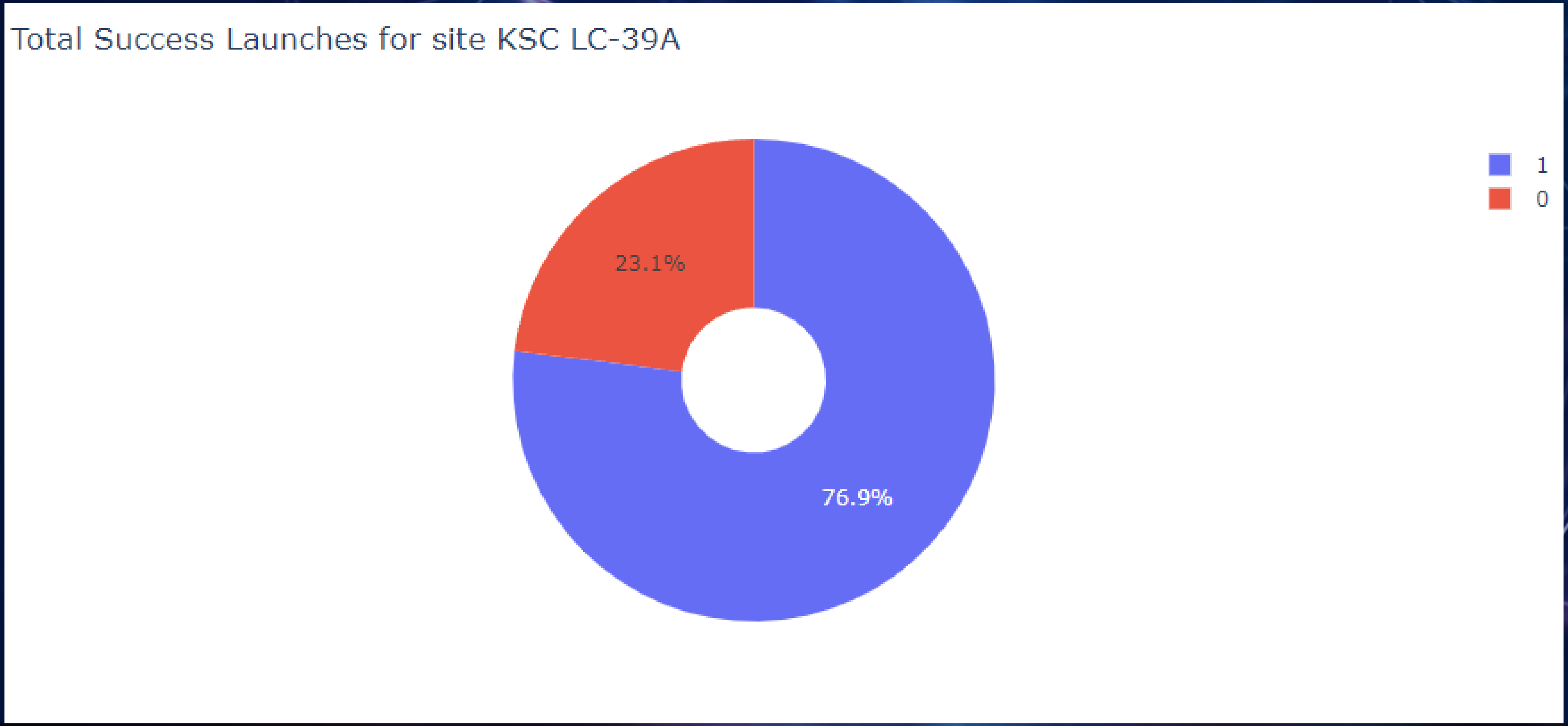
Pie Chart – Launch success count for all sites

KSC LC-39A launch site has the highest count of success with 41.7%



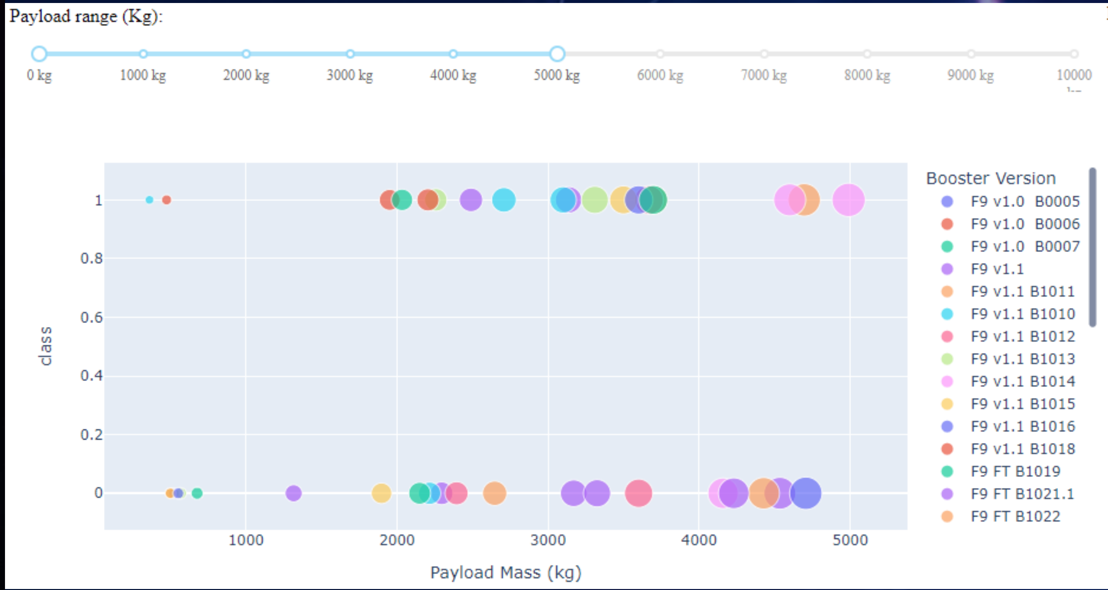
Pie Chart – Launch site with highest launch success ratio

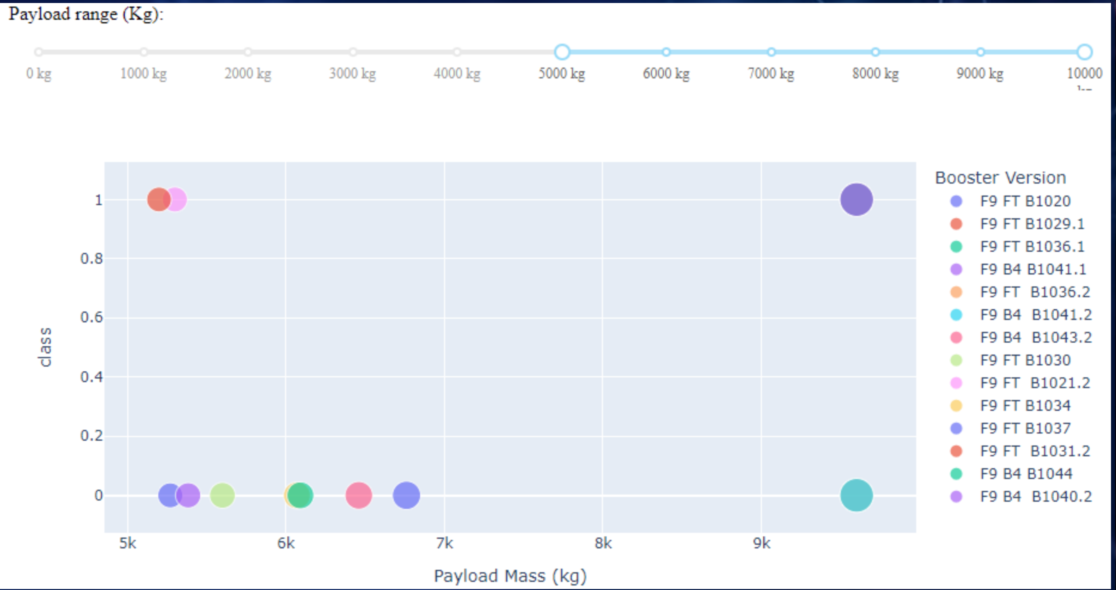
KSC LC-39A has the highest launch success ratio with 76.9%



Scatter Plot – Launch Outcome vs. Payload Mass (kg)

Payload range (kg) 0 kg – 5000 kg have a higher success rate than Payload Range (kg) 5000 kg – 10000 kg



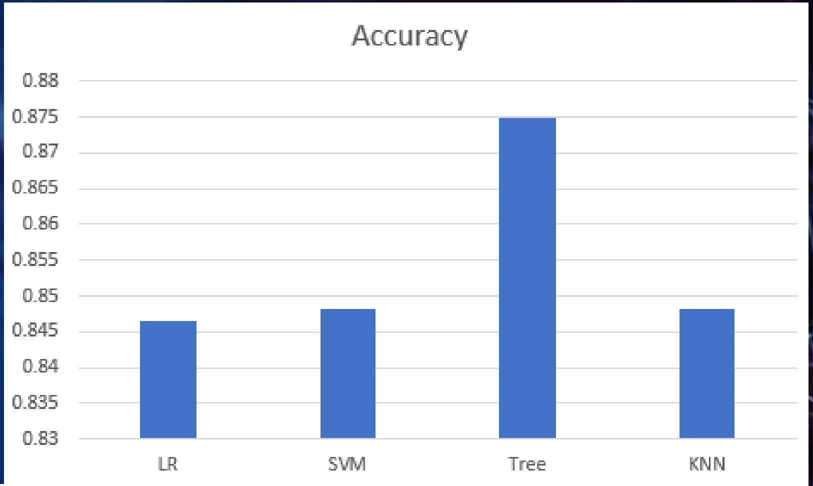


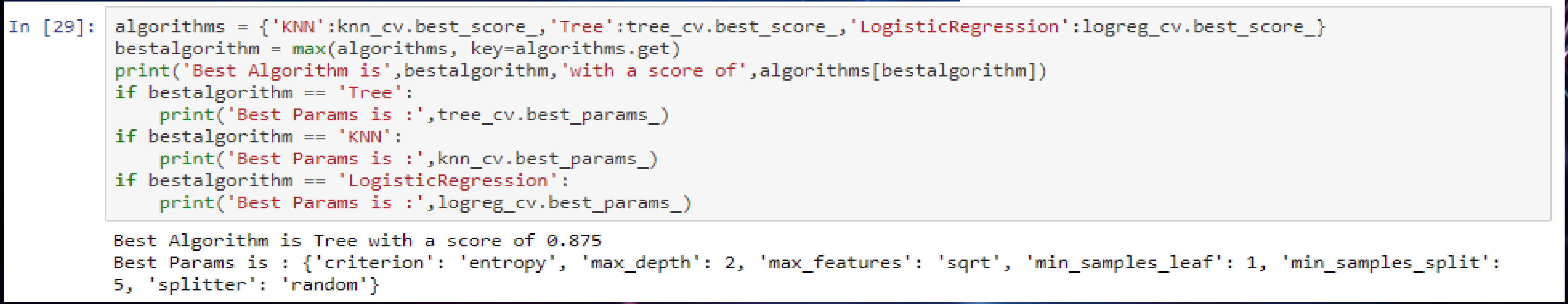
Predictive Analysis (Classification)

Classification Accuracy

Based on the bar chart; Logistic Regression (LR), Support Vector Machine (SVM) and K-nearest neighbor (KNN) have very similar accuracy result.

The best algorithm based on the bar chart is Tree Classifier with 0.875 accuracy.

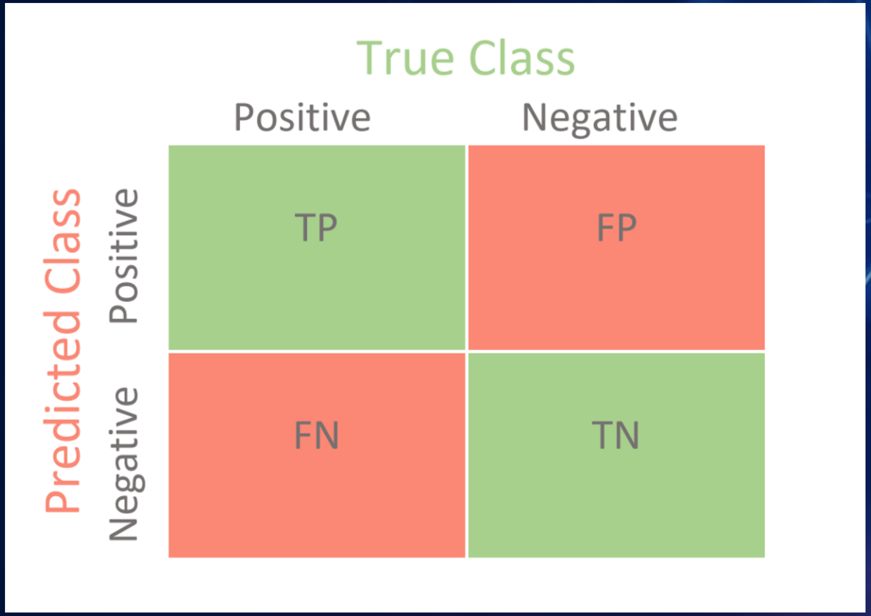


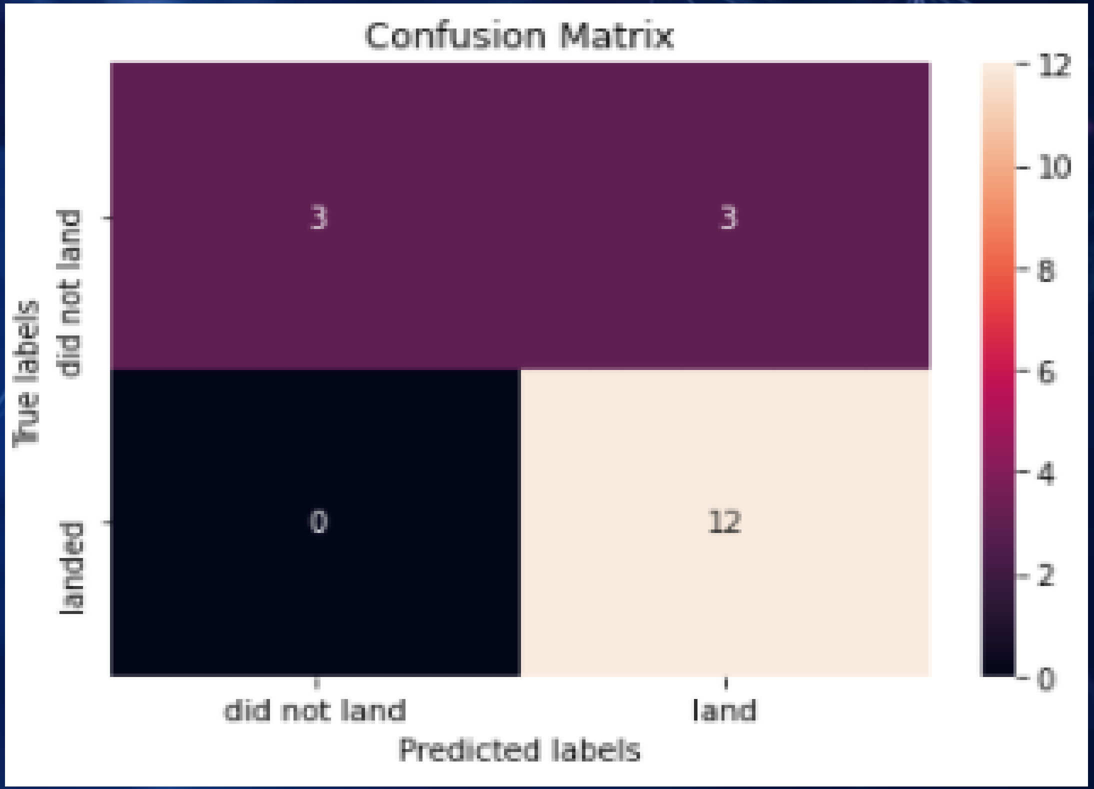


```
In [29]: algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
         bestalgorithm = max(algorithms, key=algorithms.get)
         print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
         if bestalgorithm == 'Tree':
             print('Best Params is :',tree_cv.best_params_)
         if bestalgorithm == 'KNN':
             print('Best Params is :',knn_cv.best_params_)
         if bestalgorithm == 'LogisticRegression':
             print('Best Params is :',logreg_cv.best_params_)

         Best Algorithm is Tree with a score of 0.875
         Best Params is : {'criterion': 'entropy', 'max_depth': 2, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split':
         5, 'splitter': 'random'}
```

Confusion Matrix

Based on the confusion matrix, we are able to observe Tree can distinguish True Positive, False Negative and True Negative correctly, but it have a problem in False Positive.





Conclusions

- Tree Classifier Algorithm is the best for Machine Learning for this dataset.
- Low weighted payloads perform better than the heavier payloads.
- Success rate of SpaceX launches will be higher as time goes longer.
- KSC LC-39A has the most successful launches from all the sites.
- Orbit type ES-L1, GEO, HEO and SSO has the best success rate.

Appendix

None

Thank You