IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Olena Kirieva
14.01.2025

Outline

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

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

- Problems you want to find answers

  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

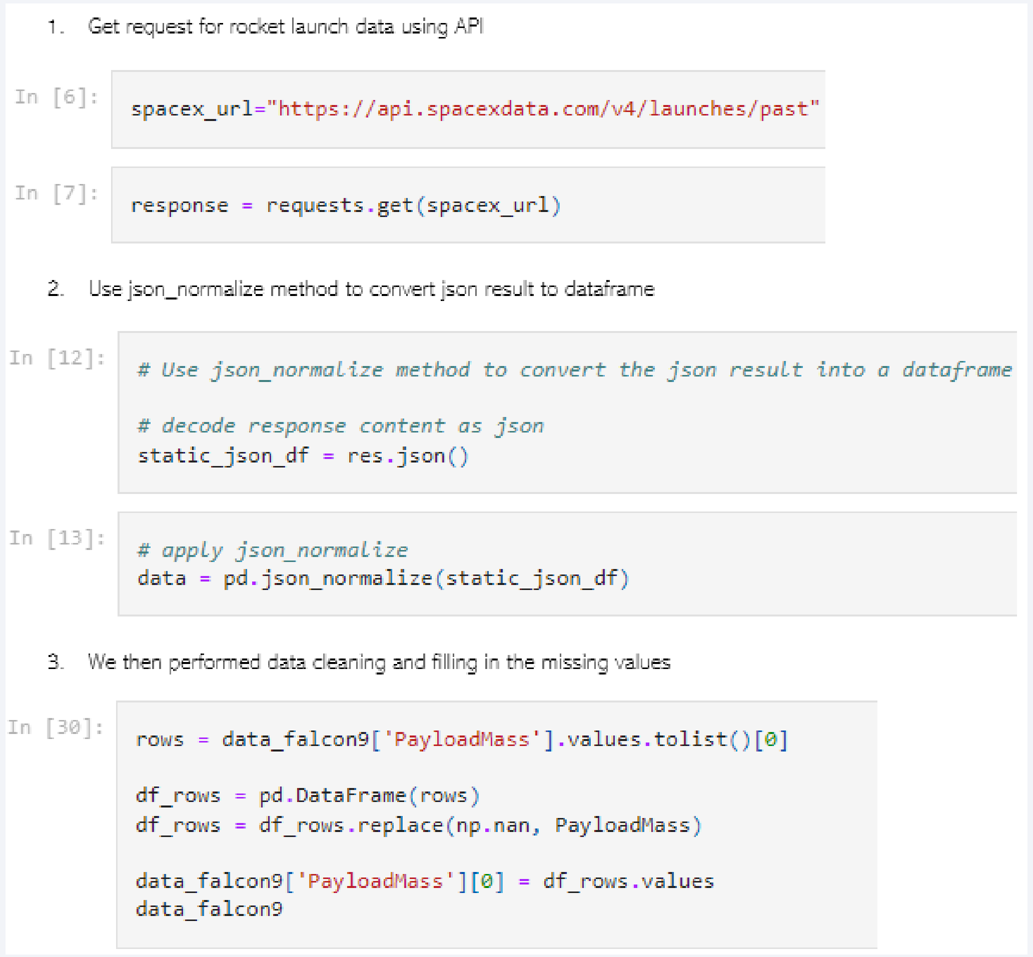
- Data collection methodology:
    - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
    - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - How to build, tune, evaluate classification models

Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is github



1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe

         # decode response content as json
         static_json_df = res.json()
```

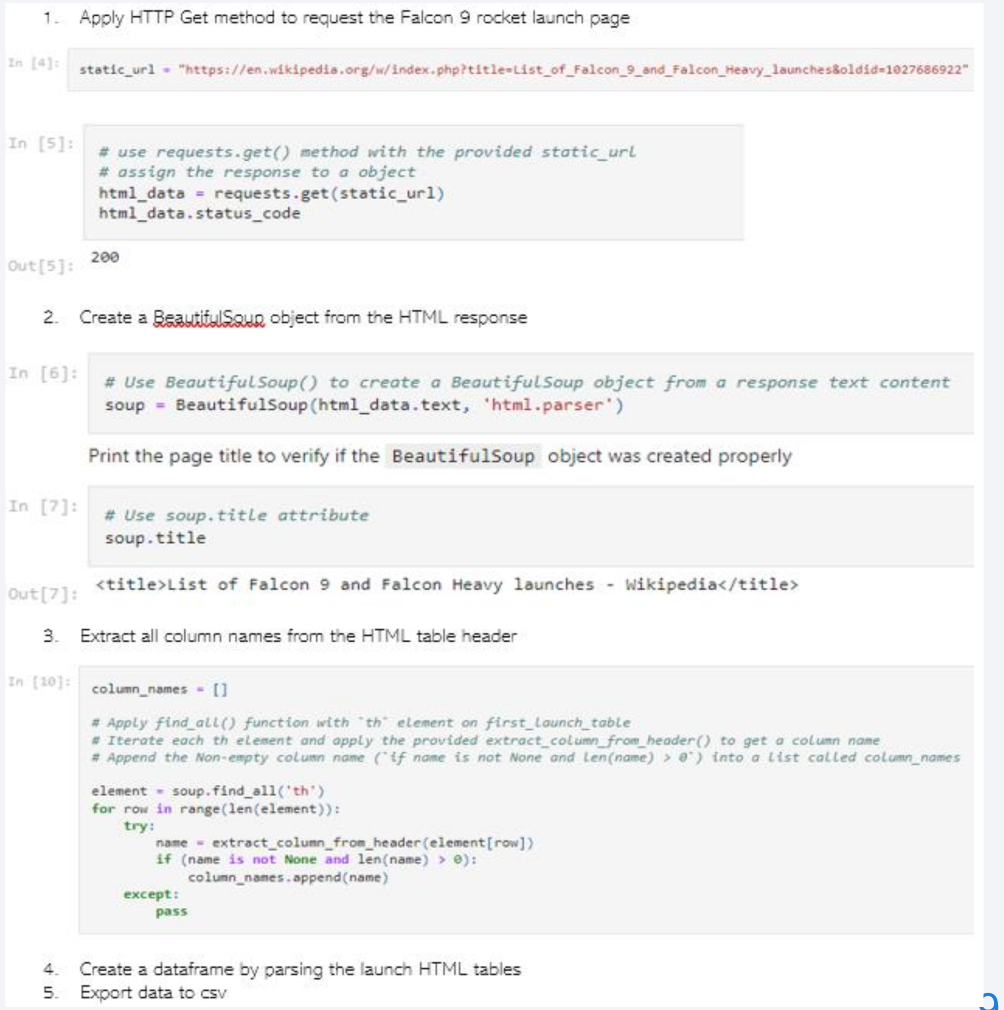
```
In [13]: # apply json_normalize
         data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

         df_rows = pd.DataFrame(rows)
         df_rows = df_rows.replace(np.nan, PayloadMass)

         data_falcon9['PayloadMass'][0] = df_rows.values
         data_falcon9
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe github



1. Apply HTTP Get method to request the Falcon 9 rocket launch page

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

Out[5]: 200
```

2. Create a BeautifulSoup object from the HTML response

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

Print the page title to verify if the `BeautifulSoup` object was created properly

```
In [7]: # Use soup.title attribute
        soup.title

Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

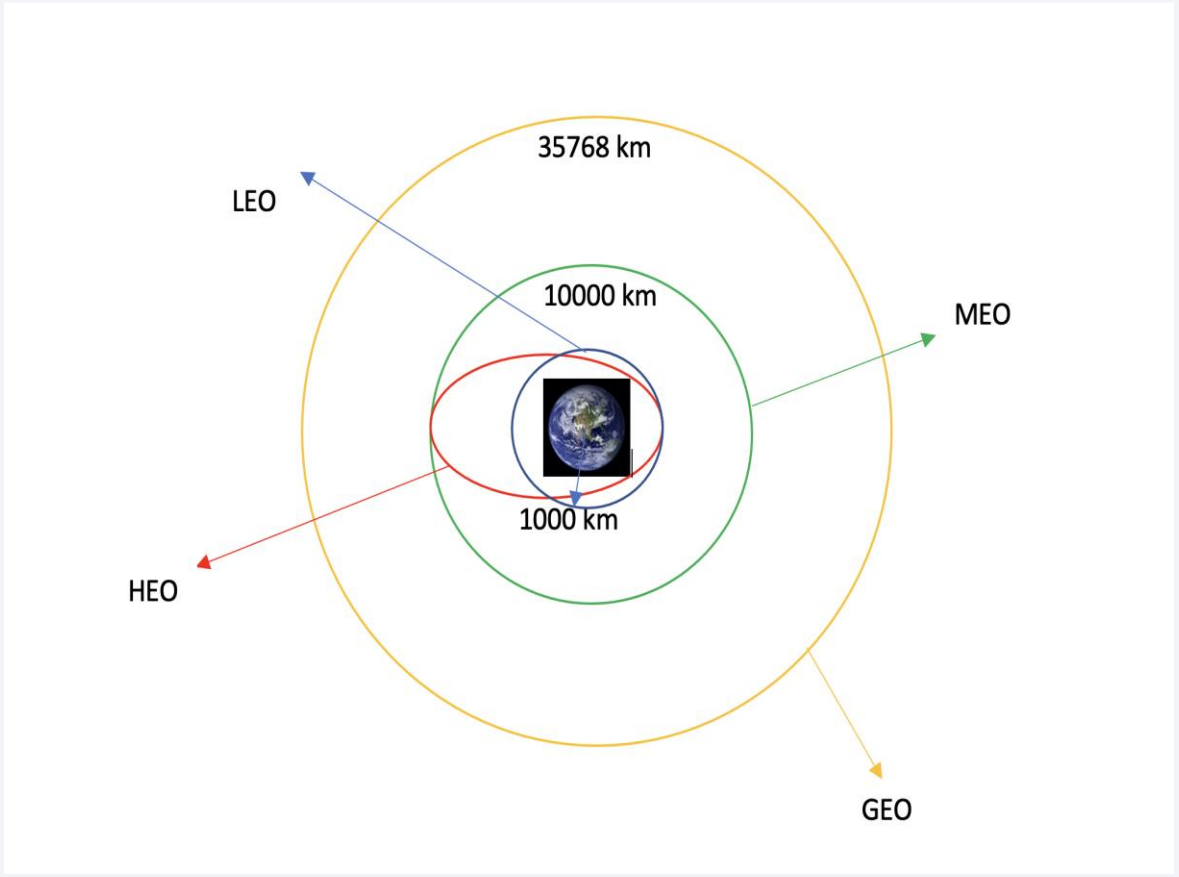
3. Extract all column names from the HTML table header

```
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

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

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

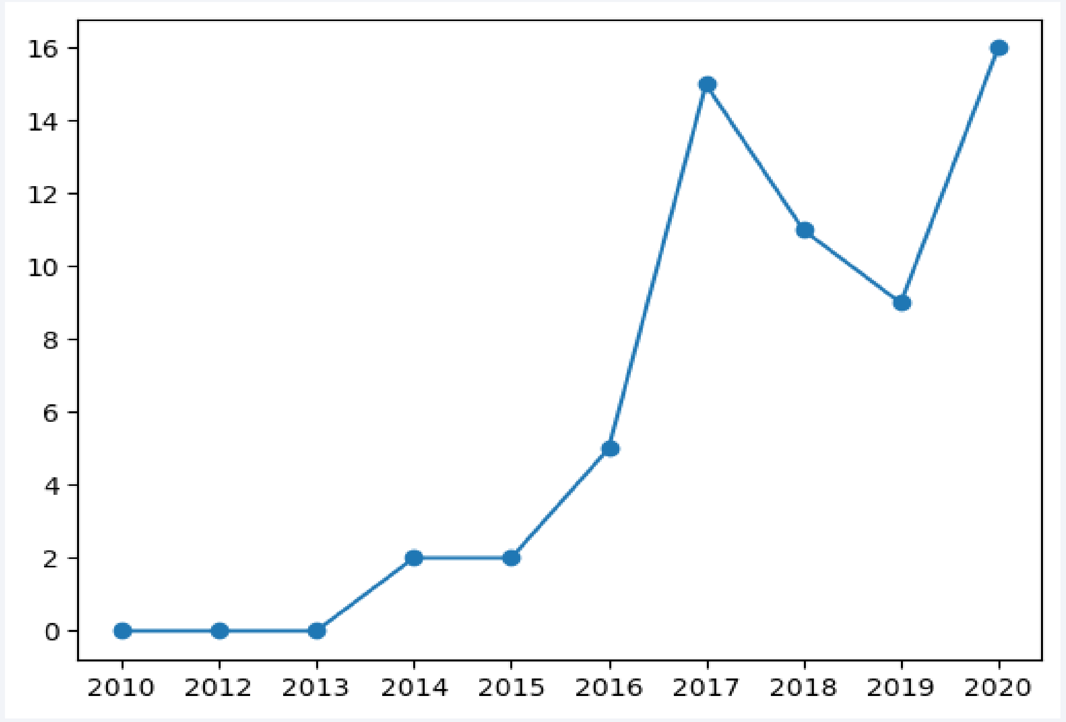
Data Wrangling

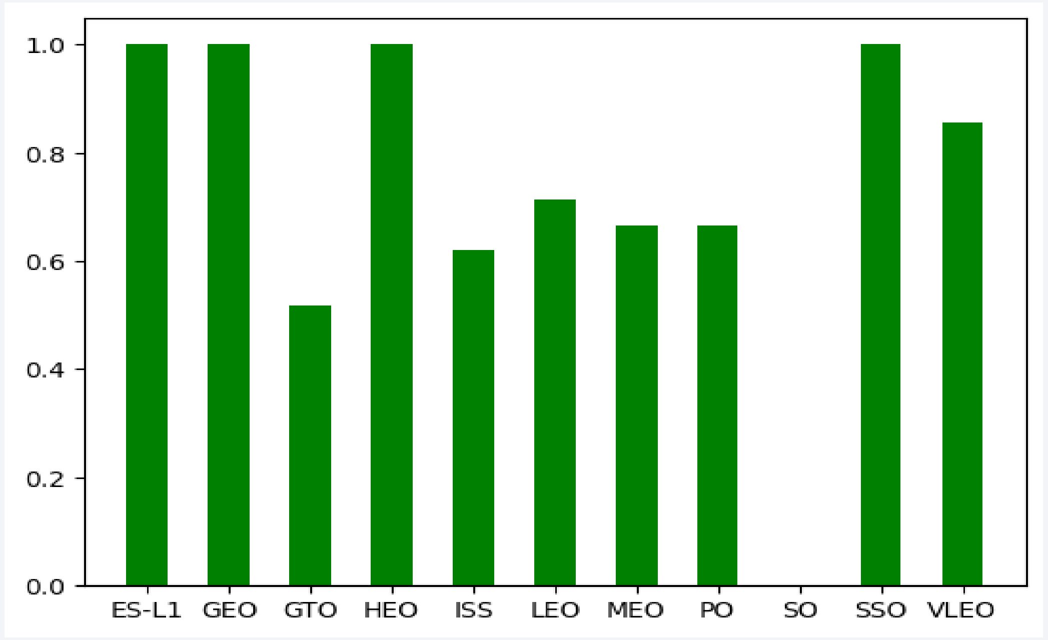
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is github



EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly





- Plot the launch success yearly trend
- The link to the notebook is [github]

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
    - The names of unique launch sites in the space mission.
    - The total payload mass carried by boosters launched by NASA (CRS)
    - The average payload mass carried by booster version F9 v1.1
    - The total number of successful and failure mission outcomes
    - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is github

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

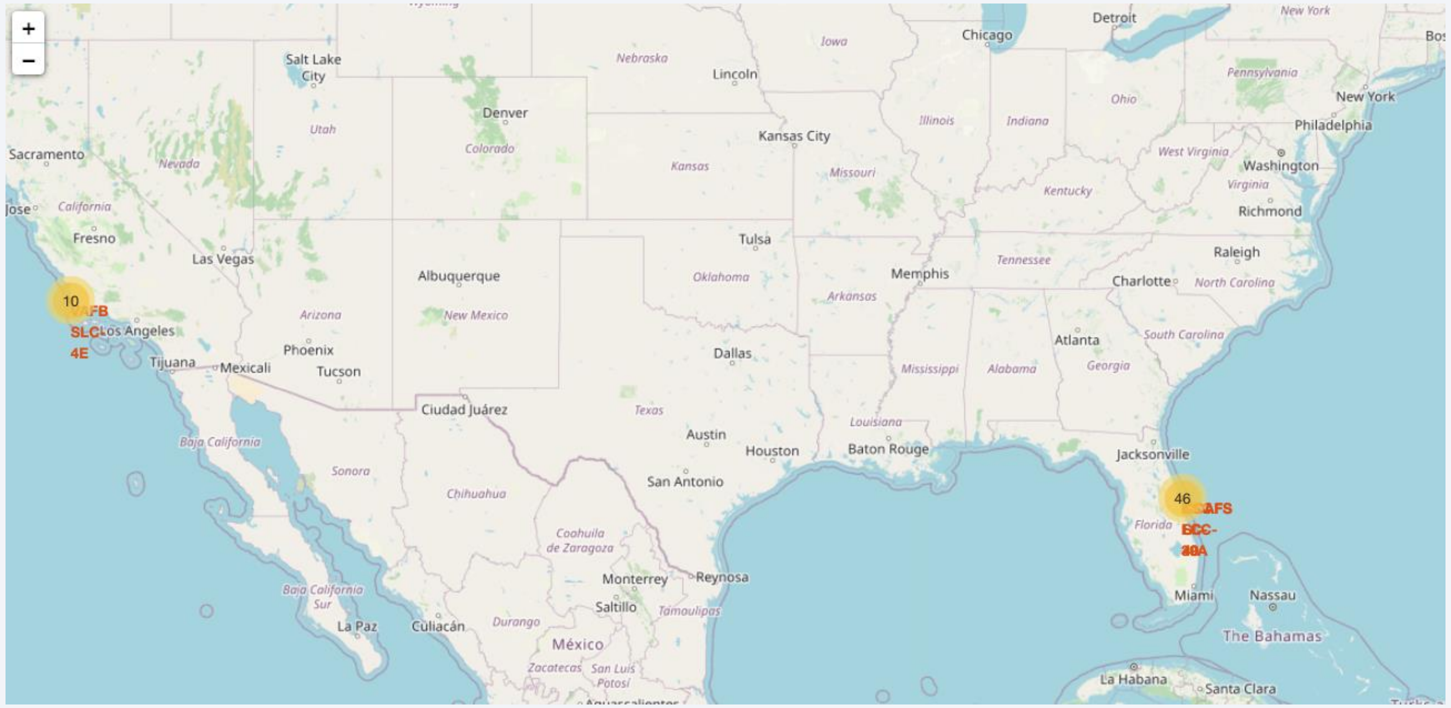
- SWe built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is github

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is github

Results

- Using interactive analytics was possible to identify that launch sites use to be in safety places, near sea, for example and have a good logistic infrastructure around.
- Most launches happens at east cost launch sites.

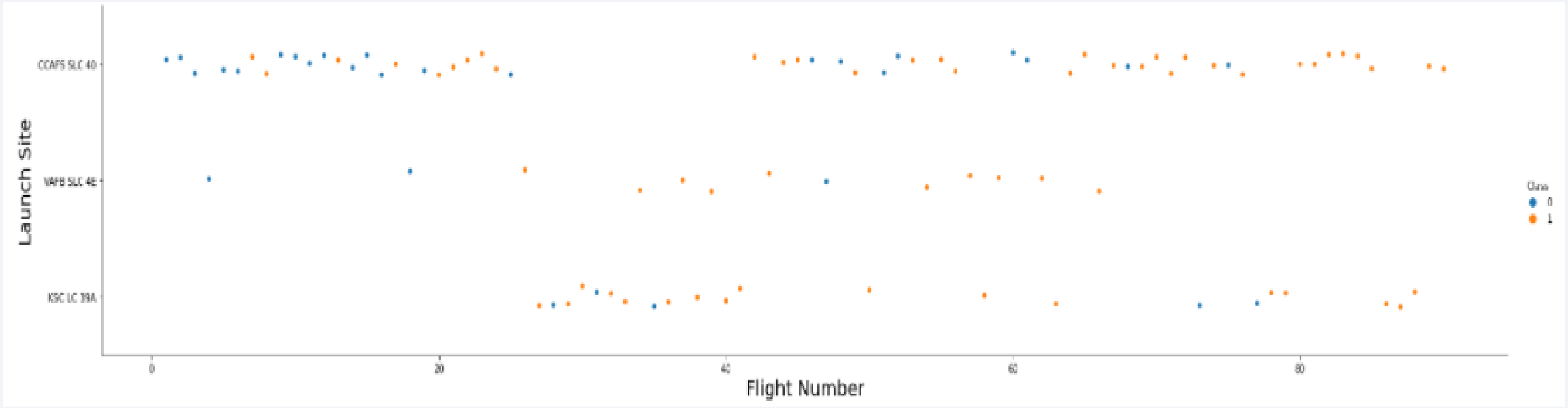


Section 2

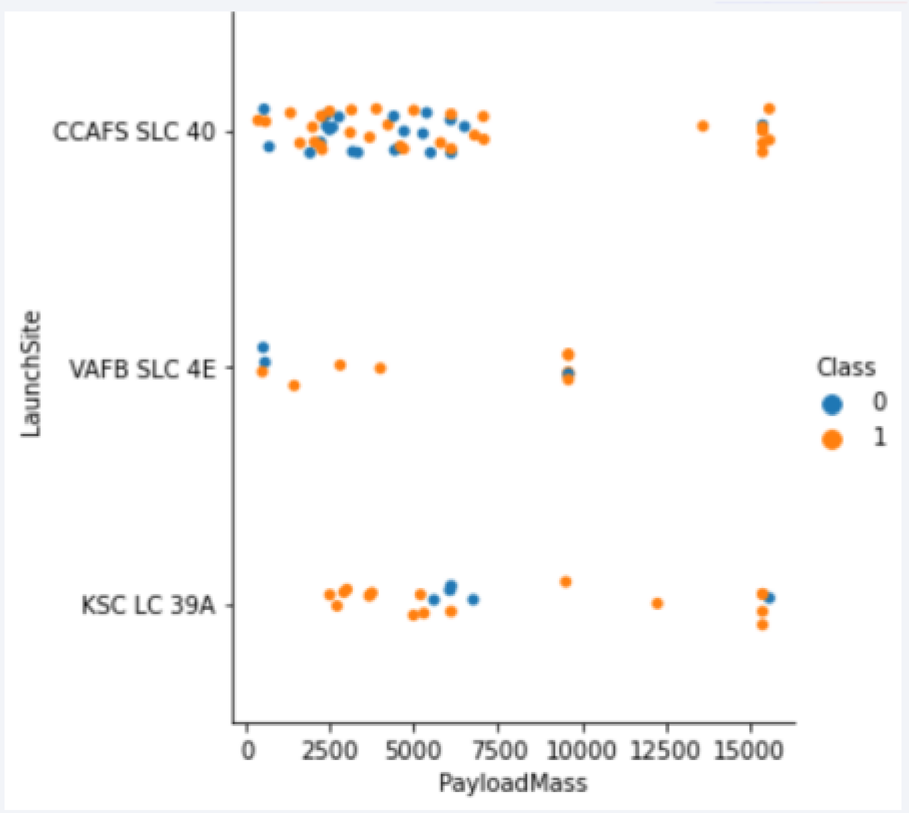
Insights drawn from EDA

Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

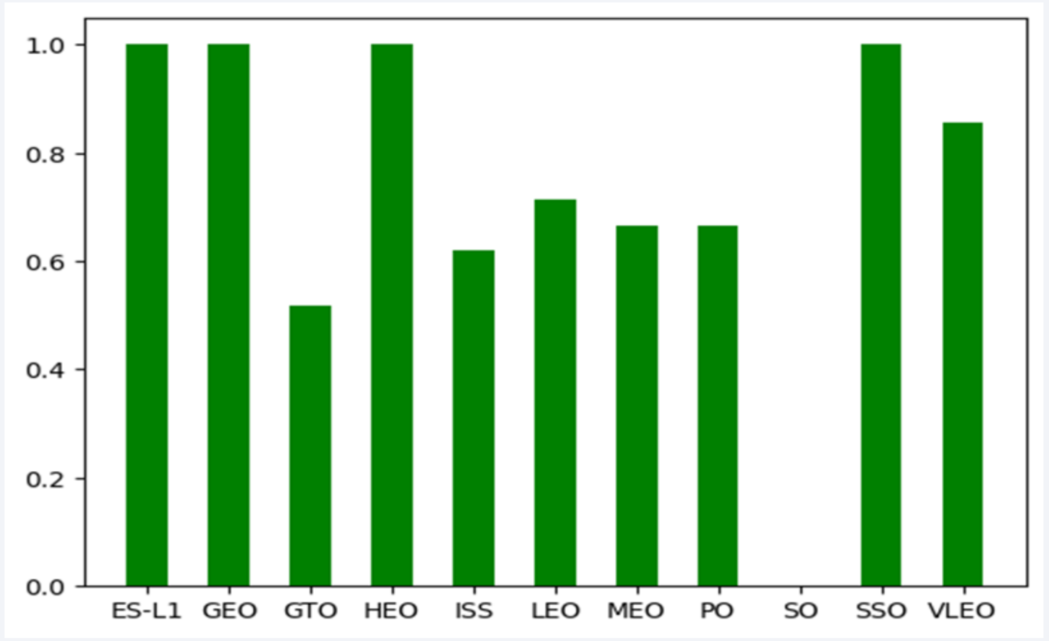


Payload vs. Launch Site



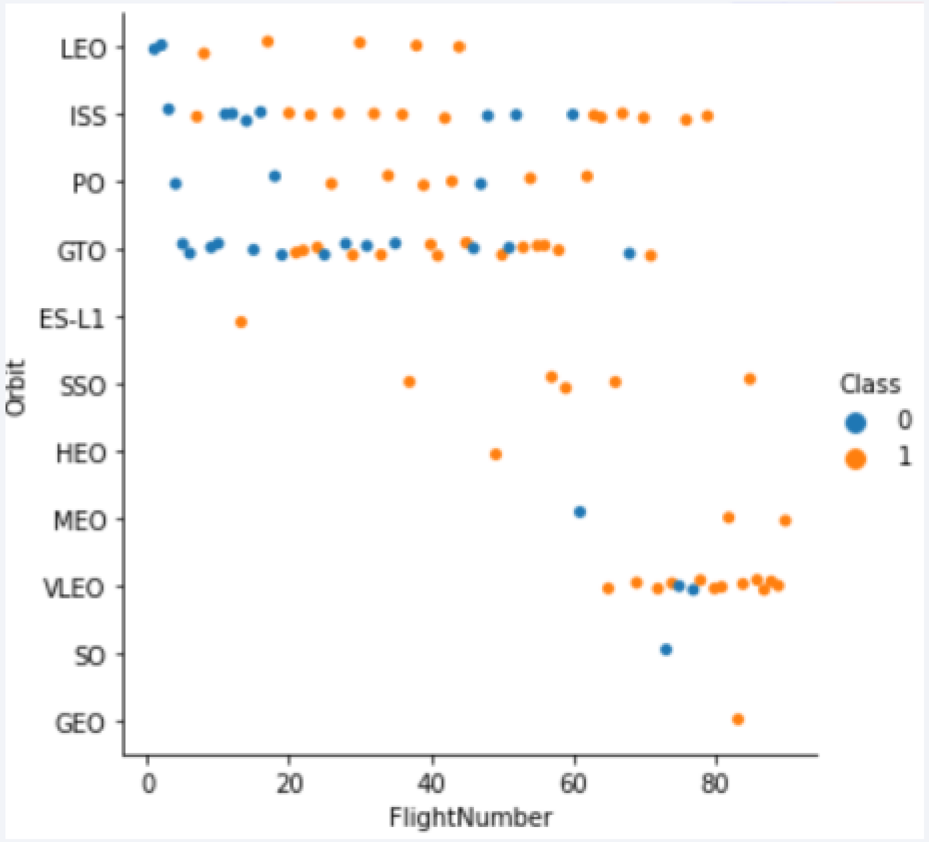
- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.

Success Rate vs. Orbit Type



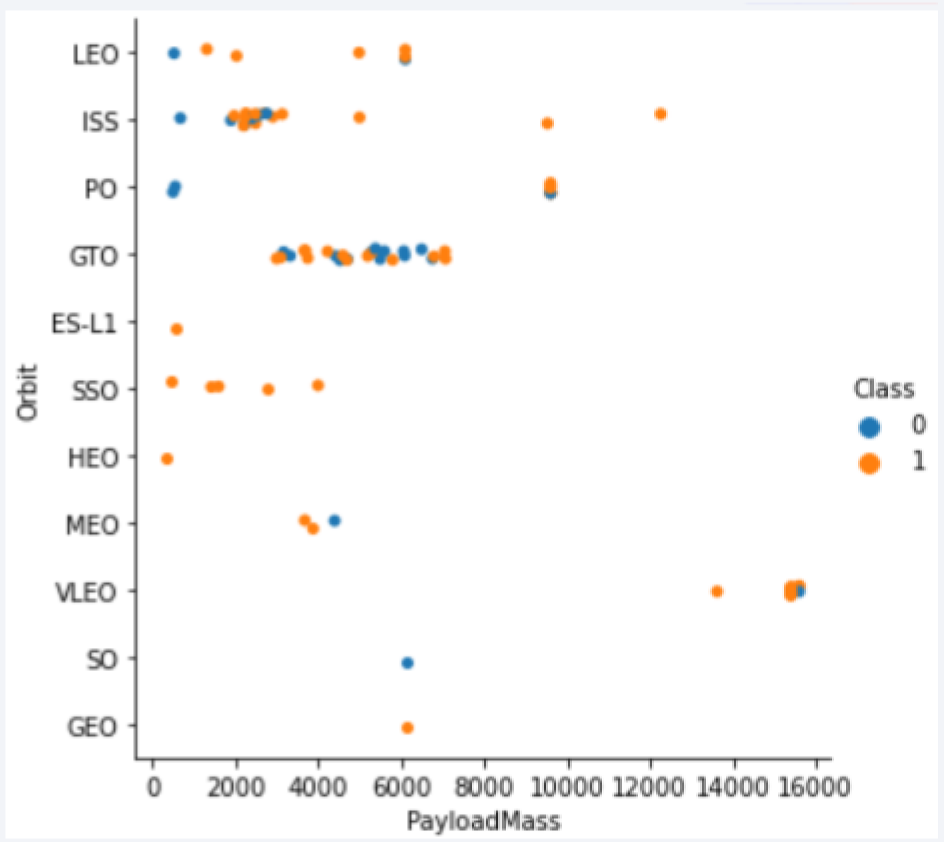
- The orbits ES-L1, GEO, HEO, and SSO have an average success rate of 100%; excluding SO landing sites with a 0% of success rate, generally speaking, all landing sites have an average success greater than 50%.

Flight Number vs. Orbit Type



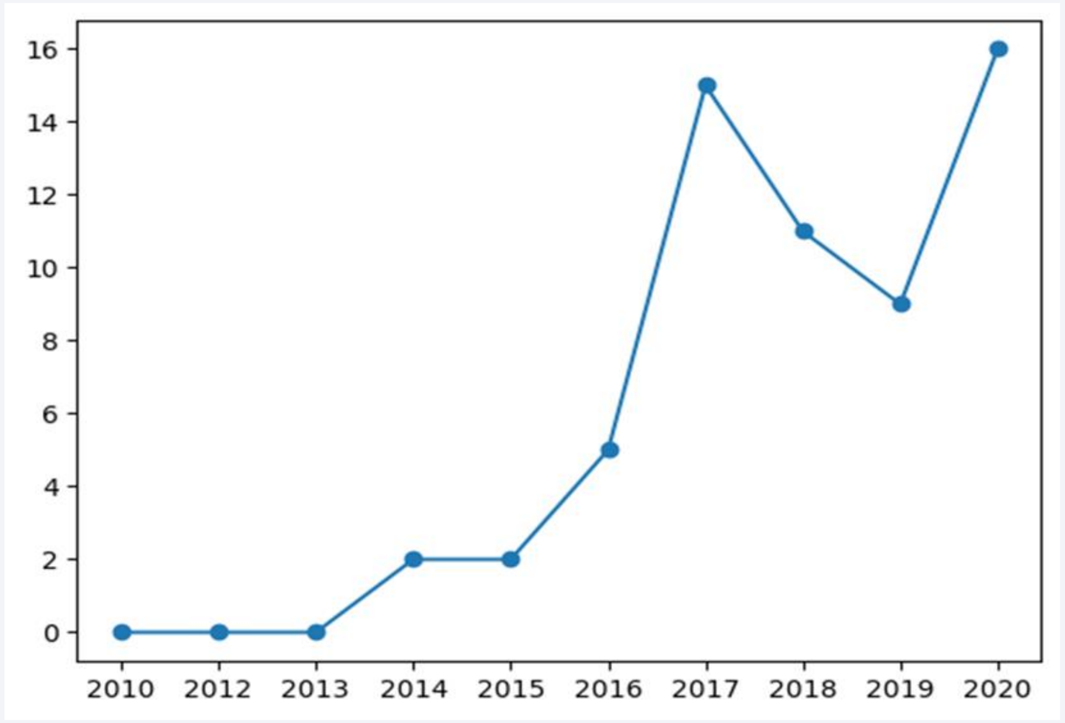
- The LEO orbit the Success appears related to the number of flights.
- There seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



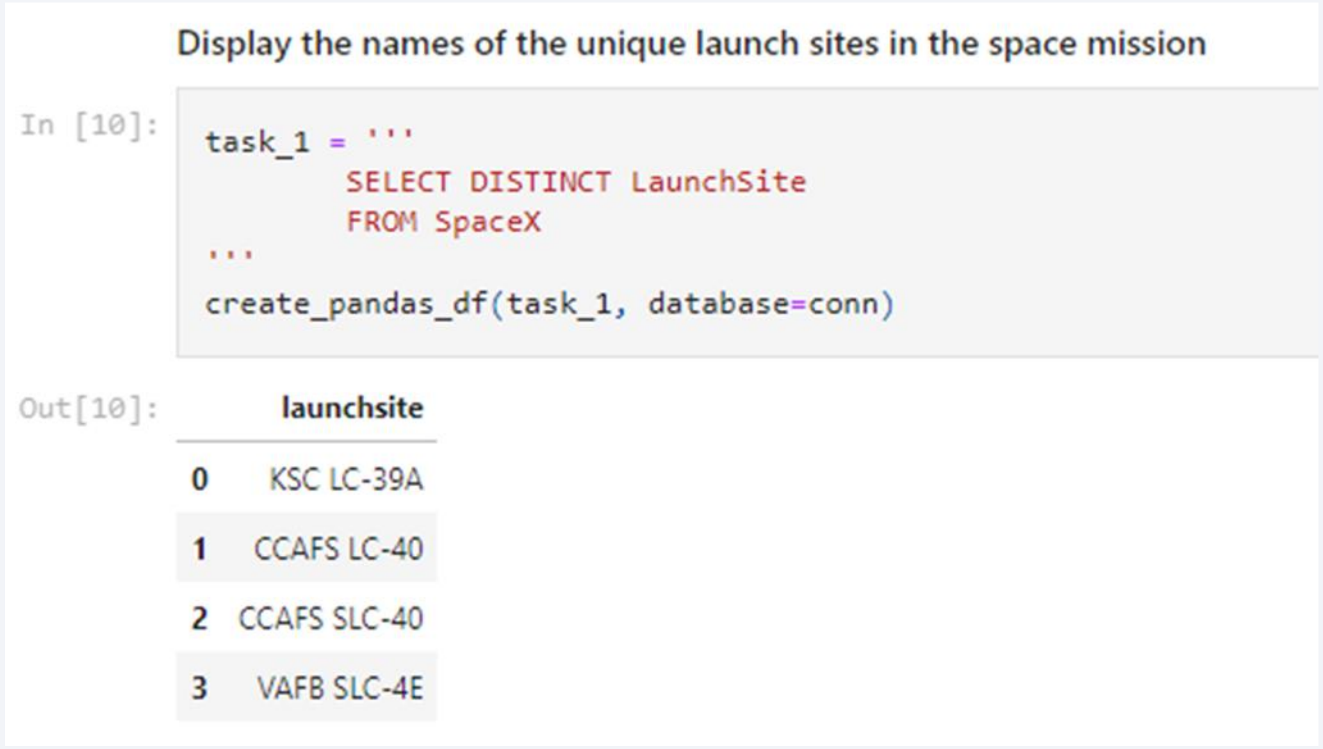
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO, and ISS.
- For GTO we cannot distinguish this well as both positive landing rate and negative landing (unsuccessful mission) are both there here.

Launch Success Yearly Trend



- The success rate since 2013 kept increasing till 2020.

All Launch Site Names



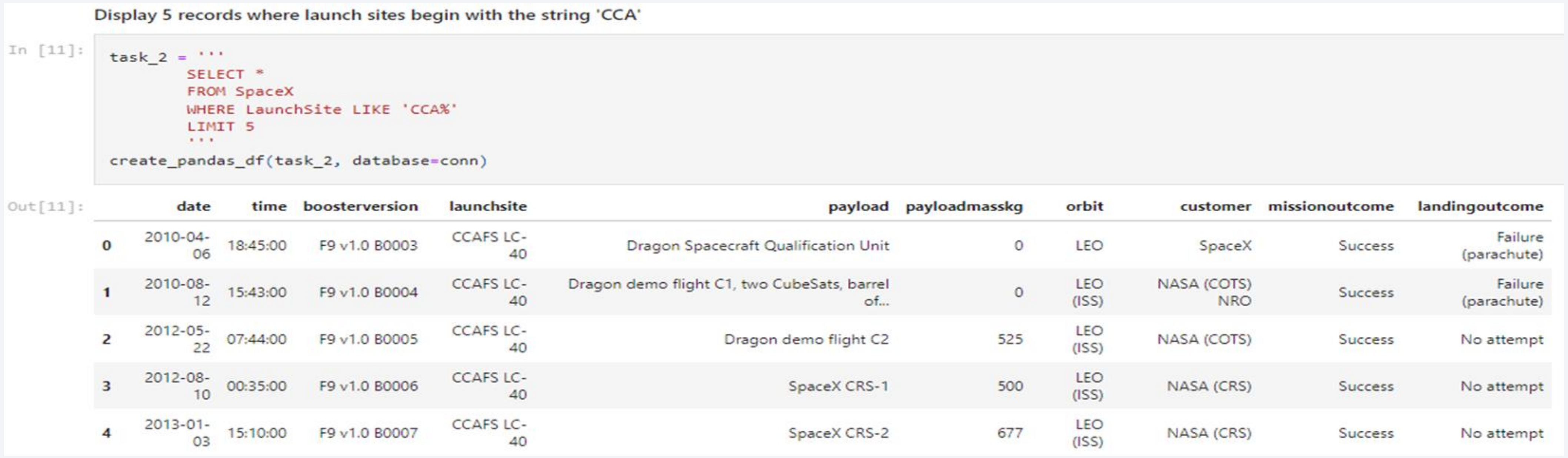
Display the names of the unique launch sites in the space mission

```
In [10]:  task_1 = '''
                  SELECT DISTINCT LaunchSite
                  FROM SpaceX
          '''
          create_pandas_df(task_1, database=conn)
```

```
Out[10]:
       launchsite
  0    KSC LC-39A
  1    CCAFS LC-40
  2    CCAFS SLC-40
  3    VAFB SLC-4E
```

- The names of the unique launch sites are CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, VAFB SLC-4E.

Launch Site Names Begin with 'CCA'



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

```
In [11]: task_2 = '''
                 SELECT *
                 FROM SpaceX
                 WHERE LaunchSite LIKE 'CCA%'
                 LIMIT 5
                 '''
         create_pandas_df(task_2, database=conn)
```

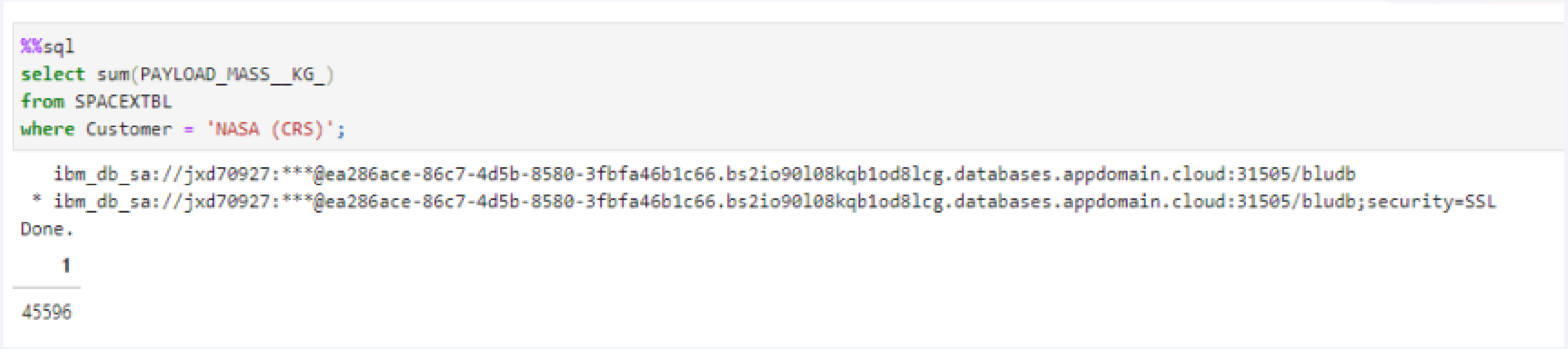
Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	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	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- The image shows 5 records where launch sites begin with ‘CCA’.

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below



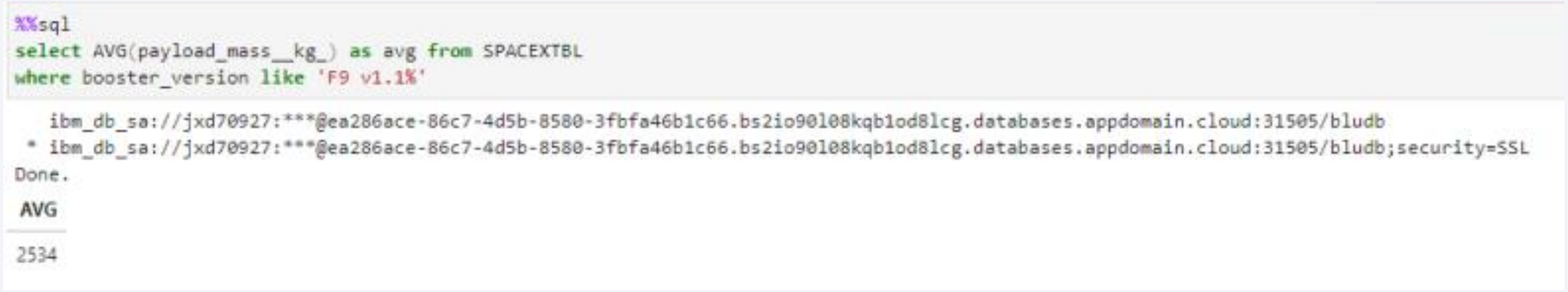
```
%%sql
select sum(PAYLOAD_MASS__KG_)
from SPACEXTBL
where Customer = 'NASA (CRS)';
```

```
   ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
 * ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb;security=SSL
Done.
```

1
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1



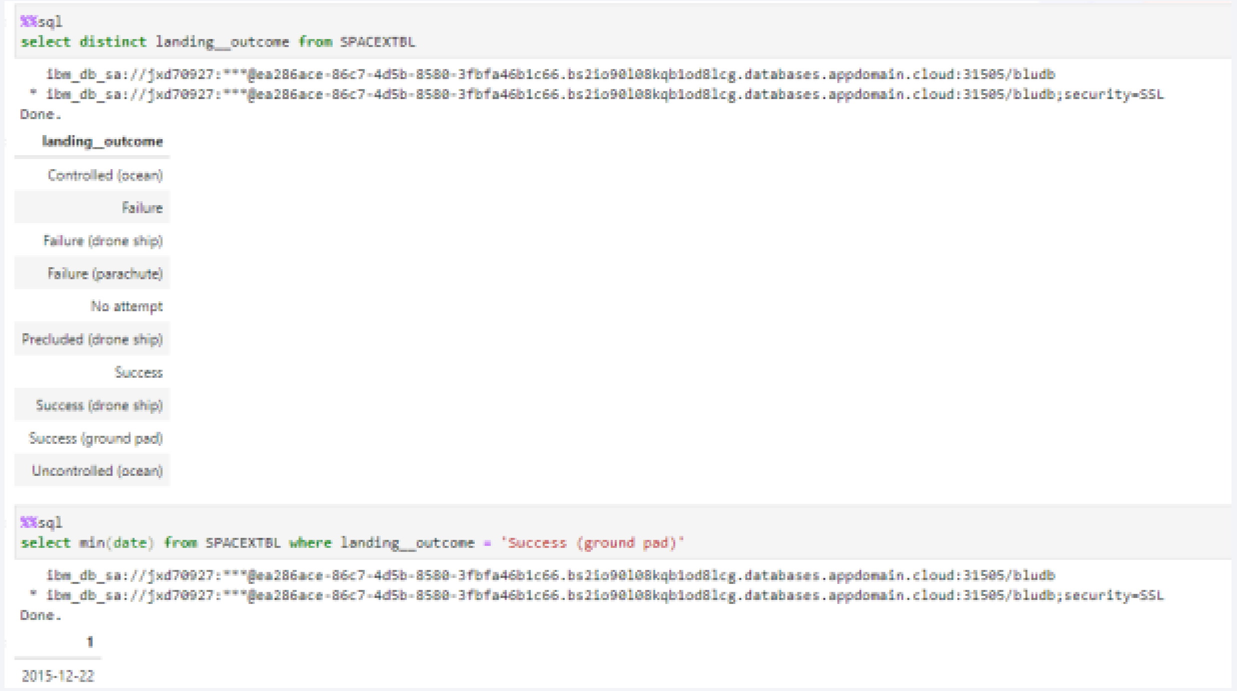
```
%%sql
select AVG(payload_mass__kg_) as avg from SPACEXTBL
where booster_version like 'F9 v1.1%'
```

```
   ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
 * ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb;security=SSL
Done.
```

AVG
2534

First Successful Ground Landing Date

- We used DISTINCT to find the right value representing successful ground landing and then used MIN-function found the dates of the first successful landing outcome on ground pad



```
%%sql
select distinct landing__outcome from SPACEXTBL
```

```
   ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
 * ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb;security=SSL
Done.
```

landing__outcome
Controlled (ocean)
Failure
Failure (drone ship)
Failure (parachute)
No attempt
Precluded (drone ship)
Success
Success (drone ship)
Success (ground pad)
Uncontrolled (ocean)

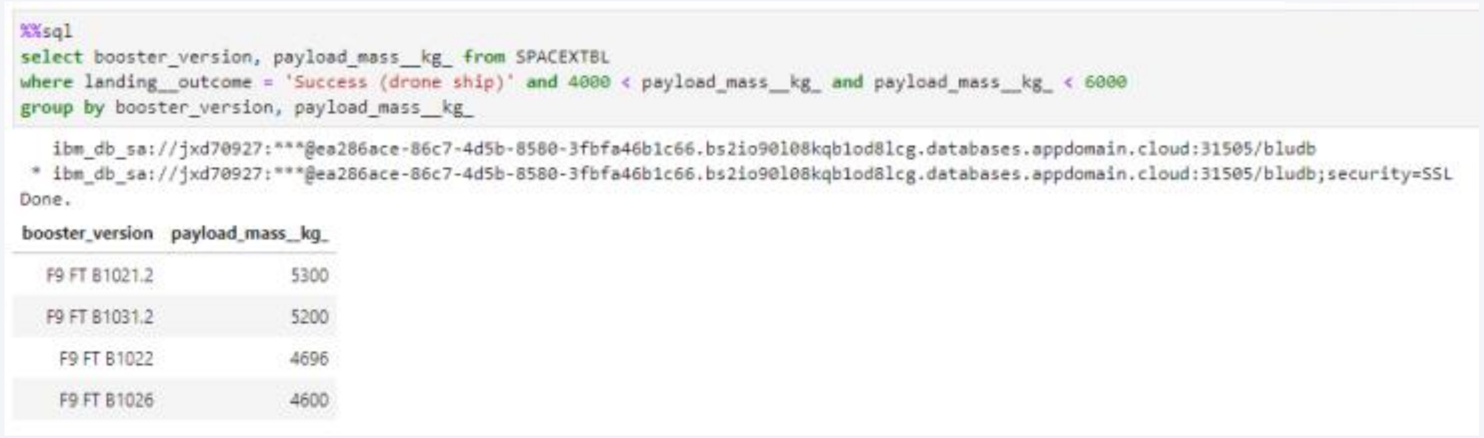
```
%%sql
select min(date) from SPACEXTBL where landing__outcome = 'Success (ground pad)'
```

```
   ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
 * ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb;security=SSL
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied AND condition to determine successful landing with payload mass greater than 4000 but less than 6000



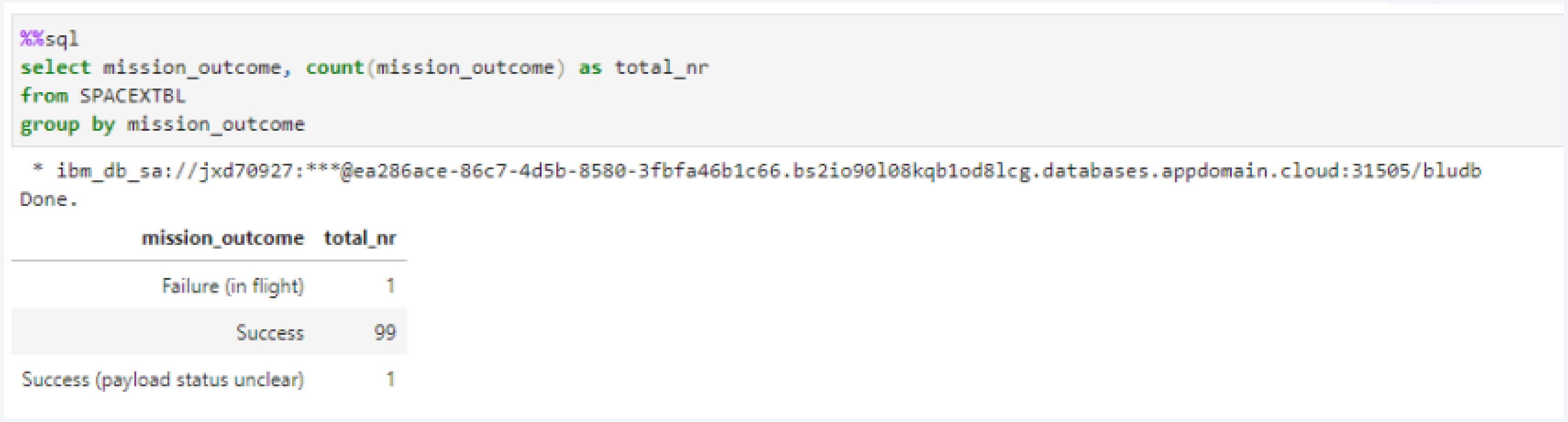
```
%%sql
select booster_version, payload_mass__kg_ from SPACEXTBL
where landing__outcome = 'Success (drone ship)' and 4000 < payload_mass__kg_ and payload_mass__kg_ < 6000
group by booster_version, payload_mass__kg_
```

```
   ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
 * ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb;security=SSL
Done.
```

booster_version	payload_mass__kg_
F9 FT B1021.2	5300
F9 FT B1031.2	5200
F9 FT B1022	4696
F9 FT B1026	4600

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like '%' to filter for WHER Mission Outcome was a success or a failure.



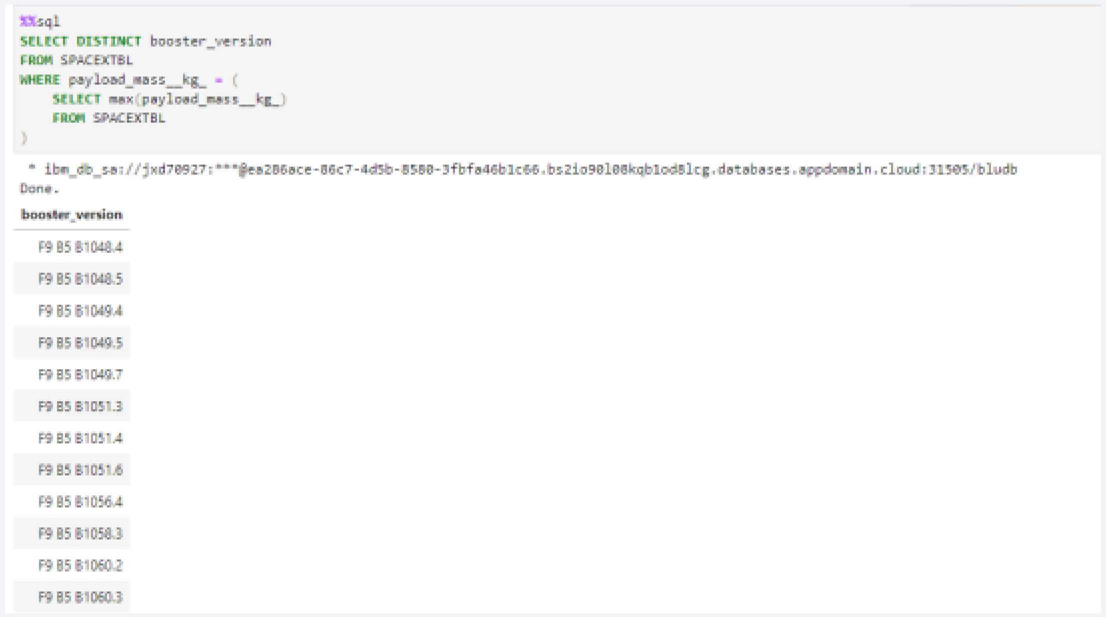
```
%%sql
select mission_outcome, count(mission_outcome) as total_nr
from SPACEXTBL
group by mission_outcome
```

```
 * ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

mission_outcome	total_nr
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.



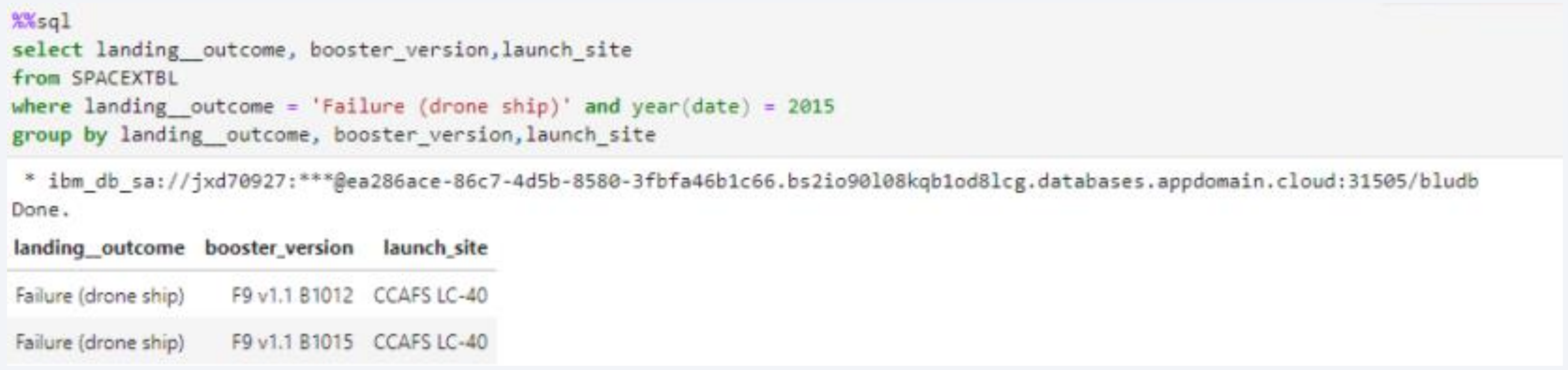
```
%%sql
SELECT DISTINCT booster_version
FROM SPACEXTBL
WHERE payload_mass__kg_ = (
    SELECT max(payload_mass__kg_)
    FROM SPACEXTBL
)
```

```
 * ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

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

2015 Launch Records

- We used a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015



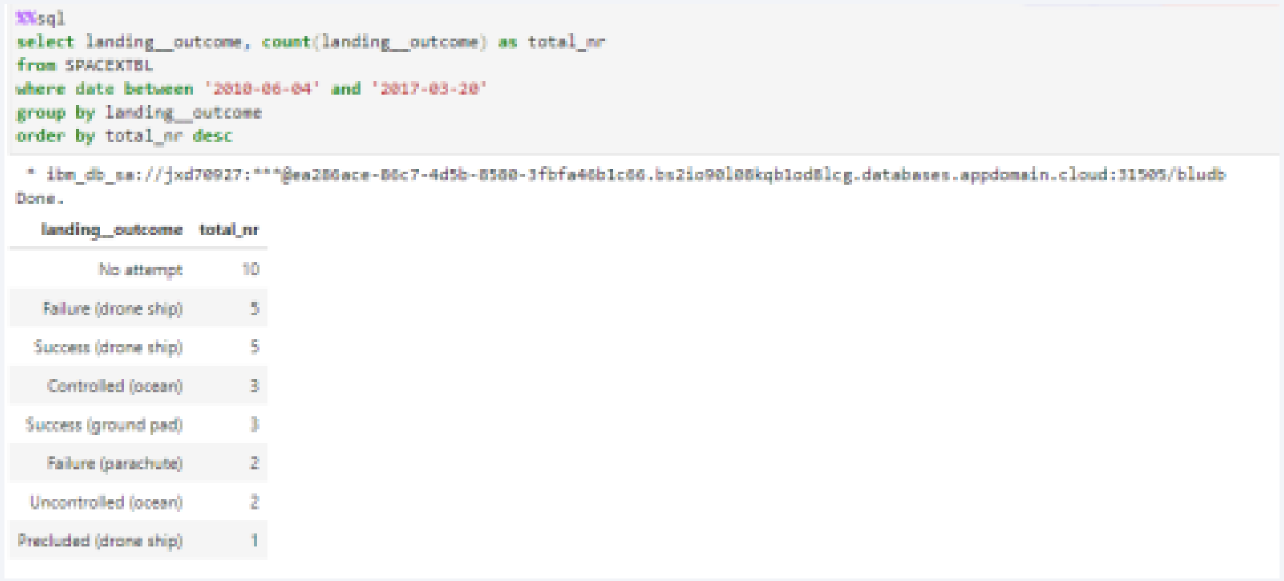
```
%%sql
select landing__outcome, booster_version,launch_site
from SPACEXTBL
where landing__outcome = 'Failure (drone ship)' and year(date) = 2015
group by landing__outcome, booster_version,launch_site
```

```
 * ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- We used GROUP BY and ORDER BY to rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between 2016-06-04 to 2010-03-20 in a descending order.



```
%%sql
select landing__outcome, count(landing__outcome) as total_nr
from SPACEXTBL
where date between '2010-06-04' and '2017-03-20'
group by landing__outcome
order by total_nr desc
```

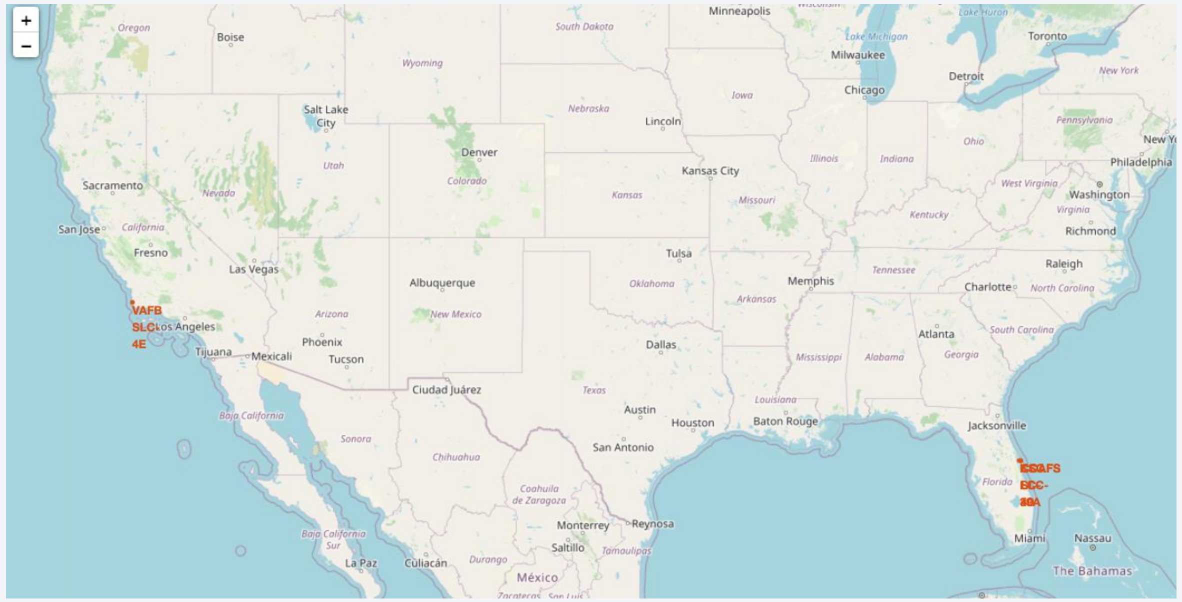
```
 * ibm_db_sa://jxd70927:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

landing__outcome	total_nr
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

Section 3

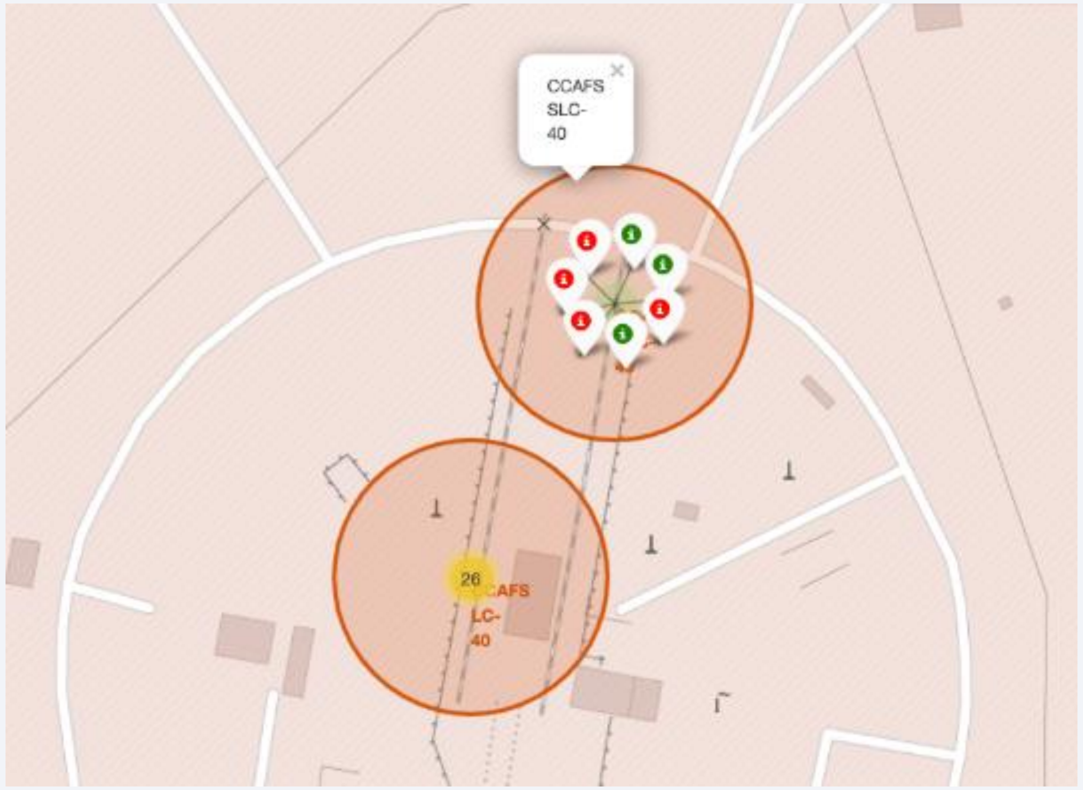
Launch Sites Proximities Analysis

Launch sites in Folium Map



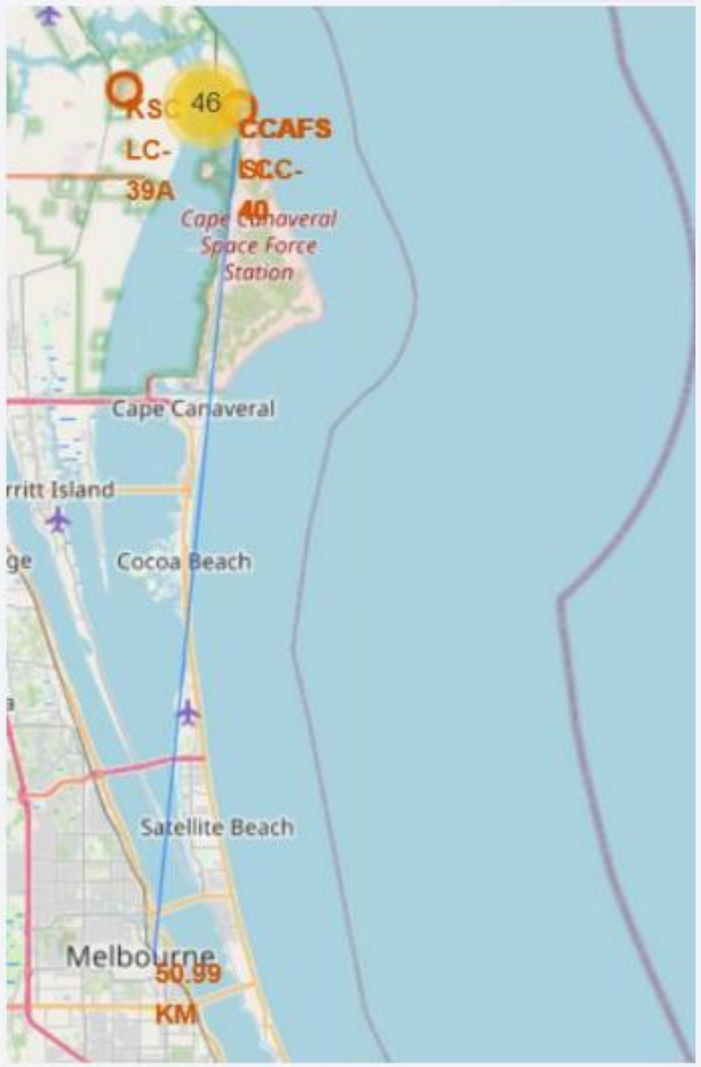
- It’s possible to notice two cluster launch sites in Florida and California.

Outcome feature per launch site with Folium Map



- Folium allows us to visually analyze the outcome feature per launch site.
- For instance, for the launch site KSC LC 40 in Florida, there are 10 successful landings and 3 failed landings.

Distance between CCAFS SLC-40 launch site and its proximities

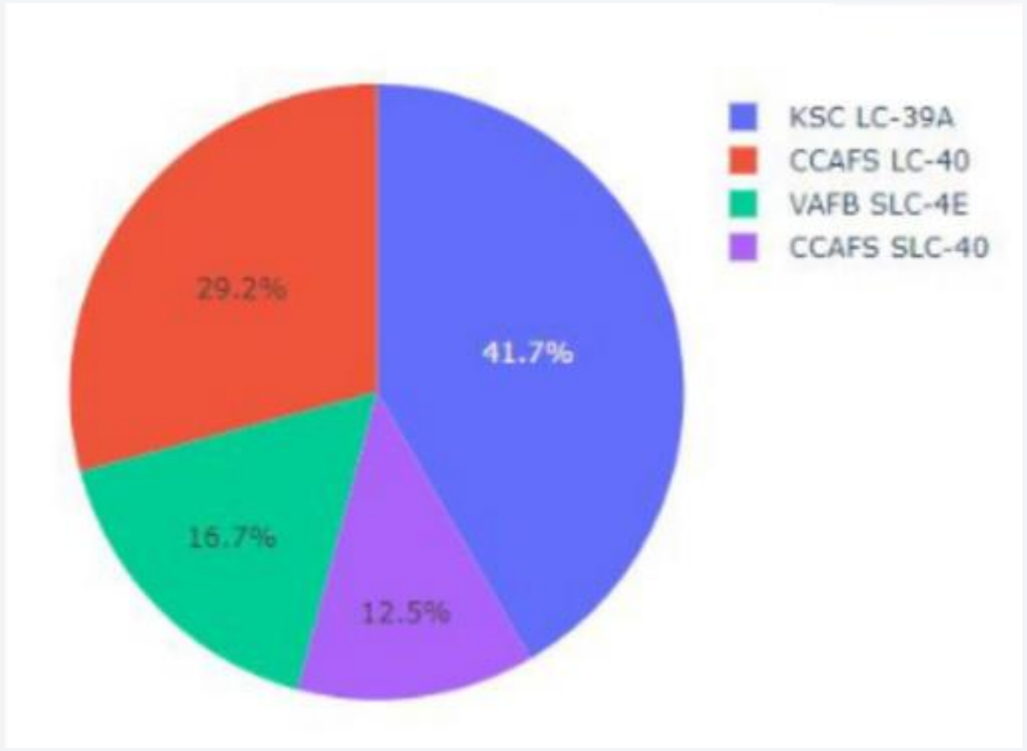


- The selected area for the distance analysis between the CCAFS SLC-40 launch site was Melbourne city.
- The distance between CCAFS SLC-40 and Melbourne city is 50.99 km.

Section 4

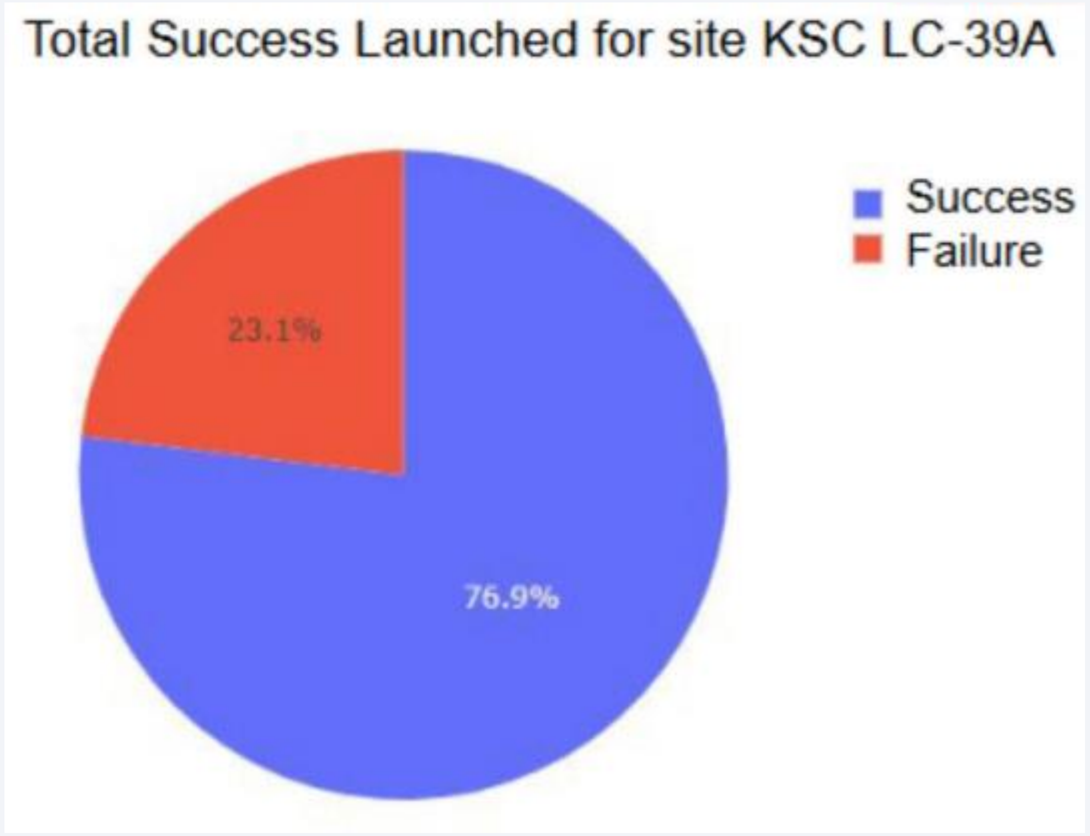
Build a Dashboard with Plotly Dash

Total Success Launches by site



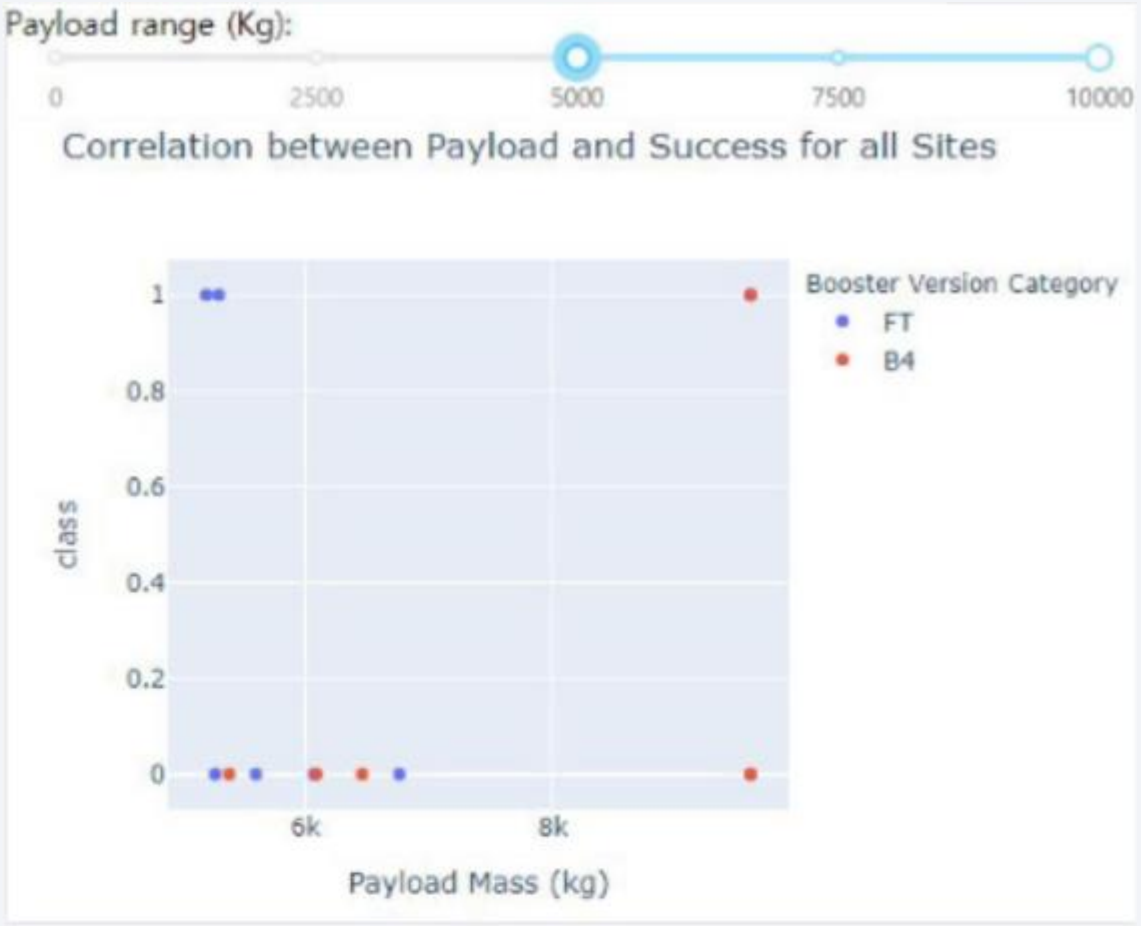
- Through the pie chart, it’s possible to determine that the highest launch success rate is KSC LC-39 A with 41.7%, followed by CCAFS LC-40 with 29.2% and at the bottom the VAFB SLC4E, and CCAFS SLC-40 with 16.7% and 12.5%, respectively.

KSC LC-39A total Success Launches



- Through the pie chart is possible to determine that KSC LC-39A has a 76.9% success rate.

Correlation between payload and success for all sites

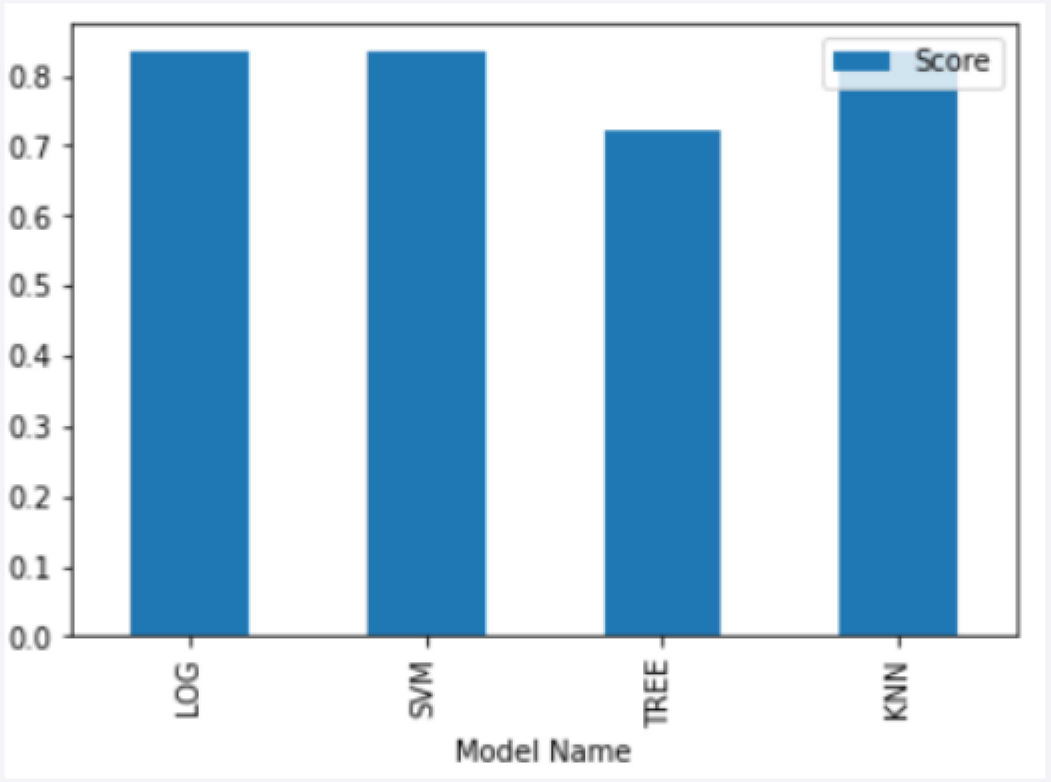


- The plot shows the correlation between payload and success for all sites with a [5000,10000] range.
- With heavier payload mass it appears to end in an unsuccessful landing.

Section 5

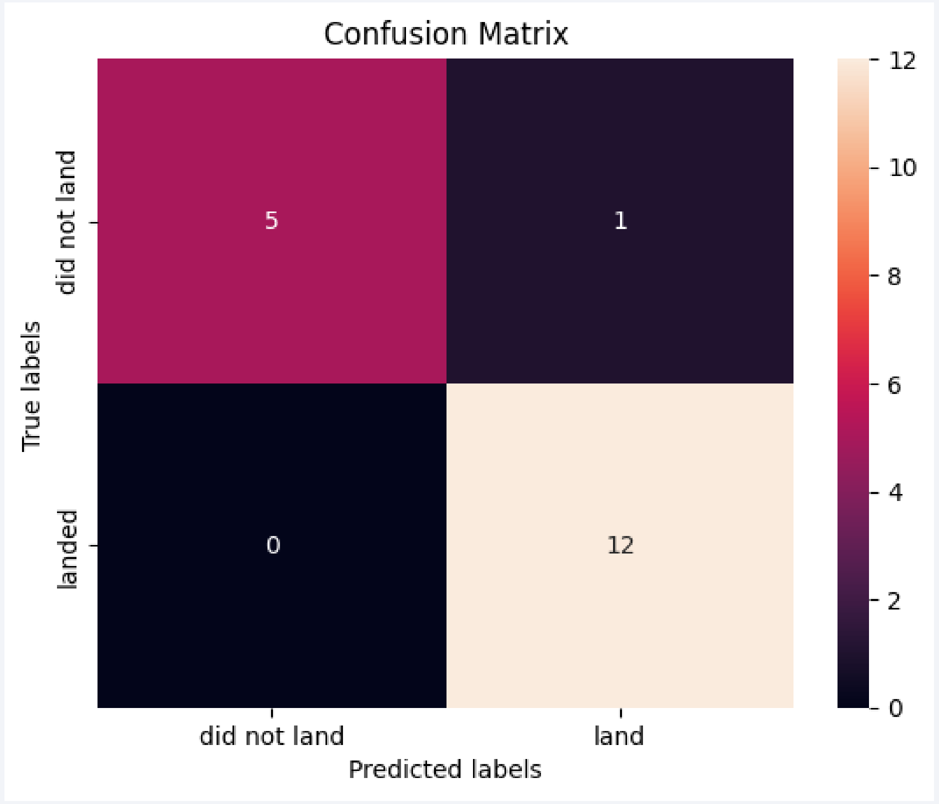
Predictive Analysis (Classification)

Classification Accuracy



The best performance models classifier were the logistic regression, SVM, and KNN, each with a score more of 80%.

Confusion Matrix



- Support vector machine with the best parameters gives an excellent result

Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Support vector machine is the best machine learning algorithm for this task.

Appendix

- • The repository for this project is available here

Thank you!