IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Kevin Rio Harristyando
21 November 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
  - ✓ Data Collection Through API
  - ✓ Data Collection with Web-Scraping
  - ✓ Data Wrangling
  - ✓ Exploratory Data Analysis (EDA) with SQL
  - ✓ Perform interactive visual analytics using Folium and Plotly Dash
  - ✓ Perform predictive analysis using classification models
- 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. The primary challenge addressed in this machine learning project is to predict whether the first stage of a Falcon 9 rocket will successfully land or not. By accurately predicting the landing outcome, potential clients or competing companies can make informed decisions when considering SpaceX as a launch provider or bidding against SpaceX for a rocket launch contract.

- Problems you want to find answers

  - ❑ Can we accurately predict whether the first stage of a Falcon 9 rocket will successfully land or not?
  - ❑ How can the predictions from the machine learning model be utilized in the decision-making process for potential clients or competing companies considering SpaceX for a rocket launch?
  - ❑ What insights can be gained from the exploratory data analysis regarding the distribution of key features and potential patterns related to landing outcomes?
  - ❑ How accurate, precise, and reliable is the model in predicting first stage landing outcomes?

Section 1

Methodology

Methodology

Executive Summary

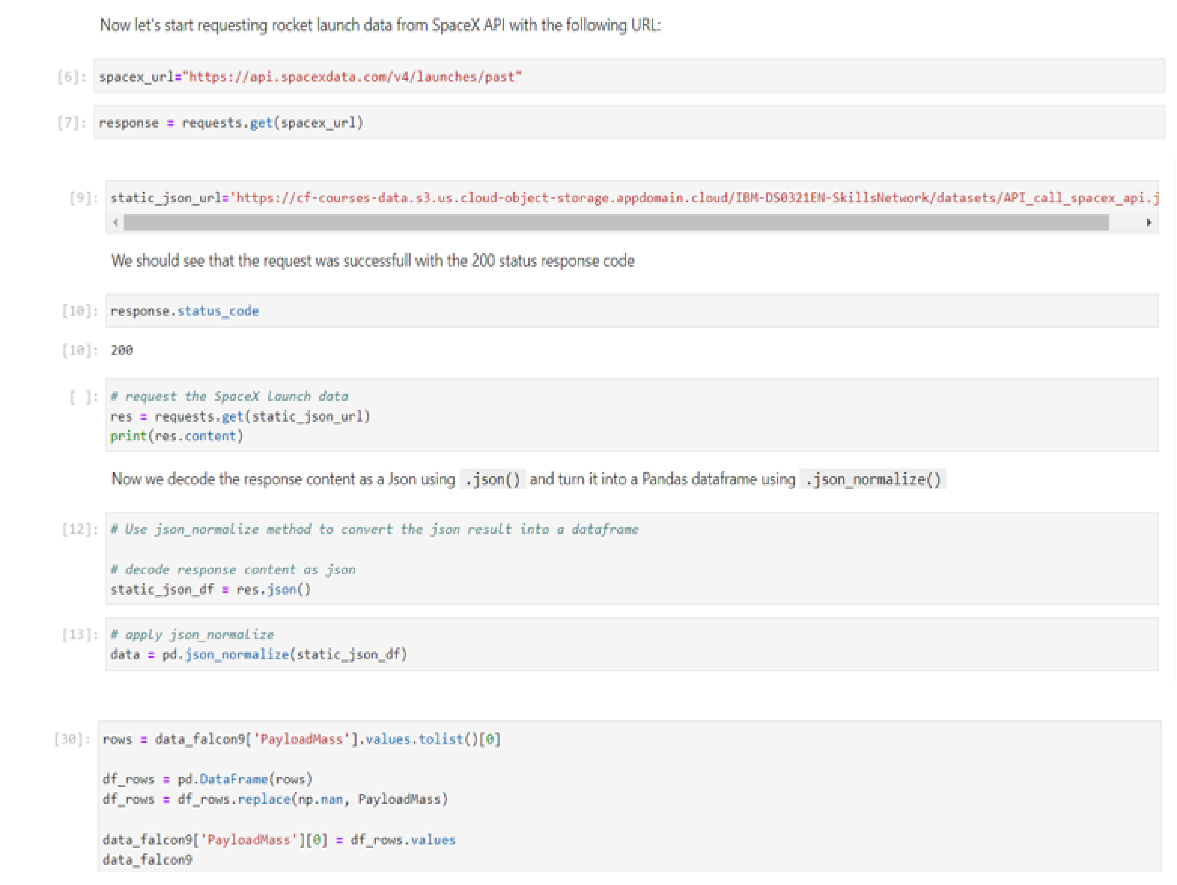
- Data collection methodology:
  - Data was collected using SpaceX API and Web-Scraping from Wikipedia
- Perform data wrangling
  - One-hot encoding was applied to categorical features and Data Standardization
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive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

Data Collection

- Data Collection was initiated the data collection process by utilizing a GET request to the SpaceX API.
- Following this, we decoded the response content using the .json() function and transformed it into a pandas dataframe using the .json_normalize() method.
- Subsequently, we conducted data cleaning, examined for any missing values, and addressed them as needed.
- Additionally, we engaged in web scraping from Wikipedia to retrieve Falcon 9 launch records using BeautifulSoup, in order to extract the launch records presented in an HTML table, parse the table, and convert the information into a pandas dataframe for subsequent 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.
- Link Notebook: https://github.com/kevinrioharris/IBM-Courses-Data-Science/blob/main/Applied%20Data%20Science%20Capstone/Data%20Collection%20API_FinalPresentation.ipynb



Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

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

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

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

```
[10]: response.status_code
```

```
[10]: 200
```

```
[ ]: # request the SpaceX launch data
     res = requests.get(static_json_url)
     print(res.content)
```

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

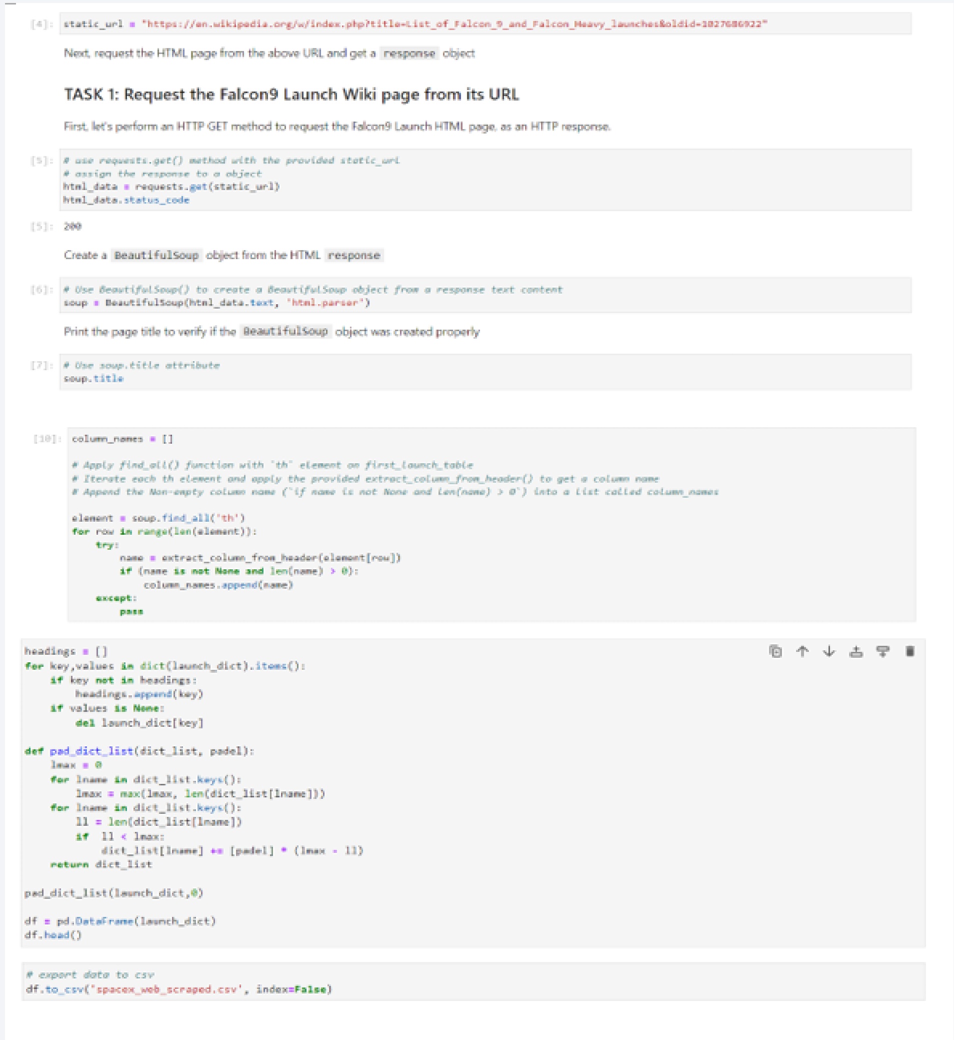
```
[12]: # Use json_normalize method to convert the json result into a dataframe

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

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

```
[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.
- Link Notebook: https://github.com/kevinrioharris/IBM-Courses-Data-Science/blob/main/Applied%20Data%20Science%20Capstone/Data%20Collection%20with%20Web%20Scraping_FinalPresentation.ipynb



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

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

[5]:
```python
# 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
```

[5]:
```
200
```

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

[6]:
```python
# 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

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

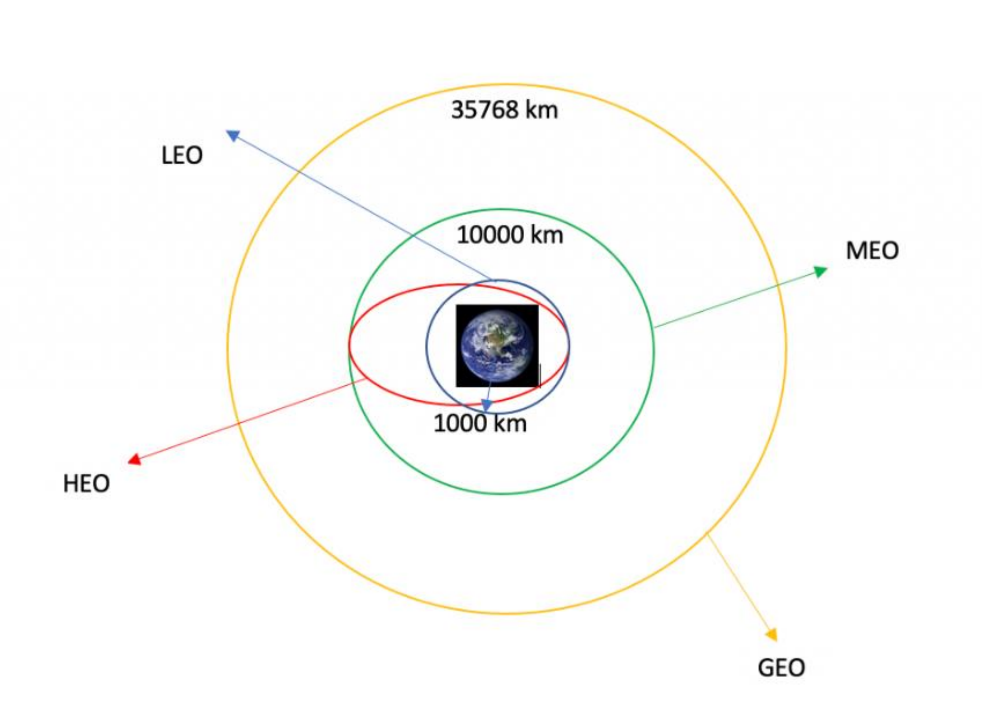
[10]:
```python
column_names = []

# 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
```

```python
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

df = pd.DataFrame(launch_dict)
df.head()
```

```python
# export data to csv
df.to_csv('spacex_web_scraped.csv', index=False)
```

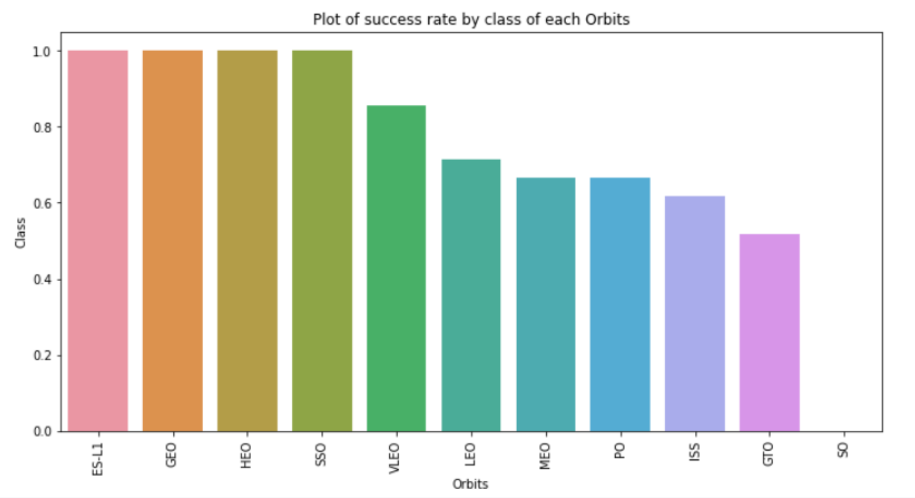
Data Wrangling

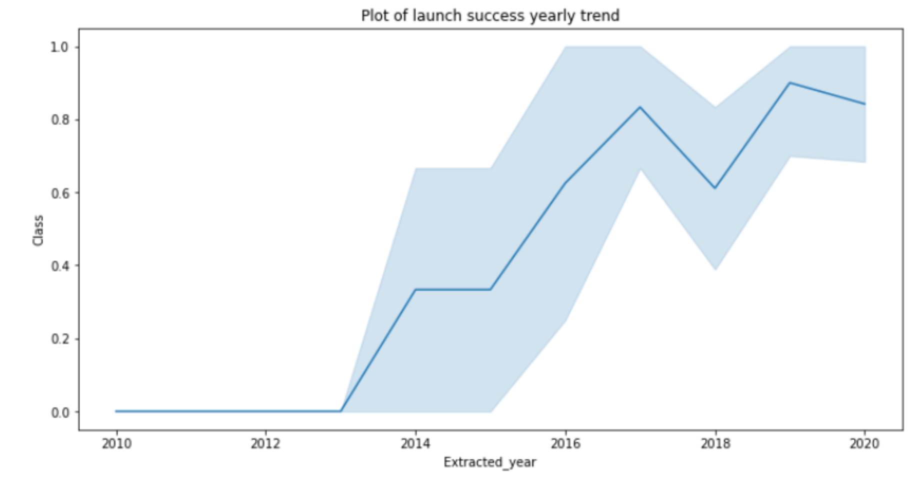
- We conducted exploratory data analysis to establish training labels, computed launch counts per site, and analyzed orbit frequencies.
- Additionally, we derived landing outcome labels from the outcome column and exported the findings to a CSV file.
- Link notebook: https://github.com/kevinrioharris/IBM-Courses-Data-Science/blob/main/Applied%20Data%20Science%20Capstone/Data%20Wrangling_FinalPresentation.ipynb



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 trend.
- And As we can see that the sucess rate since 2013 kept increasing till 2020
- Link Notebook: https://github.com/kevinrioharris/IBM-Courses-Data-Science/blob/main/Applied%20Data%20Science%20Capstone/EDA%20with%20Data%20Visualization_FinalPresentation.ipynb





EDA with SQL

- We seamlessly imported the SpaceX dataset into a PostgreSQL database within the Jupyter notebook.
- Employing EDA with SQL for exploratory data analysis, we extracted key insights, including:
  - 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.
- Link Notebook: https://github.com/kevinrioharris/IBM-Courses-Data-Science/blob/main/Applied%20Data%20Science%20Capstone/EDA%20with%20SQL_FinalPresentation.ipynb

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.
- Link notebook: https://github.com/kevinrioharris/IBM-Courses-Data-Science/blob/main/Applied%20Data%20Science%20Capstone/Interactive%20Map%20with%20Folium_FinalP.ipynb

Build a Dashboard with Plotly Dash

- We 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.
- Link Code: https://github.com/kevinrioharris/IBM-Courses-Data-Science/blob/main/Applied%20Data%20Science%20Capstone/PlotlyApps.py

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.
- Link Notebook: https://github.com/kevinrioharris/IBM-Courses-Data-Science/blob/main/Applied%20Data%20Science%20Capstone/Machine%20Learning%20Prediction_FinalP.ipynb

Results

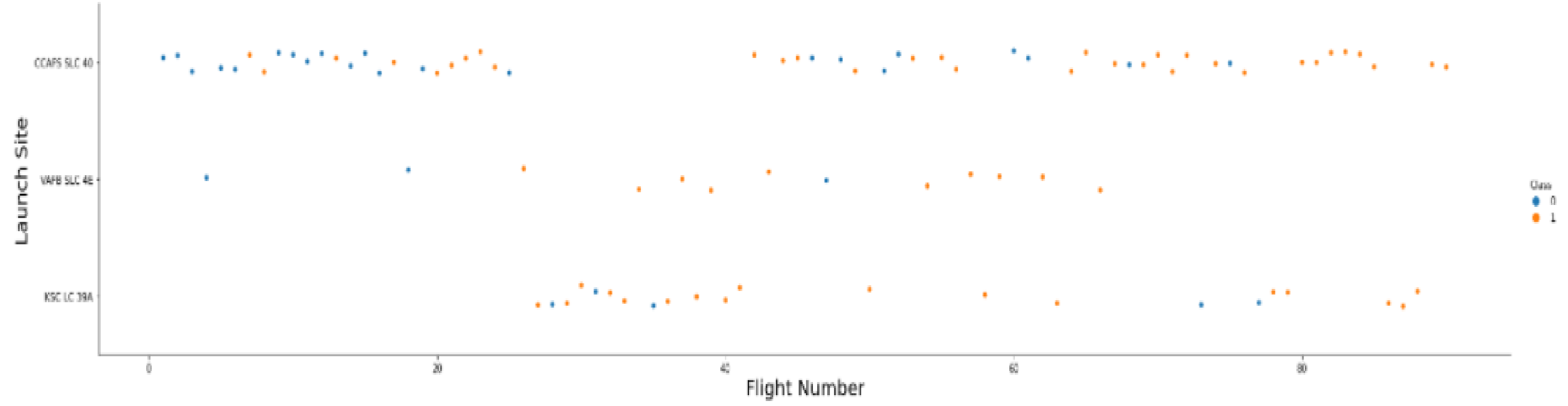
- Best classification model is **DecisionTree** with a accuracy score of 0.8732142857142856

Section 2

Insights drawn from EDA

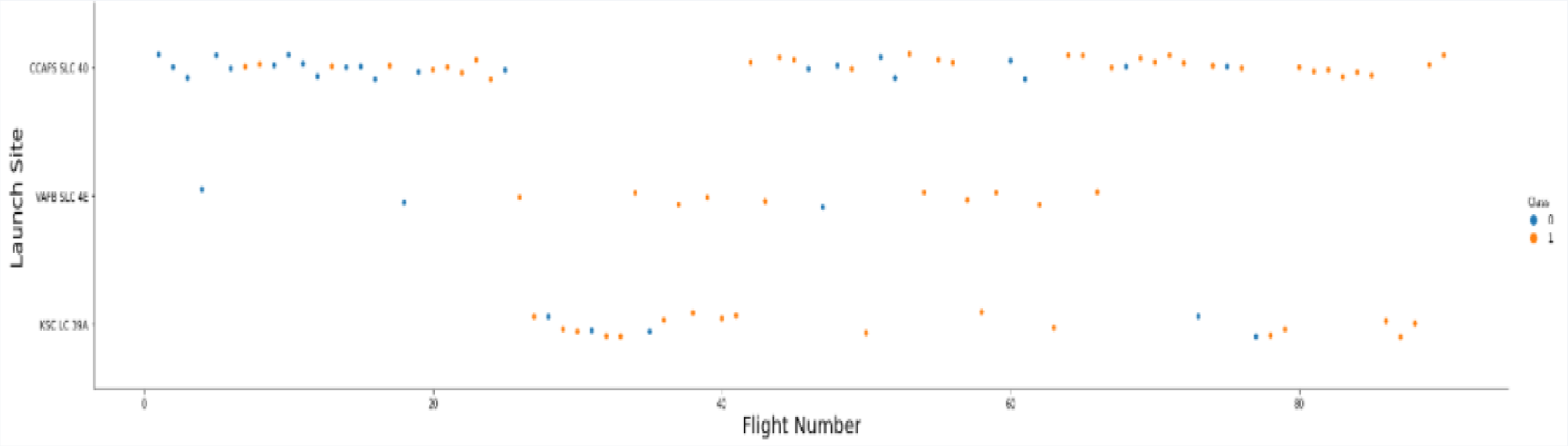
Flight Number vs. Launch Site

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



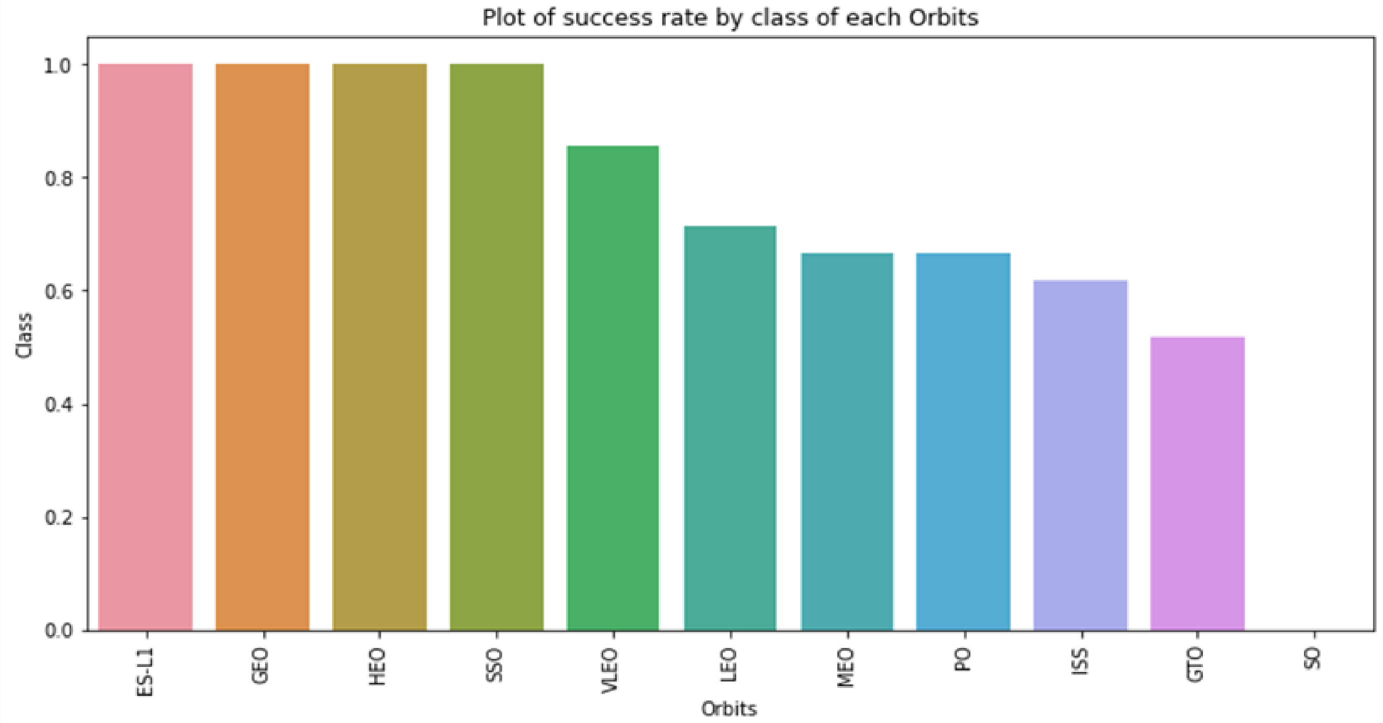
Payload vs. Launch Site

The Greate the Payload mass for launch site CCAFS SLC 40, The higher the success rate for the rocket.



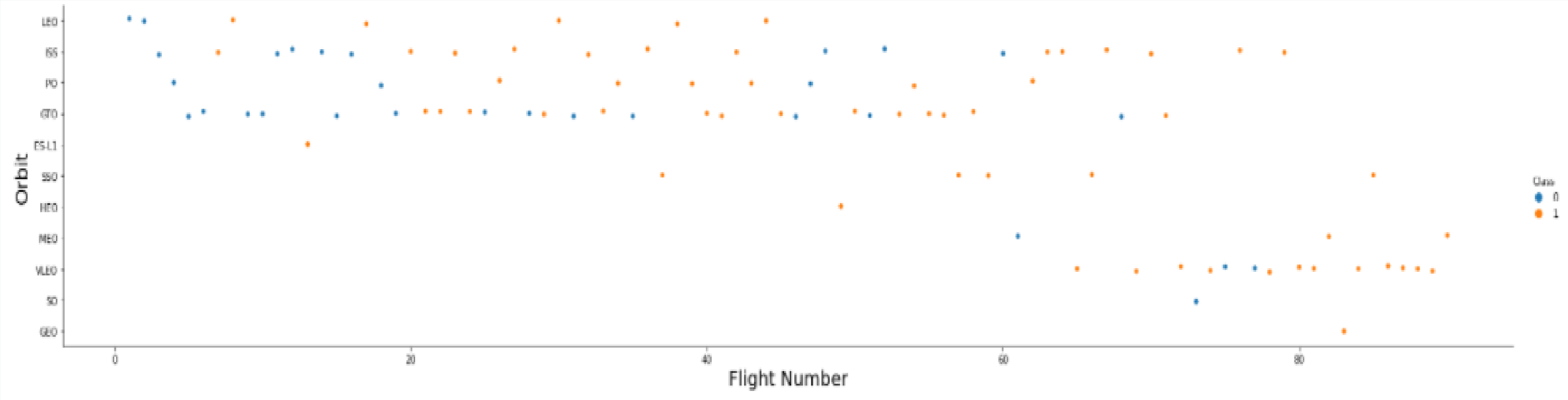
Success Rate vs. Orbit Type

From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



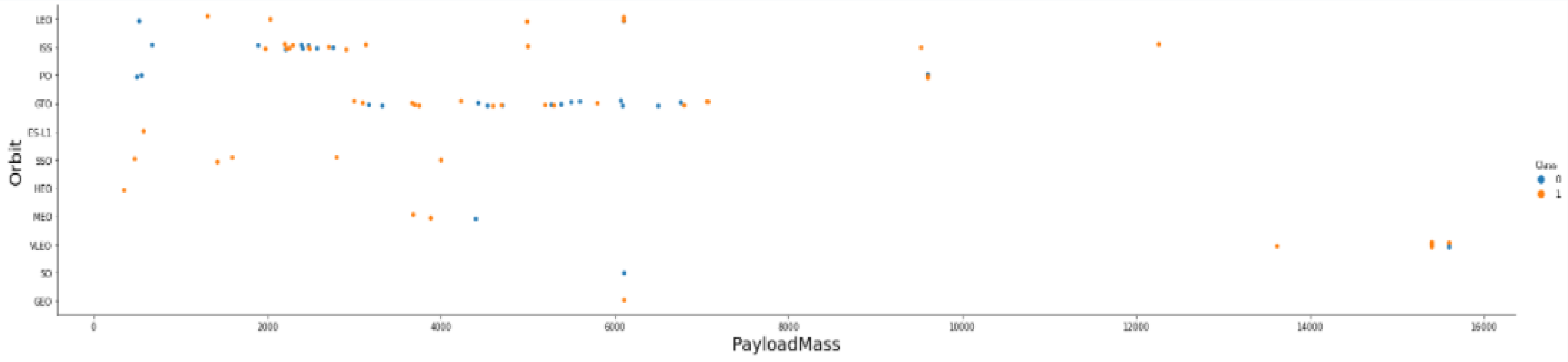
Flight Number vs. Orbit Type

The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.

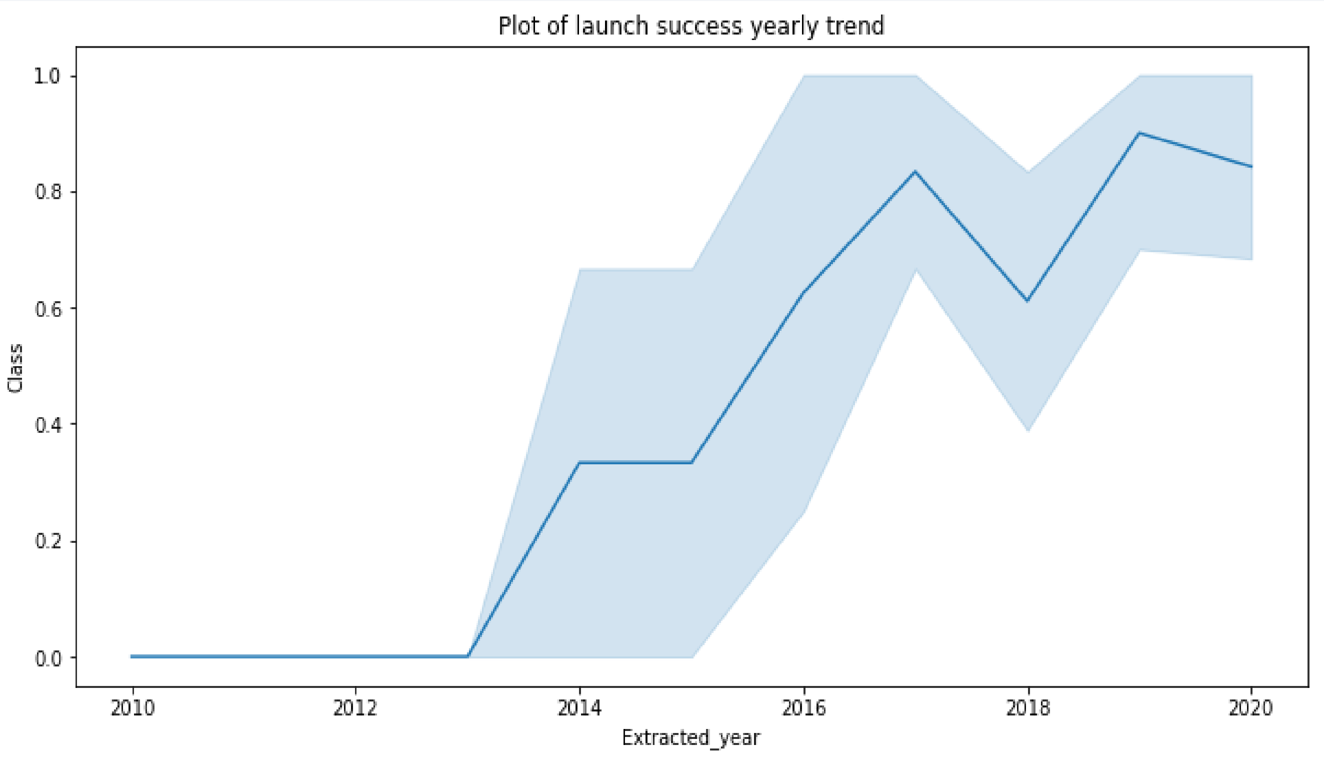


Payload vs. Orbit Type

We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend



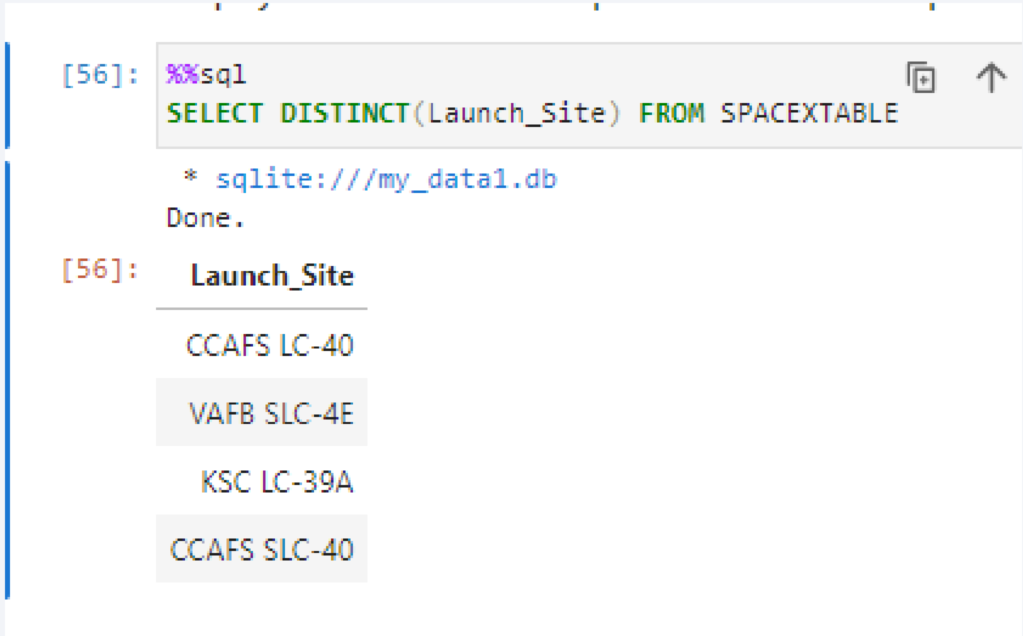
From the plot, we can observe that success rate since 2013 kept on increasing till 2020.

All Launch Site Names

We used the keyword **DISTINCT** on the LaunchSite to show only unique launch sites from the SpaceX data.

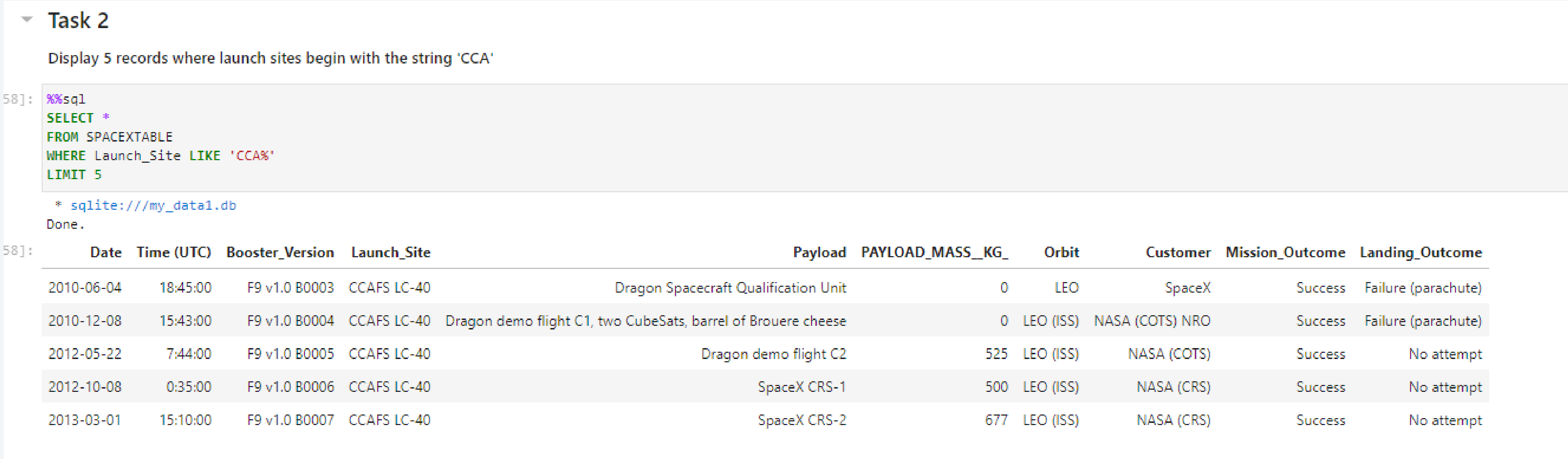
It shows

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



```
[56]: %%sql
      SELECT DISTINCT(Launch_Site) FROM SPACEXTABLE

       * sqlite:///my_data1.db
      Done.

[56]:   Launch_Site
        CCAFS LC-40
        VAFB SLC-4E
         KSC LC-39A
       CCAFS SLC-40
```

Launch Site Names Begin with 'CCA'



Task 2

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

```
58]: %%sql
     SELECT *
     FROM SPACEXTABLE
     WHERE Launch_Site LIKE 'CCA%'
     LIMIT 5
```

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

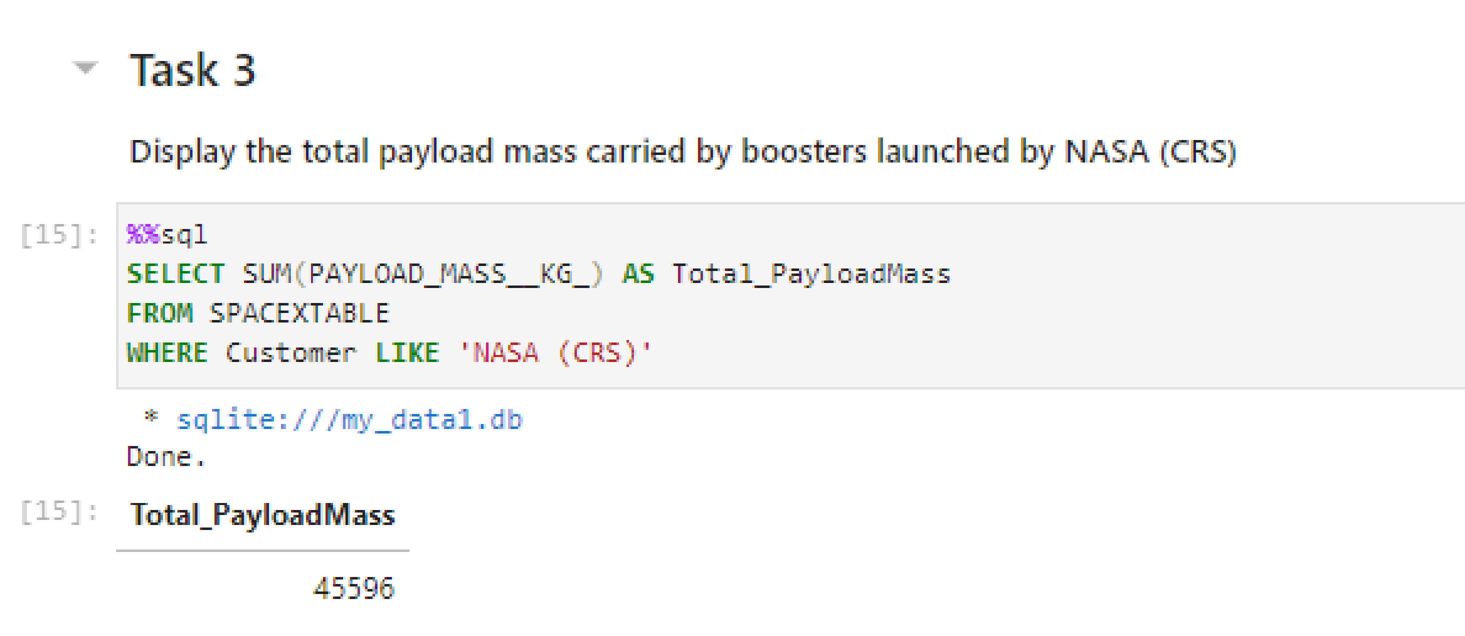
58]:

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

We used the query above to display 5 records where launch sites begin with `CCA` by using LIKE function and LIMIT for displaying only 5

Total Payload Mass

- We calculated the total payload carried by boosters launched by NASA is **45596** using the query below



Task 3

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

[15]:

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass
FROM SPACEXTABLE
WHERE Customer LIKE 'NASA (CRS)'
```

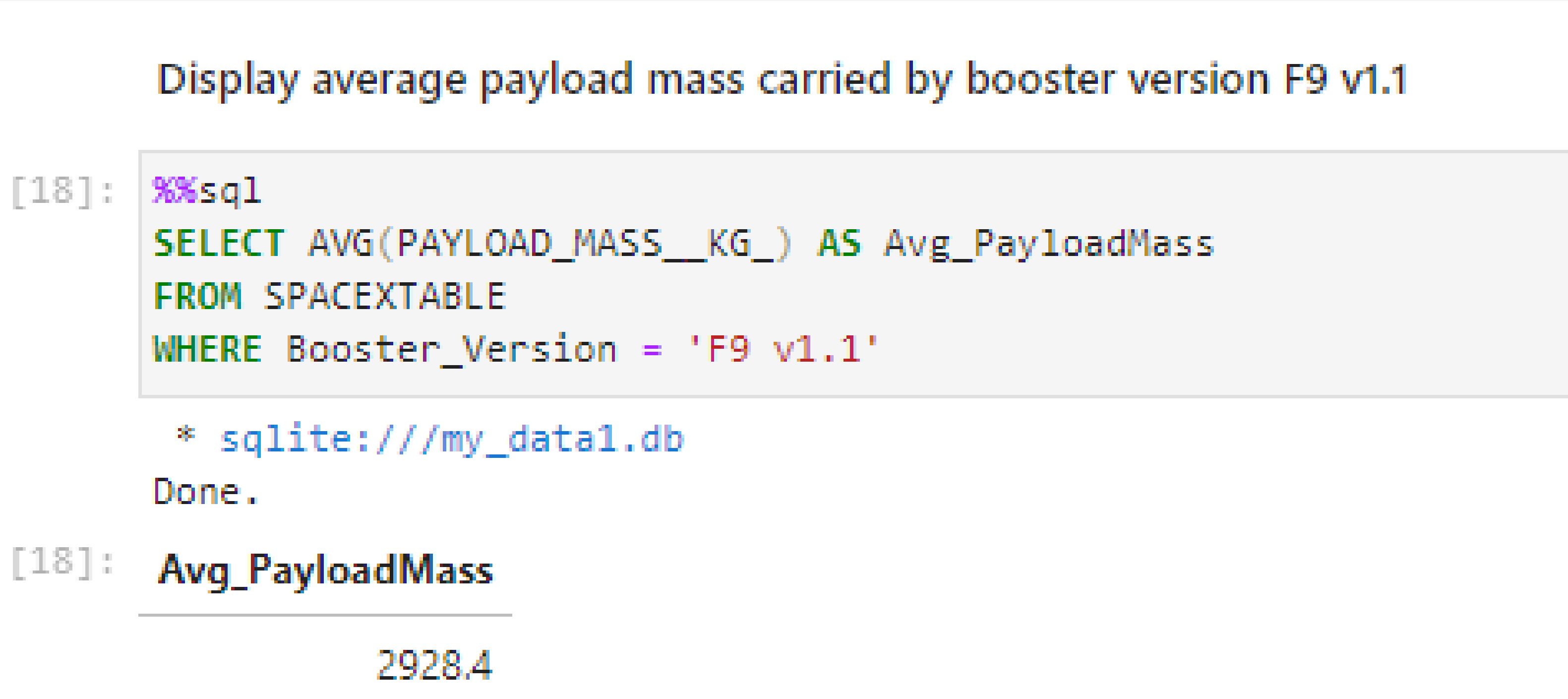
```
 * sqlite:///my_data1.db
Done.
```

[15]:

Total_PayloadMass
45596

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 is **<u>2928.4</u>**



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

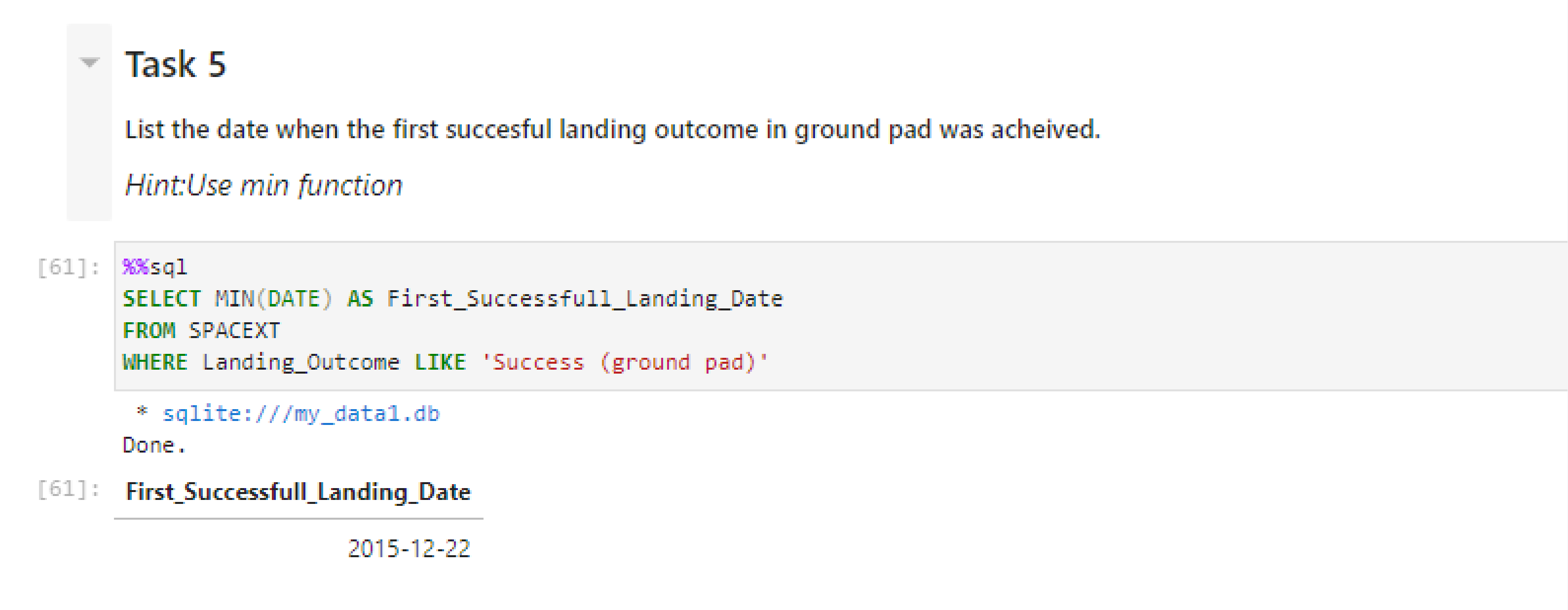
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_PayloadMass
FROM SPACEXTABLE
WHERE Booster_Version = 'F9 v1.1'
```

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

Avg_PayloadMass
2928.4

First Successful Ground Landing Date

We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015



Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

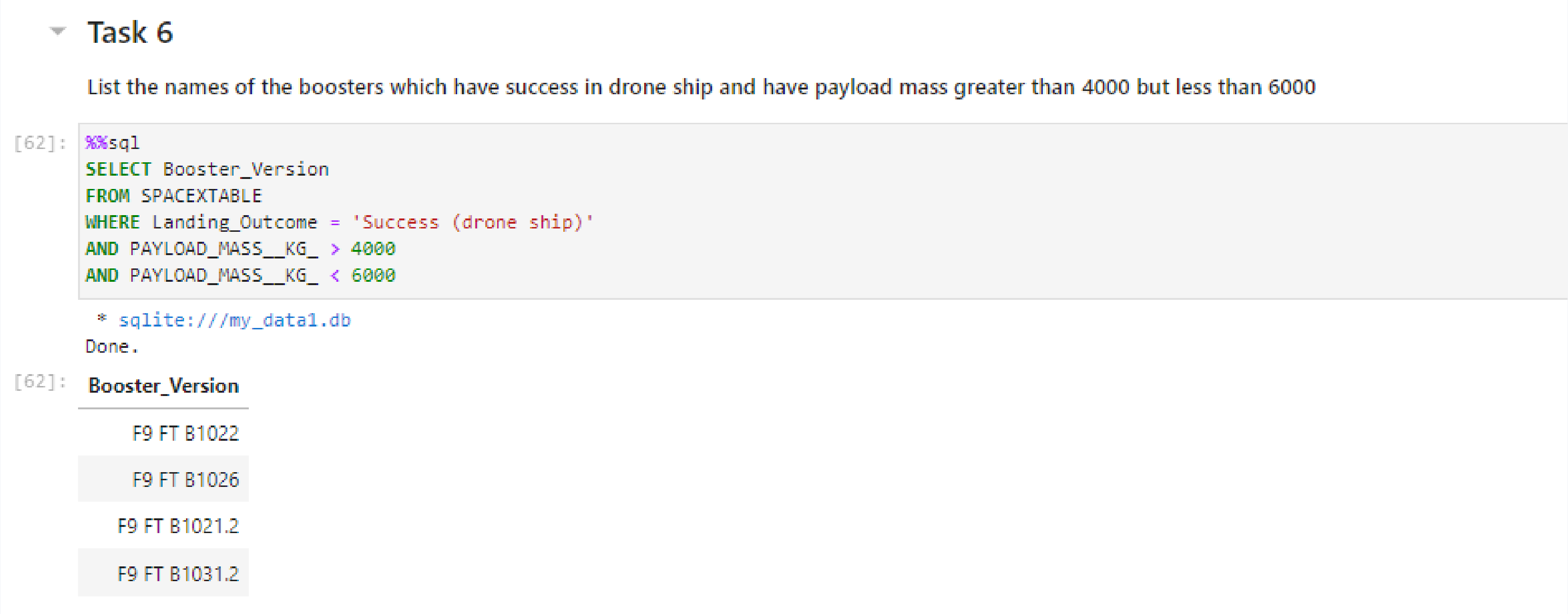
```
[61]: %%sql
SELECT MIN(DATE) AS First_Successfull_Landing_Date
FROM SPACEXT
WHERE Landing_Outcome LIKE 'Success (ground pad)'
```

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

[61]:

First_Successfull_Landing_Date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000



Task 6

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

[62]:

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

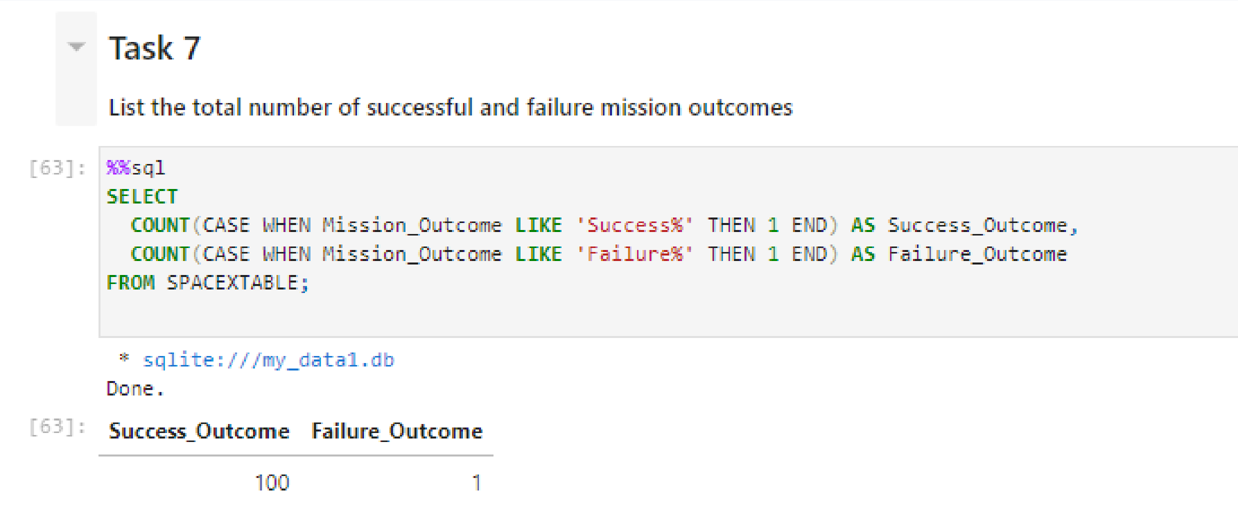
```
 * sqlite:///my_data1.db
Done.
```

[62]:

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

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

Total Number of Successful and Failure Mission Outcomes



Task 7

List the total number of successful and failure mission outcomes

```
[63]: %%sql
      SELECT
        COUNT(CASE WHEN Mission_Outcome LIKE 'Success%' THEN 1 END) AS Success_Outcome,
        COUNT(CASE WHEN Mission_Outcome LIKE 'Failure%' THEN 1 END) AS Failure_Outcome
      FROM SPACEXTABLE;
```

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

[63]:

Success_Outcome	Failure_Outcome
100	1

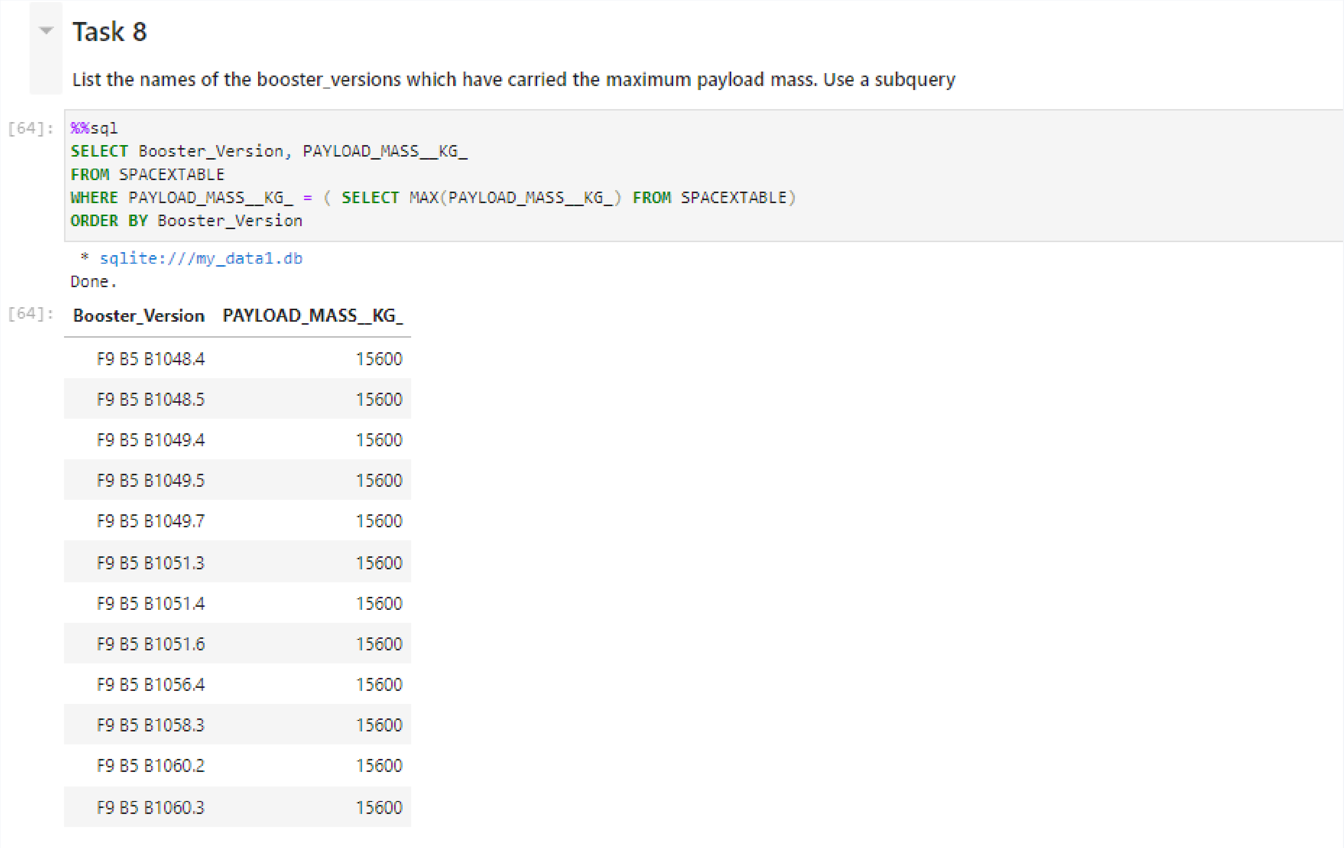
We used wildcard like ‘%’ to filter for **WHERE** MissionOutcome was a success or a failure.

We used **CASE** function to make it shorter code for doing both Mission_Outcome : **Success** or **Failure**

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.

Where The maximum payload is 15600 KG



Task 8

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

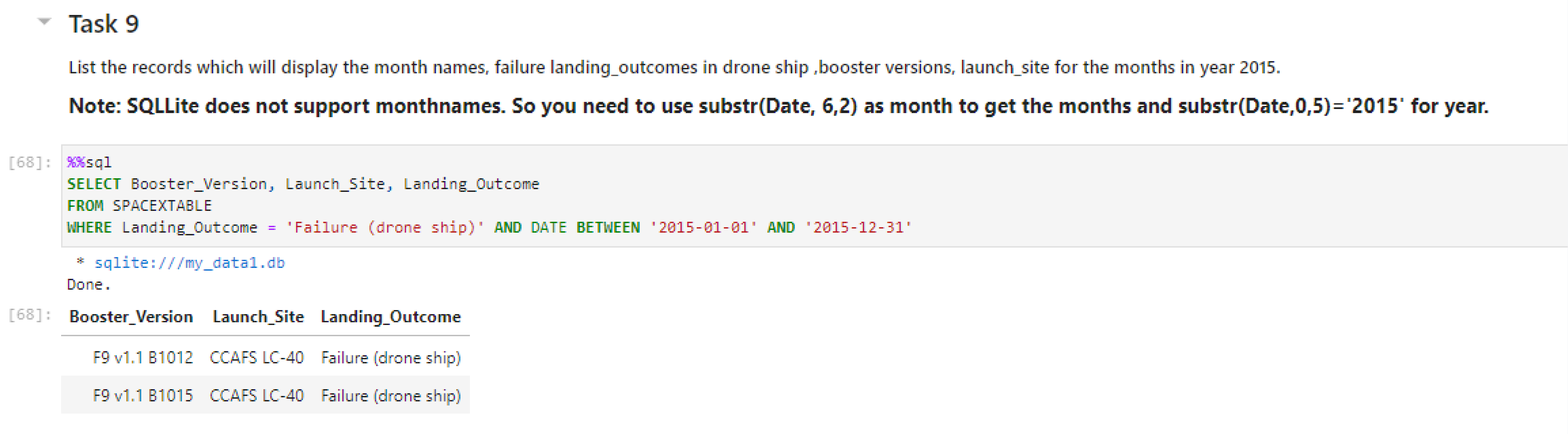
[64]:
```
%%sql
SELECT Booster_Version, PAYLOAD_MASS__KG_
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = ( SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
ORDER BY Booster_Version
```

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

[64]:

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

2015 Launch Records



Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[68]:

```
%%sql
SELECT Booster_Version, Launch_Site, Landing_Outcome
FROM SPACEXTABLE
WHERE Landing_Outcome = 'Failure (drone ship)' AND DATE BETWEEN '2015-01-01' AND '2015-12-31'
```

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

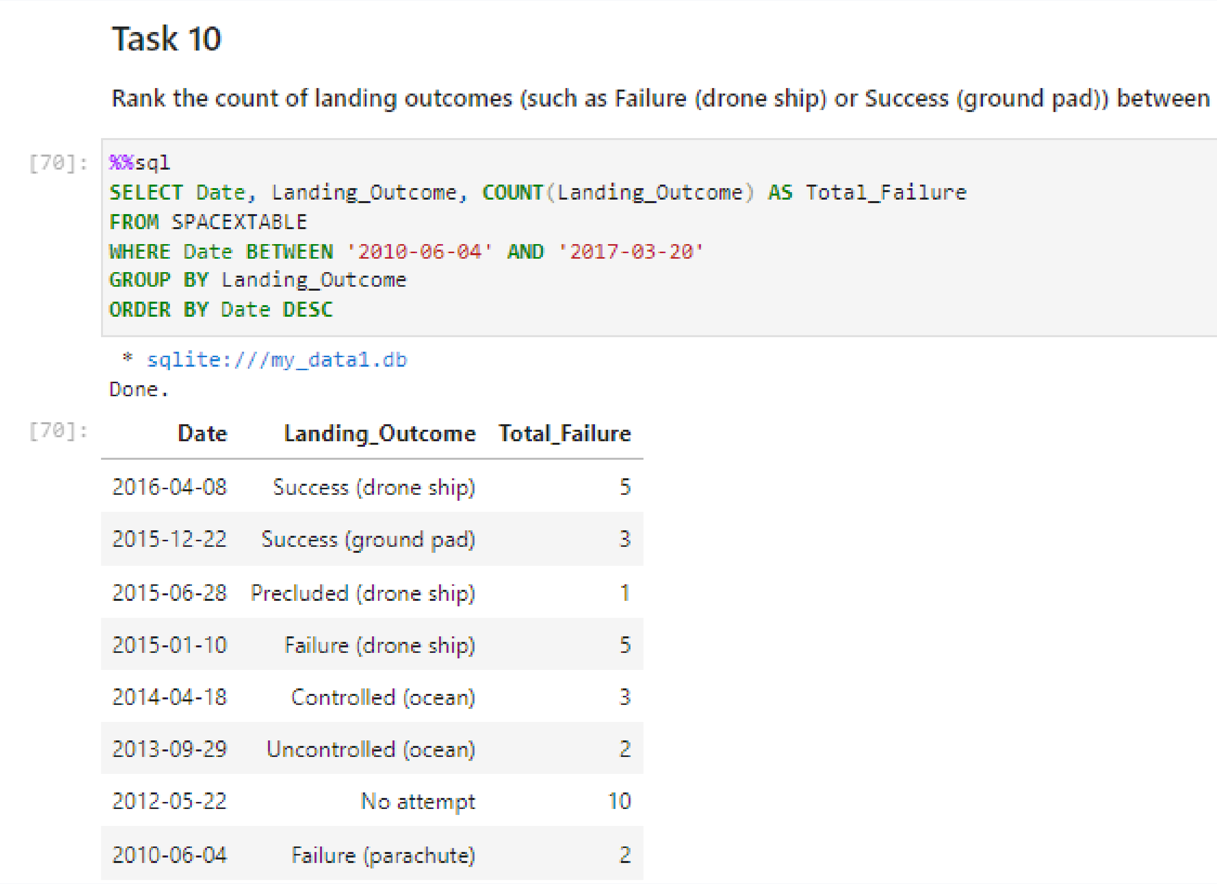
[68]:

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

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

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

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order based on date in descending order.



Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between

```
[70]: %%sql
SELECT Date, Landing_Outcome, COUNT(Landing_Outcome) AS Total_Failure
FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Date DESC
```

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

[70]:

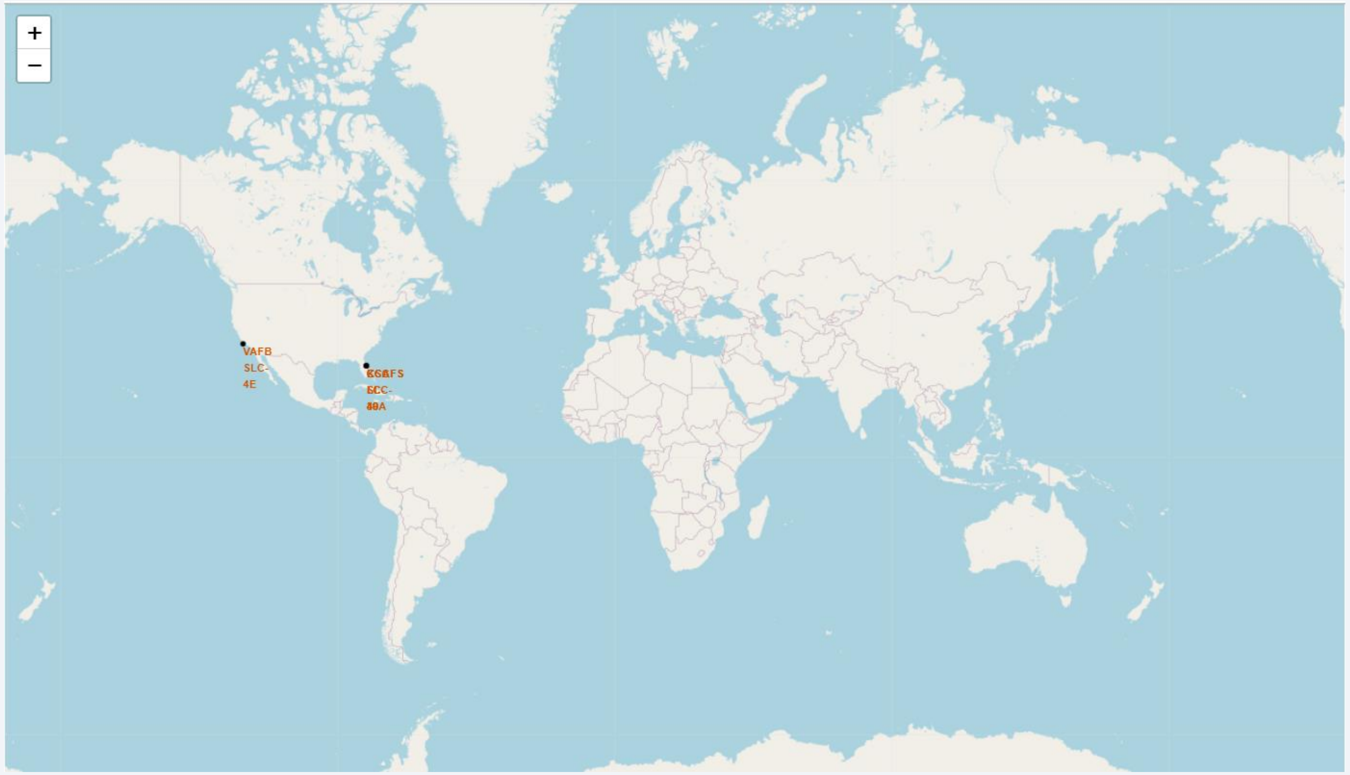
Date	Landing_Outcome	Total_Failure
2016-04-08	Success (drone ship)	5
2015-12-22	Success (ground pad)	3
2015-06-28	Precluded (drone ship)	1
2015-01-10	Failure (drone ship)	5
2014-04-18	Controlled (ocean)	3
2013-09-29	Uncontrolled (ocean)	2
2012-05-22	No attempt	10
2010-06-04	Failure (parachute)	2

Section 3

Launch Sites Proximities Analysis

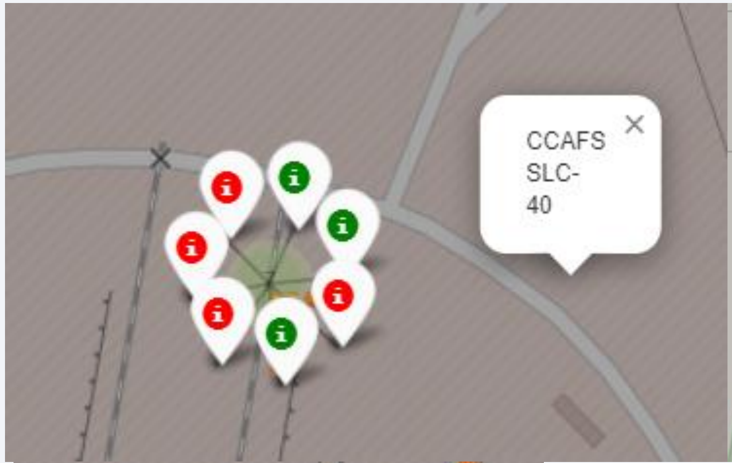
All Launch Sites Global Map Markers

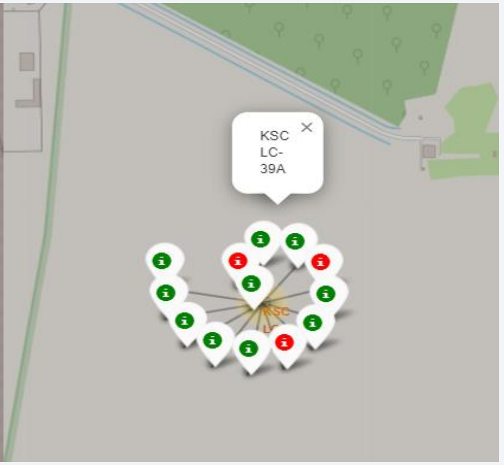
We can see that the SpaceX Launch Site are in the United States of America coasts: Florida and California

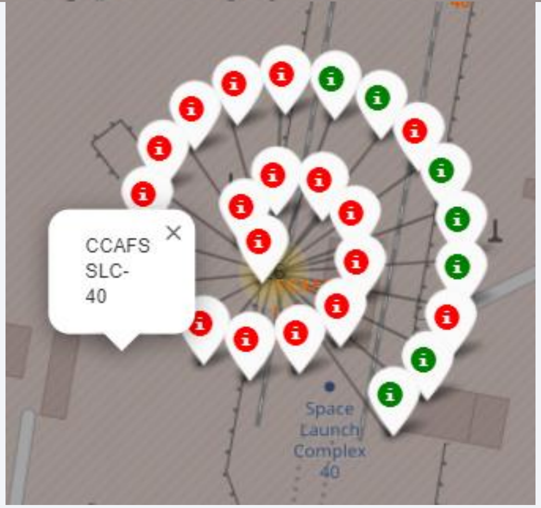


Markers showing successful launch using color labels

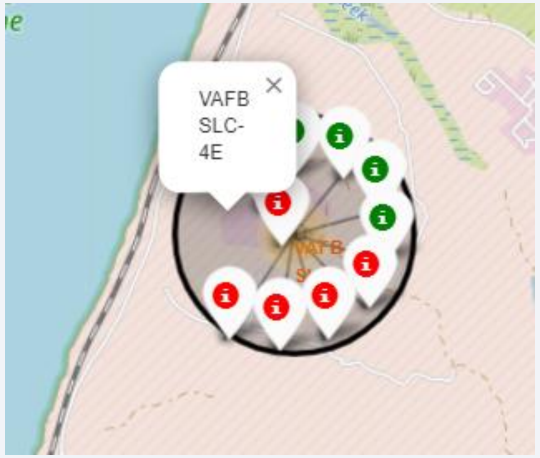
Florida Launch Sites







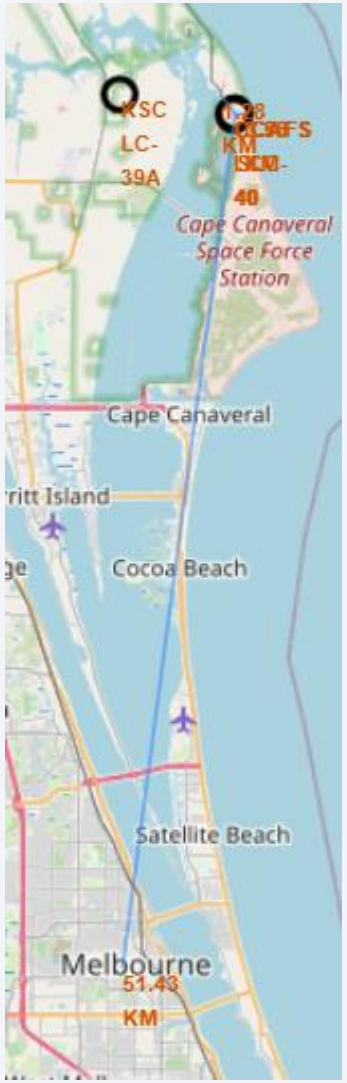
California Launch Sites

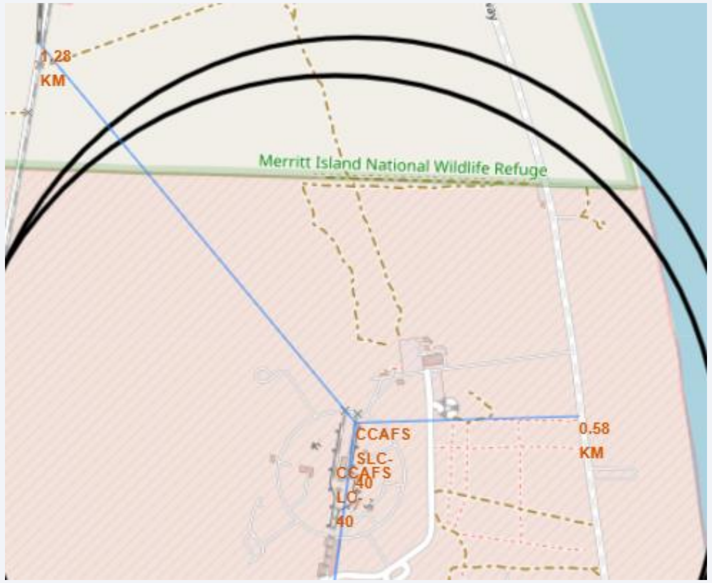


Green Marker Shows successful Launch

Red Marker Shows Failures Launch

Launch Site (CCAFS SLC-40) Distance to Landmarks

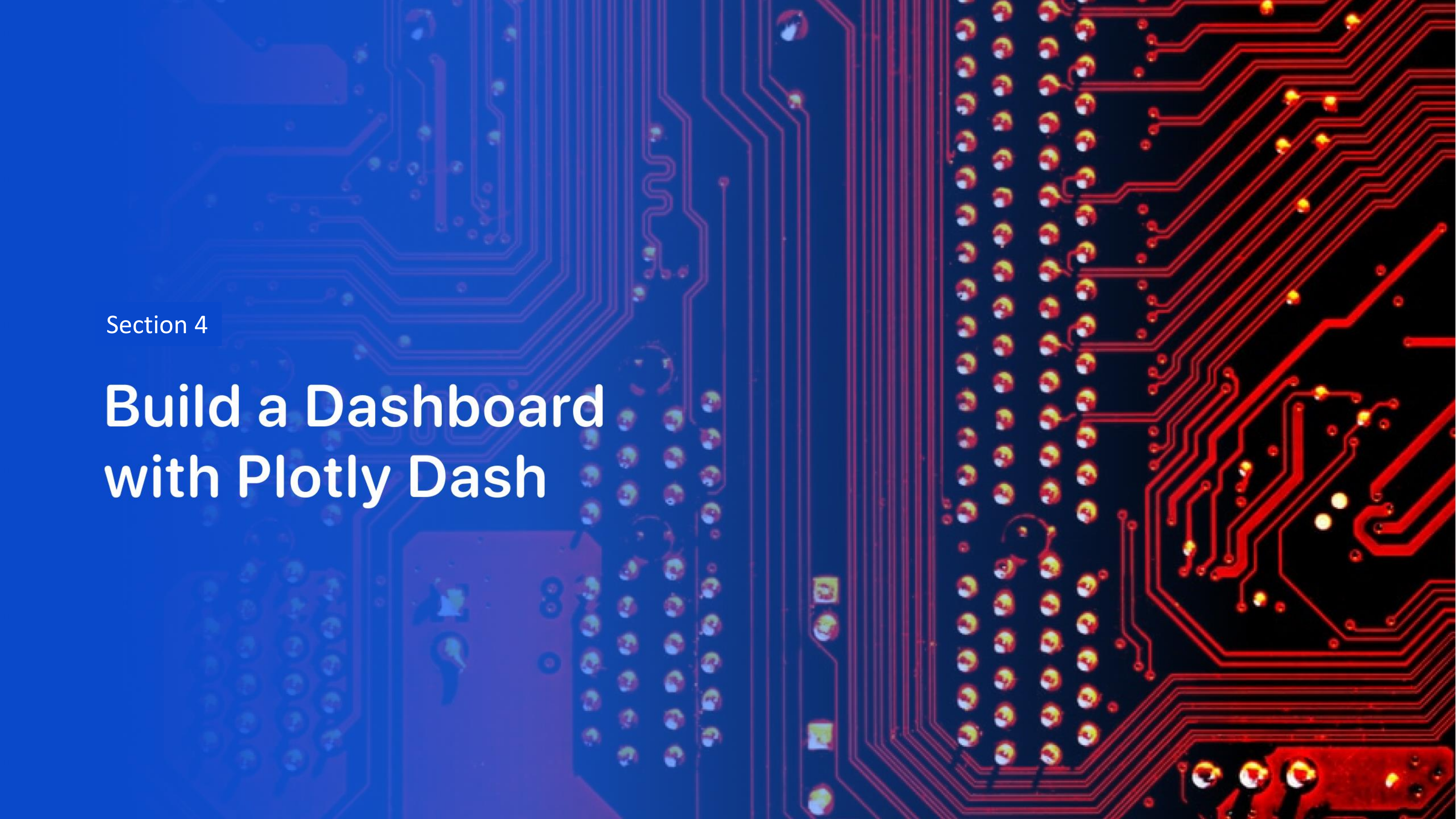




P
33
CCAFS SLC-40
CCAFS LC-40
Samuel C Phillips Parkway
0.58 KM

distance_highway = 0.5834695366934144 km
distance_railroad = 1.2845344718142522 km
distance_city = 51.434169995172326 km

Are launch sites in close proximity to railways? **No**
Are launch sites in close proximity to highways? **No**
Are launch sites in close proximity to coastline? **Yes**
Do launch sites keep certain distance away from cities? **Yes**

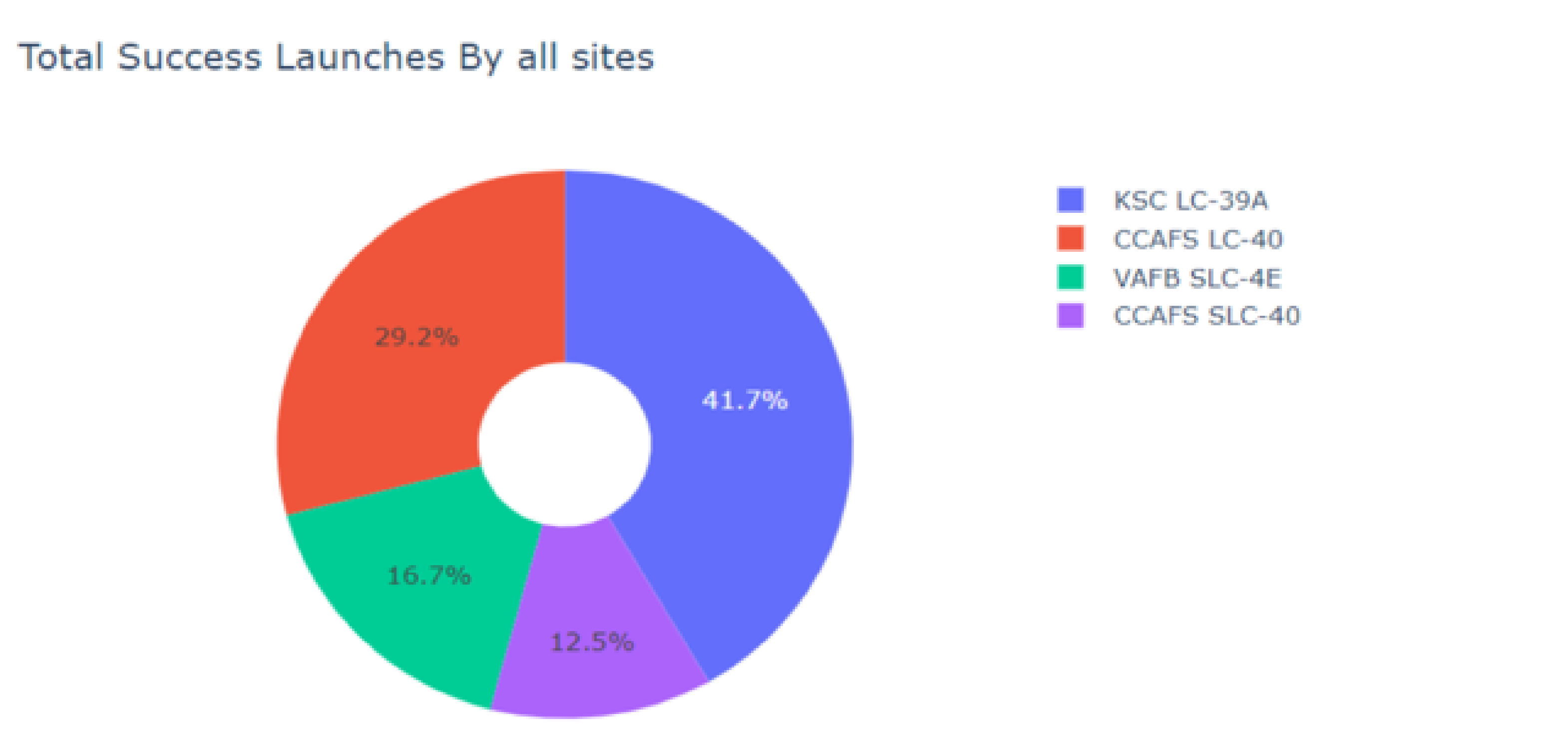


Section 4

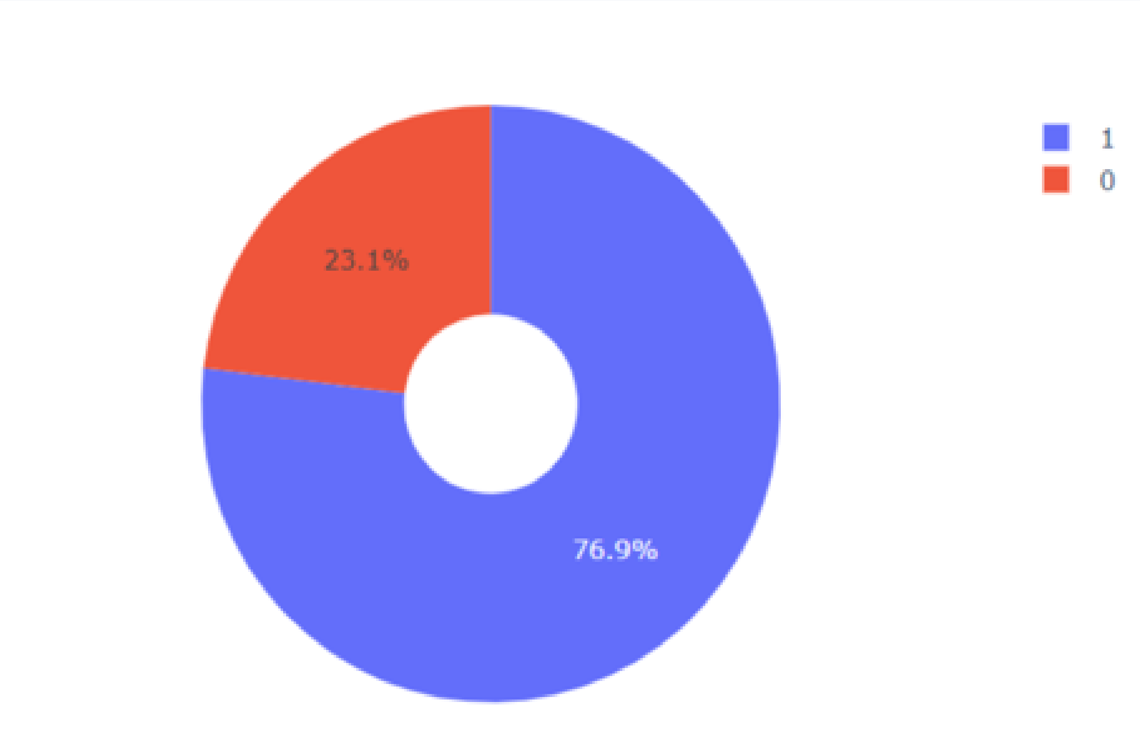
Build a Dashboard with Plotly Dash

Pie Chart Showing The Success Percentage Achieved from Each Launch Site

We can see that KSC LC-39A had the most successful launches from all Launch Sites.



Pie Chart Showing The Launch Site with The Highest Launch Success Ratio

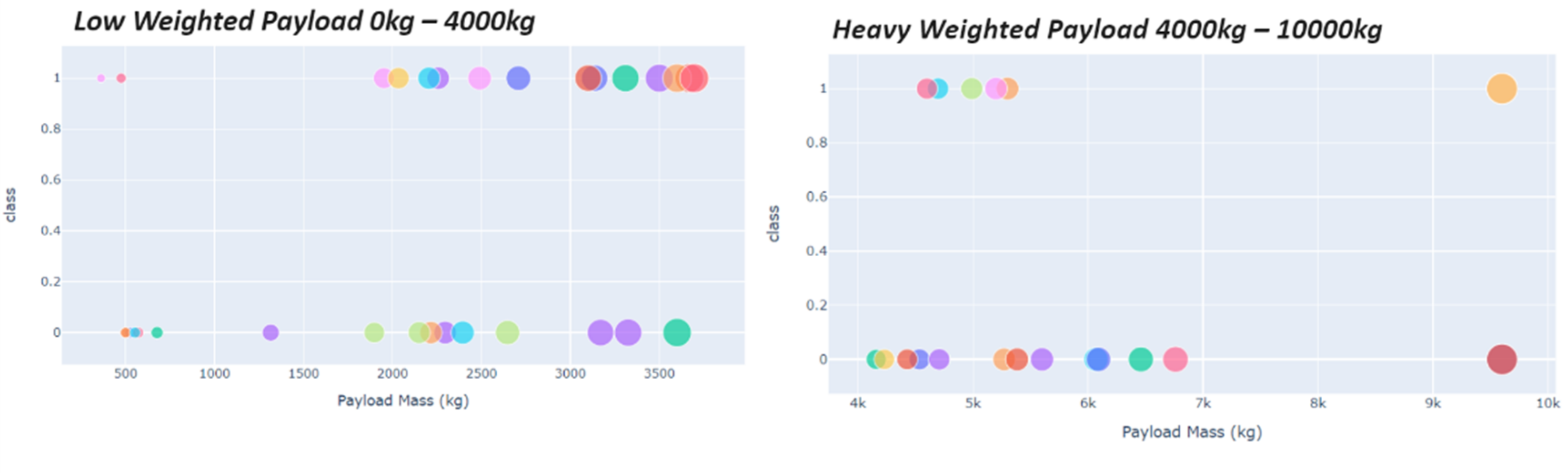


KSC LC-39A Achieved 76,9% success rate while getting 23,1% failure rate

1 means Success

0 means Failure

Scatter Plot of Payload vs Launch Outcome for All Launch Sites (with Different Payloads)

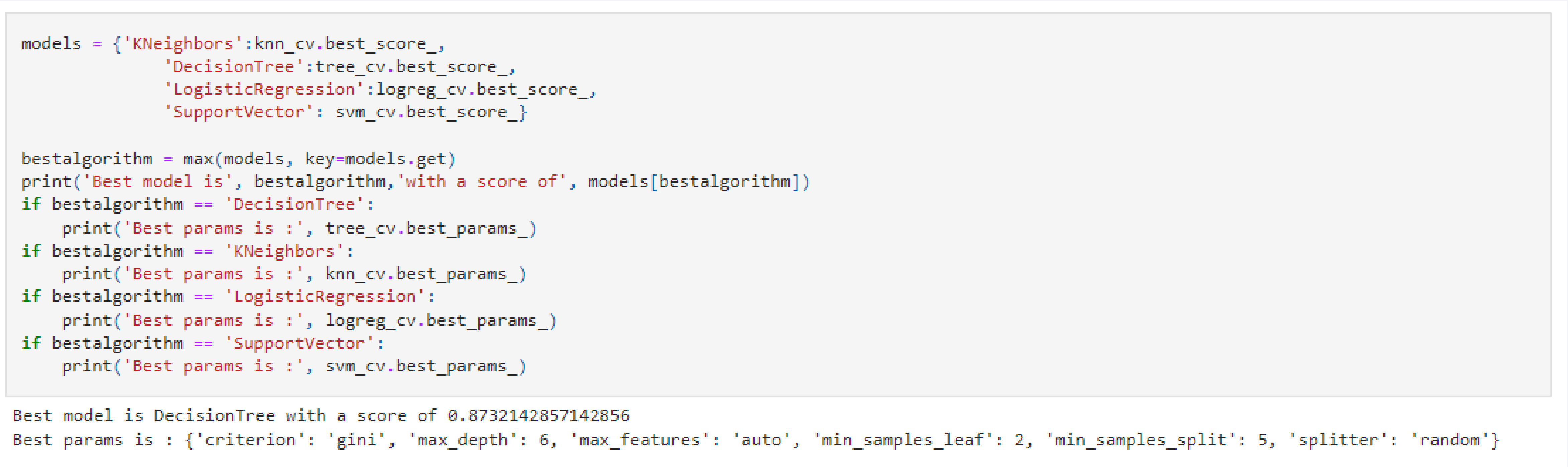


Based on the graph above, We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy



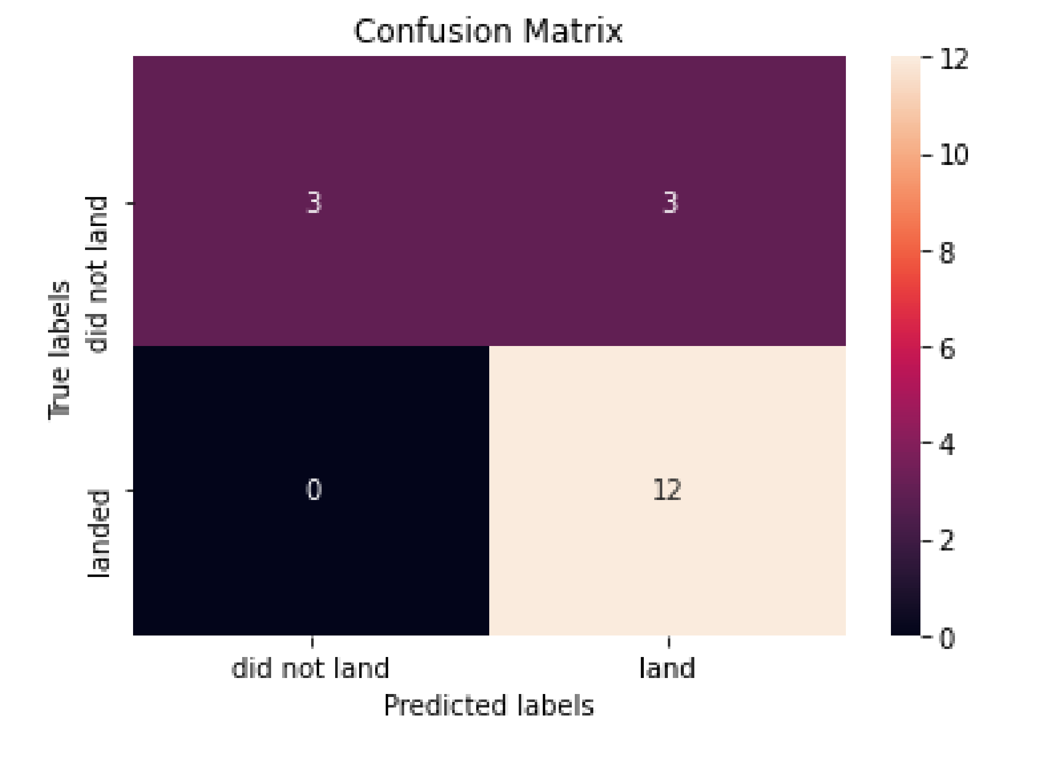
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Based on the code above, The **decision tree classifier** is the model with the highest classification accuracy

Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

In summary:

- A launch site's success rate is positively correlated with the magnitude of flights conducted there.
- The launch success rate witnessed a continuous rise from 2013 to 2020.
- Orbits ES-L1, GEO, HEO, SSO, and VLEO exhibited the highest success rates.
- Among all launch sites, KSC LC-39A boasted the highest number of successful launches.
- The Decision tree classifier stands out as the most effective machine learning algorithm for this analysis.

Thank you!