SpaceX with Data Science

By Viktor Berg

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API's
  - Data Collection with Web Scrapping
  - 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 results
  - Interactive Analytics
  - Predictive Analysis results

Introduction

- Project background

SpaceX advertises Falcon 9 launches on its website for a cost of 62 million dollars. Compared to other rocket providers costing approximately 165 million dollars each. Most of these savings come from the fact that SpaceX can reuse the first stage. Meaning, if we can predict if the first stage will land safely, we can estimate the cost of launch. This information can be used if a competitor company wants to bid against SpaceX for a rocket launch of competitive price. The goal of this project is to create a machine learning pipeline to predict if the first stage of the Falcon 9 will land successfully.

Introduction

- Problems you want to find answers
- What are the major factors that will determine if the rocket will land safely?
- What operating conditions must be in place to ensure a successful landing program?
    - And how sensitive is the system?
- How will the interactions between varying features and systems affect the prediction success rate?

Methodology

Executive Summary

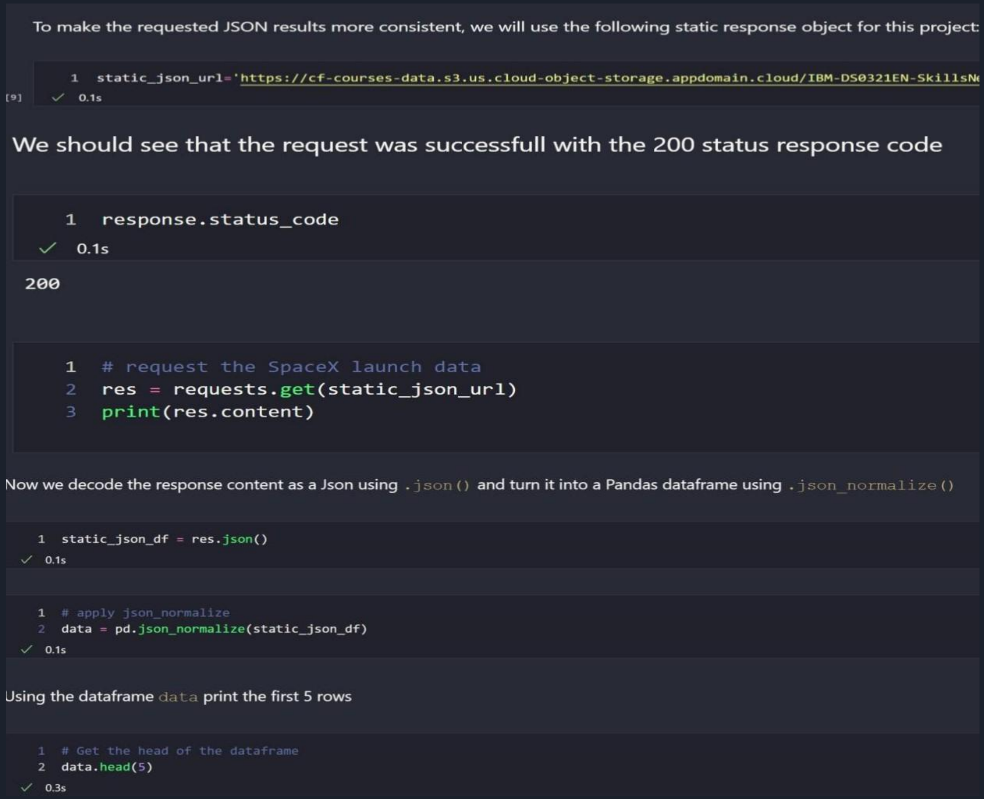
- Data collection methodology:
  - Data was collected using SpaceX API and Webscraping.
- Perform data wrangling
  - One-hot encoding applied to categorical features.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash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

Data Collection

- The data was collected using the following methods:
  - Collection was done using get requests to the SpaceX API.
  - Next was decoding the response content as a JSON using .json() function to then turn it into a pandas data frame using .json_normalize().
  - Next was cleaning the data.
    - Identified missing values
    - Input missing values
  - Performed Webscraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The overall object was to extract the launch records as HTML table, parse the table and convert it to a pandas data frame for later 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.



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

```
[9] static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsN
    ✓ 0.1s
```

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

```
response.status_code
✓ 0.1s
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()`

```
static_json_df = res.json()
✓ 0.1s
```

```
# apply json_normalize
data = pd.json_normalize(static_json_df)
✓ 0.1s
```

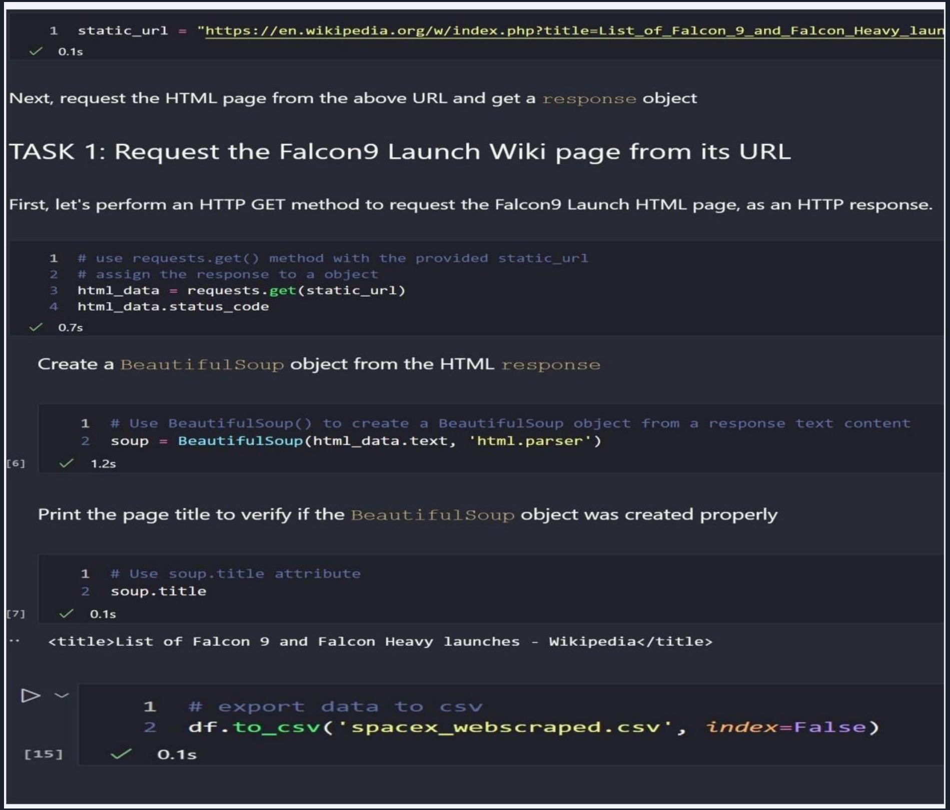
Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head(5)
✓ 0.3s
```

Data Collection - Scraping

- Web scrapping was applied to webscrap Falcon 9 launch records with BeautifulSoup

- Tables were parsed and converted it into a pandas dataframe .



```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_laun
✓ 0.1s
```

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.

```
# 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
✓ 0.7s
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
[6] ✓ 1.2s
```

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

```
# Use soup.title attribute
soup.title
[7] ✓ 0.1s
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

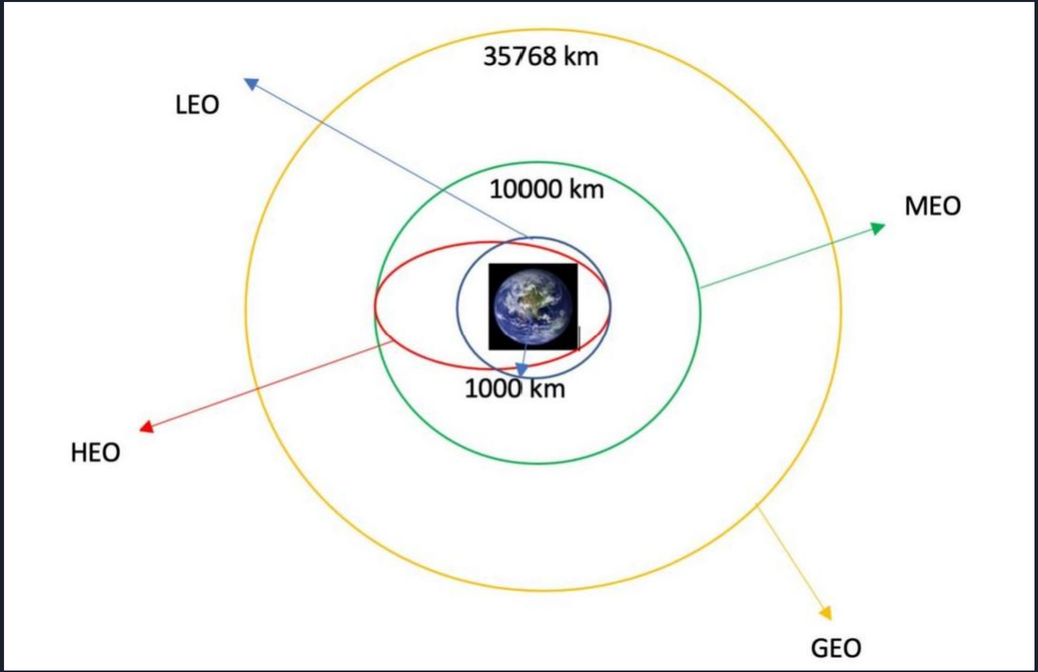
```
# export data to csv
df.to_csv('spacex_webscraped.csv', index=False)
[15] ✓ 0.1s
```

Data Wrangling

We performed exploratory data analysis and determined the training labels.

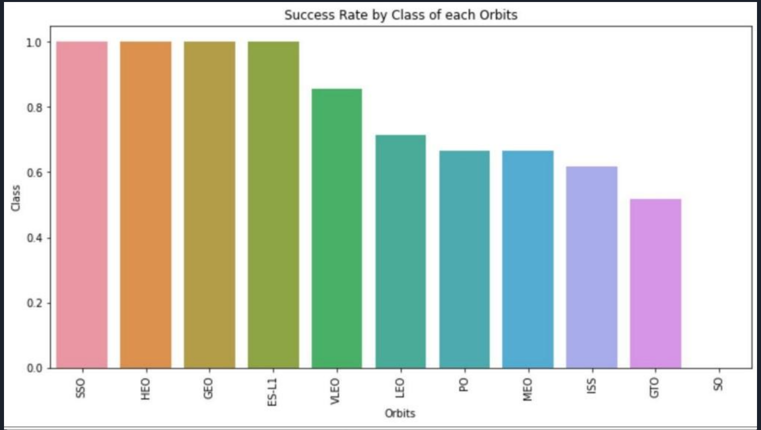
• We calculated the number of launches at each site, and the number and occurrence of each orbits

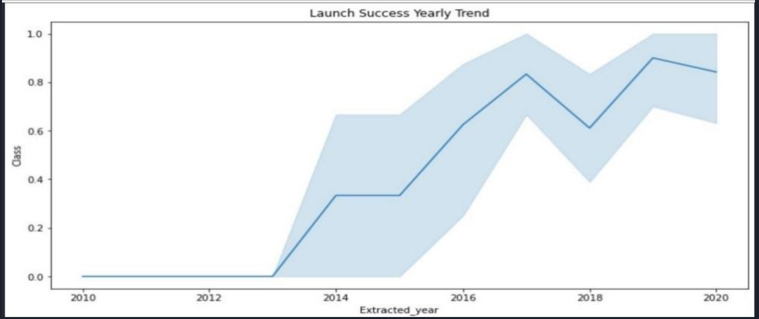
• We created landing outcome label from outcome column and exported the results to csv.



EDA with Data Visualization

The data was explored 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.





EDA with SQL

• SpaceX dataset was loaded into a SQL database without leaving the jupyter notebook.

• EDA was applied 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.

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

- 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.

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.

Results

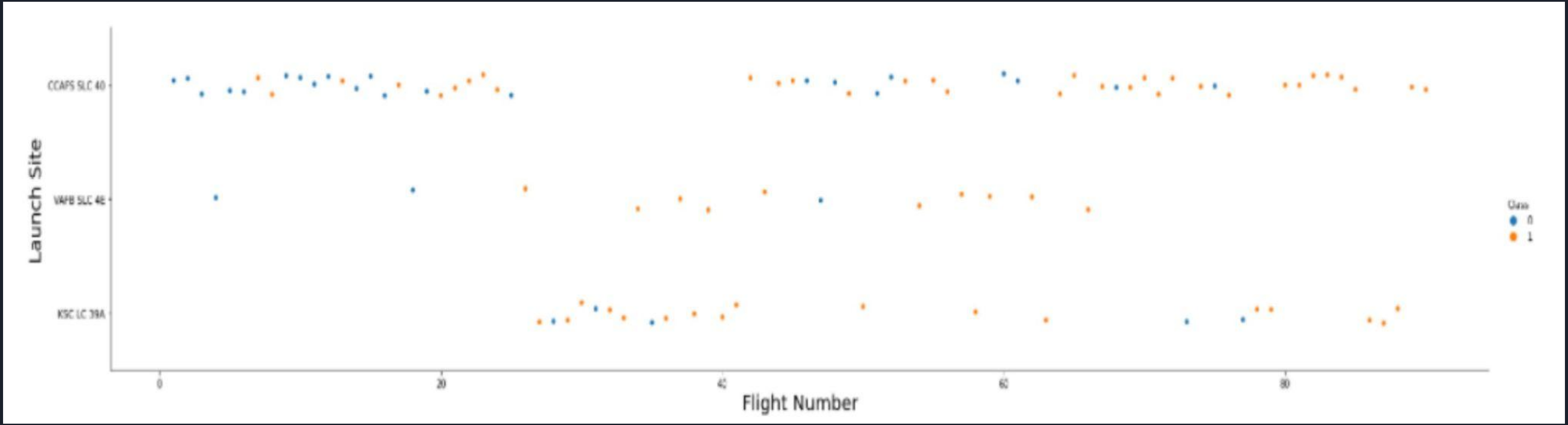
Exploratory data analysis results

- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2
Insights drawn from EDA

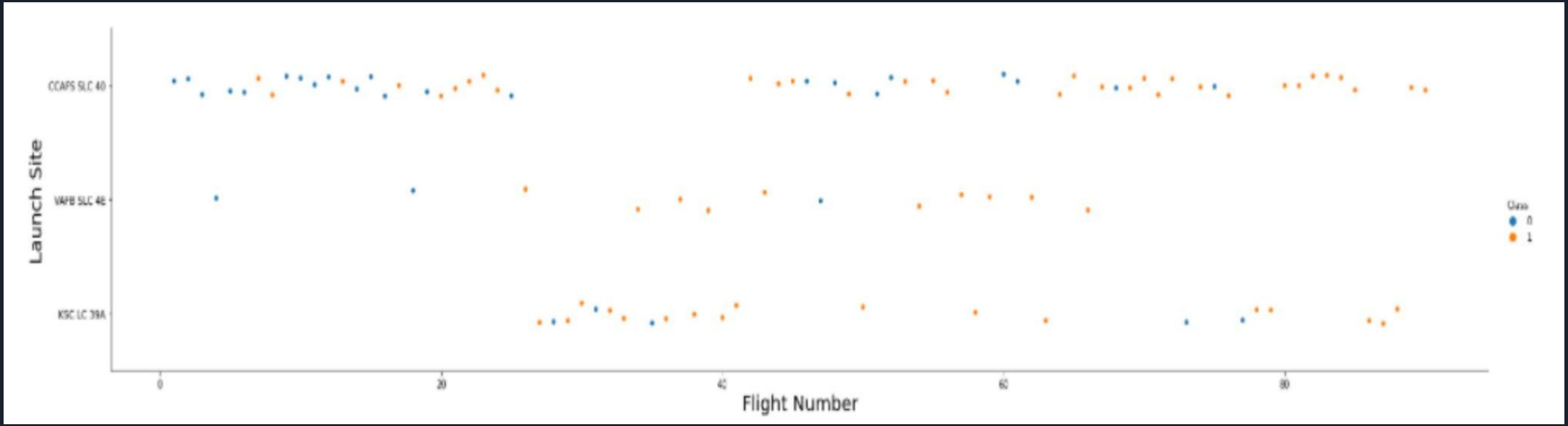
Flight Number vs. Launch Site

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



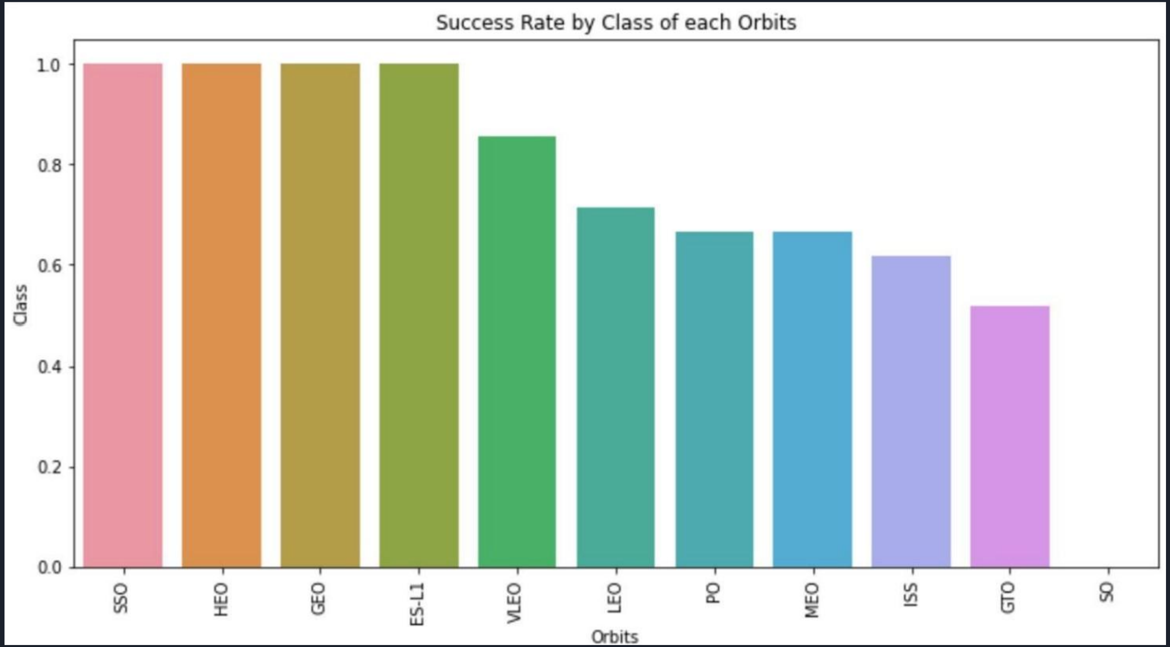
Payload vs. Launch Site

Relationship: the greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the launch



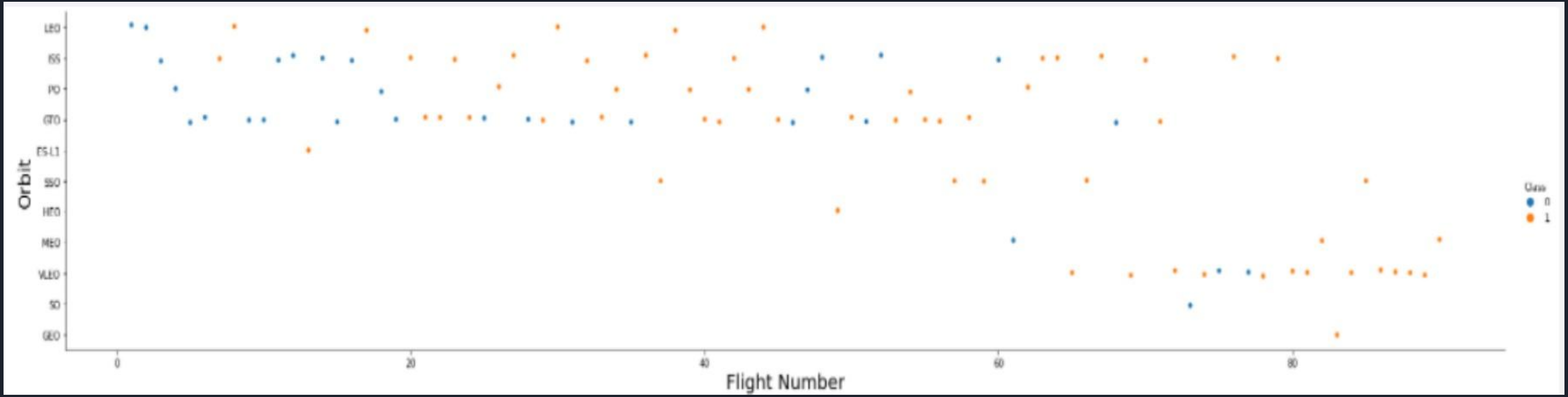
Success Rate vs. Orbit Typ

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



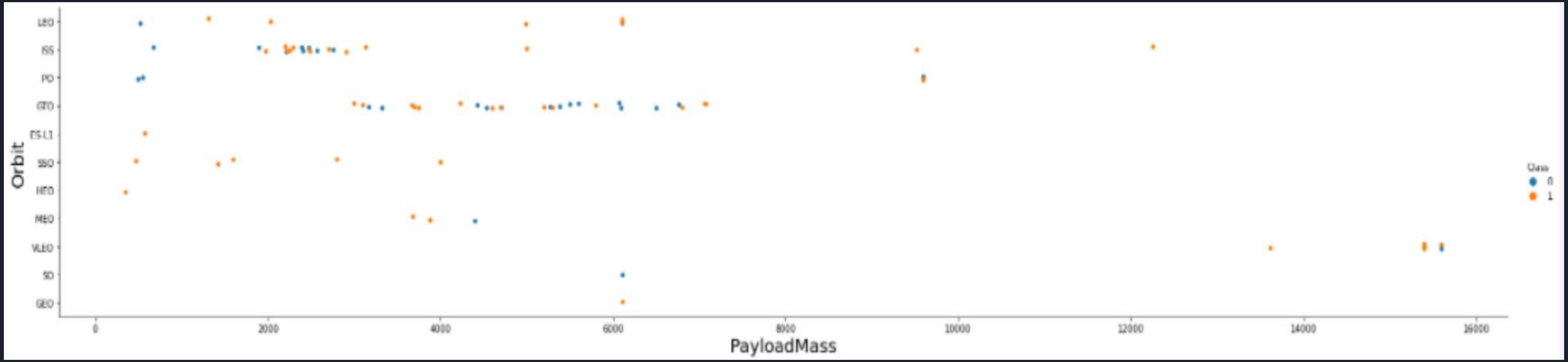
Flight Number vs. Orbit Type

• 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.



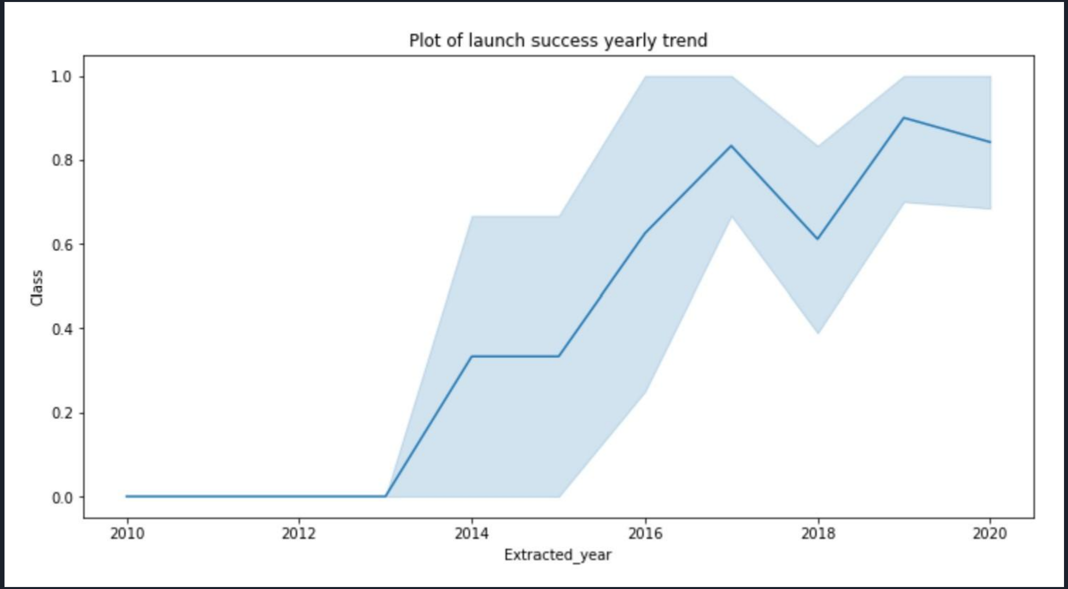
Payload vs. Orbit Type

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



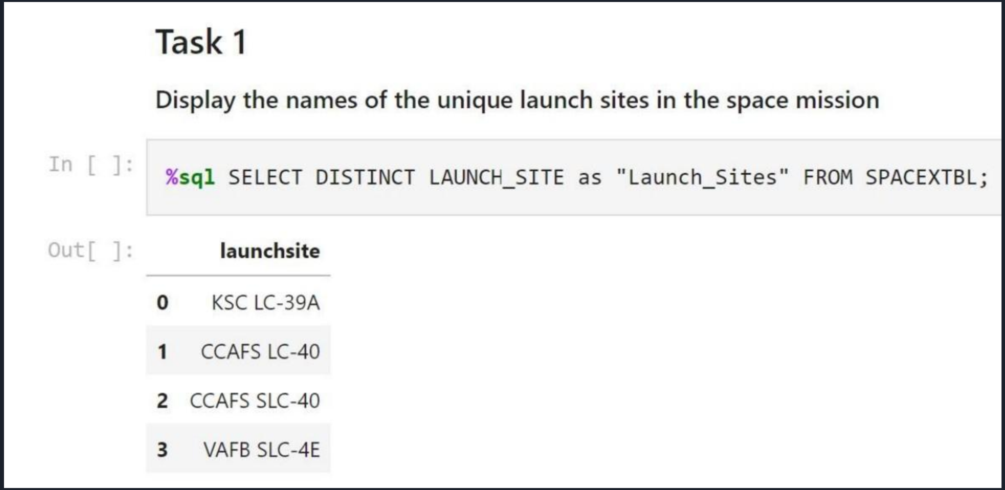
Launch Success Yearly Trend

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



All Launch Site Names

We used the key word DISTINCT to show only unique launch sites from the SpaceX data



Task 1

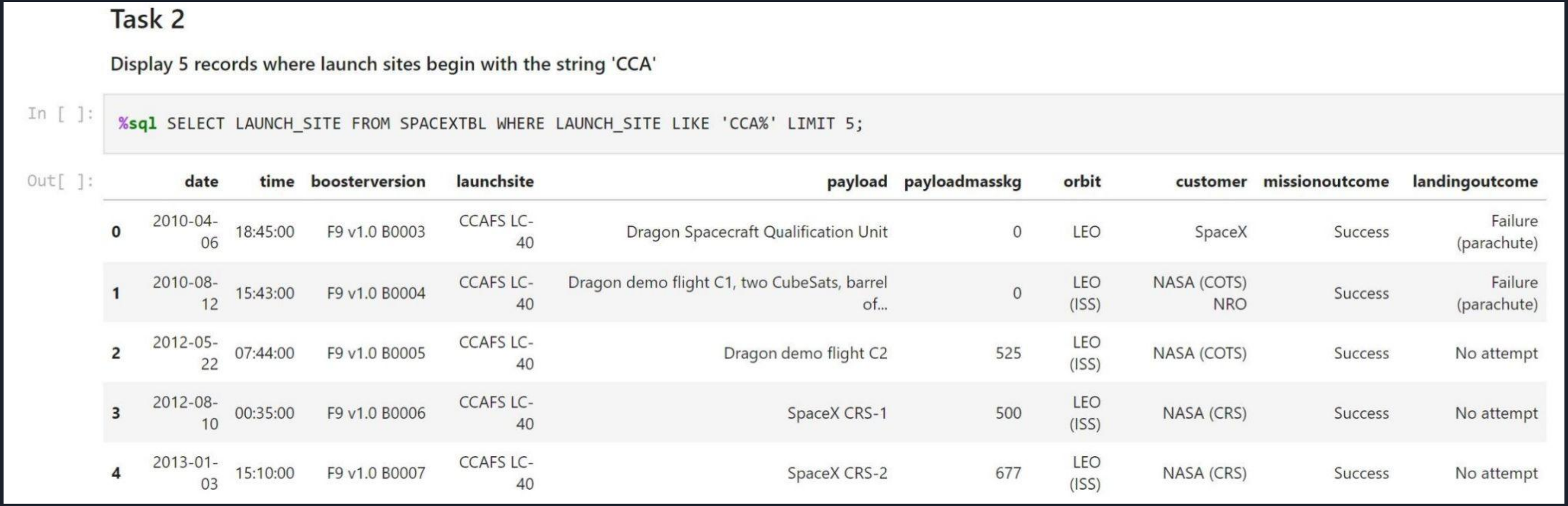
Display the names of the unique launch sites in the space mission

```
In [ ]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

Launch Site Names Begin with 'CCA'

The query below was used to display 5 records where launch sites begin with `CCA`



Task 2

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

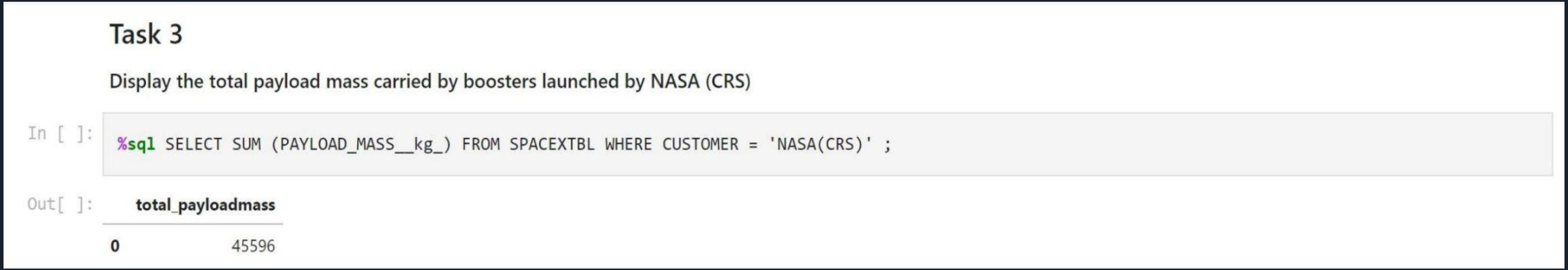
```
In [ ]: %sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

Out[ ]:

	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

Total Payload Mass

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



Task 3

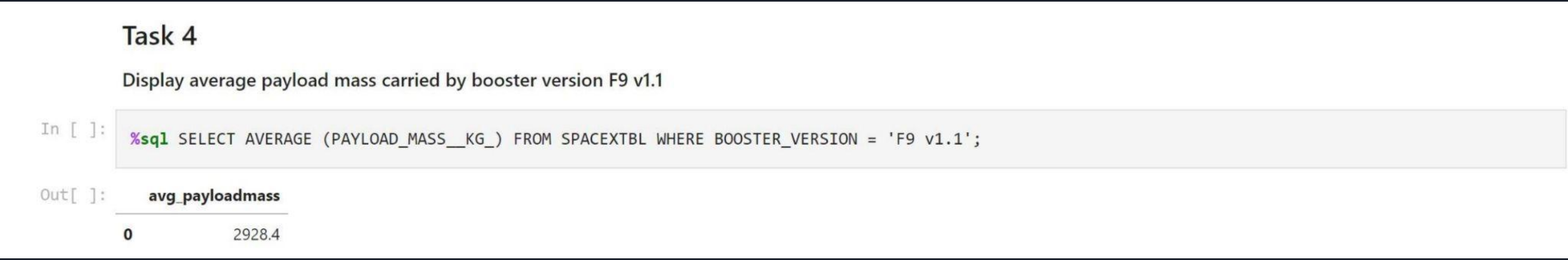
Display the total payload mass carried by boosters launched by NASA (CRS)

```
In [ ]: %sql SELECT SUM (PAYLOAD_MASS__kg_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA(CRS)' ;
```

```
Out[ ]:    total_payloadmass
        0              45596
```

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4



Task 4

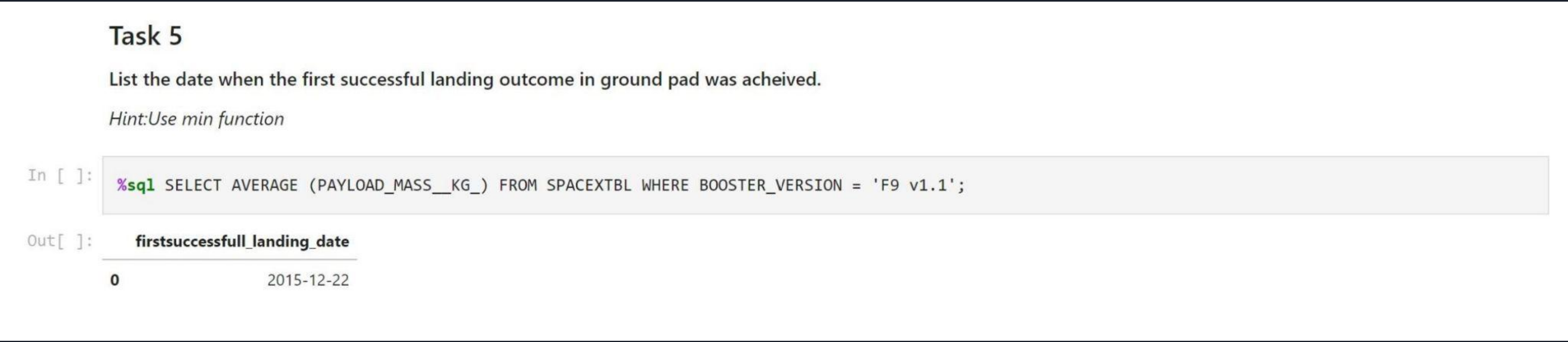
Display average payload mass carried by booster version F9 v1.1

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

```
Out[ ]:    avg_payloadmass
        0           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 successful landing outcome in ground pad was acheived.

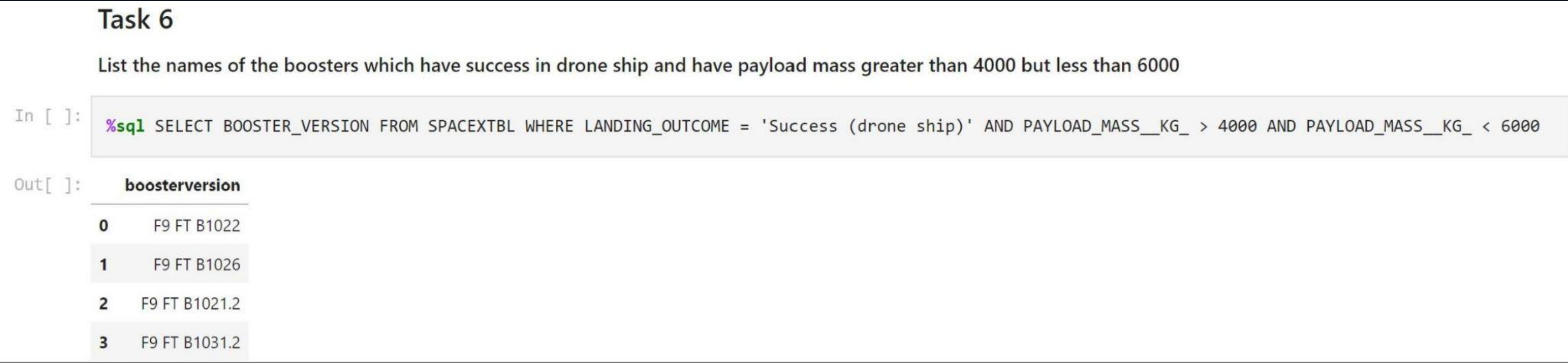
Hint:Use min function

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

```
Out[ ]:    firstsuccessfull_landing_date
        0                     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 the AND condition to determine successful landing with payload mass greater than 4000 but less than 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

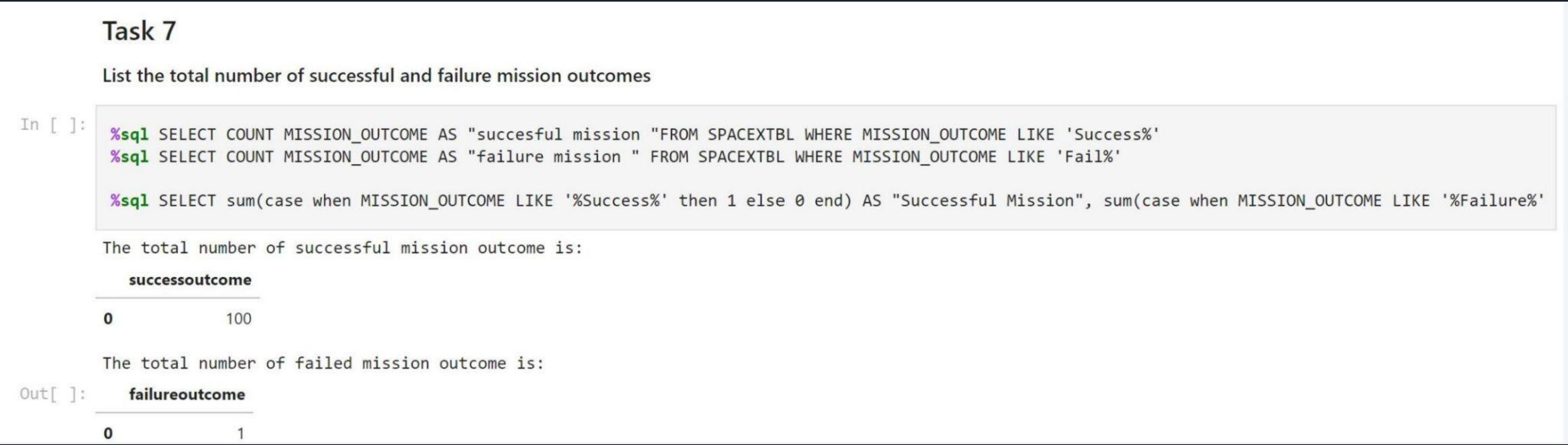
```
In [ ]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000
```

Out[ ]:

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

Total Number of Successful and Failure Mission Outcomes

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



Task 7

List the total number of successful and failure mission outcomes

In [ ]:

```
%sql SELECT COUNT MISSION_OUTCOME AS "succesful mission "FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%'
%sql SELECT COUNT MISSION_OUTCOME AS "failure mission " FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Fail%'

%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", sum(case when MISSION_OUTCOME LIKE '%Failure%'
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

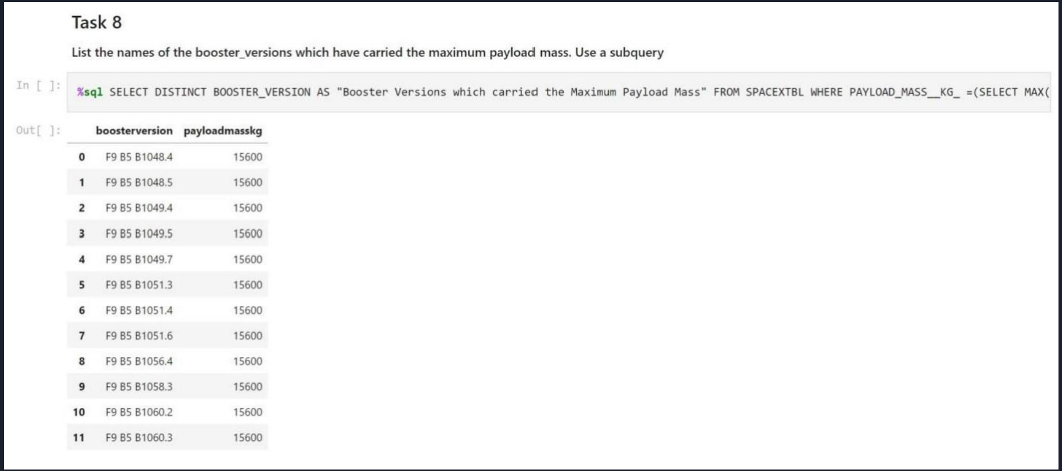
```
The total number of failed mission outcome is:
```

Out[ ]:

	failureoutcome
0	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.



Task 8

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

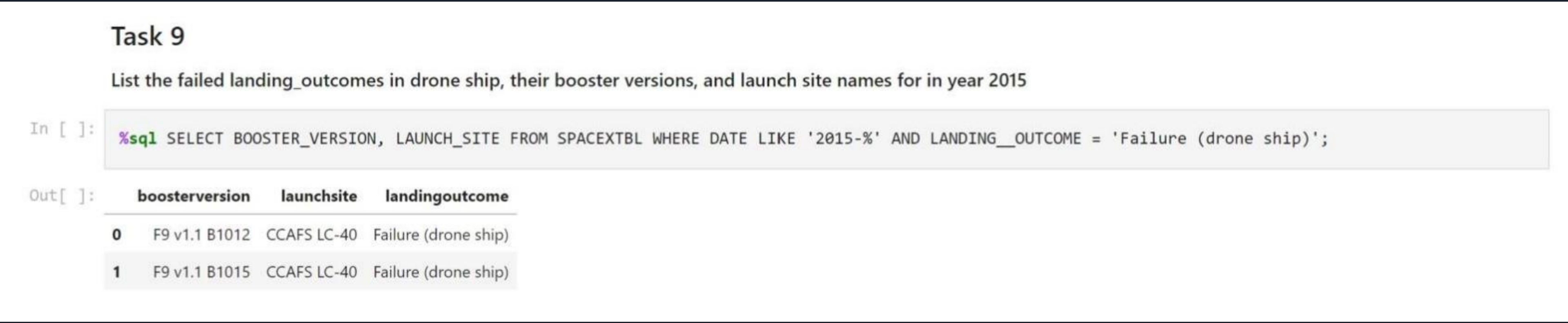
```
In [ ]: %sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(
```

Out[ ]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

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



Task 9

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

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

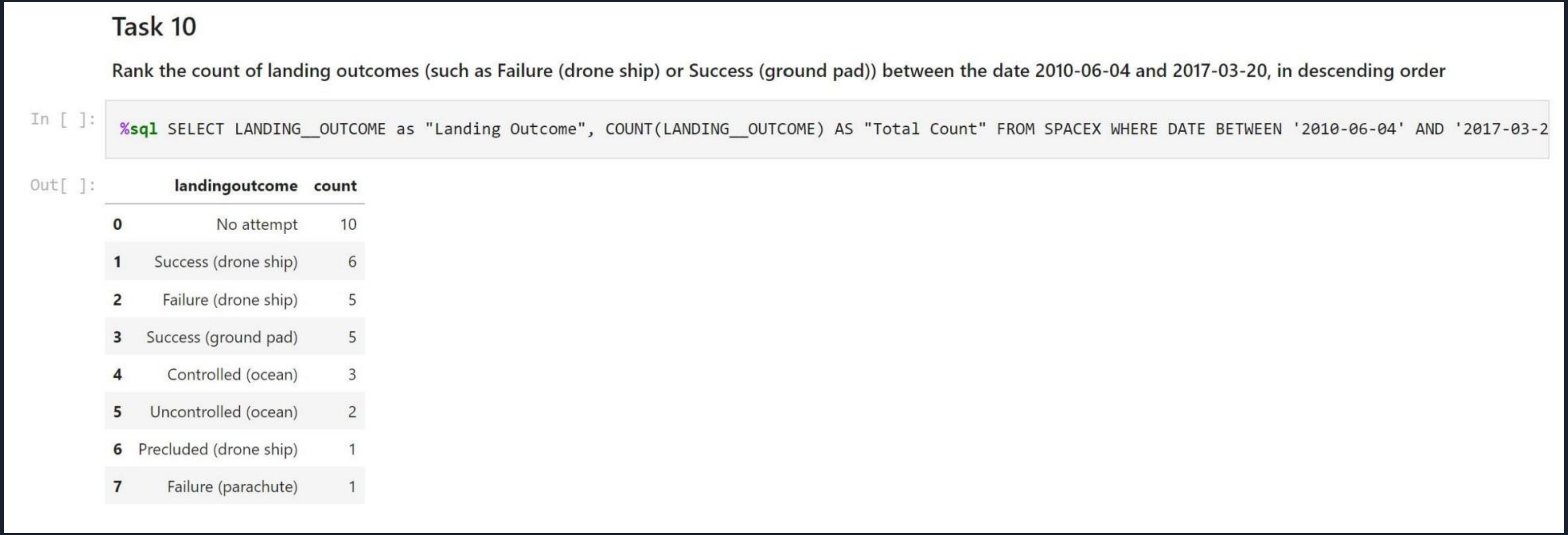
Out[ ]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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 the grouped landing outcome in descending order.



Task 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

In [ ]:

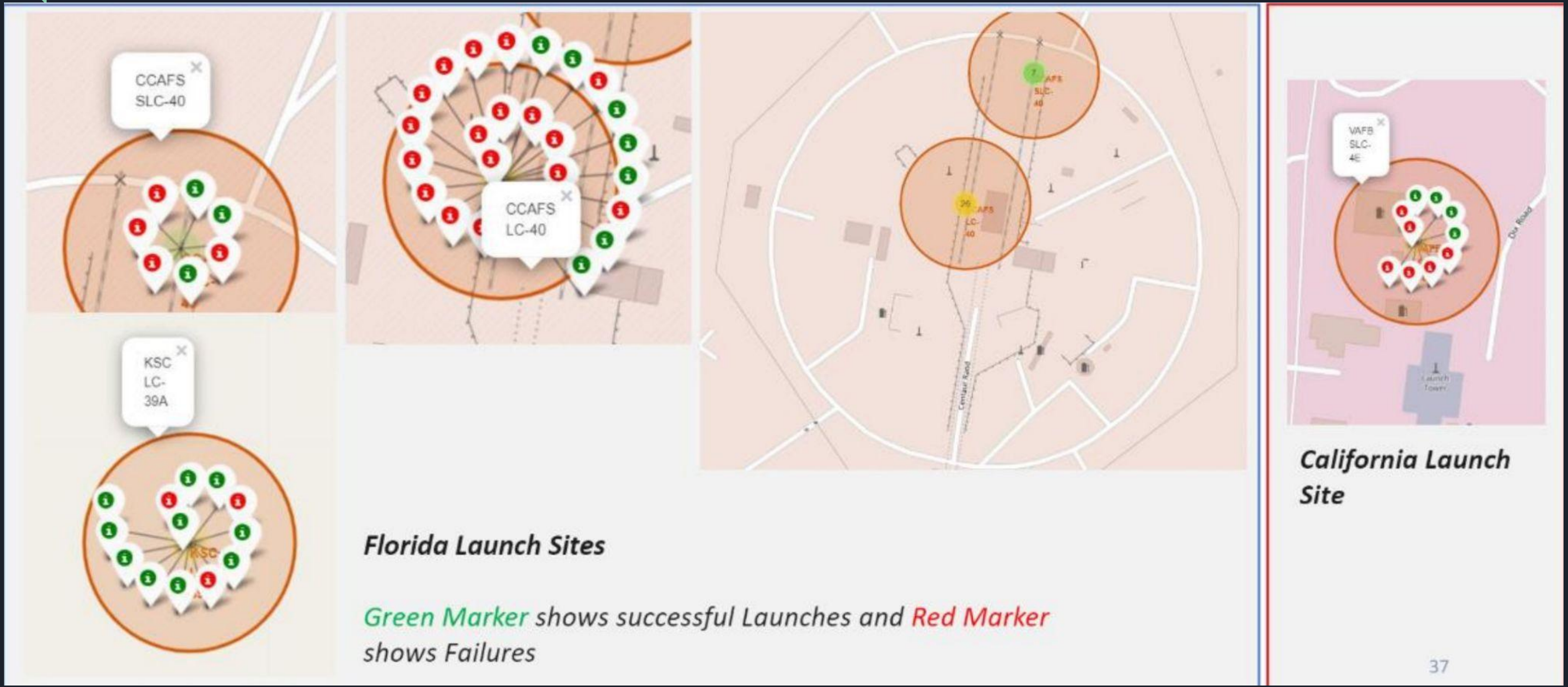
```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX WHERE DATE BETWEEN '2010-06-04' AND '2017-03-2
```

Out[ ]:

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Section 3
Launch Sites Proximities Analysis

Color Labelled Launch Sites



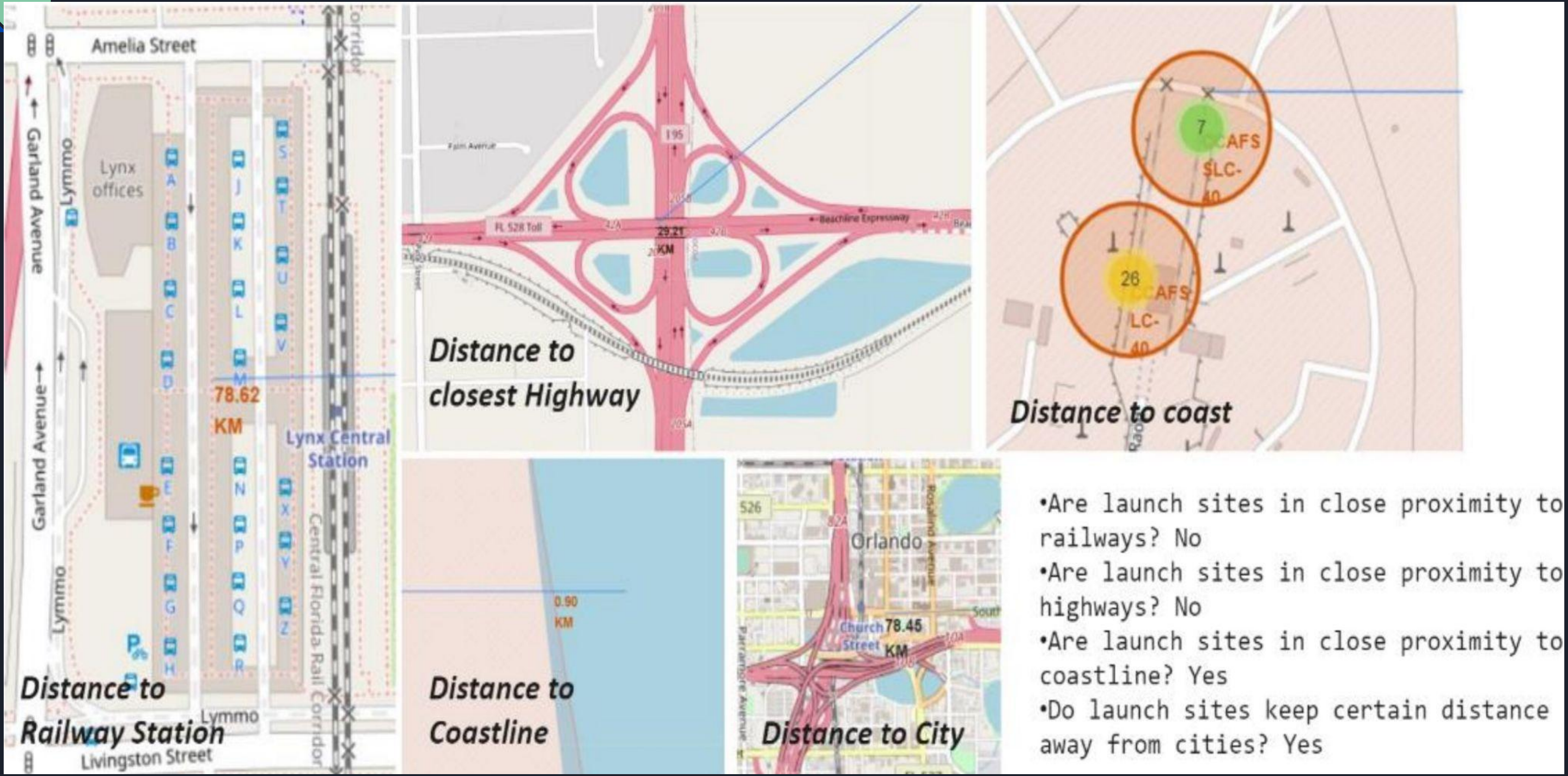
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

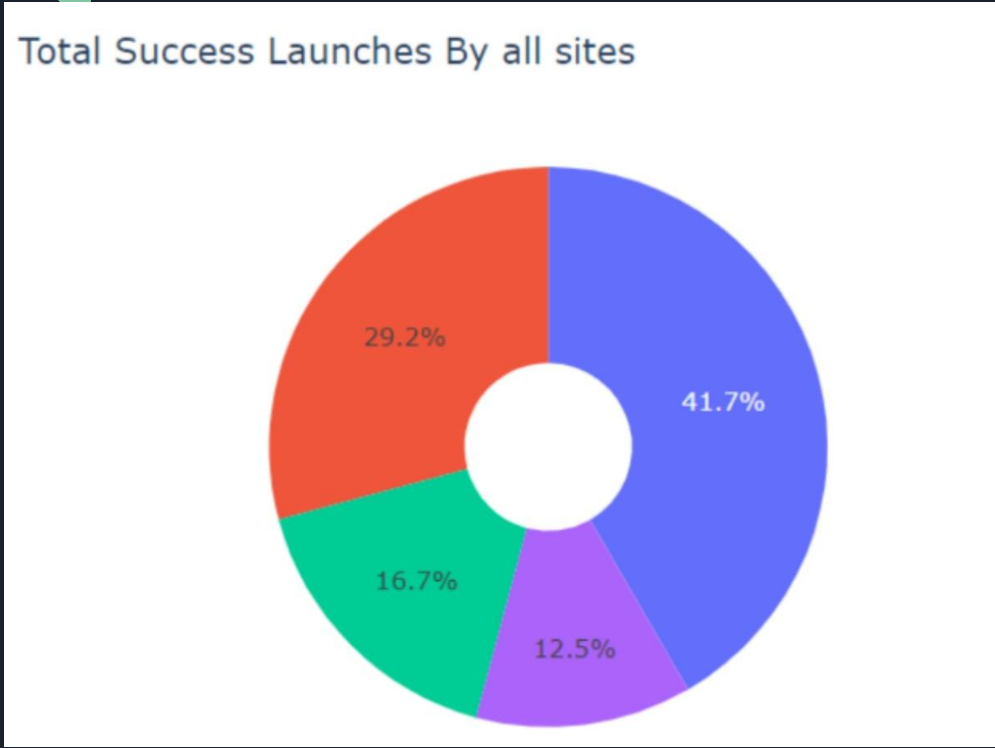
Launch Sites distance to Landmarks



•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

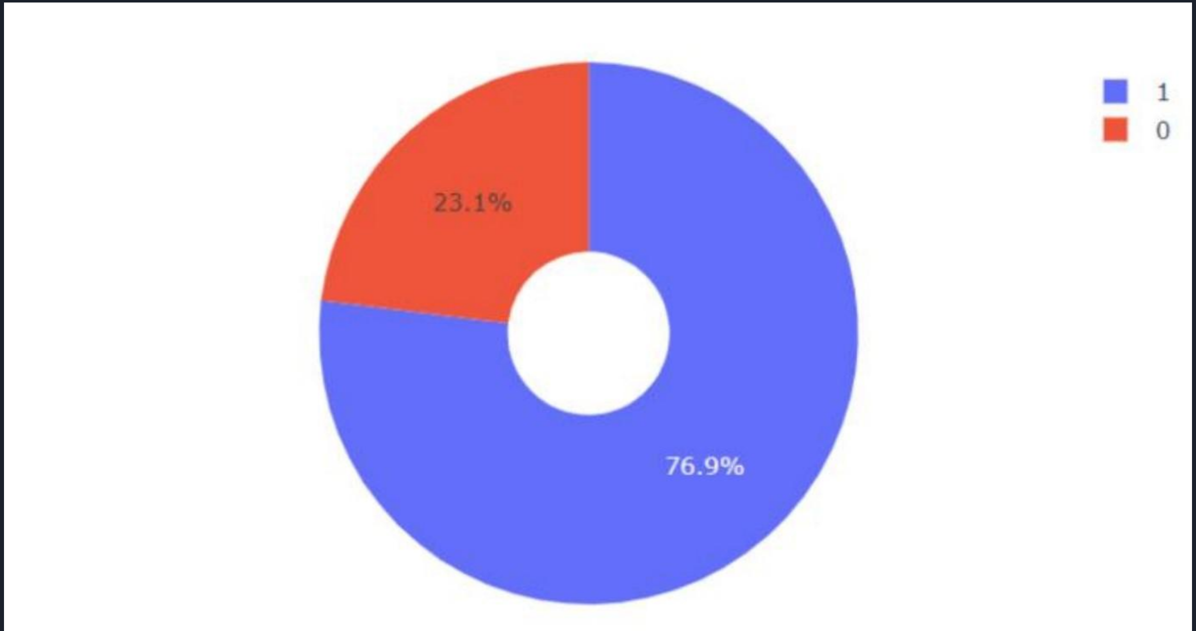
Section 4
Build a Dashboard with Plotly Dash

Plotly Pie Chart showcasing success by Launch



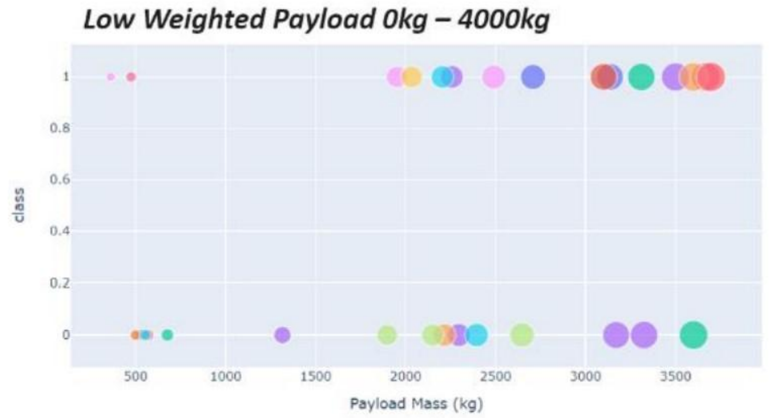
We can see that KSC LC-39A had the most successful launches from all the sites

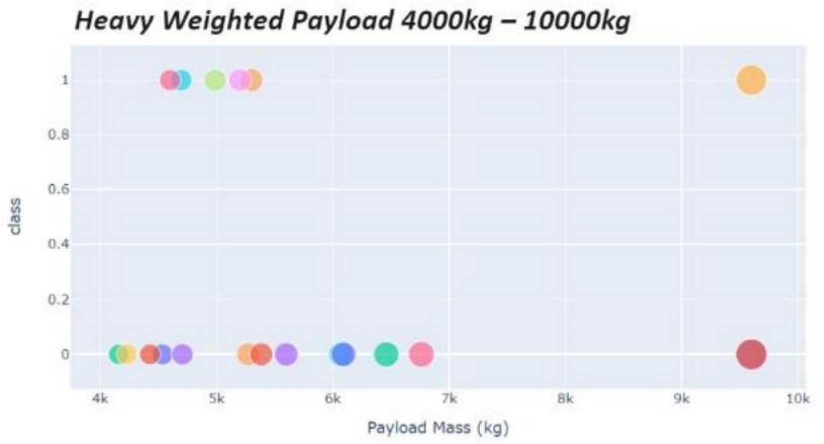
Plotly Pie Chart Showcasing Launch Site with highest launch success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Scatter plot of Payload vs Launch Outcome for all sites, with varying payload selected in the range slider



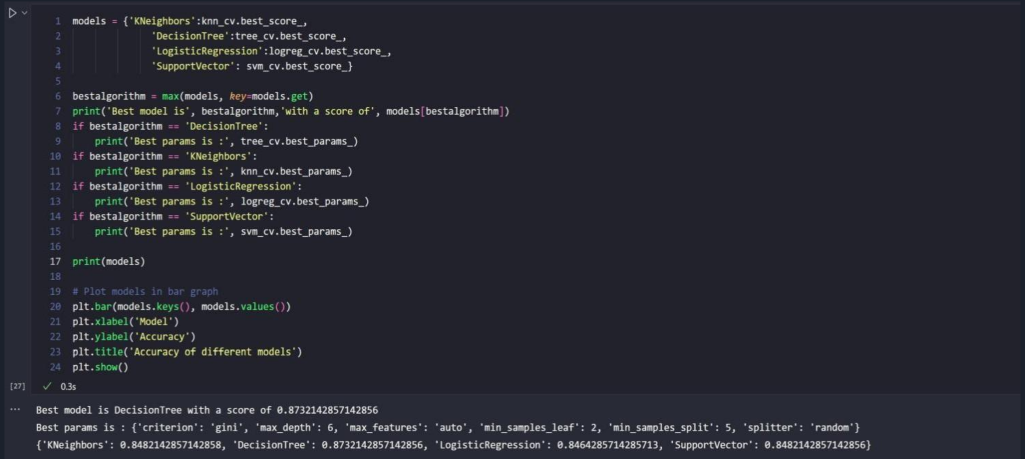


We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5
Predictive Analysis

Classification Accuracy

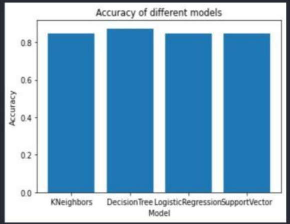
The decision tree classifier is the model with the highest classification accuracy



```python
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_)

print(models)

# Plot models in bar graph
plt.bar(models.keys(), models.values())
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.title('Accuracy of different models')
plt.show()
```

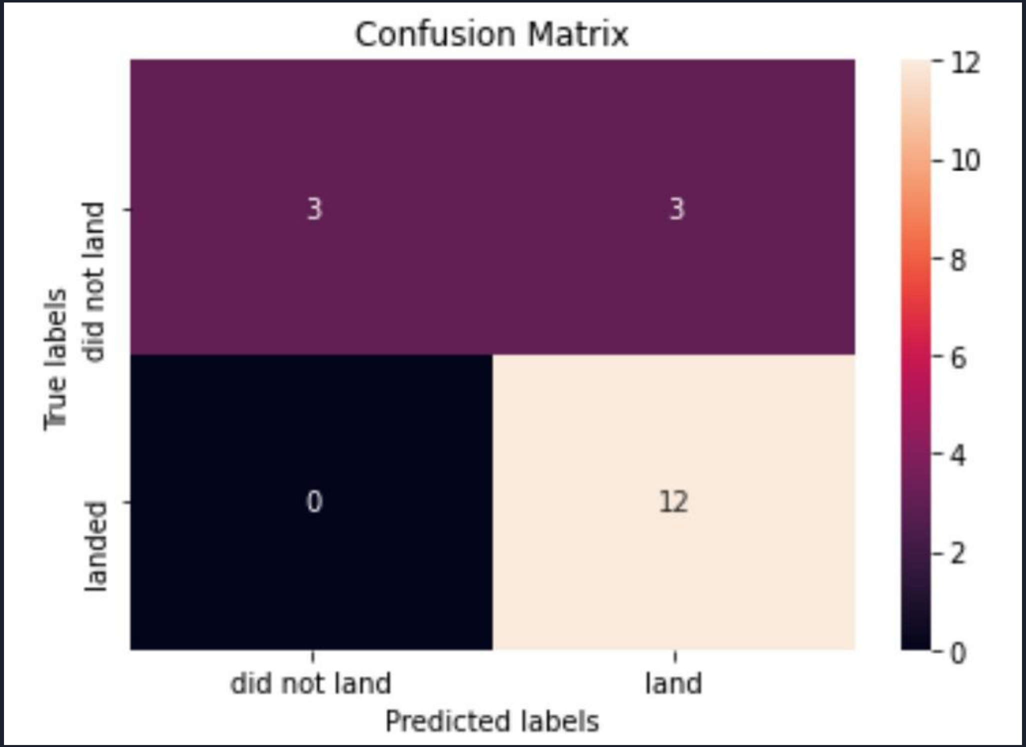
[27] ✓ 0.3s

```
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'}
{'KNeighbors': 0.8482142857142858, 'DecisionTree': 0.8732142857142856, 'LogisticRegression': 0.8464285714285713, 'SupportVector': 0.8482142857142856}
```



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

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 Decision tree classifier is the best machine learning algorithm for this task.