IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

- **Summary of methodologies**
  - This research project developed a model to **predict** whether **SPACEX will successfully land** the **first stage** of their rockets (i.e. the bottom part) in each launch, as this can significantly reduce costs and the prices they charge. In order to develop such a model, the following steps were undertaken:
    - **Data Collection** through the SPACEX Public API and Web Scraping techniques.
    - **Data Wrangling** in order to prepare the data for modelling.
    - **Exploratory Data Analysis (EDA)** through SQL queries and data visualization.
    - **Interactive Visualizations** with maps and a dashboard.
    - **Prediction** through Machine Learning classification models.

Executive Summary

- **Summary of results**
  - EDA:
    - The most important predictor of landing success is Flight Number, as the success rate dramatically increased over the years, from 0% in 2010-2013 to 33% in 2014-2015, and finally to between 61% and 90% in 2016-2020.
    - Payload Mass also seems relevant, but the relationship is not linear. A manually built decision tree shows launches with lighter payloads performed significantly better in early years (2010-2015), while flights with heavier payloads performed better in later years (2016-2020). This could mean a change in technology or processes that allowed the company to handle heavier payloads better and more often.
    - The role of Orbit Types and Booster Versions for landing success is not clear.

Executive Summary

- **Summary of results**
    - Interactive Visualizations:
        - All launch sites in the southern part of the country (nearer to the equator), 3 of them in Florida and one in California. This can be explained by the fact that, the nearer you are to the equator, the more you can take optimum advantage of the earth's rotational speed to aid your launches.
        - All of the launch sites are near the coast and relatively isolated from highways, railways and cities, probably due to safety concerns but also regulations about noise and pollution.
        - The launch site with highest number of successes was KSC-LC-39-A, with 41,7% of the total instances.

Executive Summary

- **Summary of results**
  - Predictive Analysis:
    - All models performed equally on the testing data (83,3% accuracy), but decision tree slightly outperformed other models on the training data (87,5% accuracy).
    - The decision tree model returned to false negatives, but it did have a problem with returning false positives, so there's still room for improvement.

Introduction

- **Project background and context**
  - Space X advertises Falcon 9 rocket launches on its website at a cost of 62 million dollars, while other providers charge upward to 165 million dollars each. Much of the savings is because Space X can reuse the first stage (the bottom part of the rocket), provided it is landed successfully.
  - If we can determine if the first stage will land, we can determine the approximate cost of that launch. This information can be used if an alternate company wants to bid against space X for a rocket launch.
- **Problems we want to find answers to**:
  - Can we predict, based on features of the launch sites and rockets, whether the first stage will be landed successfully?
  - What is the most accurate model for this prediction? Is this accuracy satisfactory?

Section 1

Methodology

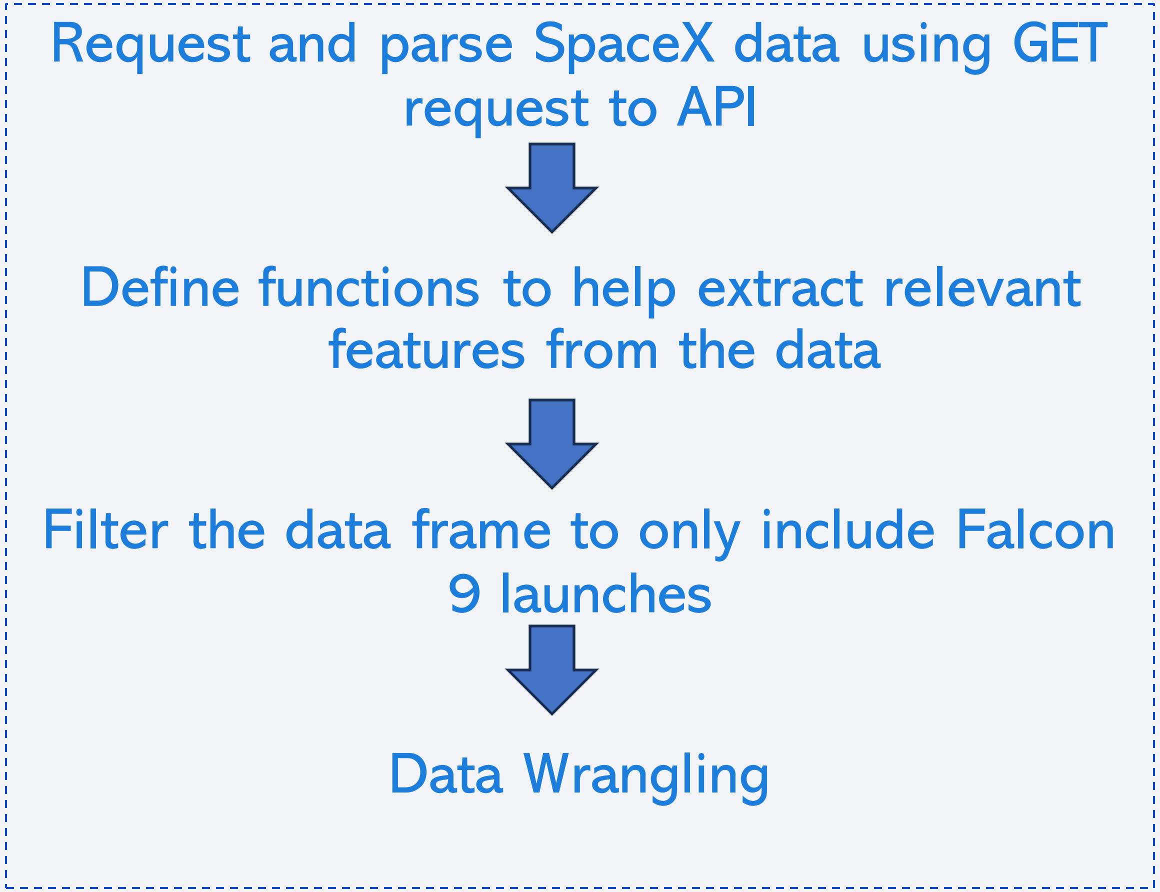
Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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 – SpaceX API

- We used the get request to the SpaceX API to collect the data, then selected the features and did some basic data wrangling and formatting such as replacing missing values.
- The corresponding notebook can be seen here: https://github.com/Rafael-Leite/SPACEX-Machine-Learning-Project/blob/main/01-Data-Collection-API.ipynb



Request and parse SpaceX data using GET request to API

Define functions to help extract relevant features from the data

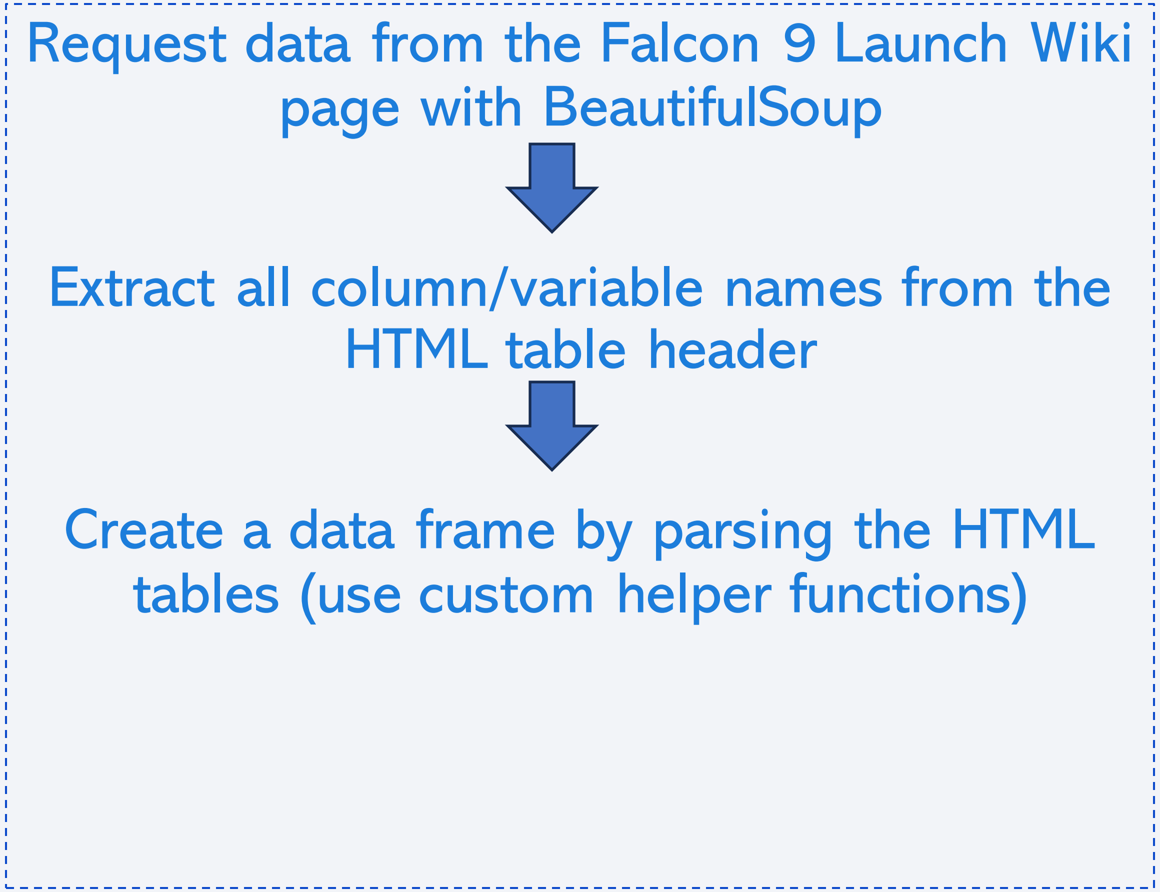
Filter the data frame to only include Falcon 9 launches

Data Wrangling

Data Collection – Scraping

- We used BeautifulSoup library to web scrape the wiki page for Falcon 9 launches, and parsed the data turn it into a data frame with the help of some custom helper functions.

- The notebook can be seen here : https://github.com/Rafael-Leite/SPACEX-Machine-Learning-Project/blob/main/02-Web-Scraping.ipynb



Request data from the Falcon 9 Launch Wiki page with BeautifulSoup

Extract all column/variable names from the HTML table header

Create a data frame by parsing the HTML tables (use custom helper functions)

Data Wrangling

- Exploratory data analysis was performed to understand the data better and determine the outcome variable to be predicted.
- Calculations were made to learn the number of launches at each site, as well as the number and occurrence of each orbits in the missions.
- A dummy variable indicating the success/failure of the landing of the first stage was created from a multiclass categorical variable, and it was chosen as the dependent variable.
- The notebook can be seen here: https://github.com/Rafael-Leite/SPACEX-Machine-Learning-Project/blob/main/03-Data-Wrangling.ipynb

EDA with SQL

- With sqlite module, SQL queries were written within the Jupyter notebook to gather more familiarity with the data, for instance:
    - Names of unique launch sites.
    - Average payload mass carried by booster version F9 v1.1
    - The total number of successful and failure mission outcomes
    - The names of the booster_versions which have carried the maximum payload mass.
- The notebook can be seen here: https://github.com/Rafael-Leite/SPACEX-Machine-Learning-Project/blob/main/04-EDA-sql-sqllite.ipynb

EDA with Data Visualization

- Line plots, scatter plots and bar plots were produced in order to uncover the relationships between variables and their relevance to the outcome of the landing mission, in particular: Flight Number, Payload Mass, Launch Site and Orbit Type.
- A final step was to further prepare the data for machine learning, by applying one-hot encoding to turn categorical variables into dummies.
- The notebook can be seen here: https://github.com/Rafael-Leite/SPACEX-Machine-Learning-Project/blob/main/05-EDA-Dataviz-seaborn.ipynb

Build an Interactive Map with Folium

- Interactive maps were built with Folium, including markers and circles for identifying launch sites and their respective numbers of successes and failures in landing attempts.
- Colored lines were used to indicate the distance from launch sites to nearest cities, highways, railways and coastline.
- The notebook can be seen here: https://github.com/Rafael-Leite/SPACEX-Machine-Learning-Project/blob/main/06-Interactive-Maps-folium.ipynb

Build a Dashboard with Plotly Dash

- Pie charts were added to the dashboard in order to easily visualize the success rates across different launch sites, as well as their participation in the total success landings of the company.
- Scatter plots were also added, in order to visualize the landing outcomes across different payload masses, both overall and across each launch site.
- The notebook can be seen here: https://github.com/Rafael-Leite/SPACEX-Machine-Learning-Project/blob/main/07-Interactive-Dashboard-dash.py

Predictive Analysis (Classification)

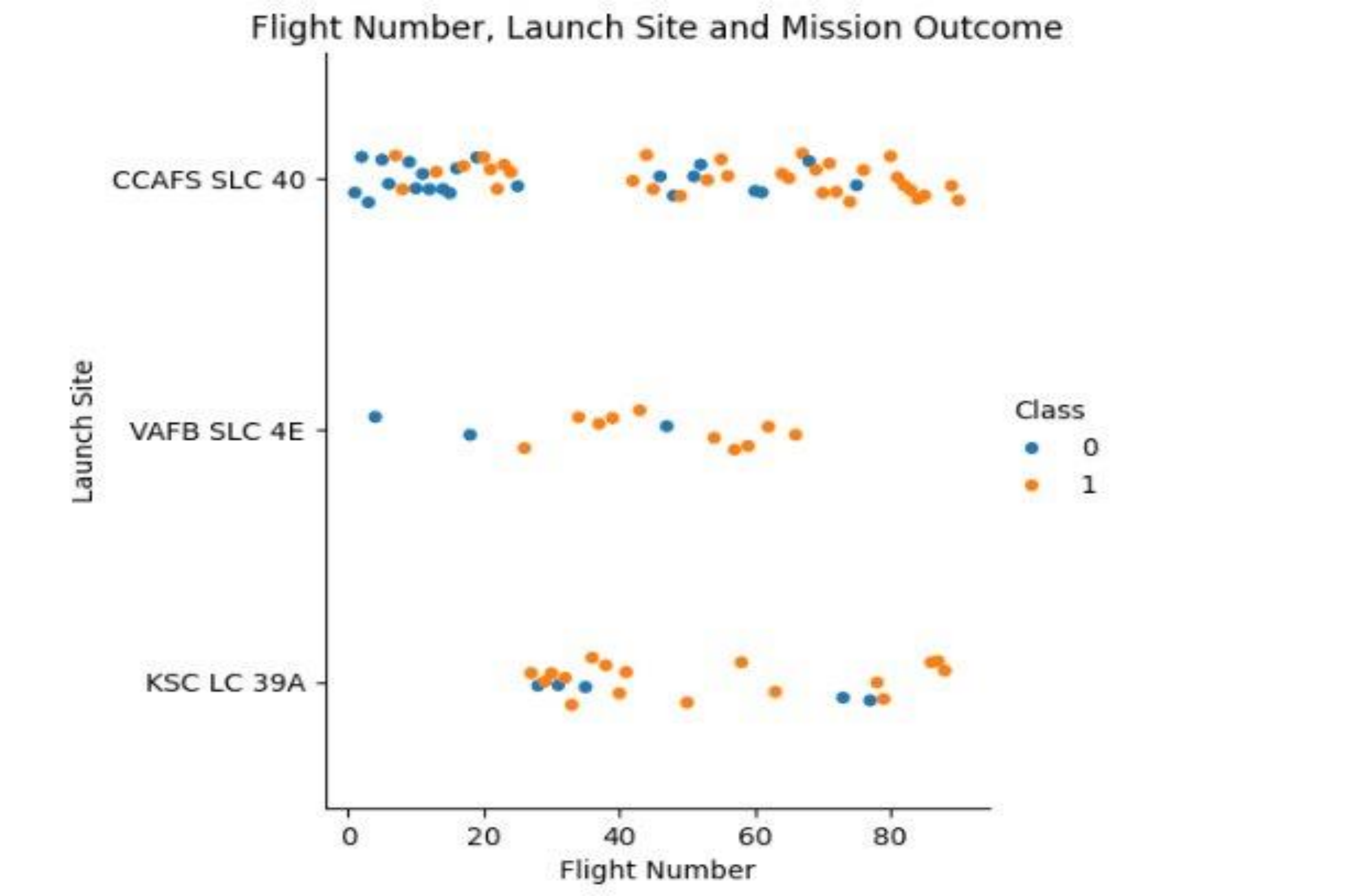
- The data was standardized and split into training and testing samples, with a 20% test size.
- Models were trained for Logistic Regression, Support Vector Machines, Decision Trees and K-Nearest Neighbors, all using sklearn.
- Each model's hyperparameters were tuned using GridSearchCV(), with a cross-validation of 10 folds.
- The R-squared measures for both training and test data were compared across models to find the best one.
- The notebook can be seen here: https://github.com/Rafael-Leite/SPACEX-Machine-Learning-Project/blob/main/08-Machine-Learning-Prediction.ipynb

Section 2

Insights drawn from EDA

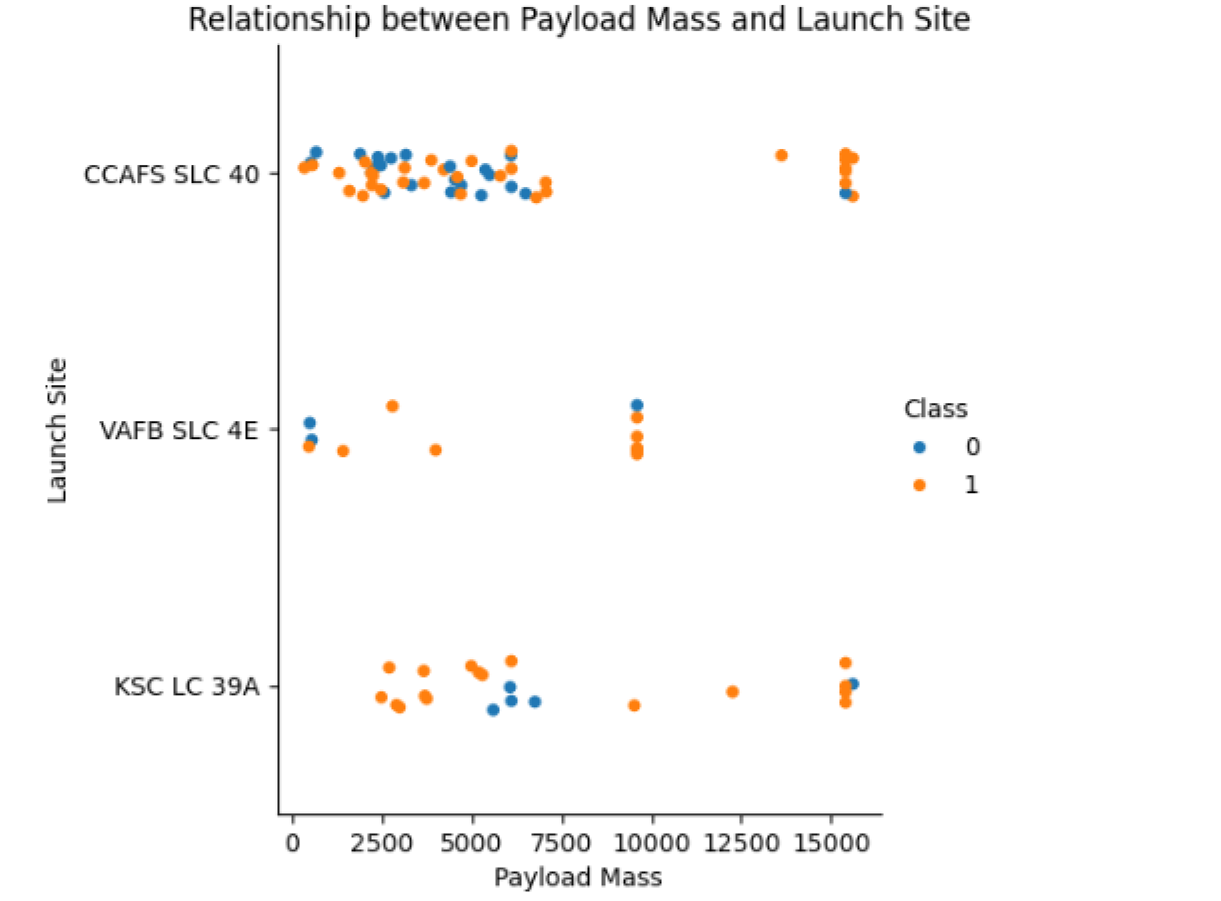
Flight Number vs. Launch Site

- A scatter plot shows how almost all of the 20 first launches were made in launch site CCAFS, but as success rates started to increase with time, operations started to expand to other sites:



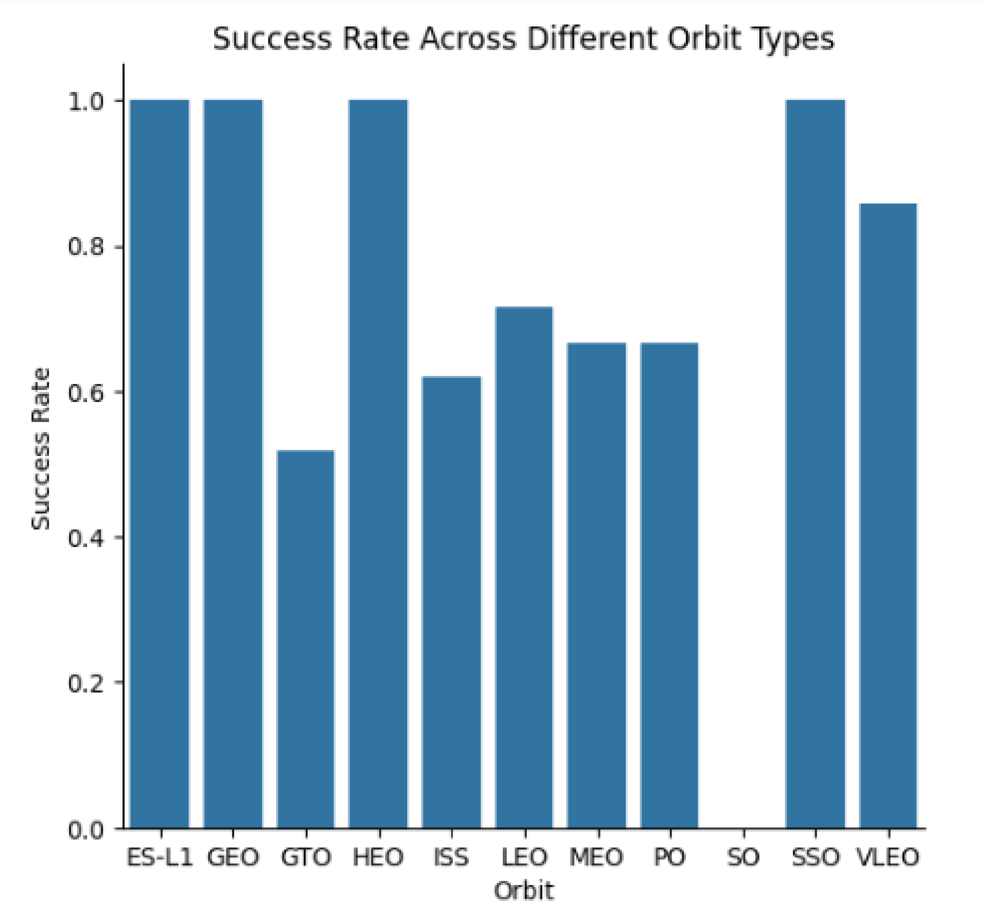
Payload vs. Launch Site

- A scatter plot shows the Payload handled by each launch site. VAFB did not handle anything beyond 10.000 kg:



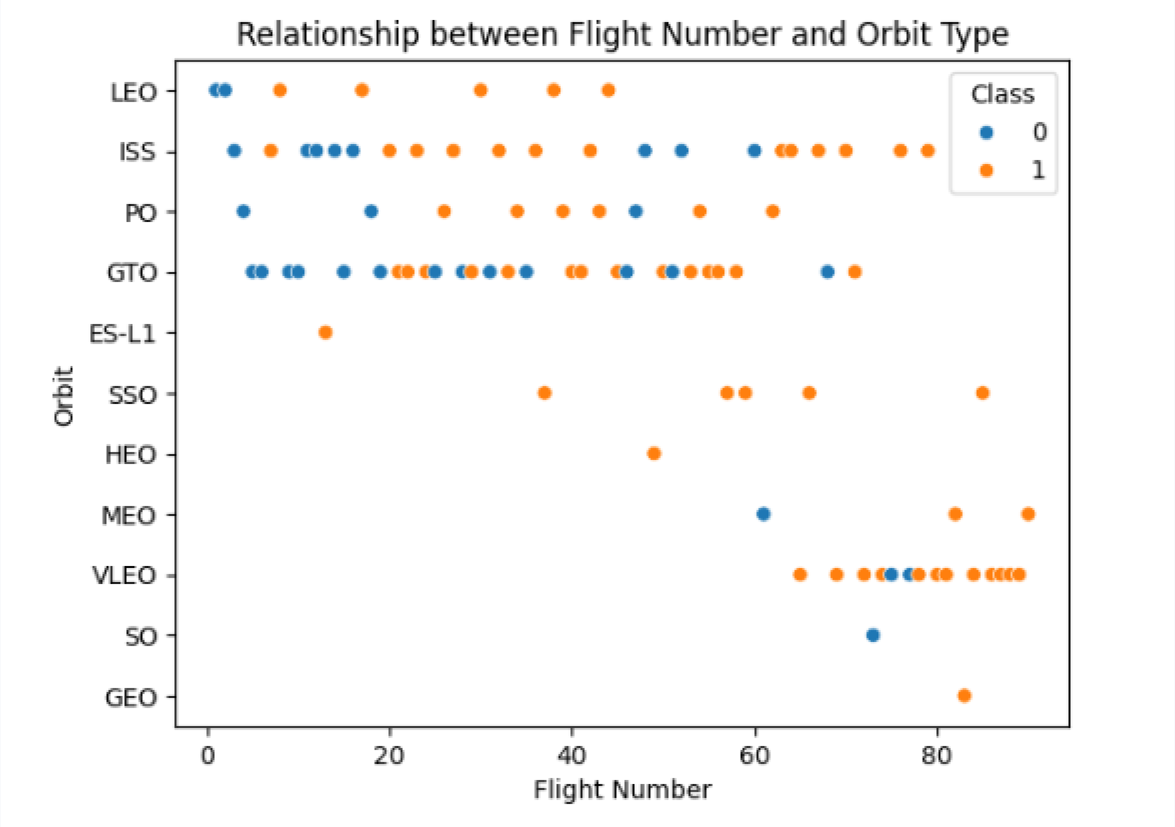
Success Rate vs. Orbit Type

- A bar chart shows the success rates for each orbit type, with 100% success for ES-L1, GEO, HEO, SSO, and 0% for SO. All those orbits, though, had only 1 attempt each (with the exception of SSO, with 5 attempts), so these results are not conclusive enough. GTO had the most attempts (27), with a success rate of 51,8%:



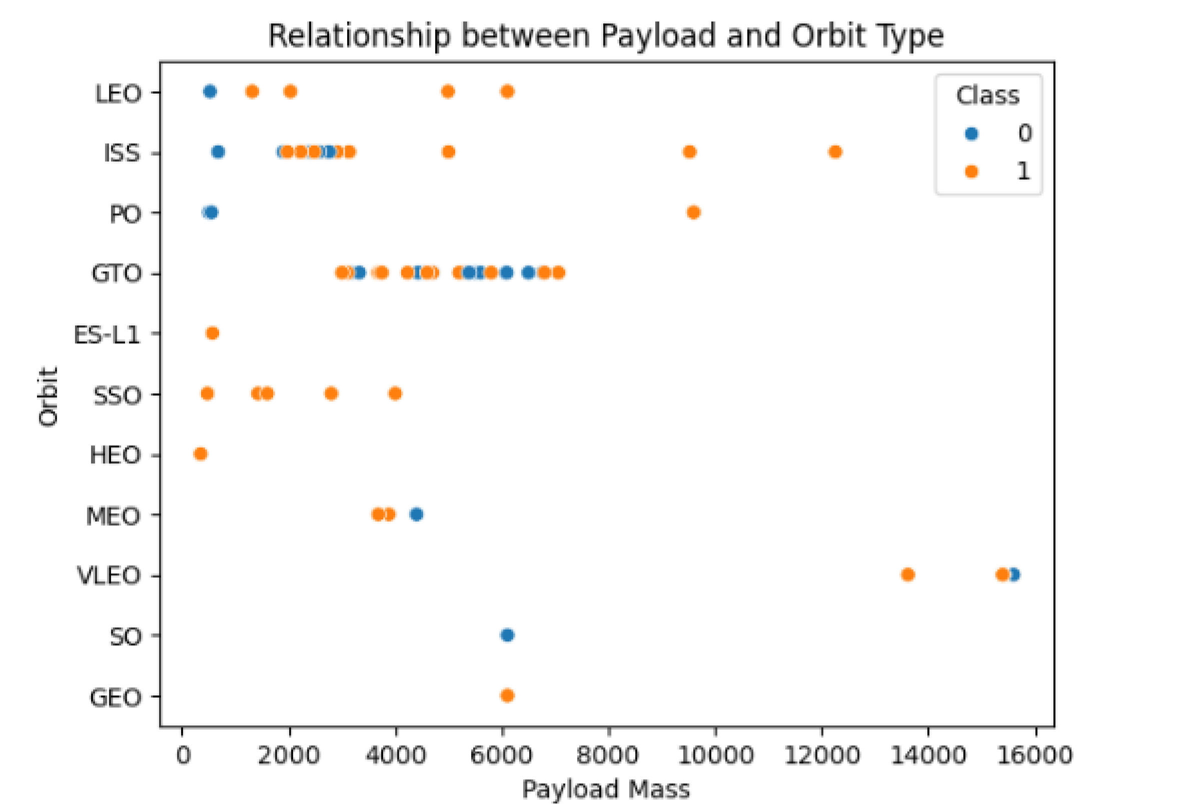
Flight Number vs. Orbit Type

- A scatter plot shows how most early flights were clustered around LEO, ISS, PO and GTO orbits, later expanding to the other ones, especially VLEO (Very Low Earth Orbit). It can also be seen that in LEO, later flights are related to greater success, but the same is harder to attest in GTO:



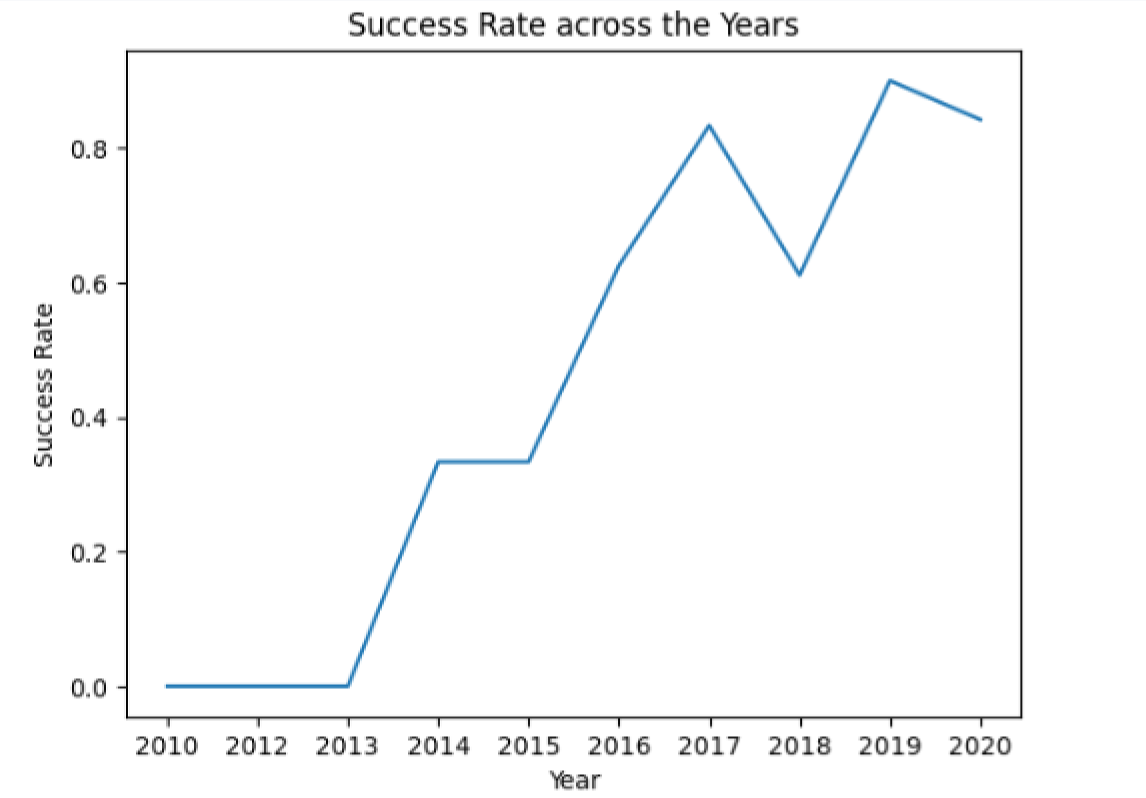
Payload vs. Orbit Type

- A scatter plot shows that with heavier payloads, success rates are greater for PO, LEO and ISS. One can also see that the heaviest payload went to VLEO (Very Low Earth Orbit):



Launch Success Yearly Trend

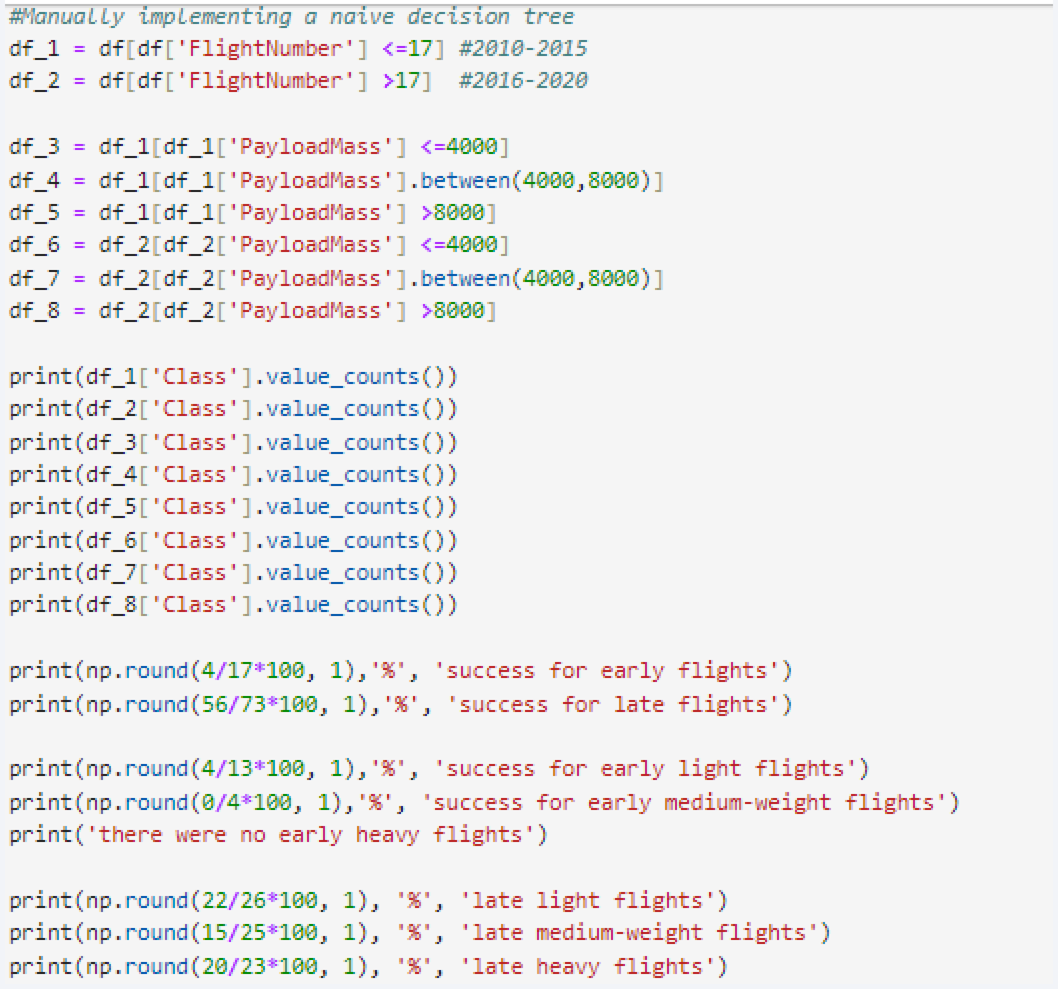
- Grouping the data by year, one can see the success rate dramatically increased over the years, from 0% in 2010-2013 to between 61% and 90% in 2016-2020:



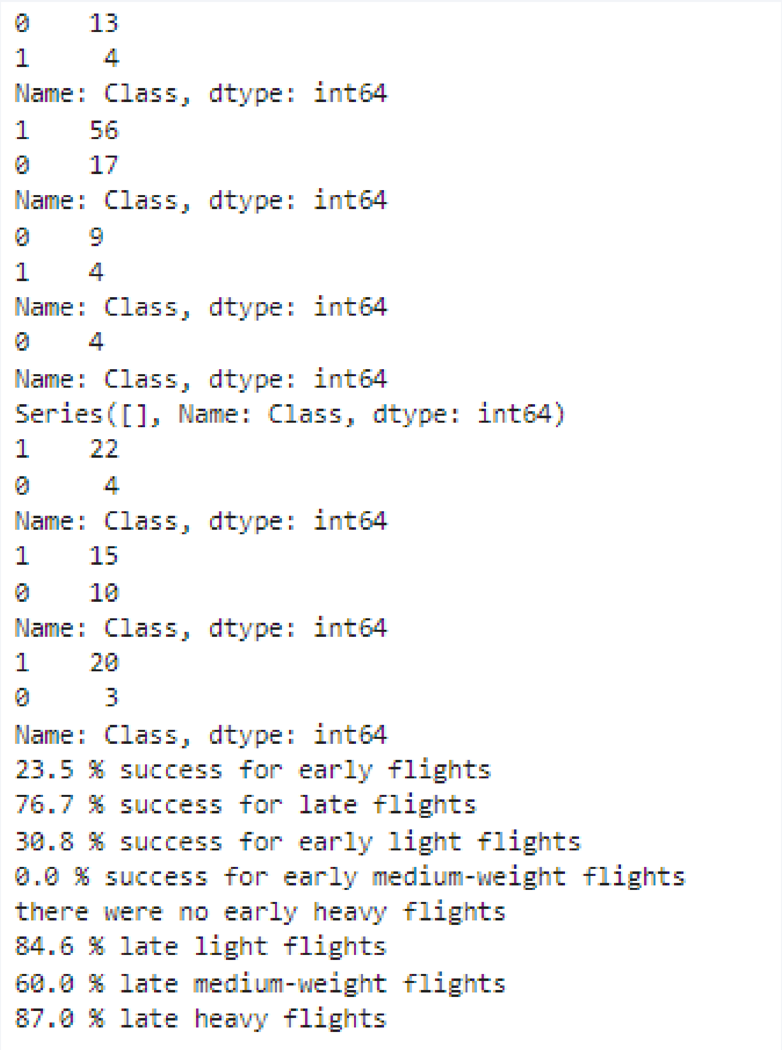
Manually built decision tree

- Since the effect of payload mass on landing success rate seems to be non-linear, a decision tree was manually built to try to get a better grasp of this effect. Both the code and the result can be seen in the next slide.
- The results show that during early years (2010-2015), lighter payloads (<4,000 kg) performed better than medium-weight loads (between 4,000 and 8,000kg), while no payloads heavier than 8,000 kg were attempted during this period.
- On the other hand, in the later years (2016-2020), heavy payloads above 8,000 kg outperformed both light and medium loads.

Manually built decision tree



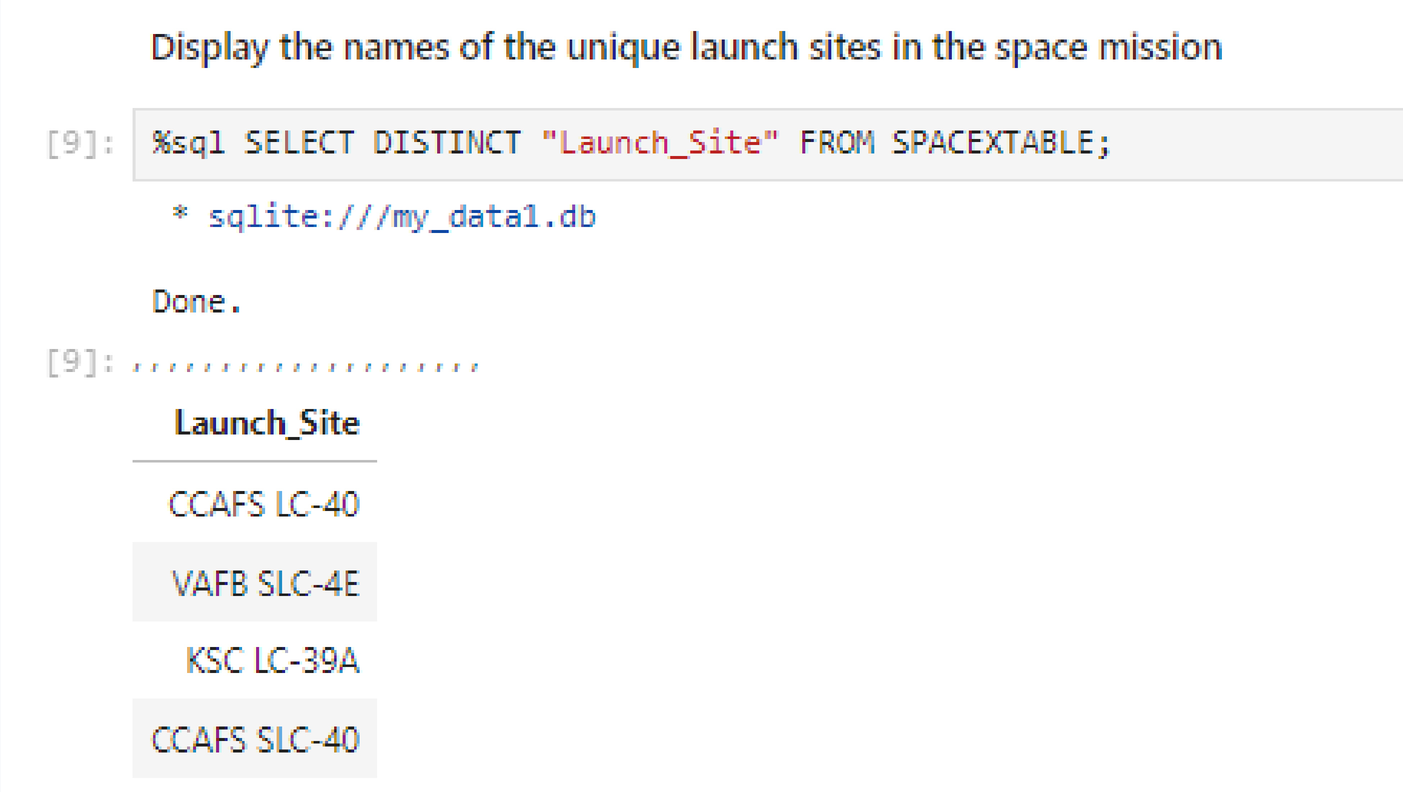
```
#Manually implementing a naive decision tree
df_1 = df[df['FlightNumber'] <=17] #2010-2015
df_2 = df[df['FlightNumber'] >17]  #2016-2020

df_3 = df_1[df_1['PayloadMass'] <=4000]
df_4 = df_1[df_1['PayloadMass'].between(4000,8000)]
df_5 = df_1[df_1['PayloadMass'] >8000]
df_6 = df_2[df_2['PayloadMass'] <=4000]
df_7 = df_2[df_2['PayloadMass'].between(4000,8000)]
df_8 = df_2[df_2['PayloadMass'] >8000]

print(df_1['Class'].value_counts())
print(df_2['Class'].value_counts())
print(df_3['Class'].value_counts())
print(df_4['Class'].value_counts())
print(df_5['Class'].value_counts())
print(df_6['Class'].value_counts())
print(df_7['Class'].value_counts())
print(df_8['Class'].value_counts())

print(np.round(4/17*100, 1),'%', 'success for early flights')
print(np.round(56/73*100, 1),'%', 'success for late flights')

print(np.round(4/13*100, 1),'%', 'success for early light flights')
print(np.round(0/4*100, 1),'%', 'success for early medium-weight flights')
print('there were no early heavy flights')

print(np.round(22/26*100, 1), '%', 'late light flights')
print(np.round(15/25*100, 1), '%', 'late medium-weight flights')
print(np.round(20/23*100, 1), '%', 'late heavy flights')
```



```
0    13
1     4
Name: Class, dtype: int64
1    56
0    17
Name: Class, dtype: int64
0    9
1    4
Name: Class, dtype: int64
0    4
Name: Class, dtype: int64
Series([], Name: Class, dtype: int64)
1    22
0     4
Name: Class, dtype: int64
1    15
0    10
Name: Class, dtype: int64
1    20
0     3
Name: Class, dtype: int64
23.5 % success for early flights
76.7 % success for late flights
30.8 % success for early light flights
0.0 % success for early medium-weight flights
there were no early heavy flights
84.6 % late light flights
60.0 % late medium-weight flights
87.0 % late heavy flights
```

All Launch Site Names

- We can use the DISTINCT key word to return unique Launch Site names:



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

```
[9]: %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;
```

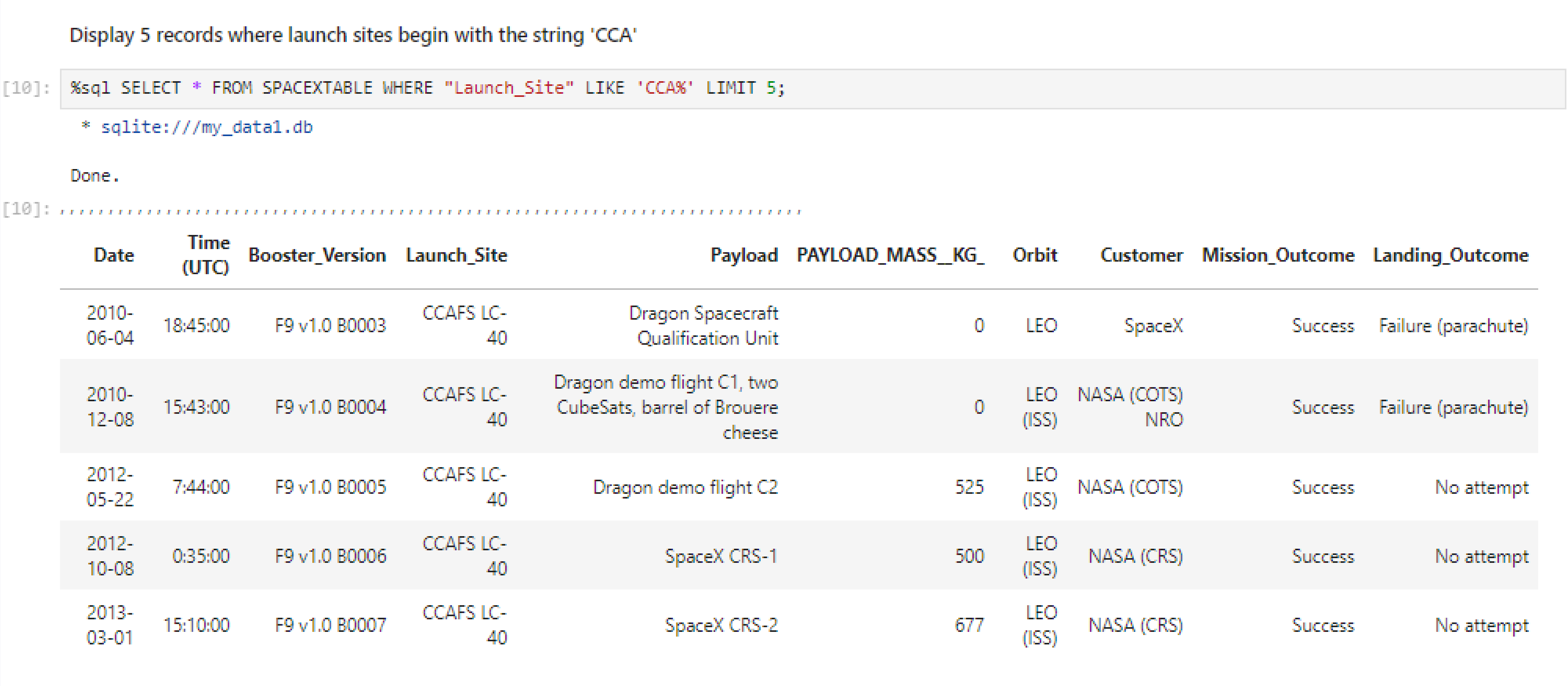
```
 * sqlite:///my_data1.db
Done.
```

[9]:

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

Launch Site Names that Begin with 'CCA'

- We can use 'LIMIT' to restrict our query to the first 5 launches of the bases that start with 'CCA':



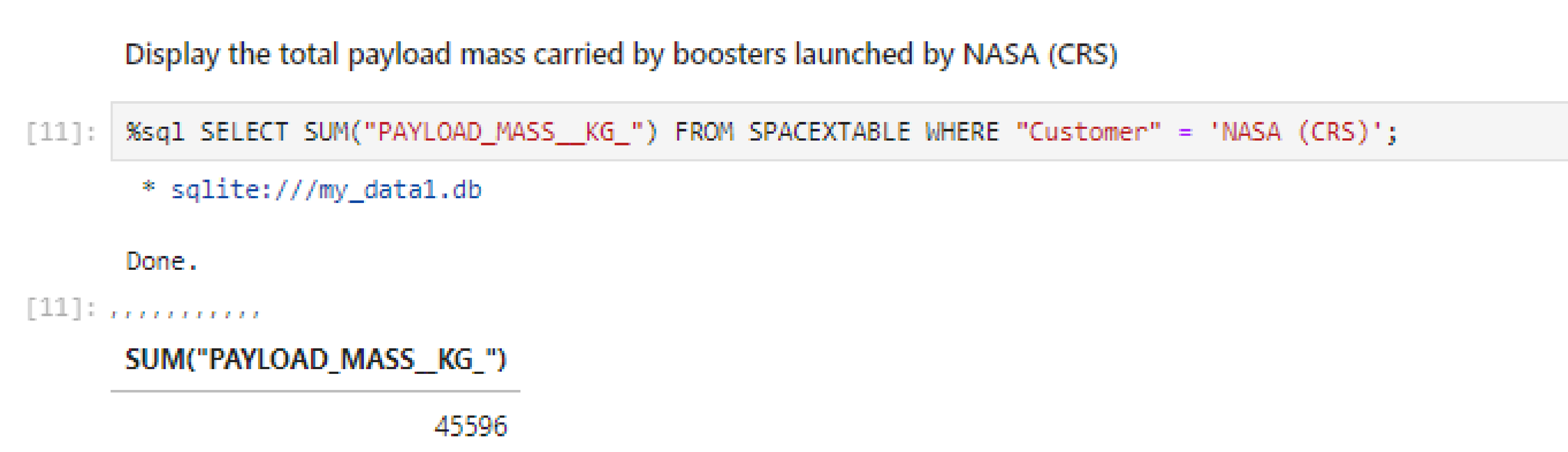
Display 5 records where launch sites begin with the string 'CCA'

```
[10]: %sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;

 * sqlite:///my_data1.db

Done.

[10]: ,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
```

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

Total Payload Mass from NASA

- We can calculate the total payload carried by boosters from NASA, by using conditions with 'WHERE' clauses:



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

```
[11]: %sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';
```

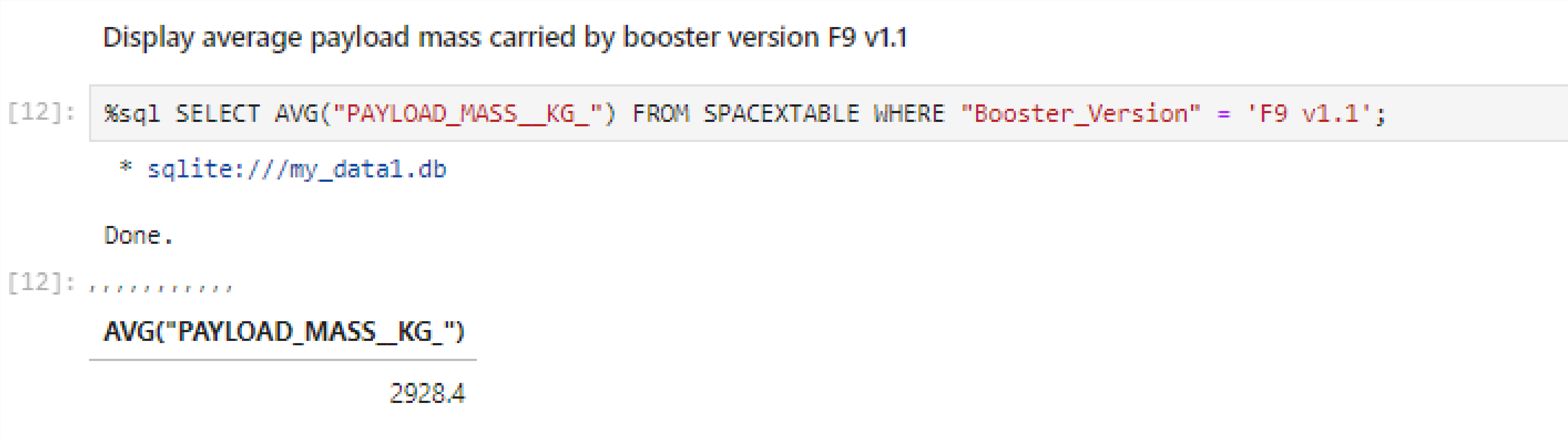
```
 * sqlite:///my_data1.db
Done.
```

[11]: ,,,,,,,,,,,

SUM("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- We can calculate the average payload mass carried by booster version F9 v1.1 using the aggregate function AVG():



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

```
[12]: %sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';
```

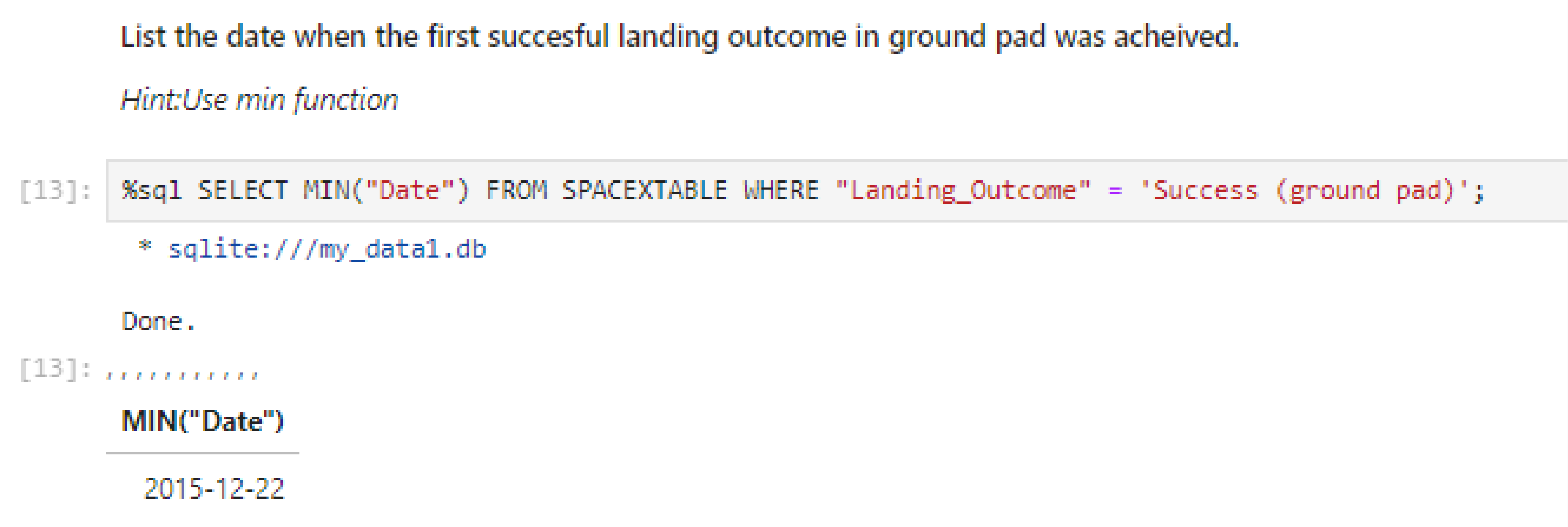
```
 * sqlite:///my_data1.db
Done.
```

[12]:

AVG("PAYLOAD_MASS__KG_")
2928.4

First Successful Ground Landing Date

- We can find the dates of the first successful landing outcome on ground pad using the MIN() function of 'Date' column:



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

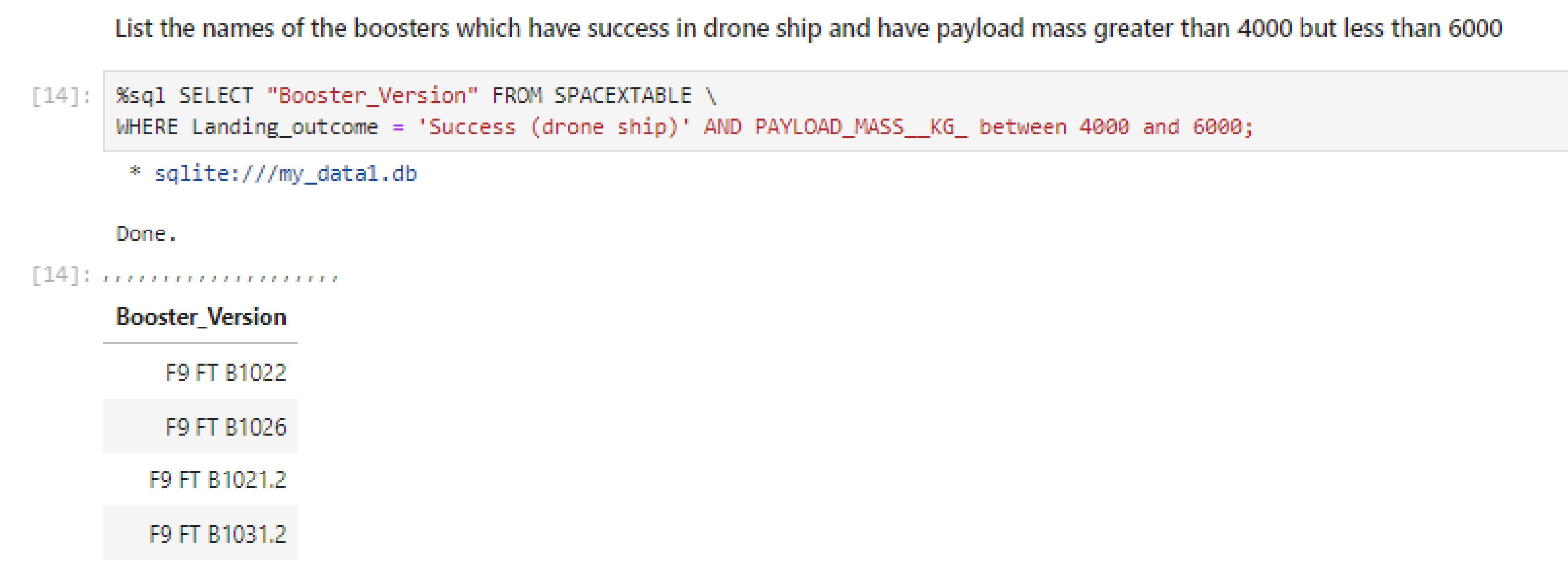
Hint:Use min function

```
[13]: %sql SELECT MIN("Date") FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';

 * sqlite:///my_data1.db
Done.

[13]: ,,,,,,,,,,,
```

MIN("Date")
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We can list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000. For that, we use 'and' to state a double condition, and 'between' to return values within the specified range:



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

```
[14]: %sql SELECT "Booster_Version" FROM SPACEXTABLE \
      WHERE Landing_outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ between 4000 and 6000;
```

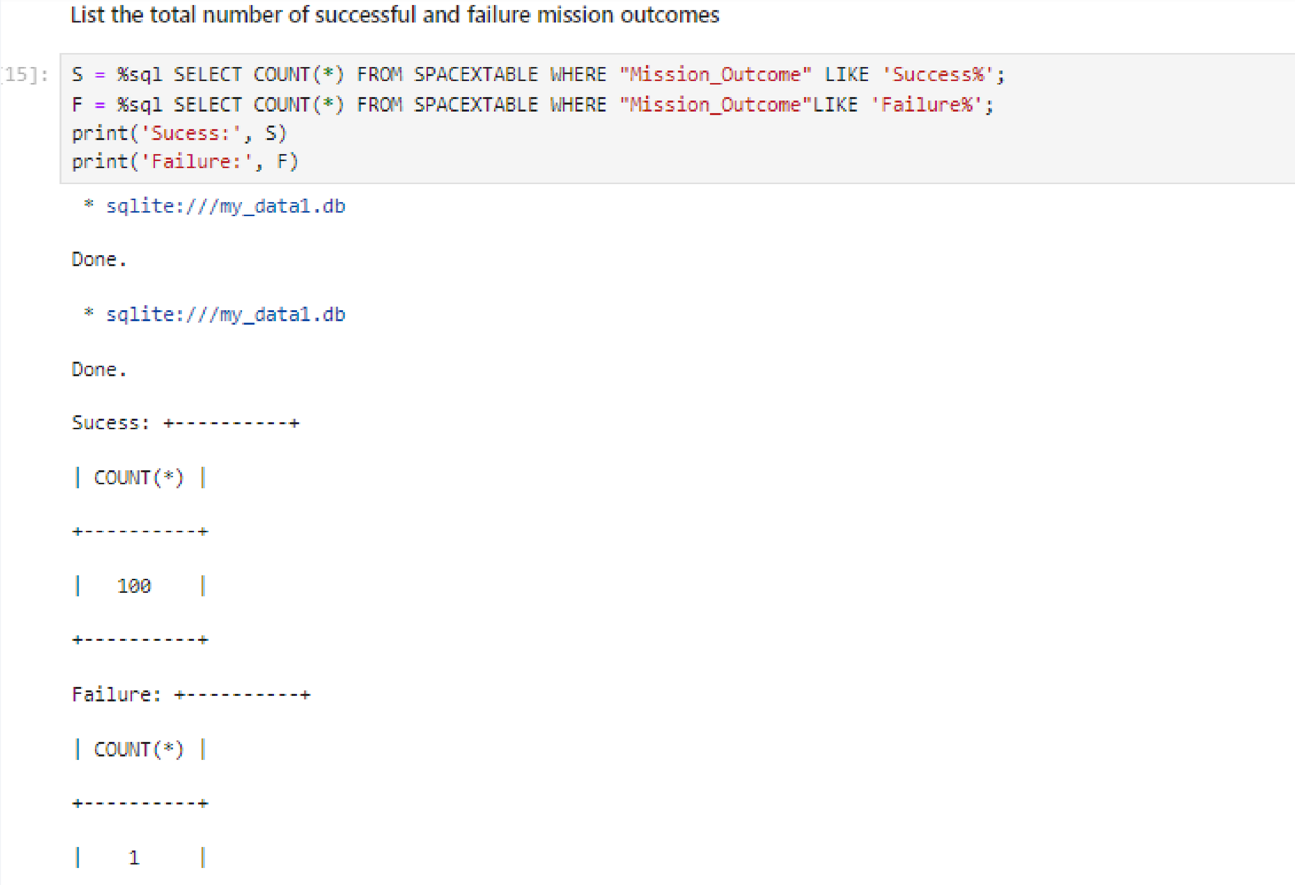
```
 * sqlite:///my_data1.db

Done.
```

[14]:

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

Total Number of Successful and Failure Mission Outcomes

- We can calculate the total number of successful and failure mission outcomes using 'COUNT', together with a condition that used 'LIKE' to return any mission outcome similar to the word 'success':



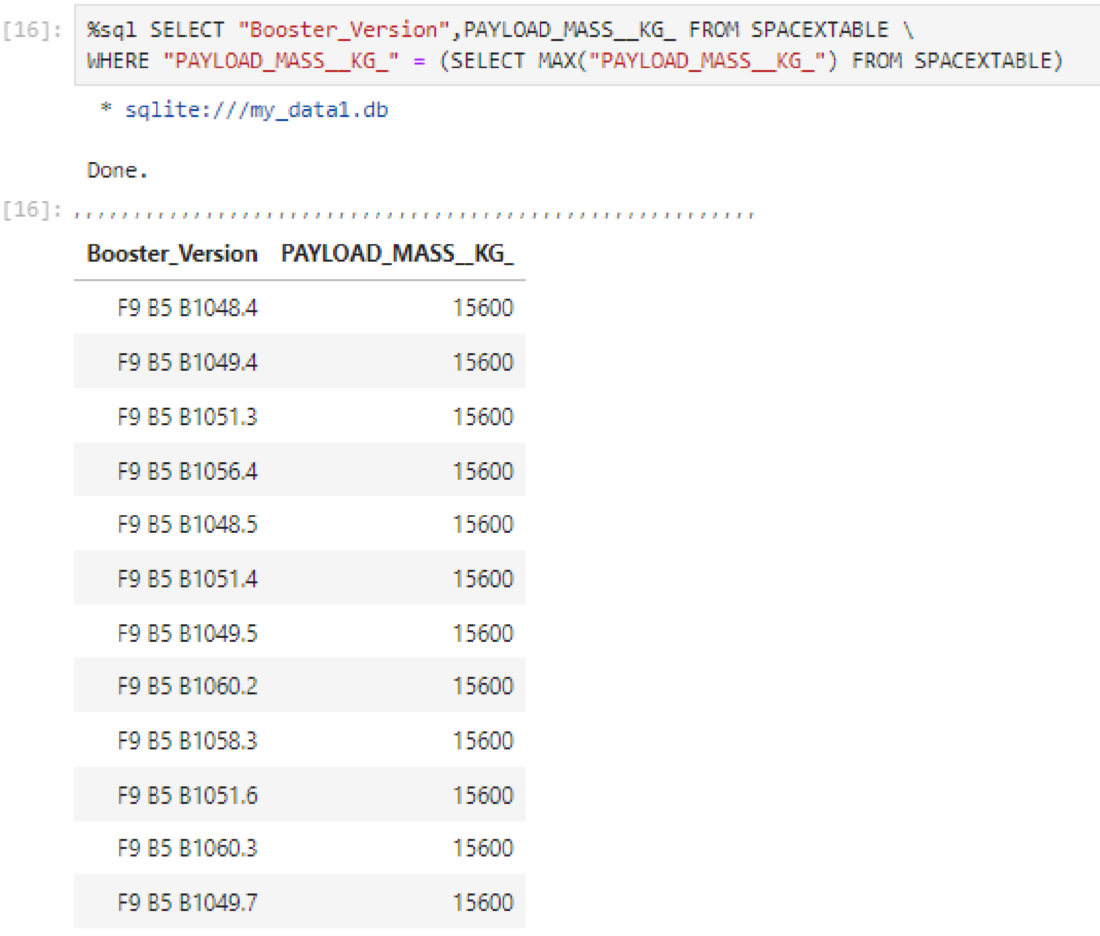
List the total number of successful and failure mission outcomes

```
15]: S = %sql SELECT COUNT(*) FROM SPACEXTABLE WHERE "Mission_Outcome" LIKE 'Success%';
     F = %sql SELECT COUNT(*) FROM SPACEXTABLE WHERE "Mission_Outcome"LIKE 'Failure%';
     print('Sucess:', S)
     print('Failure:', F)
```

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

Done.

 * sqlite:///my_data1.db

Done.

Sucess: +----------+

| COUNT(*) |

+----------+

|   100    |

+----------+

Failure: +----------+

| COUNT(*) |

+----------+

|    1     |
```

Boosters Carried Maximum Payload

- We can list the names of the boosters which have carried the maximum payload mass by using a subquery within the WHERE condition:

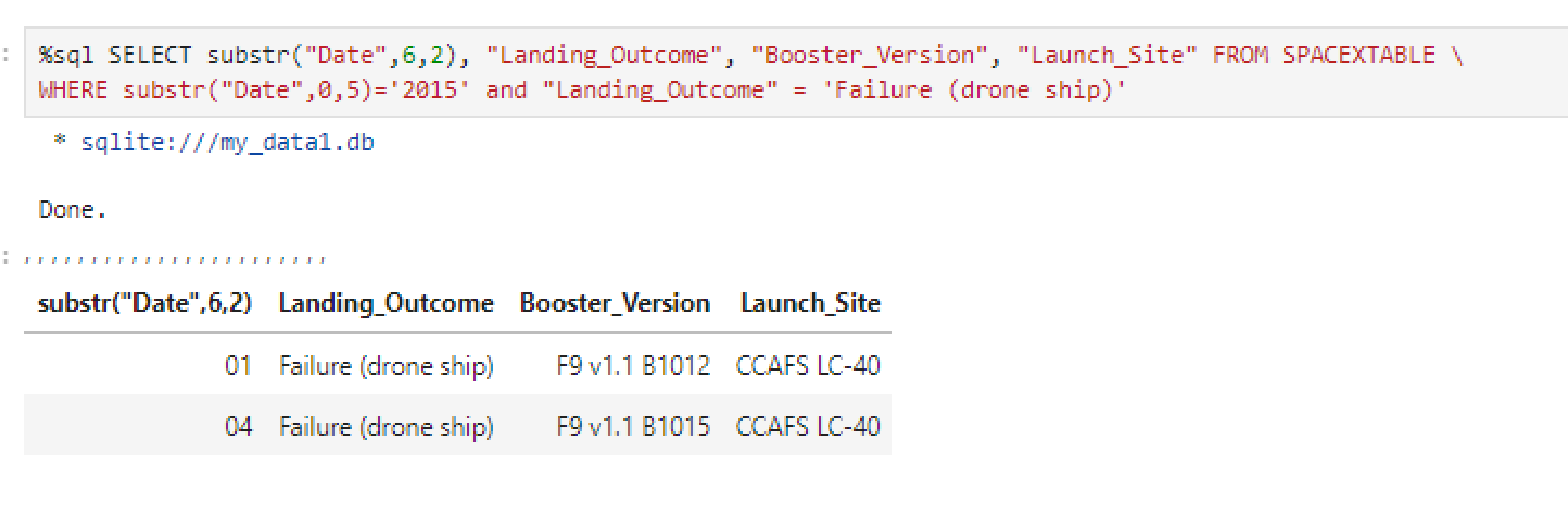


```
[16]: %sql SELECT "Booster_Version",PAYLOAD_MASS__KG_ FROM SPACEXTABLE \
      WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE)

       * sqlite:///my_data1.db

      Done.
[16]:
```

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

2015 Launch Records

- We can list the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015 by using 'substr()' to remove year from the date column:



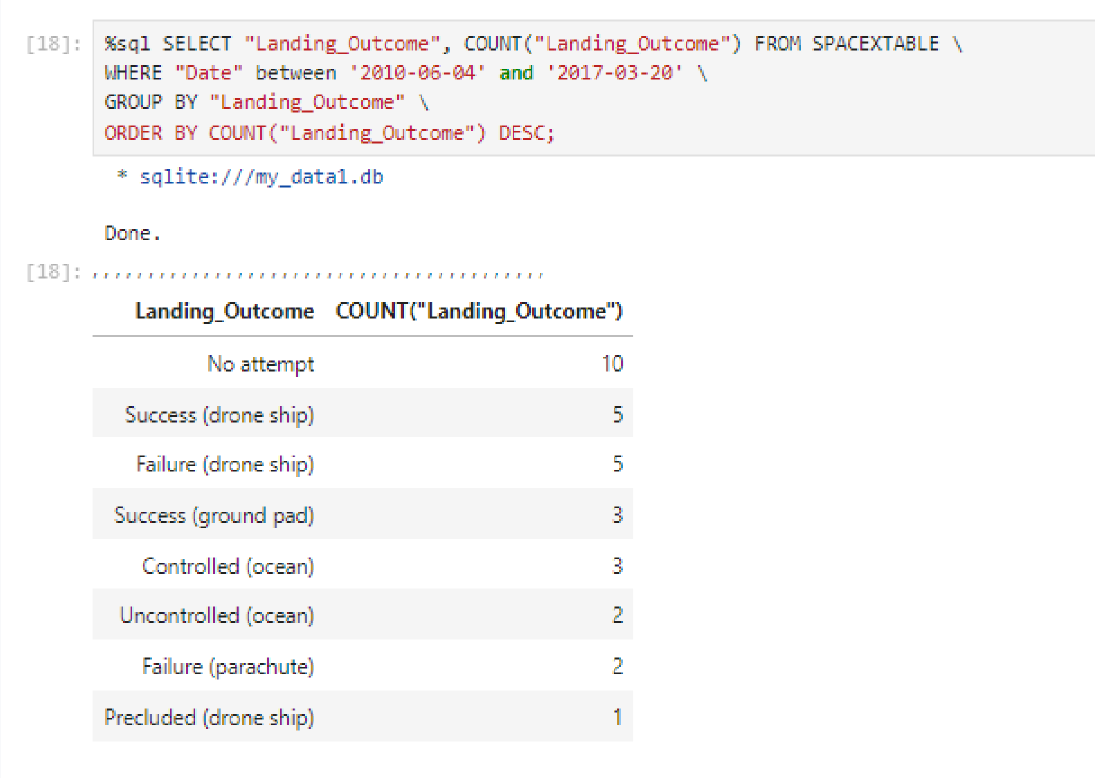
```
%sql SELECT substr("Date",6,2), "Landing_Outcome", "Booster_Version", "Launch_Site" FROM SPACEXTABLE \
WHERE substr("Date",0,5)='2015' and "Landing_Outcome" = 'Failure (drone ship)'
```

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

Done.
```

substr("Date",6,2)	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- We can 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. First, we use 'between' in the WHERE clause to restrict the dates. Then, we use the command ORDER BY COUNT("Landing_Outcome") DESC to order the results from most common to least common:



```
[18]: %sql SELECT "Landing_Outcome", COUNT("Landing_Outcome") FROM SPACEXTABLE \
      WHERE "Date" between '2010-06-04' and '2017-03-20' \
      GROUP BY "Landing_Outcome" \
      ORDER BY COUNT("Landing_Outcome") DESC;

       * sqlite:///my_data1.db

      Done.

[18]:
```

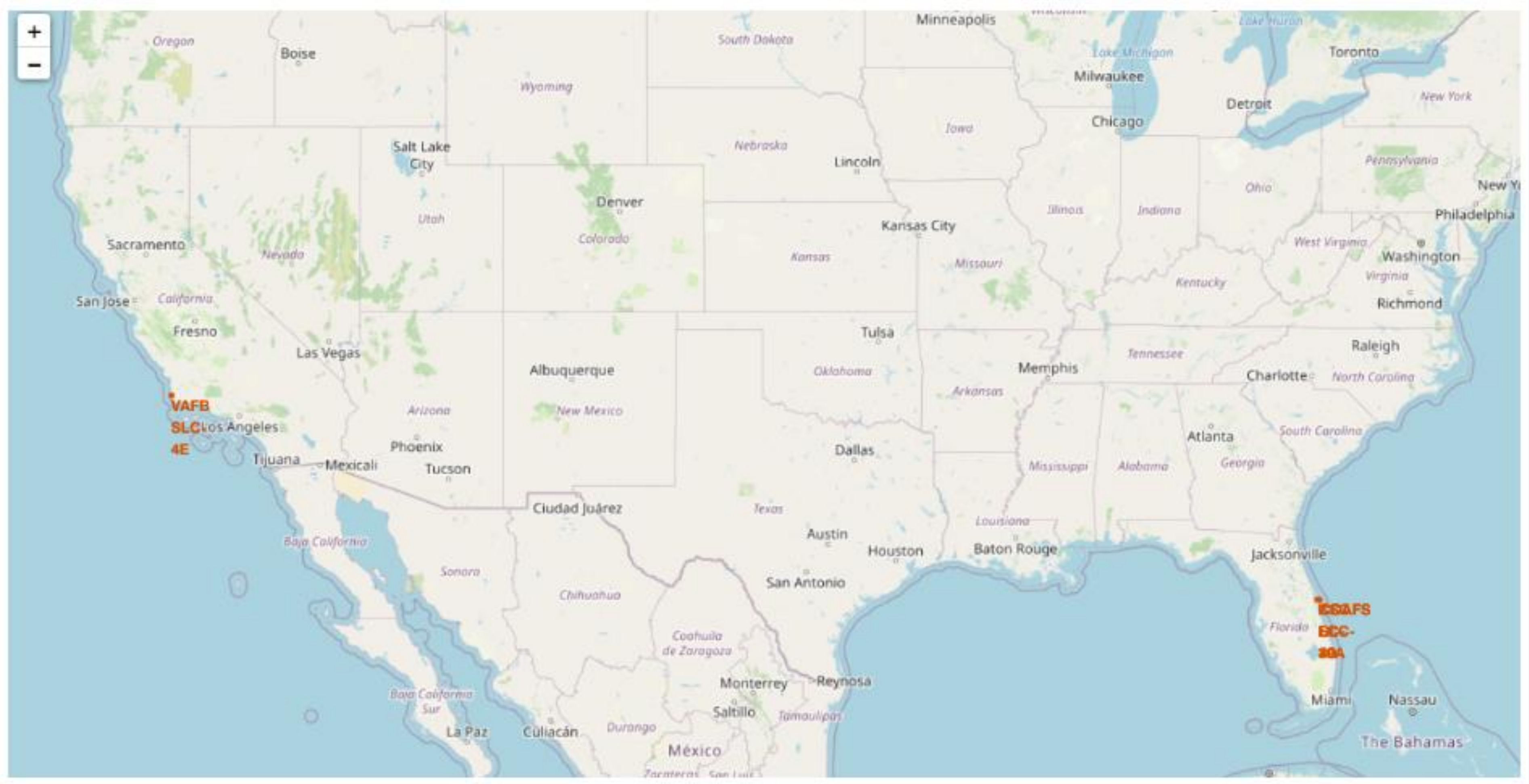
Landing_Outcome	COUNT("Landing_Outcome")
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

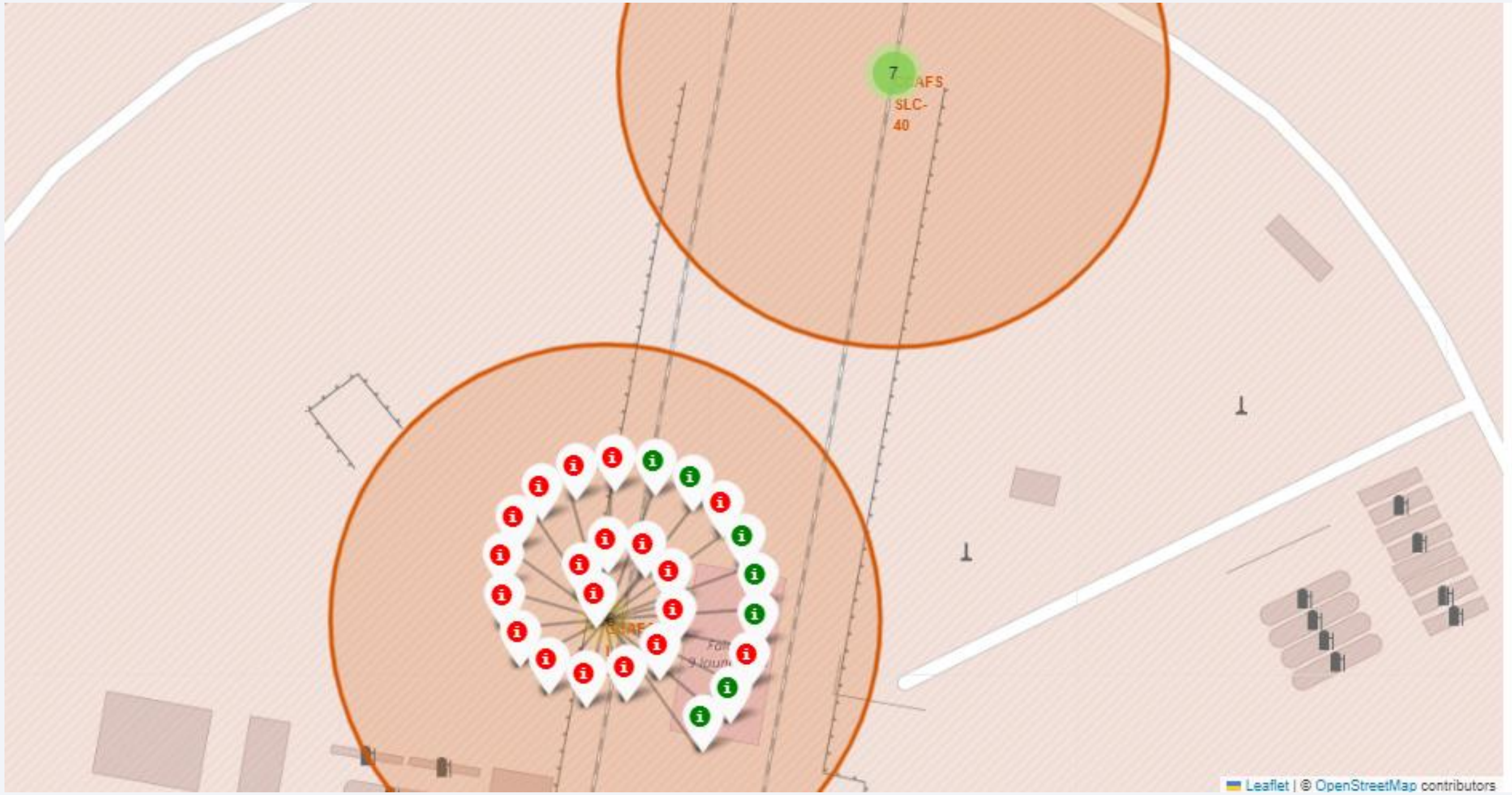
Marking all launch sites on a map

- We can see all launch sites are near the coast and down south (nearer to the equator), 3 of them in Florida and one in California:



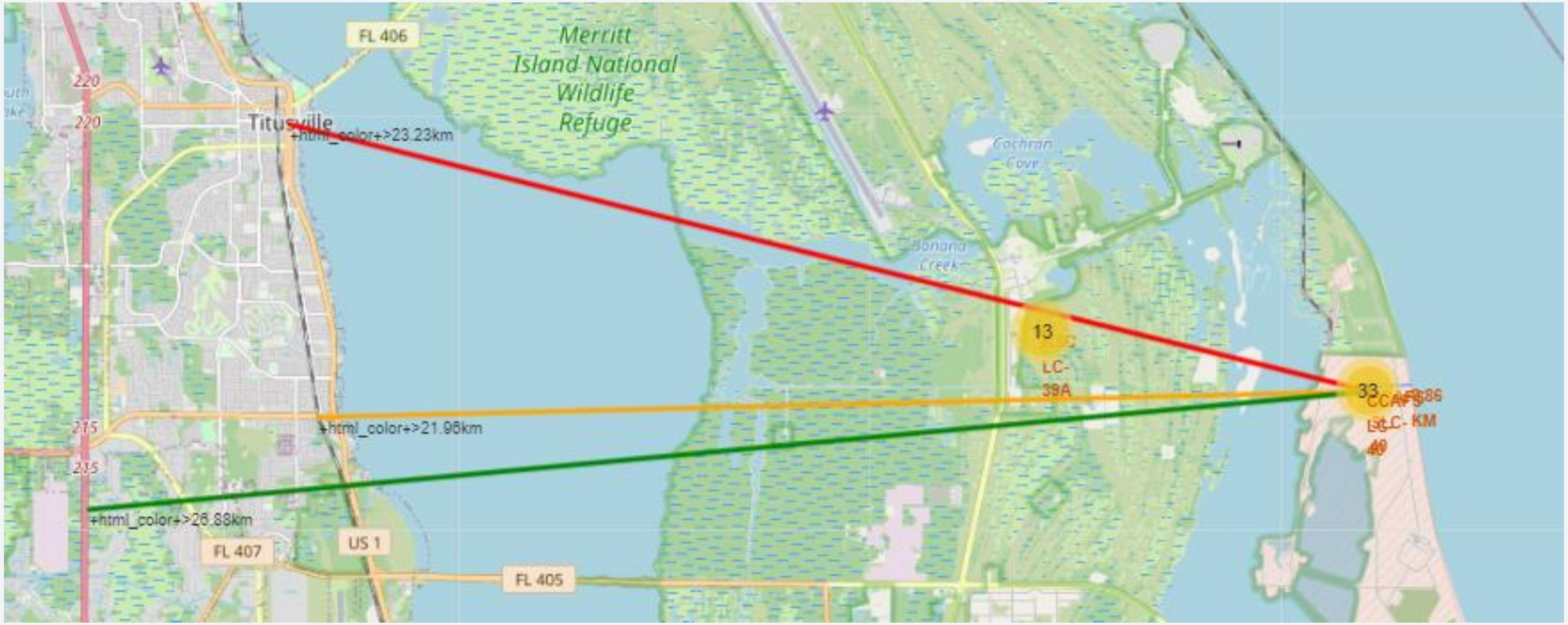
Marking the success/failed launches for each site on the map

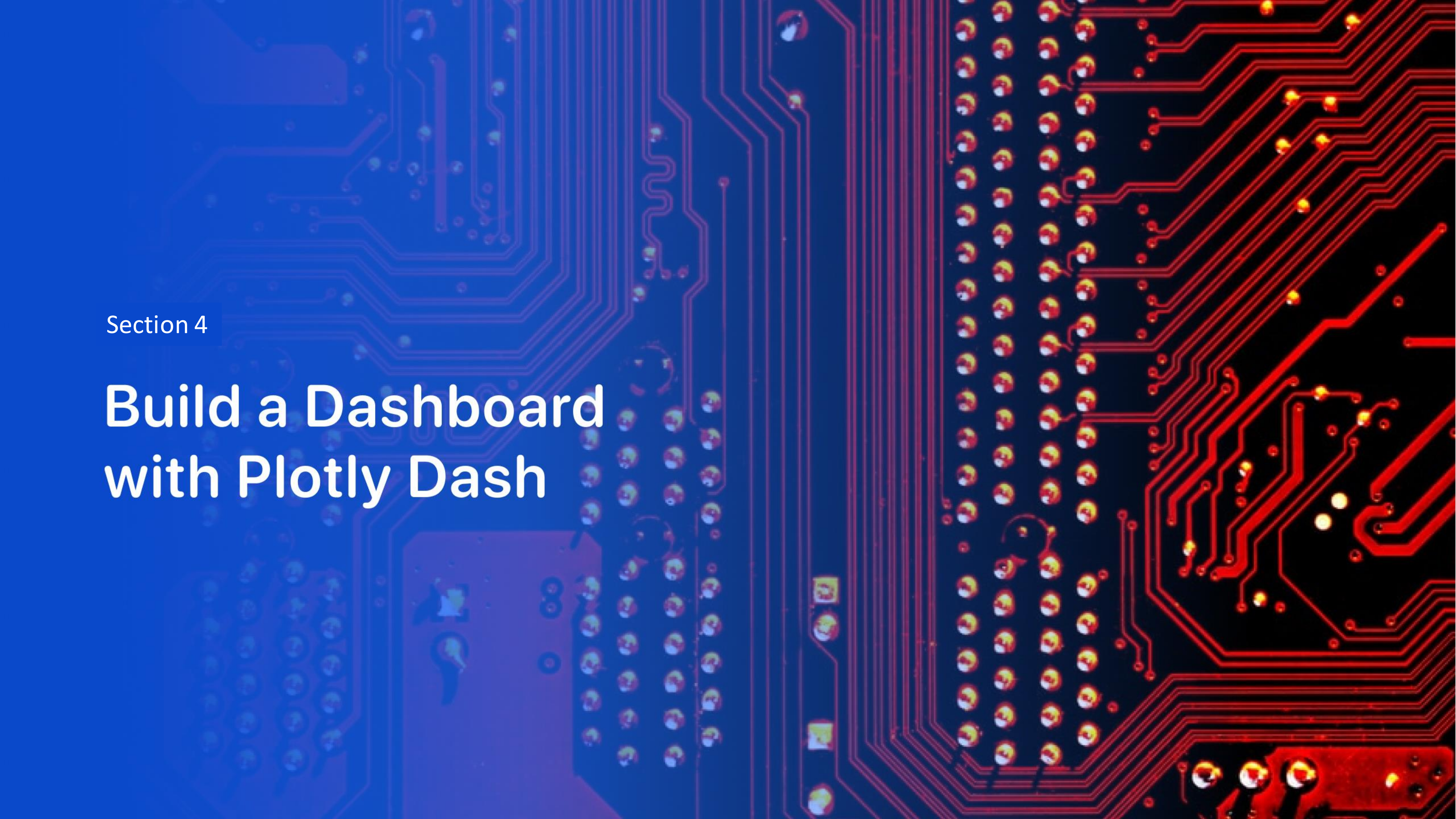
- Once you zoom in a launch site, you're able to see the succesful (green) and failed (red) landings there:



Distances between a launch site and its proximities

- We can see that launch sites are relatively isolated from highways, railways and cities:



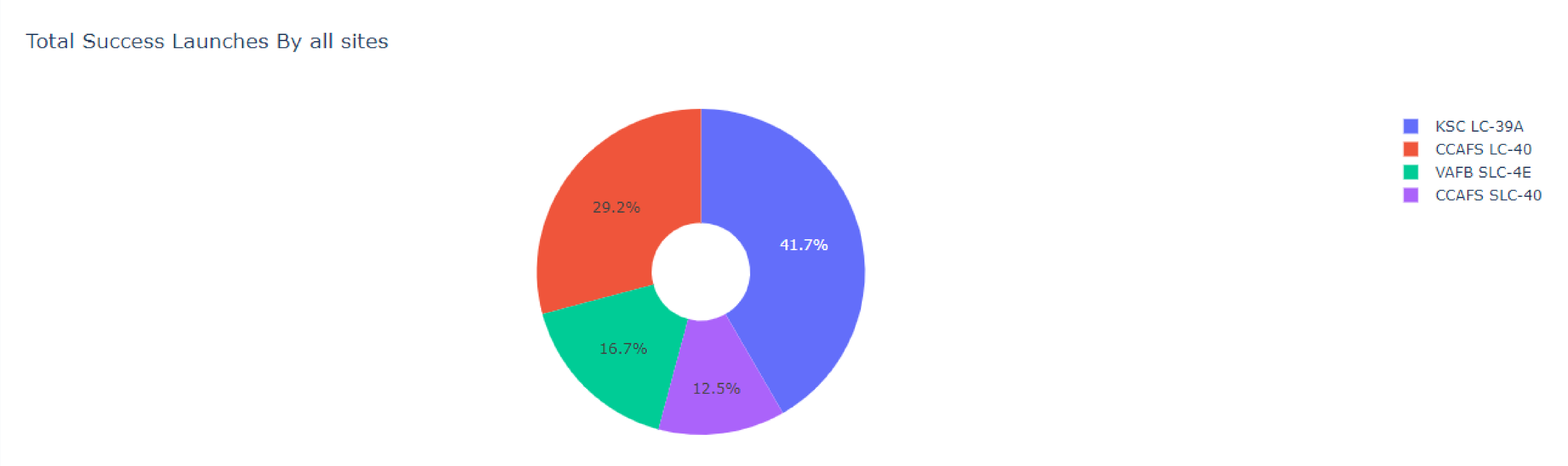


Section 4

Build a Dashboard with Plotly Dash

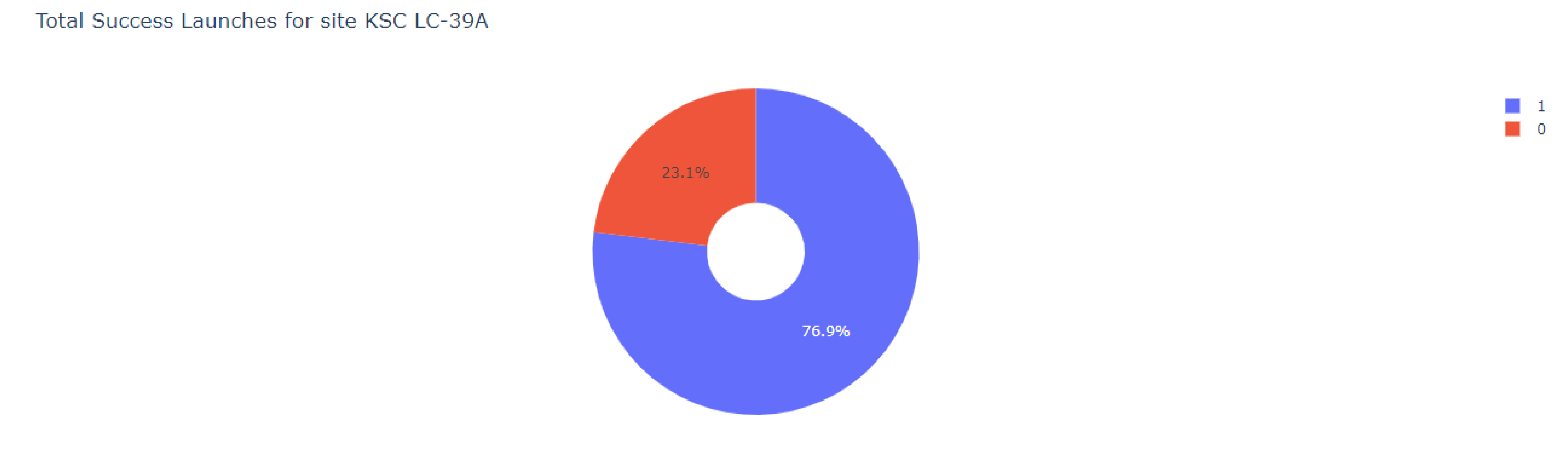
Success Landings by Launch Site

- The launch site with highest number of successes was KSC-LC-39-A, with 41,7% of the total instances:



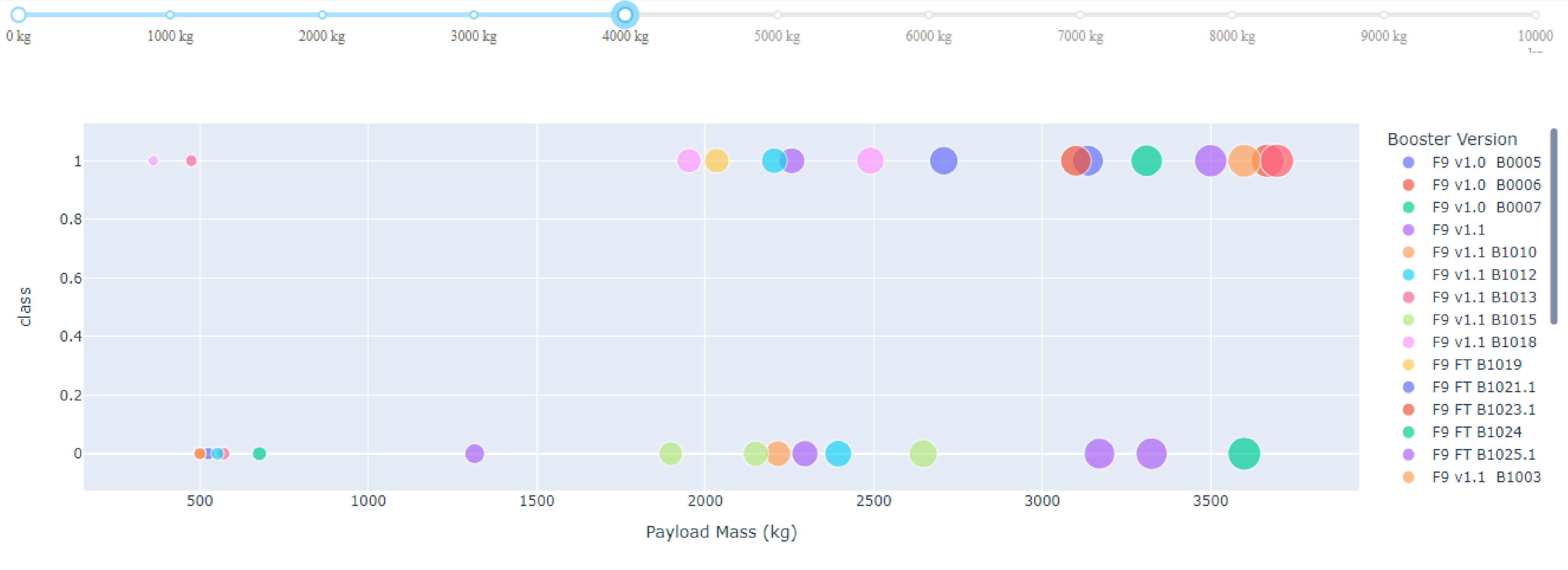
Launch site with highest success ratio

- The Launch site with the highest success rate was KSC LC-39A, with a rate of 76.9%

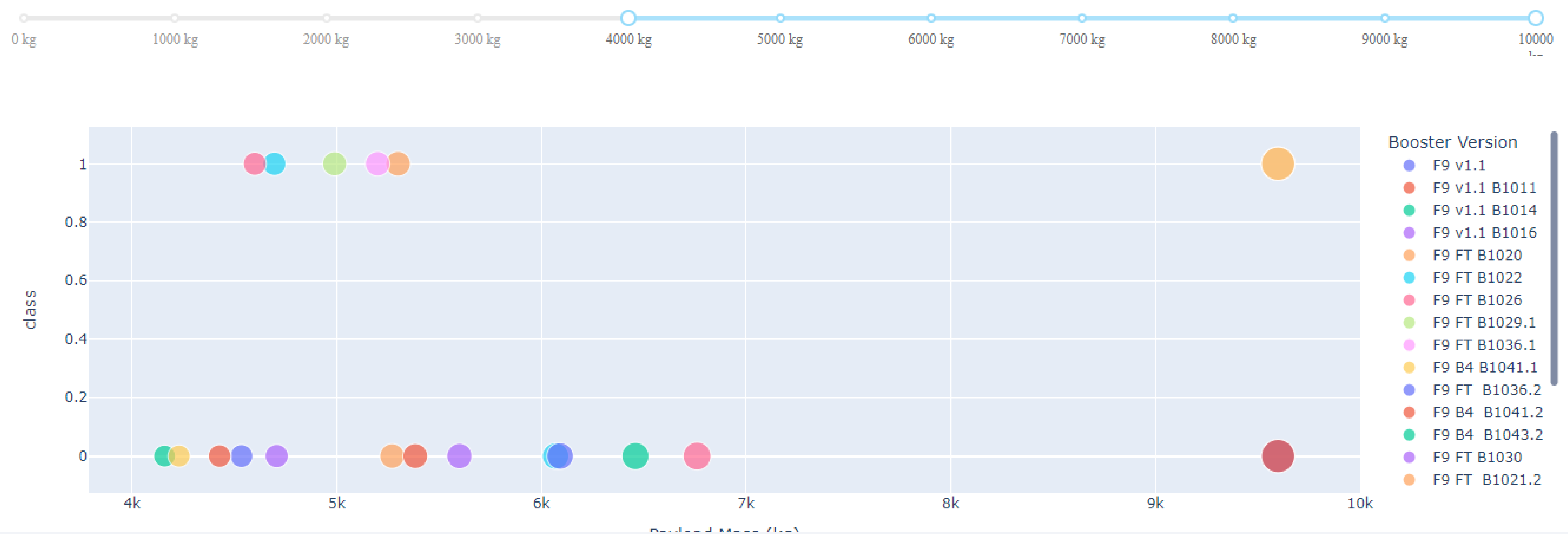


Payload vs. Launch Outcome (0-4.000 kg)

- We can see the success rate of light payloads (0-4000kg) is greater than that of heavier payloads (4.000-10.000 kg) - see the latter at the next slide:



Payload vs. Launch Outcome (4.000-10.000 kg)

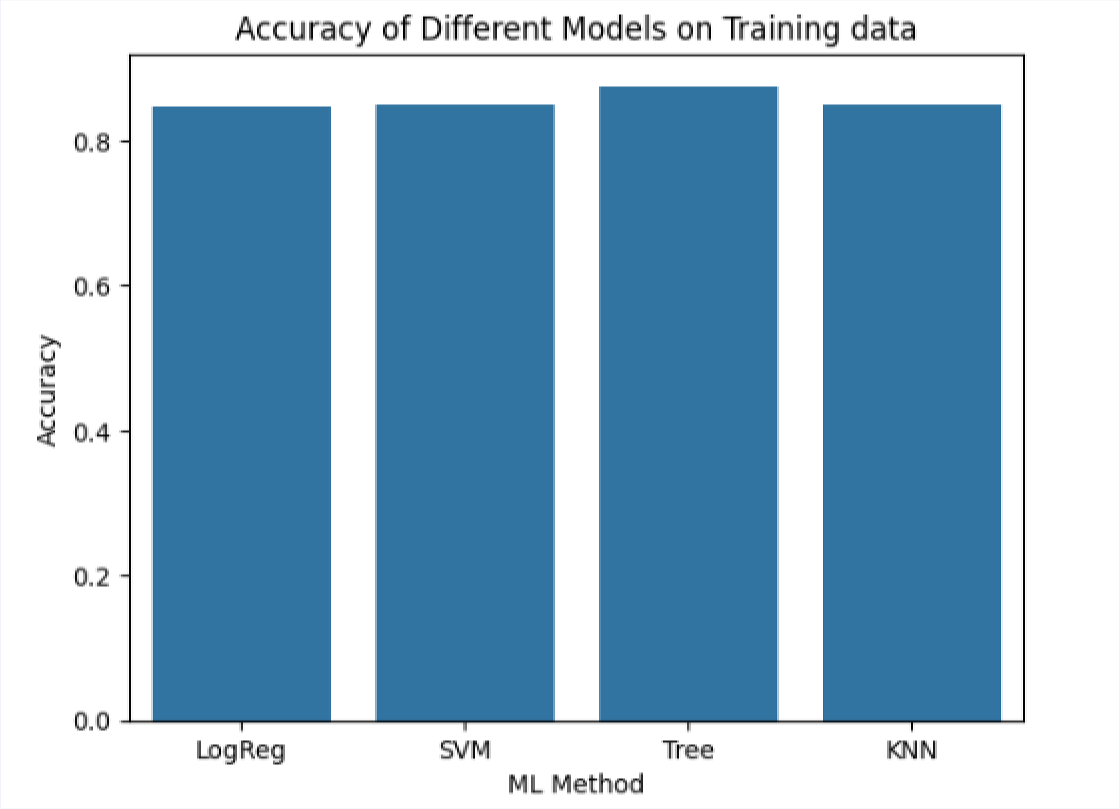


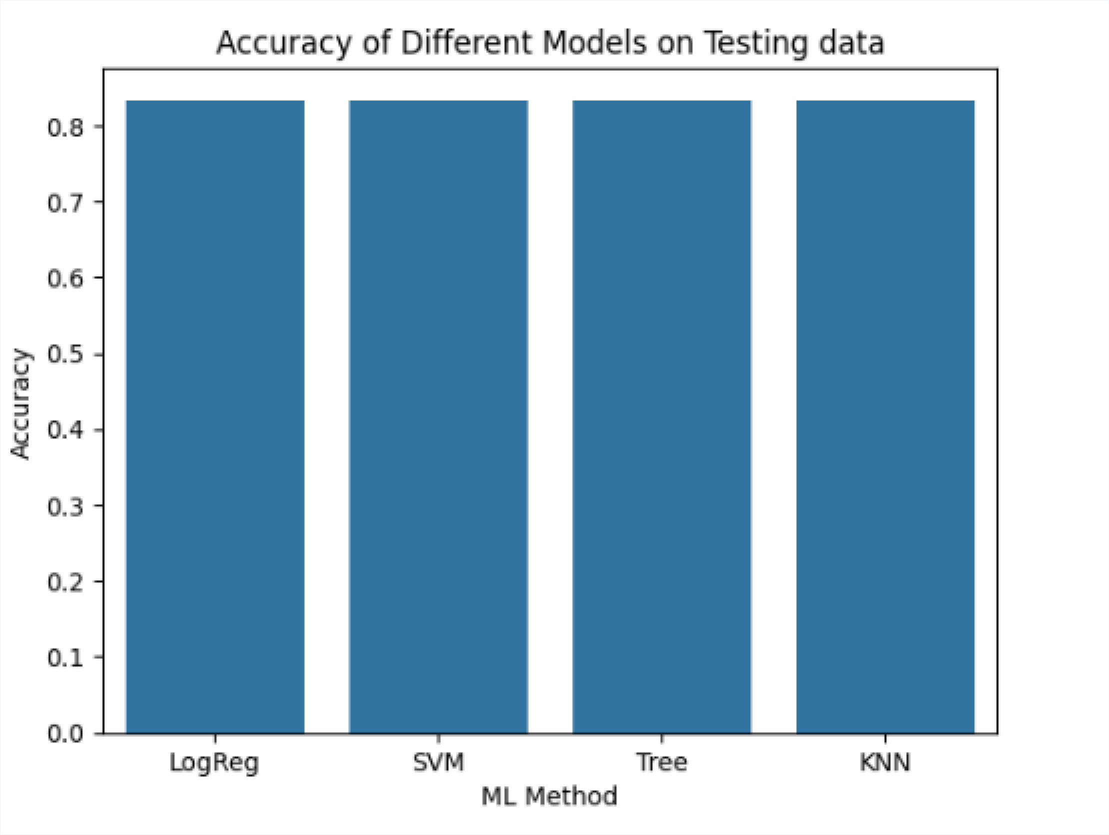
Section 5

Predictive Analysis (Classification)

Classification Accuracy

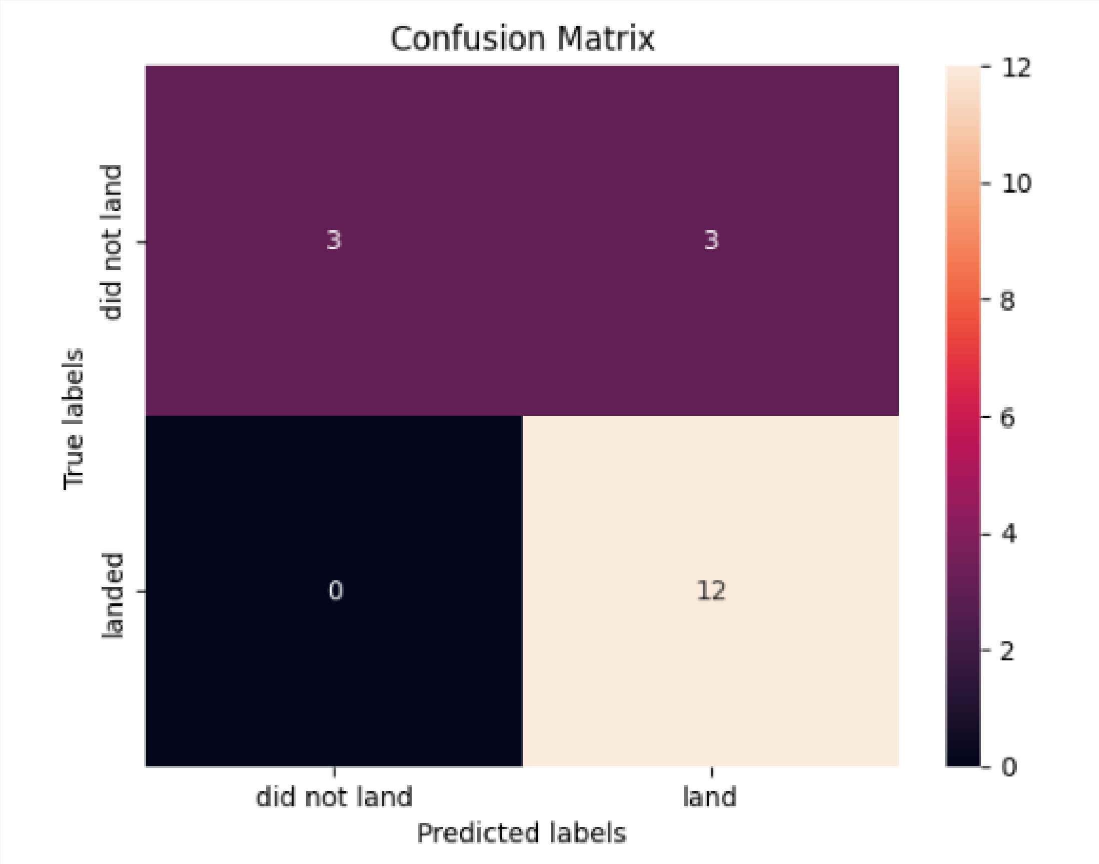
- With a bar plot, we can see the accuracy for the training data is slightly higher for the decision tree model (87.5%), while the accuracy on the test data is the same across all models (83.3%). All models had their hyperparameters tuned with GridSearchCV():





Confusion Matrix

- Below is the confusion matrix for the decision tree model. We can see that it correctly classifies all 12 successful landings, but it generates 3 false positives when looking at the 6 failed landings.



Conclusions

- Flight number (year) and payload mass are most significant for explaining success in landing the first stage.
- More data is needed to understand the role of other features, as well as technical knowledge about aerospace and the Earth's orbits.
- All classification models used had good accuracy score, going beyond 80%. Decision tree was the top performer, with 87,5%.
- XGBoost, often pointed by the literature as the best model for solving both regression and classification problems, was not tested in this study, but it could outperform the other classifications models used here.

Thank you!