IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Federico Orlandini
11th May 2024

Outline

Executive Summary

- Summary of methodologies
  - Data Collection through API and 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 from Machine Learning Lab

Introduction

- SpaceX is a revolutionary company who has disrupt the space industry by offering a rocket launches specifically Falcon 9 as low as 62 million dollars; while other providers cost upward of 165 million dollar each. Most of this saving thanks to SpaceX astounding idea to reuse the first stage of the launch by re-land the rocket to be used on the next mission. Repeating this process will make the price down even further. As a data scientist of a startup rivaling SpaceX, the goal of this project is to create the machine learning pipeline to predict the landing outcome of the first stage in the future. This project is crucial in identifying the right price to bid against SpaceX for a rocket launch.
- The problems include:
  - Identifying all factors that affect the landing outcome.
  - The relationship between each variables and how it is affecting the outcome.
  - The best condition needed to increase the probability of successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data has been collected querying the SpaceX REST API and scraping data from the SpaceX Wikipedia page
- Perform data wrangling
  - Data was processed to identify and prepare the variable that represent the outcome of each launch
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Training and performance evaluation for different classification algorithms to select the best one

Data Collection

- Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. As mentioned, the dataset was collected using the SpaceX REST API and scrapping the SpaceX Wikipedia page
- For the SpaceX REST API, we used Http Get request to retrieve the data then, we decoded the response content as Json and turn it into a pandas DataFrame object to simplify the data clean up and preparation.
- We collected additional in formation scrapping the SpaceX Wikipedia we page,using the BeautifulSoup to extract the launch records from the web page

Data Collection – SpaceX API

- Collecting SpaceX data consuming the SpaceX API, via Http requests
  - Different API endpoints for different data (rockets, launchpads, payloads, etc.)
- Converting Http responses from Json to Python Pandas DataFrames
- Cleaning up the data
- Python notebook: https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/jupyter-labs-spacex-data-collection-api.ipynb

```
#----- GETTING THE DATA -----

# Get the data using Http GET request
static_json_url='https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBM-DS0321EN-
SkillsNetwork/datasets/API_call_spacex_api.json'
response = requests.get(static_json_url)

#----- CONVERTING DATA TO DATAFRAMES -----

# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())

#----- DATA CLEAN UP -----

# Lets take a subset of our dataframe keeping only the features we want and the
flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number',
'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2
extra rocket boosters and rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single
value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting
the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Data Collection - Scraping

- Get the SpaceX Wikipedia web page content (https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches)
- Read the data contained in the main table in the web page using BeautifulSoap Python library
- Python notebook: https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/jupyter-labs-webscraping.ipynb

```
# Get the web page content
static_url =
https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_
Heavy_launches&oldid=1027686922
response = requests.get(static_url)

# Use BeautifulSoap to navigate the HTML page
bs_object = BeautifulSoup(response.content)
html_tables = bs_object.find_all('table')

# Parse the column name from the HTML table
for header_item in html_tables[2].find_all('th'):
    name = extract_column_from_header(header_item)
    if( name is not None and len(name) > 0 ):
          column_names.append(name)

# Parse the data in the HTML table rows
for table_number,table in
enumerate(bs_object.find_all('table',"wikitable plainrowheaders 
collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding
to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
                ...other code
```

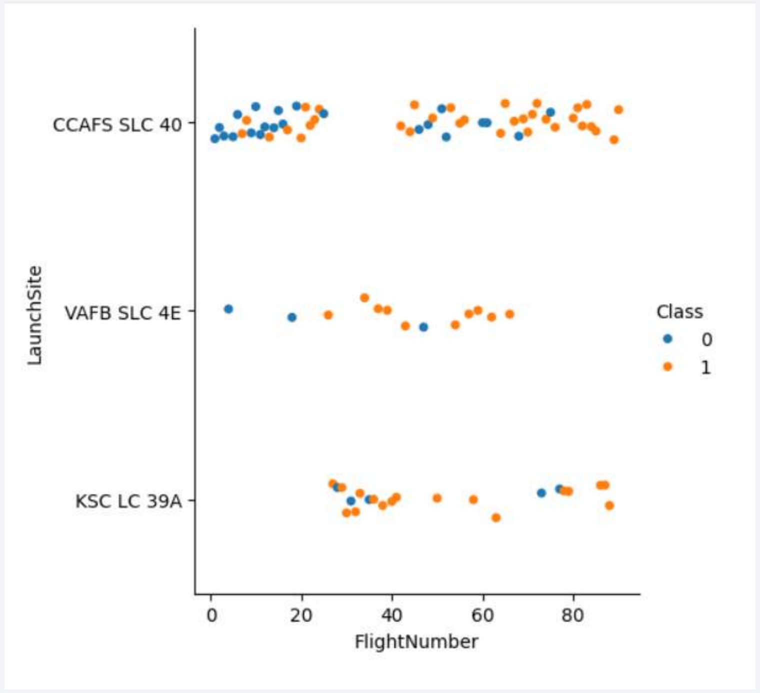
Data Wrangling

- Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA)

- Data wrangling process key steps:
  - Identify and calculate the percentage of the missing values in each attribute in the dataset
  - Identify which columns are numerical and categorical
  - Calculate the number of launches on each site
  - Calculate the number and occurrence of mission outcome per orbit type
  - Create a landing outcome label from Outcome column
- Jupyter notebook: https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/labs-jupyter-spacex-Data%20wrangling.ipynb

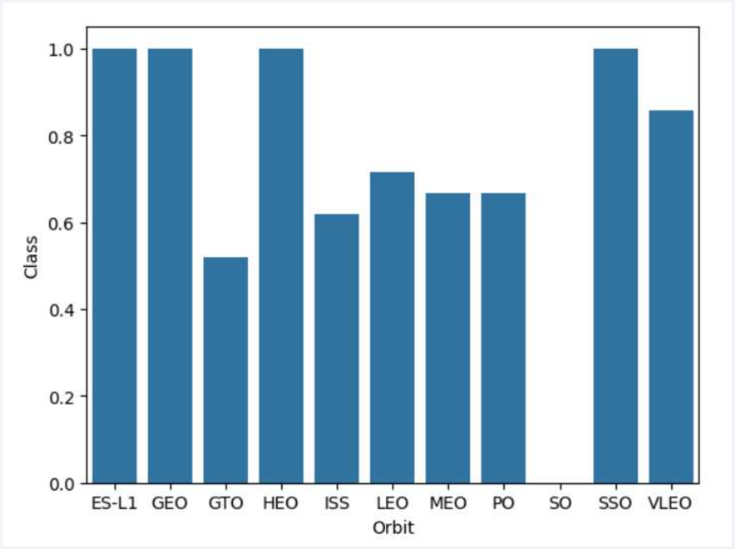
EDA with Data Visualization

- We used scatter graph to investigate the relationship between the attributes such as between:
  - Payload and Flight Number.
  - Flight Number and Launch Site.
  - Payload and Launch Site.
  - Flight Number and Orbit Type.
  - Payload and Orbit Type.
- Scatter plots show dependency of attributes on each other and they are very useful in finding which factors affecting the most to the success of the landing outcomes
- Jupyter notebook: https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/jupyter-labs-eda-dataviz.ipynb



EDA with Data Visualization

- Once we get a hint of the relationships using scatter plot, we will then use further visualization tools such as bar graph and line plots graph for further analysis.
- In this case, we will use the bar graph to determine which orbits have the highest probability of success.
- We then use the line graph to show the launch success yearly trend.
- We then use Feature Engineering to be used in success prediction in the future module by created the dummy variables to categorical columns.
- Jupyter notebook: https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/jupyter-labs-eda-dataviz.ipynb



EDA with SQL

- Storing the data in a relational database and using SQL, we have been able to better understand the data in the dataset. For example:
  - Displaying the names of the launch sites.
  - Displaying 5 records where launch sites begin with the string ‘CCA’.
  - Displaying the total payload mass carried by booster launched by NASA (CRS).
  - Displaying the average payload mass carried by booster version F9 v1.1.
  - Listing the date when the first successful landing outcome in ground pad was achieved.
  - Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
  - Listing the total number of successful and failure mission outcomes.
  - Listing the names of the booster_versions which have carried the maximum payload mass.
  - Listing the failed landing_outcomes in drone ship, their booster versions, and launch sites names for in year 2015.
  - Rank the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order.

Jupyter notebook:
https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We used maps to investigate potential correlation between lunch success rate and the launch site location
  - We marked each site's location on a map using site's latitude and longitude coordinates using red and green markers and MarkerCluster objects on the map
  - We marked the success/failed launches for each site on the map
  - We calculated the distances between a launch site to its proximities
- We then used the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:
  - How close the launch sites with railways, highways and coastlines?
  - How close the launch sites with nearby cities?
- Jupyter notebook: https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allowing the user to analyze the data using a graphical interface.
  - 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.

- URL python application:
  - https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/spacex_dash_app.py

- URL screenshots:
  - https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/Dashboard%20-%20screenshot%201.PNG
  - https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/Dashboard%20-%20screenshot%202.PNG

Predictive Analysis (Classification)

- Building the Model
  - Load the dataset into NumPy and Pandas
  - Transform the data and then split into training and test datasets
  - Select which type of ML algorithm to use
  - Set up GridSearchCV to optimize the hyperparameters and fit the dataset
- Evaluating the Model
  - Check the accuracy for each model
  - Get tuned hyperparameters for each type of algorithms.
  - Plot the confusion matrix
- Improving the Model
  - Use Feature Engineering and Algorithm Tuning
- Find the Best Model
  - The model with the best accuracy score will be the best performing model
- URL Jupyter notebook: https://github.com/federicoorlandini/IBMDataScience/blob/main/AppliedData%20ScienceCapstone/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

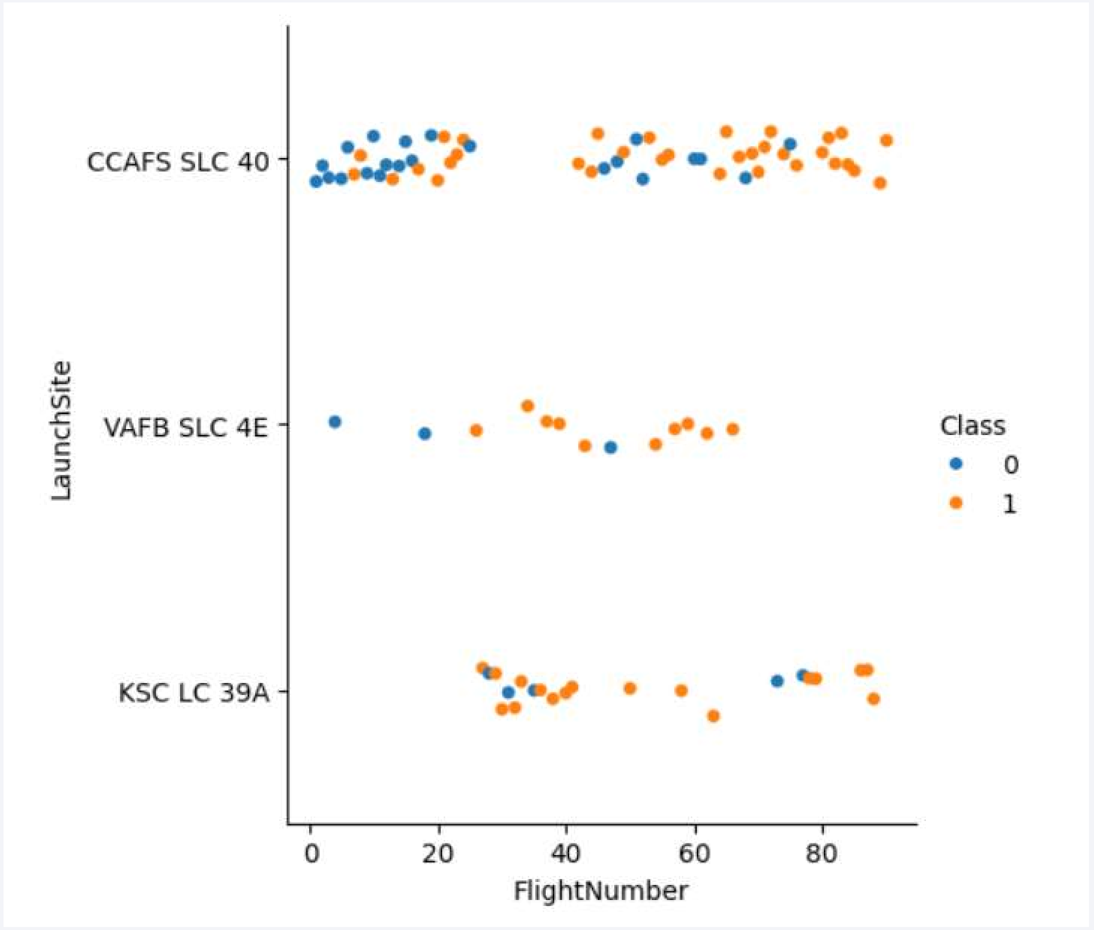
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

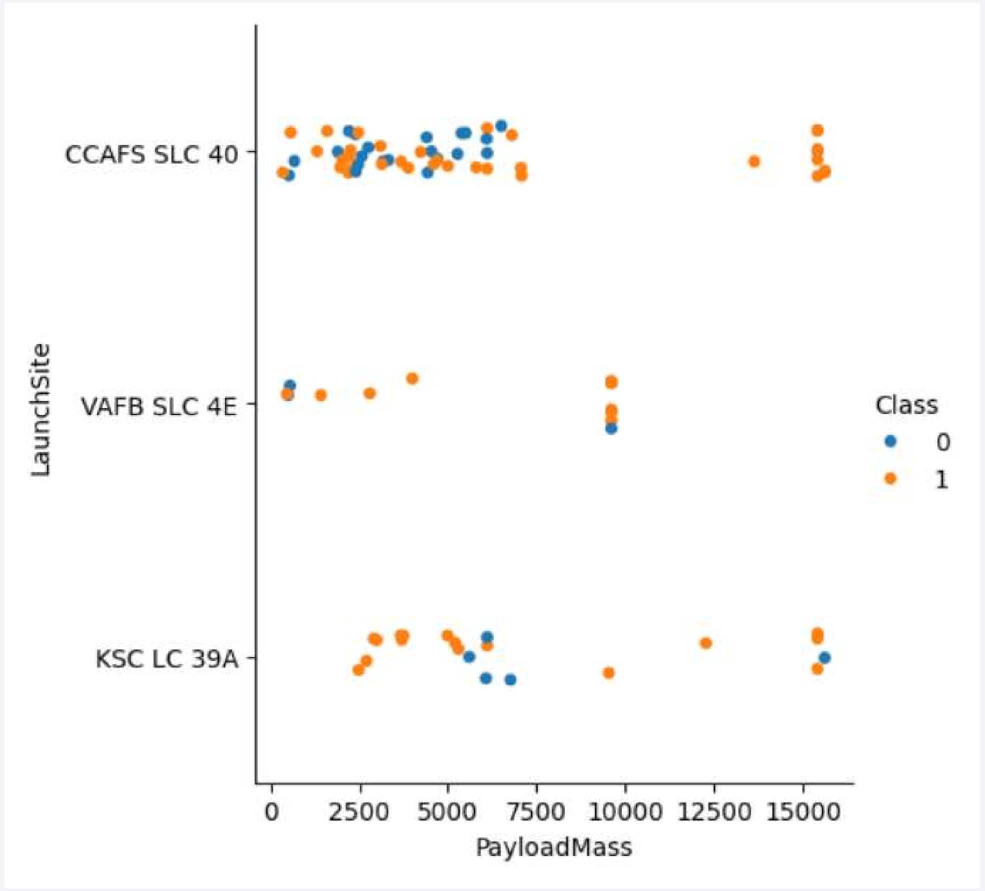
Flight Number vs. Launch Site

This scatter plot shows that the larger the flights amount of the launch site, the greater the success rate will be, even if site CCAFS SLC40 shows the least pattern of this



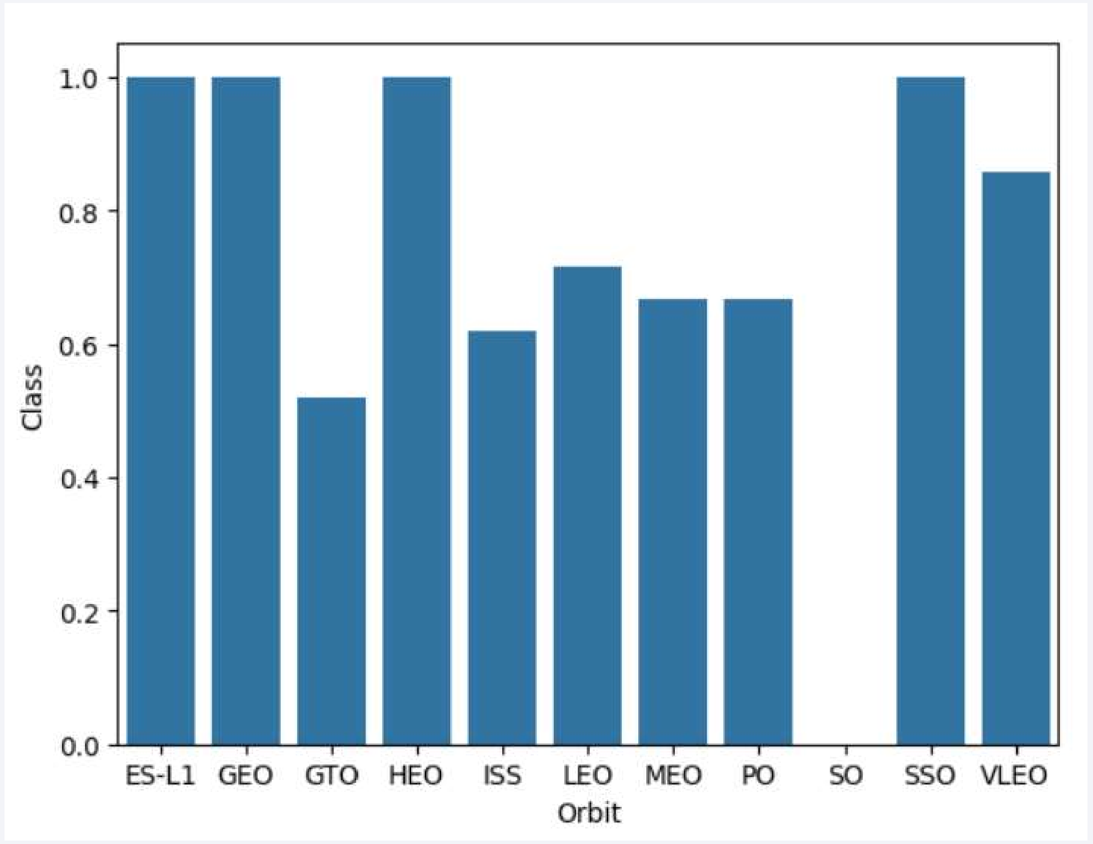
Payload vs. Launch Site

This scatter plot shows that the success rate is higher if the Payload Mass is greater than 7000 kg.



Success Rate vs. Orbit Type

- This bra graph shows the correlation between the orbits and the landing outcomes. The figure shows that some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.
- However some of this orbits has only 1 occurrence such as GEO, SO, HEO and ES-L1 which mean that we need more data to enrich our dataset to see pattern or trend before we draw any conclusion.

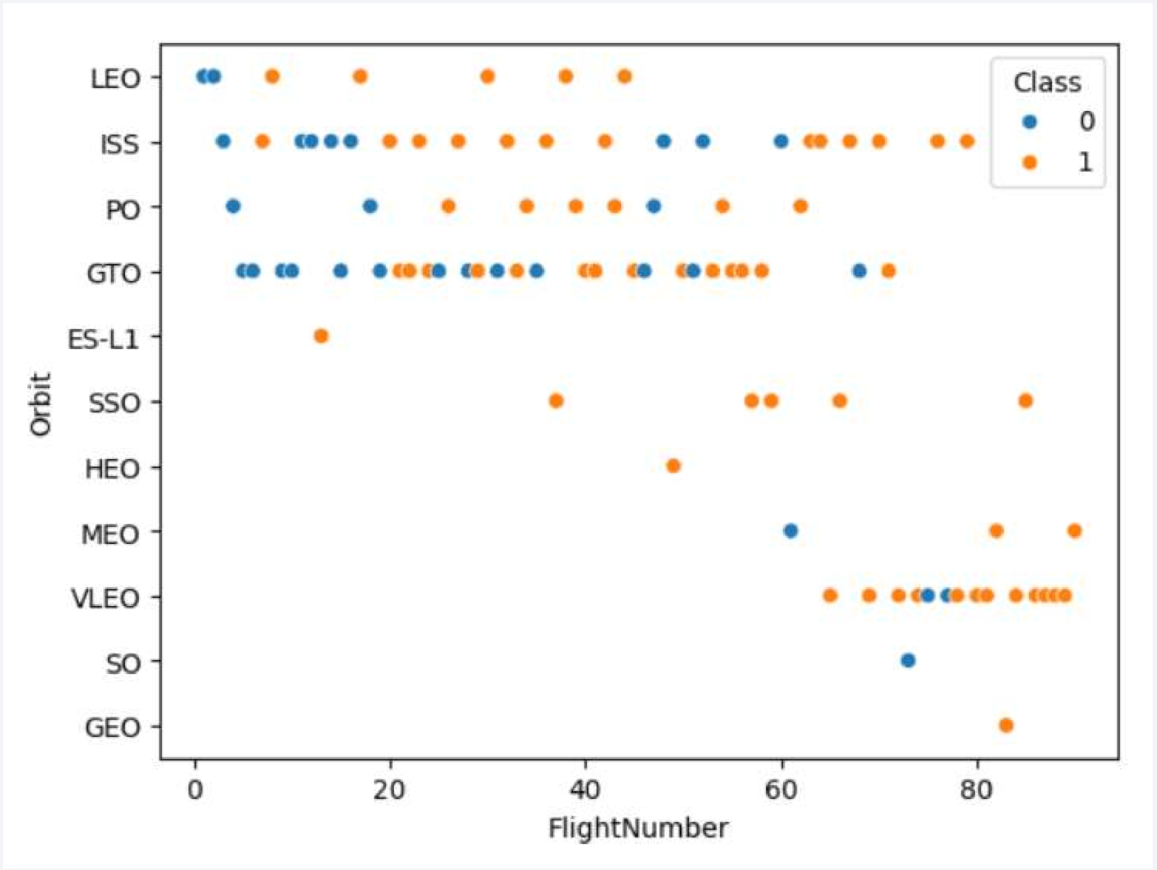


Flight Number vs. Orbit Type

This scatter plot shows that in general larger the flight number on each orbits, the greater the success rate.

However, the GTO orbit seems to be the exception since it doesn't show the same trend.

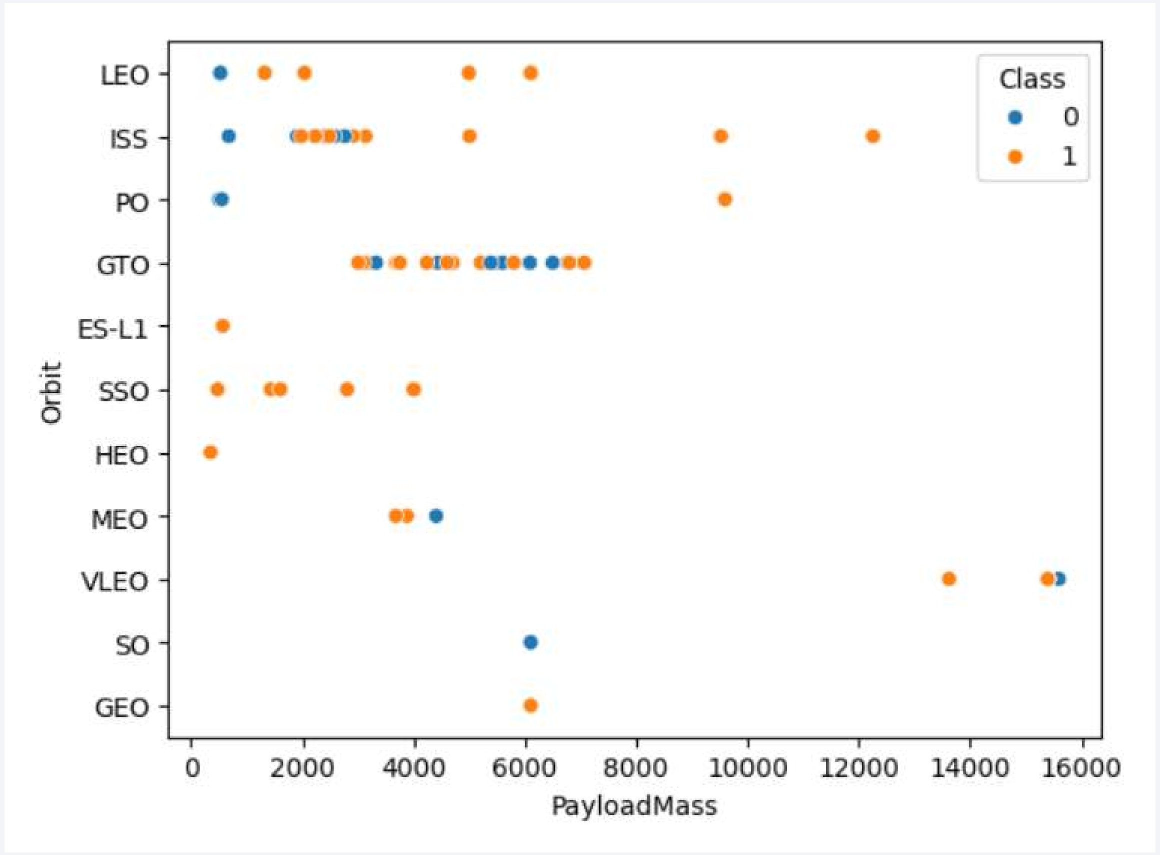
Some of the orbit types have only one occurrence of data and they should be excluded by the statement as a bigger data set is needed



Payload vs. Orbit Type

With heavier payloads the successful landing or positive landing rate are more for Polar, LEO and ISS, while it has negative impact on MEO and VLEO orbit.

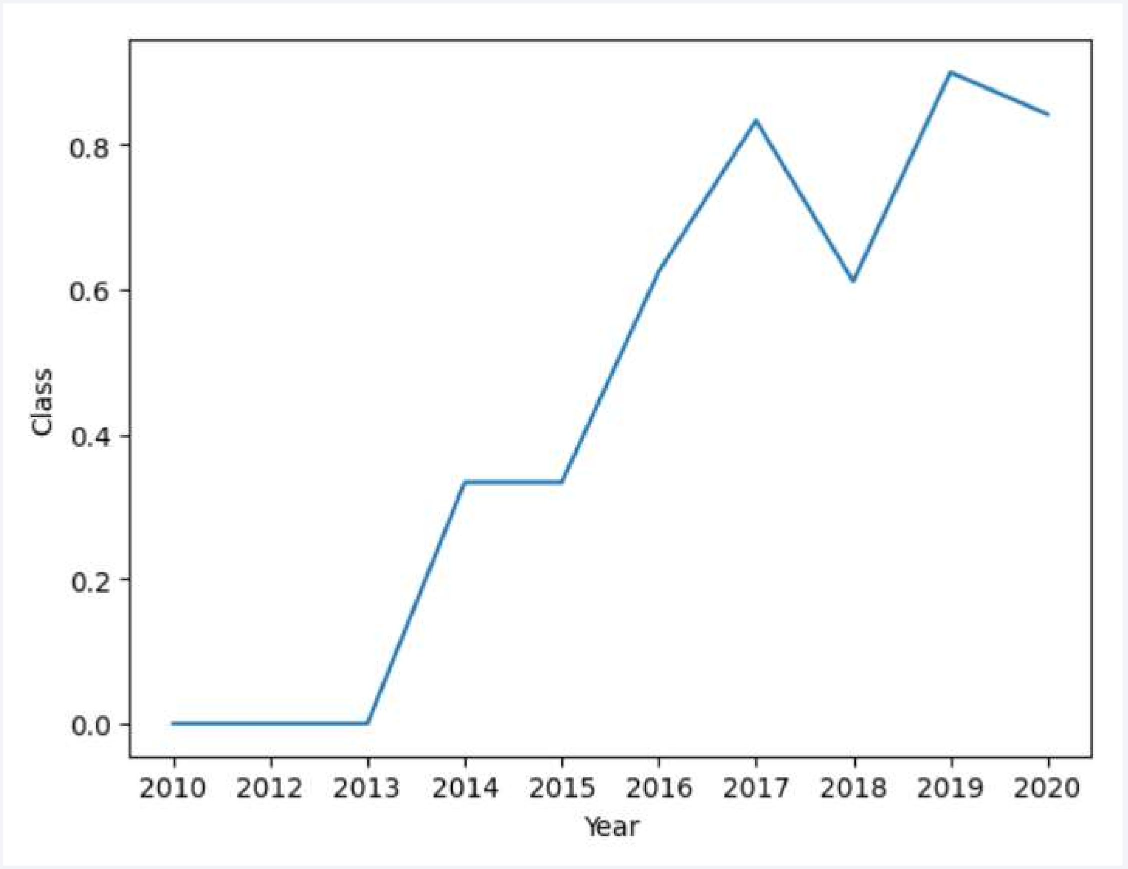
For GTO we cannot distinguish this well as both positive landing rate and negative landing are both there here.



Launch Success Yearly Trend

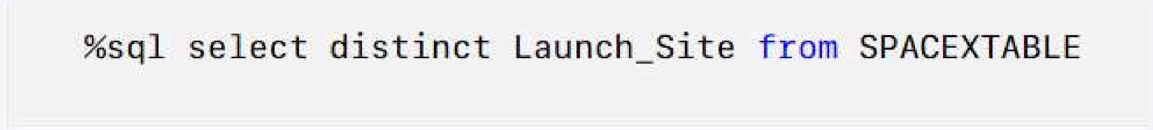
This figures clearly shows the increasing trend from the year 2013 until 2020, with a slightly decrease in 2018.

However, the launch success rate in the last years is close to 100%.



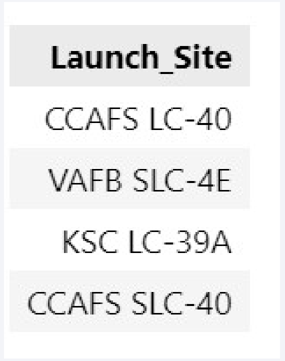
All Launch Site Names

- We found the list of all the launch site names loading the dataset in a relational database (SQLite) and using the SQL DISTINCT statement to retrieve the list:



```
%sql select distinct Launch_Site from SPACEXTABLE
```

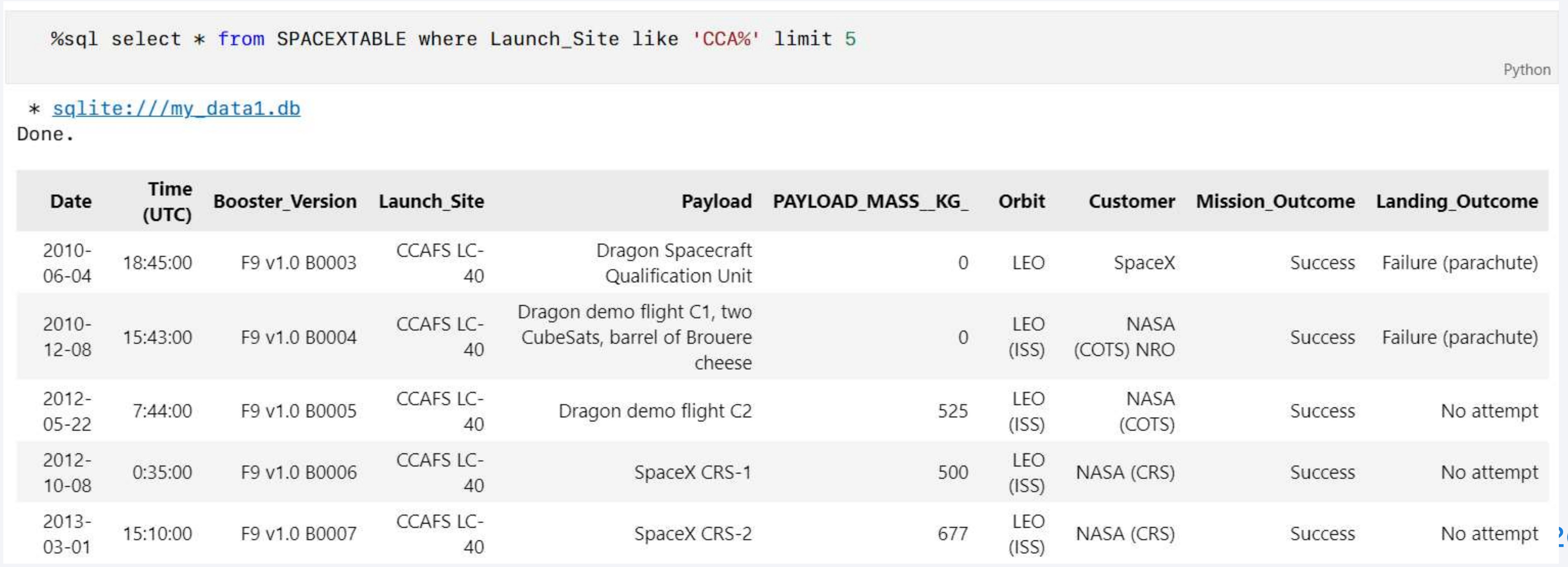
- Present your query result with a short explanation here



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

Launch Site Names Begin with 'CCA'

- Using the dataset data imported in the relational database, we also found the list of the first 5 names of the launch sites where the site name begin with ‘CCA’. We used the combination of the LIKE statement and the wildcard ‘%’



```
%sql select * from SPACEXTABLE where Launch_Site like 'CCA%' limit 5
```

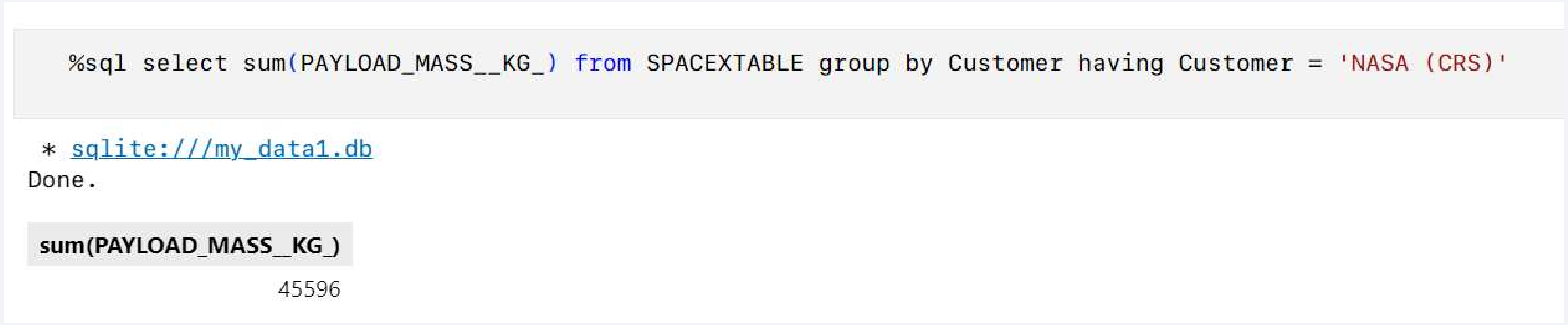
Python

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

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

- We were also able to calculate the total amount of Payload Mass (in Kg) that have been carried by boosters launched by NASA (CRS) using the SUM() fucntion to sum all the payloads in the result set



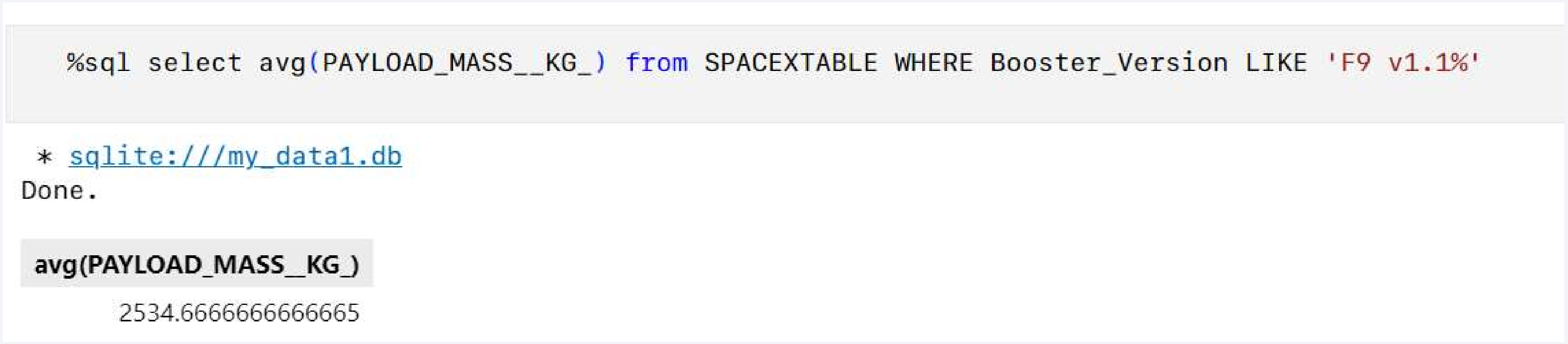
```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE group by Customer having Customer = 'NASA (CRS)'
```

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

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster F9 v1.1 is equal to 2534,67 Kg using the AVG() function to compute the average and the LIKE statement with wildcard '%' to filter the data



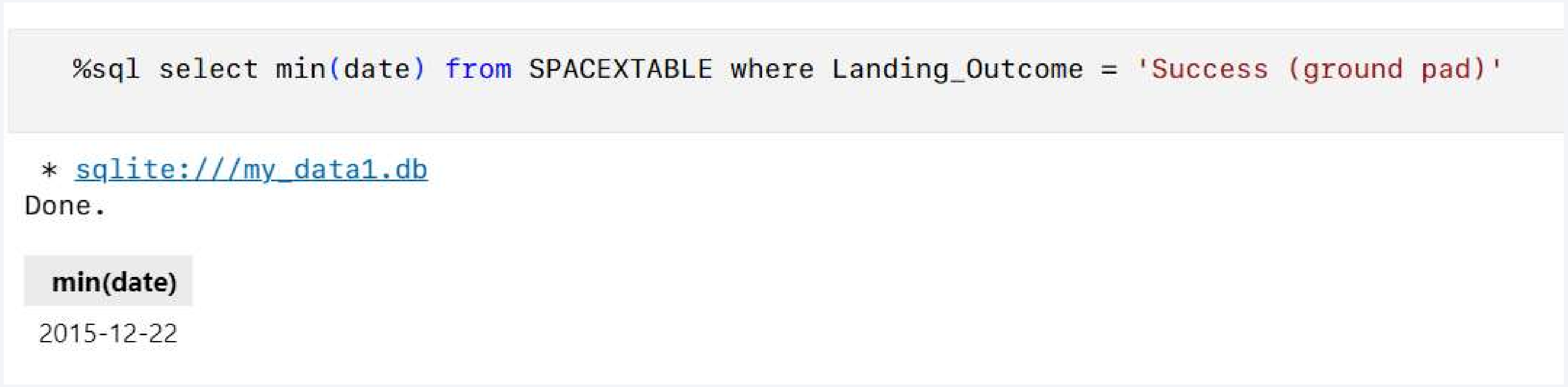
```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%'
```

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

avg(PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

- The first succesful landing outcome in ground pad was achieved on the 22nd December 2015, using the MIN() function



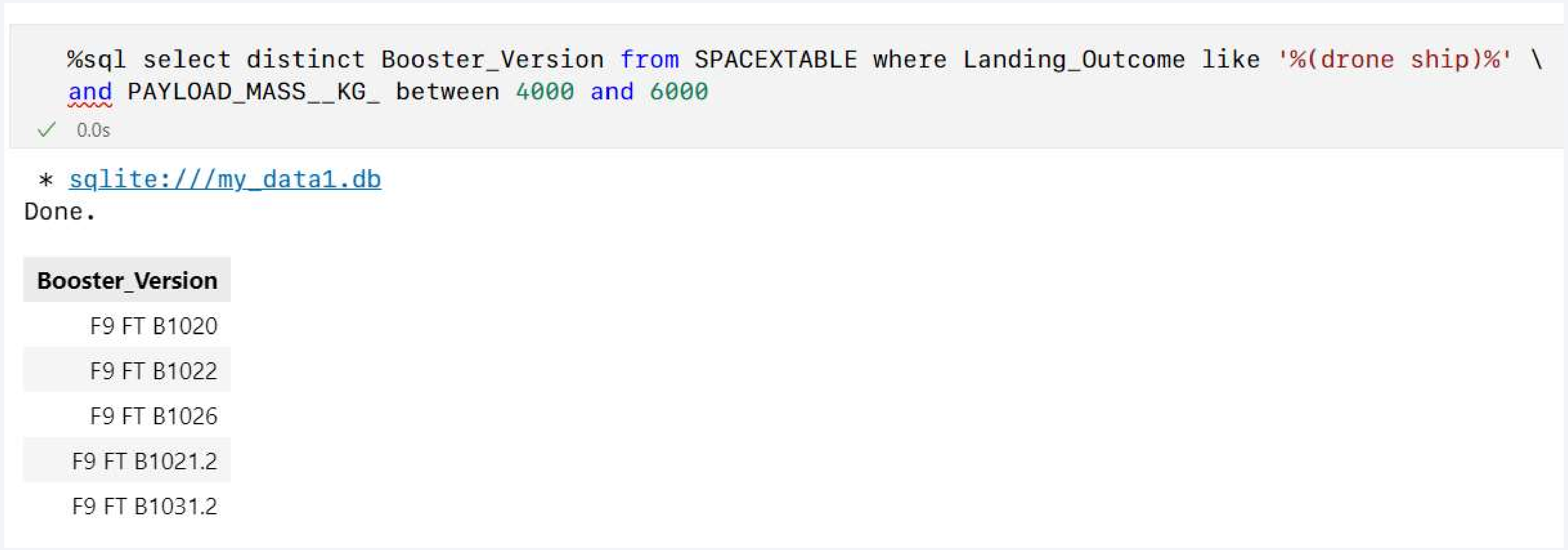
```
%sql select min(date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)'
```

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

min(date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- This is the list of the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000, using the BETWEEN statement in the WHERE condition



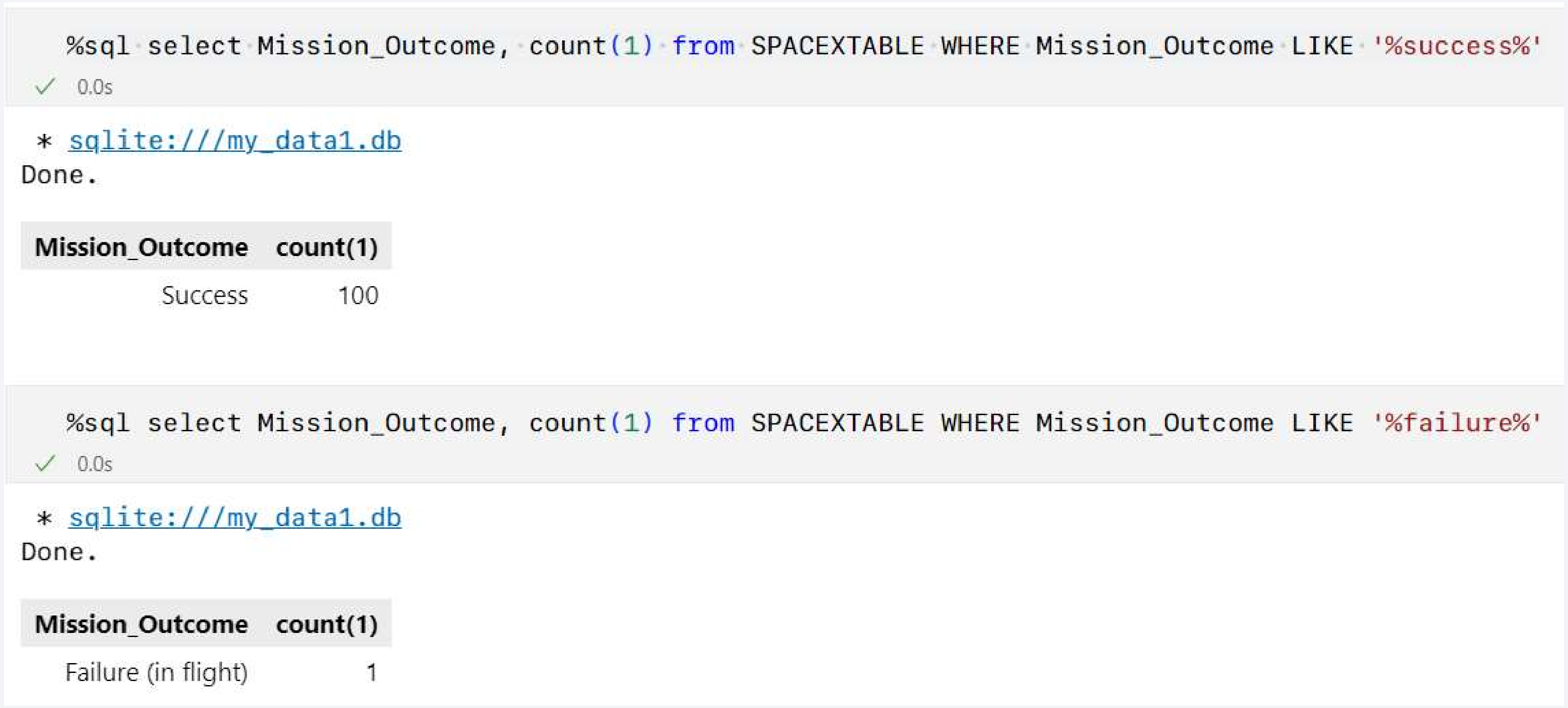
```
%sql select distinct Booster_Version from SPACEXTABLE where Landing_Outcome like '%(drone ship)%' \
and PAYLOAD_MASS__KG_ between 4000 and 6000
✓ 0.0s

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

Booster_Version
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- We calculated the total number of successful and failure mission outcomes, using the wildcard '%' in the WHERE SQL statement



```
%sql select Mission_Outcome, count(1) from SPACEXTABLE WHERE Mission_Outcome LIKE '%success%'
```

✓ 0.0s

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

Mission_Outcome	count(1)
Success	100

```
%sql select Mission_Outcome, count(1) from SPACEXTABLE WHERE Mission_Outcome LIKE '%failure%'
```

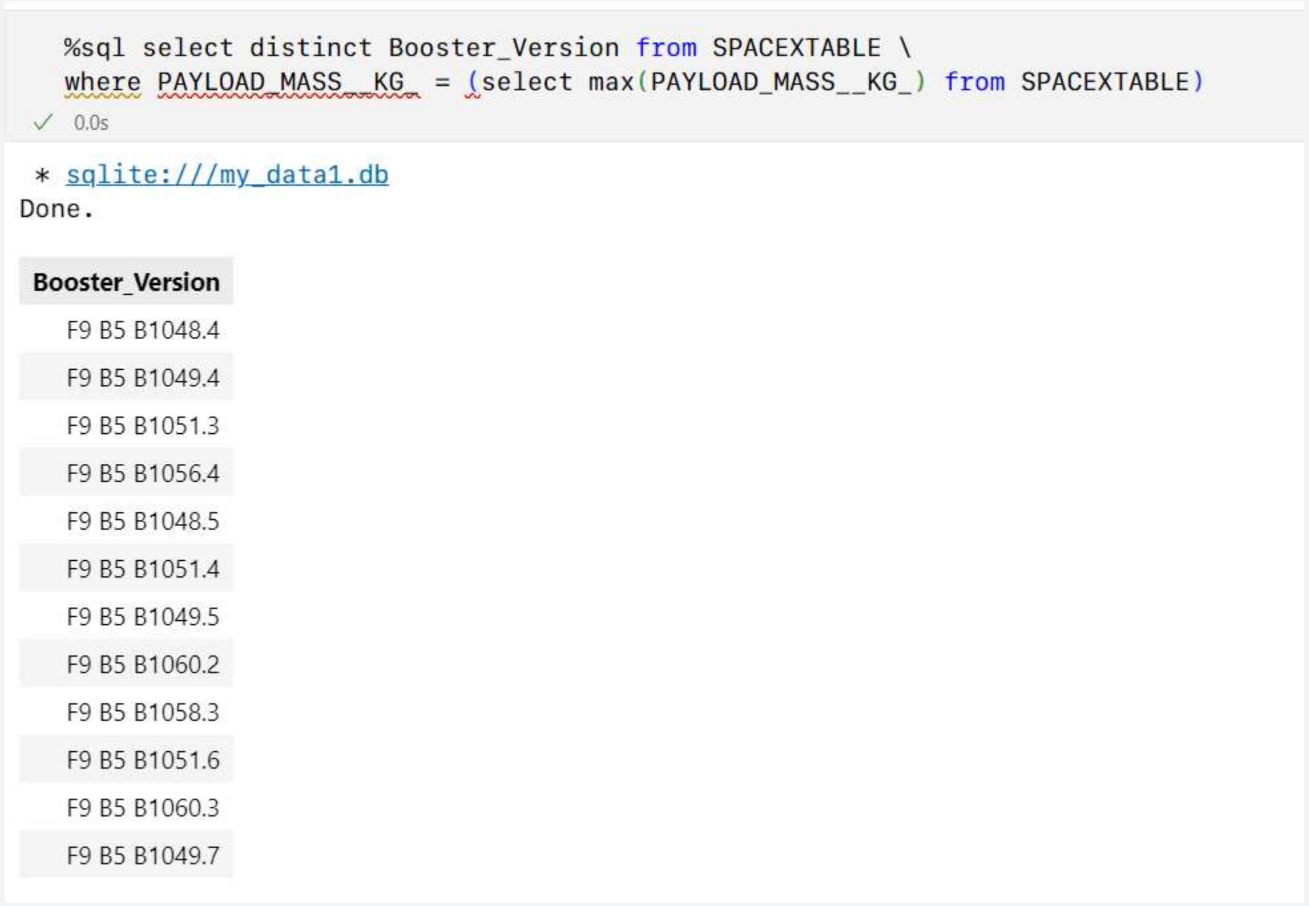
✓ 0.0s

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

Mission_Outcome	count(1)
Failure (in flight)	1

Boosters Carried Maximum Payload

- We listed the names of the booster which have carried the maximum payload mass, using a sub-query and the MAX() function



```
%sql select distinct Booster_Version from SPACEXTABLE \
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

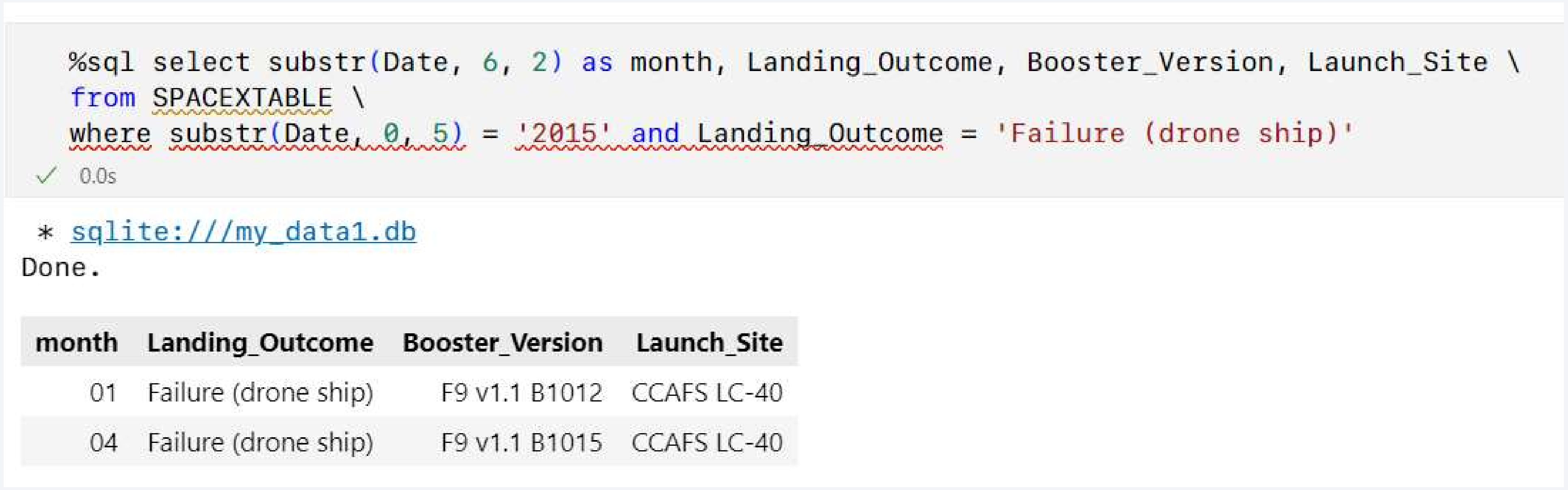
✓ 0.0s

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

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

2015 Launch Records

- We retrieved the list of the failed landing_outcomes in drone ship in the year 2015, including the booster versions, and launch site names. Since the SQLite hasn't a MounthName() function natively, we used the sustr(Date, 0, 5) to extract the year from the date field



```
%sql select substr(Date, 6, 2) as month, Landing_Outcome, Booster_Version, Launch_Site \
from SPACEXTABLE \
where substr(Date, 0, 5) = '2015' and Landing_Outcome = 'Failure (drone ship)'
```

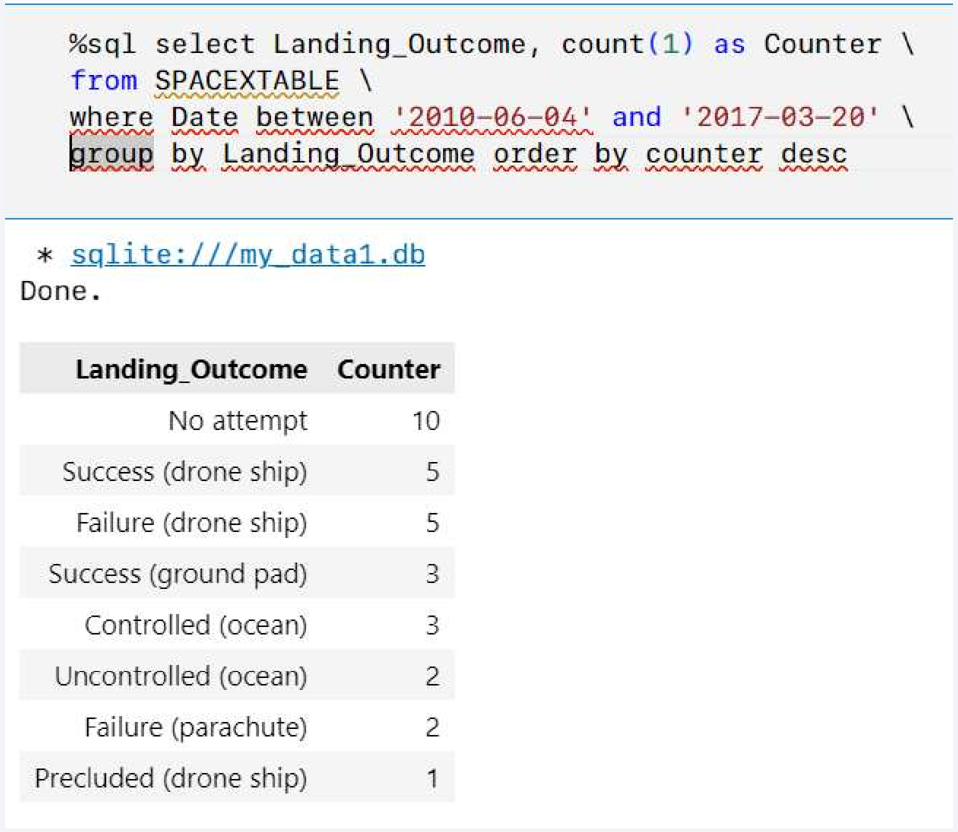
✓ 0.0s

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

month	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 ranked the count of landing outcomes between the date 2010-06-04 and 2017-03-20, sorting the list in descending order using the BETWEEN statement to filter the data and the GROUP BY statement and COUNT() function to count the number of records for each Landing_Outcome value



```
%sql select Landing_Outcome, count(1) as Counter \
from SPACEXTABLE \
where Date between '2010-06-04' and '2017-03-20' \
group by Landing_Outcome order by counter desc
```

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

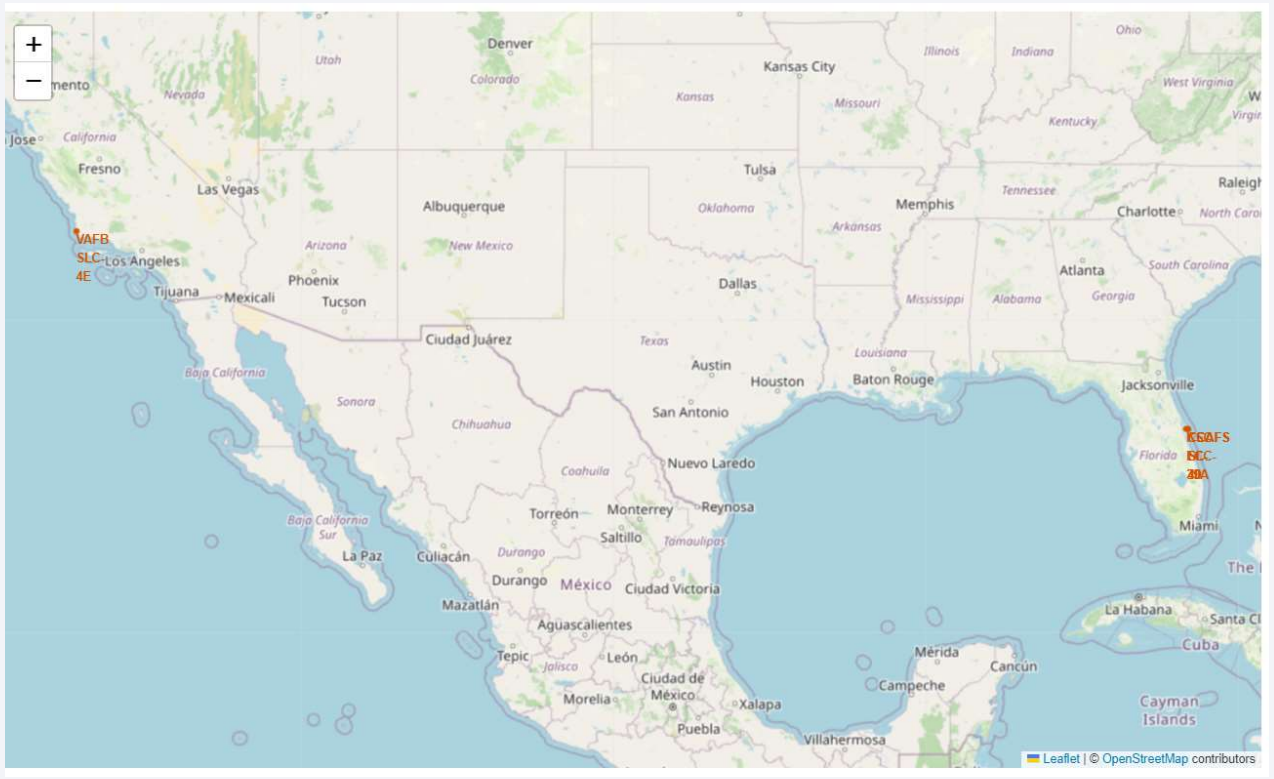
Landing_Outcome	Counter
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

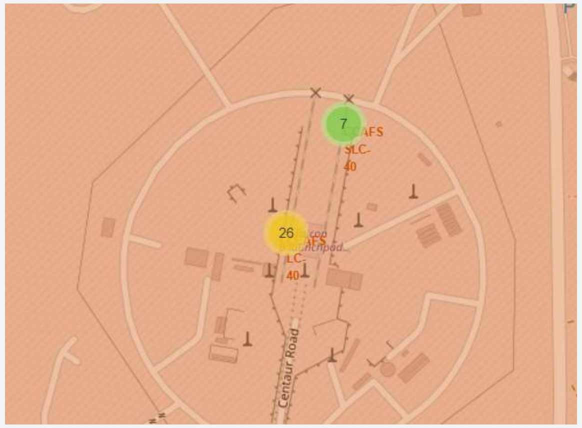
Location for all the launch sites

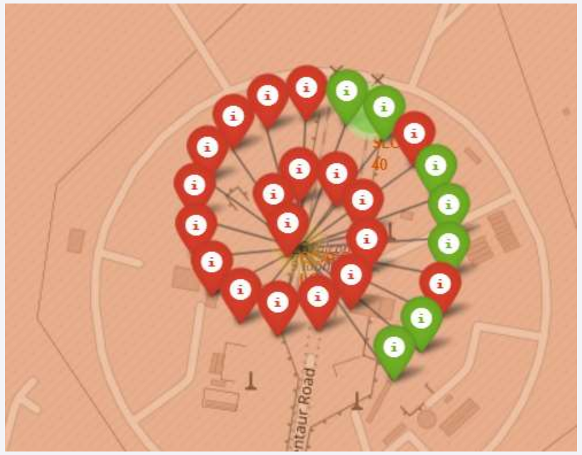
- We used the Folio map marker feature to show the location of all the launch sites on a global map
- All the launch sites are located in the United Stated, on the west and east coast



Launch site details

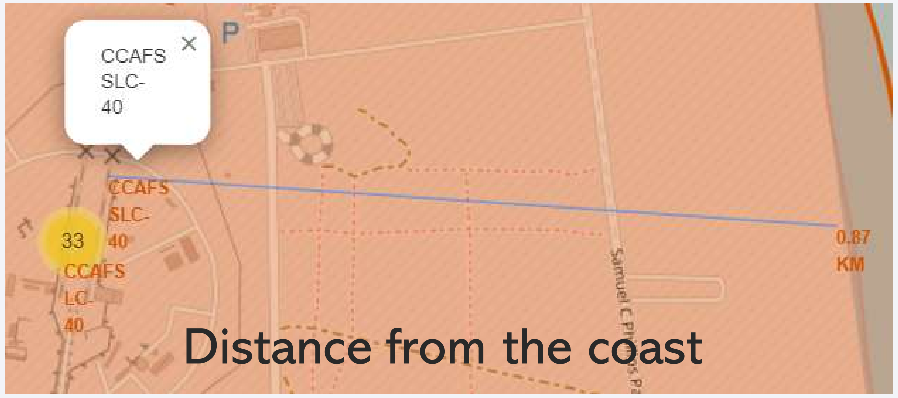
- We added additional details to the location on the global map, for each location
  - Icon indicating the overall number of launch attempts for the specific launch site
  - The icon representation for each launch with:
    - A green icon in case of a successful result
    - A red icon in case of an unsuccessful result



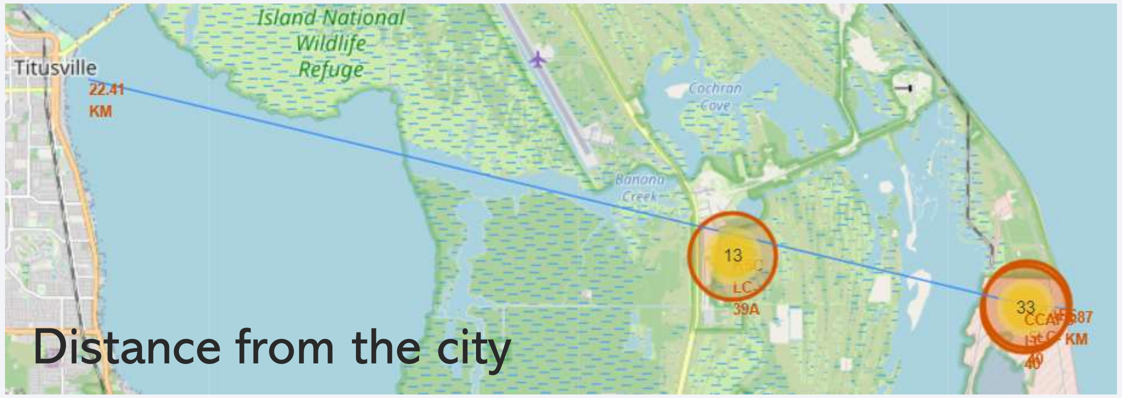


Launch sites and proximity with other sites

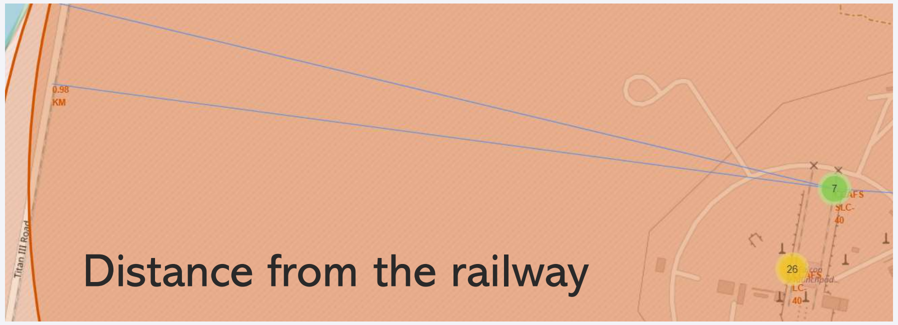
- Considering the launch site CCAFS SLC-40, we measured distances from relevant sites on the map



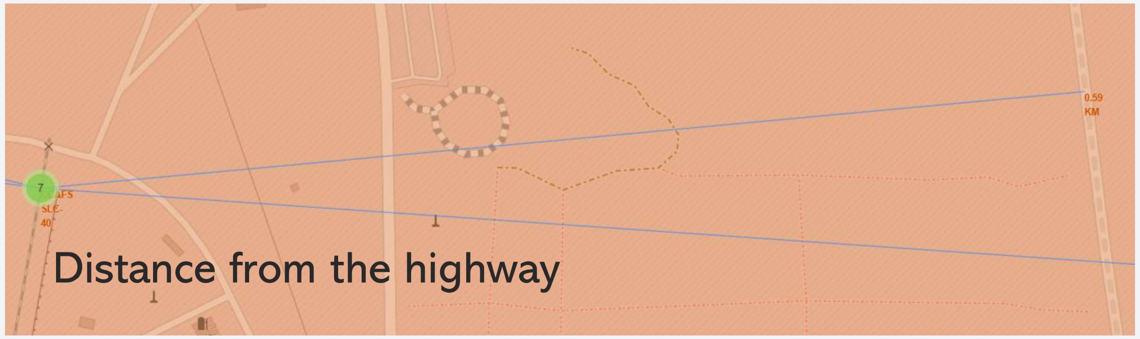
Distance from the coast



Distance from the city



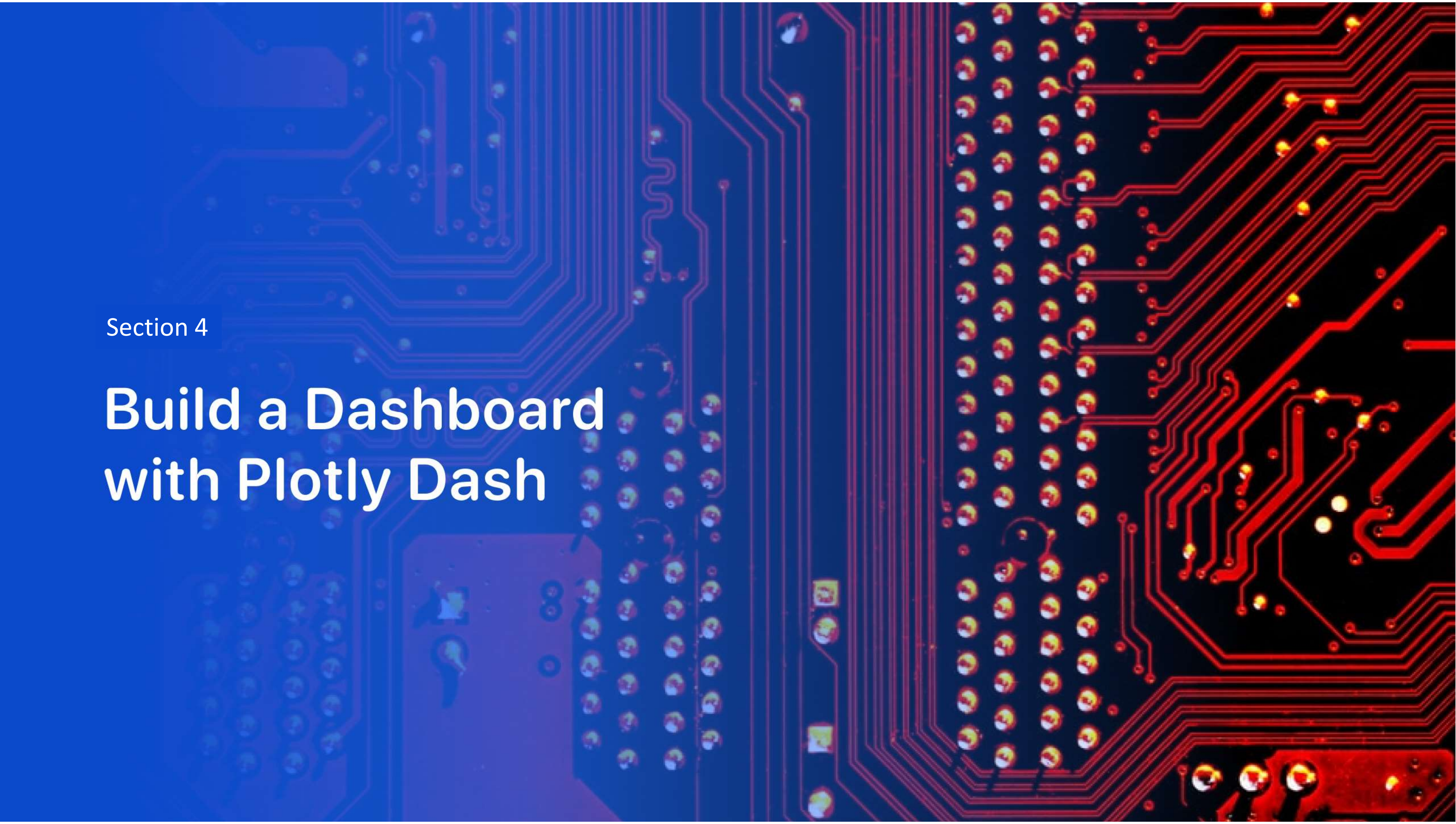
Distance from the railway



Distance from the highway

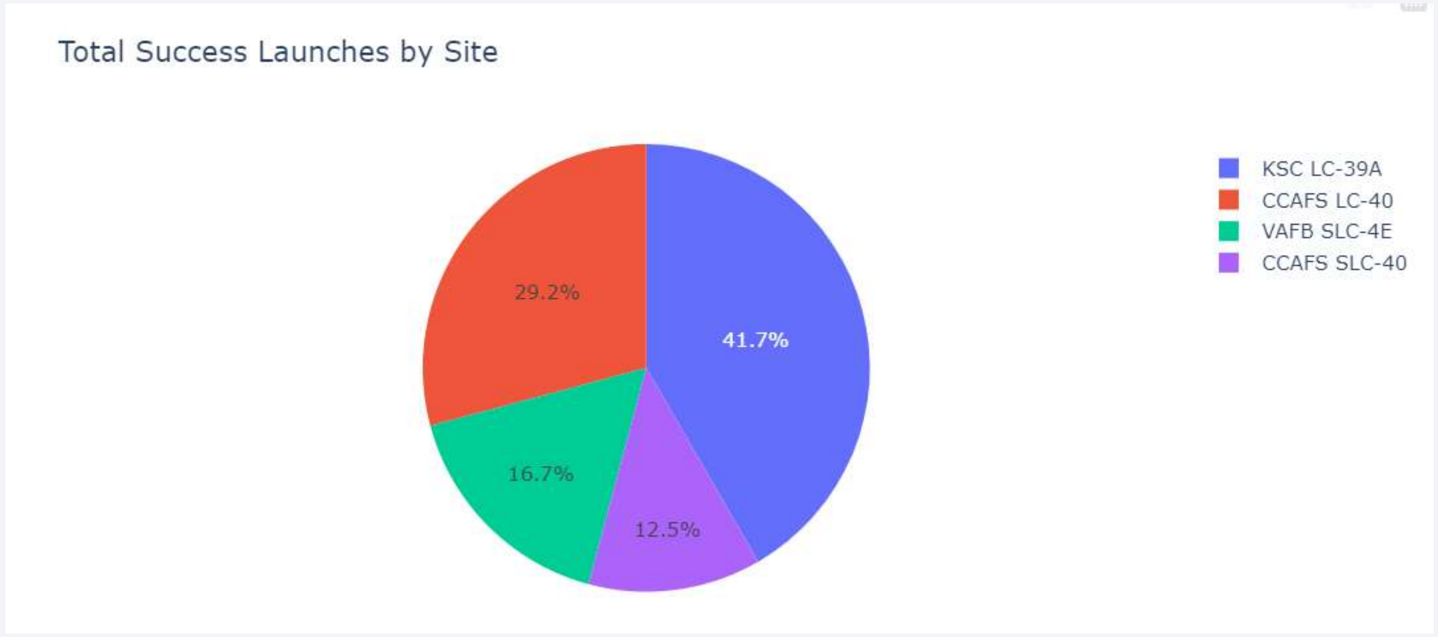
Section 4

Build a Dashboard with Plotly Dash



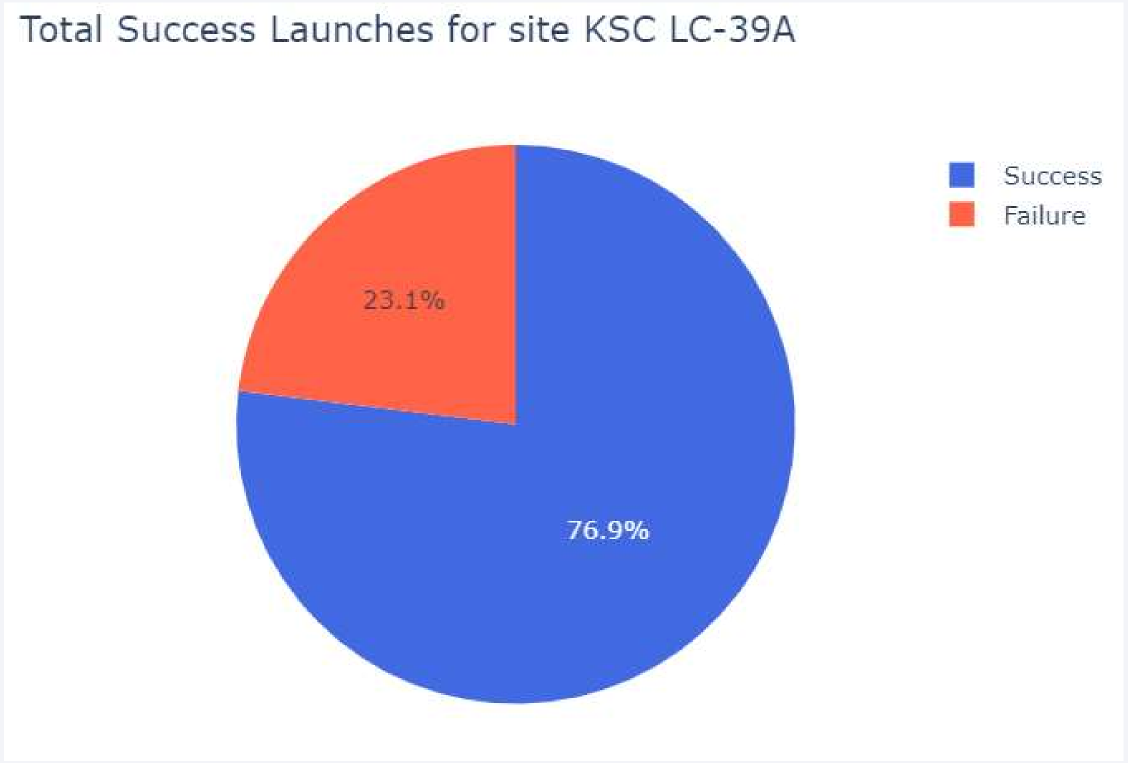
Success ratio for each launch site

We can clearly see that the launch site KSC LC-39A has the highest successful rate from all the sites



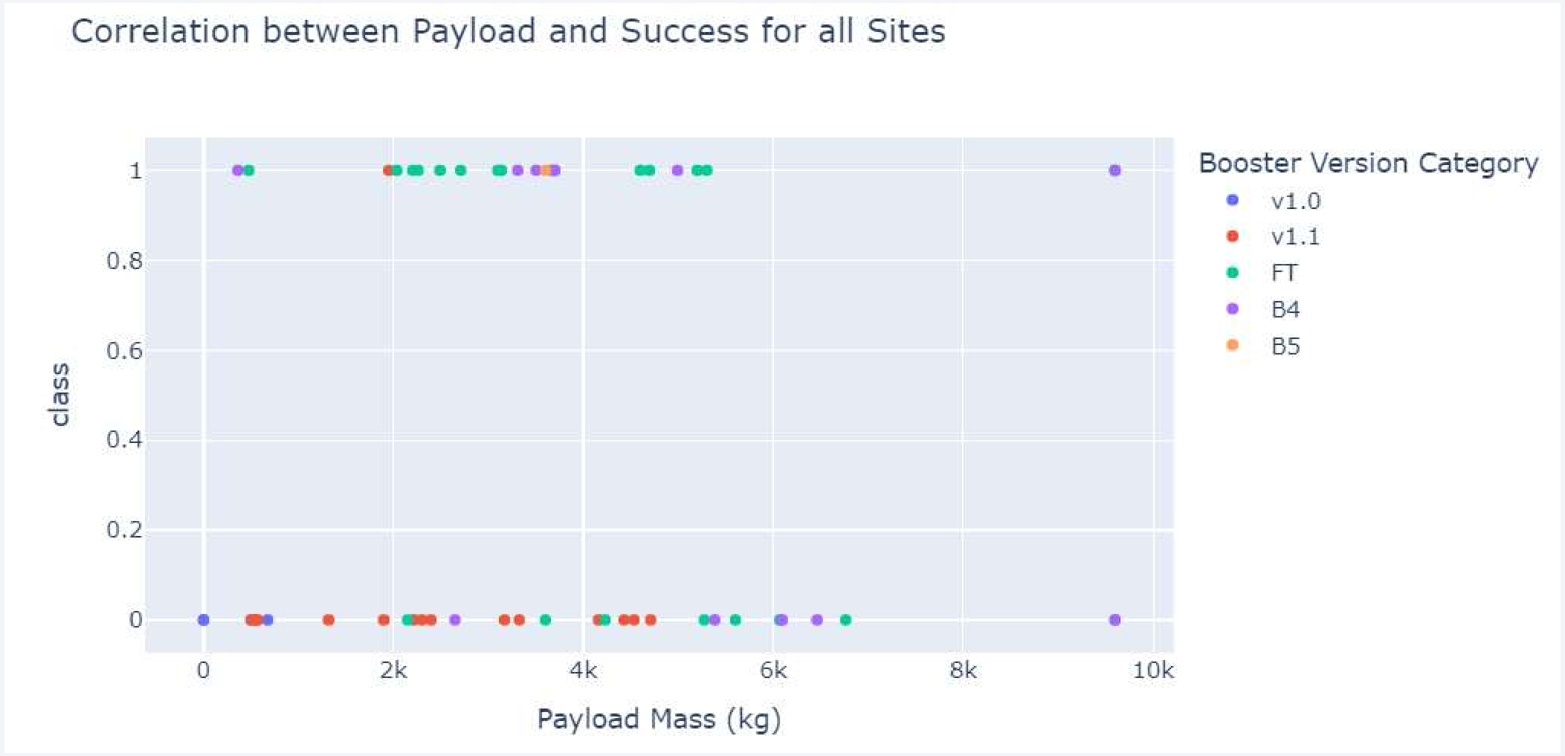
Success launch rate for the KSC LC-39A site

- The launch site KSC LC-39A achieved a 76.9% rate of successful launches while getting a 23.1% failure rate



Payload range and launch outcome

- The payload range 2000 kg – 6000 kg has higher success rate than other ranges
- We have too few data to properly evaluate the success rate with heavyweight oalyloads (>9000kg)

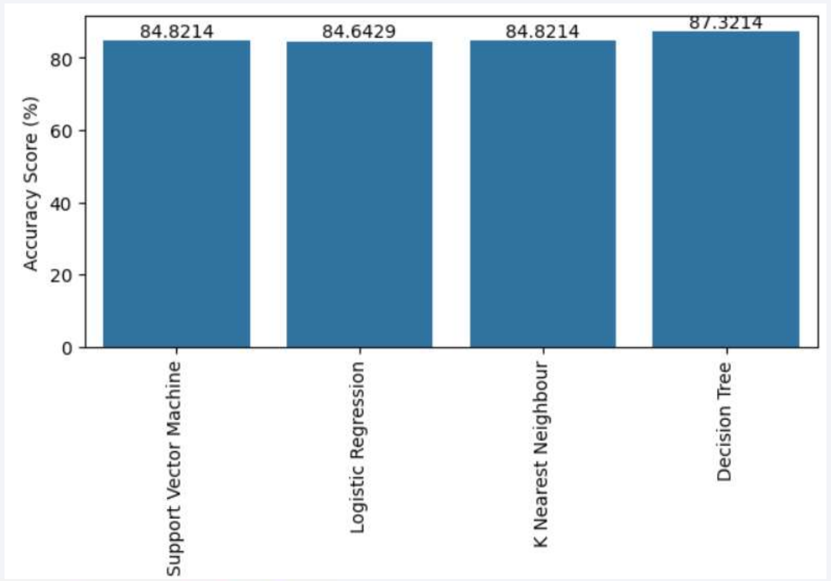


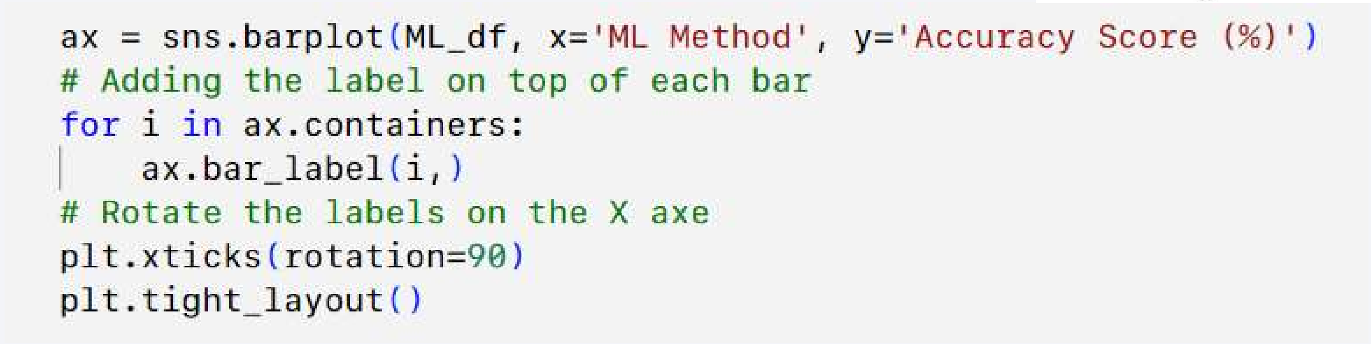
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Comparison of the accuracy for each model using a bar chart
- The Decision Tree model shows the highest accuracy

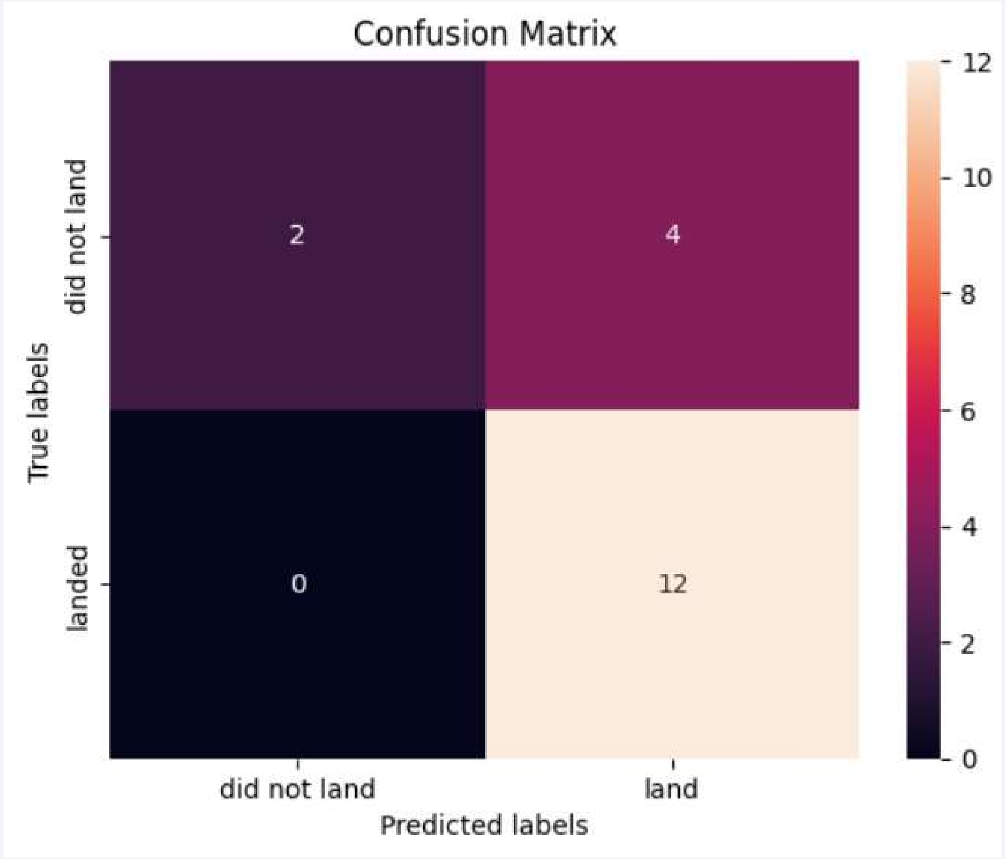




```
ax = sns.barplot(ML_df, x='ML Method', y='Accuracy Score (%)')
# Adding the label on top of each bar
for i in ax.containers:
    ax.bar_label(i,)
# Rotate the labels on the X axe
plt.xticks(rotation=90)
plt.tight_layout()
```

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. As you can see in the confusion matrix below, the major problem is the false: unsuccessful landing marked as successful landing by the classifier.



Conclusions

- The Tree Classifier Algorithm is the best Machine Learning approach for this dataset.
- The low weighted payloads (which define as 4000kg and below) performed better than the heavy weighted payloads.
- Starting from the year 2013, the success rate for SpaceX launches is increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.
- KSC LC-39A have the most successful launches of any sites; 76.9%
- SSO orbit have the most success rate; 100% and more than 1 occurrence

Thank you!