IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Aliaksandr Karpovich

12/10/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Collecting data via REST API and Web Scrapping
  - Exploratory Data Analysis through visualisation
  - EDA with SQL
  - Interact with Map, Markers by using Folium
  - Interactive controls with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - EDA observations
  - Geo-data observations
  - Graphs with interactive controls
  - Predicted results

Introduction

- Project background and context
  - The idea of the project is to predict outcome of Falcon 9 landing. Since launch of the Falcon 9 rocket costs 62 million dollars it's reasonable to use different attributes and params to predict its landing and by doing this determine the cost of a launch.
- Problems you want to find answers
  - What attributes make landing successful?
  - How each of the attributes affect on landing result?
  - What launch sites allow SpaceX to achieve the best success rate?

Section 1

Methodology

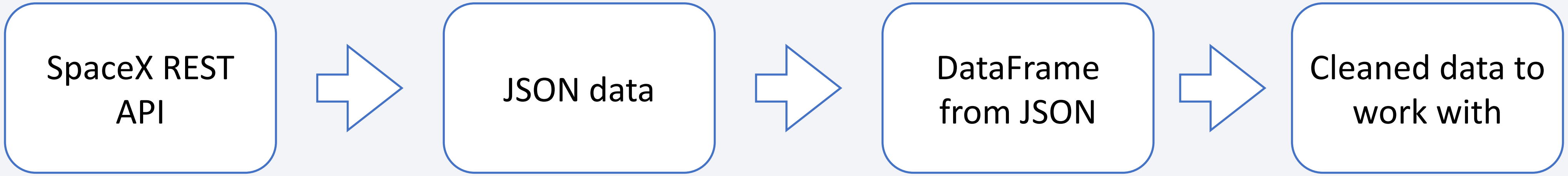
Methodology

Executive Summary

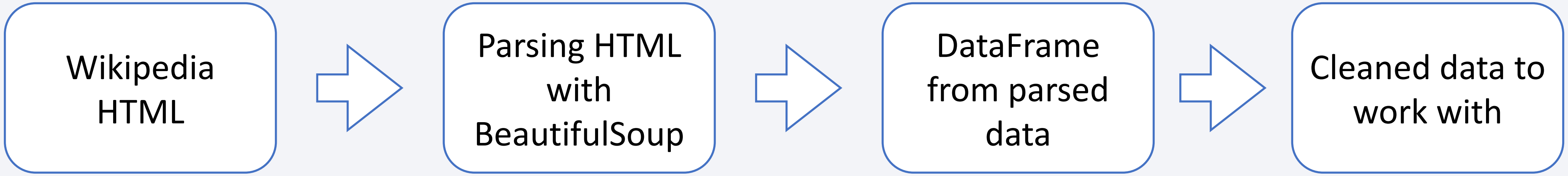
- Data collection methodology:
  - SpaceX REST API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Dropping unnecessary columns
  - One Hot Encoding for categorical attributes
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The info obtained by the SpaceX REST API using the endpoint api.spacexdata.com/v4/ is rocket, launches, payload mass.

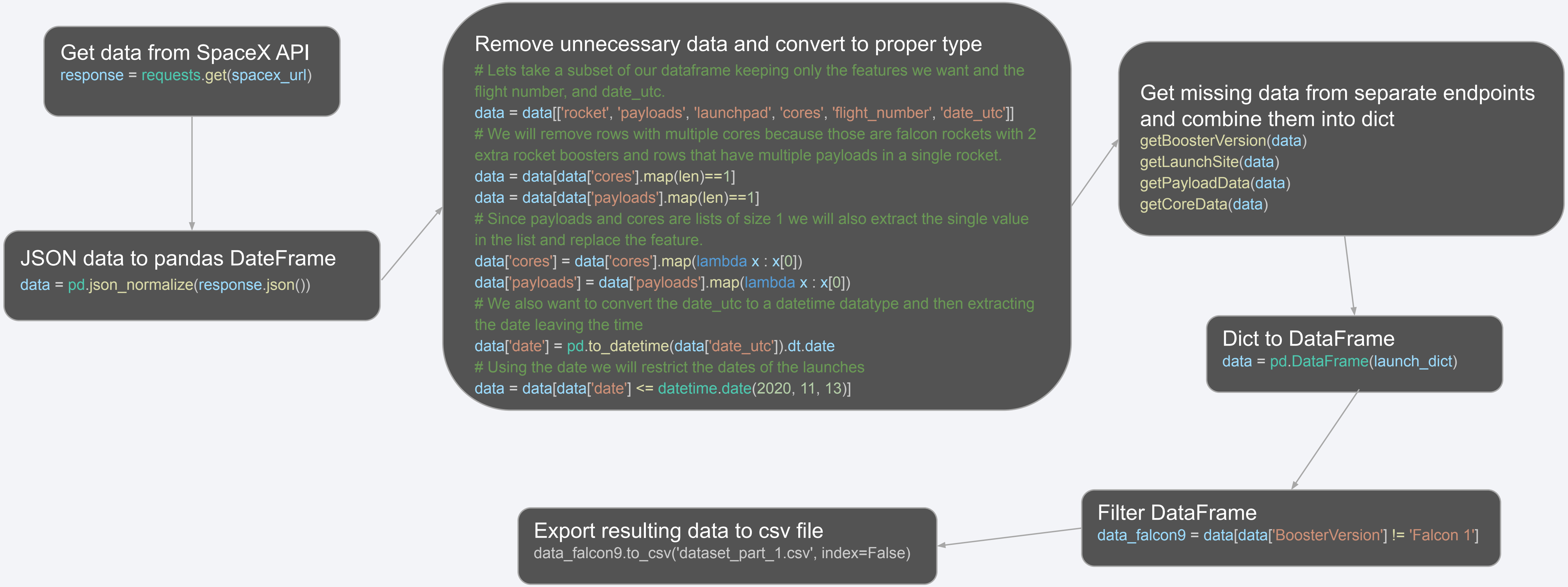


- The info obtained from Wikipedia *https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922* by using web scrapping.


Wikipedia HTML
Parsing HTML with BeautifulSoup
DataFrame from parsed data
Cleaned data to work with

Data Collection – SpaceX API

Collecting data from SpaceX API:



Get data from SpaceX API

```
response = requests.get(spacex_url)
```

JSON data to pandas DateFrame

```
data = pd.json_normalize(response.json())
```

Remove unnecessary data and convert to proper type

```
# 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)]
```

Get missing data from separate endpoints and combine them into dict

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

Dict to DataFrame

```
data = pd.DataFrame(launch_dict)
```

Filter DataFrame

```
data_falcon9 = data[data['BoosterVersion'] != 'Falcon 1']
```

Export resulting data to csv file

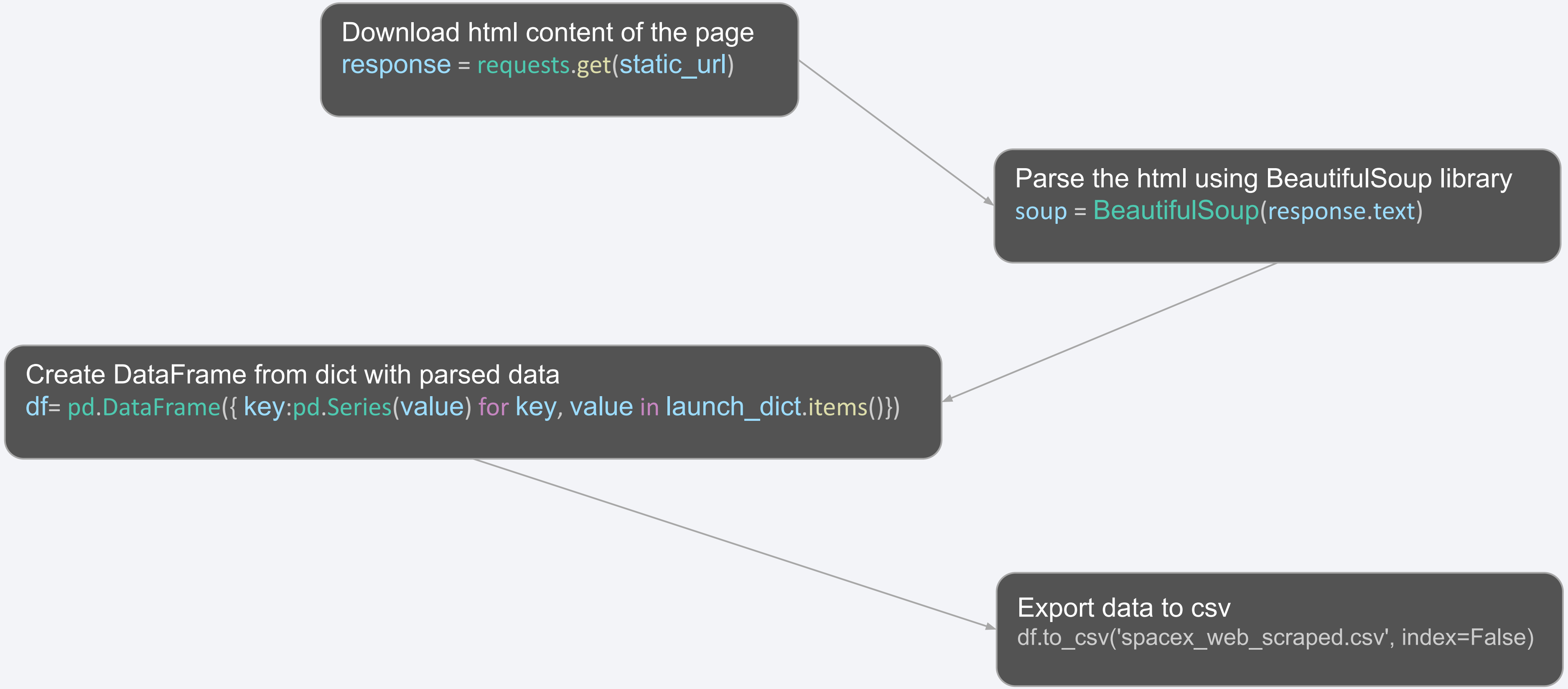
```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

[Link to Github with the notebook]()

Data Collection - Scraping

Collecting Data by using web scrapping

static_url = “https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"



Download html content of the page

```
response = requests.get(static_url)
```

Parse the html using BeautifulSoup library

```
soup = BeautifulSoup(response.text)
```

Create DataFrame from dict with parsed data

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items()})
```

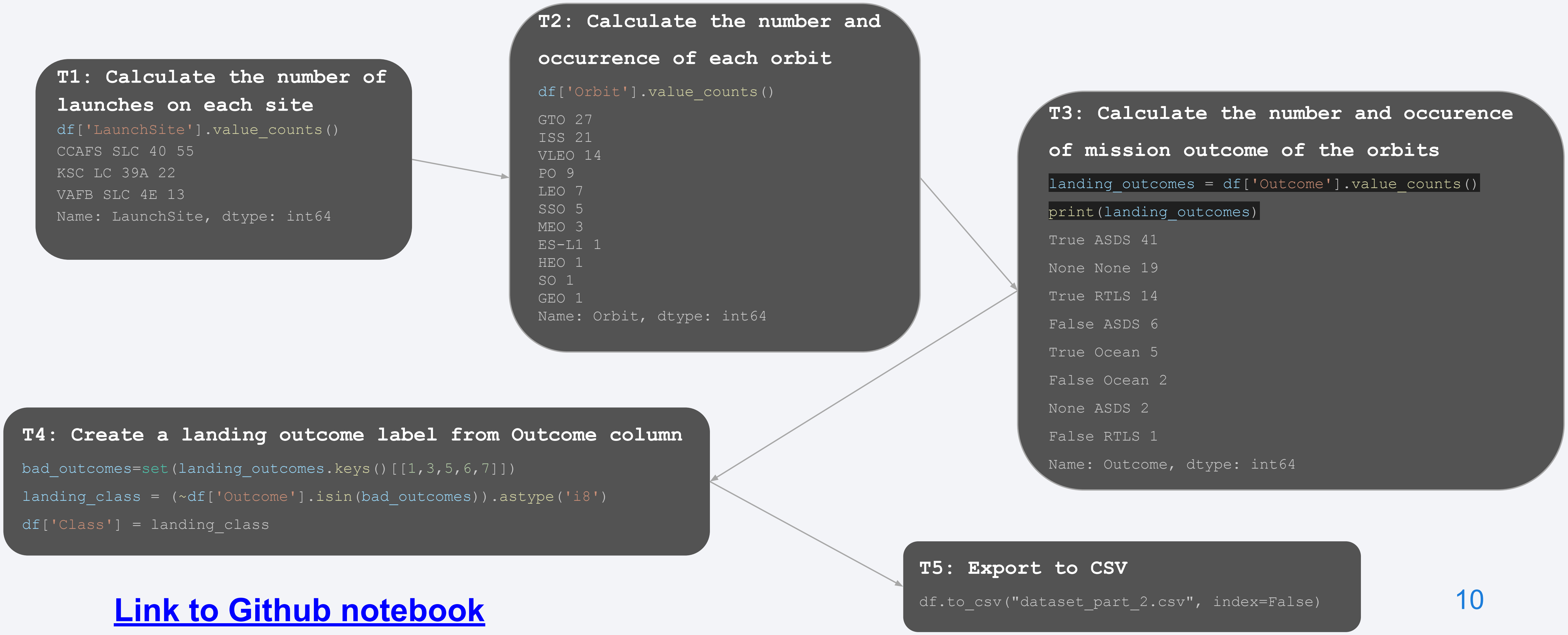
Export data to csv

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Link to Github with the notebook

Data Wrangling

- The Outcome string variable has to be transformed into categorical variable which equals to 1 if land is successful and 0 otherwise.



T1: Calculate the number of launches on each site

```
df['LaunchSite'].value_counts()
CCAFS SLC 40 55
KSC LC 39A 22
VAFB SLC 4E 13
Name: LaunchSite, dtype: int64
```

T2: Calculate the number and occurrence of each orbit

```
df['Orbit'].value_counts()

GTO 27
ISS 21
VLEO 14
PO 9
LEO 7
SSO 5
MEO 3
ES-L1 1
HEO 1
SO 1
GEO 1
Name: Orbit, dtype: int64
```

T3: Calculate the number and occurence of mission outcome of the orbits

```
landing_outcomes = df['Outcome'].value_counts()
print(landing_outcomes)
True ASDS 41
None None 19
True RTLS 14
False ASDS 6
True Ocean 5
False Ocean 2
None ASDS 2
False RTLS 1
Name: Outcome, dtype: int64
```

T4: Create a landing outcome label from Outcome column

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
landing_class = (~df['Outcome'].isin(bad_outcomes)).astype('i8')
df['Class'] = landing_class
```

T5: Export to CSV

```
df.to_csv("dataset_part_2.csv", index=False)
```

Link to Github notebook

EDA with Data Visualization

Scatter Graphs

- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Orbit vs. Flight Number
- Payload vs. Orbit Type
- Orbit vs. Payload Mass

shows the relationship between two quantitative variables measured for the same individuals

Bar Graph

- Success rate vs. Orbit

shows the relationship between a numeric and a categorical variable

Line Graph

- Success rate vs. Year

shows data along a number line with Xs or dots recorded above the responses to indicate the number of occurrences a response appears in the data set

Link to Github notebook

EDA with SQL

- **We performed SQL queries to gather and understand data from dataset:**
  - Displaying the names of the unique launch sites in the space mission.
  - Display 5 records where launch sites begin with the string 'CCA'
  - Display the total payload mass carried by boosters launched by NASA (CRS).
  - Display average payload mass carried by booster version F9 v1.1.
  - List the date when the first succesful landing outcome in ground pad was acheived.
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
  - List the total number of successful and failure mission outcomes.
  - List the names of the booster_versions which have carried the maximum payload mass.
  - 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.
  - 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.

Link to Github notebook

Build an Interactive Map with Folium

Completed tasks:

- Created a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.
  nasa_coordinate = [29.559684888503615, -95.0830971930759]
  site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
- Created and added `folium.Circle` and `folium.Marker` for each launch site on the site map
- For each launch result in `spacex_df` data frame, added a `folium.Marker` to `marker_cluster`
- *Calculate the distances between a launch site to its proximities*
- *Added* a `MousePosition` on the map to get coordinate for a mouse over a point on the map

```
formatter = "function(num) {return L.Util.formatNum(num, 5);};"
mouse_position = MousePosition(
    position='topright',
    separator=' Long: ',
    empty_string='NaN',
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)
site_map.add_child(mouse_position)
```

- *Mark* down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site
- *Draw* a `PolyLine` between a launch site to the selected coastline point
- *Created a marker with distance to a closest railway and drew a line between the marker to the launch site*

It helps to answer such questions as:

- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities?

Link to Github notebook

Build a Dashboard with Plotly Dash

Dashboard has dropdown, pie chart, rangeslider and scatter plot components:

- Dropdown allows a user to choose either specific launch site or ALL.
- Pie chart shows either success percentage among ALL launch sites or success/failure percentage for specific launch site.
- Rangeslider allows a user to control a payload mass range.
- Scatter chart shows the relationship between Success vs Payload Mass.

Link to Github file

Predictive Analysis (Classification)

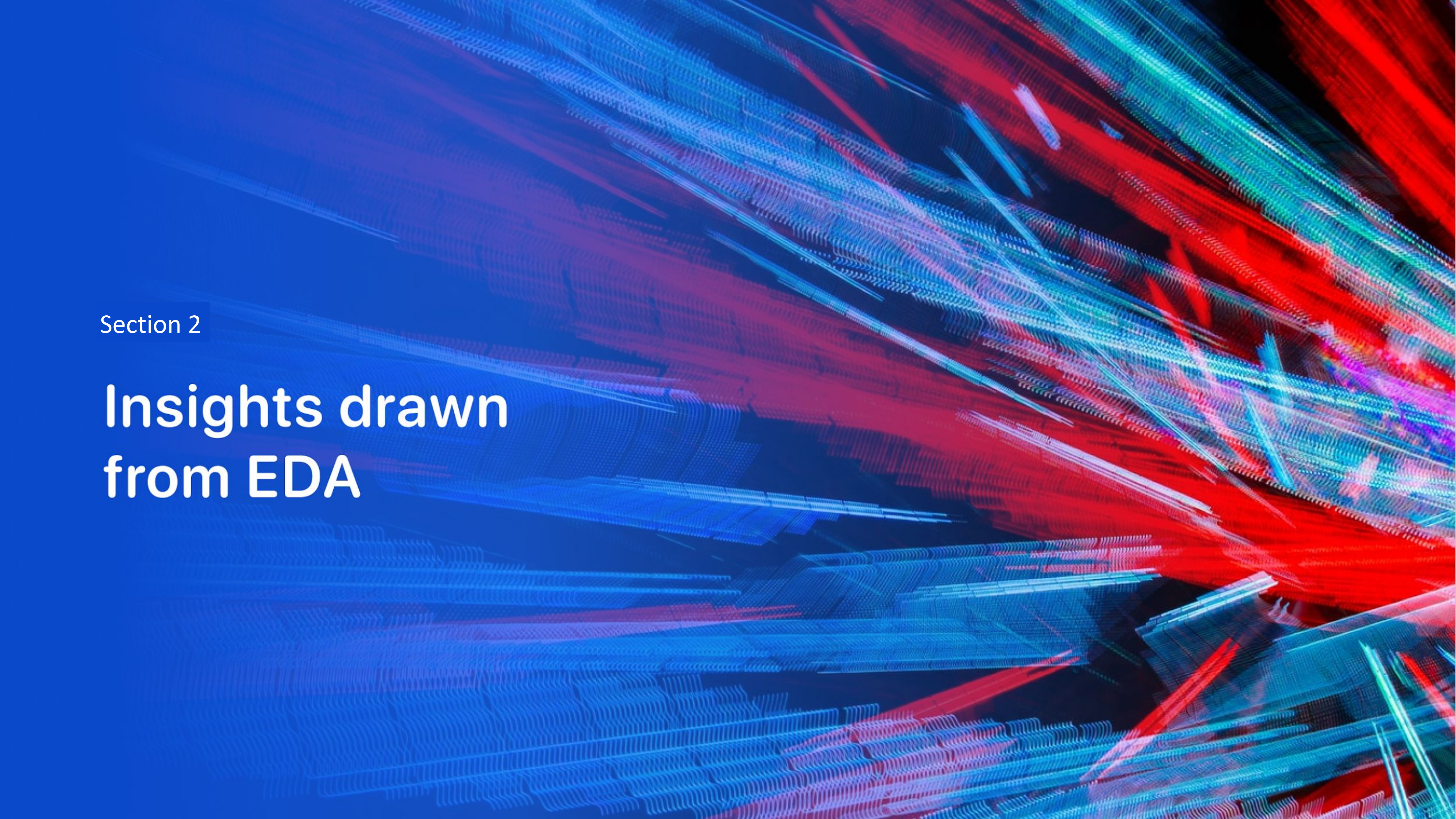
Performed steps to find the best model and its params:

- Load data
- Pick X features and Y labels variables
- Standardize X variables
- Split data into train and test subsets
- Select the best hyperparameters to describe training data using GridSearchCV and the following models:
  - Logistic regression
  - Support vector machine
  - Decision tree classifier
  - K nearest neighbors
- Compare models on test data

Link to Github notebook

Results

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

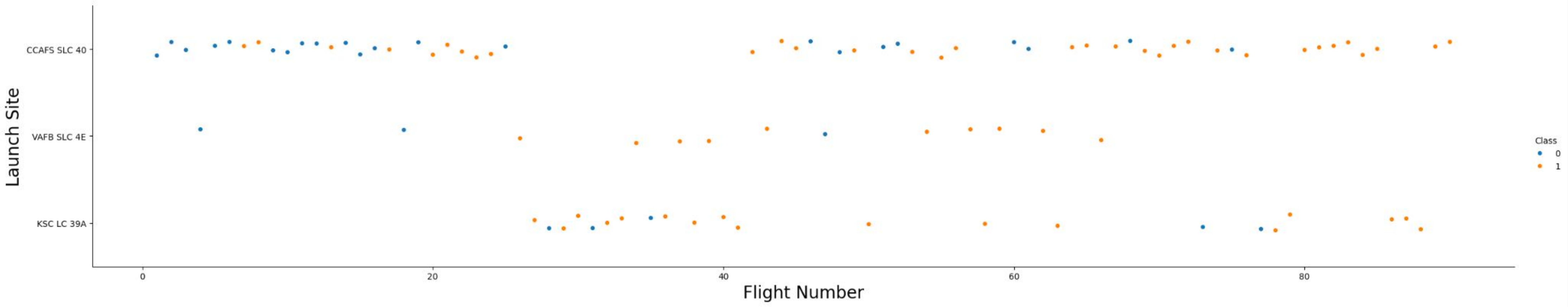


Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

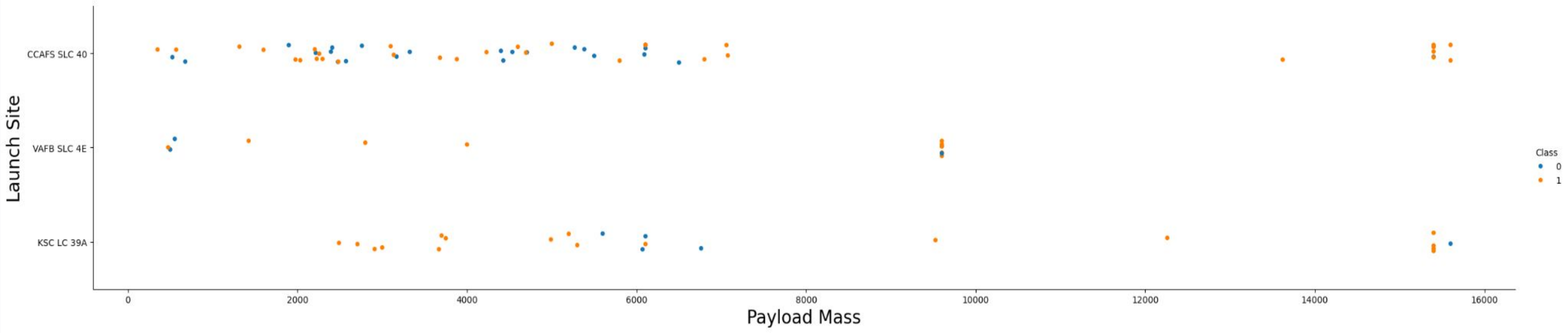
- Show a scatter plot of Flight Number vs. Launch Site



- We observe that with bigger FlightNumber success rate also increases.

Payload vs. Launch Site

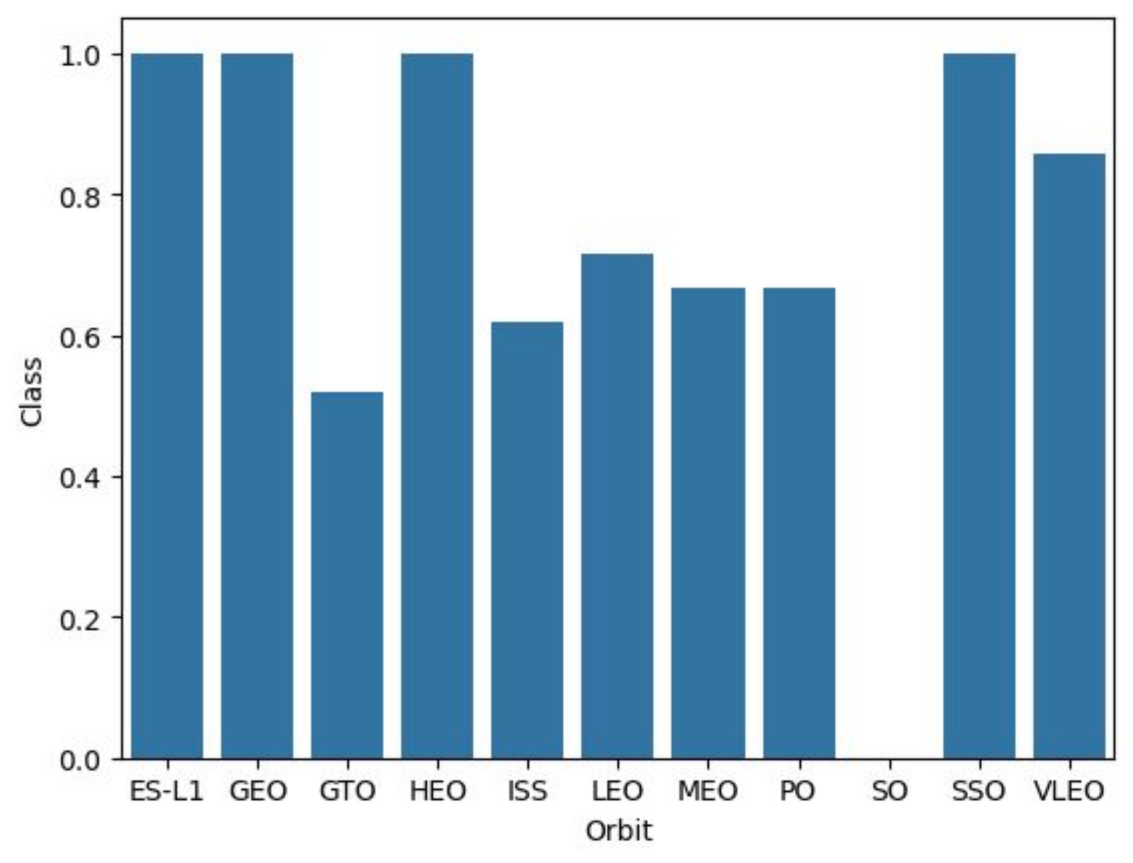
- Show a scatter plot of Payload vs. Launch Site



- Heavier payload may lead to successful landing but if it’s too heavy it leads to failure.

Success Rate vs. Orbit Type

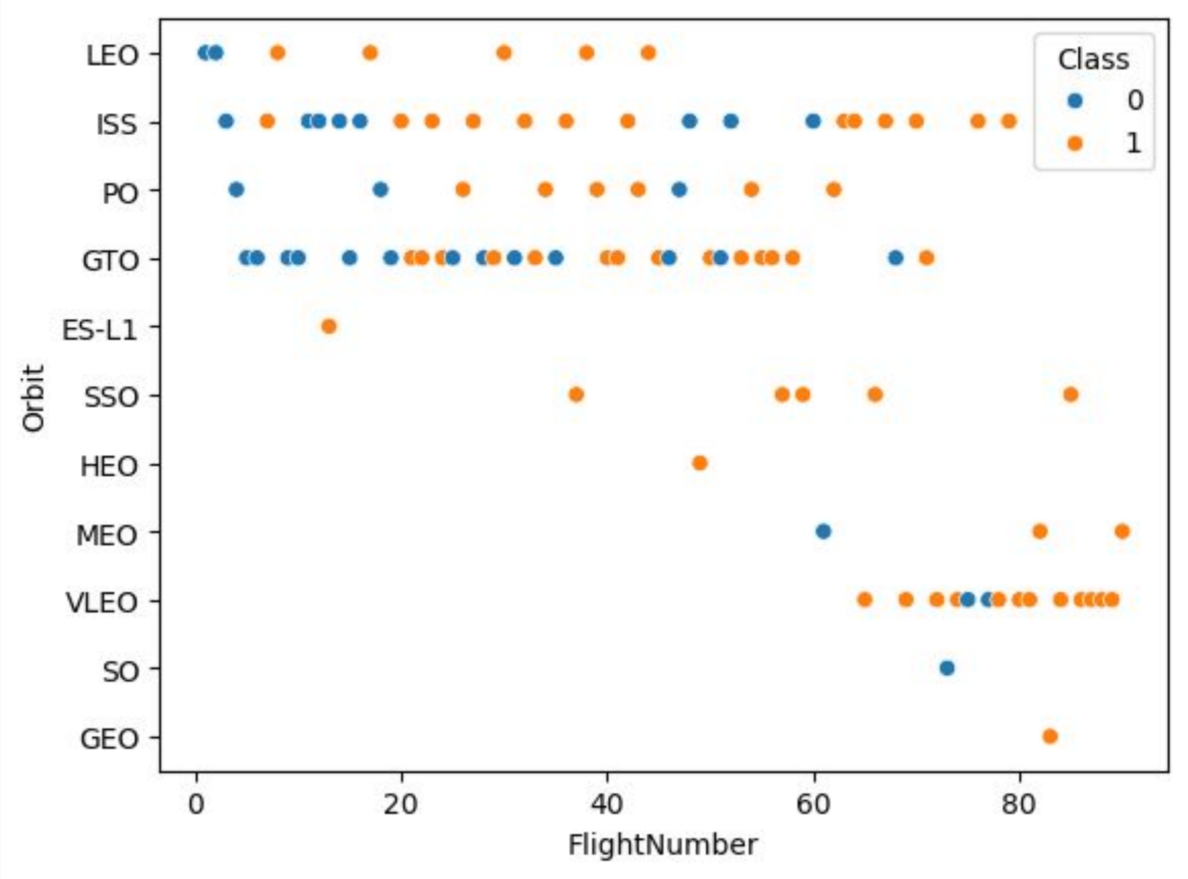
- Show a bar chart for the success rate of each orbit type



From the bar chart we can see that ES-L1, GEO, HEO, SSO have the best success rate.

Flight Number vs. Orbit Type

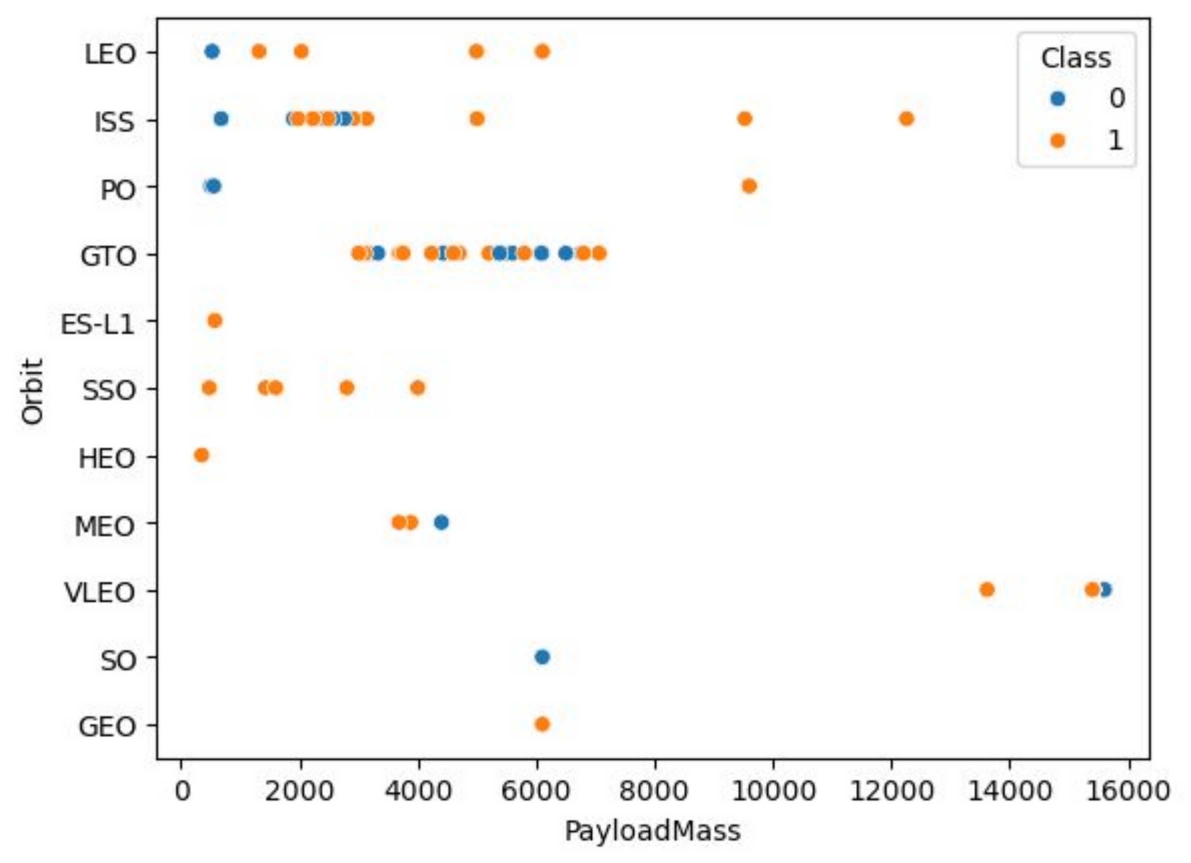
- Show a scatter point of Flight number vs. Orbit type



Success rate increases with the greater flight number but there are such orbits as GTO for example, where no relations with flight number found.

Payload vs. Orbit Type

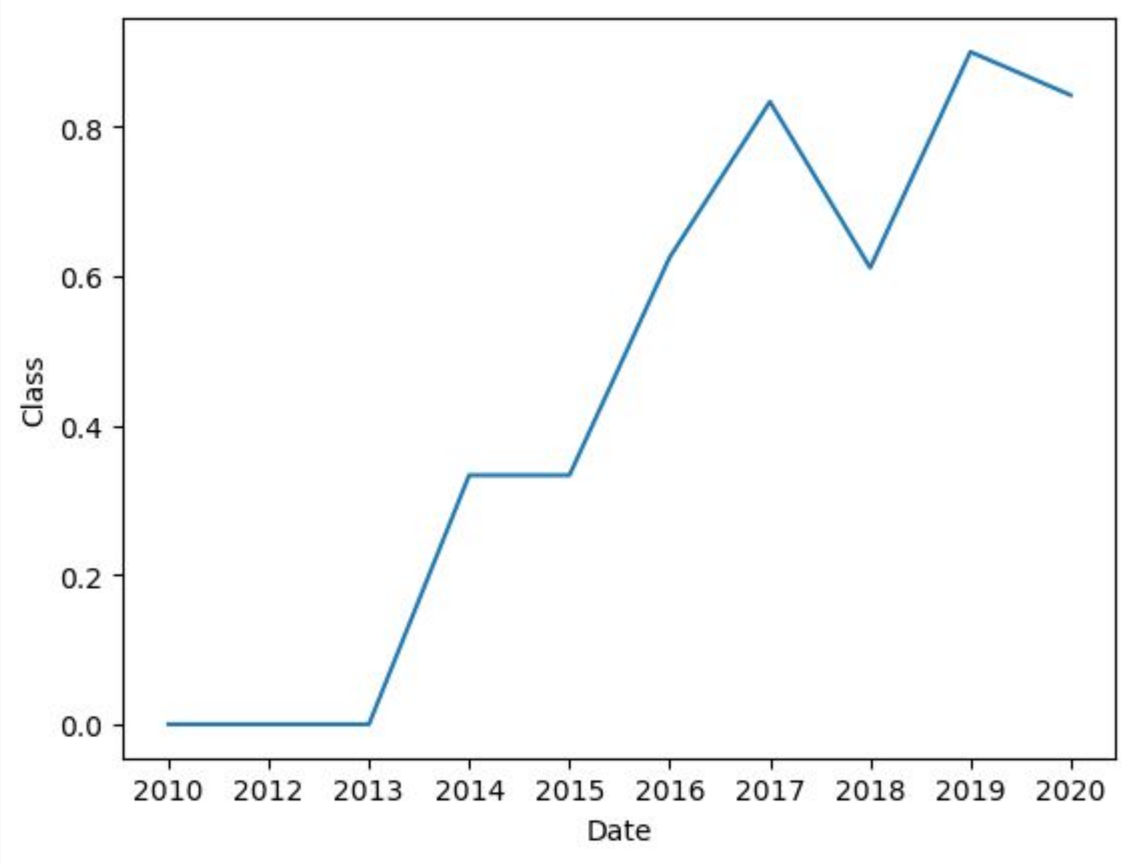
- Show a scatter point of payload vs. orbit type



Payload has great correlation with certain Orbit types. For example LEO, ISS have greater success rate with greater payload mass but for GTO, MEO it doesn’t work.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate



Since 2013 success rate keeps increasing with little drops.

All Launch Site Names

SQL Query:

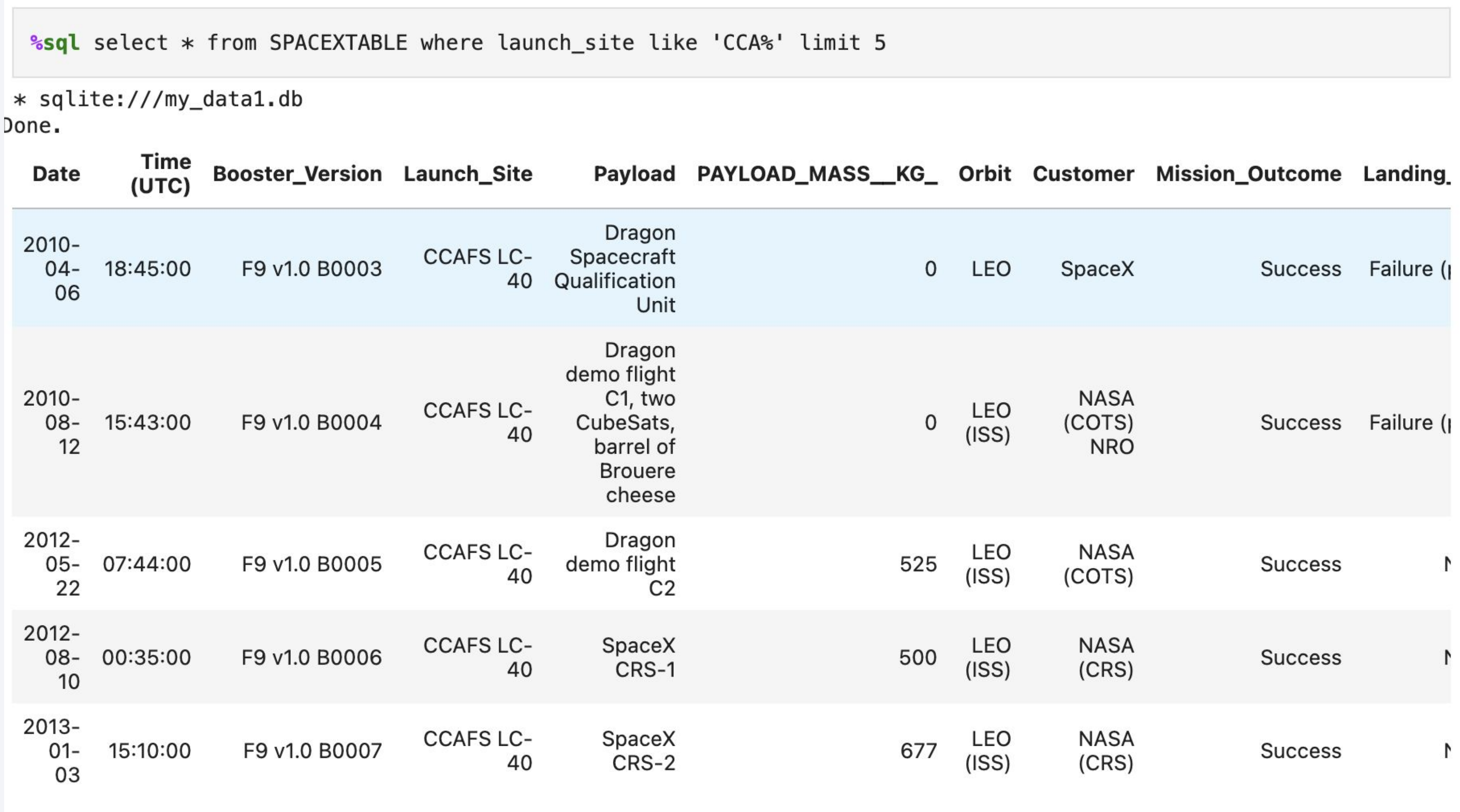
select distinct launch_site from SPACEXTABLE

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

Explanation

It runs through all launch_site records and returns unique with DISTINCT operator.

Launch Site Names Begin with 'CCA'



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

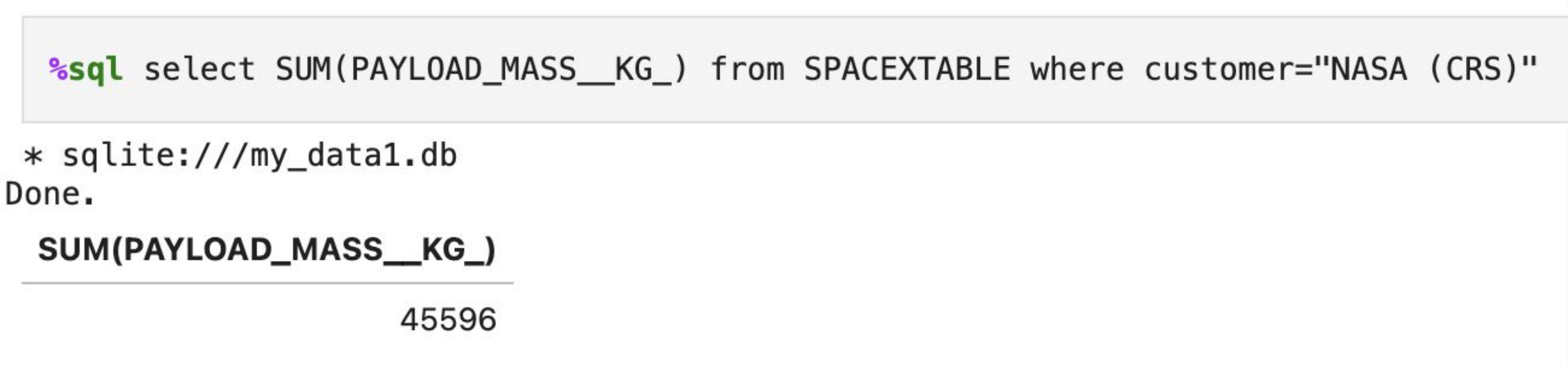
```
* sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (I
2010-08-12	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 (I
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Explanation

It filters records by launch_site to start with CCA and limits the output to 5 records

Total Payload Mass



```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEXTABLE where customer="NASA (CRS)"
```

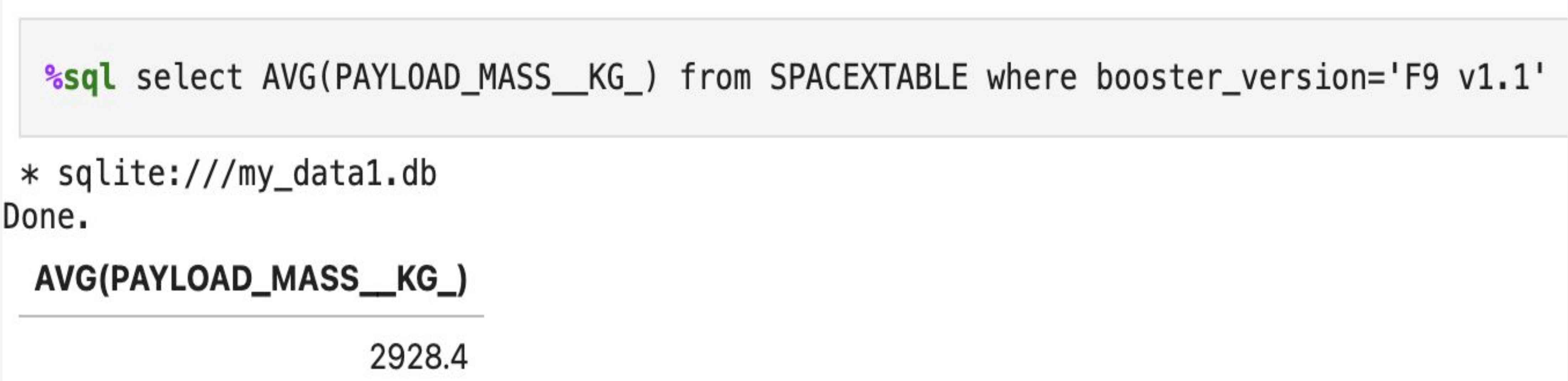
```
 * sqlite:///my_data1.db
Done.
```

SUM(PAYLOAD_MASS__KG_)
45596

Explanation:

It sums payload_mass of all NASA (CRS) customers.

Average Payload Mass by F9 v1.1



```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTABLE where booster_version='F9 v1.1'
```

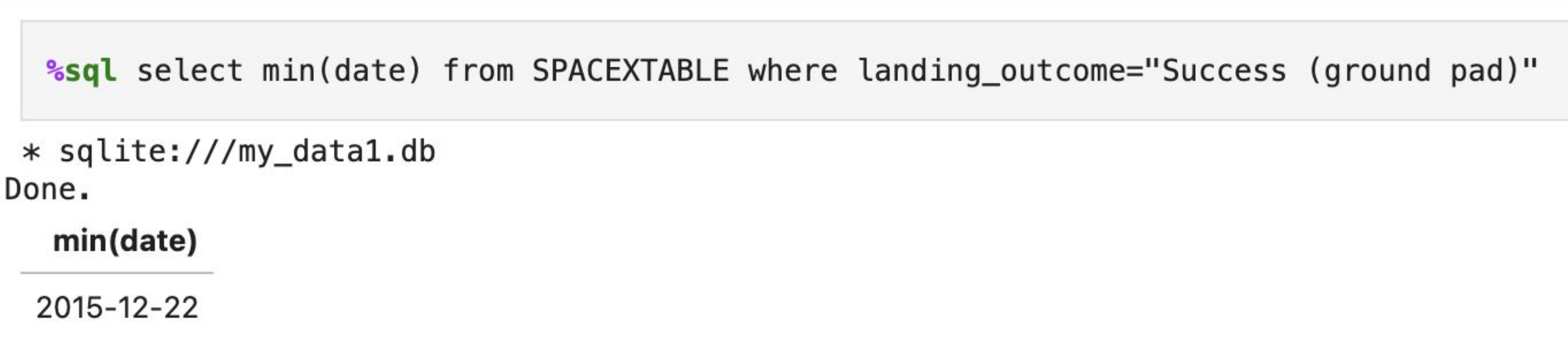
```
 * sqlite:///my_data1.db
Done.
```

```
 AVG(PAYLOAD_MASS__KG_)
------------------------
                  2928.4
```

Explanation:

It filters records by booster_version to be ‘F9 v1.1’ and calculates average of payload_mass using built-in AVG function.

First Successful Ground Landing Date



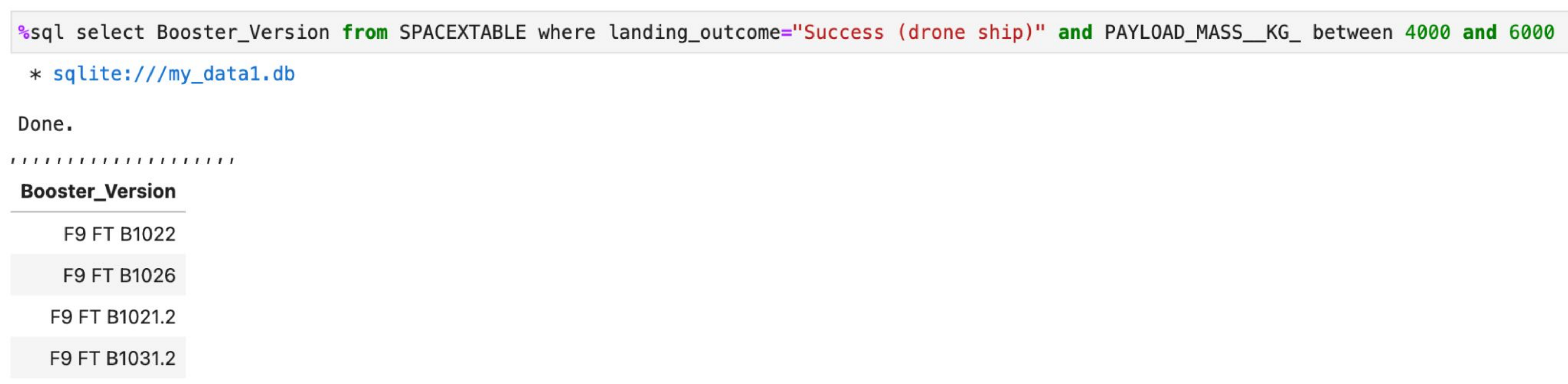
```
%sql select min(date) from SPACEXTABLE where landing_outcome="Success (ground pad)"
```

```
 * sqlite:///my_data1.db
Done.
   min(date)
  2015-12-22
```

Explanation:

It filters records by landing_outcome to be “Success (ground pad)” and finds minimum date using built-in MIN function.

Successful Drone Ship Landing with Payload between 4000 and 6000



```
%sql select Booster_Version from SPACEXTABLE where landing_outcome="Success (drone ship)" and PAYLOAD_MASS__KG_ between 4000 and 6000
```

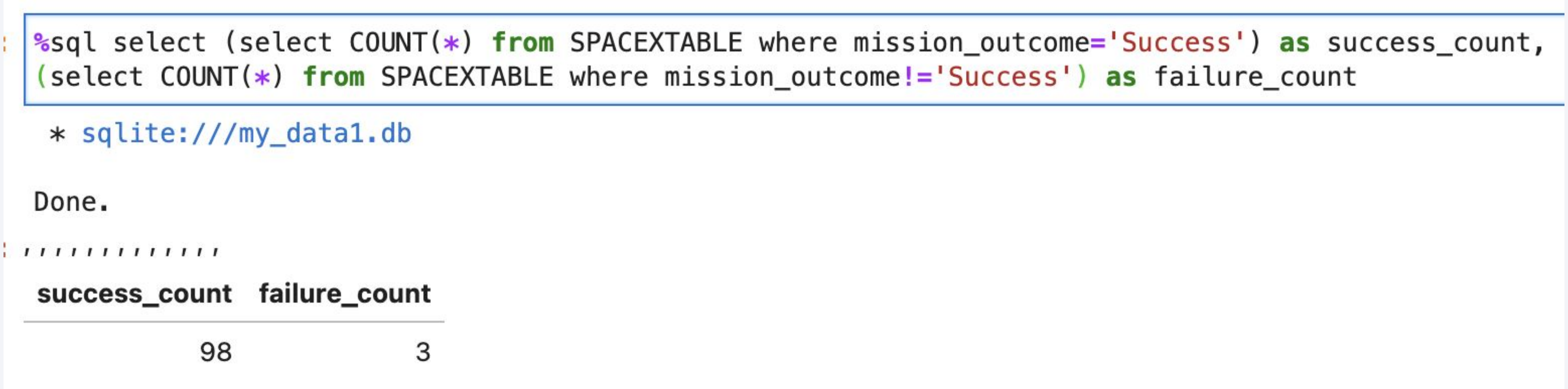
```
 * sqlite:///my_data1.db
Done.
```

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

Explanation:

It filters records by landing outcome to be “Success (drone ship)” and payload mass to be in range [4000, 6000] kg and returns only booster version.

Total Number of Successful and Failure Mission Outcomes



```
%sql select (select COUNT(*) from SPACEXTABLE where mission_outcome='Success') as success_count,
(select COUNT(*) from SPACEXTABLE where mission_outcome!='Success') as failure_count
```

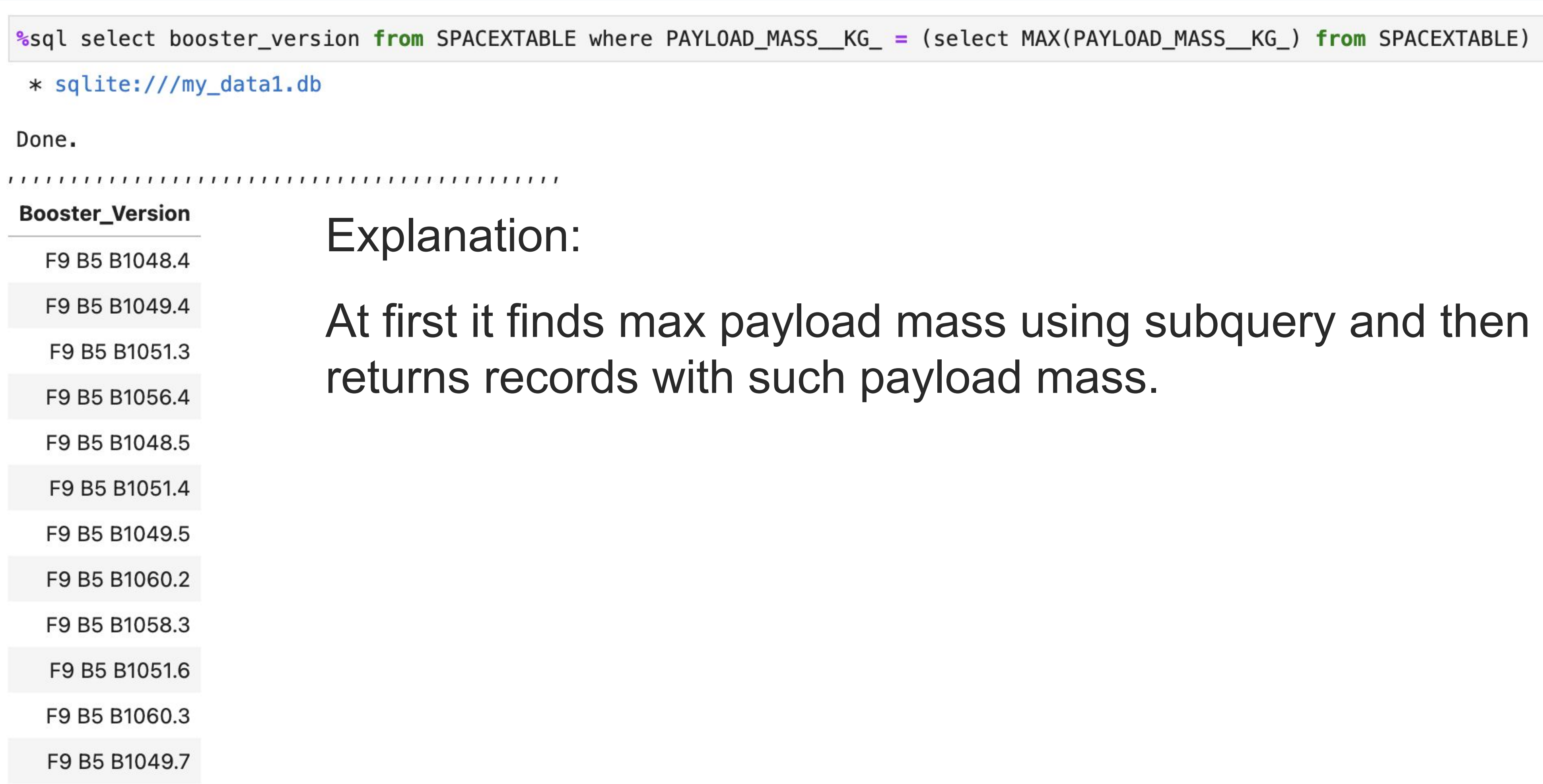
```
 * sqlite:///my_data1.db

Done.
```

success_count	failure_count
98	3

Explanation:

It uses 2 subqueries, one of them counts all records with successful mission outcome and another with not successful.

Boosters Carried Maximum Payload



```
%sql select booster_version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

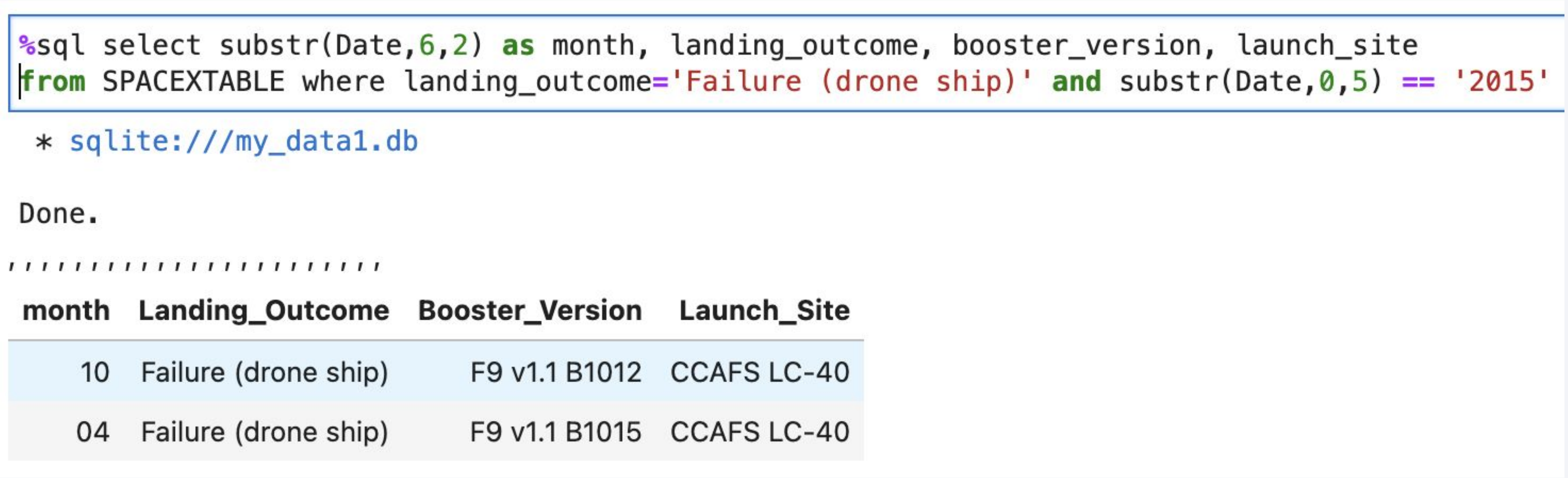
```
 * sqlite:///my_data1.db
Done.
''''''''''''''''''''''''''''''''''''''''''''
```

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

Explanation:

At first it finds max payload mass using subquery and then returns records with such payload mass.

2015 Launch Records



```
%sql select substr(Date,6,2) as month, landing_outcome, booster_version, launch_site
from SPACEXTABLE where landing_outcome='Failure (drone ship)' and substr(Date,0,5) == '2015'
```

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

Done.
```

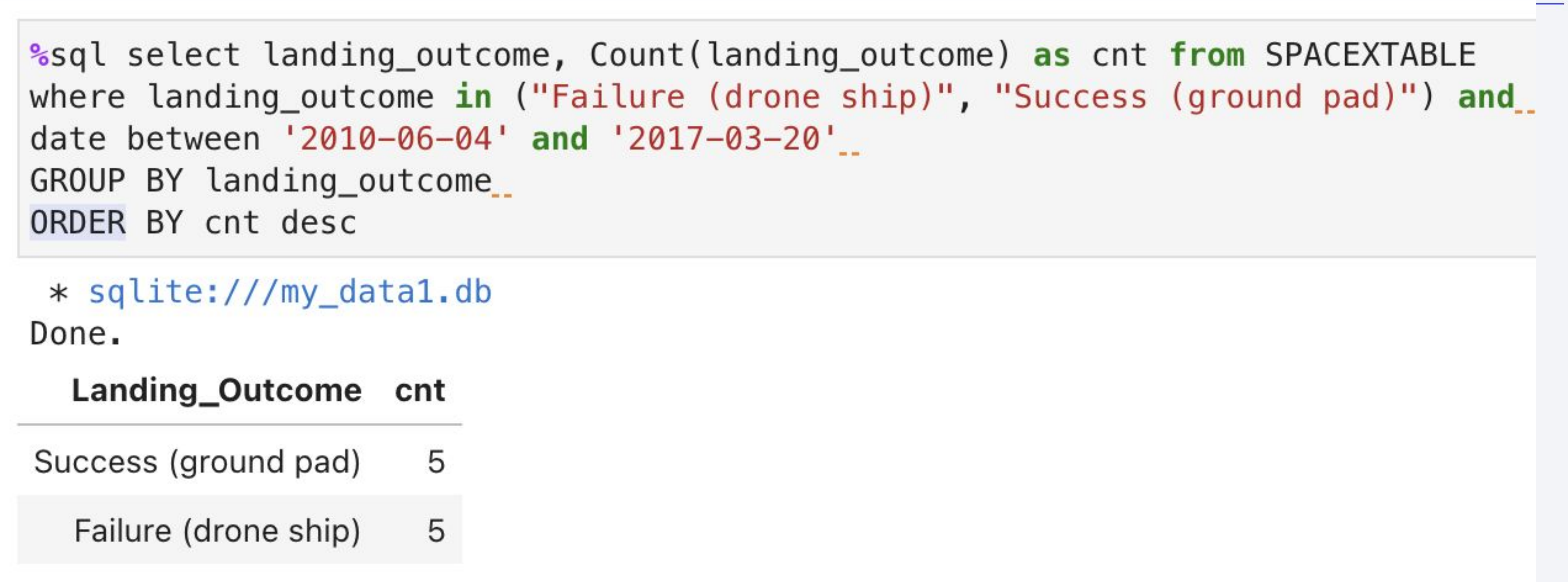
'''''''''''''''''''''''

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

Explanation:

It filters records by landing outcome to be Failure (drone ship) and by year to be 2015. To get year from a date it uses substr function.

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



```
%sql select landing_outcome, Count(landing_outcome) as cnt from SPACEXTABLE
where landing_outcome in ("Failure (drone ship)", "Success (ground pad)") and
date between '2010-06-04' and '2017-03-20'
GROUP BY landing_outcome
ORDER BY cnt desc
```

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

Landing_Outcome	cnt
Success (ground pad)	5
Failure (drone ship)	5

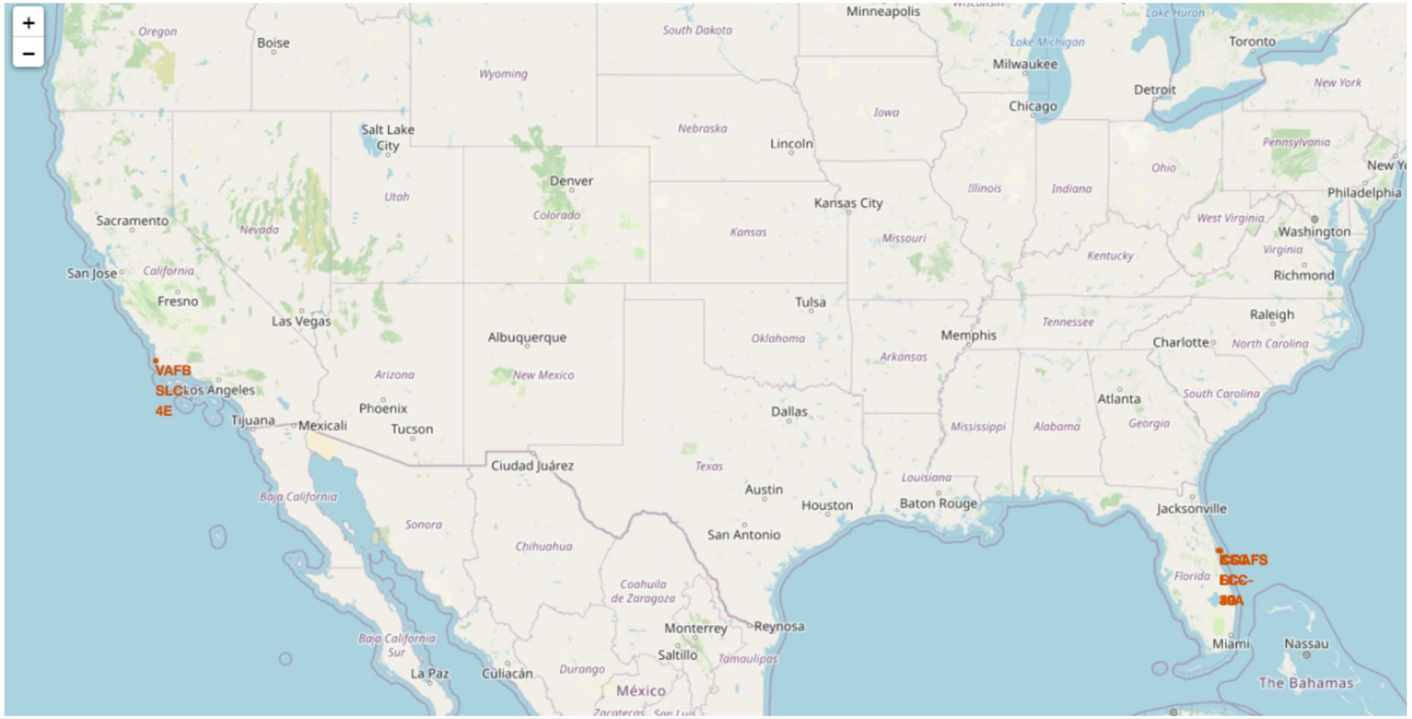
Explanation:

It groups records filtered records by landing outcome and for each outcome counts its records

Section 3

Launch Sites Proximities Analysis

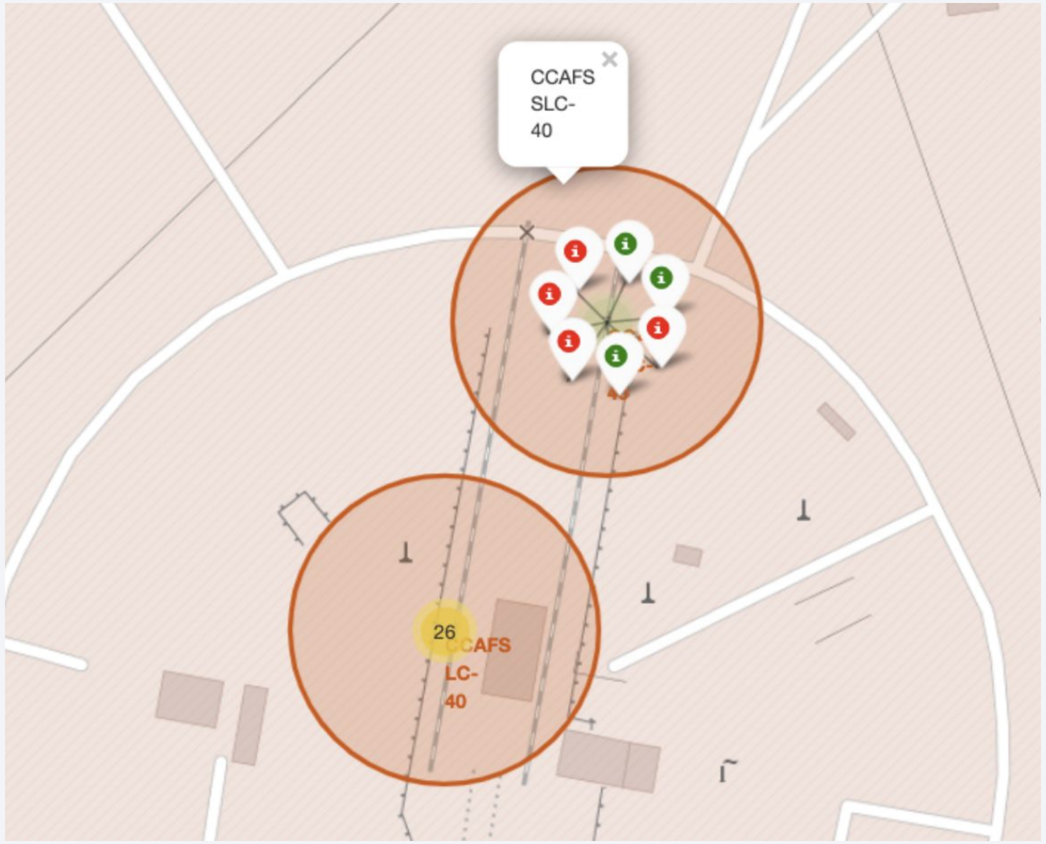
Folium Map - Ground stations



Explanation:

On that map we see that SpaceX launch sites located on the coast of US.

Folium Map - Marker clusters and labels

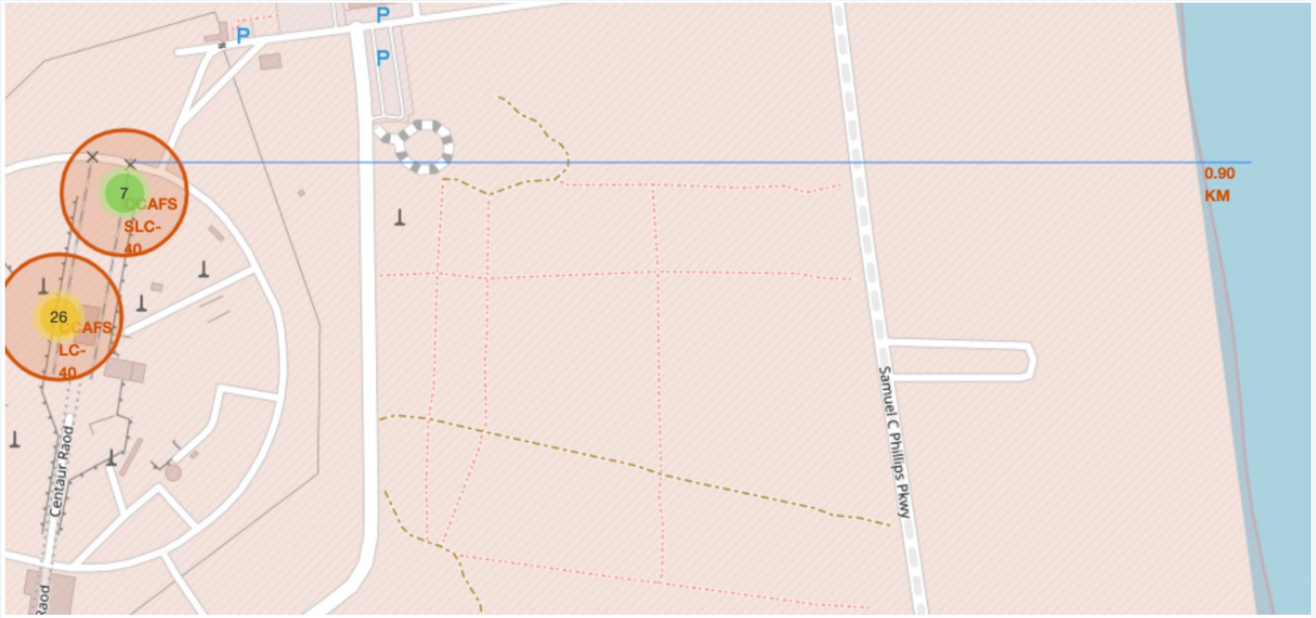


Explanation:

Red labels represent Failed launches and green labels represent Succesful launches.

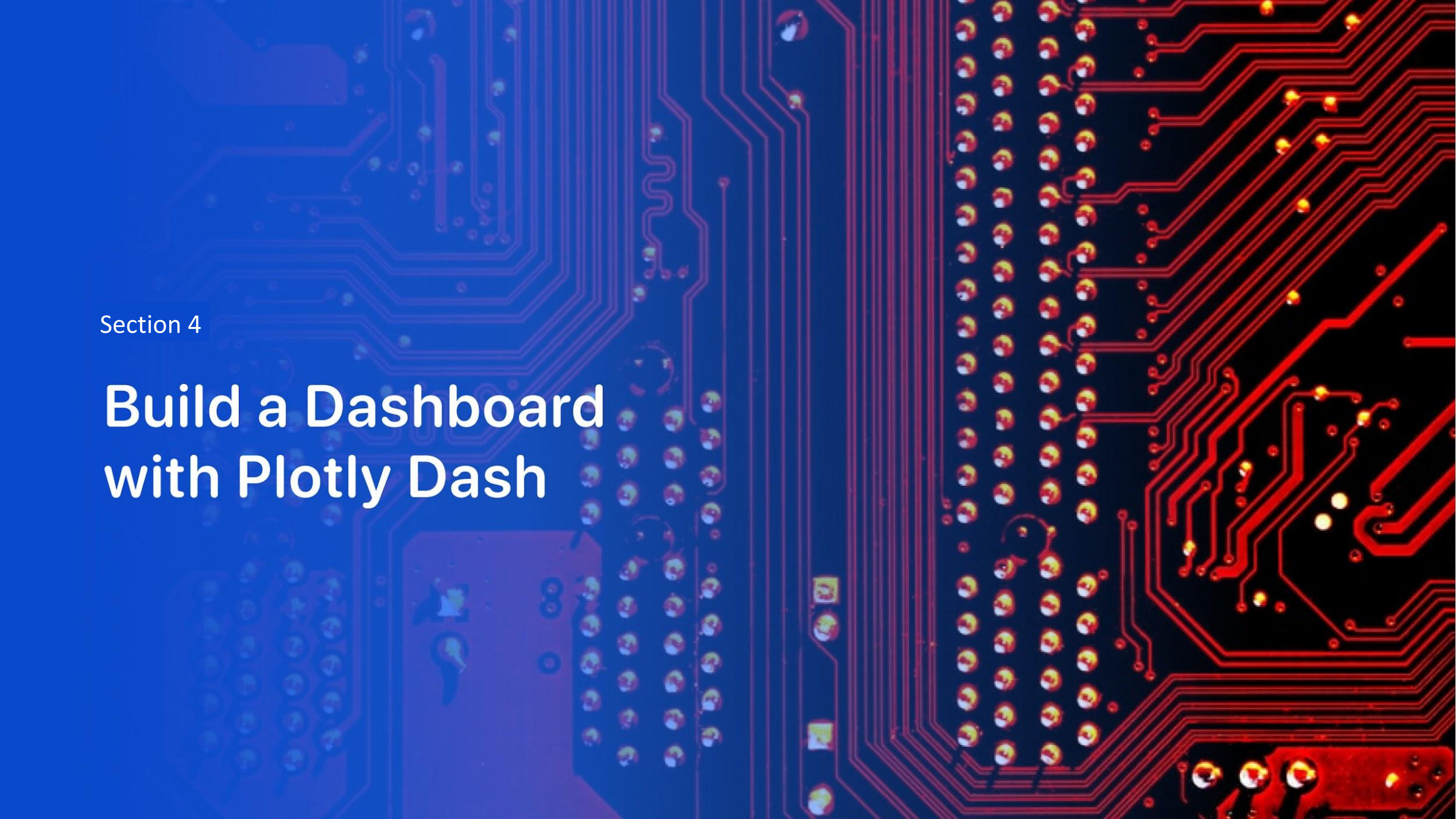
The most successful launch site is KSC LC-39A.

Folium Map - Distances between CCAFS SLC-40 and its proximities



Explanation:

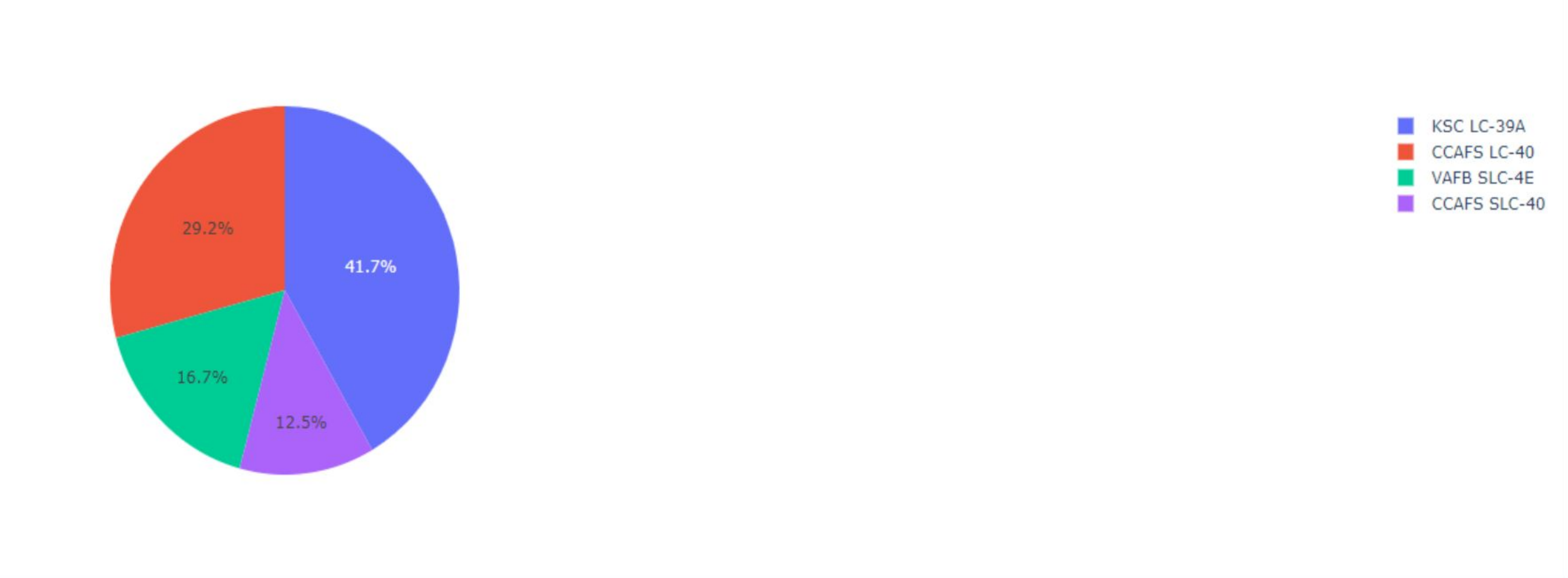
CCAFS SLC-40 is pretty close to all of its proximities such as railways, highways, coastline but it doesn’t keep distance away from cities.



Section 4

Build a Dashboard with Plotly Dash

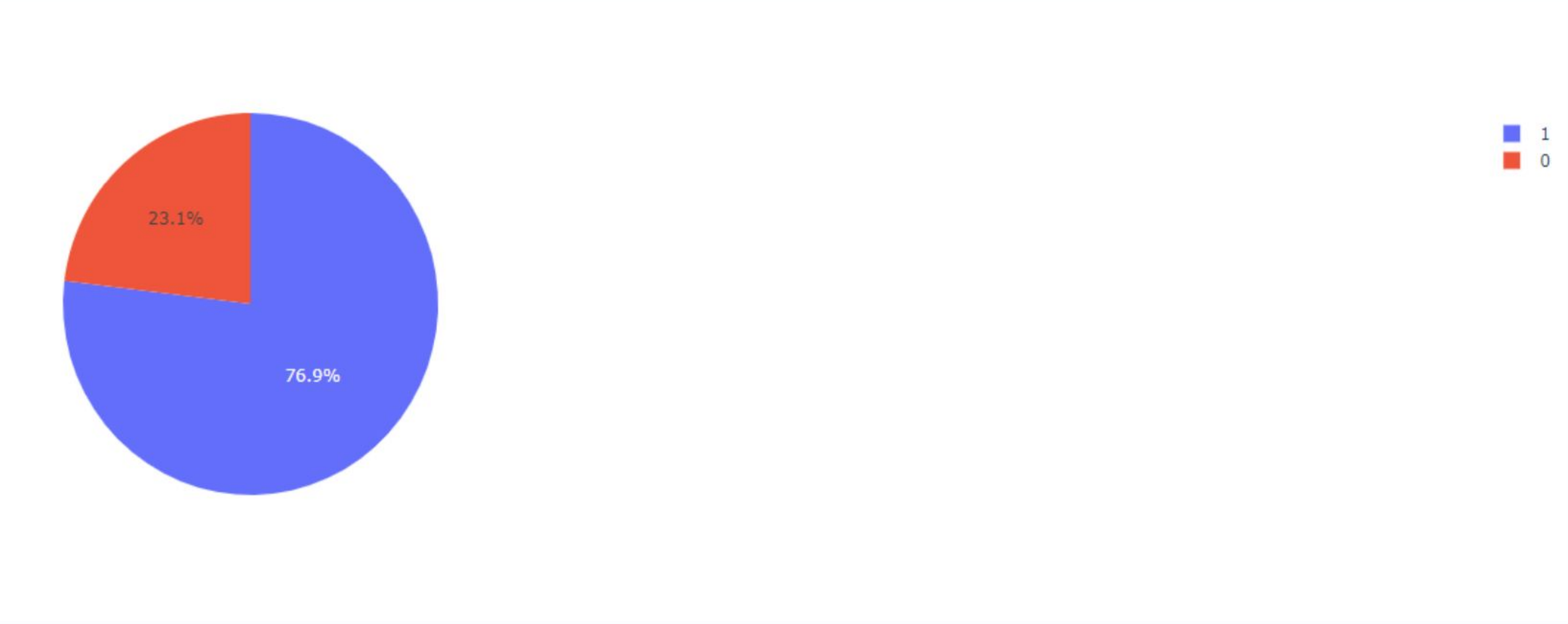
Plotly Dash - Total success rate



Explanation:

KSL LC-39A has the highest success rate.

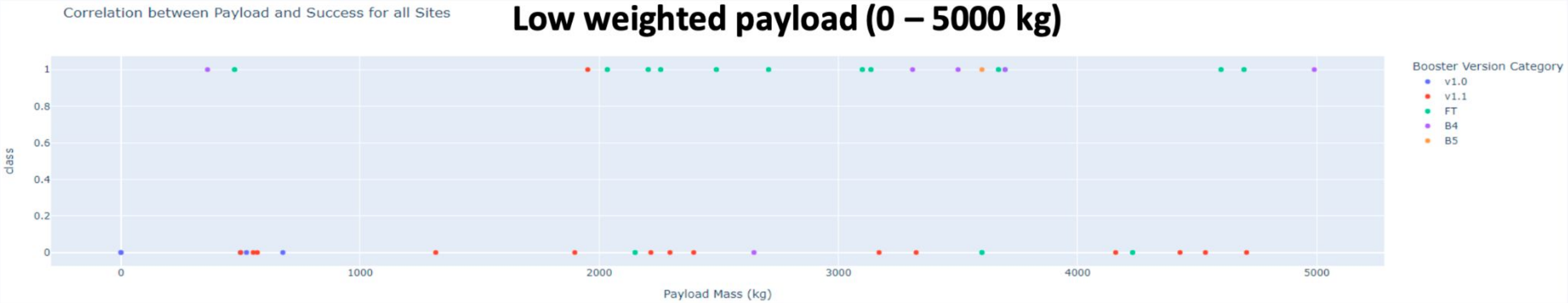
Plotly Dash - KSL LC-39A success rate



Explanation:

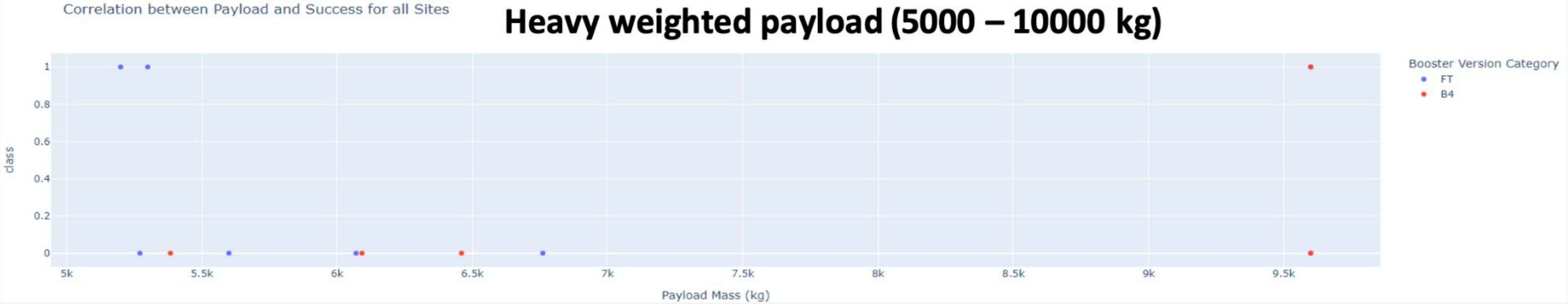
The launch site has 76.9% success launches and only 23.1% failed.

Plotly Dash - Correlation between Payload Mass vs Outcome for all launch sites



Correlation between Payload and Success for all Sites

Low weighted payload (0 – 5000 kg)



Correlation between Payload and Success for all Sites

Heavy weighted payload (5000 – 10000 kg)

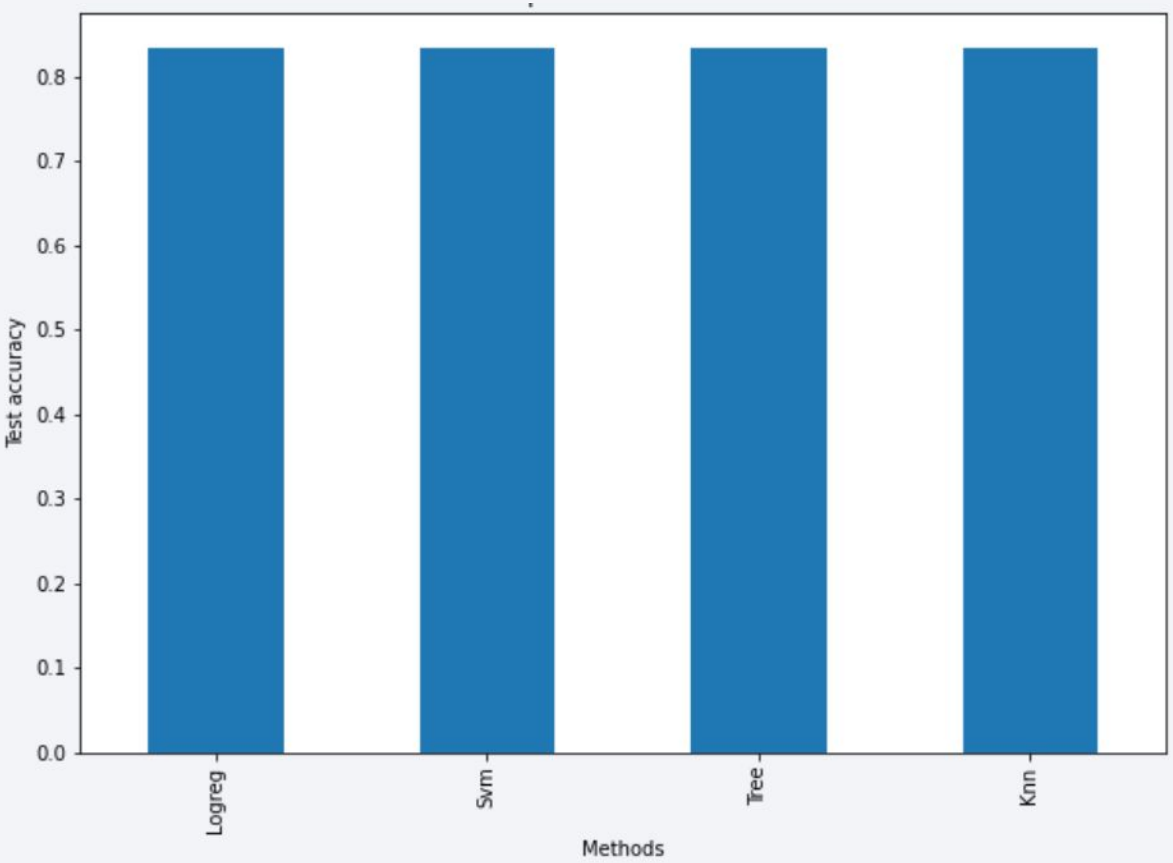
Explanation:

By comparing the plots we can see that success outcome for specific Booster versions is highest payload in [2000, 5000] kg.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



Explanation:

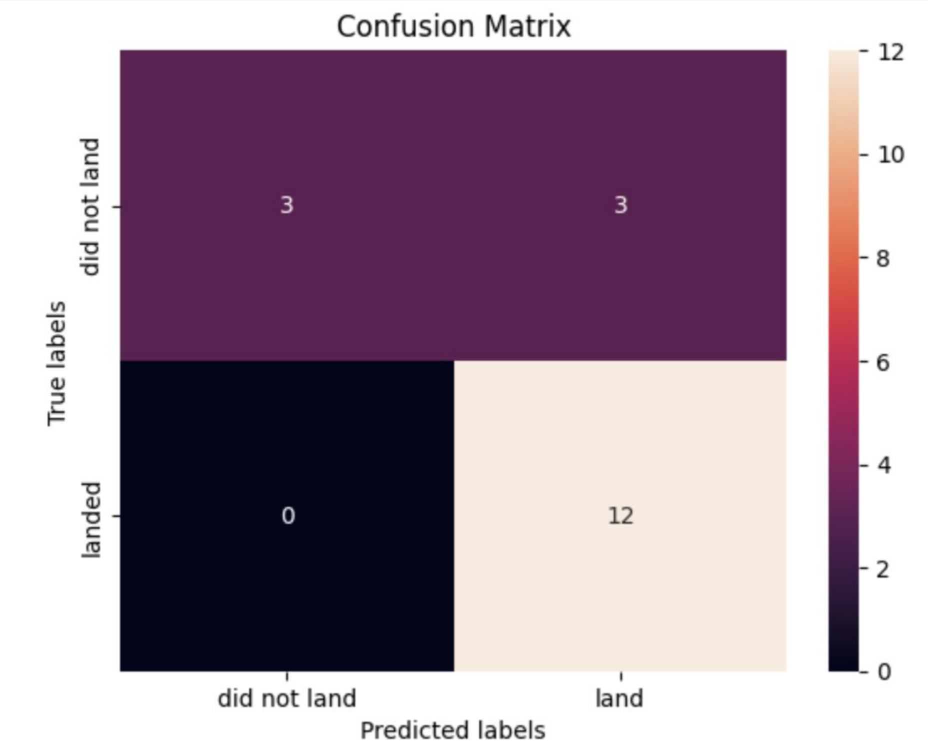
From the bar plot we can see that accuracy on test data for all methods are the same.

But on training data the best result has decision trees, its accuracy is 88.9%.

Decision Tree best params:

```
{'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'best'}
accuracy : 0.8892857142857142
```

Confusion Matrix



Explanation:
Confusion Matrix plots look the same for all methods. We can see from that graph that the main problem is with false positives.

Conclusions

- We can see that success outcome correlates with such variables as flight number, launch site, the orbit.
- The best success rates we can see on the following orbits: GEO, HEO, SSO, ES-L1.
- The payload mass highly correlated with successful outcome, payloads mass in such range as [2000, 5000] kg leads to successful landing but it’s true only for specific orbits.
- We cannot explain correlation between launch site and successful landing with the provided data.
- The best result of prediction corresponds to Decision Tree, even when on test dataset we see the same results among methods on training data Decision Tree shows the best accuracy 88.9%.

Appendix

- https://github.com/alexkarpovich/datascience-coursera/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb
- https://github.com/alexkarpovich/datascience-coursera/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb
- https://github.com/alexkarpovich/datascience-coursera/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb
- https://github.com/alexkarpovich/datascience-coursera/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb
- https://github.com/alexkarpovich/datascience-coursera/blob/main/jupyter-labs-spacex-data-collection-api.ipynb
- https://github.com/alexkarpovich/datascience-coursera/blob/main/jupyter-labs-webscraping.ipynb
- https://github.com/alexkarpovich/datascience-coursera/blob/main/labs-jupyter-spacex-Data%20wrangling%20(1).ipynb
- https://github.com/alexkarpovich/datascience-coursera/blob/main/spacex_dash_app.py

Thank you!