IBM Data Science Capstone Project

26.07.2025

Vlad Trifautan

Executive Summary

1. Data collection with API
2. Data collection with web scrapping
3. Data wrangling
4. Exploratory Data Analysis with SQL
5. Exploratory Data Analysis with Visualization
6. Interactive Visual Analytics with Folium map
7. Interactive Dashboard with Plotly Dash
8. Machine Learning prediction

Introduction

SpaceX is one of the most fast-growing rocket-launching company from the last decades. It accomplished fascinating results and succeeded to radically change the way we launch stuff in space. It's the first company that started successfully reusing the first stage of the rocket, saving a ton of money.

We'll dive deeper into the details of the launches of SpaceX rockets, specifically into Falcon 9 launches.

Methodology

Data Collection

I used two methods for data collection: using a REST API and using web scraping

- The REST API method is the easiest, because the data is already formatted and different queries are possible by using search params. You can specify which data you want to receive. The data is requested using HTTP requests to the SpaceX API.
- The web scraping method is less elegant, and is not so flexible. You are limited on the data you see on the web page. I used BeautifulSoup to parse the data from tables.

Data Wrangling

For data wrangling I used pandas and numpy.

In the process of data wrangling I did the following things:

- Calculated the number of launches on each site
- Calculated the number and occurence of each orbit
- Calculated the number and occurence of mission outcome of the orbits
- Created a landing outcome column (0 or 1)

This way I prepared the data for analysis. I needed the outcome to later calculate the correlation between different parameters and the outcome.

Exploratory Data Analysis using SQL

In the first step of the EDA I used a Jupyter notebook with the %sql magic function to be able to make sql queries.

I explored different things like the unique launch sites, average payload mass carried by Falcon 9, the landing outcomes of different landing sites (ground pad, drone ship), the list of boosters with maximum payload mass and more.

Exploratory Data Analysis with Visualization - charts

For this part, I used pandas, numpy, seaborn and matplotlib.

I visualized the relationship between flight number and launch site, payload mass and launch site, success rate of each orbit type, flight number and orbit type, payload mass and orbit type, launch success yearly trend.

For this I used scatterplot, barchart, and lineplot.

Exploratory Data Analysis with Visualization - Folium Map

For the second part of EDA, I used pandas and Folium maps.

I used the Folium's markers to visualize the launching sites and to find reasons of the placement of these sites.

I calculated the proximity of different infrastructure elements (highway, railroad, shore).

Exploratory Data Analysis with Visualization - Plotly Dash

I used Plotly Dash to create a simple dashboard where I displayed the success rates of different sites. I used pie charts and line charts.

Machine Learning Prediction for landing outcomes

To accomplish this, I used pandas, numpy, matplotlib, seaborn and sklearn.

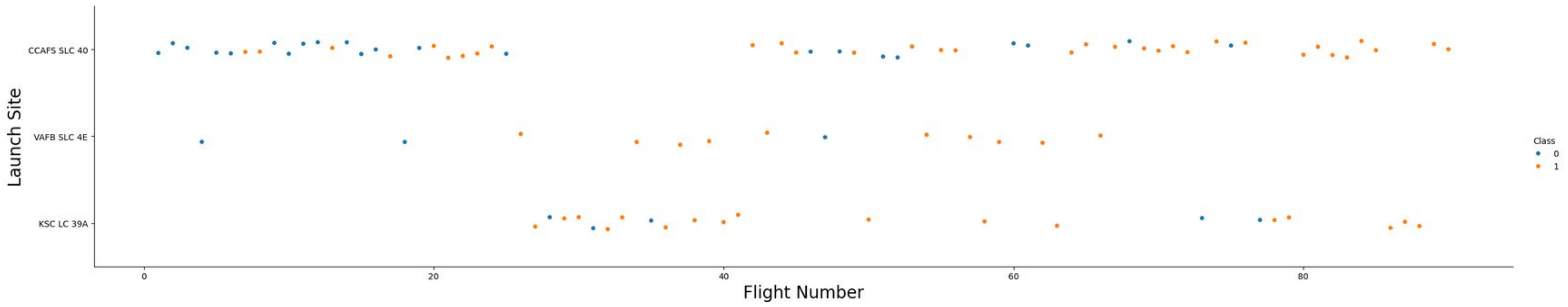
I used the following ML objects:

- Logistic Regression
- Support Vector Machine (SVM)
- Decision Tree
- K Nearest Neighbors (KNN)

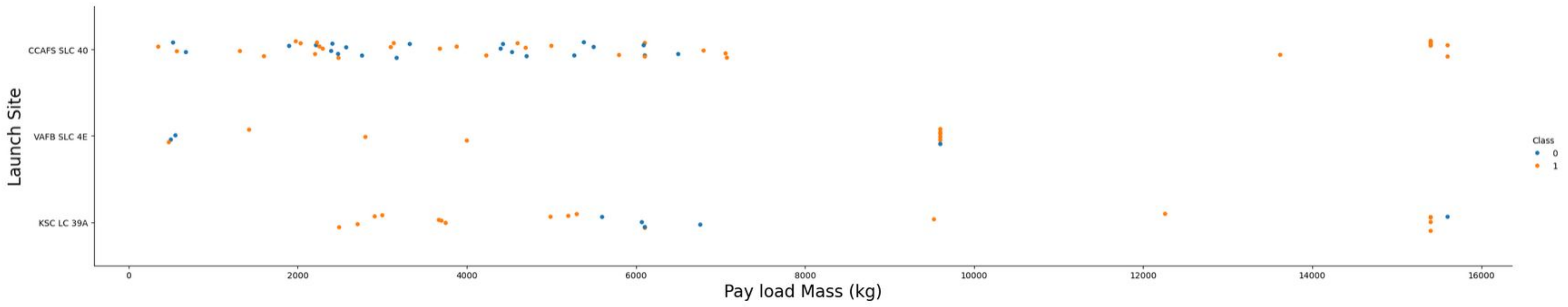
Results

Exloratory Data Analysis with visualization - charts

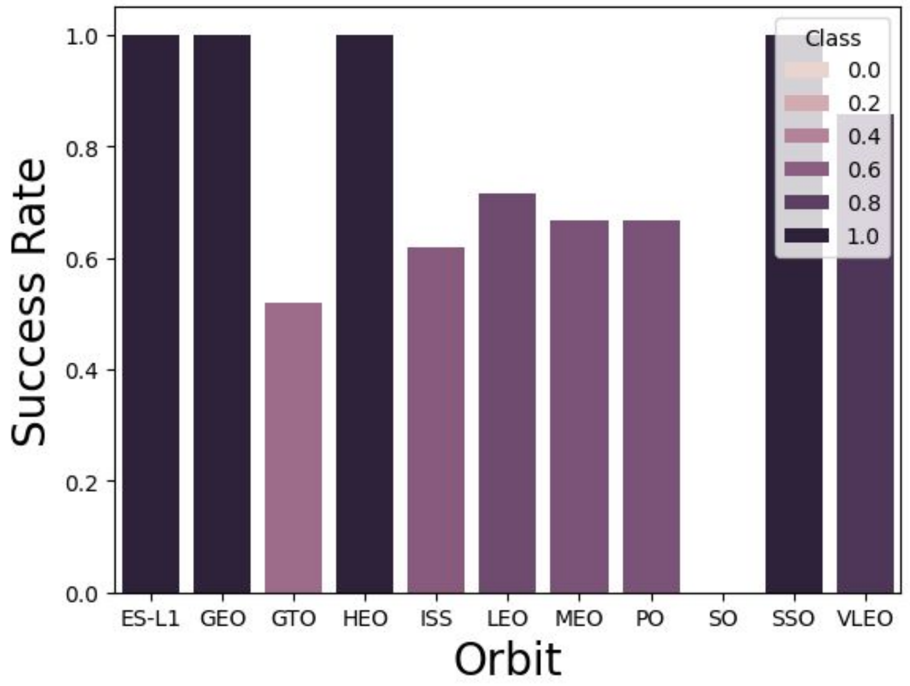
Relationship between Flight Number and Launch Site



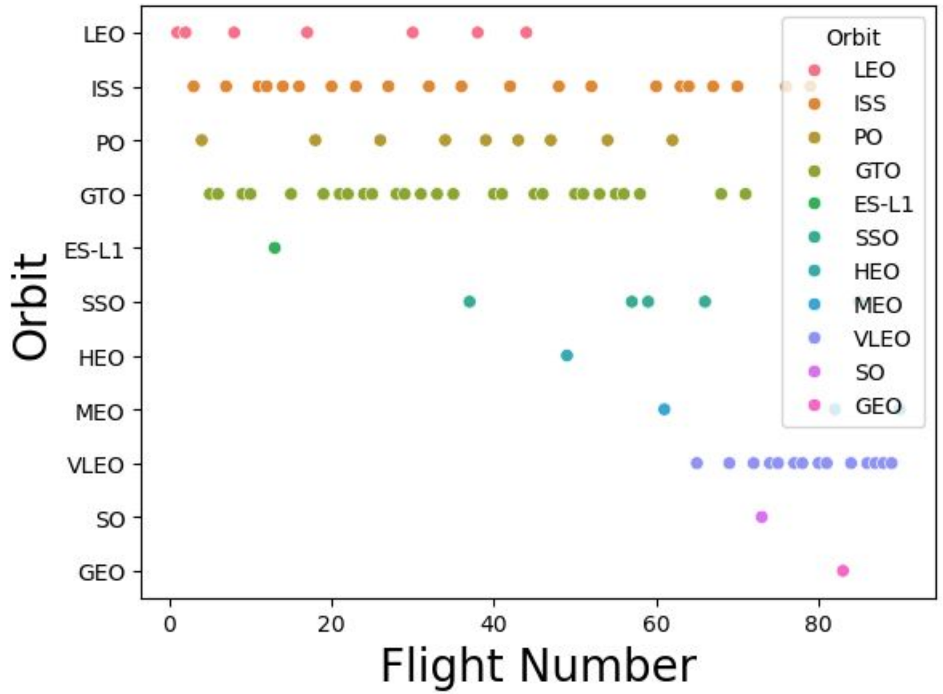
Relationship between payload mass and launch site



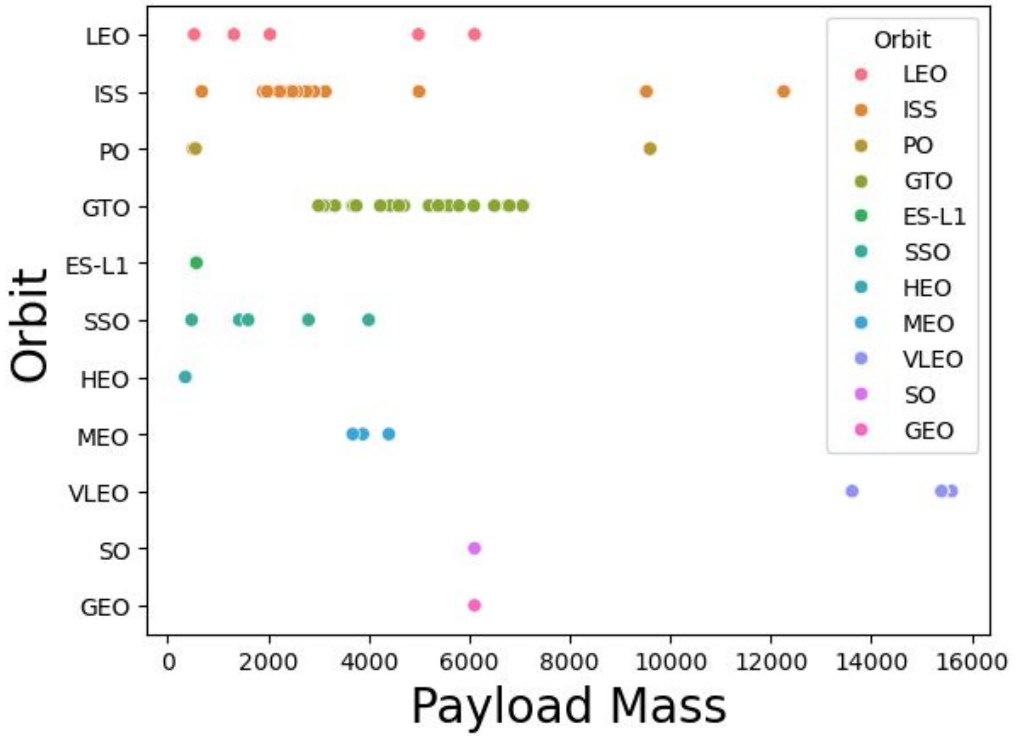
Success rate of each orbit type



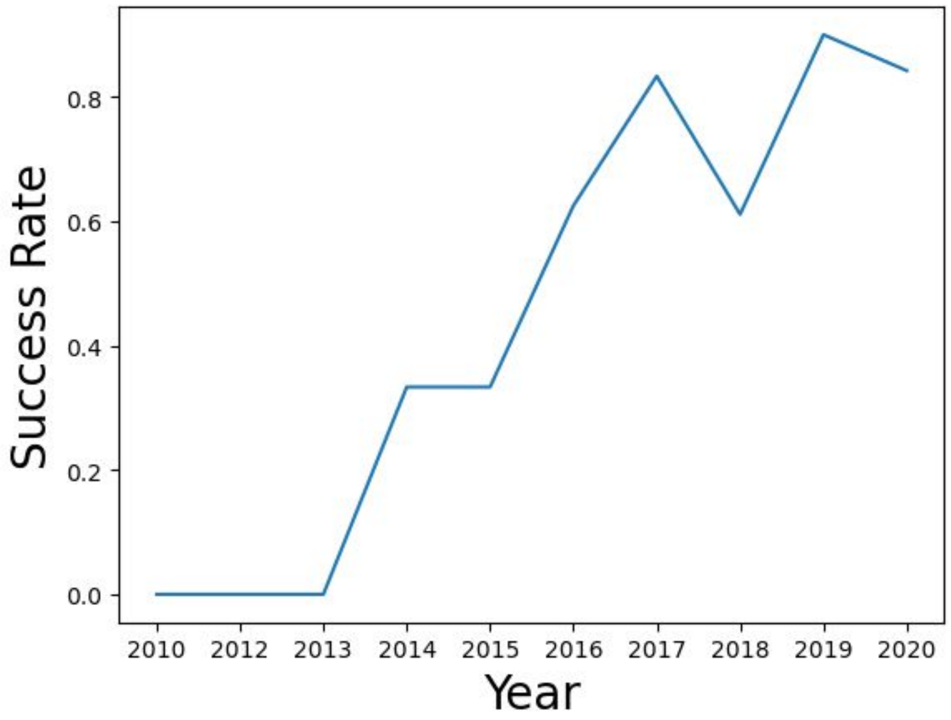
Relationship between flight number and orbit type



Relationship between payload mass and orbit type



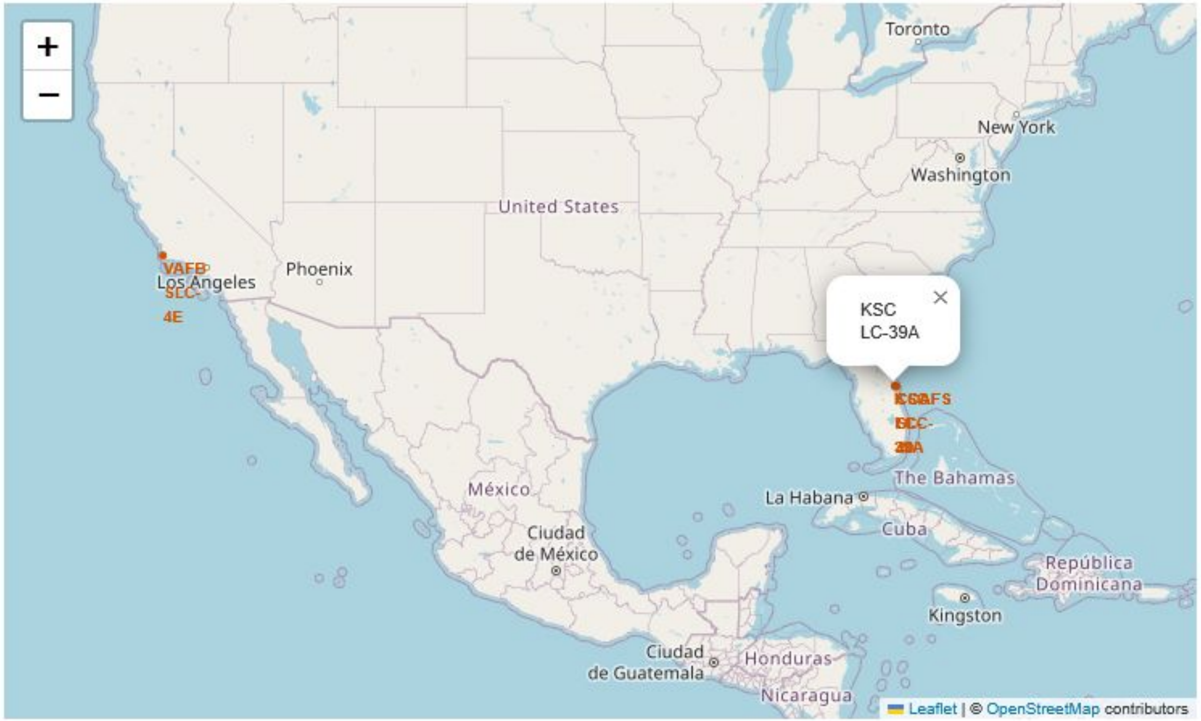
Launch success yearly trend



Exloratory Data Analysis with visualization - Folium Maps

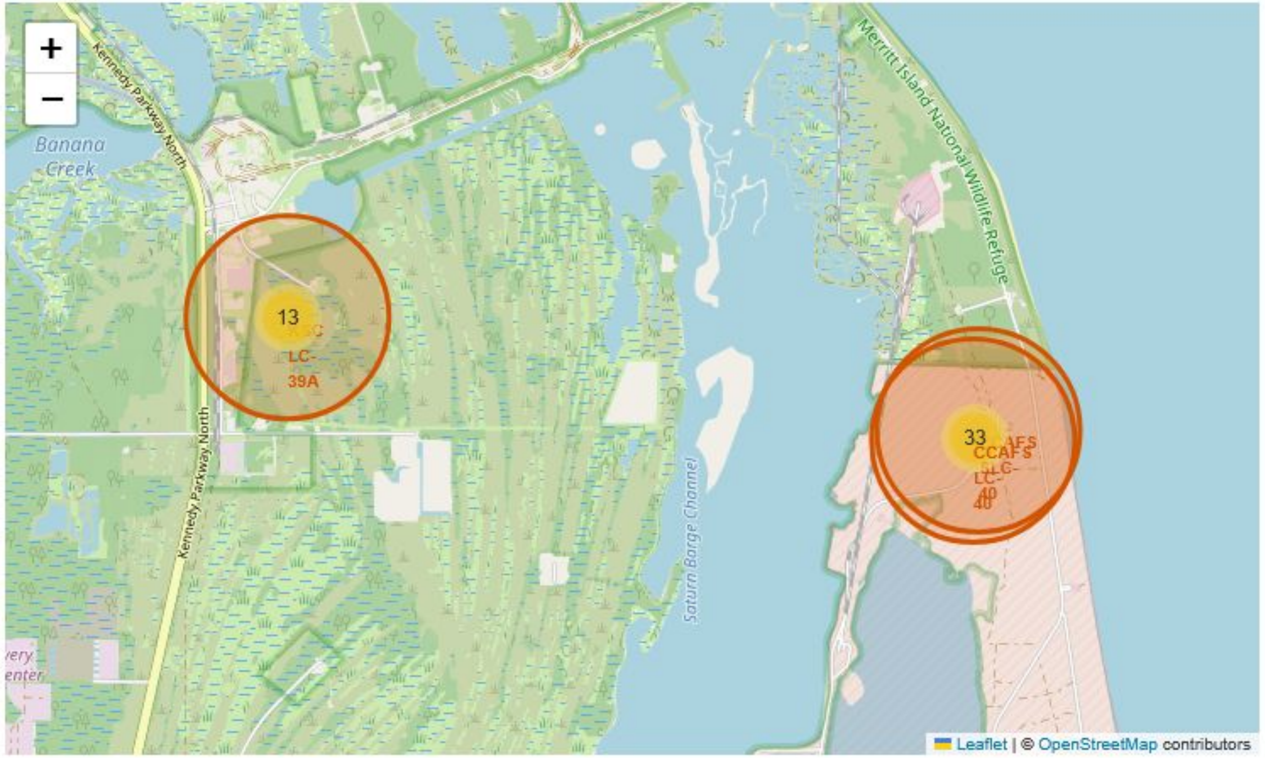
Launch sites simple markers

[9]:

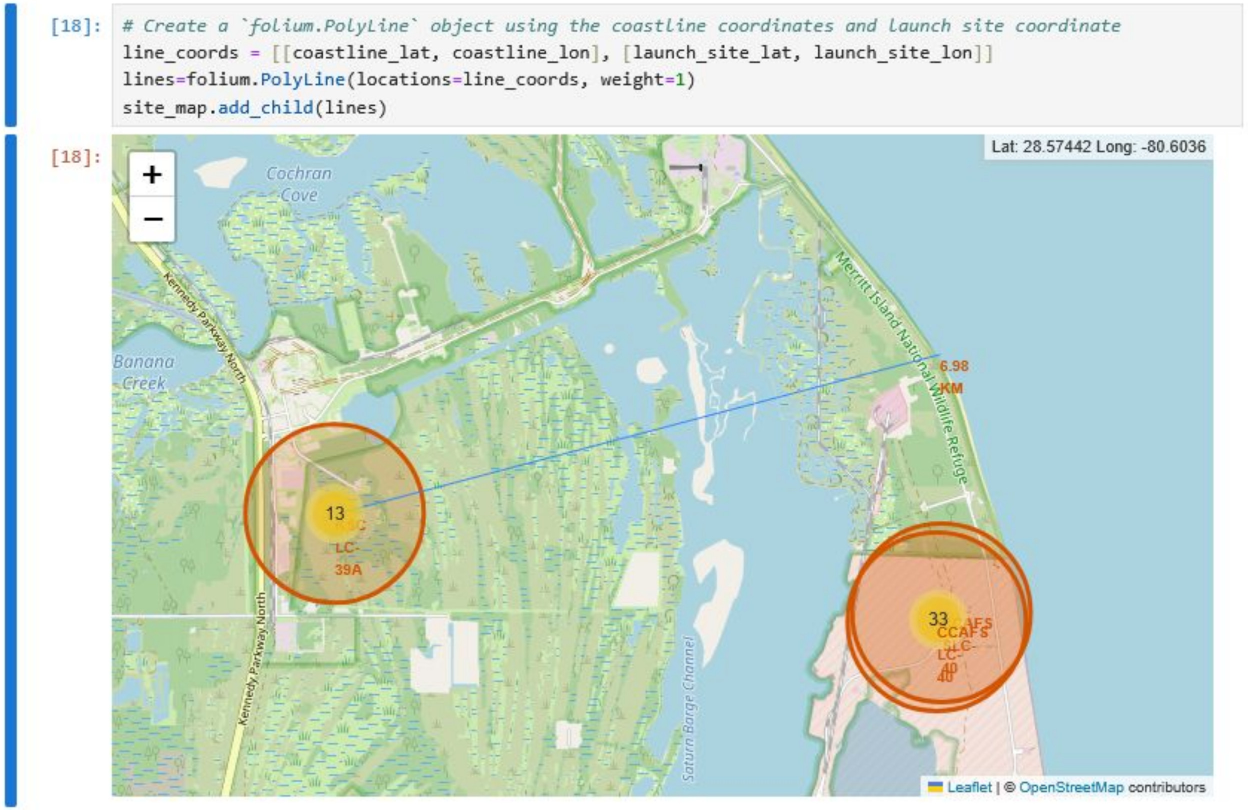


Launch Sites marker cluster

[13]:


+
−
Banana Creek
Kennedy Parkway North
Merritt Island National Wildlife Refuge
Saturn Barge Channel
13
LC-
39A
33
CCAFS
LC-
40
Leaflet | © OpenStreetMap contributors

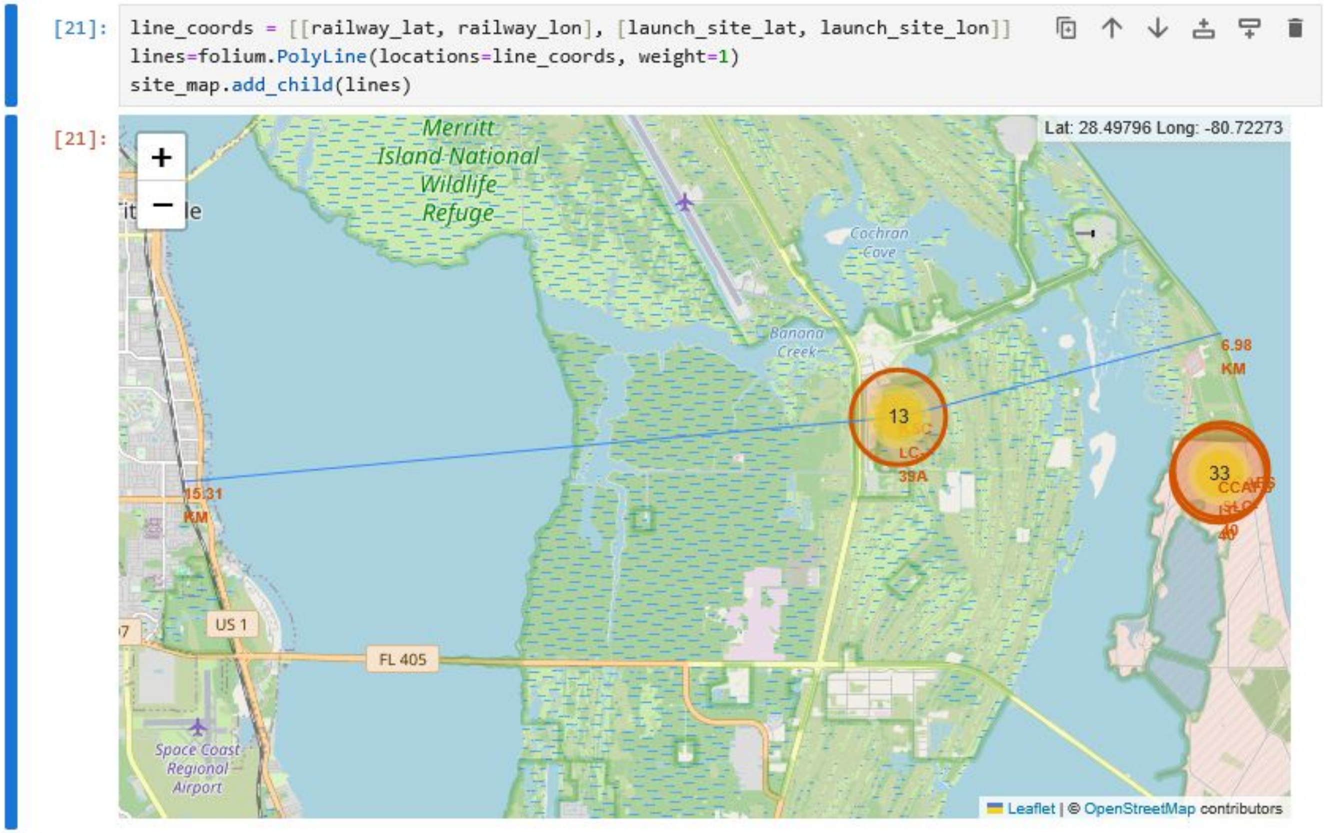
Distance to closest coastline



```
[18]: # Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
line_coords = [[coastline_lat, coastline_lon], [launch_site_lat, launch_site_lon]]
lines=folium.PolyLine(locations=line_coords, weight=1)
site_map.add_child(lines)
```

[18]:

Distance to closest railway



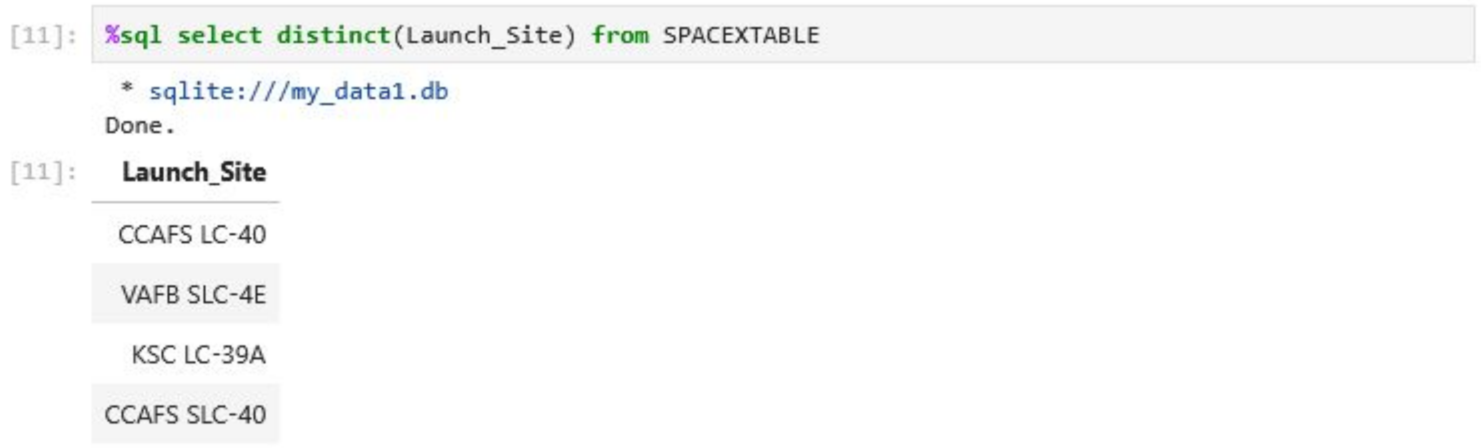
```
[21]: line_coords = [[railway_lat, railway_lon], [launch_site_lat, launch_site_lon]]
lines=folium.PolyLine(locations=line_coords, weight=1)
site_map.add_child(lines)
```

[21]:

Lat: 28.49796 Long: -80.72273
Merritt Island National Wildlife Refuge
Cochran Cove
Banana Creek
6.98 KM
13
LC-39A
33
CCAFS SLC-40
15.31 KM
US 1
FL 405
Space Coast Regional Airport
Leaflet | © OpenStreetMap contributors

Exloratory Data Analysis with SQL

Unique Launch Sites



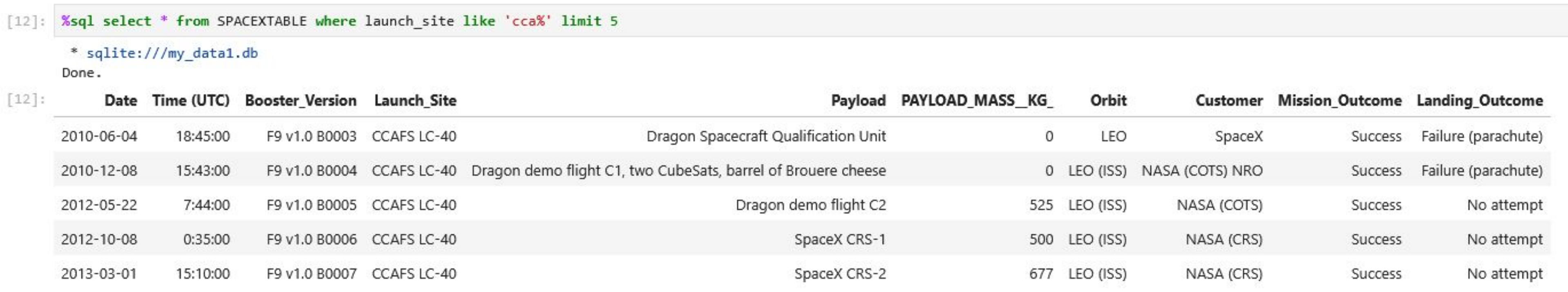
```
[11]: %sql select distinct(Launch_Site) from SPACEXTABLE

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

[11]:

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

Launch sites beginning with 'CCA'



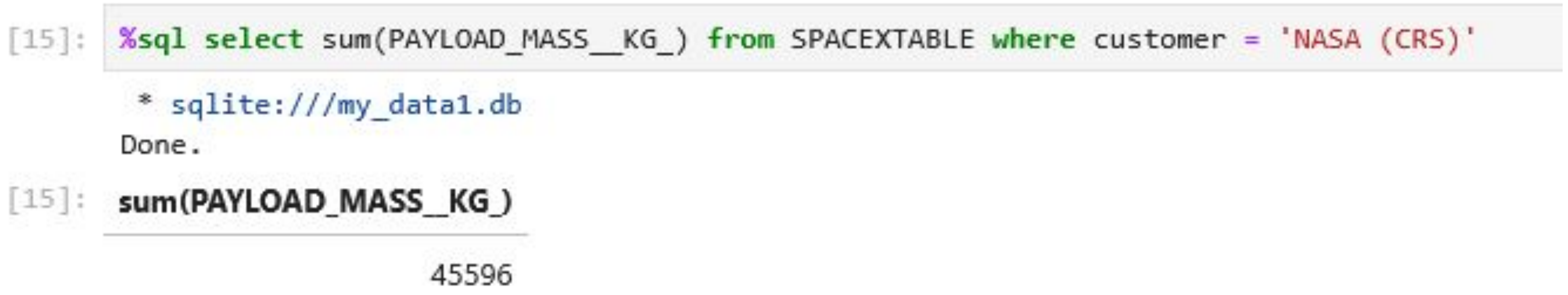
```
[12]: %sql select * from SPACEXTABLE where launch_site like 'cca%' limit 5

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

[12]:

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 carried by boosters launched by NASA (CRS)

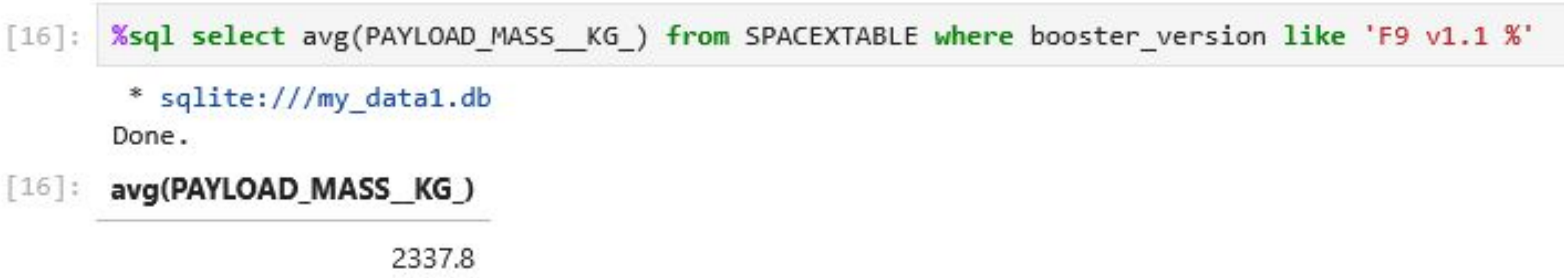


```
[15]: %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where customer = 'NASA (CRS)'

 * sqlite:///my_data1.db
Done.

[15]: sum(PAYLOAD_MASS__KG_)
                      45596
```

Average payload mass carried by Falcon 9 V1.1

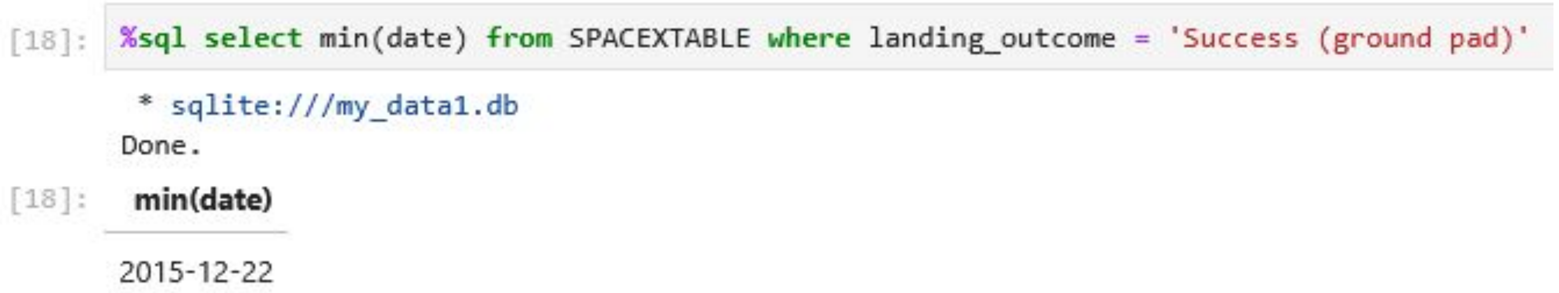


```
[16]: %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where booster_version like 'F9 v1.1 %'

 * sqlite:///my_data1.db
Done.

[16]: avg(PAYLOAD_MASS__KG_)
                      2337.8
```

Date of the first successful landing outcome in ground pad



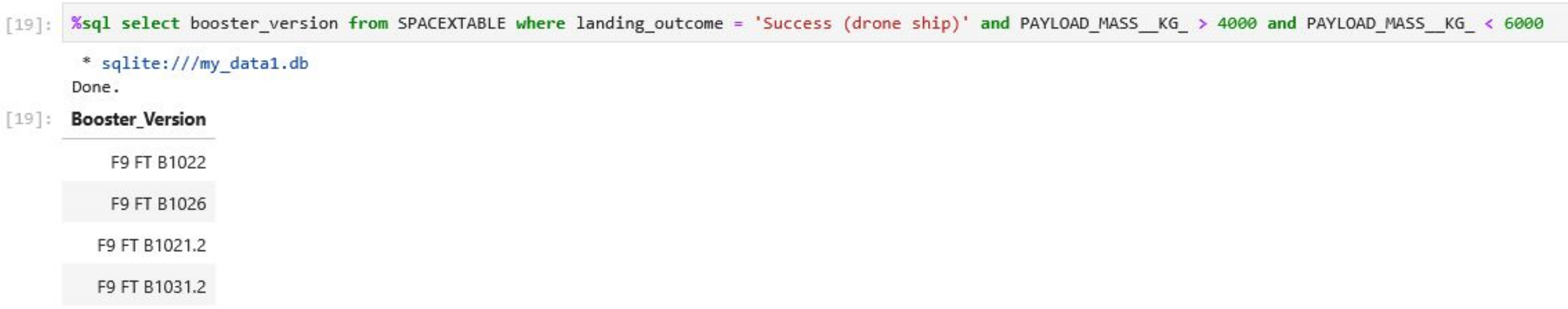
```
[18]: %sql select min(date) from SPACEXTABLE where landing_outcome = 'Success (ground pad)'

 * sqlite:///my_data1.db
Done.

[18]: min(date)

      2015-12-22
```

Boosters with successful landing in drone ship with payload mass greater than 4000 and less than 6000



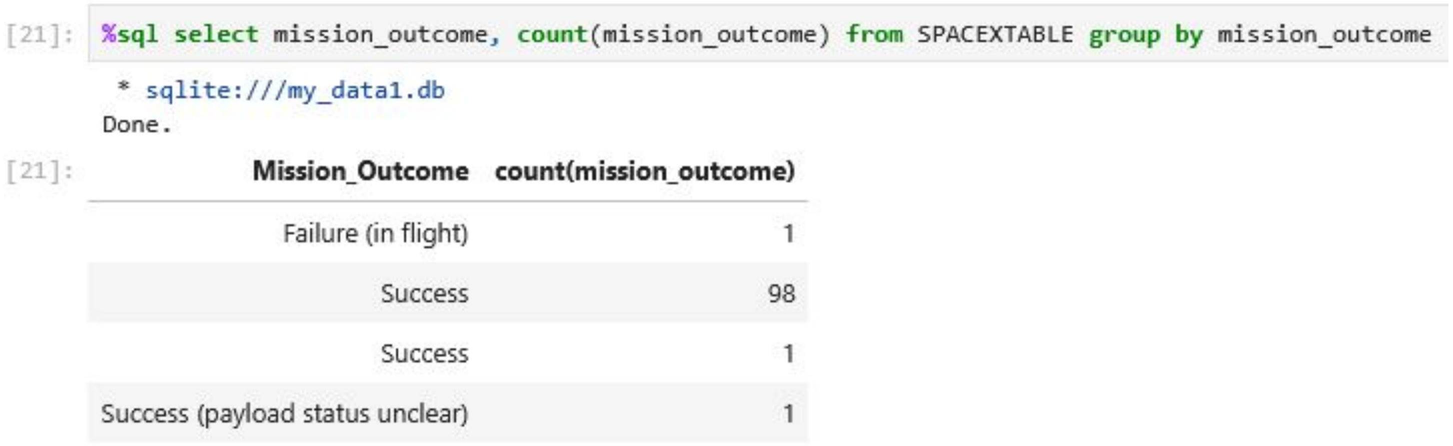
```
[19]: %sql select booster_version from SPACEXTABLE where landing_outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000

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

[19]:

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

Mission outcomes



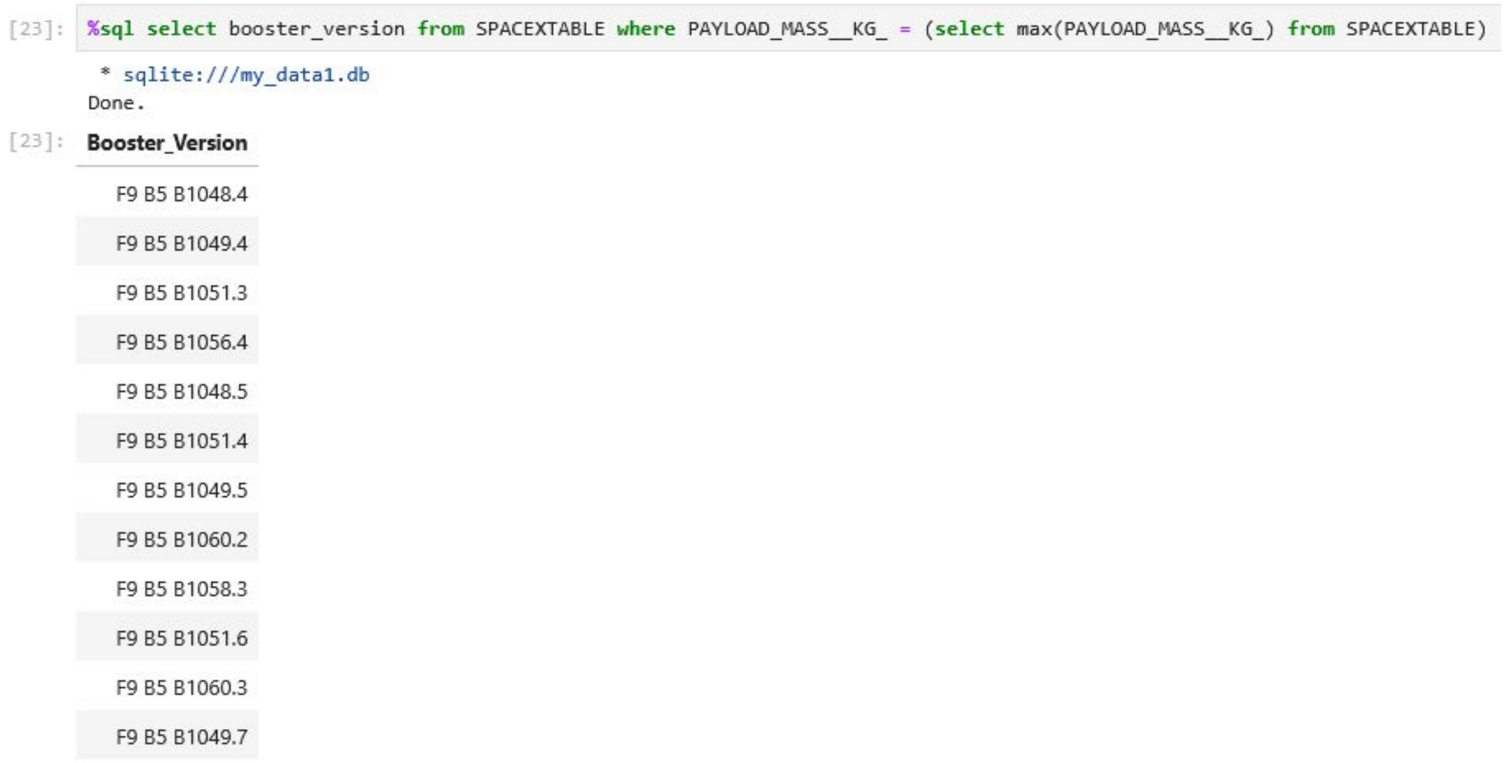
```
[21]: %sql select mission_outcome, count(mission_outcome) from SPACEXTABLE group by mission_outcome

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

[21]:

Mission_Outcome	count(mission_outcome)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Booster versions that carried maximum payload mass



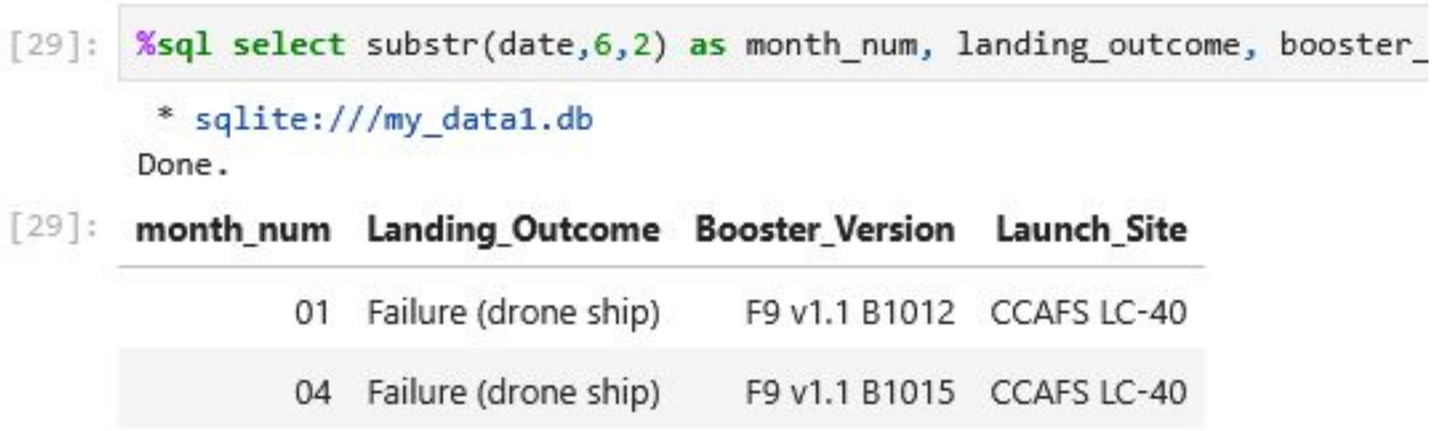
```
[23]: %sql select booster_version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)

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

[23]:

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

Failures in drone ship landings in 2025



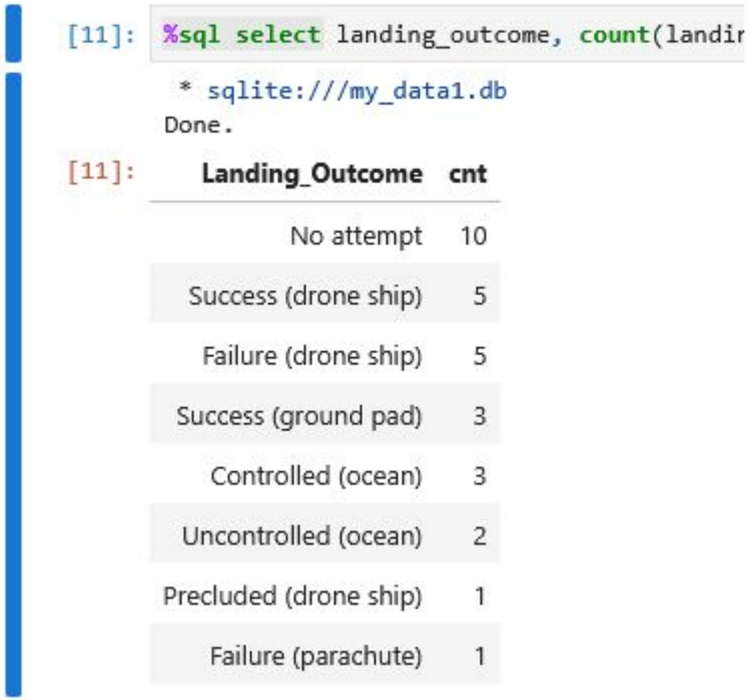
```
[29]: %sql select substr(date,6,2) as month_num, landing_outcome, booster_
```

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

[29]:

month_num	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

Landing outcomes from 2010 to 2017



```
[11]: %sql select landing_outcome, count(landir
```

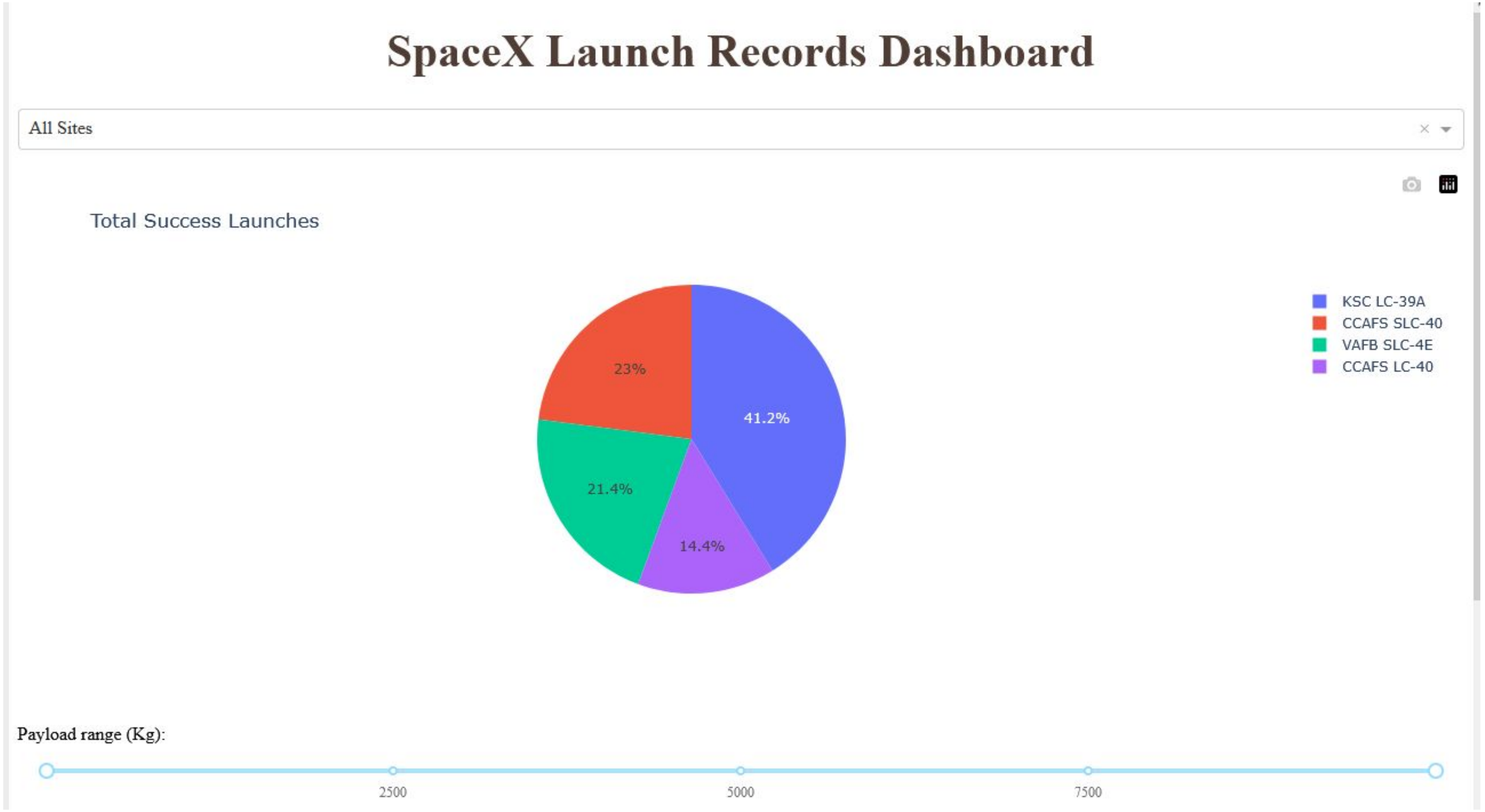
```
 * sqlite:///my_data1.db
Done.
```

[11]:

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

Exloratory Data Analysis with visualization - Plotly Dash

Total success launches

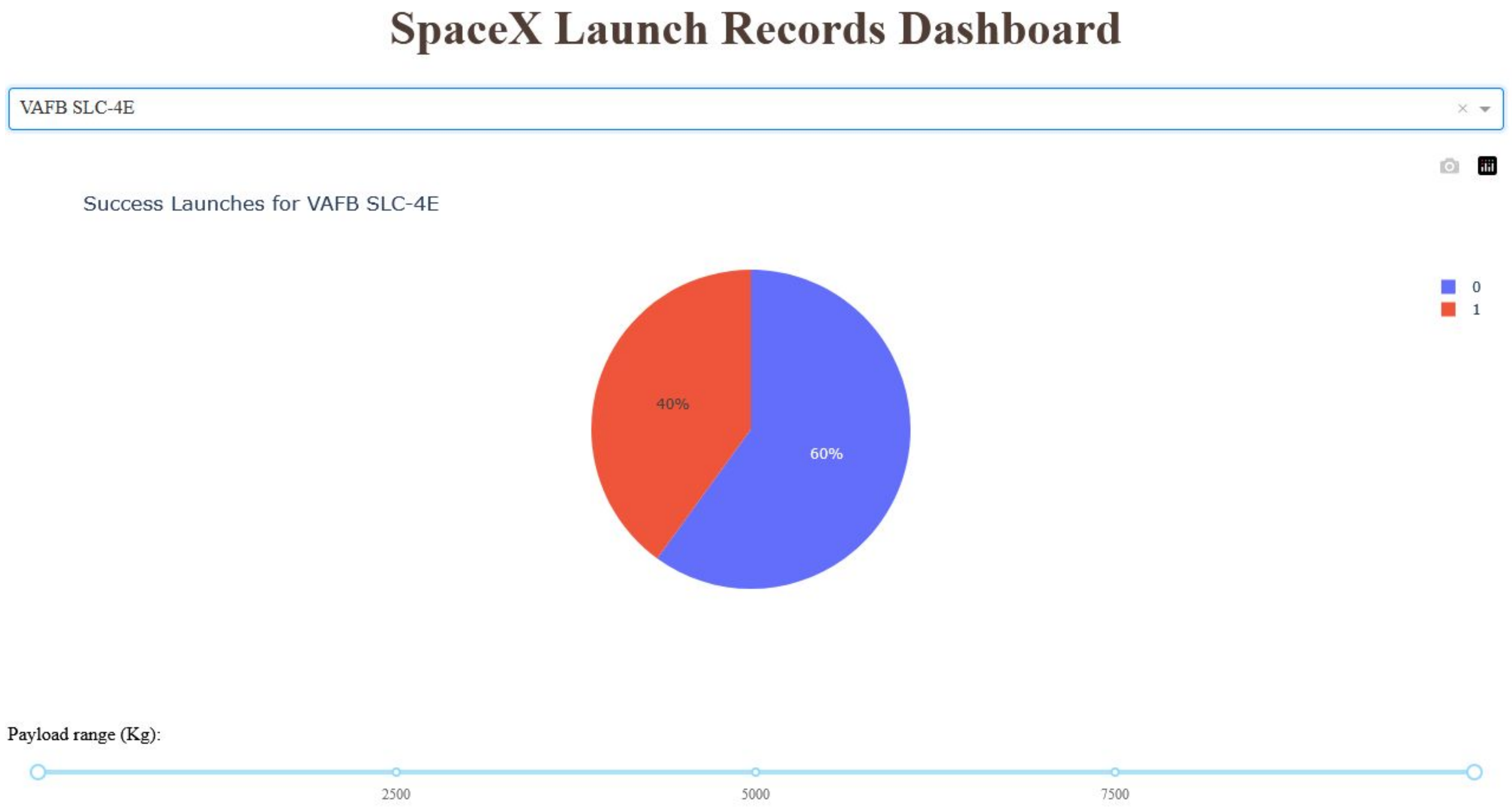


SpaceX Launch Records Dashboard

All Sites

Total Success Launches

Success Launches for a specific site



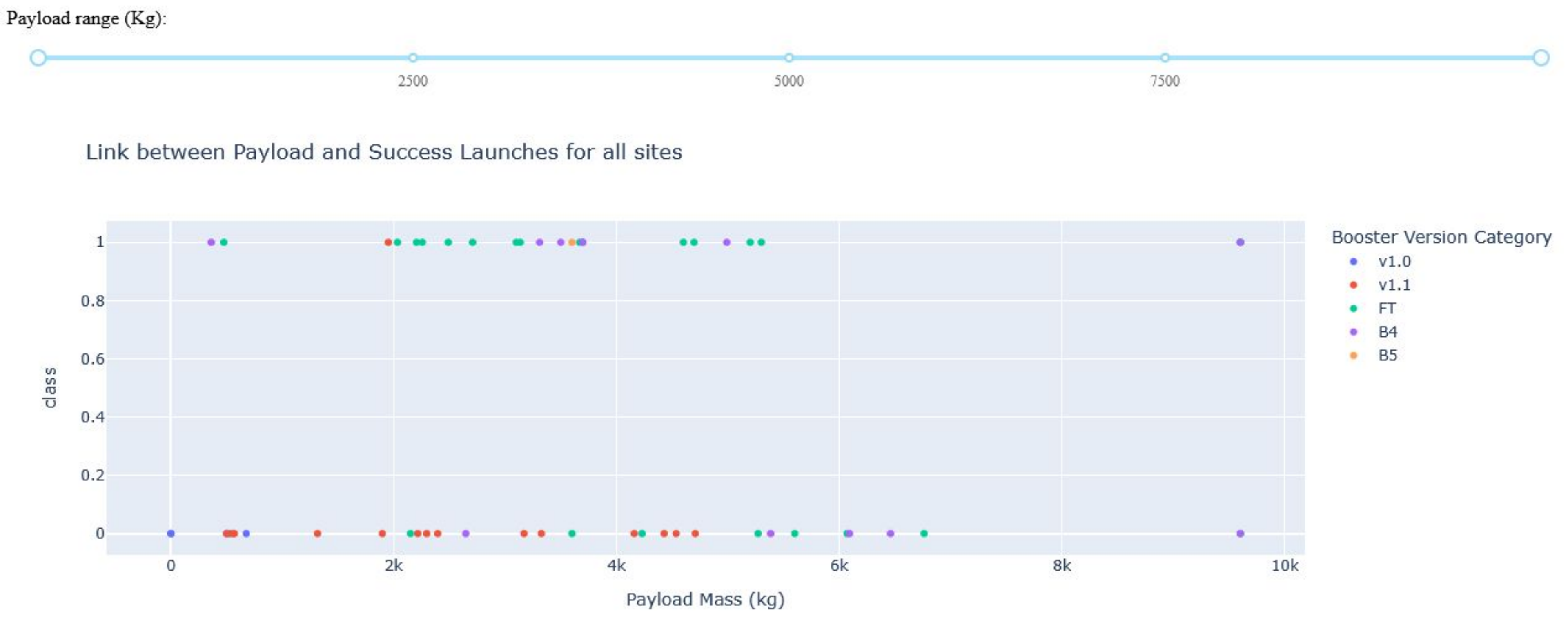
SpaceX Launch Records Dashboard

VAFB SLC-4E

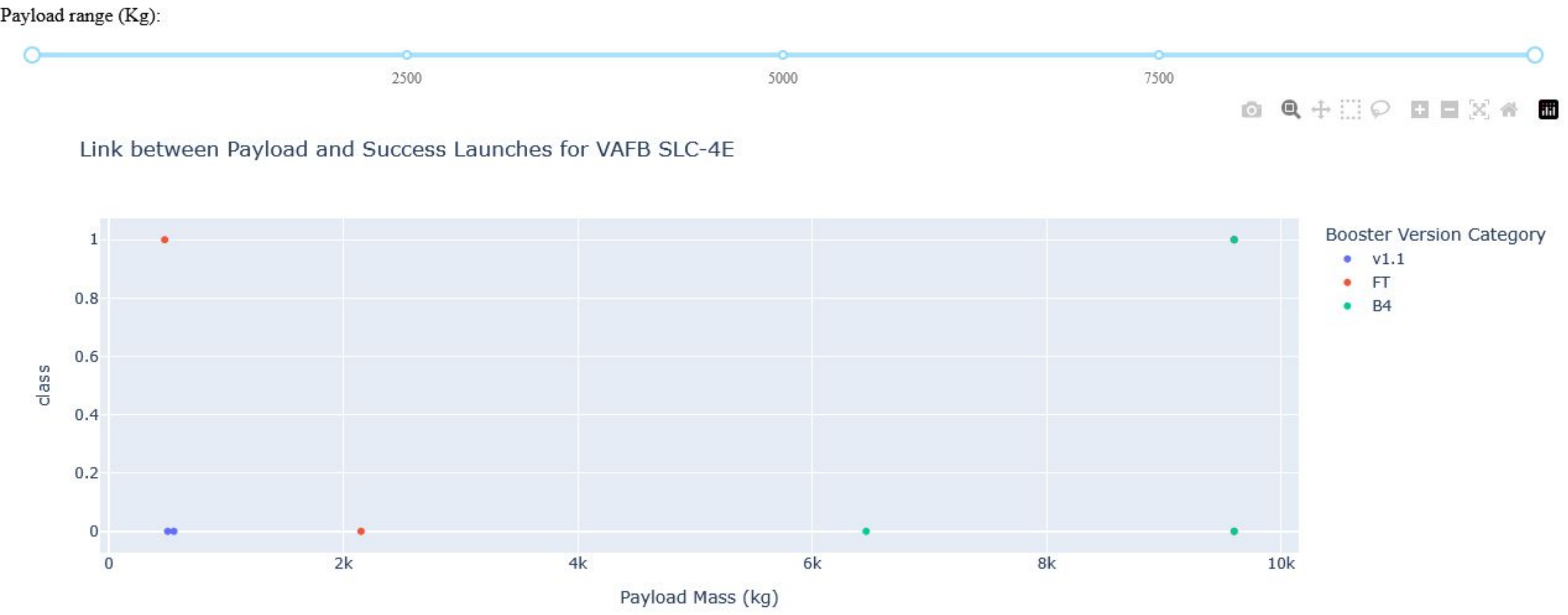
Success Launches for VAFB SLC-4E

Payload range (Kg):

Link between payload and success launches for all sites

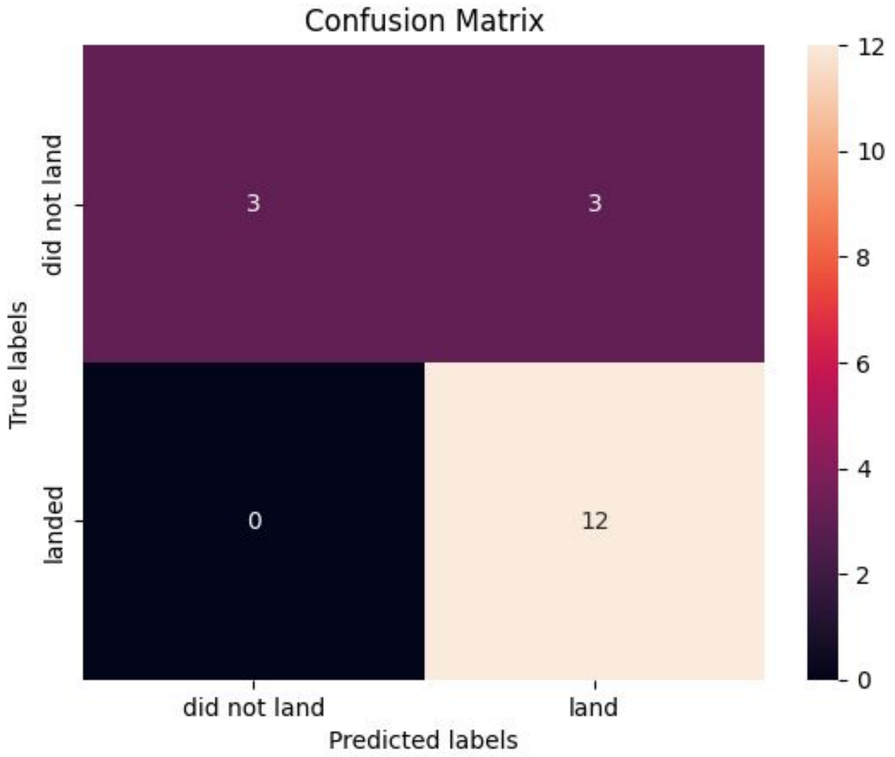


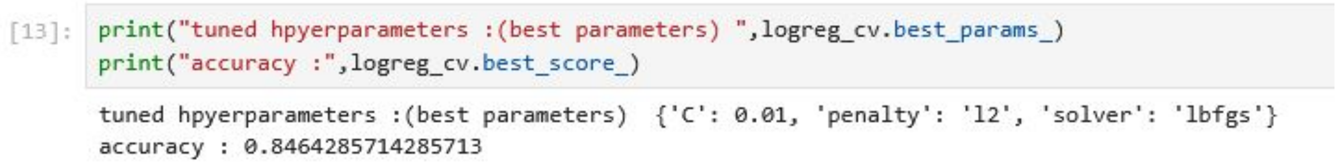
Link between payload and success launch for a specific site



Machine Learning First Stage Landing Prediction

Logistic Regression

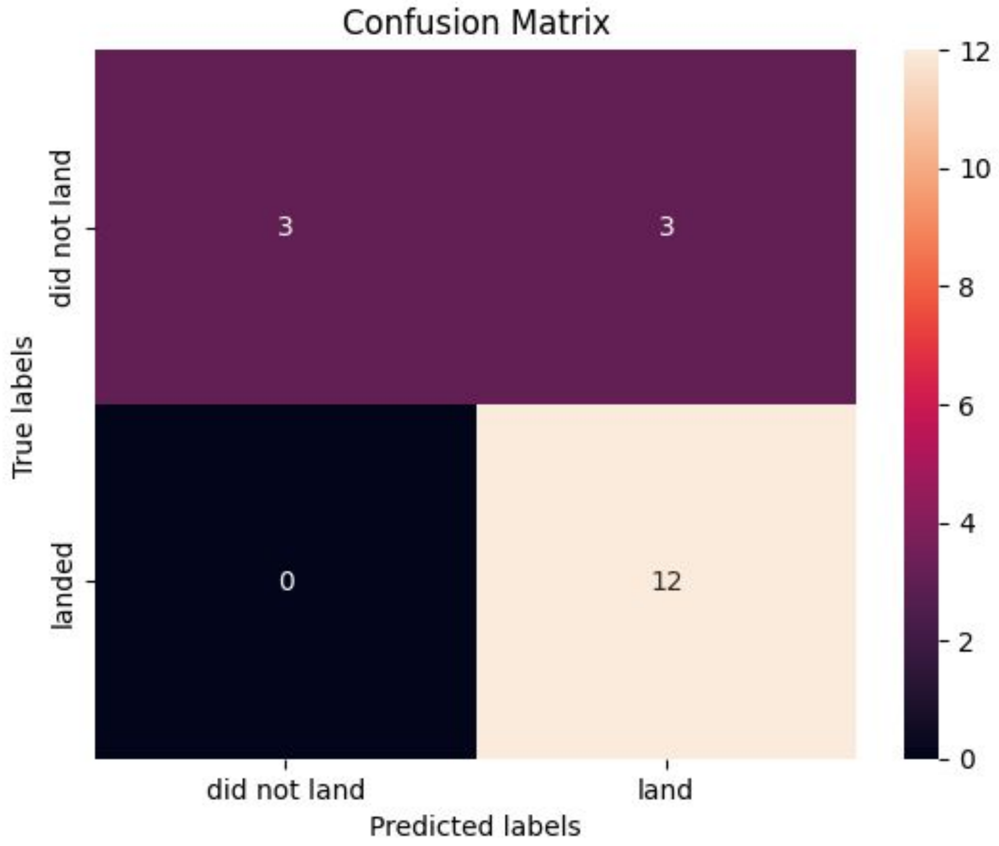




```
[13]: print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
      print("accuracy :",logreg_cv.best_score_)

      tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
      accuracy : 0.8464285714285713
```

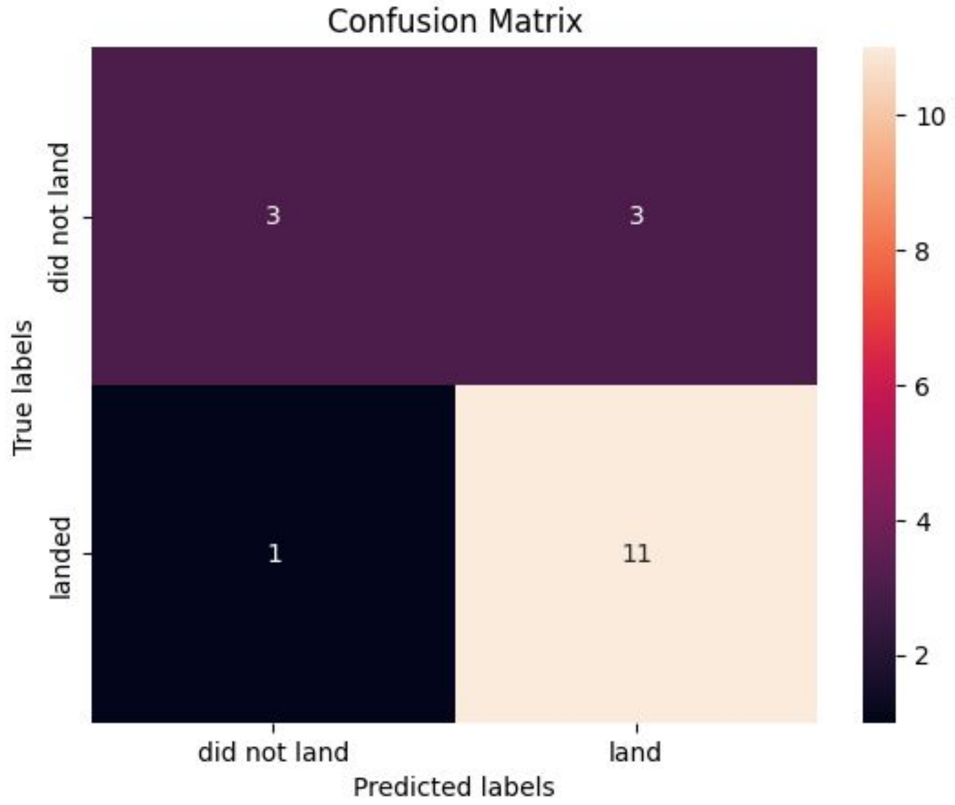
Support Vector Machine

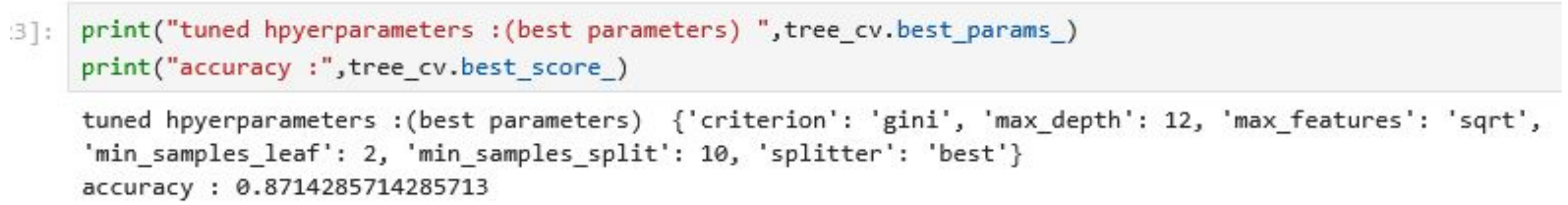


```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

Decision Tree Classifier

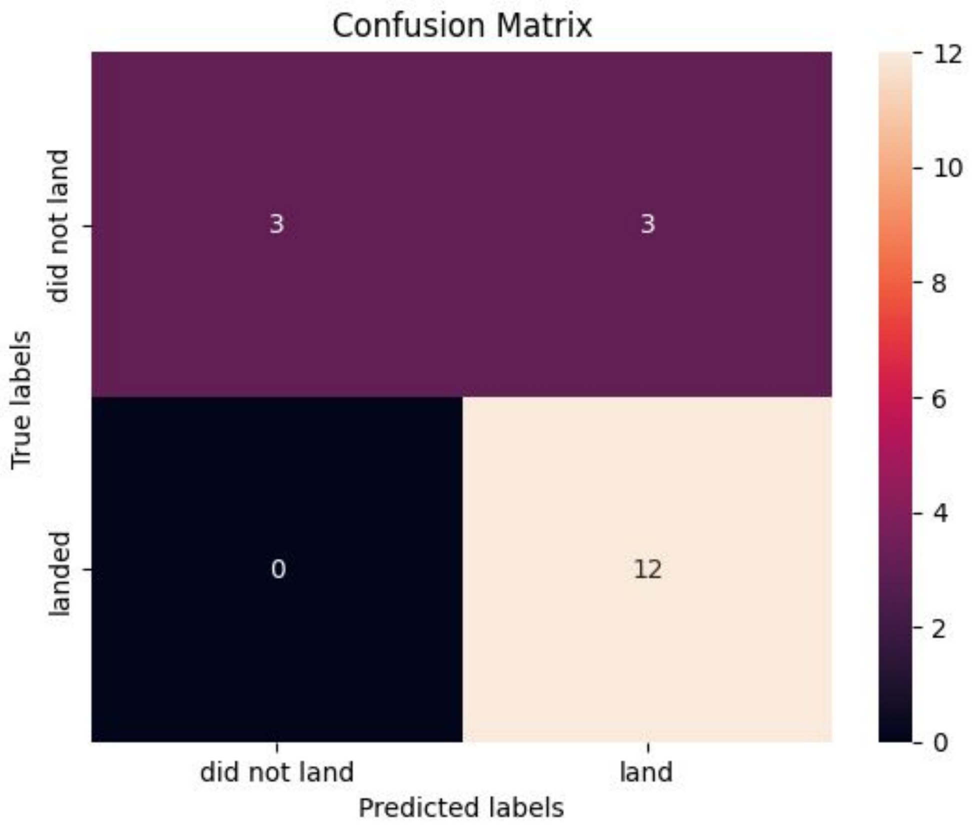




```
3]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
    print("accuracy :",tree_cv.best_score_)

    tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 12, 'max_features': 'sqrt',
    'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'best'}
    accuracy : 0.8714285714285713
```

K Nearest Neighbors (KNN)



```
[29]: print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
      print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

Conclusion

Conclusion

In conclusion, I can say that the most accurate model in this case was Decision Tree Classifier. It provided predictions with 84% accuracy, which is a very good indicator considering the relatively small size of data.