IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Elton Shinji Onari
June, 04 2024

Outline

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

Executive Summary

- Objective: Predict the successful landing of Falcon 9 first stage.
- Methods: API requests, web scraping, data wrangling, EDA, SQL, Folium, Plotly Dash, predictive modeling.
- Key Results: Identified patterns, created interactive dashboards, built predictive models with high accuracy.
- Conclusion: Provided actionable insights for cost-effective rocket launches.

Introduction

- Background: SpaceX's cost-effective Falcon 9 rocket launches.
- Problem Statement: Determine the likelihood of a successful first stage landing.
- Importance: Significant cost savings and competitive advantage in the aerospace industry.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - API requests and web scraping
- Perform data wrangling
  - data cleaning, and formatting
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression, SVM, Decision Trees, KNN.

Data Collection

- Data Sources: SpaceX API, Wikipedia.
- Methods: API requests, web scraping using BeautifulSoup, data cleaning, and formatting.
- Tools: Python, Pandas, Numpy.

Data Collection – SpaceX API

- https://github.com/EltonSO/Applied_DS_Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

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

In [10]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

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

In [11]:
```
response.status_code
```

Out[11]:
```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

In [19]:
```
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url)
data_json = response.json()
data = pd.json_normalize(data_json)
```

Data Collection - Scraping

- https://github.com/EltonSO/Applied_DS_Capstone/blob/main/jupyter-labs-webscraping.ipynb

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

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

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

In [7]:
```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data Wrangling

- Identified missing values and dealt with them
- Calculated number and occurrence of mission outcome of the orbits and created a landing outcome label ('Class') from outcome column

https://github.com/EltonSO/Applied_DS_Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

In [8]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df[['Outcome']].value_counts()
landing_outcomes
```

Out[8]:
```
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
dtype: int64
```

In [33]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

for outcome in df['Outcome']:
    outcome_tuple = (outcome,)
    if outcome_tuple in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [35]:
```
df['Class']=landing_class
df[['Class']].head(5)
```

EDA with Data Visualization

- Initial Analysis: Examined Flight Number and Payload effects on landing success.
- Visualization: Used Matplotlib for scatter plots and bar charts.
- Feature Engineering: Applied OneHotEncoder to categorical variables.

https://github.com/EltonSO/Applied_DS_Capstone/blob/main/edadataviz.ipynb

EDA with SQL

- Unique Launch Sites: Displayed names.
- Records with Specific Launch Sites: Displayed 5 records.
- Total Payload Mass by NASA (CRS): 45596 kg.
- Average Payload Mass by Booster Version F9 v1.1: 2928.4 kg.
- First Successful Ground Pad Landing: 2015-12-22.
- Boosters with Success on Drone Ship and Specific Payload Mass: Listed.
- Mission Outcomes: Total success and failures.
- Booster Versions with Maximum Payload: Listed.
- Month Names and Outcomes in 2015: Detailed records.
- Ranked Landing Outcomes: Displayed count.

https://github.com/EltonSO/Applied_DS_Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Launch Sites Proximity: Equator and coast.
- Success/Failure Markers: Clustered for each site.
- Distance to Proximities: Railways, highways, coastline, cities

https://github.com/EltonSO/Applied_DS_Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Graphs:
  - Success Pie Chart by Launch Site.
  - Success-Payload Scatter Plot.
- Content:
  - Insights: Largest successful launches and highest success rate by site, payload ranges with highest/lowest success rates, F9 Booster version with highest success rate.

https://github.com/EltonSO/Applied_DS_Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Content:
  - Best Model: Decision Tree with 87.32% accuracy.
  - Model Performance: Logistic Regression, SVM, KNN.
  - Hyperparameter Tuning: Optimal parameters for each model.

https://github.com/EltonSO/Applied_DS_Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

In [51]:
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

In [52]:
```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

Out[52]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [53]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_sampl
es_leaf': 1, 'min_samples_split': 5, 'splitter': 'random'}
accuracy : 0.8732142857142857
```

Results

- Exploratory Data Analysis Results:
  - Flight Number vs. Launch Site: Higher flight numbers show increased success across all launch sites.
  - Payload vs. Launch Site: Heavy payloads (>10,000 kg) are absent at VAFB-SLC, impacting success rates.
  - Success Rate by Orbit: ES-L1, GEO, HEO, and SSO have higher success rates.
  - Success Rate Over Time: Success rate increased consistently from 2013 to 2020.

Interactive Analytics Demo:

- Folium Map:
  - Markers indicating success/failure at each launch site.
  - Proximity to Equator, coastlines, and railways highlighted.
- Plotly Dash:
  - Success Pie Chart: KSC LC-39A has the highest success rate (76.9%).
  - Success-Payload Scatter Plot: Payload range 2000-4000 kg shows highest success.

Predictive Analysis Results:

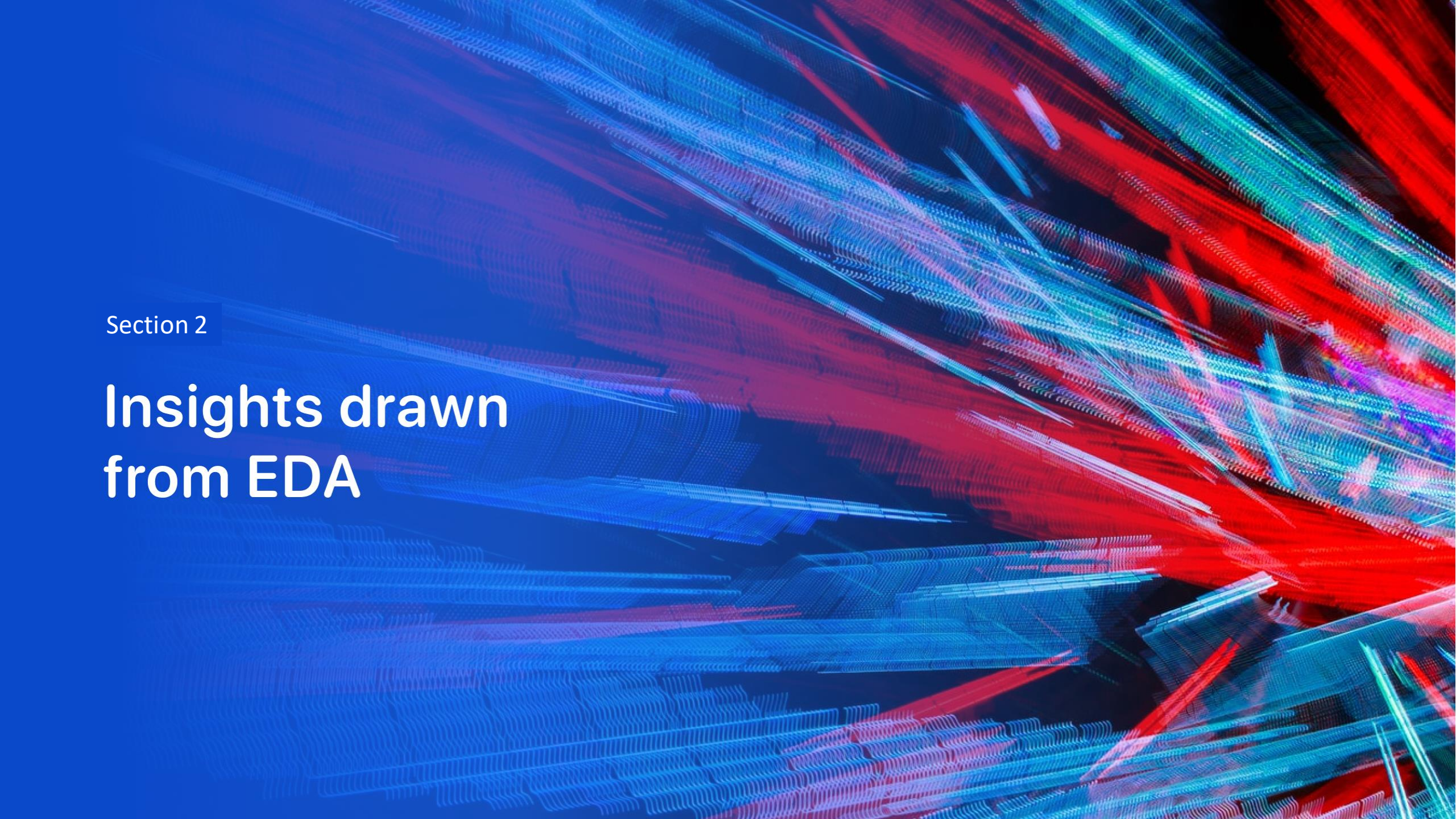
- Best Model: Decision Tree
- Accuracy: 87.32%

Visuals:

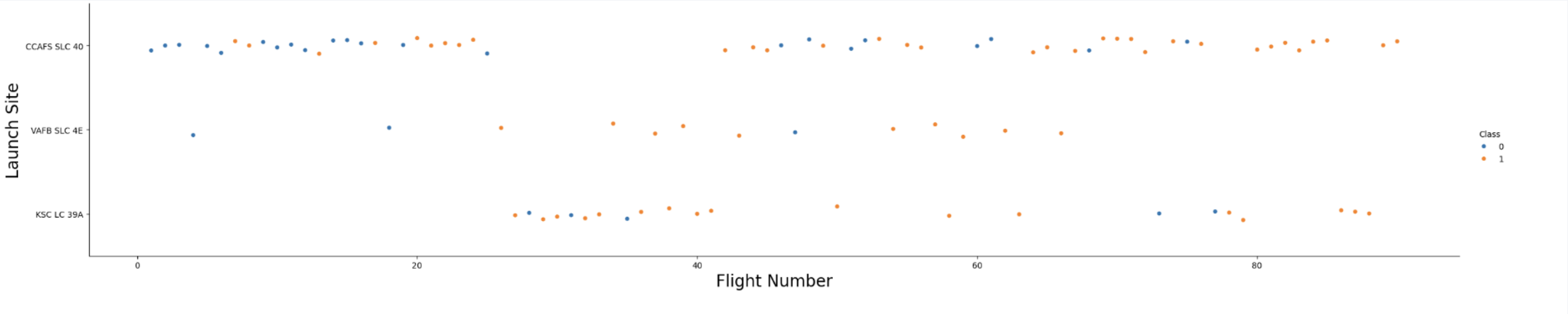
- Screenshots of Folium map and Plotly Dash for interactive insights.

Section 2

Insights drawn from EDA

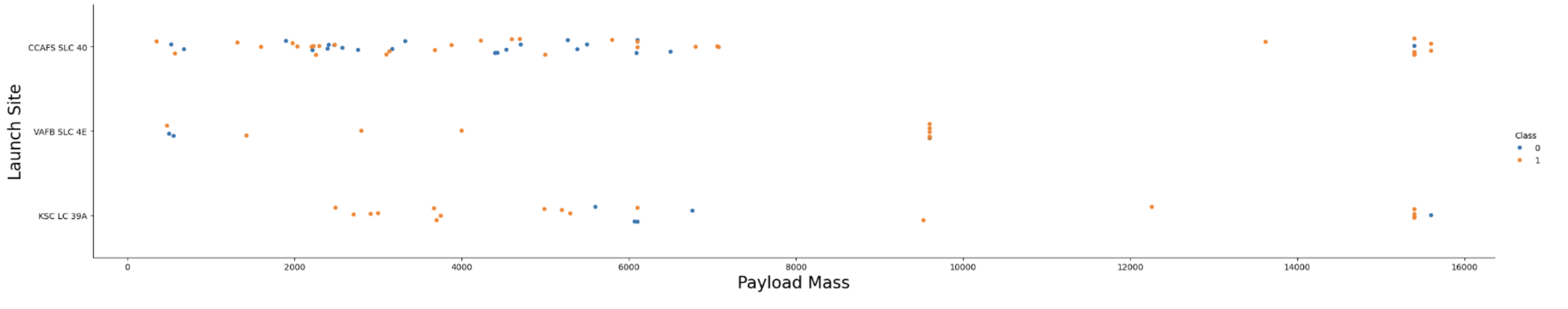


Flight Number vs. Launch Site



- Higher flight numbers show increased success across all launch sites.

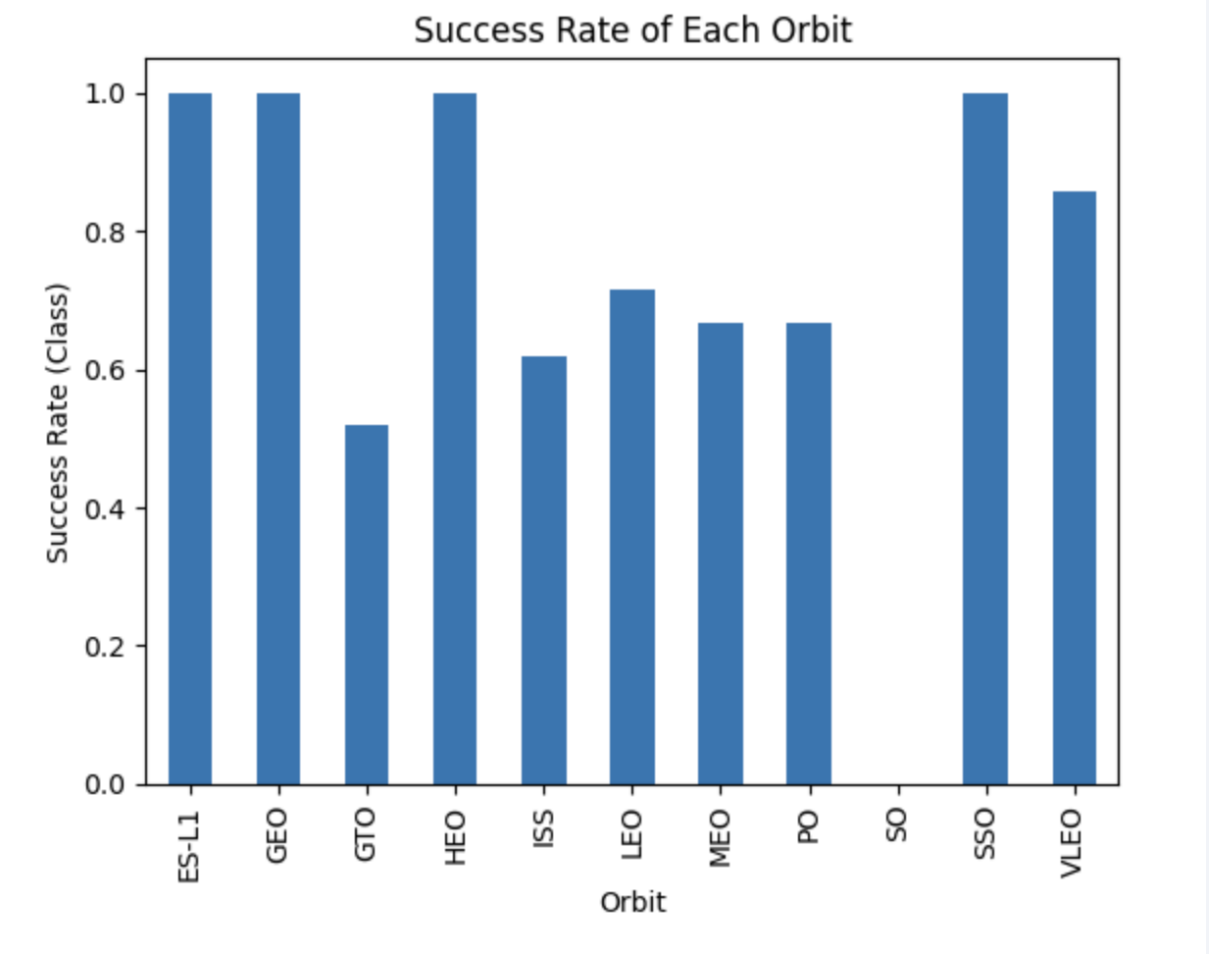
Payload vs. Launch Site



- Heavy payloads (>10,000 kg) are absent at VAFB-SLC, impacting success rates.
- It seems the more massive the payload (> 7,000 kg) , the more likely the first stage will return.

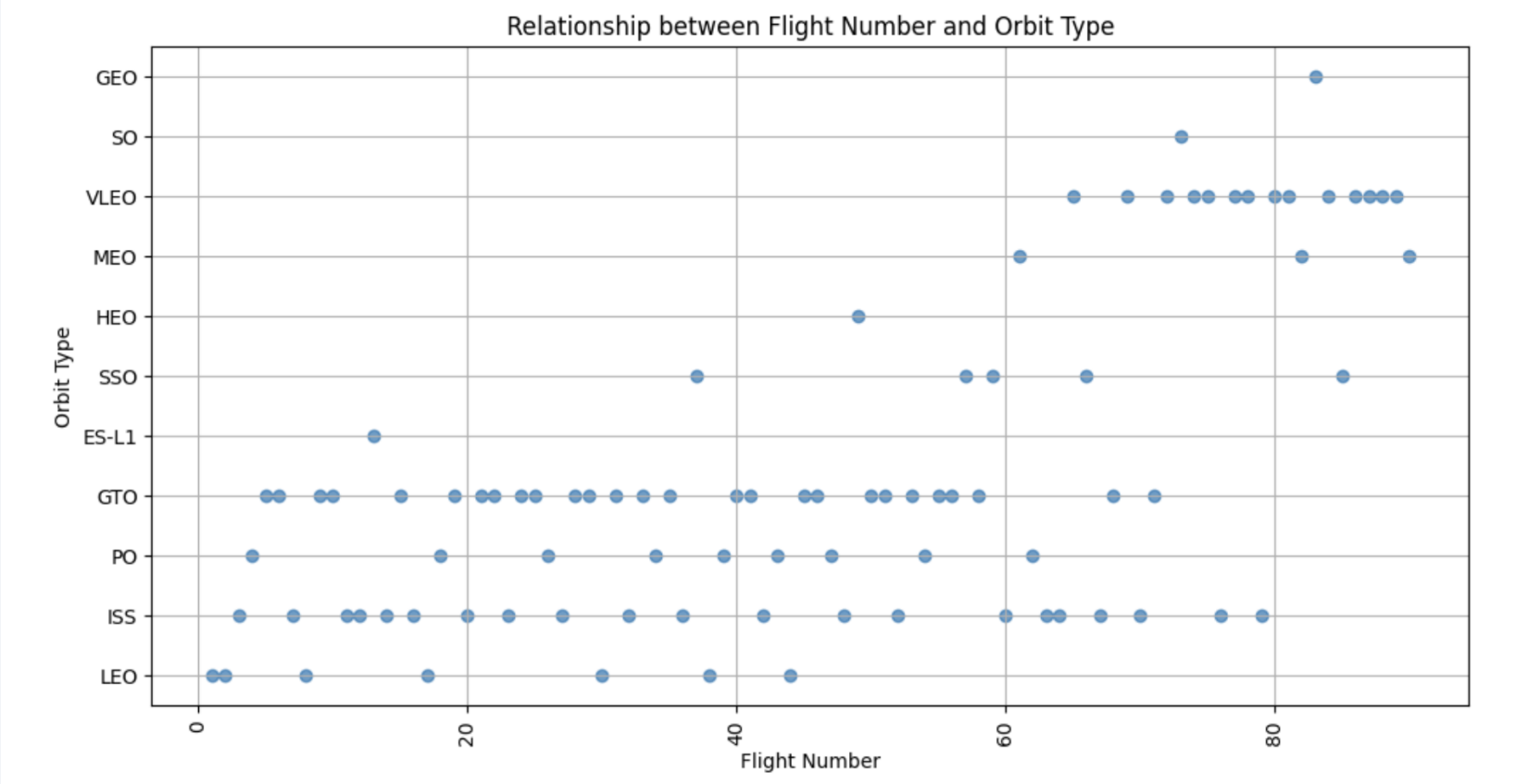
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, and SSO have higher success rates.



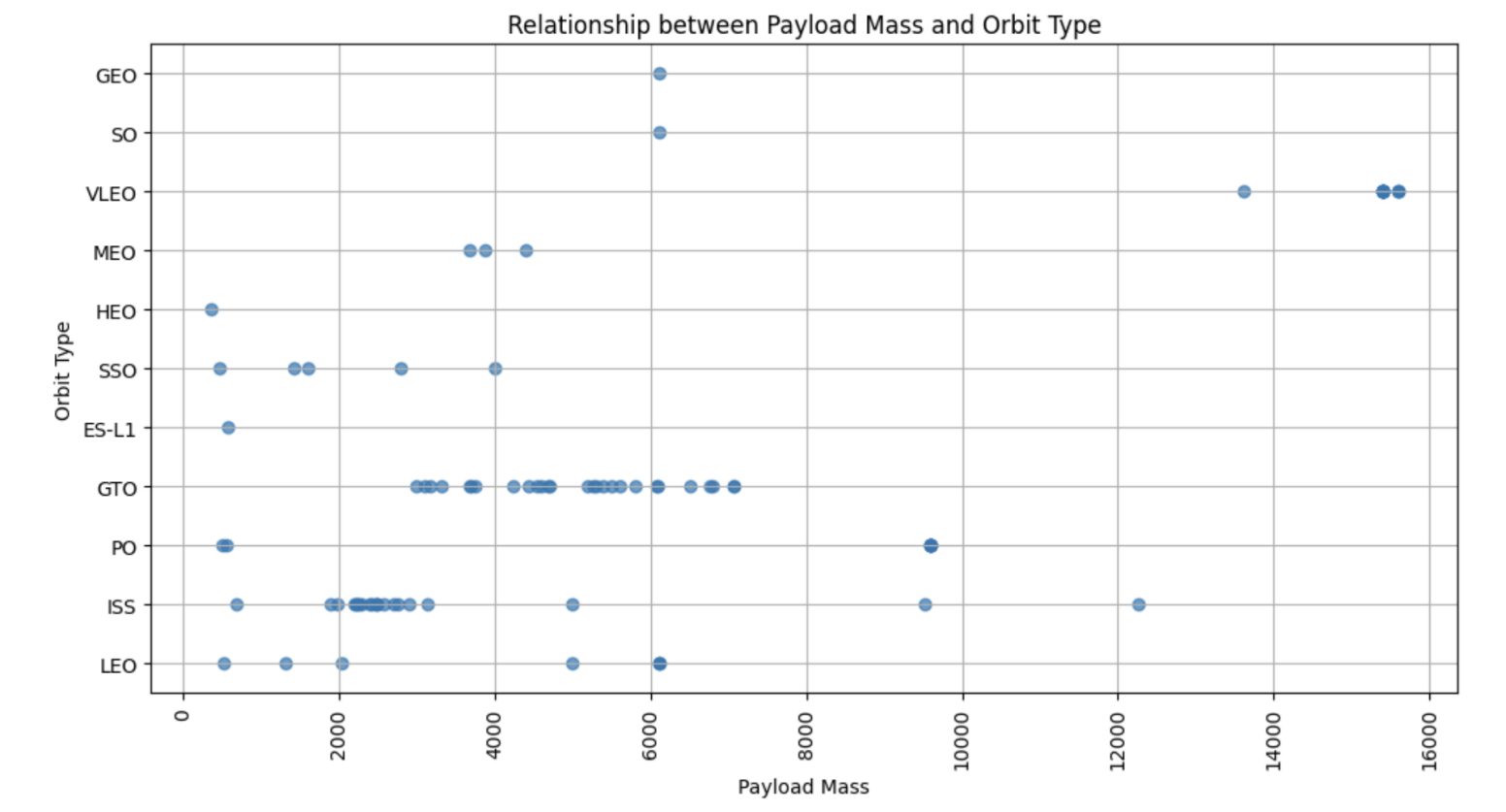
Flight Number vs. Orbit Type

- At beginning, more flights from LEO, ISS, PO and GTO. Then it changed to SSO, MEO, VLEO, SO and GEO.



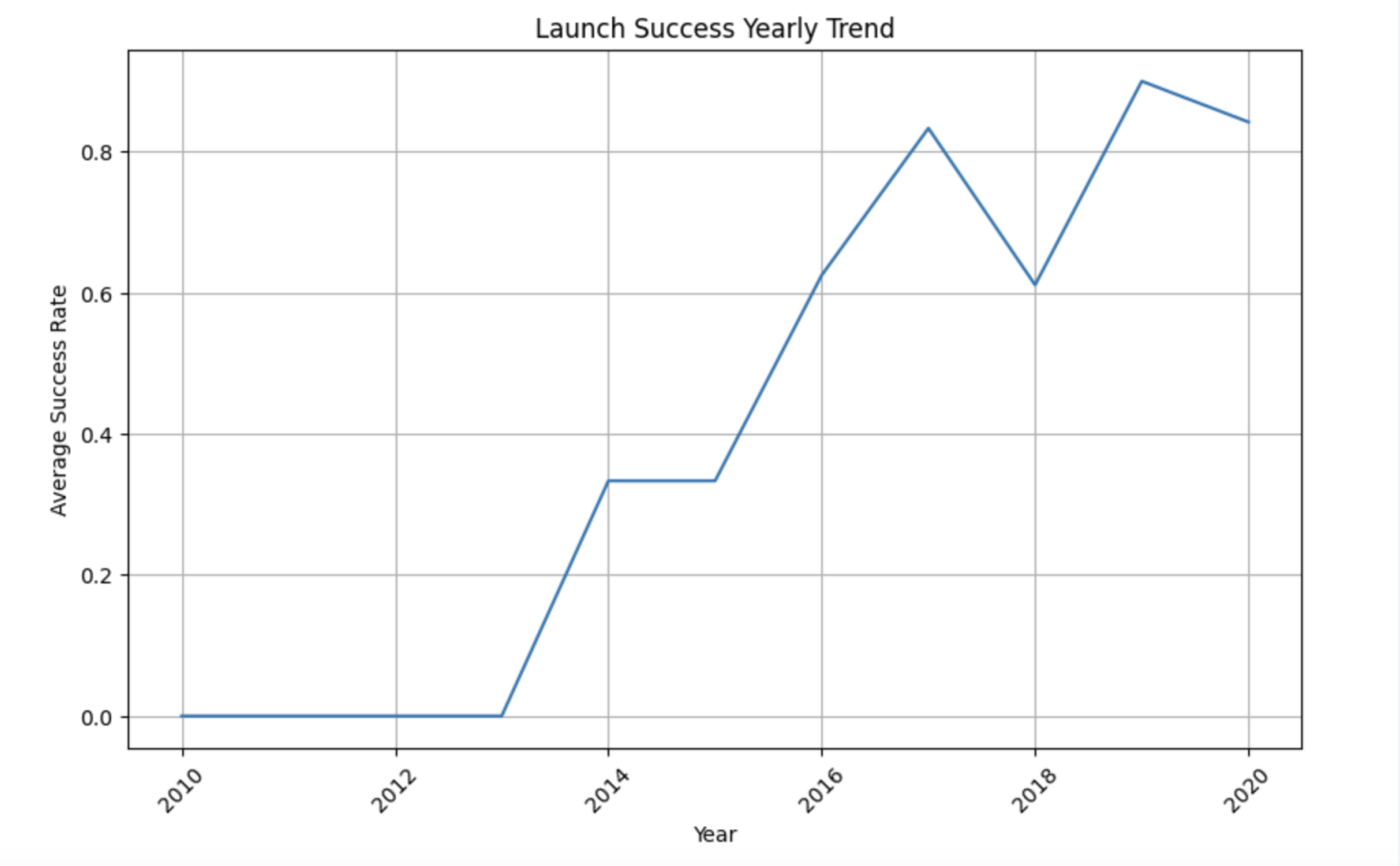
Payload vs. Orbit Type

- With heavy payloads the orbit type more common is VLEO
- For all others orbit types, a lighter PayLoad Mass was prefered



Launch Success Yearly Trend

- Success rate increased consistently from 2013 to 2020.



All Launch Site Names

Task 1

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

In [35]:
```
%sql select distinct "Launch_site" from SPACEXTABLE;
```

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

Out[35]:

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

Launch Site Names Begin with 'CCA'

Task 2

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

In [24]:

```
%%sql
SELECT * FROM SPACEXTABLE
WHERE "Launch_site" LIKE 'CCA%'
LIMIT 5;
```

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

Out[24]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Total Payload Mass

Task 3

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

In [34]:
```
%%sql
select sum("PAYLOAD_MASS__KG_")
from SPACEXTABLE
where "Customer" = "NASA (CRS)";
```

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

Out[34]:

sum("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

Task 4

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

In [33]:

```
%%sql
select avg("PAYLOAD_MASS__KG_") as AVERAGE_PAYLOAD_MASS__KG_
from SPACEXTABLE
where "Booster_Version" =  "F9 v1.1";
```

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

Out[33]:

AVERAGE_PAYLOAD_MASS__KG_
2928.4

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

In [32]:
```
%%sql
select min("Date")
from SPACEXTABLE
where "Landing_Outcome" like "Success%";
```

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

Out[32]:

min("Date")
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

In [30]:
```
%%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.
```

Out[30]:

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

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [36]:
```
%%sql
select "Landing_Outcome", count(*) as Total
from SPACEXTABLE
where "Landing_Outcome" in ('Success', 'Failure')
group by "Landing_Outcome";
```

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

Out[36]:

Landing_Outcome	Total
Failure	3
Success	38

Boosters Carried Maximum Payload

Task 8

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

In [38]:

```
%%sql
select "Booster_Version"
from SPACEXTABLE
where "Payload_Mass__kg_" = (select max("Payload_Mass__kg_") from SPACEXTABLE);
```

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

Out[38]:

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

Task 9

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [39]:

```
%%sql
select
    CASE substr(Date, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END as Month,
    "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.
```

Out[39]:

Month	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

In [40]:

```
%%sql
select "Landing_Outcome", count(*) as "Outcome_Count"
from SPACEXTABLE
where Date between '2010-06-04' and '2017-03-20'
group by "Landing_Outcome"
order by "Outcome_Count" desc;
```

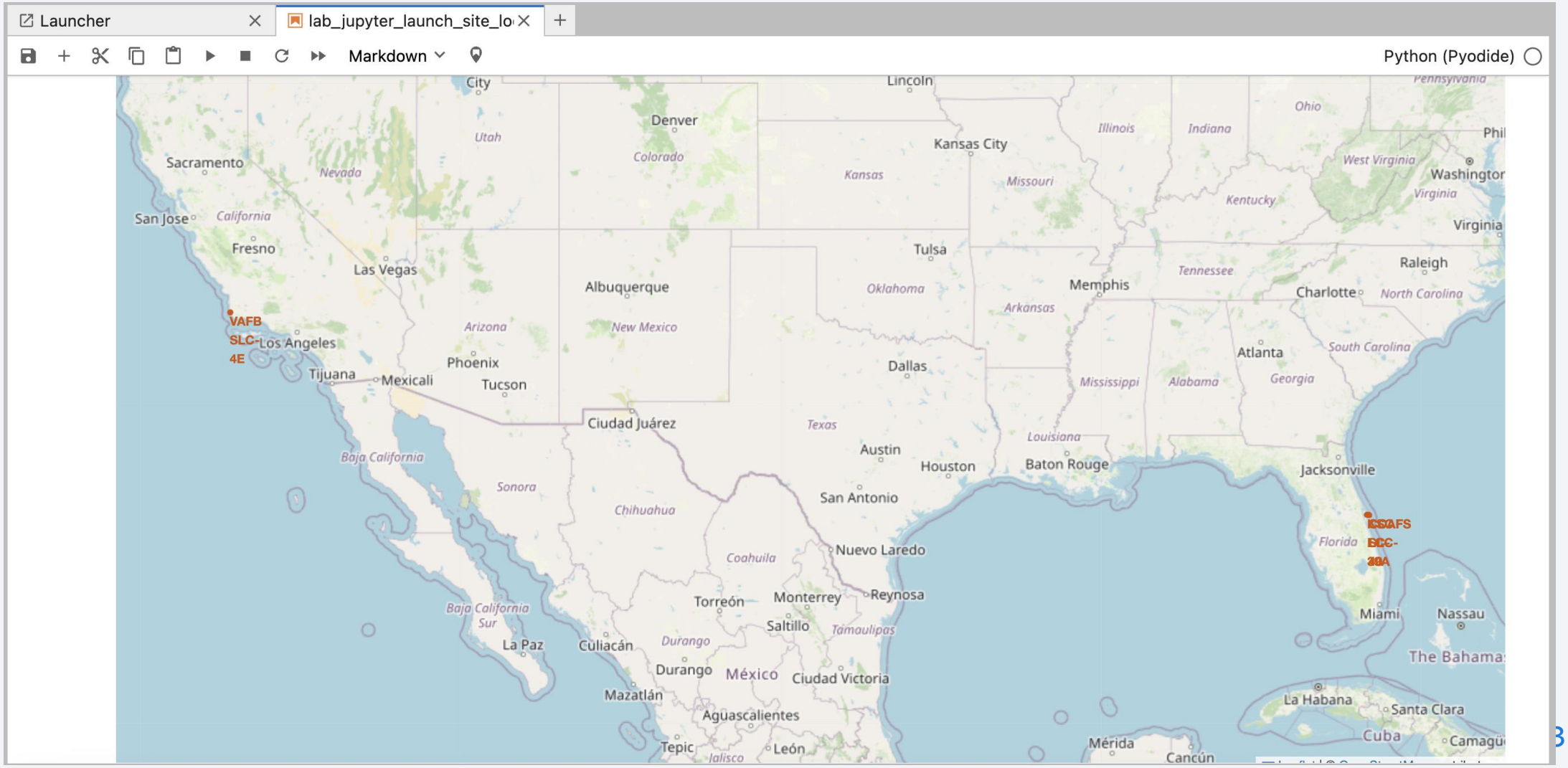
```
 * sqlite:///my_data1.db
Done.
```

Out[40]:

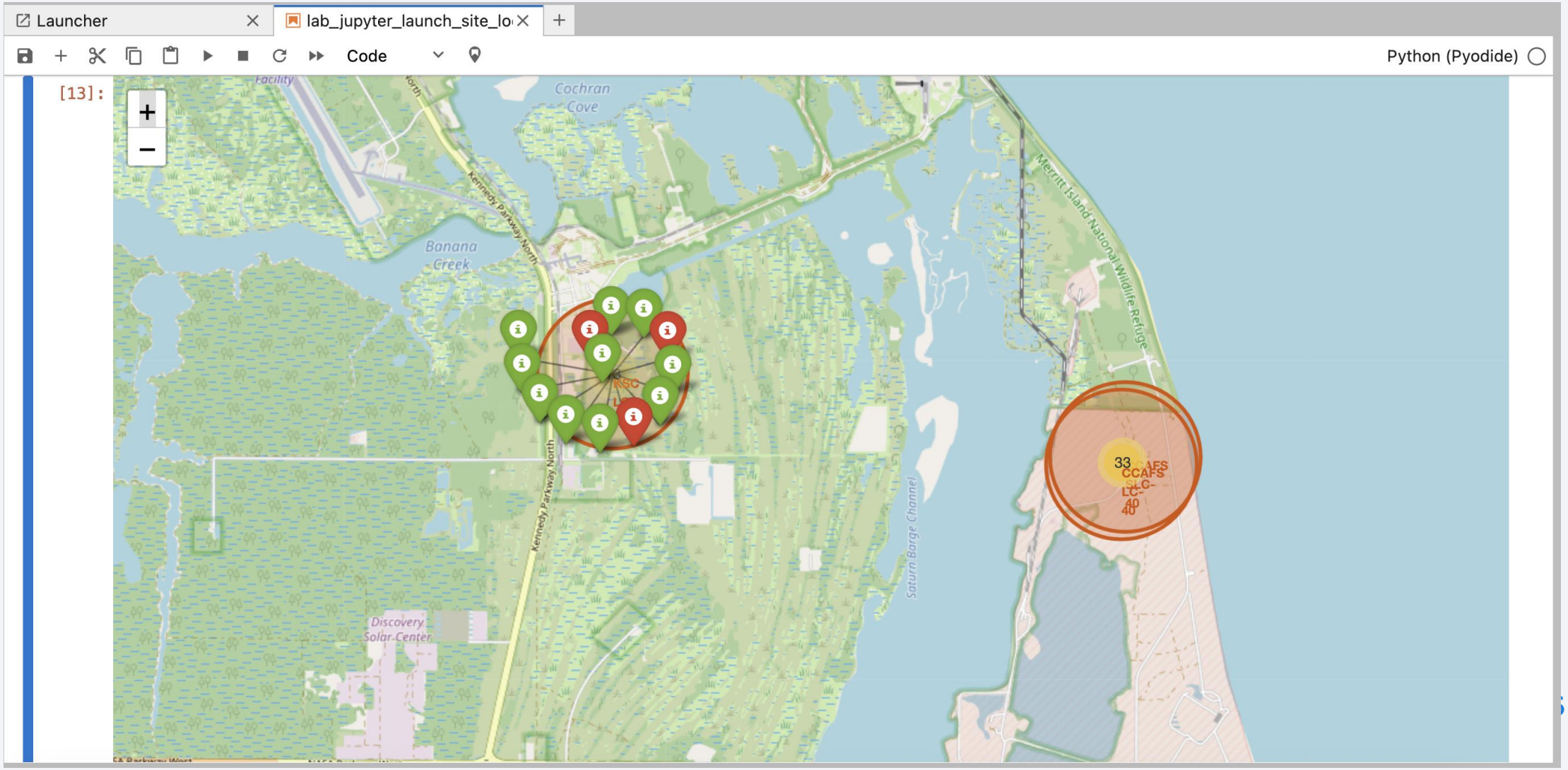
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1


Section 3
Launch Sites
Proximities Analysis

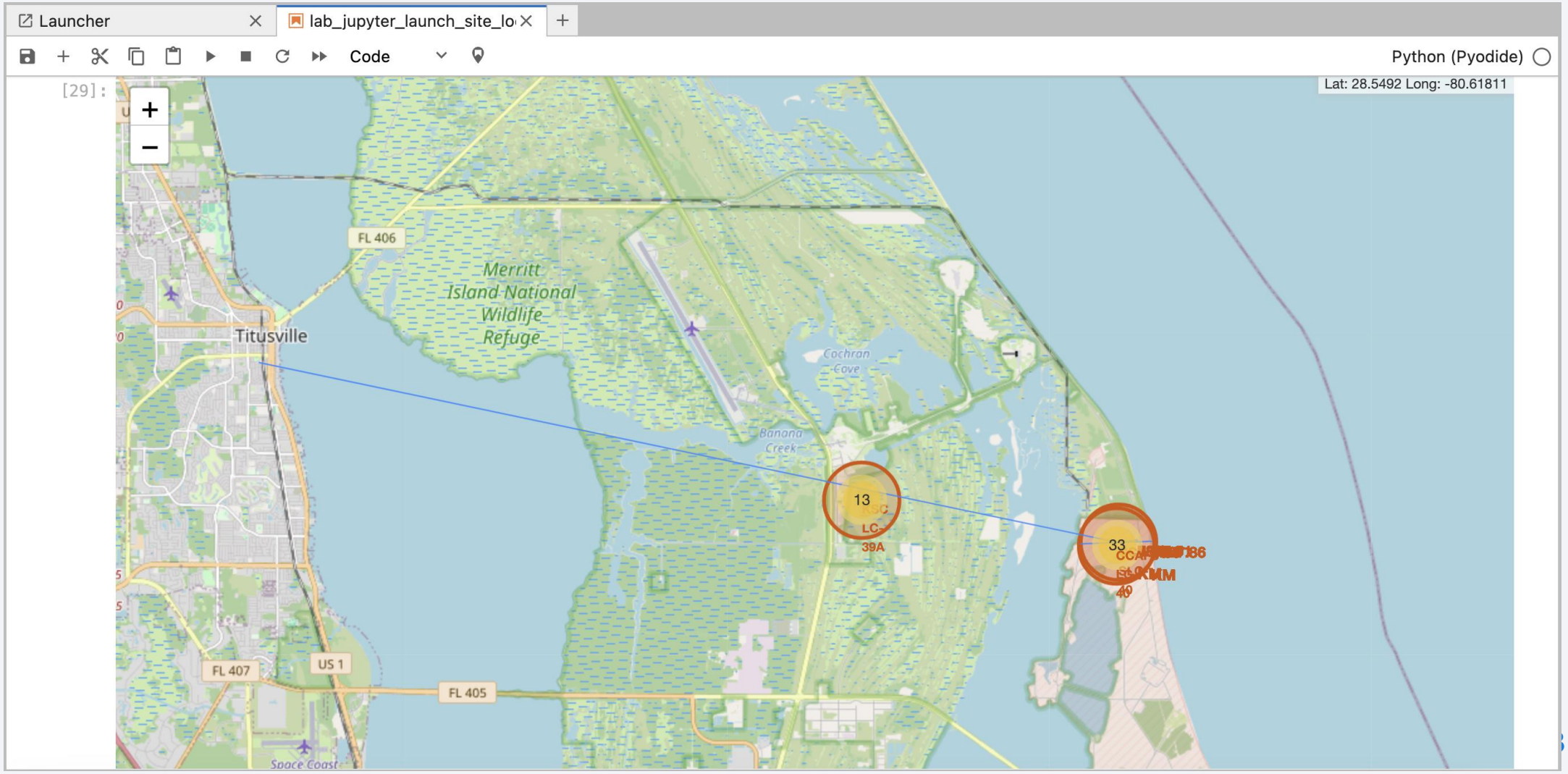
All launch sites on a global map

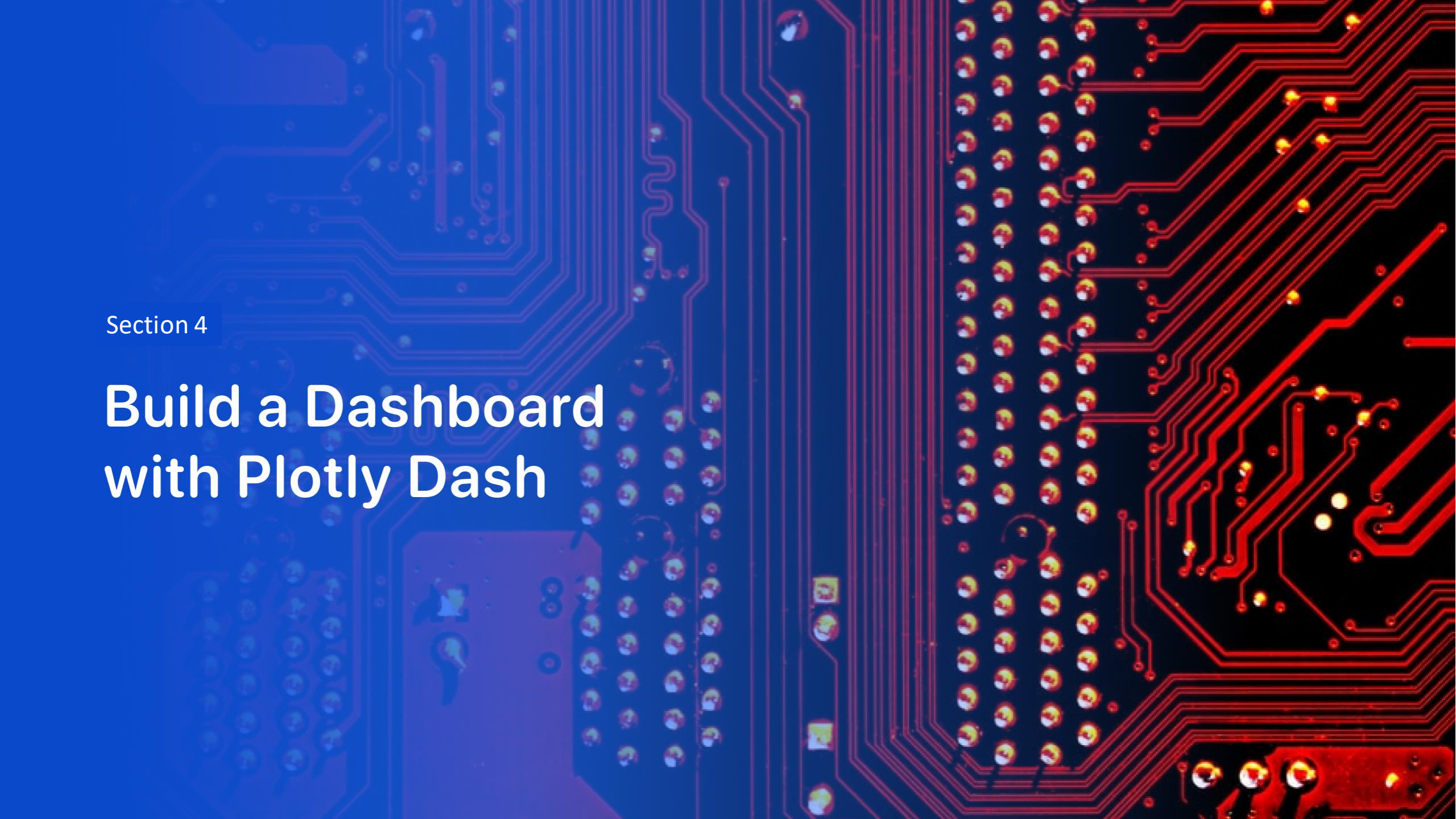


Marked the success/failed launches for each site on the map



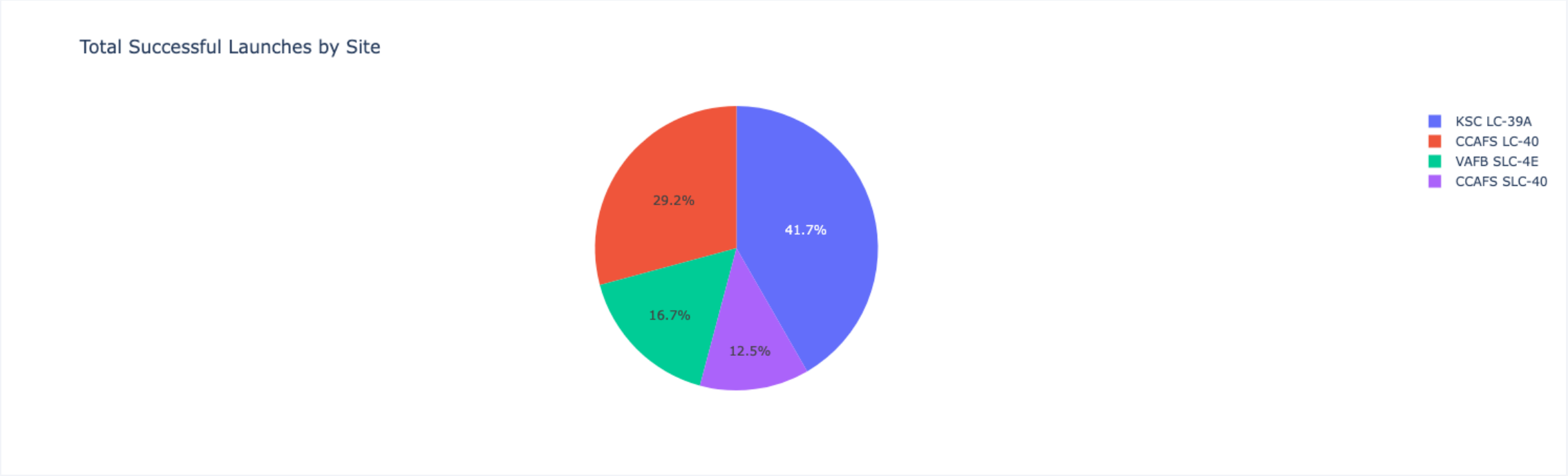
Calculated the distances between a launch site to its proximities




Section 4
Build a Dashboard with Plotly Dash

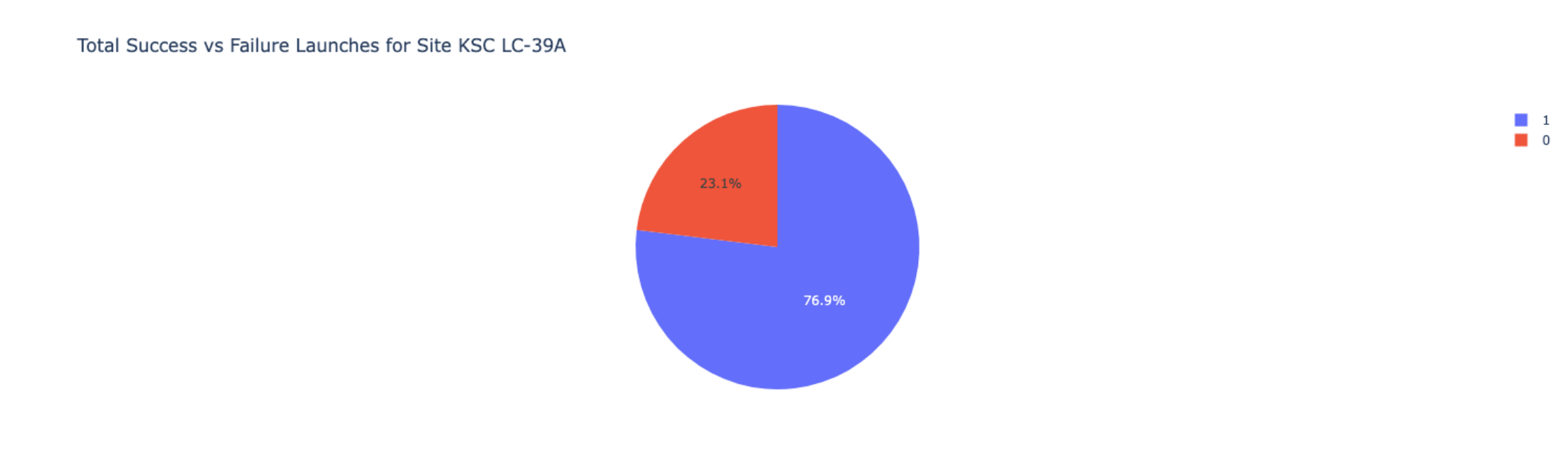
Total Successful Launches by Site using Plotly

- KSC LC-39A with 41,7% is the most successful site



<Dashboard Screenshot 2>

- KSC LC-39A had 76,9% of success rate



Payload vs. Launch Outcome scatter plot for all sites Dashboard using Plotly

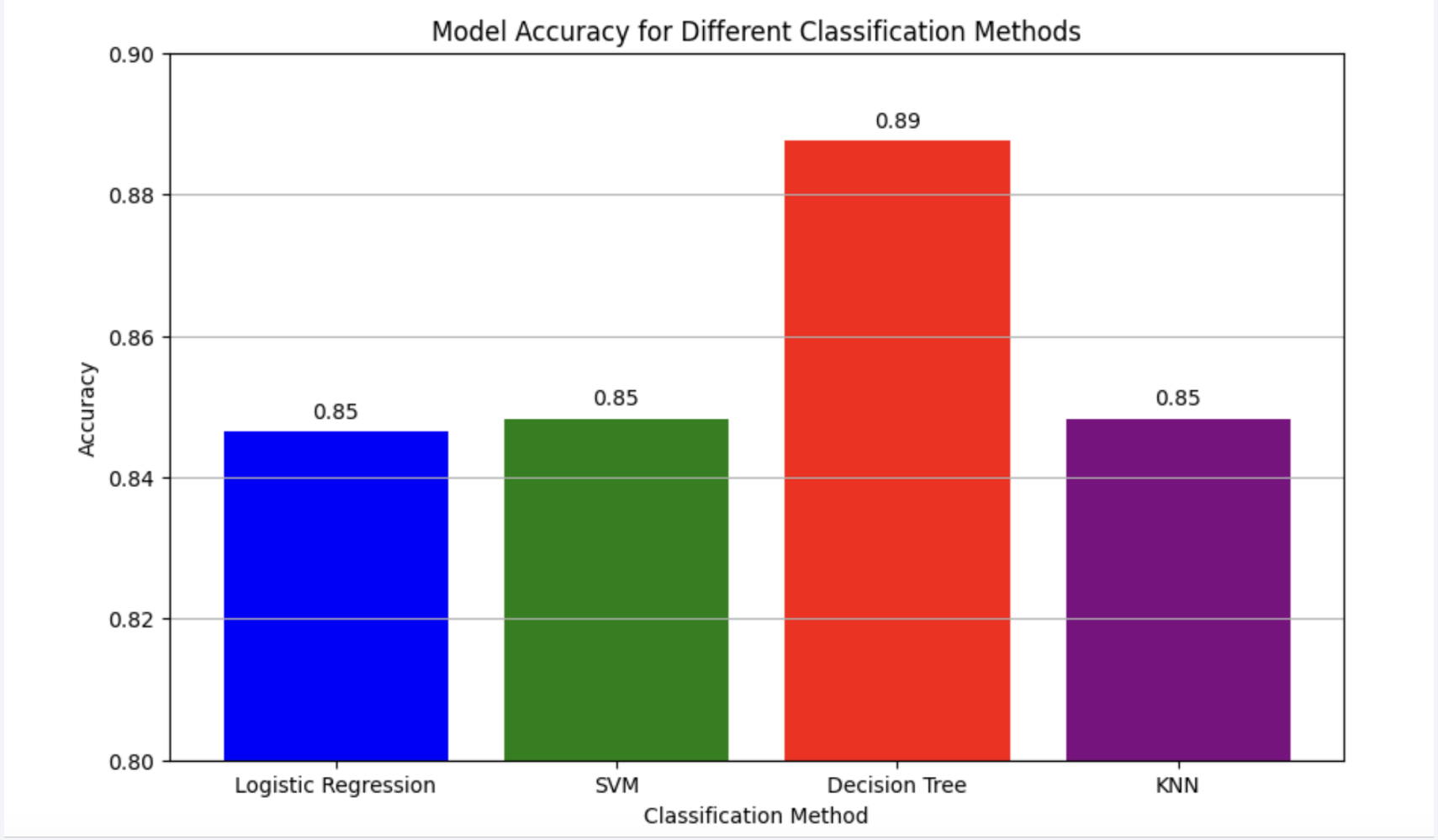
- Payload range with highest launch success rate: 2000 - 4000 kg
- Booster Version Category with highest launch success rate: FT

Section 5

Predictive Analysis (Classification)

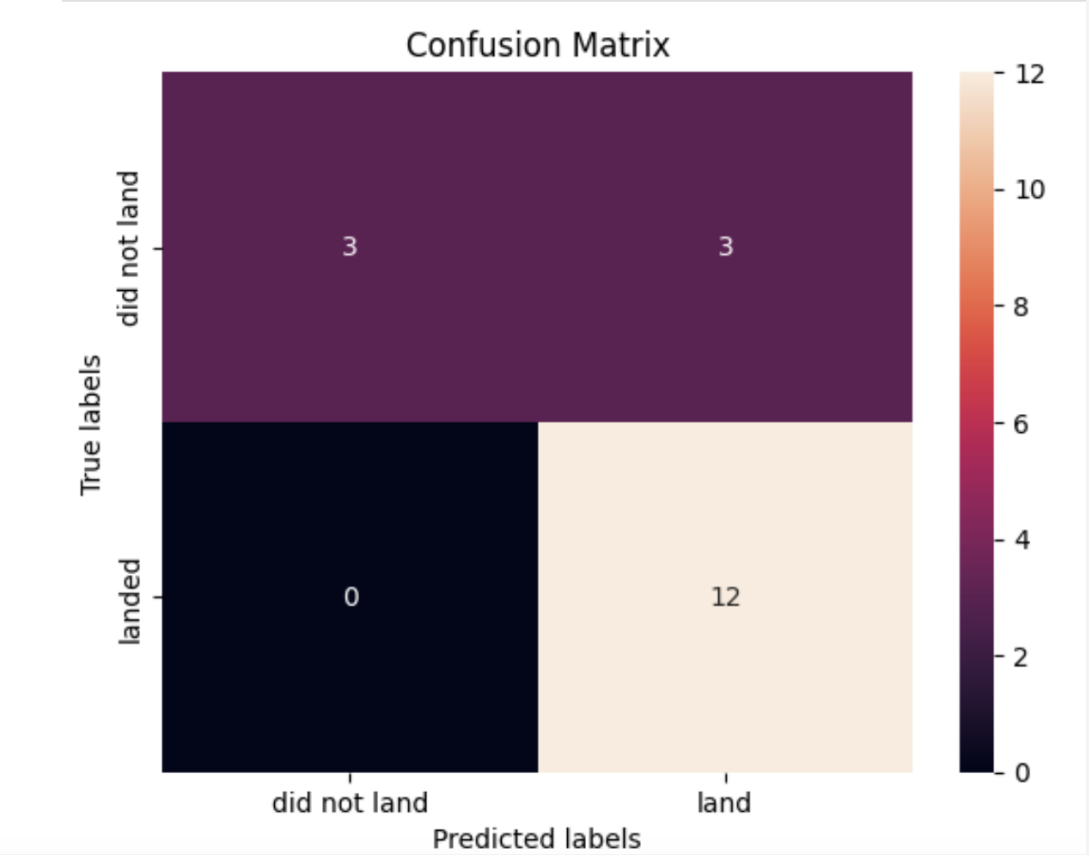
Classification Accuracy

- Decision Tree Classification has the best accuracy



Confusion Matrix

- Major problem is false positives.
- No false negatives and 100% true positives



Conclusions

- Summary of Findings: Key insights from EDA, SQL, Folium, Plotly Dash, and predictive models.
- Impact: Data-driven decision-making for cost-effective launches.
- Future Work: Further model improvements, additional features, real-time data integration.

Appendix

- GitHub link: https://github.com/EltonSO/Applied_DS_Capstone/tree/main

Thank you!