IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Xun Zhu
2025.3

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project Context:

- SpaceX's Falcon 9 reusability reduces launch costs from $165M to $62M.
- Objective: Predict first-stage landing success to estimate mission costs.
- Key Questions:

- Which launch sites and payload ranges correlate with higher success rates?
- How do booster versions and orbits impact landing outcomes?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected : with api。use python request to obtain data。then covert to datafrme for future use。BeautifulSoup package to web scrape some HTML
- Perform data wrangling
  - Describe how data was processed : with api sampling data dealing with null filter data calculate mean and replace null one hot encoding
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

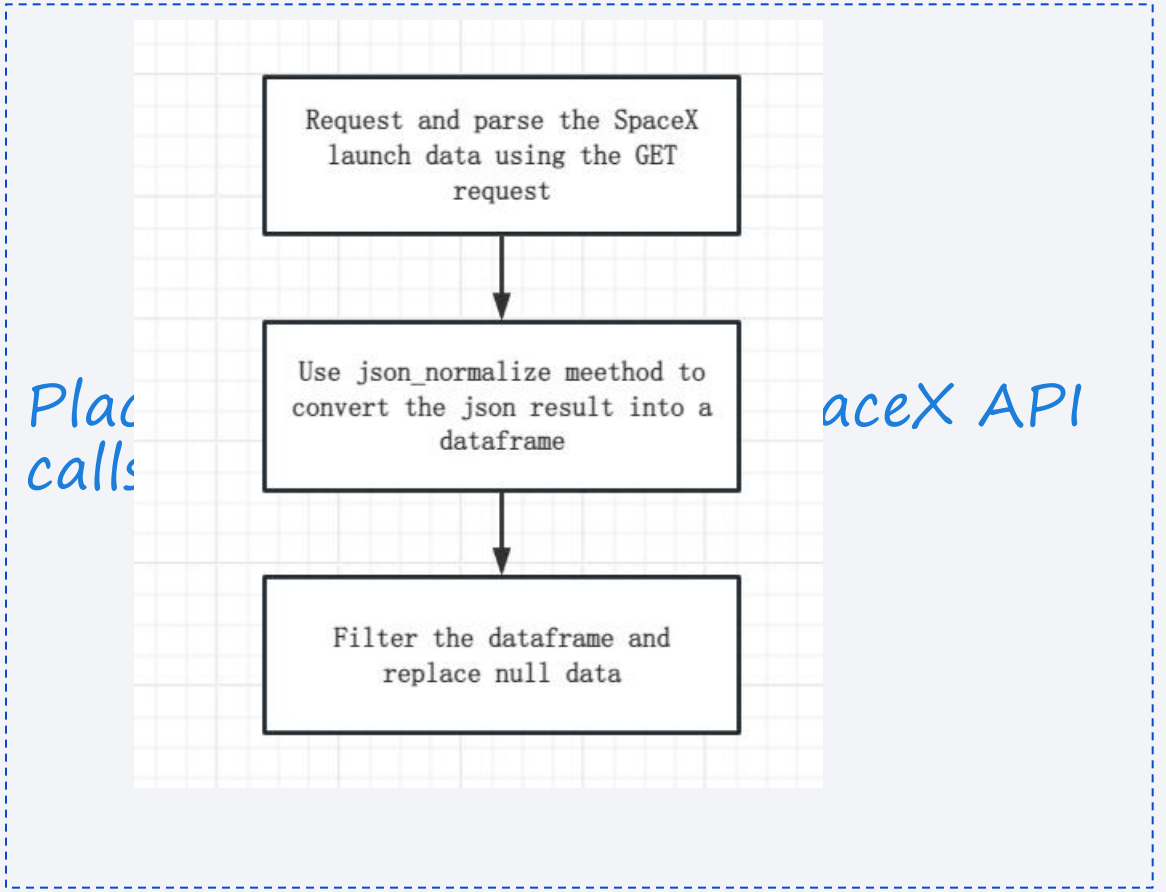
Data Collection

- Describe how data sets were collected.
- request.get()
- data1=response.json()
- data=pd.json_normalize(data1)
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

- Use request.get()
- Use repalce to get rid of the null data

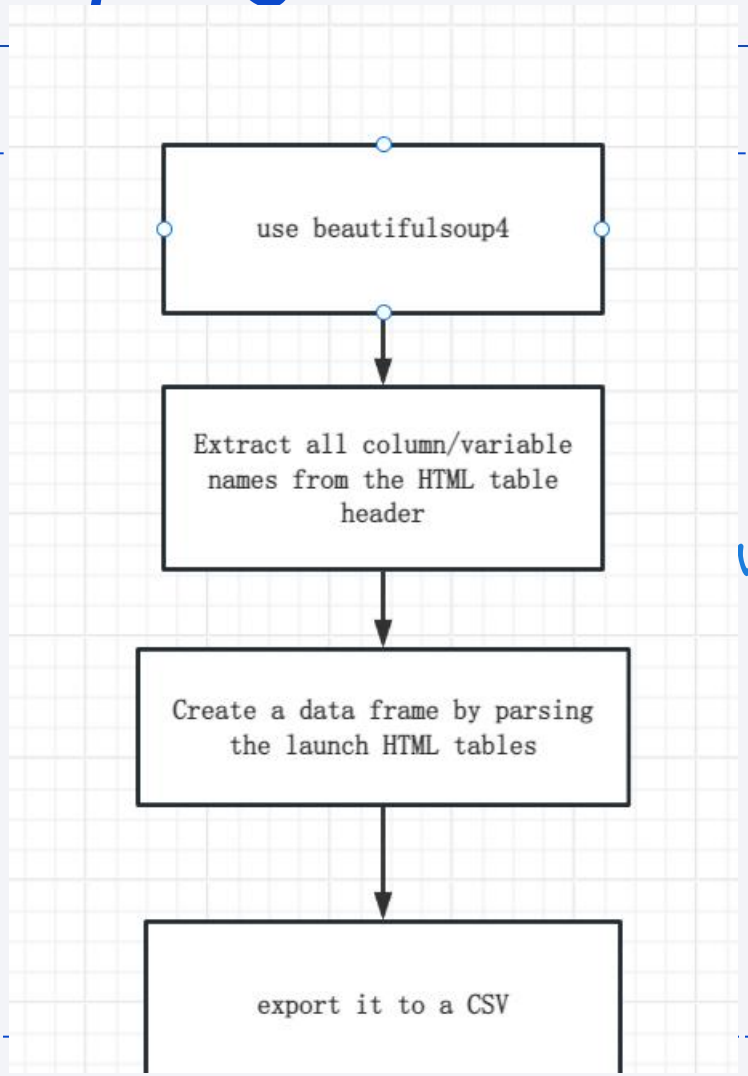
- https://github.com/Emma0804/IBM_Data_Science-project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



Data Collection - Scraping

- useBeautifulSoup(response.text, 'html.parser')
- use find_all() to extract data
- use pd.DataFrame() to get a dataframe dataset

- https://github.com/Emma0804/IBM_Data_Science-project/blob/main/jupyter-labs-webscraping-bak-2025-03-07-07-08-20Z.ipynb



web scraping

Data Wrangling

- https://github.com/Emma0804/IBM_Data_Science-project/blob/main/jupyter-labs-webscraping-bak-2025-03-07-07-08-20Z.ipynb

EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts
- catplot
- scatterplot
- plot(line chart)
- https://github.com/Emma0804/IBM_Data_Science-project/blob/main/edadataviz.ipynb

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
- DISTINCT/where like
- count/sum/avg/min
- https://github.com/Emma0804/IBM_Data_Science-project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

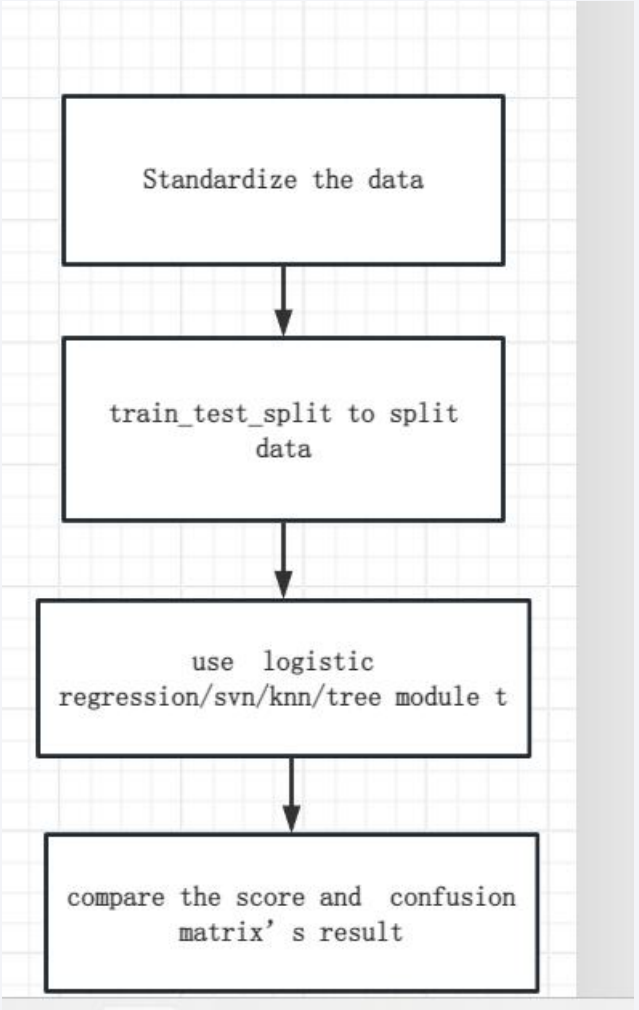
- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
- circle/marker/PolyLine
- Explain why you added those objects
- the reason to use map to get a clear impression of the realtion between a location and the success rate.
- https://github.com/Emma0804/IBM_Data_Science-project/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- piechart
- scatterplot
- Explain why you added those plots and interactions
- to see the relationship of locaiton/Payload Mass and success rate
- https://github.com/Emma0804/IBM_Data_Science-project/blob/main/dash_launch.py

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- use logistic regression/svn/knn/tree module
- then compare the score and confusion matrix's result
- https://github.com/Emma0804/IBM_Data_Science-project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb



Results

- Exploratory data analysis results

```
accuracy = logreg_cv.score(X_test,Y_test)
```

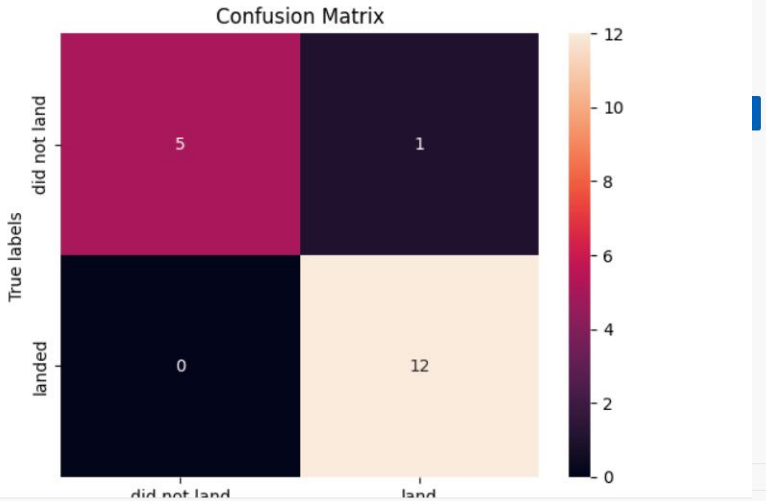
```
[23]
```

```
0.9444444444444444
```

Lets look at the confusion matrix:

```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

```
[24]
```



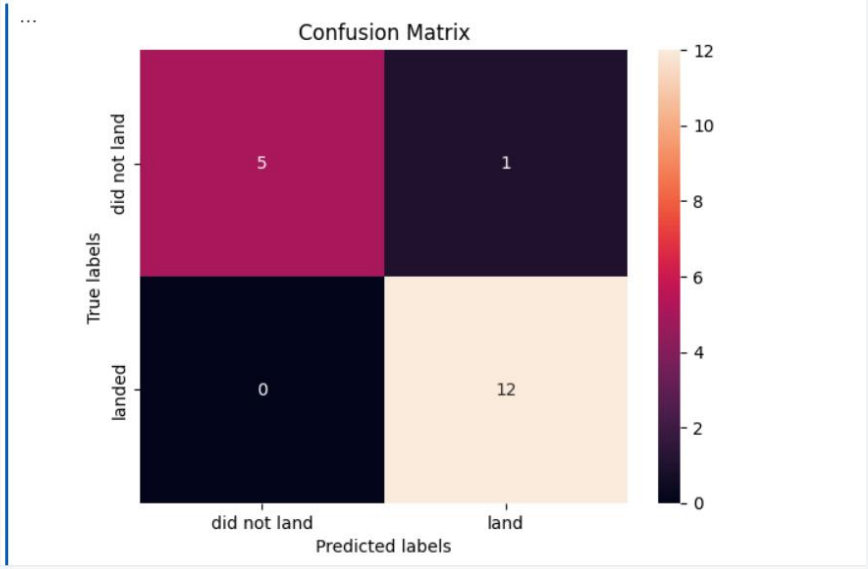

```
[30]
```

```
0.9444444444444444
```

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

```
[31]
```

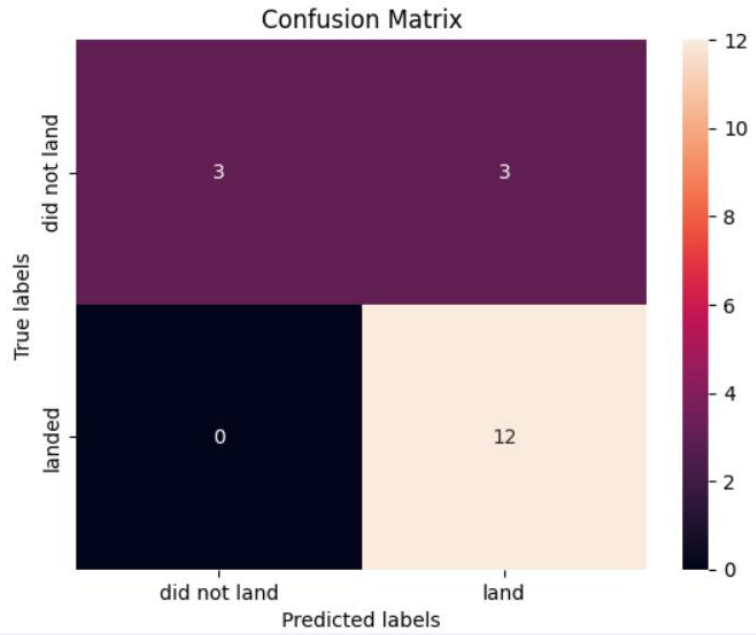


Results

0.8333333333333334

We can plot the confusion matrix

```
[37]
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```




```
[43]
accuracy_knn
```

0.9444444444444444

We can plot the confusion matrix

```
[44]
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Section 2

Insights drawn from EDA

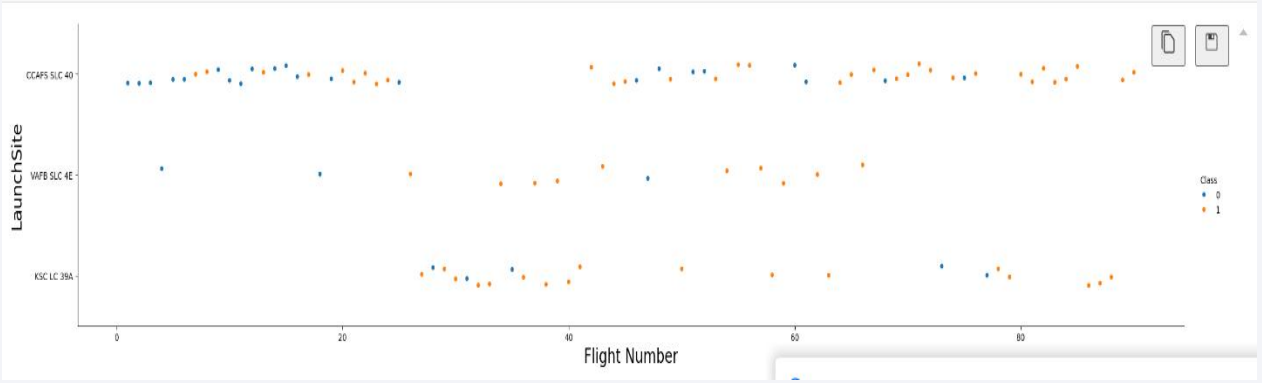
Flight Number vs. Launch Site

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

- Show the screenshot of the scatter plot with explanations

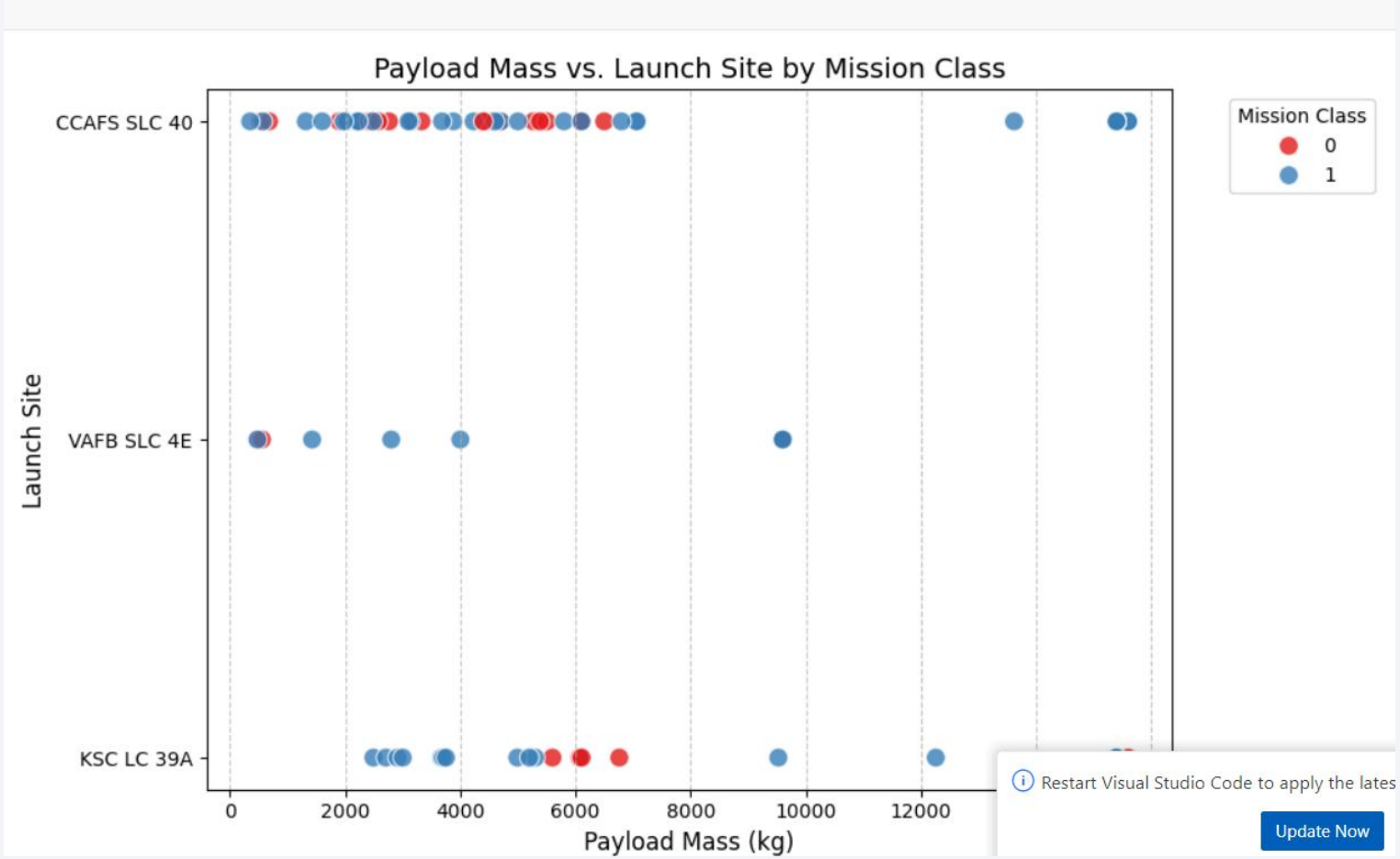
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

Python



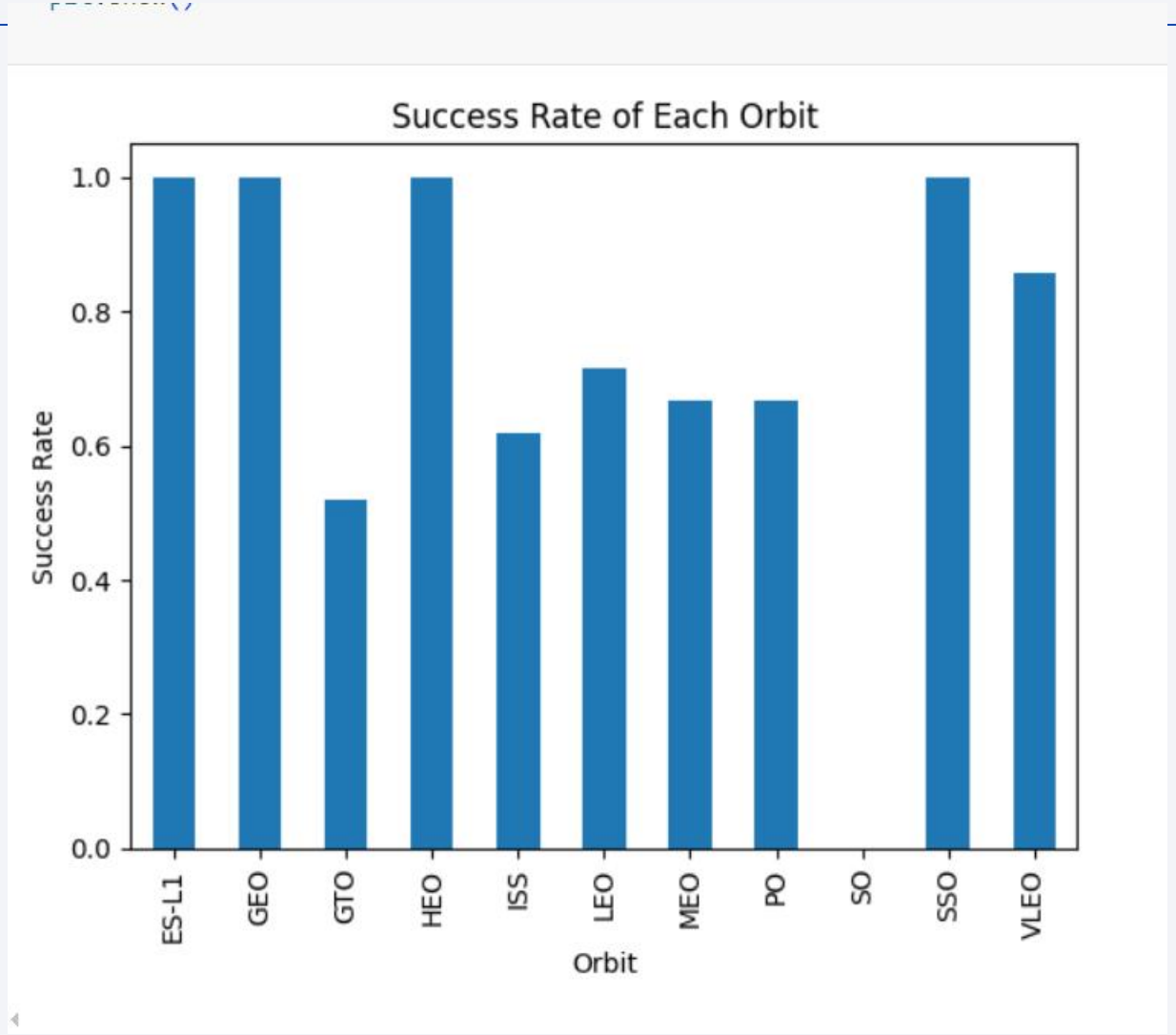
Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site
- Show the screenshot of the scatter plot with explanations



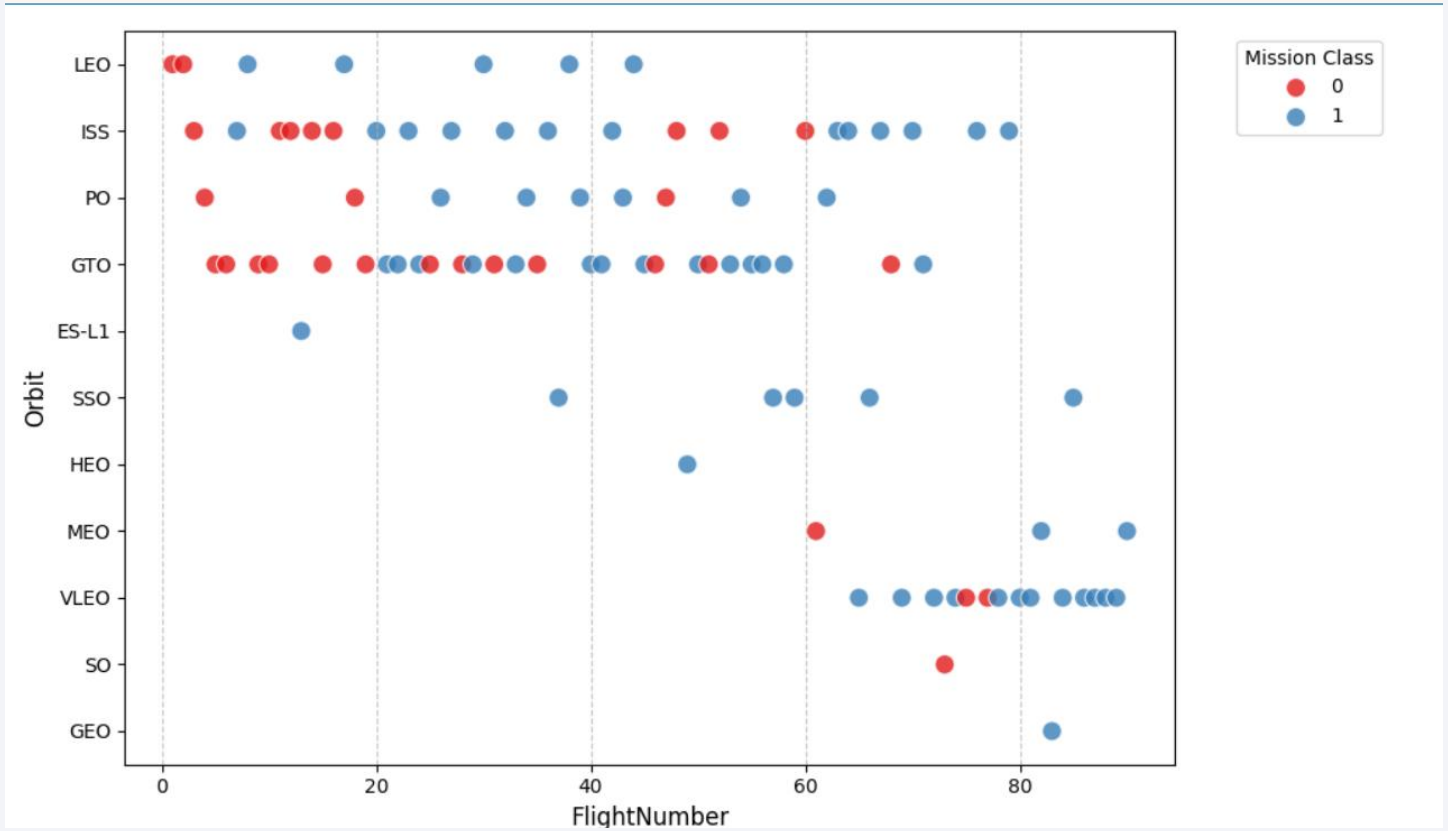
Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type
- Show the screenshot of the scatter plot with explanations



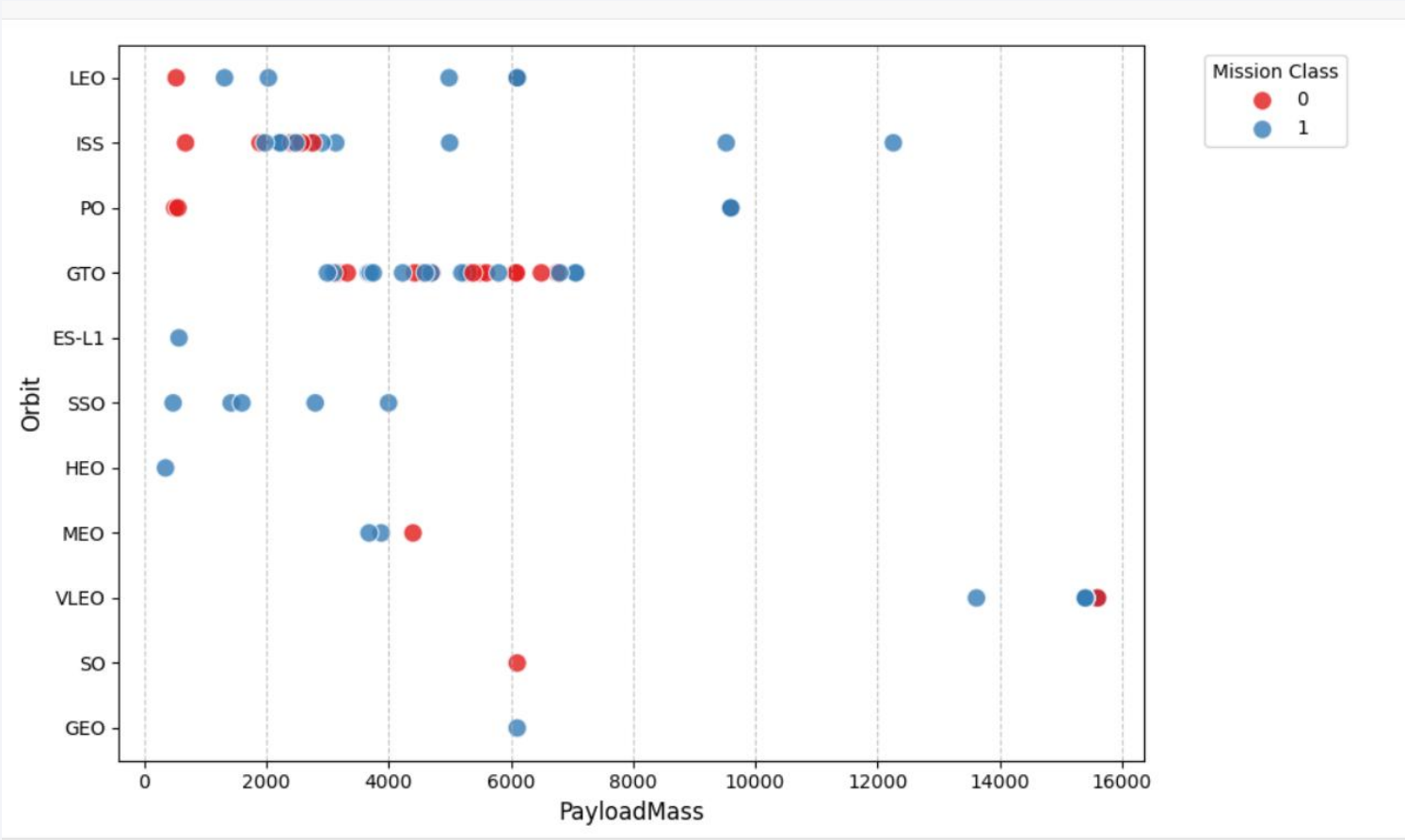
Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type
- Show the screenshot of the scatter plot with explanations



Payload vs. Orbit Type

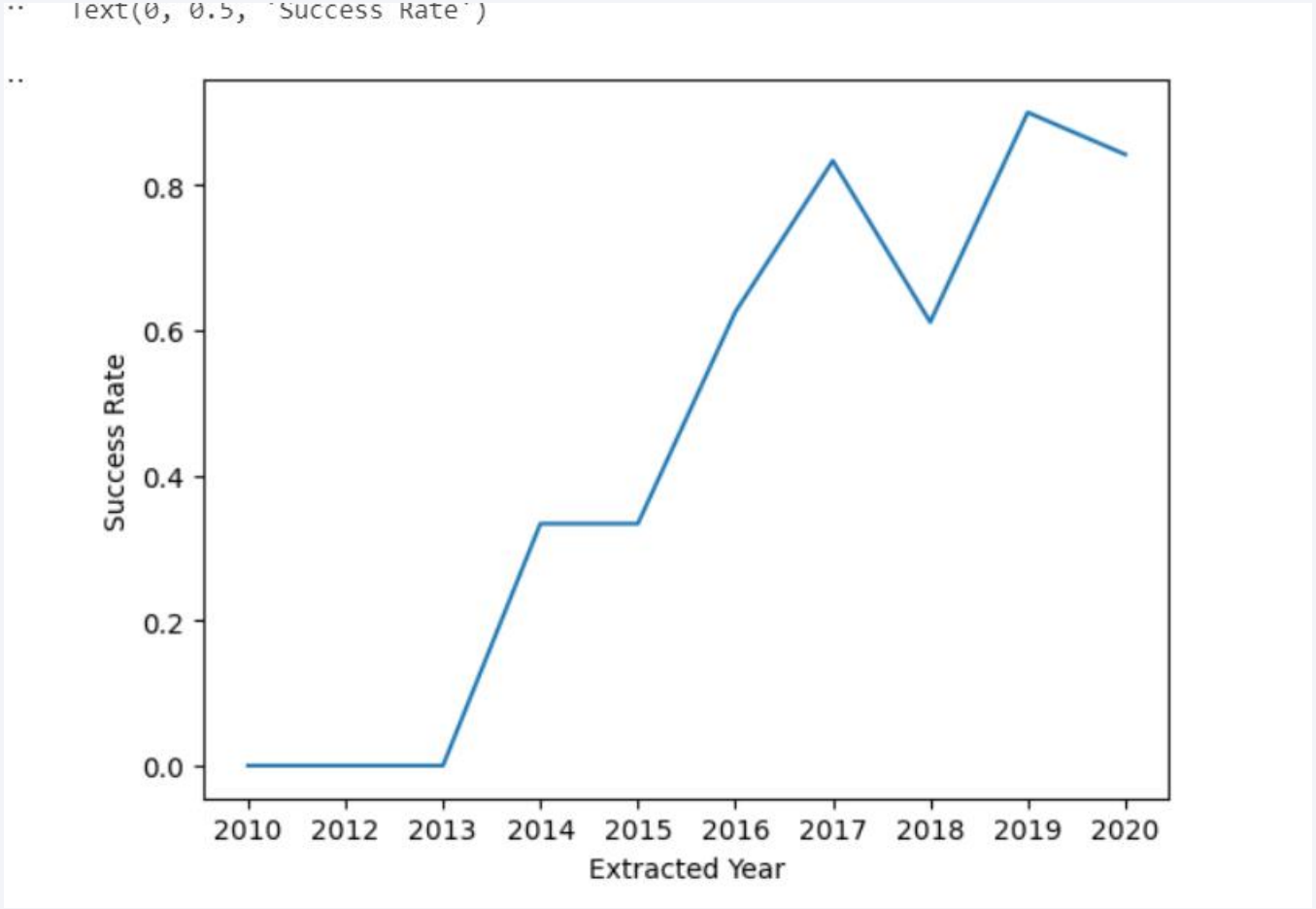
- Show a scatter point of payload vs. orbit type
- Show the screenshot of the scatter plot with explanations



Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- Show the screenshot of the scatter plot with explanations



All Launch Site Names

- Find the names of the unique launch sites
- Present your query result with a short explanation here

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

[12]

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

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Present your query result with a short explanation here

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

[16] 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

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

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

```
[20] %sql  SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)'
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

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

```
[21] %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

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here

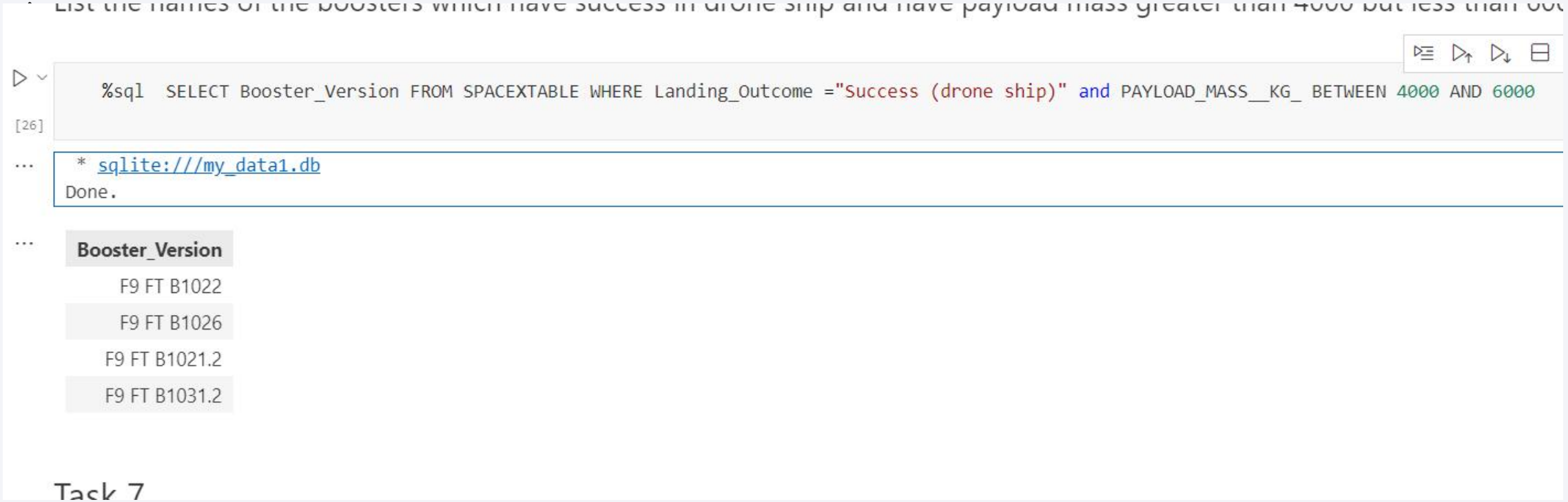
```
%sql  SELECT min(Date) FROM SPACEXTABLE WHERE Landing_Outcome LIKE "%Success%"
```

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less



Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

List the total number of successful and failure mission outcomes

```
%sql SELECT COUNT(CASE WHEN LOWER(Landing_Outcome) LIKE '%success%' THEN 1 END) AS success_count,COUNT(CASE WHEN LOWER(Landing_Outcome) LIKE'
```

[32]

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

success_count	fail_count
61	10

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
%sql SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ =( SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

[36]

```
 * 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

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
            WHEN substr(Date, 6, 2) = '02' THEN 'February'
            WHEN substr(Date, 6, 2) = '03' THEN 'March'
            WHEN substr(Date, 6, 2) = '04' THEN 'April'
            WHEN substr(Date, 6, 2) = '05' THEN 'May'
            WHEN substr(Date, 6, 2) = '06' THEN 'June'
            WHEN substr(Date, 6, 2) = '07' THEN 'July'
            WHEN substr(Date, 6, 2) = '08' THEN 'August'
            WHEN substr(Date, 6, 2) = '09' THEN 'September'
            WHEN substr(Date, 6, 2) = '10' THEN 'October'
            WHEN substr(Date, 6, 2) = '11' THEN 'November'
            WHEN substr(Date, 6, 2) = '12' THEN 'December'
        END AS MonthName,
        Landing_Outcome,
        Booster_Version,
        Launch_Site
    FROM SPACEXTABLE
    WHERE
        substr(Date, 1, 4) = '2015'
        AND Landing_Outcome LIKE '%fail%'
        AND Landing_Outcome LIKE '%drone ship%'
    ORDER BY substr(Date, 6, 2);
```

[38]

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

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

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

- 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

- Pre lanation here

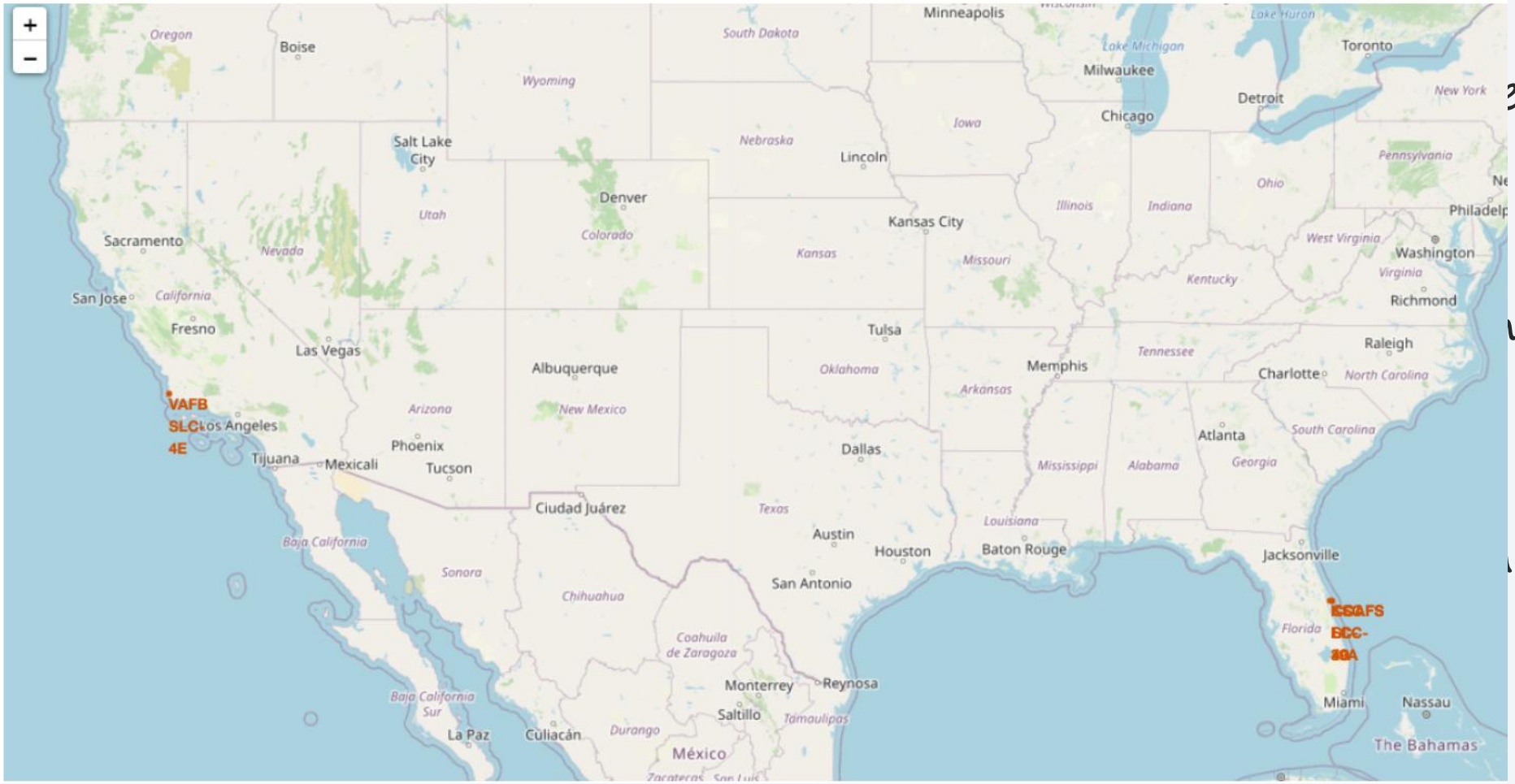
```
%%sql
SELECT
    Landing_Outcome,
    COUNT(*) AS outcome_count
FROM SPACEXTABLE
WHERE
    Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY outcome_count DESC;
```

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

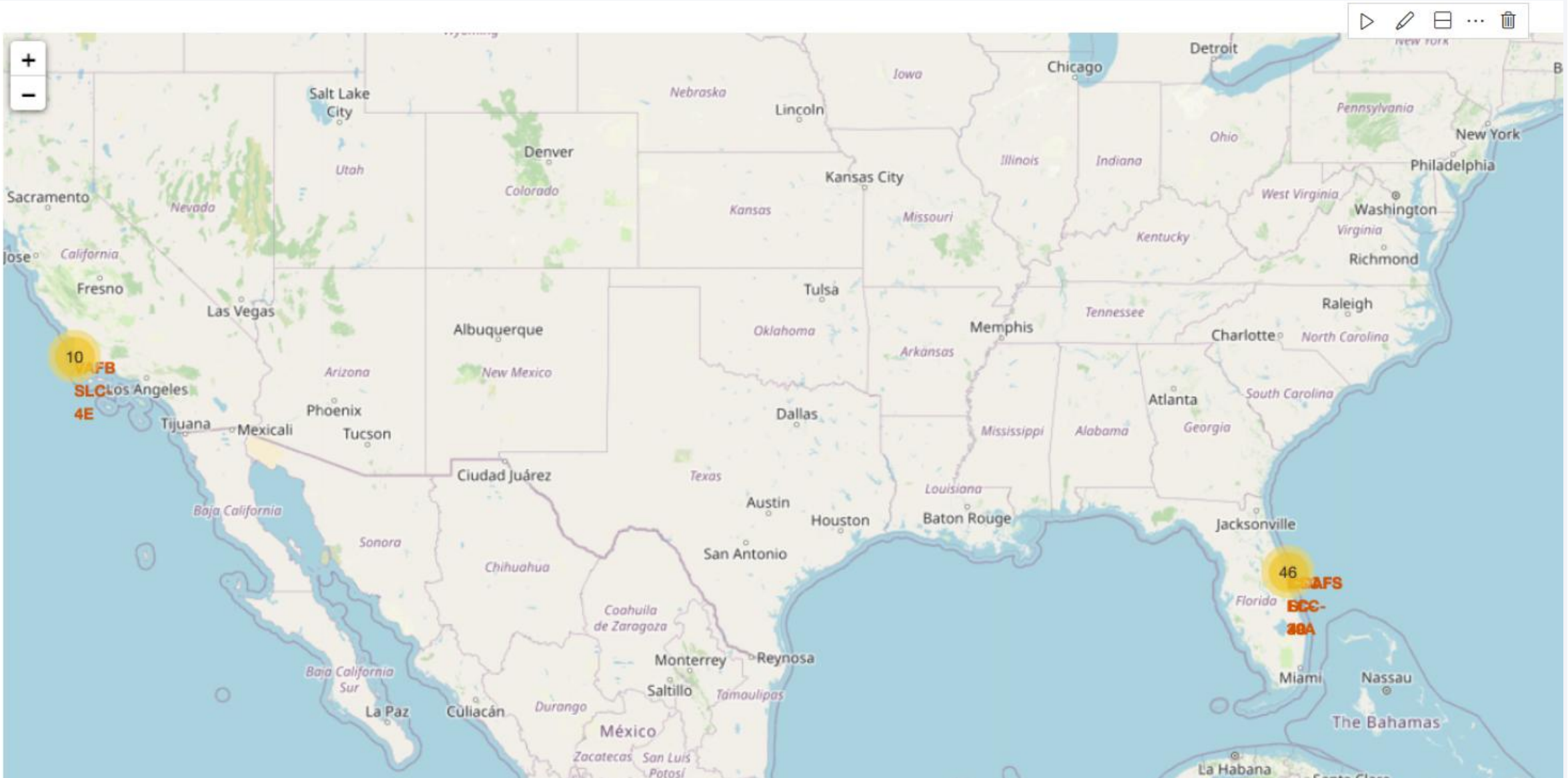
Landing_Outcome	outcome_count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1


Section 3
Launch Sites
Proximities Analysis

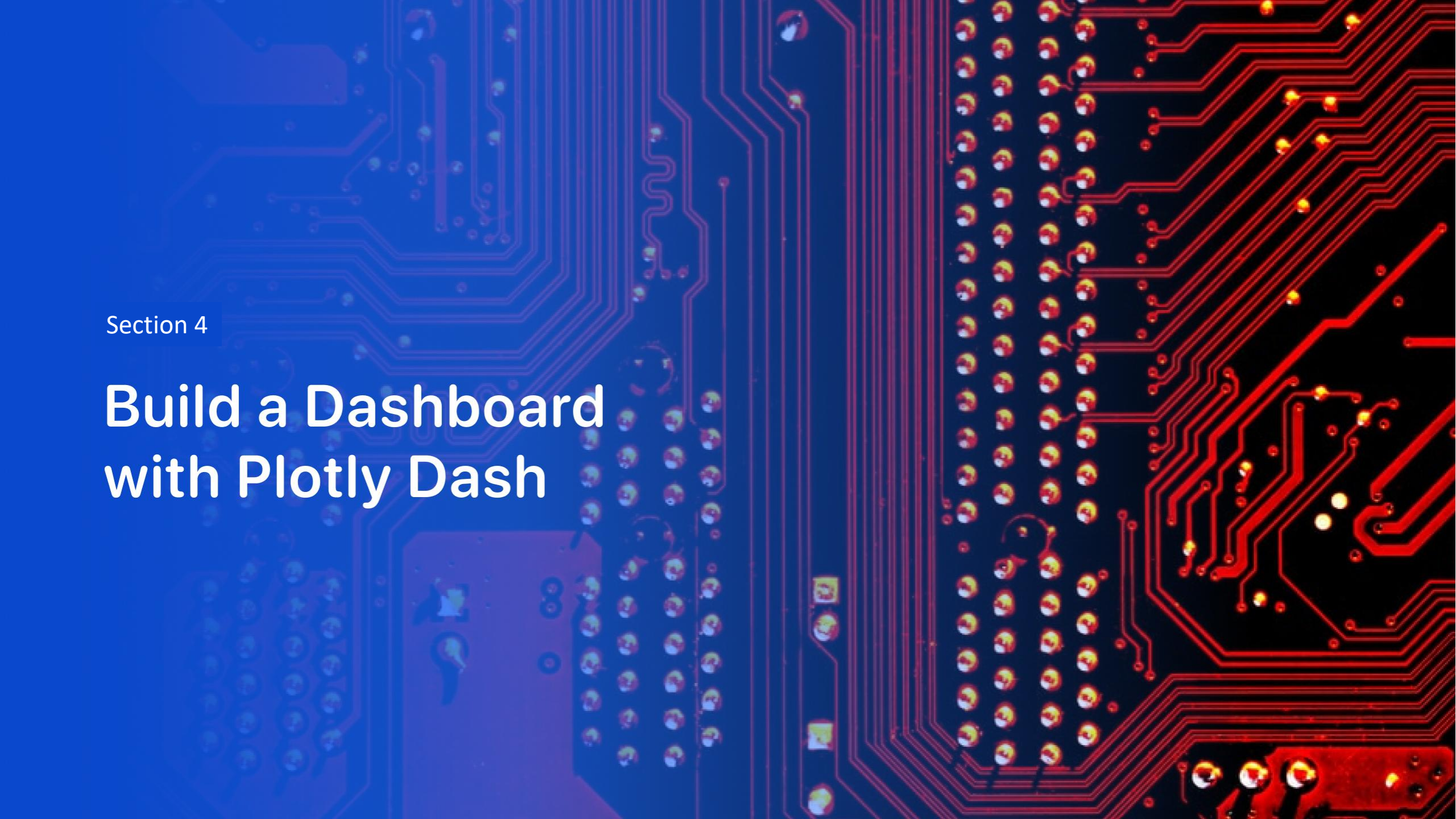
<launch sites'>



<color-labeled launch outcome>



<closest coastline>


Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Replac … opriate title
- Show … ces, in a piecha…
- Explai… creenshot

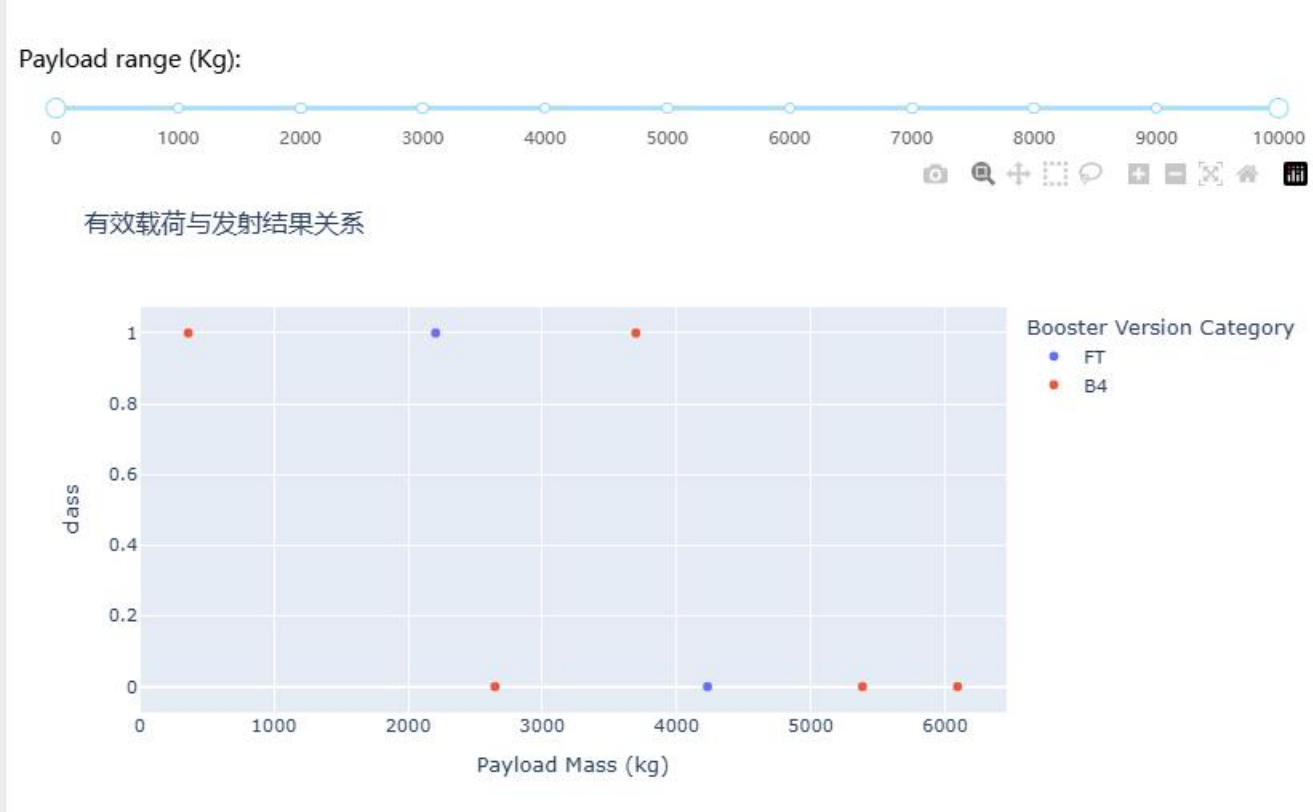
<Dashboard Screenshot 2>

. appropriate title

nch site with highest

ı the screenshot

<Dashboard Screenshot 3>



- ...n an appropriate title
- ...come scatter plot for all ... range slider
- ...s on the screenshot, such ...ave the largest success

Section 5

Predictive Analysis (Classification)

Classification Accuracy

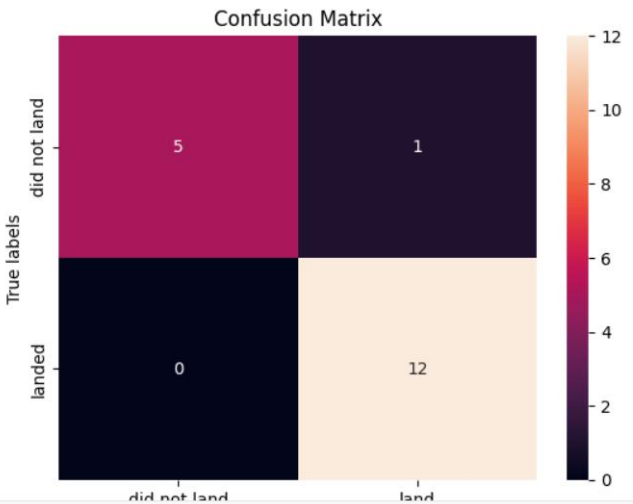
- Vis
  for
  in
- Fin
  cla

```
accuracy = logreg_cv.score(X_test,Y_test)
```
[23]

```
0.9444444444444444
```

Lets look at the confusion matrix:

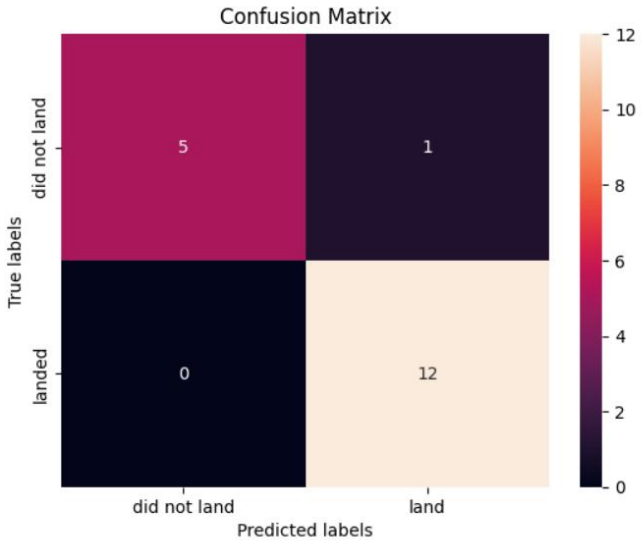
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```
[24]




```
0.9444444444444444
```
[30]

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```
[31]

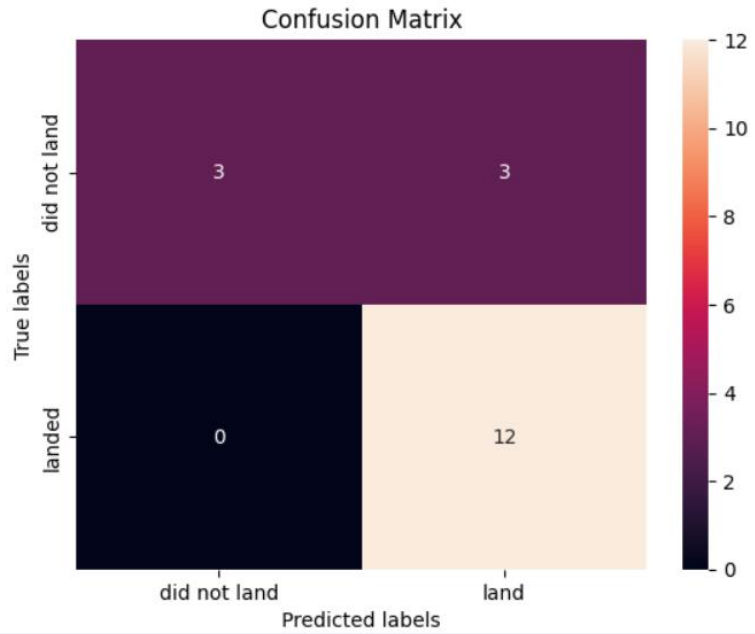


Confusion Matrix

```
0.8333333333333334
```

We can plot the confusion matrix

```
[37]
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



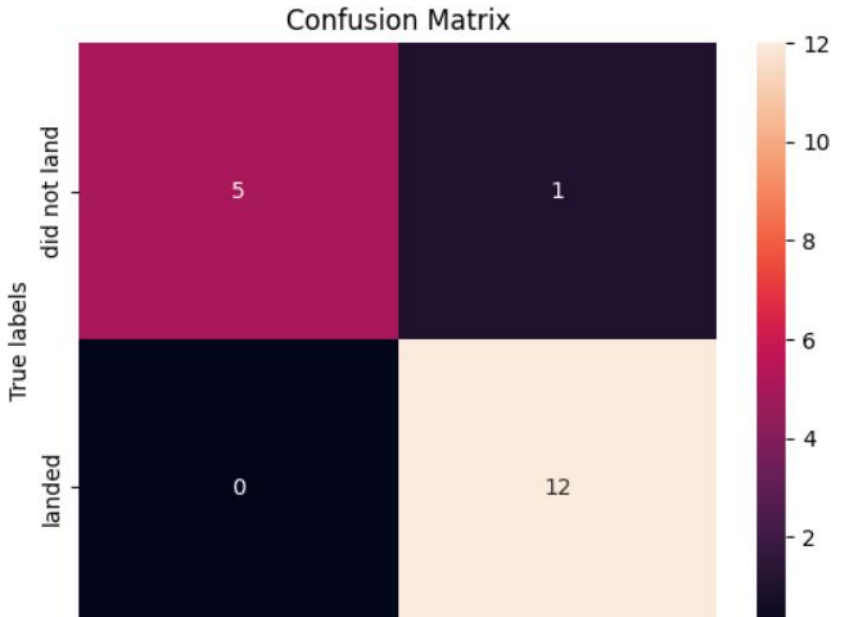
Code | Markdown | Run All | Clear All Outputs | Outline

```
accuracy_knn
[43]
```

```
0.9444444444444444
```

We can plot the confusion matrix

```
[44]
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- Point 1
- Point 2
- Point 3
- Point 4
- …

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!