IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Kevin Yang>
<Dec 27th 2023>

Outline

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

Executive Summary

- Data was collected from Space X API followed by data wrangling. The was subsequently explored through SQL and visualization. The processed data was used for folium-based visualization on map and also for creating an interactive dashboard. To further help predict successful rate, the features, which includes flight number, payload mass etc were used to train machine learning model for future decision making. The accuracy of the model was further validated by confusion matrix and accuracy scores.
- Summary of all results

Introduction

Background

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch.

Objective

The goal of this project is to create a machine learning pipeline to predict if the first stage will land given the data from the preceding labs.

Section 1

Methodology

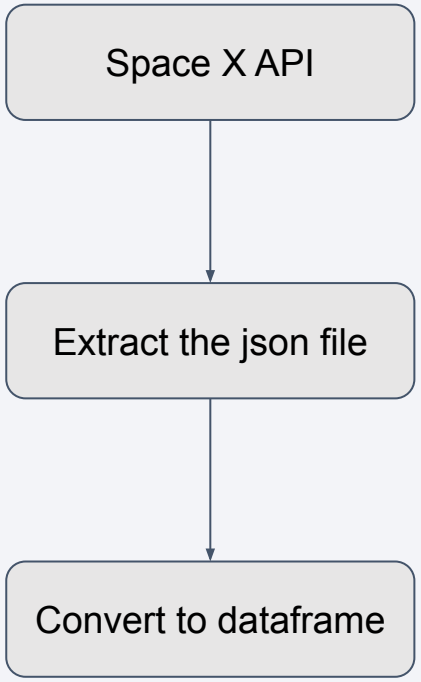
Methodology

Executive Summary

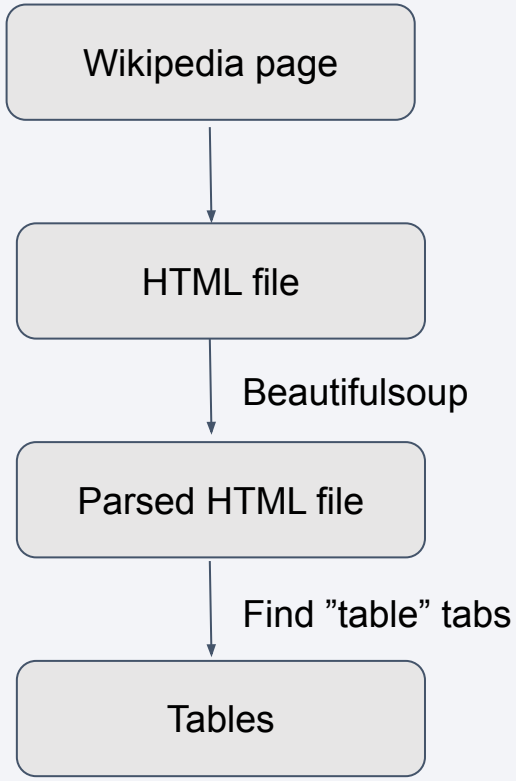
- Data collection methodology:
  - Data was collected through requesting to the SpaceX API or scraping records from Wikipedia
- Perform data wrangling
  - Data was subsequently processed: replace null with average, convert integer to floats etc.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Various model were tested and their best parameters were identified through GridSearchCV function on sklearn.

Data Collection

API



Web scraping



Data Collection – SpaceX API

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain
.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

Executed at 2023.12.22 17:16:39 in 11ms

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

```
response.status_code
```

Executed at 2023.12.22 17:16:45 in 13ms

```
200
```

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

```
# Use json_normalize method to convert the json result into a dataframe
df = pd.read_json(static_json_url, orient='records')

```

Executed at 2023.12.22 17:17:05 in 924ms

[Full Python codes on Github]

Data Collection - Scraping

- Provide URL

```
static_url = "https://en.wikipedia.org/w/index
 .php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
Executed at 2023.12.26 11:27:46 in 4ms
```

- Use BeautifulSoup to scrape the Wikipedia article

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, "html.parser")
Executed at 2023.12.26 11:31:08 in 569ms
```

Full Python code on Github

Data Collection - Scraping (continued)

- Scrape the table from the Wikipedia article

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all("table")
```

Executed at 2023.12.26 11:34:57 in 21ms

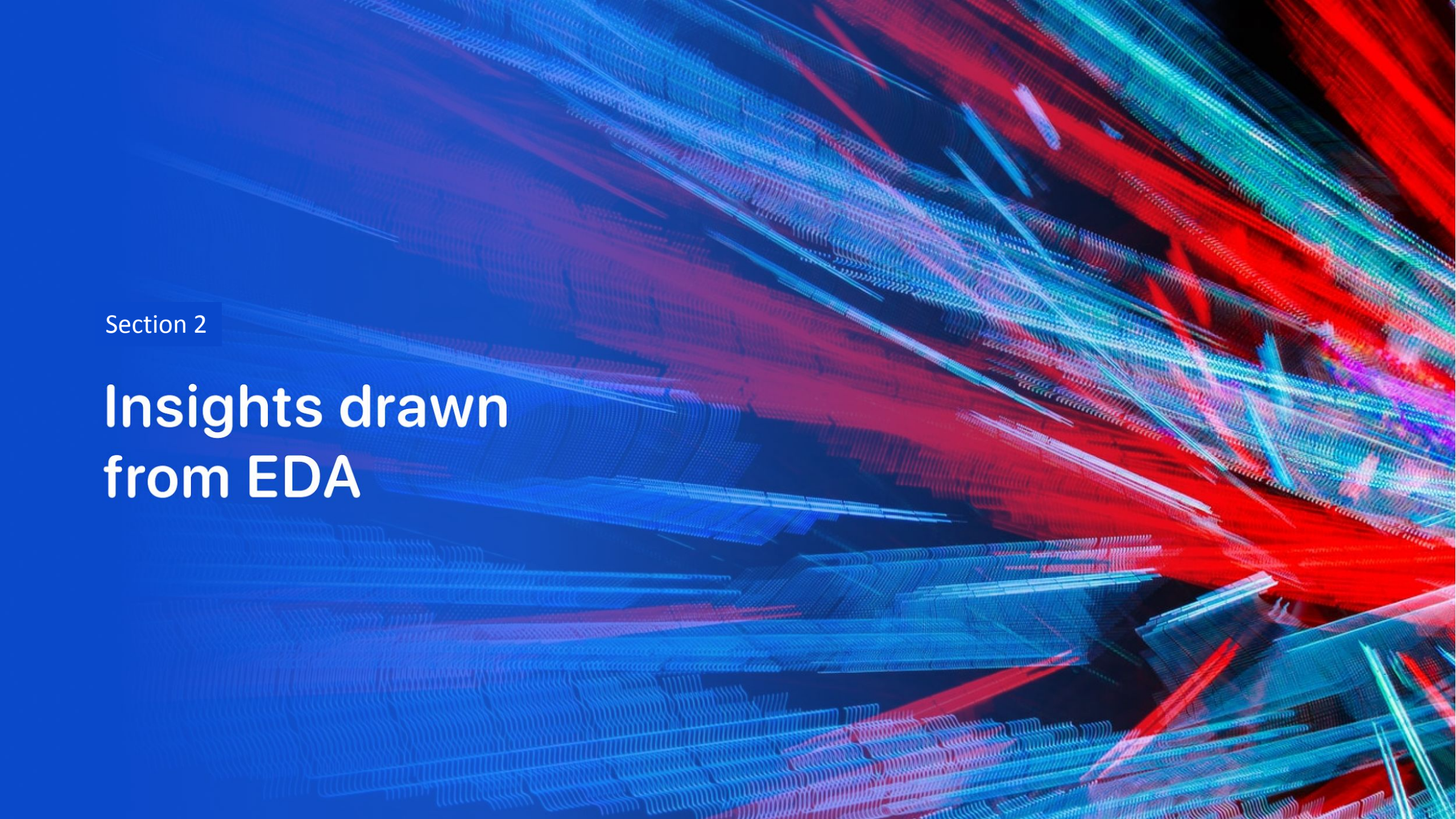
- Trim the collected tables to preserve only those that contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

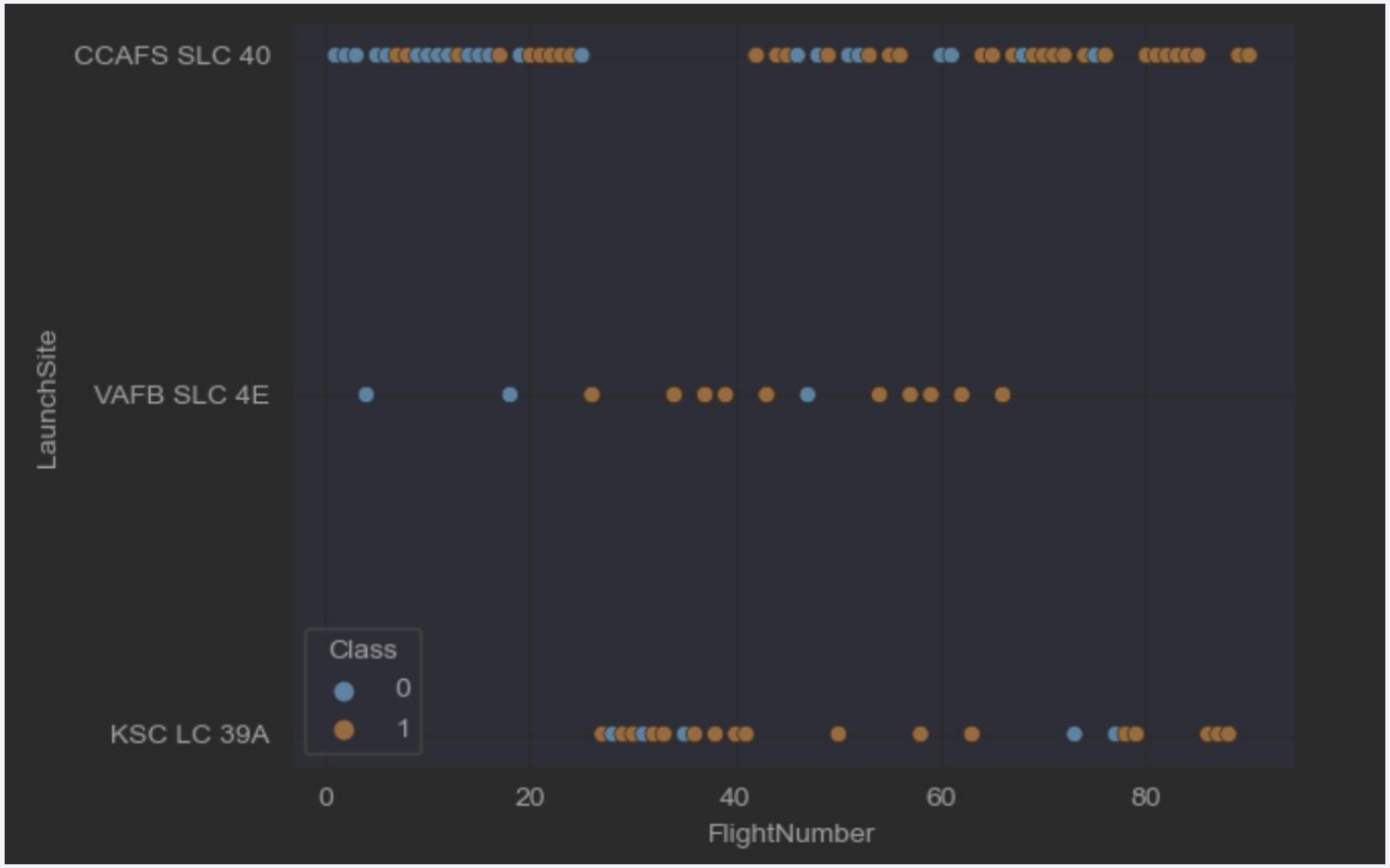
Executed at 2023.12.26 11:35:00 in 8ms

Section 2

Insights drawn from EDA



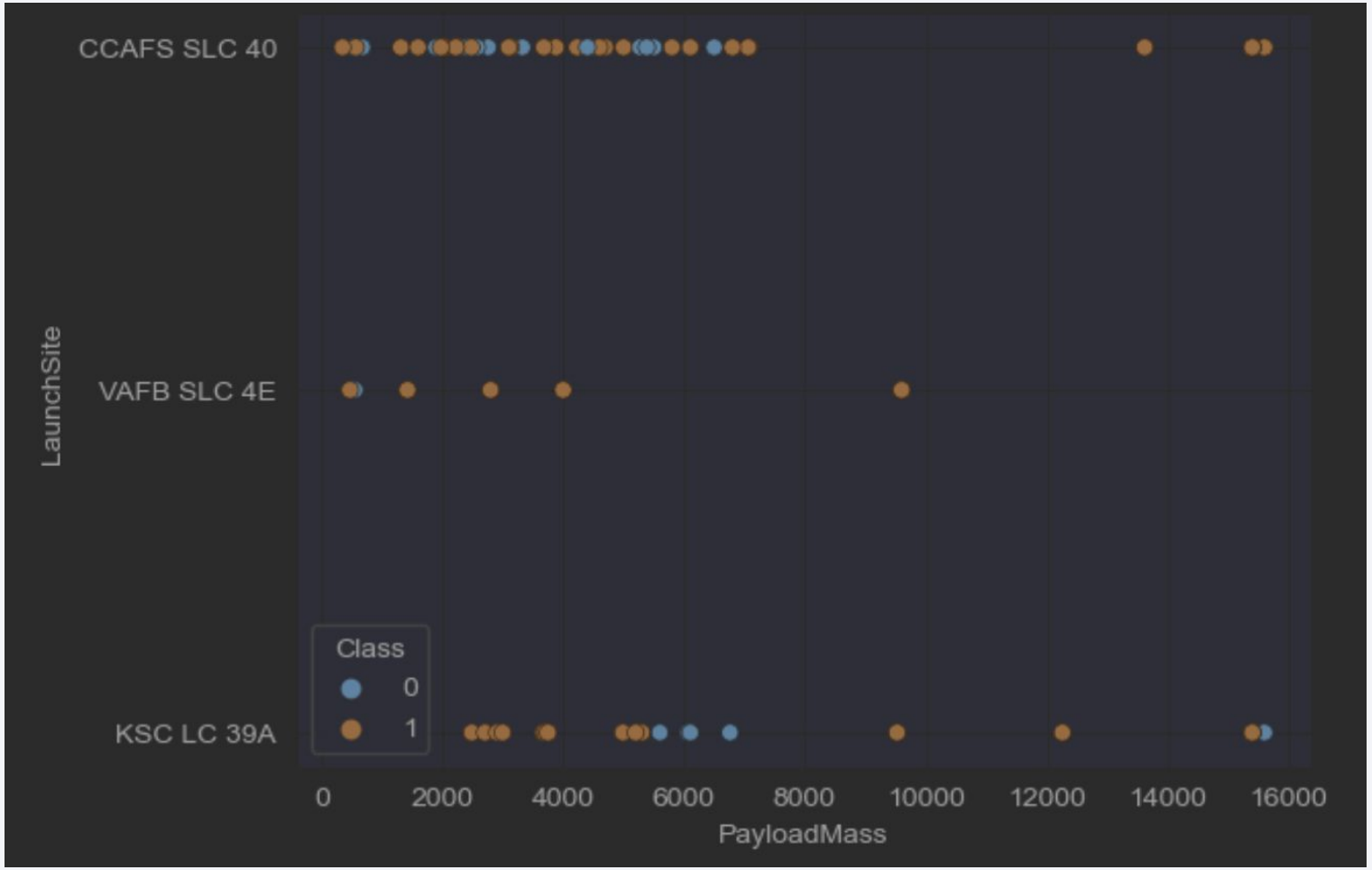
Flight Number vs. Launch Site



Both *CCAFS SLC 40* and *VAFB SLC 4E* launch sites have higher success rates upon increasing the flight number

Full Python code on Github

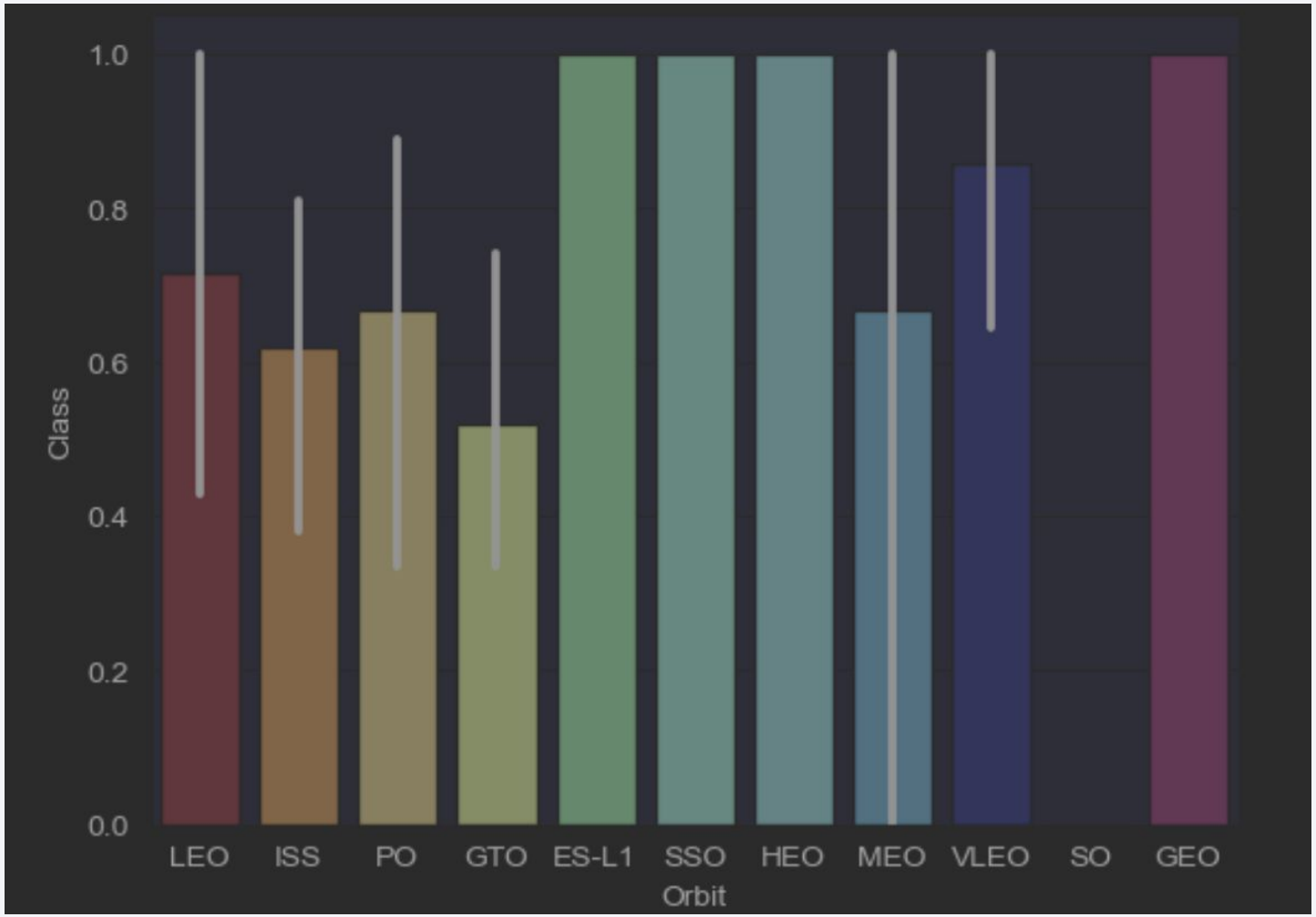
Payload vs. Launch Site



VAFB SLC 4E launch site has no rockets launched for heavy payload mass (> 10,000 Kg)

Full Python code on Github

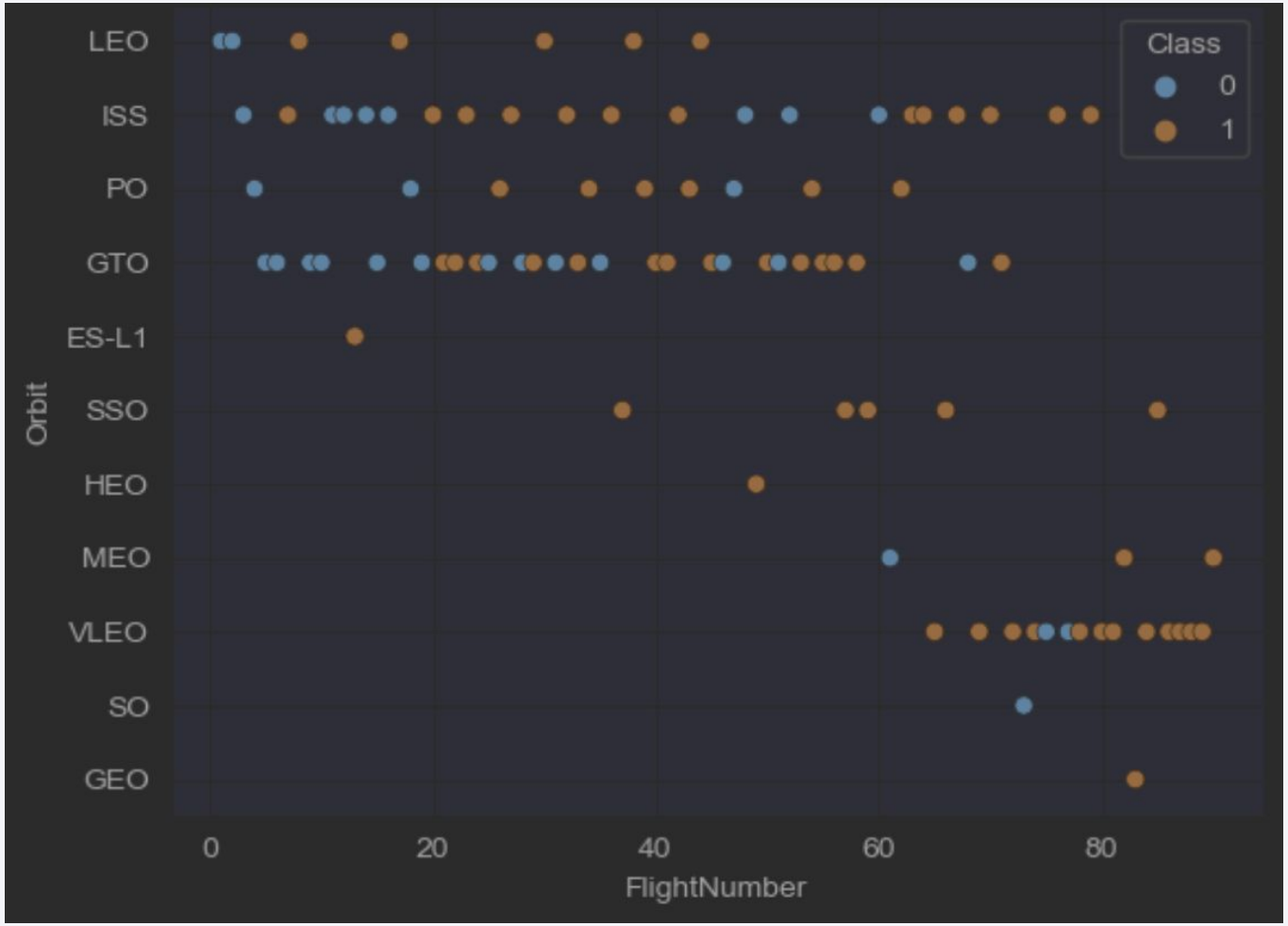
Success Rate vs. Orbit Type



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

Full Python code on Github

Flight Number vs. Orbit Type

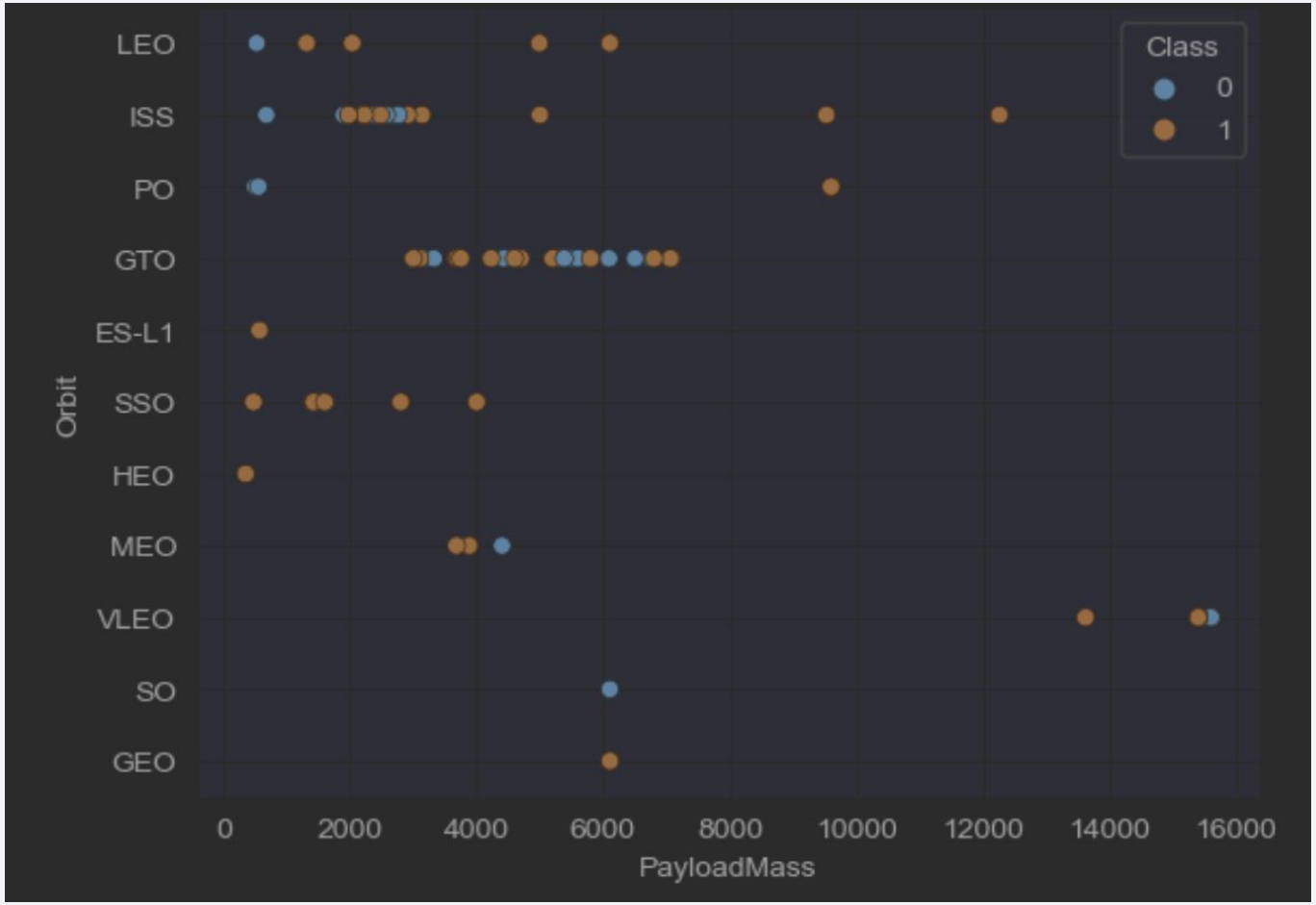


LEO orbit success rate is related to the number of flight.

No relationship between flight number and success rate in GTO orbit.

[Full Python code on Github]

Payload vs. Orbit Type

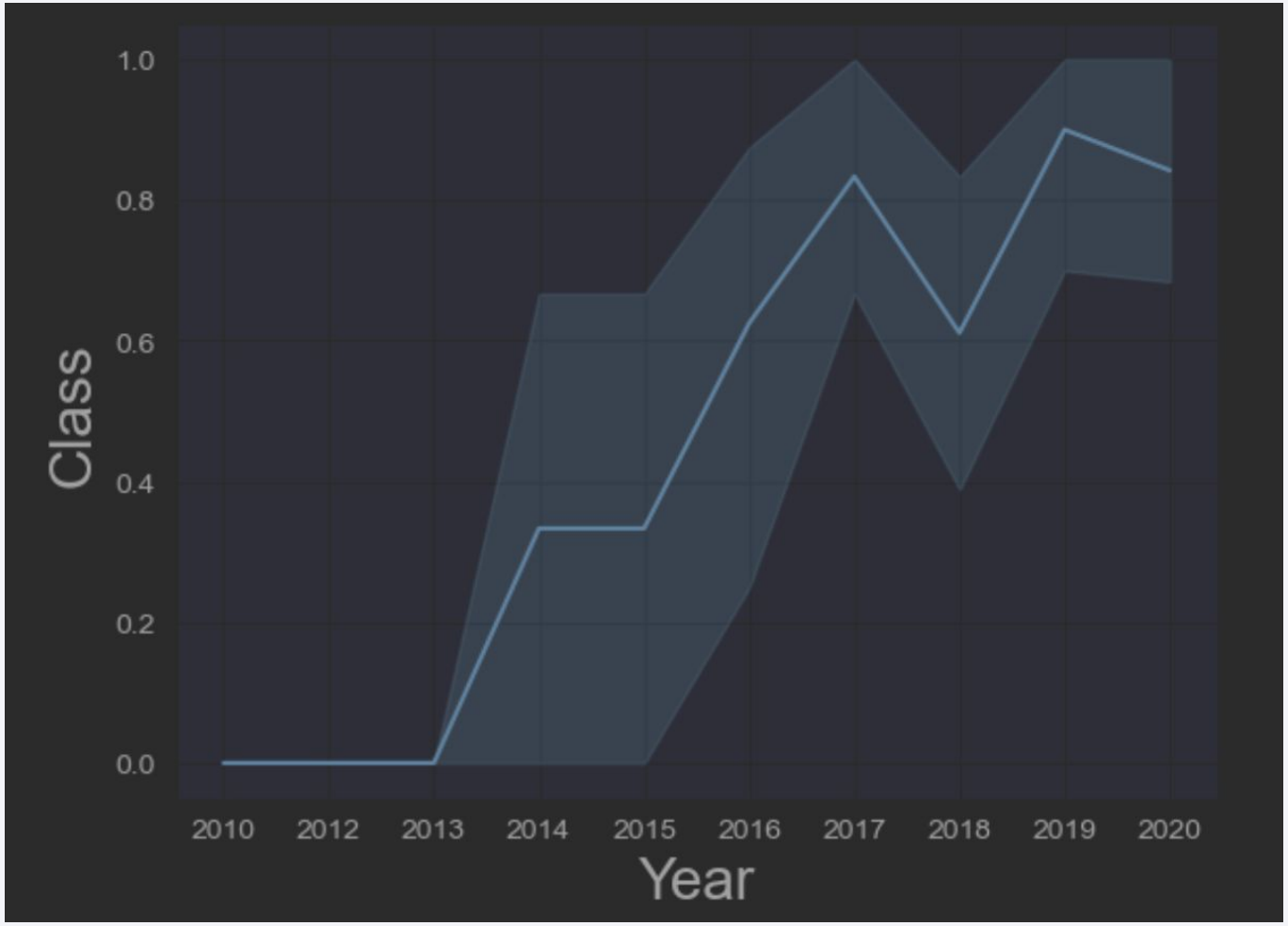


More successful landing with higher payload mass for orbit Polar (PO), LEO and ISS orbit.

No relationship between payload mass and success rate for orbit GTO.

Full Python code on Github

Launch Success Yearly Trend



Success rate sharply increased since 2013 and reached plateau in 2017.

Full Python code on Github

All Launch Site Names

- Explore the unique launch sites in the space mission

```
query = "select distinct Launch_Site from SPACEXTBL"
cur.execute(query)
launch_sites = cur.fetchall()
for site in launch_sites:
    print(site)
```

Executed at 2023.12.26 15:31:21 in 4ms

```
('CCAFS LC-40',)
('VAFB SLC-4E',)
('KSC LC-39A',)
('CCAFS SLC-40',)
```

Four distincted launch sites

[Full Python Code on Github]

Launch Site Names Begin with 'CCA'

- Explore the launch sites begin with the string “CCA” and display the top 5

```
query = "select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5"
cur.execute(query)
CCA = cur.fetchall()
for c in CCA:
    print(c)
```

Executed at 2023.12.26 15:33:56 in 5ms

```
('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')
```

Full Python Code on Github

Total Payload Mass

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

```
query = "select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)'"
cur.execute(query)
total_payload = cur.fetchone()
print(total_payload)
```

Executed at 2023.12.26 15:43:16 in 10ms

```
(45596,)
```

Total payload mass is 45,596 Kg

Full Python Code on Github

Average Payload Mass by F9 v1.1

- Average payload mass carried by booster version F9 v1.1

```
query = "select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version = 'F9 v1.1'"
cur.execute(query)
avg_mass = cur.fetchone()
print(avg_mass)
Executed at 2023.12.26 15:44:52 in 6ms

(2928.4,)
```

Averaged payload mass is 2,928 Kg

Full Python Code on Github

First Successful Ground Landing Date

- When was the first successful landing outcome

```
query = "select min(Date) from SPACEXTBL where Landing_Outcome = 'Success'"
cur.execute(query)

suc = cur.fetchone()
print(suc)
```

Executed at 2023.12.26 15:47:47 in 4ms

```
('2018-07-22',)
```

First successful landing is on 2018-07-22

Full Python Code on Github

Successful Drone Ship Landing with Payload between 4000 and 6000

- Boosters which have success in drone ship and have payload mass between 4000-6000 kg

```
query = "select Booster_Version from SPACEXTBL where Mission_Outcome = 'Success' and
 Payload_MASS__KG_ >4000 and Payload_MASS__KG_ <6000 "
cur.execute(query)
boosters = cur.fetchall()
for b in boosters:
    print(b)
```

Executed at 2023.12.26 15:51:34 in 4ms

```
('F9 v1.1',)
('F9 v1.1 B1011',)
('F9 v1.1 B1014',)
('F9 v1.1 B1016',)
('F9 FT B1020',)
```

Full Python Code on Github

Total Number of Successful and Failure Mission Outcomes

- Total number of successful and failure mission outcomes

```
query_success = "select count(*) from SPACEXTBL where Mission_Outcome = 'Success' "
query_failure = "select count(*) from SPACEXTBL where Mission_Outcome like '%Failure%' "
dic_mission = {'Success': query_success,
               'Failure': query_failure}
def mission_outcome(outcome, query):
    cur.execute(query)
    results = cur.fetchall()
    print(str(outcome) + ": " + str(results[0]))
for key, values in dic_mission.items():
    mission_outcome(key, values)
```

Executed at 2023.12.26 16:05:19 in 5ms

```
Success: (98,)
Failure: (1,)
```

Full Python Code on Github

Boosters Carried Maximum Payload

- Boosters that have carried the maximum payload mass

```
query = "select Booster_Version from SPACEXTBL where Payload_MASS__KG_ = (select max
 (Payload_MASS__KG_) from SPACEXTBL)"
cur.execute(query)
boosters_max_payload = cur.fetchall()
for b in boosters_max_payload:
    print(b)
```

Executed at 2023.12.26 16:09:09 in 4ms

```
('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 ',)
```

[Full Python Code on Github]

2015 Launch Records

- Records of launches for the months in 2015

```
query = """SELECT SUBSTR(Date, 6,2) as Month, Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTBL
WHERE Landing_Outcome like '%Failure%'  AND SUBSTR(Date, 0,5) = '2015'  """

cur.execute(query)
records = cur.fetchall()
df = pd.DataFrame(records, columns=['Month',
                                    'Landing_Outcome',
                                    'Booster_Version',
                                    'Launch_Site'])
print(df)
```

Executed at 2023.12.26 16:24:58 in 4ms

	Month	Landing_Outcome	Booster_Version	Launch_Site
0	01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
1	04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Full Python Code on Github

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

- Landing outcomes between 2010-06-04 to 2017-03-20

```
query = """
SELECT substr(Date, 0,5), substr(Date,6,2),  Landing_Outcome, COUNT(*)
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY COUNT(*) DESC
"""
cur.execute(query)
ranks = cur.fetchall()
df = pd.DataFrame(ranks, columns=['Year',
                                  'Month',
                                  'Landing_Outcome',
                                  'Counts'])
print(df)
```

Executed at 2023.12.26 16:31:02 in 4ms

	Year	Month	Landing_Outcome	Counts
0	2012	05	No attempt	10
1	2016	04	Success (drone ship)	5
2	2015	01	Failure (drone ship)	5
3	2015	12	Success (ground pad)	3
4	2014	04	Controlled (ocean)	3
5	2013	09	Uncontrolled (ocean)	2
6	2010	06	Failure (parachute)	2
7	2015	06	Precluded (drone ship)	1

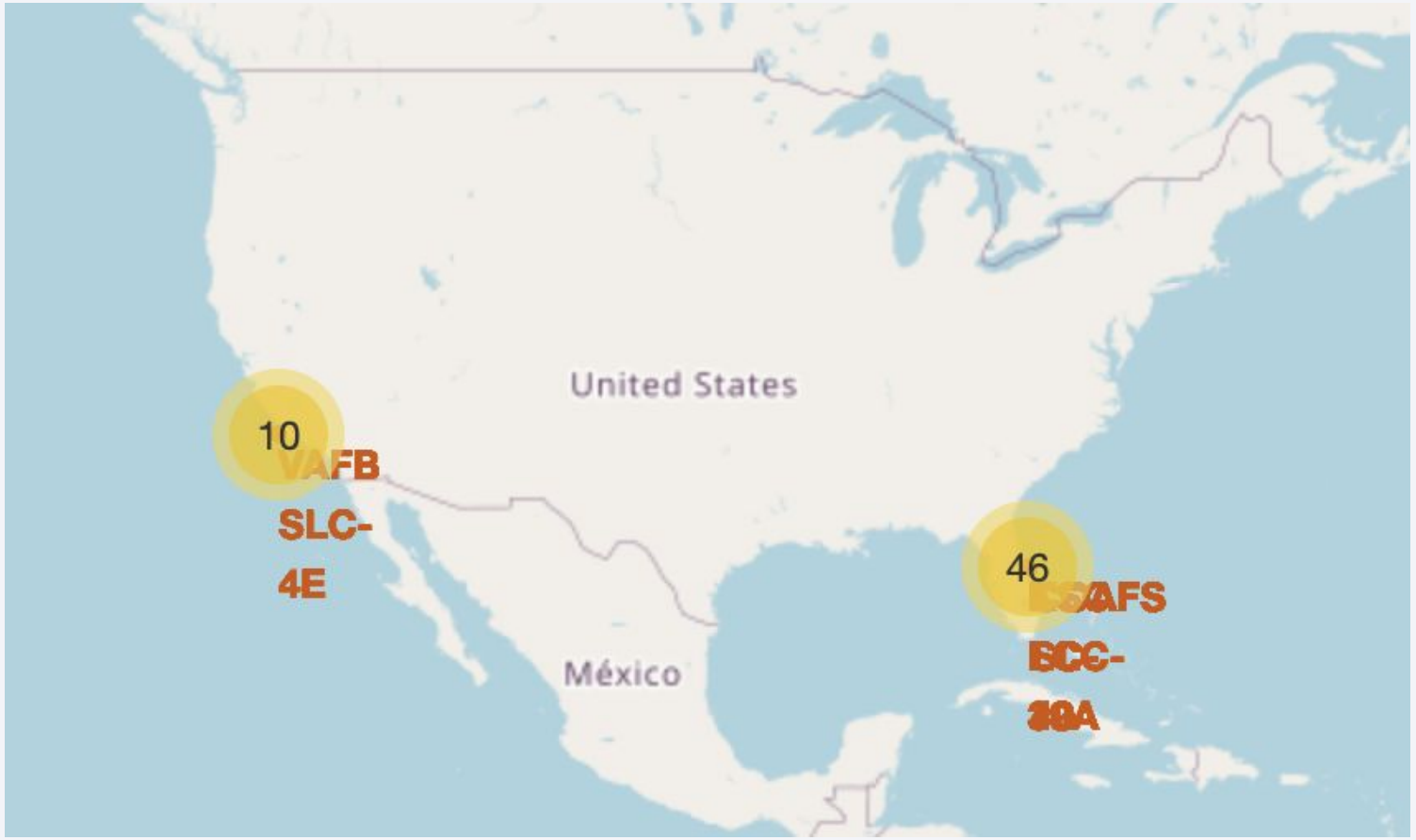

Section 3
Launch Sites
Proximities Analysis

Launch Sites in the US

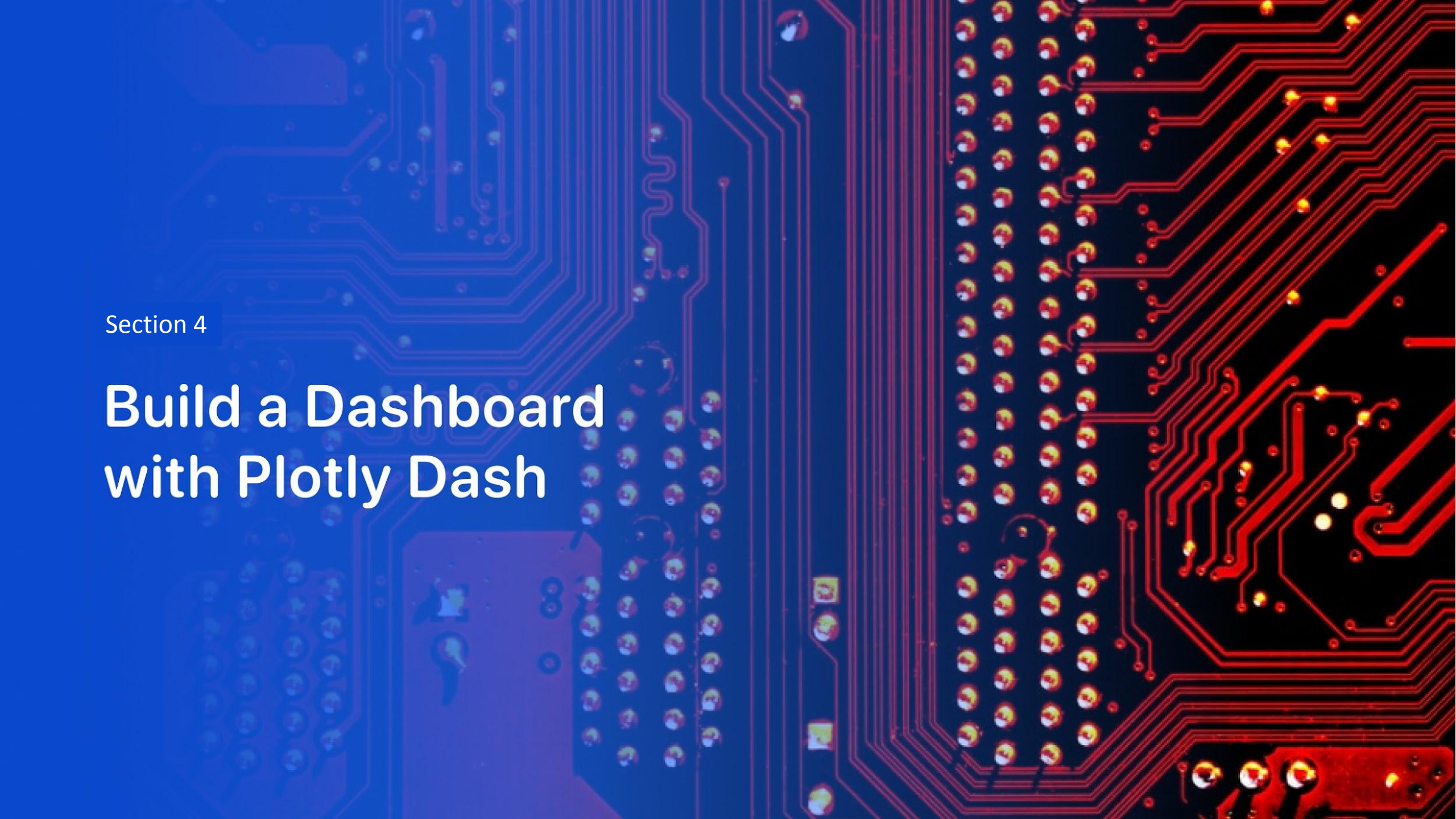
Total payload mass is 45,596 Kg

Full Python code on Github

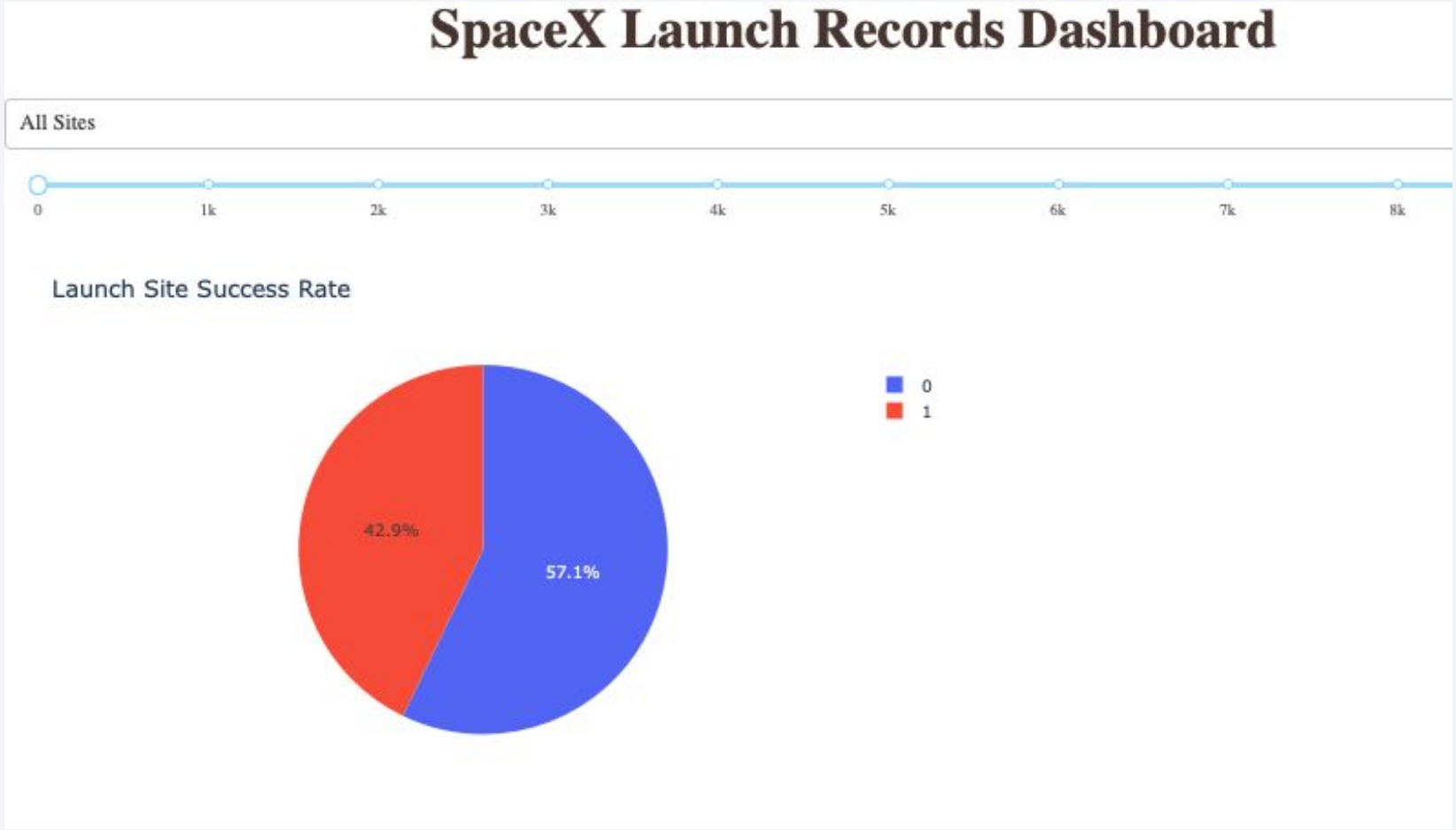
Successful/failed launches for each site



Full Python code on Github


Section 4
Build a Dashboard with Plotly Dash

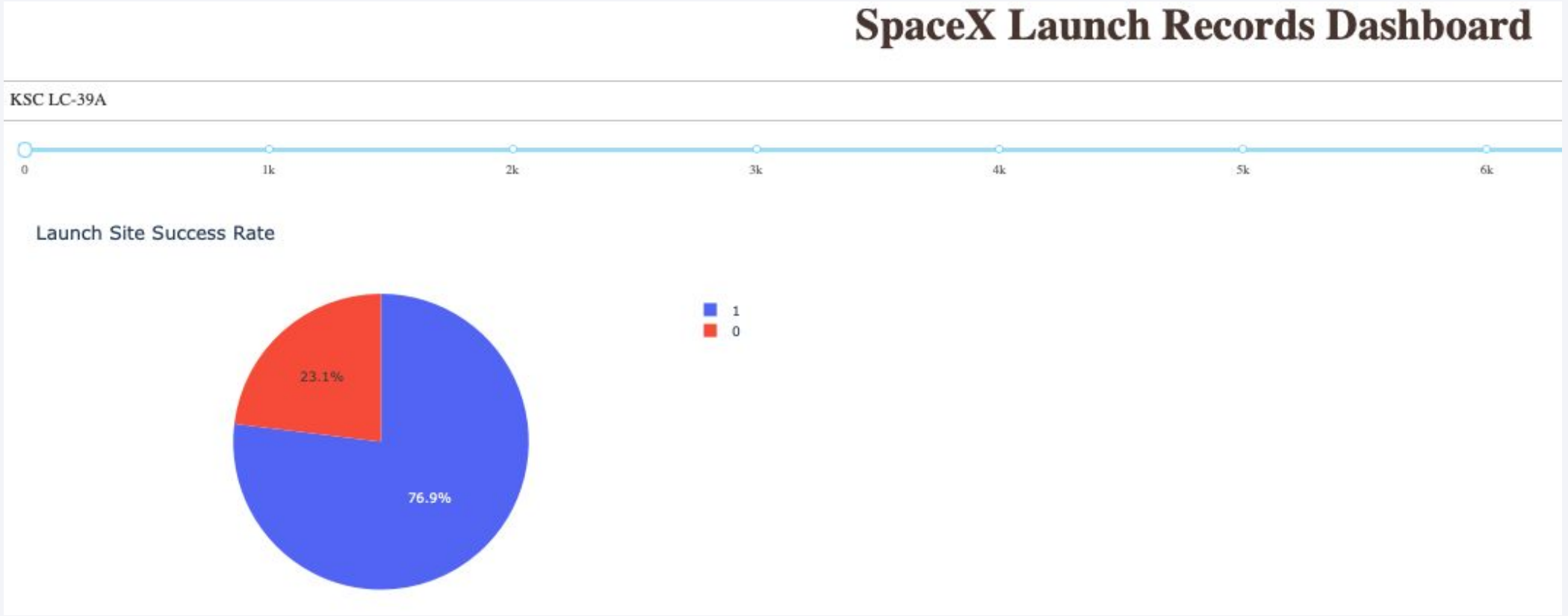
<Dashboard Screenshot 1>



Approximately 43% successful rate for all sites

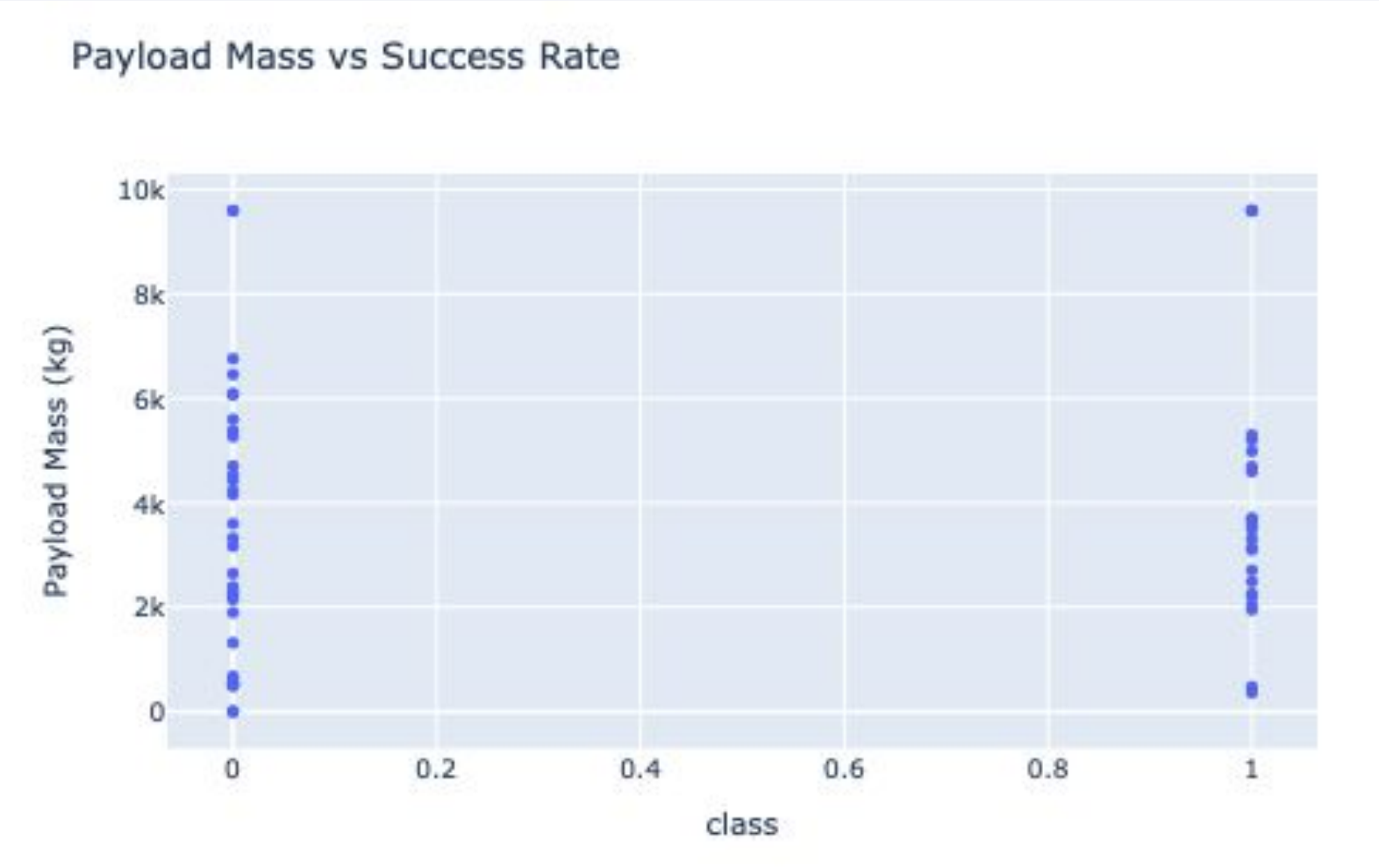
[Full Python code on Github]

KSCLC-39A site has the highest successful rate



Full Python code on Github

Payload mass vs successful rate for all sites



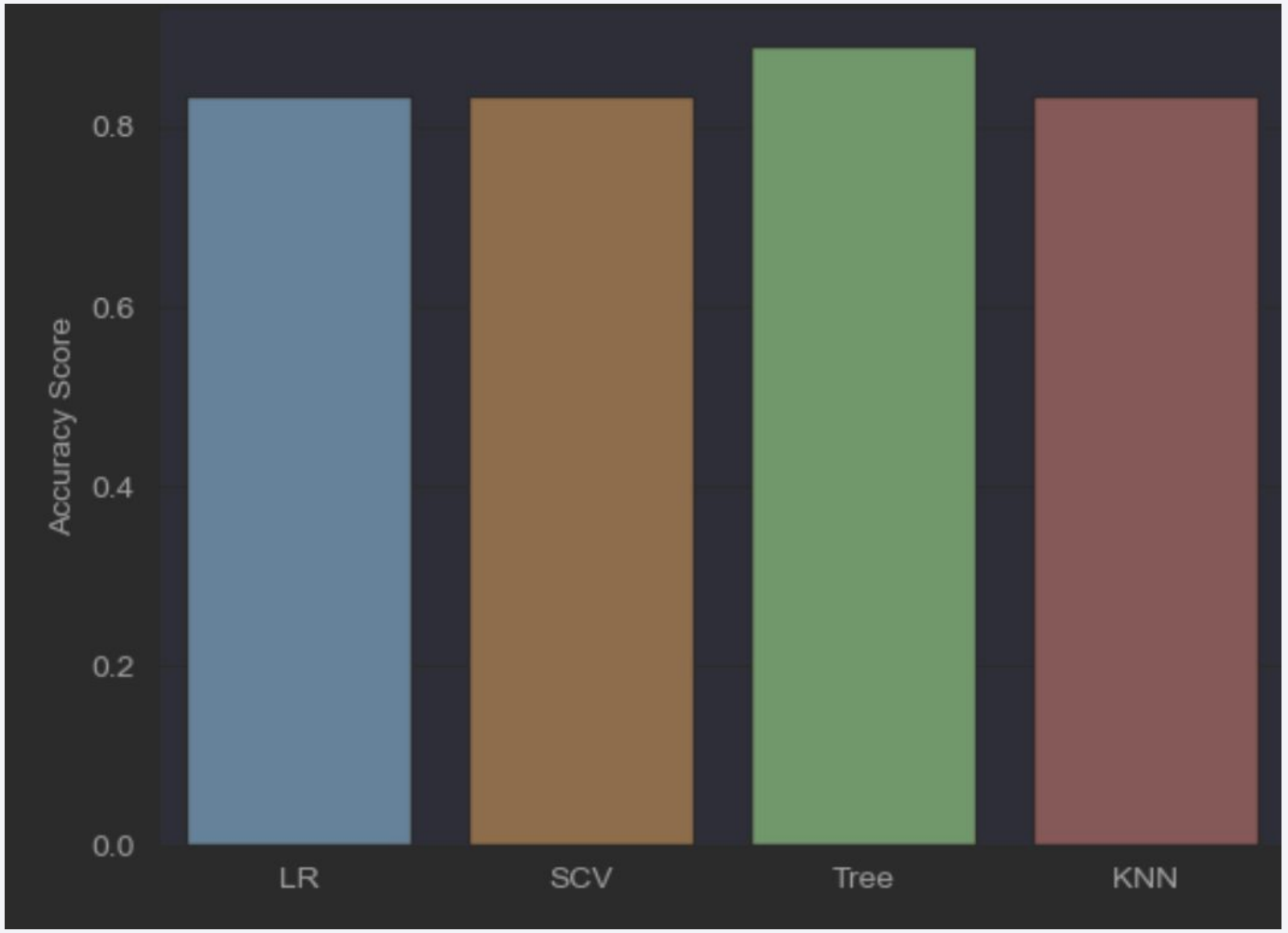
Successful launch are concentrated within 2000-6000 Kg payload mass

Full Python code on Github

Section 5

Predictive Analysis (Classification)

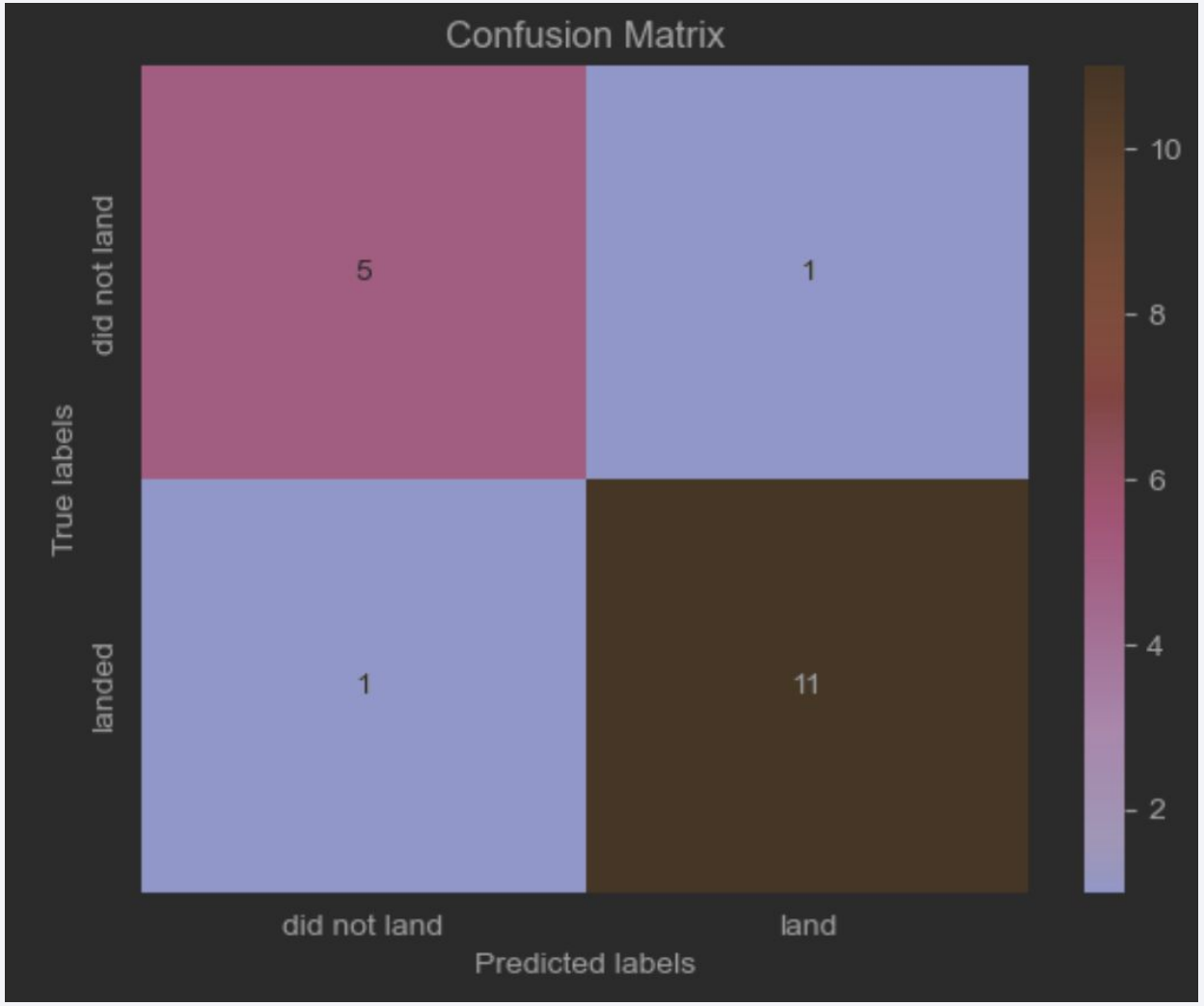
Classification Accuracy



Decision tree has the highest accuracy score.

Full Python code on Github

Confusion Matrix



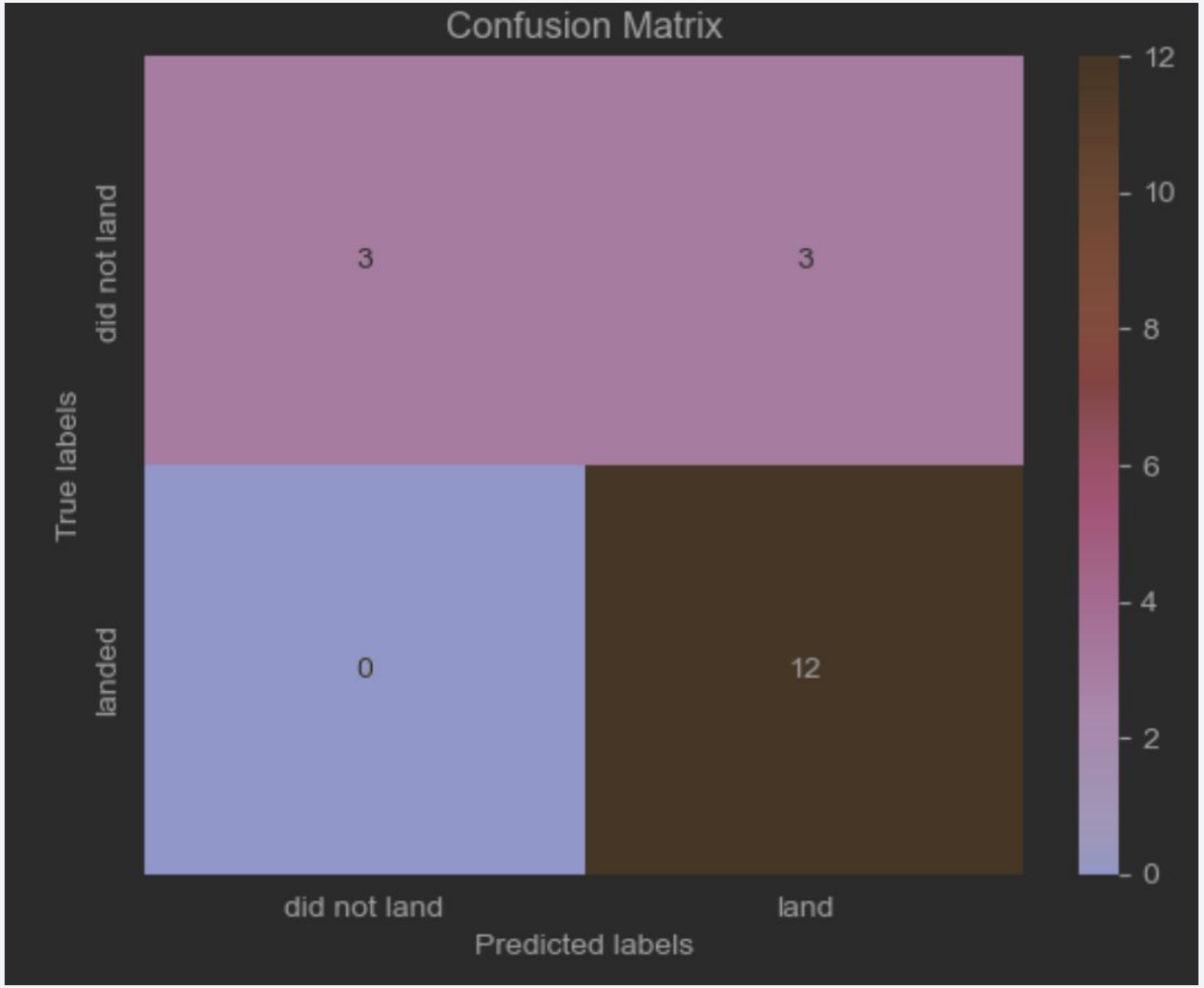
Most of the class are correctly predicted, either the landed or the not landed.

Full Python code on Github

Conclusions

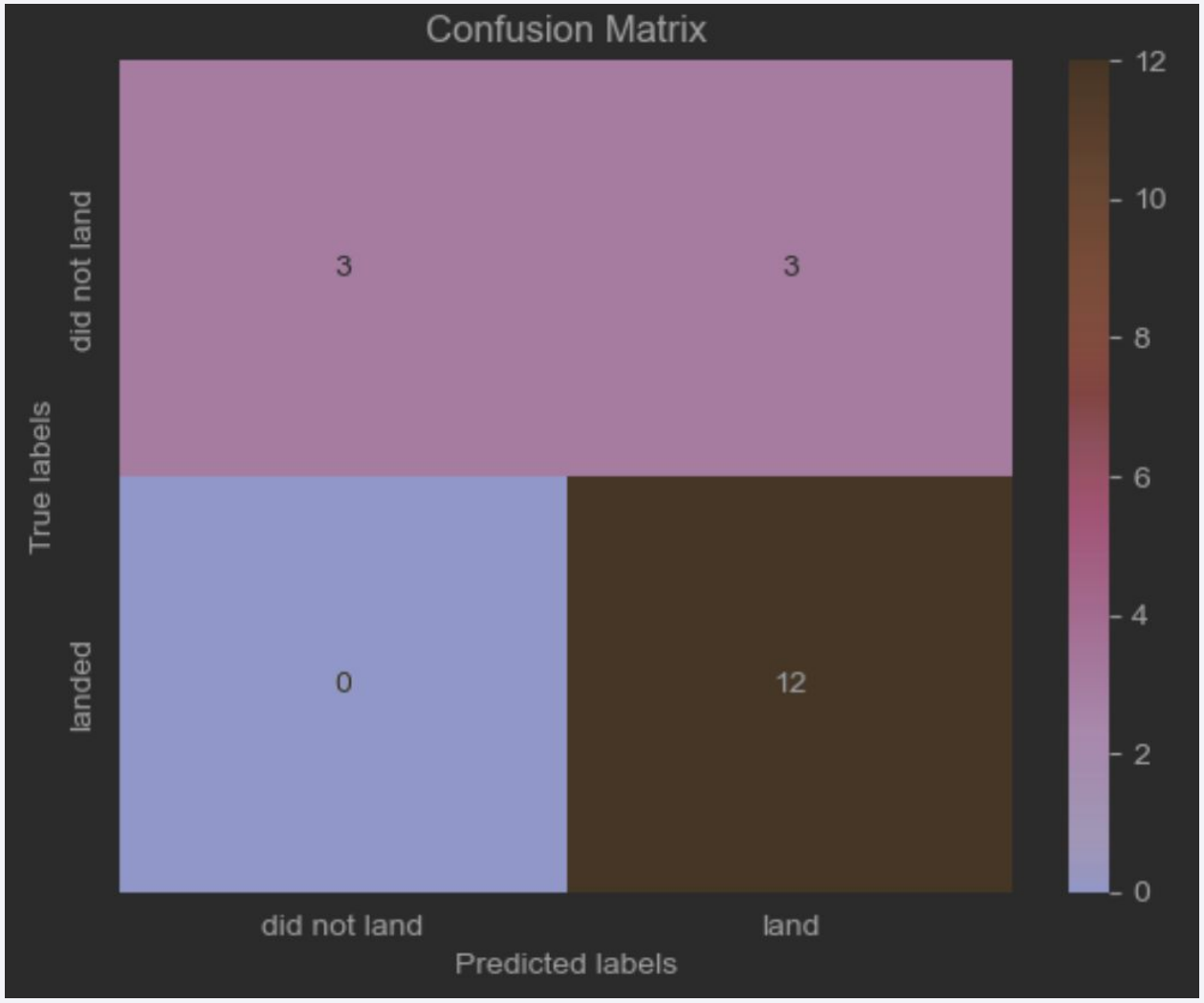
- Orbit ES-L1, SSO, HEO, and GEO have higher success rate.
- Found feature interactions (e.g., orbit & flight number) in terms of ensuring successful rate.
- Success rate sharply increased since 2013 and reached plateau in 2017.
- KSCLC-39A site has the highest successful rate
- Decision tree can be used to predict successful rate with an accuracy score of 0.89

Appendix



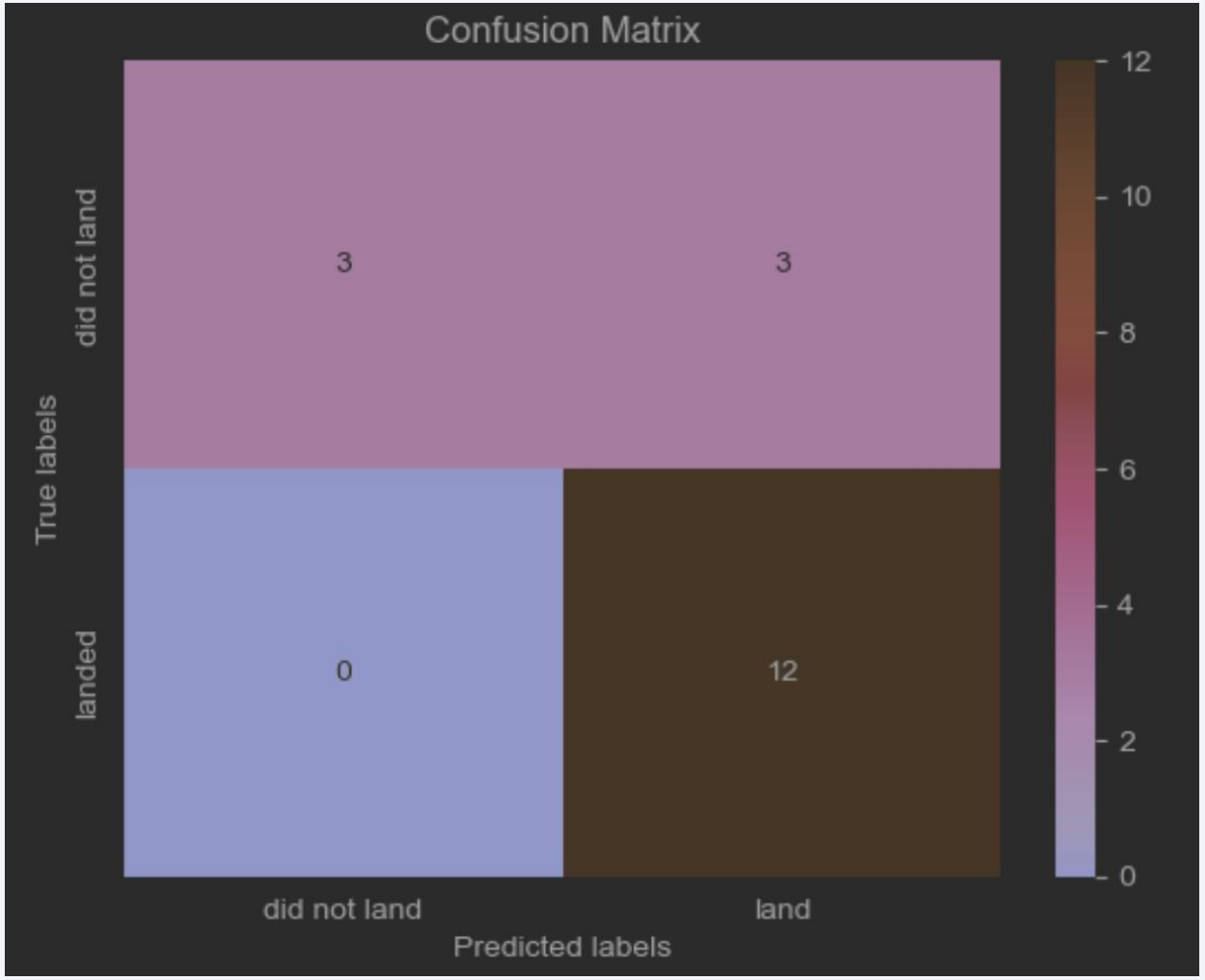
Confusion matrix for
Logistic Regression model.

Appendix



Confusion matrix for Support vector machine.

Appendix



Confusion matrix for KNN

Thank you!