IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nerea Caldés
13/08/2023

Outline

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

Executive Summary

I. Methodologies:
   I. Data Collection
   II. Data Wrangle
   III. EDA with SQL
   IV. EDA with Data Visualization
   V. Interactive visual analytics and Dashboards

II. Results:
   I. EDA
   II. Visualization
   III. Predictive analytics

Introduction

- Project background and context:

  SpaceX offers Falcon 9 rocket launches at $62 million, a significant cost reduction compared to other providers due to their ability to reuse the first stage. Predicting the first stage's landing outcome allows for accurate cost estimation of a launch. This data is valuable for competing companies considering bidding against SpaceX for a rocket launch. The module provides an introduction to the issue and equips participants with essential tools for course completion.

- Problems you want to find answers:

  - Trend in successful landings over a period

  - Elements that will influence the successful landing

  - Specific circumstances are necessary to guarantee the success landing

Methodology

Executive Summary

- **Data collection**: Data is collected using SpaceX REST API and also web scraping techniques (Wikipedia)

- Data Wrangle: Identify missing values, filter data, using One Hot encoding.

- **Perform exploratory data analysis (EDA) using visualization and SQL:**

    - **SQL**: Using SQL queries to extract necessary data from data base

    - Visualization: relationship between different parameters, using dummy variables and encoding to represent data in Scatter Graphs, Bar Graphs, Line Graphs…

- **Perform interactive visual analytics using Folium and Plotly Dash**

- Perform predictive analysis using classification models: How to build, tune, evaluate classification models

Data Collection – SpaceX API

Steps to get data from SpaceX API:

I. Use get function from library “requests“ to make HTTP request to API URL.

II. Decode response and normalize data using functions from “json” library

III. Get specific columns from API (with obtained IDs) and store it in lists

IV. Create a dictionary from lists data combining columns

V. Create Pandas data frame

Data Collection – SpaceX API

```
# Make HTTP request
import requests
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
# Decode response
import json
decoded = json.loads(response.content)
df = pd.json_normalize(decoded)
# Get specific data
data = df[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
# Create dictionary
launch_dict = {'FlightNumber': list(data['flight_number']),'Date': list(data['date']),'BoosterVersion':BoosterVersion,'PayloadMass':PayloadMass,
               'Orbit':Orbit,'LaunchSite':LaunchSite,'Outcome':Outcome,'Flights':Flights,'GridFins':GridFins,'Reused':Reused,'Legs':Legs,
               'LandingPad':LandingPad,'Block':Block,'ReusedCount':ReusedCount,'Serial':Serial,'Longitude': Longitude,'Latitude': Latitude}
# Create a data from launch_dict
pdf = pd.DataFrame(launch_dict)
```

Link to Github

Data Collection - Scraping

Steps to get data from SpaceX API:

- Request the HTML page from Wikipedia URL and get response
- Create a BeautifulSoup object from the HTML response
- Find tables and check content from a targeted table
- Iterate through <th> elements to extract the column names
- Create a dictionary with keys from the extracted column names
- Fill dictionary with data in the table
- Create Pandas data frame from dictionary

Data Collection – Scraping

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
response = requests.get(static_url)
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, "html.parser")
# Find tables and target the 3rd
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
# Fill data
column_names = []
html_th = first_launch_table.find_all('th')
for th_element in html_th:
    if th_element is not None and len(th_element) > 0:
        column_names.append(th_element.text.strip())
launch_dict= dict.fromkeys(column_names)
# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
# Here goes all the snippet to extract information from the rows
# Create data frame from dictionary
df=pd.DataFrame(launch_dict)
```

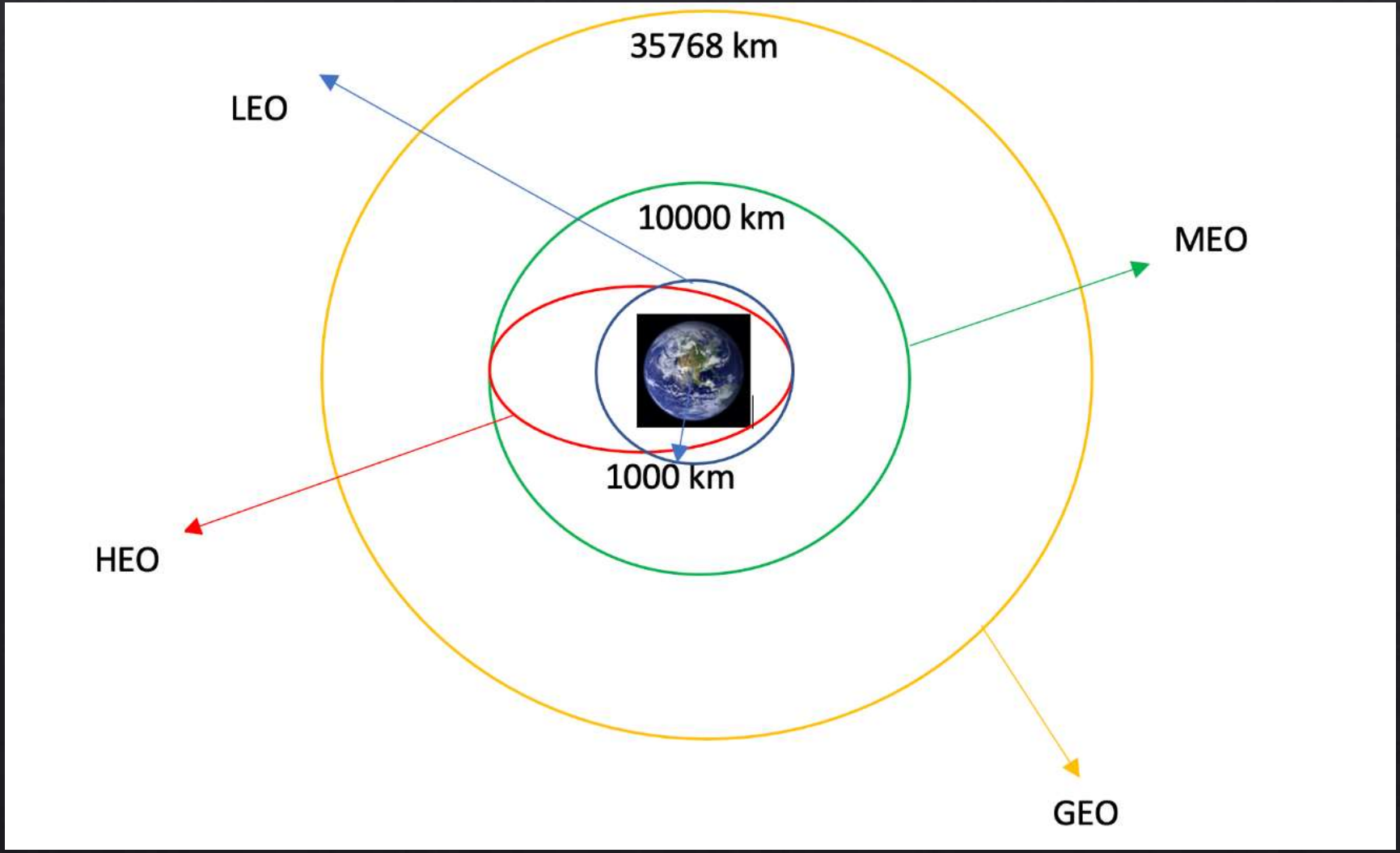
Link to Github

Data Wrangling

Steps to data wrangling:

- Perform EDA and determine training labels
- Calculate number of launches on each site
- Calculate number and occurrence of each orbit
- Calculate number and occurrence of mission outcome per orbit type
- Create a landing outcome label from outcome column considering 0 as bad outcome and 1 otherwise

Data Wrangling

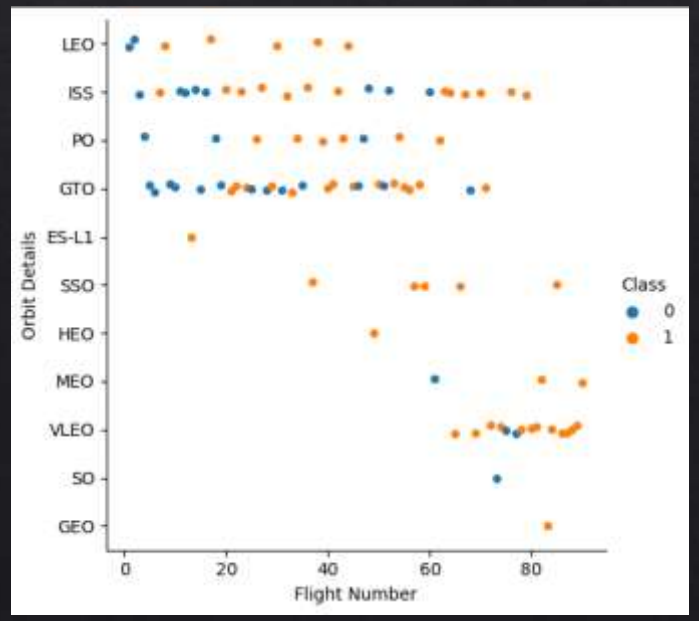


Link a Github

EDA with Data Visualization

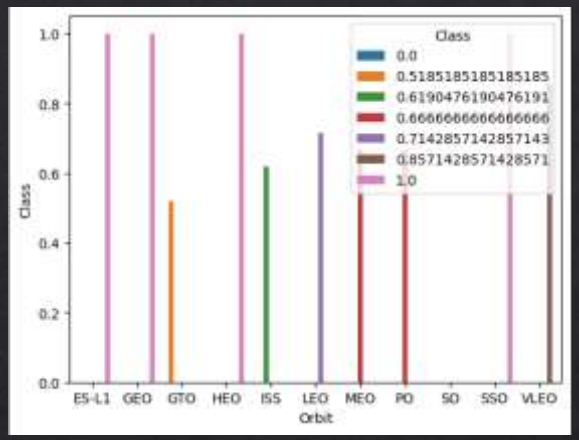
Scatter Graph for correlation analysis:

- Flight number vs Payload Mass
- Flight number vs Launch Site
- Payload vs Launch Site
- Orbit vs Flight number
- Payload vs Outcome
- Orbit vs Payload Mass



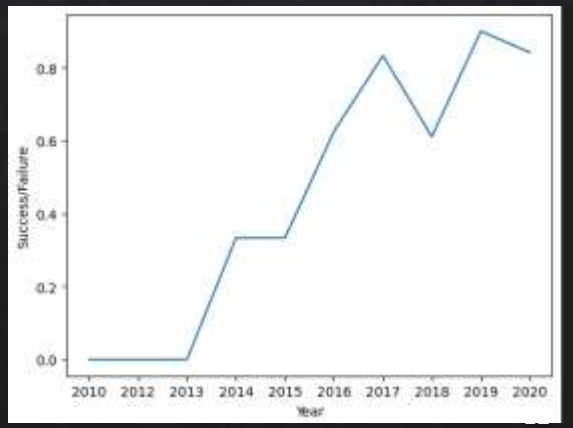
Bar Graphs:

- Orbit types



Line Graphs:

- Success Rate vs year



Link to Github

EDA with SQL

Connect to data base from Jupyter Notebook and few queries. Some of them are:

- Display names of the unique launch site in the space mission
- Display 5 records where launch sites begin with the string 'KSC’
- Display average payload mass carried by booster version F9 v1.1
- List the total number of successful and failure mission outcomes

Link to GitHub

Build an Interactive Map with Folium

- Mapped all launch sites, added markers, circles, and lines indicating launch success/failure on the folium map.
- Assign launch outcomes (0 is failure and 1 is success)
- Calculate distance from the launch site to different landmarks and draw lines in the map to measure distance

Link to GitHub

Build a Dashboard with Plotly Dash

- Added a launch site drop-down input component
- A callback function to render a pie chart for success records based on drop-down
- A range slider to select Payload
- A callback function to render the scatter chart for success records

Link to GitHub

Predictive Analysis (Classification)

- Load data and create an array using numpy.
- Standardize data and split it into training and testing data
- Create the following machine learning objects (Logistic regression, Vector machine , Decision Tree Classifier, k nearest neighbors) also with GridSearchCV and fit it to find the best parameters from given dictionaries.
- Calculate the accuracy on the test data using the method score() to find predictive data

Link to GitHub

Results

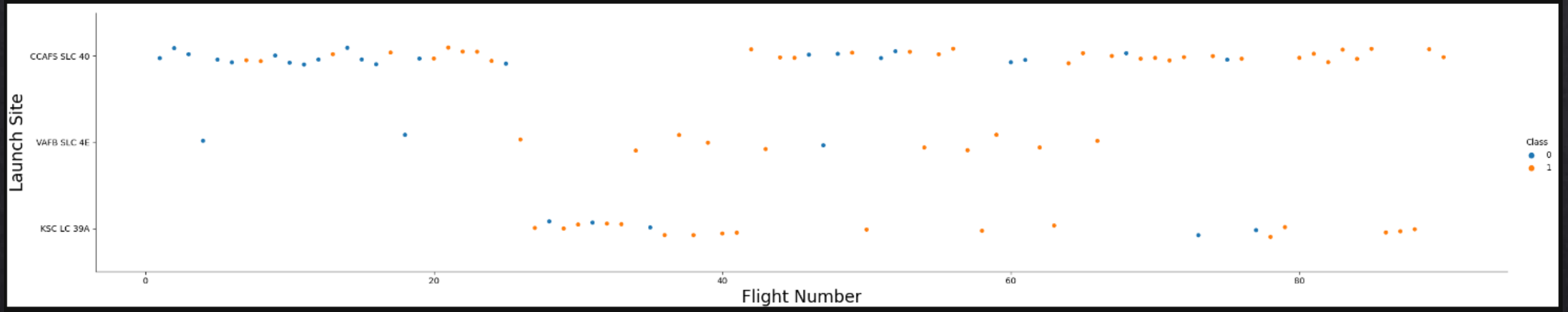
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

```
# Visualize the relationship between Flight Number and Launch Site
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

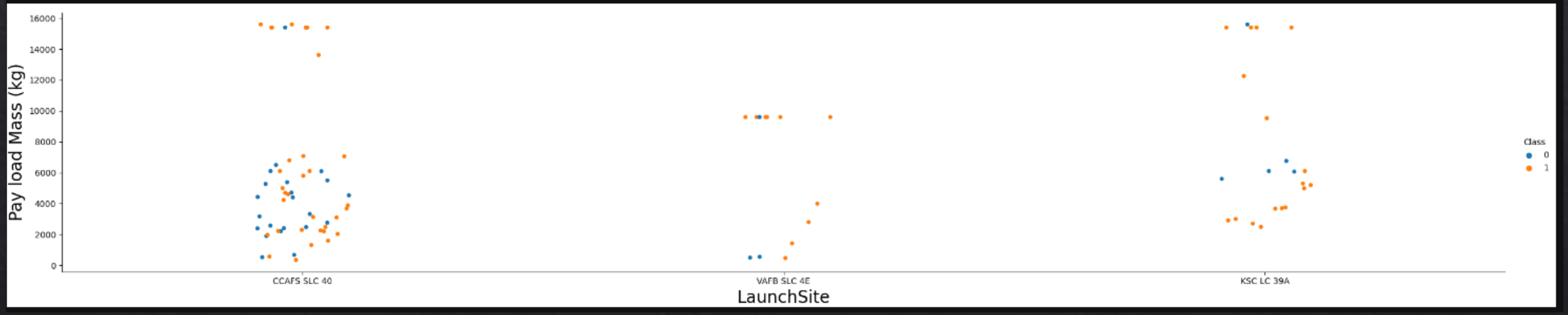


It is shown that the more amount of flight number is at launch site, the greater the success rate at launch site

Link to GitHub

Payload vs. Launch Site

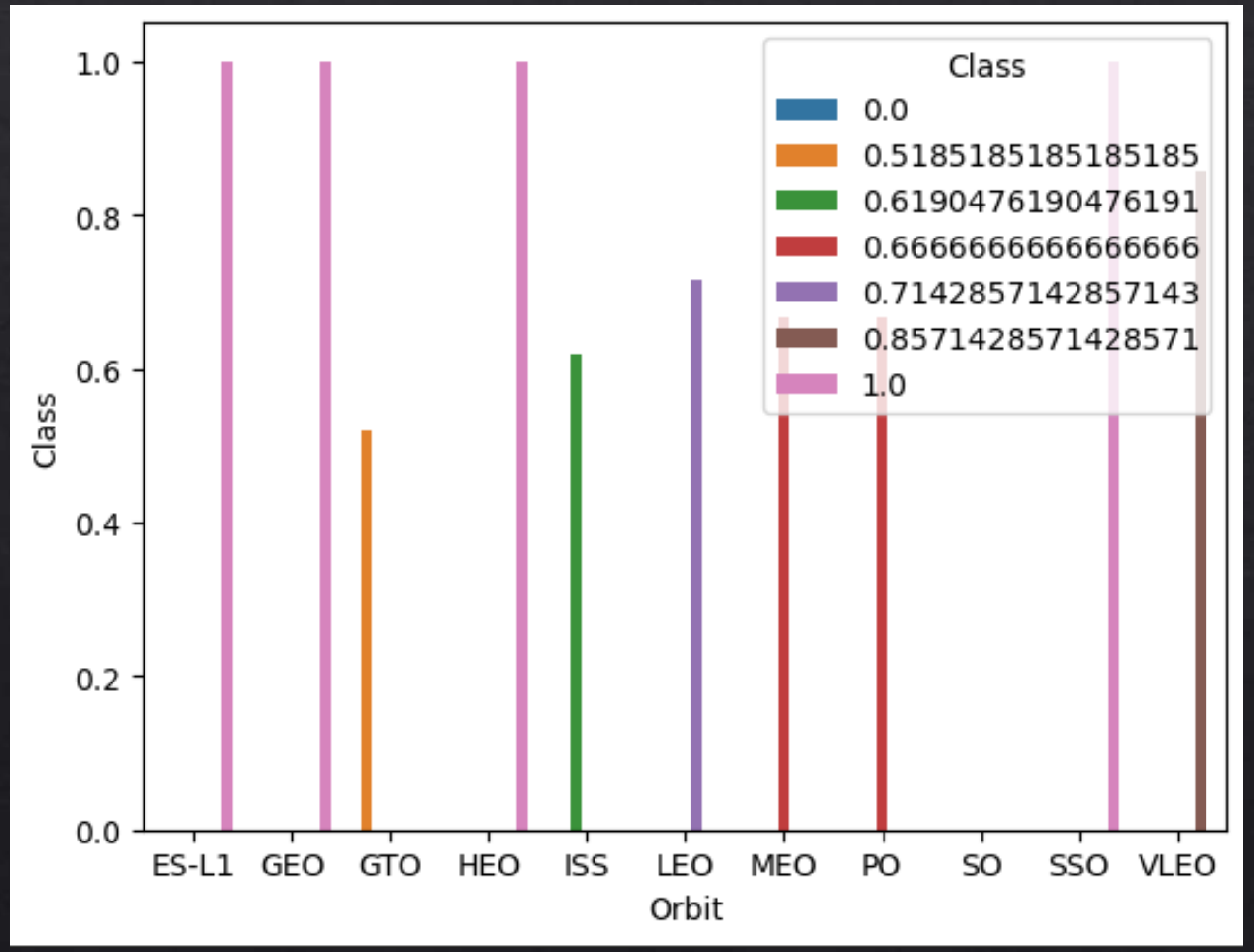
```
# Visualize the relationship between Payload and Launch Site
sns.catplot(y="PayloadMass", x="LaunchSite", hue="Class", data=df, aspect = 5)
plt.xlabel("LaunchSite",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



Here, as higher the Payload Mass is, the higher the success rate.

Link to GitHub

Success Rate vs. Orbit Type

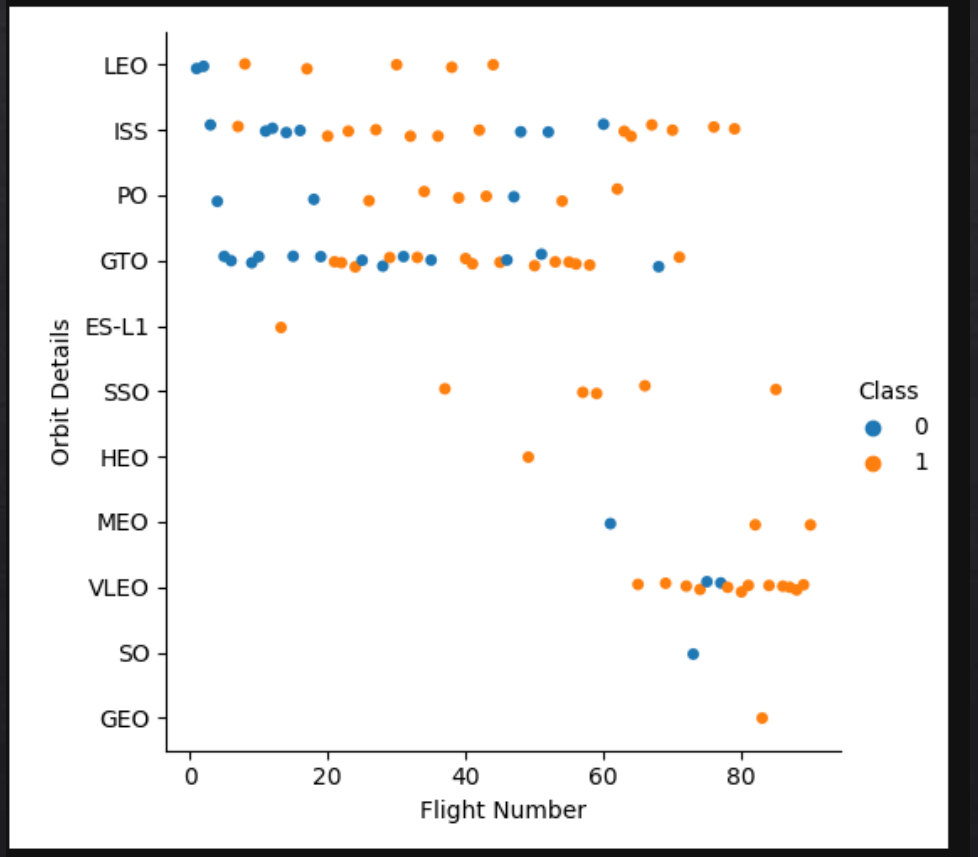


Orbits ES-L1, GEO, HEO and SSO have 100% Success Rate. Lowest is SO

Link to GitHub

Flight Number vs. Orbit Type

```
### TASK  4: Visualize the relationship between FlightNumber and Orbit type
sns.catplot(x='FlightNumber',y='Orbit',data=df,hue='Class')
plt.xlabel('Flight Number')
plt.ylabel('Orbit Details')
plt.show()
```

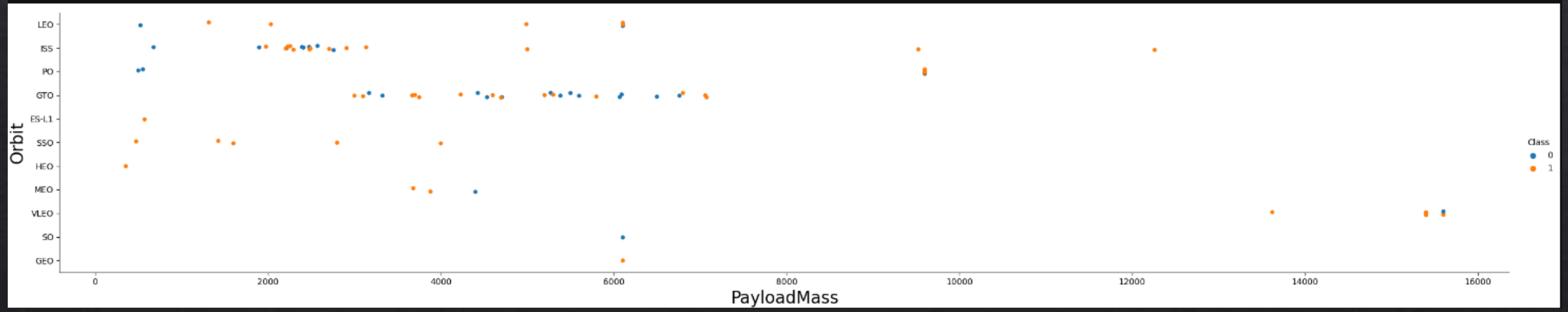


LEO orbit: Success tied to flight count.
GTO orbit: No correlation.

Link to GitHub

Payload vs. Orbit Type

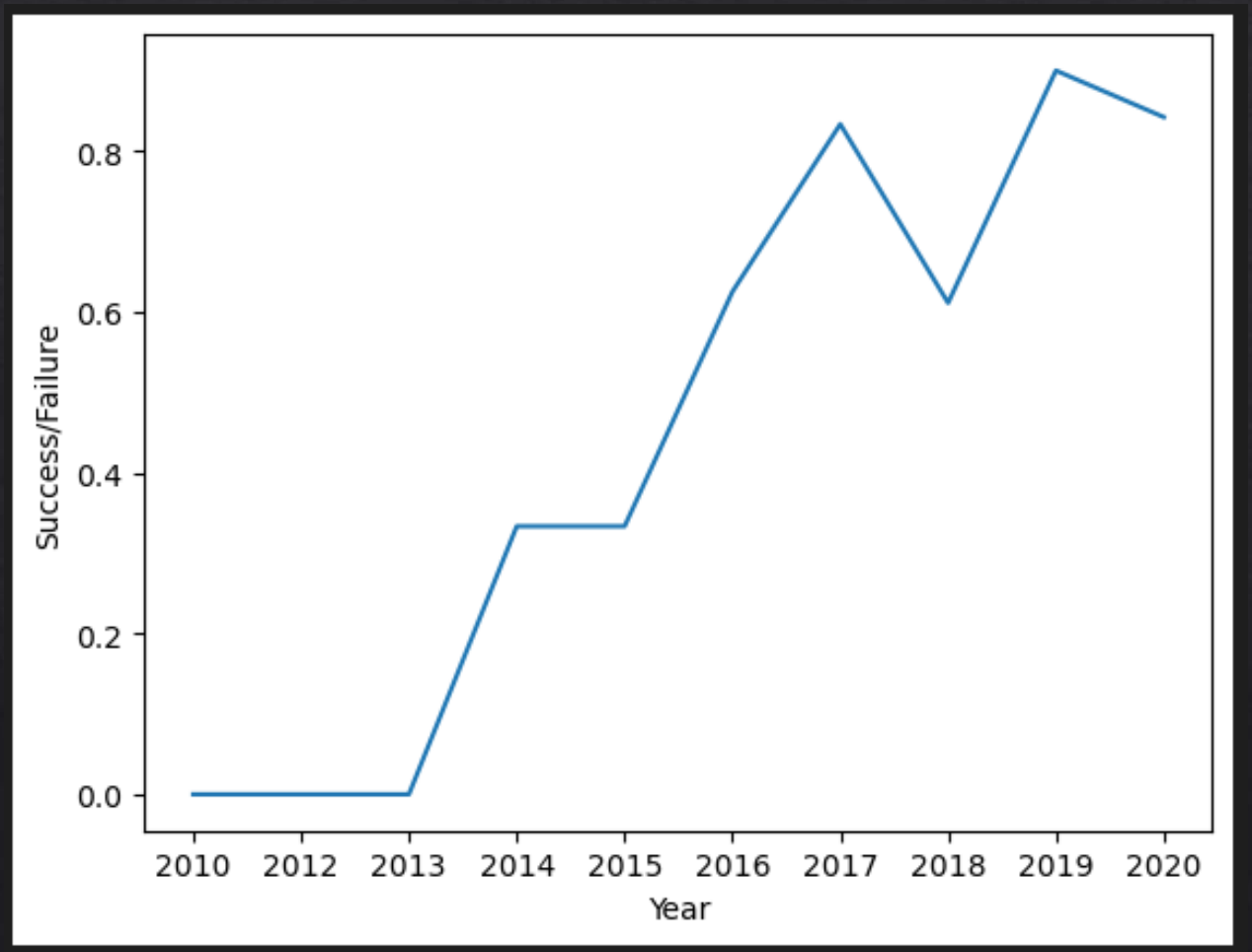
```
### TASK  5: Visualize the relationship between Payload and Orbit type
sns.catplot(x='PayloadMass',y='Orbit',data=df,hue='Class', aspect= 5)
plt.xlabel('PayloadMass',fontsize=20)
plt.ylabel('Orbit',fontsize=20)
plt.show()
```



Heavier payloads correlate with higher successful landings in PO, LEO, and ISS orbits.

Link to GitHub

Launch Success Yearly Trend



The plot shows a consistent increase in success rate from 2013 to 2020.

Link to GitHub

All Launch Site Names

```
%sql select distinct(LAUNCH_SITE) from SPACEXTBL;
```

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

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

"Distinct" keyword used to display unique launch sites in SpaceX data.

Link to GitHub

Launch Site Names Begin with 'KSC'

```
%sql SELECT * from SPACEXTBL where (LAUNCH_SITE) LIKE 'KSC%' LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
19/02/2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490.0	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
16/03/2017	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600.0	GTO	EchoStar	Success	No attempt
30/03/2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300.0	GTO	SES	Success	Success (drone ship)
05/01/2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300.0	LEO	NRO	Success	Success (ground pad)
15/05/2017	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070.0	GTO	Inmarsat	Success	No attempt

Used query to show 5 'KSC' launch site records.

Link to GitHub

Total Payload Mass

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

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

payloadmass
619967.0

Query yielded NASA's booster payload sum: 45596.

Link to GitHub

Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

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

payloadmass
6138.287128712871

Average payload mass of F9 v1.1 booster: 2928.4.

Link to GitHub

First Successful Ground Landing Date

```
%sql select min(DATE) from SPACEXTBL;
```

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

min(DATE)
01/06/2014

First successful ground pad landing:
June 1, 2014.

Link to GitHub

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select BOOSTER_VERSION from SPACEXTBL 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

Filtered for successful drone ship landings and payload mass between 4000 and 6000 using WHERE and AND clauses.

Link to GitHub

Total Number of Successful and Failure Mission Outcomes

```
%sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

missionoutcomes
0
1
98
1
1

Utilized wildcard '%' to filter WHERE MissionOutcome as success or failure.

Link to GitHub

Boosters Carried Maximum Payload

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

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

boosterversion
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

Identified max payload-carrying booster using subquery in WHERE with MAX() function.

Link to GitHub

2017 Launch Records

```
%sql SELECT substr(Date, 4, 2),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where substr(Date,7,4)='2017'
 * sqlite:///my_data1.db
Done.
```

substr(Date, 4, 2)	Mission_Outcome	Booster_Version	Launch_Site
01	Success	F9 FT B1029.1	VAFB SLC-4E
02	Success	F9 FT B1031.1	KSC LC-39A
03	Success	F9 FT B1030	KSC LC-39A
03	Success	F9 FT B1021.2	KSC LC-39A
01	Success	F9 FT B1032.1	KSC LC-39A
05	Success	F9 FT B1034	KSC LC-39A
03	Success	F9 FT B1035.1	KSC LC-39A
06	Success	F9 FT B1029.2	KSC LC-39A
06	Success	F9 FT B1036.1	VAFB SLC-4E
05	Success	F9 FT B1037	KSC LC-39A
08	Success	F9 B4 B1039.1	KSC LC-39A
08	Success	F9 FT B1038.1	VAFB SLC-4E
07	Success	F9 B4 B1040.1	KSC LC-39A
09	Success	F9 B4 B1041.1	VAFB SLC-4E
11	Success	F9 FT B1031.2	KSC LC-39A
10	Success	F9 B4 B1042.1	KSC LC-39A
12	Success	F9 FT B1035.2	CCAFS SLC-40
12	Success	F9 FT B1036.2	VAFB SLC-4E

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

Link to GitHub

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

```
%sql SELECT LANDING_OUTCOME FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;
```

Landing_Outcome
No attempt
Success (ground pad)
Success (ground pad)
Success (drone ship)
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)
Controlled (ocean)

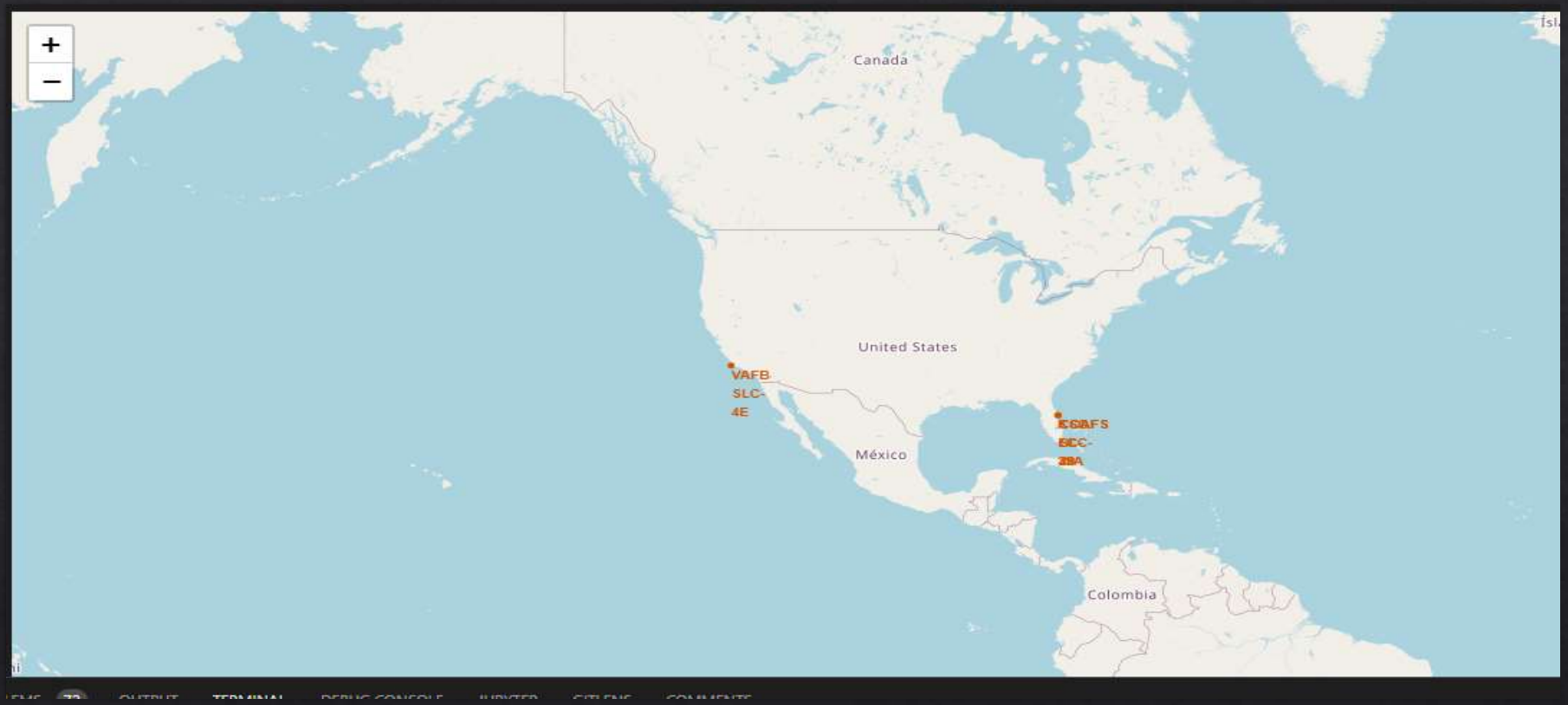
Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Link to GitHub

Section 3

Launch Sites Proximities Analysis

All launch sites map markers



Proximity to equator aids equatorial launches, leveraging Earth's rotation for prograde orbits. Near-equator sites gain natural boost, saving fuel and costs.

Link to GitHub

<Folium Map Screenshot 2>

- Replace <Folium map screenshot 2> title with an appropriate title

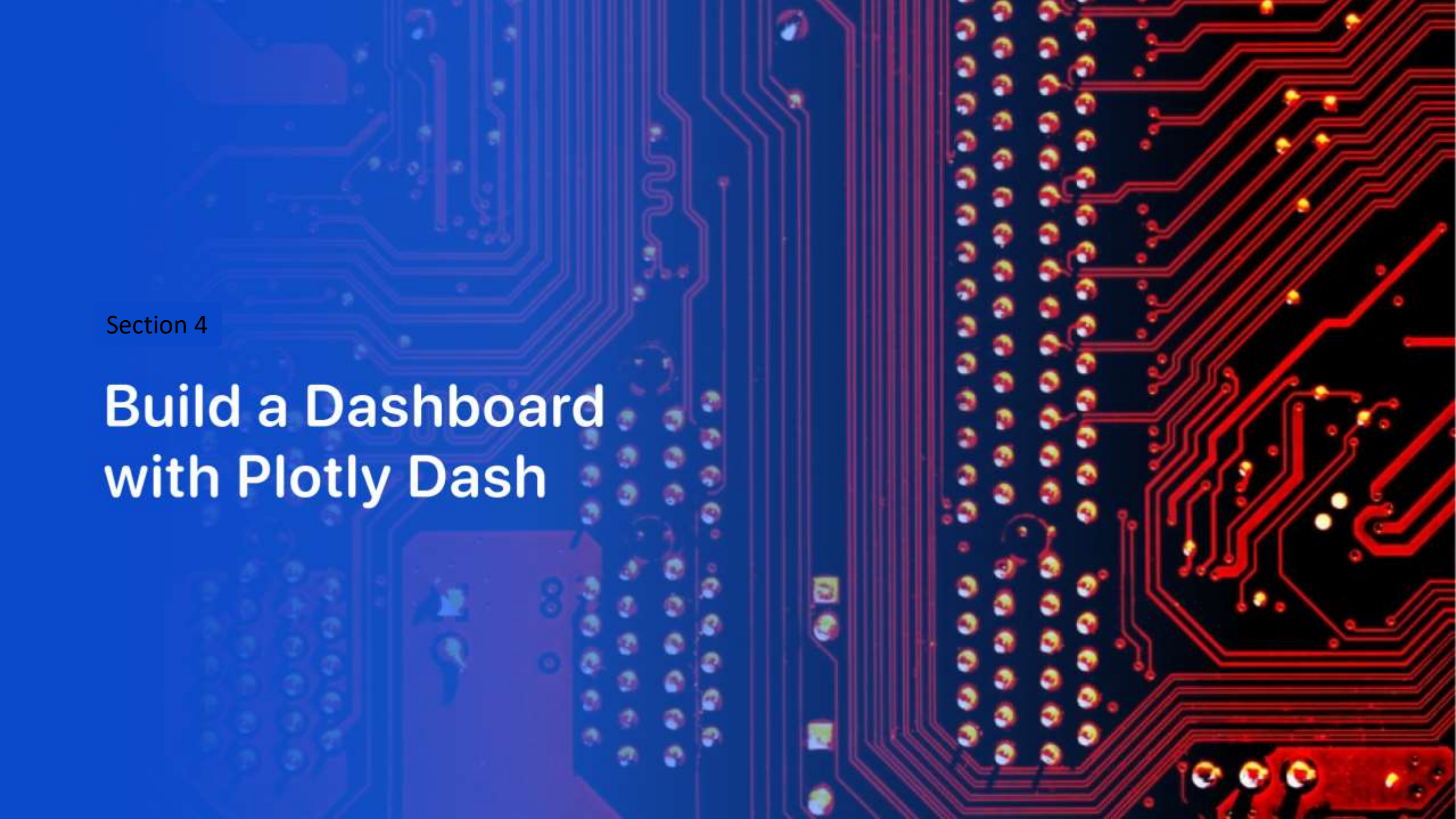
- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map

- Explain the important elements and findings on the screenshot

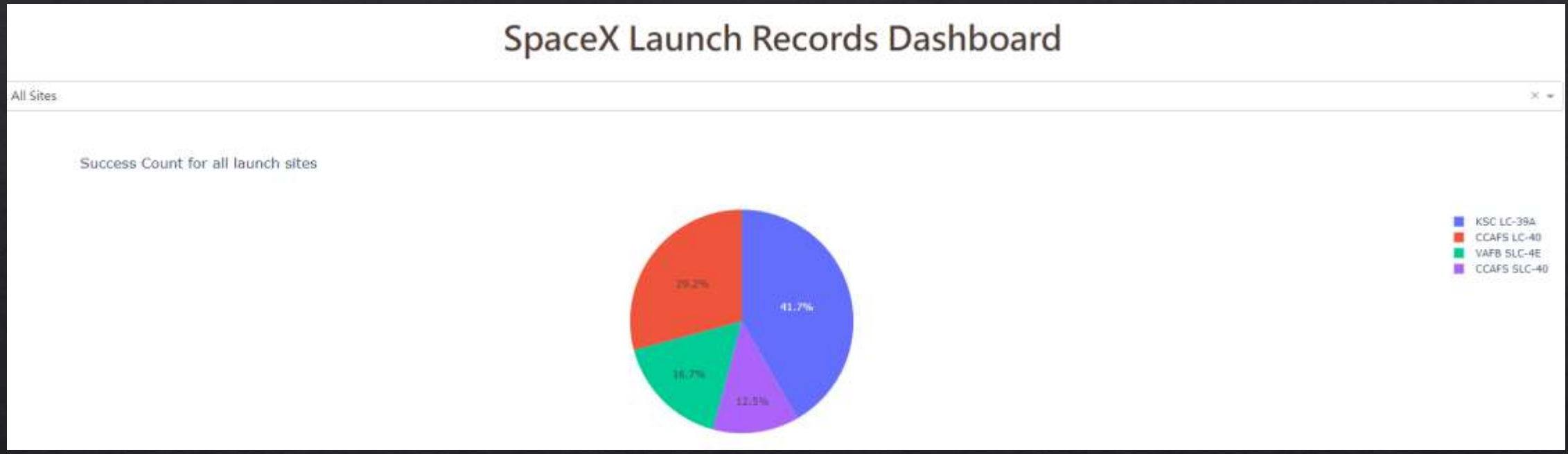
Link to GitHub

Section 4

Build a Dashboard with Plotly Dash



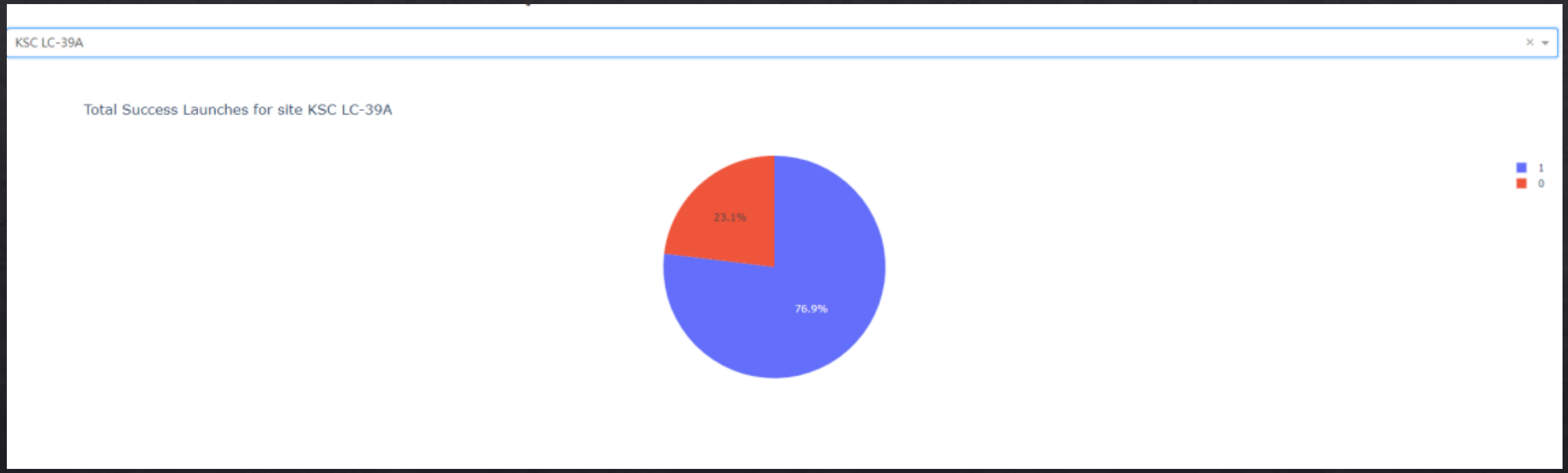
Success launches Pie Chart



KSC LC-39A has the most successful launches from all sites.

Link to GitHub

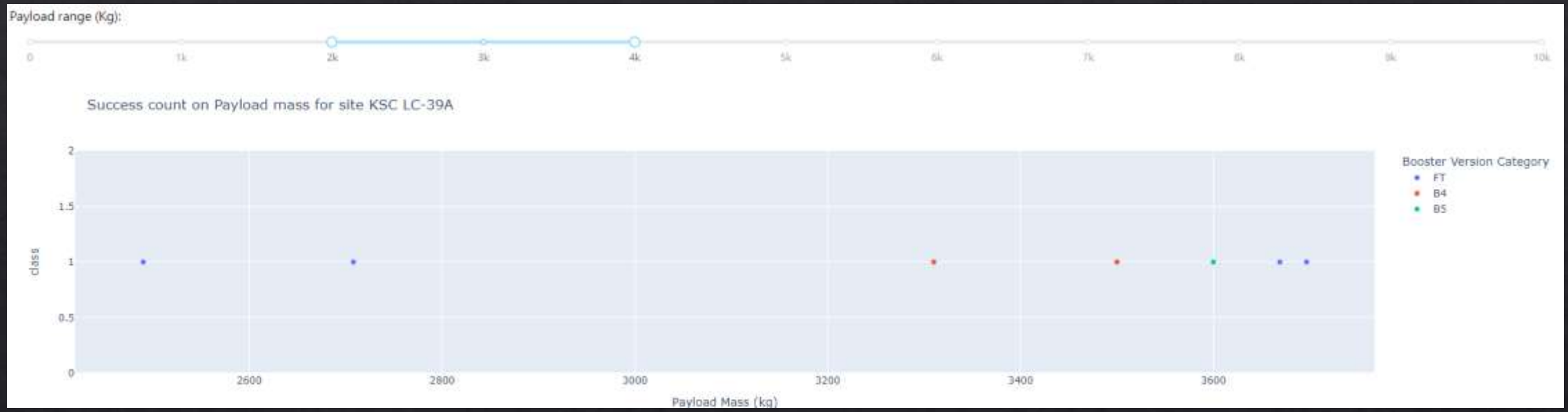
High launch success ratio pie chart



KSC LC-39A has the highest success rate amongst launch sites

Link to GitHub

Payload Mass and Success



Link to GitHub

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
models = {'KNeigbors':knn_cv.best_score_, 'DecisionTree':tree_cv.best_score_, 'LogisticRegression':logreg_cv.best_score_, 'SupportVector':svm_cv.best_score_}
best_algorithm = max(models, key=models.get)
print('Best model is ', best_algorithm, 'with score = ', models[best_algorithm])
if best_algorithm == 'KNeigbors':
    print('Best params are: ', knn_cv.best_params_)
if best_algorithm == 'DecisionTree':
    print('Best params are: ', tree_cv.best_params_)
if best_algorithm == 'LogisticRegression':
    print('Best params are: ', logreg_cv.best_params_)
if best_algorithm == 'SupportVector':
    print('Best params are: ', svm_cv.best_params_)
```

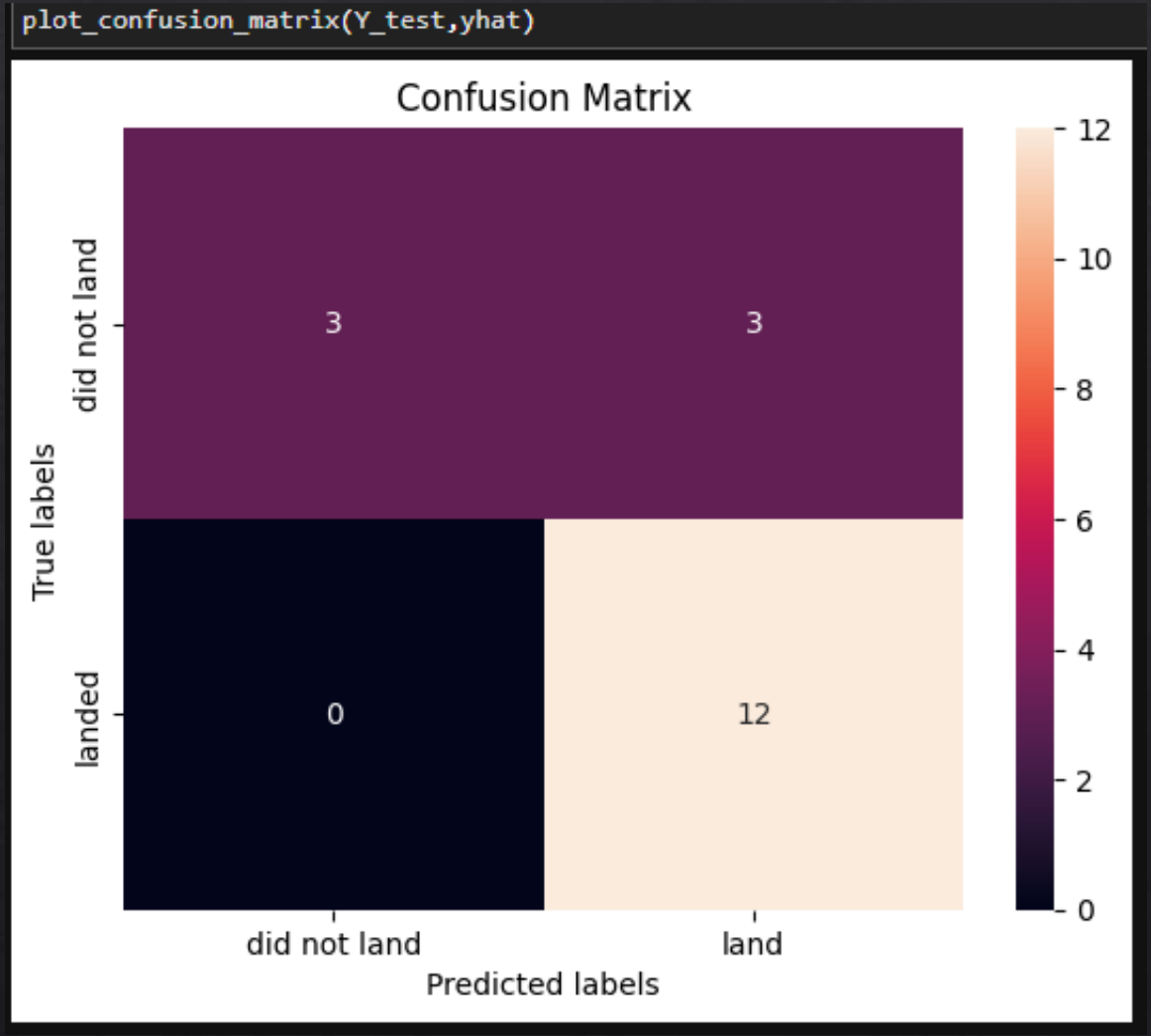
```
Best model is  DecisionTree with score =  0.8732142857142857
Best params are:  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'random'}
```

The decision tree classifier is the model with the highest classification accuracy

Link to GitHub

Confusion Matrix

```
plot_confusion_matrix(Y_test,yhat)
```



Decision tree matrix differentiates classes but struggles with false positive: unsuccessful landings misclassified as successful.

Link to GitHub

Conclusions

- Positive correlation between launch site flight volume and success rate.
- Orbits: ES-L1, GEO, HEO, SSO, VLEO excelled in success.
- KSC LC-39A dominated successful launches.
- Decision tree classifier emerged as optimal ML algorithm for task.

Link to GitHub

Thank you!