IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

Summary of methodologies

- Data collection Via API
- Web Scraping
- SQL
- Interactive Map with Folium
- Creating Dashboard with Plotly
- Predictive Analysis

Summary of all results

- Exploratory Data Analysis results
- Interactive Map
- Predictive results

Introduction

- **Project background and context**

We will predict whether the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website at a cost of $62 million, while other providers charge upwards of $165 million each. Much of the cost savings come from SpaceX's ability to reuse the first stage. Therefore, by determining if the first stage will land, we can estimate the cost of a launch. This information is valuable for other companies looking to compete with SpaceX for rocket launches. In this module, you will receive an overview of the problem and the tools necessary to achieve our goal.

- **Problems you want to find answers**

Is the first stage going to land successfully or not?

What attributes are correlated with successful landings?

What are the conditions which will allow SpaceX to achieve the best landing success rate ?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scrapping
- Perform data wrangling
  - Dropping unnecessary columns
- 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

We worked with SpaceX launch data obtained from the SpaceX REST API. This API provided information about launches, including rocket details, payloads, launch and landing specifications, and outcomes. Our objective was to predict whether SpaceX would attempt to land a rocket. The API's base URL was api.spacexdata.com/v4/, with endpoints such as /capsules, /cores, and /launches/past. We used the requests library to fetch past launch data, which was returned as a JSON list of launch objects. We converted this JSON data into a DataFrame using the json_normalize function for analysis.

Additionally, we used web scraping with the BeautifulSoup package to extract Falcon 9 launch data from HTML tables on relevant Wiki pages, converting them into Pandas DataFrames for further analysis. We aimed to transform raw data into a clean dataset, addressing data wrangling, sampling, and handling null values. For example, some columns contained ID numbers instead of detailed data, requiring additional API calls to endpoints like Booster, Launchpad, Payload, and Core. We filtered out Falcon 1 launches and handled null values, particularly in the PayloadMass column, by replacing them with the column's mean value. The LandingPad column was dealt with using one-hot encoding later.

Data Collection – SpaceX API

1. Getting Response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Convert Response to JSON File

```
data = response.json()
data = pd.json_normalize(data)
```

3. Transform data

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
getBoosterVersion(data)
```

4. Create dictionary with data

```
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}
```

5. Create dataframe

```
data = pd.DataFrame.from_dict(launch_dict)
```

6. Filter dataframe

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

7. Export to file

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

1. Getting Response from HTML

```
response = requests.get(static_url)
```

2. Create BeautifulSoup Object

```
soup = BeautifulSoup(response.text, "html5lib")
```

3. Find all tables

```
html_tables = soup.findAll('table')
```

4. Get column names

```
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0 :
        column_names.append(name)
```

5. Create dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# 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'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Add data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is a
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.stri
                flag=flight_number.isdigit()
```

<u>**See notebook for the rest of code**</u>

7. Create dataframe from dictionary

```
df=pd.DataFrame(launch_dict)
```

8. Export to file

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

• In the dataset, there are several cases where the booster did not land successully. • True Ocean, True RTLS, True ASDS means the mission has been successful.

• False Ocean, False RTLS, False ASDS means the mission was a failure.

• We need to transform string variables into categorical variables where 1 means the mission has been successful and 0 means the mission was a failure.

1. Calculate launches number for each site

```
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

2. Calculate the number and occurence of each orbit

```
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
SO        1
ES-L1     1
HEO       1
GEO       1
Name: Orbit, dtype: int64
```

3. Calculate number and occurrence of mission outcome per orbit type

```
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
None ASDS       2
False Ocean     2
False RTLS      1
Name: Outcome, dtype: int64
```

4. Create landing outcome label from Outcome column

```
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
df['Class']=landing_class
```

5. Export to file

```
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

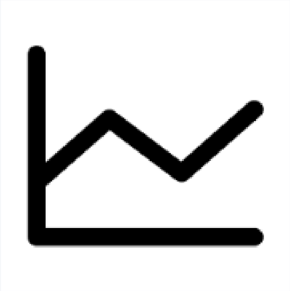
- Scatter Graphs
  - Flight Number vs. Payload Mass
  - Flight Number vs. Launch Site
  - Payload vs. Launch Site
  - Orbit vs. Flight Number
  - Payload vs. Orbit Type
  - Orbit vs. Payload Mass

Scatter plots show relationship between variables. This relationship is called the correlation.

- Bar Graph
  - Success rate vs. Orbit

Bar graphs show the relationship between numeric and categoric variables.

- Line Graph
  - Success rate vs. Year



Line graphs show data variables and their trends. Line graphs can help to show global behavior and make prediction for unseen data.

EDA with SQL

- We performed SQL queries to gather and understand data from dataset.

The links will be provided in the last slide.

Build an Interactive Map with Folium

- Folium map object is a map centered on NASA Johnson Space Center at Houson, Texas
- The objects were created in order to understand better the problem and the data. We can show easily all launch sites, their surroundings and the number of successful and unsuccessful landings.

Build a Dashboard with Plotly Dash

- Dashboard has dropdown, pie chart, rangeslider and scatter plot components
- Dropdown allows a user to choose the launch site or all launch sites (dash_core_components.Dropdown).
- Pie chart shows the total success and the total failure for the launch site chosen with the dropdown component (plotly.express.pie).
- Rangeslider allows a user to select a payload mass in a fixed range (dash_core_components.RangeSlider).
- Scatter chart shows the relationship between two variables, in particular Success vs Payload Mass (plotly.express.scatter)

Predictive Analysis (Classification)

Data preparation

- Load dataset
- Normalize data
- Split data into training and test sets.
- Model preparation
- Selection of machine learning algorithms
- Set parameters for each algorithm to GridSearchCV
- Training GridSearchModel models with training dataset

Model evaluation

- Get best hyperparameters for each type of model
- Compute accuracy for each model with test dataset
- Plot Confusion Matrix

Model comparison

- Comparison of models according to their accuracy
- The model with the best accuracy will be chosen (see Notebook for result)

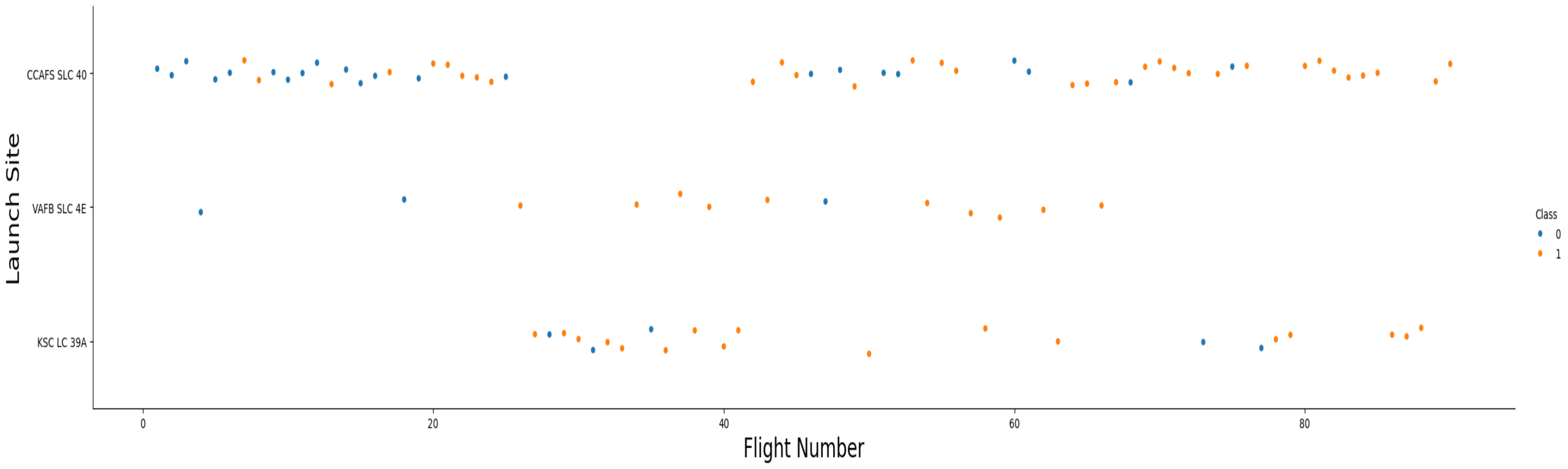
Results

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

Section 2

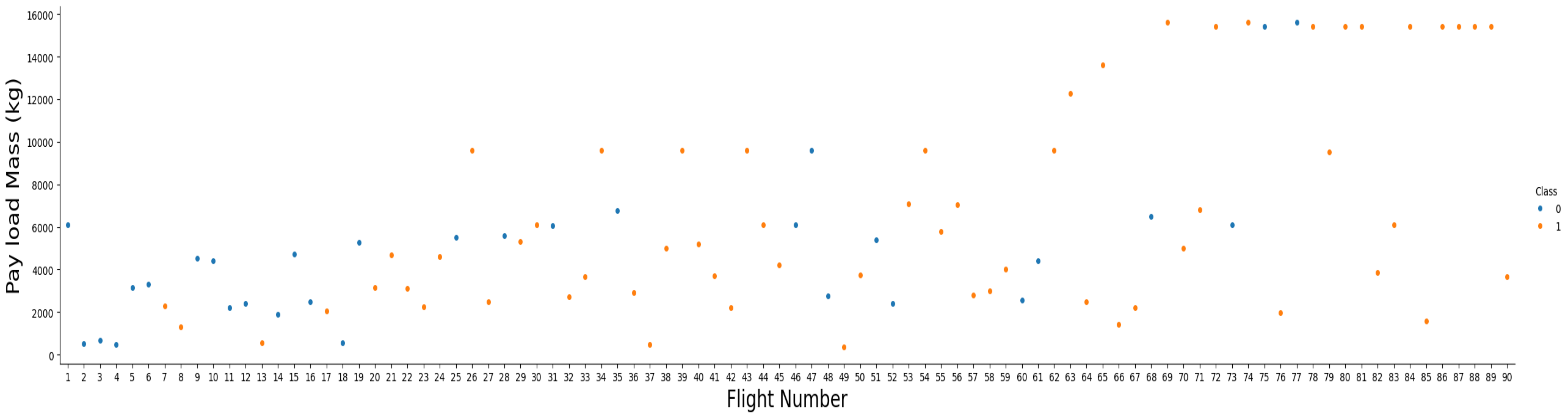
Insights drawn from EDA

Flight Number vs. 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()
```

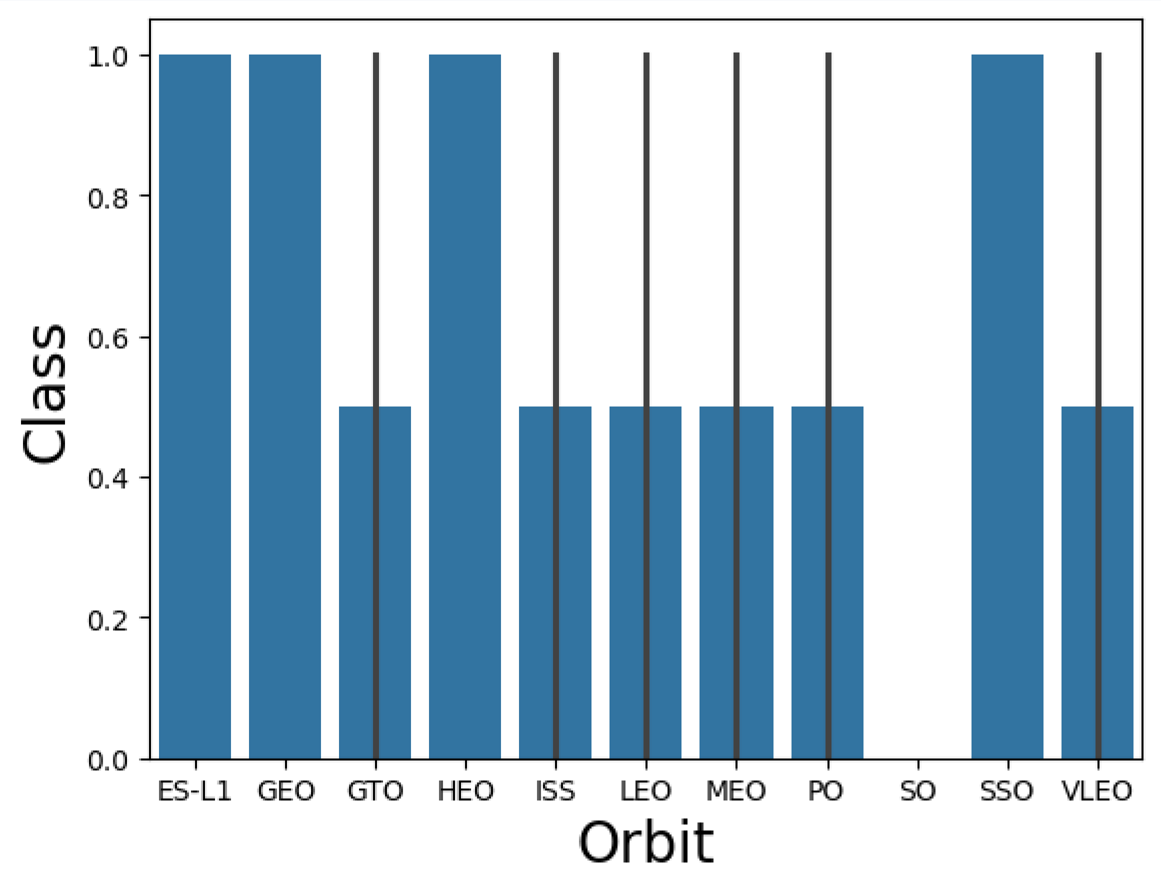
Payload vs. Launch Site




```
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("PayloadMass",fontsize=20)
plt.show()
```

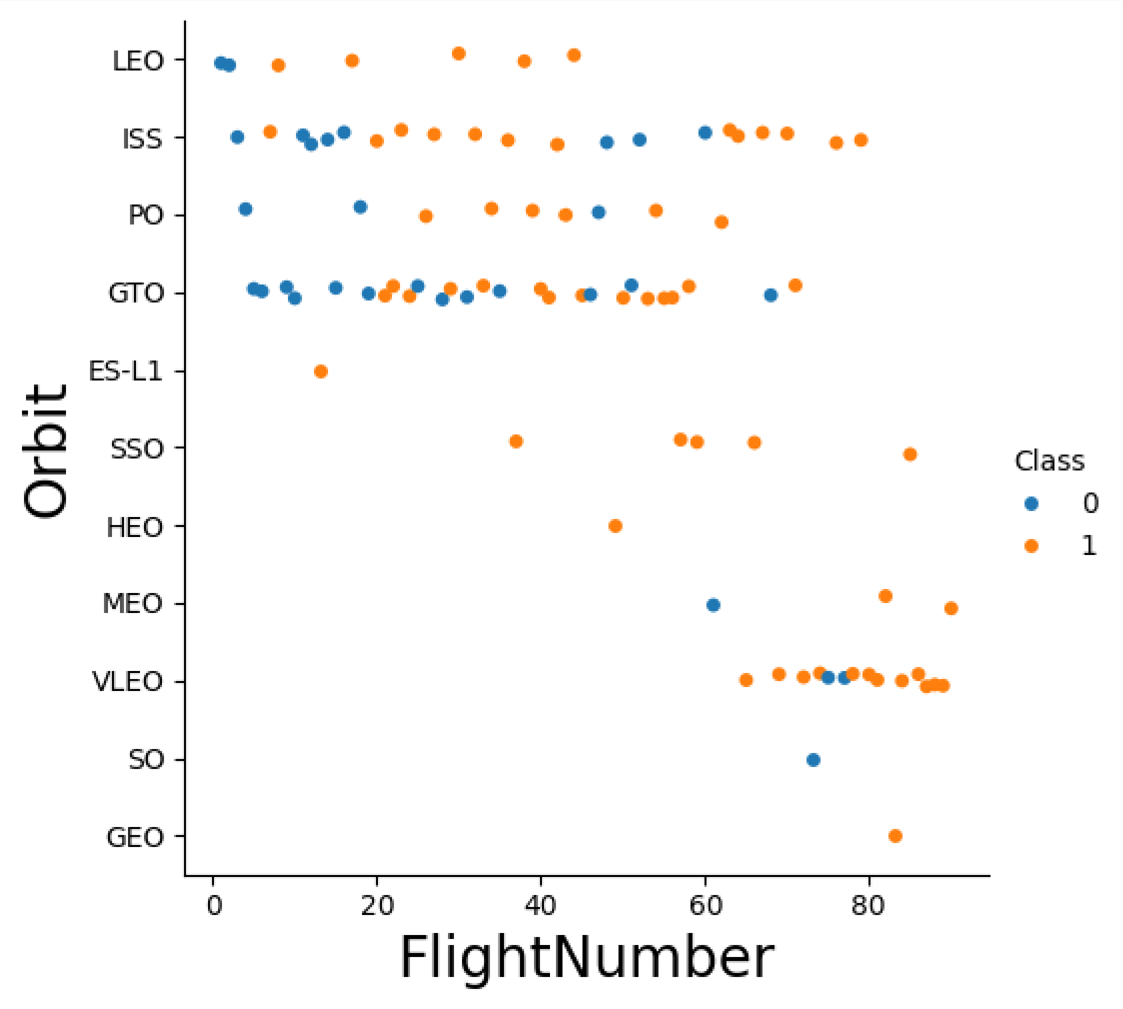
Success Rate vs. Orbit Type

- `t = df.groupby(['Orbit', 'Class'])['Class'].agg(['mean']).reset_index()`
- `sns.barplot(y="Class", x="Orbit", data=t)`
- `plt.xlabel("Orbit",fontsize=20)`
- `plt.ylabel("Class",fontsize=20)`
- `plt.show()`



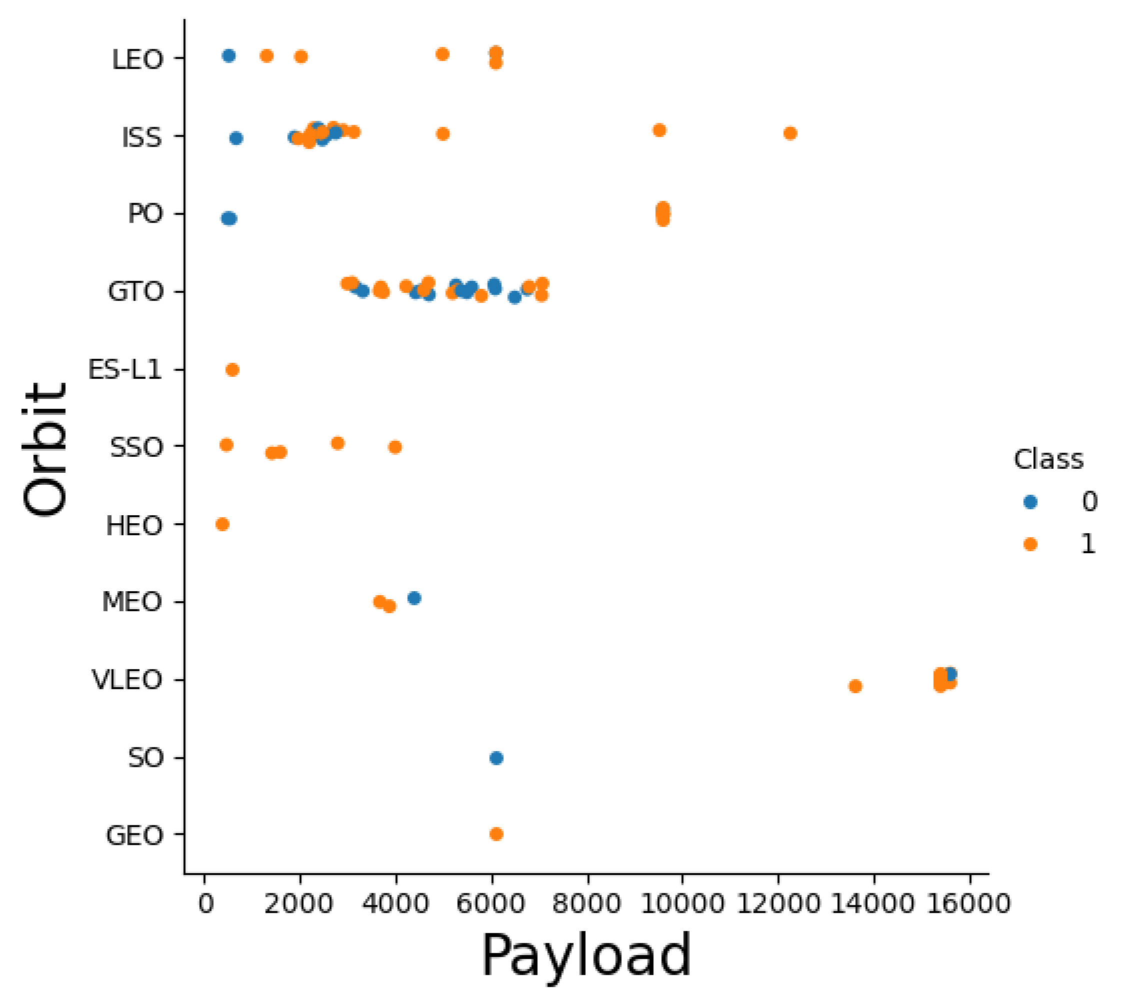
Flight Number vs. Orbit Type

```
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()
```



Payload vs. Orbit Type

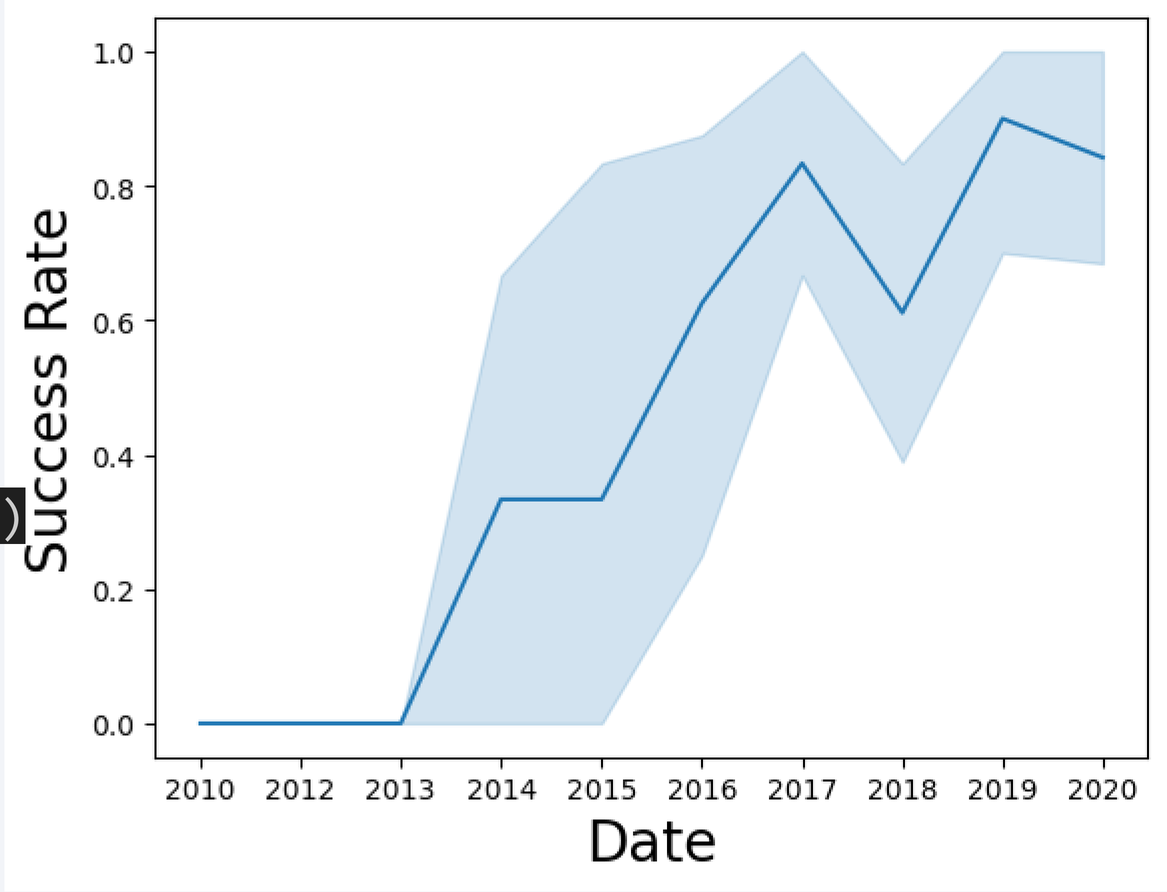
```
sns.catplot(y="Orbit", x="PayloadMass",
hue="Class", data=df)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

```
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
```

```
sns.lineplot(data=df, x="Date", y="Class")
plt.xlabel("Date",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



All Launch Site Names

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

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

Launch Site Names Begin with 'CCA'

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX
08-12-2010	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
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)

Total Payload Mass

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

SUM("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_)  FROM
SPACEXTABLE WHERE Booster_Version ='F9 v1.1'
```

AVG("PAYLOAD_MASS__KG_")
2534.6666666666665

First Successful Ground Landing Date

```
%sql SELECT MIN("Date") FROM SPACEXTABLE WHERE
Landing_Outcome= "Success (drone ship)"
```

MIN("DATE")
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
SELECT Booster_Version FROM SPACEXTABLE
WHERE Landing_Outcome = 'Success (drone ship)'
AND PAYLOAD_MASS__KG_ > 4000
AND PAYLOAD_MASS__KG_ < 6000;
```

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

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT  COUNT(Mission_Outcome) FROM
SPACEXTABLE WHERE Mission_Outcome like
'Success%'
```

SUCCESS	FAILURE
100	1

Boosters Carried Maximum Payload

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

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

```
%%sql
SELECT Landing_Outcome , Booster_Version ,
Launch_Site , 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 MonthName
FROM SPACEXTABLE WHERE substr(Date,0,5)='2015'
AND Landing_Outcome LIKE 'Failure (drone ship)'
```

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

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

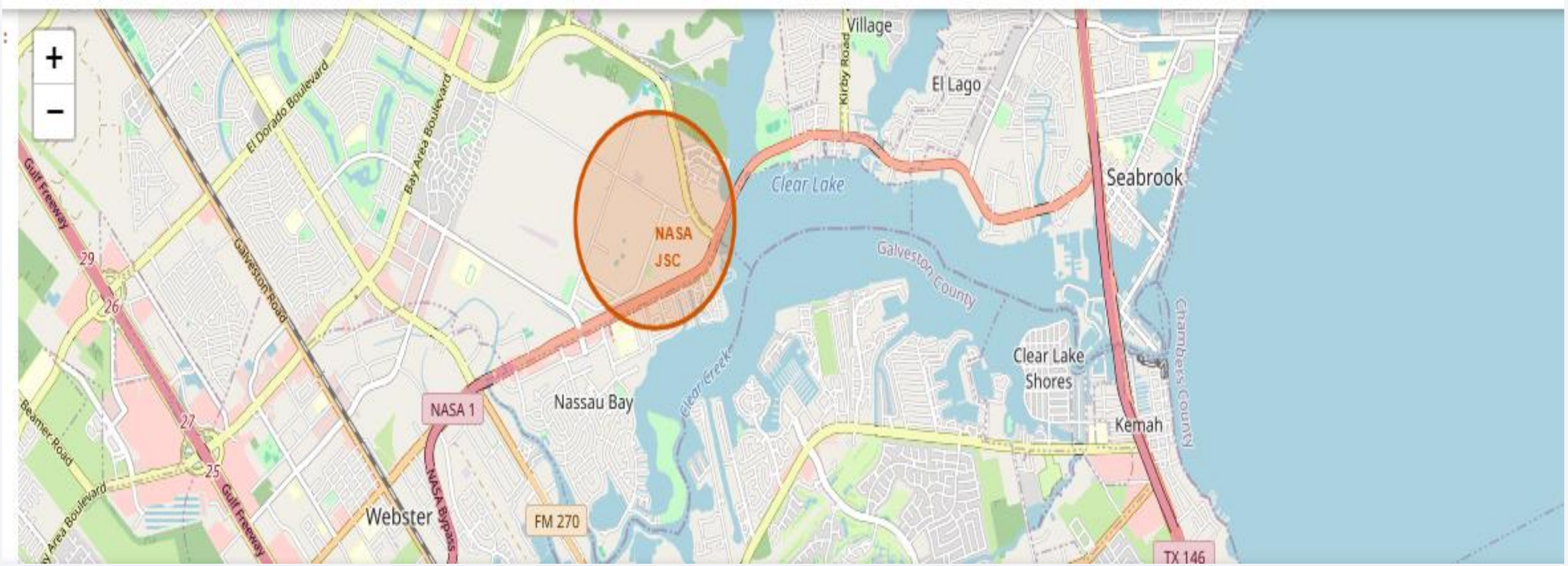
```
%sql SELECT COUNT(DISTINCT Landing_Outcome) As
counts , Date FROM SPACEXTABLE WHERE Date
BETWEEN '2010-06-04' AND '2017-03-20' group by
Date order by counts desc
```

Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

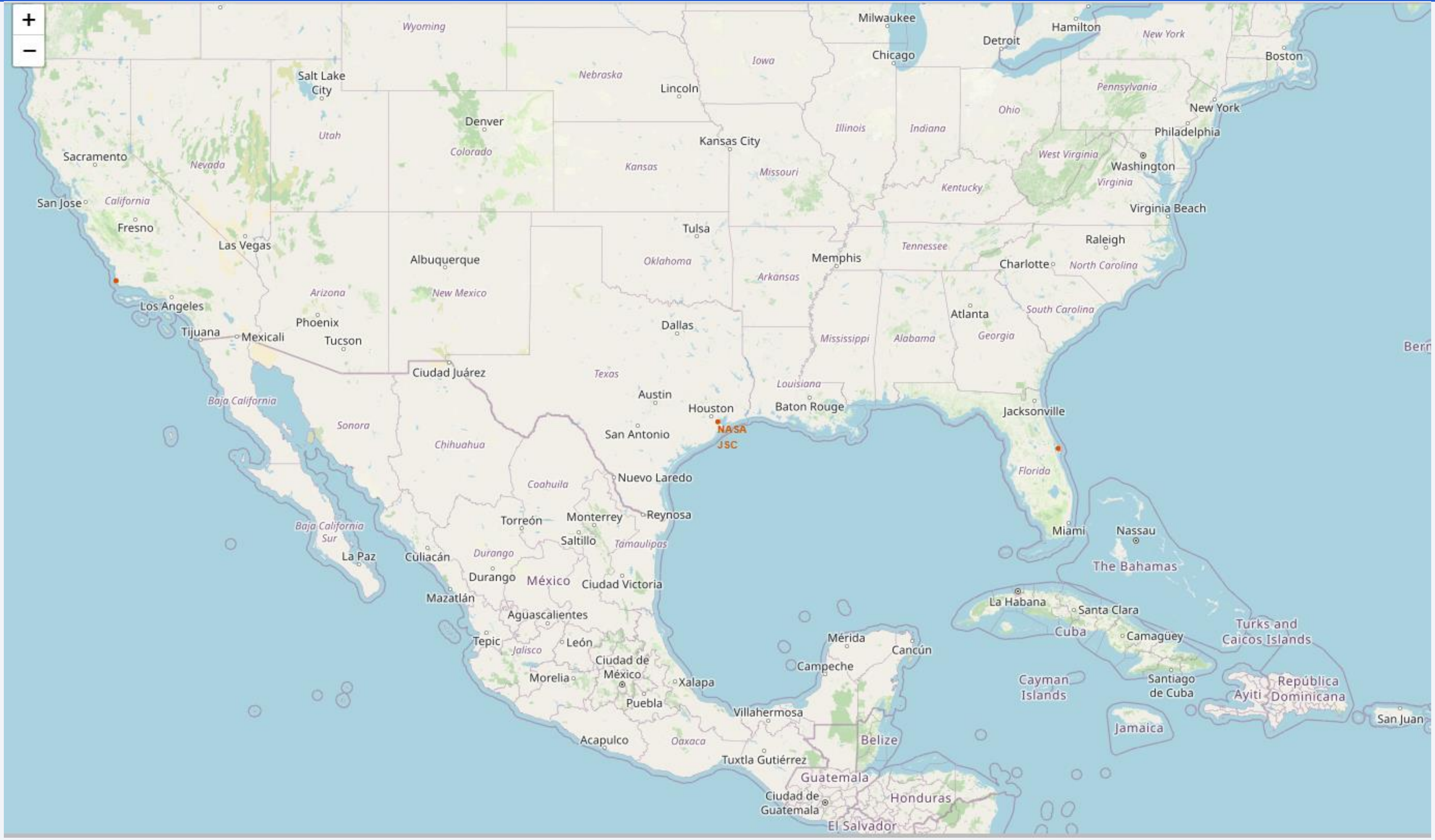
Section 3

Launch Sites Proximities Analysis

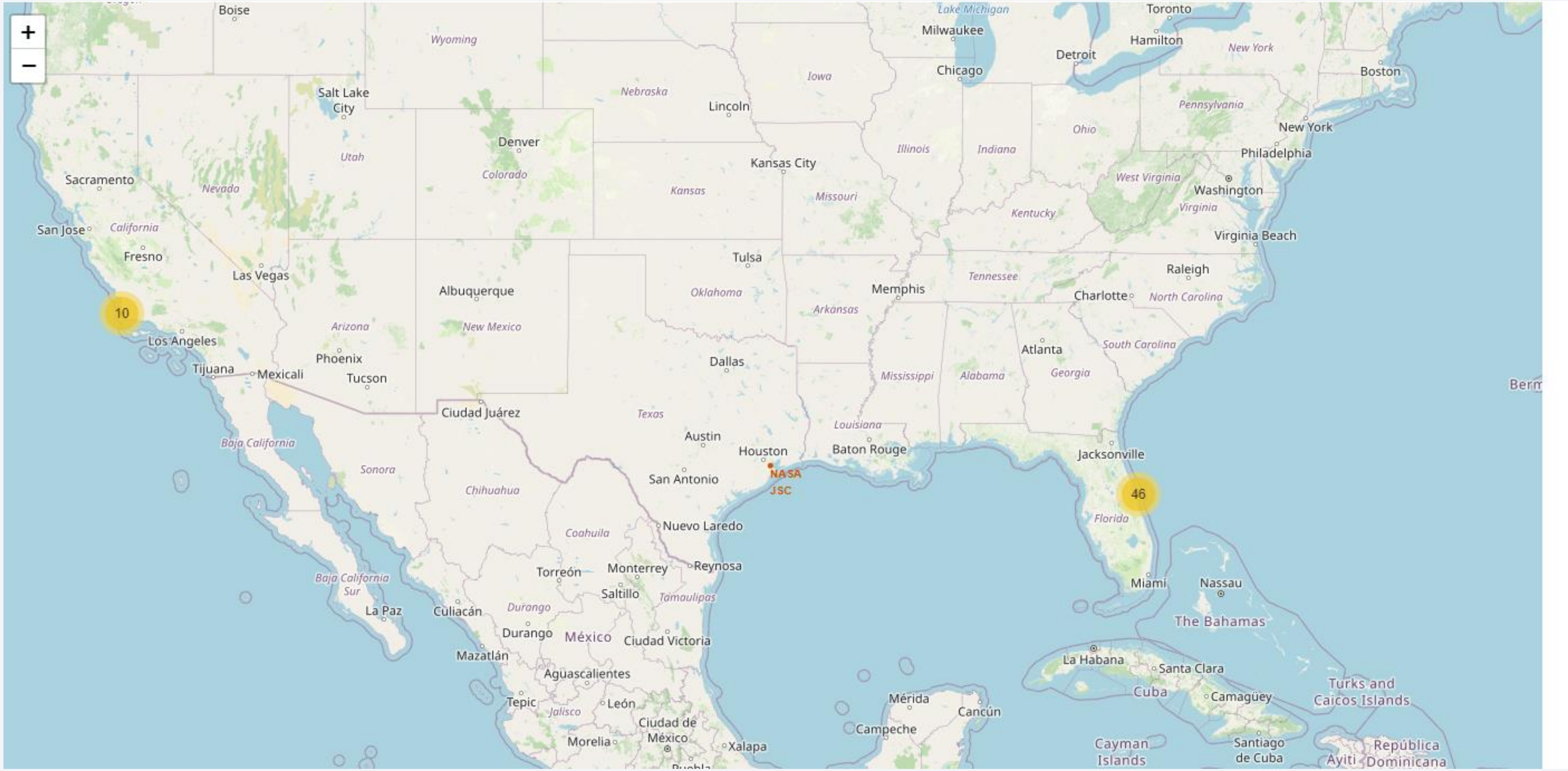
<Folium Map Screenshot 1>



<Folium Map Screenshot 2>



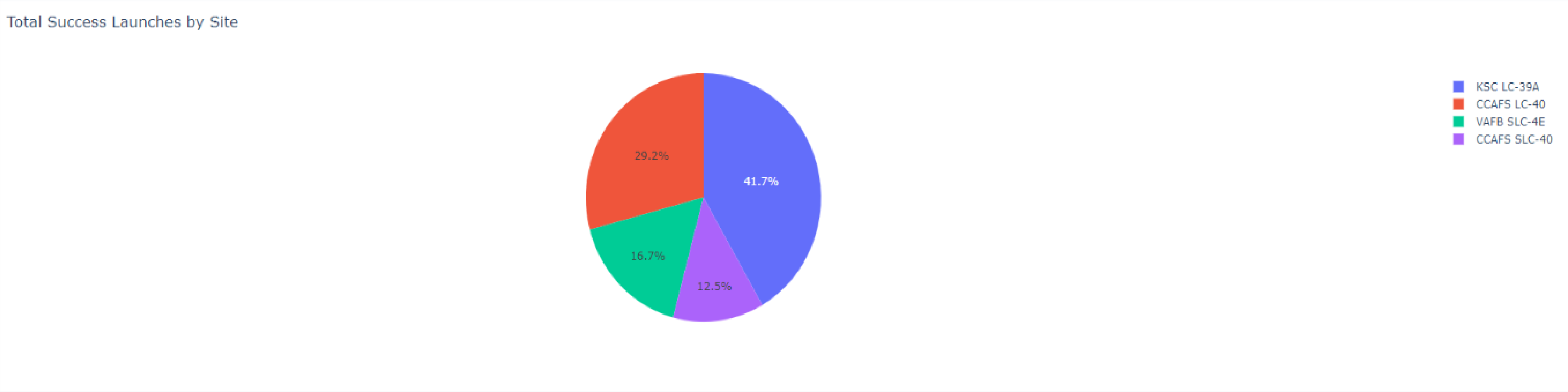
<Folium Map Screenshot 3>



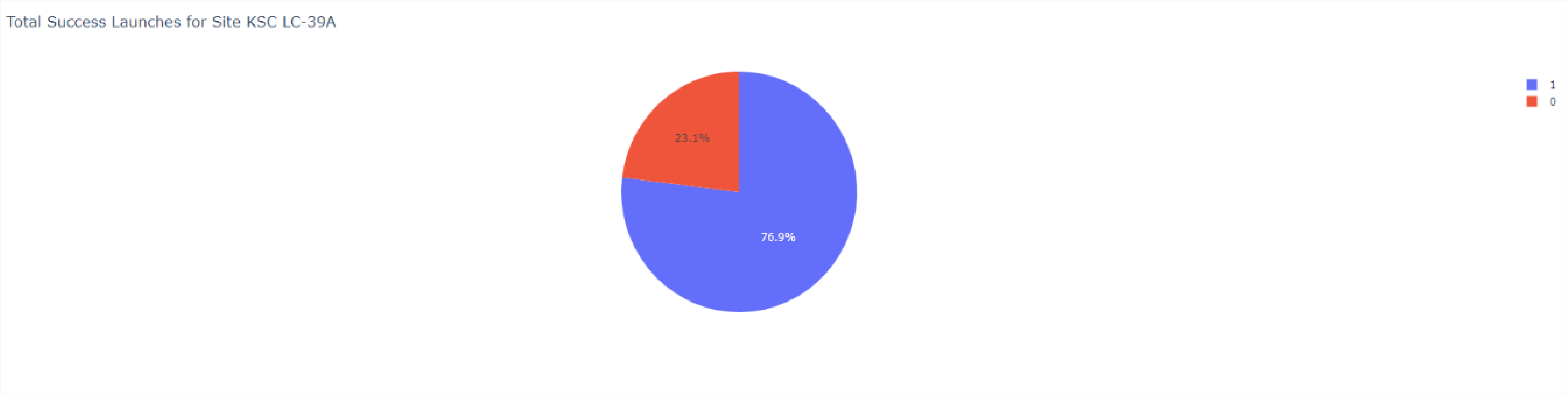
Section 4

Build a Dashboard with Plotly Dash

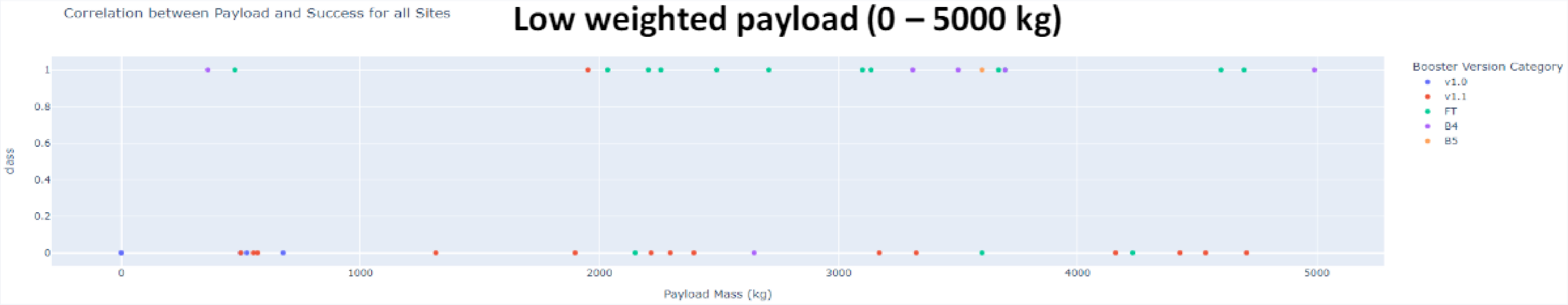
<Dashboard Screenshot 1>

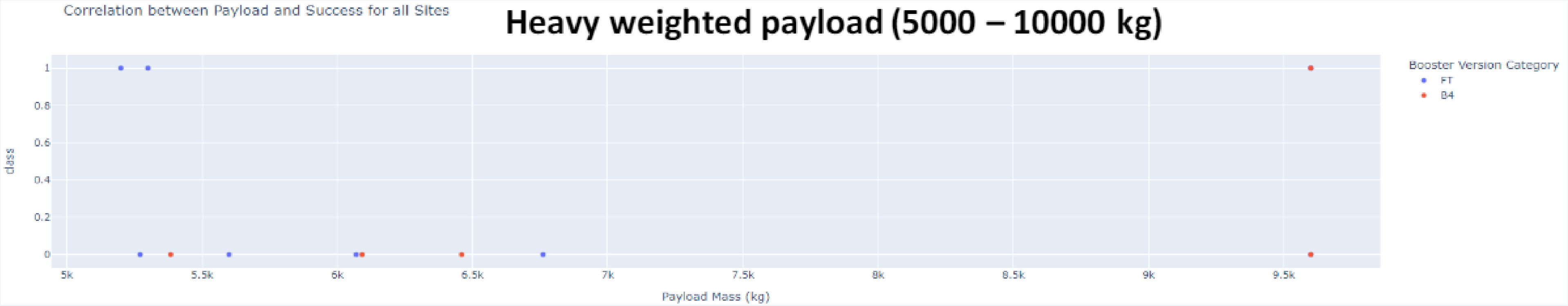


<Dashboard Screenshot 2>



<Dashboard Screenshot 3>



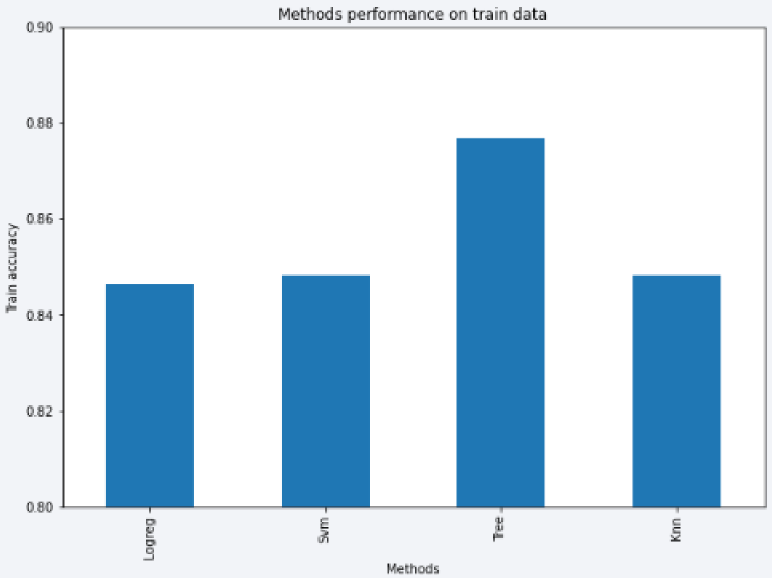


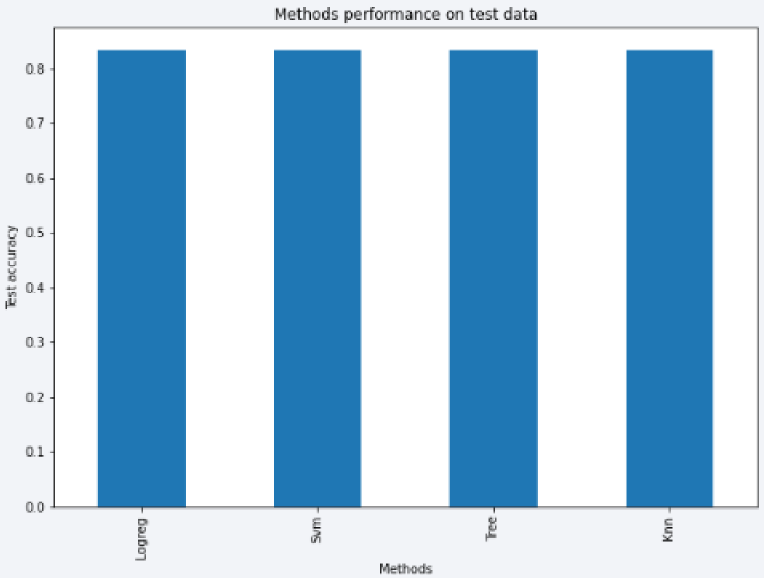
Section 5

Predictive Analysis (Classification)

Classification Accuracy

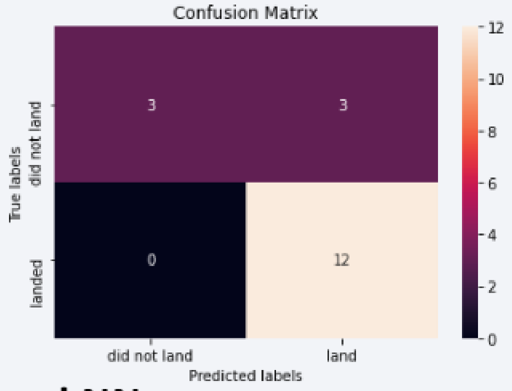
	Accuracy Train	Accuracy Test
Tree	0.876786	0.833333
Knn	0.848214	0.833333
Svm	0.848214	0.833333
Logreg	0.846429	0.833333





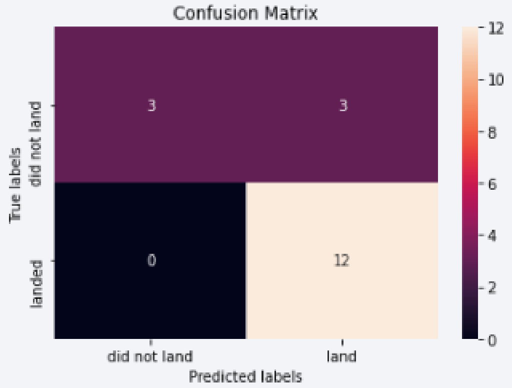
Confusion Matrix

Logistic regression

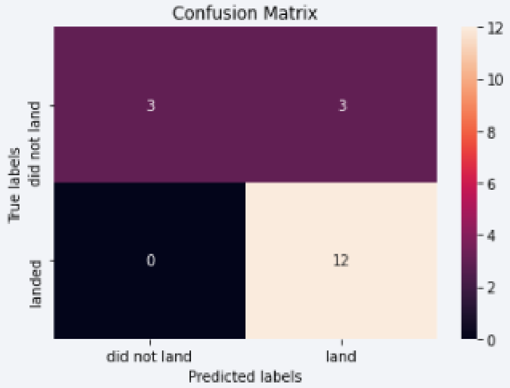


Decision Tree

kNN



SVM



Conclusions

The success of a mission can be explained by several factors such as the launch site, the orbit and especially the number of previous launches. Indeed, we can assume that there has been a gain in knowledge between launches that allowed to go from a launch failure to a success.

- The orbits with the best success rates are GEO, HEO, SSO, ES-L1.
- Depending on the orbits, the payload mass can be a criterion to take into account for the success of a mission. Some orbits require a light or heavy payload mass. But generally low weighted payloads perform better than the heavy weighted payloads.
- With the current data, we cannot explain why some launch sites are better than others (KSC LC-39A is the best launch site). To get an answer to this problem, we could obtain atmospheric or other relevant data.
- For this dataset, we choose the Decision Tree Algorithm as the best model even if the test accuracy between all the models used is identical. We choose Decision Tree Algorithm because it has a better train accuracy.

Thank you!

https://github.com/Shathabajes/FinalProject.git