IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Regina Temino Boes>
<12/08/2023>

Outline

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

Executive Summary

Summary of methodologies:

- Data collection
- Data wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)

Summary of all results:

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

Introduction

SpaceX's accomplishments include sending spacecraft to the International Space Station, starlink, a satellite internet constellation providing satellite Internet access, and sending manned missions to Space. One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch.

QUESTIONS
What are the main characteristics of a successful or failed landing ?
What are the effects of each relationship of the rocket variables on the success or failure of a landing ?
What are the conditions which will allow SpaceX to achieve the best landing success rate ?

Section 1

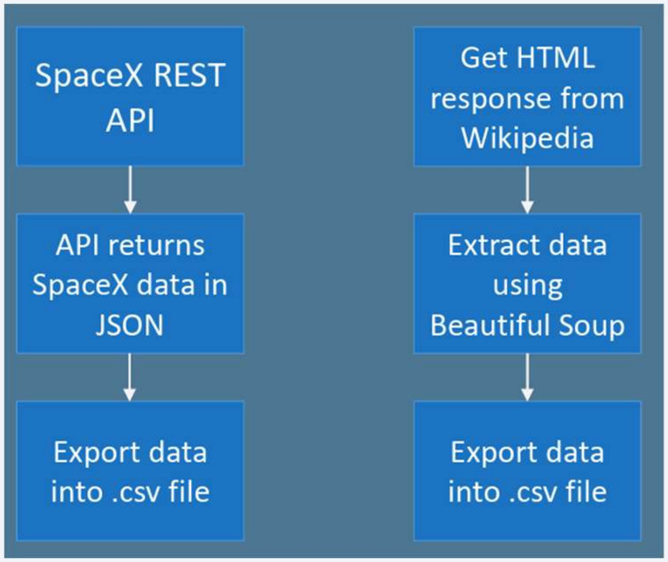
Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web scrapping
- Perform data wrangling
  - Drop unnecessary columns and missing values
  - 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
  - Build, tune, evaluate classification models

Data Collection



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)
```

GitHub URL of the completed SpaceX API calls notebook:

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()
```

See notebook for the rest of code

7. Create dataframe from dictionary

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

8. Export to file

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

GitHub URL of the completed web scraping notebook:

Data Wrangling

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

```
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)
```

GitHub URL of your completed data wrangling related notebooks:

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.

GitHub URL of your completed EDA with data visualization notebook:

EDA with SQL

- Queried the names of the launch sites
- Displayed records of launch sites that began with “CCA”
- Queried the total payload mass carried by boosters launched by NASA
- Displayed average payload mass carried by booster version F9 v1.1
- Found date when the first successful landing outcome in ground pad was achieved
- Queried the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Gathered the total number of successful and failure mission outcomes
- Queried names of the booster versions which have carried the maximum payload mass.
- Queried records that displayed the month names, failure landing outcomes in drone ship ,booster versions, launch site for the months in year 2015
- Ranked 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

GitHub URL of your completed EDA with SQL notebook:

Build an Interactive Map with Folium

Objects Used:

- Markers
    - Indicate launch sites
- Circles
    - Indicate launch areas (i.e Space Centers)
- Clusters
    - Indicate several launch sites within a specific area
- Lines
    - Indicate distance between these launches

GitHub URL of your completed interactive map with Folium map:

Build a Dashboard with Plotly Dash

- Visualizations surrounding percentage of launches per site and payload range
  - Plots
  - Graphs

- Along with being visually appealing and easy to analyze payload ranges and launch site usage, it is easy to draw conclusions such as best launch locations

GitHub URL of your completed Plotly Dash lab:

Predictive Analysis (Classification)

Four classification methods were used and compared in this project

- Logistic regression
- K-nearest-neighbors
- Decision tree
- Support vector machine

GitHub URL of your completed predictive analysis lab:

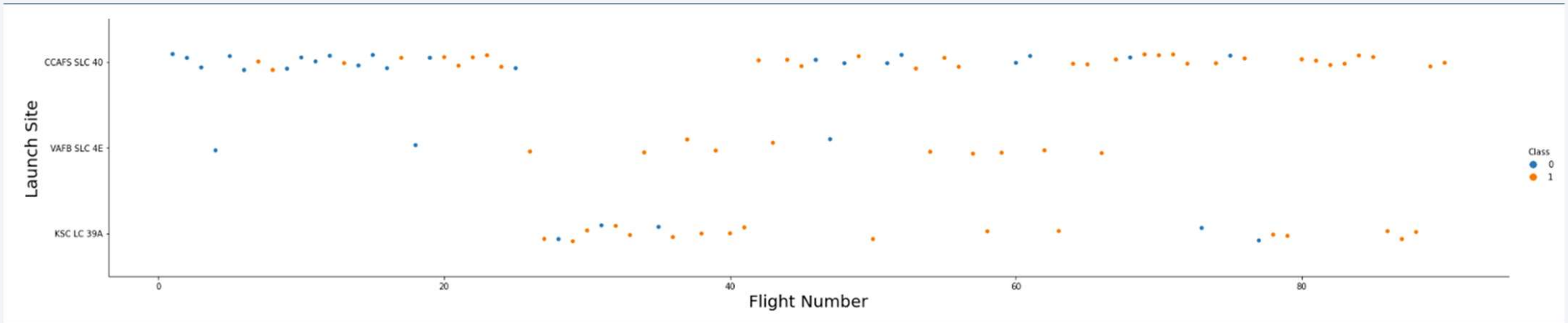
Results

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

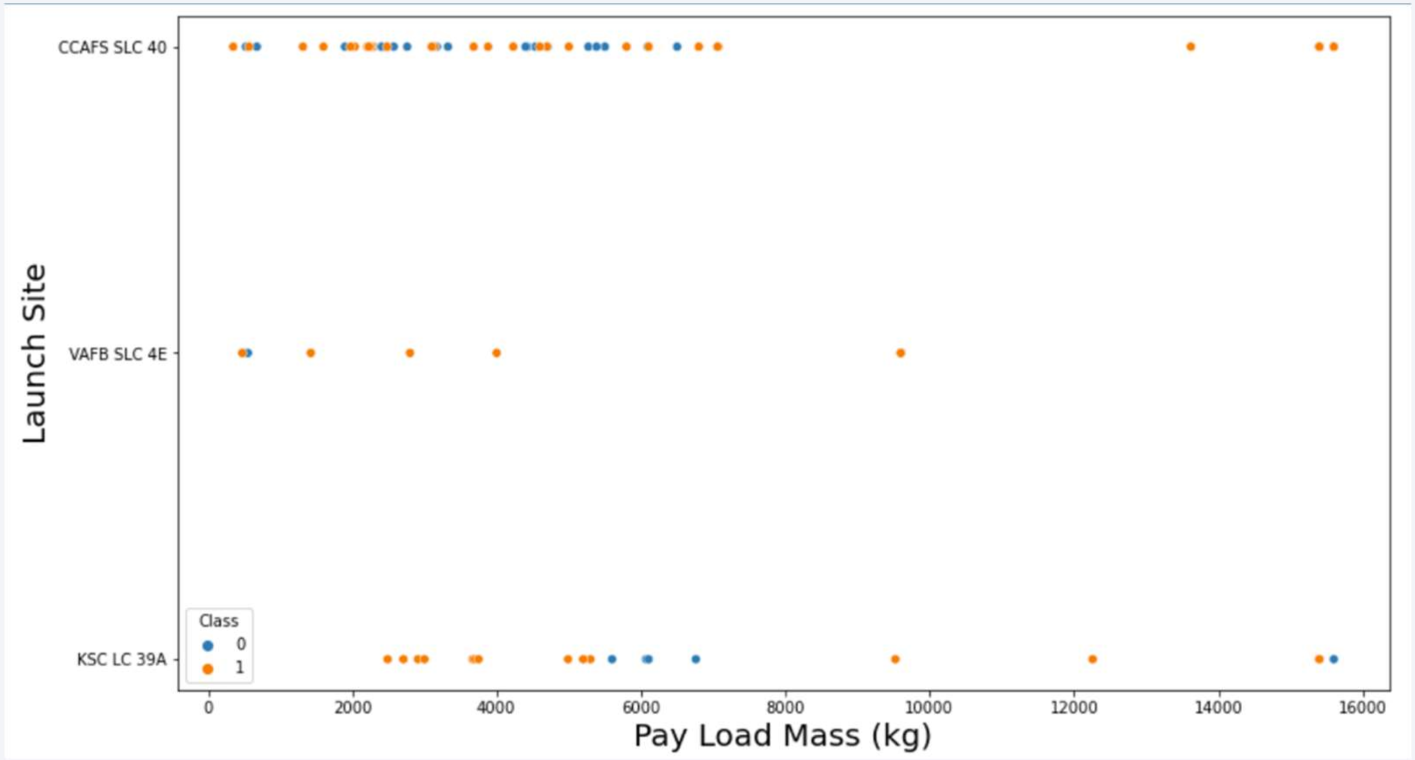
Section 2

Insights drawn from EDA

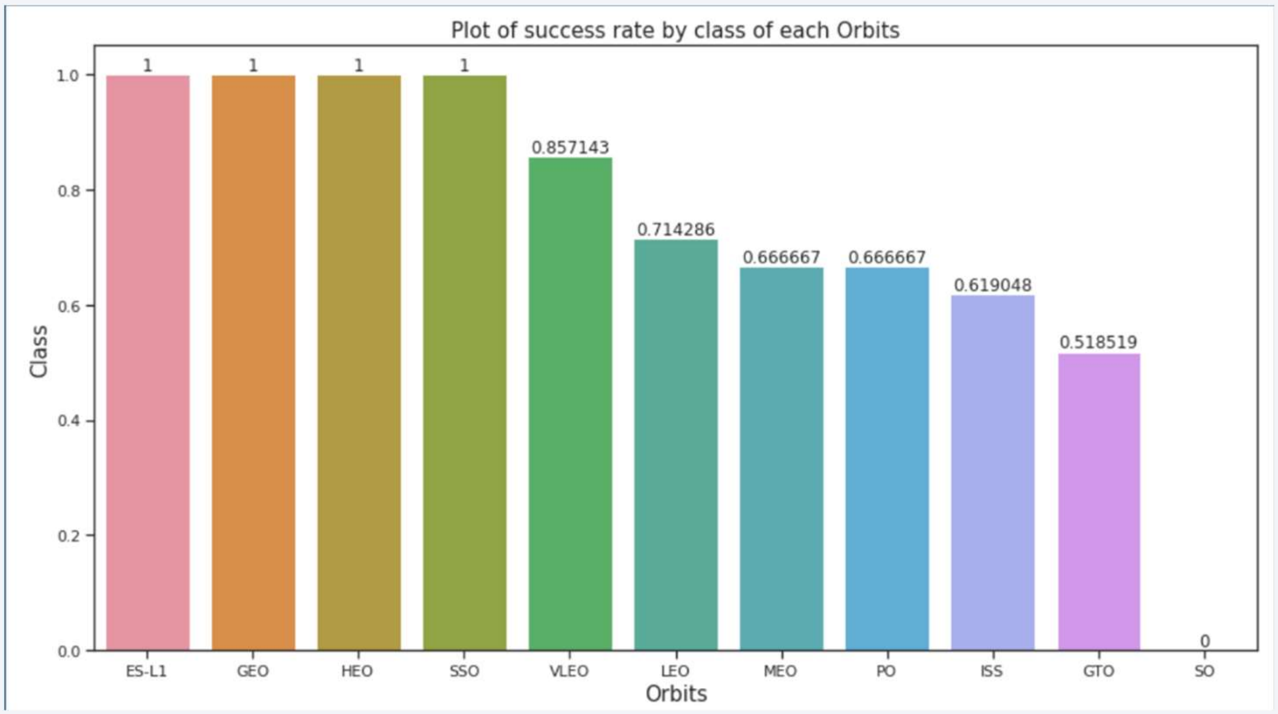
Flight Number vs. Launch Site



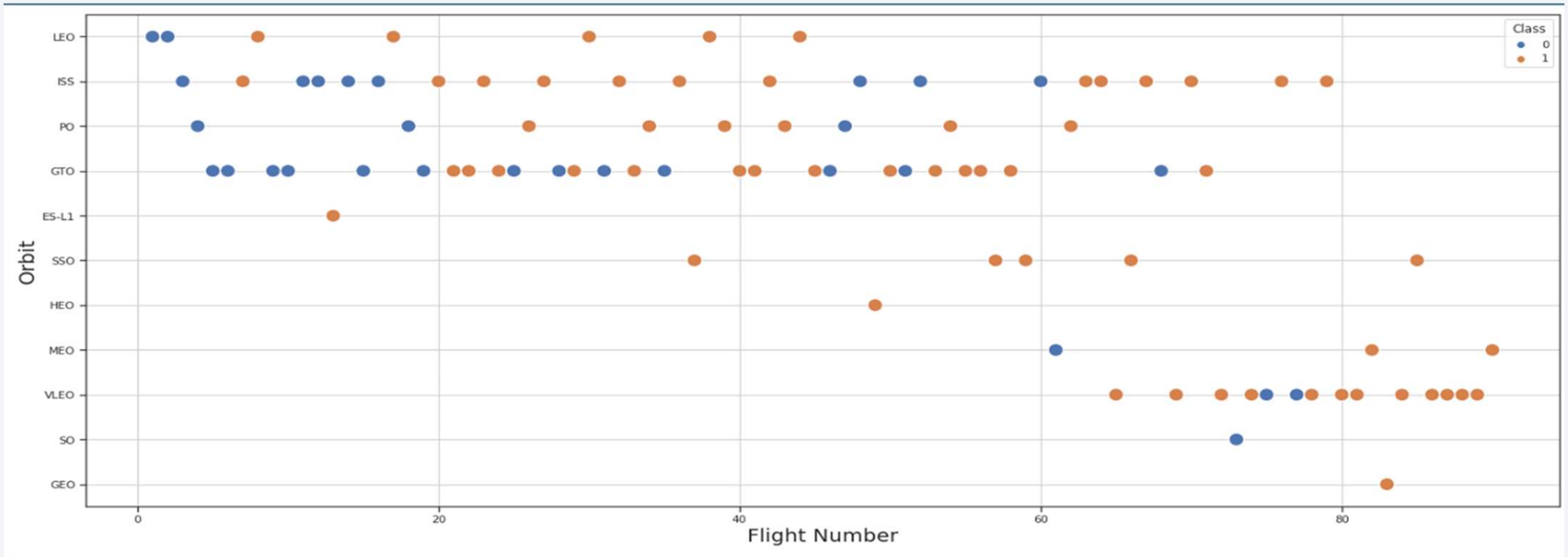
Payload vs. Launch Site



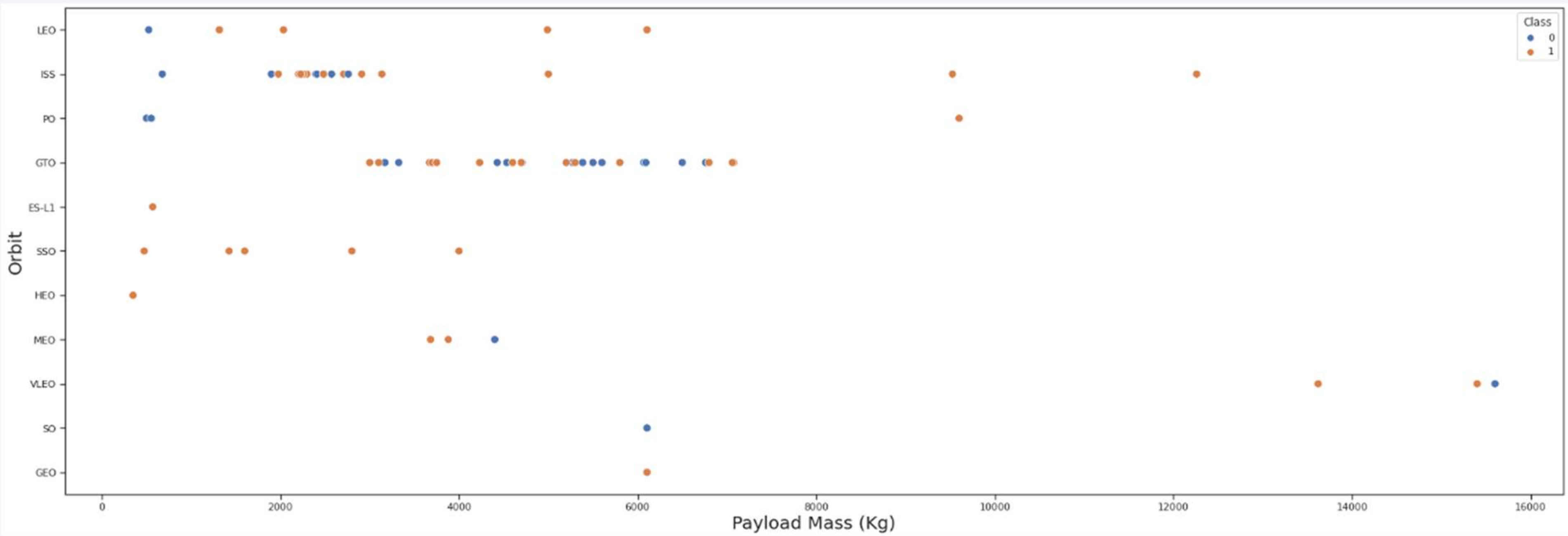
Success Rate vs. Orbit Type



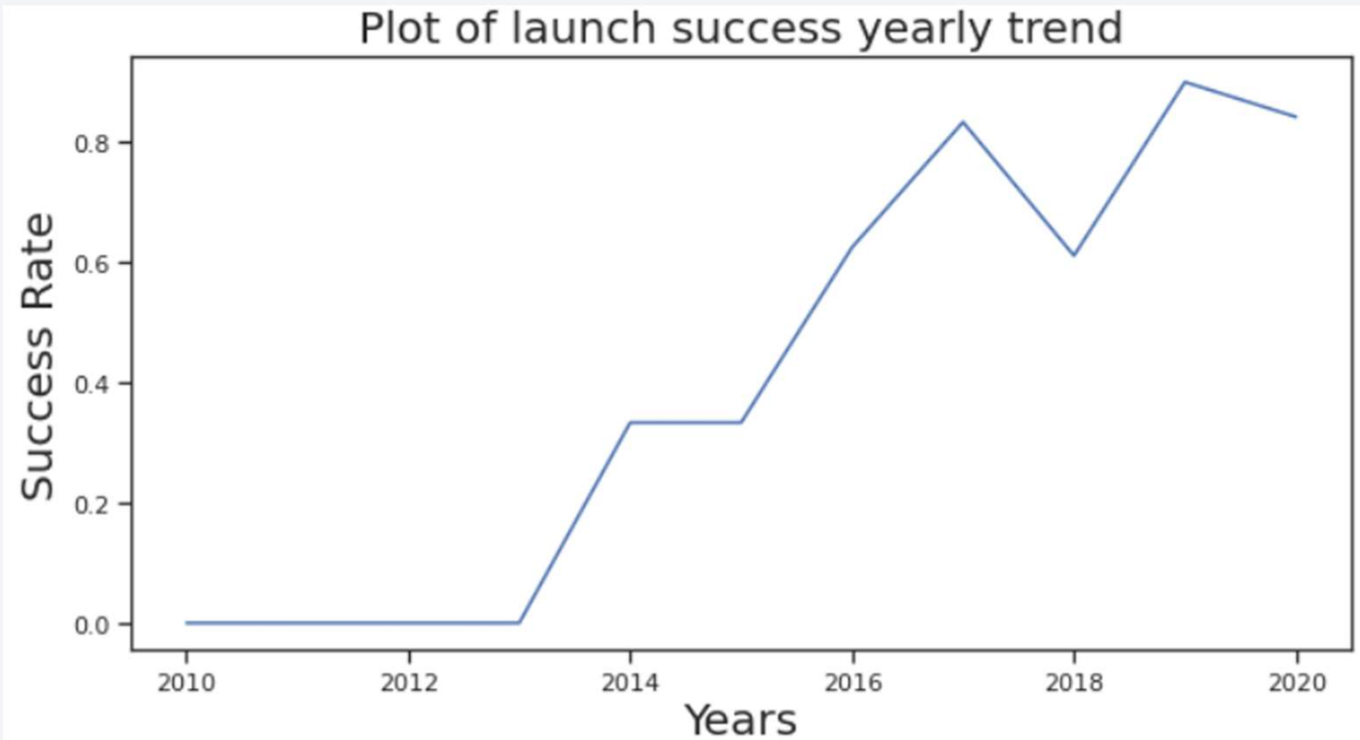
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

The DISTINCT statement used in this query ensures that the query only returns unique launch site names from the launch_site column.

```
SELECT DISTINCT "LAUNCH_SITE" FROM SPACEXTBL
```

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

Launch Site Names Begin with 'CCA'

This query used the LIKE operator to find the top 5 records for the launch site name with a string that started with ‘CCA’.

```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

This query used the SUM function to total the amount of payload mass from the column payload_mass_kg_ launched by the customer ‘NASA (CRS)’.

```
%sql SELECT SUM(payload_mass__kg_) AS "TOTAL PAYLOAD MASS BY NASA(CRS)" FROM SPACEXTBL WHERE customer = 'NASA (CRS)';
```

TOTAL PAYLOAD MASS BY NASA(CRS)
45596

Average Payload Mass by F9 v1.1

This query used the AVG function to retrieve the average payload mass from column payload_mass_kg_ that is carried by the booster ‘F9 v1.1’.

```
%sql SELECT AVG(payload_mass__kg_) AS "average payload mass carried by booster version F9 v1.1"  FROM SPACEXTBL WHERE booster_version like 'F9 v1.1';
```

average payload mass carried by booster version F9 v1.1
2928

First Successful Ground Landing Date

The query used the MIN function on the launch date to retrieve the first date of which the condition of the “Landing_Outcome” is ‘success (ground pad)’.

```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEXTBL WHERE "Landing _Outcome" = 'Success (ground pad)';
```

First Succesful Landing Outcome in Ground Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

This query was used to retrieve the booster version that successfully landed on a drone ship and also carried a payload mass between 4000kg and 6000kg.

```
%%sql
SELECT BOOSTER_VERSION FROM SPACEXTBL
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

This query is used to sum up, the amount of success and failure missions separately by using the LIKE operator on the string in mission_outcome column that consists of ‘success’ or ‘failure’.

```
%%sql
SELECT SUM(case when mission_outcome LIKE '%Success%' then 1 else 0 end) AS "Successful Mission",sum(case when mission_outcome LIKE '%Failure%' then 1
```

Successful Mission	Failure Mission
100	1

Boosters Carried Maximum Payload

This query was used to retrieve distinct booster versions and payload mass with the condition that the booster carried the maximum payload mass. This query also used sub-query as its condition to find the maximum payload mass.

```
%%sql
SELECT DISTINCT booster_version, payload_mass__kg_ FROM SPACEXTBL
WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL);
```

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

2015 Launch Records

This query retrieved the date, landing outcome, booster version, and launch site for launch in the year 2015 and has an outcome of ‘Failure (drone ship)’

```
%sql SELECT DATE, "Landing _Outcome",BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE DATE LIKE '2015-%' AND "Landing _Outcome" LIKE 'Failure (drone ship)';
```

DATE	Landing _Outcome	booster_version	launch_site
2015-01-10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

This query was used to retrieve the number of launches that resulted in each landing outcome between 2010-06-04 and 2017-03-20. The GROUP BY statement is to group the count of each landing_outcome, and then ORDER BY was used to rank the landing_outcome columns based on the number of each total launch in descending order.

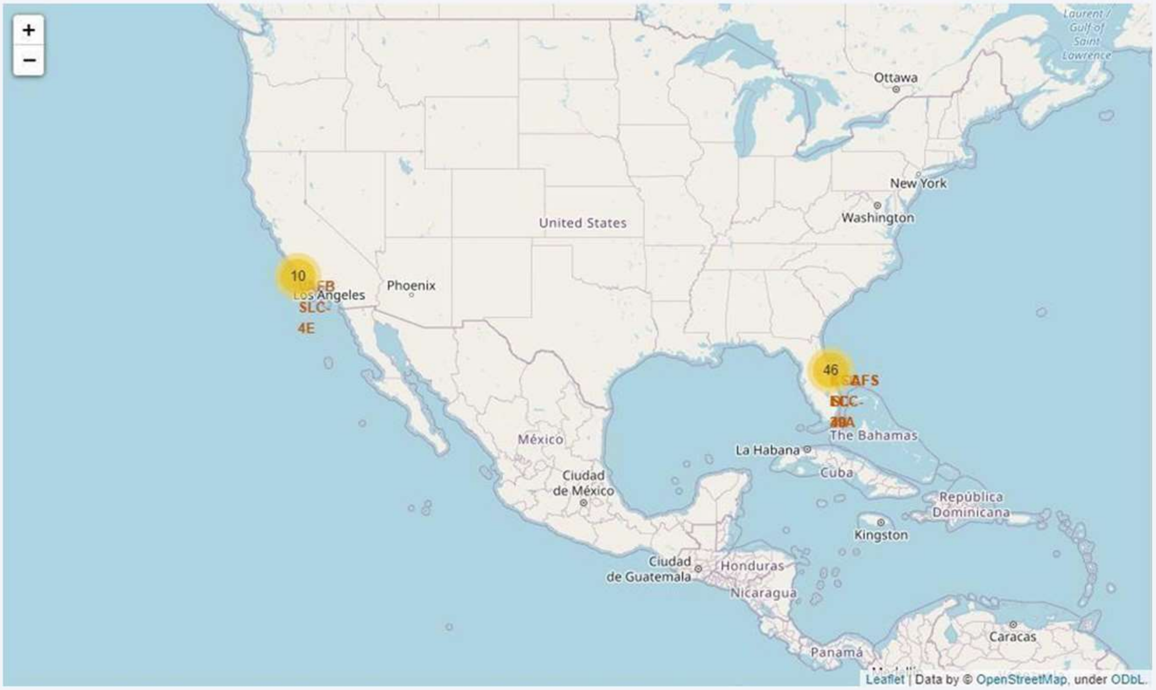
```
%%sql
SELECT "Landing _Outcome", COUNT("Landing _Outcome") AS "TOTAL LAUNCH" FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing _Outcome"
ORDER BY COUNT("Landing _Outcome") DESC
```

Landing _Outcome	TOTAL LAUNCH
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

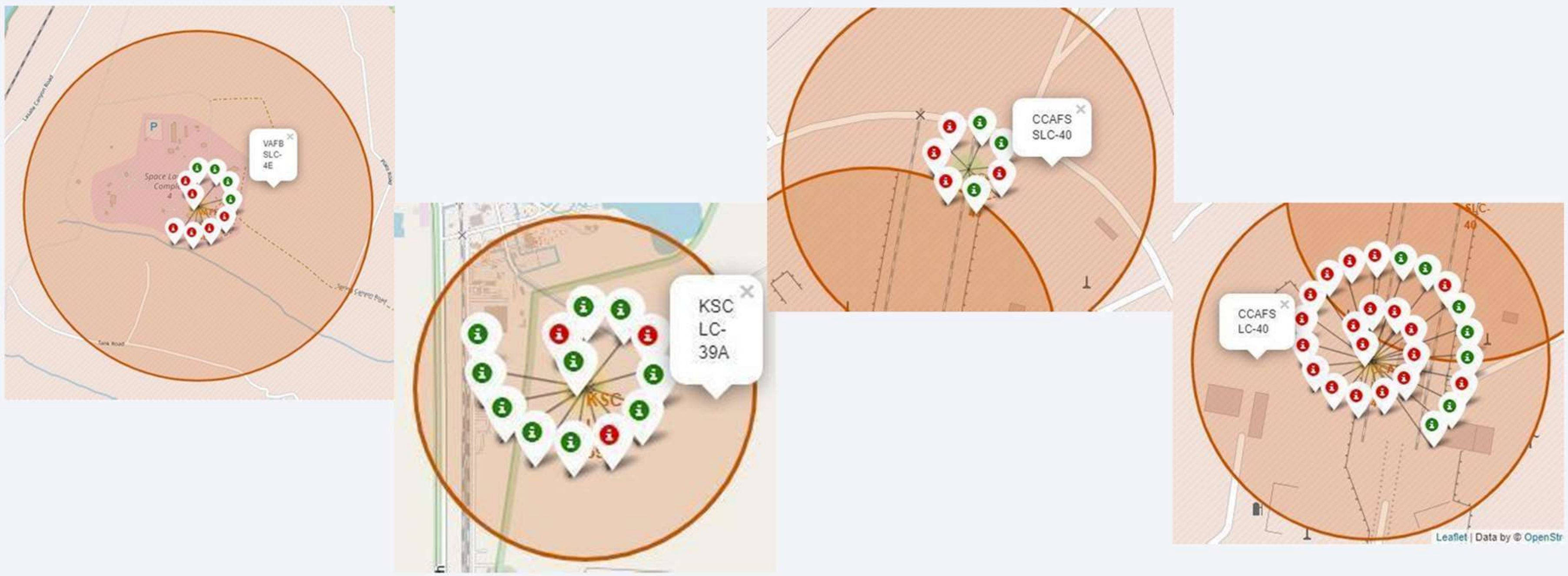
Section 3

Launch Sites Proximities Analysis

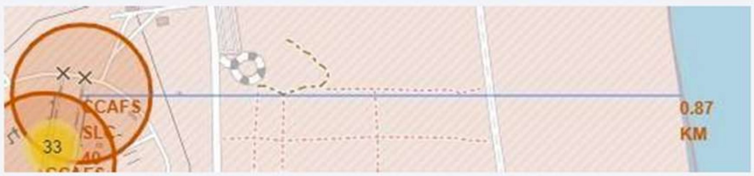
Folium Map - Stations

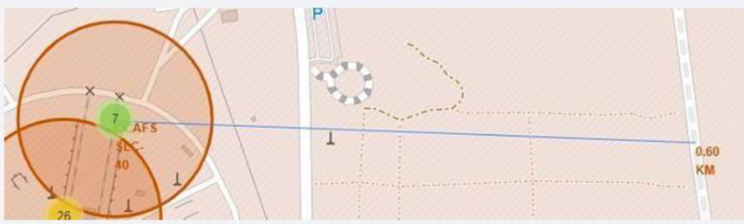


Folium Map – Labeled Markers



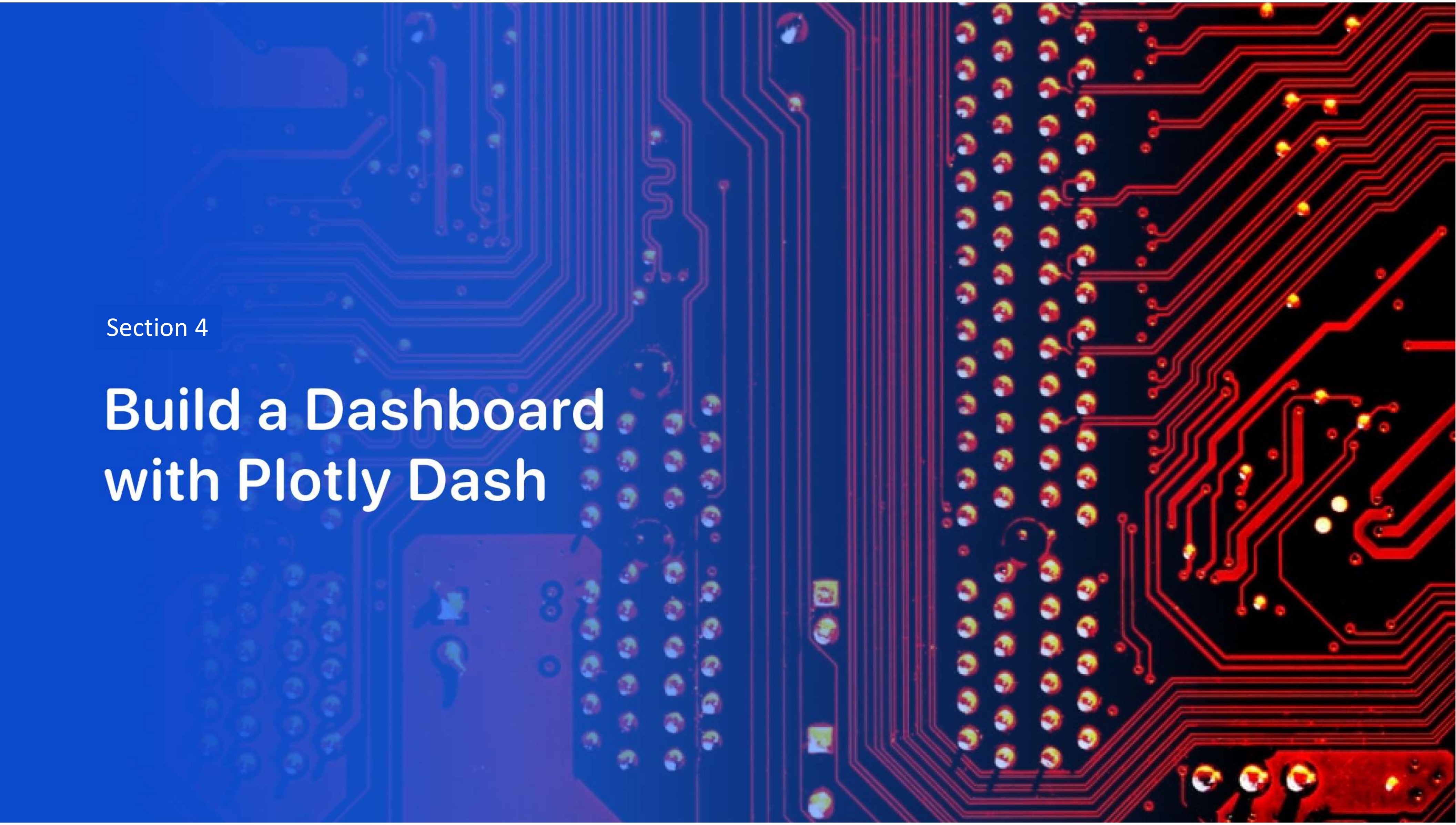
Folium Map - Distances



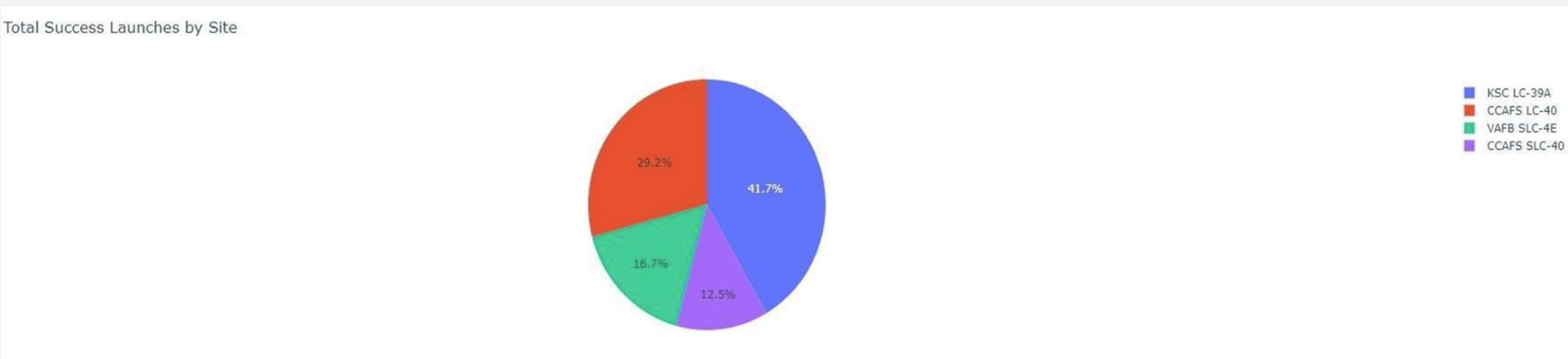


Section 4

Build a Dashboard with Plotly Dash



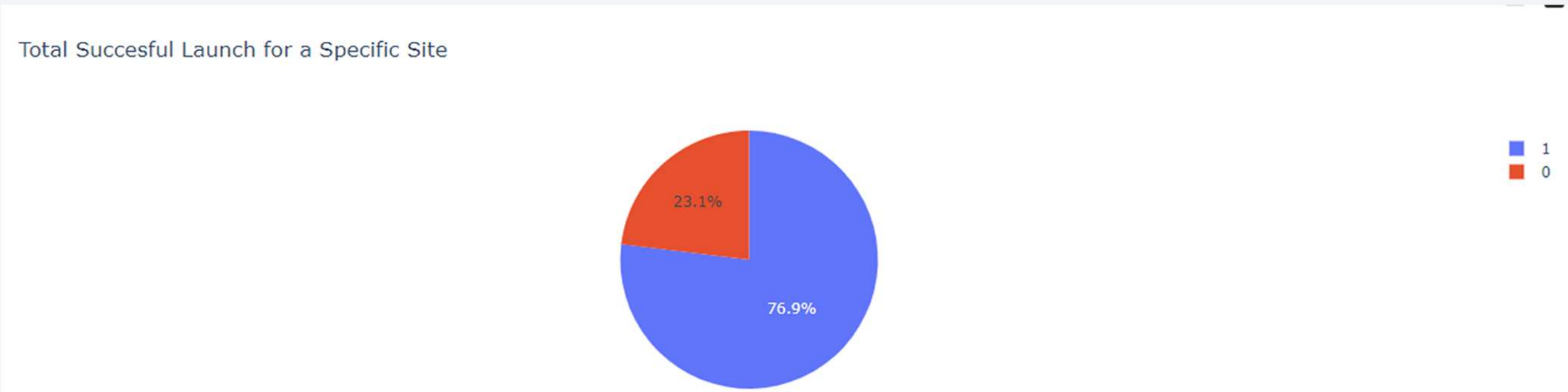
Success Percentages for Launch Sites



Kennedy Space Center Launch Complex 39A (KSC LC-39A) had the highest share of a successful landing.

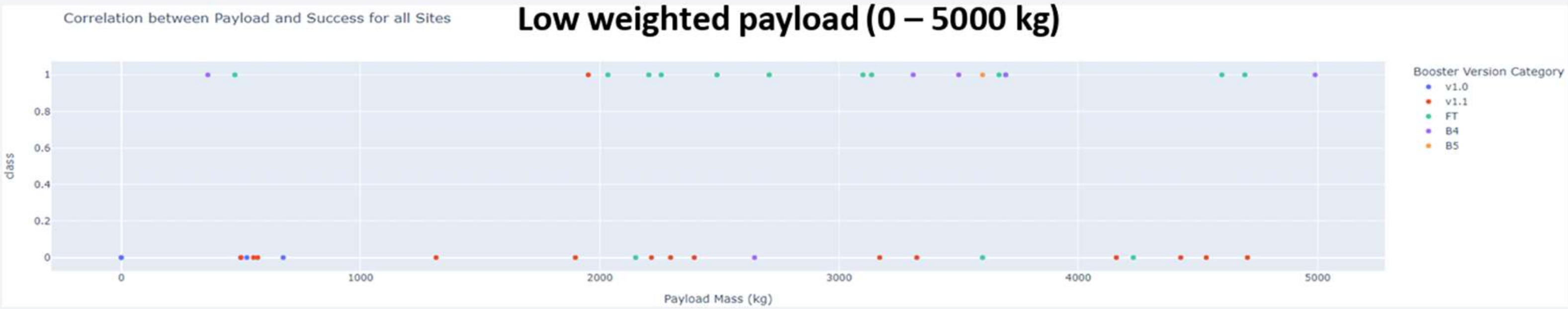
Kennedy Space Center Launch

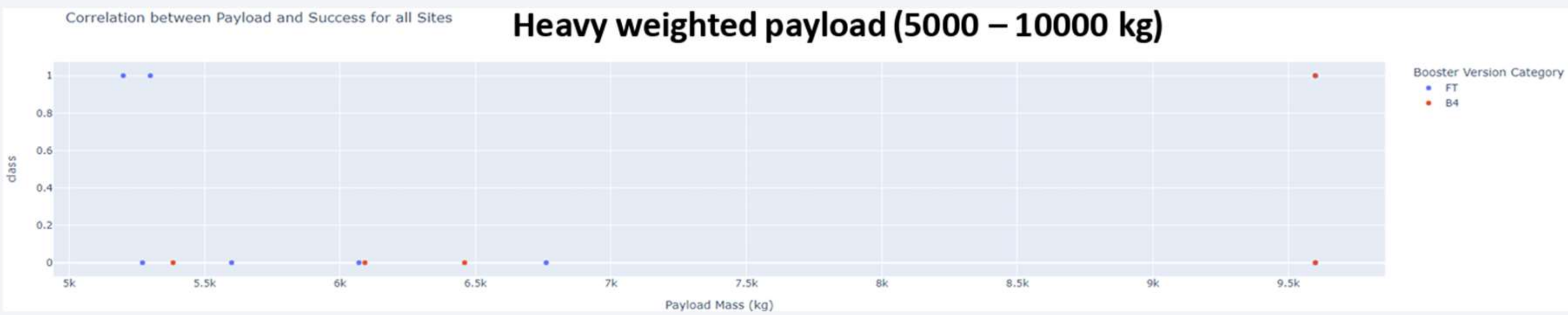
Kennedy Space Center Launch Complex 39A (KSC LC-39A)



Kennedy Space Center Launch Complex 39A (KSC LC-39A) managed to achieve almost 77% of success (class 1) and 23% of landing (class 0) for Falcon 9 first-stage landing.

<Dashboard Screenshot 3>





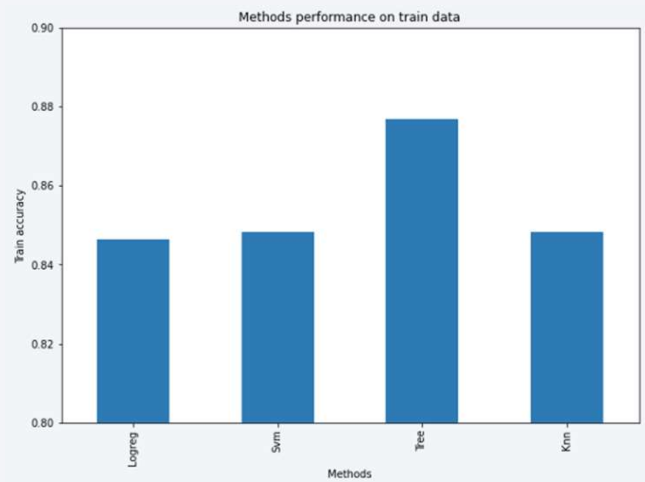
Low weighted payloads have a better success rate than the heavy weighted payloads.

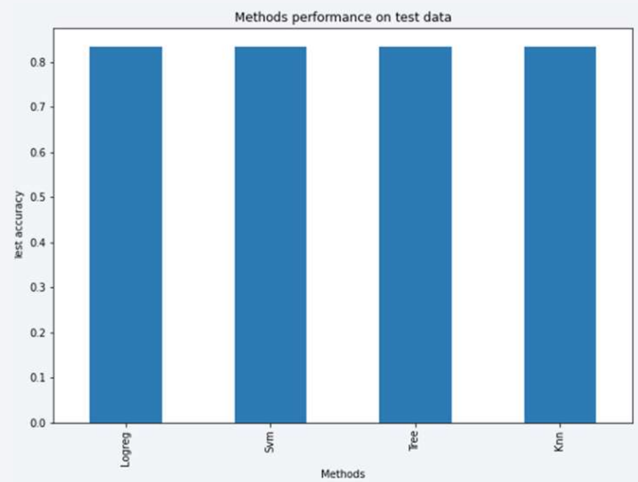
Section 5

Predictive Analysis (Classification)

Classification Accuracy

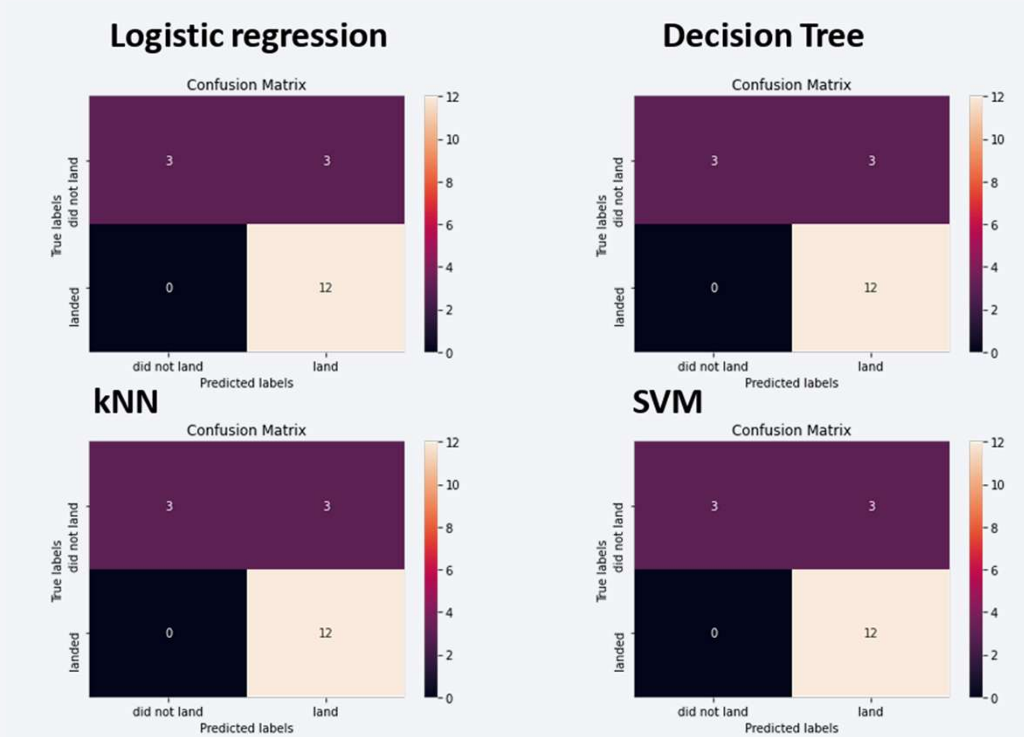
	Accuracy Train	Accuracy Test
Tree	0.876786	0.833333
Knn	0.848214	0.833333
Svm	0.848214	0.833333
Logreg	0.846429	0.833333





For accuracy test, all methods performed similar. We could get more test data to decide between them.

Confusion Matrix



Conclusions

- Under the default test_size value given for this project, all tested algorithms performed similarly for both test accuracy score and confusion matrix output.
- When using test_size 23%, the decision tree algorithm got the highest accuracy and increased its performance by 14% compared to the model using the default test size value.
- FT boosters are the best booster version for payload between 2k-4k Kg
- Kennedy Space Center Launch Complex 39A (KSC LC-39A) had the highest share of successful landing
- For the orbit destination, ES-L1, GEO, HEO, and SSO have the best success rate.


Thank you!