IBM Developer
SKILLS NETWORK

IBM Data Science Capstone Project

Space X Falcon 9 Landing Prediction

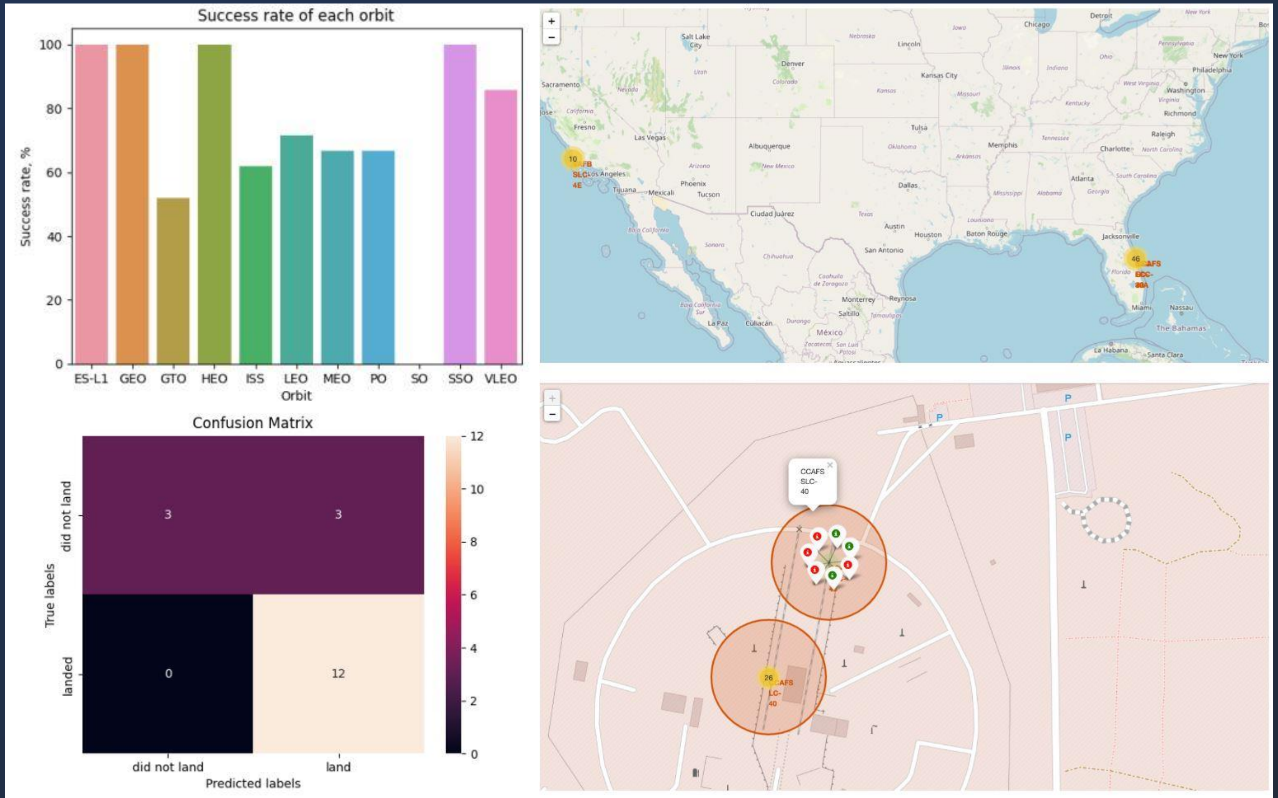
Jeffrey Hsieh

Outline

Executive Summary

IBM Developer
SKILLS NETWORK

- **Summary of Methodologies:**
  1. Data Collection
  2. Data Wrangling
  3. Exploratory Data Analysis
  4. Interactive Visual Analytics
  5. Predictive Analysis (Classification)



- **Summary of Results:**
  1. Exploratory Data Analysis (EDA) results
  2. Geospatial analytics
  3. Interactive dashboard
  4. Predictive analysis of classification models

Introduction

IBM Developer SKILLS NETWORK

- Project background:

  SpaceX 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 SpaceX can reuse the first stage.

- Problems Statement:

  Predict if the first stage will land successfully as well as the cost of a launch in order to bid against SpaceX for a rocket launch.

Methodology

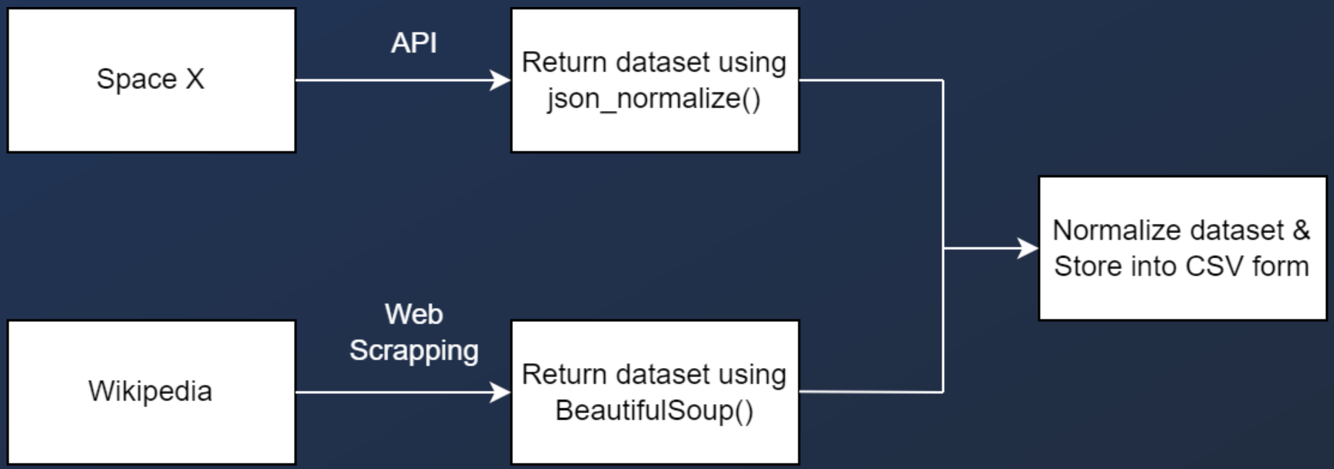
IBM Developer SKILLS NETWORK

1. Data Collection:
   - SpaceX API
   - Web Scrapping from Wikipedia
2. Data Wrangling:
   - Data cleaning, e.g., replacing null value with mean or median
   - One Hot Encoding
3. Exploratory Data Analysis (EDA) using Visualization and SQL
4. Interactive Visual Analytics using Folium and Plotly Dash
5. Predictive Analysis using Classification Models
   - Model comparison : Linear Regression, Decision Tress, K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machine (SVM)
   - Model evaluation : R2, RMSE, Accuracy Score, F1-Score, Jaccard Index, Log-Loss-Loss

Data Collection

IBM Developer SKILLS NETWORK

- The data sets were collected through below two methods:
  - SpaceX API
  - Web Scrapping from Wikipedia
- Data collection process:



Data Collection – SpaceX API

IBM Developer SKILLS NETWORK

Data collection with SpaceX REST calls

1. Request rocket launch data from SpaceX API
2. Request and parse SpaceX launch data with a GET request and turning the JSON into a Pandas dataframe
3. Use the functions created for using the API to extract information
4. Construct dataset from data obtained from API and GET functions into a pandas dictionary with updating columns and rows.
5. Replace null values with mean and export to CSV file.

[Click here for more detail and code.]

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

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

```
response.status_code
```

```
200
```

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

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

Using the dataframe `data` print the first 5 rows

```
data.head()
```

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown	[]	[]	[]	[5eb0e4b6b6

```
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(data_falcon9['PayloadMass'].mean())
data_falcon9.isnull().sum()
```

```
FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

IBM Developer SKILLS NETWORK

Data Collection – Web Scraping

Data collection with web scraping process

1. Request Falcon9 Launch HTML page HTTP method
2. Create BeautifulSoup object
3. Extract column name by iterate through each <th> element from HTML table header.
4. Fill launch dictionary with launch records extracted from table rows with for loop to append into.
5. Create dataframe from launch_dict with parsed launch values now added and export to CSV file.

Click here for more detail and code.

```
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)
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
```

```
soup = BeautifulSoup(response.text, 'html.parser')
```

```
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(extract_column_from_header(th))
```

Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

```
extracted_row = 0
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        row=rows.find_all('td')
        if flag:
            extracted_row += 1
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            time = datatimelist[1]
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)
            launch_site = row[2].a.string
            payload = row[3].a.string
            payload_mass = get_mass(row[4])
            orbit = row[5].a.string
            customer = row[6].a.string
            launch_outcome = list(row[7].strings)[0]
            booster_landing = landing_status(row[8])
```

Data Wrangling

IBM Developer SKILLS NETWORK

1. Load dataset from last section with read_csv function
2. Calculate the number of column on each site with value_counts saperately.
3. Use for loop to assign numbers to landing_outcomes, and list the failed landing outcomes
4. Present list of outcomes with Class assigned as '0' or '1' for failure or success, respectively and export to CSV file.

Click here for more detail and code.

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

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

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

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

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

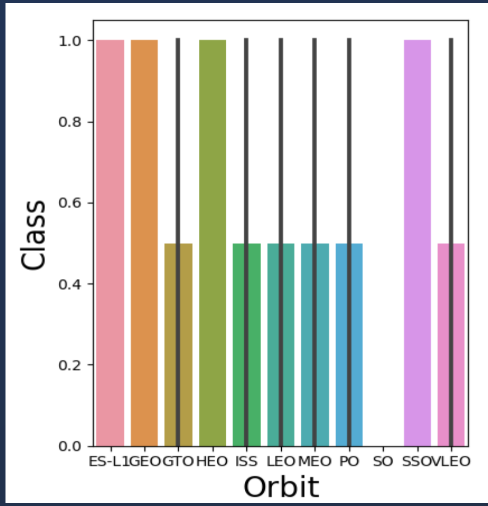
```
landing_class =[]
for i in df['Outcome']:
    if i in set(bad_outcomes):
        landing_class.append(0)
    else:
        landing_class.append(1)
```

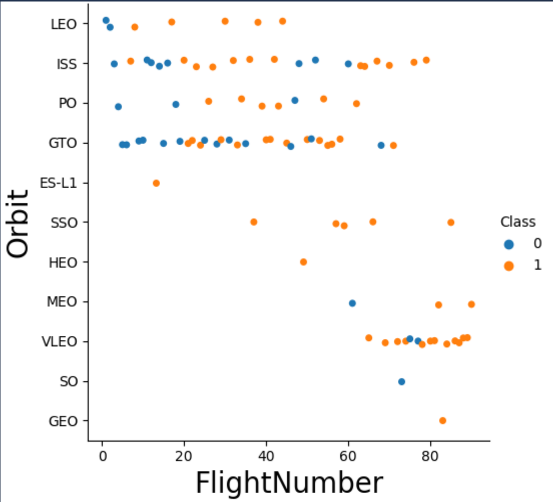
```
df['Class']=landing_class
df[['Class']].head(8)
```

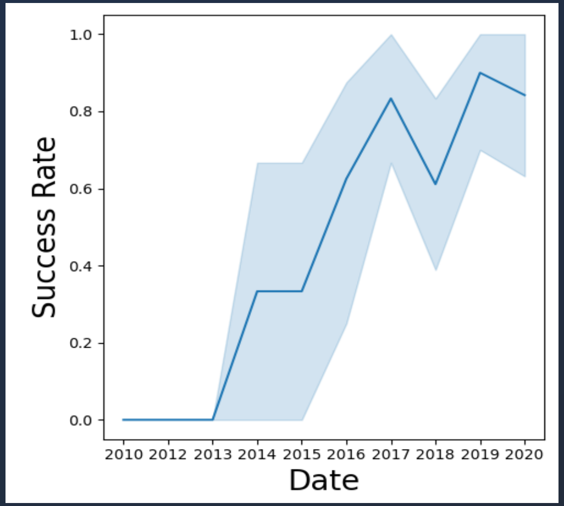
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

EDA with Data Visualization

1. Visualize the relationship between success rate of each orbit type
2. Visualize the relationship between FlightNumber and Orbit type
3. Visualize the launch success yearly trend
4. Create dummy variables to categorical columns







[Click here for more detail and code.]

EDA with SQL

IBM Developer SKILLS NETWORK

The SQL queries performed on the data set were used to:

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string ‘CCA’
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display the average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome on a ground pad was achieved
6. List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
7. List the total number of successful and failed mission outcomes
8. List the names of the booster versions which have carried the maximum payload mass
9. List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
10. 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

Click here for more detail and code.

Build an Interactive Map with Folium

IBM Developer SKILLS NETWORK

1. Mark all launch sites on a map
   - Initialise the map using a Folium Map object
   - Add a folium.Circle and folium.Marker for each launch site on the launch map
2. Mark the success/failed launches for each site on a map
   - As many launches have the same coordinates, it makes sense to cluster them together.
   - Before clustering them, assign a marker colour of successful (class = 1) as green, and failed (class = 0) as red.
   - To put the launches into clusters, for each launch, add a folium.Marker to the MarkerCluster() object.
   - Create an icon as a text label, assigning the icon_color as the marker_colour determined previously.
3. Calculate the distances between a launch site to its proximities
   - To explore the proximities of launch sites, calculations of distances between points can be made using the Lat and Long values.
   - After marking a point using the Lat and Long values, create a folium.Marker object to show the distance.
   - To display the distance line between two points, draw a folium.PolyLine and add this to the map.

Click here for more detail and code.

Build a Dashboard with Plotly Dash

1. Pie chart (px.pie()) showing the total successful launches per site
   - This makes it clear to see which sites are most successful
   - The chart could also be filtered (using a dcc.Dropdown() object) to see the success/failure ratio for an individual site
2. Scatter graph (px.scatter()) to show the correlation between outcome (success or not) and payload mass (kg)
   - This could be filtered (using a RangeSlider() object) by ranges of payload masses
   - It could also be filtered by booster version

Click here for more detail and code.

Predictive Analysis (Classification)

1. Model Development
   - Create a GridSearchCV object and a dictionary of parameters
   - Fit the object to the parameters
   - Use the training data set to train the model
2. Model Evaluation
   - Using the output GridSearchCV object
   - Plot and examine the Confusion Matrix
3. Finding the Best Classification Model
   - Review the accuracy scores for all chosen algorithms and record the highest one

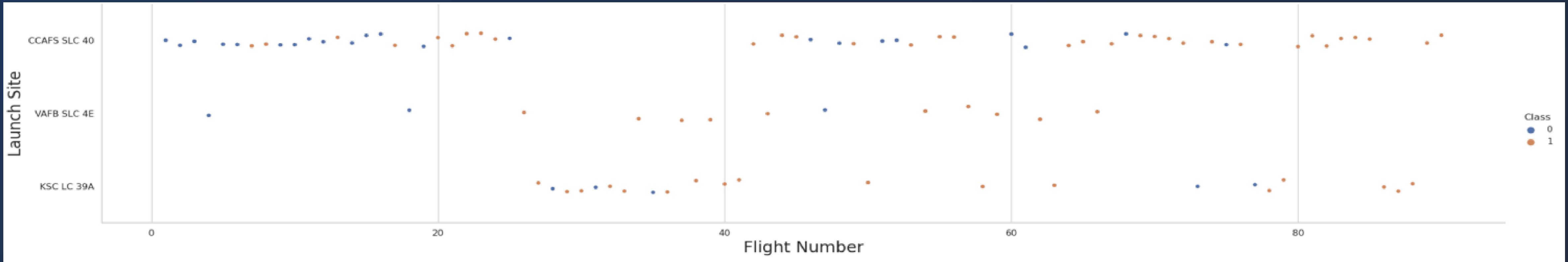
Click here for more detail and code.

Results

- Exploratory data analysis results
  - The success rate kept increasing since year 2013.
  - Orbit at ES-L1, GEO, HEO & SSO with a 100% success rate for rocket launching.
  - Based on scatter point chart, you will find that the VAFB-SLC launch site has no rockets
  - launched for heavy payload mass (greater than 10000).
- Interactive analytics demo in screenshots
  - KSC LC-39A had the most successful launches compared with the other sites.
- Predictive analysis results
  - Decision tree is the best model in terms of prediction accuracy in this dataset

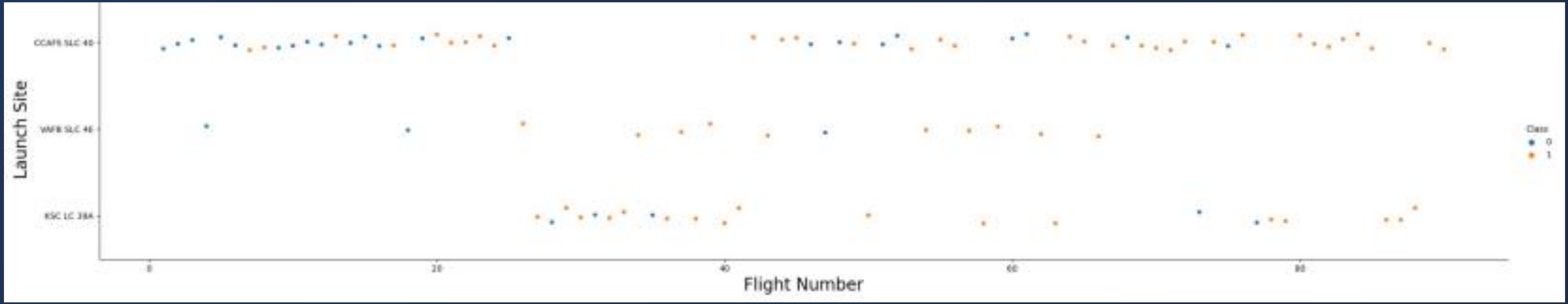
Data Visualization

Flight Number vs. Launch Site



Flight Number increases, the first stage is more likely to land successfully with more results of Class "1" for Successful Outcome. This indicates a relationship.

Payload vs. Launch Site

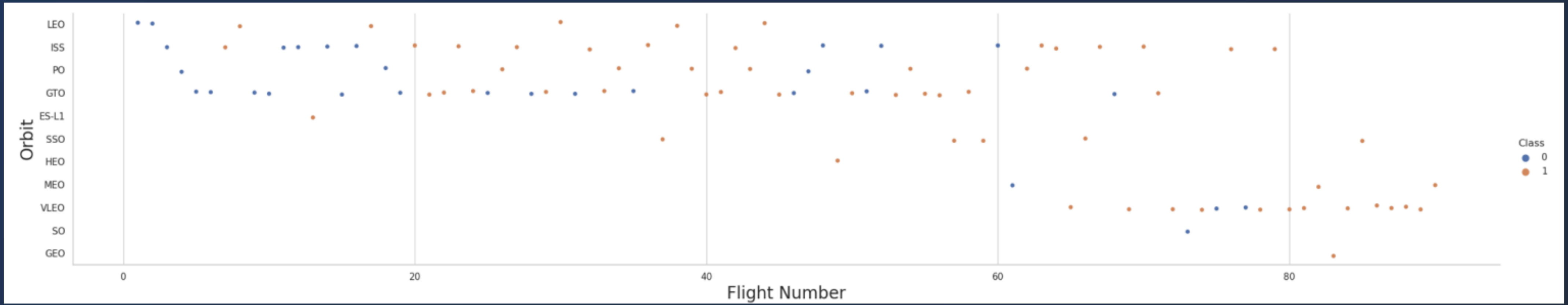


Rocket launching from the launch site of CCAFS SLC 40 are significantly higher than the rest of other sites.

Success Rate vs. Orbit Type

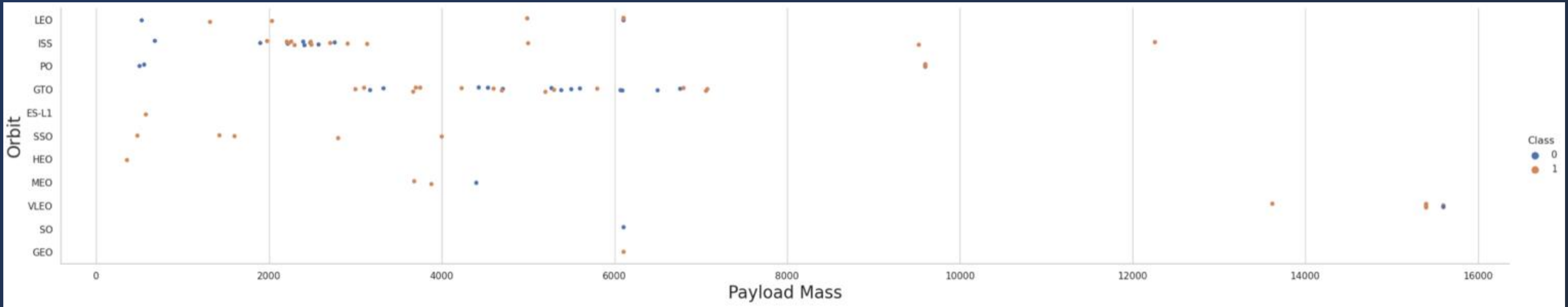
Based on scatter point chart, you will find that the VAFB-SLC launch site has no rockets launched for heavy payload mass (greater than 10000).

Flight Number vs. Orbit Type



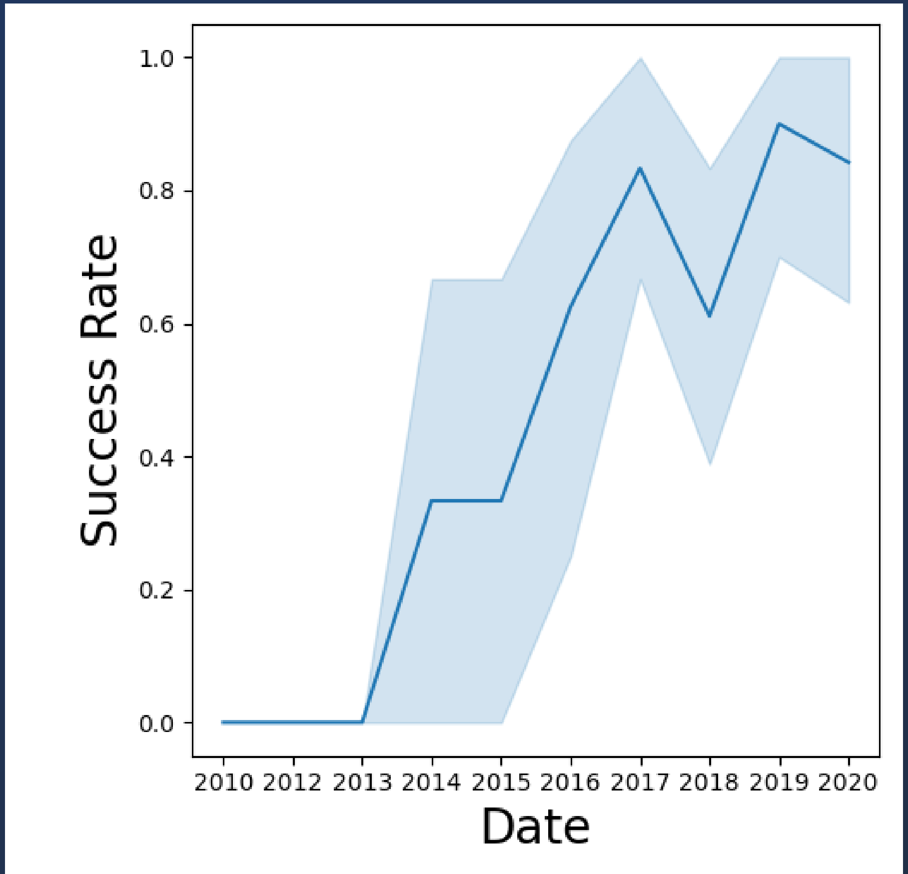
The scatterplot can show some cases where the Orbit Type can be considered related to the Flight Number, and other cases where there is no definitive relationship to infer.

Payload vs. Orbit Type



The more likely they are successful landings. This can mean a relationship is present between mass and the orbit type with the success of the landing.

Launch Success Yearly Trend



- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.

All Launch Site Names

- %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEX ORDER BY 1

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

Launch Site Names Begin with 'CCA'

- %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEX ORDER BY 1

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

IBM Developer SKILLS NETWORK

- • %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEX WHERE CUSTOMER = 'NASA (CRS)'

1
45596

Average Payload Mass by F9 v1.1

- %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEX WHERE BOOSTER_VERSION = 'F9 v1.1'

1
2928

First Successful Ground Landing Date

- %sql SELECT DATE FROM SPACEX WHERE LANDING__OUTCOME = 'Success (ground pad)' LIMIT 1

DATE
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- %%sql
  SELECT DISTINCT BOOSTER_VERSION
  FROM SPACEX
  WHERE LANDING__OUTCOME = 'Success (drone ship)’
  AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
  ORDER BY 1

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

Total Number of Successful and Failure Mission Outcomes

- %%sql
```
SELECT MISSION_OUTCOME, COUNT(*) AS REC
FROM SPACEX
GROUP BY MISSION_OUTCOME
ORDER BY MISSION_OUTCOME
```

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- %%sql
```
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEX
WHERE PAYLOAD_MASS__KG_ =
        (SELECT MAX(PAYLOAD_MASS__KG_)
         FROM SPACEX)
ORDER BY 1
```

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

2015 Launch Records

- %%sql

```
SELECT
LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEX
WHERE LANDING__OUTCOME = 'Failure (drone ship)’
        AND EXTRACT(YEAR FROM DATE) = 2015
GROUP BY LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
ORDER BY LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- %%sql

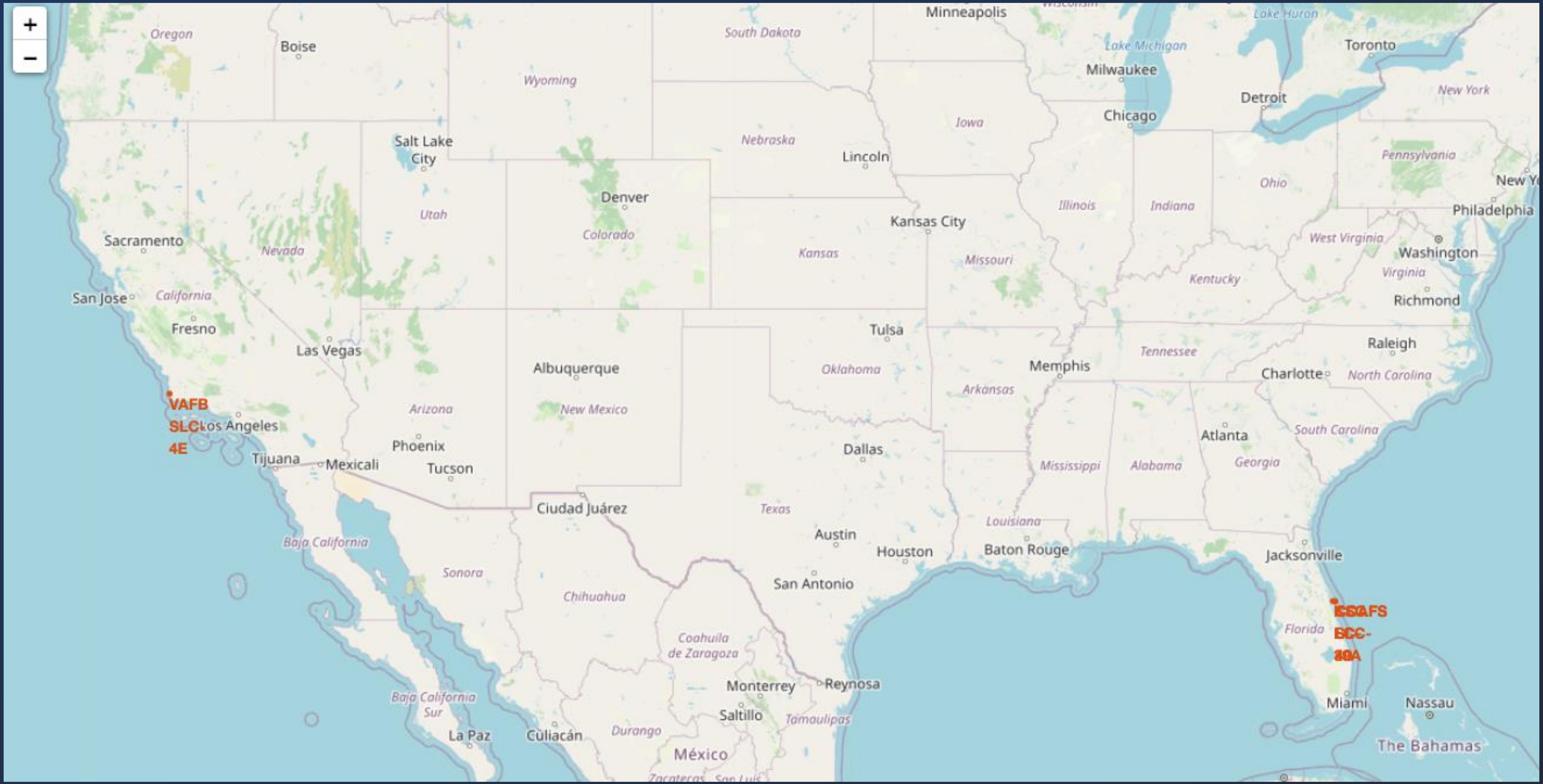
```
SELECT *, RANK() OVER (ORDER BY REC DESC) AS RANK
FROM ( SELECT LANDING__OUTCOME, COUNT(*) AS REC
        FROM SPACEX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20’
        GROUP BY LANDING__OUTCOME
        )
```

landing__outcome	rec	RANK
No attempt	10	1
Failure (drone ship)	5	2
Success (drone ship)	5	2
Controlled (ocean)	3	4
Success (ground pad)	3	4
Failure (parachute)	2	6
Uncontrolled (ocean)	2	6
Precluded (drone ship)	1	8

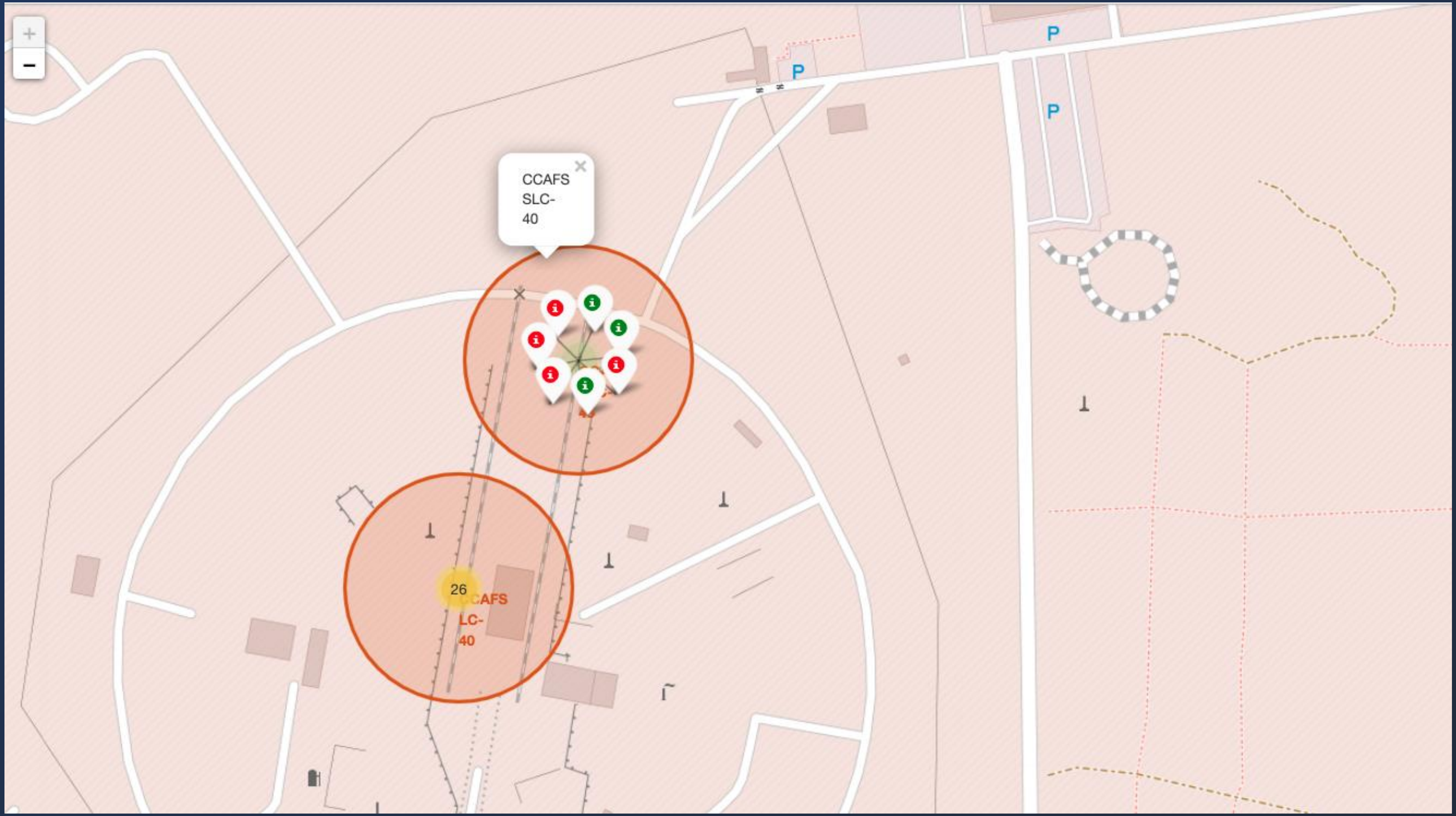
LAUNCH SITES PROXIMITY ANALYSIS

<Folium Map Screenshot 1>

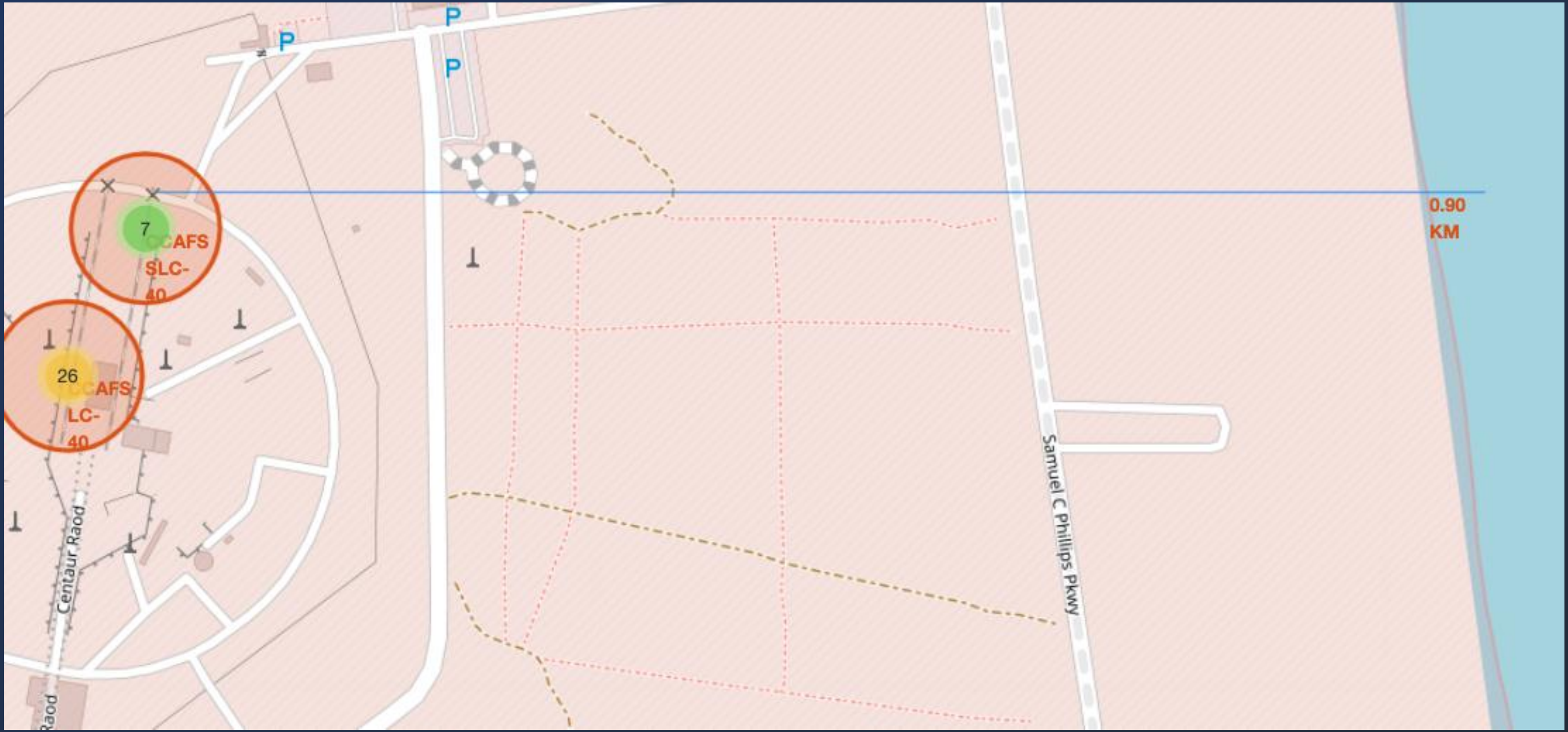
IBM Developer SKILLS NETWORK



<Folium Map Screenshot 2>

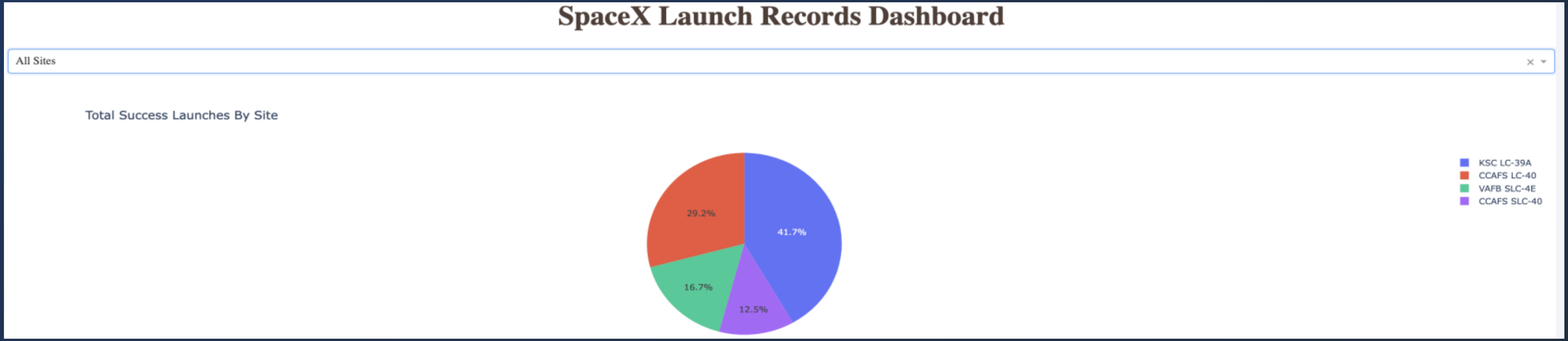


<Folium Map Screenshot 3>

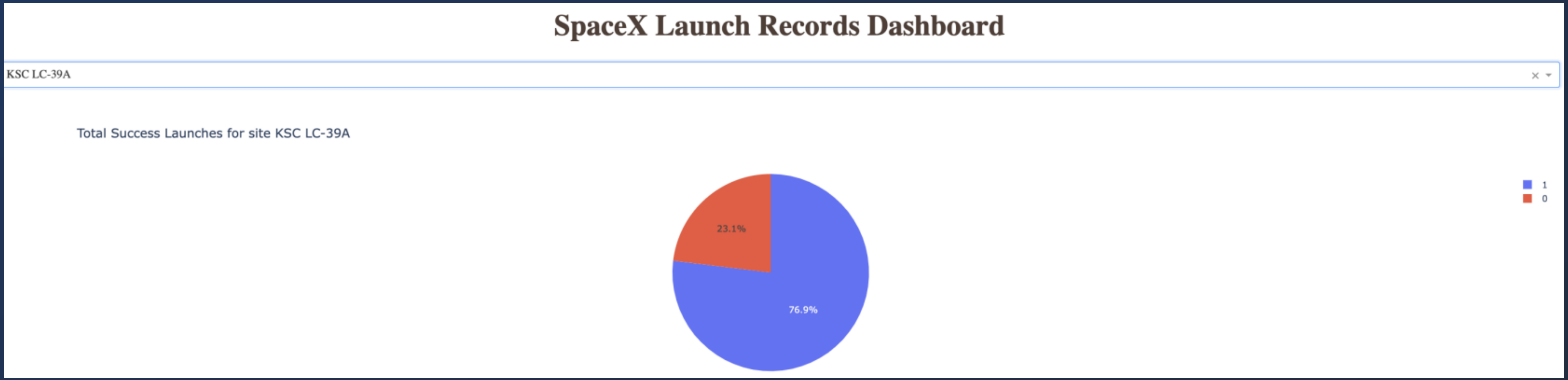



Build a Dashboard
With Plotly Dash

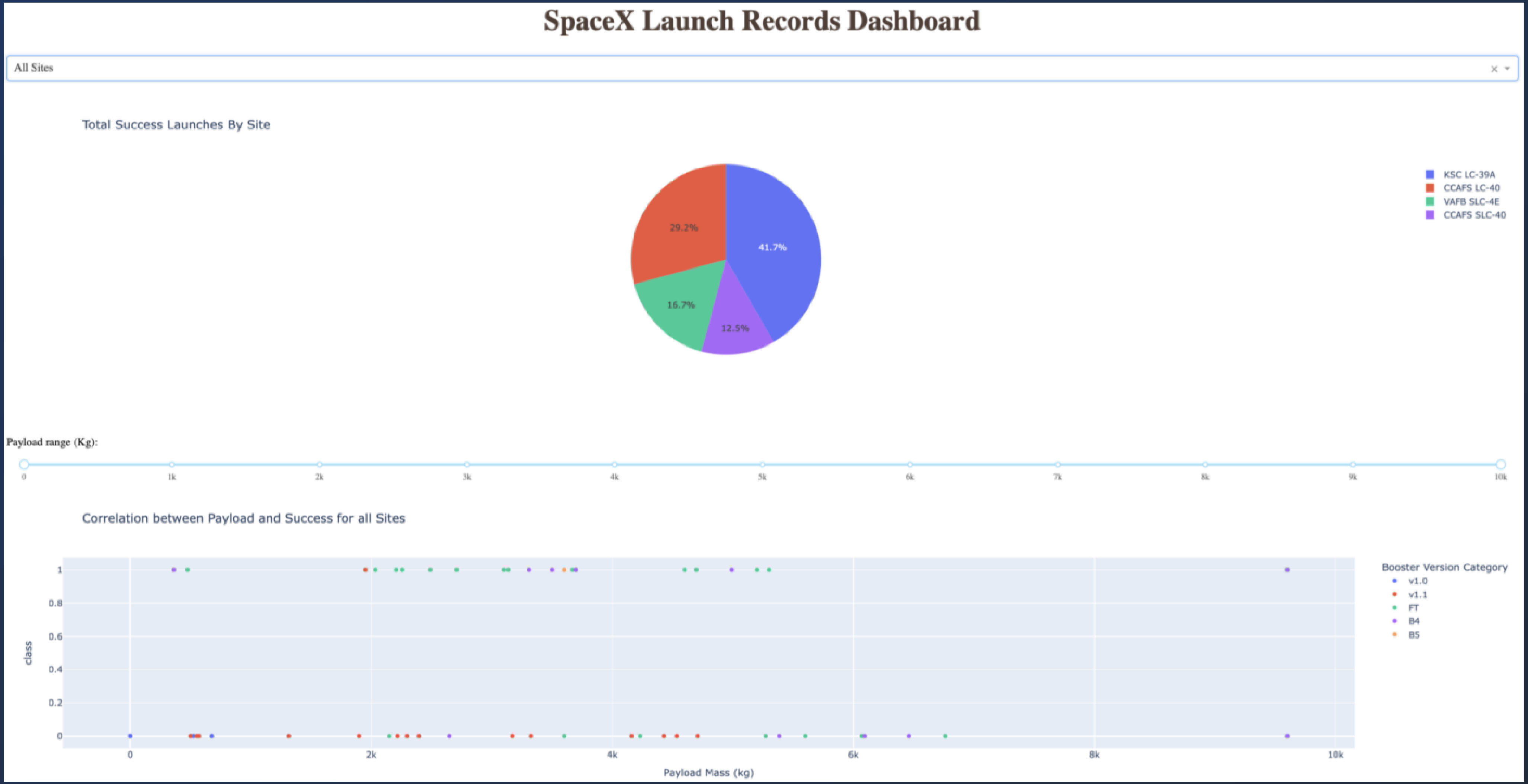
<Dashboard Screenshot 1>



<Dashboard Screenshot 2>



<Dashboard Screenshot 3>

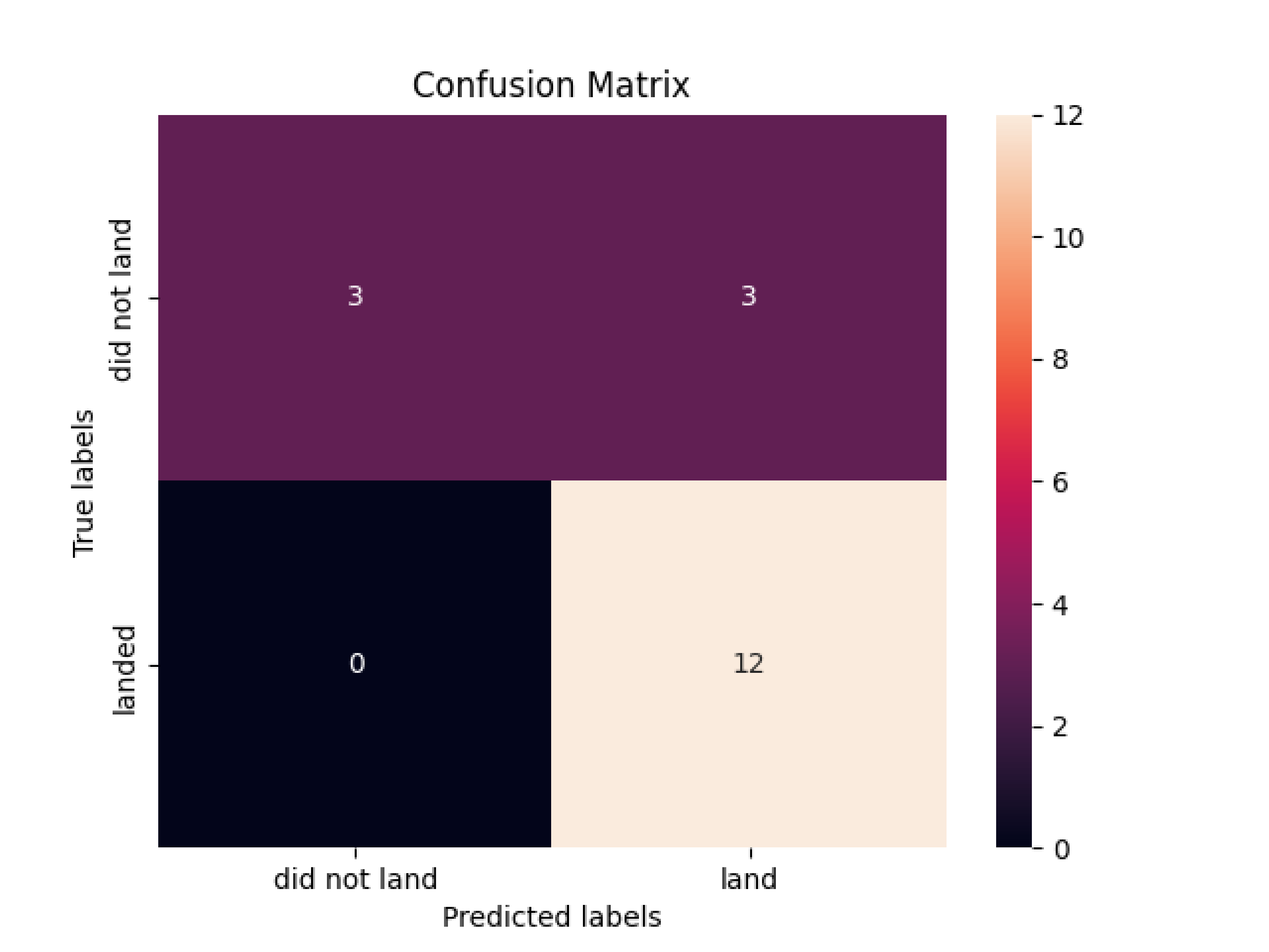


Predictive Analysis (Classification)

Classification Accuracy

- Decision tree is the best model in terms of prediction accuracy in this dataset.

Confusion Matrix



Conclusions

1. Decision tree is the best model in terms of prediction accuracy in this dataset.
2. Low weighted payloads perform better than the heavier payloads.
3. 4 classification method models used: Logistic Regression, SVM, Decision Tree, and KNN.

Appendix

IBM Developer SKILLS NETWORK

- In case you are interested in exploring the coding files used in this presentation, you can visit my GitHub page, where all of the files are available. I have added the individual links to the files throughout the presentation.
- Thank you for taking the time to read my presentation. I hope you found it informative and enjoyable.


Thank You