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 scrapping
  - Exploratory DATA Analysis (EDA) with Data visualization
  - EDA with SQL
  - Interactive Map with Folium
  - Dashboards with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis
  - Interactive maps and dashboard
  - Predictive results

# Introduction

- Project background and context
  - Our aim is to predict successful landings of Space X's Falcon 9 rocket. This results in more efficient space exploration modules which can be reused, reducing the cost of this endeavor
- Problems you want to find answers
  - What are the main characteristics of successful landings
  - What features affect a successful or a failed landing?
  - Can we predict when a landing will be successful?
  - What conditions allow SpaceX to achieve higher success rates?

Section 1

# Methodology

Acc

Accuracy of Logistic Regression: 0.8333333333333334 Accuracy of Support Vector Machine: 0.8333333333333334 Accuracy of Decision Tree: 0.8333333333333334 Accuracy of K-Nearest Neighbors: 0.8333333333333334

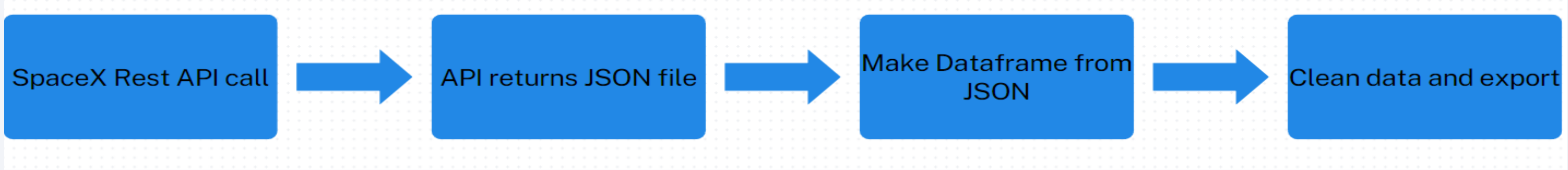
# Methodology

## Executive Summary

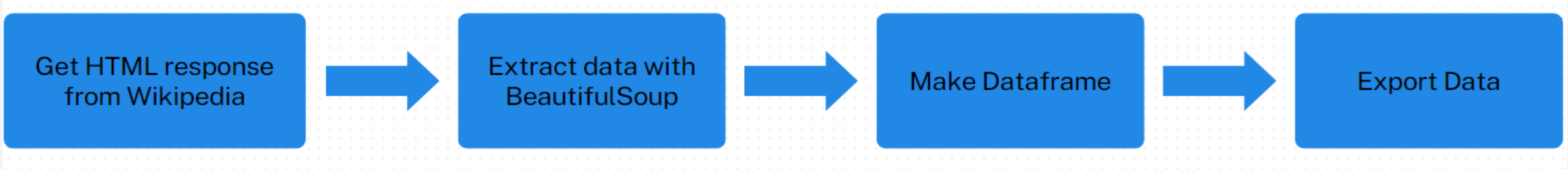
- Data collection methodology:
  - SpaceX REST API
  - Web Scraping from Wikipedia
- Perform data wrangling
  - Dropping unnecessary columns
  - One-hot encoding for classification models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Built Logistic Regression, SVM, Decision tree and KNN models to predict Falcon 9 successful landings with tuning hyperparameters

# Data Collection

- Data sets collected from REST SPACEX API and web scrapping Wikipedia
  - SpaceX Rest API URL is api.spacexdata.com/v4




- Data sets were collected web scrapping Wikipedia (updated 06/2021):
  - URL:https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922



# 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 to JSON

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

3. Transform data

```
# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)

# Call getBoosterVersion
getBoosterVersion(data)
```

4. 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}
```

5. Create Dataframe

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

6. Filter dataframe

```
data_falcon9 = df.loc[df['BoosterVersion']!='Falcon 1']
data_falcon9
```

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, 'html.parser')
```

2. Find all tables

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

4. Get column names

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

4. 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('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
```

etc...

7. Create dataframe from dictionary

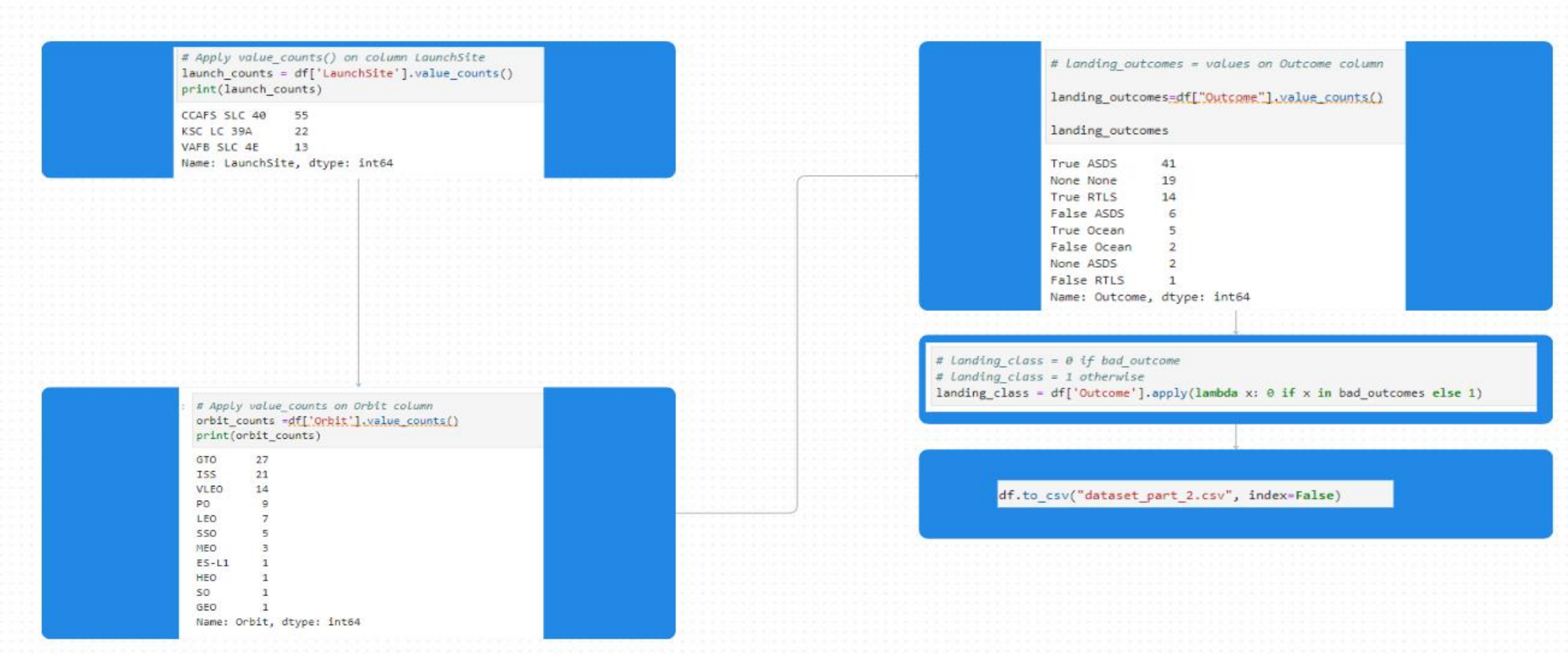
```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

8. Export to file

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

# Data Wrangling

- There are cases where booster did not land successfully:
    - TrueOcean, TrueRTLS, TrueASDS means the mission has been successful
    - False Ocean, False RTLS, False ASDS means the mission was a failure
- Transform string variables into categorical variables where 1 means the mission has been successful and 0 means the mission was a failure.



# EDA with Data Visualization

- Scatter Graphs: show relationship between variables (i.e. correlation)
  - 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
- Bar Graph: show the relationship between numeric and categoric variables
  - Success rate vs. Orbit.
- Line Graph: show data variables and their trends
  - Success rate vs. Year

# EDA with SQL

- Performed SQL queries to gather and understand dataset:
  - Displaying the names of the unique launch sites
  - Display 5 records where launch sites begin with the string 'CCA’
  - Display the total payload mass carried by boosters launched by NASA (CRS).
  - Display average payload mass carried by booster version F9 v1.1.
  - List the date when the first successful landing outcome in ground pad was achieved.
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
  - List the total number of successful and failure mission outcomes.
  - List the names of the booster_versions carrying maximum payload mass.
  - List the records with month names, failure landing_outcomes in drone ship, booster versions, launch_site for the months in year 2015.
  - Rank successful landing_outcomes between the date 2010 and 2017 in descending order.

# Build an Interactive Map with Folium

- Folium map object is a map centered on NASA Johnson Space Center at Houston
  - Red circle at NASA Johnson Space Center's coordinate with label showing its name (folium.Circle, folium.map.Marker).
  - Red circles at each launch site coordinates with label showing launch site name (folium.Circle, folium.map.Marker, folium.features.DivIcon).
  - The grouping of points in a cluster to display multiple and different information for the same coordinates (folium.plugins.MarkerCluster).
  - Markers to show successful and unsuccessful landings. Green for successful landing and Red for unsuccessful landing. (folium.map.Marker, folium.Icon).
  - Markers to show distance between launch site to key locations (railway, highway, coastway, city) and plot a line between them. (folium.map.Marker, folium.PolyLine, folium.features.DivIcon)

# Build a Dashboard with Plotly Dash

- Dashboard has dropdown, pie chart, range slider and scatter plot components
  - Pie chart shows the total success and the total failure for the launch site chosen with the dropdown component (plotly.express.pie).
  - Range slider 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)

- Preparation
  - Load data
  - Normalize
  - Split into train/test sets
- Models:
  - Used four models: Logistic regression, SVM, decision tree and KNN
  - Tuned hyperparameters to increase accuracy
  - Computed accuracy and confusion matrix
- Compare
  - Based on accuracy, selecting best model

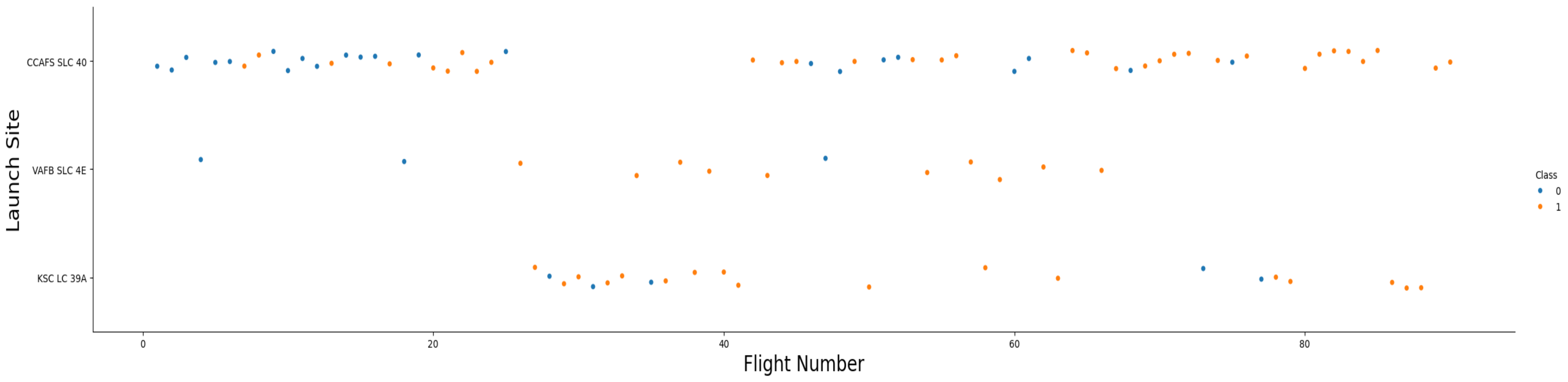
# Results

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

Section 2

# Insights drawn from EDA

# Flight Number vs. Launch Site

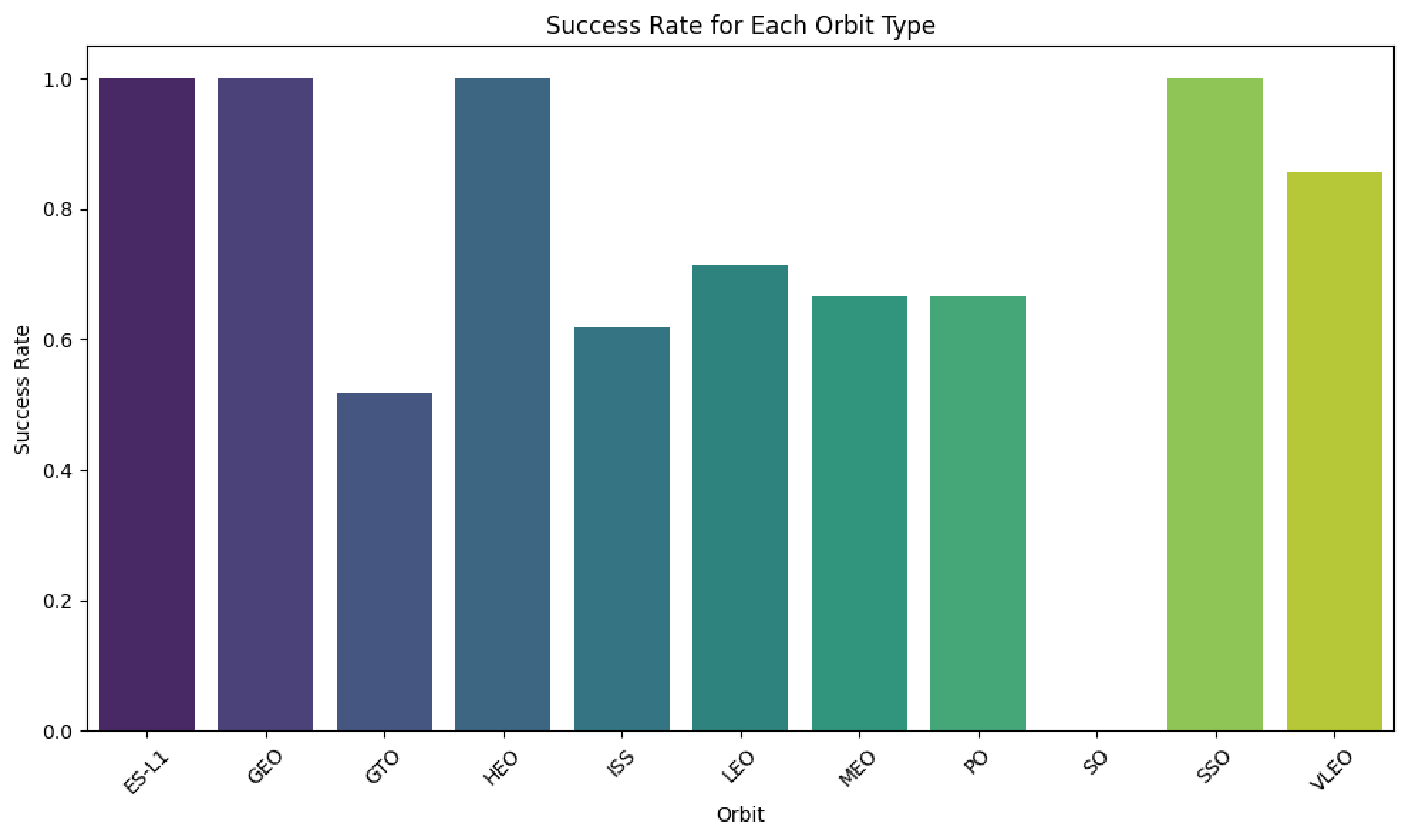



- In general, the success increases with flight number for each site.
- KSL site has some failed flights even at large number

# Payload vs. Launch Site

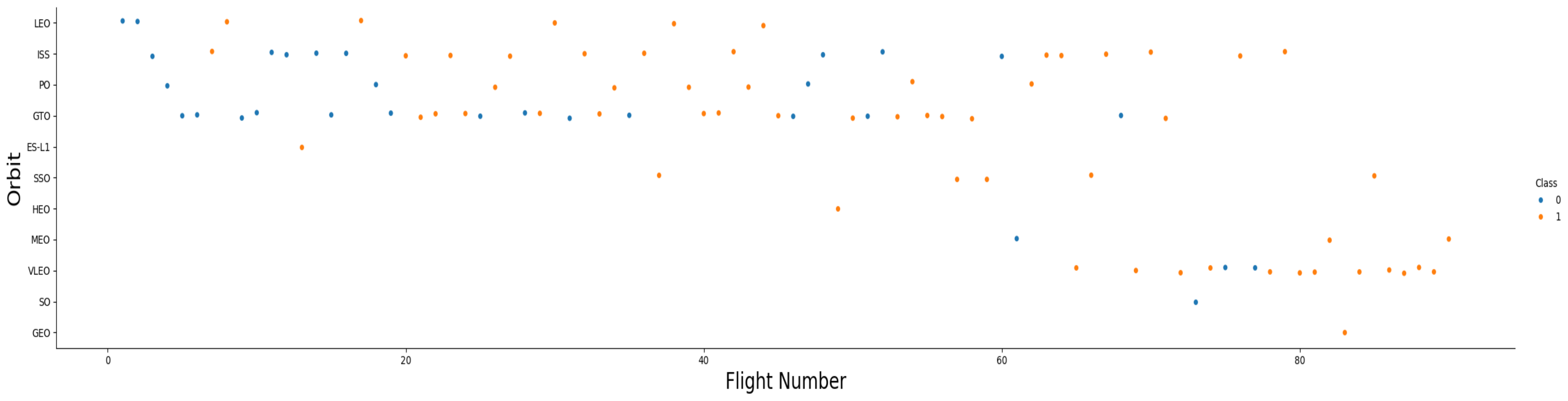

- For CCAFS site high payload has more failed landing while for VAFB payload has less effect on landing success

# Success Rate vs. Orbit Type



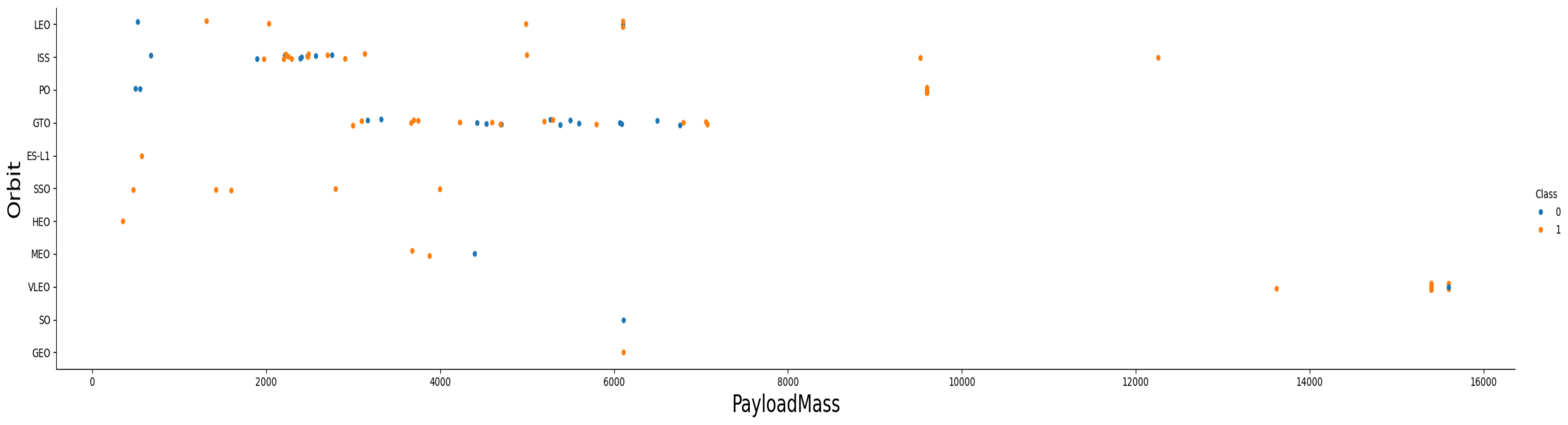

- Some orbits have more success rate than others, namely: ES-L1, GEO, HEO and SSO.

# Flight Number vs. Orbit Type



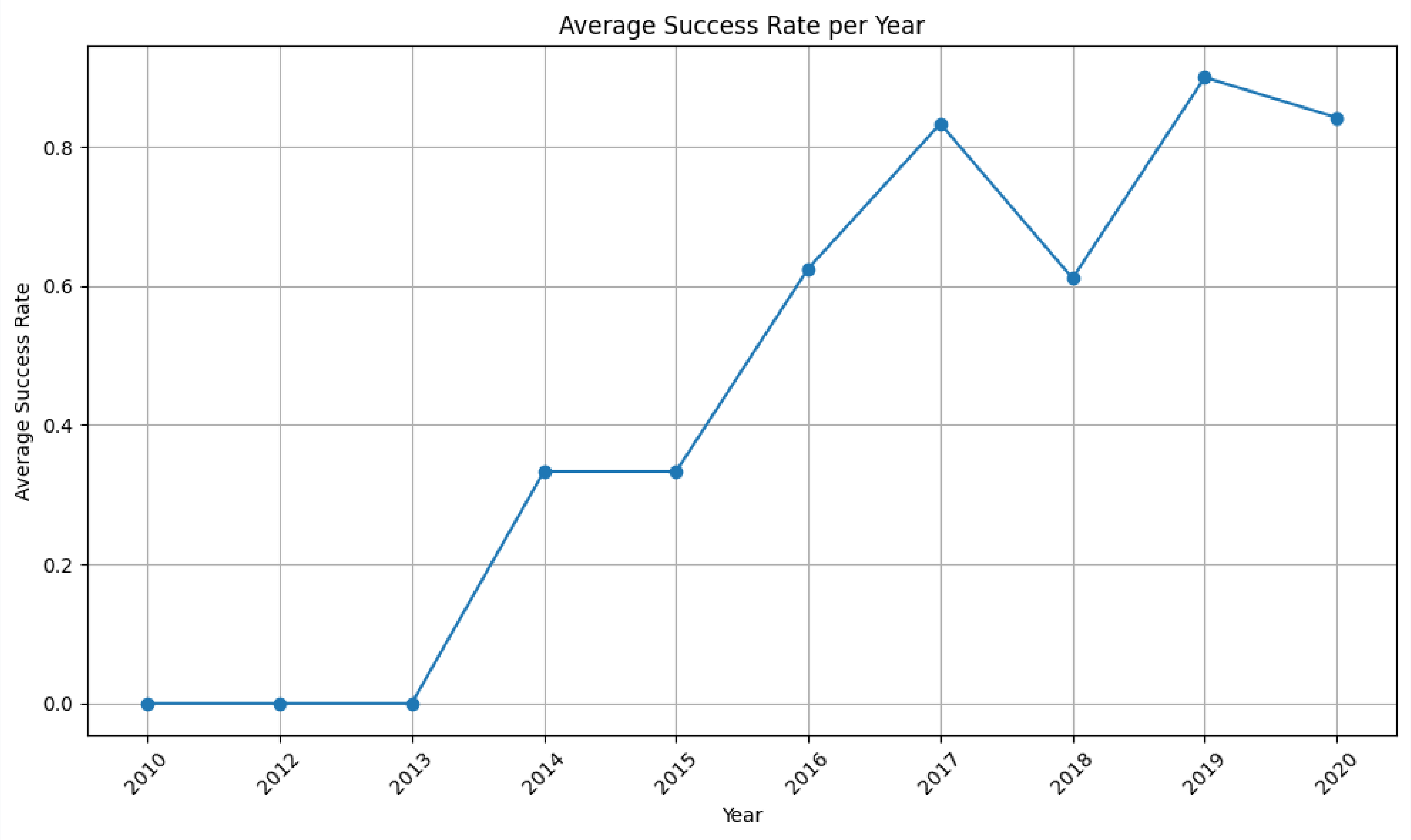

- Some of the orbits have increased success with number of orbits. This can explain the former success rates in the previous slide (i.e. more trial and error makes for better landings)

# Payload vs. Orbit Type




- Some orbits like GTO has more failed landings for heavier payloads than others. However, LEO or PO can sustain higher payloads with lower failures.

# Launch Success Yearly Trend




- The success rate increases per year. This can be due to more knowledge on repeating previous successes and preventing previous errors, while better technology.

# All Launch Site Names

```python
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

| | Launch Site | Lat | Long |
|---|---|---|---|
| 0 | CCAFS LC-40 | 28.562302 | -80.577356 |
| 1 | CCAFS SLC-40 | 28.563197 | -80.576820 |
| 2 | KSC LC-39A | 28.573255 | -80.646895 |
| 3 | VAFB SLC-4E | 34.632834 | -120.610745 |

```sql
%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 |

- We find the distinct launch sites by using groupby function.
- Alternatively we could use SQL query: SELECT DISTINCT "LAUNCH_SITE" FROM SPACEX

# Launch Site Names Begin with 'CCA'

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

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

| 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 | 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 |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

- We use the WHERE clause followed by LIKE to constrain the search. Then we use LIMIT to find 5 records

# Total Payload Mass

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "CUSTOMER" = 'NASA (CRS)'
```

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

| SUM("PAYLOAD_MASS__KG_") |
|---|
| 45596 |

- Use SUM on the Payload_Mass column constrained to costumer being NASA (CRS)

# Average Payload Mass by F9 v1.1

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%'
```

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

| AVG("PAYLOAD_MASS__KG_") |
| --- |
| 2534.6666666666665 |

- Calculate the average of the Payload_Mass column constraining the booster_version type to F9 v1.1

# First Successful Ground Landing Date

```
%sql SELECT MIN("DATE") FROM SPACEXTBL WHERE "Landing_Outcome" LIKE '%Success%'
```

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

| MIN("DATE") |
| --- |
| 2015-12-22 |

- We find the first date using MIN function where the Landing_Outcome has some Success in its entry

# 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_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

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

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

- We use WHERE to constrain the successful landings where the Payload_Mass entry is between the given boundaries

# Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT (SELECT COUNT("Mission_Outcome") FROM SPACEXTBL WHERE "Mission_Outcome" LIKE '%Success%') AS SUCCESS, \
(SELECT COUNT("Mission_Outcome") FROM SPACEXTBL WHERE "Mission_Outcome" LIKE '%Failure%') AS FAILURE
```

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

| SUCCESS | FAILURE |
|---|---|
| 100 | 1 |

- We select the successes and failures and store them under their respective names

# Boosters Carried Maximum Payload

```
%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTBL \
WHERE "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

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

| 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 |

- We use the command SELECT DISTINCT to choose the distinct boosters and WHERE to select those carrying payload equal to maximum payload

# 2015 Launch Records

```
%sql SELECT substr("Date",6, 2) AS MONTH, "Booster_Version", "Launch_Site" FROM SPACEXTBL\
WHERE "Landing_Outcome" = 'Failure (drone ship)' and substr("Date",0,5) = '2015'
```

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

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

- Select those records by displaying the month names, failure landing_outcomes in drone ship, booster version, launch site for the months in 2015

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

```
%sql SELECT "Landing_Outcome", COUNT("Landing_Outcome") FROM SPACEXTBL\
WHERE "Date" >= '2010-06-04' and "Date" <= '2017-03-20' and "Landing_Outcome" LIKE '%Success%'\
GROUP BY "Landing_Outcome" \
ORDER BY COUNT("Landing_Outcome") DESC ;
```

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

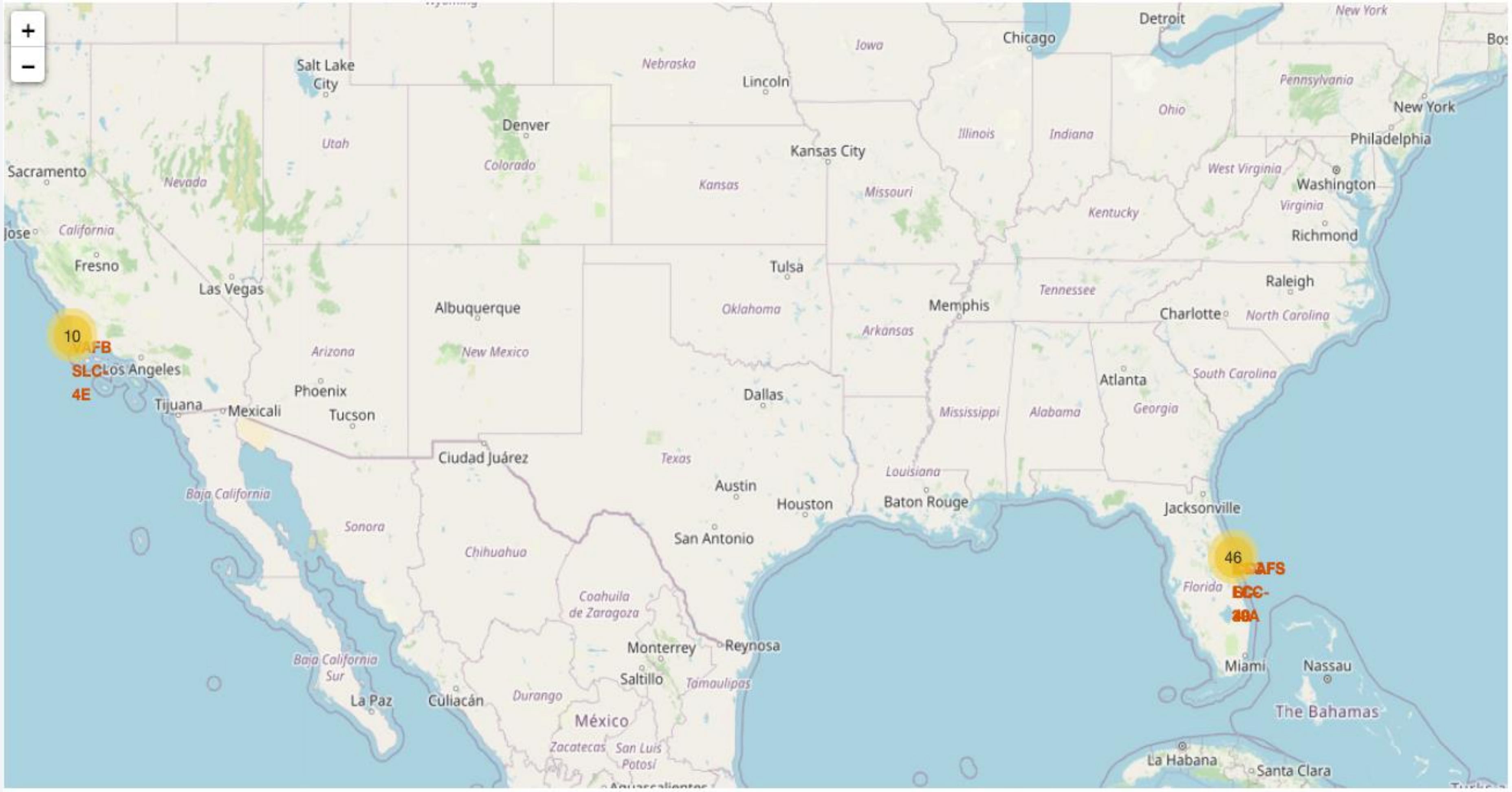
| Landing_Outcome | COUNT("Landing_Outcome") |
|---|---|
| Success (drone ship) | 5 |
| Success (ground pad) | 3 |

- Count from a selection where the landing outcome is success for drone and ground.

Section 3

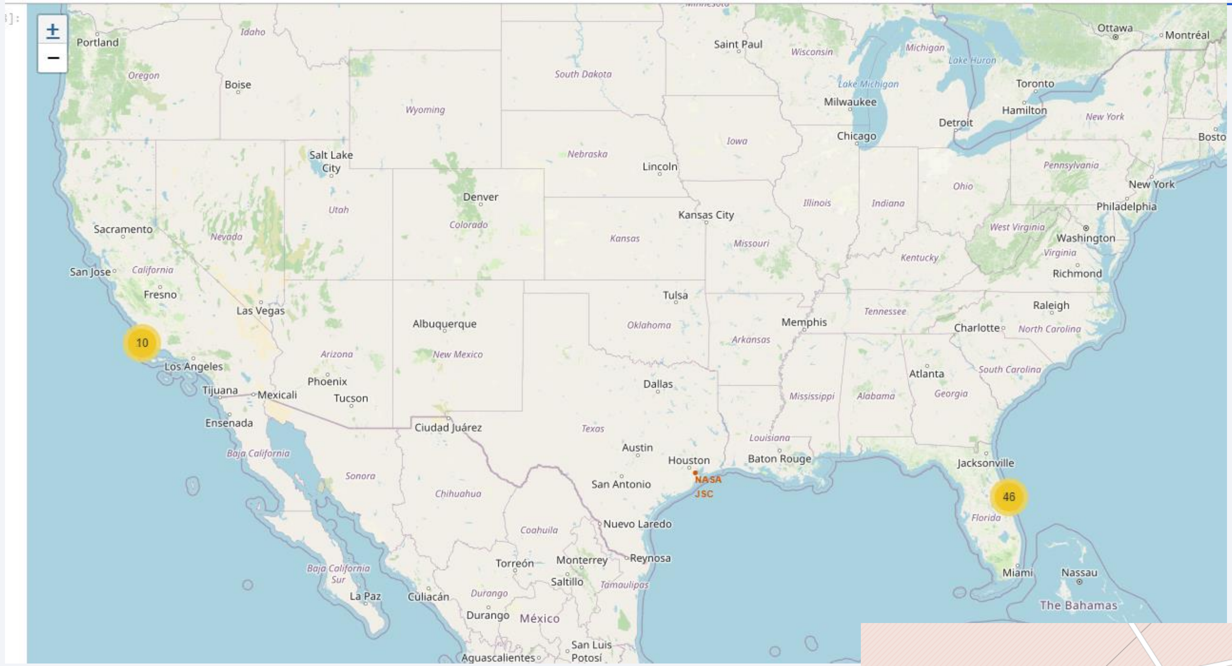
# Launch Sites Proximities Analysis

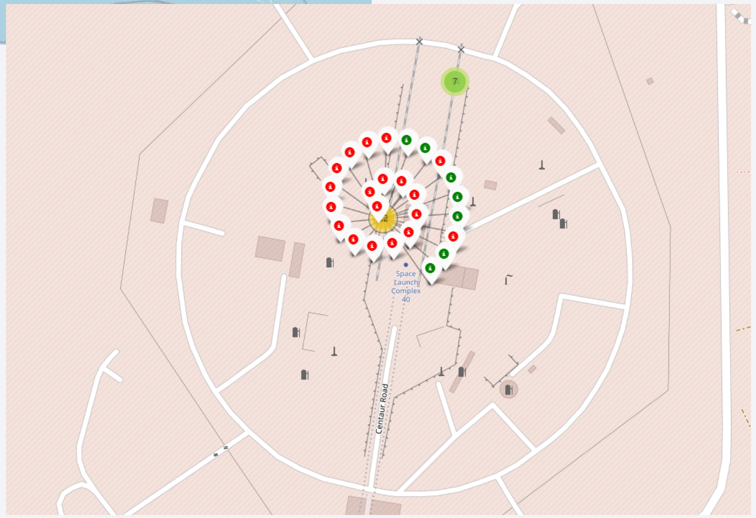
# <Folium Map Screenshot 1>




- SpaceX locations are on the coast of California and Florida states (USA)

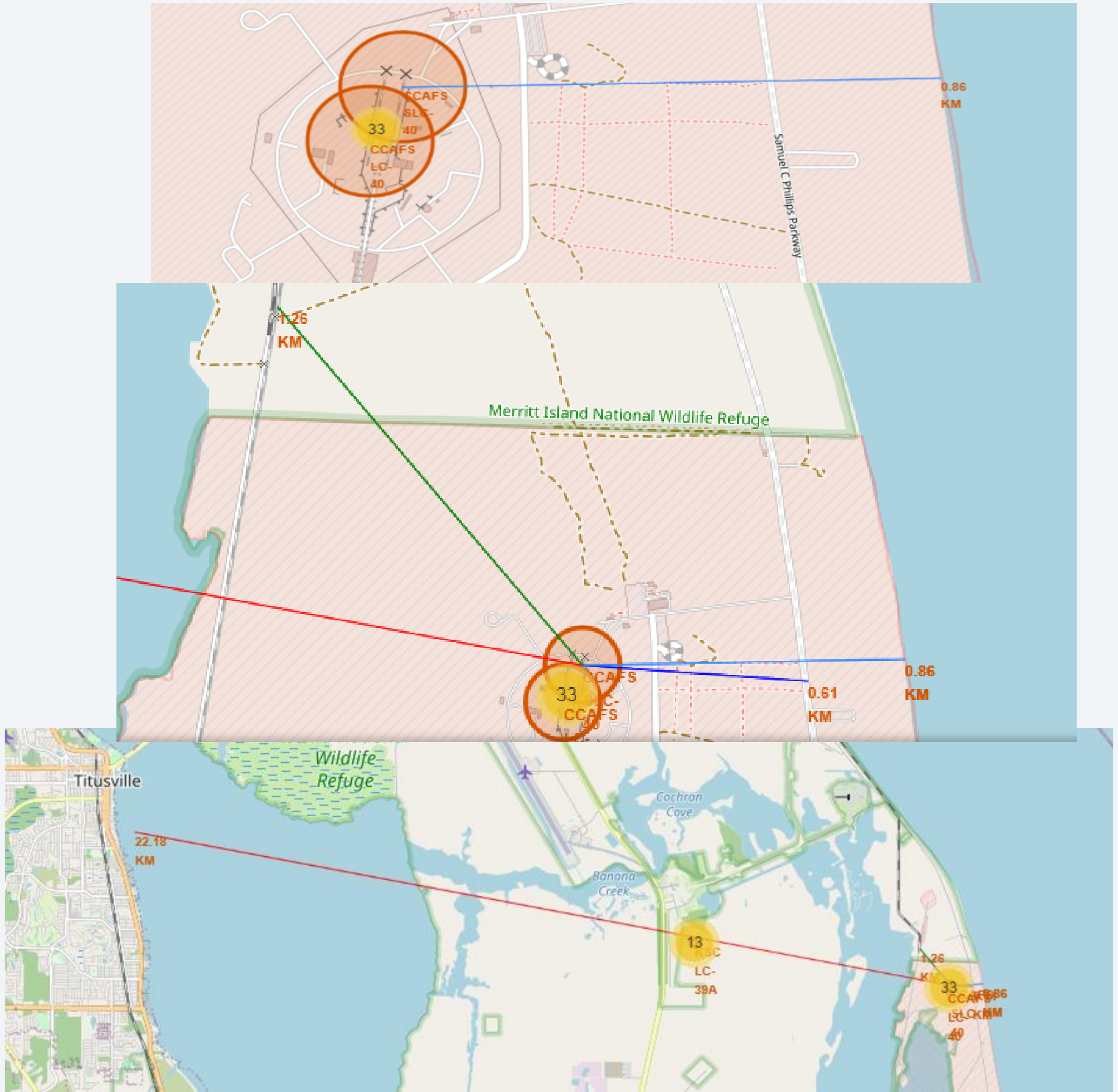
# <Folium Map Screenshot 2>




- As an example, we show one of the locations with green markers (showing successful landings) and red markers (failed landings).

# <Folium Map Screenshot 3>

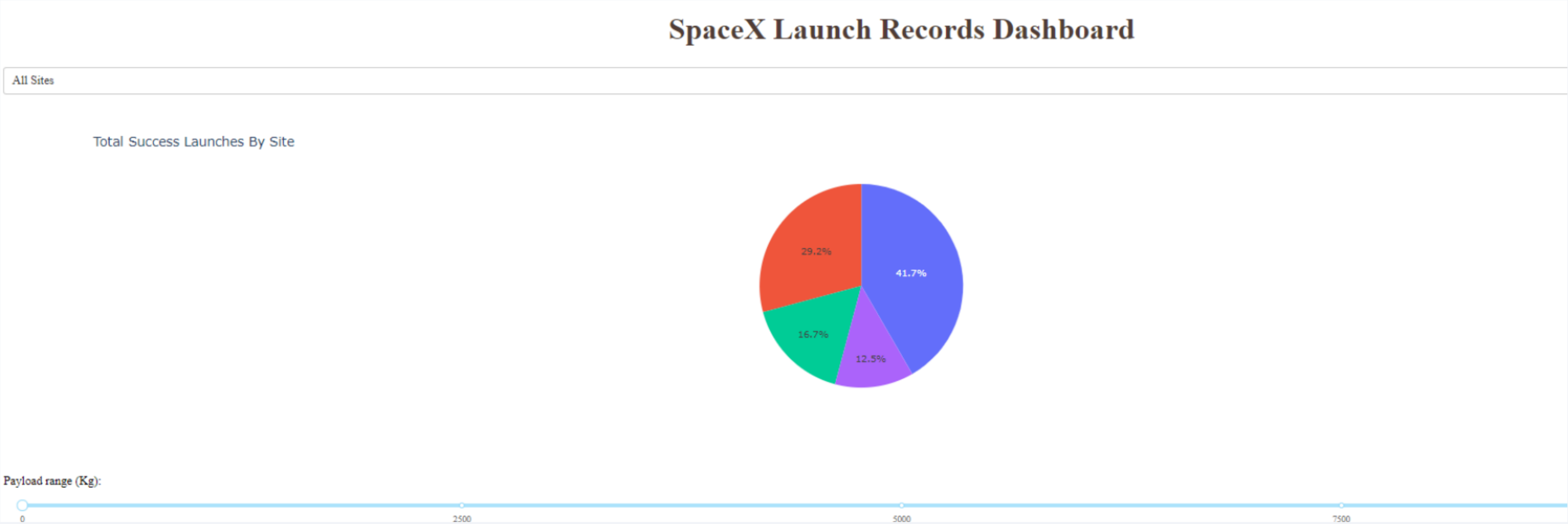



- We see how the proximity to coastlines
- To roads
- To railways
- To cities

Section 4

# Build a Dashboard with Plotly Dash

# Total Success by site

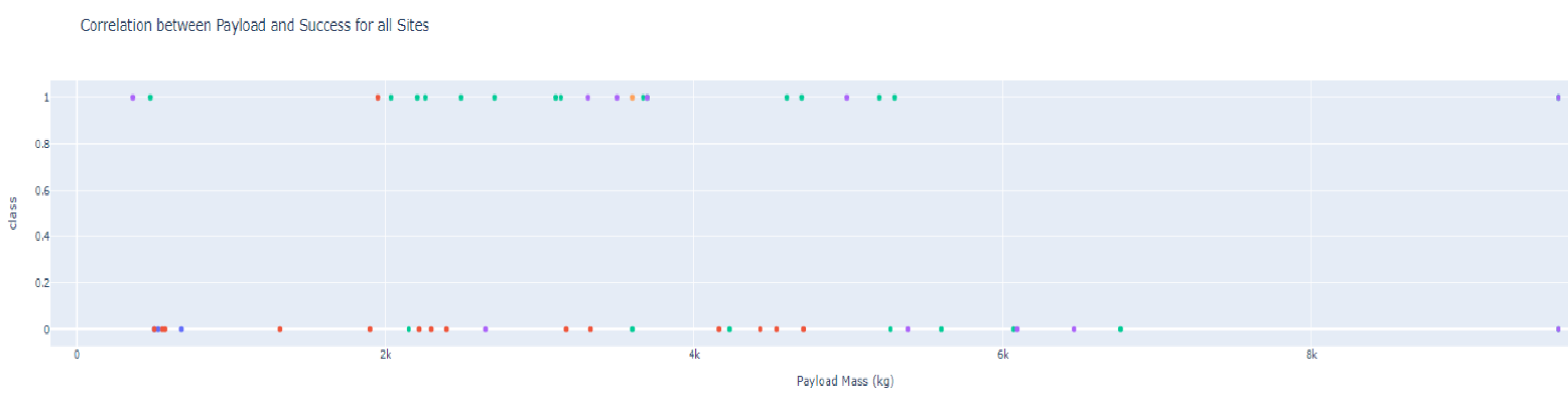



- KSC LC-39A has the largest success rate

# Total success – CCAFS LC-40


- CCAFS LC-40 has a success rate of 73.1% and failure rate of 26.1%

# Payload and success



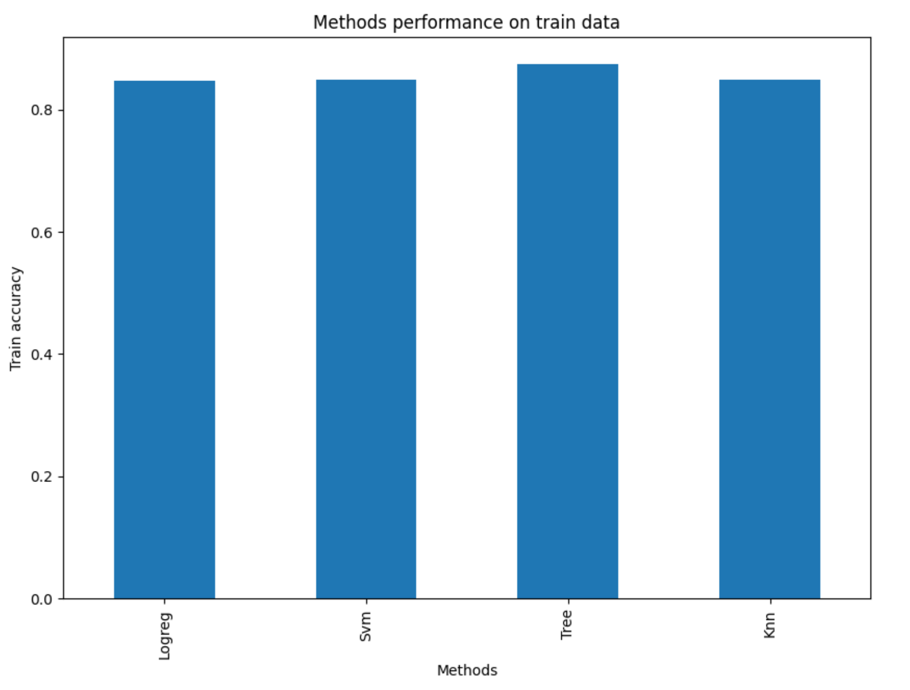

- Correlation between payload and success for all sites

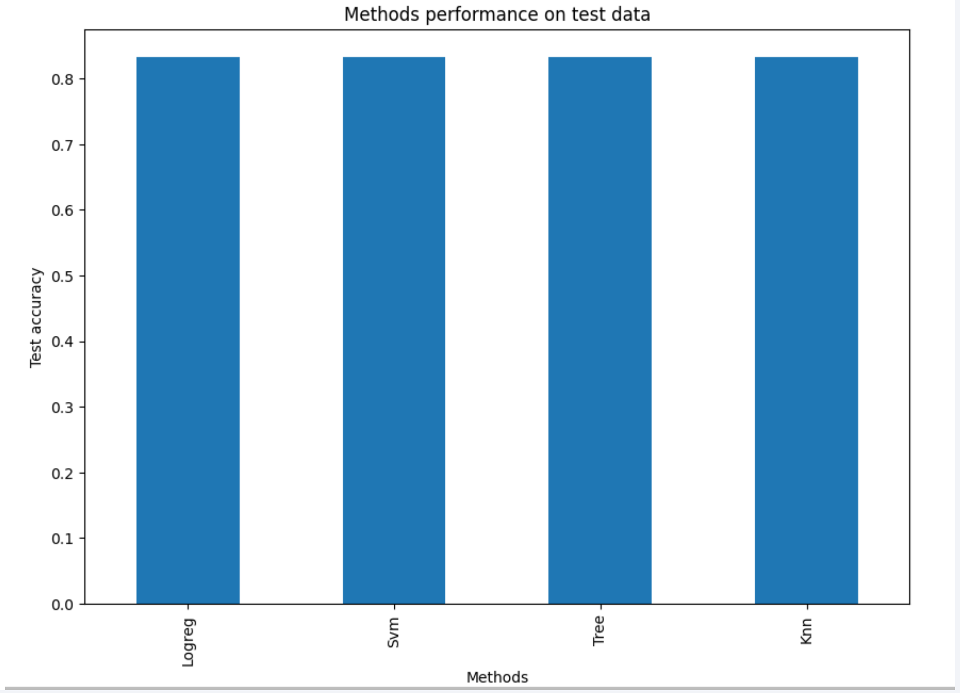
Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

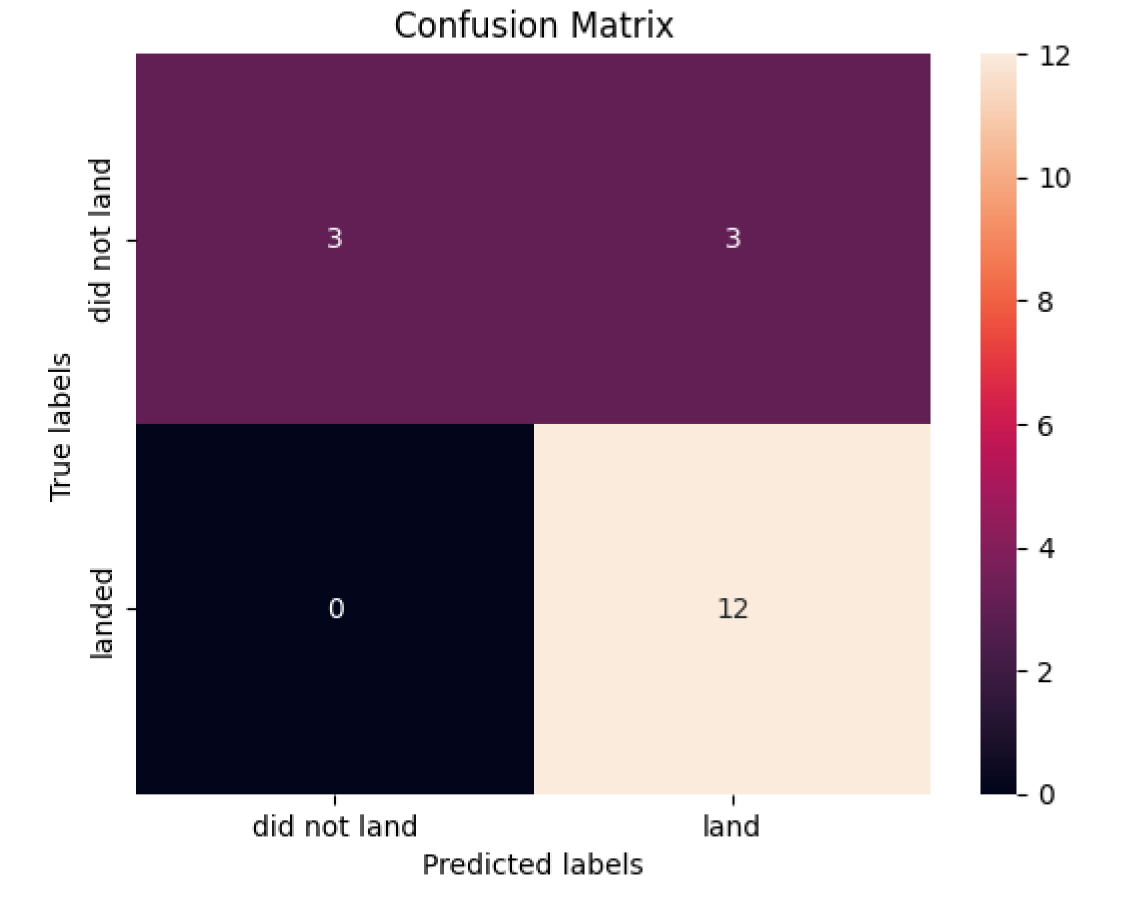
| | Accuracy Train | Accuracy Test |
|---|---|---|
| **Logreg** | 0.846429 | 0.833333 |
| **Svm** | 0.848214 | 0.833333 |
| **Tree** | 0.875000 | 0.833333 |
| **Knn** | 0.848214 | 0.833333 |




- All methods performed roughly the same. Performance on test set was the same

# Confusion Matrix




- Confusion matrix for the logistic regression

# Conclusions

- **Mission Success Factors**: Success of a mission relies on multiple factors such as launch site, orbit, and the cumulative knowledge gained from prior launches. The accumulation of knowledge from past missions likely contributes to the transition from launch failures to successful missions.
- **Successful Orbits:** Orbits with notably high success rates include GEO (Geostationary Earth Orbit), HEO (Highly Elliptical Orbit), SSO (Sun-Synchronous Orbit), and ES-L1 (Earth-Sun L1 Lagrange Point).
- **Payload Considerations:** Payload mass is a critical factor depending on the orbit. Different orbits might require light or heavy payloads. Generally, missions with lower-weighted payloads tend to have better success rates compared to heavier payloads.
- **Unexplained Success of Launch Sites:** Certain launch sites, particularly KSC LC-39A, exhibit higher success rates without clear explanations. Obtaining additional atmospheric or relevant data could provide insights into these success rates.
- **Model Selection for Dataset:** All models performed the same with Logistic regression performing slightly better on training set.

# Appendix

- The notebooks for this assignment are in:

# Thank you!