Winning Space Race with Data Science

Olorunsuyi Oluwakayode
22/12/22

Outline

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

Executive Summary

- Summary of methodologies
  - Date Collection
  - Data Collection via API, Web Scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - EDA with SQL
  - Interactive Mapping with Folium
  - Dashboard Visualization with Plotly Dash
  - Predictive Machine Learning Analysis
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive maps and dashboard
  - Predictive results

Introduction

- Project background and context
  - The goal of this project is to predict the success of the Falcon 9 first stage landing. According to SpaceX's website, the cost of launching a Falcon 9 rocket is $62 million, while other providers charge upwards of $165 million per launch. The lower cost for SpaceX is due to the fact that the first stage of the Falcon 9 can be reused, which is why predicting the success of the landing is important for determining the cost of a launch. This information would be valuable to another company looking to compete with SpaceX in the rocket launch industry.
- Problems you want to find answers
  - What are the key characteristics that distinguish a successful landing from a failed landing?
  - How do the various relationships between the rocket variables affect the likelihood of a successful or failed landing?
  - Under what conditions is SpaceX most likely to achieve a high landing success rate?

Section 1

Methodology

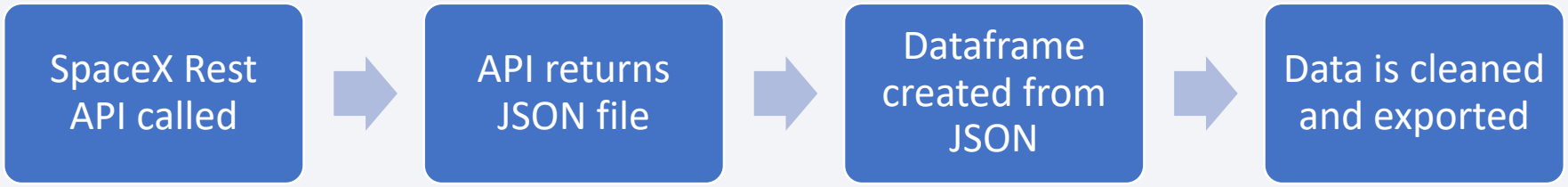
Methodology

Executive Summary

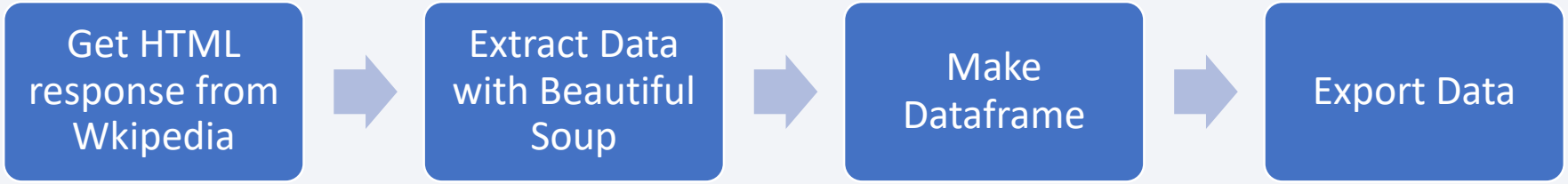
- Data collection methodology:
  - SpaceX REST API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Removing unnecessary columns
  - Using one hot encoding for classification models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Datasets are collected from Rest Space X API and web scraping
  - The API provides information about rockets, launches and payloads.
    - SpaceX REST API URL: https://api.spacexdata.com/v4/



- The data obtained through web scraping Wikipedia includes information about launches, landings and payloads.
  - URL: List of Falcon 9 and Falcon Heavy Launches



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 JSON Content Response to pandas dataframe

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

3. Transform Data

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(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 & Filter dataframe to include only Falcon 9 launches

```
df = pd.DataFrame.from_dict(launch_dict)
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

6. Export to File

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

Link to Code

Data Collection - Scraping

1. Getting Response from HTML Page

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

2. Create BeautifulSoup Object

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

3. Find All tables

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

4. Extract Column Names

```
column_names = []

temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5. Create Dictionary with keys as Column Names

```
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 Values 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"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
```

7. Create Dataframe from Dictionary and save as file

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

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

Link to Code

Data Wrangling

The data set had multiple instances of failed booster landings. The mission outcome can be successful or unsuccessful depending on the location (ocean, ground pad, or drone ship) and whether it was due to an accident. The outcomes was converted into training labels with 1 for successful landings and 0 for unsuccessful landings.

1. Count Launches per site

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

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

2. Count number of orbits occured

```
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
```

3. Count mission outcomes per orbits occured

```
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
```

4. Create landing outcome lable from Outcome

```
landing_class = df['Outcome'].replace({'False Ocean': 0,
df['Outcome'] = df['Outcome'].astype(int)
```

```
df['Class']=landing_class
```

5. Export to file

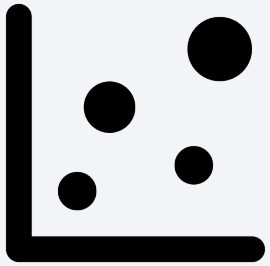
```
df.to_csv("dataset_part_2.csv", index=False)
```

Link to Code

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 variable correlation.

BAR GRAPHs

- Success rate vs. Orbit

Bar graphs show numeric-categoric relationship.

LINE GRAPHs

- Success rate vs Year

Line graphs show trends and predictability over time. They can be used to identify patterns and anomalies in a dataset.

Link to Code

EDA with SQL

We peformed SQL queries to gather and understand data from dataset:

- Displaying the names of the unique launch sites in the space mission.
- Display 5 records where launch site begin with the string ‘CCA’
- Display the total payload mass carried by boosters launched by NASA(CRS)
- Display the 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 name of the boosters which have success in drone ship and have payload mass (4000<mass<6000)
- List the total number of successful and failure mission outcomes
- List the name of the booster versions which have carried the maximum payloaod mass
- List the records which will display the month names, failure landing_outcomes in drone ship, booster versions, launch_site for the months in year 2015
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order

Link to Code

Build an Interactive Map with Folium

Folium map object is a map centered on NASA Johnson Space Center at Houston, Texas

- Red Circle at NASA Johnson Space Center's coordinate with label showing its name(folium.circle, folium.map.Marker)
- Red Circle 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 landing. Green for successful launding 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)

We create these objects above to more effectively understand the problem and data. Using these objects, we easily highlight all launch sites, their surroundings, and the number of successful and unsuccessful landings.

Link to Code

Build a Dashboard with Plotly Dash

Dashboard built has dropdown, pie chart, rangeslider and scatter plot components constructed

- Dropdown allows user to select the launch site or all launch sites (dash_core_components.Dropdown)
- A pie chart displays the total number of successful and failed launches for the selected launch site using a dropdown component. (plotly.express.pie)
- A user can use the range slider to choose a payload mass within a predetermined range.(dash_core_components.rangeslider)
- Scatter chart illustrates the relationship between two variables, specifically the success rate versus payload mass. (plotly.express.scatter)

Link to Code

Predictive Analysis (Classification)

Data Preparation

- Load dataset
- Normalize data
- Split data into train and test sets

Model Preparation

- ML Algorithm Section
- GridSearchCV Parameter tuning
- Training Models with Data

Model Evaluation

- Find best hyperparameters
- Calculate model accuracy with test dataset
- Plot confusion matrix

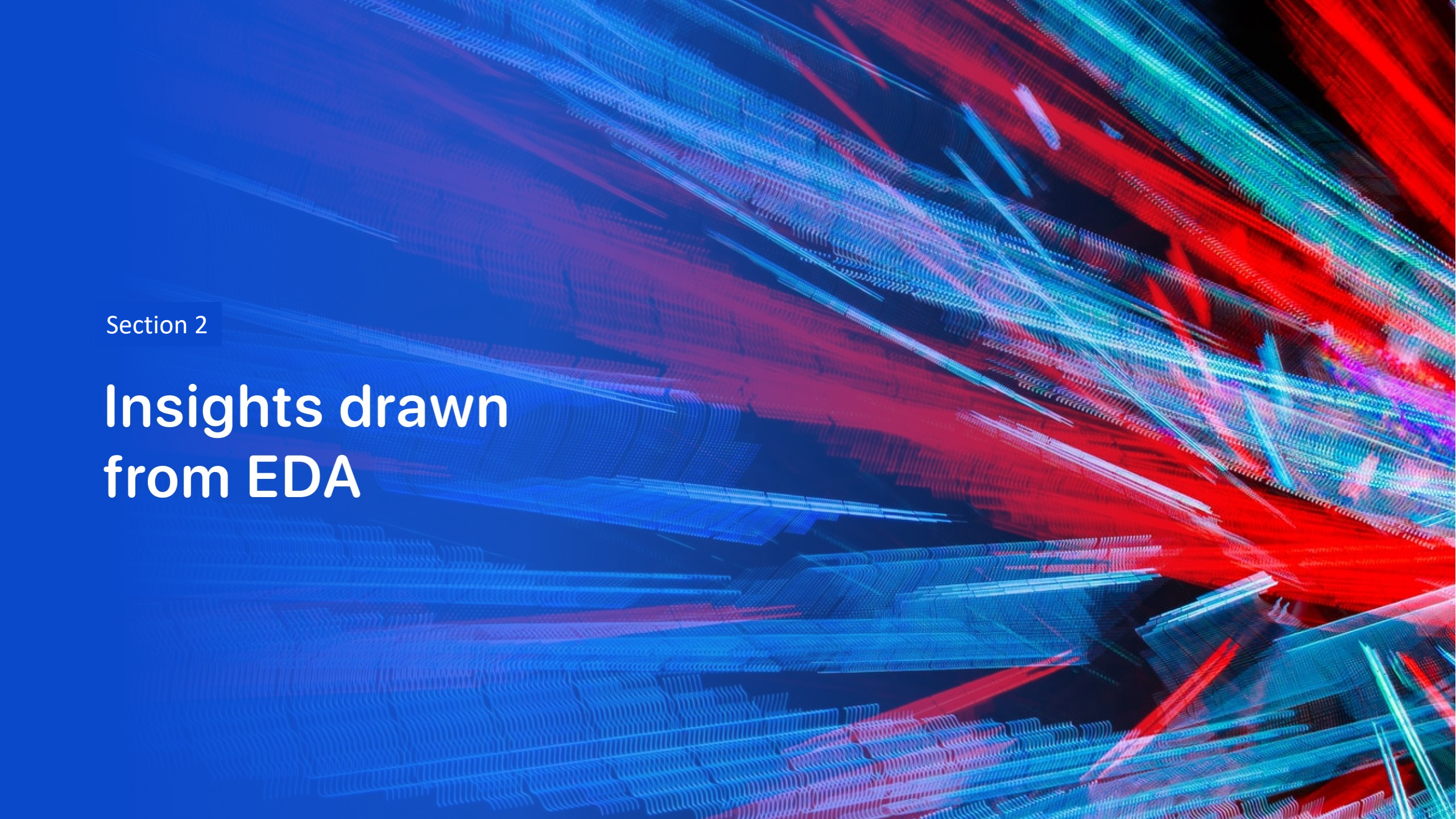
Model Comparison

- Compare model accuracy
- Choose best accuracy model
- See Notebook for further result

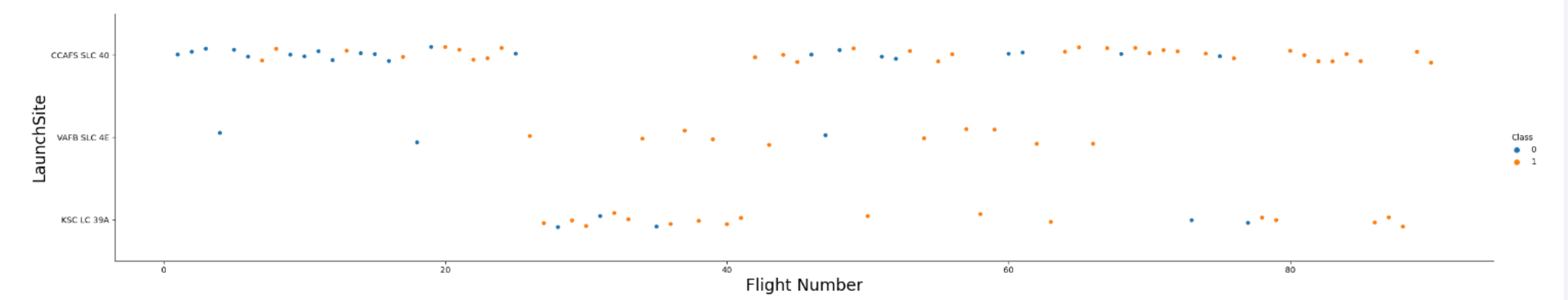
Link to Code

Results

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

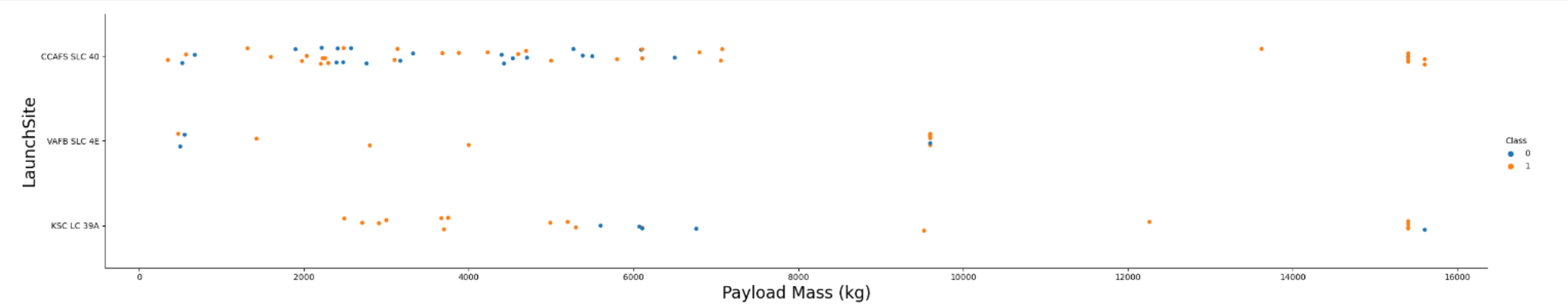

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



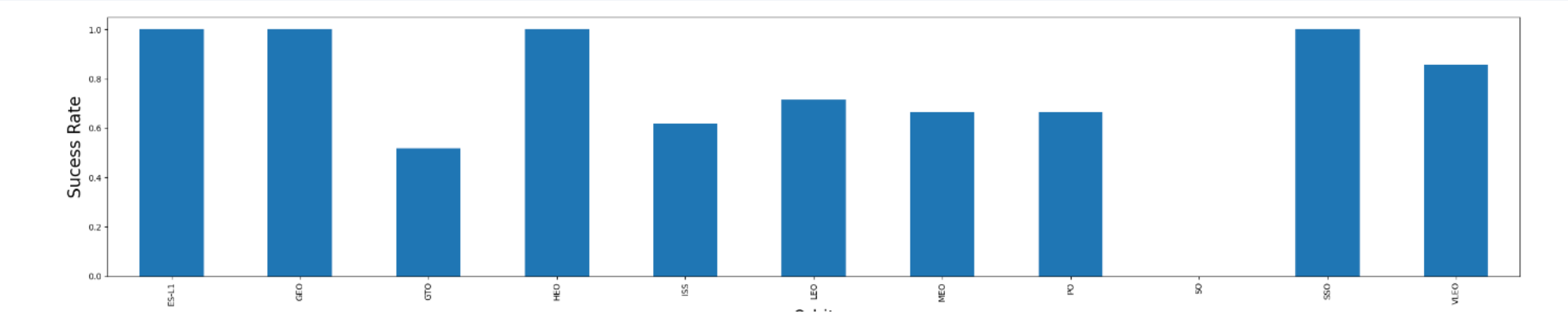
We notice that the success rate for each site is increasing

Payload vs. Launch Site



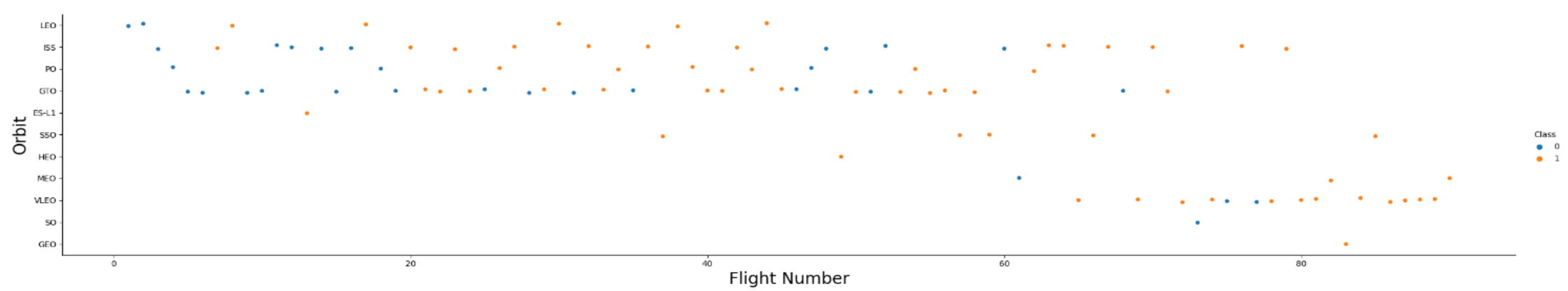
The success of a landing may depend on the weight of the payload and the launch site. While a heavier payload may increase the likelihood of a successful landing, it can also cause the landing to fail if it is too heavy.

Success Rate vs. Orbit Type



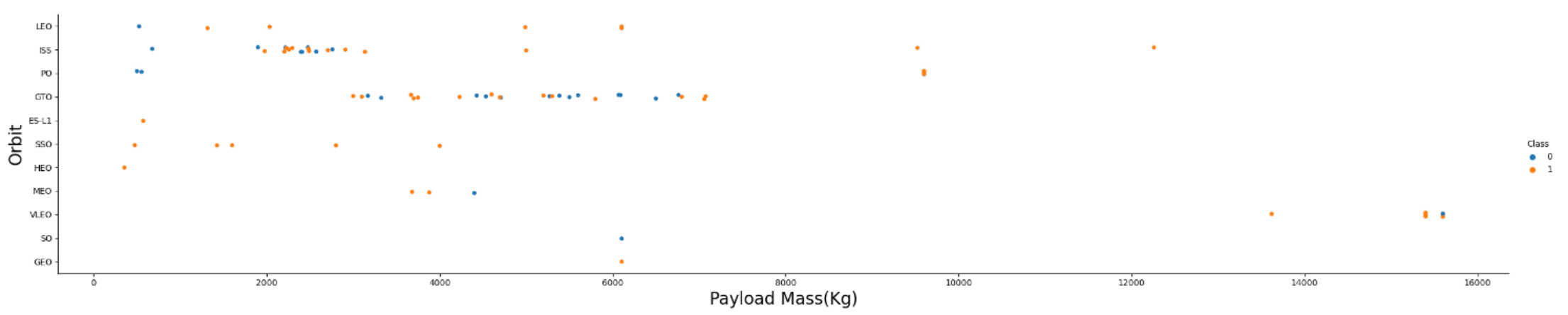
Plot shows success rate by orbit type, with ES-L1, GEO, HEO, and SSO having highest success rates.

Flight Number vs. Orbit Type



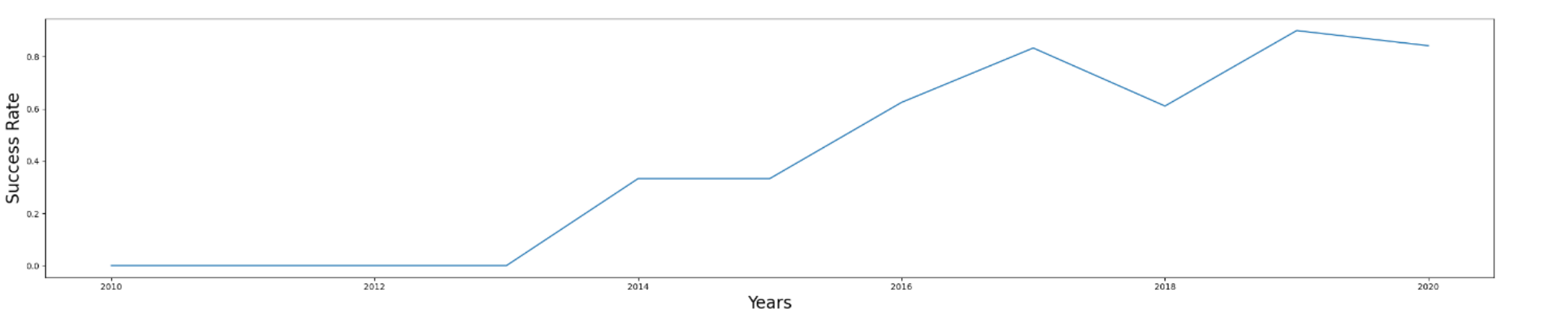
Success rate for LEO orbits increases with number of flights. For orbits like GTO, success rate not related to number of flights. High success rates of SSO, HEO possibly due to previous launch knowledge in other orbits.

Payload vs. Orbit Type



Payload weight influences success rate in certain orbits. For example,Heavier payloads better for LEO, lighter payloads better for GTO.

Launch Success Yearly Trend



Since 2013, we can see an increase in the Space X Rocket Success Rate

All Launch Site Names

SQL QUERY

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

RESULTS

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

Explanation

The DISTINCT keyword in the query is used to eliminate duplicate values from the results set for the LAUNCH_SITE column. This ensures that each unique value for LAUNCH_SITE is only returned once in the results.

Launch Site Names Begin with 'CCA'

SQL QUERY

```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5;
```

RESULTS

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Explanation

The WHERE clause in this query is used to filter the results based on the contents of the LAUNCH_SITE column. The LIKE clause is used to specify a pattern to match against the values in the LAUNCH_SITE column, in this case the substring 'CCA'. The LIMIT clause is used to limit the number of records returned in the results to 5. This means that the query will return a maximum of 5 records for launch sites that contain the substring 'CCA' in their name."

Total Payload Mass

SQL QUERY

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE customer = 'NASA (CRS)';
```

RESULTS

sum(PAYLOAD_MASS__KG_)
45596

Explanation

This query returns the sum of all payload masses where the customer is NASA(CRS)

Average Payload Mass by F9 v1.1

SQL QUERY

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where booster_version = 'F9 v1.1';
```

RESULTS

avg(PAYLOAD_MASS__KG_)
2928.4

Explanation

This query returns the average of all payload masses where the booster version contains the substring “F9 v1.1”

First Successful Ground Landing Date

SQL QUERY

```
%sql select min(DATE) from SPACEXTBL where Landing_Outcome = 'Success (ground pad)';
```

RESULTS

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: Landing_Outcome
[SQL: select min(DATE) from SPACEXTBL where Landing_Outcome = 'Success (ground pad)';
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Explanation

”Landing_Outcome” seems not to be running in the console

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL QUERY

```
%sql select BOOSTER_VERSION from SPACEXTBL where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

RESULTS

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: Landing_Outcome
[SQL: select BOOSTER_VERSION from SPACEXTBL where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000
and 6000;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Explanation

”Landing_Outcome” seems not to be running in the console

Total Number of Successful and Failure Mission Outcomes

SQL QUERY

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

RESULTS

missionoutcomes
1
98
1
1

Explanation

With the first SELECT, we show the subqueries that return results. The first subquery counts the successful mission. The second subquery counts the unsuccessful mission.

Boosters Carried Maximum Payload

SQL QUERY

```
%sql select booster_version, payload_mass__kg_ from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL)
```

RESULTS

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

EXPLANATION

In order to filter the data, we used a subquery that returns the maximum payload mass using the MAX function. The main query then uses the results of the subquery to return unique booster versions (using the SELECT DISTINCT clause) that are capable of carrying the heaviest payload mass.

2015 Launch Records

SQL Query and Results

```
%sql select count(landing_outcome), landing_outcome from SPACEXTBL where DATE between '2010-06-04' and '2017-03-20' group by landing__outcome orde
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: landing_outcome
[SQL: select count(landing_outcome), landing_outcome from SPACEXTBL where DATE between '2010-06-04' and '2017-03-20' group by landing__outcome or
der by count(landing__outcome) desc;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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

SQL Query and Results

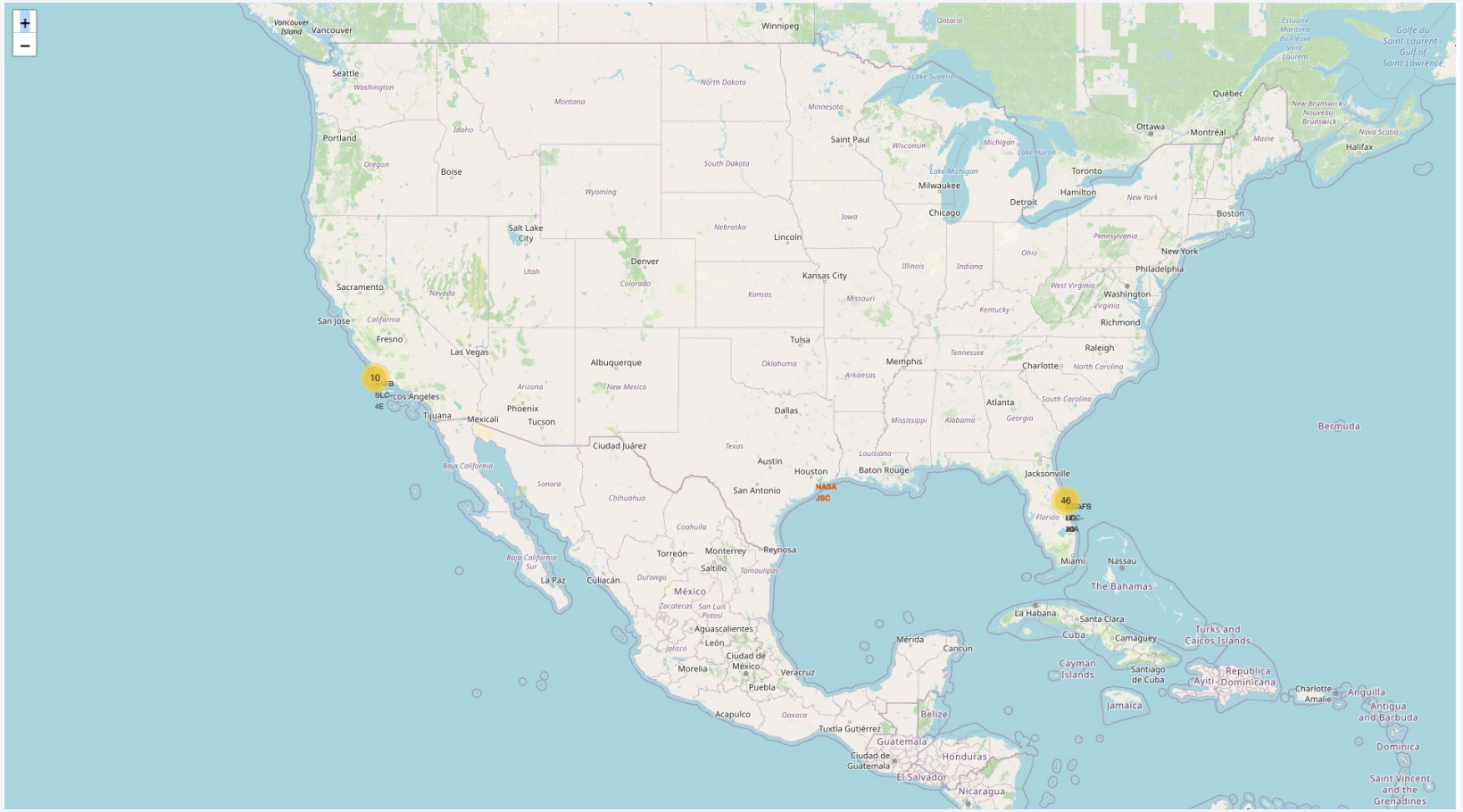
Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql select count(landing_outcome), landing_outcome from SPACEXTBL where DATE between '2010-06-04' and '2017-03-20' group by landing__outcome order by count(landing__outcome) desc;
```

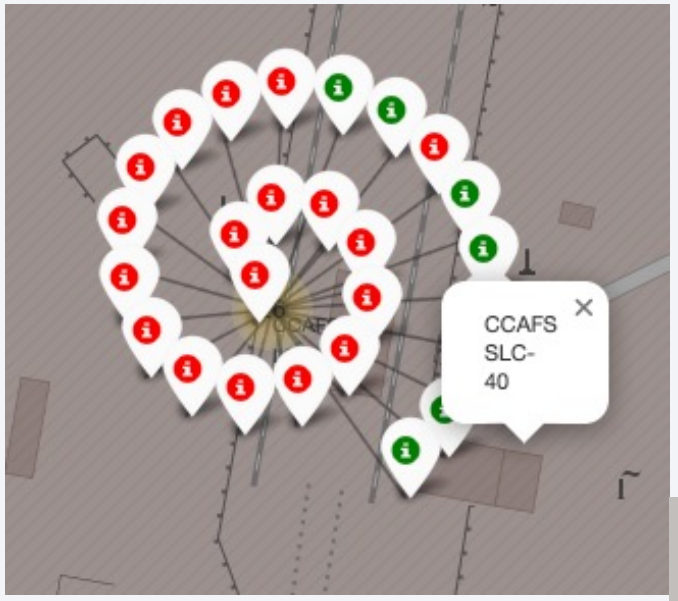
```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: landing_outcome
[SQL: select count(landing_outcome), landing_outcome from SPACEXTBL where DATE between '2010-06-04' and '2017-03-20' group by landing__outcome order by count(landing__outcome) desc;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

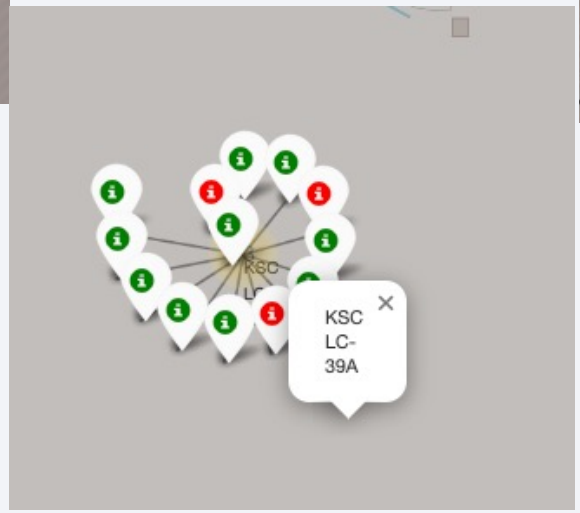

Section 3
Launch Sites
Proximities Analysis

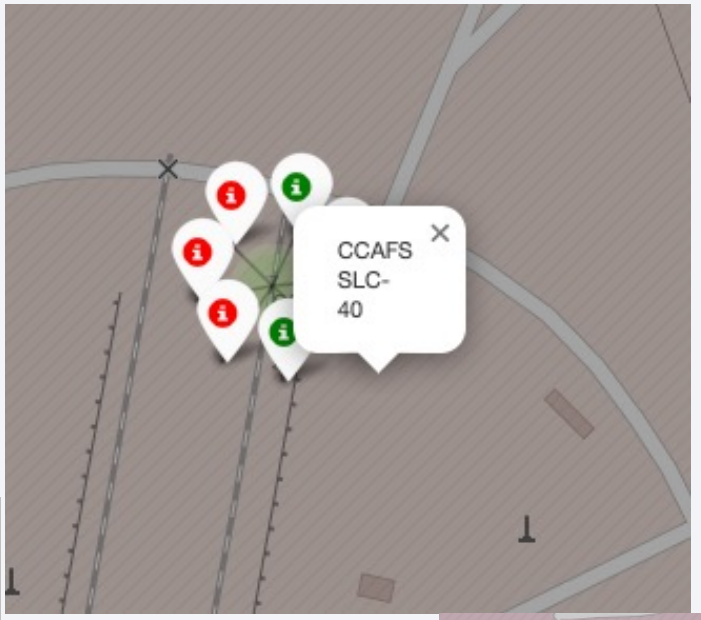
Folium Map – Ground Station & Launch Sites' location markers

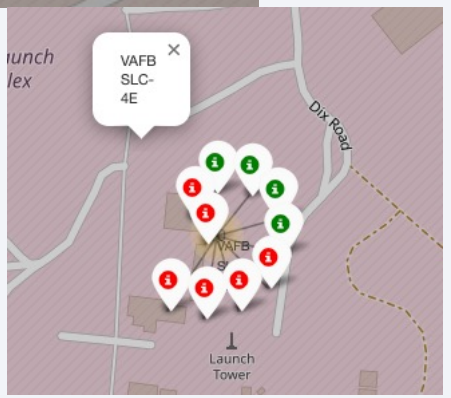


Folium Map – Color Labeled Markers



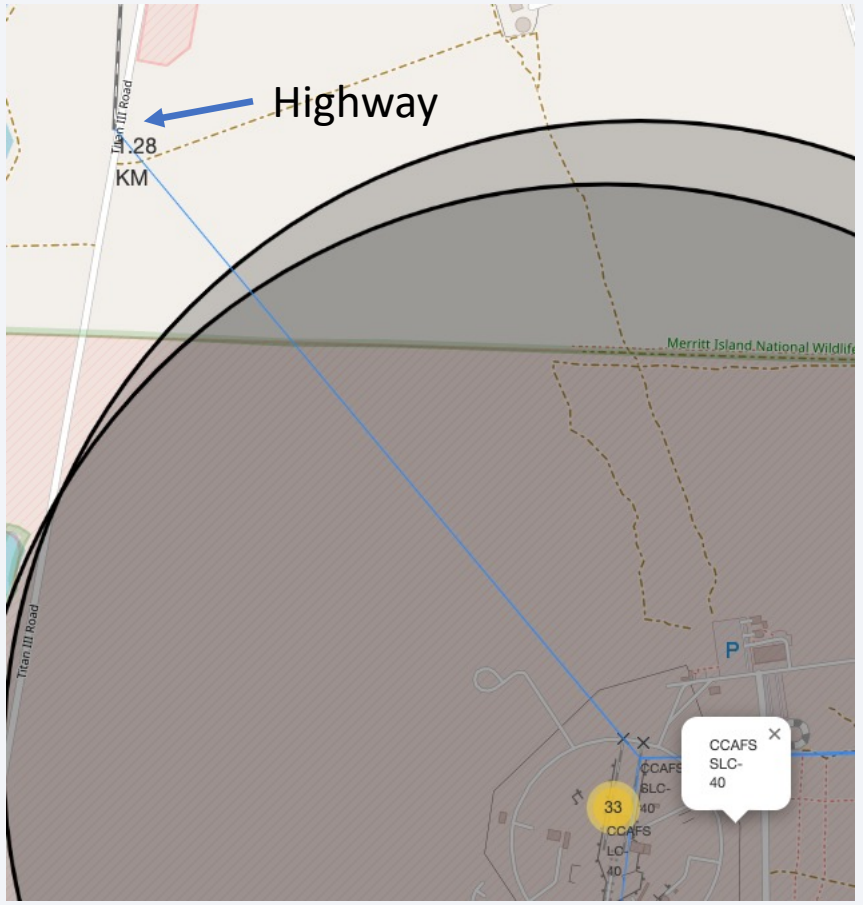




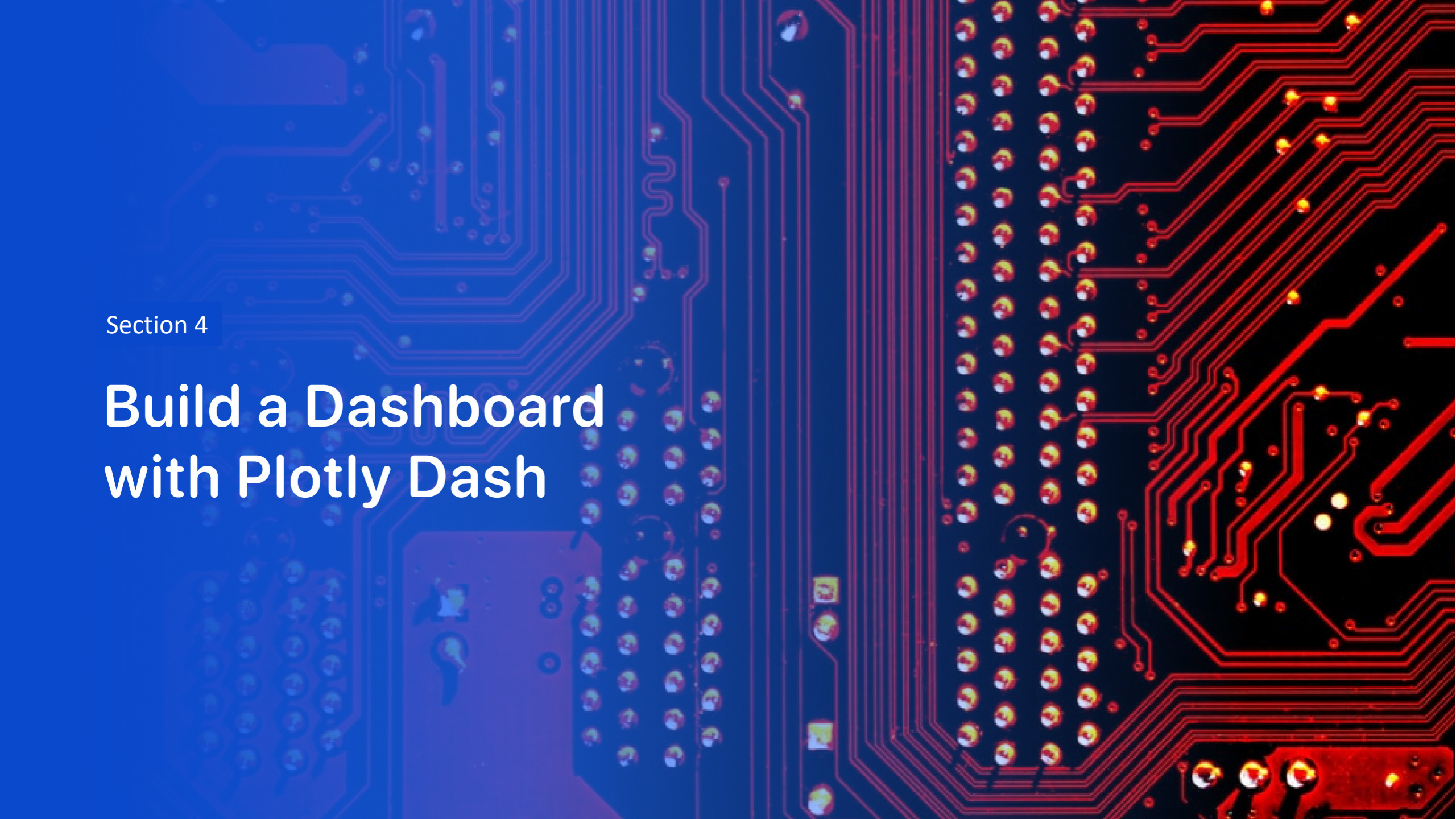


Green marker means successful launches. Red marker means unsuccessful launches. KSC LC-39A has a higher launch success rate

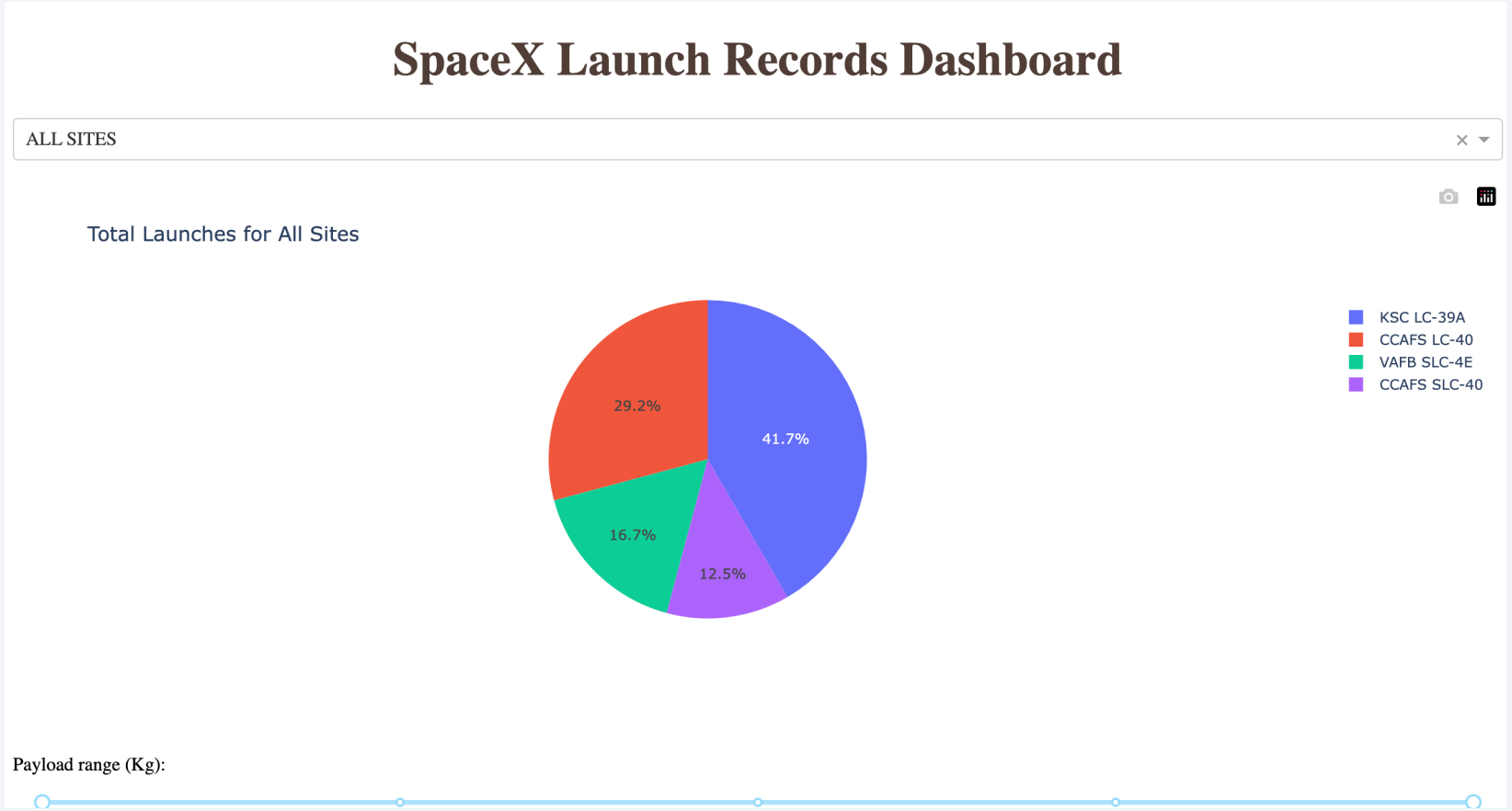
Folium Map – Distances between CCAFS SLC-40 and its proximities



Is CCAFS SLC-40 in close proximity to railways ? Yes
Is CCAFS SLC-40 in close proximity to highways ? Yes
Is CCAFS SLC-40 in close proximity to coastline ? Yes
Do CCAFS SLC-40 keeps certain distance away from cities ? No

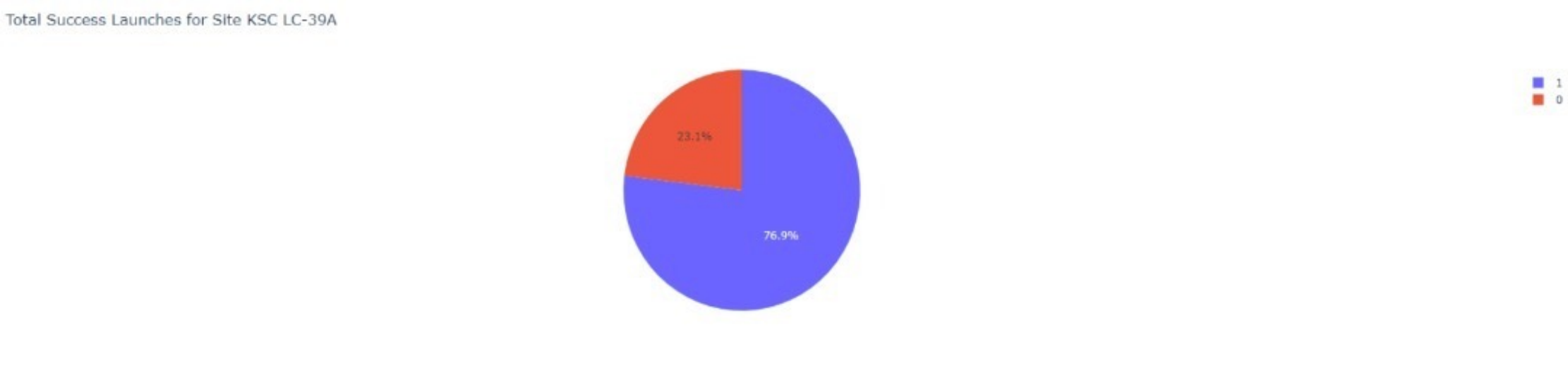

Section 4
Build a Dashboard with Plotly Dash

Dashboard – Total success by Site



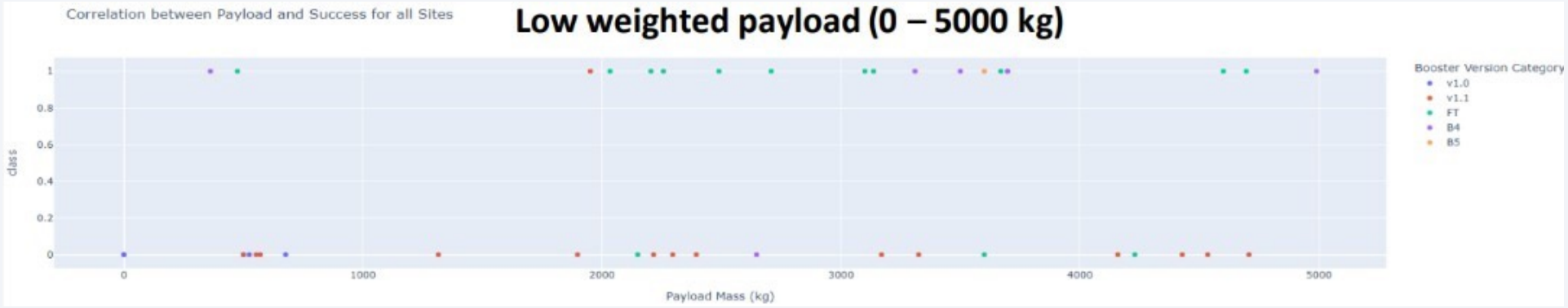
From the pie chart, LSC LC-93A has the best success rate of launches

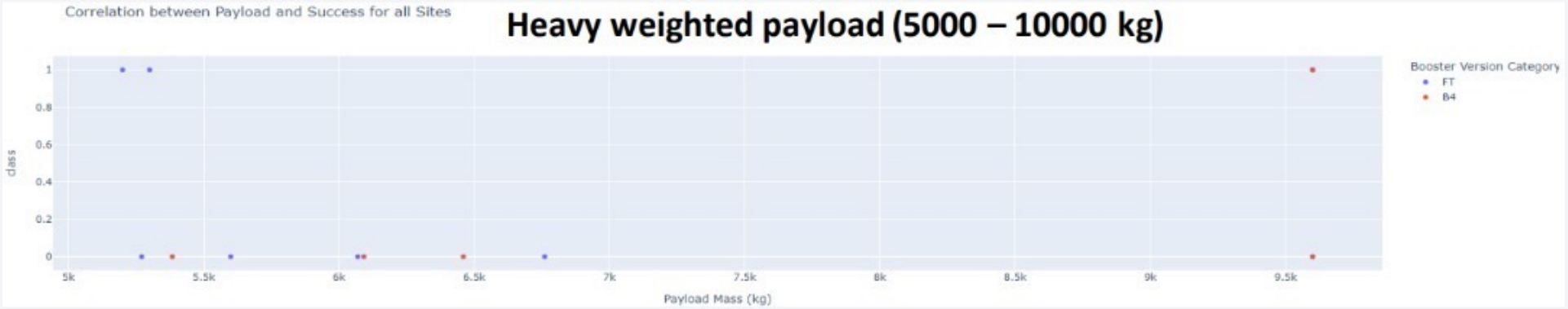
Dashboard – Total Success launch for KSC LC-39A



We see that KSC LC-39A achieved 76.9% success rate while getting 23.1% failure rate

Dashboard – Payload mass vs Outcome for all sites with different payload mass selected





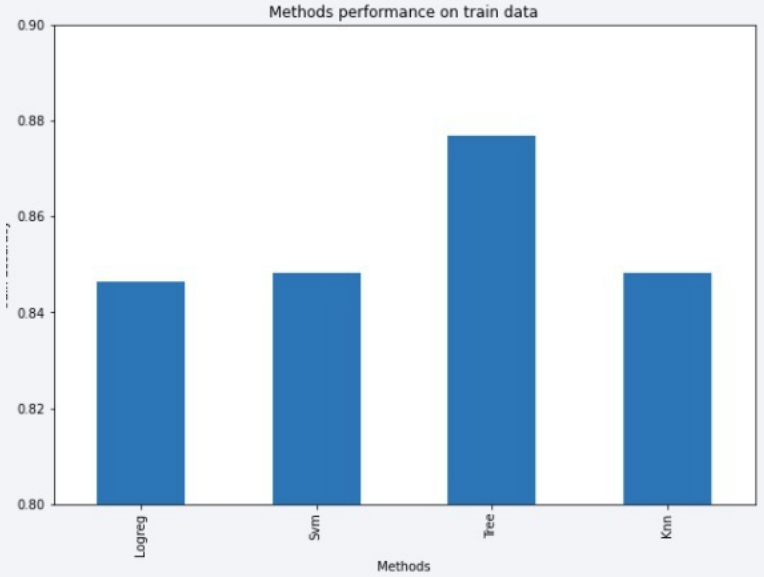
Low weighted payloads have better success rates as compared to heavy weighted payloads.

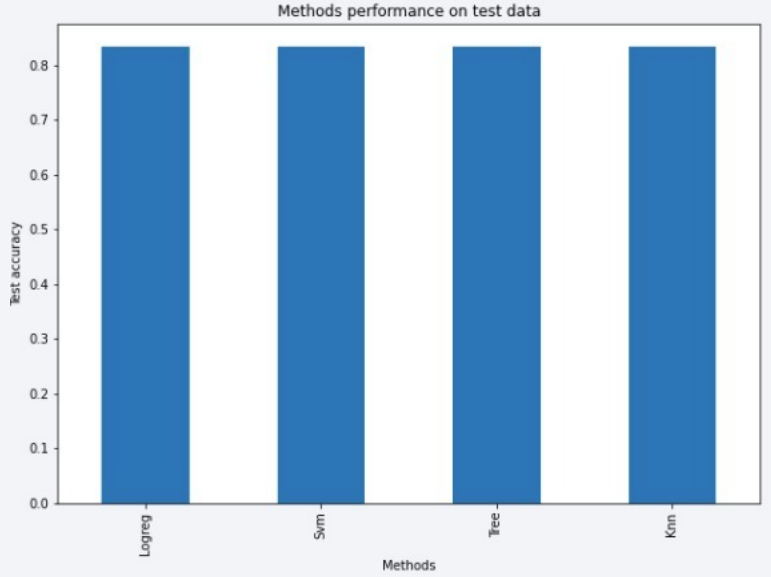
Section 5

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





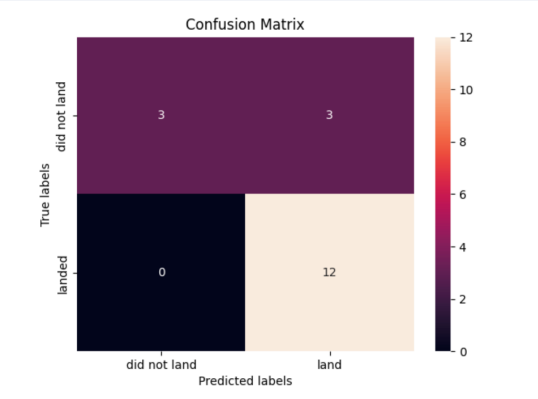
During the accuracy testing, all of the methods performed similarly. In order to confidently determine which method is most effective, it would be beneficial to gather more test data. However, if a decision must be made immediately, the decision tree method would be the most reasonable choice based on the available data.

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

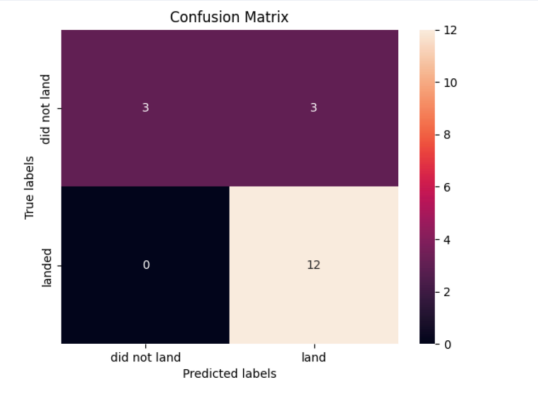
```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitte
r': 'random'}
accuracy : 0.8625
```

Confusion Matrix

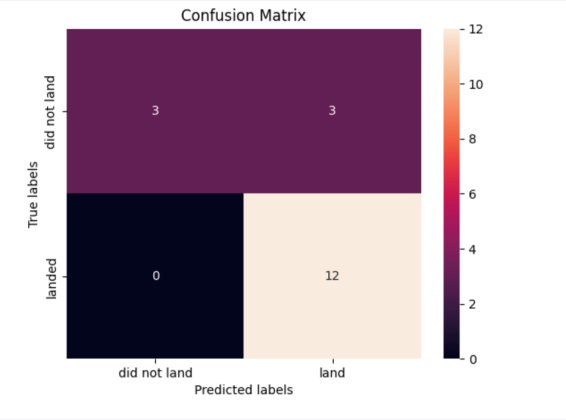
Logistic regression



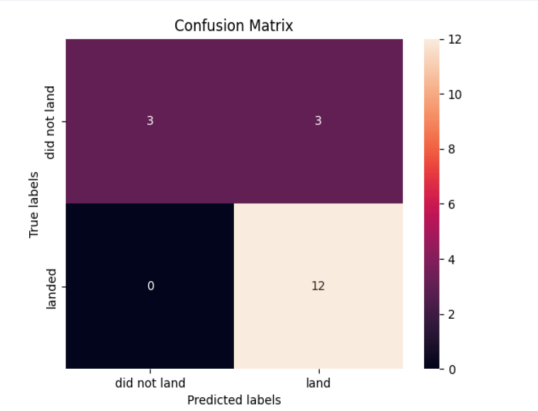
SVM



Tree



knn




```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.6666666666666666
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```

We can see the decision tree method is the best method

Conclusions

- Factors that can contribute to the success of a mission include the launch site, orbit, and number of previous launches.
- The most successful orbits are GEO, HEO, SSO, and ES-L1.
- Payload mass may also be a factor in mission success, with lighter payloads generally performing better than heavier payloads.
- It is unclear why some launch sites are more successful than others, and further data such as atmospheric data may be needed to understand this.
- The Decision Tree Algorithm was chosen as the best model for this dataset due to its higher train accuracy, even though all models had similar test accuracy.

Thank you!