IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ninth Narong
Jan 16 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API & Web Scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - Exploratory Data Analysis with SQL
  - Data Visualization with Folium
  - Interactive Dashboard with Plotly Dash
- Summary of all results
  - Exploratory Data Analysis Results
  - Interactive Maps and Dashboards
  - Predictive Analysis Results

Introduction

Project background and context

Due to technological advancements, commercial space travel is made possible as companies are competing to make it affordable for everyone. Despite competing with space giants such as Virgin Galatic, Rocket Lab and Blue origins, SpaceX came out on top. One major reason of their success is due to their inexpensive rocket launches. For comparison, SpaceX advertises that Falcon 9 rocket launches cost around 62 million dollars whereas other providers spend upwards of 165 million for each launch. The key to the significant savings is SpaceX's capability to reuse the first stage. As a potential competitor, Space Y, founded by Billionaire Allon Musk, will rely on their data science team to gather information about SpaceX and generate key business insights. SpaceY Data Science team's job is to determine the price of each launch, and instead of using rocket science, to train a machine learning model to predict SpaceX's first stage reusability.

Problems you want to find answers

- What characteristics define the success probability of a rocket landing?
- What are the correlations between the key characteristics and the success probability of the rocket landings?
- Will the trained model be able to predict the first stage reusability?

Section 1

Methodology

Methodology

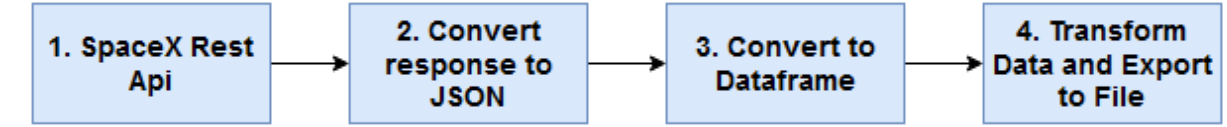
Executive Summary

- Data collection methodology:
  - API endpoints from Space X
  - Web scraping Wikipedia
- Perform data wrangling
  - Drop unnecessary columns
  - Deal with nulls
  - Joining datasets to replace IDs
  - Encode data to easily identify either success or failure
- 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

The sources of the dataset will be **SpaceX API** and **Wikipedia Web scraping**

Information about **rocket launches, launchpads** and **payloads** and will be obtained from the SpaceX REST API endpoints which start with: api.spacexdata.com/v4/



Less Information about **rocket launches** and **payloads** can be obtained from webscraping the Wikipedia page: https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Data Collection – SpaceX API

1. Request data from APIs

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
[42]: spacex_url="https://api.spacexdata.com/v4/launches/past"

[43]: response = requests.get(spacex_url)
```

2. Convert response to JSON

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

3. Transform Data

```
[20]: # Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

```
[21]: BoosterVersion[0:5]

[21]: ['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

```
[22]: # Call getLaunchSite
getLaunchSite(data)

[23]: LaunchSite[0:5]

[23]: ['Kwajalein Atoll',
 'Kwajalein Atoll',
 'Kwajalein Atoll',
 'Kwajalein Atoll',
 'CCSFS SLC 40']

[24]: # Call getPayloadData
getPayloadData(data)

[25]: # Call getCoreData
getCoreData(data)
```

4. Transform Data

```
[26]: 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

```
# Create a data from Launch_dict

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

6. Filter Dataframe

```
# Hint data['BoosterVersion']!='Falcon 1'

data_falcon9 = newData[newData['BoosterVersion']=="Falcon 9"]
data_falcon9.head()
```

Link

Data Collection - Scraping

1. Obtain data from Wikipedia via webscraping

```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
```

2. Convert to a Beautifulsop object

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(response.text, "html.parser")
```

3. Find all tables

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

4. Get all columns from Table 3

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
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 empty 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. Populate dictionary from beautiful soup object (please refer to notebook for code)

7. Create dataframe from dictionary

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

LINK

Data Wrangling

1. Calculate number of launches on each site

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

2. Calculate number and occurrence of each orbit

```
# Apply value_counts on Orbit column
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. Calculate number and occurrence of mission outcome per orbit type

```
# landing_outcomes = values on Outcome column
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. Classify landing outcome

```
tempOutcome = df['Outcome']
i = 0

while i < tempOutcome.count():
    if tempOutcome.iloc[i] in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
    i+=1
```

5. Export to file

LINK

EDA with Data Visualization

Scatter Point Plots

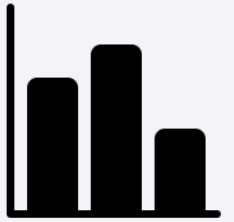
Used to show the relationship between two variables

- Flight Number and Payload
- Flight number vs. Launch Site
- Payload vs. Launch Site
- Flight Number and Orbit Type
- Payload vs. Orbit Type

Bar Chart

Used to show the relationship between numerical and categorical variables

- Success Rate vs. Orbit Type



Line Chart

Used to show data variables and their trends

- Success Rate vs. Year

EDA with SQL

- Display the names of the unique launch sites in the space mission
- 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 succesful landing outcome in ground pad was acheived.
- 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 which have carried the maximum payload 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

- Create a Red circle at NASA Johnson Space Center's coordinate with an icon and a popup label showing its name (circle, marker)
- Create Red circles for all the launch sites and labeling their names (circle, marker)
- Add markers to show successful (green) and unsuccessful (red) landings (marker).
- Cluster markers by each site (MarkerCluster())
- Lines to show distance between launch site to key locations (polyline)

Many factors contribute to a successful launch. These could be payload mass, orbit type, and so on, but it can also depend on the location and proximities of a launch site. By building an interactive map with Folium, users can now better see the launch sites, surroundings, number of successful and unsuccessful landings.

Link to code

Build a Dashboard with Plotly Dash

- Components used in the Dashboard include: **dropdown**, **pie chart**, **rangeslider,** and **scatter plot**.
- **Dropdown** allows user to visualize data between launch sites, or from all sites
- **Pie chart** is used to easily visualize the total success and total failure of selected sites
- **Rangeslider** allows users to change the payload mass and observe the data in the scatter plot
- **Scatter Plot** shows the relationship between the success rate and the payload mass

Link to code

Predictive Analysis (Classification)

Prepare Data

- Load dataset
- Standardize dataset
- Split dataset into training and test data

Prepare Model

- Select machine learning algorithms
- Create a GridSearchCV object
- Fit GridSearchCV with training data

Evaluate Model

- Find best parameters for each model
- Compute accuracy
- Plot confusion matrix

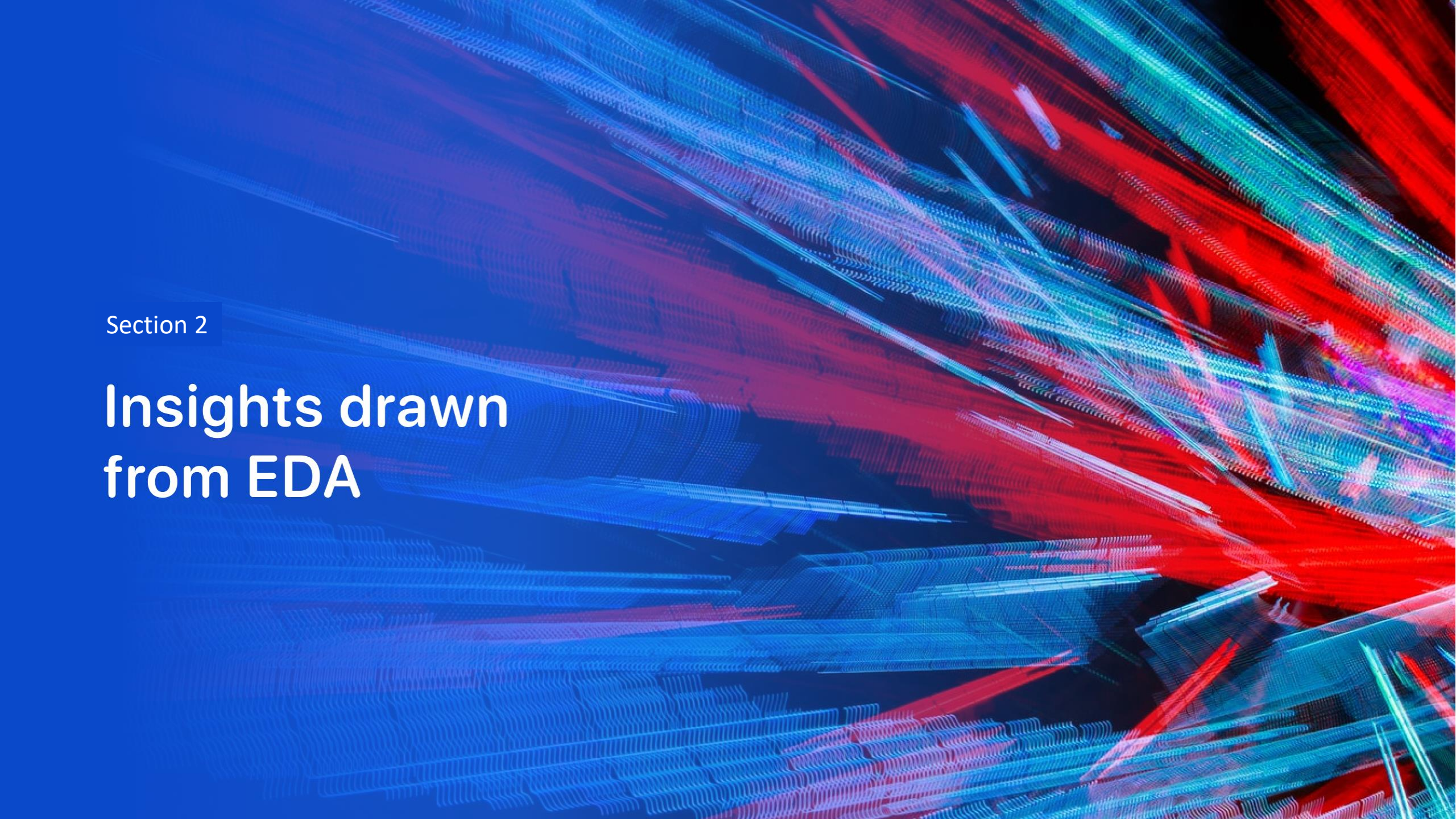
Compare Model

- Compare accuracy of models
- Select model with best accuracy

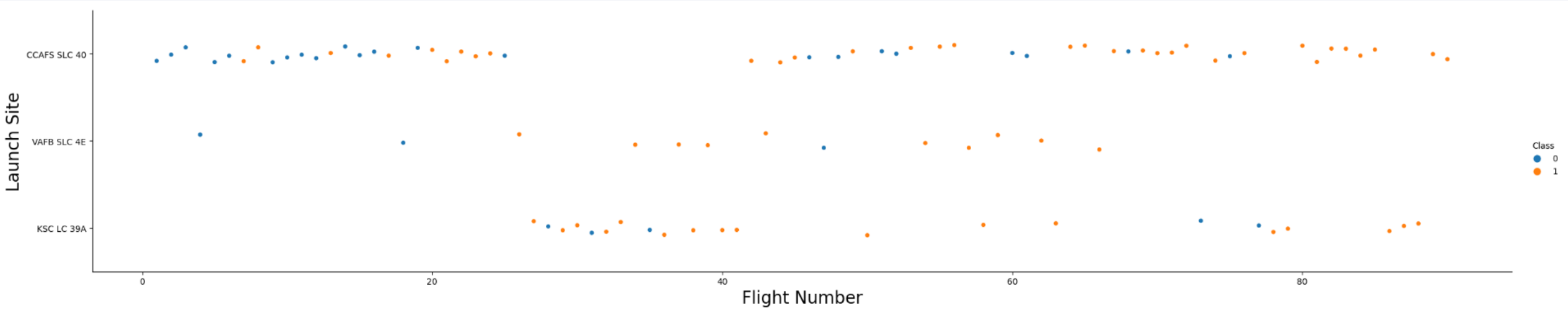
Link to code

Results

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

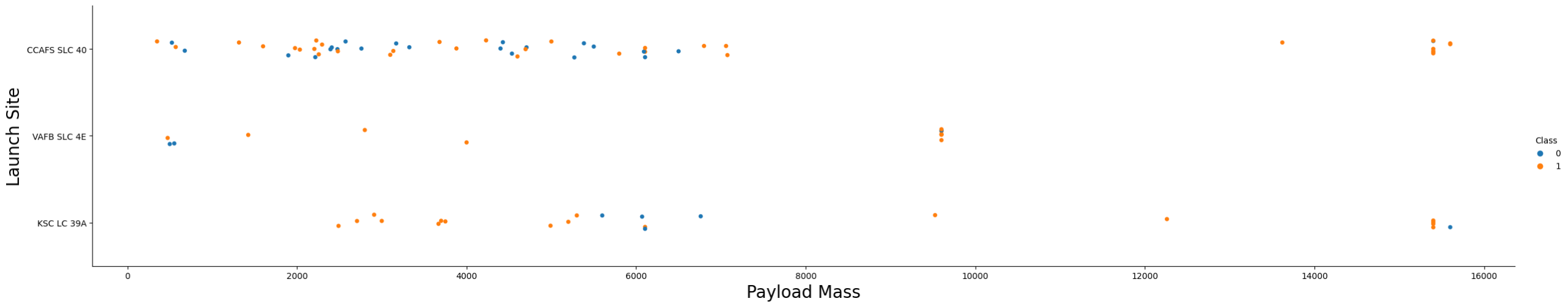

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



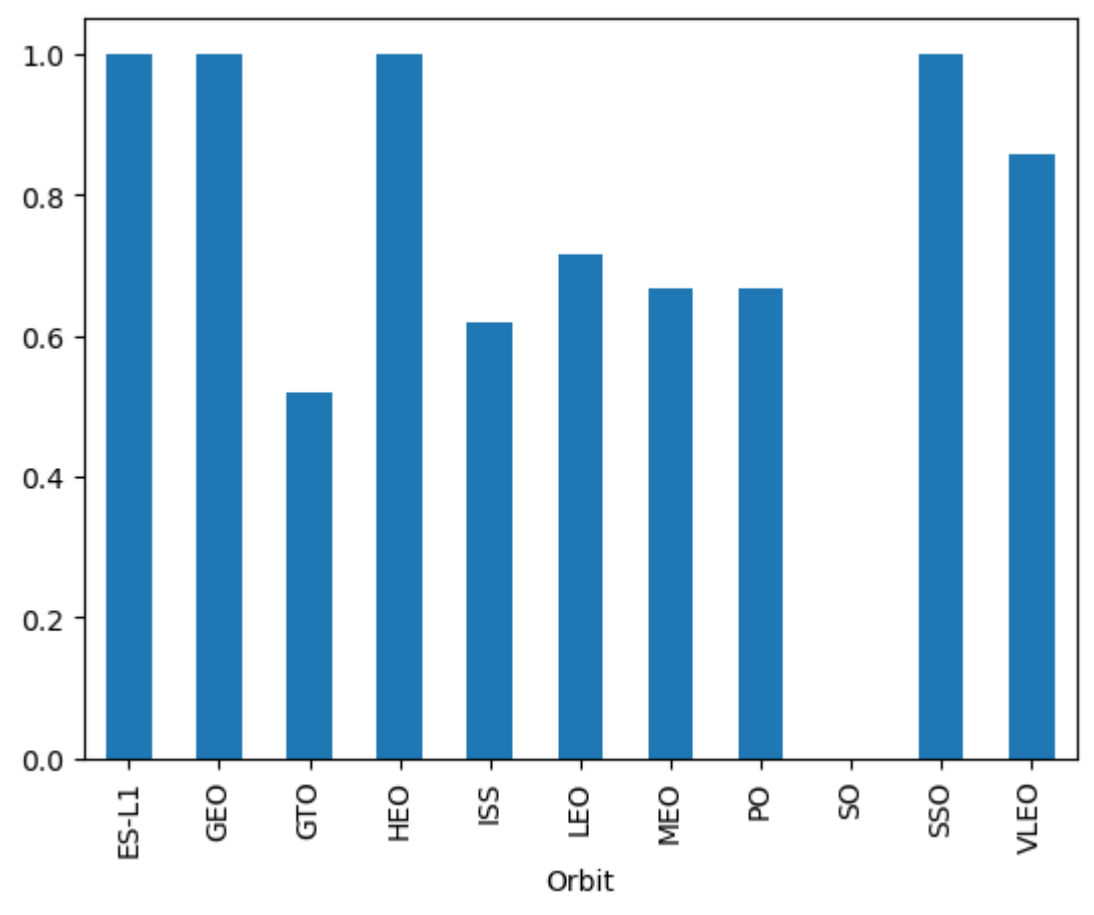
We observe that for each launch site, success increases as the flight number increases

Payload vs. Launch Site



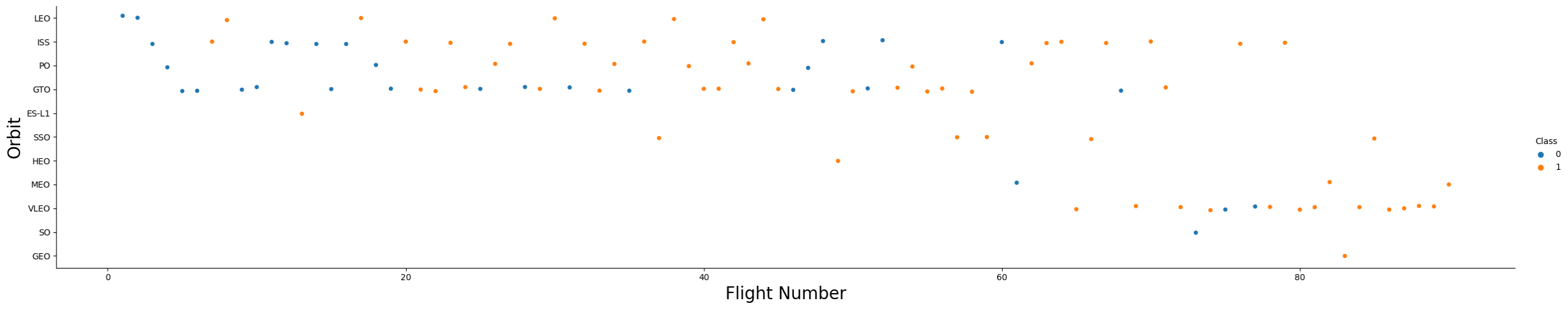
We observe that for all the launch sites, there seems to be a high chance of a successful landing if the payload is above **10,000** kg

Success Rate vs. Orbit Type



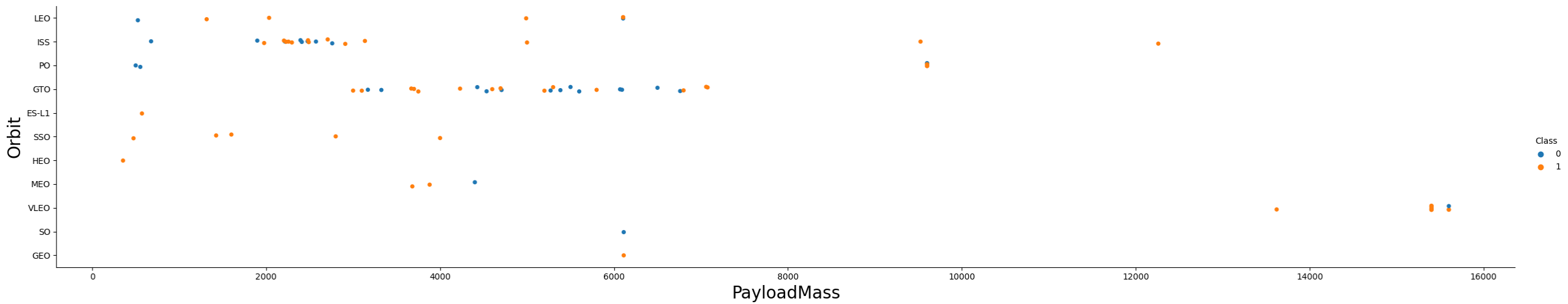
This graph allows us to observe the success rate of each Orbit. It seems like ES-L1, GEO, HEO, SSO have the **highest** success rate.

Flight Number vs. Orbit Type



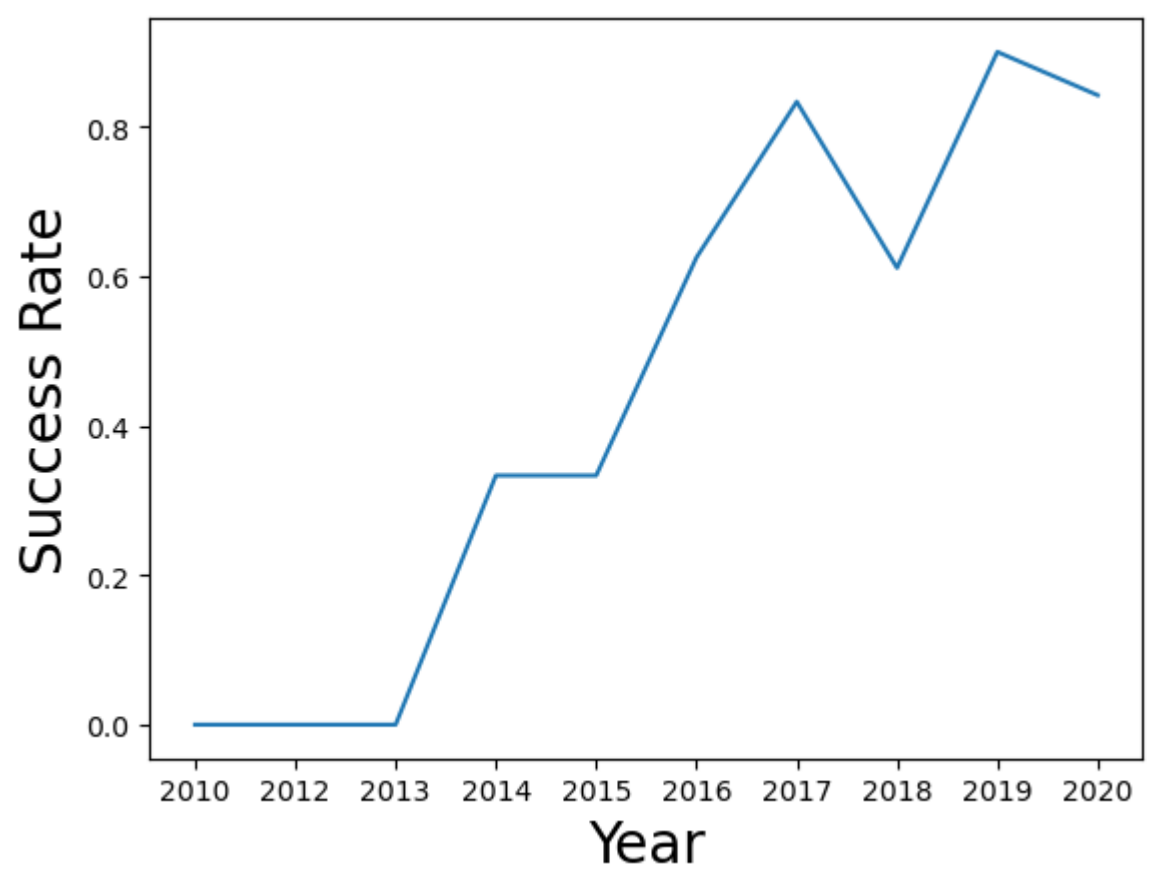
We can observe that certain orbits have more tests than the others. Despite certain orbits (GTO) showing no correlation between **success rate** and **flight number,** it can be assumed that **success rate increases as the number of flight increases** (SSO or HEO).

Payload vs. Orbit Type



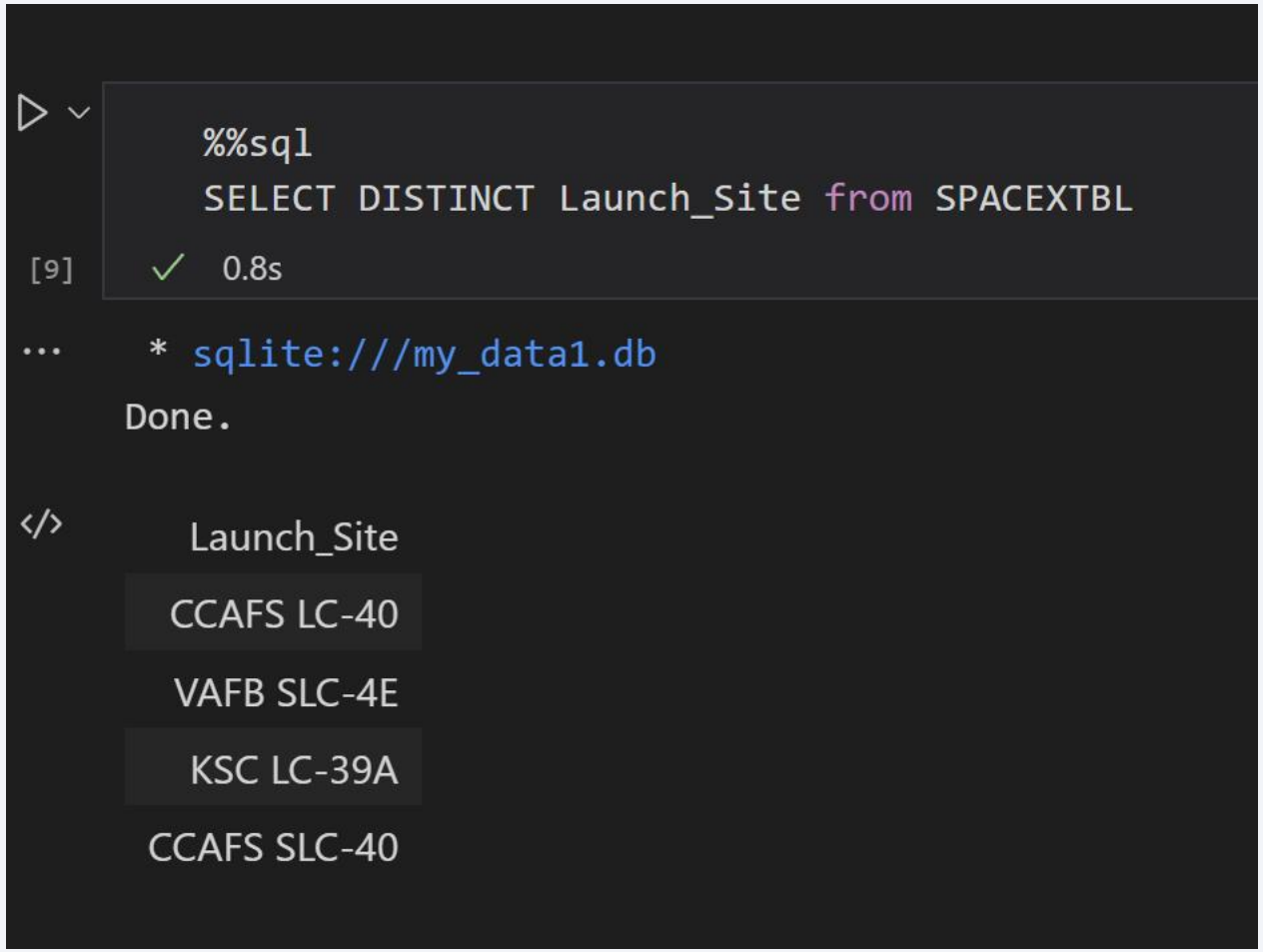
For most orbits, increasing the payload mass increases the success rate. However, for GTO, it can be observed that decreasing the payload mass improves its success rate.

Launch Success Yearly Trend



There has been significant improvement on successful landings over the years.

All Launch Site Names



'SELECT DISTINCT' QUERY allows users to retrieve only the unique records from a dataset

Launch Site Names Begin with 'CCA'

```
%%sql
Select * from SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

✓ 0.6s

0	04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Suc
1	08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Suc
2	22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Suc
3	08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Suc
4	01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Suc

WHERE clause allows users to set conditions, and **LIMIT** clause specifies the number of records to be retrieved

Total Payload Mass

```
%%sql
SELECT sum(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE Customer == 'NASA (CRS)'
```

✓ 0.7s

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

sum(PAYLOAD_MASS__KG_)
45596

SELECT SUM (column) can be used total all the numbers (compatible data)in a column

Average Payload Mass by F9 v1.1

```
%%sql
SELECT avg(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE Booster_Version == 'F9 v1.1'
```

✓ 0.1s

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

avg(PAYLOAD_MASS__KG_)
2928.4

Similarly, AVG can be used to calculate the average of a column, if the data is compatible.

First Successful Ground Landing Date

```
%%sql
SELECT min("DATE") from SPACEXTBL WHERE "Landing _Outcome" LIKE '%Success%'
```

✓ 0.1s

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

min("DATE")
01-05-2017

First, we find the records that contains “Success”, which can be done through the use of **LIKE.**
MIN will filter out the earliest date.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%%sql
SELECT "Booster_Version" from SPACEXTBL
WHERE
"Landing _Outcome" LIKE '%Success%' AND
"PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000
```

✓ 0.1s

We can specify which column to retrieve after SELECT and add one or more conditions with AND. In this case, LIKE and BETWEEN are used to retrieve our data

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 B4 B1043.1
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1046.3
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

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
```

✓ 0.1s Python Python Python Python Python

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

SUCCESS	FAILURE
100	1

We must use 2 subqueries to retrieve the result. COUNT is used to retrieve the total number, and LIKE is used to retrieve the records of Successful or Failure. We combine the 2 subqueries using another SELECT.

Boosters Carried Maximum Payload

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

✓ 0.4s

A subquery with MAX is used to obtain **the maximum payload mass**, then SELECT DISTINCT is used to ensure that only **unique booster versions** will be shown.

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  substr(Date, 4, 2) as month,
'Landing _Outcome',
Booster_Version,
Launch_Site

FROM SPACEXTBL
WHERE substr(Date,7,4)='2015'
```

month	'Landing _Outcome'	Booster_Version	Launch_Site
01	Landing _Outcome	F9 v1.1 B1012	CCAFS LC-40
02	Landing _Outcome	F9 v1.1 B1013	CCAFS LC-40
03	Landing _Outcome	F9 v1.1 B1014	CCAFS LC-40
04	Landing _Outcome	F9 v1.1 B1015	CCAFS LC-40
04	Landing _Outcome	F9 v1.1 B1016	CCAFS LC-40
06	Landing _Outcome	F9 v1.1 B1018	CCAFS LC-40
12	Landing _Outcome	F9 FT B1019	CCAFS LC-40

To query Date data, SUBSTR will be used. SUBSTR(DATE,4,2) is used to extract Month while SUBSTR(DATE,7,4) is used to extract Year.

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

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

75] ✓ 0.4s

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

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

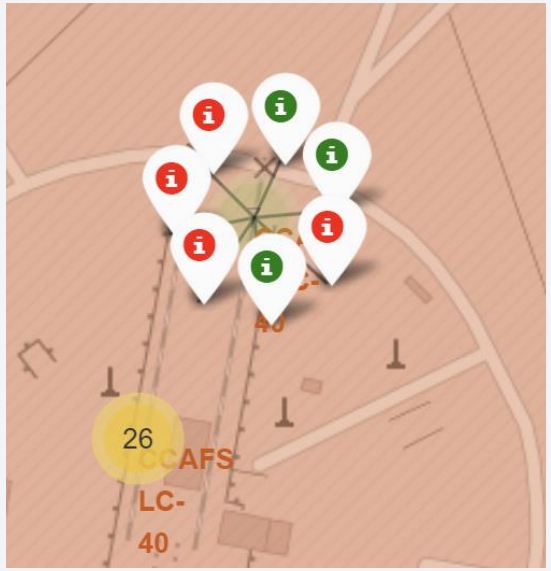
This query returns count of successful landings between 04/06/2010 and 20/03/2017. The GROUP BY clause groups records by landing outcome and ORDER BY COUNT DESC shows records in decreasing order


Section 3
Launch Sites
Proximities Analysis

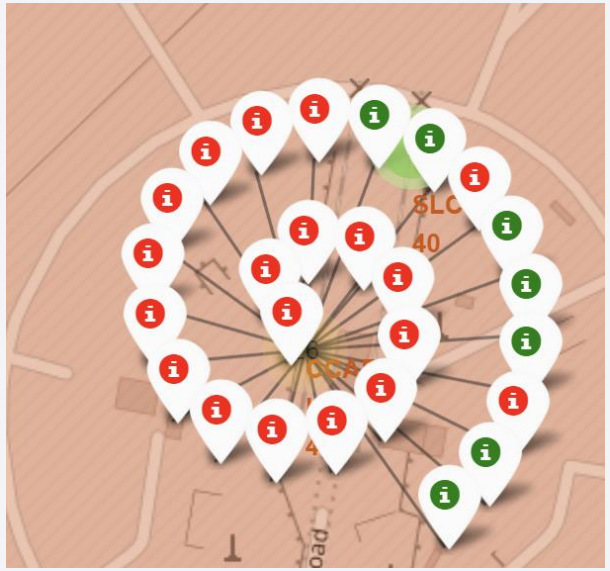
Launching Sites

It seems like the launching sites are placed on the **coasts** in the United States.

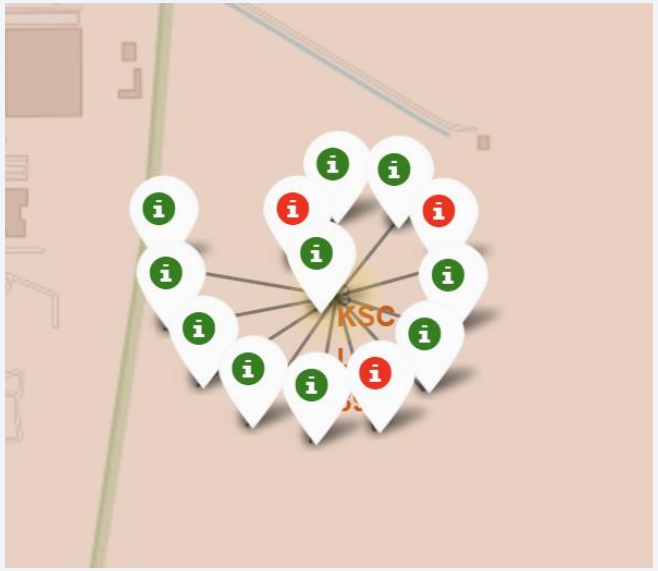
Clusters of Landing Results



CAFSSLC-40



CCAFSLC-40

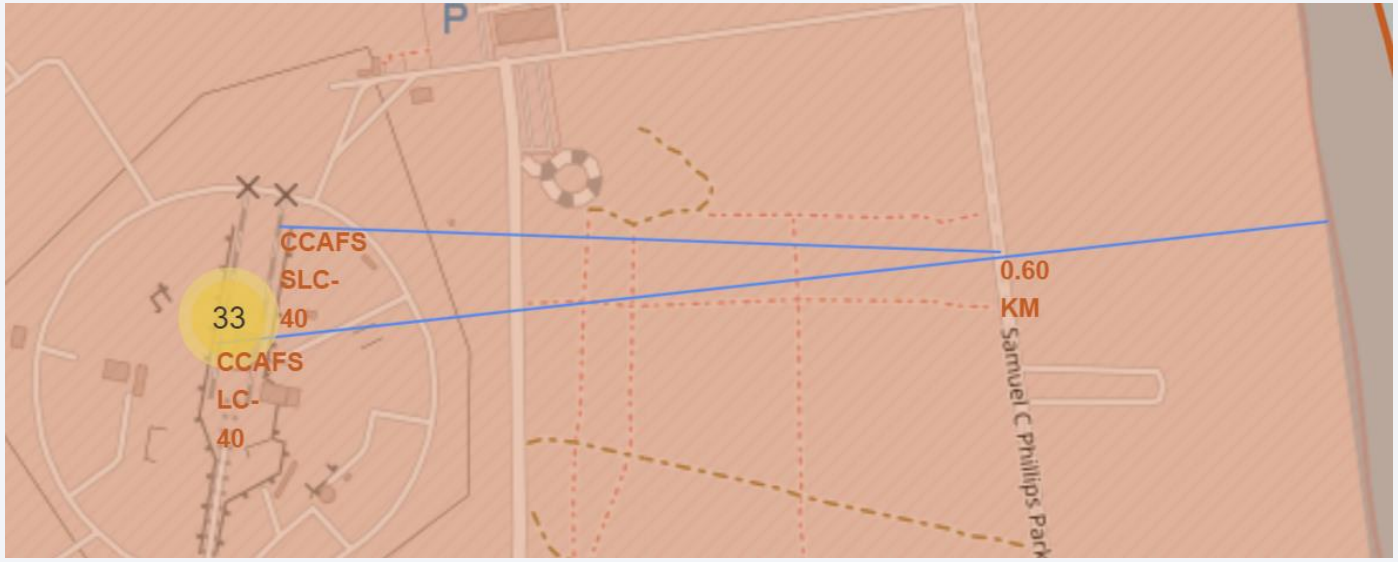


KSCLC-39A

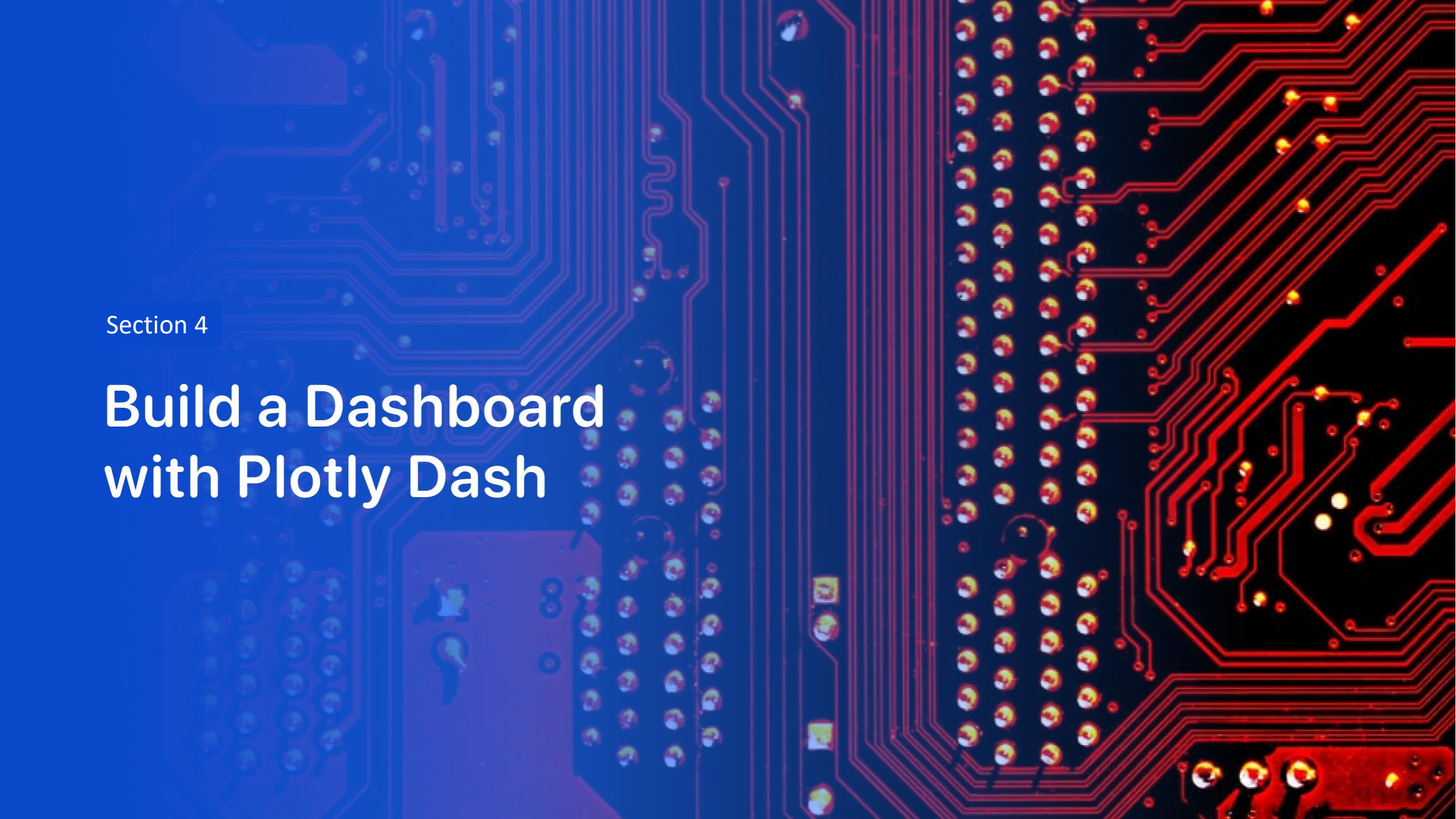
VAFESLC-4E

The Green Markers show successful landings whereas the Red Markers represent failed landings.

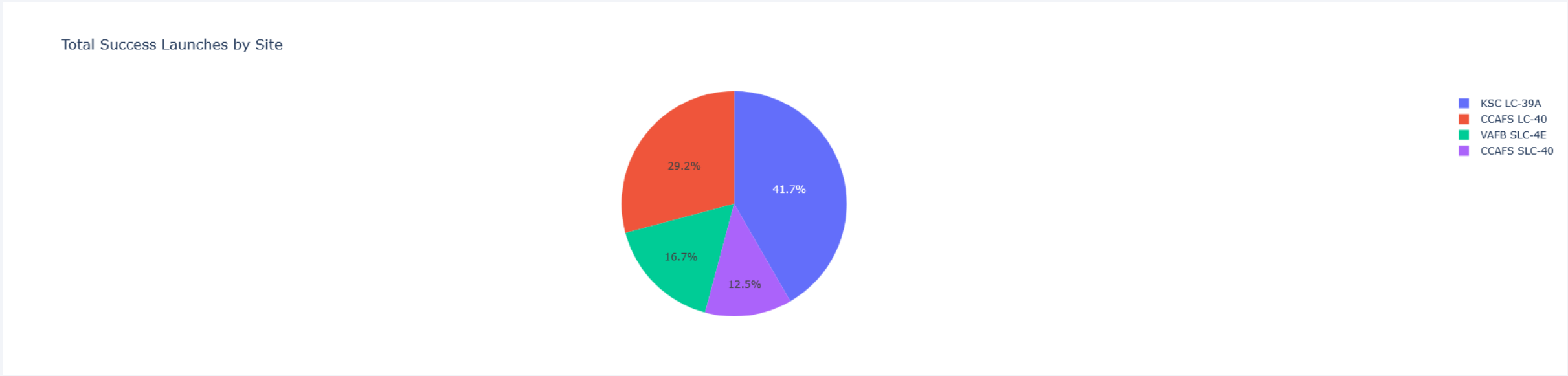
Distances between CCAFS SLC-40 and its proximities



Is CCAFS SLC-40 in close proximity to railways ? Yes
Is CCAFS SLC-40 in close proximity to highways ? Could not Mark
Is CCAFS SLC-40 in close proximity to coastline ? Yes
Do CCAFS SLC-40 keeps certain distance away from cities ? Could not Mark

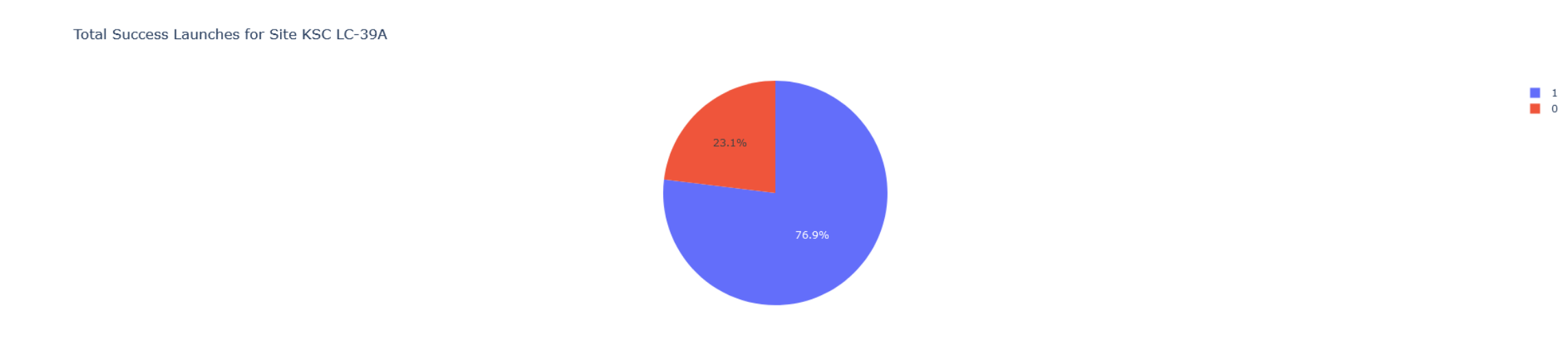

Section 4
Build a Dashboard with Plotly Dash

Success Rate by Launch Sites



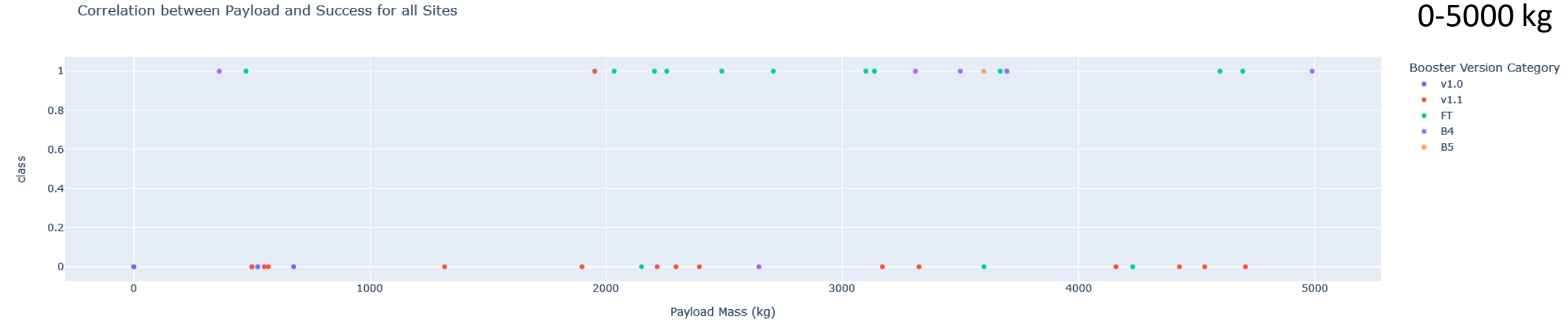
KSC LC-39A has the best success rates

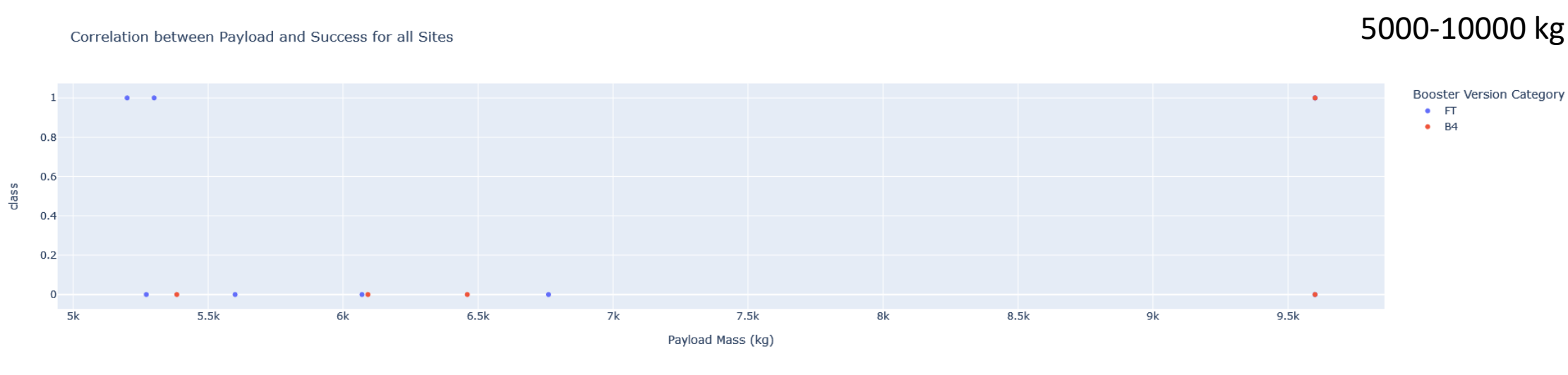
Observations on Site KSC LC-39A



KSC LC-39A has achieved a 76.9% success rate and a 23.1% failure rate.

Correlation between Payload and Success by Launch Sites at Different Weight Ranges

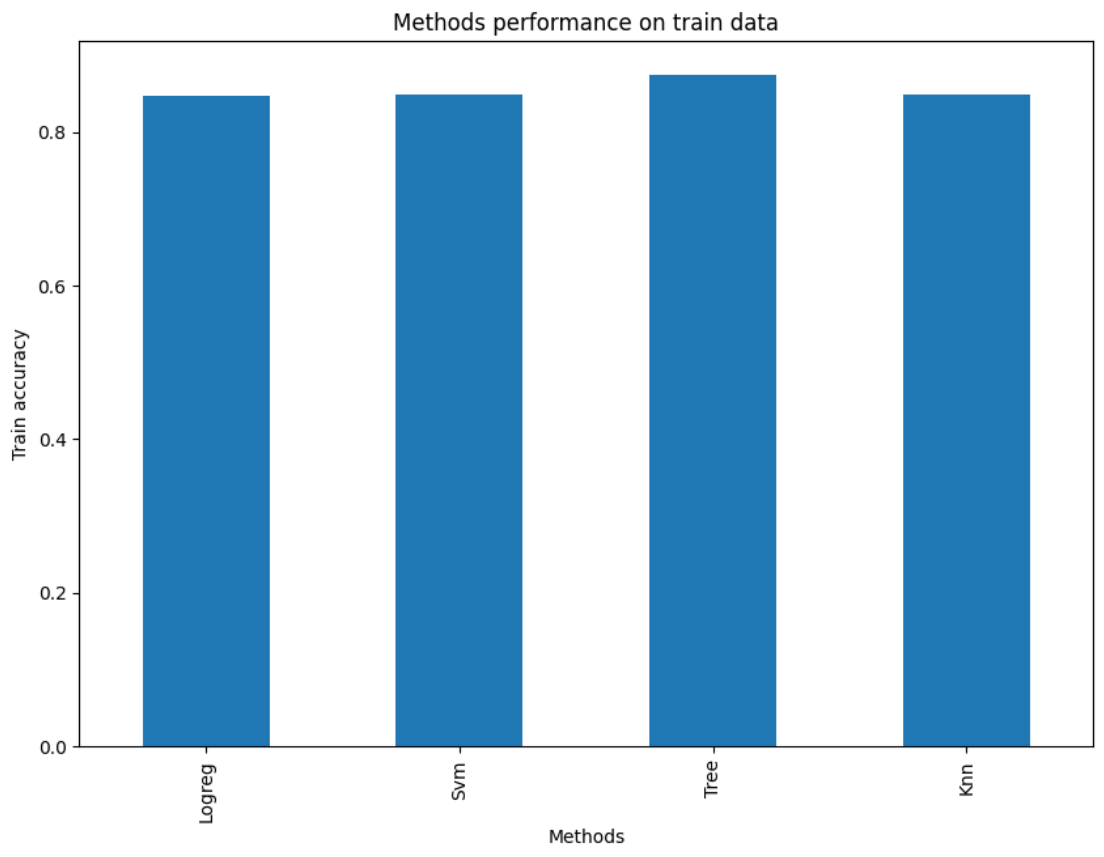




Section 5

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	0.861111

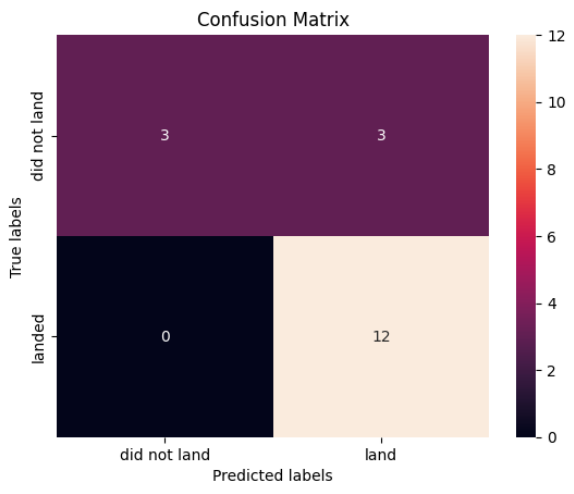
All models were quite accurate, but the best model should be Tree with 87.5%.

The best parameters are:

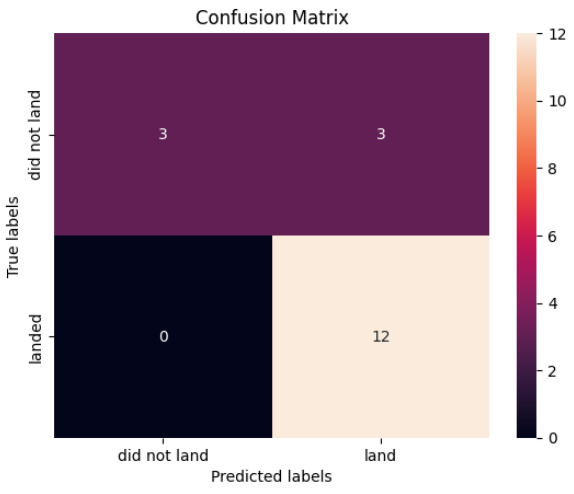
```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto',
'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.875
```

Confusion Matrix

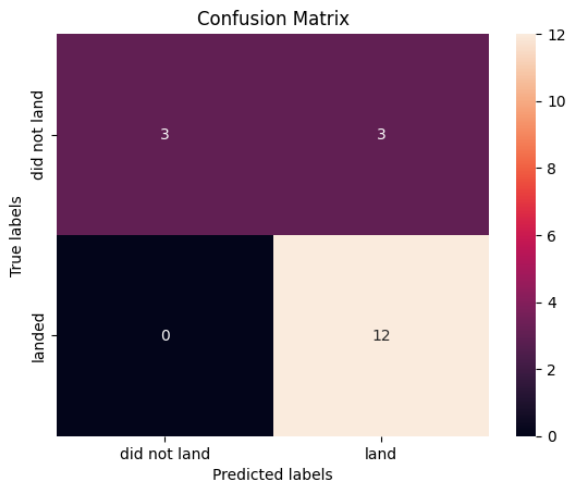
Logistics Regression



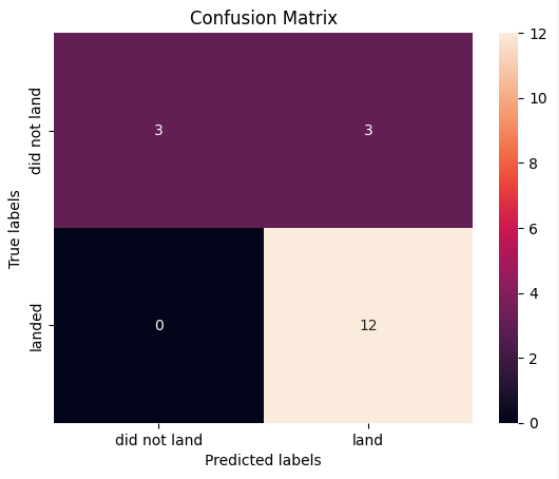
Decision Tree



SVM



KNN



Seems like the Confusion Matrix of all these models are identical

Conclusions

- Successful landing can be affected by key factors such as the launch site, orbit, payload mass and total number of previous launches. It almost appears as if more test launches result in more success.
- Payload mass can affect the success rates, and usually low weight launches result in more success.
- The orbits with the highest success rates are **GEO, HEO, SSO, ES-L1**.
- Decision Tree Algorithm seems to have the best train accuracy

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!