IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Fabio Gaiera
14/11/2021

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- Project background and context

The purpose of this report is to determine whether the Falcon 9 first stage will land successfully or not. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars, instead other providers cost upward of 165 million dollars each. Much of the savings are because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers
  - What influences if the rocket will land successfully?
  - The relationship between rocket variables will impact in determining the success rate of a successful landing.
  - What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate?

Section 1

Methodology

Methodology

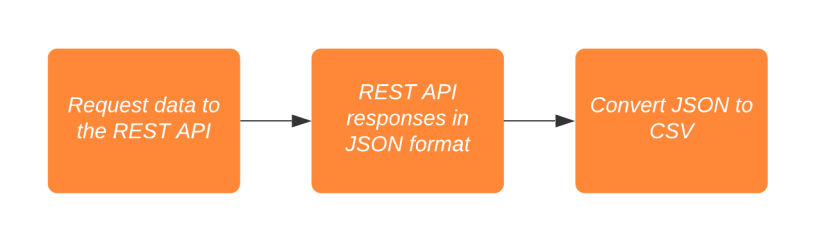
Executive Summary

- Data collection methodology:
  - By using SpaceX REST API
  - By performing web scraping towards an article from Wikipedia
- Perform data wrangling
  - One Hot Encoding data fields and dropping irrelevant columns.
- 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

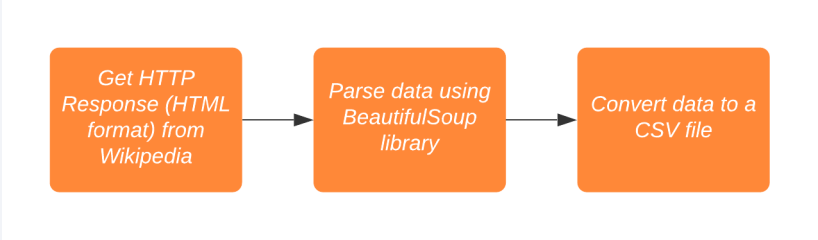
Data Collection

- We have used 2 techniques for gathering data:
  - Using the SpaceX REST API: The API provides data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome. Our goal is to use this data to predict whether a SpaceX rocket will successfully land or not.
  - Using web scraping with Beautiful Soup library towards a Wikipedia article.

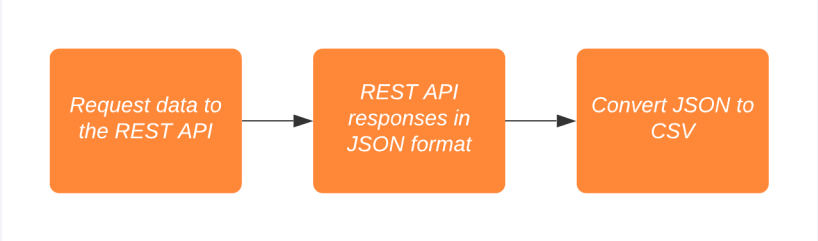
SpaceX REST API



Web scraping using Beautiful Soup



Data Collection – SpaceX API



1. Get the response from SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Convert response to JSON format

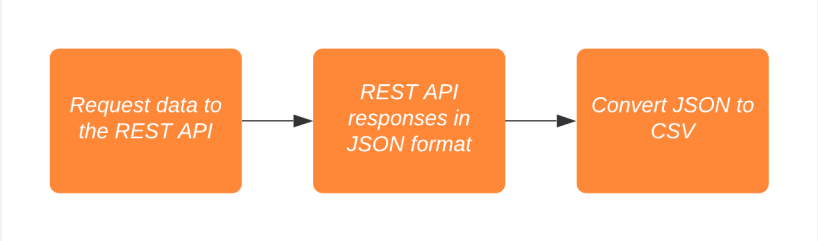
```
response_decoded = response.json()
data = pd.json_normalize(response_decoded)
```

3. Apply functions to clean data

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

GitHub repository

Data Collection – SpaceX API



GitHub repository

4. Combine columns into a 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 Pandas dataframe

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

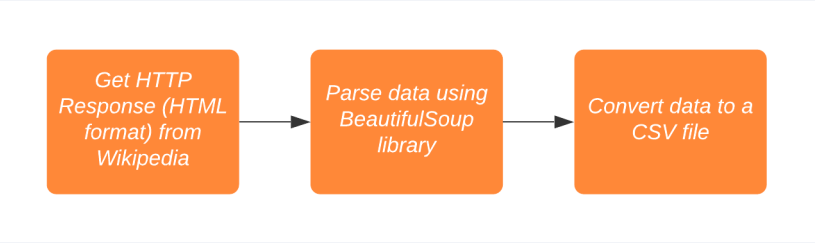
Data Collection – SpaceX API

[GitHub repository]

6. Filter dataframe and export to a CSV file

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping



GitHub repository

1. Get the response from Wikipedia URL

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

2. Create a Beautiful Soup object

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

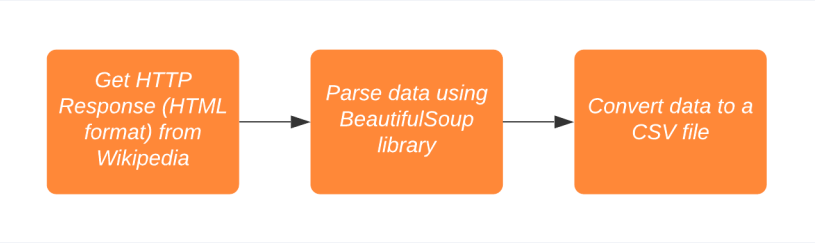
3. Search for all tables

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

4. Get the 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
```

Data Collection - Scraping



GitHub repository

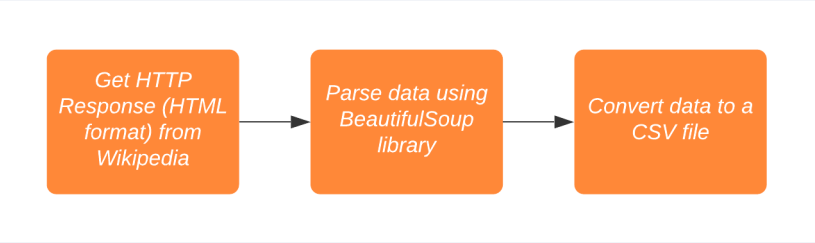
5. Create a dictionary

```
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. Append data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table head
        if rows.th:
            if rows.th.string:
```

Data Collection - Scraping



[GitHub repository]

7. Create a dataframe from dictionary

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

8. Convert dataframe to CSV file

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

Data Wrangling

Introduction

There are several cases where the booster did not land successfully. For example, **True Ocean** means the mission outcome was successfully landed to a specific region of the ocean while **False Ocean** means the mission outcome was unsuccessfully landed to a specific region of the ocean. **True RTLS** means the mission outcome was successfully landed to a ground pad whereas **False RTLS** means the mission outcome was unsuccessfully landed to a ground pad. **True ASDS** means the mission outcome was successfully landed on a drone ship whereas **False ASDS** means the mission outcome was unsuccessfully landed on a drone ship. We convert those outcomes into Training Labels as follows: 1 means the booster successfully landed and 0 means it was unsuccessful.

Process

Perform Exploratory Data Analysis on dataset

Calculate the number of launches on each site

Calculate the number and occurrence of each orbit

Calculate the number and occurence of mission outcome per orbit type

Create a landing outcome label from Outcome column

[GitHub repository]

EDA with Data Visualization

- Scatter Plots
  - Flight Number vs Pay Load Mass
  - Flight Number vs Launch Site
  - Pay Load Mass vs Launch Site
  - Flight Number vs Orbit
  - Pay Load Mass vs Orbit
- Bar Plot
  - Orbit vs Mean
- Line Chart
  - Year vs Success Rate

GitHub repository

EDA with SQL

SQL queries for retrieving data from the dataset

- 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 successful 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. Use a subquery.
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015.
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015.
- 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.

GitHub repository

Build an Interactive Map with Folium

To visualize the launch data into an interactive map, we took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.

We assigned the dataframe launch_outcomes (failures, successes) to classes 0 and 1 with red and green markers on the map in a MarkerCluster()

Using Haversine's formula we calculated the distance from the launch site to various landmarks to find various trends about what is around the launch site to measure patterns. Lines are drawn on the map to measure distance to landmarks.

Example of some trends in which the Launch Site is situated in:

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

GitHub repository

Build a Dashboard with Plotly Dash

- Graphs
  - Pie chart showing the total launches by a certain site or all sites.

  When selecting and individual site, you can observe the proportion of success / failure. When selecting all sites, you can observe the total proportion of success / failure.

  - Scatter graph showing the relationship with Outcome and Payload Mass (Kg) for different boosters

  It shows the relationship between two variables, and it is the best method to show a non-linear pattern.

GitHub repository

Predictive Analysis (Classification)

BUILDING MODEL
- Load dataset using NumPy and pandas.
- Transform Data.
- Split data into training and test data sets.
- Decide which type of machine learning algorithms we want to use.
- Set our parameters and algorithms to GridSearchCV.
- Fit our datasets into the GridSearchCV objects and train our dataset.

EVALUATING MODEL
- Check accuracy for each model.
- Get tuned hyperparameters for each type of algorithms.
- Plot Confusion Matrix.

IMPROVING MODEL
- Feature Engineering.
- Algorithm Tuning.

FINDING THE BEST PERFORMING CLASSIFICATION MODEL
- The model with the best accuracy score wins the best performing model.
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook.

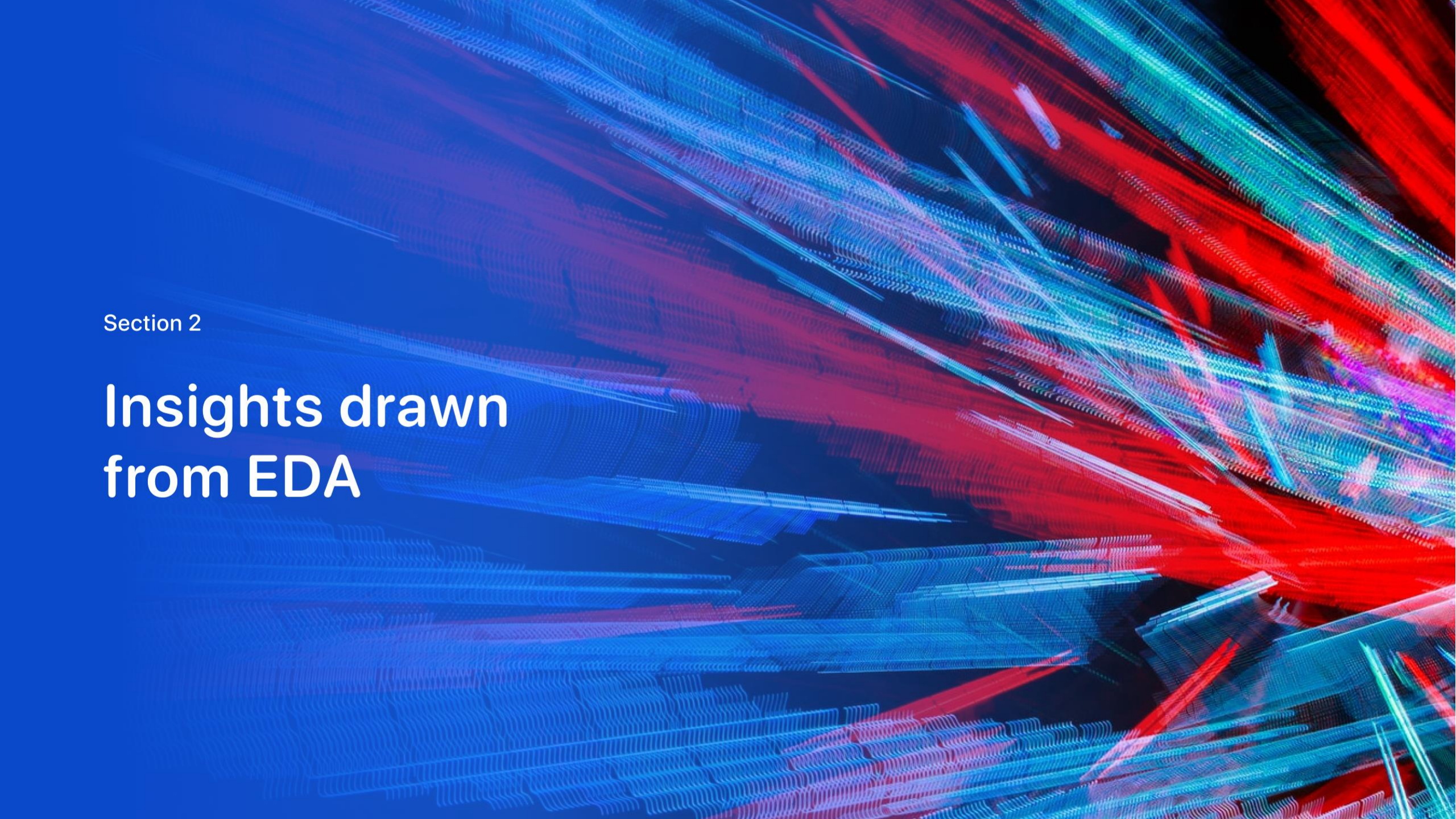
GitHub repository

Results

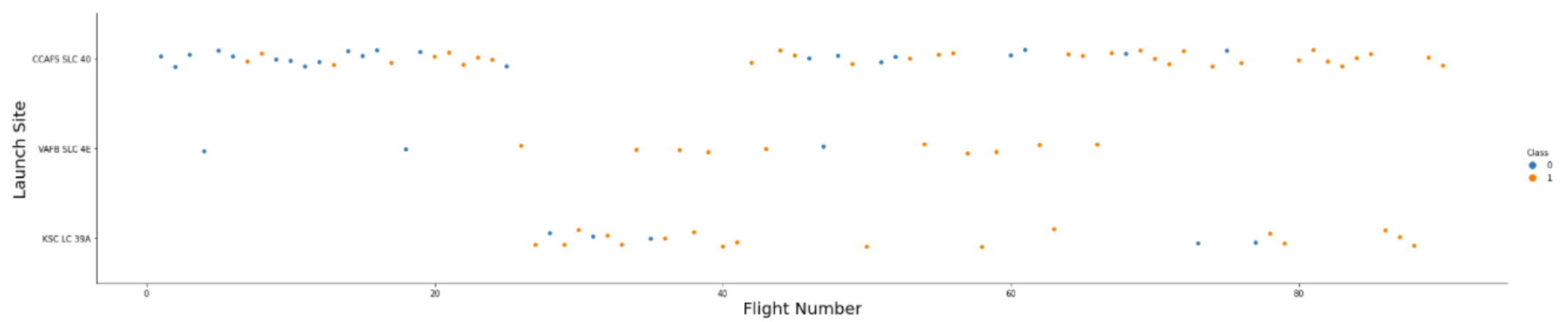
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

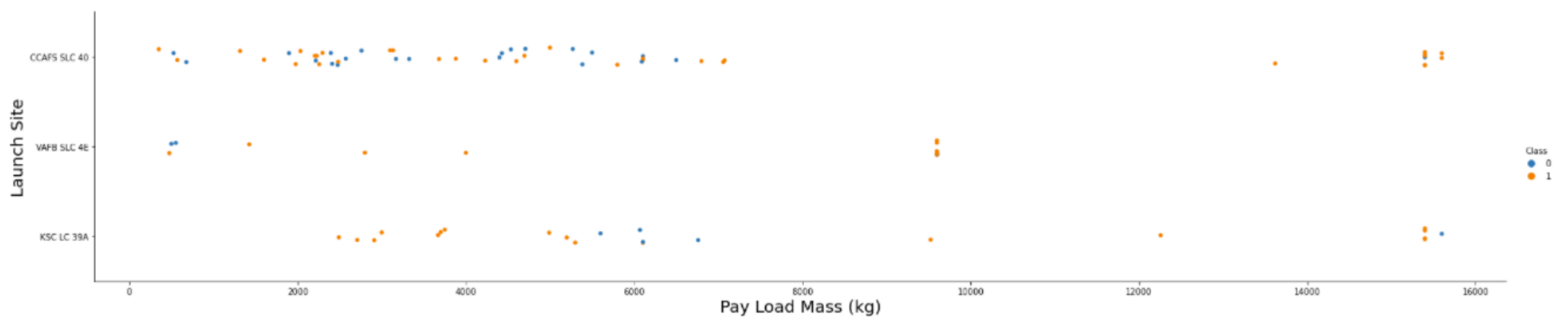


Flight Number vs. Launch Site



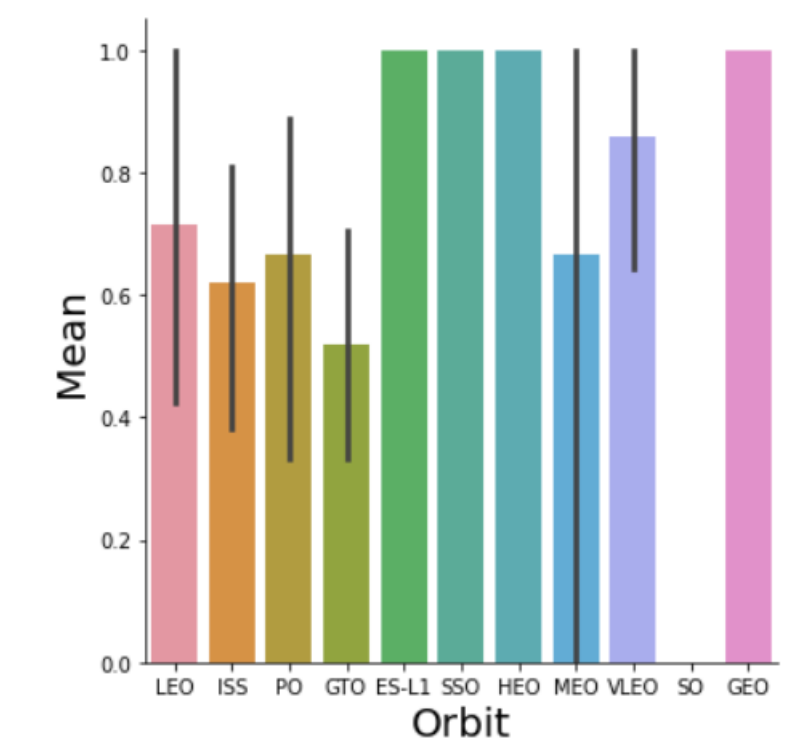
For each launch site, whenever there are more flights, the success rate increases.

Payload vs. Launch Site



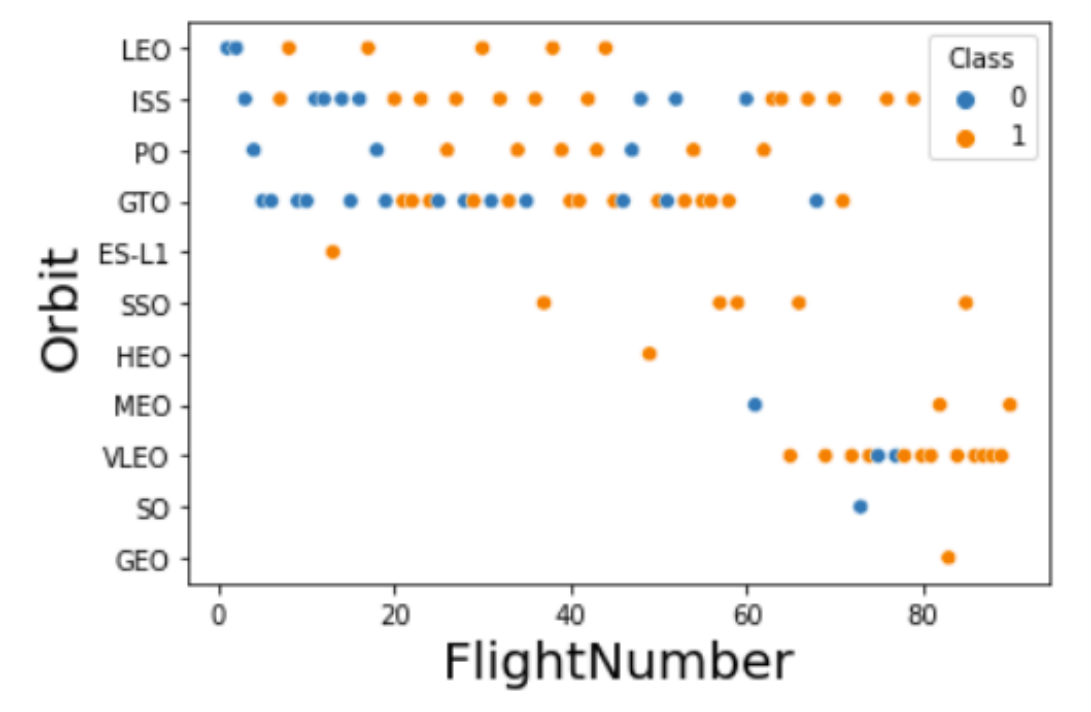
Greater is the Pay Load Mass (kg), the success rate increases. However, for CCAFS SLC 40 there is not a clear pattern that helps to determine the correlation between the Pay Load Mass (kg) and the success rate.

Success Rate vs. Orbit Type



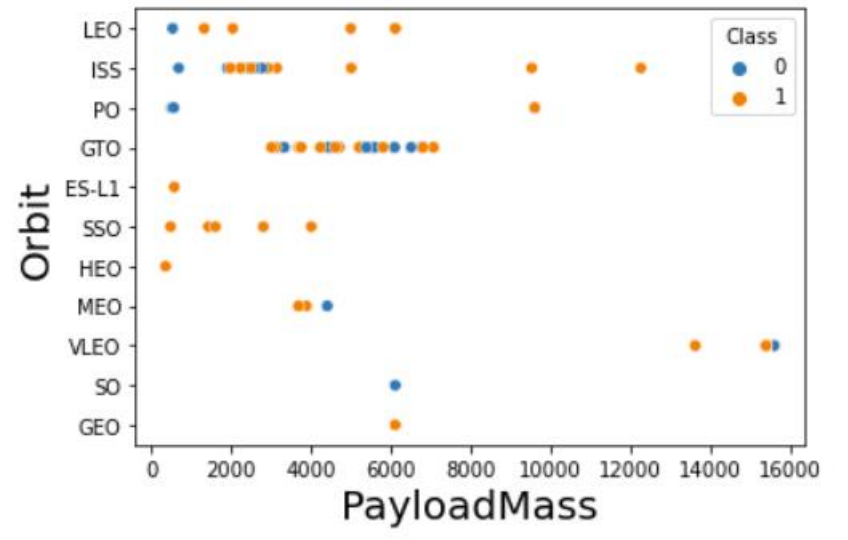
Orbits ES-L1, SSO, HEO and GEO have the best success rate.

Flight Number vs. Orbit Type



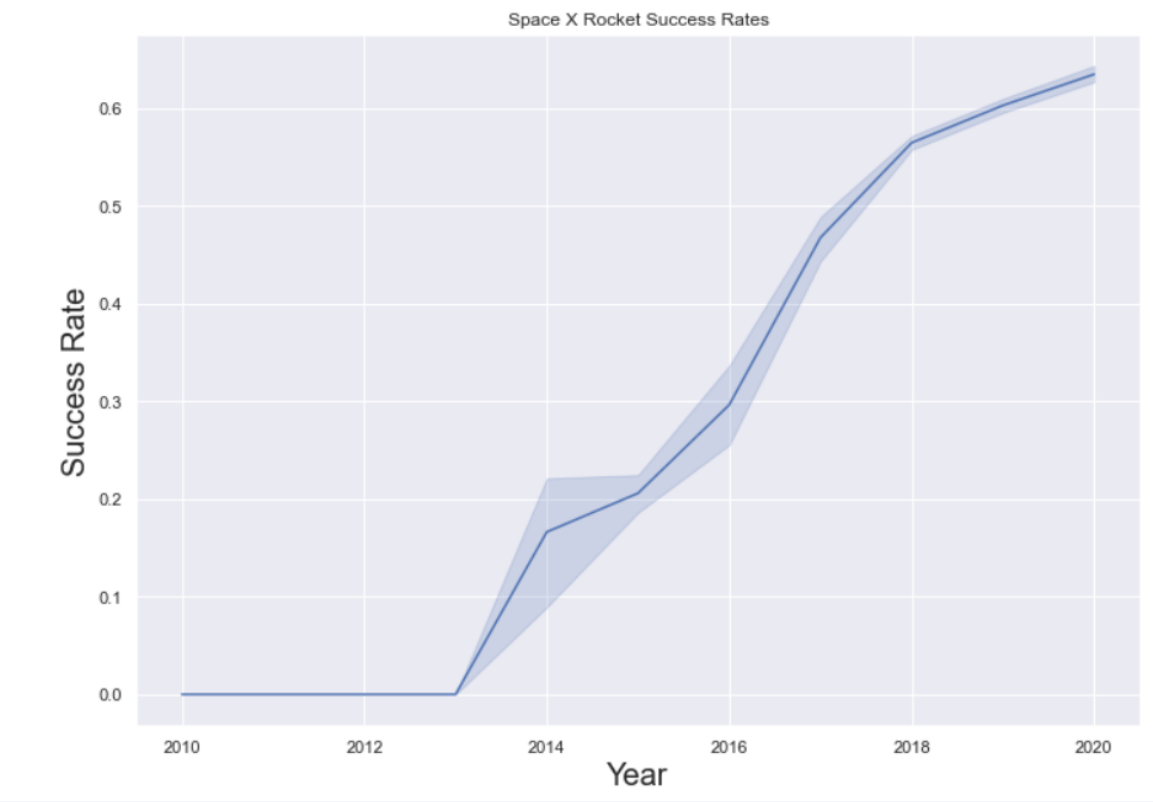
Clearly, in the LEO orbit, the success is related to the number of flights. However, for the GTO orbit there is no relationship between Flight Number and the success rate.

Payload vs. Orbit Type



The heavy payloads have a negative influence on GTO orbits. Instead, for LEO and ISS orbits have a positive influence.

Launch Success Yearly Trend



You can observe that the success rate since 2013 kept increasing till 2020.

All Launch Site Names

- Unique launch sites
  - CCAFS LC-40
  - CCAFS SLC-40
  - KSC LC-39A
  - VAFB SLC-4E
- SQL query
  - SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
  - The keyword DISTINCT means that it will only show unique values in the LAUNCH_SITE column from SPACEXTBL table.

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

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

Using the keyword LIMIT 5 in the query means that it will only show 5 records from SPACEXTBL table and LIKE keyword will filter all launch sites starting with ‘CCA’.

Total Payload Mass

- Total payload carried by boosters from NASA: 45596
- SQL query
  - SELECT SUM(payload_mass__kg_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
  - The SUM function does the addition of all values from column payload_mass__kg_ whereas the WHERE predicate filters by customer NASA (CRS).

Average Payload Mass by F9 v1.1

- Average payload mass carried by booster version F9 v1.1: 2928
- SQL query
  - SELECT AVG(payload_mass__kg_) FROM SPACEXTBL WHERE booster_version = 'F9 v1.1';
  - The AVG function does the average of all values from column payload_mass__kg_ whereas the WHERE predicate filters by booster_version F9 v1.1.

First Successful Ground Landing Date

- First successful landing outcome on ground pad occurred on 2015-12-22
- SQL query:
  - SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)';
  - The MIN function returns the minimum value (in this case for DATE column) whereas the WHERE predicate filters by LANDING__OUTCOME.

Successful Drone Ship Landing with Payload between 4000 and 6000

- Names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

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

- SQL query
  - SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
  - The BETWEEN operator selects values withing a given range.

Total Number of Successful and Failure Mission Outcomes

- Total number of successful and failure mission outcomes

Successful outcomes	Failure outcomes
100	1

- SQL queries

SELECT COUNT(*) AS SUCCESSFUL_MISSION FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Success%';

SELECT COUNT(*) AS FAILURE_MISSION FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Failure%';

The COUNT function counts the rows. In this case it will count the rows that satisfy certain condition, which is specified in the WHERE predicate.

Boosters Carried Maximum Payload

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

SQL query

```
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_)
                           FROM SPACEXTBL);
```

The WHERE predicate contains a subquery that selects the maximum PAYLOAD_MASS__KG_ from SPACEXTBL table.

2015 Launch Records

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

SQL query

SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING__OUTCOME LIKE '%Failure (drone ship)%' AND DATE LIKE '2015%';

This query selects certain columns and uses the predicate WHERE for filtering by LANDING__OUTCOME and DATE.

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

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.

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

```
SELECT LANDING__OUTCOME, COUNT(*) AS RANK

FROM SPACEXTBL

WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'

GROUP BY LANDING__OUTCOME

ORDER BY RANK DESC;
```

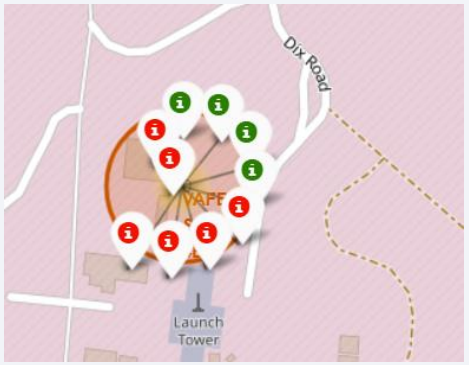
Section 4

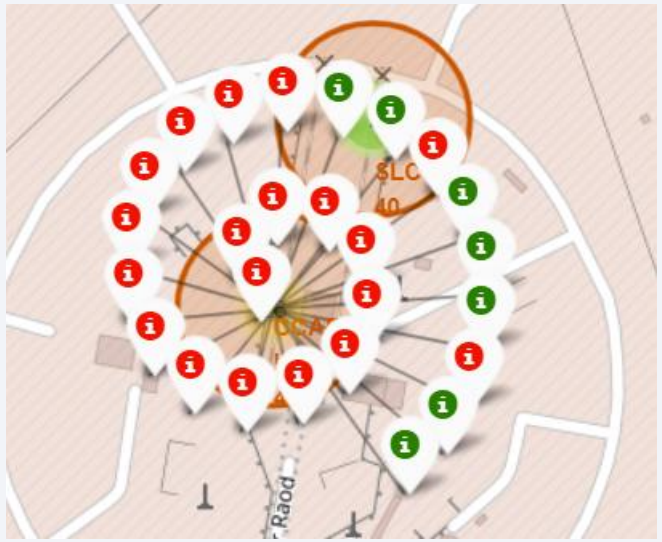
Launch Sites Proximities Analysis

All launch sites on a map

We can observe that all launch sites are placed in United States of America: States of California and Florida.

Success / failed launches for each site on the map





Green marker shows successful launches and red marker show failed launches.

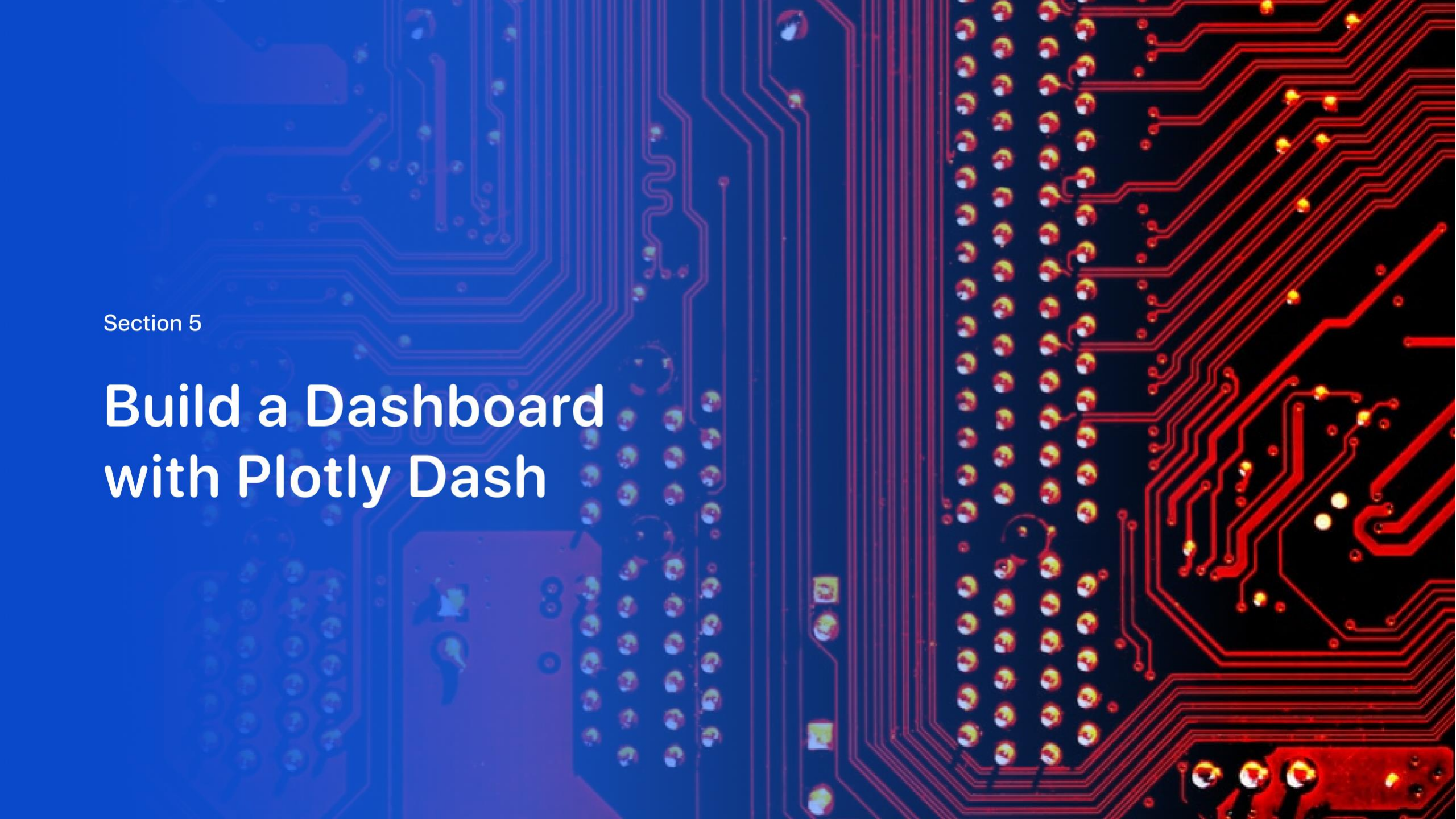
Distances between a launch site to its proximities

The above picture shows the distance to coastline. Analyzing the map, we can conclude that:

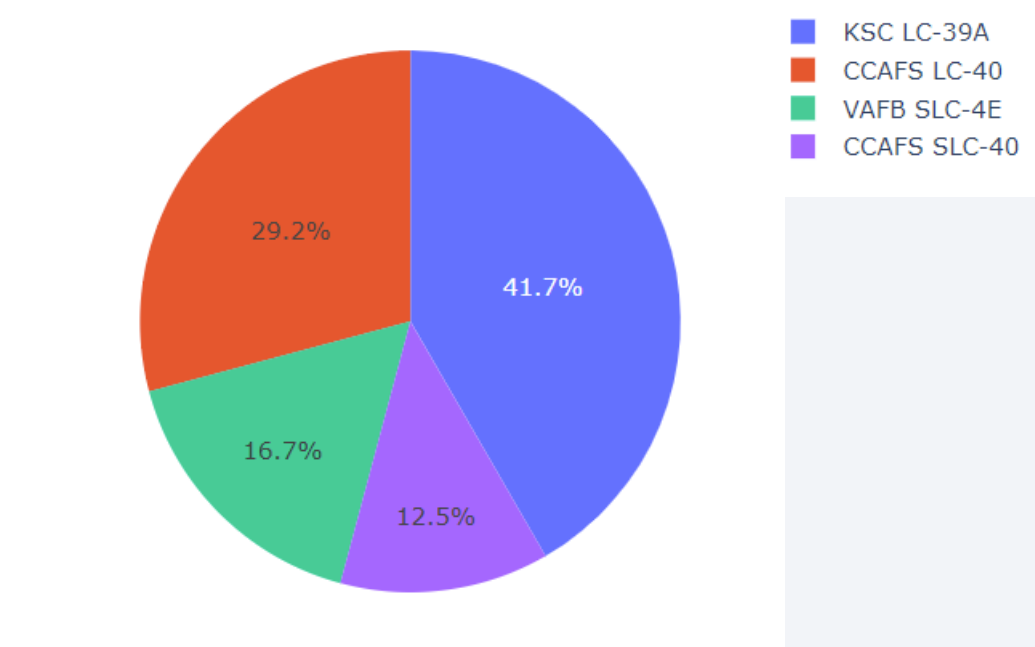
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Are launch sites in close proximity from cities? Yes

Section 5

Build a Dashboard with Plotly Dash

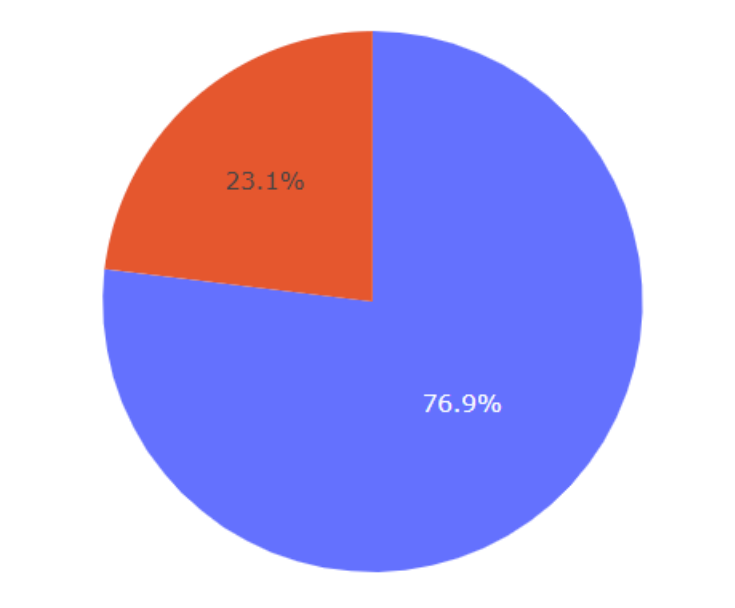


Success count for all launch sites



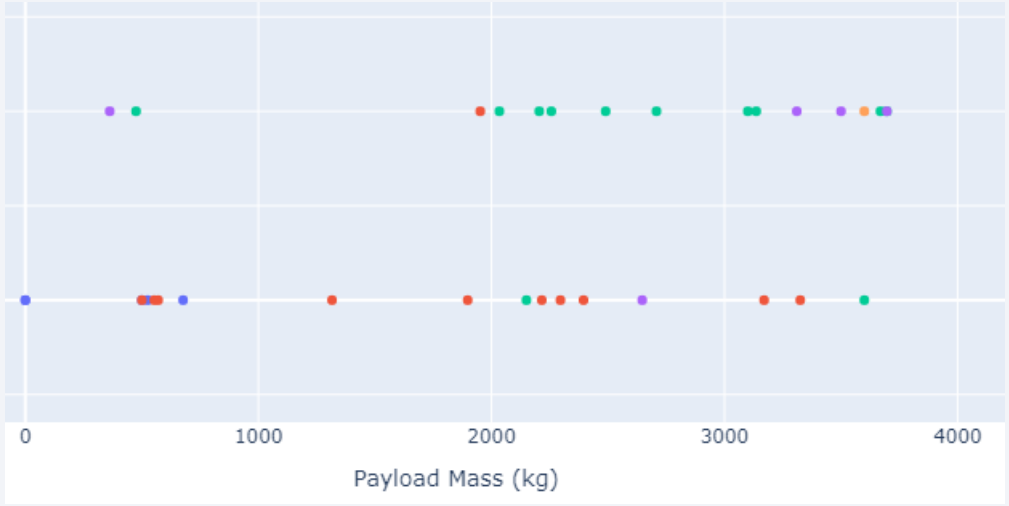
We can observe that KSC LC-39A is the most successful launch site.

Launch site with highest launch success ratio

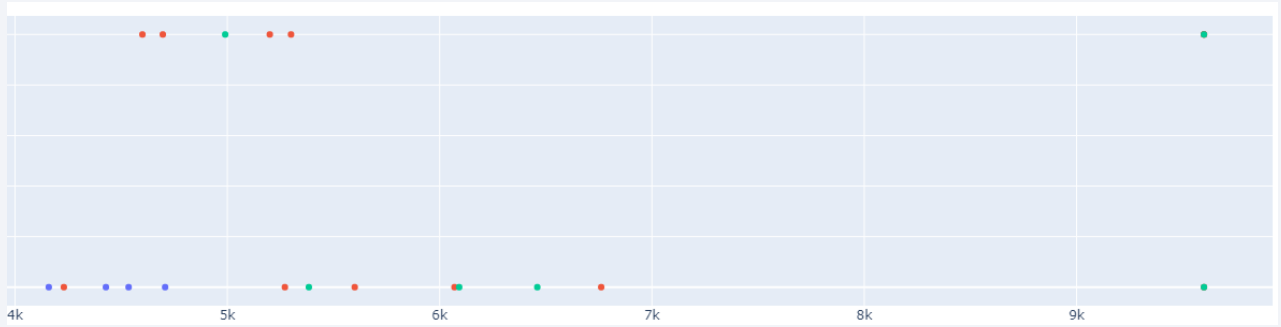


KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate.

Payload vs launch outcome scatter plot



Low weighted payload 0 kg – 4000 kg



Heavy weighted payload 4K kg – 10K kg

We can see that low weighted payloads have higher success rate than the heavy weighted payloads.

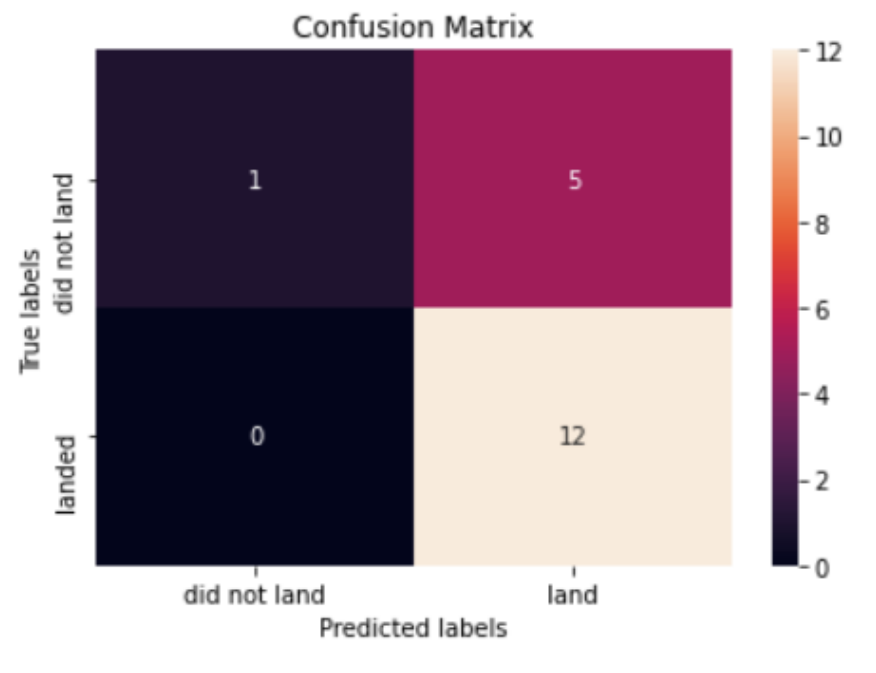
Section 6

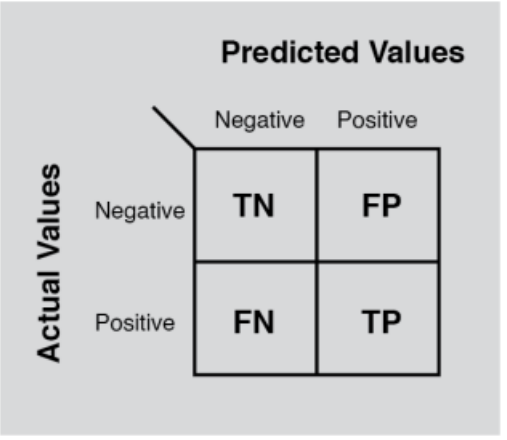
Predictive Analysis (Classification)

Classification Accuracy

```
Best Algorithm is Tree with a score of 0.8892857142857142
Best Params are: {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5,
'splitter': 'random'}
```

Confusion Matrix





We can observe that the major problem is false positives.

Conclusions

- The tree classifier algorithm provides the highest accuracy for this dataset.
- Low weight payloads perform better than heavier payloads.
- The success rate for SpaceX increases across the years.
- KSC LC-39A has the most successful launches from all the sites.
- Orbits GEO, HEO, SSO and ES-L1 have the best Success Rate.

Appendix

- Programming Language: Python
- Databases: DB2
- Libraries: NumPy, pandas, scikit-learn, Plotly, Folium
- Tools: Jupyter Notebook, Microsoft Power Point, Coursera labs.

Thank you!