IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

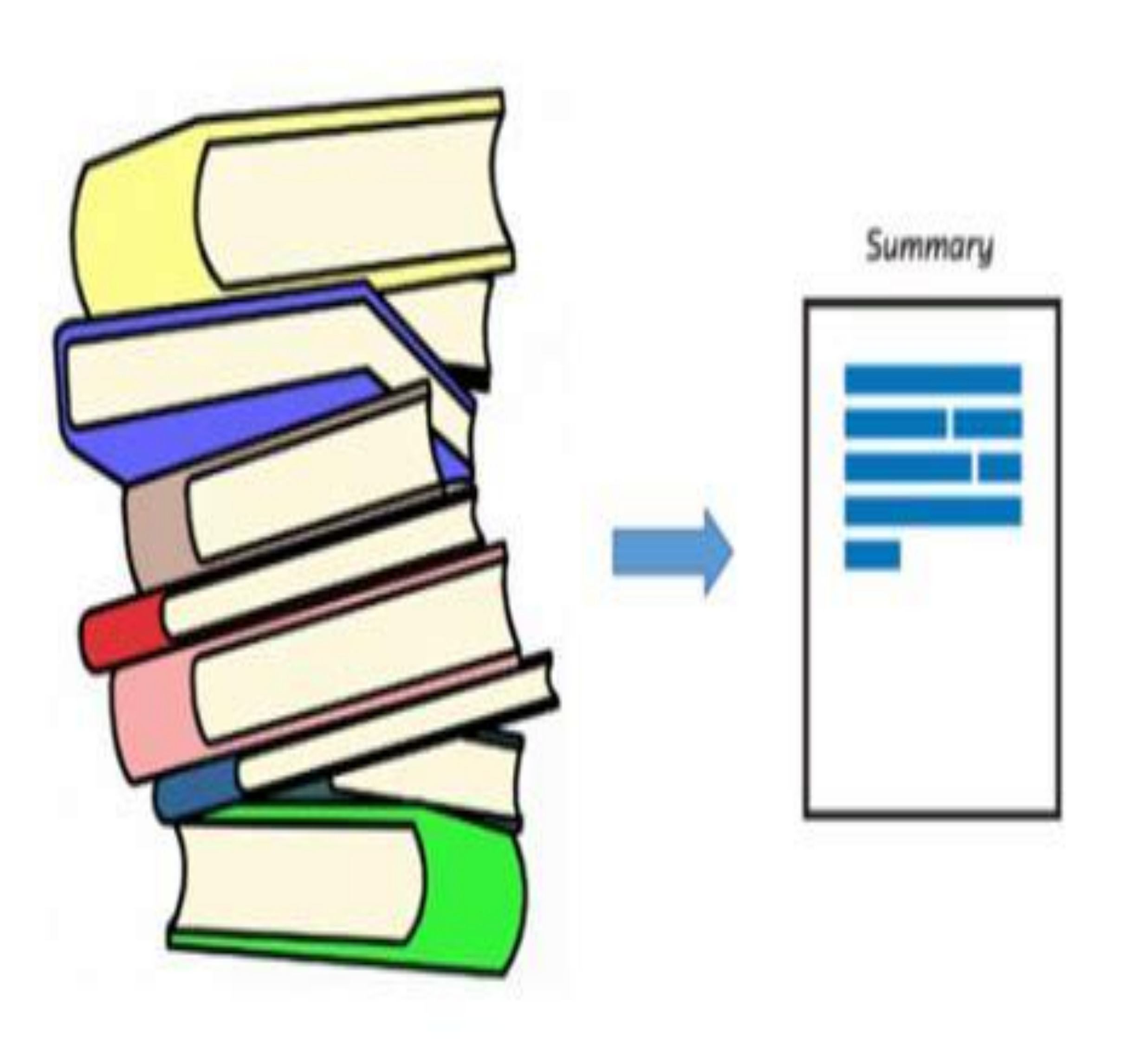
Aliyu Muhammad Aminu

11th September, 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
- Predicting analysis (Classification)

❑Summary of all results

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

Introduction

□Project background and context

We predicted if the falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. 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 effect each relationship with certain 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

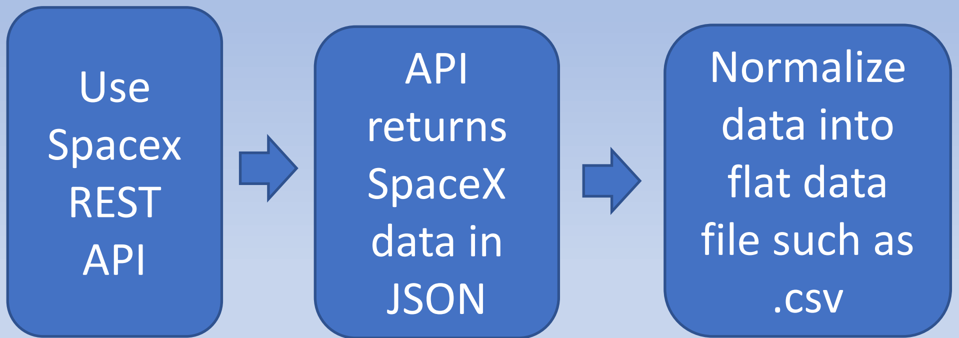
- Data collection methodology:
  - SpaceX Rest API
  - Web Scrapping from **Wikipedia**
- Perform data wrangling
  - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Plotting: Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.
- 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 following datasets was collected by

- We worked with SpaceX launch data that is gathered from the SpaceX REST API.
- This API will give us 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 SpaceX will attempt to land a rocket or not.
- The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
- Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.

SpaceX API

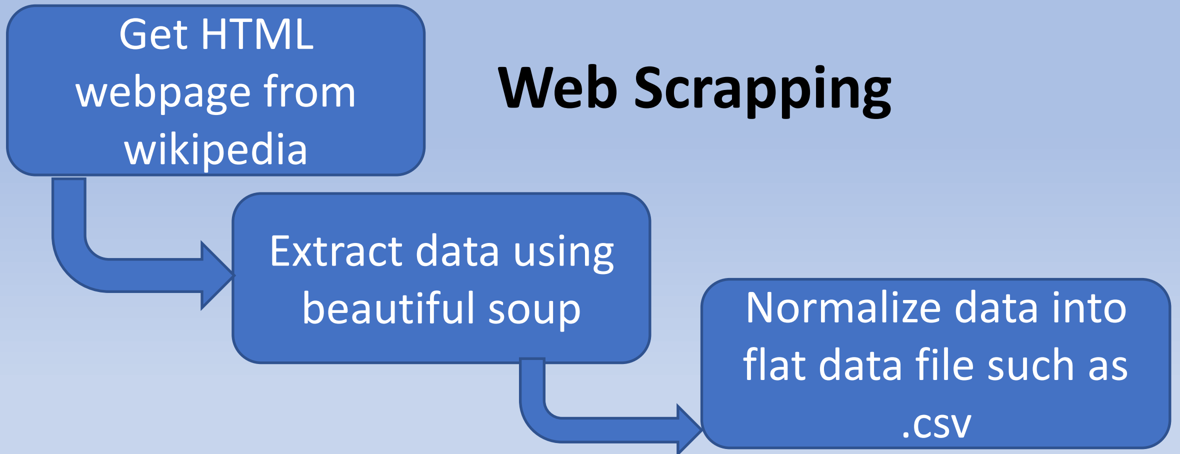


Use Spacex REST API

API returns SpaceX data in JSON

Normalize data into flat data file such as .csv

Web Scrapping

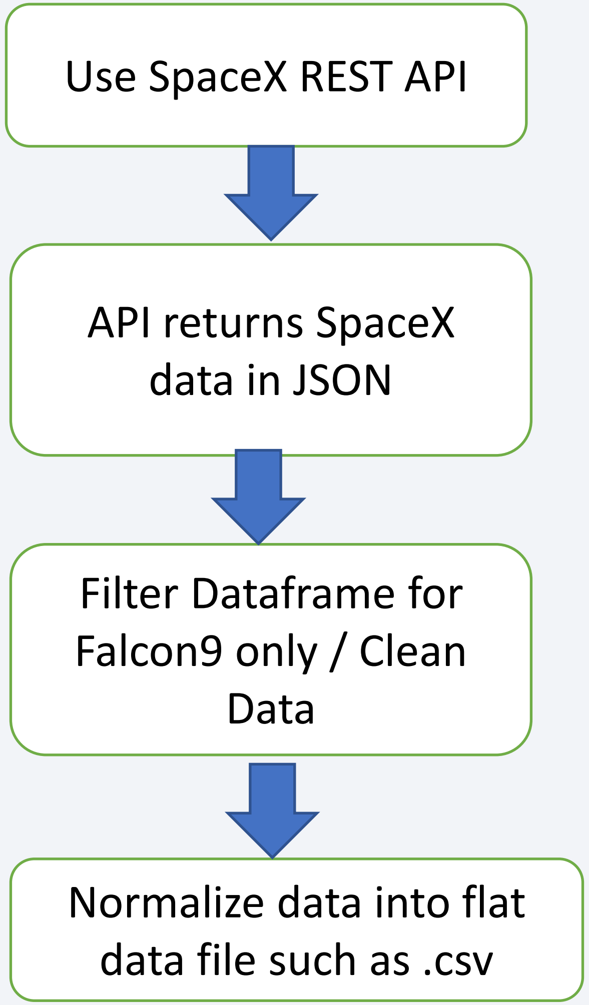


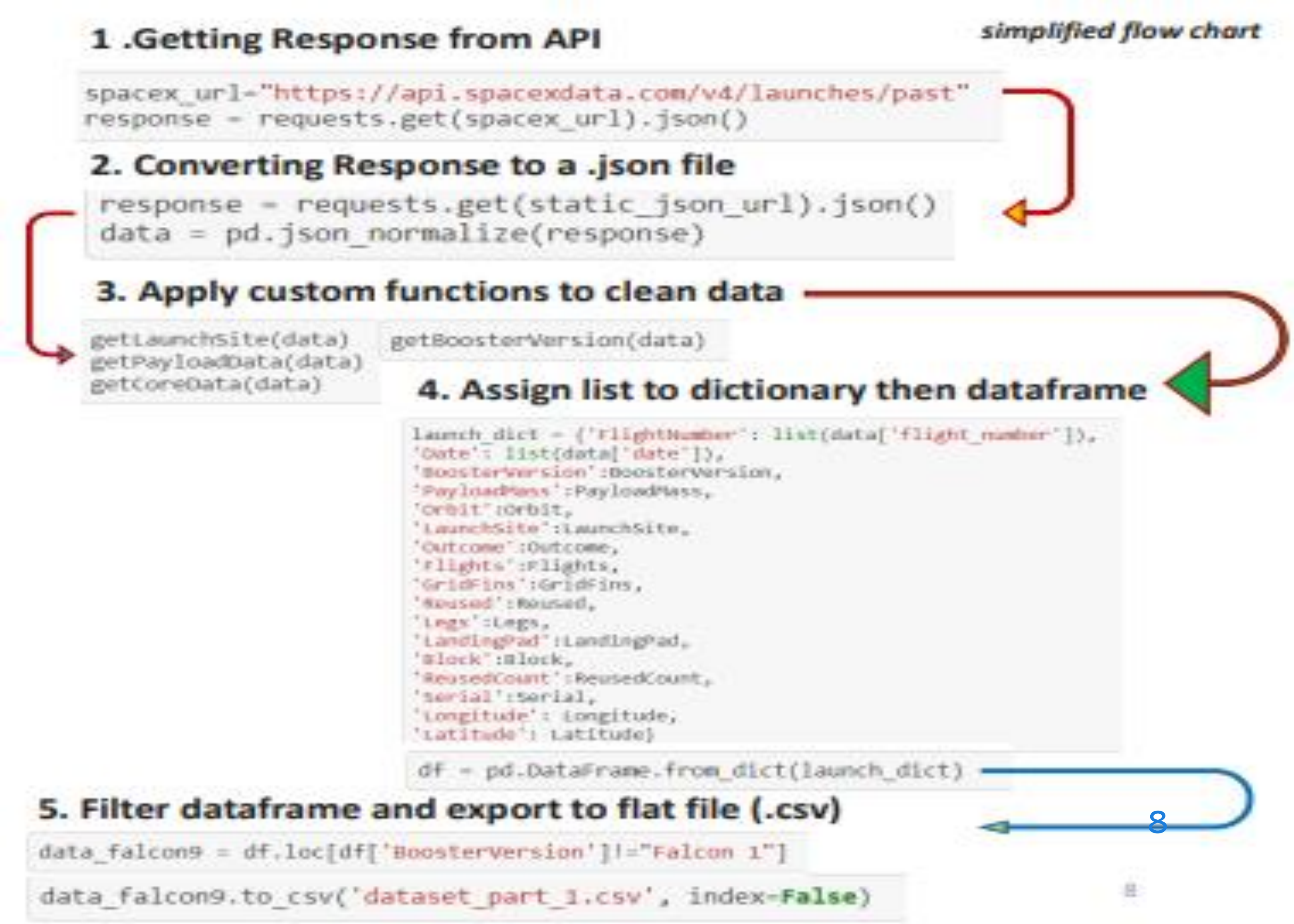
Get HTML webpage from wikipedia

Extract data using beautiful soup

Normalize data into flat data file such as .csv

Data Collection – SpaceX API


Use SpaceX REST API
API returns SpaceX data in JSON
Filter Dataframe for Falcon9 only / Clean Data
Normalize data into flat data file such as .csv


simplified flow chart

1 .Getting Response from API

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

2. Converting Response to a .json file

```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. Apply custom functions to clean data

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

```
getBoosterVersion(data)
```

4. Assign list to dictionary then dataframe

```
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}
```

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

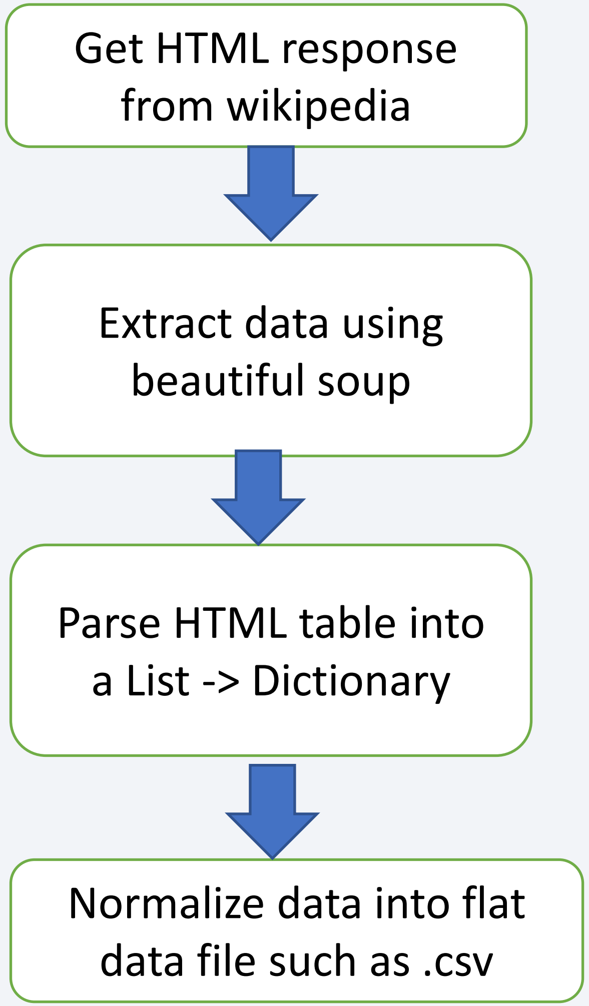
5. Filter dataframe and export to flat file (.csv)

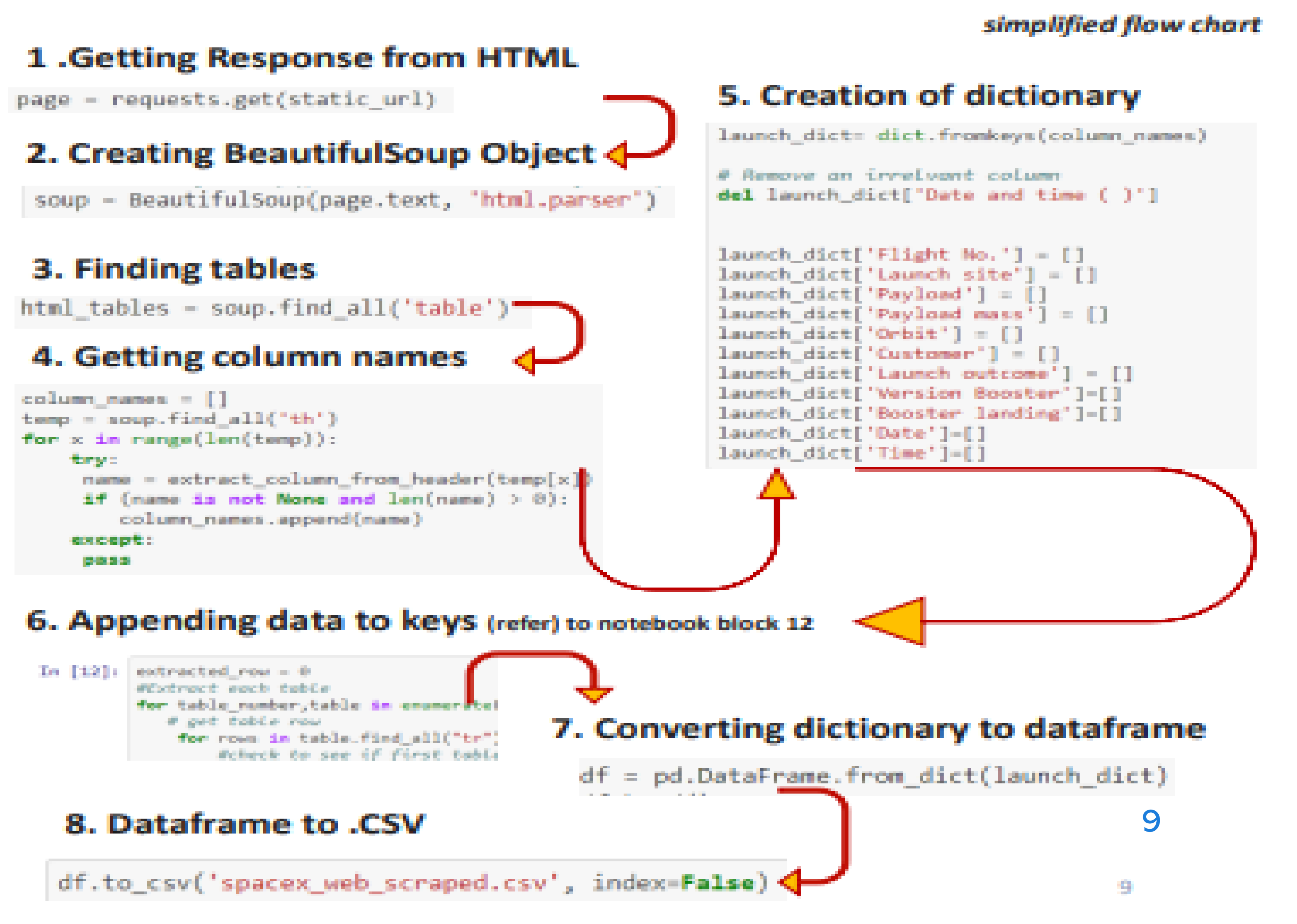
```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

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

[Github URL to Notebook]

Data Collection - Scraping


Get HTML response from wikipedia
Extract data using beautiful soup
Parse HTML table into a List -> Dictionary
Normalize data into flat data file such as .csv



simplified flow chart

1 .Getting Response from HTML

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

2. Creating BeautifulSoup Object

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

3. Finding tables

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

4. Getting 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. Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']


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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Appending data to keys (refer) to notebook block 12

```
In [12]: extracted_row = 0
         #Extract each table
         for table_number,table in enumerate(
             # get table row
              for rows in table.find_all("tr")
                  #check to see if first table
```

7. Converting dictionary to dataframe

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

8. Dataframe to .CSV

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

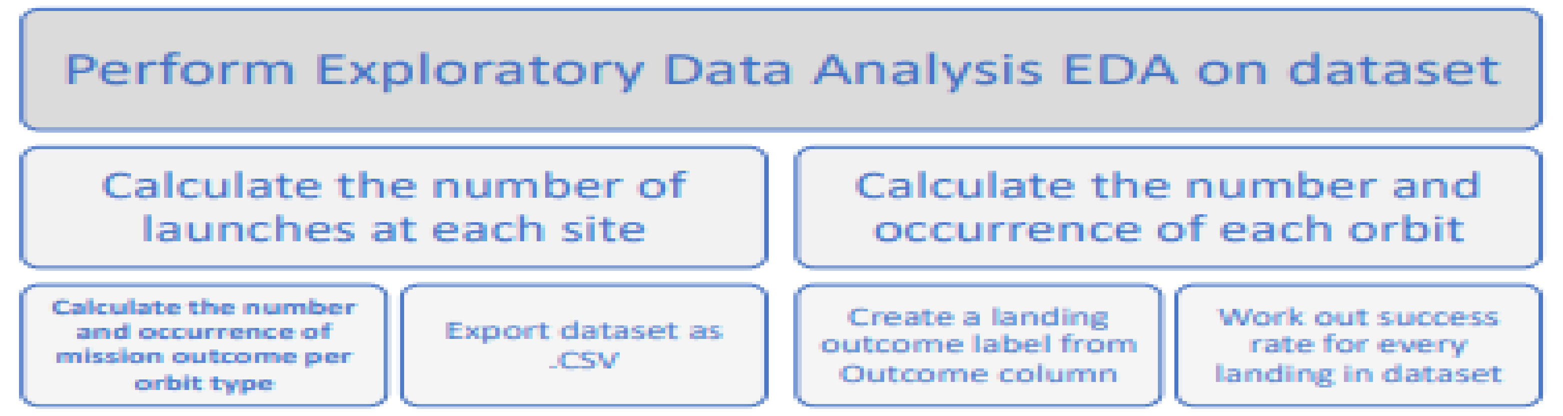
[Github URL to Notebook](#)

Data Wrangling

Introduction

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; 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 groud pad False RTLS means the mission outcome was unsuccessfully landed to a groud pad. True ASDS means the mission outcome was unsuccessfully landed on a drone ship.
We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Process



Perform Exploratory Data Analysis EDA on dataset

Calculate the number of launches at each site

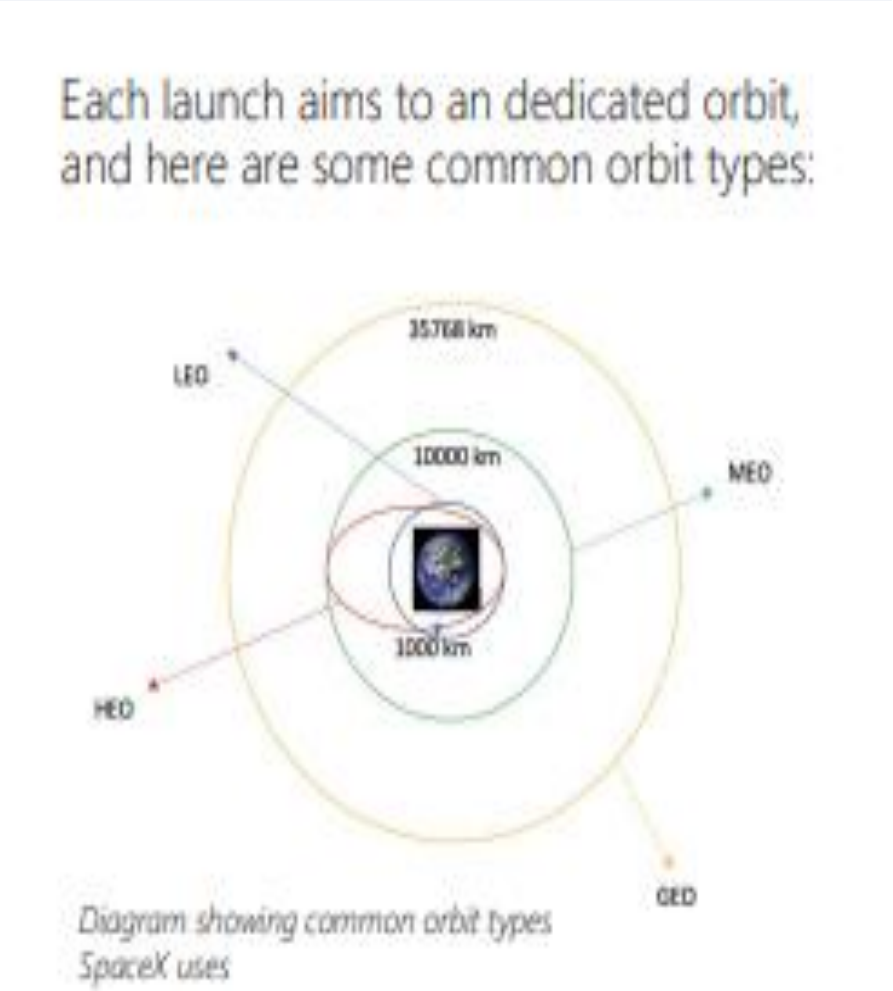
Calculate the number and occurrence of each orbit

Calculate the number and occurrence of mission outcome per orbit type

Export dataset as .CSV

Create a landing outcome label from Outcome column

Work out success rate for every landing in dataset



Each launch aims to an dedicated orbit, and here are some common orbit types:

Diagram showing common orbit types SpaceX uses

Github URL to Notebook

EDA with Data Visualization

Scatter Graphs being drawn:

Flight Number VS. Payload Mass

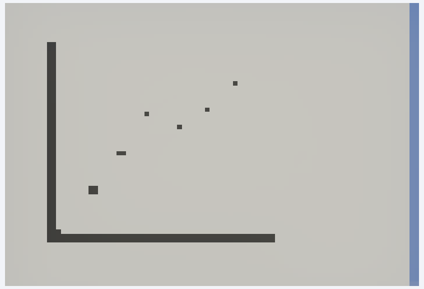
Flight Number VS. Launch Site

Payload VS. Launch Site

Orbit VS. Flight Number

Payload VS. Orbit Type

Orbit VS. Payload Mass



Scatter plots show how much one variable is affected by another. The relationship between two variable is called their correlation. Scatter plots usually consist of a large body of data.

Bar Graph being drawn:

Mean VS. Orbit

A bar diagram makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axis. Bar charts can also show big changes in data over tine.

Line Graph being drawn:

Success Rate VS. Year

Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded.

[Github URL to Notebook]

EDA with SQL

Performed SQL Queries to gather information about the dataset.

For example of some questions we wre asked about the data we needed information about.

Which we are using SQl queries to get the answers in the dataset :

- Display the names of the unique launch sited in the space mission
- Display 5 records where launch sites begin with the string ‘KSC’
- Display the total payload mass carried by booster launched by NASA (CRS)
- Display the total payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster versions which have carried the maximum payload mass.
- Listing the records which will display the month names, successful landing_outcomes in ground pad, booster versions, launch_site for the months in year 2017.
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Github URL to Notebook

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 Green and Red markers on the map in a MarkerCluster()

Using Haversines'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 somes trends in which he 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 URL to Notebook

Build a Dashboard with Plotly Dash

Used Plotly and Dash so that we can view and play with data.

- **The dashboard is build with Plotly Dash**
  - **Graphs**
    - Pie Chart showing the total launches by a certain site/all sites
    - Display relative proportions of multiply classes of data.
    - Size of the circle can be made proportional to the total quantity it represents.
  - **Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions**
    - It shows the relationship between two variables.
    - It is the best method to show you anon0linear pattern.
    - The range of data flow, i.e. maximum and minimum value can be determined.
    - Obervation and reading are straightforward.

Github URL to Dash App

Predictive Analysis (Classification)

BUILDING MODEL

- Load our dataset into Numpy and pandas
- Transform Data
- Split our data into training and testing data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fir our datasets into the GridSearchCV
- Fir our datasets into the GridSearchCV objects and train our dataset.

EVALUATING MODEL

- Check accuracy for each model
- Get tunes 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 URL to Notebook

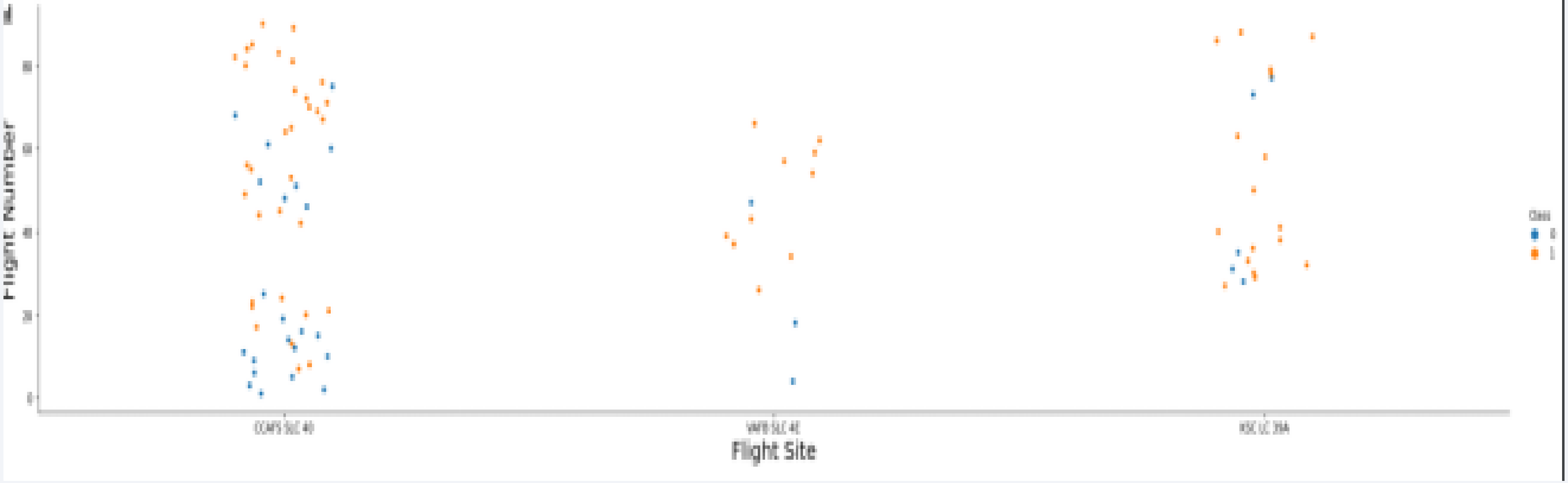
Results

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

Section 2

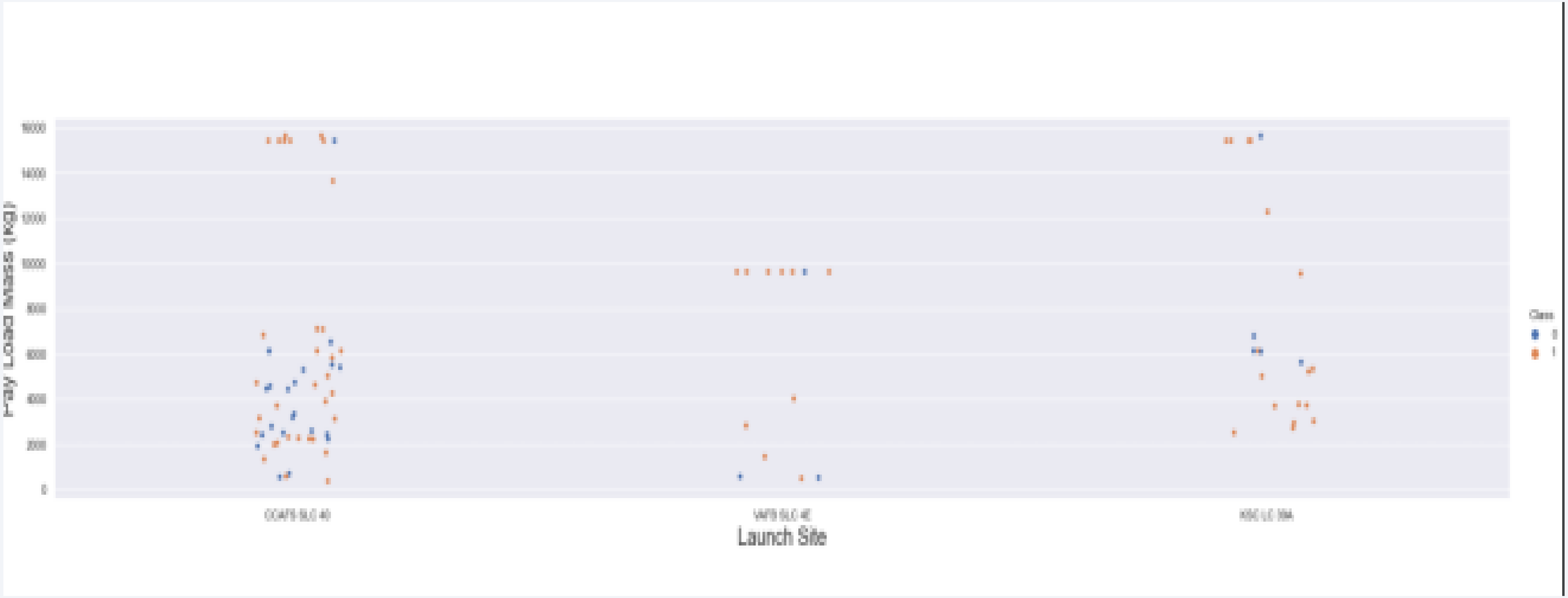
Insights drawn from EDA

Flight Number vs. Launch Site



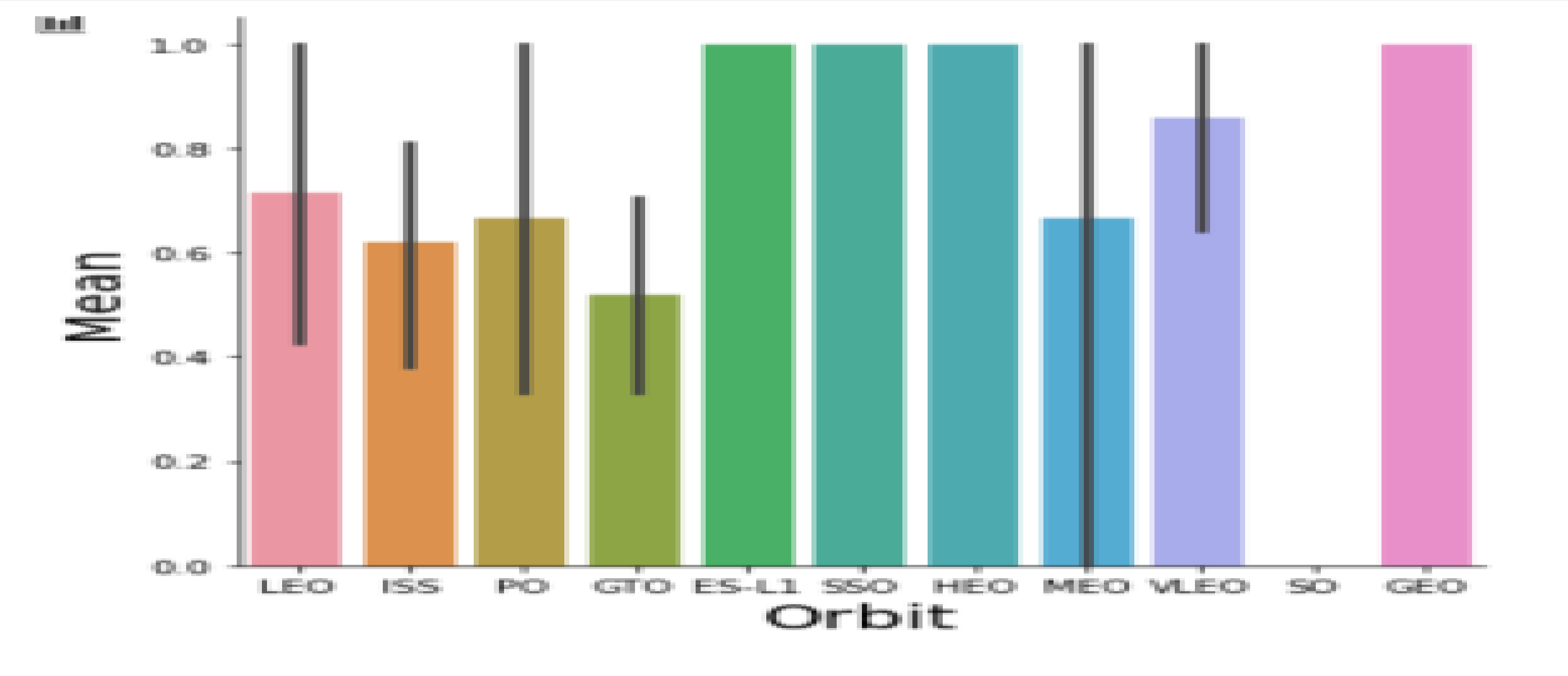
The more amount of flights at a launch site the greater the success rate at a launch site

Payload vs. Launch Site



The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Pay Load Mass for a success launch

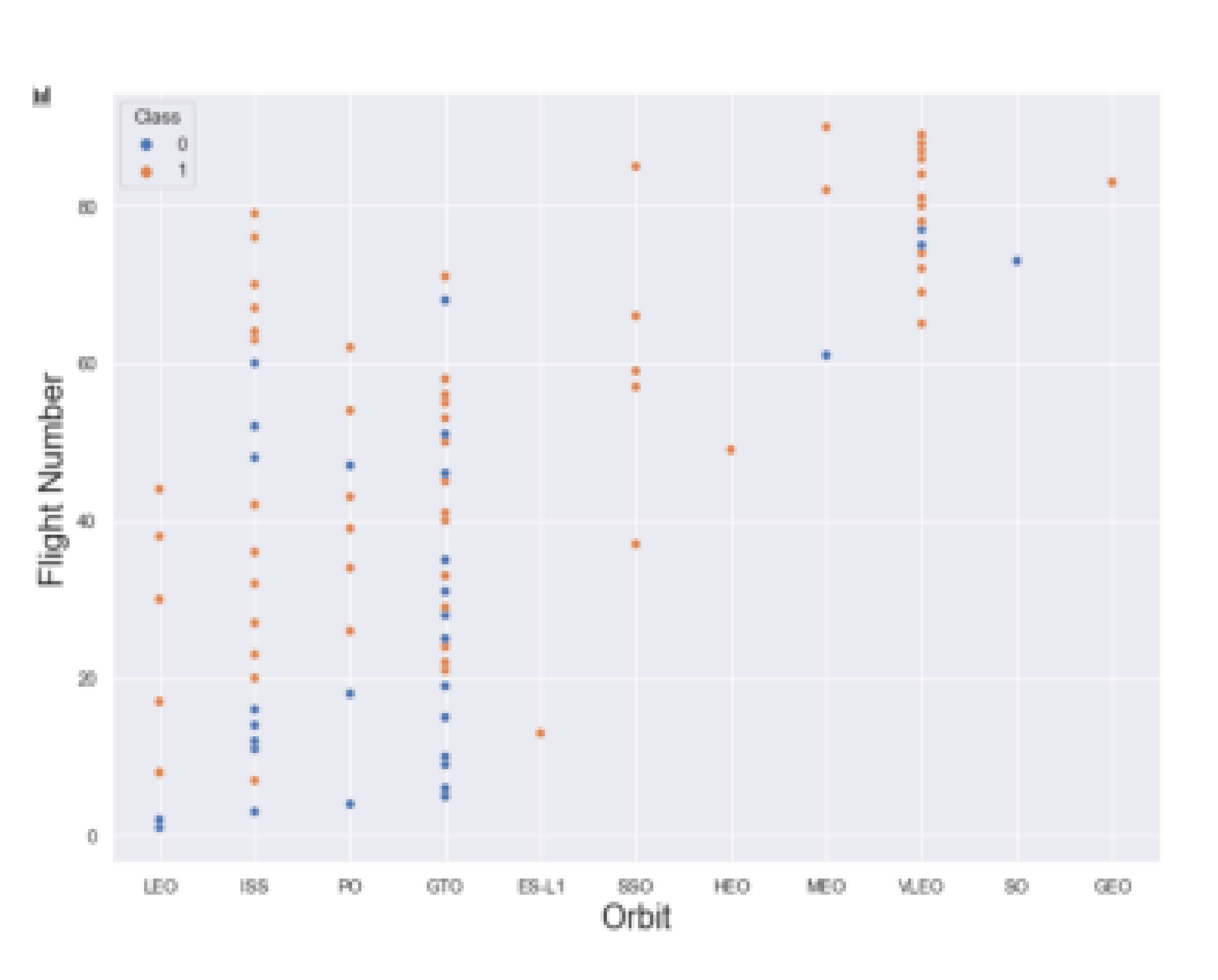
Success Rate vs. Orbit Type



Orbit GEO, HEO, SSO, ES-L1 has the best Success Rate

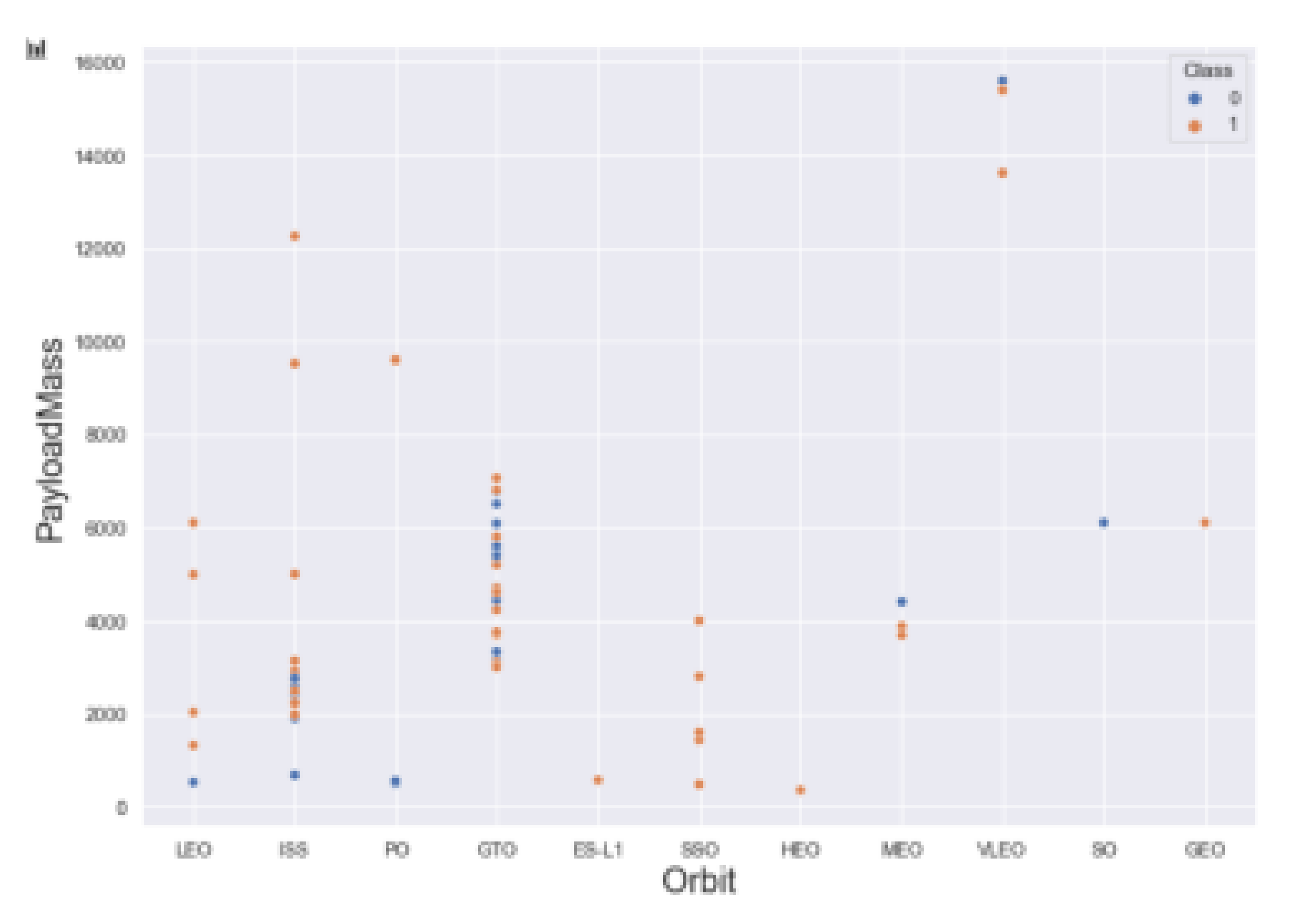
Flight Number vs. Orbit Type

You should see that in the LEO orbit the success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit



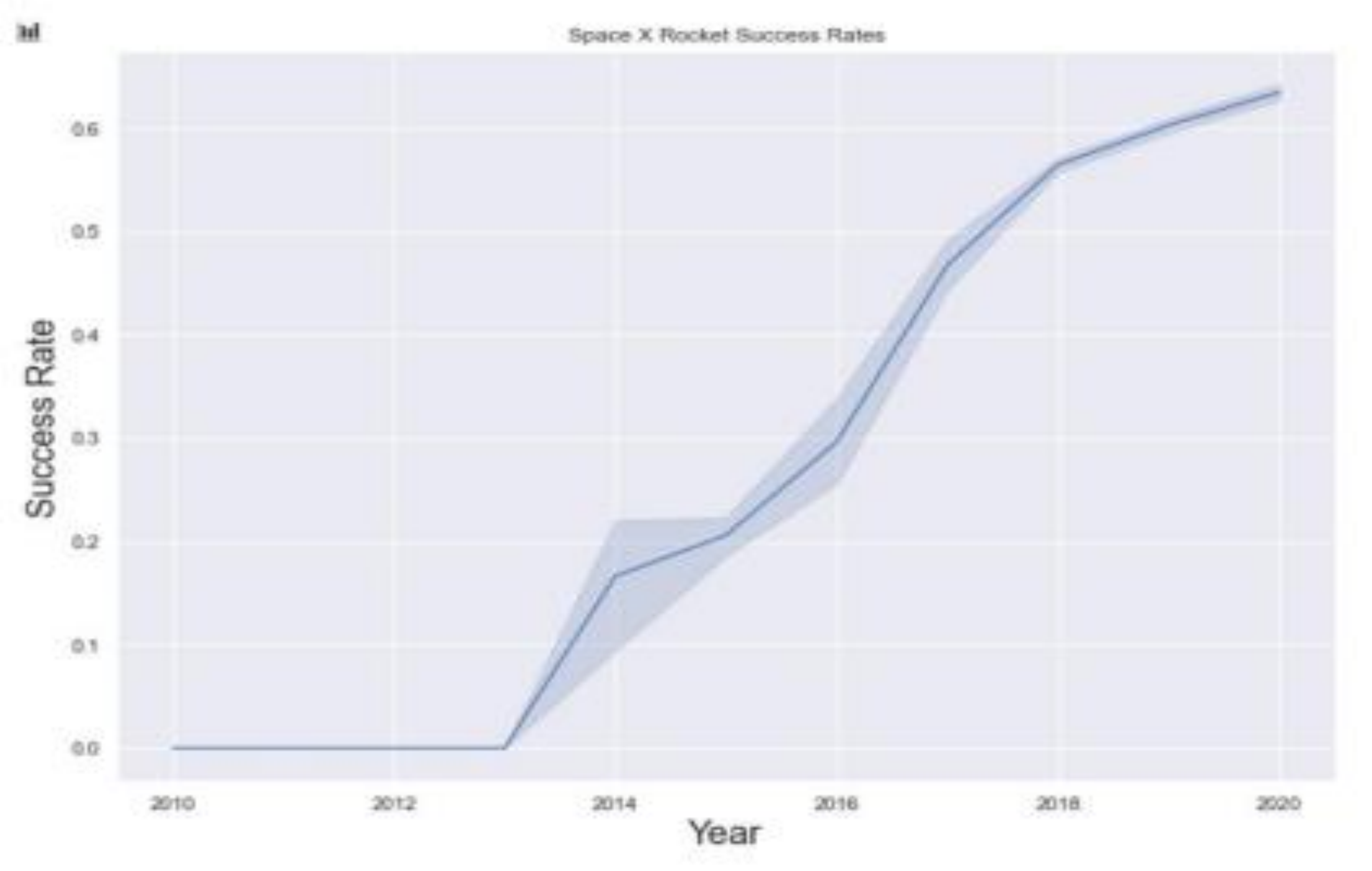
Payload vs. Orbit Type

You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.



Launch Success Yearly Trend

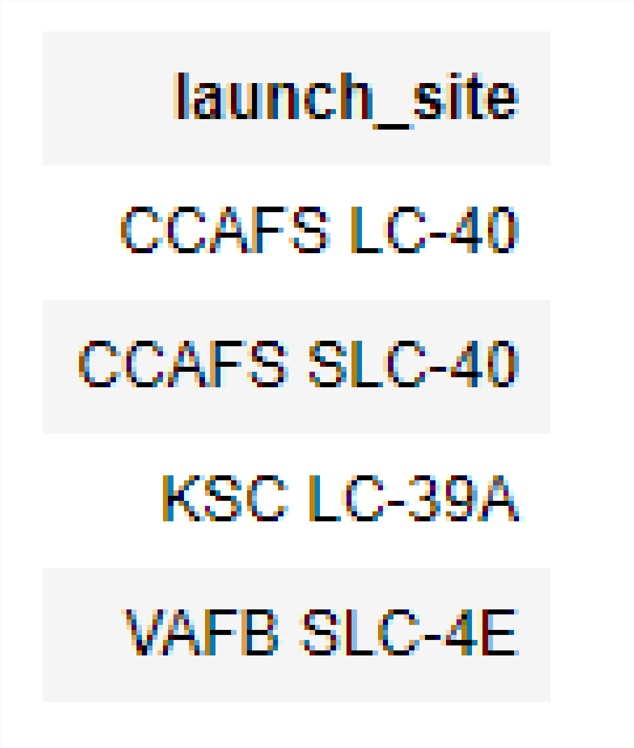
You can observe that the success rate since 2013 kept increasing till 2020



All Launch Site Names

SQL QUERY

Select DUSTINCT Launch_Site from SPACEXTBL



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

QUERY EXPLANATION

Using the word **DISTINCT** in the query means that it will only show Unique values in the **Launch_Site** column from **SPACEXTBL**

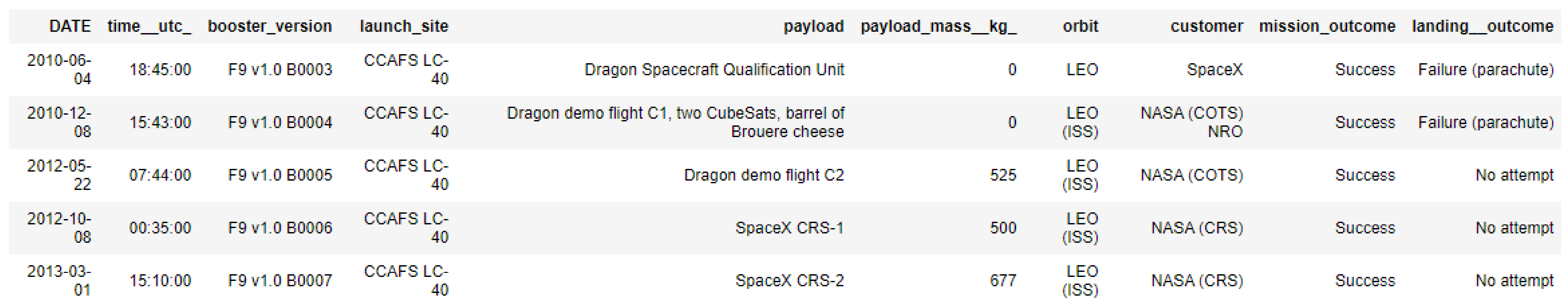
Launch Site Names Begin with 'CCA'

SQL QUERY

select * from spacextbl where launch_site like 'CCA%' limit 5

QUERY EXPLANATION

Using the word **LIMIT 5** in the query means that it will only show 5 records from **SpaceXTBL** and **LIKE** keyword has a wild card with the words **'CCA%'** the percentage in the end suggests that the **Launch_Site** name must start with CCA.

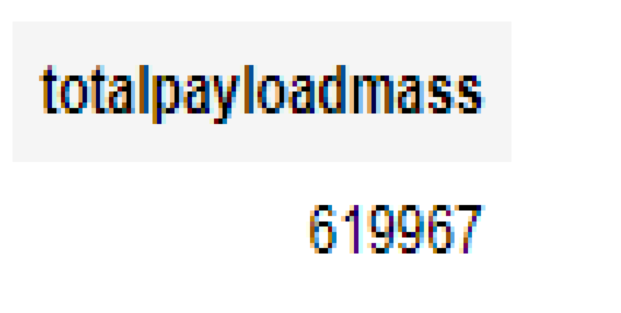


DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight 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

Total Payload Mass

SQL QUERY

select sum(payload_mass__kg_)
TotalPayloadMass from spacextbl



totalpayloadmass
619967

QUERY EXPLANATION

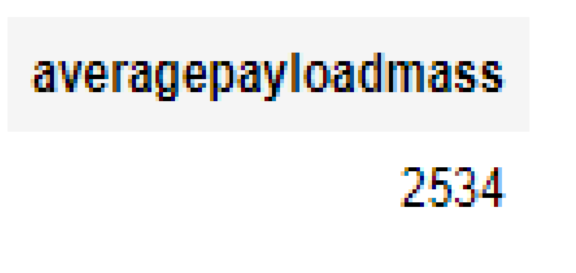
Using the function **SUM** summates the total in the column **PAYLOAD_MASS_KG_**

The **WHERE** clause filters the dataset to only perform calculations on **Customer NASA (CRS)**

Average Payload Mass by F9 v1.1

SQL QUERY

```
select AVG(PAYLOAD_MASS_KG_)
AveragePayloadMass from SpaceXTBL where
Booster_Version = 'F9 v1.1’
```



averagepayloadmass
2534

QUERY EXPLAINATION

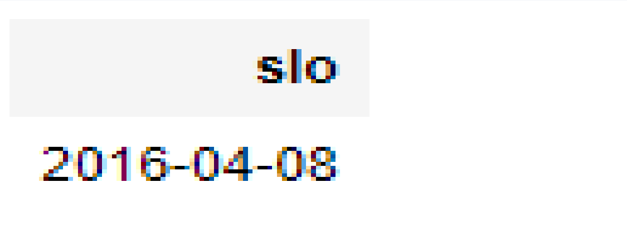
Using the function **AVG** works out the average in the column **PAYLOAD_MASS_KG_**

The **WHERE** clause filters the dataset to only perform calculations on **Booster_version F9 v1.1**

First Successful Ground Landing Date

SQL QUERY

select MIN(Date) SLO from SpaceXtbl where Landing__Outcome = 'Success (drone ship)’



slo
2016-04-08

QUERY EXPLAINATION

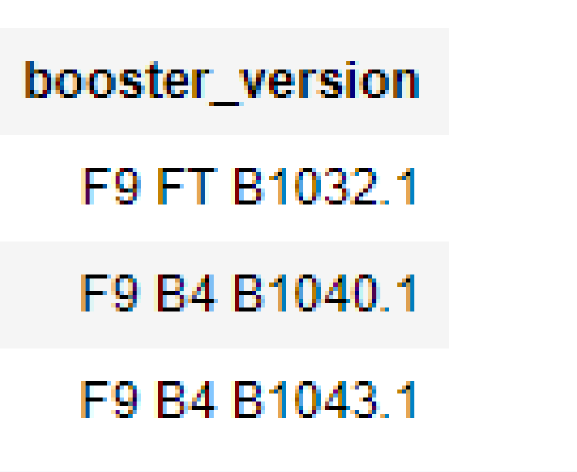
Using the function **MIN** works out the minimum date in the column Date

The **WHERE** clause filters the dataset to only perform calculations **on Landing__Outcome Success (drone ship)**

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL QUERY

select booster_version from spacextbl where Landing__Outcome = 'Success (ground pad)' and payload_mass__kg_>4000 and payload_mass__kg_ < 6000



booster_version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

QUERY EXPLAINATION

Selecting only **Booster_Version**

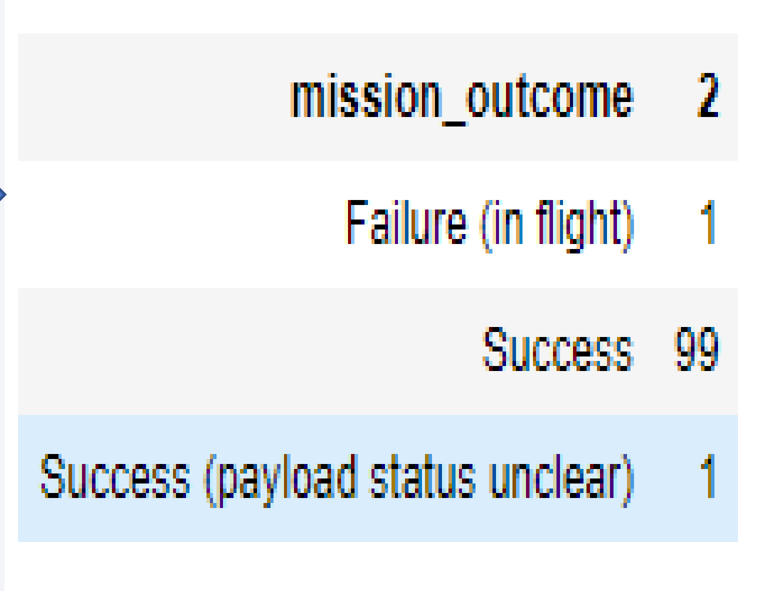
The **WHERE** clause filters the dataset to **Landing__Outcome = Success (drone ship)**

The **AND** clause specifies additional filter conditions **Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000**

Total Number of Successful and Failure Mission Outcomes

SQL QUERY

select mission_outcome, count(mission_outcome) from spacextbl group by mission_outcome



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

QUERY EXPLAINATION

We select the **mission_outcome** and **count** the number of **mission_outcome** from **spacextbl** and **group by** the result by mission_outcome inorder to have same mission outcome together

Boosters Carried Maximum Payload

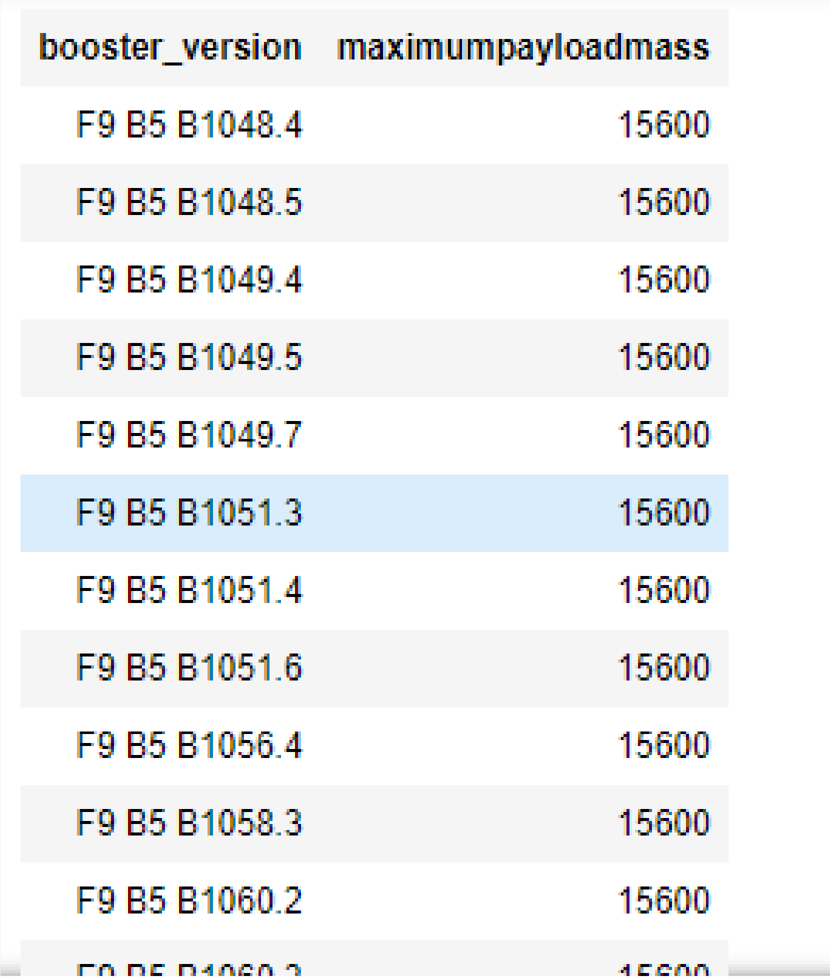
SQL QUERY

SELECT DISTINCT Booster_Version, MAX(PAYLOAD_MASS__KG_) AS MaximumPayloadMass FROM SpaceXTBL GROUP BY Booster_Version ORDER BY MaximumPayloadMass DESC

QUERY EXPLAINATION

Using the word **DISTINCT** in the query means that it will only show Unique values in the **Booster_Version** column from **tblSpaceX**

GROUP BY puts the list in order set to a certain condition. **DESC** means its arranging the dataset into descending order



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

2015 Launch Records

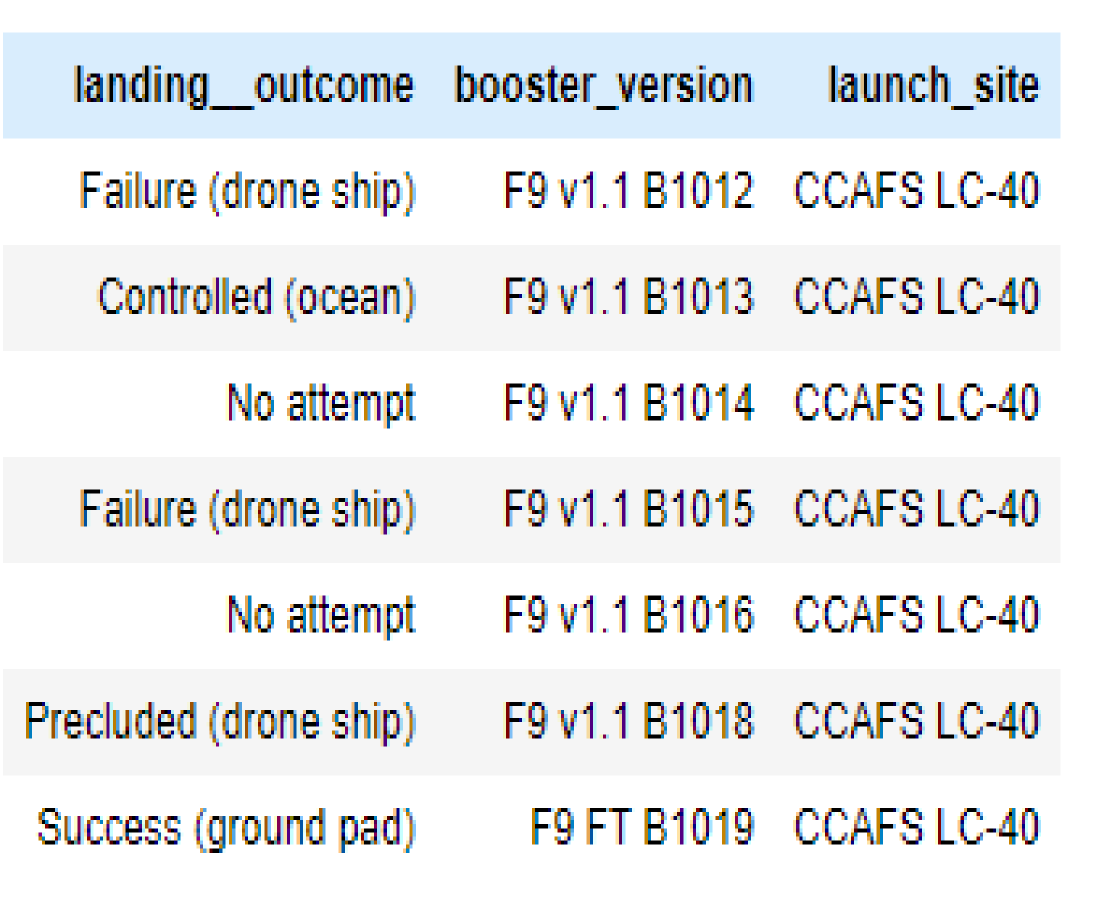
SQL QUERY

select landing__outcome, booster_version, launch_site from spacextbl where date like '2015%’

QUERY EXPLAINATION

We select the landing__outcome, booster_version and launch_site of every launch in the year 2015.

The where clause filters Date to be 2015

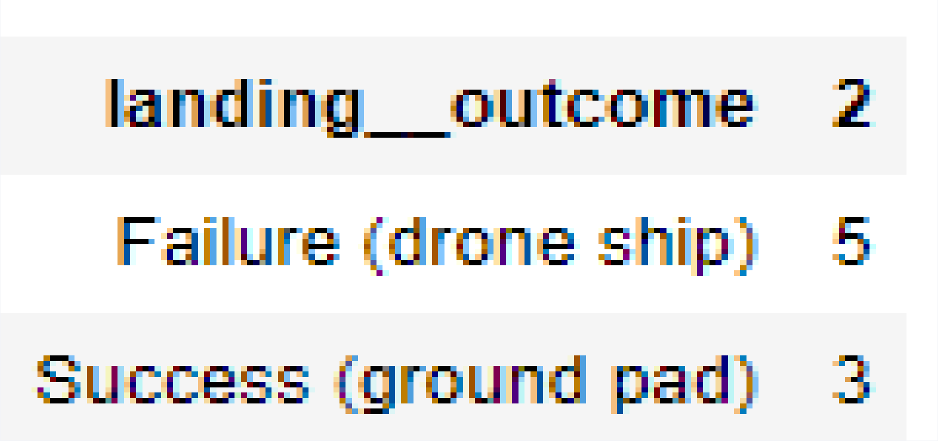


landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Controlled (ocean)	F9 v1.1 B1013	CCAFS LC-40
No attempt	F9 v1.1 B1014	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
No attempt	F9 v1.1 B1016	CCAFS LC-40
Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40
Success (ground pad)	F9 FT B1019	CCAFS LC-40

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

SSQL QUERY

```
select landing__outcome,
count(landing__outcome) from spacextbl where
landing__outcome='Failure (drone ship)' or
landing__outcome='Success (ground pad)' and
date between '2010-06-04' and '2017-03-20'
group by landing__outcome
```



landing__outcome	2
Failure (drone ship)	5
Success (ground pad)	3

QUERY EXPLAINATION

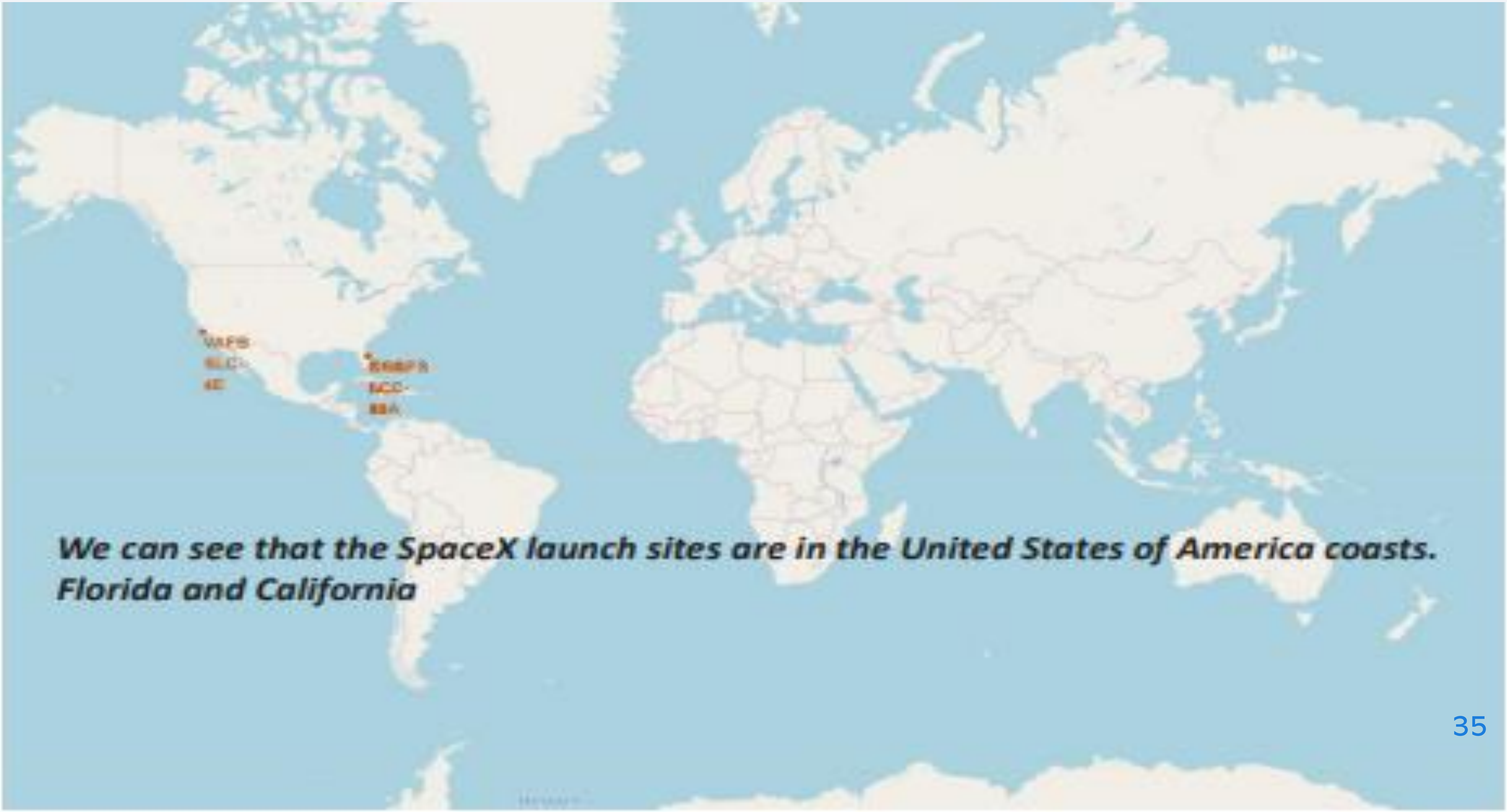
Function COUNT counts records in column WHERE filters data

LIKE (wildcard) AND (conditions) AND (conditions)

Section 4

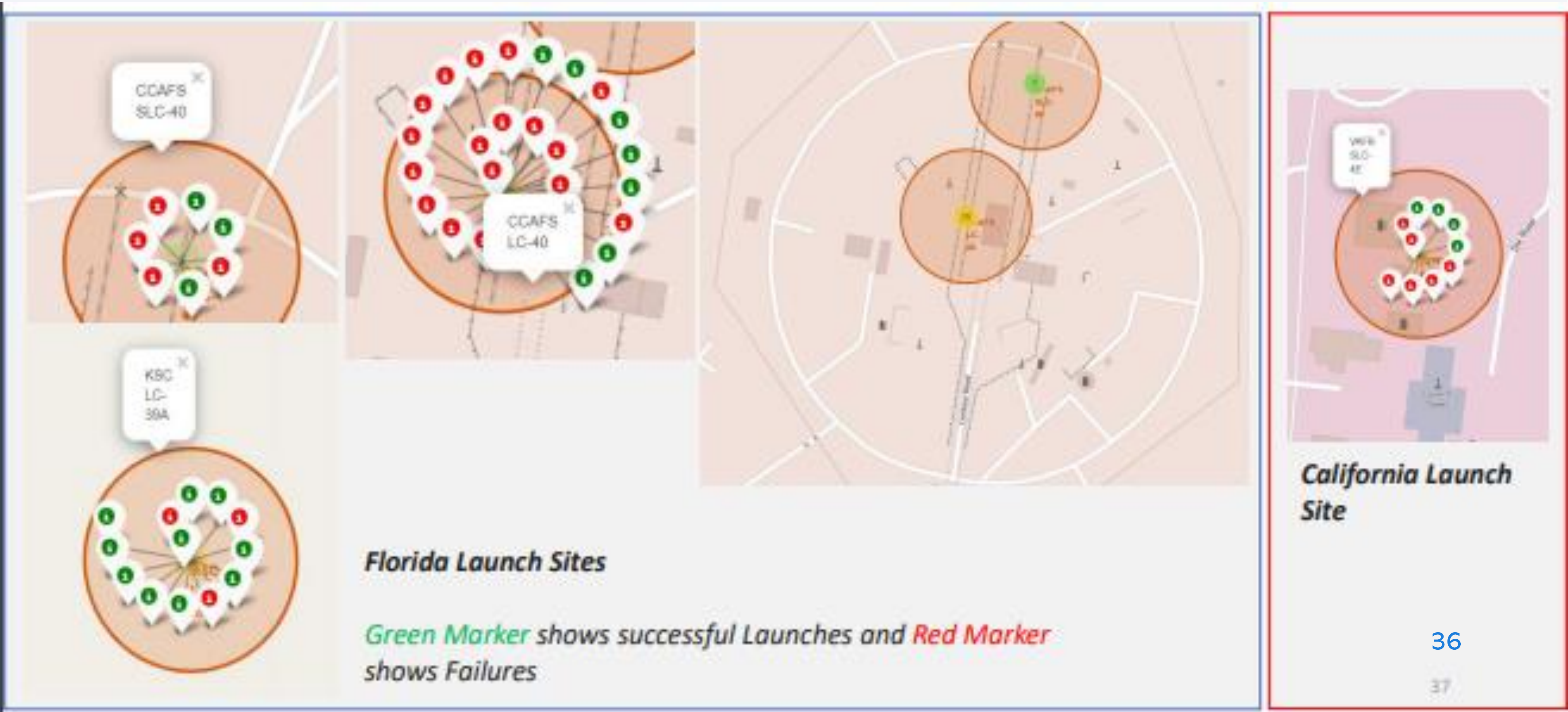
Launch Sites Proximities Analysis

All Launch Sites Global Map Markers



We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Colour Labelled Markers

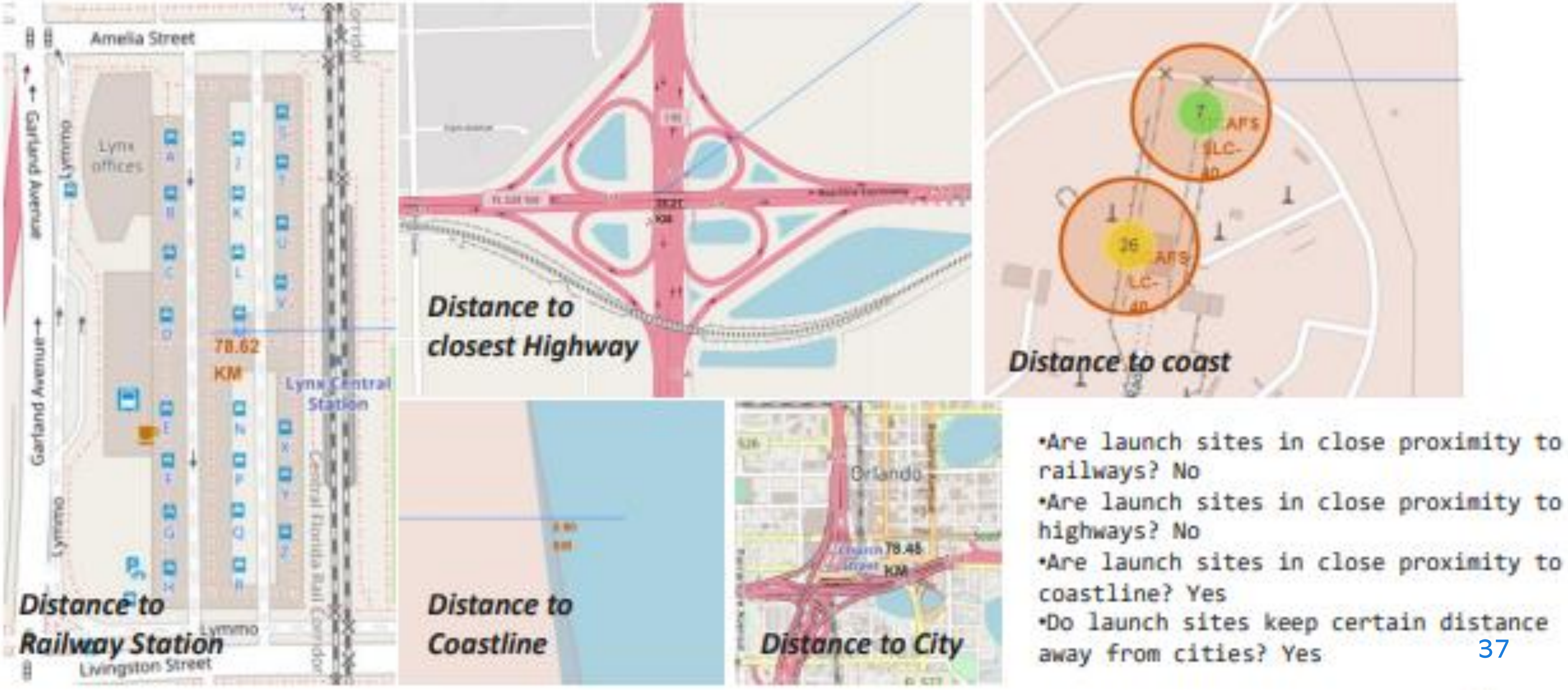


Florida Launch Sites

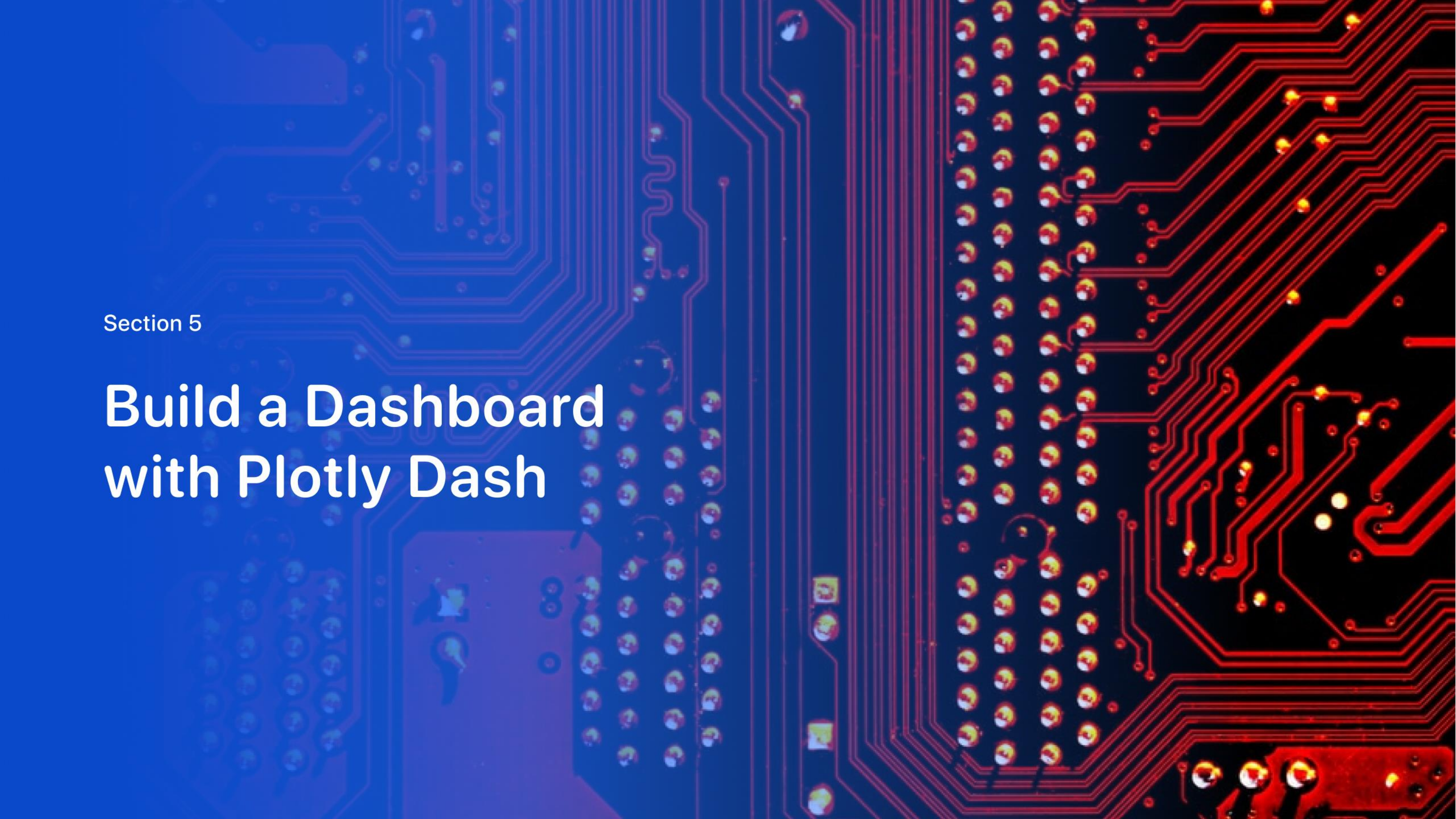
Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

Working out Launch Sites distance to landmarks to find trends with Haversine formula using CCAFS-SLC-40 as a reference



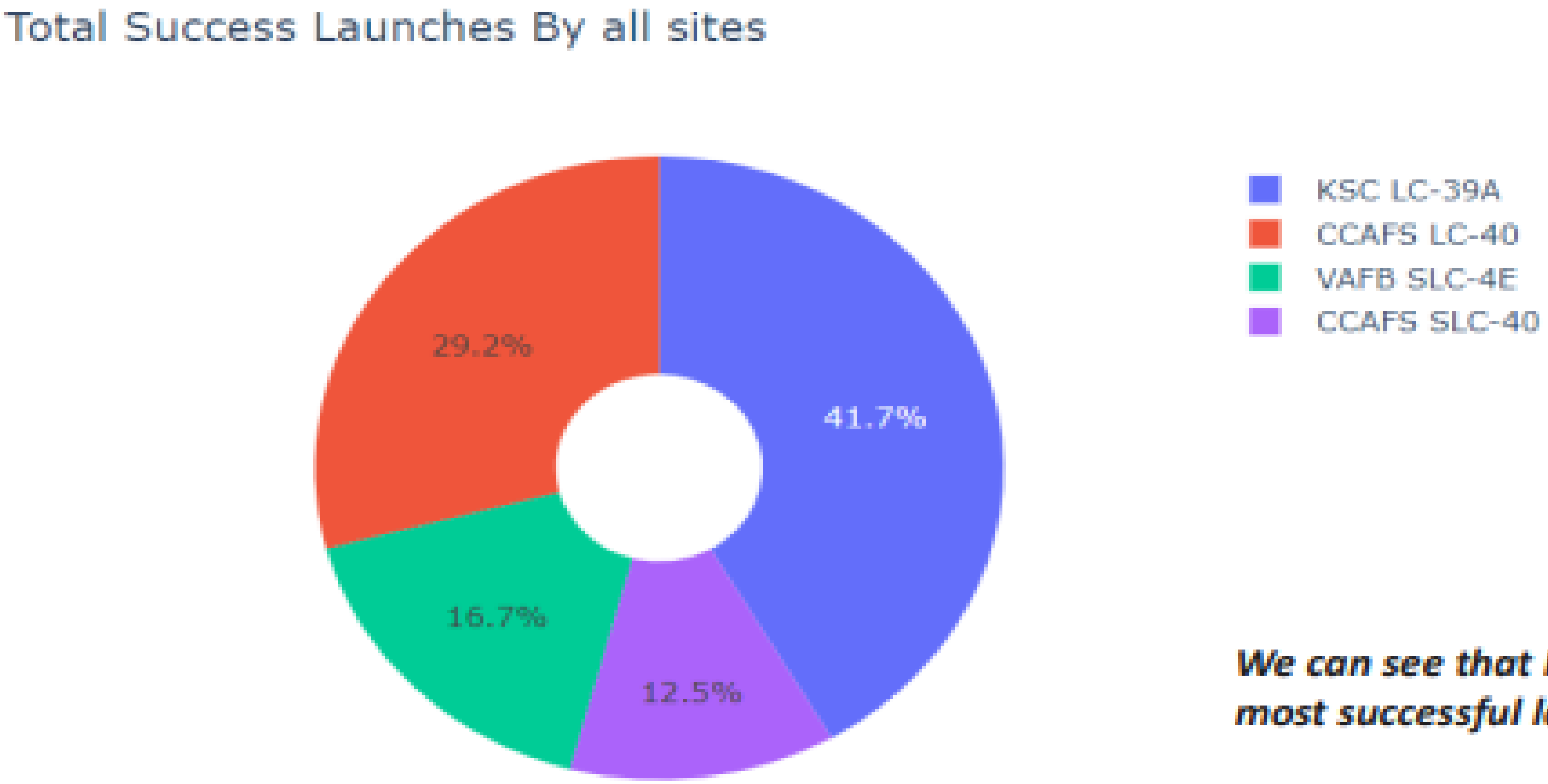
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



Section 5

Build a Dashboard with Plotly Dash

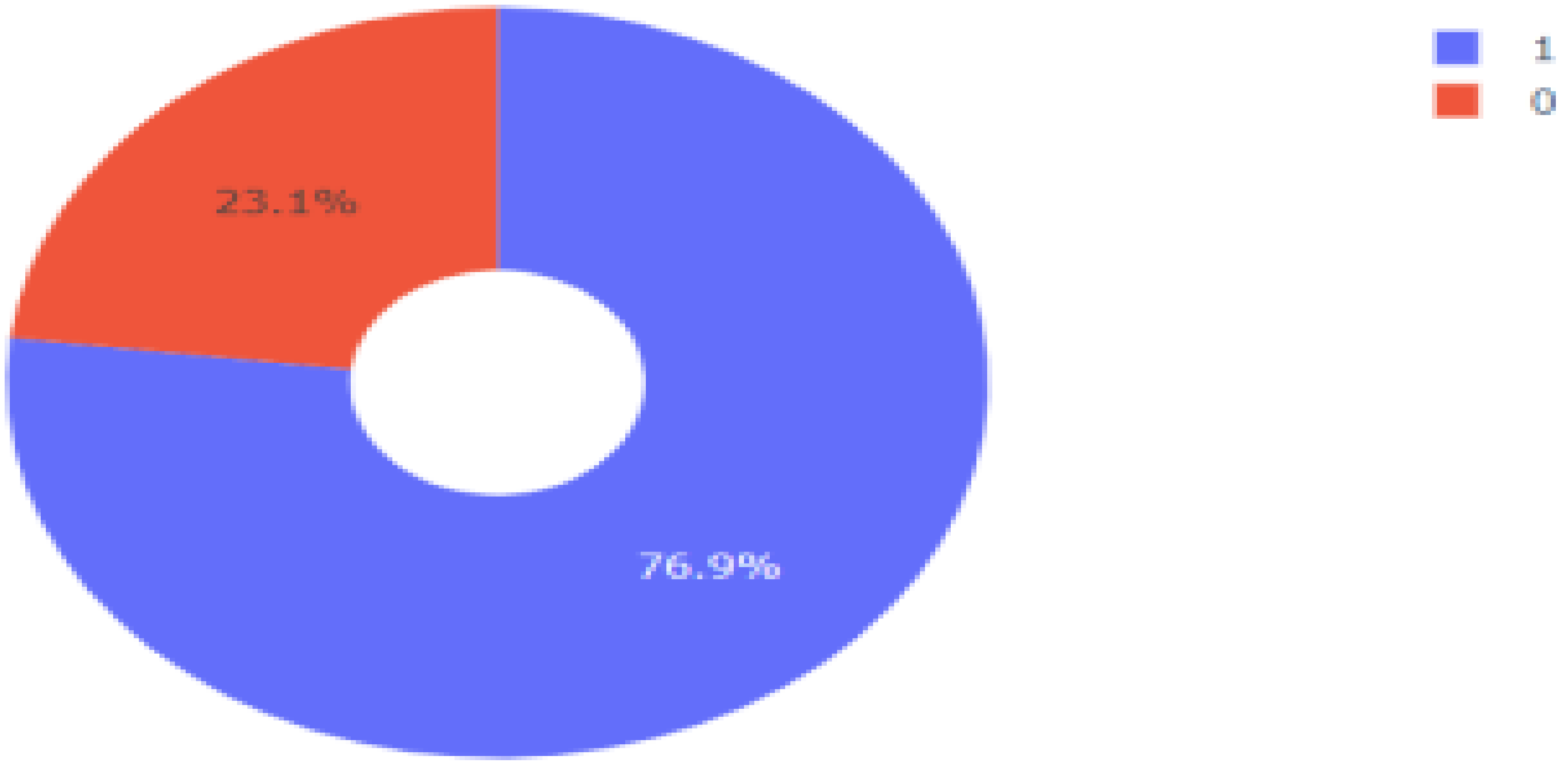
Dashboard – Pie charts showing the success percentage achieved by each launch site



Total Success Launches By all sites

We can see that KSC LC-39A had the most successful launches from all the sites

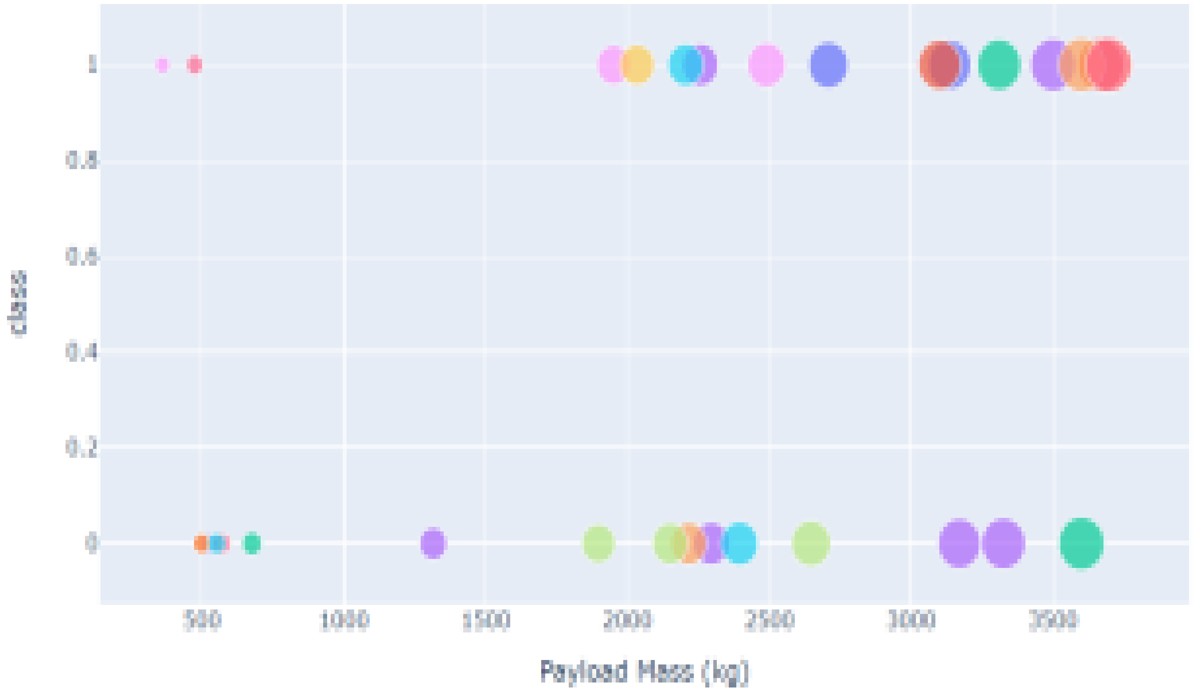
Dashboard – Pie chart for the launch site with highest launch success ratio



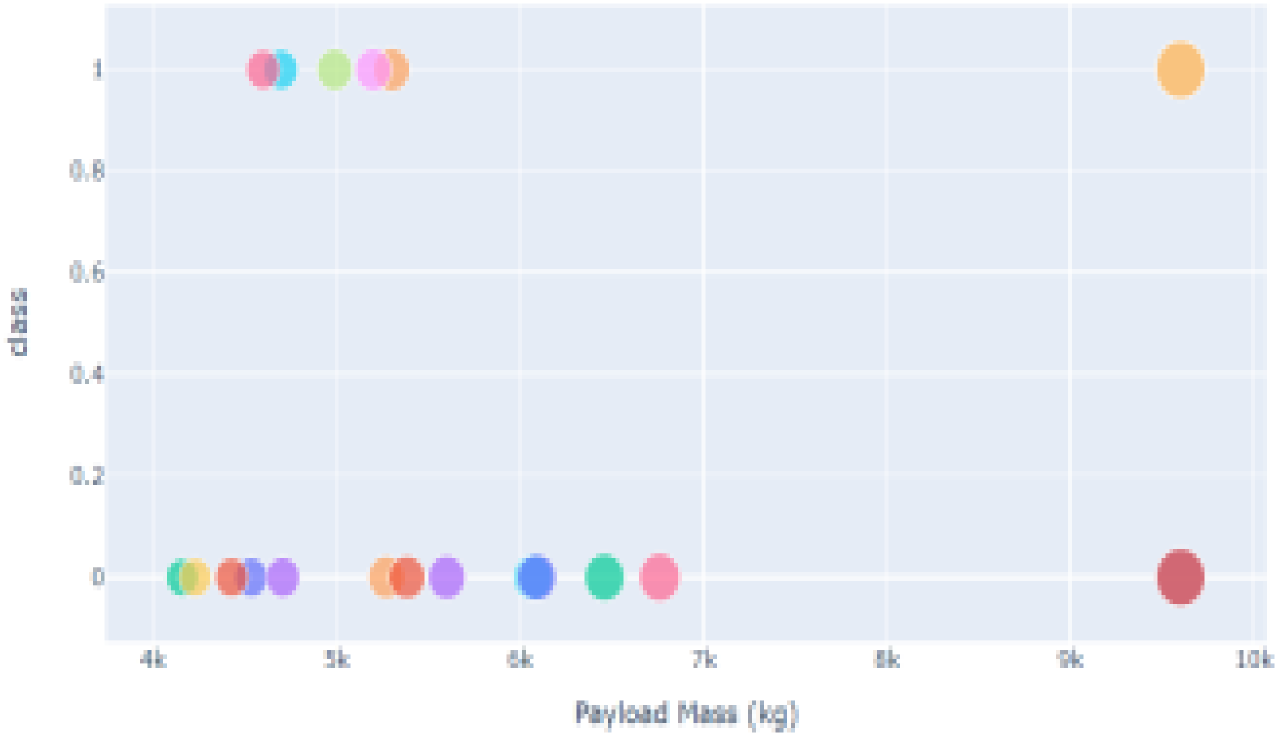
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Dashboard – Payload vs Launch Outcome scatter plot for all sites with different payload selected in the range slider

Low Weighted Payload 0kg – 4000kg



Heavy Weighted Payload 4000kg – 10000kg



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

Predictive Analysis (Classification)

Classification Accuracy

As you can see our accuracy is extremely close but we do have a winner its down to decimal places! using this function.

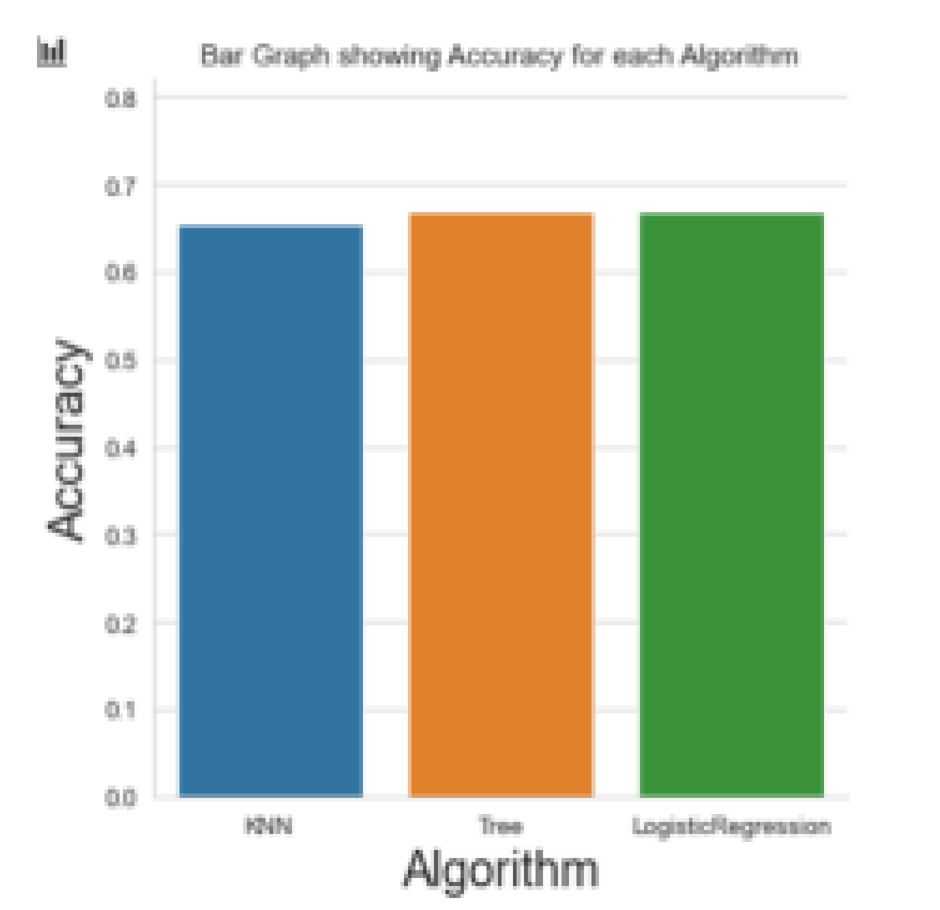
bestalgorithm= max(algorithms, key=algorithms.get)

The tree algorithm wins!!

Best Algorithm is tree with a score of 0.6678571428571429

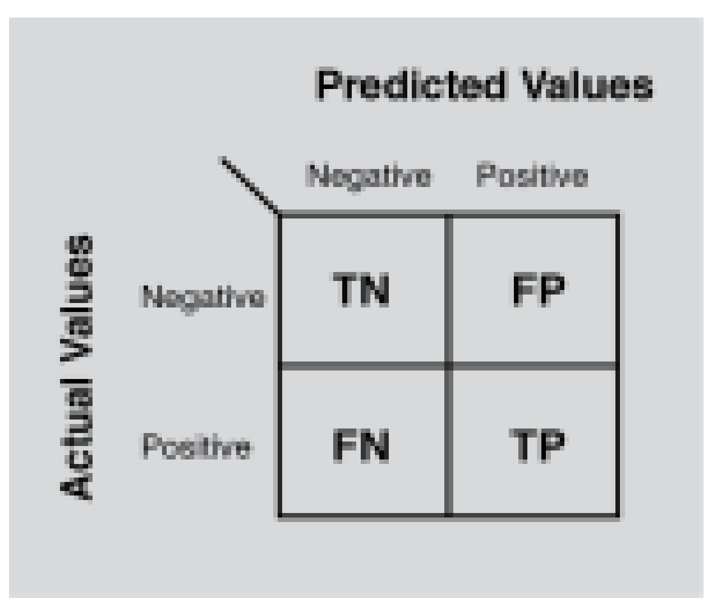
Best param is : {‘criterion’: ‘gini’, ‘max_depth’:2, ‘max_features’: ‘auto’, ‘min_sample_leaf’:1, ‘min_sample_split’:2, ‘splitter’:’best’}

After selecting the best hyperparameters for the decision tree classifier using the validation data, we achieved 83.33% accuracy on the test data.

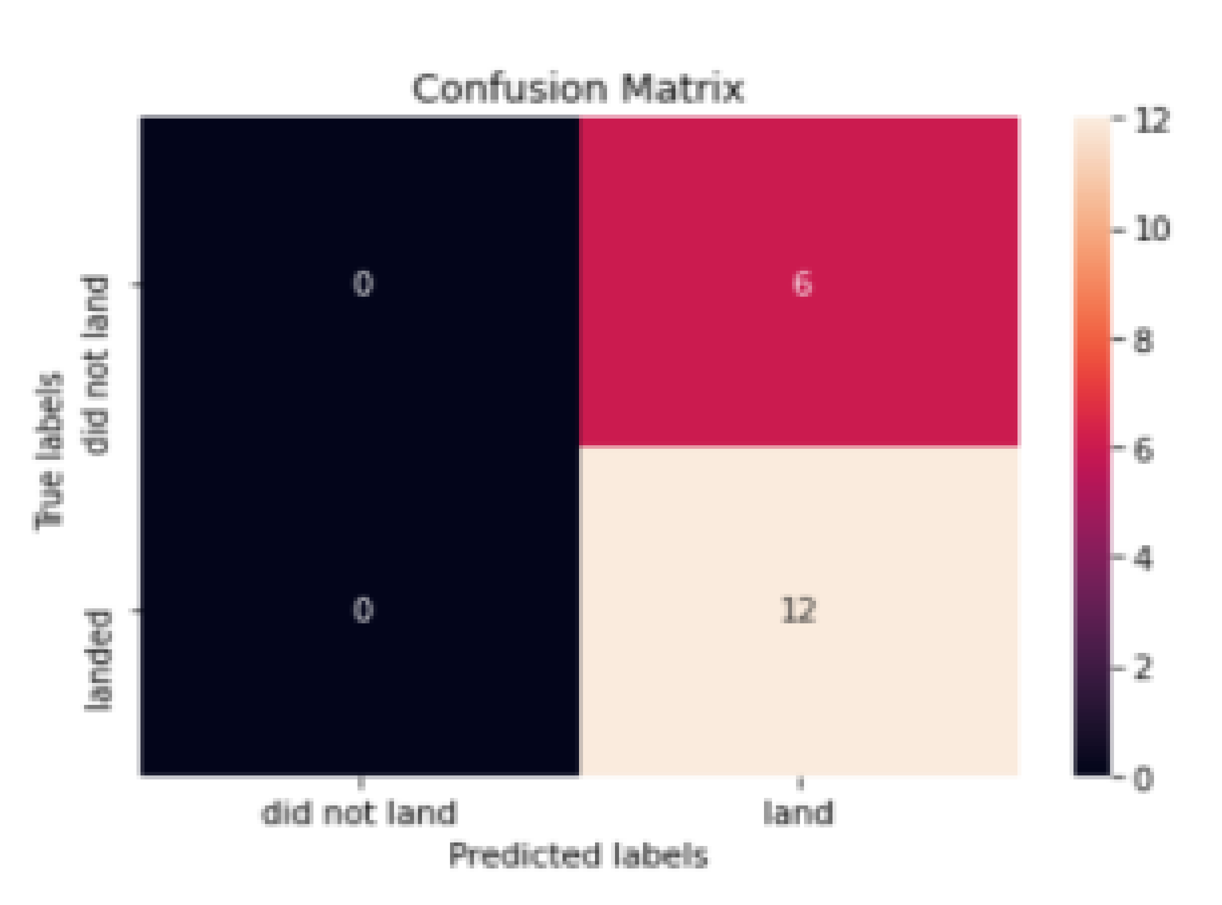


Confusion Matrix

Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the major problem is false positives.



		Predicted Values	
		Negative	Positive
Actual Values	Negative	TN	FP
	Positive	FN	TP



Conclusions

- The Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

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!