Winning Space Race with Data Science

By Sandesh Yadav

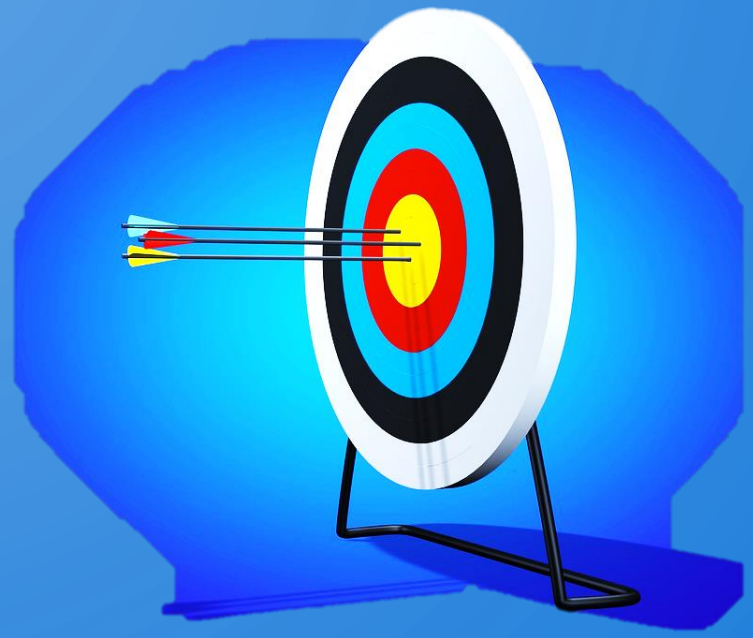
4th December 2021

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
- 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

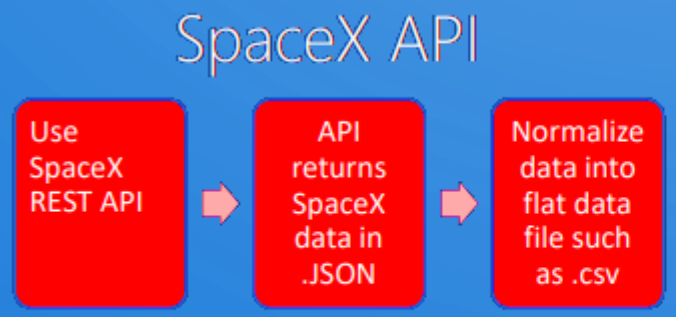
Methodology

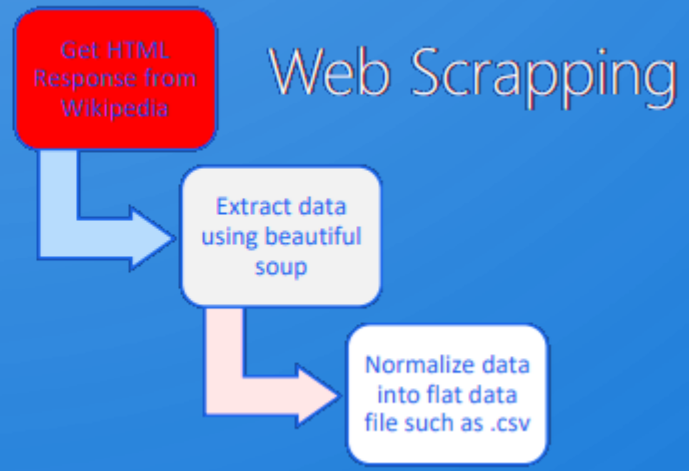
Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - (Web Scrapping) from Wikipedia
- Perform data wrangling (Transforming data for Machine Learning)
  - 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 detect any pattern
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

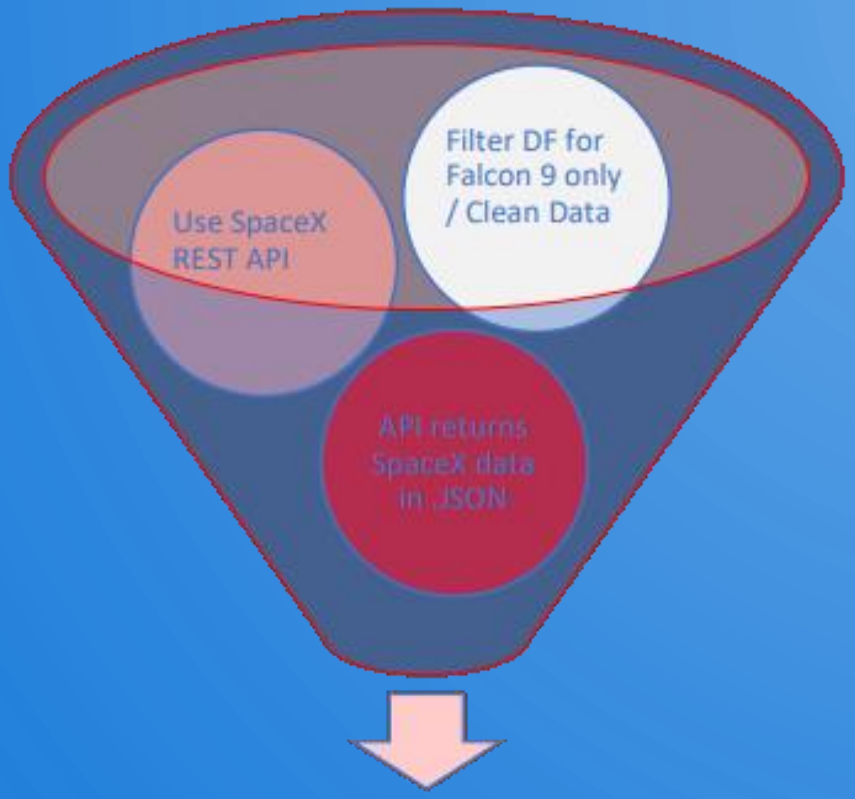
Data Collection

- They following datasets were 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.





Data Collection – SpaceX API



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)
```

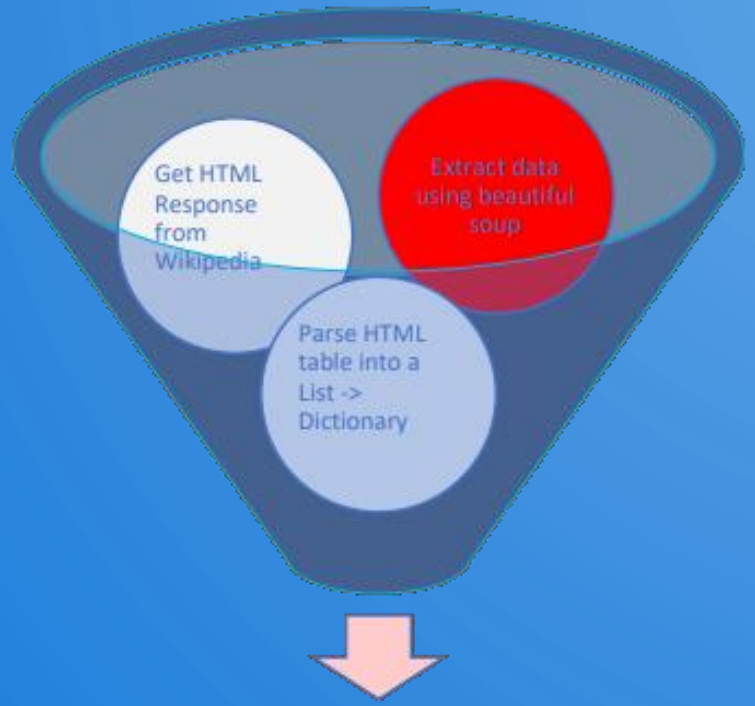
5. Filter dataframe and export to flat file (.csv)

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

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

GitHub URL to Notebook

Data Collection - Scrapping



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

```
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)
```

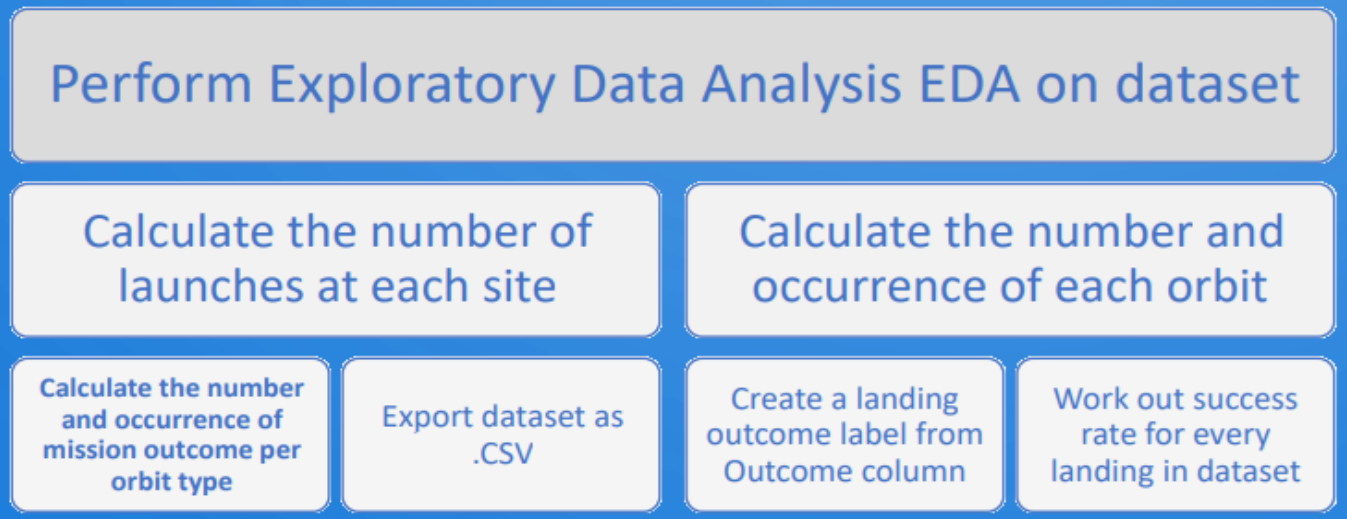
[GitHub URL to Notebook]

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 ground pad 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 False 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

GitHub URL to Notebook

- Process



EDA with Data Visualization

GitHub URL to Notebook

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 variables 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 axes. Bar charts can also show big changes in data over time.

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

EDA with SQL

- Performed SQL queries to gather information about the dataset.
- For example of some questions we were asked about the data we needed information about. Which we are using SQL queries to get the answers in the dataset :
  - Displaying the names of the unique launch sites in the space mission
  - Displaying 5 records where launch sites begin with the string 'KSC'
  - Displaying the total payload mass carried by boosters launched by NASA (CRS)
  - Displaying average 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 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

Build a Dashboard with Plotly Dash

- Used Python Anywhere to host the website live 24/7 so your can play around with the data and view the data
- The dashboard is built with Plotly Dash web framework.
- Graphs:

GitHub URL to Notebook

  - Pie Chart showing the total launches by a certain site/all sites
  - Display relative proportions of multiple 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 a non-linear pattern.
  - The range of data flow, i.e. maximum and minimum value, can be determined.
  - Observation and reading are straightforward.

Predictive Analysis (Classification) GitHub URL to Notebook

- Building Model:
  - Load our dataset into NumPy and Pandas
  - Transform Data
  - Split our data into training and test 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
  - 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 the Model:
  - Feature Engineering
  - Algorithm Tuning
  - The model with the best accuracy score will be chosen as the performing model.

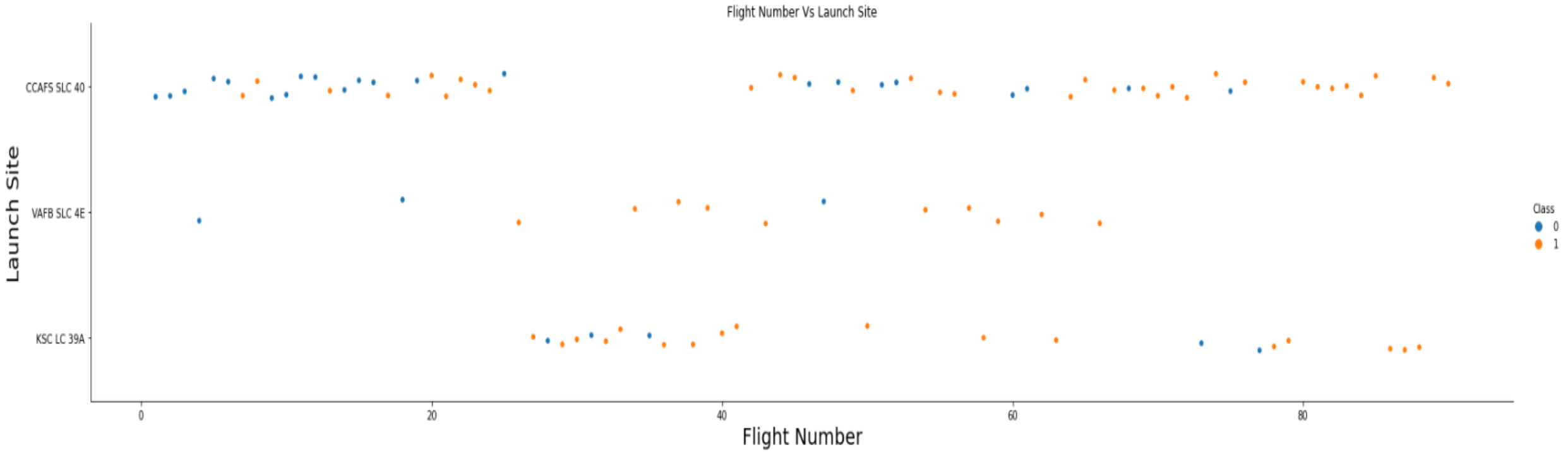
Results

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

Section 2

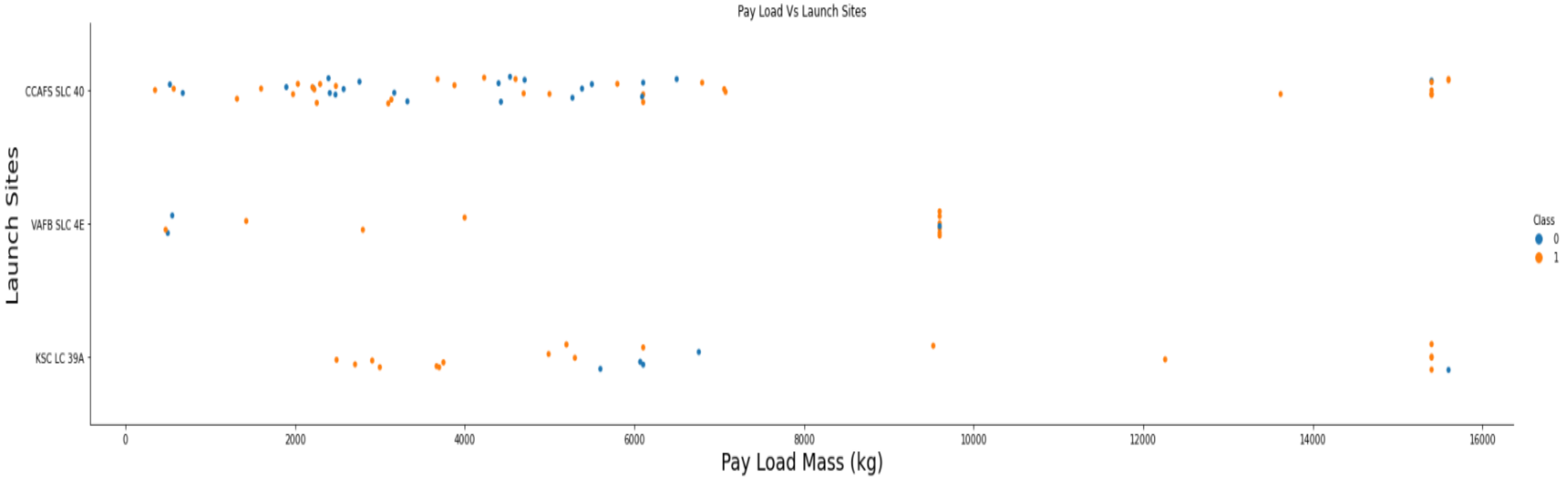
Insights drawn from EDA

Flight Number vs. Launch Site



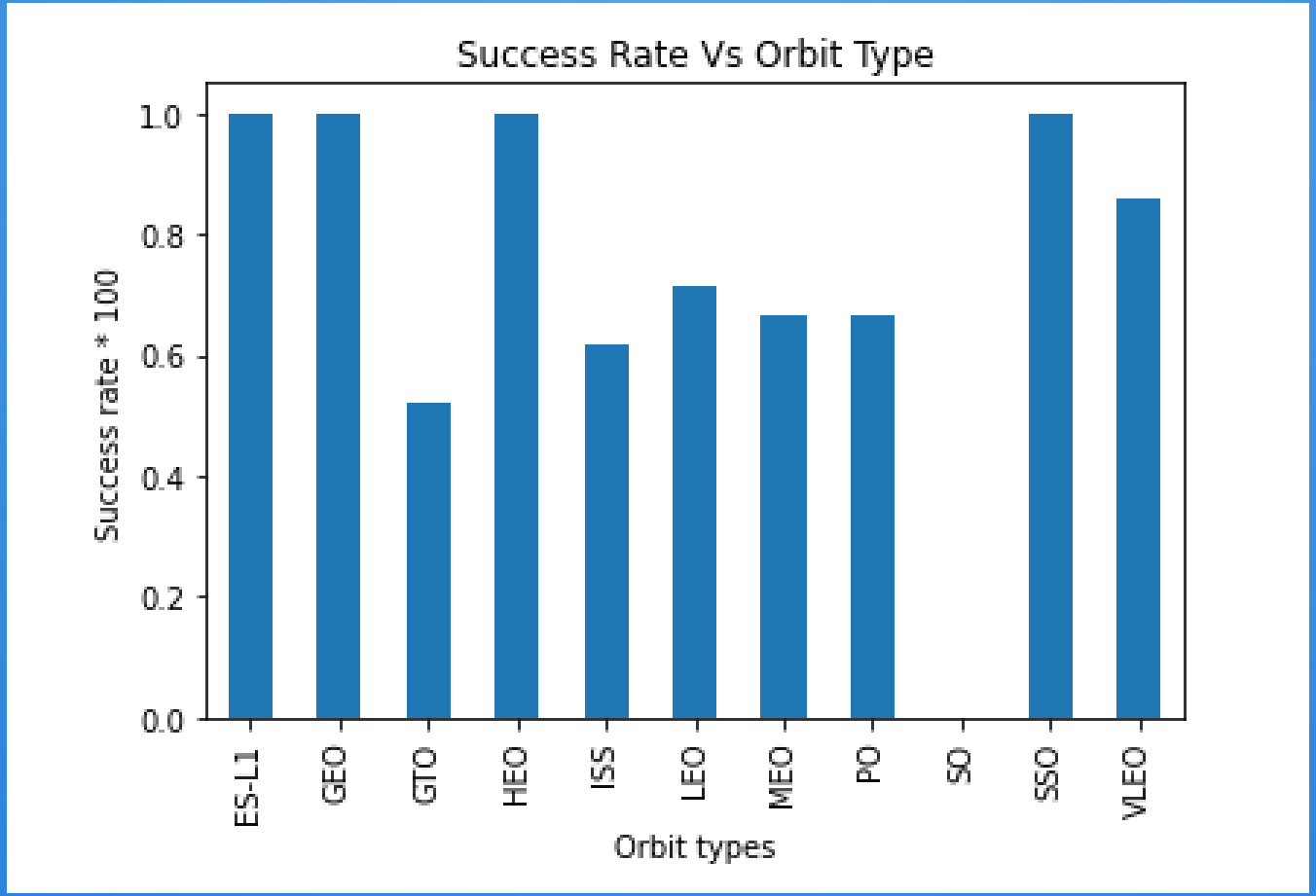
The more amount of flights at a launch site
the greater the success rate at a launch site.

Payload vs. Launch Site



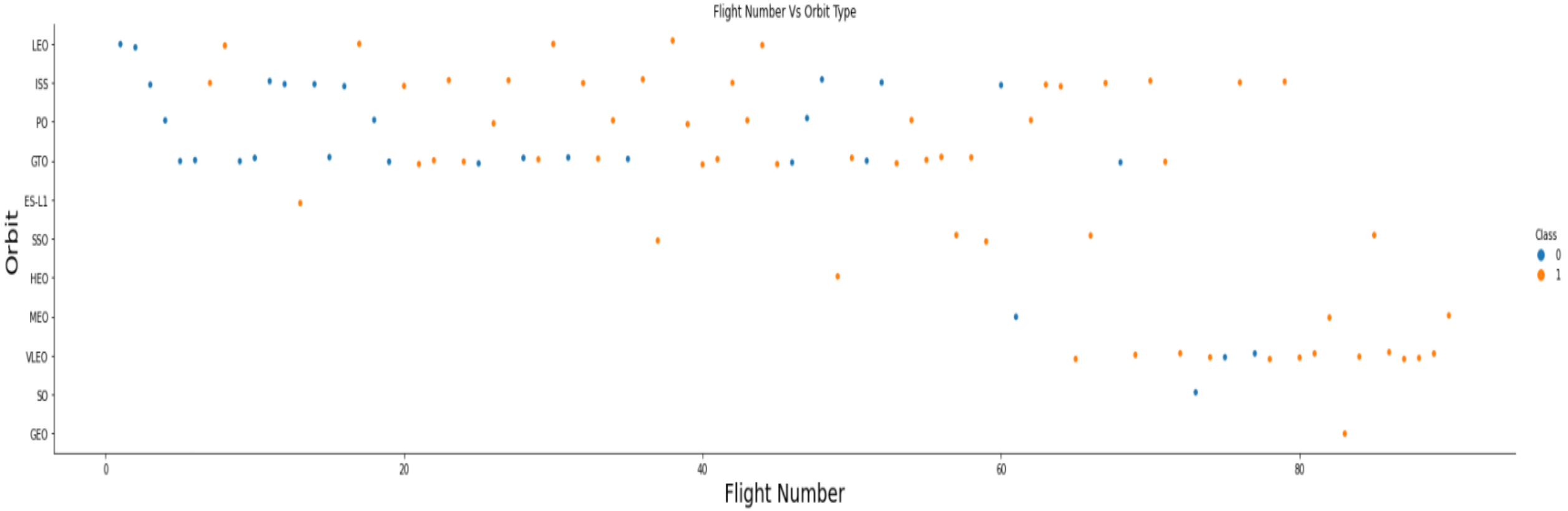
For the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).

Success Rate vs. Orbit Type



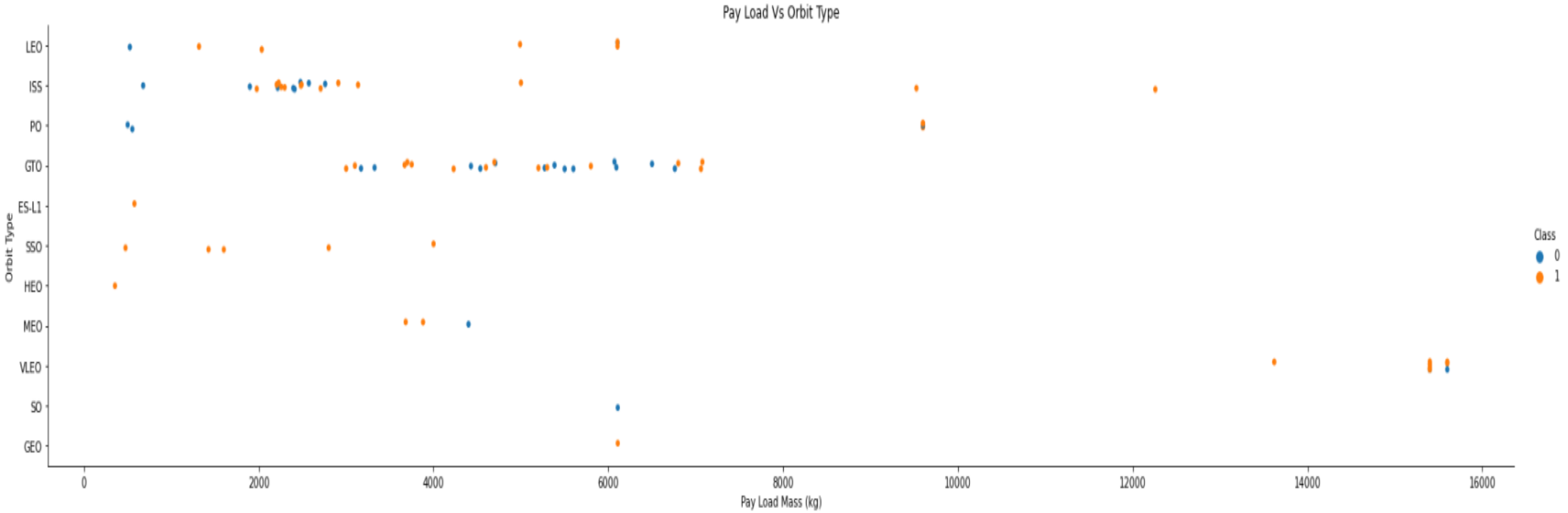
Orbits ES-L1, GEO, HEO, and SSO have the highest Success Rate

Flight Number vs. Orbit Type



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.

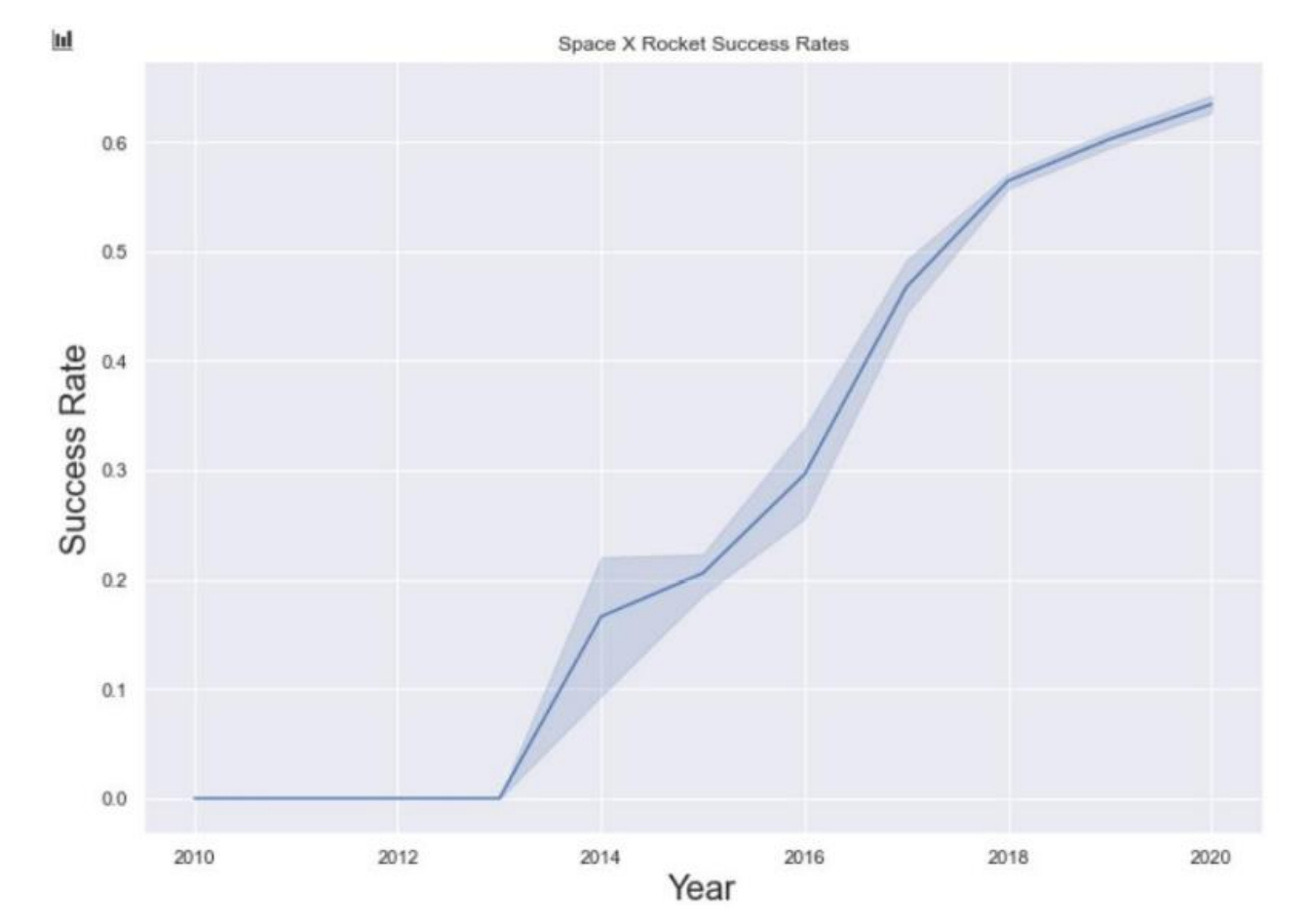
Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for PO, LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend



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

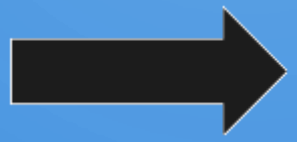
Section 3

EDA with SQL

All Launch Site Names

SQL QUERY:

```
SELECT
DISTINCT(Launch_Site)
FROM SPACEXTBL
```



Out[5]:

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

- QUERY EXPLAINATION:
- Using the word DISTINCT in the query means that it will only show Unique values in the Launch_Site column from tblSpaceX

Launch Site Names Begin with 'CCA'

SQL QUERY:

```
SELECT *
FROM SPACEXTBL
WHERE Launch_Site
LIKE “CCA%”
LIMIT 5
```

QUERY EXPLAINATION:

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.

Out[7]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-12	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt

Total Payload Mass

SQL QUERY:

```
SELECT SUM(payload_mass__kg_)
AS “Total Payload Mass(Kg)”
FROM SPACEXTBL
WHERE customer = “NASA (CRS)”
```

Out[16]:

Total Payload Mass(Kg)
45596

QUERY EXPLAINATION

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_)
AS "Average Payload Mass(Kg)"
FROM SPACEXTBL
WHERE booster_version = 'F9 v1.1'
```

Out[17]:

Average Payload Mass(Kg)
2928

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)
AS "First successful landing outcome date in ground pad"
FROM SPACEXTBL
WHERE landing__outcome = 'Success (ground pad)'
```

Out[18]:

First successful landing outcome date in ground pad
22-12-2015

QUERY EXPLANATION:

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

The WHERE clause filters the dataset to only perform calculations on Landing_Outcome Success (Ground Pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL QUERY:

```
SELECT booster_version
FROM SPACEXTBL
WHERE landing__outcome = 'Success (drone ship)'
AND payload_mass__kg_ BETWEEN 4000 AND 6000
```

Out[19]:

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

QUERY EXPLANATION:

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(SELECT Count(mission_outcome)
from SPACEXTBL
where mission_outcome LIKE '%Success%')
as Successful_Mission_Outcomes,
(SELECT Count(mission_outcome)
from SPACEXTBL
where mission_outcome LIKE '%Failure%')
as Failure_Mission_Coutcomes
FROM SPACEXTBL
LIMIT 1
```

Out[26]:

successful_mission_outcomes	failure_mission_coutcomes
100	1

QUERY EXPLANATION:

This query is a bit complicated and hard, we used subqueries here to produce the results. The LIKE '%Success%' and LIKE '%Failure' wildcards shows that in the record the given phrase is in any part of the string in the records.

Boosters Carried Maximum Payload

SQL QUERY:

```
SELECT DISTINCT(booster_version)
FROM SPACEXTBL
WHERE payload_mass__kg_ =
(SELECT MAX(payload_mass__kg_)
FROM SPACEXTBL)
```

Out[27]:

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

QUERY EXPLANATION:

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

Using the function MAX works out for the MAXIMUM payload in the column payload_mass__kg_

2015 Launch Records

SQL QUERY:

```
SELECT landing__outcome, booster_version, launch_site
FROM SPACEXTBL
WHERE YEAR(DATE) = 2015
AND landing__outcome LIKE '%Failure%'
```

Out[30]:

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

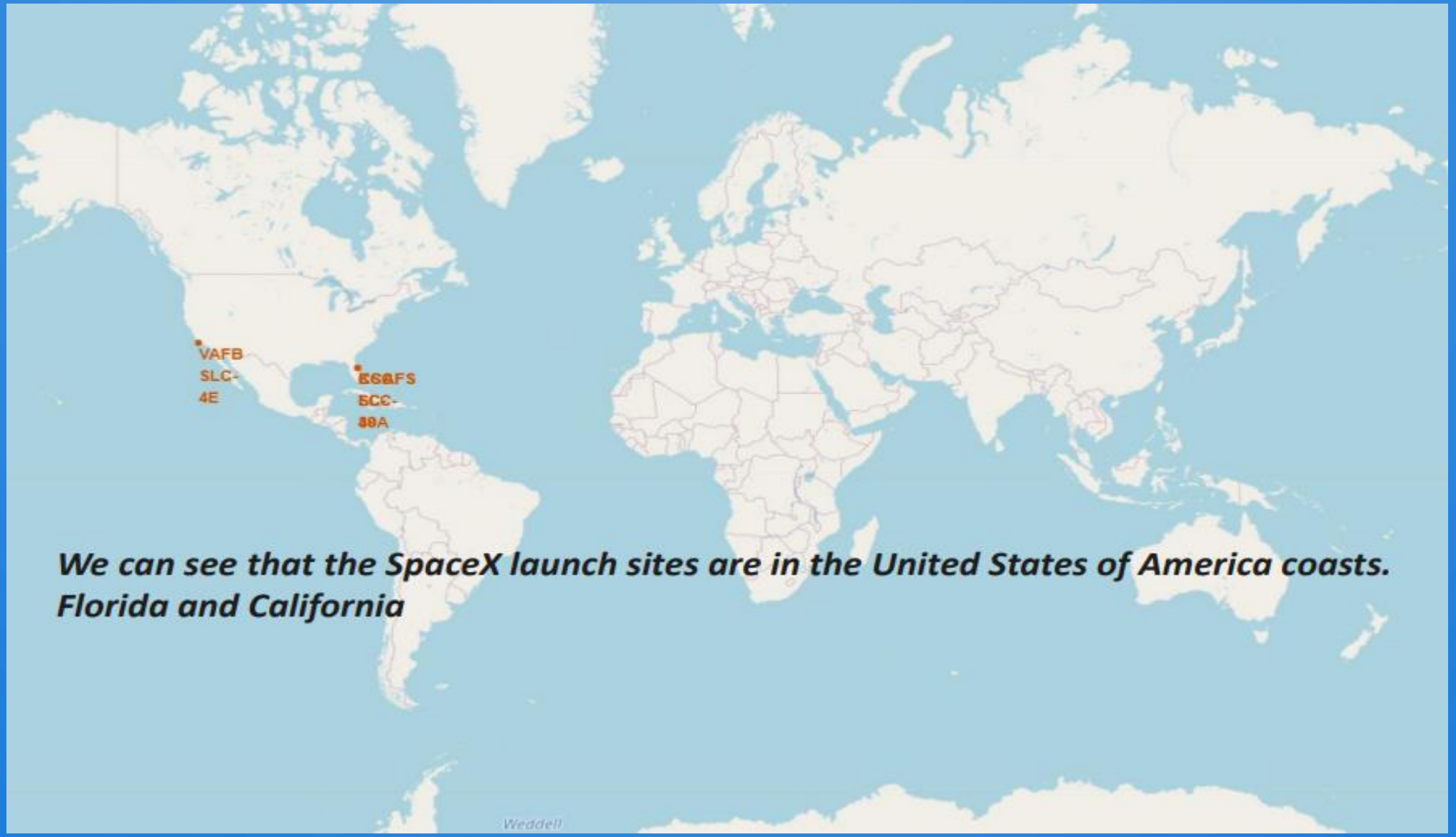
QUERY EXPLANATION:

The YEAR() function returns only the year part of the date i.e. YYYY.
WHERE clause filters Year to be 2015.

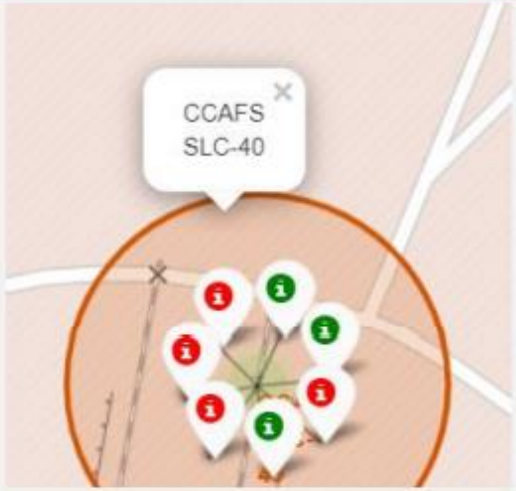
Section 4

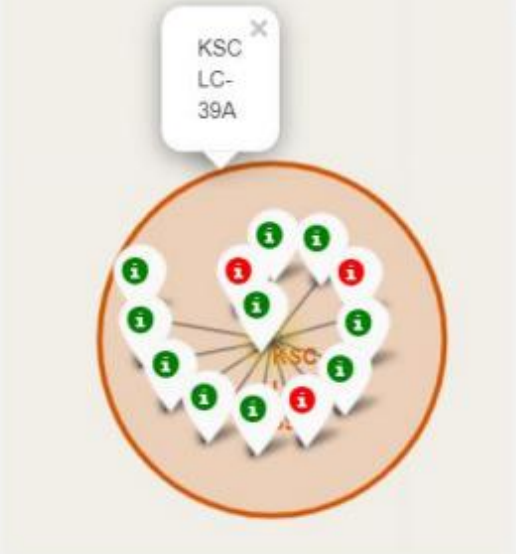
Launch Sites Proximities Analysis

All SpaceX launch sites globally



Launch Sites with labelled markers



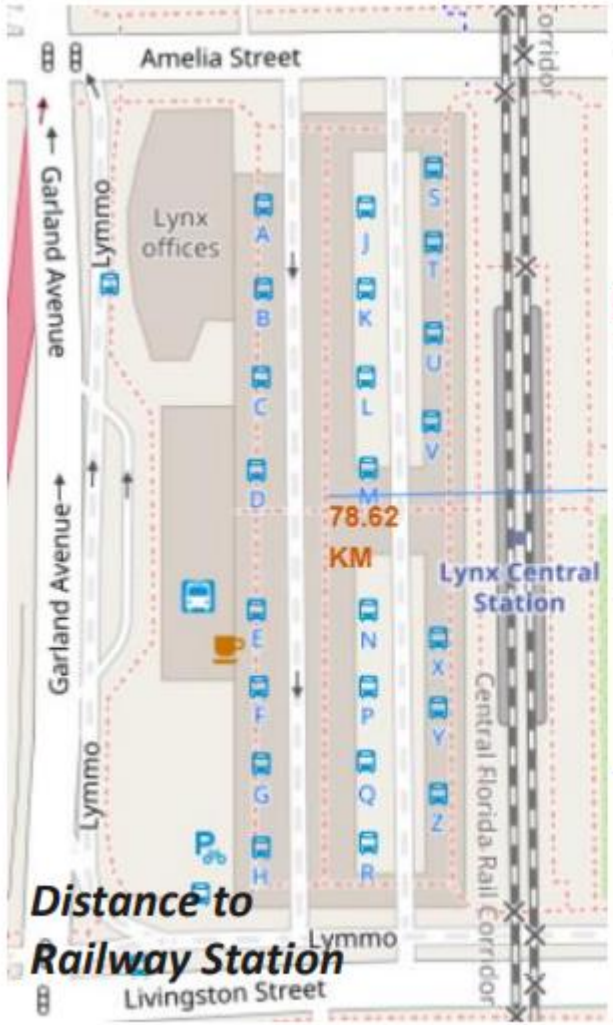


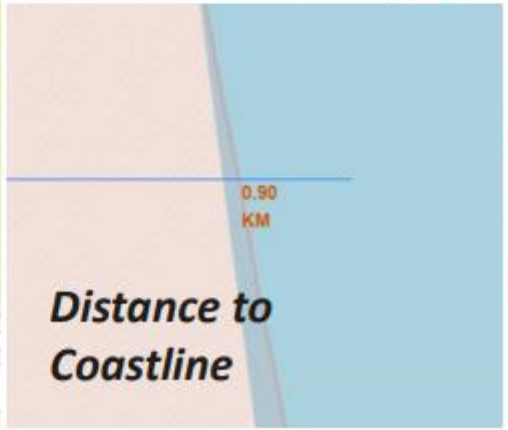
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

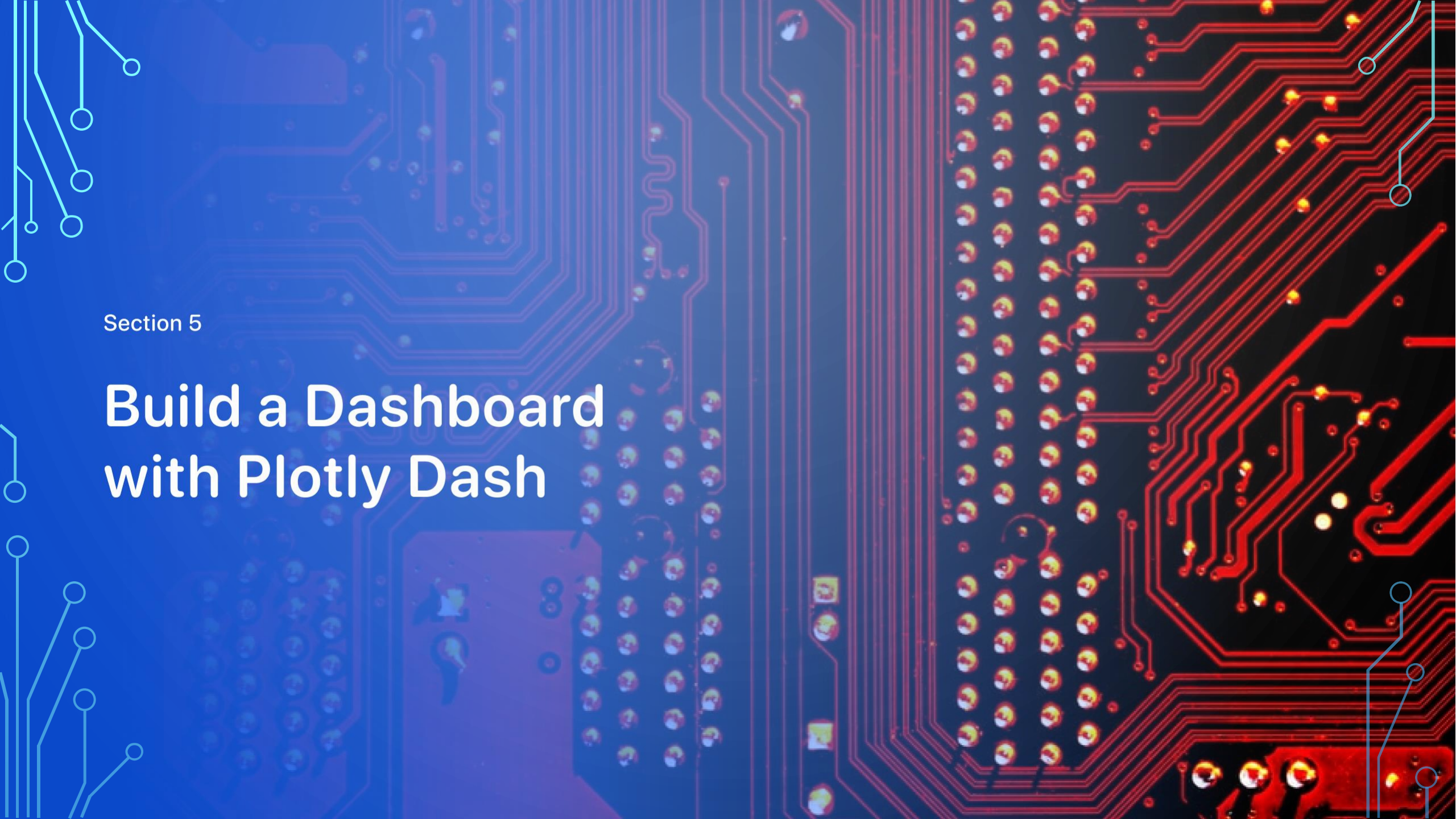
California Launch Site

Launch Sites with their distances to their proximities

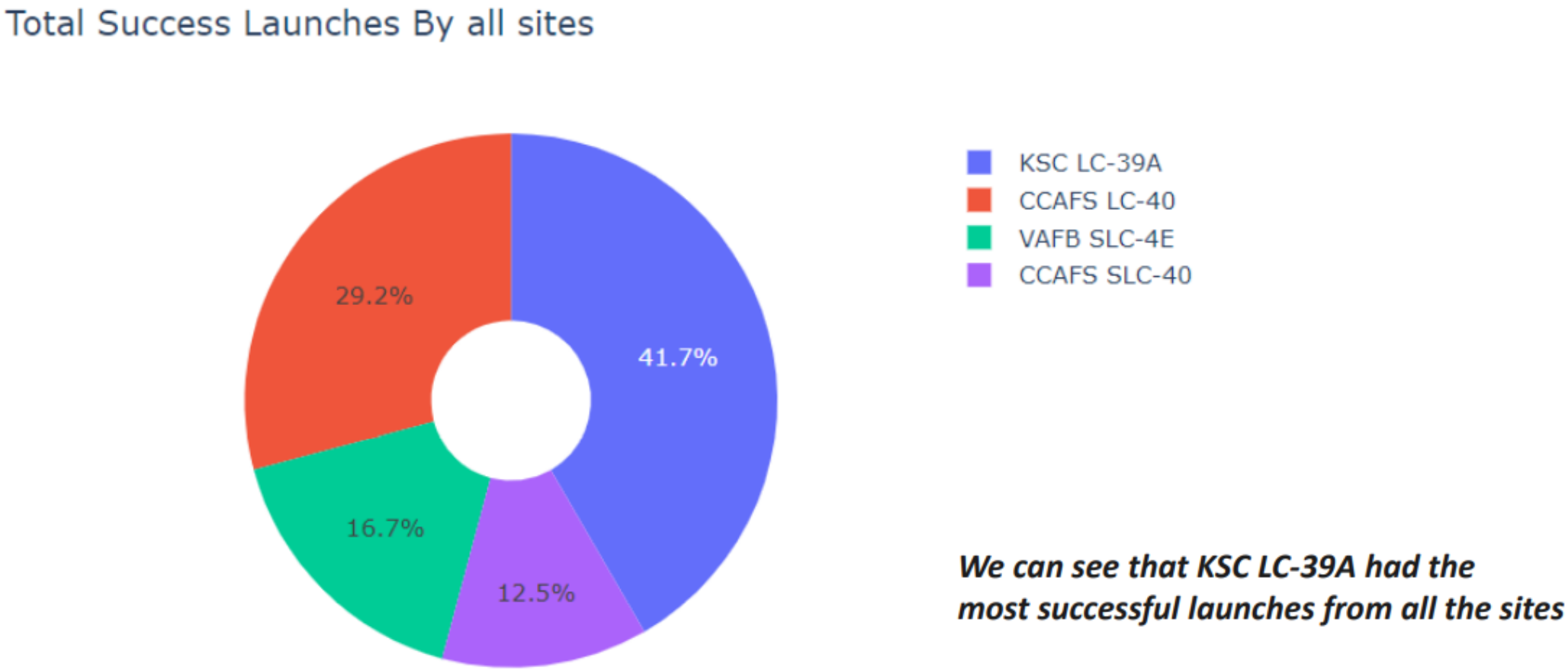




- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

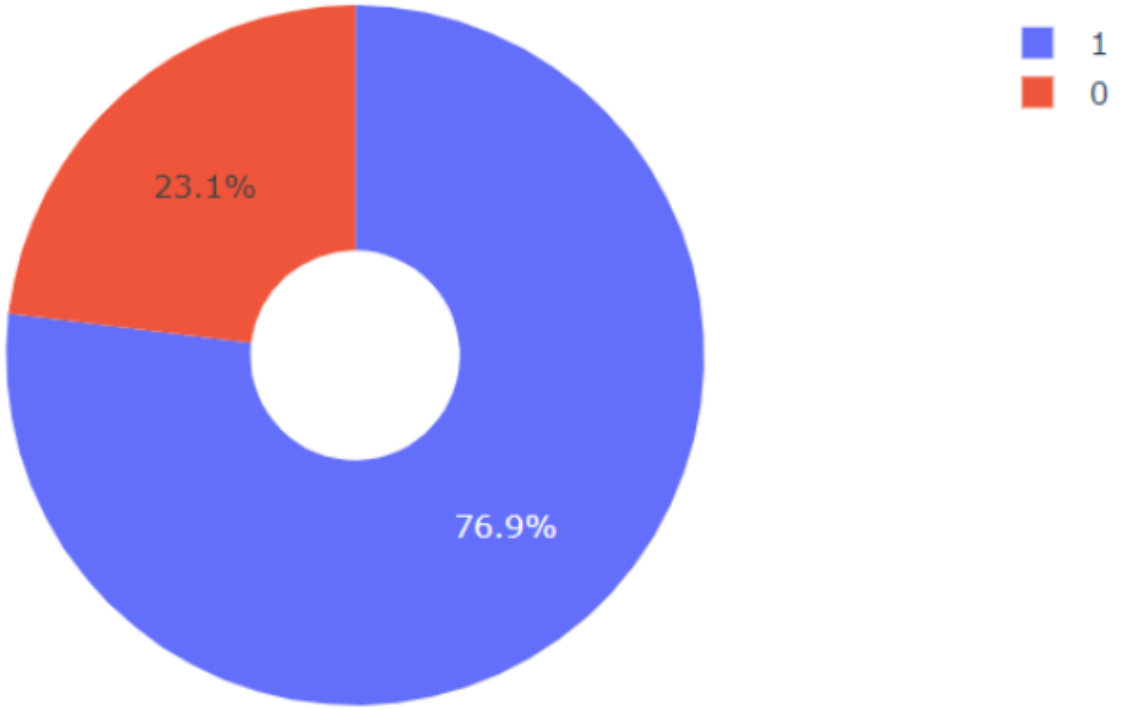

Section 5
Build a Dashboard with Plotly Dash

DASHBOARD – Pie chart showing the success percentage achieved by each launch site



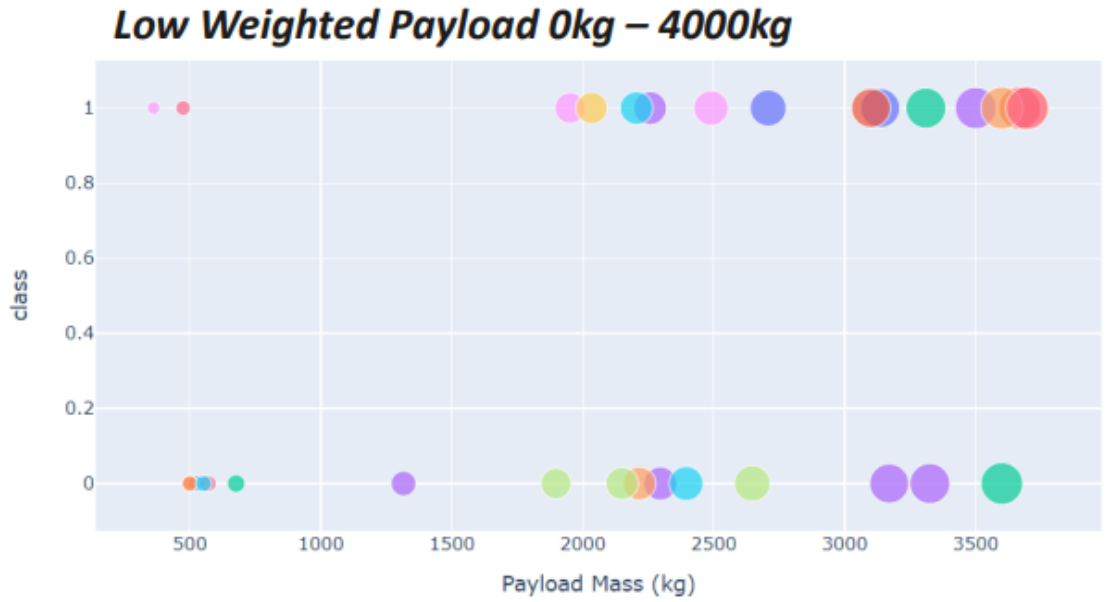
We can see that KSC LC-39A had the most successful launches from all the sites

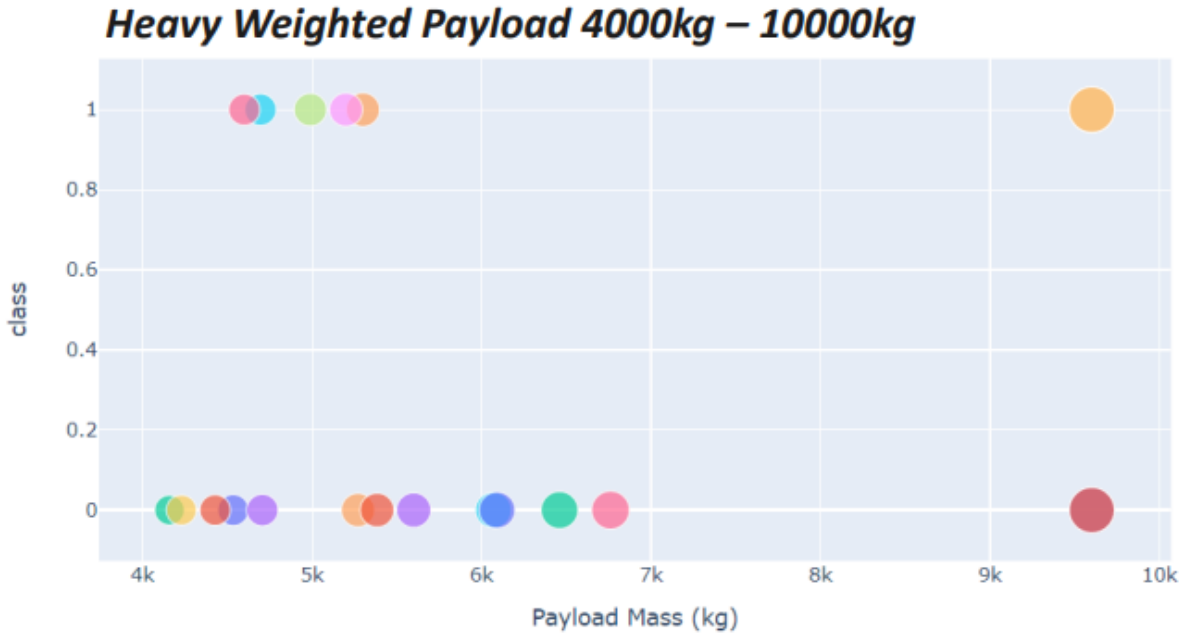
DASHBOARD – Pie chart for the launch site with highest launch success ratio



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





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

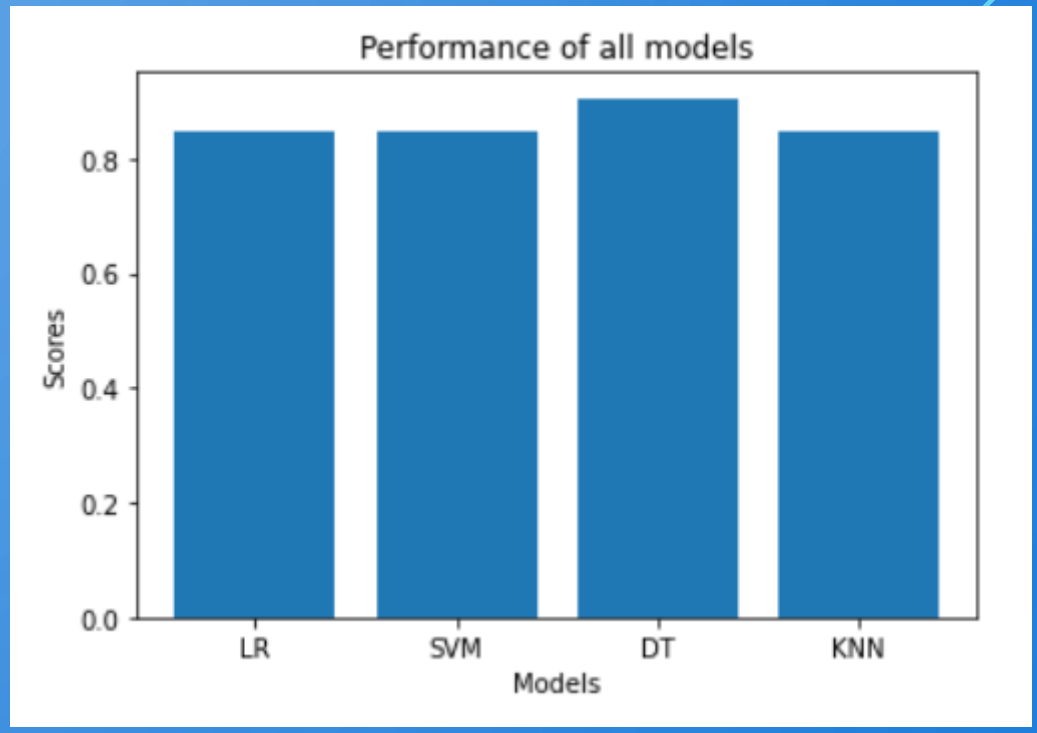
Section 6

Predictive Analysis (Classification)

Classification Accuracy using test data

As you can see that the accuracies of all the models is quite close, but the Decision Tree model seems to perform well than others.

	Model	Score
0	Logistic Regression	0.846429
1	Support Vector Machine	0.848214
2	Decision Tree	0.905357
3	K Nearest Neighbor	0.848214

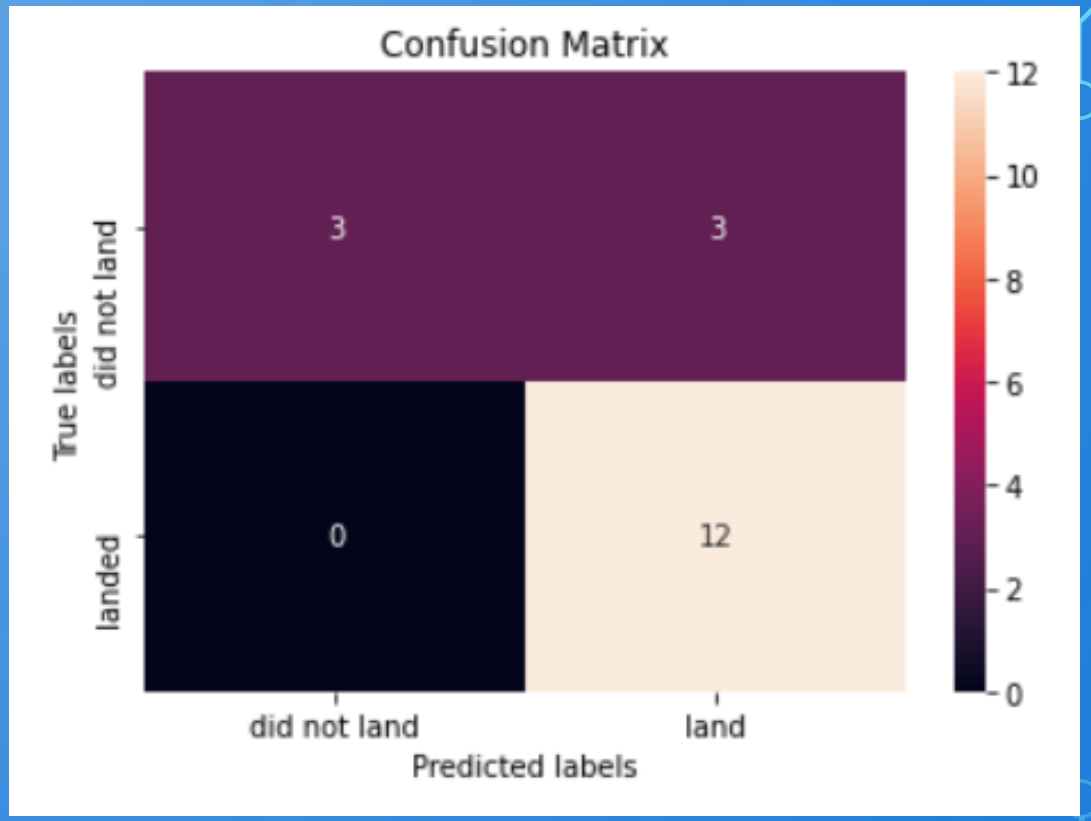


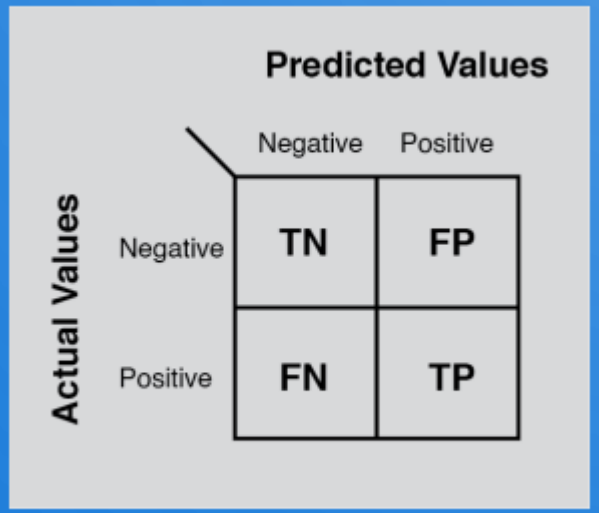

```
The best performing model is the Decision Tree model with the accuracy of : 90.53571428571429%

The Hyper Parameters used in decision tree are : {'criterion': 'entropy', 'max_depth': 14, 'max_features': 'sqrt', 'min_samples
_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
```

Confusion Matrix for the Decision Tree

Examining the confusion matrix for the Decision Tree, we observe that model can distinguish between the different classes quite well. We can also observe that there is a problem i.e. false positives.





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

- Haversine formula
- ADGGoogleMaps Module (not used but created)
- Module sqlserver (ADGSQLSERVER)
- PythonAnywhere 24/7 dashboard

Thank you!