IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Kem Diaz Mena
04-Jan-2022

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 commercial space age is here, companies are making space travel affordable for everyone. Virgin Galactic is providing suborbital spaceflights. Rocket Lab is a small satellite provider. Blue Origin manufactures sub-orbital and orbital reusable rockets. Perhaps the most successful is SpaceX.

SpaceX's accomplishments include: Sending spacecraft to the International Space Station. Starlink, a satellite internet constellation providing satellite Internet access. Sending manned missions to Space. One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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. Spaces X's Falcon 9 launch like regular rockets

Problems you want to find answers

Determine the price of each launch. To do this, you will gather information about Space X and create dashboards for your team and determine whether SpaceX will reuse the first stage. Instead of using rocket science to determine whether the first stage will land successfully, you'll train a machine learning model, powered by public information, to predict whether SpaceX will reuse the first stage.

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 useless columns
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Plotting: Scatter Graphs, Bar Graphs to show relationships between different variables and visualize patterns in 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 data sets were acquired by.

- I worked with SpaceX launch data that is collected from the SpaceX REST API.
- This API provides me with data about launches, including information about the rocket used, the payload delivered, the launch specifications, the landing specifications, and the outcome of the landing.
- This is for the purpose of using the data to predict whether SpaceX will attempt to land a rocket or not.
- SpaceX's REST API endpoints, or URLs, start with api.spacexdata.com/v4/.
- Another data source for obtaining Falcon 9 launch data was scraping the Wikipedia website using BeautifulSoup.

Data Collection – SpaceX API

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Finally lets construct our dataset using the data we have obtained. We we combine the 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}
```

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

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

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(mean)
```

https://github.com/KD96/DS-Capstone-IBM/blob/master/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

```
200
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(page.text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
# 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')
```

Data Collection - Scraping

https://github.com/KD96/DS-Capstone-IBM/blob/master/jupyter-labs-webscraping.ipynb

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
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']=[]
```

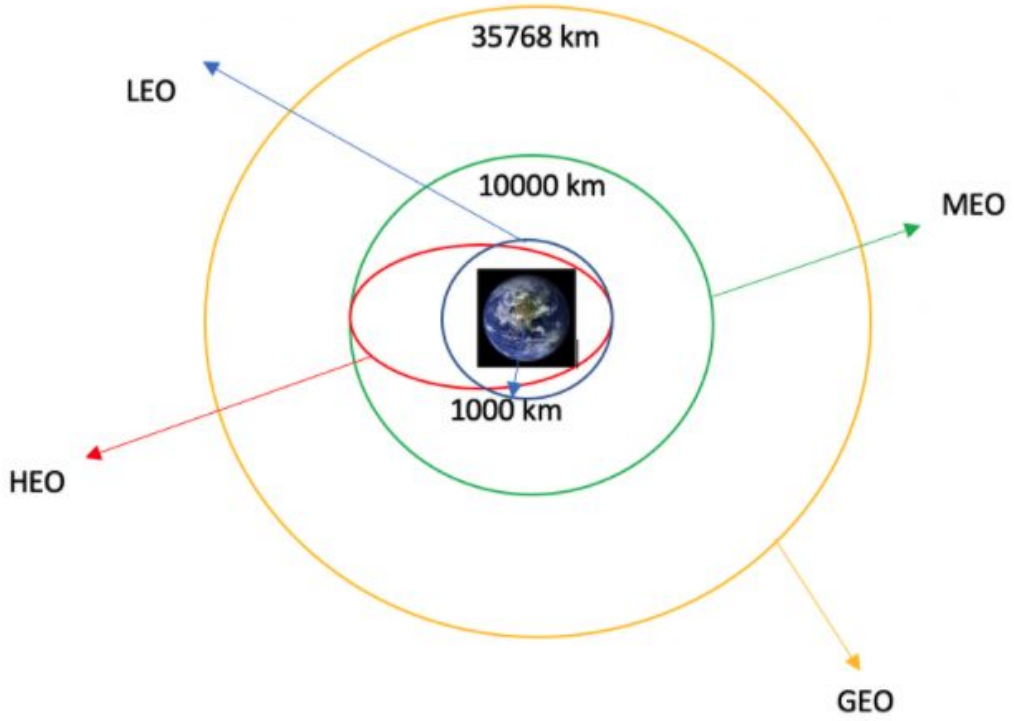
After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

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

Data Wrangling

https://github.com/KD96/DS-Capstone-IBM/blob/master/labs-jupyter-spacex-Data%20wrangling.ipynb

Diagram showing common orbit types SpaceX uses



In this lab, we will perform some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models. we will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Task:

1. **Calculate the number of launches on each site**
2. **Calculate the number and occurrence of each orbit**
3. **Calculate the number and occurence of mission outcome per orbit type**
4. **Create a landing outcome label from Outcome column**

EDA with Data Visualization

https://github.com/KD96/DS-Capstone-IBM/blob/master/jupyter-labs-eda-dataviz.ipynb

Scatter Graphs being drawn:

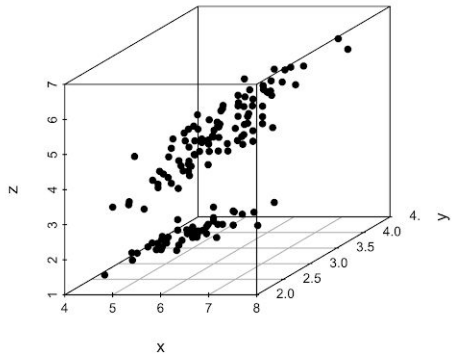
- Flight Number vs Payload Mass
- Flight Number vs Launch Site
- Payload vs Launch Site
- Orbit vs Flight Number
- Payload vs Orbit Type
- Orbit vs Payload Mass

Bar Graph being drawn:

- Mean vs Orbit

Line Graph being drawn:

- Success Rate vs Year



EDA with SQL

https://github.com/KD96/DS-Capstone-IBM/blob/master/jupyter-labs-eda-sql-coursera.ipynb

- 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

Build an Interactive Map with Folium

1. To visualize the launch data on an interactive map. It takes the latitude and longitude coordinates of each launch point and adds a circular marker around each launch point.
2. Then assign the dataframe launch_outcomes(failures, successes) to classes 0 and 1 with green and red markers on the map in a MarkerCluster().
3. Using Haversine's formula to calculate 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 the distance to the landmarks.

Build a Dashboard with Plotly Dash

https://github.com/KD96/DS-Capstone-IBM/blob/master/Dashboard%20with%20Plotly%20Dash.ipynb

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. URL Link to live website GitHub Link to source code Graphs

Predictive Analysis (Classification)

https://github.com/KD96/DS-Capstone-IBM/blob/master/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

BUILDING MODEL

1. Load our dataset into NumPy and Pandas
2. Transform Data • Split our data into training and test data sets
3. Check how many test samples we have
4. Decide which type of machine learning algorithms we want to use
5. Set our parameters and algorithms to GridSearchCV
6. Fit our datasets into the GridSearchCV objects and train our dataset.

.

Predictive Analysis (Classification)

EVALUATING MODEL

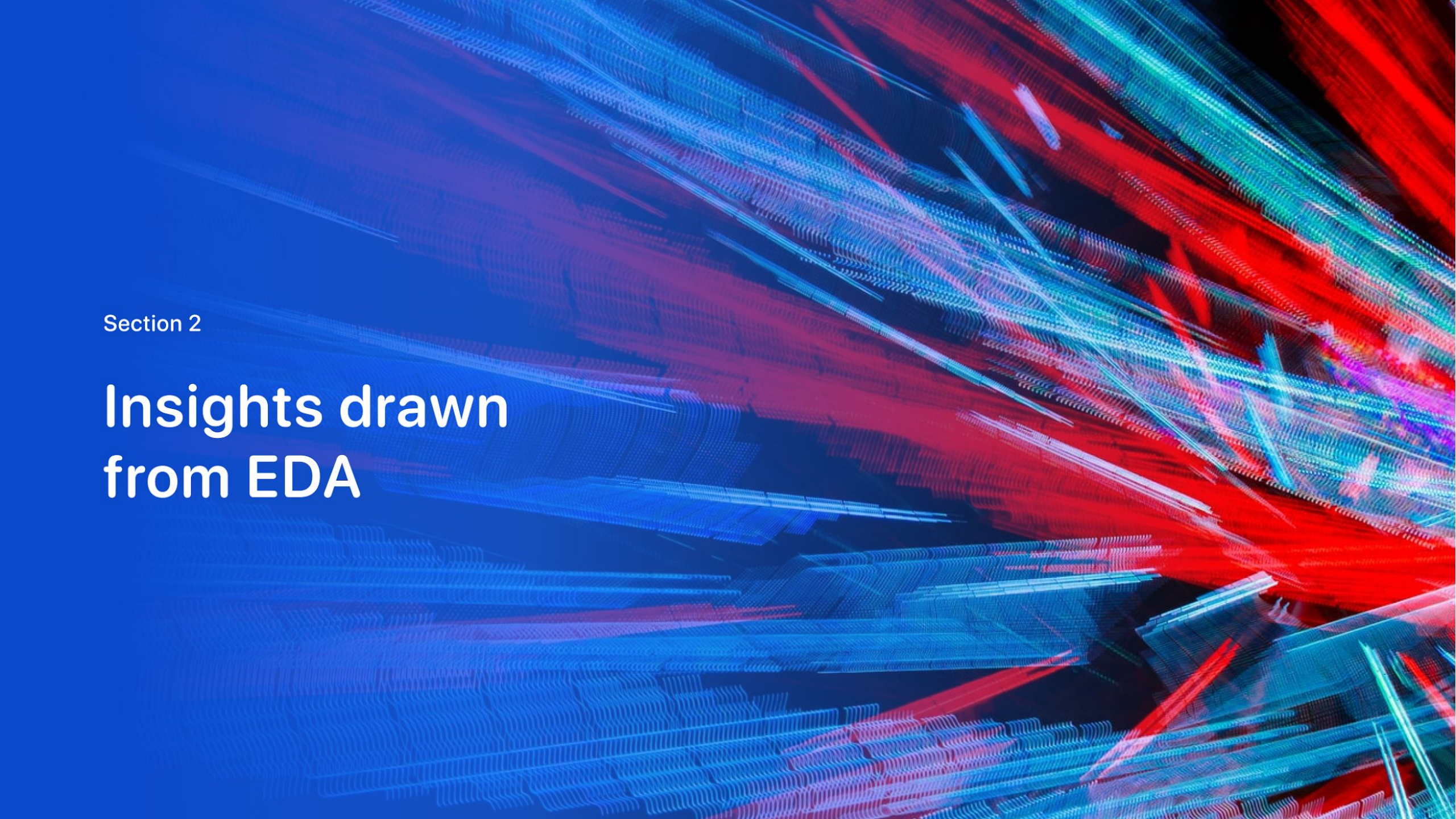
1. Check accuracy for each model
2. Get tuned hyperparameters for each type of algorithms
3. Plot Confusion Matrix IMPROVING MODEL
4. Feature Engineering
5. Algorithm Tuning

FINDING THE BEST PERFORMING CLASSIFICATION MODEL

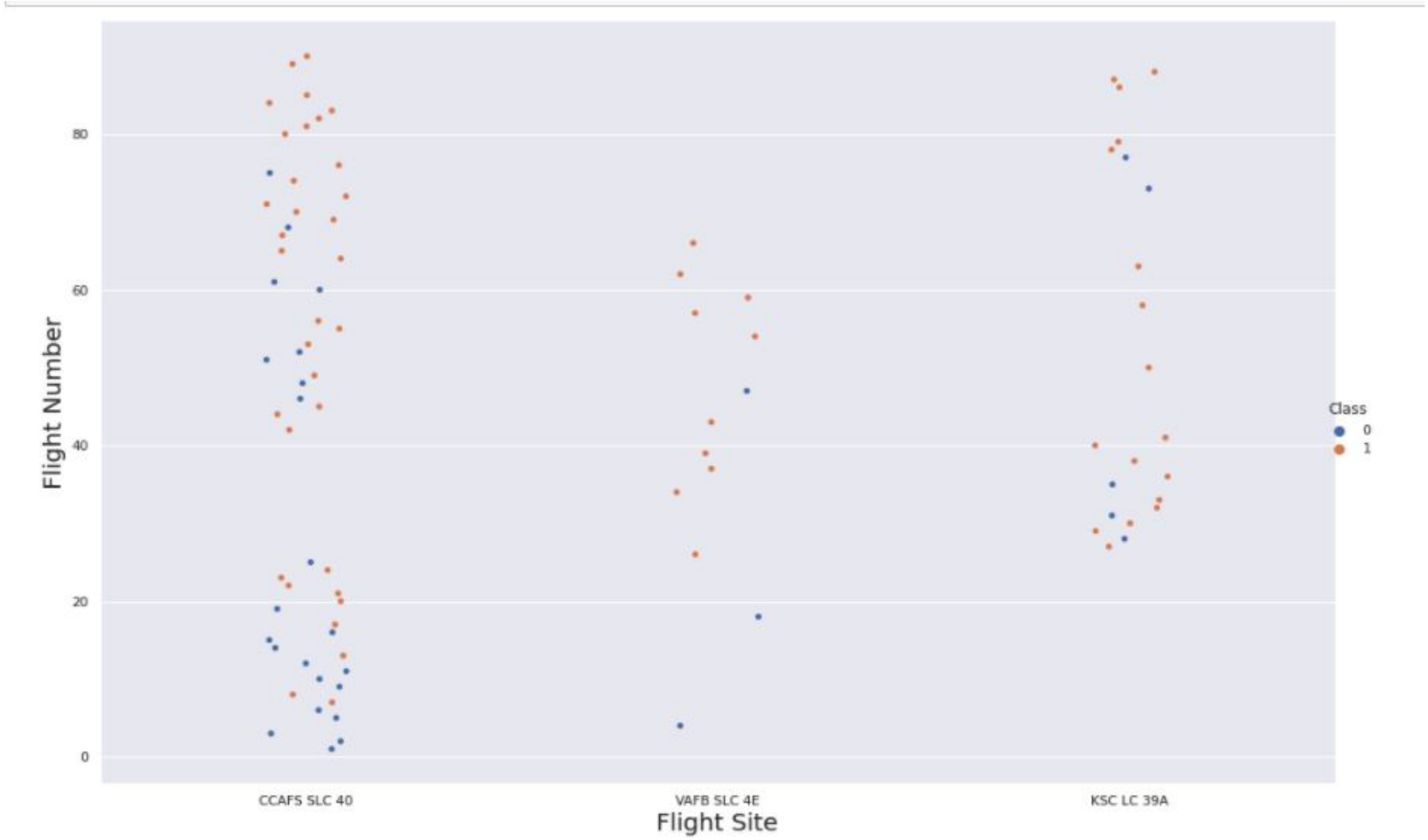
1. The model with the best accuracy score wins the best performing model
2. In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook.

Results

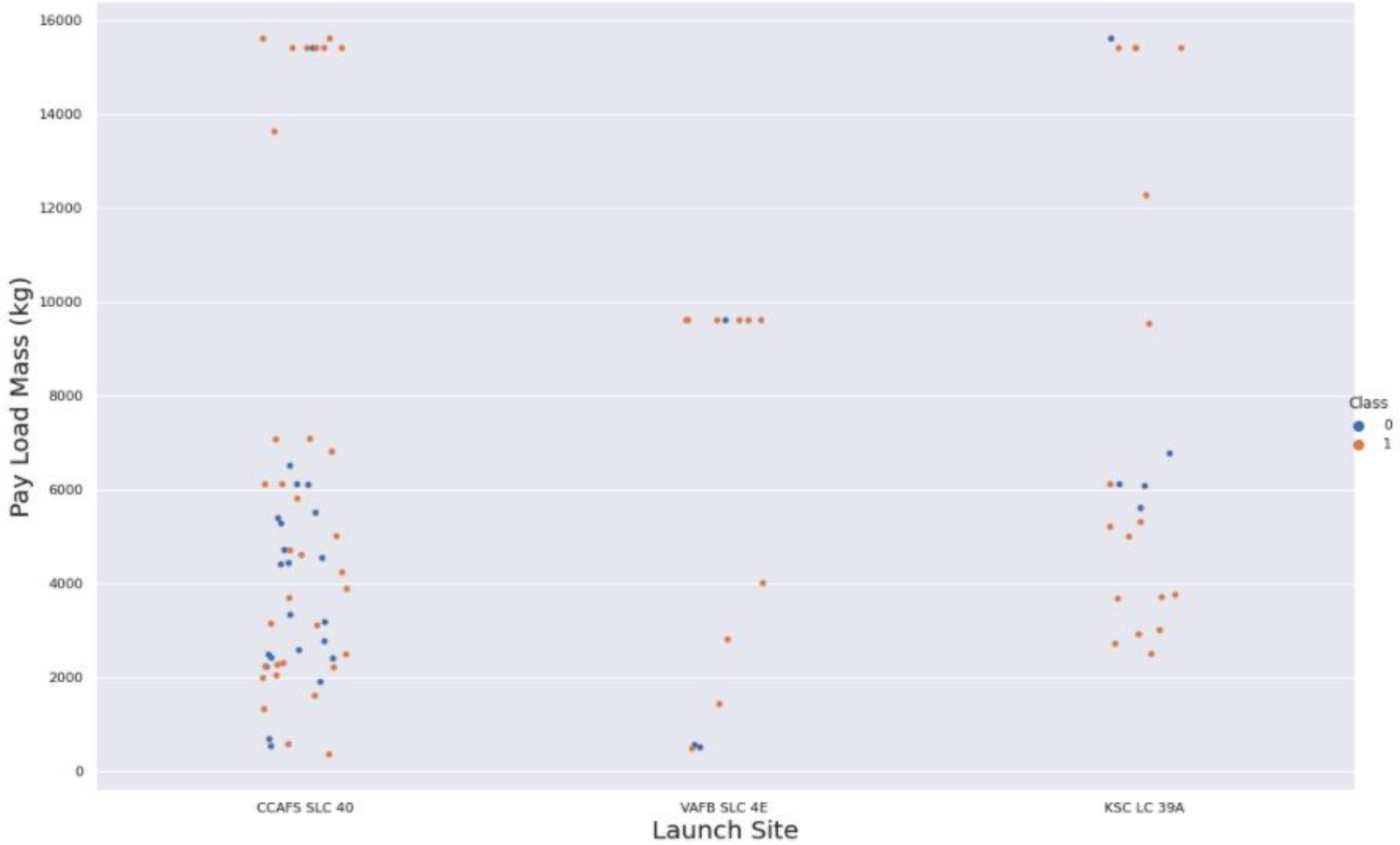
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

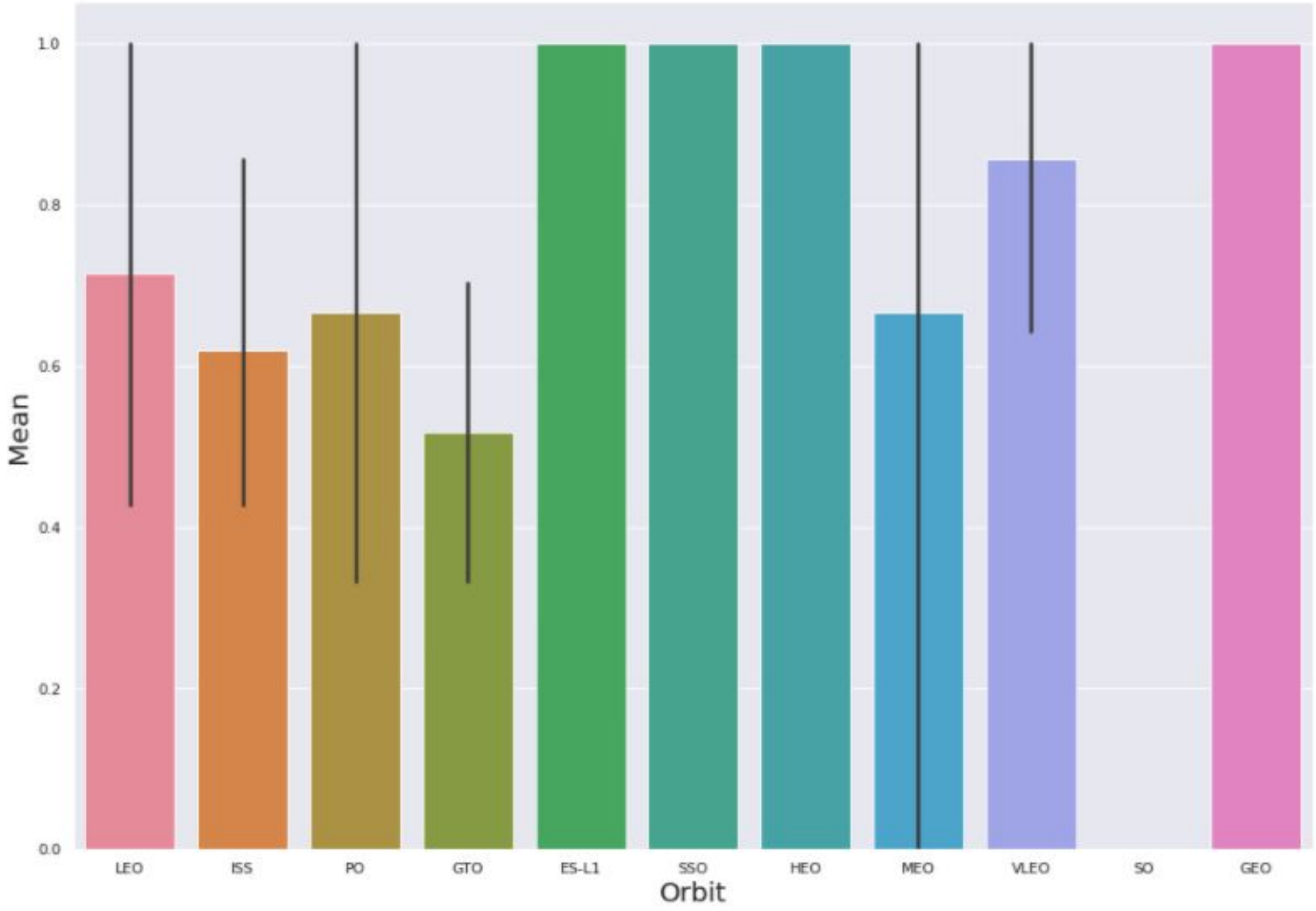
Flight Number vs. Launch Site



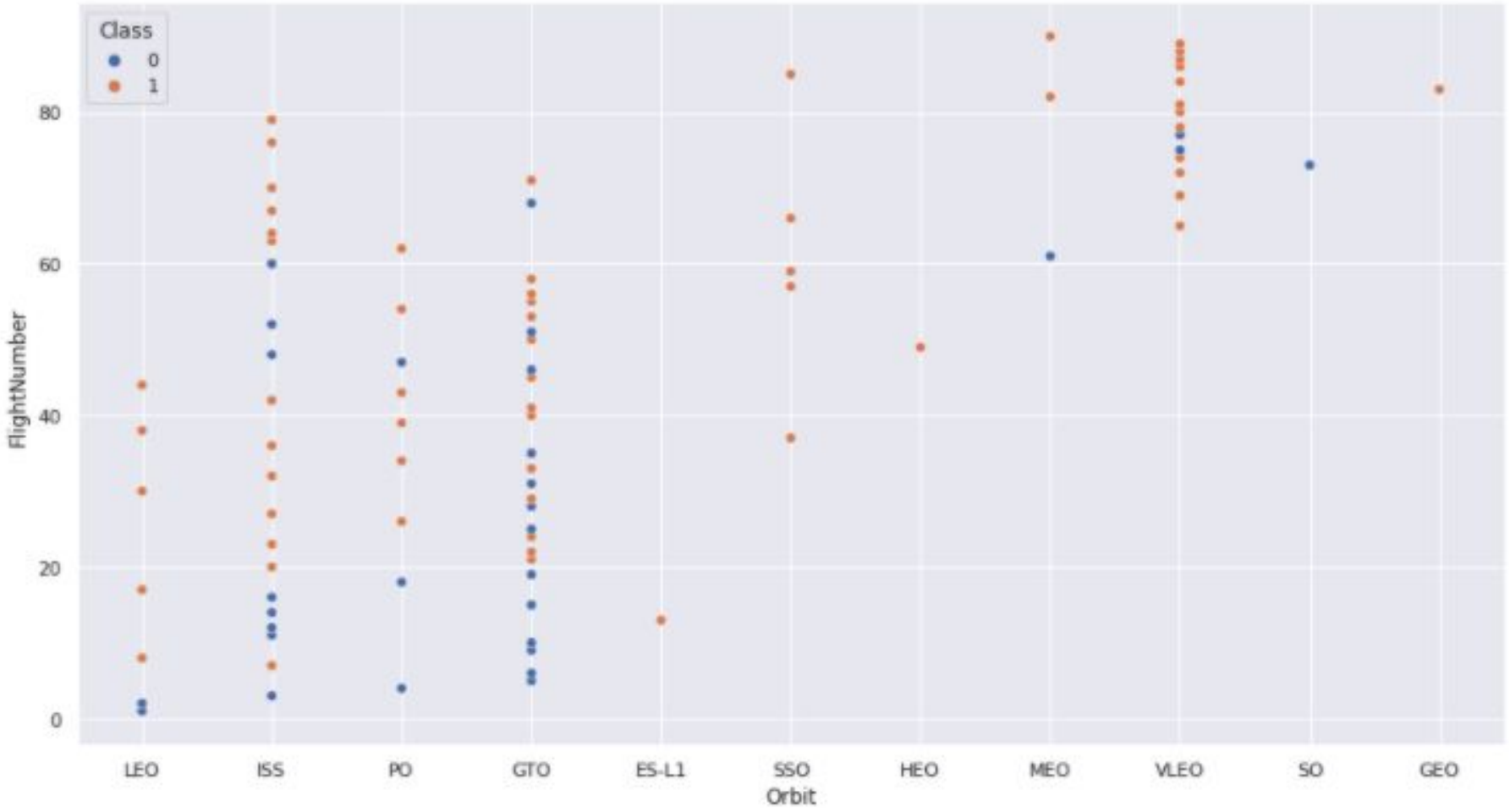
Payload vs. Launch Site



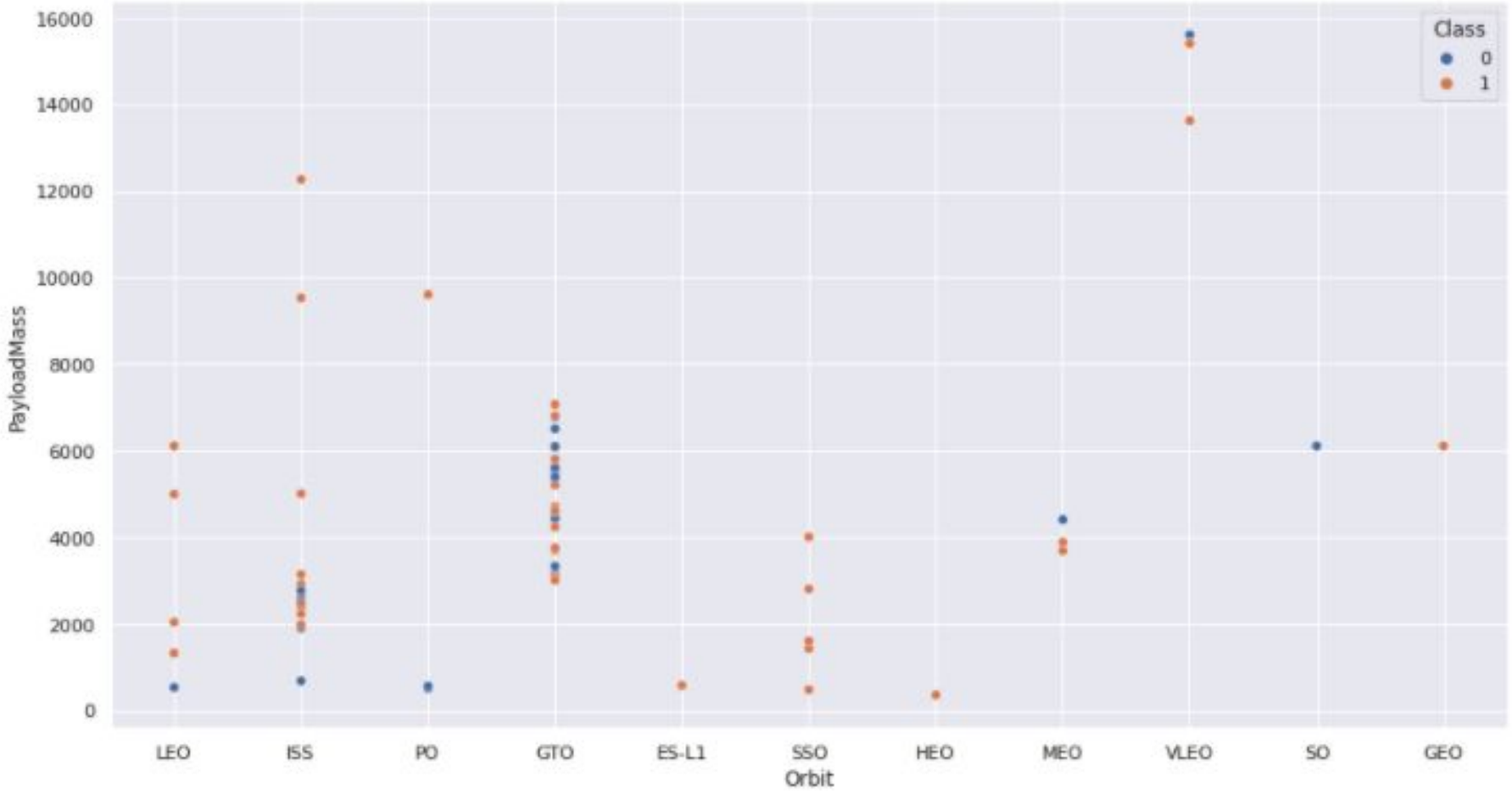
Success Rate vs. Orbit Type



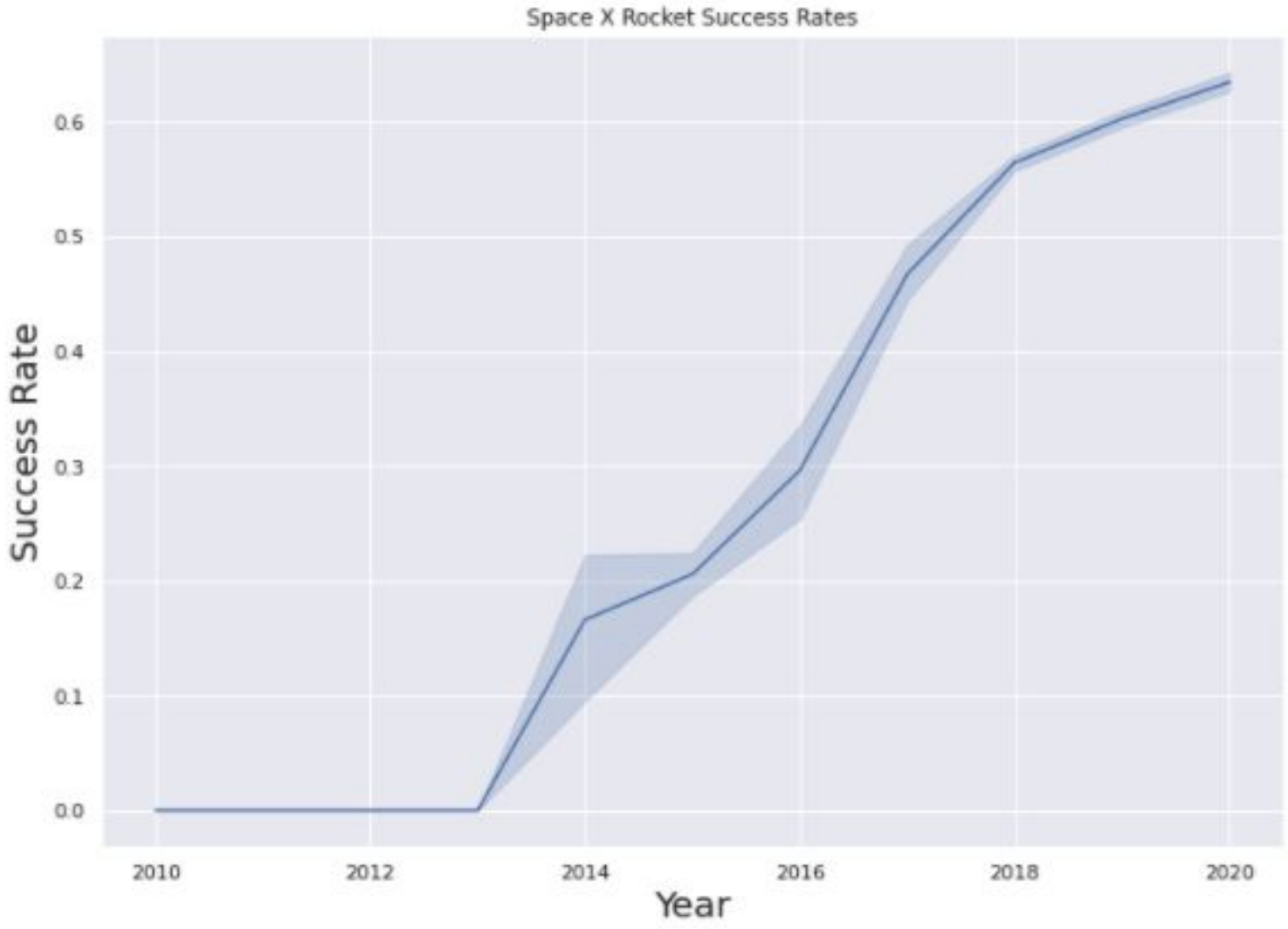
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

Task 1

Display the names of the unique launch sites in the space mission

```
%sql select Unique(LAUNCH_SITE) from ZWL17924.SPACEXDATASET;
```

```
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

```
%sql SELECT LAUNCH_SITE from ZWL17924.SPACEXDATASET where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:3119
8/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from ZWL17924.SPACEXDATASET;
```

```
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:3119
8/bludb
Done.
```

payloadmass
619967

Average Payload Mass by F9 v1.1

Task 4

Display average payload mass carried by booster version F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from ZWL17924.SPACEXDATASET;
```

```
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:3119
8/bludb
Done.
```

payloadmass
6138

First Successful Ground Landing Date

Task 5

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql select min(DATE) from ZWL17924.SPACEXDATASET;
```

```
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:3119
8/bludb
Done.
```

1
2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql select BOOSTER_VERSION from ZWL17924.SPACEXDATASET where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS
__KG_ BETWEEN 4000 and 6000;
```

```
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:3119
8/bludb
Done.
```

booster_version
F9 FT B1022
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [17]:
```
%%sql
select count(*), mission_outcome from ZWL17924.SPACEXDATASET group by mission_outcome
```

```
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

Out[17]:

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

Boosters Carried Maximum Payload

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
In [33]: %sql select BOOSTER_VERSION as boosterversion from ZWL17924.SPACEXDATASET where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from ZWL17924.SPACEXDATASET);
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

Out[33]:

boosterversion
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

Task 9

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
In [35]: %sql SELECT MONTH(DATE),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM ZWL17924.SPACEXDATASET where EXTRACT(YEAR FROM DATE)='2015';
```

```
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

Out[35]:

1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
2	Success	F9 v1.1 B1013	CCAFS LC-40
3	Success	F9 v1.1 B1014	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40
4	Success	F9 v1.1 B1016	CCAFS LC-40
6	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

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

```
In [19]: %sql SELECT LANDING__OUTCOME FROM ZWL17924.SPACEXDATASET WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;
 * ibm_db_sa://zwl17924:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqblod8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

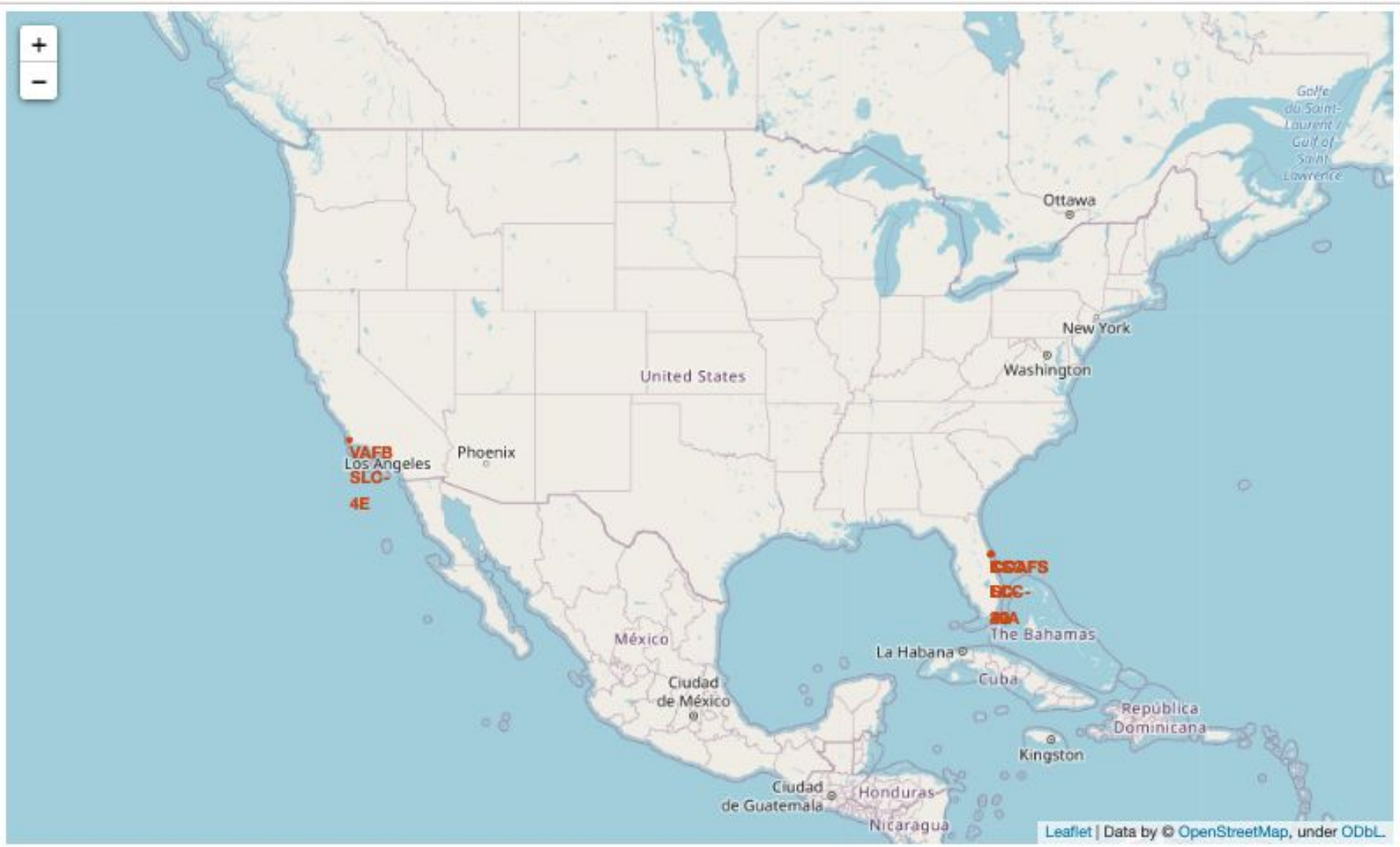
Out[19]:

landing__outcome
No attempt
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)
Precluded (drone ship)
No attempt
Failure (drone ship)
No attempt
Controlled (ocean)
Failure (drone ship)
Uncontrolled (ocean)
No attempt
No attempt
Controlled (ocean)
Controlled (ocean)
No attempt
No attempt
Uncontrolled (ocean)
No attempt
No attempt
No attempt
Failure (parachute)
Failure (parachute)

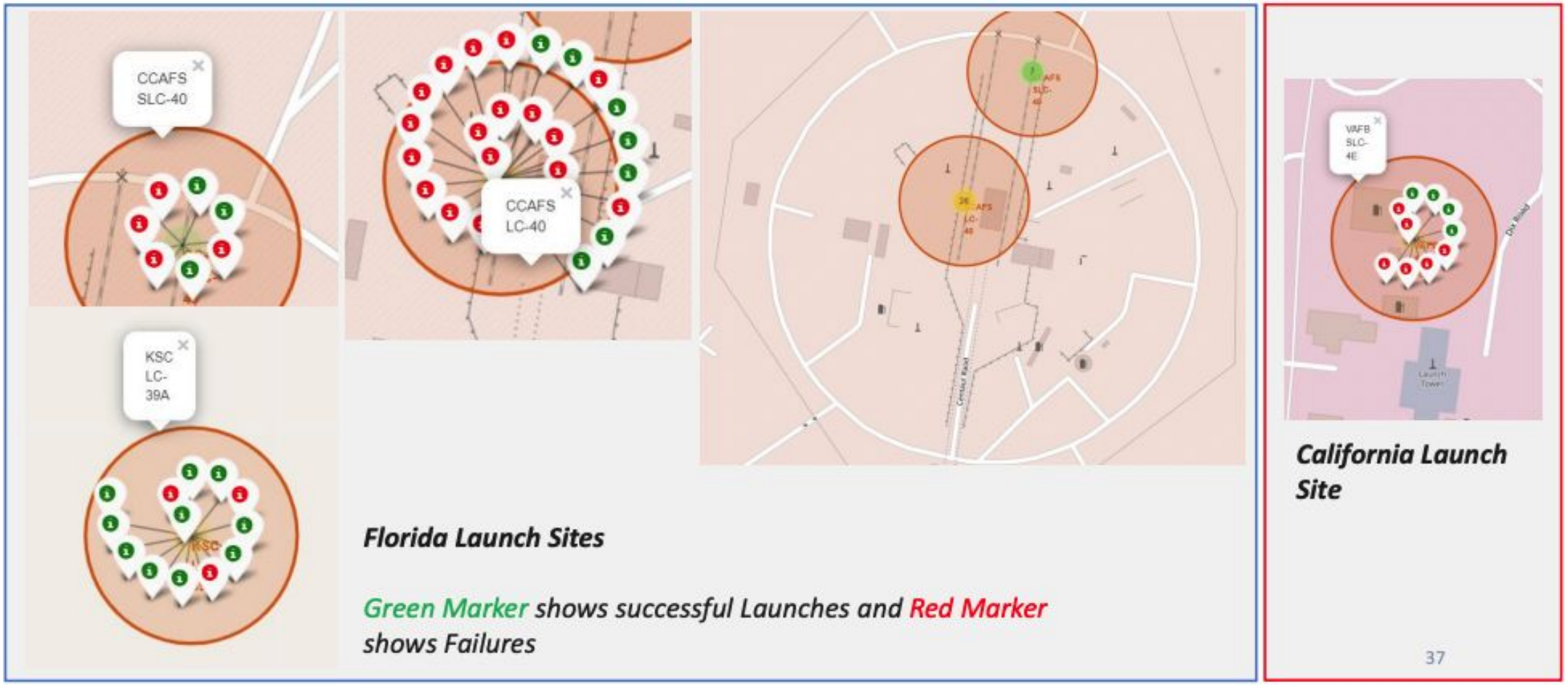
Section 4

Launch Sites Proximities Analysis

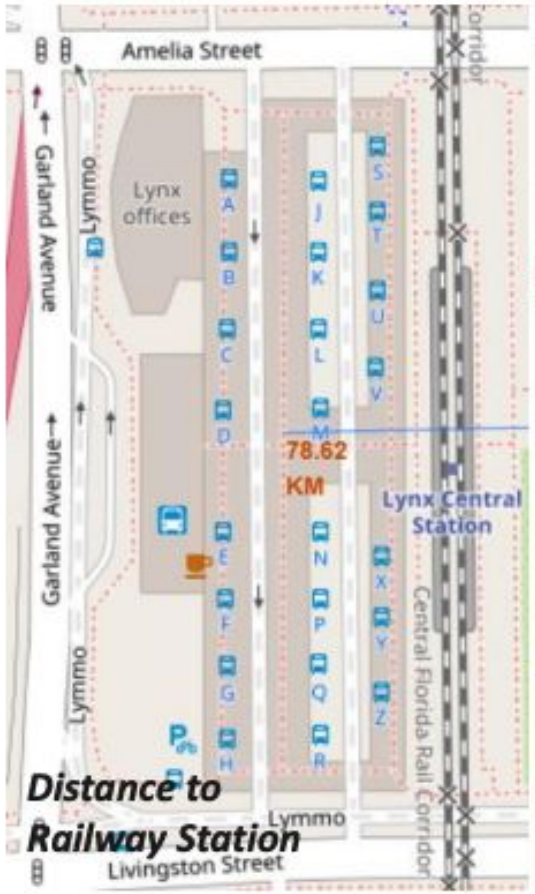
<Folium Map Screenshot 1>

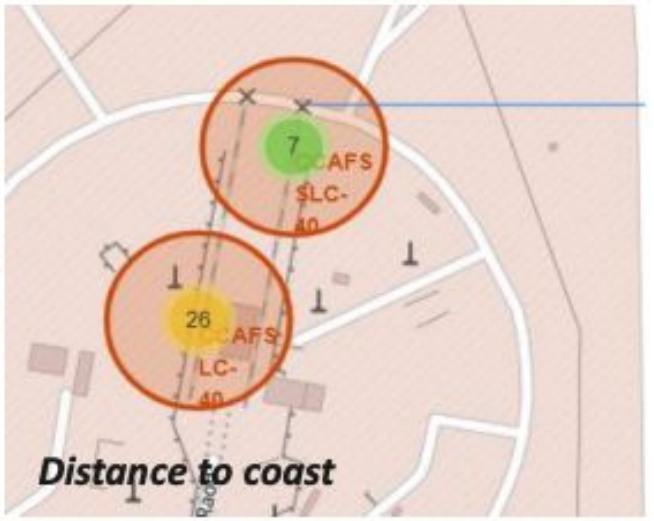


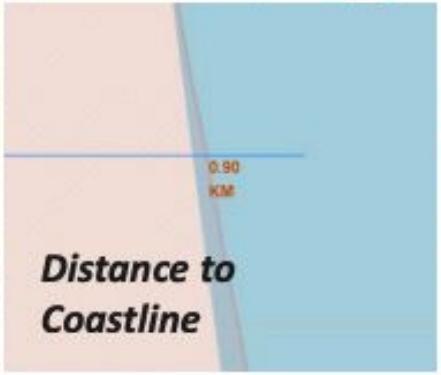
<Folium Map Screenshot 2>

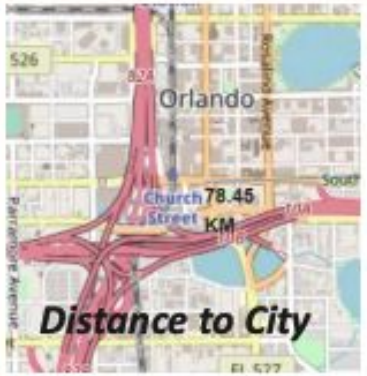


<Folium Map Screenshot 3>

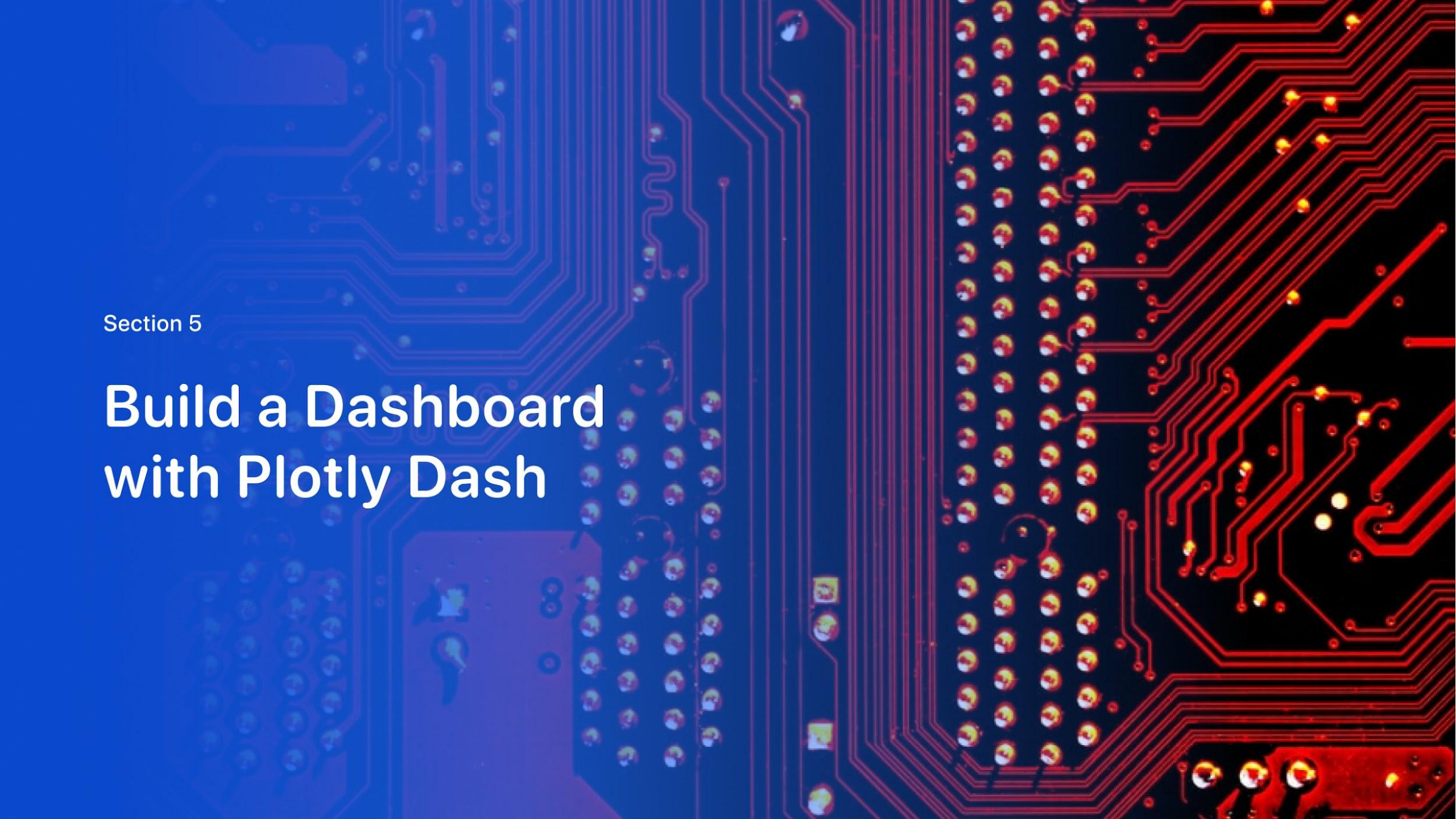




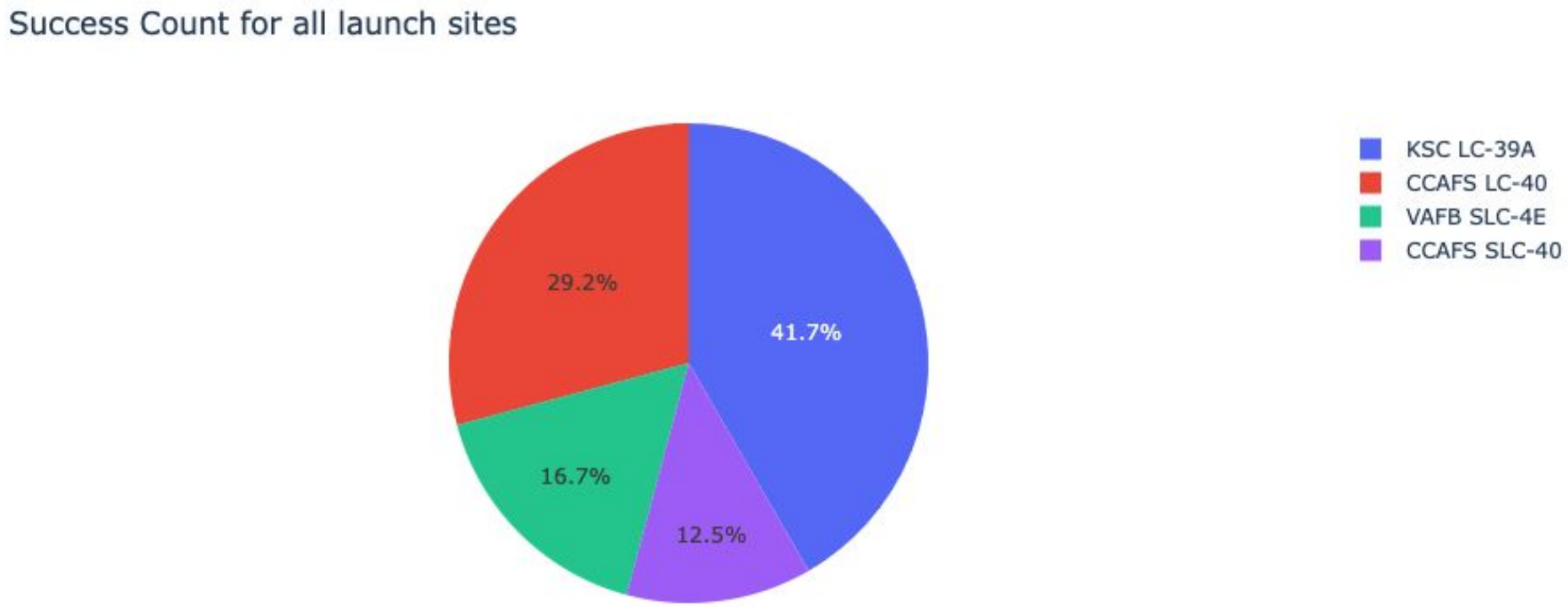




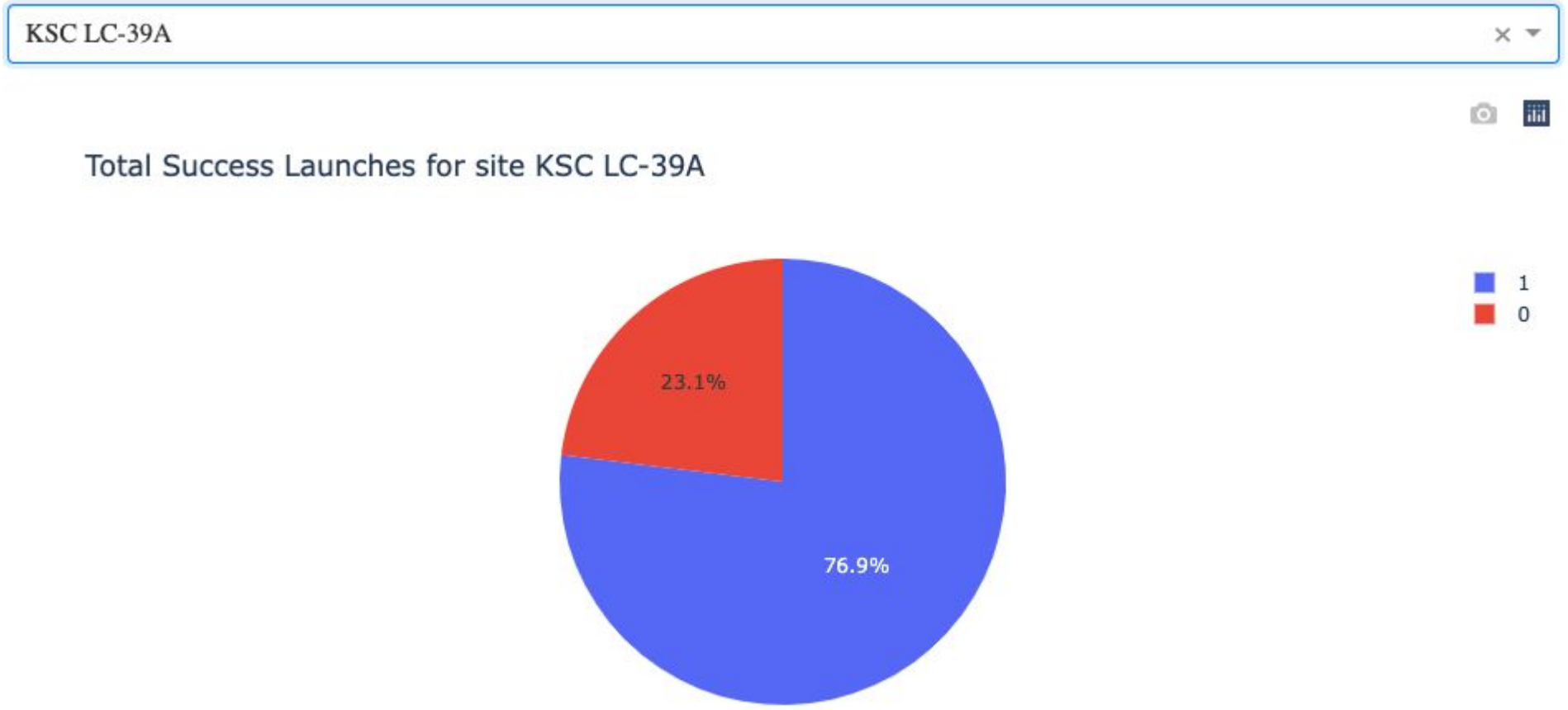
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes


Section 5
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site

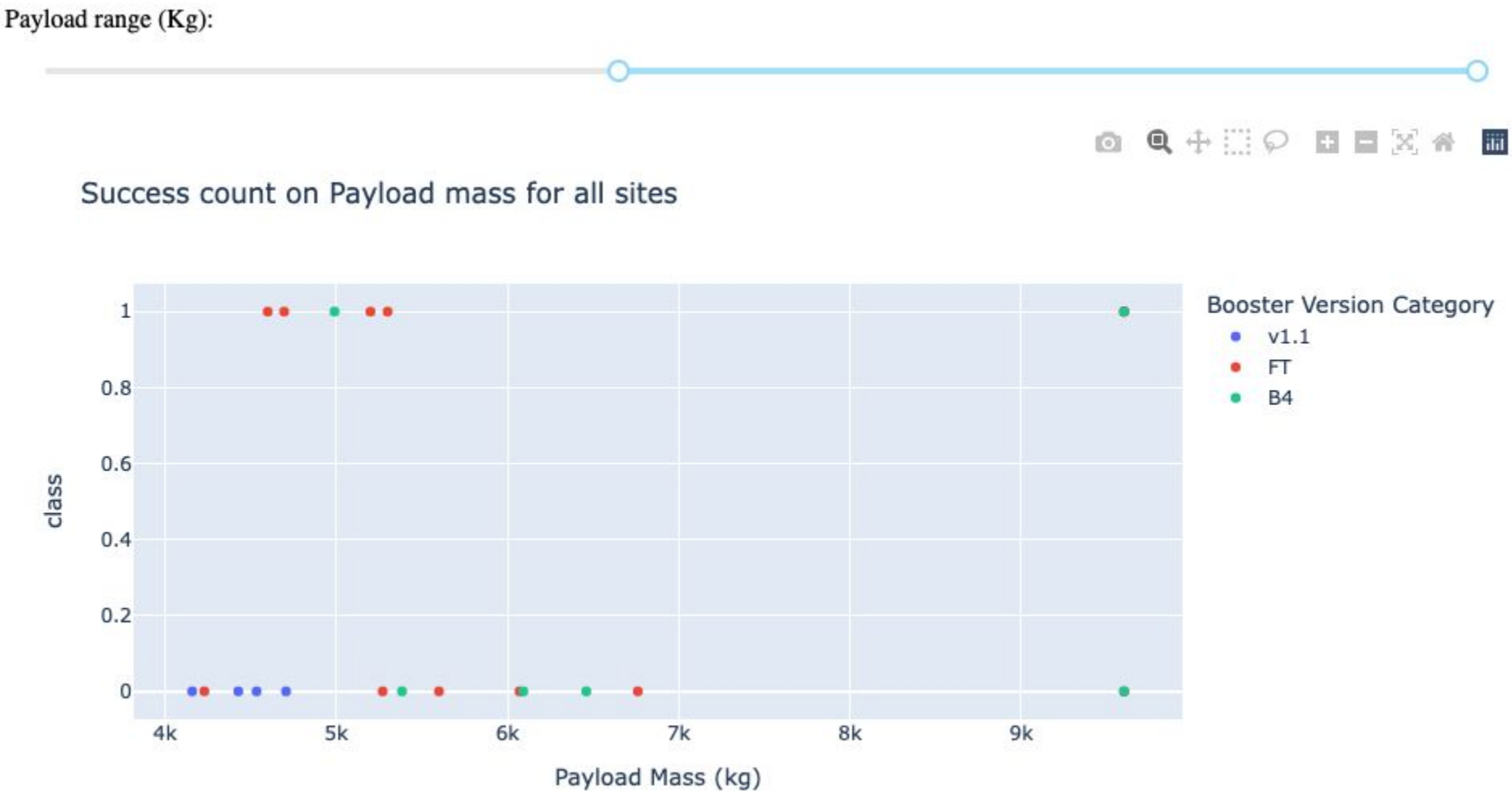


Pie chart for the launch site with highest launch success ratio



Payload vs. Launch Outcome scatter plot for all sites (payload < 4000kg)

Payload vs. Launch Outcome scatter plot for all sites (payload > 4000kg)



Section 6

Predictive Analysis (Classification)

Classification Accuracy

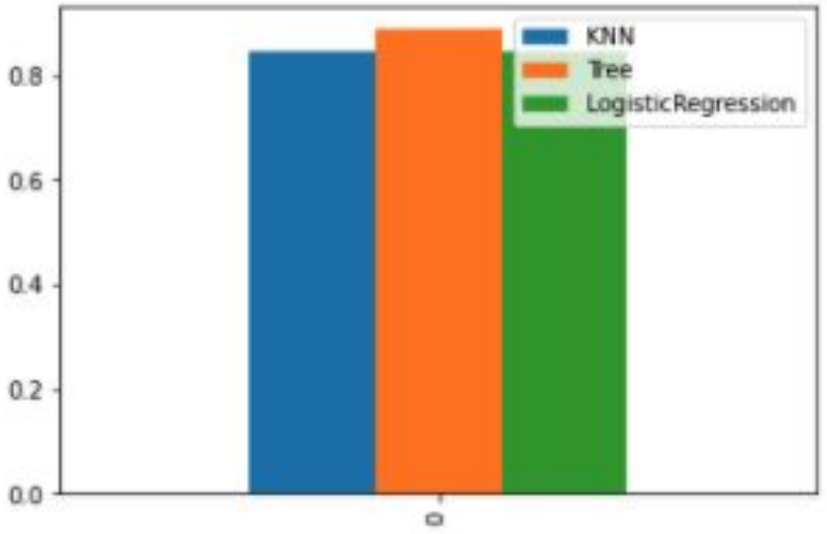
```
In [50]: pd.DataFrame([algorithms])
```

Out[50]:

	KNN	Tree	LogisticRegression
0	0.848214	0.8875	0.846429

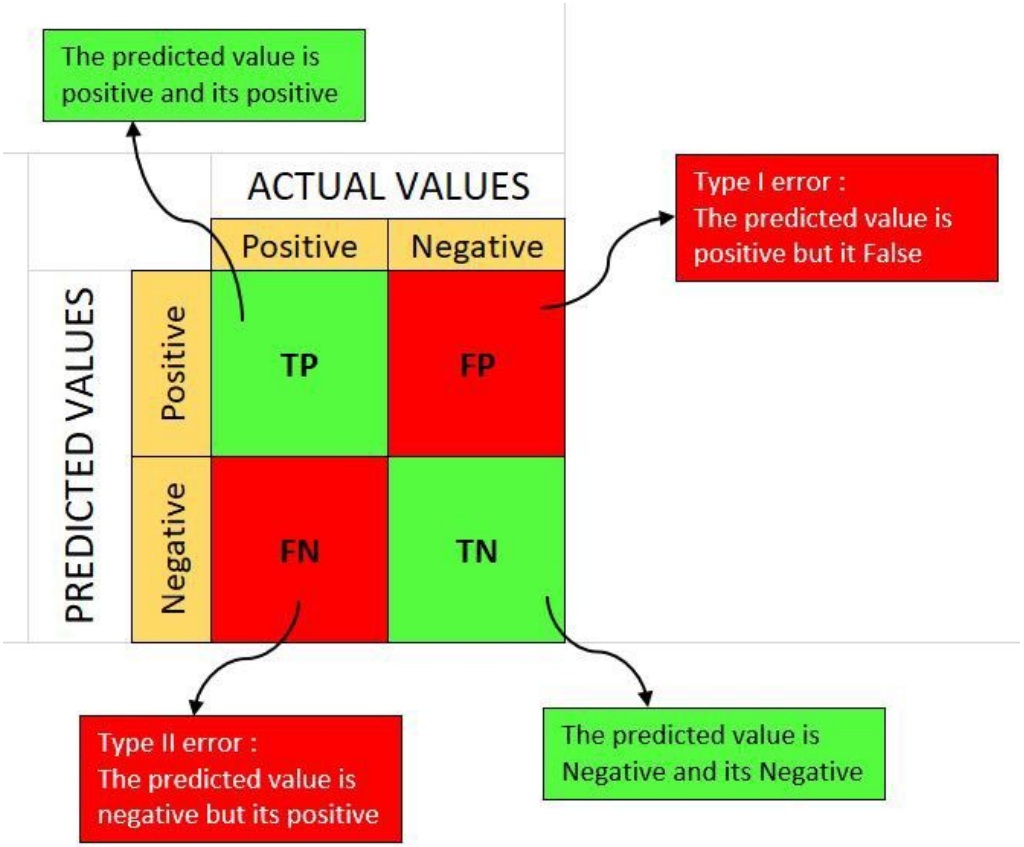
```
In [49]: pd.DataFrame([algorithms]).plot.bar()
```

Out[49]: <AxesSubplot:>

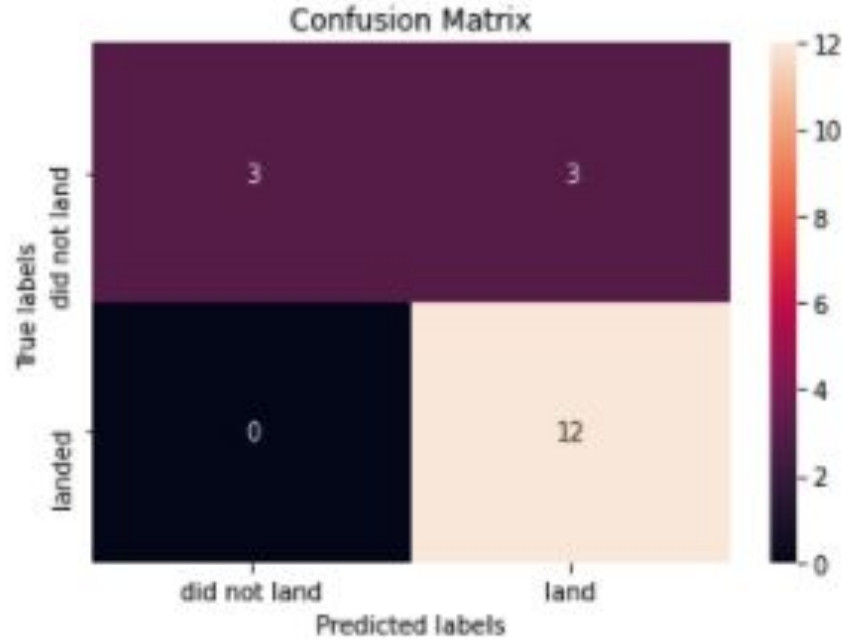


Confusion Matrix for the Tree

- The confusion matrix is a useful tool used for classification tasks in machine learning with the primary objective of visualizing the performance of a machine learning model




```
yhat = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- The Tree Classifier Algorithm is the best algorithm for machine learning for this data set with an accuracy with the training set of 0.8875
- Low-weighted payloads perform better than heavier payloads,
- SpaceX launch success rates are directly proportional to the time as the pass years that SpaceX launches are getting perfect
- FT B1022 and FT B1031.2 boosters have been successful, with payloads greater than 4000 and less than 6000.
- GEO, HEO, SSO, ES-L1 orbit has the best success rate

Thank you!