Winning Space Race with Data Science

Sankha Mitra

4th August 2023

Table of Contents

Executive Summary

- SpaceY is a new commercial rocket launch provider who wants to bid/compete against SpaceX.
- SpaceX advertises launch services starting at $62 million (for Falcon 9 launches) for missions that allow some fuel to be reserved for landing the 1st stage rocket booster, so that it can be reused.
- SpaceX states that a 1st stage Falcon 9 booster to cost upwards of $15 million to build.
- Given mission parameters such as payload mass and desired orbit, the models produced in this report were able to predict the first stage rocket booster landing successfully with an accuracy level of 86% using Decision Tree model among other models.
- As a result, Space Y will be able to make more informed bids against SpaceX by using 1st stage landing predictions as a proxy for the cost of a launch.

Introduction (Background)

- This report has been prepared as part of the Applied Data Science Capstone course.*
- In this capstone, the role of a data scientist is assumed who is working for a new rocket manufacturing company called Space Y.
- With the help of data science findings and models in this report, SpaceY will be able to make more informed bids against SpaceX for a rocket launch.

* This is the 10th course in the IBM Data Science Professional Certification.

Introduction (Business Case)

- SpaceX advertises Falcon 9 rocket launches with a cost of 62 million dollars when the first stage of their rockets can be reused.

- The first stage is estimated to cost upwards of 15 million to build without including R&D cost recoupment or profit margin.

- Sometimes SpaceX will sacrifice the first stage due to mission parameters such as payload, orbit, and customer.

- Therefore, this report aims to accurately predict the likelihood of the first stage rocket landing successfully as a proxy for the cost of a launch.

Section 1

Methodology

Methodology

For this report, the following data science methodology has been used :

1. Data collection
2. Data wrangling
3. Exploratory data analysis
4. Data visualization
5. Model development
6. Reporting and sharing of results to the stakeholders

Data Collection

- Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. As mentioned, the dataset was collected by REST API and Web Scrapping from Wikipedia
- For REST API, it is initiated by using the get request. Then, the response content is decoded as Json and converted into a pandas data frame using json_normalize(). The data is the cleaned, checked for missing values and filled with whatever values needed.
- For web scrapping, BeautifulSoup is used to extract the launch records as HTML table, the table is the parsed and converted to a pandas data frame for further analysis

Data Collection – SpaceX API

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

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

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

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and r
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Step1

Get request for rocket launch data using API

Step2

Use json_normalize method to convert json result to dataframe

Step3

Perform data cleaning and filling the missing value

From: https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/jupyter-labs-spacex-data-collection-api%20(2).ipynb

Data Collection - Scraping

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

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

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number) #TODO-1
            #print(flight_number)
            datatimelist=date_time(row[0])

------------------
            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date) #TODO-2
            #print(date)
```

Step1

Request the Falcon9 Launch Wiki page from url

Step2

Create a BeautifulSoup from the HTML response

Step3

Extract all column/variable names from the HTML header

From: https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/jupyter-labs-webscraping%20(2).ipynb

Data Wrangling

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
```

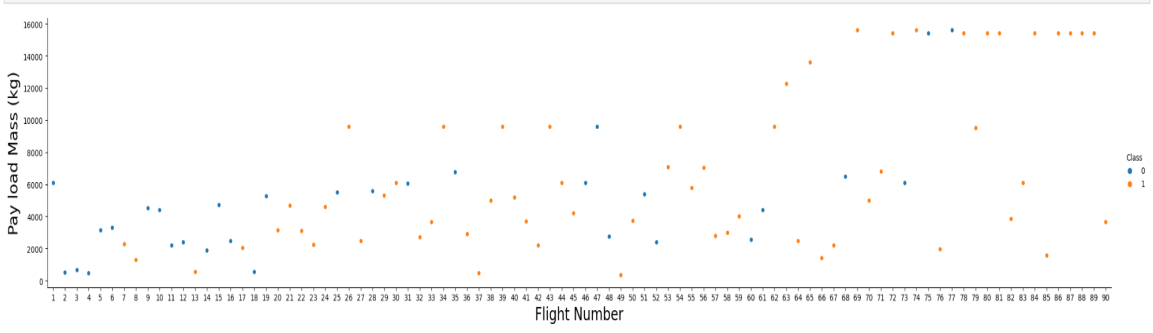
Landing Outcomes	
Sample Size = 90	
True ASDS	41
None None	19
True RTLS	14
False ASDS	6
True Ocean	5
False Ocean	2
None ASDS	2
False RTLS	1

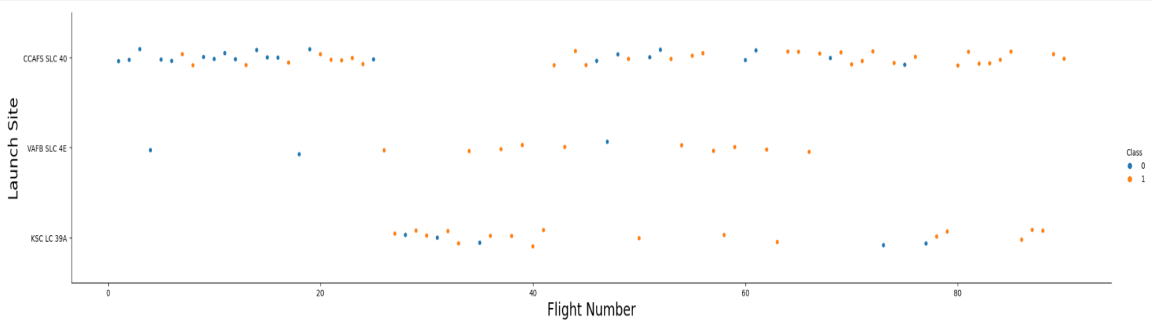
Class 0
Class 1

- Data exploration performed to determine the label for training supervised models
  - Calculated the number of launches on each site
  - Calculated the number and occurrence of each orbit
  - Calculated the number and occurrence of mission outcome per orbit type
- Created a landing outcome training label from 'Outcome' column
  - Training label: 'Class'
  - Class = 0; first stage booster did not land successfully
    - None None; not attempted
    - None ASDS; unable to be attempted due to launch failure
    - False ASDS; drone ship landing failed
    - False Ocean; ocean landing failed
    - False RTLS; ground pad landing failed
  - Class = 1; first stage booster landed successfully
    - True ASDS; drone ship landing succeeded
    - True RTLS; ground pad landing succeeded
    - True Ocean; ocean landing succeeded

From: https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite%20(1).ipynb

EDA with Data Visualization

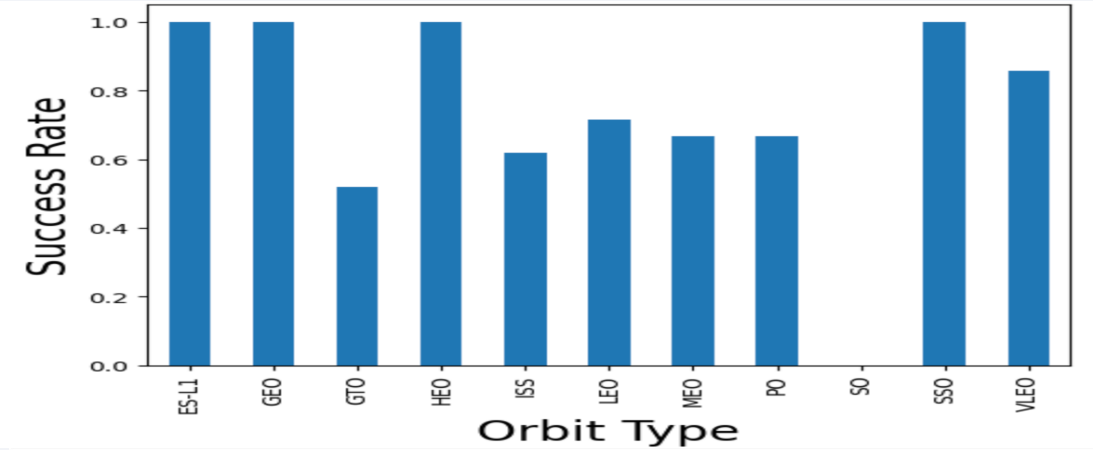


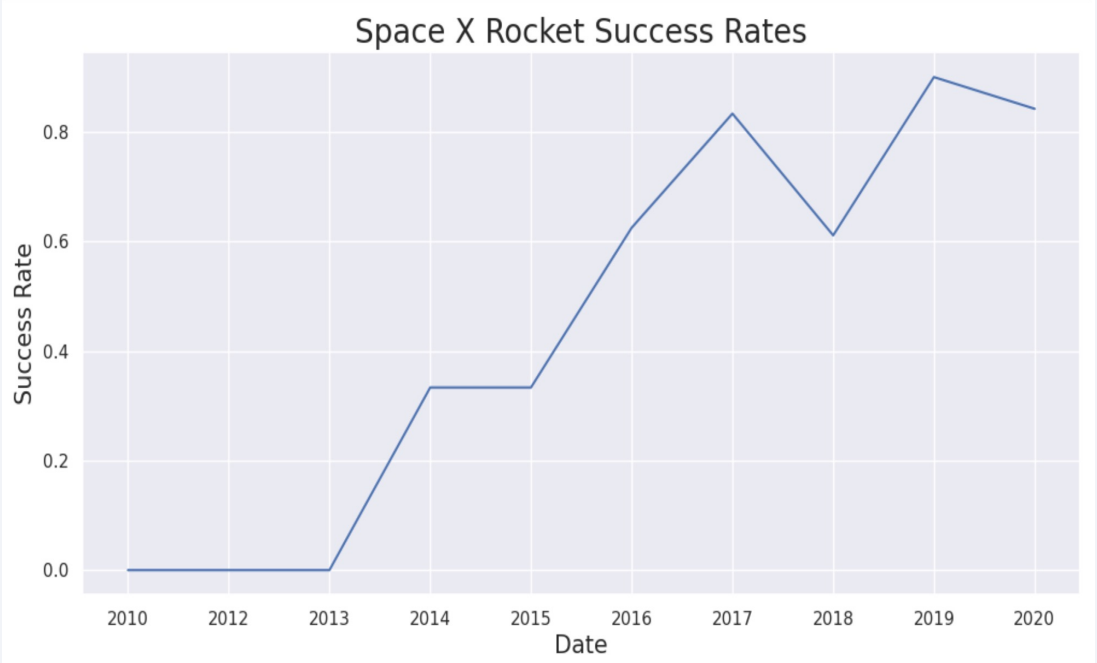


- First, scatter graph has been used to find the relationship between the attributes :
- Payload and Flight Number.
- Flight Number and Launch Site.
- Payload and Launch Site.
- Flight Number and Orbit Type.
- Payload and Orbit Type.
- Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs. It is very easy to see which factors affect most to the success of the landing outcomes.

From: https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

EDA with Data Visualization





- Once the relationships are determined using scatter plot, further visualization tools such as bar graph and line plot graphs are used for further analysis.
- Bar graph is one of the easiest way to interpret the relationship between the attributes. In this case, the bar graph has been used to determine which orbits have the highest probability of success.
- Line graph has been used to show trends or pattern of the attribute over time, which in this case, has been used for observing the launch success yearly trend.
- Feature Engineering has been used in success prediction in the future module by creating the dummy variables to categorical columns.

From:
https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

EDA with SQL

- Using SQL, queries were run to get better understanding of the dataset, Ex:
- - Displaying the names of the unique launch sites.
- - Displaying 5 records where launch sites begin with the string ‘CCA’.
- - Displaying the total payload mass carried by boosters launched by NASA (CRS).
- - Displaying the average payload mass carried by booster version F9 v1.1.
- - Listing the date when the first successful landing outcome in ground pad was achieved.
- - Listing the names of the boosters which have success in drone ship 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 failed landing_outcomes in drone ship, their booster versions, and launch sites
- names for in year 2015.
- - Rank the count of landing outcomes or success between the date 2010-06-04 and
- 2017-03-20, in descending order.

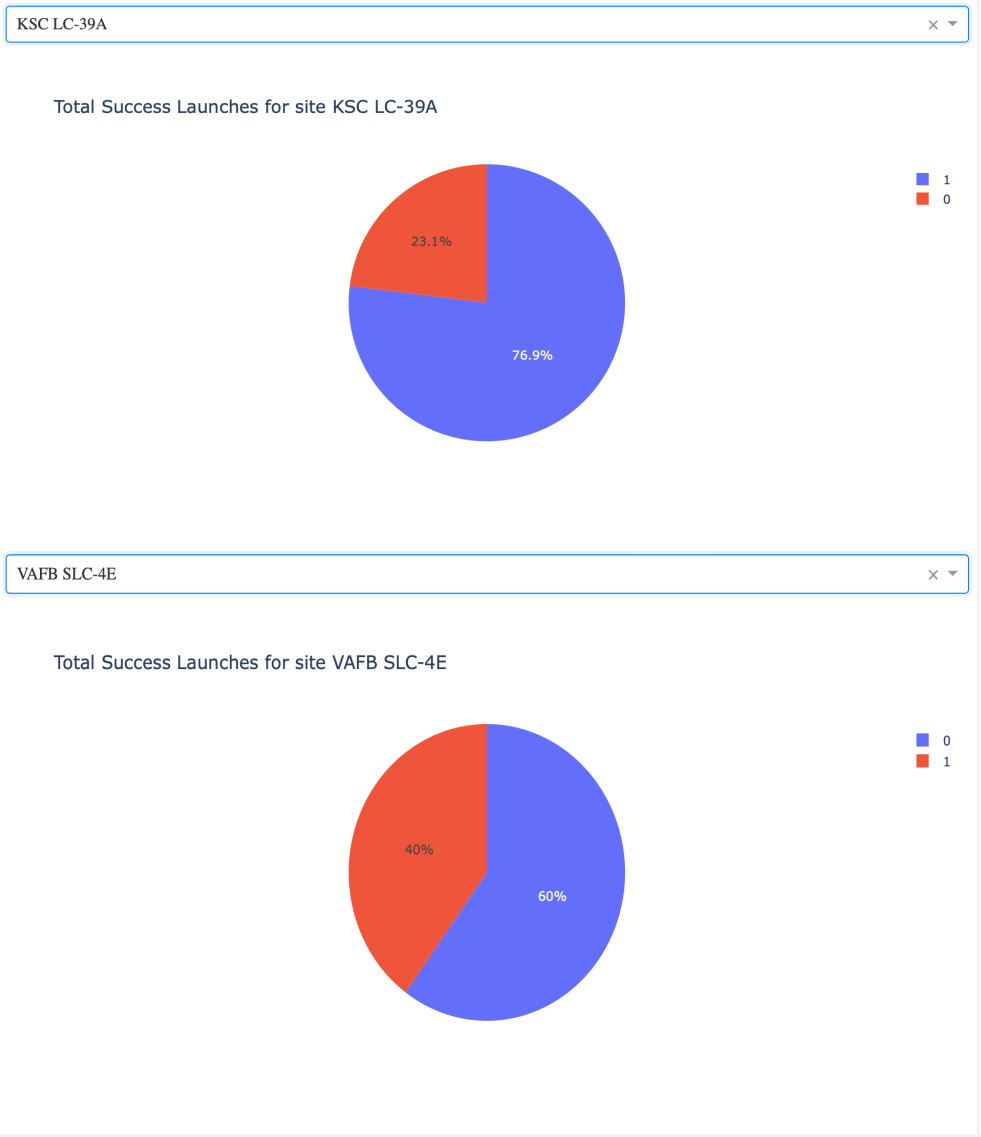
From: https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

Build an Interactive Map with Folium

- To visualize the launch data into an interactive map, the latitude and longitude coordinates were considered at each launch site and a circle marker was added around each launch site with a label of the name of the launch site.
- The dataframe launch_outcomes(failure,success) were then labelled to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().
- Haversine's formula was then used to calculated the distance of the launch sites to various landmark to find answer to the questions of:
- How close the launch sites with railways, highways and coastlines?
- How close the launch sites with nearby cities?

From: https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash



- Python interactive dashboarding library called Plotly Dash was then used to enable stakeholders to explore and manipulate data in an interactive and real-time way

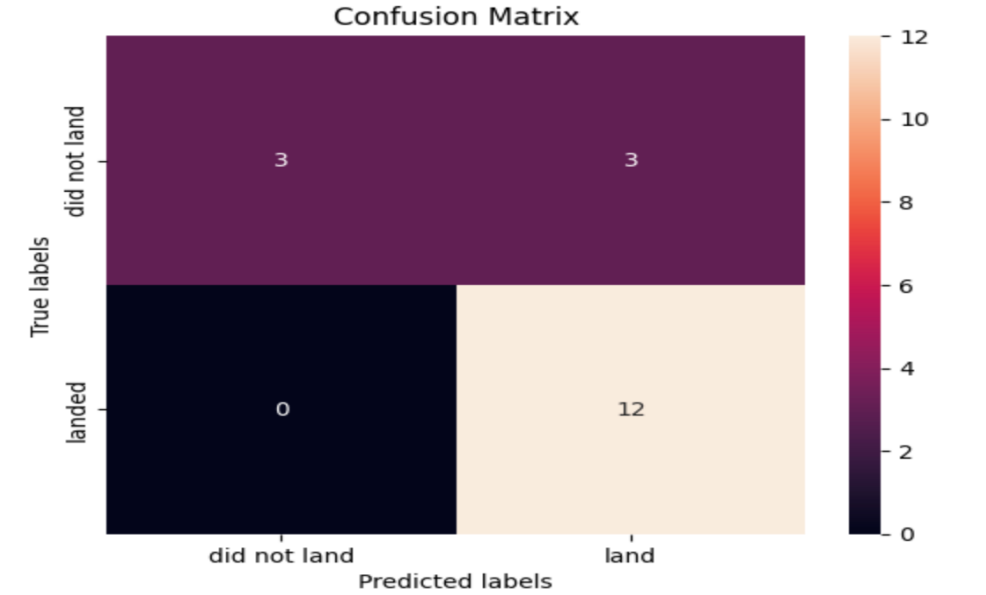
Dashboard observations:

- VAFB SLC-4E had the heaviest successful
- booster landing success
- KSC LC-39A had the highest booster landing success rate
- Payloads < 5,300 kg had the highest booster landing success rate
- Payloads > 5,300 kg had the lowest booster landing success rate

From: https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/Hands-on%20Lab_%20Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash.ipynb

Predictive Analysis (Classification)

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



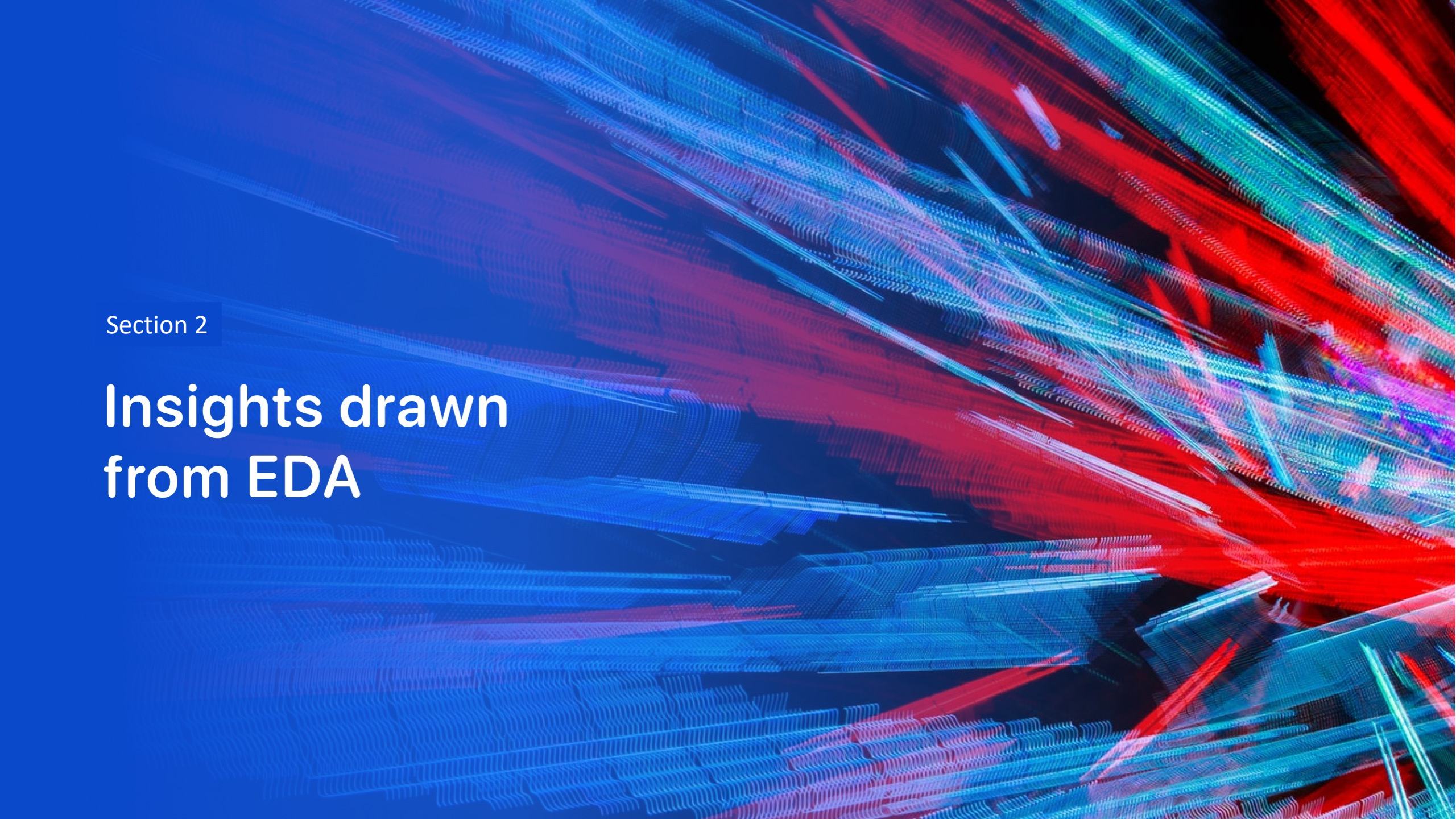
Confusion matrix of logistic regression model, showing 15 correct predictions and 3 false positives

- Libraries imported and defined function to create confusion matrix
  - Pandas
  - Numpy
  - Matplotlib
  - Seaborn
  - Sklearn
- Loaded the dataframe created during data collection
- Created a column for our training label 'Class' created during data wrangling
- Standardized the data
- Split the data into training data and test data
- Fit the training data to various model types
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree Classifier
  - K Nearest Neighbors Classifier
- Used a cross-validated grid-search over a variety of hyperparameters to select the best ones for each model (Enabled by Scikit-learn library function GridSearchCV)
- Evaluated accuracy of each model using test data to select the best model

From: https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

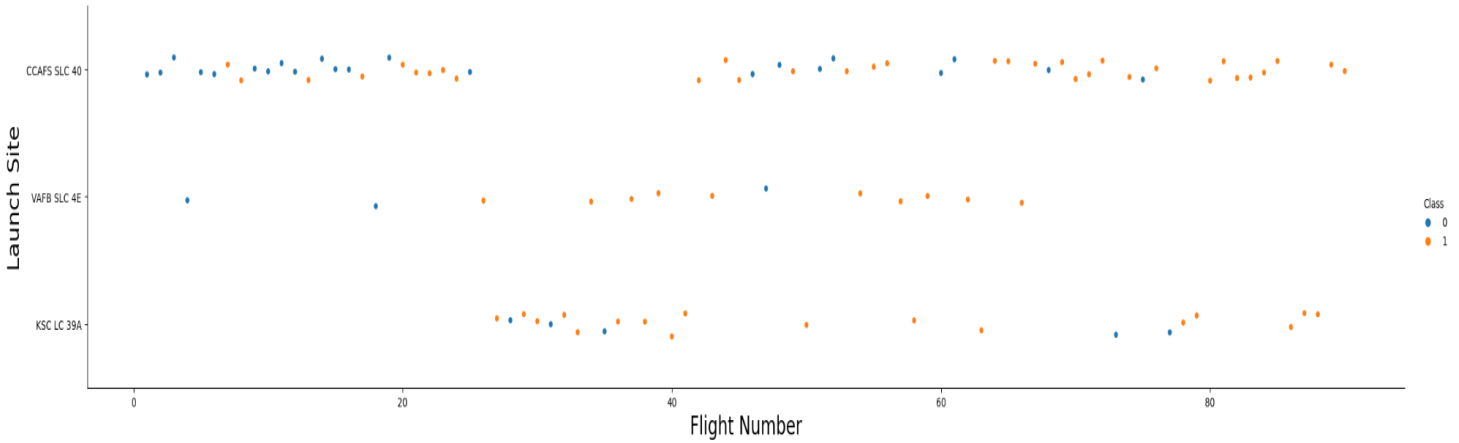
Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

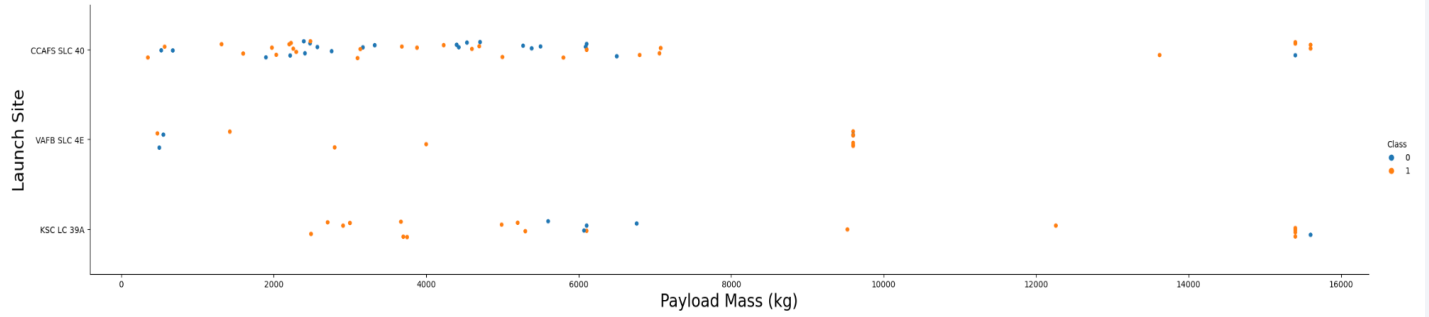
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



An analysis of Flight Number vs Launch Site through a scatter plot diagram which shows that CCAFS SLC 40 appears to have been where most of the early 1st stage landing failures took place

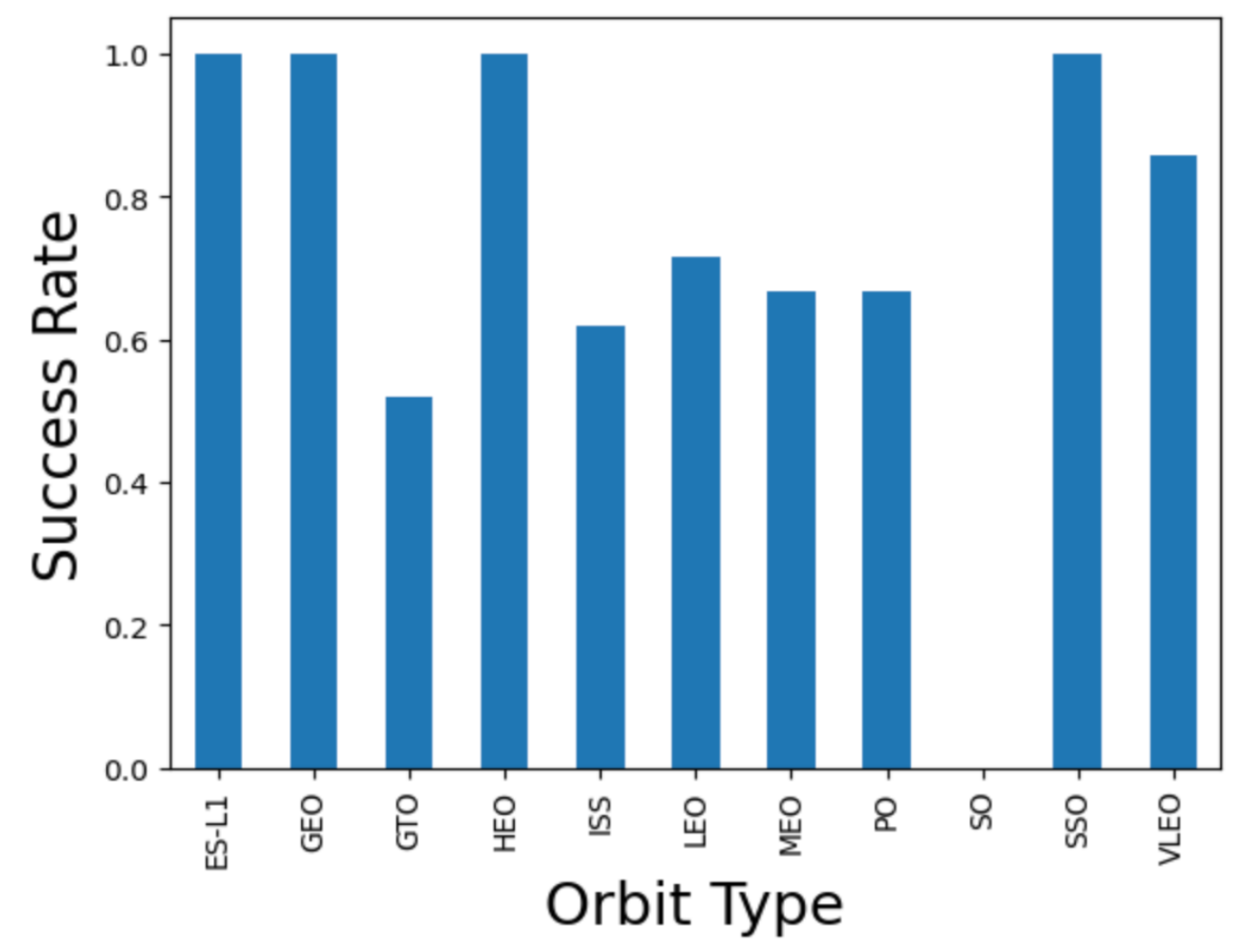
Payload vs. Launch Site

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class va
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



- Analysis of Payload Mass vs Launch Site through scatter plot diagram which shows that CCAFS SLC 40 and KSC LC 39A appear to be favoured for heavier payloads

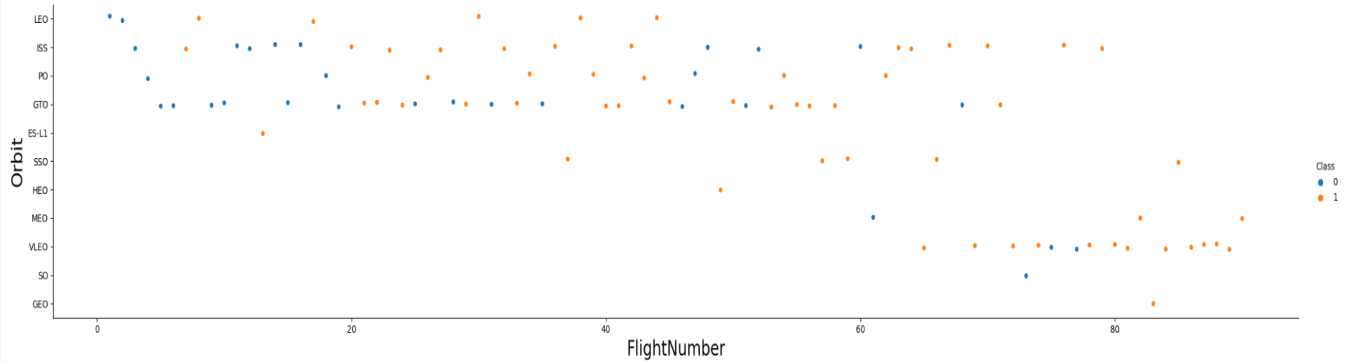
Success Rate vs. Orbit Type



- Analysis of Orbit type vs Success rate where it is observed that all orbit types except 'SO' have had successful 1st stage landings

Flight Number vs. Orbit Type

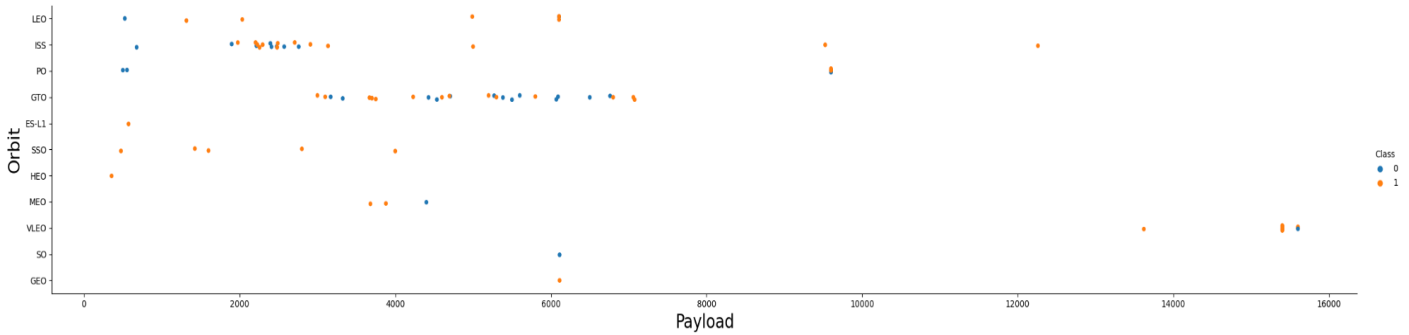
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



- Analysis of Flight Number vs Orbit type, which shows that flight number positively correlated with 1st stage recovery for all orbit types

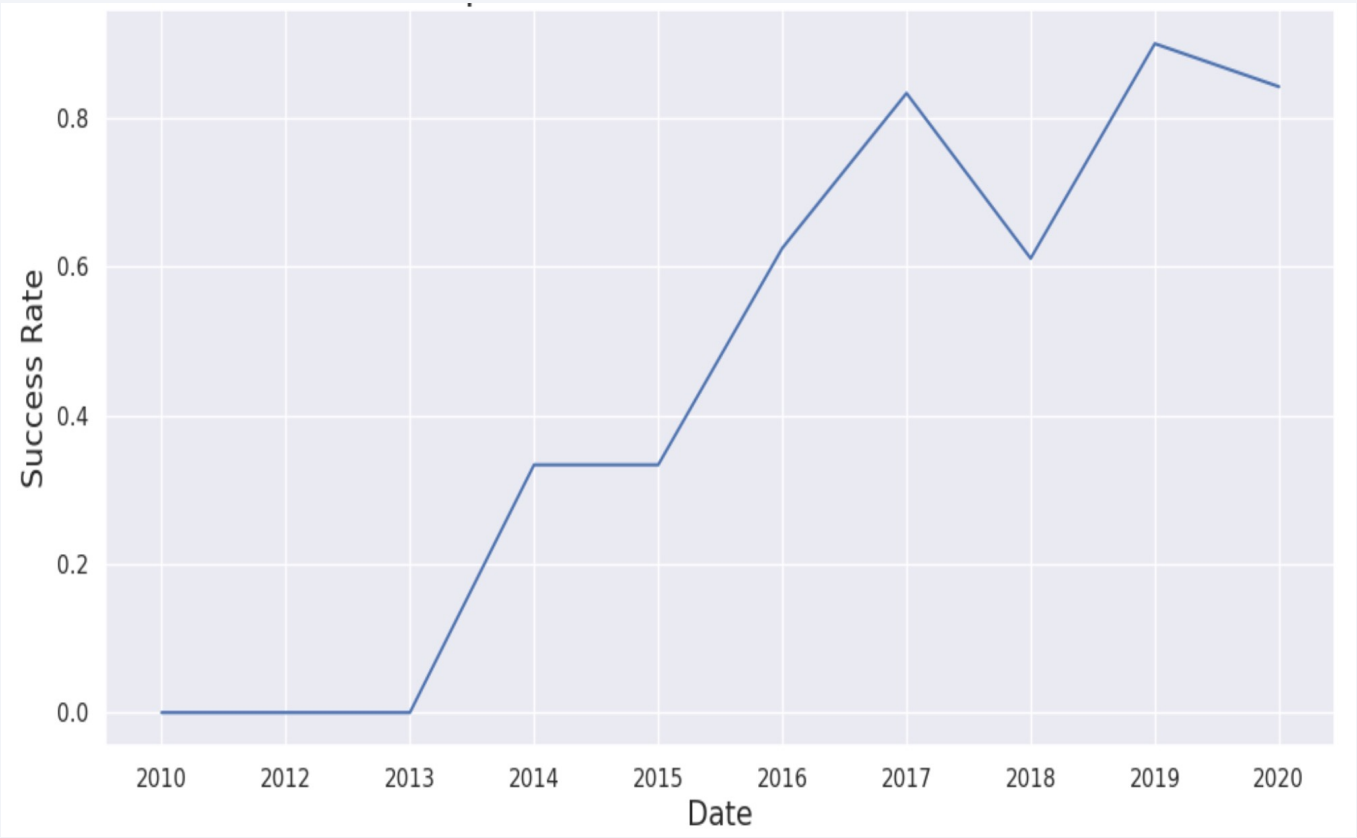
Payload vs. Orbit Type

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



- Analysis of Payload Mass vs Orbit type which shows that heavier payloads have a negative influence on GTO orbits and positive influence on ISS orbits.

Launch Success Yearly Trend



- An analysis of the Year vs Success rate shows that the success rate trends positively on a yearly basis from 2013 till 2020.

All Launch Site Names

- The key word DISTINCT has been used to display unique launch sites only from the SpaceX data.

[9]:
```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL;
```

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

[9]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

Launch Site Names Begin with 'CCA'

The query to display 5 records where launch sites begin with `CCA`

[13]:
```
%%sql
SELECT LAUNCH_SITE
FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

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

[13]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- The total payload mass carried by boosters from NASA was 45596.0 as shown using the query below

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596.0

Average Payload Mass by F9 v1.1

- Average payload mass carried by booster version F9 v1.1 calculated as 340.4 KG

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.0%';
```

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

AVG(PAYLOAD_MASS__KG_)
340.4

First Successful Ground Landing Date

- Min() function used to find the result
- It is observed that the date of the first successful landing outcome on ground pad was 1st August 2018

```
%sql SELECT MIN (DATE) AS "First Successful Landing" FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

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

First Successful Landing
01/08/2018

Successful Drone Ship Landing with Payload between 4000 and 6000

- WHERE clause has been used to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ between 4000 and 6000 AND LANDING_OUTCOME = 'Succes
```

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

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

Total Number of Successful and Failure Mission Outcomes

- The following query was used to find the Mission Outcome as a success or a failure:
- Success Count - 100,
- Failure Count - 1.

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

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

Mission_Outcome	TOTAL_NUMBER
None	0
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- The boosters that have carried the maximum payload were displayed using a subquery in the WHERE clause and the MAX() function.

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

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- A combination of the WHERE clause, LIKE, AND, and BETWEEN conditions has been used to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

```
%%sql
SELECT LANDING_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE Landing_Outcome = 'Failure (drone ship)'
    AND "DATE" LIKE '%2015%';
```

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

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

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

- Landing outcomes selected along with the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.
- GROUP BY clause applied to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

```
%%sql
SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE DATE BETWEEN '04/06/2010' AND '20/03/2017'
GROUP BY LANDING_OUTCOME
ORDER BY TOTAL_NUMBER DESC;
```

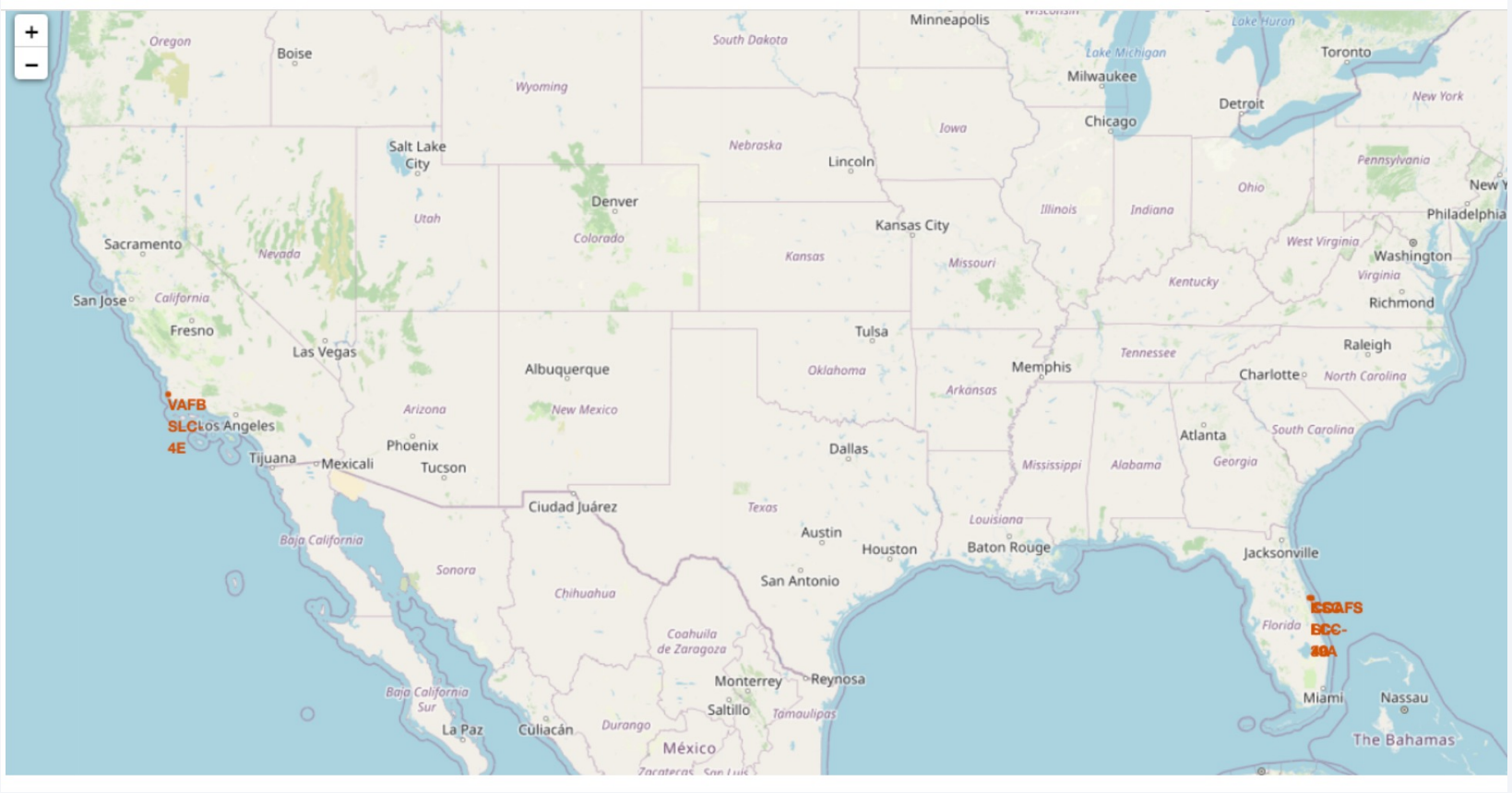
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	TOTAL_NUMBER
Success	20
No attempt	9
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

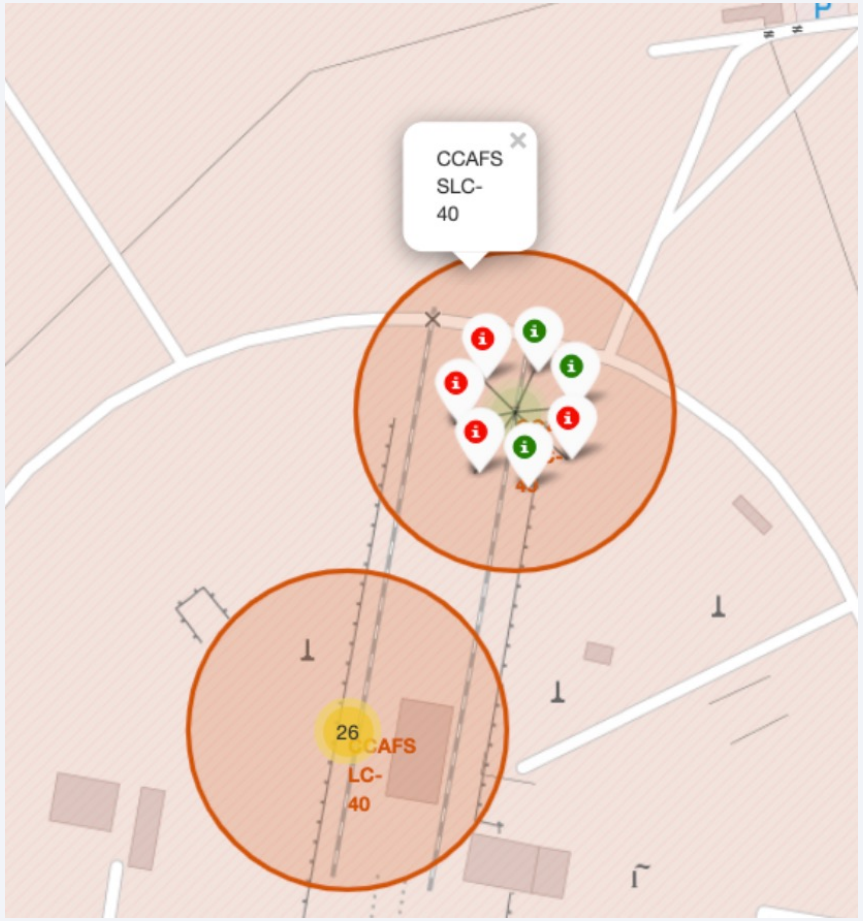

Section 3
Launch Sites
Proximities Analysis

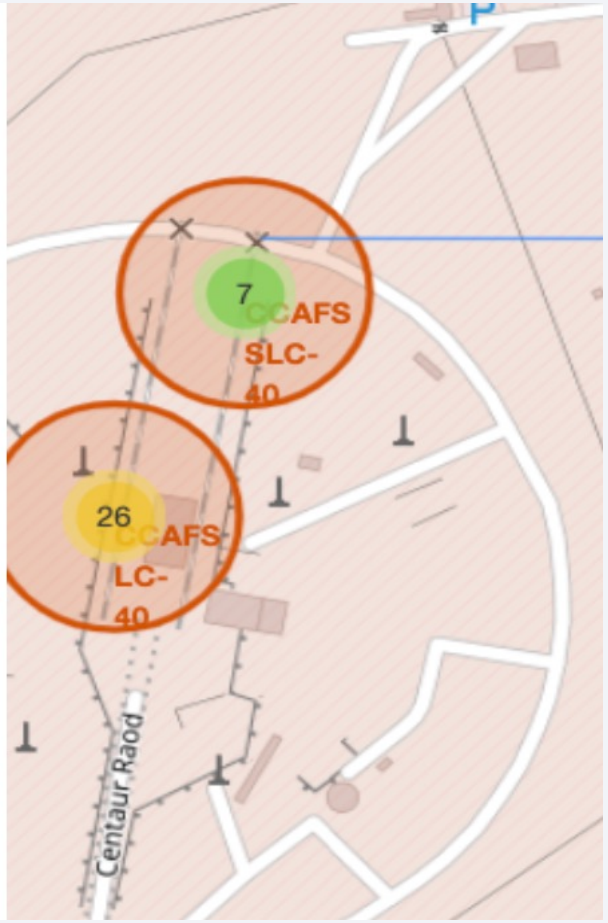
Location of Launch Sites

- All the SpaceX launch sites are located inside the United States highlighting the importance of the launch sites near the coastline and the equator



Markers showing launch sites with colour labels





Results

- Visualizing the railway, highway, coastline, and city proximities for each launch site allows us to see how close each is, for example:
- Proximities for CCAFS SLC-40:
- Railway:1.28 km
- Transporting heavy cargo • highway: 0.58 km
- Transporting personnel and equipment
- Coastline: 0.86 km
  - optionality to abort launch and attempt water landing
  - minimizing risk from falling debris
- City: 51.43 km
- minimizing danger to population dense areas

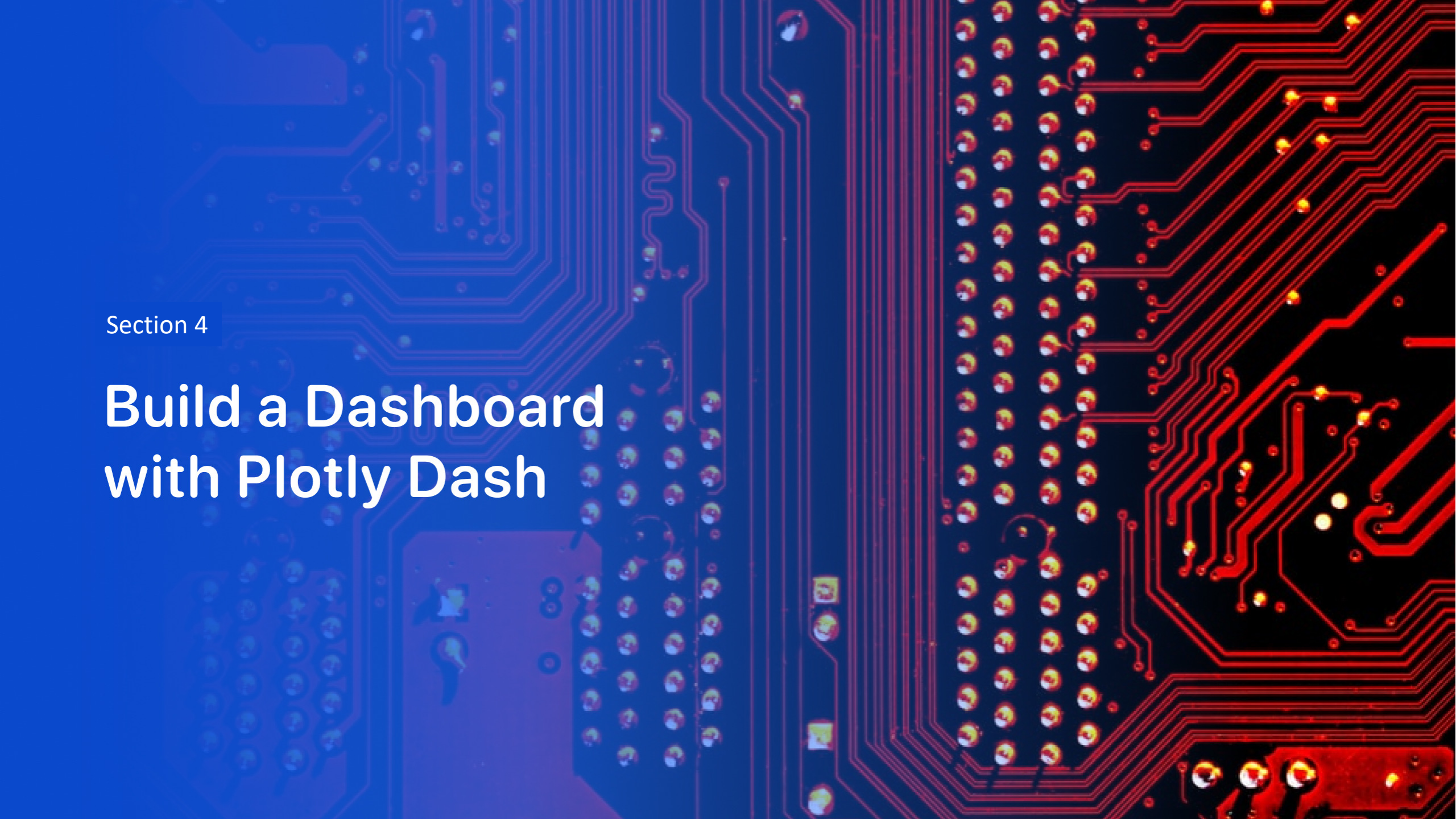
```
# find coordinate of the closet coastline
# e.g.,: Lat: 28.56367  Lon: -80.57163
# distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
launch_site_lat = 28.563197
launch_site_lon = -80.576820
coastline_lat = 28.56334
coastline_lon = -80.56799
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
print(distance_coastline,' km')
```

```
0.8627671182499878  km
```

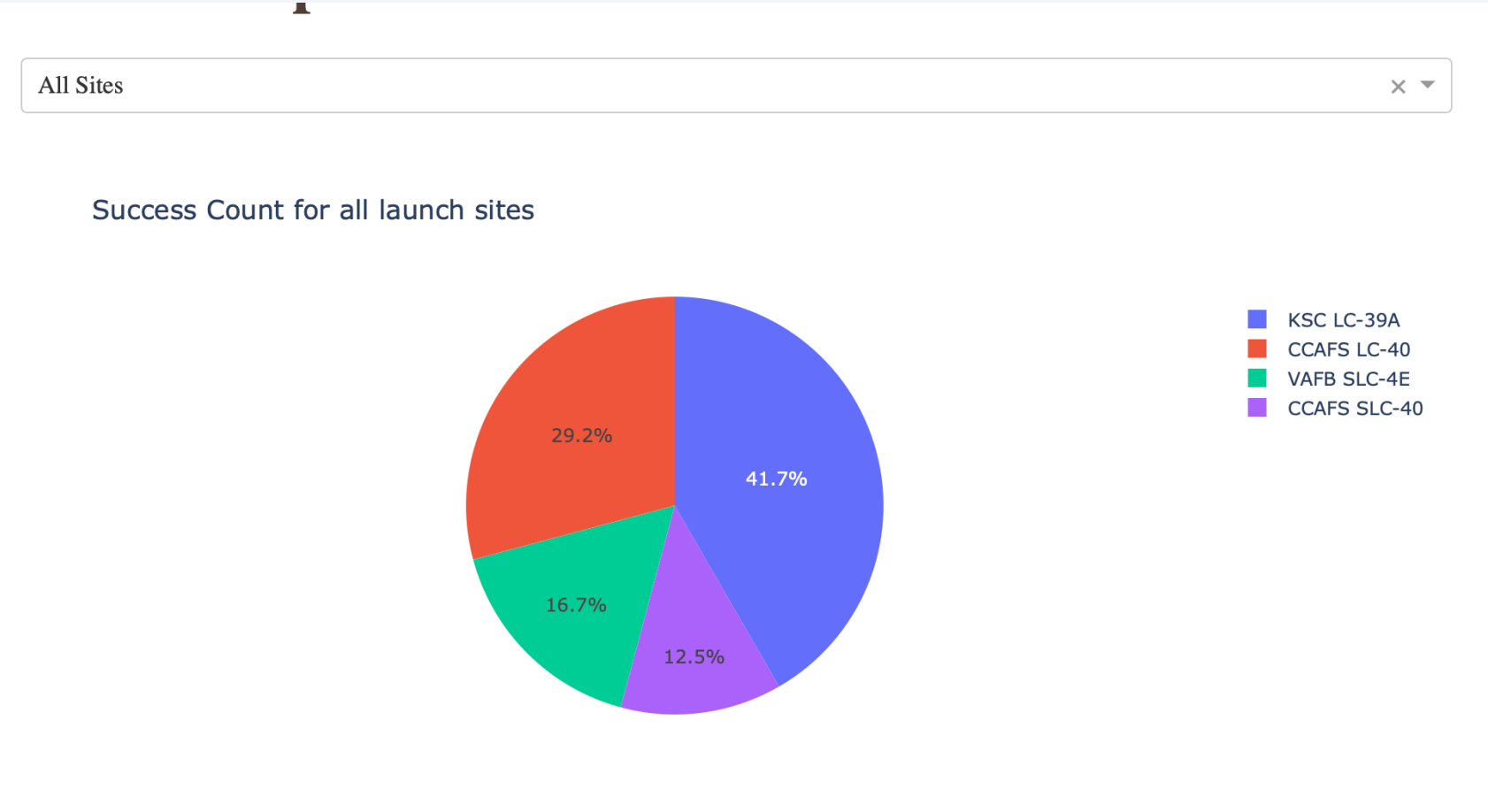
```
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the marker to the launch site
closest_highway = 28.56335, -80.57085
closest_railroad = 28.57206, -80.58525
closest_city = 28.10473, -80.64531
```

```
distance_highway = calculate_distance(launch_site_lat, launch_site_lon, closest_highw
print('distance_highway =',distance_highway, ' km')
distance_railroad = calculate_distance(launch_site_lat, launch_site_lon, closest_rail
print('distance_railroad =',distance_railroad, ' km')
distance_city = calculate_distance(launch_site_lat, launch_site_lon, closest_city[0],
print('distance_city =',distance_city, ' km')
```

```
distance_highway = 0.5834695366934144  km
distance_railroad = 1.2845344718142522  km
distance_city = 51.434169995172326  km
```

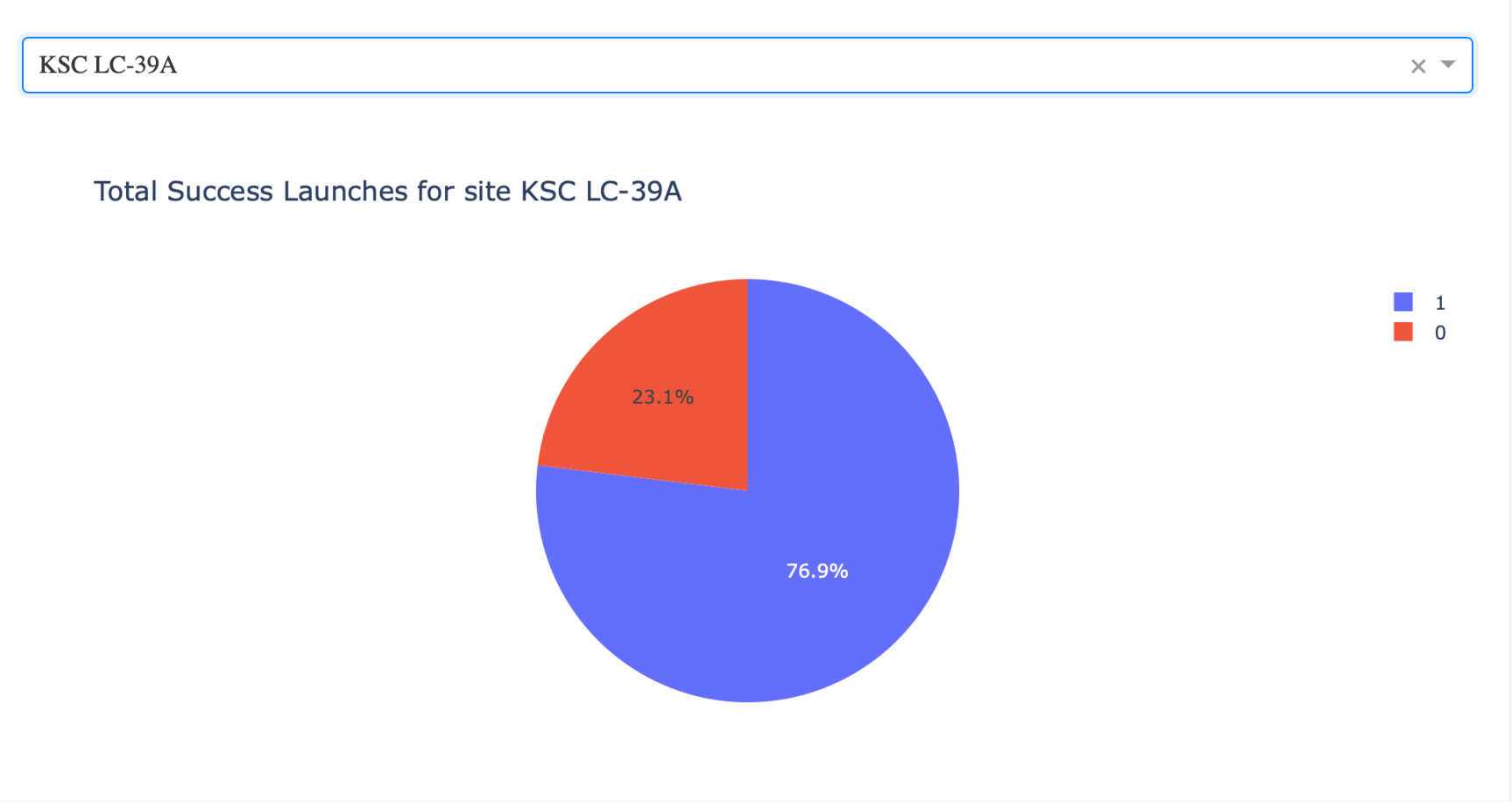

Section 4
Build a Dashboard with Plotly Dash

The success percentage by each sites.



The graph demonstrates that KSC LC-39A had the most successful launch compared to others.

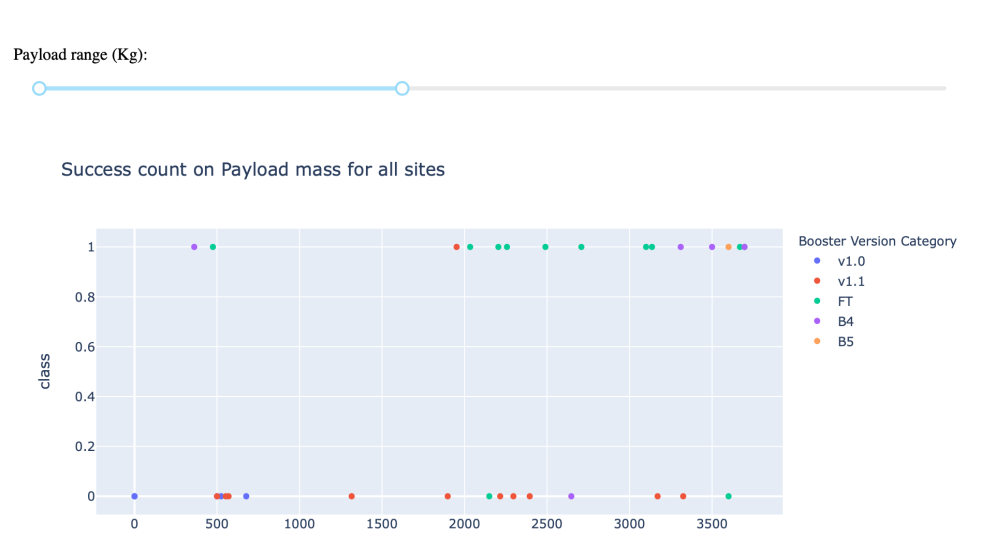
The highest launch-success ratio: KSC LC-39A



KSC LC-39A had a success rate of 76.9% compared to a failure rate of 23.1%

Payload vs Launch Outcome Scatter Plot

- It is observed that the success rate for low weighted payload is higher than heavy weighted payloads



Section 5

Predictive Analysis (Classification)

Classification Accuracy

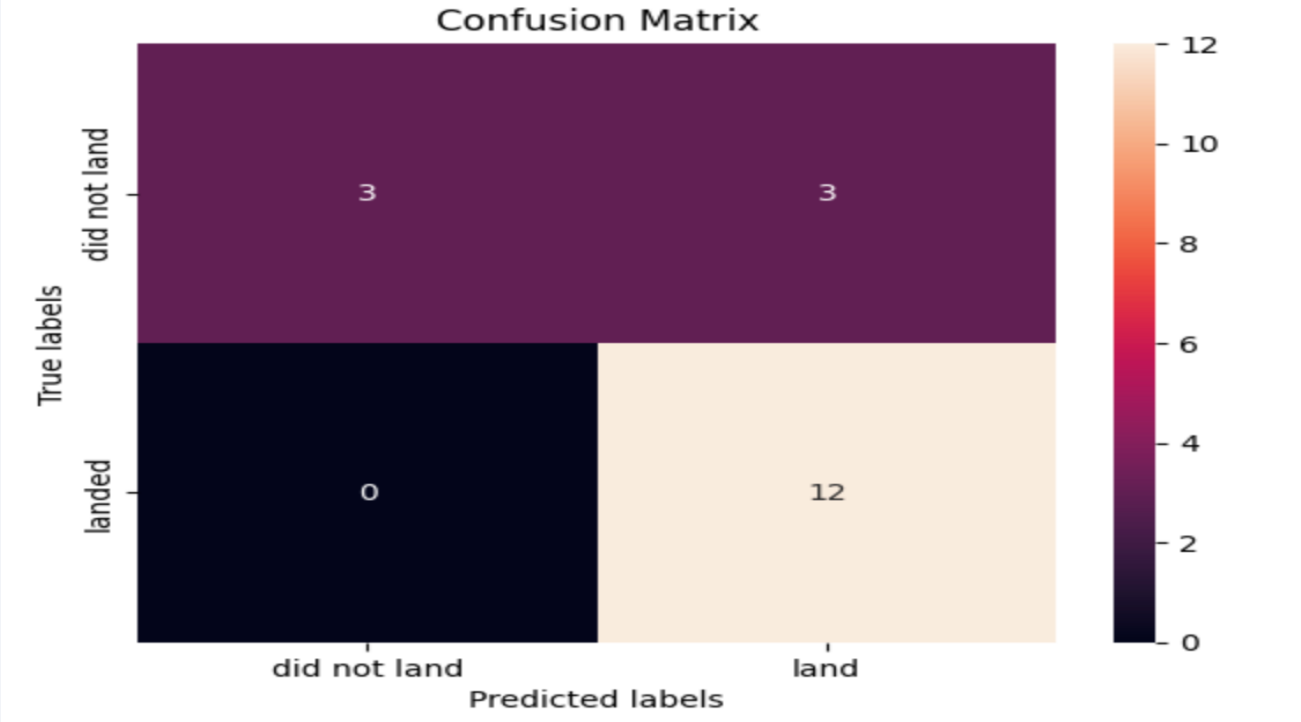
- Using the code below it has been identified that the best algorithm is the Tree Algorithm which has the highest classification accuracy.

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8625
Best Params is : {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10,
'splitter': 'random'}
```

Confusion Matrix

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



- The confusion matrices of the best performing models (4-way-tie) are the same
- The major problem is false positives as evidenced by the models incorrectly predicting the 1st stage booster to land in 3 out of 18 samples in the test set

Conclusions

- We can conclude that:
- Using the models from this report SpaceY can predict when SpaceX will successfully land the 1st stage booster with 86% accuracy
- The Tree Classifier Algorithm is the best Machine Learning approach for this dataset.
- The low weighted payloads (which is defined as 4000kg and below) performed better than the heavy weighted payloads.
- Starting from the year 2013, the success rate for SpaceX launches increased with time in years to 2020, which tends towards perfection and it will eventually perfect the launches in the future.
- KSC LC-39A have the most successful launches of any sites; 76.9%
- SSO orbit have the most success rate; 100% and more than 1 occurrence.

Appendix

- **GitHub references:**
- https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/jupyter-labs-spacex-data-collection-api%20(2).ipynb
- https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/jupyter-labs-webscraping%20(2).ipynb
- https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite%20(1).ipynb
- https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb
- https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb
- https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb
- https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/Hands-on%20Lab_%20Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash.ipynb
- https://github.com/mitras077/Shonxfw/blob/8a5cd893f10a0cd77cf5871da70a605e4be23eab/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

- **Acknowledgement:**

‘ Thank you’ - To all the instructors for creating the course and learning materials with excellent insight.

Thank you!