IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Kindy

January 29th 2024

# Outline

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

# Executive Summary

The purpose of this research is to gather and analyze SpaceX Falcon 9 data and employ Machine Learning models to predict a successful first stage landing, thereby deriving the cost of a launch. This information can be employed by alternate company to bid against SpaceX for a rocket launch.

- Summary of methodologies
  - The following methods were applied to collect and analyze data, build and evaluate machine learning models, and make predictions:
    - Collect data through API and Web scraping
    - Transform data through data wrangling
    - Conduct exploratory data analysis with SQL and data visuals
    - Build an interactive map with folium to analyze launch site proximity
    - Build a dashboard to analyze launch records interactively with Plotly Dash
    - Utilized several predictive models to predict a successful landing of the first stage of Falcon 9
- Summary of all results
  - The report will share results as follows:
    - Data analysis results
    - Data visuals, interactive dashboards
    - Predictive model analysis results

# Introduction

- Project background and context
  - Given the recent successes in space travel, the space industry is becoming more mainstream and accessible to the general population. The cost of launch remains the main barrier to entry for new company looking to join the space race.
  - SpaceX is currently leading the industry. The linchpin of its success is its partially reusable Falcon 9 rocket, as boosters makeup over half the manufacturing cost of the rocket, and elegantly landing and reusing these allows for major savings. It costs SpaceX around $67 million per launch while competitors spend anywhere from $167 to $4,000 million
- Problems you want to find answers
  - Determine if the first stage of SpaceX Falcon 9 will land successfully
  - Impact of different parameters/variables on the landing outcomes (e.g., launch site, payload mass, booster version, etc.)
  - Correlations between launch sites and success rates

Section 1

# Methodology

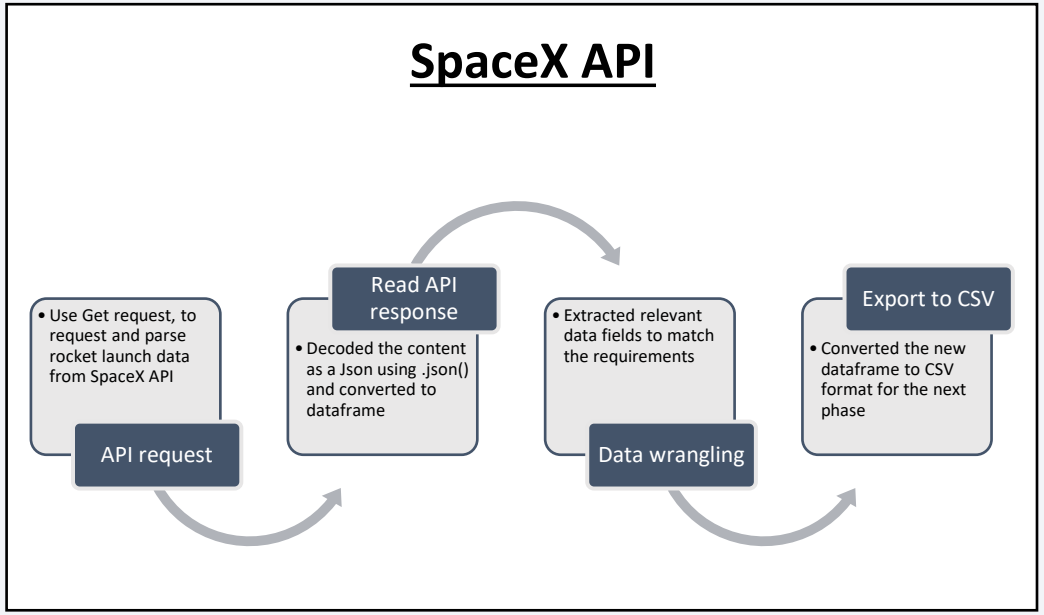
# Methodology

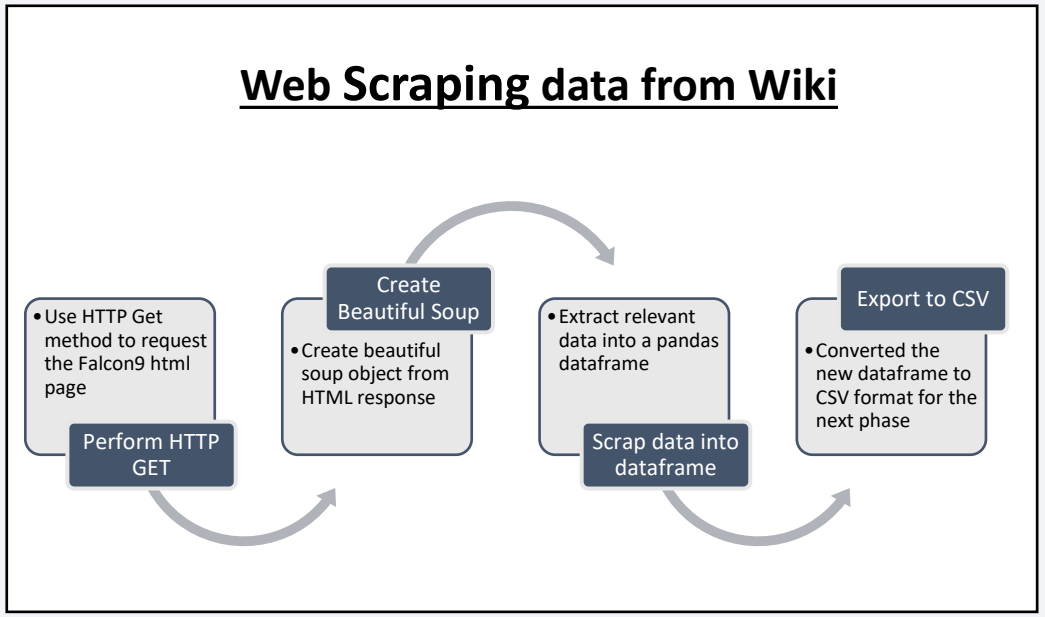
## Executive Summary

- Data collection methodology:
  - SpaceX API
  - Web scrap Falcon 9 and Falcon Heavy launch records from Wikipedia (link)
- Perform data wrangling
  - Determine patterns within the data and creating labels for the supervised models by converting mission outcomes to training labels (0-unsuccessful, 1-successful)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Split the data into training data and test data to find the best Hyperparameter for SVM, Classification Trees, and Logistic Regression.

# Data Collection

- For this project data was collected from two sources using two method. The first is by making a get-request to the SpaceX API while the second is by web scraping Wikipedia for Falcon 9 historical launch records



# Data Collection – SpaceX API

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

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

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

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

```
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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

```
# Create a data from launch_dict
launch_df = pd.DataFrame(launch_dict)
```

```
# Calculate the mean value of PayloadMass column
avg_PayloadMass = data_falcon9['PayloadMass'].astype("float").me

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, np.round(avg_Payload
data_falcon9.isnull().sum()
```

**1. API Get Request and read to dataframe**

- Use requests.get(spacex_url) to request the launch data from the API
- Decode the json response content and turn it into a Pandas dataframe
- Majority of the data are IDs. We use these IDs to extract the desired data from the API

**2. Declare global variables & apply helper functions**

- A set of global variables are declared as lists
- Using helper functions, ID data is used to extract the desired data from the API
- A new dataframe is created with this data

**Data Wrangling**

- Payload Mass and Landing Pad variable both contained missing values
- Landing Pad variable retained none values to represent when landing pads were not used. Payload Mass missing values were replaced by average payload mass value

**Export dataset to CSV**

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

More details available here

# Data Collection - Scraping

```
wiki_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
spaceX_data = requests.get(wiki_url).text
soup = BeautifulSoup(spaceX_data, "html5lib")
```

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

```
column_names = []
for hd in first_launch_table.find_all('th'):
    name = extract_column_from_header(hd)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

```
launch_dict = dict.fromkeys(column_names)
```

```
launch_dict = dict.fromkeys(column_names)

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']=[]
```

```
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)
            #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)
            #print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            #print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            #print(bv)
            launch_dict['Version Booster'].append(bv)
```

1. HTTP Get Request and Create BeautifulSoup object
   - Apply the get method to request the launch data from the Wiki URL
   - Create a beautiful soup object from the HTML response
2. Extract variable names from HTML table reader
   - Use the find_all method on the BeautifulSoup object, to extract relevant column names from the HTML table header
   - Use this column names to create a dictionary
3. Parse through HTML tables and create Pandas dataframe
   - Create global variables as list
   - Parse through HTML table and extract the relevant data into the list
   - Create a dataframe with the extracted data

Export dataset to CSV
- df.to_csv('spacex_web_scraped.csv', index=False)

GitHub

# Data Wrangling

- We conduct Exploratory Data Analysis (EDA) to uncover potential patterns in the data and determine labels for training supervised models.
- The data set contained various mission outcomes that were converted into Training Labels with 1 meaning the booster successfully landed and 0 meaning booster was unsuccessful in landing. Following landing scenarios were considered to create labels:
  - True Ocean means the mission outcome was successfully landed to a specific region of the ocean
  - False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean
  - 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

**1. Load dataset into Dataframe**

- Load SpaceX dataset (csv) into Dataframe

**2. Find patterns in Data**

- i. Calculate the number of launches on each site
- ii. Calculate the number and occurrence of each orbit
- iii. Calculate number/occurrence of mission outcomes per orbit type

**3. Create landing outcome label**

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset
```

```
pd.value_counts(df['LaunchSite'])
```

| | |
|---|---|
| CCAFS SLC 40 | 55 |
| KSC LC 39A | 22 |
| VAFB SLC 4E | 13 |

```
pd.value_counts(df['Orbit'])
```

| | |
|---|---|
| GTO | 27 |
| ISS | 21 |
| VLEO | 14 |
| PO | 9 |
| LEO | 7 |
| SSO | 5 |
| MEO | 3 |
| ES-L1 | 1 |
| HEO | 1 |
| SO | 1 |
| GEO | 1 |

```
landing_class = []
for x in df['Outcome']:
    if x in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)

df['Class']=landing_class
df[['Class']].head(8)
```

| | Class |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |

# EDA through Data Visualization

- As part of the Exploratory Data Analysis (EDA), following charts were plotted to gain further insights into the dataset:
    1. Scatter plot:
        - Shows relationship or correlation between two variables making patterns easy to observe
        - Plotted following charts to visualize
            - Relationship between Flight Number and Launch Site
            - Relationship between Payload and Launch Site
            - Relationship between Flight Number and Orbit Type
            - Relationship between Payload and Orbit Type
    2. Bar Chart
        - Indicates how different categories differ from one another.
        - Categorized success frequency by orbit type to visualize the relationship between success rate at each orbit type
    3. Line Chart
        - Indicates changes over a period of time.
        - Plotted success rate over time to visualized the launch success yearly trend

# EDA with SQL

- To better understand SpaceX data set, following SQL operations were performed on an IBM DB2 cloud instance:
  - I. Identified the names of the unique launch sites in the space mission
    - i. Displayed 5 records where launch sites begun with the string 'CCA'
  - II. Derived the total payload mass carried by boosters launched by NASA (CRS)
  - III. Derived the average payload mass carried by booster version F9 v1.1
  - IV. Identified the date when the first successful landing in ground pad was achieved
  - V. Obtained the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - VI. Listed the total number of successful and failure mission outcomes
  - VII. Obtained the names of the booster versions which have carried the maximum payload mass.
  - VIII. Listed the failed landing outcomes in drone ship for the year of 2015, as well as the corresponding month, booster version and launch site
  - IX. Ranked the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order.

# Build an Interactive Map with Folium

- Folium interactive map helps analyze geospatial data to perform more interactive visual analytics and better understand factors such location and proximity of launch sites that impact launch success rate.
- The followed object objects were created and added to the map:
  - Added ‘folium.circle’ and ‘folium.marker’ to clearly identify each launch site on the map.
  - Added a ‘MarkerCluster()’ to show launch success (green) and failure (red) markers for each launch site.
  - Calculated distances between a launch site to its proximities (e.g., coastline, railroad, highway, city)
    - Added ‘MousePosition() to get coordinate for a mouse position over a point on the map
    - Added ‘folium.Marker()’ to display distance (in KM) on the point on the map (e.g., coastline, railroad, highway, city)
    - Added ‘folium.Polyline()’ to draw a line between the point on the map and the launch site
    - Repeated steps above to add markers and draw lines between launch sites and proximities –coastline, railroad, highway, city)

# Build a Dashboard with Plotly Dash

- Built a Plotly Dash web application to perform interactive visual analytics on SpaceX launch data in real-time.
  I. Added a Launch Site Drop-down Input component to the dashboard to provide an ability to filter Dashboard visual by all launch sites or a particular launch site
  II. Added a Pie Chart to the Dashboard to show total success launches when 'All Sites' is selected and show success and failed counts when a particular site is selected
  III. Added a Payload range slider to the Dashboard to easily select different payload ranges to identify visual patterns
  IV. Added a Scatter chart to observe how payload may be correlated with mission outcomes for selected site(s). The color-label Booster version on each scatter point provided missions outcomes with different boosters

# Predictive Analysis (Classification)

1. Read dataset into dataframe and create a 'Class' array
2. Standardize the data
3. Split the data into training and testing set
4. Create and Refine Models
5. Find the best performing model

1. Loaded SpaceX dataset and created a NumPy array from the class in data:

```
URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.
resp1 = await fetch_data(URL1)
#text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
data = pd.read_csv(io.StringIO(resp1))
Y = data['Class'].to_numpy()
print(type(Y))
```

2. Standardize data and assign to variable X

```
# students get this
#transform = preprocessing.StandardScaler()
X = preprocessing.StandardScaler().fit(X).transform(X.as
```

3. Split the data into train and test sets, with test parameter as 20%

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2,
random_state=2) #what does the random_state parameter control
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

4. Created four different models and calculated predictive accuracy using the score method and a confusion matrix to determine the best model

i. Model 1: Logistic Regression

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
# Define the hyperparameter grid
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1
# Create a Logistic Regression model
lr = LogisticRegression()
# Create a GridSearchCV object
logreg_cv = GridSearchCV(lr, parameters, cv=10, scoring='accuracy')
# Fit the GridSearchCV object to the data
logreg_cv.fit(X_train, Y_train)
```

ii. Find and display best hyperparameters and accuracy score

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

iii. Repeated for three other models:

- ✓ Decision Tree
- ✓ KNN
- ✓ SVM

5. Find the best performing model

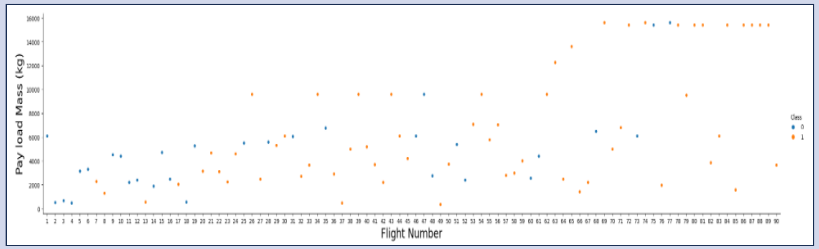
```
Model_Performance_df = pd.DataFrame({
    'Algorithm Type': ['Logistic Regression', 'SVM', 'Decision Tree', 'KNN'],
    'Accuracy Score': [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_s
    'Test Data Accuracy Score': [logreg_cv.score(X_test, Y_test), svm_cv.score(X
        tree_cv.score(X_test, Y_test), knn_cv.score(X_test, Y_test)]})

Model_Performance_df.sort_values(['Accuracy Score'],
                                 ascending = False, inplace = True)

Model Performance df
```

| | Algorithm Type | Accuracy Score | Test Data Accuracy Score |
|---|---|---|---|
| 2 | Decision Tree | 0.875000 | 0.722222 |
| 3 | KNN | 0.848214 | 0.833333 |
| 1 | SVM | 0.848214 | 0.833333 |
| 0 | Logistic Regression | 0.846429 | 0.833333 |

# Results

## Exploratory data analysis results

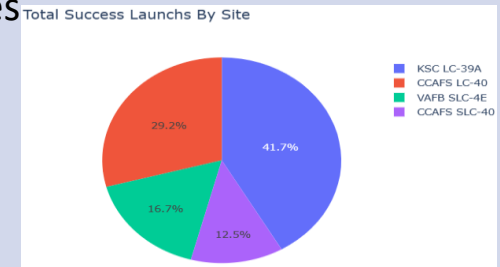
- Samples

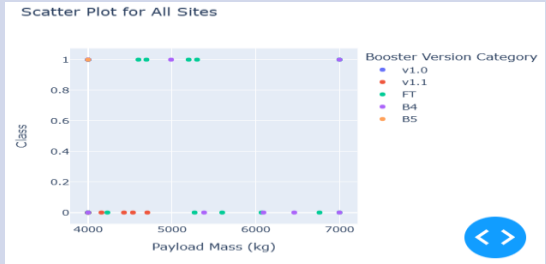




## Interactive analytics demo in screenshots

- Samples

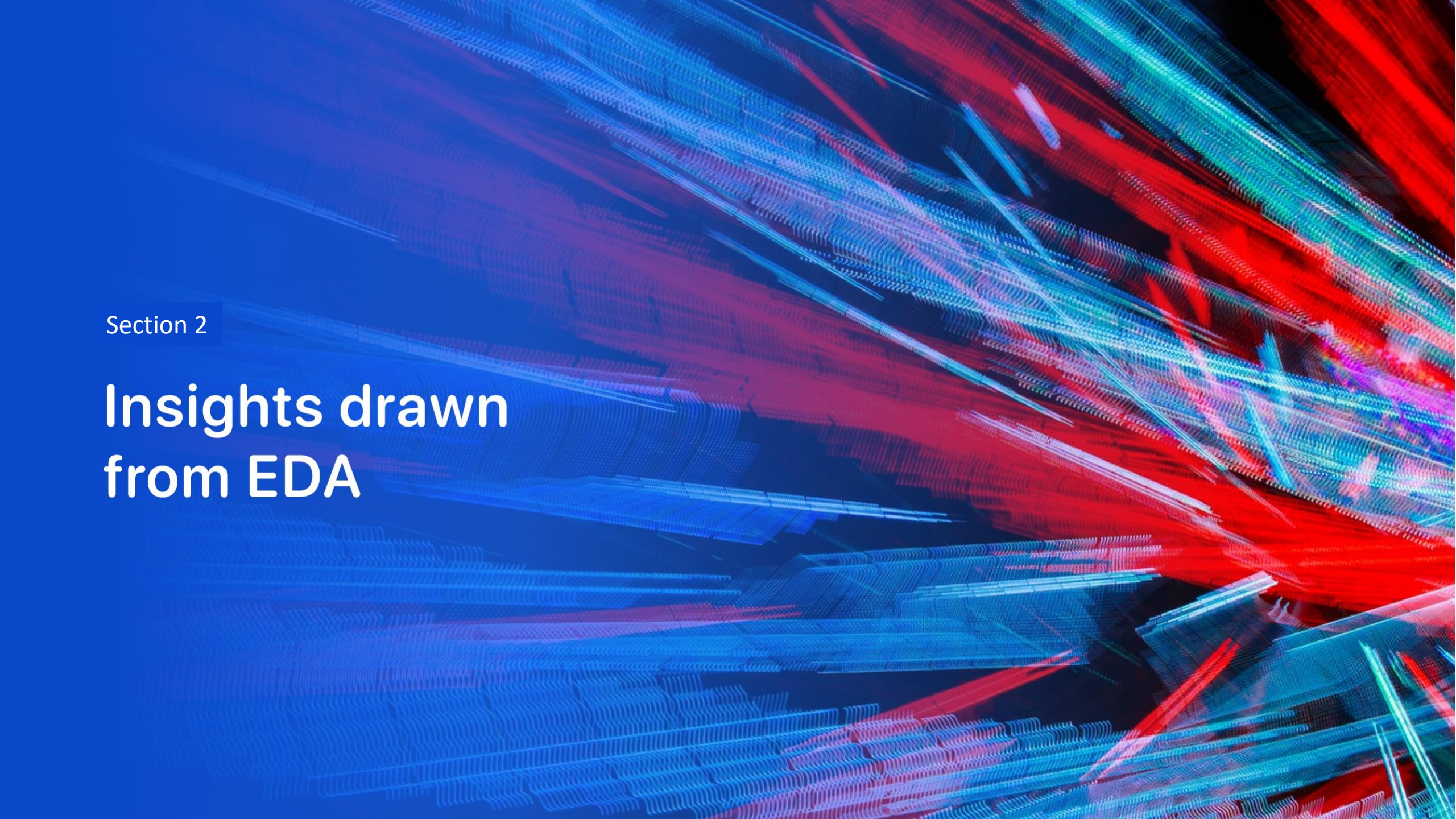


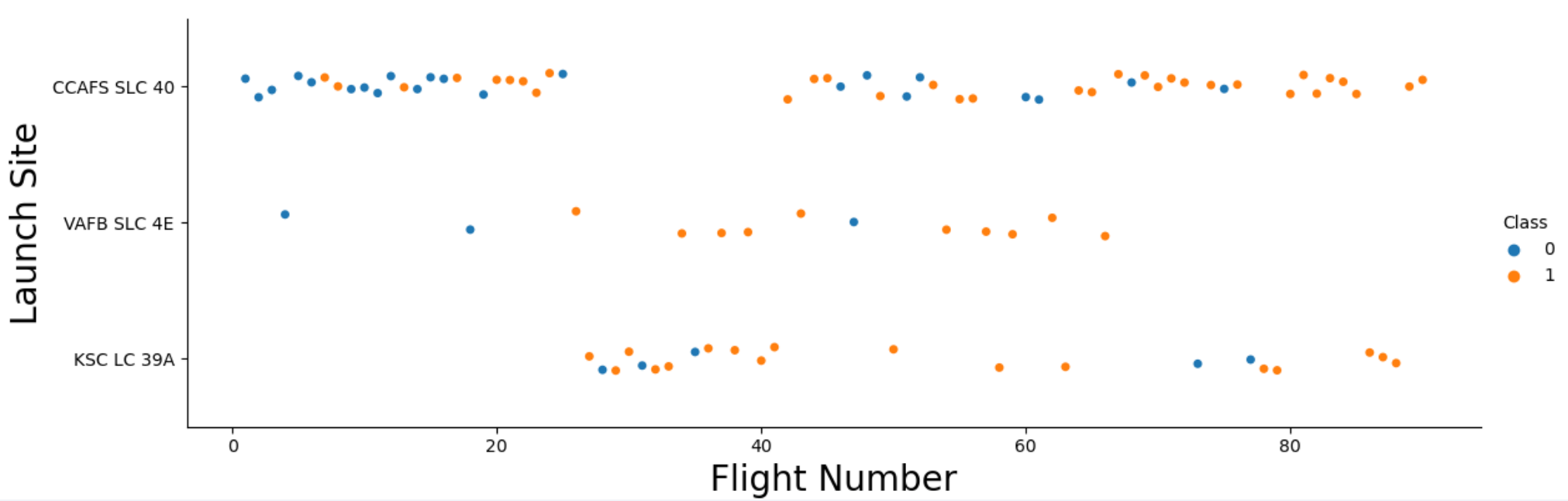

## Interactive analytics demo in screenshots

- Samples

| | Classification Model | Accuracy Score |
|---|---|---|
| 2 | Decision Tree | 0.875000 |
| 3 | KNN | 0.848214 |
| 1 | SVM | 0.848214 |
| 0 | Logistic Regression | 0.846429 |


Section 2
Insights drawn from EDA

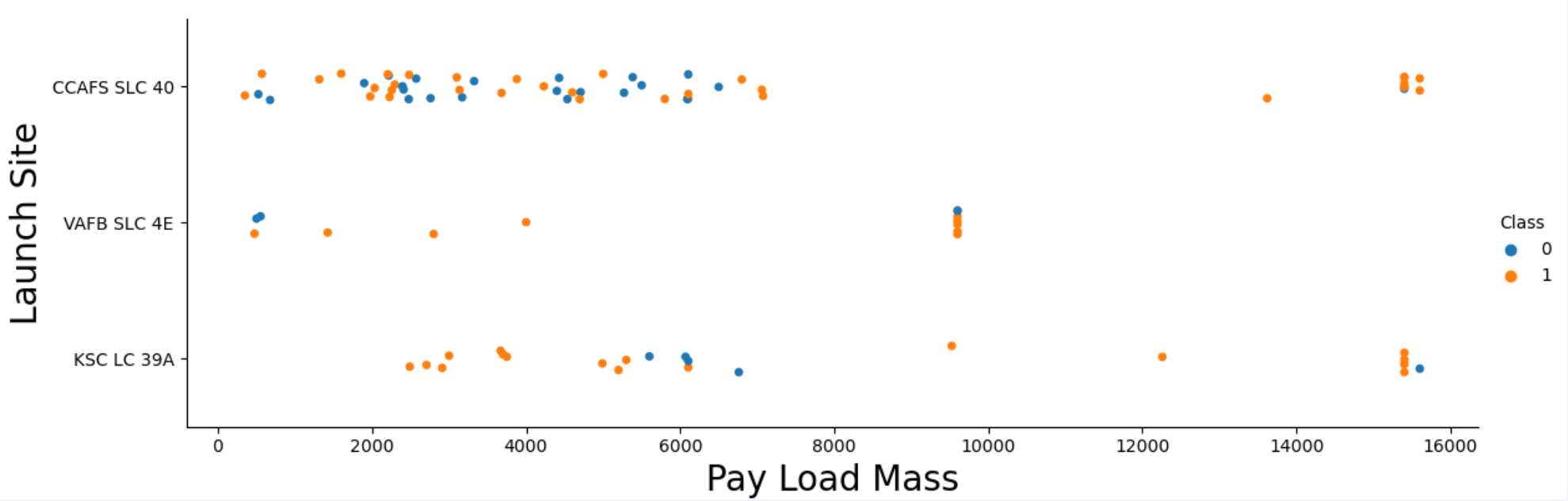
# Flight Number vs. Launch Site




Observations:

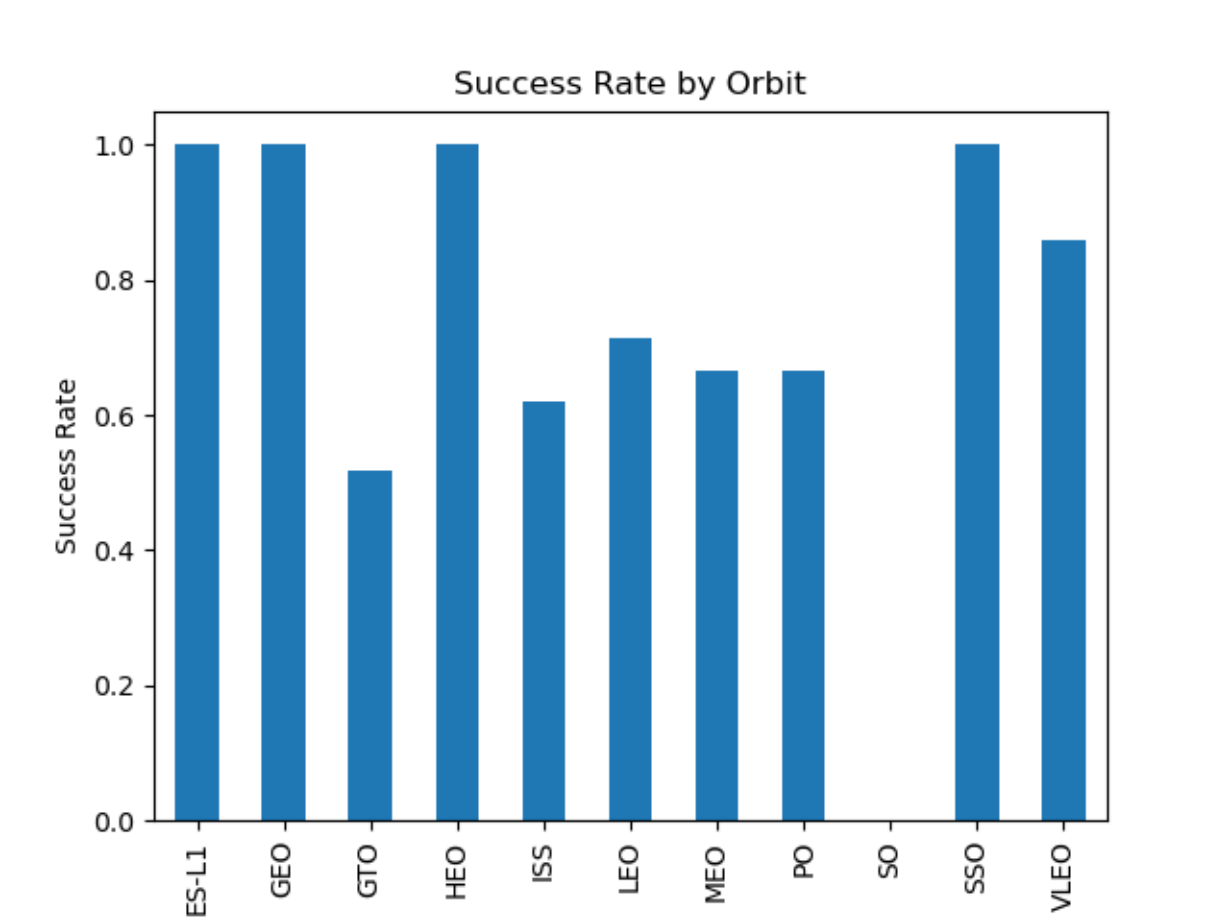
- Success rates (Class=1) increases as the number of flights increase
- Launch sites ‘KSC LC 39A’ and ‘VAFB SLC 4E’ have a higher success rate than CCAFS SLC 40

# Payload vs. Launch Site



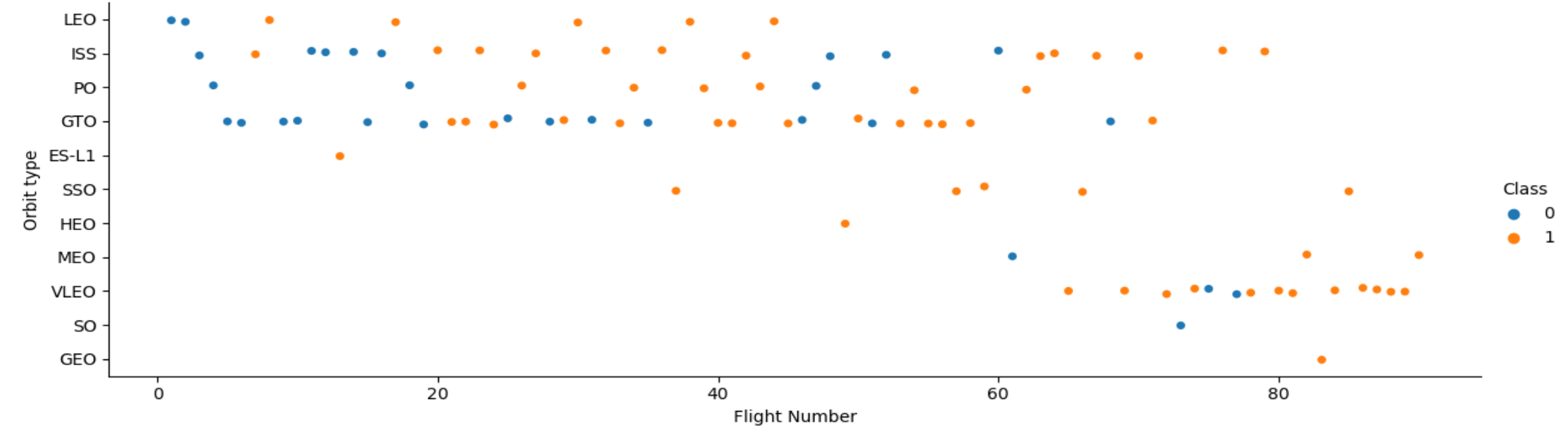

- For the VAFB-SLC launch site there are no rockets launched for heavy payload mass (i.e., 10,000 kg)
- There appears to be a higher success rate at heavier payload masses

# Success Rate vs. Orbit Type



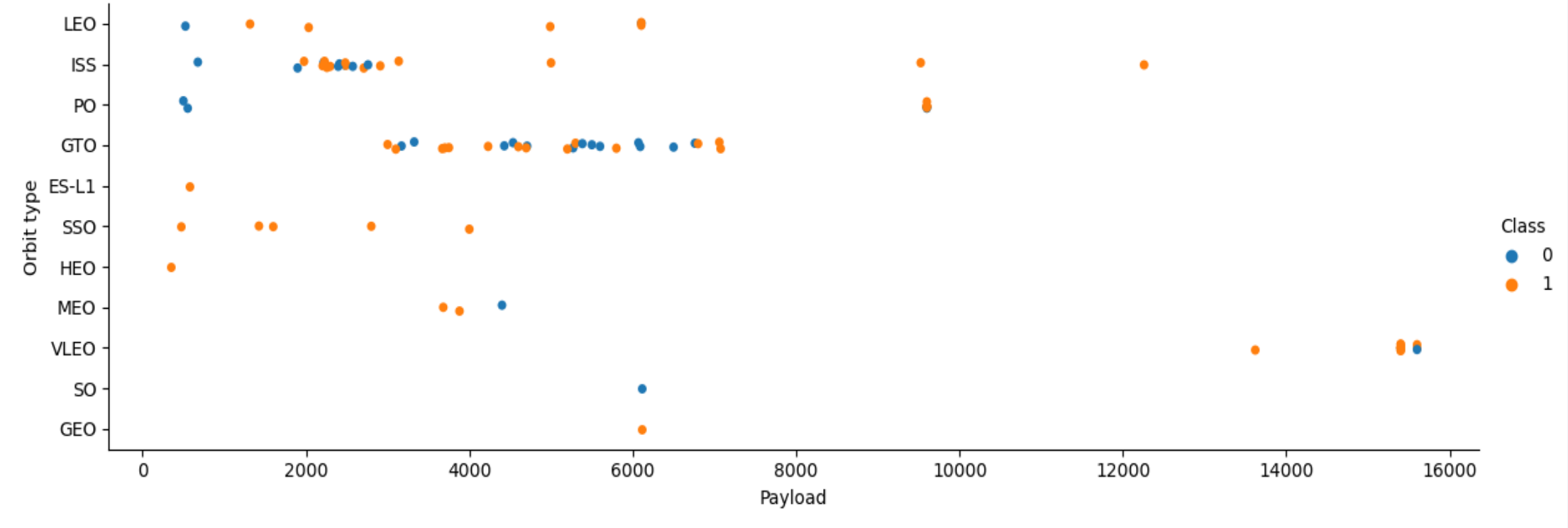

- Orbits ES-LI, GEO, HEO, and SSO have the highest success rates
- GTO orbit has the lowest success rate

# Flight Number vs. Orbit Type




- For orbit VLEO, first successful landing (class=1) doesn't occur until 60+ number of flights
- For most orbits (LEO, ISS, PO, SSO, MEO, VLEO) successful landing rates appear to increase with flight numbers
- There is no relationship between flight number and orbit for GTO

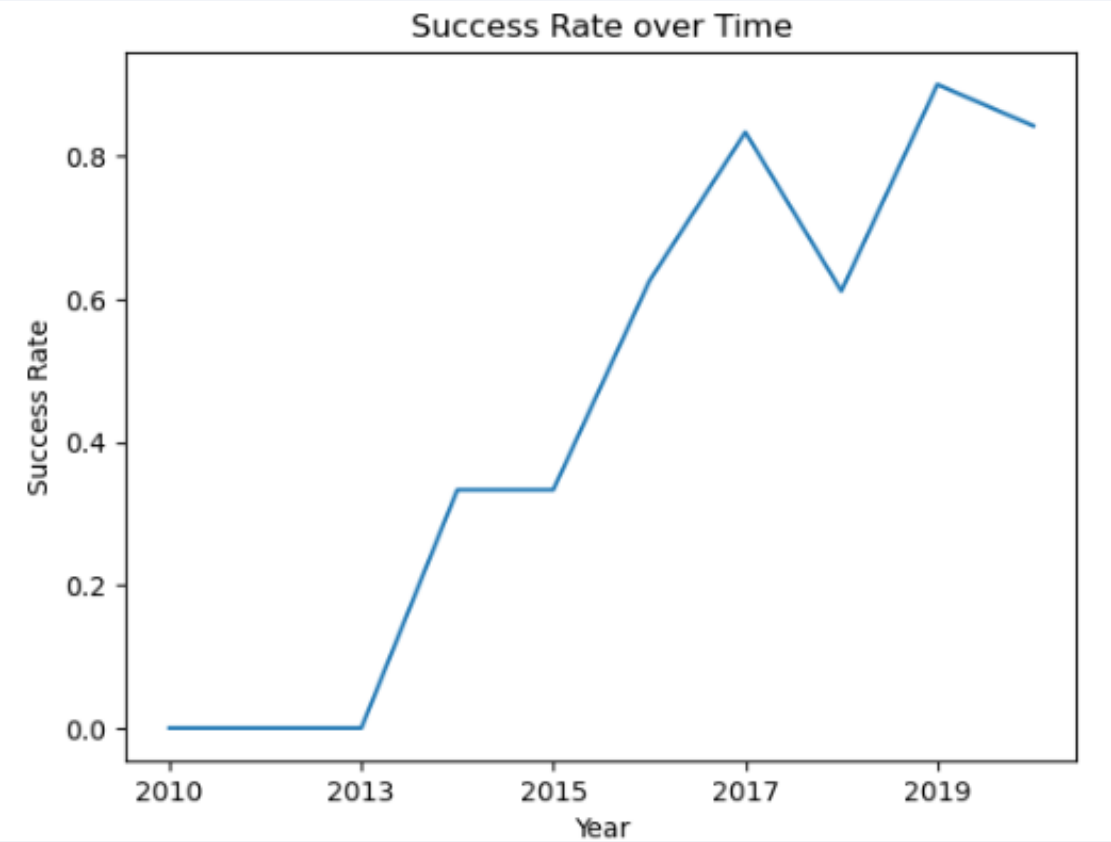
# Payload vs. Orbit Type




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

For GTO orbit, we cannot determine a clear relationship as both positive landing rate and negative landing appear at similar payload values

# Launch Success Yearly Trend




- Success rate (Class=1) increased to about 80% between by 2020
- The success rate remained stable before 2013 and from 2014 to 2015
- It took a dip in 2018 and again between 2019 and 2020

# All Launch Site Names

- Query

```
%sql select DISTINCT "Launch_Site" from SPACEXTABLE
```

- Description:
  - ‘distinct’ returns only unique values from the queries column (Launch_Site)
  - There are 4 unique launch sites
- Results:

| Launch_Site |
|---|
| CCAFS LC-40 |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |

# Launch Site Names Begin with 'CCA'

- Query

- D
```
%sql select * from SPACEXTABLE where "Launch_Site" like 'CCA%' limit 5
```
  - Using keyword ‘Like’ and format ‘CCA%’, returns records where ‘Launch_Site’ column starts with “CCA”.
  - Limit 5, limits the number of returned records to 5

- Results:

| 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 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 0:35:00 | 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

- Query

```
%sql select sum("PAYLOAD_MASS__KG_") from SPACEXTABLE where "Customer" = 'NASA (CRS)';
```

- Description:
  - ‘sum’ adds column ‘PAYLOAD_MASS_KG’ and returns total payload mass for customers named ‘NASA (CRS)’
- Results:

| sum(PAYLOAD_MASS__KG_) |
|---|
| 45596 |

# Average Payload Mass by F9 v1.1

- Query;

```
%sql select AVG("PAYLOAD_MASS__KG_") from SPACEXTABLE where "Booster_Version" = 'F9 v1.1';
```

- Description:
  - ‘avg’ keyword returns the average of payload mass in ‘PAYLOAD_MASS_KG’ column where booster version is ‘F9 v1.1’
- Results:

| AVG(PAYLOAD_MASS__KG_) |
|---|
| 2928.4 |

# First Successful Ground Landing Date

- Query:

```
%sql select MIN("Date") from SPACEXTABLE where "Landing_Outcome" = 'Success (ground pad)';
```

- Description:
  - ‘min(Date)’ selects the first or the oldest date from the ‘Date’ column where first successful landing on group pad was achieved
  - Where clause defines the criteria to return date for scenarios where ‘Landing_Outcome’ value is equal to ‘Success (ground pad)’

- Results:

| MIN(Date) |
| --- |
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

- Query

```
%sql select "Booster_Version" from SPACEXTABLE where "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_"
between 4000 and 6000;
```

- Description:
  - The query finds the booster version where payload mass is greater than 4000 but less than 6000 and the landing outcome is success in drone ship
  - The 'and' operator in the where clause returns booster versions where both conditions in the where clause are true

Results:

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

# Total Number of Successful and Failure Mission Outcomes

- Query:

```
%sql select "Mission_Outcome", count("Mission_Outcome") as Count from SPACEXTABLE group by "Mission_Outcome"
```

- Description:
  - The ‘group by’ keyword arranges identical data in a column in to group
  - In this case, number of mission outcomes by types of outcomes are grouped in column ‘counts’

- Results:

| Mission_Outcome | Count |
|---|---|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

- Query:

```
%sql select "Booster_Version" from SPACEXTABLE where "PAYLOAD_MASS__KG_" = (select Max("PAYLOAD_MASS__KG_") from SPACEXTABLE);
```

- Description:
  - The sub query returns the maximum payload mass by using keywork ‘max’ on the pay load mass column
  - The main query returns booster versions and respective payload mass where payload mass is maximum with value of 15600

- Results:

| 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

- Query:

```
%%sql

select substr("Date", 6, 2) as monthnames, "Landing_Outcome", "booster versions", "Launch_Site"
from SPACEXTABLE
where substr(Date,0,5)='2015' and "Landing_Outcome" = 'Failure (drone ship)'
```

- Description:
  - The query lists landing outcome, booster version, and the launch site where landing outcome is failed in drone ship and the year is 2015
  - The ‘and’ operator in the where clause returns booster versions where both conditions in the where clause are true
  - The ‘year’ keywork extracts the year from column ‘Date
  - The results identify launch site as ‘CCAFS LC-40’ and booster version as F9 v1.1 B1012 and B1015 that had failed landing outcomes in drop ship in the year 2015
- Results:

| monthnames | Landing_Outcome | "booster versions" | Launch_Site |
|---|---|---|---|
| 01 | Failure (drone ship) | booster versions | CCAFS LC-40 |
| 04 | Failure (drone ship) | booster versions | CCAFS LC-40 |

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

- Query:

```
%%sql

select "Landing_Outcome",count("Landing_Outcome") as Count
from SPACEXTABLE
where "Date" BETWEEN '2010-06-04' AND '2017-03-20'
group by "Landing_Outcome"
order by Count DESC;
```

- Description:
  - The ‘group by’ key word arranges data in column ‘Landing_Outcome’ into groups
  - The ‘between’ and ‘and’ keywords return data that is between 2010 06 04 and 2017 03 20
  - The ‘order by’ keyword arranges the counts column in descending order
  - The result of the query is a ranked list of landing outcome counts per the specified date range

- Results:

| Landing_Outcome | Count |
|---|---|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

Section 3

# Launch Sites Proximities Analysis

# SpaceX Falcon9 - Launch Sites Map

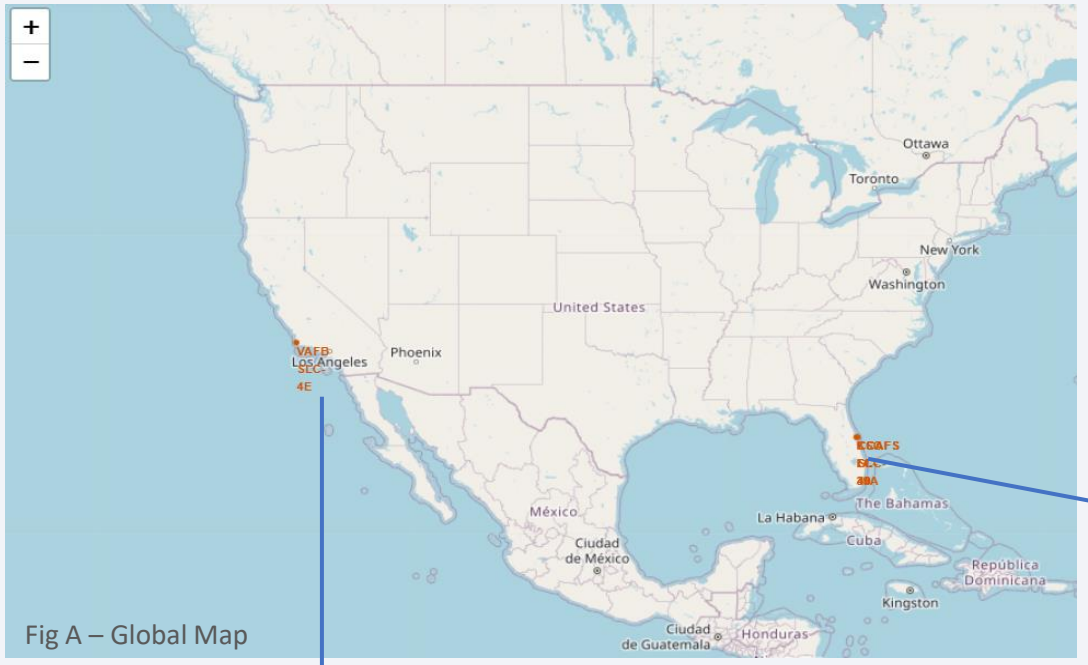


Fig A – Global Map

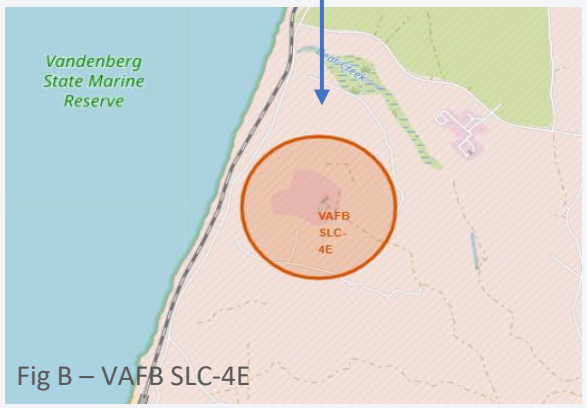


Fig B – VAFB SLC-4E

Figure A, on the left displays the Global map showing the general location of the Falcon 9 launch sites in the United States (Florida and California). We depict each launch site with a circle highlighted in orange, a label and a popup. Fig B and C are the zoomed in version of the Global map with Fig C showing:

- CCAFS LC-40
- CCAFS SLC-40
- KSC LC-39A

Fig B, shows VAFB SLC-4E

Fig C – Remains Sites

# SpaceX Falon9 – Success/Failed Launch Map for all Launch Sites

Fig A – Global Map

Figure A, the US map with all the Launch Sites. The numbers on each site depicts the total number of successful and failed launches.

Figure B to E, are the zoomed in version at each site and displays the success/fail markers with green as success and red as failed

As shown in Figure C, KSC LC-39A launch site has the greatest number of successful launches

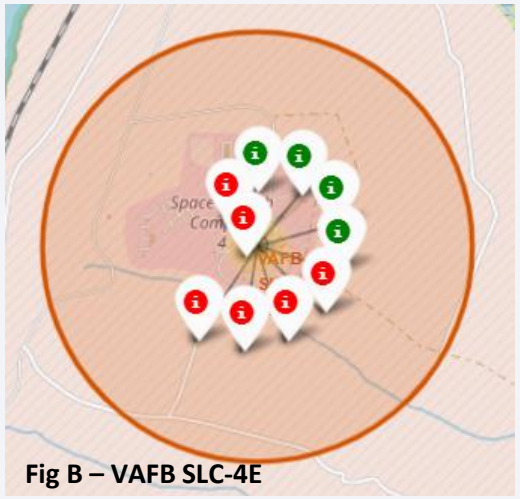


Fig B – VAFB SLC-4E

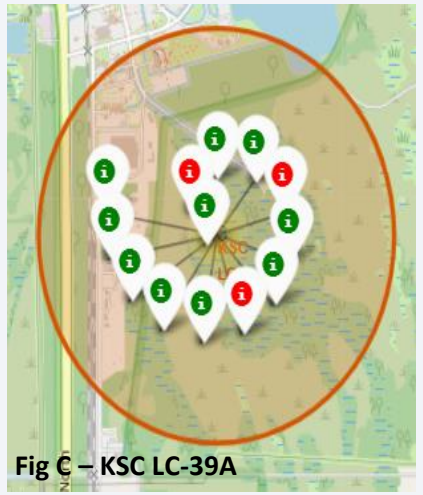


Fig C – KSC LC-39A

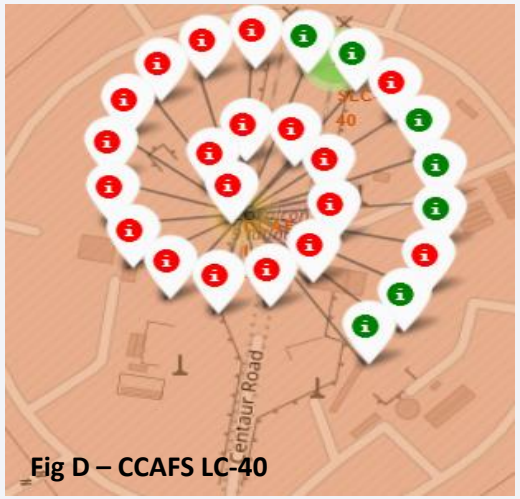


Fig D – CCAFS LC-40

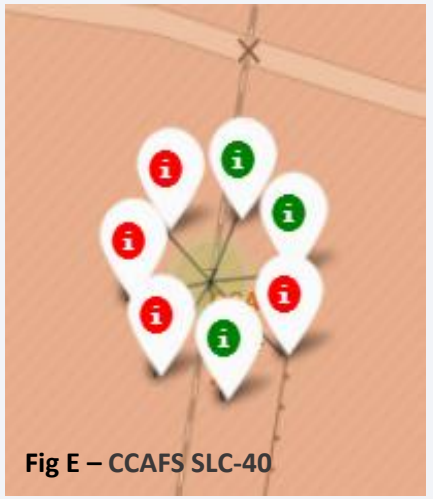
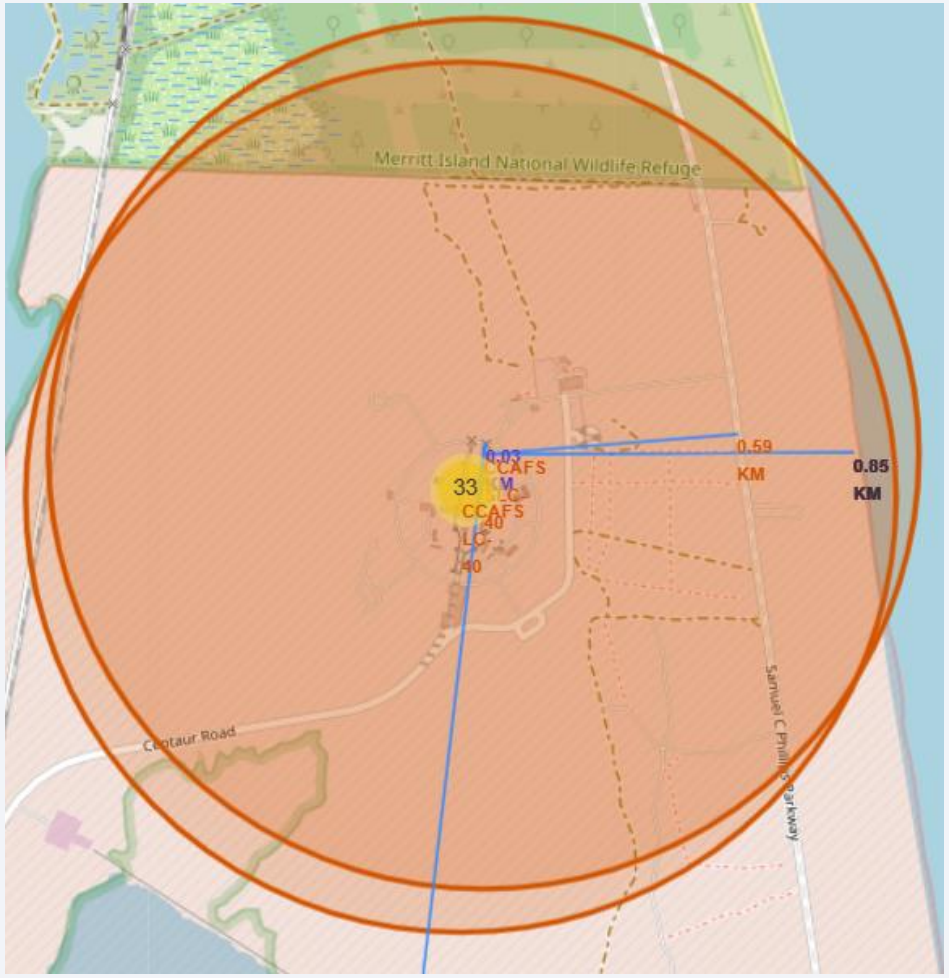


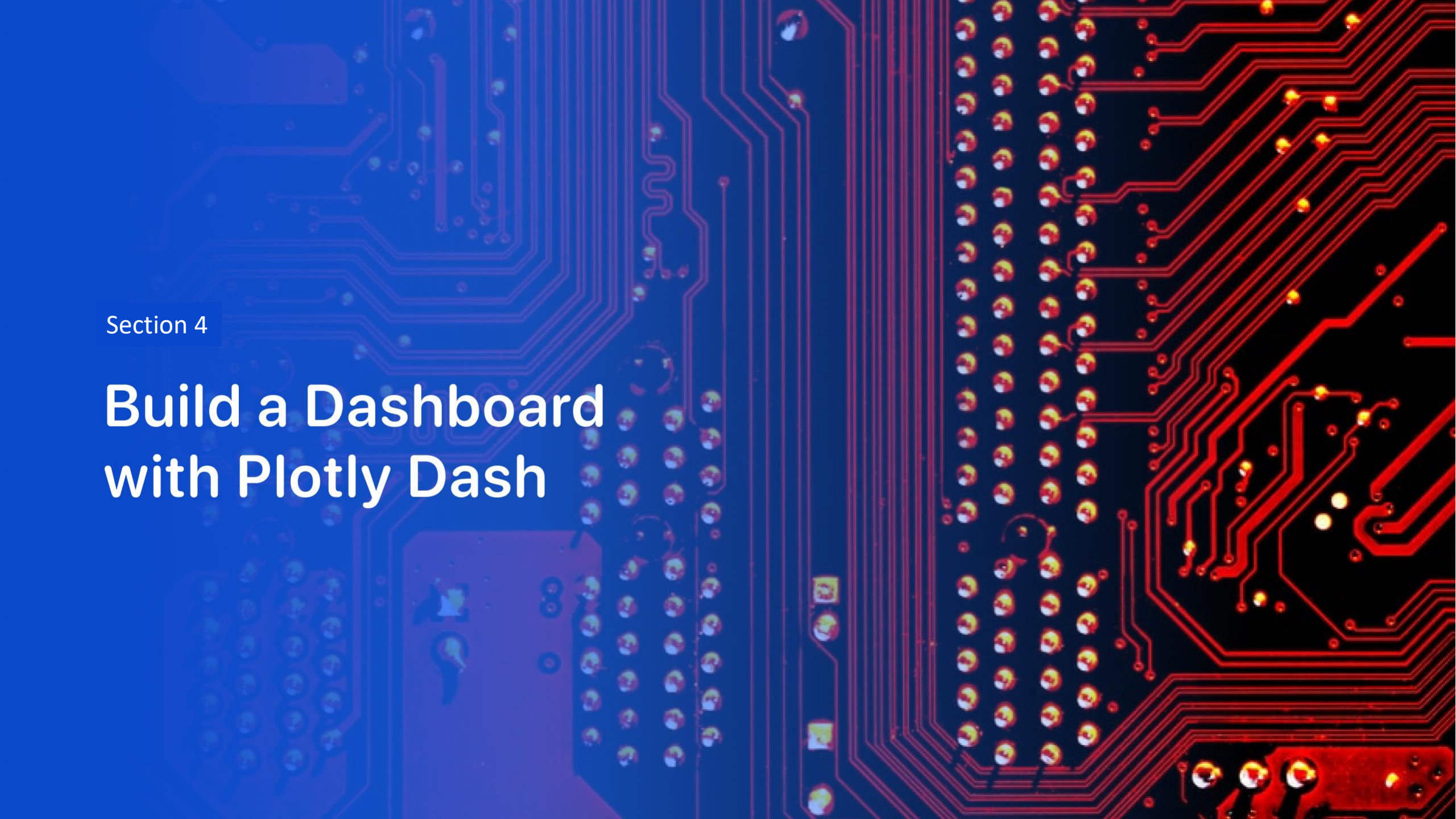
Fig E – CCAFS SLC-40

# SpaceX Falcon9 - Launch Site to proximity Distance Map

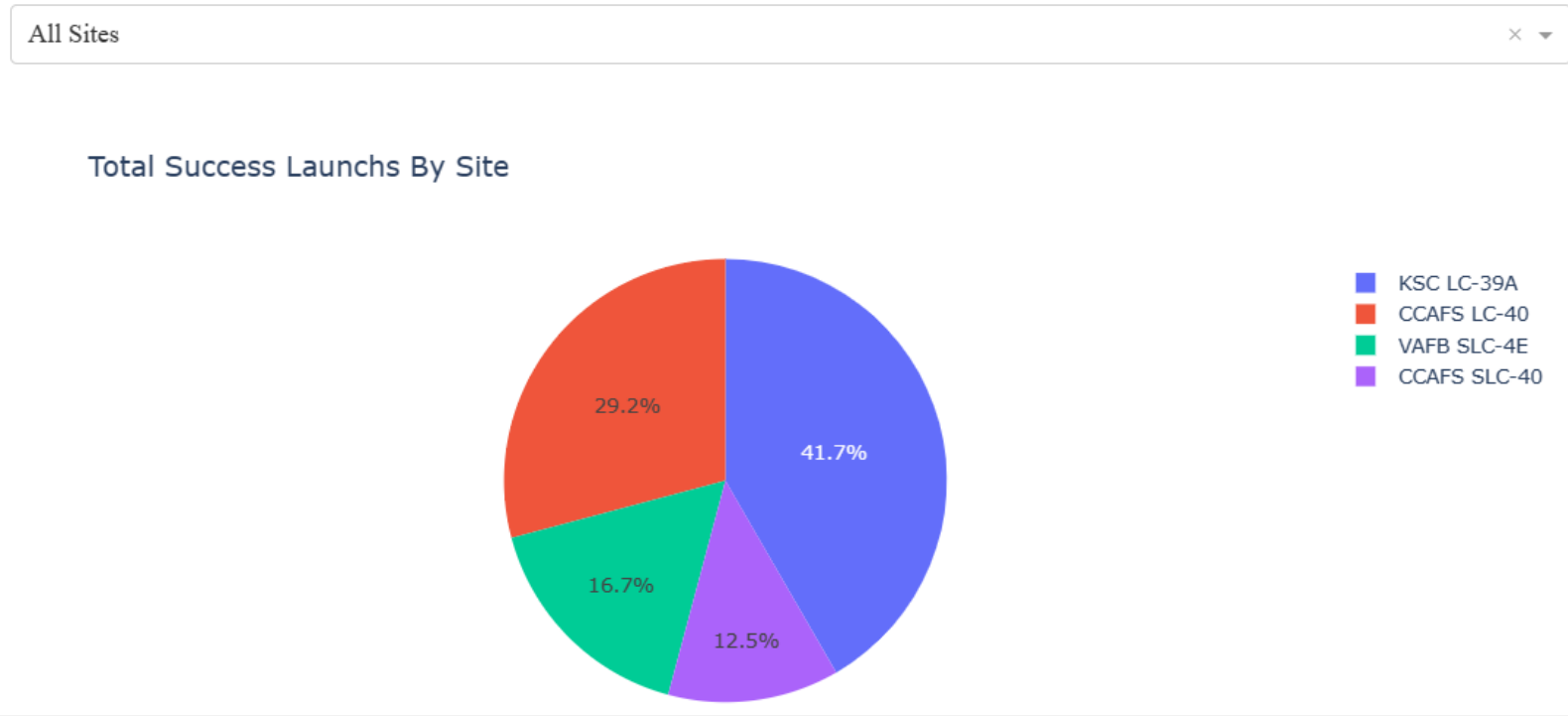
The figure displays all the proximity sites marked on the map for Launch Site CCAFS SLC-40. The city of Melbourne cannot be view in the figure as it is 51km from the site compared to coastline, railroad and highway.

In general, cities are located away from the Launch Sites to minimize impacts of any accidental impacts to the general public and infrastructure. Launch Sites are strategically located near the coastline, railroad, and highways to provide easy access to resources.



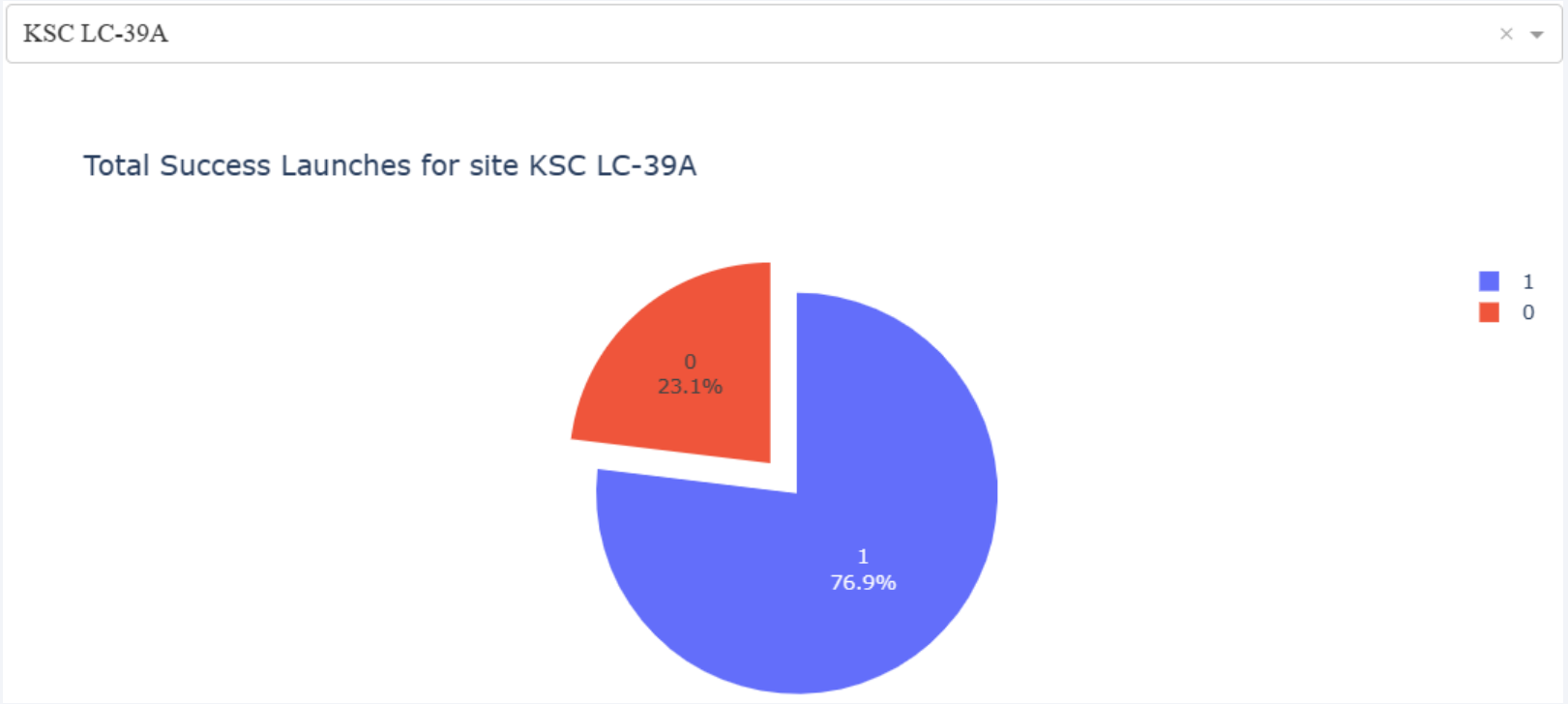

Section 4
Build a Dashboard with Plotly Dash

# Launch Success Counts For All Sites



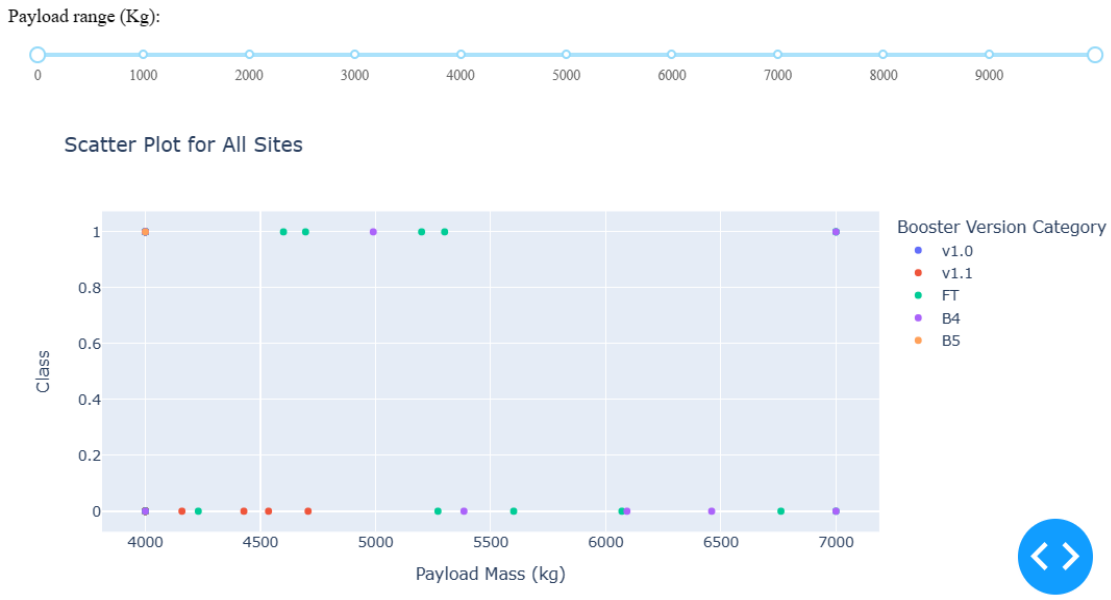

- Launch Site ‘KSC LC-39A’ has the highest launch success rate
- Launch Site ‘CCAFS SLC-40’ has the lowest launch success rate

# Launch Site with Highest Launch Success Ratio




- KSC LC-39A Launch Site has the highest launch success rate and count
- Launch success rate is 76.9%
- Launch success failure rate is 23.1%

# Payload vs. Launch Outcome Scatter Plot for All Sites



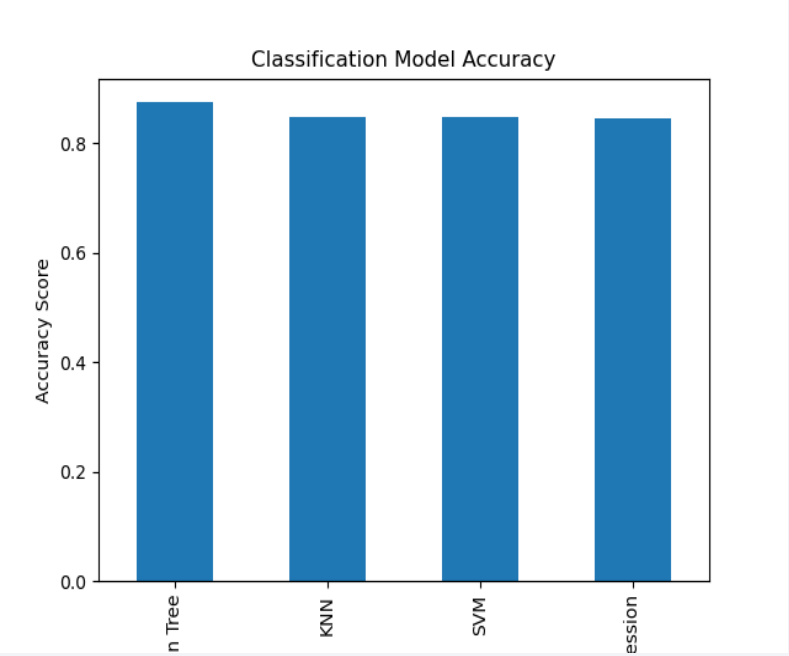


- Most successful launches are in the payload range from 2000 to about 6000
- Booster version category ‘FT’ has the most successful launches
- Only booster with a success launch when payload is greater than 6,000 is ‘B4’

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy




| | Classification Model | Accuracy Score | Test Data Accuracy Score |
|---|---|---|---|
| 2 | Decision Tree | 0.875000 | 0.944444 |
| 3 | KNN | 0.848214 | 0.833333 |
| 1 | SVM | 0.848214 | 0.833333 |
| 0 | Logistic Regression | 0.846429 | 0.833333 |

- Based on the Accuracy scores and as also evident from the bar chart, Decision Tree algorithm has the highest classification score with a value of .8750
- Accuracy Score on the test data however appears higher for this model but is the same for all other classification algorithms with a value of .8333
- Given that the Accuracy scores for Classification algorithms are very close and the test scores are the same, we may need a broader dataset to further tune the models

# Confusion Matrix

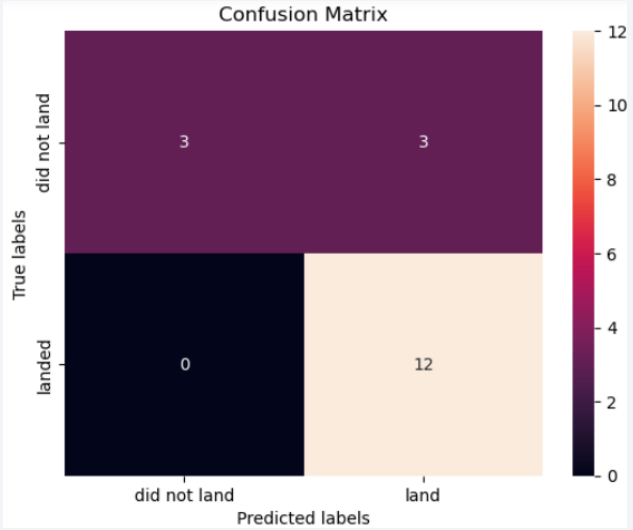



Fig A: Logistic Regression

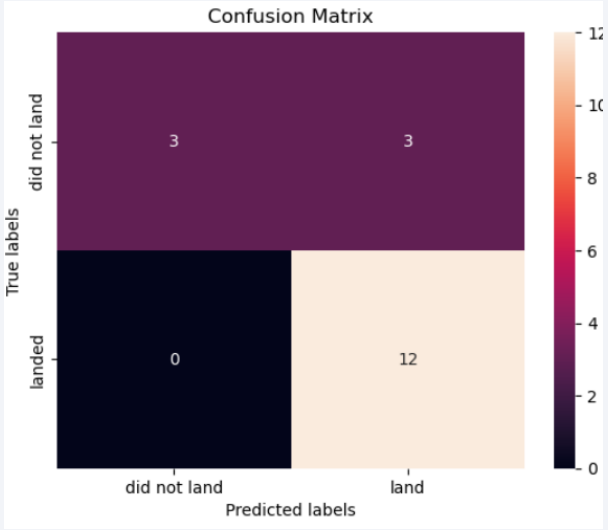



Fig B: SVC

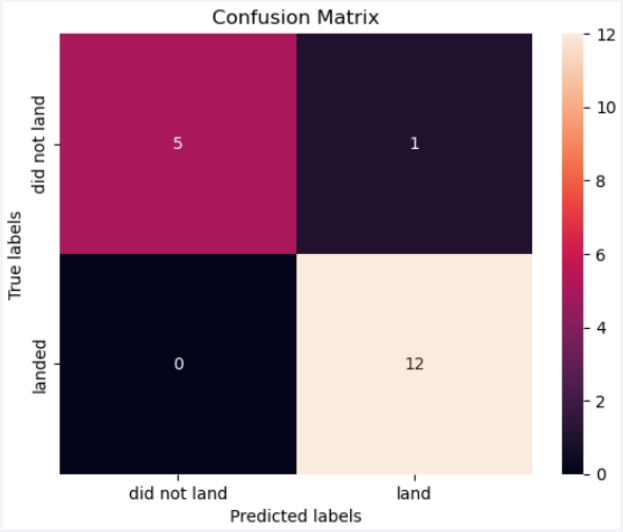



Fig C: Decision Tree

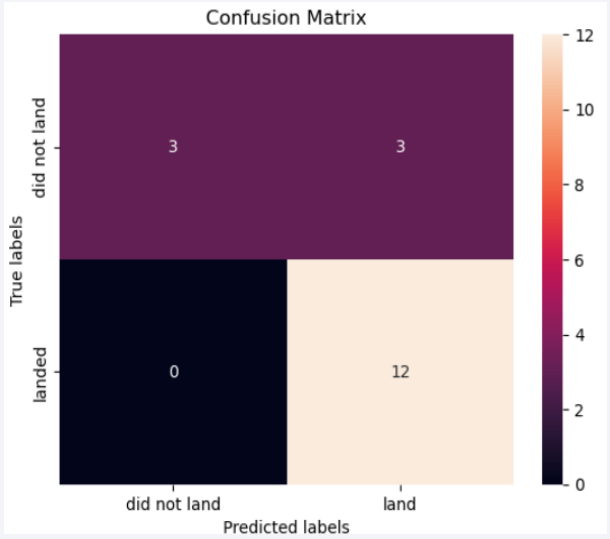



Fig D: KNN

- The confusion matrix are nearly all the same for all the models.
- As per the confusion matrix, the classifier made 18 predictions
- 12 scenarios were predicted Yes for landing and actually landed successfully (True positive)
- 3 scenarios (top left) were predicted No for landing and actually did not land successfully (True negative); exception is the decision tree model which predicted 5 correctly.
- 3 scenarios (top right) were predicted Yes for landing and actually did not land successfully (False positive); exception is the decision tree model which predicted 1 incorrectly.
- Overall, the classifier is correct about 83% of the time ((TP + TN) / Total) with a
- misclassification or error rate ((FP + FN) / Total) of about 16.5%. However, for the decision tree model it is 94% and 6%.

# Conclusions

- As the numbers of flights increase, the first stage is more likely to land successfully
- Success rates appear go up as Payload increases but there is no clear correlation between Payload mass and success rates
- Launch success rate increased by about 80% from 2013 to 2020
- Launch Site ‘KSC LC-39A’ has the highest launch success rate and Launch Site ‘CCAFS SLC-40’ has the lowest launch success rate
- Orbits ES-L1, GEO, HEO, and SSO have the highest launch success rates and orbit GTO the lowest
- Launch sites are located strategically away from the cities and closer to coastline, railroads, and highways
- The best performing Machine Learning Classification Model is the Decision Tree with an accuracy of about 87.5%.
- When the models were scored on the test data, the accuracy score was about 94% for the Decision Tree model but 83% for all other models. More data may be needed to further tune the models and find a potential better fit.

# Thank you!