Winning Space Race with Data Science

Name : JAIDHEV ANANDHEV
Date : 13-09-2023

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context

- Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  - Explain the factors that determine if the rocket will land successfully?
  - The interaction among various features that determine the success rate of a successful landing.
  - Conditions needed to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data is collected from SpaceX REST API and Wikipedia pages.
- Perform data wrangling
  - Applying One hot encoding to the column “Outcome” and convert the data into numerical representation.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

1. We will provide a GET request using requests library from the SpaceX REST API.
2. The result called can be viewed by calling the .json() method.
3. We call .json_normalize() to convert the JSON file to a data frame file.
4. We clean the data, fill the missing values with the mean values.
5. Perform web scraping from a Wiki page that contain Falcon9 records which is possible using Python BeautifulSoup package.
6. Finally, we convert the data to Pandas Dataframe for data visualization and further analysis.

Data Collection – SpaceX API

- We made a GET request from SpaceX API and normalized the JSON to Dataframe file for EDA and further analysis.

- **GITHUB LINK:**
- https://github.com/JaidhevAnandhev/Data_Science_Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

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:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_
```

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

In [11]:
```
# request the SpaceX launch data
res = requests.get(static_json_url)
print(res.content)
```

...

In [13]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
df = res.json()
```

In [14]:
```
# apply json_normalize
data = pd.json_normalize(df)
```

Using the dataframe `data` print the first 5 rows

In [15]:
```
# Get the head of the dataframe
df_data = pd.DataFrame(data)
data.head(5)
```

Out[15]:

static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	p

Data Collection - Scraping

- We perform web scraping to collect Falcon 9 historical launch records from a Wikipedia page
- Parse the table and convert it into a Pandas data frame
- BeautifulSoup is used to parse the HTML files as it is created here.

GITHUB LINK:

- https://github.com/JaidhevAnandhev/Data_Science_Capstone/blob/main/jupyter-labs-webscraping.ipynb

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.

In [5]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [6]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
resp = requests.get(static_url)
```

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

In [7]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(resp.text, "html.parser")
soup.title
```

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

In [10]:
```
# 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')
```

Starting from the third table is our target table contains the actual launch records.

In [11]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

Data Wrangling

1. Here we perform Exploratory Data Analysis and determine Training Labels. Make data accessible and easy to use.
2. At first, We calculated the number of launches on each site using value_counts() function.
3. Calculate the number and occurrence of each orbit using value_counts() function.
4. Create a landing outcome label from Outcome column using Onehot encoding.

GIT-HUB LINK:

- https://github.com/JaidhevAnandhev/Data_Science_Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

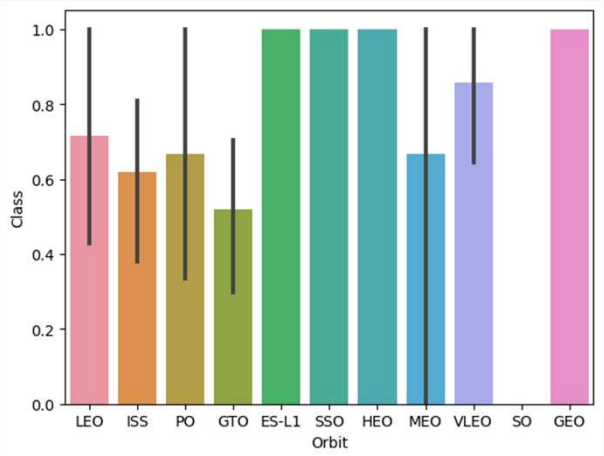
TASK 4: Create a landing outcome label

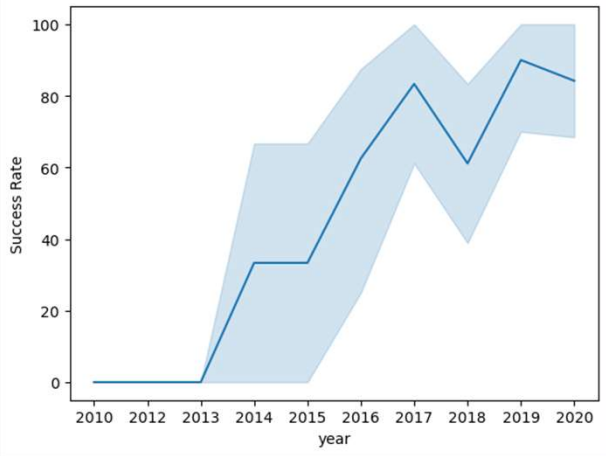
Using the `Outcome`, create a list where the element is it to the variable `landing_class`:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
def onehot(item):
    if item in bad_outcomes:
        return 0
    else:
        return 1
landing_class = df["Outcome"].apply(onehot)
landing_class
```

```
0     0
1     0
2     0
3     0
4     0
     ..
85    1
86    1
87    1
88    1
89    1
```

EDA with Data Visualization





1. Here we visualize the data using the popular python packages like NumPy, Pandas and Seaborn.
2. They help to provide a user-friendly data visualization to the users so that the data scientist can decide what to do with the data.
3. Using the available functions, we can easily check for the information of the whole database easily

GITHUB LINK:

- https://github.com/JaidhevAnandhev/Data_Science_Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

1. Connect to the database and load them into the Jupyter notebook.

2. To get some of the insights of the data:
   1. names of the unique launch sites in space mission.
   2. Display 5 records where launch sites begin with the string 'CCA'
   3. Display the total payload mass carried by boosters launched by NASA (CRS)
   4. Display average payload mass carried by booster version F9 v1.1
   5. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
   6. List the total number of successful and failure mission outcomes.

3. **GITHUB LINK:**

https://github.com/JaidhevAnandhev/Data_Science_Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines in map and mark the success or failure of launches for each site on the folium map.
- We assigned marker colors for each value of class column i.e.; 0 or 1 0 for failure and 1 for success.
- We calculated the distances between a launch site to its coastline point. We answered some question for instance:
  - Are launch sites near railways?
  - Are launch sites near highways?
  - Are launch sites near coastline?
  - Do launch sites keep certain distance away from cities?

- GITHUB LINK :
  https://github.com/JaidhevAnandhev/Data_Science_Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- An interactive dashboard is created by using Plotly Dash.
- It displays the launch sites details using a pie chart and Payload range by creating a slider for finding values for a particular range
- The payload range is shown in a scatter plot for different booster versions.
- I have added them for visualizing the data in a user-friendly manner and so data scientists can easily visualize and take necessary decisions.
- GITHUB LINK:
- https://github.com/JaidhevAnandhev/Data_Science_Capstone/blob/main/spacex_dash_app.py

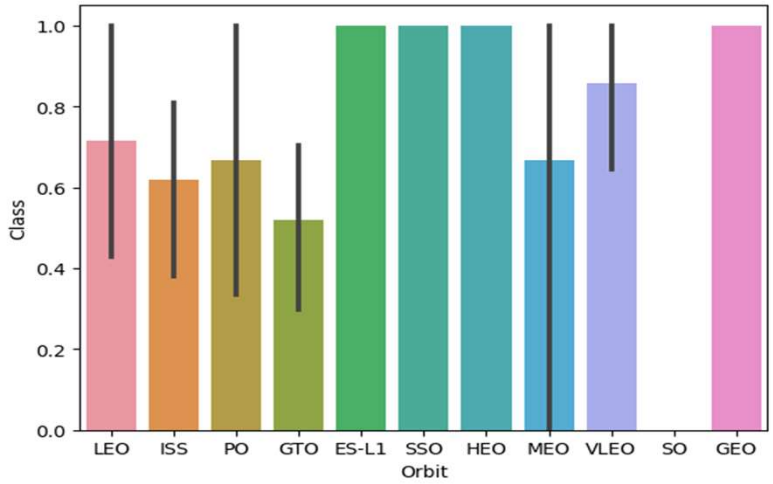
Predictive Analysis (Classification)

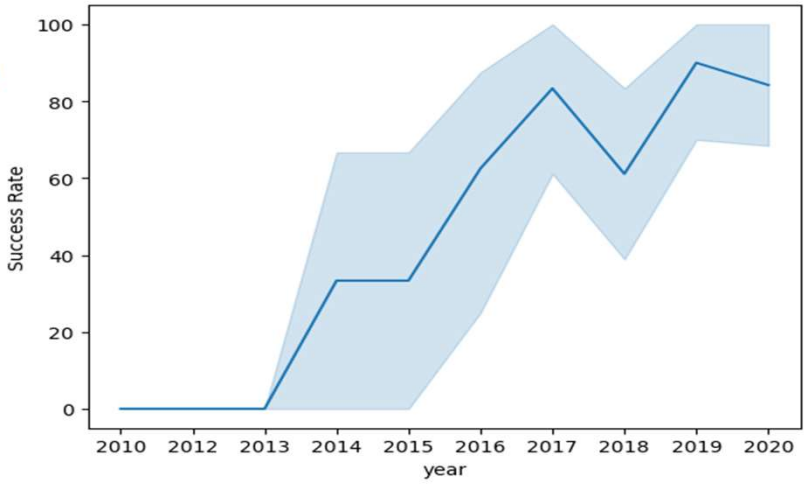
- For finding the best classification model for evaluation of our data NumPy, Pandas, Seaborn, Matplotlib are being used.
- At first, we split the data as Training set and Test Set and tuning the data parameters using GridSearchCV( ) for logistic regression, SVM, Decision trees and KNN.
- Next, we test the accuracy of the data using SCORE method and create a confusion matrix to distinguish between the different classes.
- Thus, we found the best performing classification model.
- GITHUB LINK:
- http://localhost:8888/notebooks/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

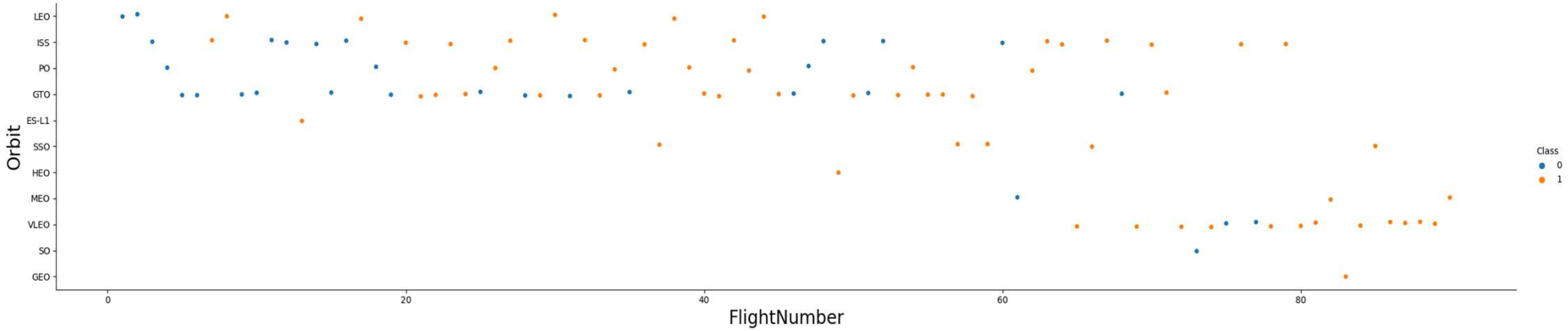
Results

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

Results


Class
Orbit
LEO
ISS
PO
GTO
ES-L1
SSO
HEO
MEO
VLEO
SO
GEO


Success Rate
year

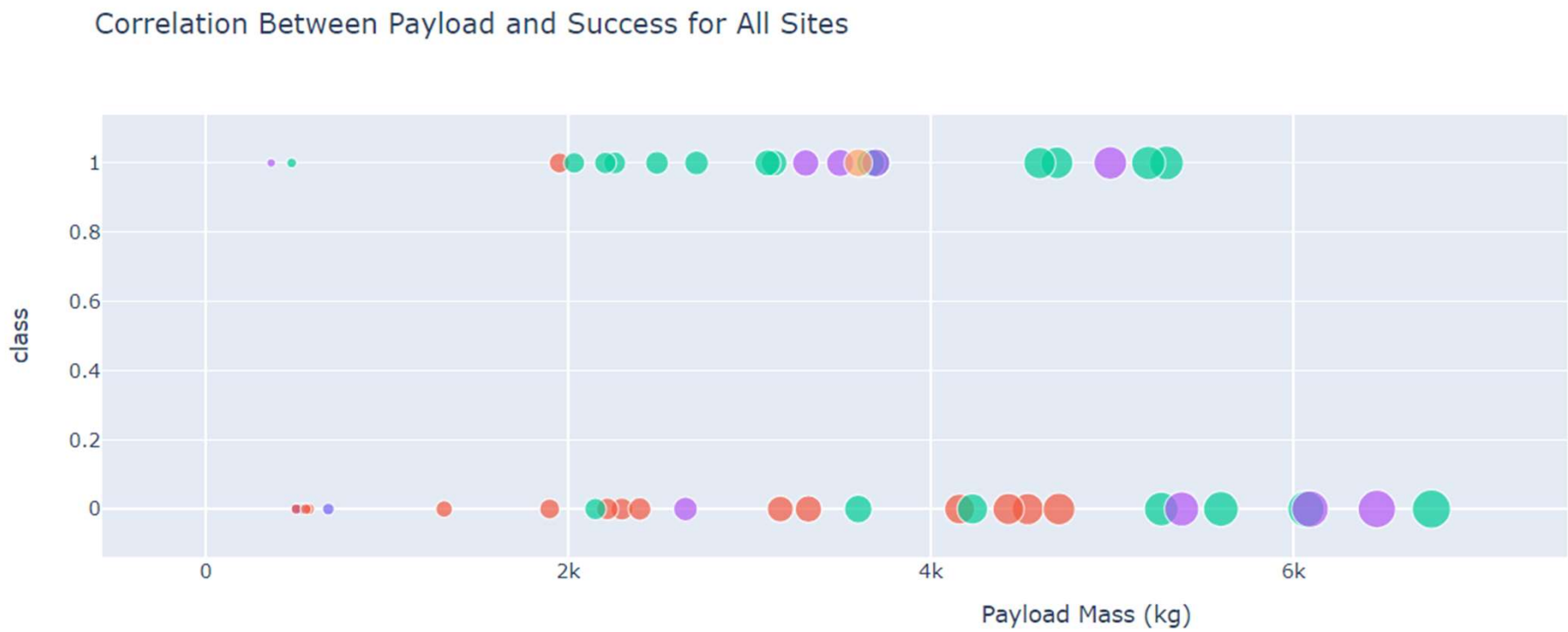

Orbit
FlightNumber
Class

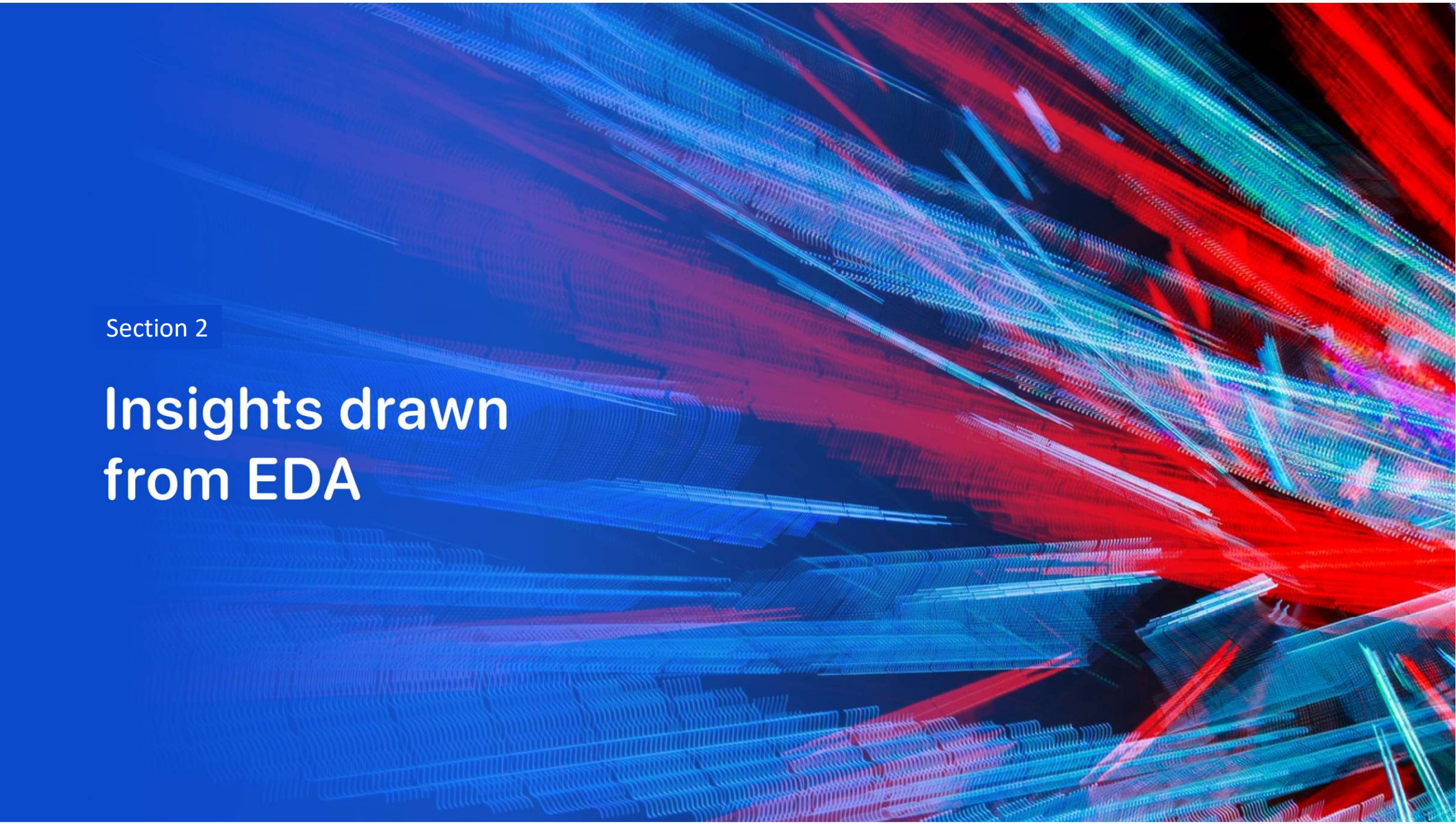
SpaceX Launch Records Dashboard

ALL SITES

Total Launches for All Sites

Payload range (Kg):

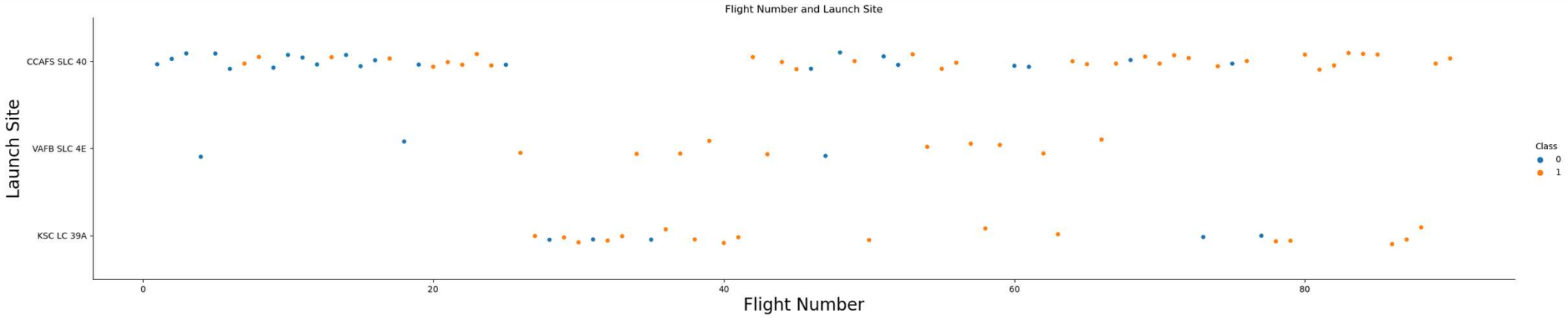




Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

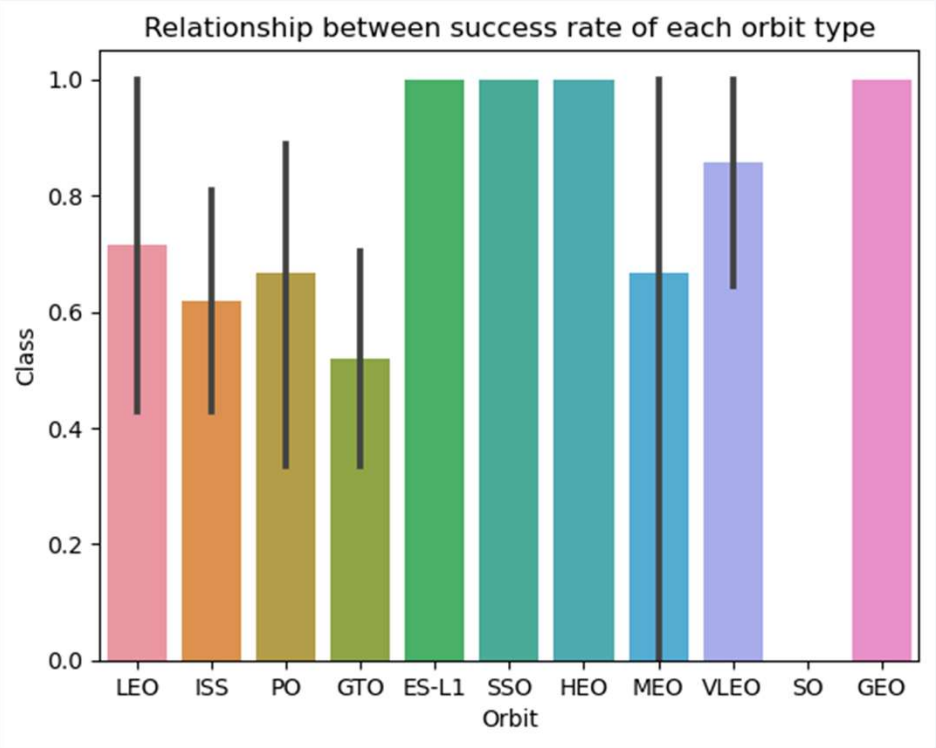


- **EXPLANATION :**
- Here we we use the function catplot( ) to plot FlightNumber vs LaunchSite, set the parameter **x parameter** to FlightNumber, set the **y parameter** to Launch Site and set the parameter **hue** to 'class'
- Based on the column 'class', values the color of the marker will be updated automatically.
- Lastly, we use xlabel, ylabel and title functions to provide an understanding to the user.

Payload vs. Launch Site

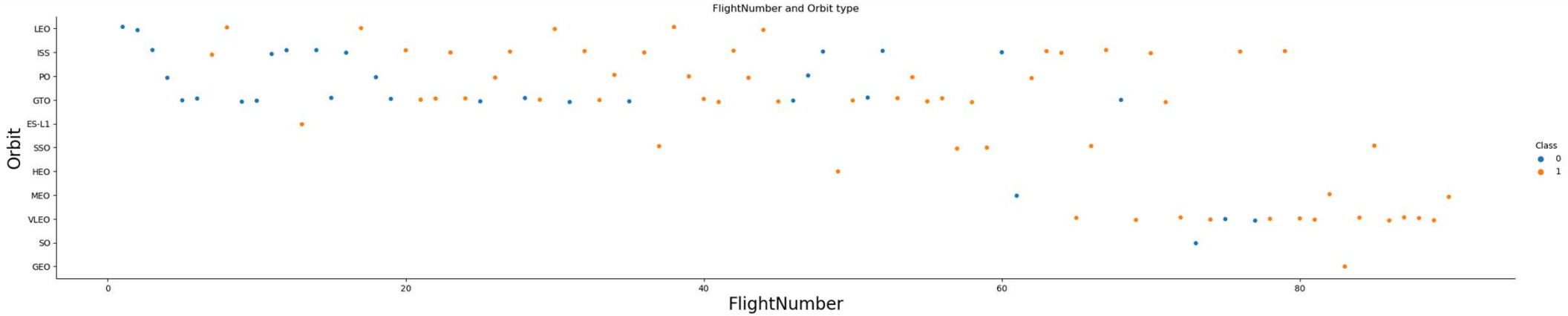
- **EXPLANATION :**
- Here we we use the function catplot( ) to plot LaunchSite vs PayloadMass, set the parameter **x parameter** to LaunchSite, set the **y parameter** to PayloadMass and set the parameter **hue** to 'class'
- Based on the column 'class', values the color of the marker will be updated automatically.
- Lastly, we use xlabel, ylabel and title functions to provide an understanding to the user.

Success Rate vs. Orbit Type



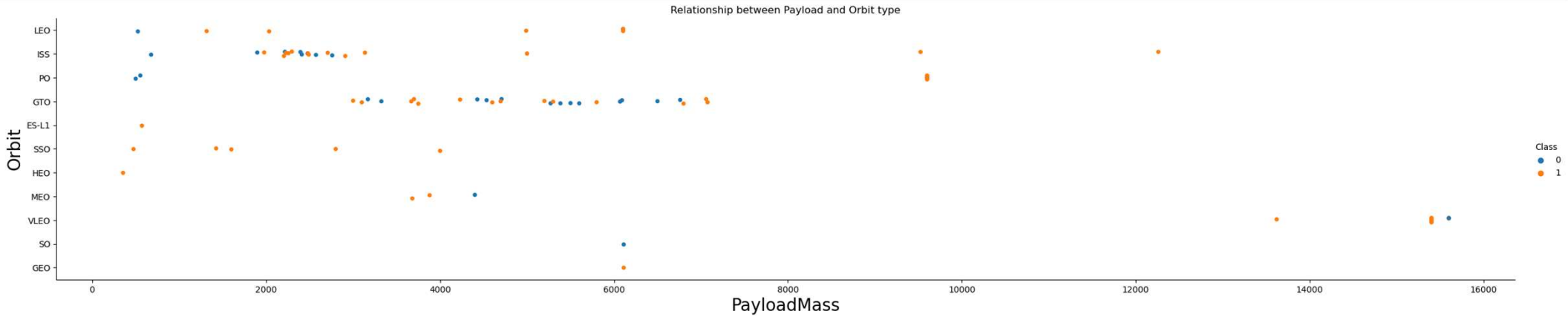
- **EXPLANATION :**
- Here we use the function barplot( ) from seaborn library to plot Orbit vs Class, set the parameter **x parameter** to Orbit, set the **y parameter** to Class.
- From the graph we can get that ES-L1, SSO, HEO and GEO have same class value of 1 and others with relatively less values.
- Lastly, we use xlabel, ylabel and title functions to provide an understanding to the user.

Flight Number vs. Orbit Type



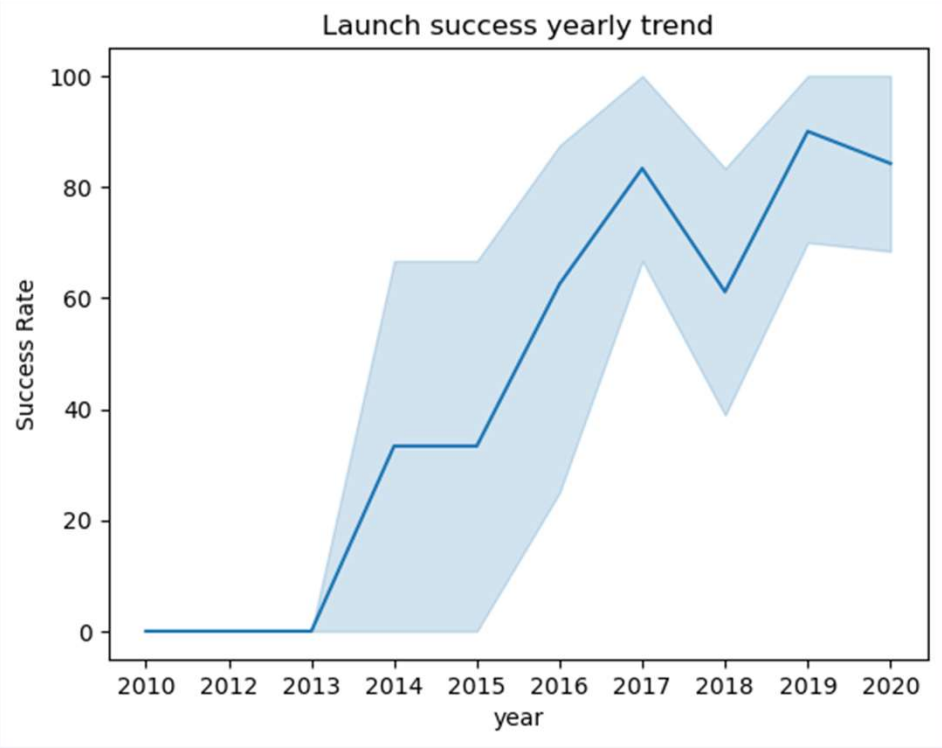
- **EXPLANATION :**
- Here we we use the function catplot( ) to plot FlightNumber vs Orbit, set the parameter **x parameter** to FlightNumber, set the **y parameter** to Orbit and set the parameter **hue** to 'class'
- Based on the column 'class', values the color of the marker will be updated automatically.
- Lastly, we use xlabel, ylabel and title functions to provide an understanding to the user.

Payload vs. Orbit Type



- **EXPLANATION :**
- Here we we use the function catplot( ) to plot PayloadMass vs Orbit, set the parameter **x parameter** to PayloadMass, set the **y parameter** to Orbit and set the parameter **hue** to 'class'
- Based on the column 'class', values the color of the marker will be updated automatically.
- Lastly, we use xlabel, ylabel and title functions to provide an understanding to the user.

Launch Success Yearly Trend



- **EXPLANATION :**
- Here we use the function lineplot ( ) from seaborn library to plot launch success yearly trend, set the parameter **x parameter** to year, set the **y parameter** to Success Rate.
- From the graph we can get that there is a rise in success rate for the rise in the year.
- Lastly, we use xlabel, ylabel and title functions to provide an understanding to the user.

All Launch Site Names

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

```
In [8]: %sql select DISTINCT LAUNCH_SITE from SPACEXDATASET

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

Out[8]:

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

- The SQL statement is used to display the launch site names
- It is observed that the Keyword DISTINCT is used the eliminate the repeated words of the launch site
- Thus, the unique values are displayed into the dataframe.

Launch Site Names Begin with 'CCA'

Display 5 records where launch sites begin with the string 'CCA'

In [9]:
```
%sql select * from SPACEXDATASET where launch_site like 'CCA%' limit 5
```

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

Out[9]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Here we use CCA% to search the words starting with CCA in Launch_site.
- SQL statement is used to extract the names that begin with CCA.

Total Payload Mass

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

```
In [10]: %sql select sum(payload_mass__kg_) as sum from SPACEXDATASET where customer like 'NASA (CRS)'
 * sqlite:///my_data1.db
Done.
```

Out[10]:

sum
45596

- **EXPLANATION :**
- Here in the SQL statement, we need to check for the total payload mass by using SUM( ) function and naming the column name as sum where the customers are separated based on the value 'NASA (CRS)'

Average Payload Mass by F9 v1.1

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

```
In [11]: %sql select avg(payload_mass__kg_) as Average from SPACEXDATASET where booster_version like 'F9 v1.1%'
 * sqlite:///my_data1.db
Done.
```

Out[11]:

Average
2534.6666666666665

- <u>EXPLANATION</u>:
- Here in the SQL statement, we are said to calculate the average of PayloadMass and name the column as Average and data is selected based on the values in the column booster_version where the values are like 'F9 v1.1% '

First Successful Ground Landing Date

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

Hint:Use min function

In [12]:
```
%sql select min(date) as Date from SPACEXDATASET where mission_outcome like 'Success'
```

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

Out[12]:

Date
2010-04-06

- From the output we can get that the first successful landing date on the ground pad was 06-04-2010.
- Here the data is obtained based on the value ‘Success’ in the mission_outcome
- Also, the resultant column is named as Date and using min( ) function to obtain the first landing date.

Successful Drone Ship Landing with Payload between 4000 and 6000

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

In [13]:
```
%%sql select booster_version from SPACEXDATASET where (mission_outcome like 'Success')
AND (payload_mass__kg_ BETWEEN 4000 AND 6000) AND (Landing_Outcome like 'Success (drone ship)')
```

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

Out[13]:

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

- Here the data of Booster_Version with successful Drone Ship Landing and with Payload between 4000 and 6000 so AND is used in SQL statement so that only when both the conditions are true the statement will execute the output.

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [14]:
```
%%sql SELECT mission_outcome, count(*) as Count FROM SPACEXDATASET
GROUP by mission_outcome ORDER BY mission_outcome
```

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

Out[14]:

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

- From the dataframe we can understand that the Count of failure and success has been noted and displayed. Here we use GROUP BY and ORDER BY to arrange the data alphabetically and based on the columns specified in the query.

Boosters Carried Maximum Payload

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

```
In [15]: maxm = %sql select max(payload_mass__kg_) from SPACEXDATASET
         maxv = maxm[0][0]
         %%sql select booster_version from SPACEXDATASET
         where PAYLOAD_MASS__KG_= (SELECT MAX(PAYLOAD_MASS__KG_) from SPACEXDATASET)
```

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

- Top 3 names of the booster which have carried the maximum payload mass are:
  - F9 B5 B1048.4
  - F9 B5 B1049.4
  - F9 B5 B1051.3
- Here the MAX( ) function [In WHERE clause] is used to get the maximum value of Payload mass from the Dataset and thus produced the DataFrame.

2015 Launch Records

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [28]:
```
%%sql select (Date) as Month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXDATASET
WHERE DATE like '2015%' AND Landing_Outcome LIKE 'Failure (drone ship)'
```

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

Out[28]:

Month	Landing_Outcome	Booster_Version	Launch_Site
2015-10-01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- Here the failure landing outcomes for the months in the year 2015 and the required columns are displayed.
- The selected column Date is named as Month and selection is based on Landing_outcome column like Failure(drone ship) values.

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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
In [30]: %%sql select Landing_Outcome, count(*) as count from SPACEXDATASET
         where Date >= '2010-06-04' AND Date <= '2017-03-20'
         GROUP by Landing_Outcome ORDER BY count Desc
```

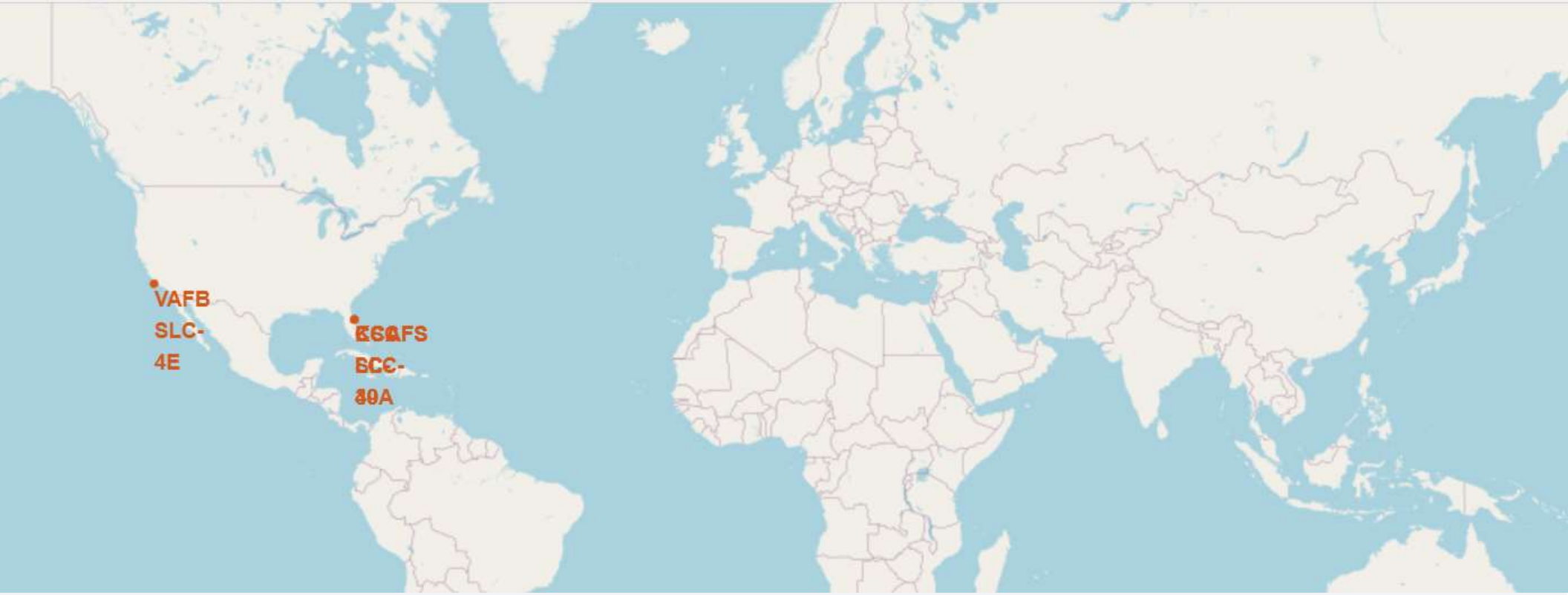
Landing_Outcome	count
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

- Here the Dates are sorted based on the rank of the landing outcomes between 2010-06-04 and 2017-03-20.
- AND keyword is used to sort the date in descending order.
- Data is grouped based on Landing_outcome and ordered by Count in descending order.

Section 3

Launch Sites Proximities Analysis

All launch site's location markers



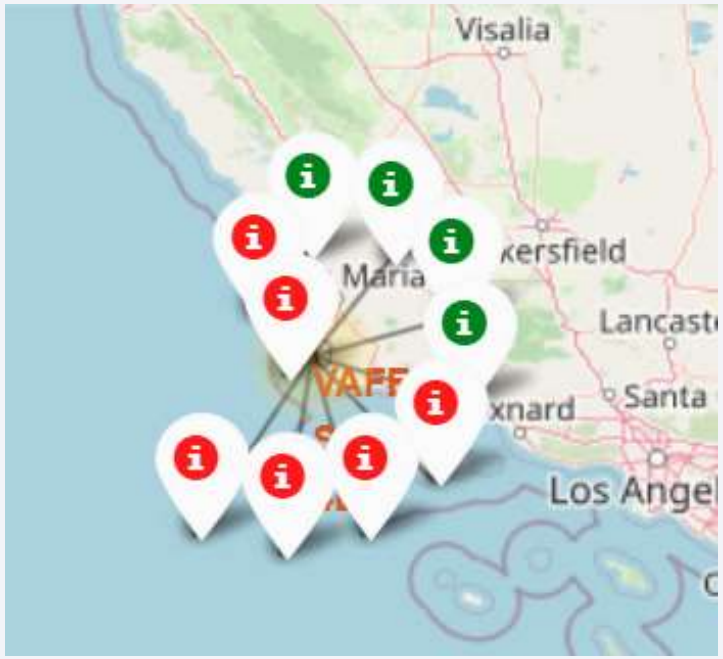
The locations mentioned in the map are

1. CCAFS SLC-40
2. KSC LC-39A
3. VAFB SLC-4E

We can be able to insert the Marker and Circle mark to that particular Latitude and longitude using the Folium for interactive visual analytics.

Markers success/failed launches for each site on the map

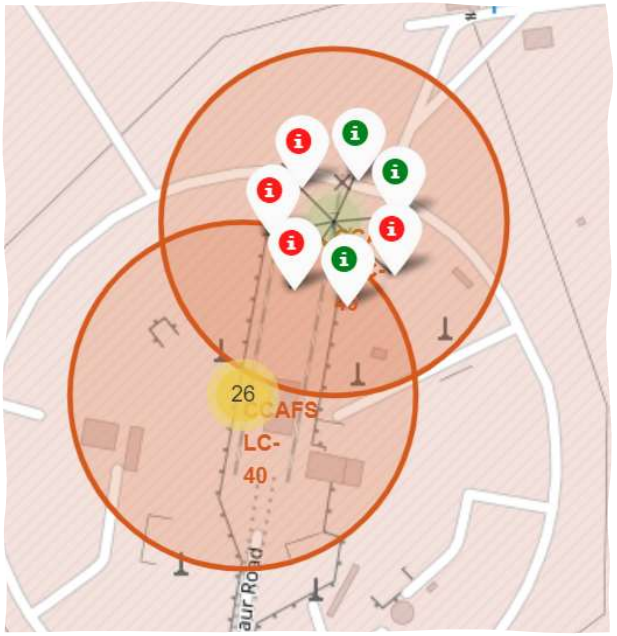
CALIFORNIA LAUNCH SITE

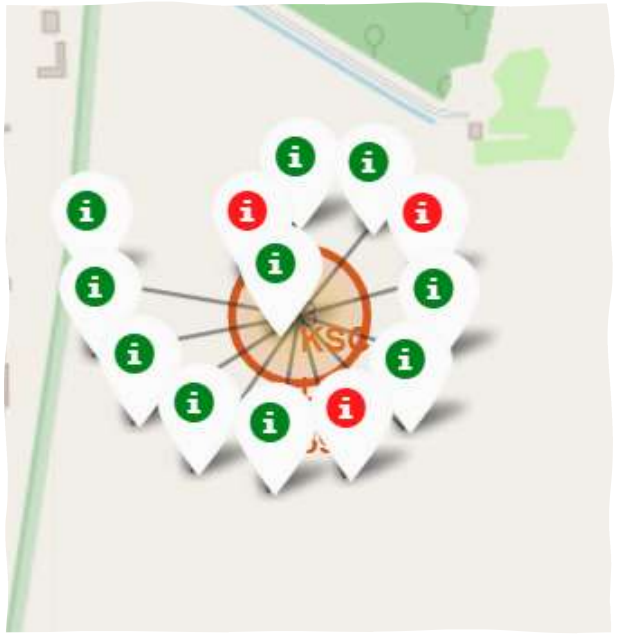


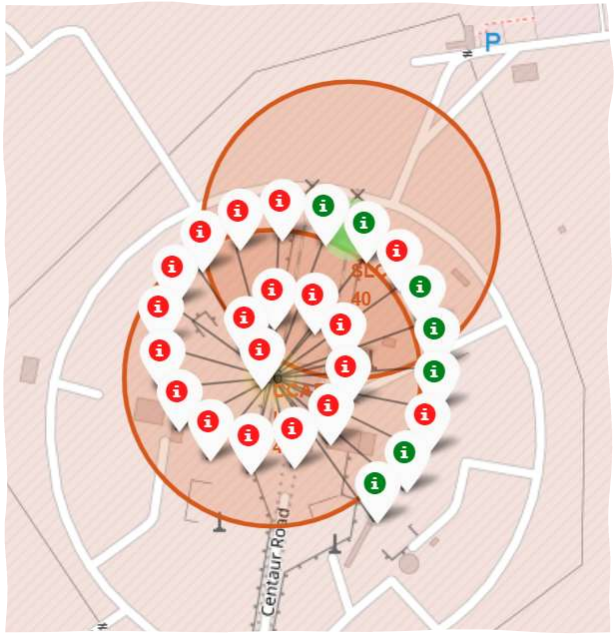
- Here the marker is produced using Folium.marker package and mentioning how to show the data and also showing popup that displays the Launch site in California.
- Here the Green mark shows the successful launches and red mark shows the failure landings.

Florida Launch Sites showing the markers and color labels

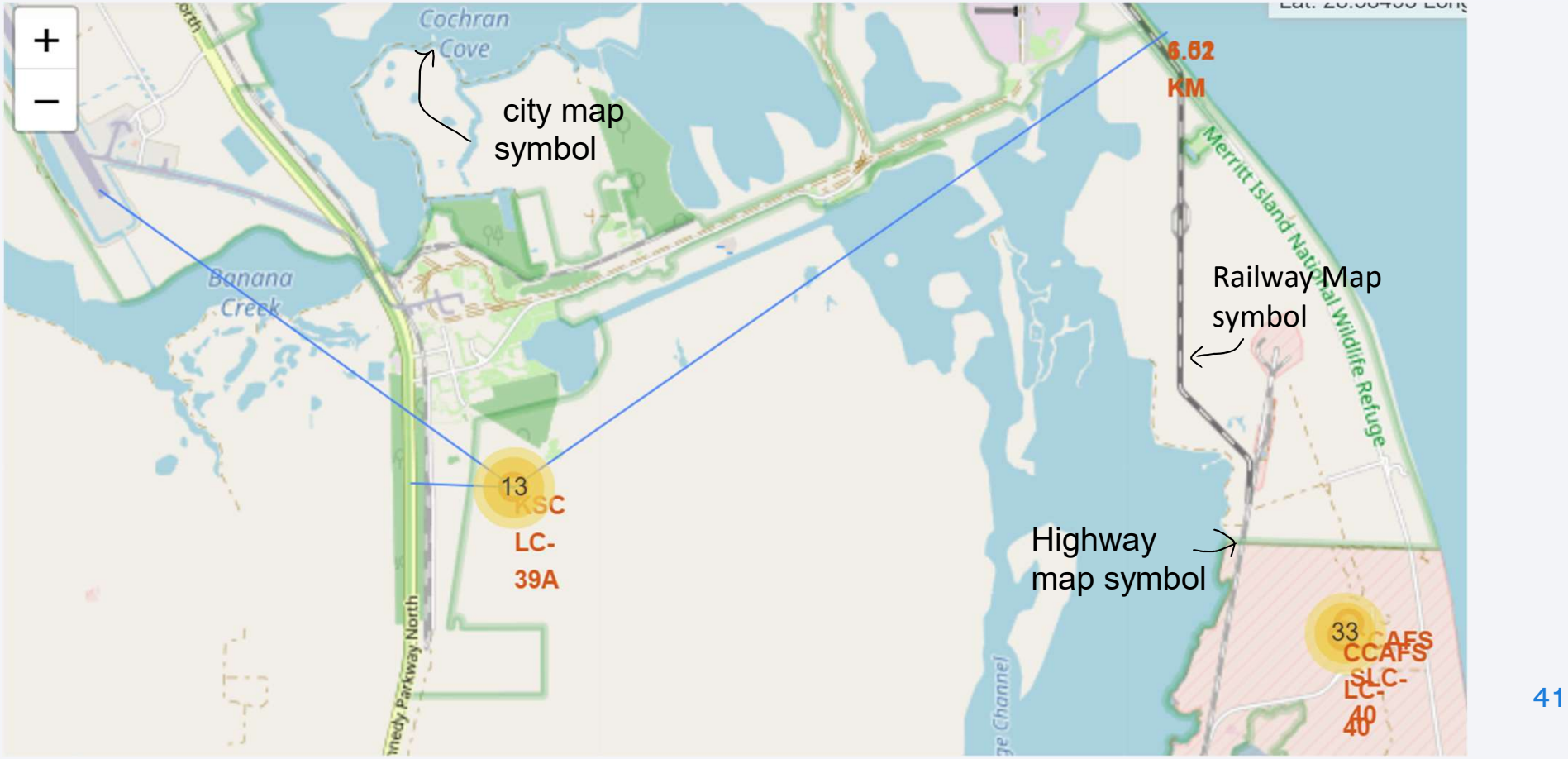
Here Green label shows the successful landing and Red label show the failure landings

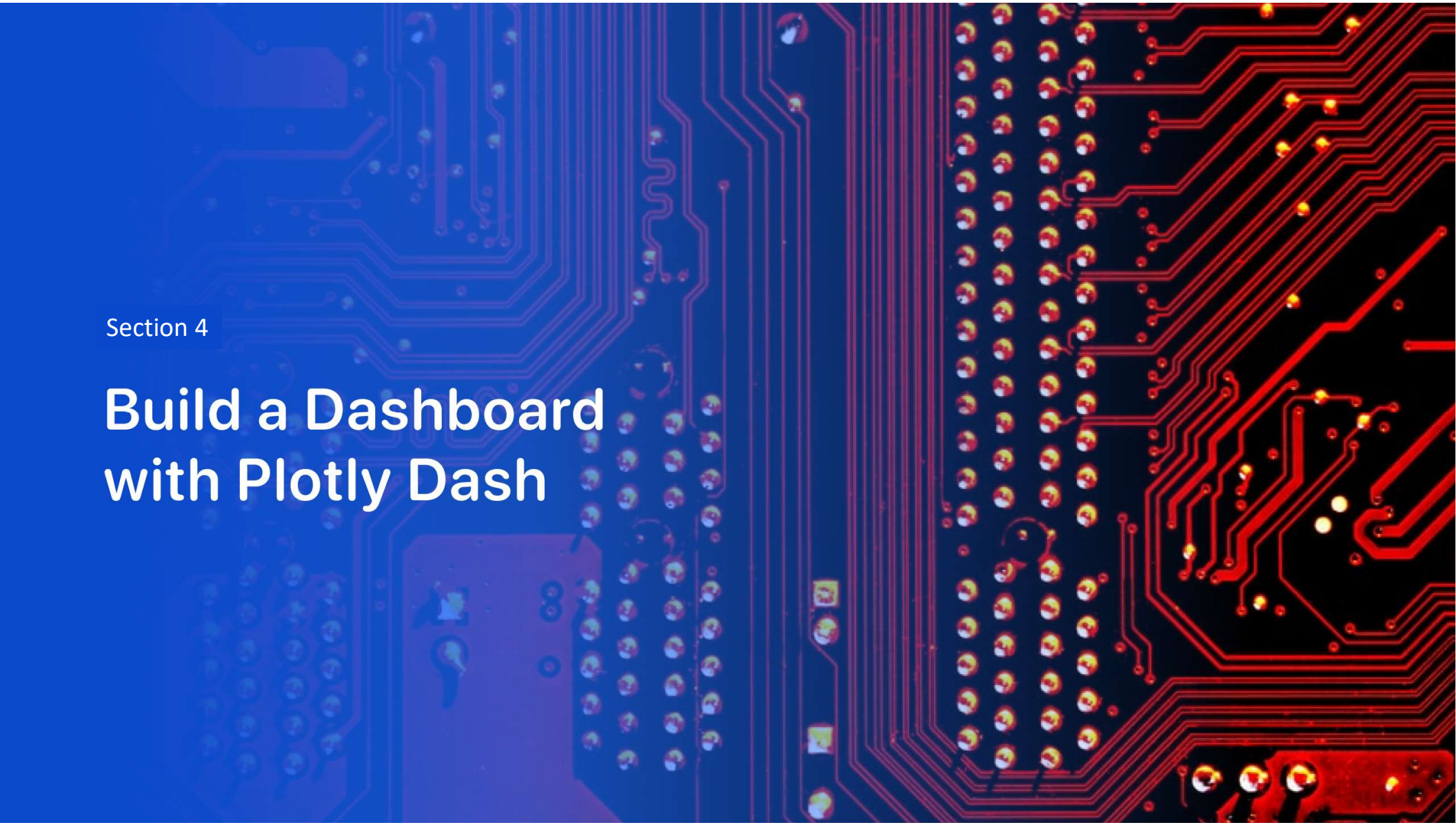






Launch Site Data to landmarks

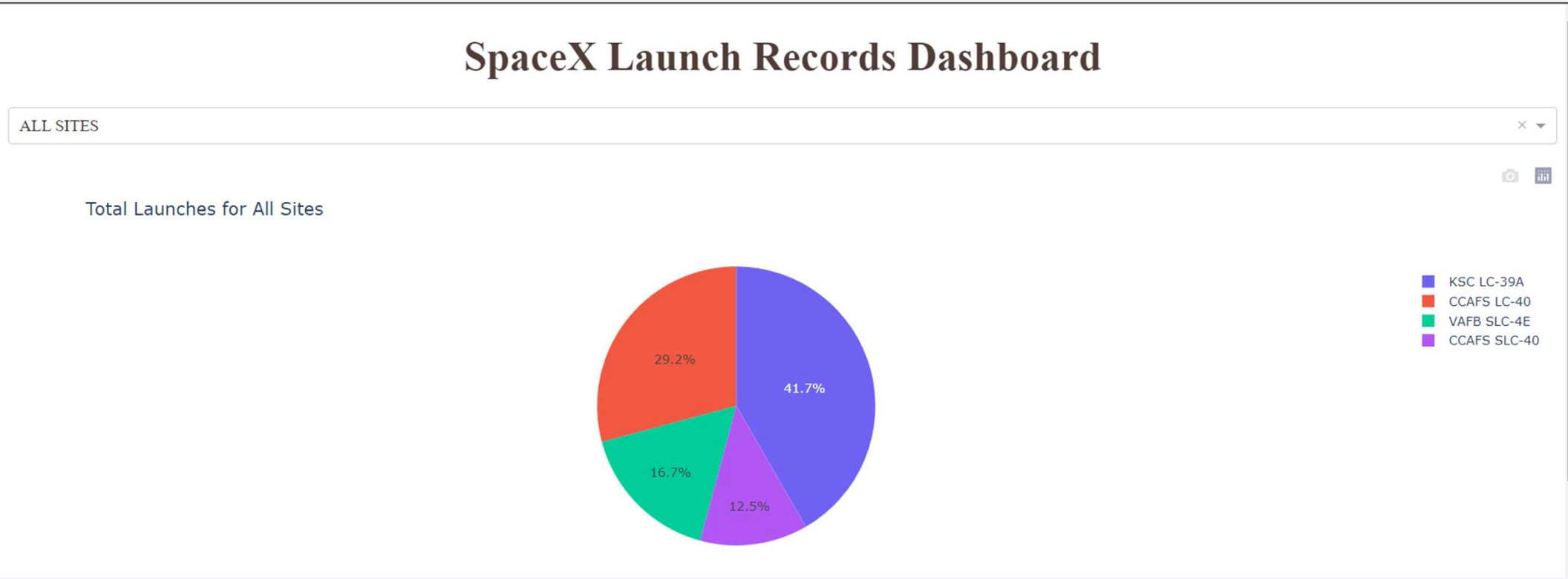




Section 4

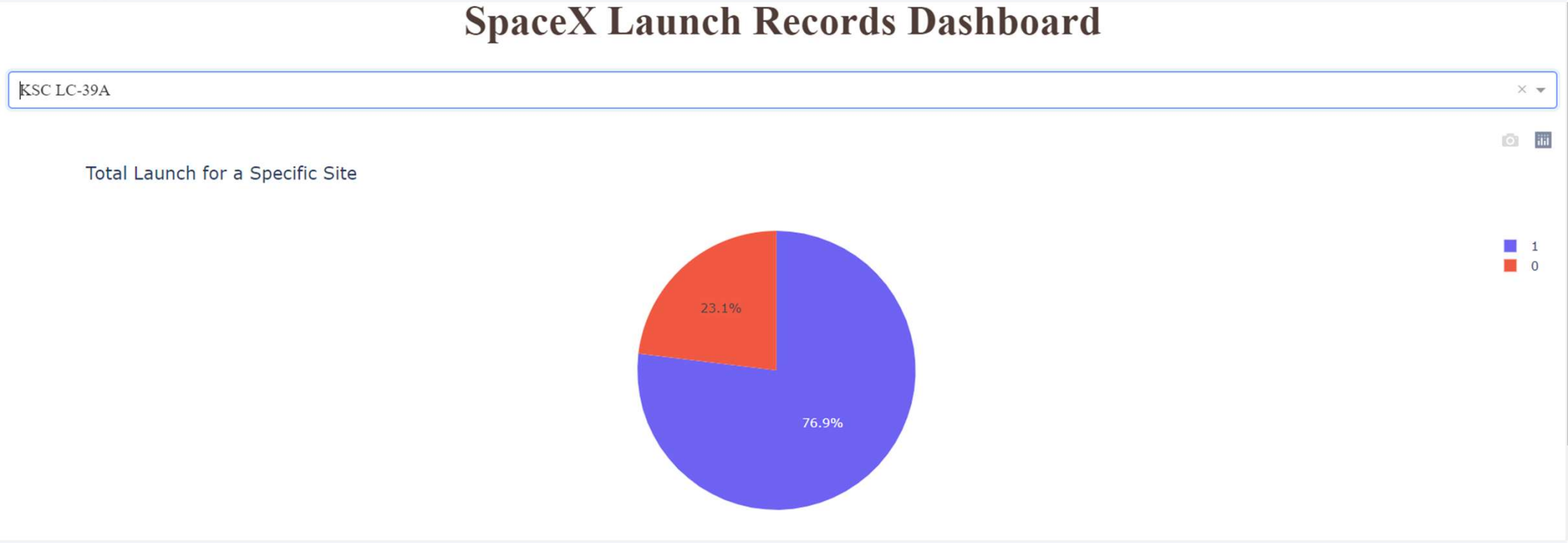
Build a Dashboard with Plotly Dash

Launch Success of All Sites in a Pie Chart



- This pie chart describes about the Launch success of all the sites with the legends present on the right and it
- From the chart we can tell that the KSC LC-39A has the highest success percentage.

Launch site with the highest launch success ratio



- As we know from the previous information KSC LC-39A has the highest launch success ratio we selected that site and checked for the class ratio.

Scatter plot for Payload vs. Launch Outcome

- This is a scatter plot describing about the success rate and payload rate for all the sites categorized based on the Booster Version Category
- Also, a slider is created to describe about the range of payload range.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

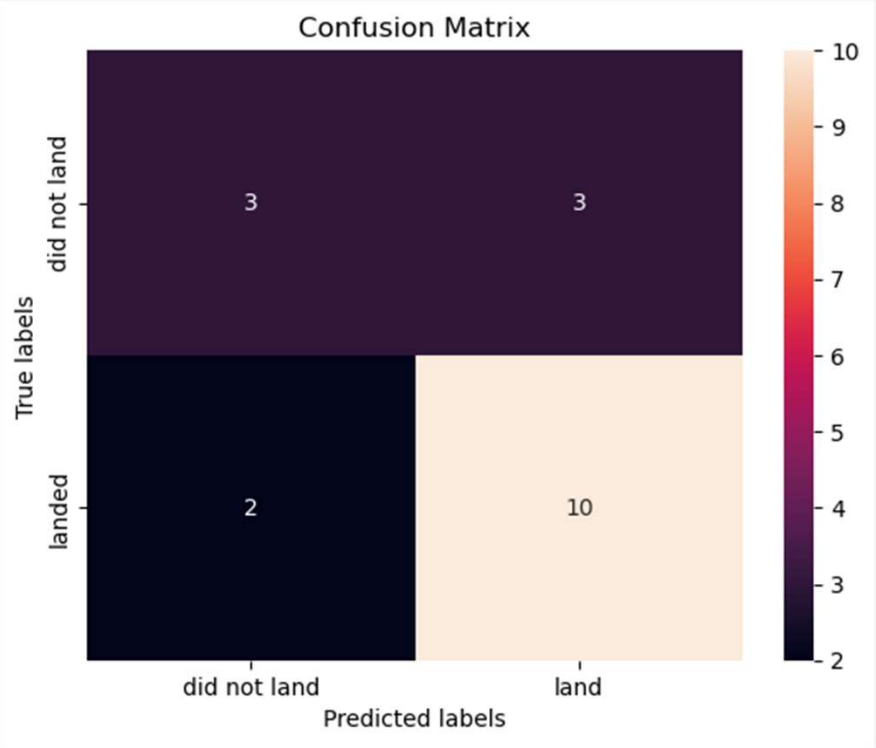
Find the method performs best:

In [39]:
```
models = {'KNeighbors' : knn_cv.best_score_,
'DecisionTree':tree_cv.best_score_,
'LogisticRegression': logreg_cv.best_score_,
'SupportVector': svm_cv.best_score_}
bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm, 'with a score of', models [bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is : ', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
          print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
          print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
          print('Best params is, svm_cv'.best_params_)
```

```
Best model is DecisionTree with a score of 0.875
Best params is :  {'criterion': 'entropy', 'max_depth': 2, 'max_features': 'sqrt', 'min_samples_lea
f': 1, 'min_samples_split': 10, 'splitter': 'best'}
```

Confusion Matrix

- The best performing model with explanation is Decision trees with accuracy score of 0.875
- It helps in distinguishing different types of the classes.
- This matrix compares the predicted and true labels of the landings
- The rate is given from 2 to 10 in form of colors.
- This helps the user to clearly find the ratio between the Predicted and True labels.



Conclusions

- Thus, we can conclude that if the cost of flight is high then the success rate would also be high.
- Launch rate is increased from 2010 till 2020.
- ES-L1, GEO, HEO, SSO, VLEO is the orbit with the highest rate of success.
- KSC LC-39A is the most successful launch area than the other launch sites.
- The decision tree algorithm is the best machine learning algorithm used for our project.

Appendix – Some of the libraries throughout the courses

```
# Pandas is a software library written for the Python p
import pandas as pd
# NumPy is a library for the Python programming languag
import numpy as np
# Matplotlib is a plotting library for python and pyplo
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based o
import seaborn as sns
# Preprocessing allows us to standarsize our data
from sklearn import preprocessing
# Allows us to split our data into training and testing
from sklearn.model_selection import train_test_split
# Allows us to test parameters of classification algori
from sklearn.model_selection import GridSearchCV
# Logistic Regression classification algorithm
from sklearn.linear_model import LogisticRegression
# Support Vector Machine classification algorithm
from sklearn.svm import SVC
# Decision Tree classification algorithm
from sklearn.tree import DecisionTreeClassifier
# K Nearest Neighbors classification algorithm
from sklearn.neighbors import KNeighborsClassifier
```

```
import folium
import wget
import pandas as pd
```

```
# Import folium MarkerCluster plugin
from folium.plugins import MarkerCluster
# Import folium MousePosition plugin
from folium.plugins import MousePosition
# Import folium DivIcon plugin
from folium.features import DivIcon
```

```
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px
```

Thank you!