IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Alicja Gawalska
22 August 2023

Outline

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

Executive Summary

- **Summary of methodologies**
  - SpaceX Data Collection using SpaceX API
  - SpaceX Data Collection with Web Scraping
  - SpaceX Data Wrangling
  - SpaceX Exploratory Data Analysis using SQL
  - Space-X EDA DataViz Using Python Pandas and Matplotlib
  - Space-X Launch Sites Analysis with Folium-Interactive Visual Analytics and PlotyDash
  - SpaceX Machine Learning Landing Prediction
- **Summary of all results**
  - EDA results
  - Interactive Visual Analytics and Dashboards
  - Predictive Analysis(Classification)

Introduction

- Project background and context

On its website, SpaceX promotes Falcon 9 rocket launches at a price of $62 million, a stark contrast to other providers whose charges soar to $165 million or more per launch. The substantial cost reduction is largely attributed to SpaceX's innovative practice of reusing the initial stage of the rocket. Consequently, gauging the potential successful landing of this first stage enables the estimation of the launch expenditure. This valuable insight holds significance for rival companies aiming to compete with SpaceX in securing contracts for rocket launches.

- Problems you want to find answers

Within this capstone project, our objective revolves around forecasting the potential success of the initial stage landing of the Falcon 9. To accomplish this, we will leverage data sourced from the Falcon 9 rocket launches, as showcased on SpaceX's official website.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

- Data sets were collection.

The initial step involved gathering data through the utilization of SpaceX API, a RESTful API. This entailed initiating a GET request to the SpaceX API, facilitated by a set of auxiliary functions designed to streamline the extraction of launch information using identification numbers within the launch data. Subsequently, the rocket launch data was acquired from the designated SpaceX API URL.

In order to ensure uniformity in the acquired JSON results, the SpaceX launch data was retrieved and processed using a GET request. The response content was decoded as a JsonResult, subsequently transformed into a Pandas dataframe to enhance data management and analysis.

In addition to the API-based approach, web scraping techniques were employed to compile historical launch records of the Falcon 9 from a Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches." These records were embedded within an HTML structure. By harnessing the capabilities of the BeautifulSoup and request libraries, the Falcon 9 launch records were extracted from the Wikipedia page's HTML table. Following this extraction, the table data was parsed and converted into a Pandas dataframe, allowing for further analysis and integration with the API-acquired data.

Data Collection – SpaceX API

- The data acquisition process involved utilizing the SpaceX API, a RESTful API. This commenced with executing a GET request to access the SpaceX API, followed by the retrieval and parsing of the SpaceX launch data through another GET request. The obtained data was then decoded from its response content in JSON format, subsequently transformed into a JsonResult, and ultimately converted into a Pandas dataframe for streamlined analysis.
- GitHub URL of the completed SpaceX API calls notebook: https://github.com/gaw-ala/Applied_Data_Science_Capstone-Coursera_Project/blob/main/Lab1-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_
```

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

In [10]:
```
response.status_code
```

Out[10]: 200

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

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

In [12]:
```
# Get the head of the dataframe
data.head()
```

Data Collection - Scraping

- Employed web scraping techniques using BeautifulSoup and the request library to gather historical launch records of the Falcon 9 from a Wikipedia page. The objective was to extract the Falcon 9 launch information embedded within an HTML table on the Wikipedia page. This involved parsing the launch-related HTML content and subsequently assembling the extracted data into a structured dataframe.

- Add the GitHub URL: https://github.com/gaw-ala/Applied_Data_Science_Capstone-Coursera_Project/blob/main/Lab2-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]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, 'html.parser')
```

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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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

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

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

In [8]:
```
# 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 [9]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

Data Wrangling

- Following the acquisition and construction of a Pandas dataframe from the amassed dataset, a filtration process was implemented. Specifically, the data was filtered using the "BoosterVersion" column to exclusively retain records associated with Falcon 9 launches. Subsequently, attention was directed towards handling missing values within the "LandingPad" and "PayloadMass" columns.
- To address missing values within the "PayloadMass" column, a strategy was adopted involving the replacement of these gaps with the mean value derived from the column's existing data.
- Furthermore, an Exploratory Data Analysis (EDA) was conducted to unveil potential patterns within the dataset. This analysis aimed to discern meaningful insights that would ultimately guide the determination of an appropriate label for training supervised machine learning models.
- GitHub URL: https://github.com/gaw-ala/Applied_Data_Science_Capstone-Coursera_Project/blob/main/Lab3-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

In [12]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df['Class'].value_counts()
```

Out[12]:
```
1    60
0    30
Name: Class, dtype: int64
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

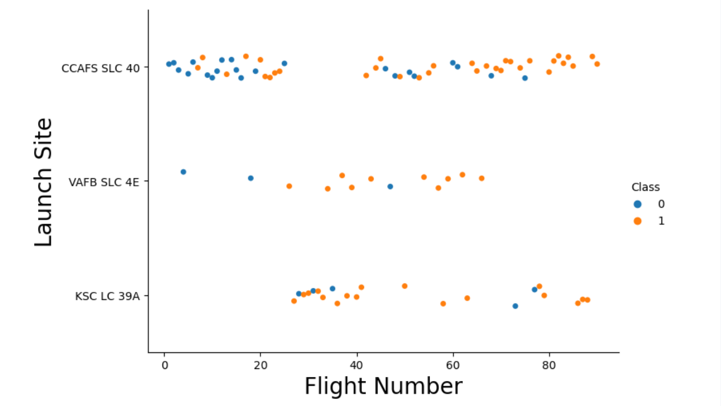
In [15]:
```
landing_class=df['Class']
df[['Class']].head(8)
```

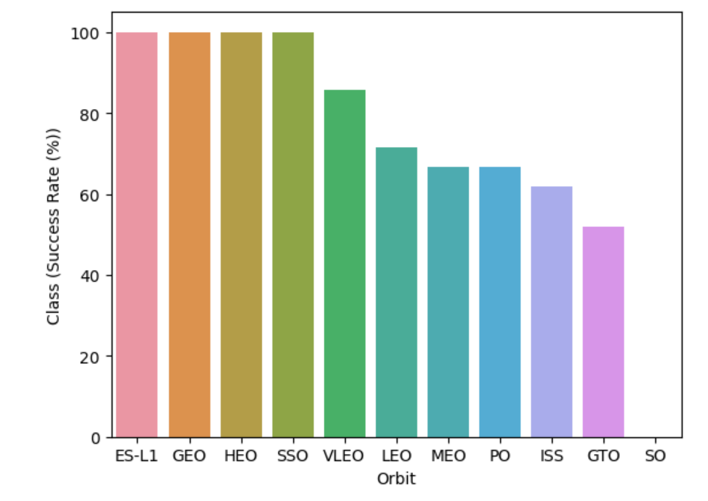
Out[15]:

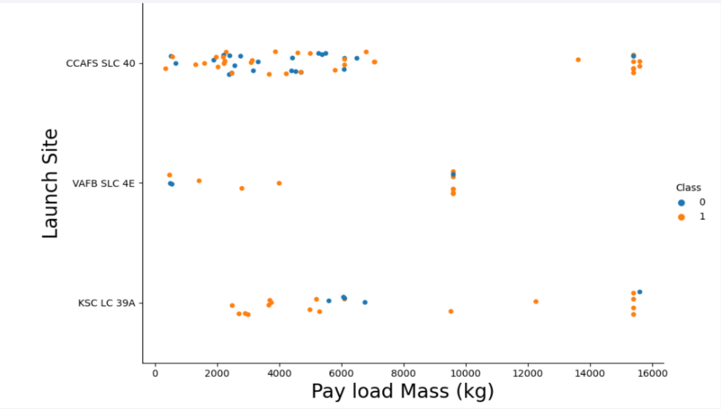
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

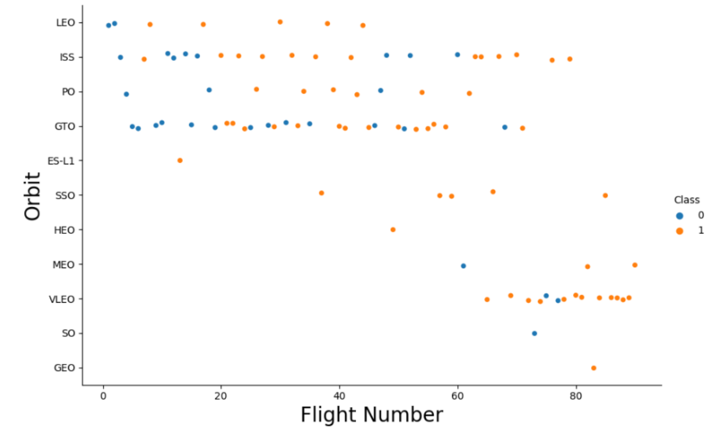
EDA with Data Visualization

- Conducted comprehensive data analysis and feature engineering utilizing the Pandas and Matplotlib libraries. This process encompassed:
  - Engaging in Exploratory Data Analysis (EDA)
  - Undertaking data preparation and feature engineering
  - Employing scatter plots to visually portray correlations between variables such as Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit Type, as well as Payload and Orbit Type
  - Utilizing bar charts to effectively illustrate the connection between the success rates of different orbit types
  - Employing line plots to graphically depict the annual trend of launch success rates.
- GitHub URL: https://github.com/gaw-ala/Applied_Data_Science_Capstone-Coursera_Project/blob/main/Lab5-jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb









EDA with SQL

- During the exploratory data analysis (EDA) phase, the subsequent SQL queries were executed:
  - Presented the names of distinct launch sites engaged in space missions.

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

  - Exhibited five records where launch sites commence with the character sequence 'CCA.'

```
%sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

  - Showcased the cumulative payload mass transported by boosters launched under NASA's (CRS) program

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

  - Disclosed the mean payload mass transported by booster version F9 v1.1.

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version
```

EDA with SQL

- Retrieve the date of the initial achievement of a successful landing on a ground pad.

```
%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
```

- Enumerate the names of boosters that have accomplished successful landings on a drone ship, and simultaneously feature a payload mass ranging between 4000 and 6000 kilograms. (%sqlSELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG > 4000 AND PAYLOAD_MASS__KG_ < 6000;)

```
%sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)" AND PAYLOAD_
```

- Summarize the overall count of missions categorized into both successful and unsuccessful outcomes.

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

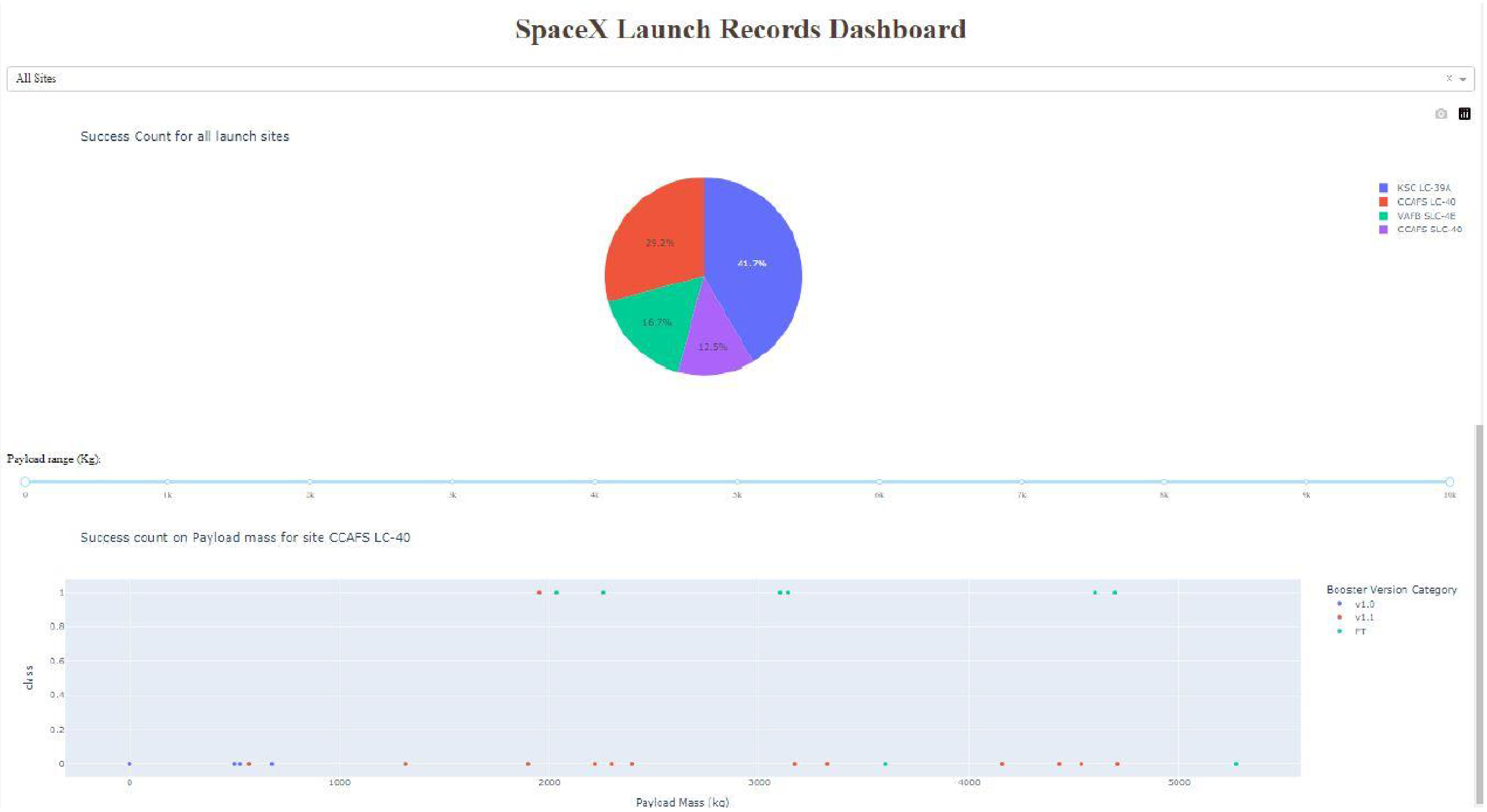
- GitHub URL: https://github.com/gaw-ala/Applied_Data_Science_Capstone-Coursera_Project/blob/main/Lab4-jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Generated a Folium map to visually represent all launch sites, employing various map elements like markers, circles, and lines to signify the outcome (success or failure) of launches associated with each specific launch site.
- Formulated a set of launch outcomes, classified as either failure (0) or success (1).
- GitHub URL: https://github.com/gaw-ala/Applied_Data_Science_Capstone-Coursera_Project/blob/main/Lab6-jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Constructed an interactive dashboard application using Plotly Dash, comprising the following steps:
  - Incorporating a dropdown input component for selecting launch sites.
  - Establishing a callback function to visualize a success-based pie chart corresponding to the chosen launch site from the dropdown.
  - Introducing a range slider to facilitate payload selection.
  - Implementing a callback function to display a scatter plot depicting the correlation between success outcomes and payload values.
- GitHub URL: https://github.com/gaw-ala/Applied_Data_Science_Capstone-Coursera_Project/blob/main/Lab7-Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash%20-%20spacex_dash_app.py


SpaceX Launch Records Dashboard
All Sites
Success Count for all launch sites
41.7%
29.2%
16.7%
12.5%
KSC LC-39A
CCAFS LC-40
VAFB SLC-4E
CCAFS SLC-40
Payload range (Kg):
0
1k
2k
3k
4k
5k
6k
7k
8k
9k
10k
Success count on Payload mass for site CCAFS LC-40
class
Payload Mass (kg)
Booster Version Category
v1.0
v1.1
FT

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- You need present your model development process using key phrases and flowchart
- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

Once the data was imported into a Pandas DataFrame, my focus shifted towards conducting exploratory data analysis (EDA) and establishing training labels. This was achieved through the following steps:

- I generated a NumPy array from the "Class" column within the dataset, utilizing the method to_numpy(). This array was then assigned to the variable Y, representing the outcome variable.
- Subsequently, I standardized the feature dataset (denoted as x) by applying the preprocessing.StandardScaler() function from the Sklearn library.
- Following standardization, I partitioned the data into training and testing sets using the train_test_split function from sklearn.model_selection. This division entailed configuring the test_size parameter as 0.2 and setting the random_state to 2 to ensure consistent randomization.

With the aim of identifying the optimal machine learning model/method that delivers superior performance on the test data among SVM, Classification Trees, k-nearest neighbors, and Logistic Regression, the following approach was taken:

- First, I instantiated an object for each of the aforementioned algorithms. Additionally, I created a GridSearchCV object for hyperparameter tuning and assigned a predefined set of parameters for each model.
- For the models under scrutiny, I established separate GridSearchCV objects with a cross-validation parameter (cv) set to 10. Subsequently, I trained each GridSearch object on the training data to determine the best hyperparameters.
- After the training phase, I extracted the GridSearchCV objects for each model. I retrieved the optimal parameters using the best_params_ attribute and showcased the accuracy on the validation data through the best_score_ attribute.
- Lastly, I utilized the score method to compute the accuracy on the test data for each model. Furthermore, I generated a confusion matrix for each model by comparing the test outcomes with the predicted results, facilitating a visual representation of model performance.

Results

- The provided table illustrates the accuracy scores for the test data across various methods, allowing for a comparison to determine the best-performing approach among SVM, Classification Trees, k-nearest neighbors, and Logistic Regression.

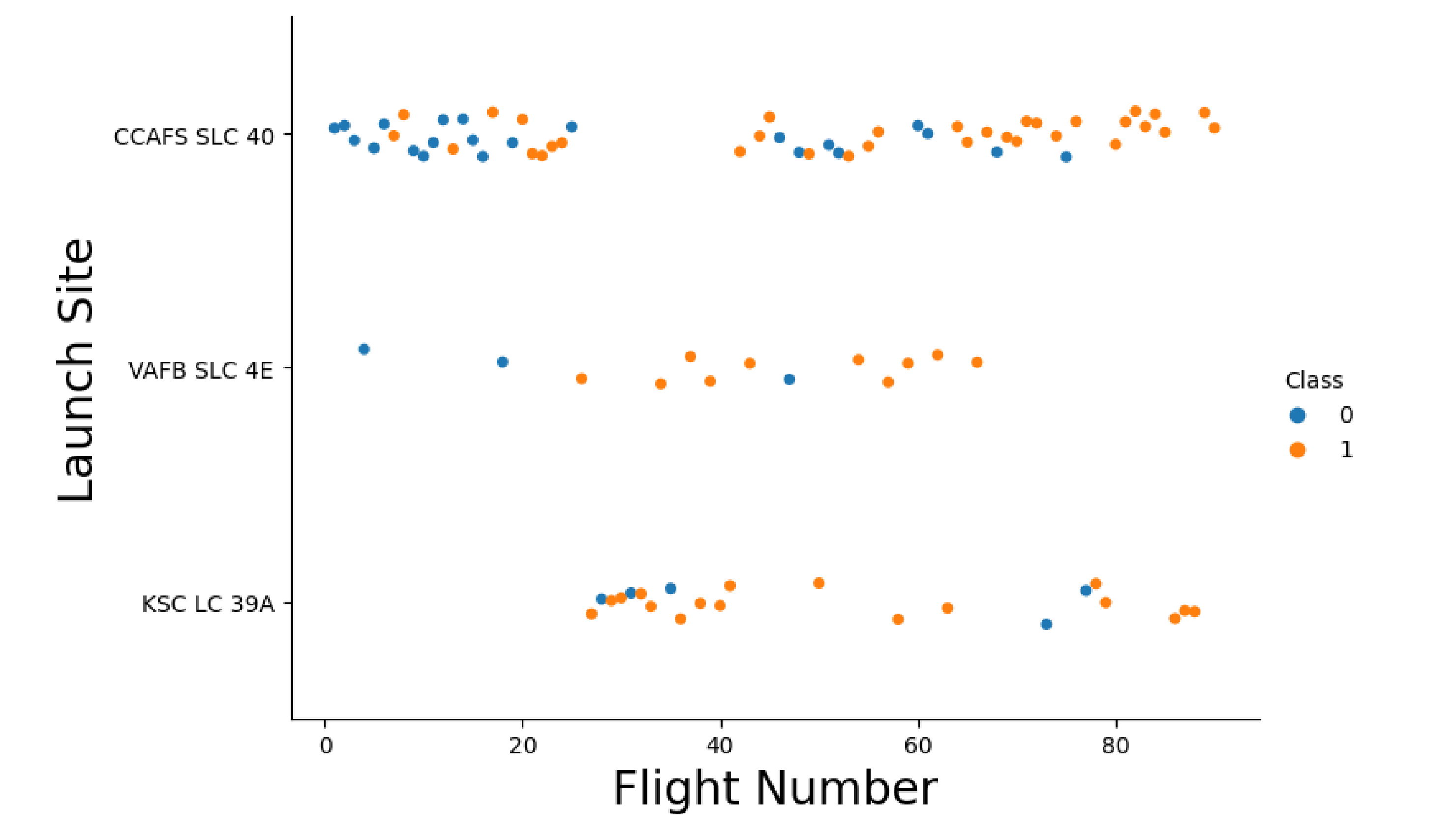
	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

- GitHub URL: https://github.com/gaw-ala/Applied_Data_Science_Capstone-Coursera_Project/blob/main/Lab8-IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Section 2

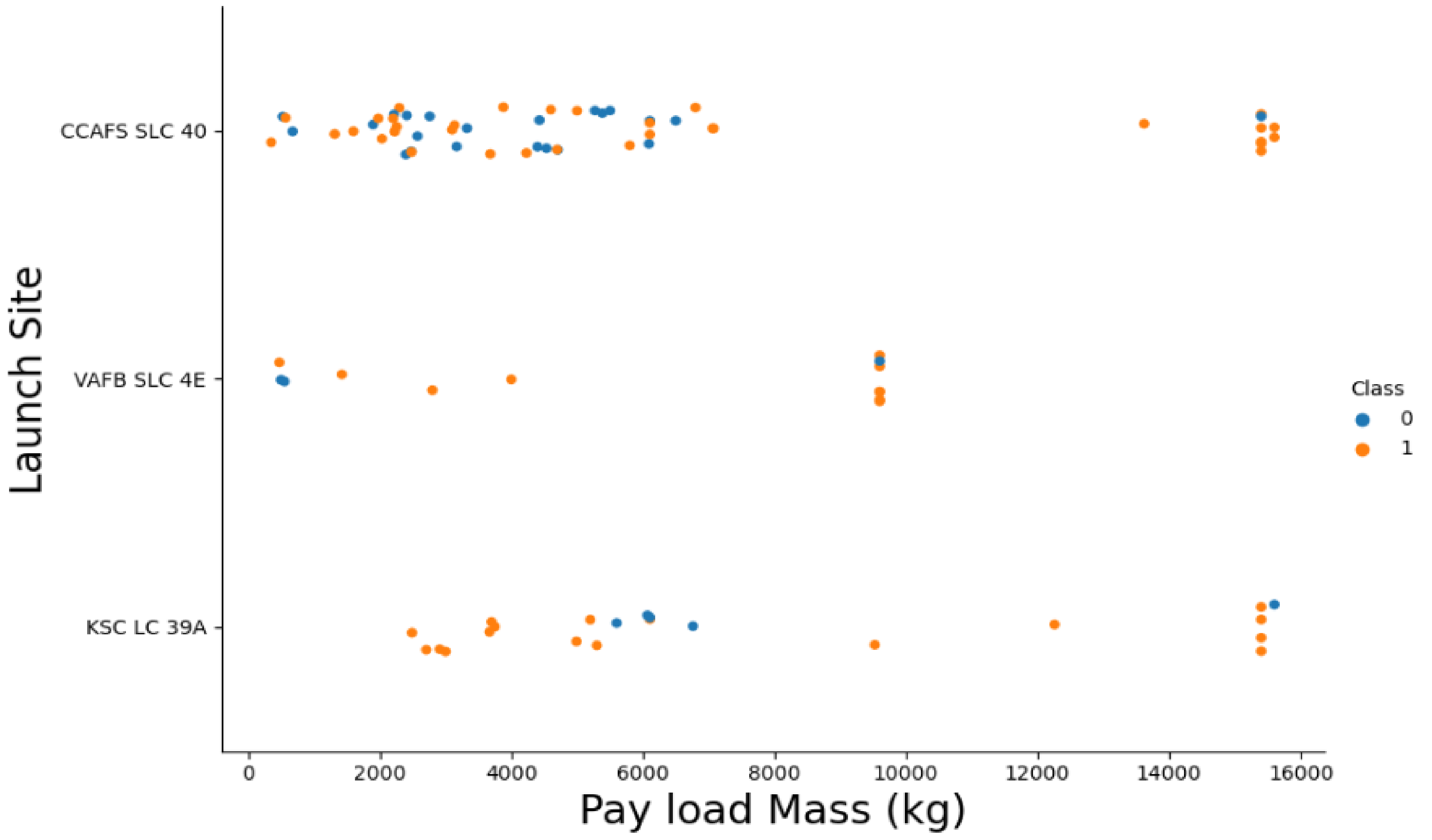
Insights drawn from EDA

Flight Number vs. Launch Site



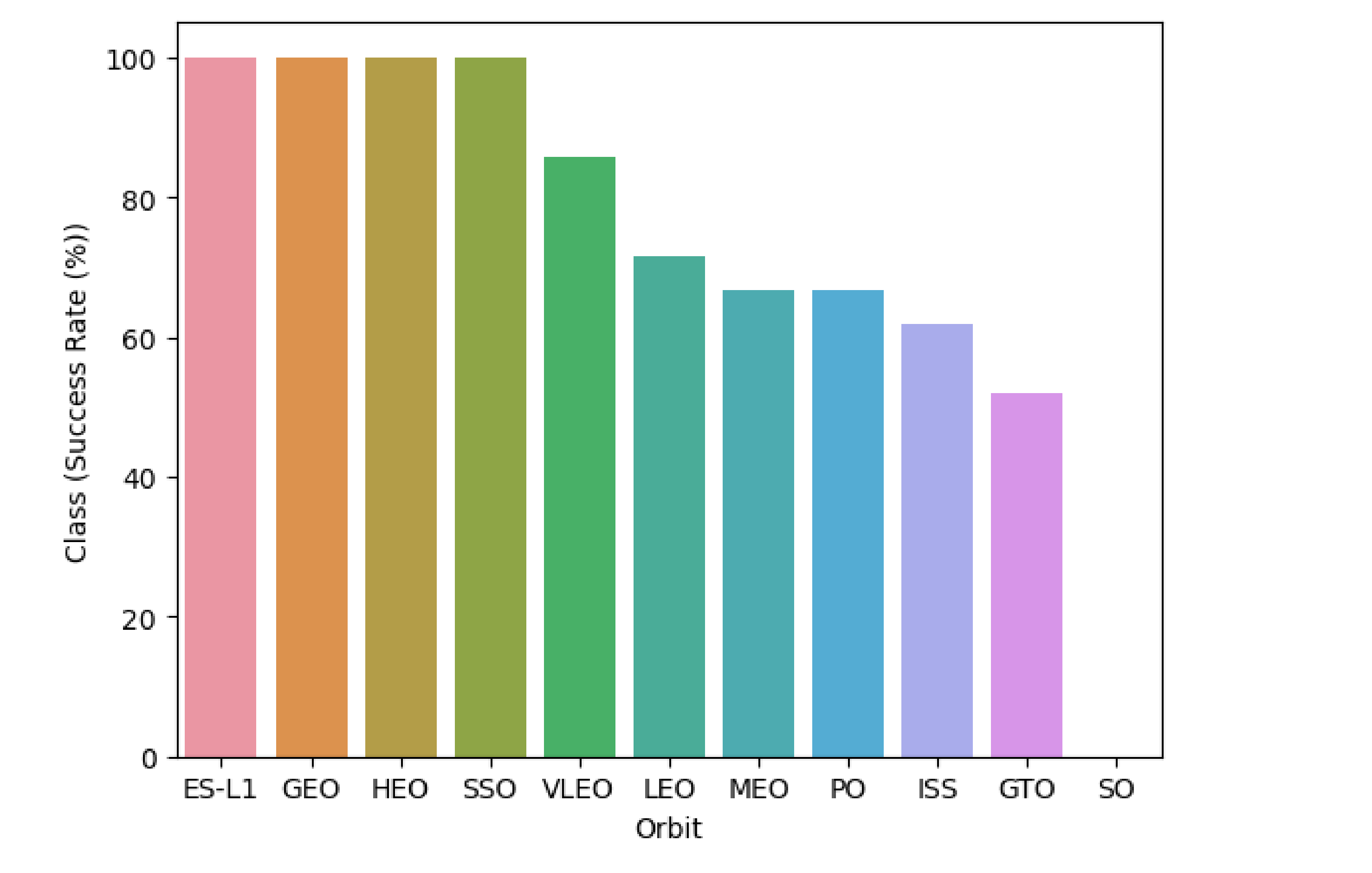
- It can be inferred that with an increase in the flight number across the three launch sites, the corresponding success rate also rises. For example, the VAFB SLC 4E launch site exhibits a 100% success rate after Flight number 50. Similarly, both the KSC LC 39A and CCAFS SLC 40 sites achieve a 100% success rate beyond the 80th flight.

Payload vs. Launch Site



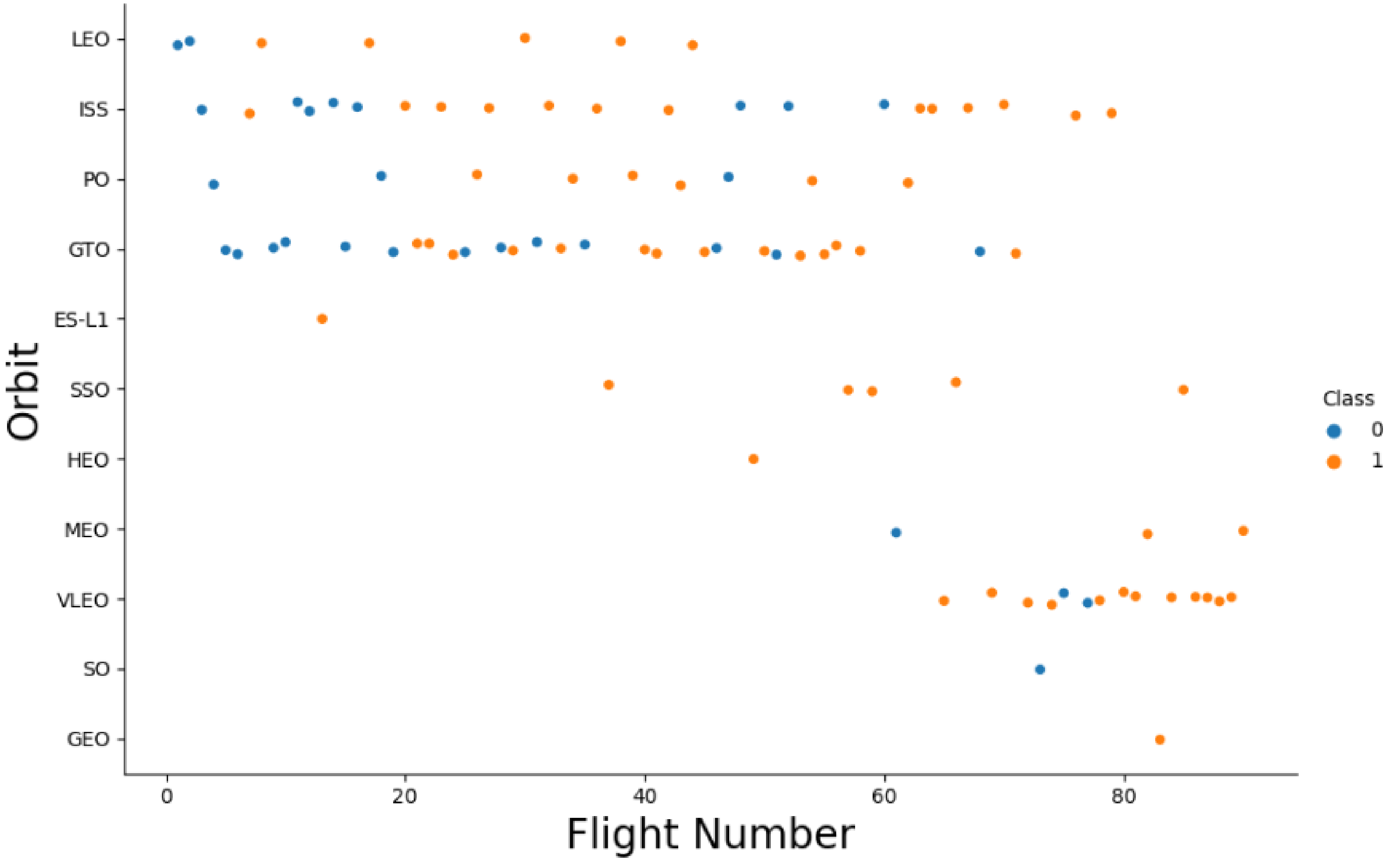
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type



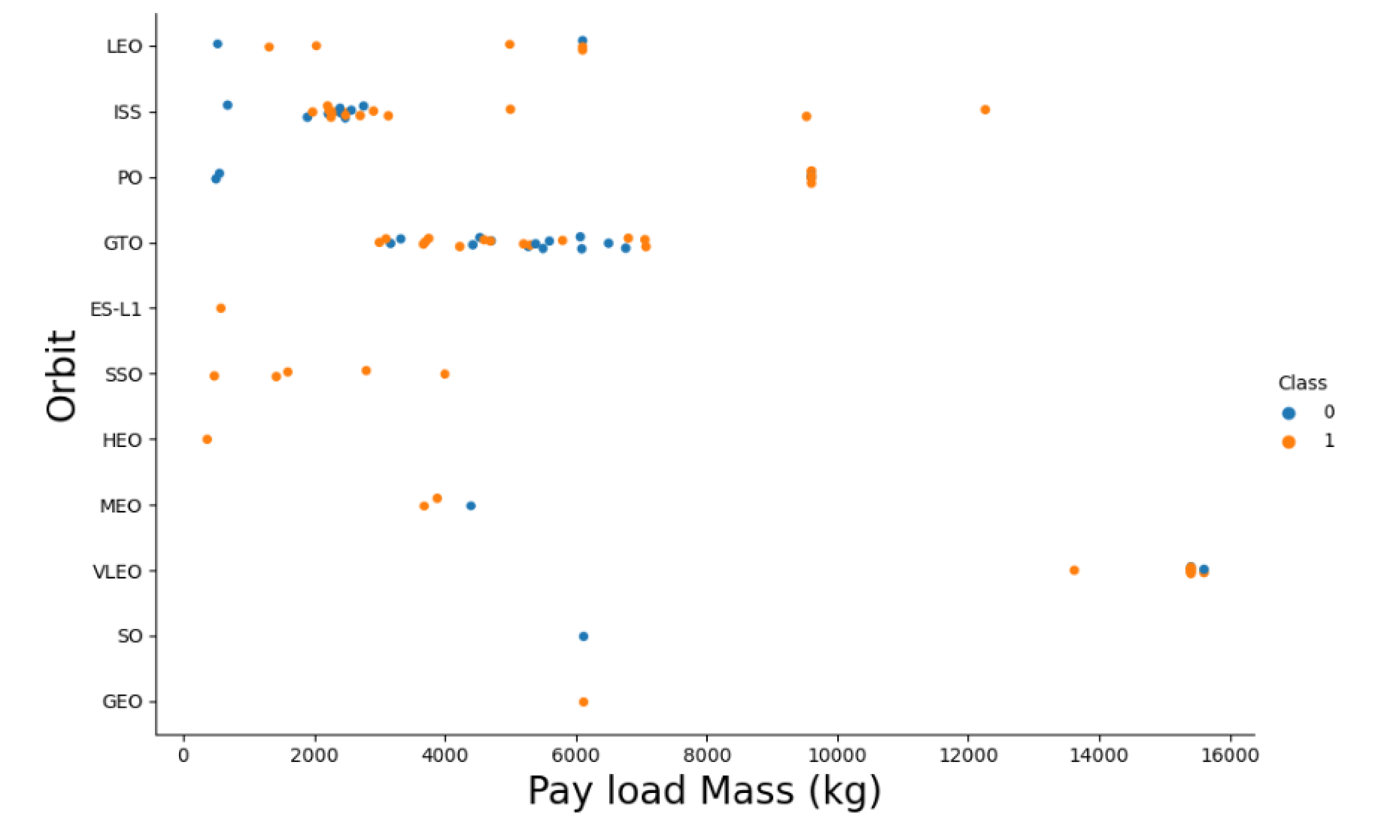
Among the orbits, ES-L1, GEO, HEO, and SSO demonstrate exceptional success rates of 100%, while the SO orbit exhibits the lowest success rate at approximately 50%. Notably, the SO orbit experiences a 0% success rate.

Flight Number vs. Orbit Type



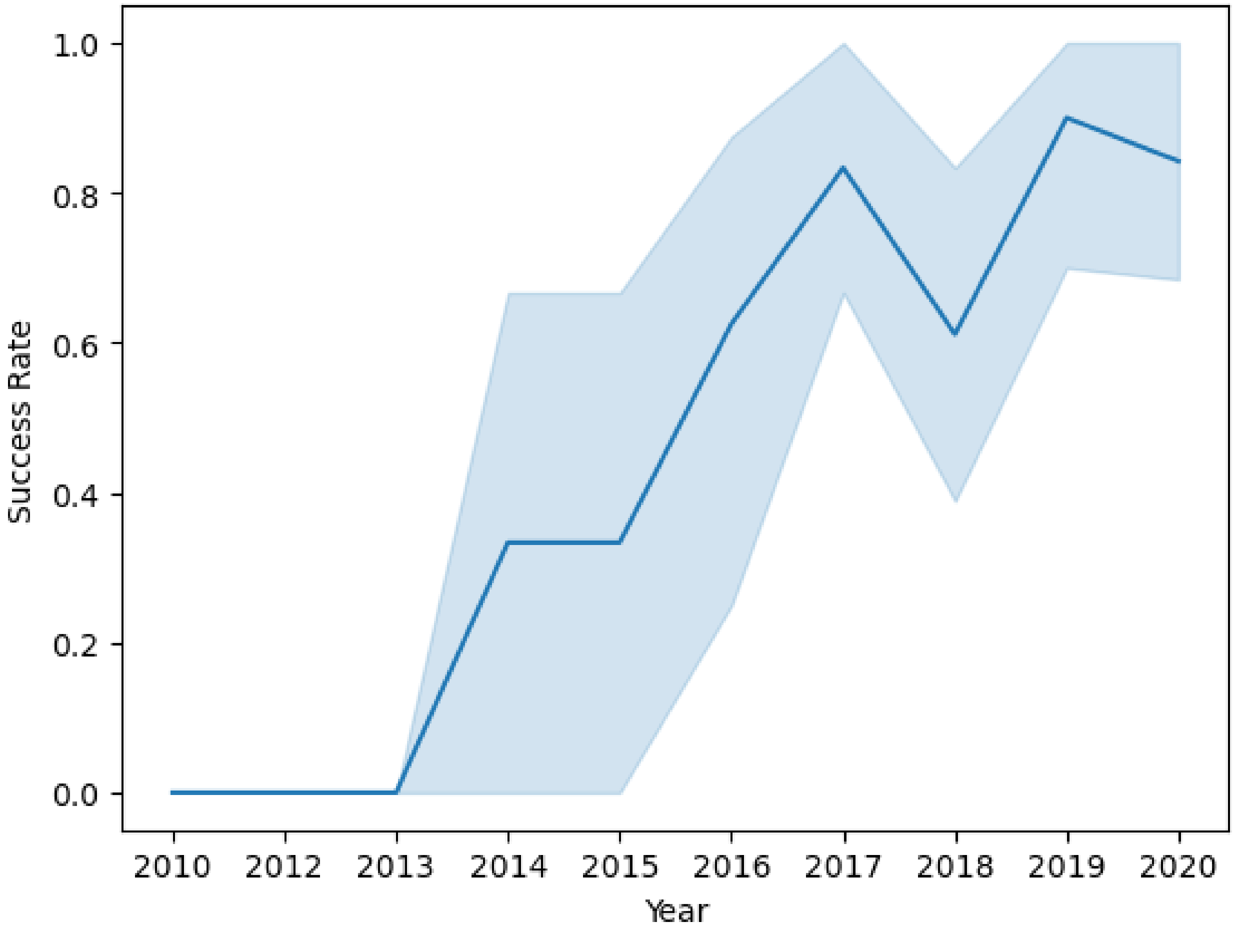
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



- When dealing with heavy payloads, there is a notable increase in the rate of successful or positive landings for orbits such as Polar, LEO, and ISS. On the contrary, in the case of GTO, it is challenging to distinguish clearly, as the chances of both positive landing rates and negative landings (unsuccessful missions) are nearly equal.

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

Task 1

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

In [8]:
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

Out[8]:

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

- Used ‘SELECT DISTINCT’ statement to return only the unique launch sites from the ‘LAUNCH_SITE’ column of the SPACEXTBL table

Launch Site Names Begin with 'CCA'

Task 2

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

In [9]:
```
%sql SELECT * FROM 'SPACEXTBL' 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

- Used ‘LIKE’ command with ‘%’ wildcard in ‘WHERE’ clause to select and dispay a table of all records where launch sites begin with the string 'CCA'

Total Payload Mass

Task 3

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

In [10]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

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

Out[10]:

Total Payload Mass(Kgs)	Customer
45596	NASA (CRS)

- Used the ‘SUM()’ function to return and dispaly the total sum of ‘PAYLOAD_MASS_KG’ column for Customer ‘NASA(CRS’

Average Payload Mass by F9 v1.1

Task 4

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

```
In [11]: %sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version I

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

Out[11]:

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

- Used the ‘AVG()’ function to return and dispaly the average payload mass carried by booster version F9 v1.1

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
```

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

MIN(DATE)
01-05-2017

- Used the ‘MIN()’ function to return and dispaly the first (oldest) date when first successful landing outcome on ground pad ‘Success (ground pad)’happened.

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

```
]: # %sql SELECT * FROM 'SPACEXTBL'
```

```
]: %sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOA
```

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

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

- Used ‘Select Distinct’ statement to return and list the ‘unique’ names of boosters with operators >4000 and <6000 to only list booster with payloads btween 4000-6000 with landing outcome of ‘Success (drone ship)’.

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [14]:
```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

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

Out[14]:

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

- Used the 'COUNT()' together with the 'GROUP BY' statement to return total number of missions outcomes

Boosters Carried Maximum Payload

Task 8

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

In [15]:
```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_
```

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

Out[15]:

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

- Using a Subquerry to return and pass the Max payload and used it list all the boosters that have carried the Max payload of 15600kgs

2015 Launch Records

Task 9

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

```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome", "Landing _Outcome"
```

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

substr(Date,7,4)	substr(Date, 4, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing _Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone ship)

- Used the ‘subsrt()’ in the select statement to get the month and year from the date column where substr(Date,7,4)='2015' for year and Landing_outcome was ‘Failure (drone ship’) and return the records nmatching the filter.

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

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql SELECT * FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORDER BY Date DESC;
```

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

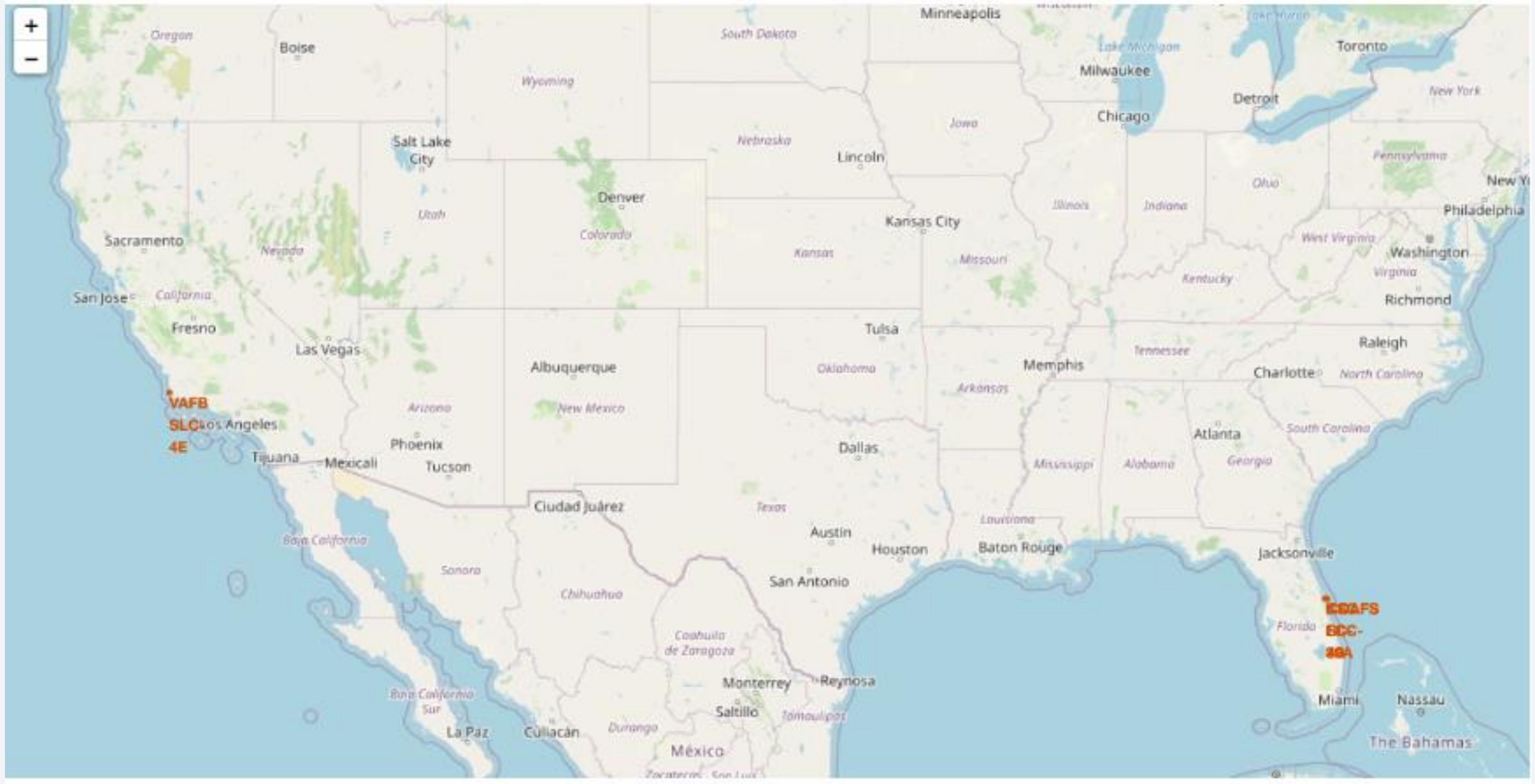
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-10-2020	12:25:57	F9 B5 B1051.6	KSC LC-39A	Starlink 13 v1.0, Starlink 14 v1.0	15600	LEO	SpaceX	Success	Success
18-08-2020	14:31:00	F9 B5 B1049.6	CCAFS SLC-40	Starlink 10 v1.0, SkySat-19, -20, -21, SAOCOM 1B	15440	LEO	SpaceX, Planet Labs, PlanetIQ	Success	Success
18-07-2016	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-04-2018	22:51:00	F9 B4 B1045.1	CCAFS SLC-40	Transiting Exoplanet Survey Satellite (TESS)	362	HEO	NASA (LSP)	Success	Success (drone ship)

- 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

Section 3

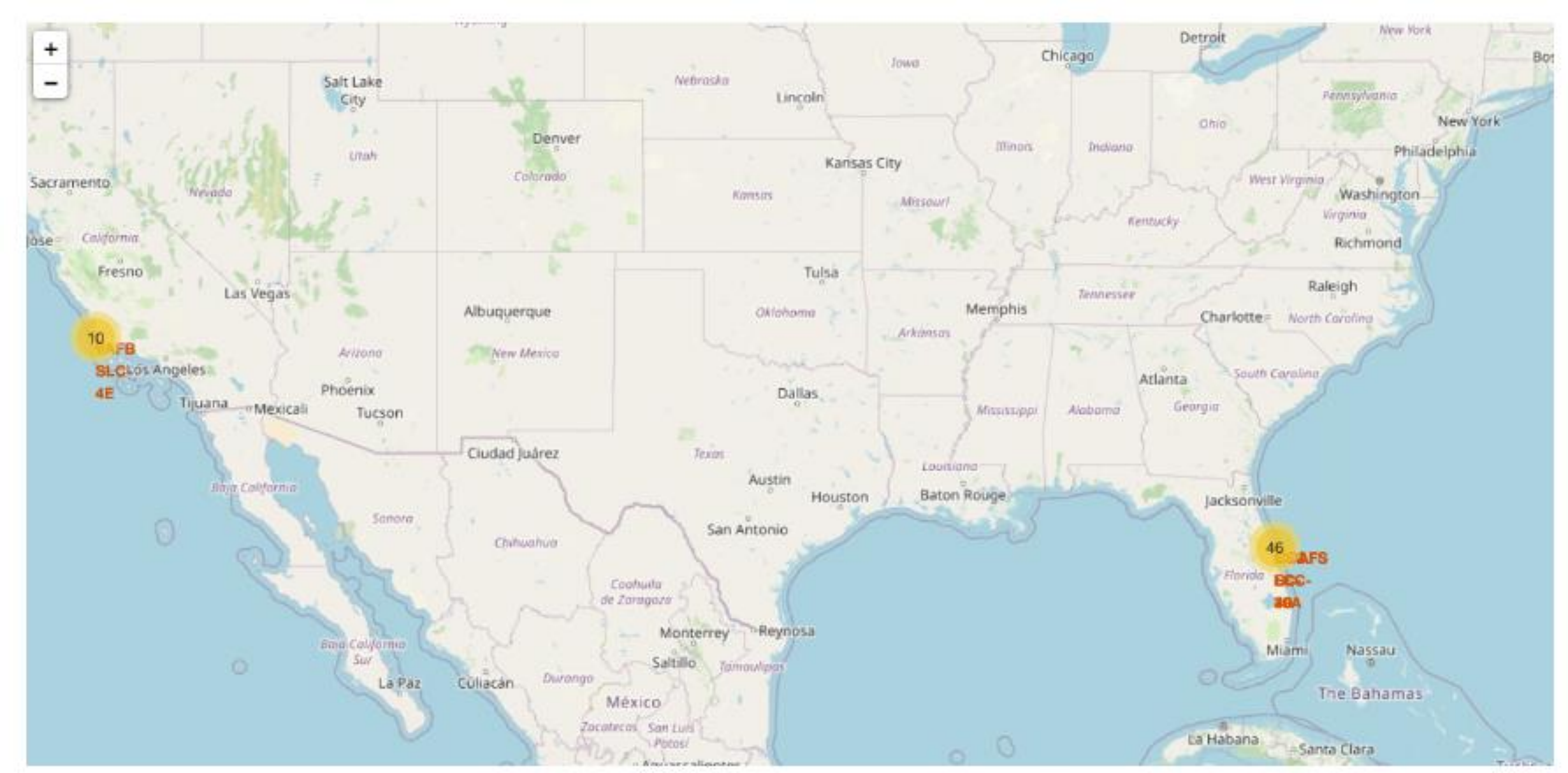
Launch Sites Proximities Analysis

Markers of all launch sites on global map

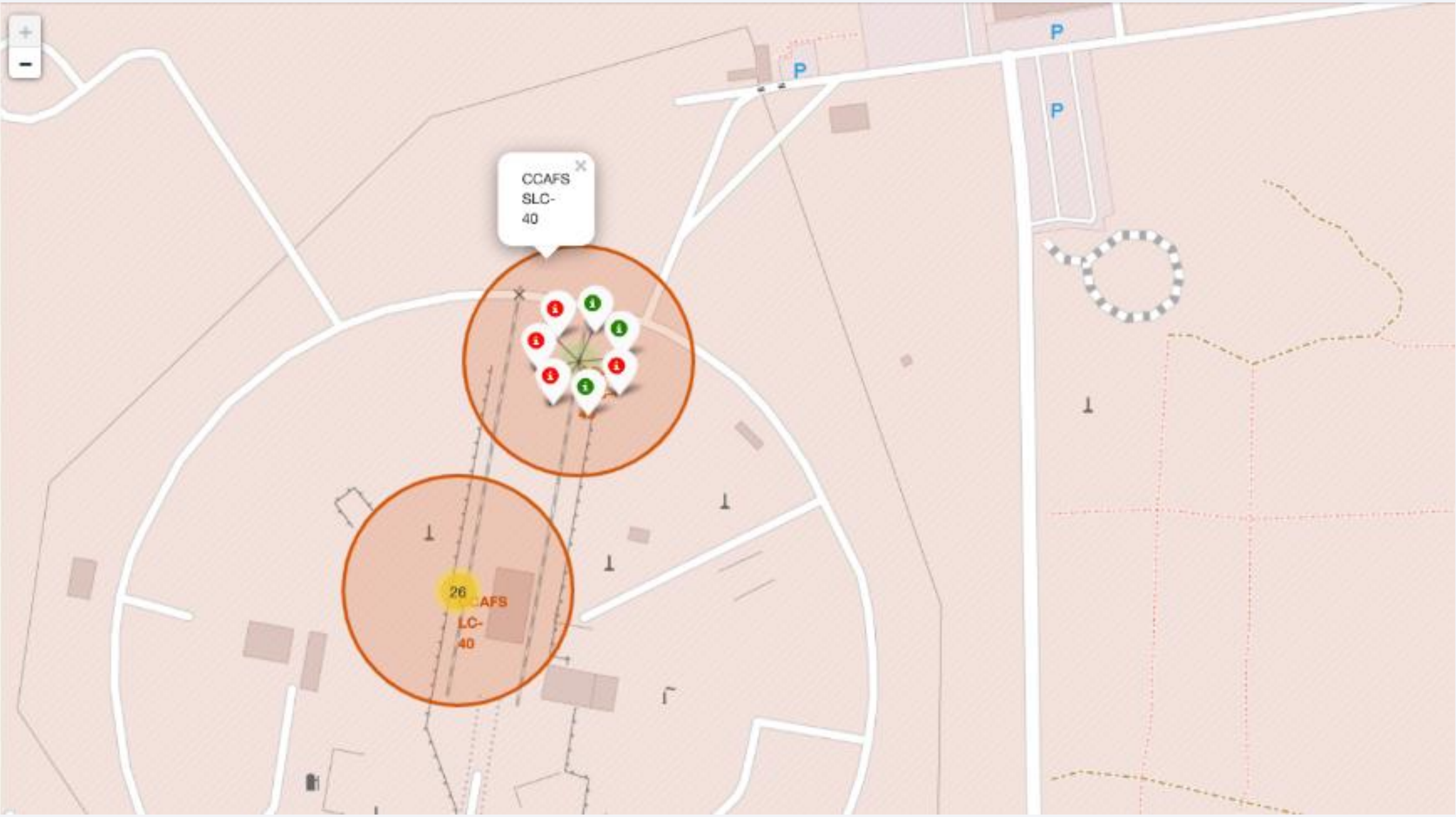


All launch sites are in proximity to the Equator, (located southwards of the US map). Also all the launch sites are in very close proximity to the coast.

Launch outcomes for each site on the map With Color Markers



Launch outcomes for each site on the map With Color Markers



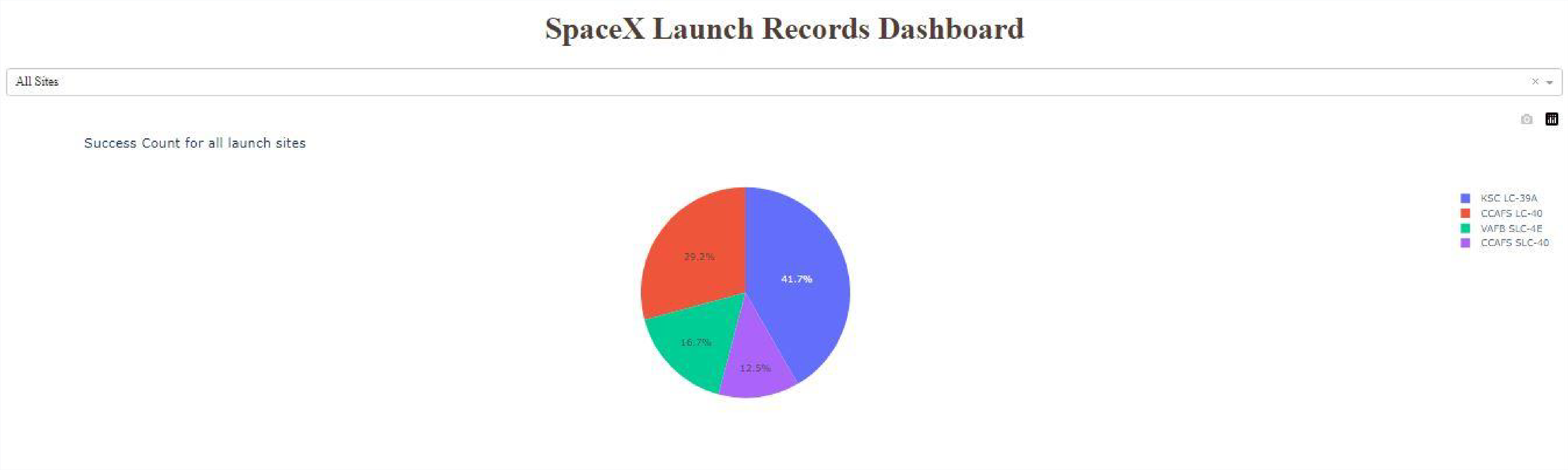
Distances between a launch site to its proximities

Launch siteCCAFS SLC-40proximity to coastline is 0.9km

Section 4

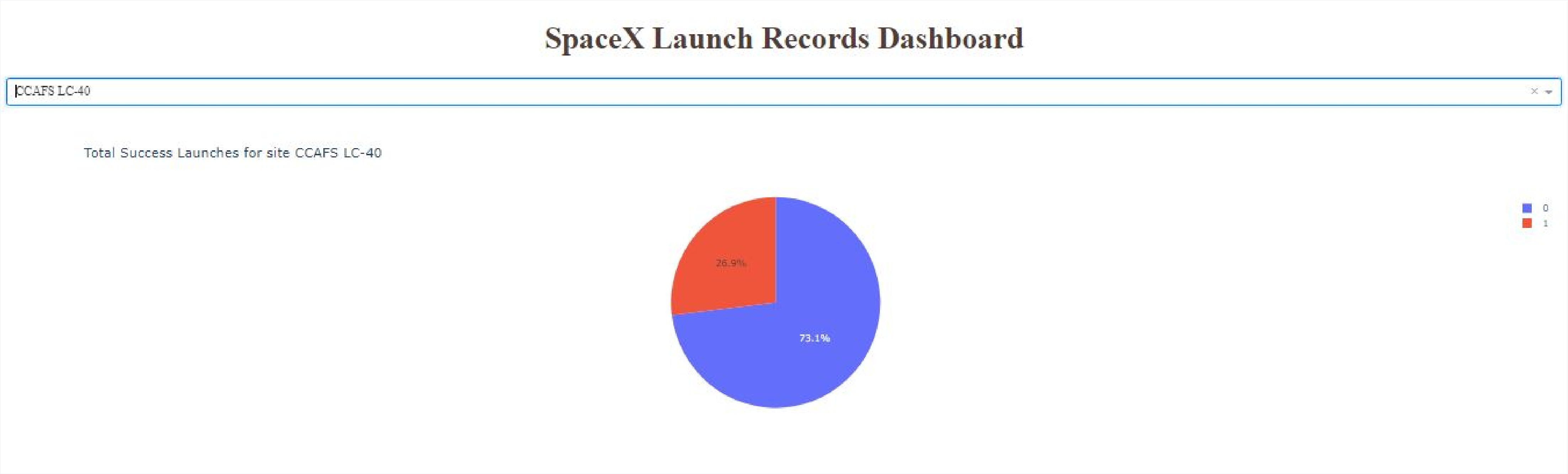
Build a Dashboard with Plotly Dash

Pie-Chart for launch success count for all sites



- Launch site KSC LC-39A has the highest launch success rate at 42% followed by CCAFS LC-40 at 29%, VAFB SLC-4E at 17% and lastly launch site CCAFS SLC-40 with a success rate of 13%

Pie chart for the launch site with 2ndhighest launch success ratio



- Launch site CCAFS LC-40 had the 2ndhighest success ratio of 73% success against 27% failed launches

Payload vs. Launch Outcome scatter plot for all sites

- For Launch site CCAFS LC-40 the booster version FT has the largest success ratefrom a payload mass of >2000kg

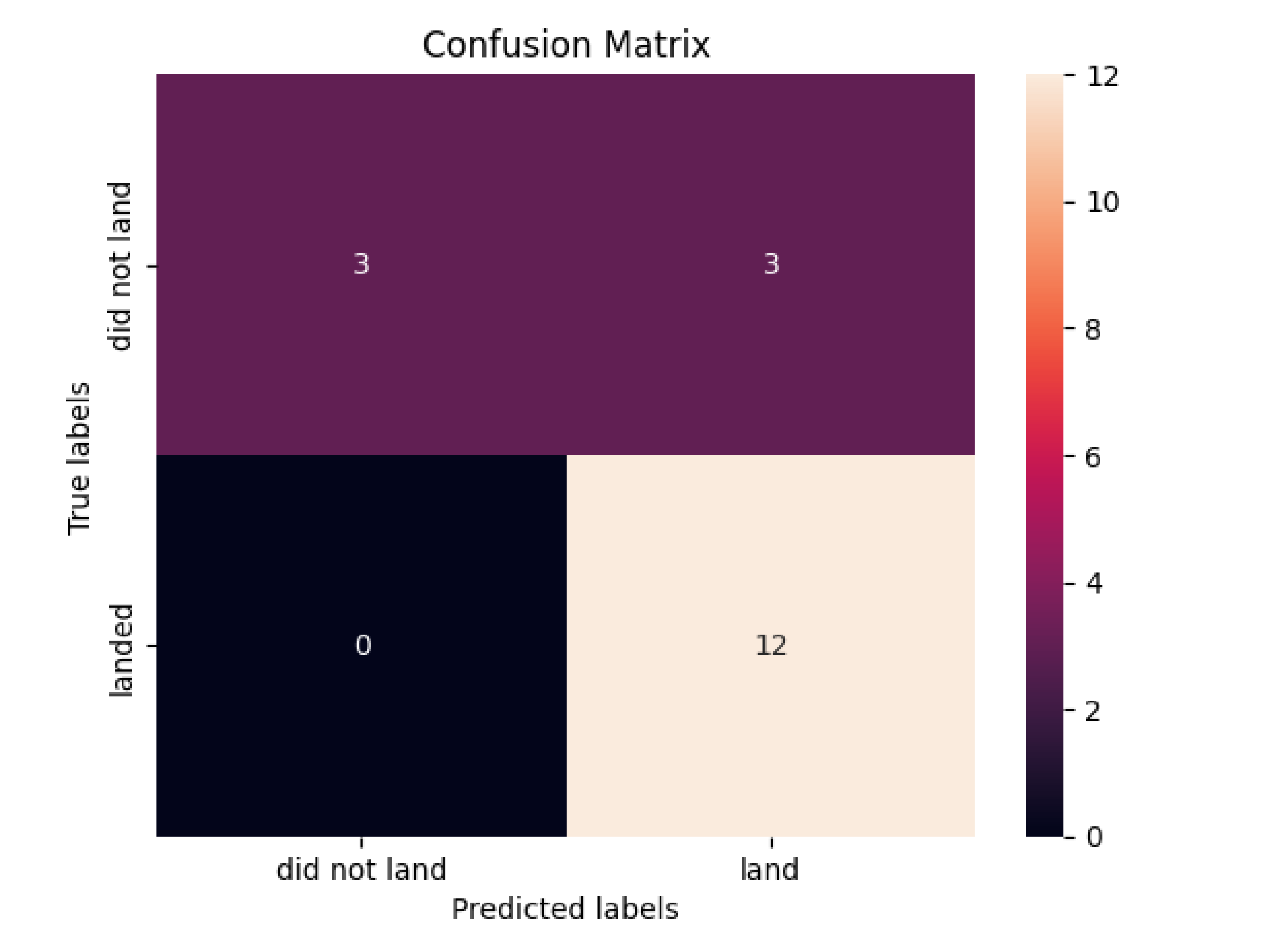
Section 5

Predictive Analysis (Classification)

Classification Accuracy

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

Confusion Matrix



Conclusions

- Distinct launch sites exhibit varying success rates. CCAFS LC-40, for instance, boasts a 60% success rate, while KSC LC-39A and VAFB SLC 4E register a higher success rate of 77%.
- It becomes apparent that, for each of the three launch sites, the success rate increases with the flight number. A notable example is the VAFB SLC 4E launch site, which attains a 100% success rate beyond Flight number 50. Similarly, both KSC LC 39A and CCAFS SLC 40 achieve a 100% success rate following the 80th flight.
- If we examine the scatter point chart depicting Payload vs. Launch Site, it becomes evident that there are no instances of heavy payload launches (greater than 10000) at the VAFB-SLC launch site.
- Orbits ES-L1, GEO, HEO, and SSO exhibit the most robust success rates, all at 100%. In contrast, the SO orbit records a lower success rate of around 50%, with a 0% success rate for the SO orbit.
- In the LEO orbit, success rates seem connected to the number of flights. Conversely, in the GTO orbit, there appears to be no discernible relationship between flight number and success rate.

Conclusions

- When dealing with heavy payloads, the rates of successful or positive landings are notably higher for Polar, LEO, and ISS orbits. However, in the case of GTO, the distinction is not as clear since both positive landing rates and negative landings (unsuccessful missions) are prevalent.
- Lastly, it's important to note that the success rate has consistently increased since 2013 and continued to rise until 2020.

Thank you!