IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Dmitrii Simonenko
02/20/2025

Outline

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

Executive Summary

- **Summary of methodologies**

1. SpaceX Data Collection using SpaceX API
2. SpaceX Data Collection with Web Scraping
3. SpaceX Data Wrangling
4. SpaceX Exploratory Data Analysis using SQL
5. SpaceX EDA DataViz Using Python
6. SpaceX Launch Sites Analysis with Folium-Interactive Visual Analytics and Plotly
7. SpaceX Machine Learning Landing Prediction

- **Summary of all results**

1. EDA results
2. Visual Analytics and Dashboards
3. Predictive Analysis

Introduction

- **Project background and context**

SpaceX 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 SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- **Problems you want to find answers**

We will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
- Perform data wrangling
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

Data collection

- Data was first collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API
- To make the requested JSON results more consistent, the SpaceX launch data was requested and parsed using the GET request and then decoded the response content as a Json result which was then converted into a Pandas dataframe
- performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page. Using BeautifulSoup and request Libraries, I extract the Falcon 9 launch HTML table records from the Wikipedia page, Parsed the table and converted it into a Pandas data frame

Data Collection – SpaceX API

- Data collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API then requested and parsed the SpaceX launch data using the GET request and decoded the response content as a Json result which was then converted into a Pandas data frame
- https://github.com/dimasik99/course_final_project/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:

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

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

```
[10]: response=requests.get(static_json_url)
```

```
[11]: response.status_code
```

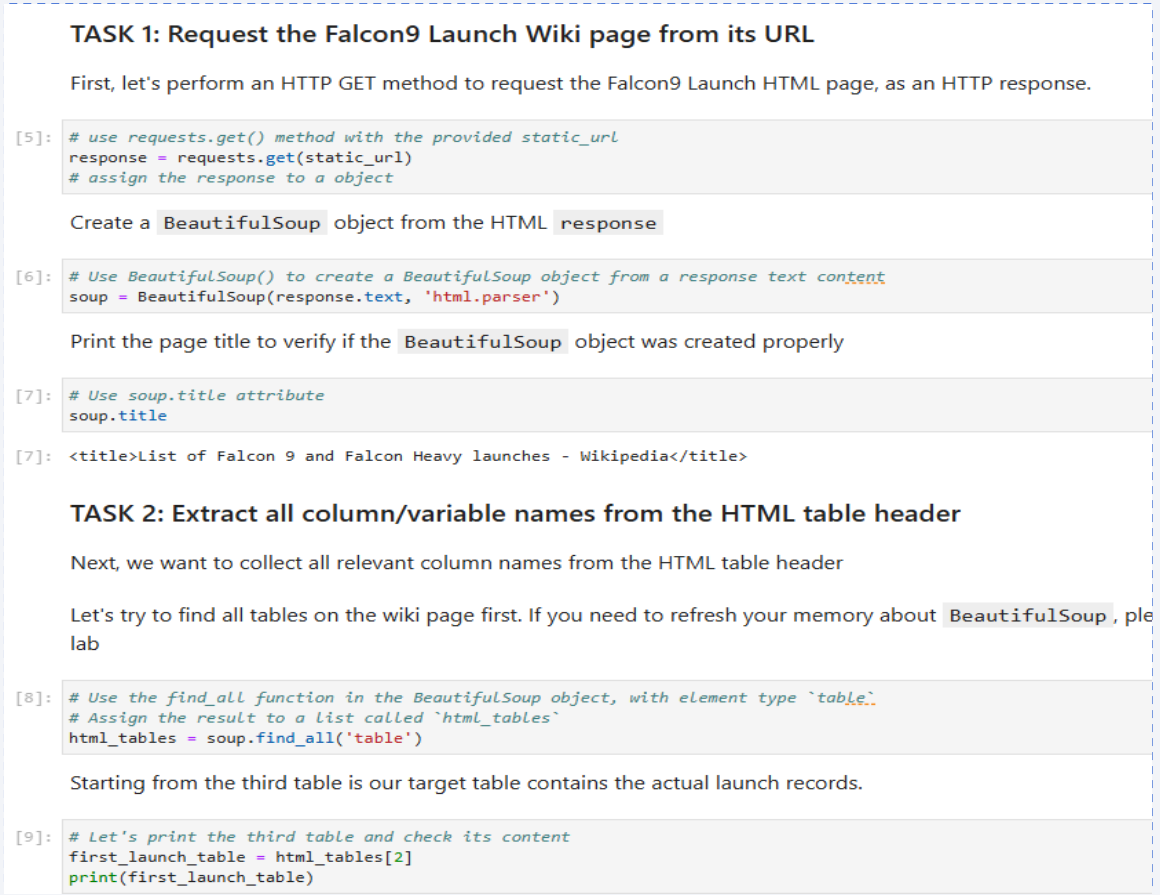
```
[11]: 200
```

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

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

Data Collection - Scraping

- Performed web scraping to collect Falcon 9 historical launch records from a Wikipedia using BeautifulSoup and request, to extract the Falcon 9 launch records from HTML table of the Wikipedia page, then created a data frame by parsing the launch HTML.
- https://github.com/dimasik99/course_final_project/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.

[5]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
```

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

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

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

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

[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`, ple
lab

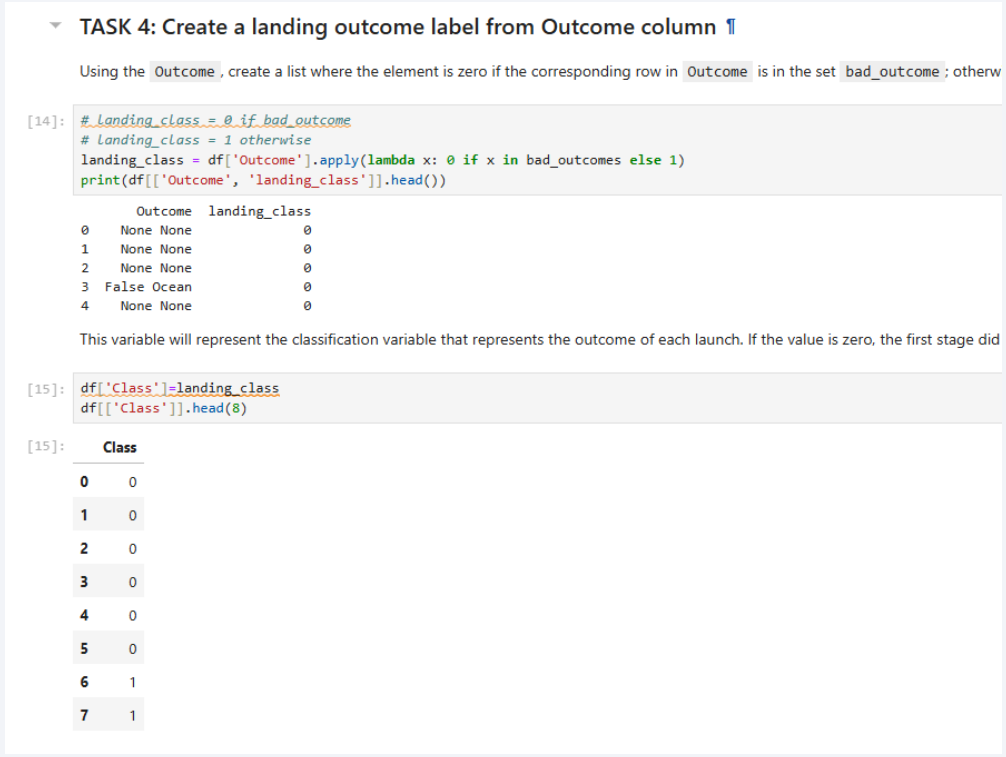
[8]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

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

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

Data Wrangling

- After obtaining and creating a Pandas DF from the collected data, data was filtered using the **BoosterVersion** column to only keep the Falcon 9 launches, then dealt with the missing data values in the **LandingPad** and **PayloadMass** columns. For the **PayloadMass** , missing data values were replaced using mean value of column.
- https://github.com/dimasik99/course_final_project/blob/main/labs-jupyter-spacex-Data%20wrangling.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`; otherw

[14]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
print(df[['Outcome', 'landing_class']].head())
```

```
      Outcome  landing_class
0    None None              0
1    None None              0
2    None None              0
3  False Ocean              0
4    None None              0
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did

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

[15]:

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

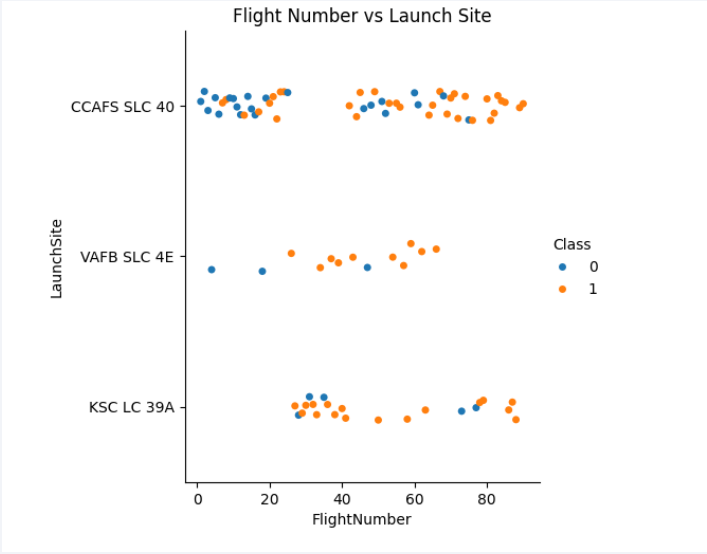
EDA with Data Visualization

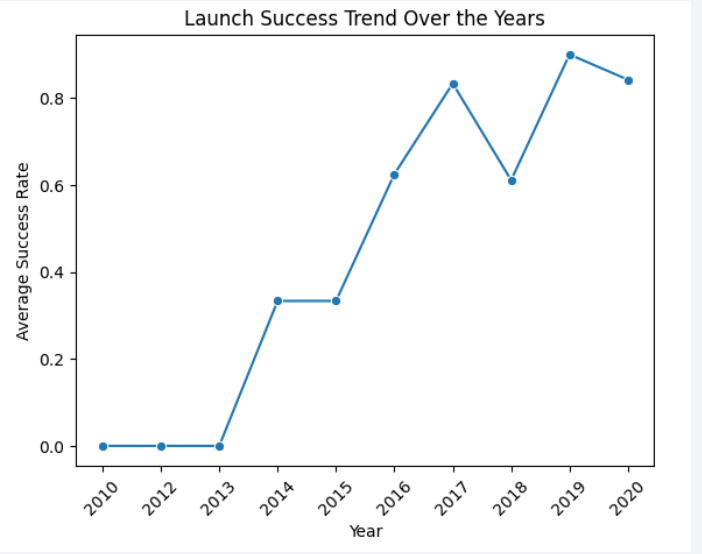
- Performed data Analysis and Feature Engineering using Pandas and Matplotlib:

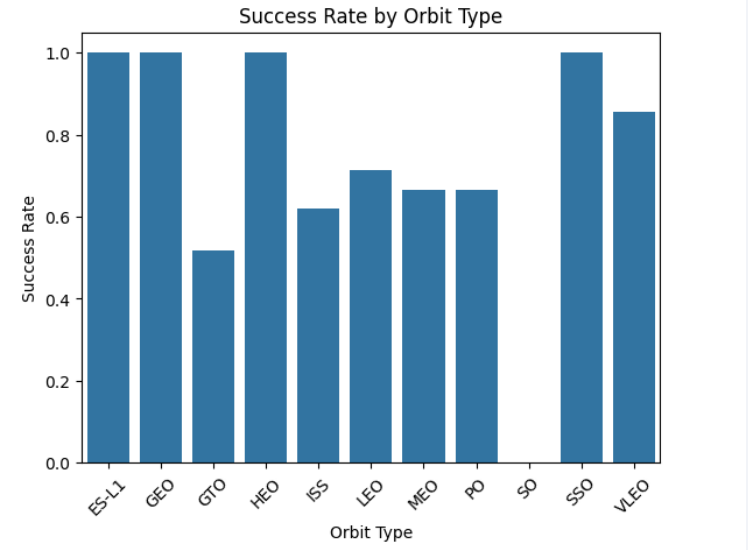
1. Exploratory Data Analysis
2. Preparing Data Feature Engineering

- Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, FlightNumber and Orbit type, Payload and Orbit type
- Used Bar chart to Visualize the relationship between success rate of each orbit type
- Line plot to Visualize the launch success yearly trend
- https://github.com/dimasik99/course_final_project/blob/main/edadataviz.ipynb

EDA with Data Visualization







EDA with SQL

The following SQL queries were performed for EDA

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

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

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

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

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass FROM
SPACEXTABLE WHERE Customer LIKE '%NASA (CRS)%';
```

EDA with SQL

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';
```

- List the date when the first successful landing outcome in ground pad was achieved

```
%sql SELECT MIN(Date) AS First_Successful_Landing FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';Display 5 records where launch sites begin with the string 'CCA'
```

- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000 Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND Payload_Mass__kg_ BETWEEN 4000 AND 6000;
```

EDA with SQL

- List the total number of successful and failure mission outcomes

```
%sql SELECT Mission_Outcome, COUNT(*) as Total_Count FROM SPACEXTABLE
GROUP BY Mission_Outcome;
```

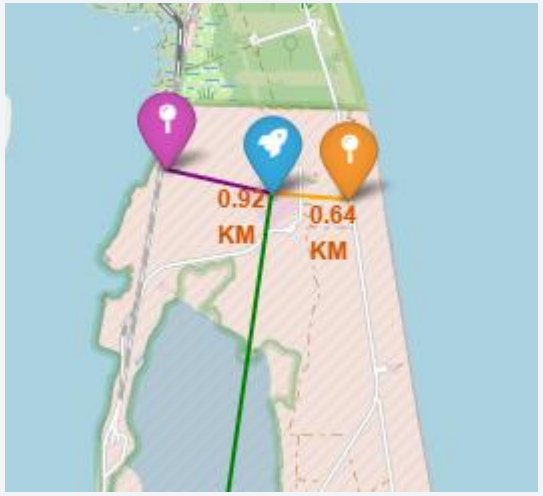
- List the names of the booster_versions which have carried the maximum payload mass

```
%sql SELECT booster_version FROM SPACEXTABLE WHERE
PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM
SPACEXTABLE)
```

- https://github.com/dimasik99/course_final_project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Created folium map to marked all the launch sites, and created map objects such as markers, circles, lines to mark the success or failure of launches for each launch site
- Created a launch set outcomes (failure=0 or success=1)
- https://github.com/dimasik99/course_final_project/blob/main/lab_jupyter_launch_site_location.ipynb



Build a Dashboard with Plotly Dash

- Built an interactive dashboard application with Plotly dash by:

1. Adding a Launch Site Drop-down Input Component
2. Adding a callback function to render success-pie-chart based on selected site dropdown
3. Adding a Range Slider to Select Payload
4. Adding a callback function to render the success-payload-scatter-chart scatter plot

- https://github.com/dimasik99/course_final_project/blob/main/spacex_dash_app.py

SpaceX Launch Records Dashboard

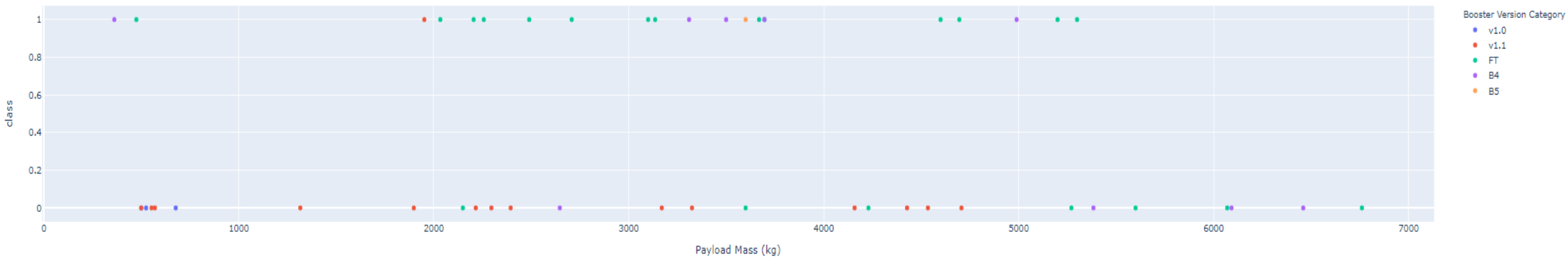
All Sites

Total Success Launches By Site

Payload range (Kg):

0 2500 5000 7500 10000

Correlation between Payload and Success for all Sites



Predictive Analysis (Classification)

- After loading the data as a Pandas Dataframe, I set out to perform exploratory Data Analysis and determine Training Labels by:

1. creating a NumPy array from the column Class in data, by applying the method to_numpy() then assigned it to the variable Y as the outcome variable.
2. Then standardized the feature dataset (x) by transforming it using preprocessing.StandardScaler() function from Sklearn.
3. After which the data was split into training and testing sets using the function train_test_split from sklearn.model_selection with the test_size parameter set to 0.2 and random_state to 2You need present your model development process using key phrases and flowchart

Predictive Analysis (Classification)

- In order to find the best ML model/ method that would performs best using the test data between SVM, Classification Trees, k nearest neighbors and Logistic Regression:

1. First created an object for each of the algorithms then created a GridSearchCV object and assigned them a set of parameters for each model.
2. For each of the models under evaluation, the GridsearchCV object was created with cv=10, then fit the training data into the GridSearch object for each to Find best Hyperparameter.
3. After fitting the training set, we output GridSearchCV object for each of the models, then displayed the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.
4. Finally using the method score to calculate the accuracy on the test data for each model and plotted a confussion matrix for each using the test and predicted outcomes.

Predictive Analysis (Classification)

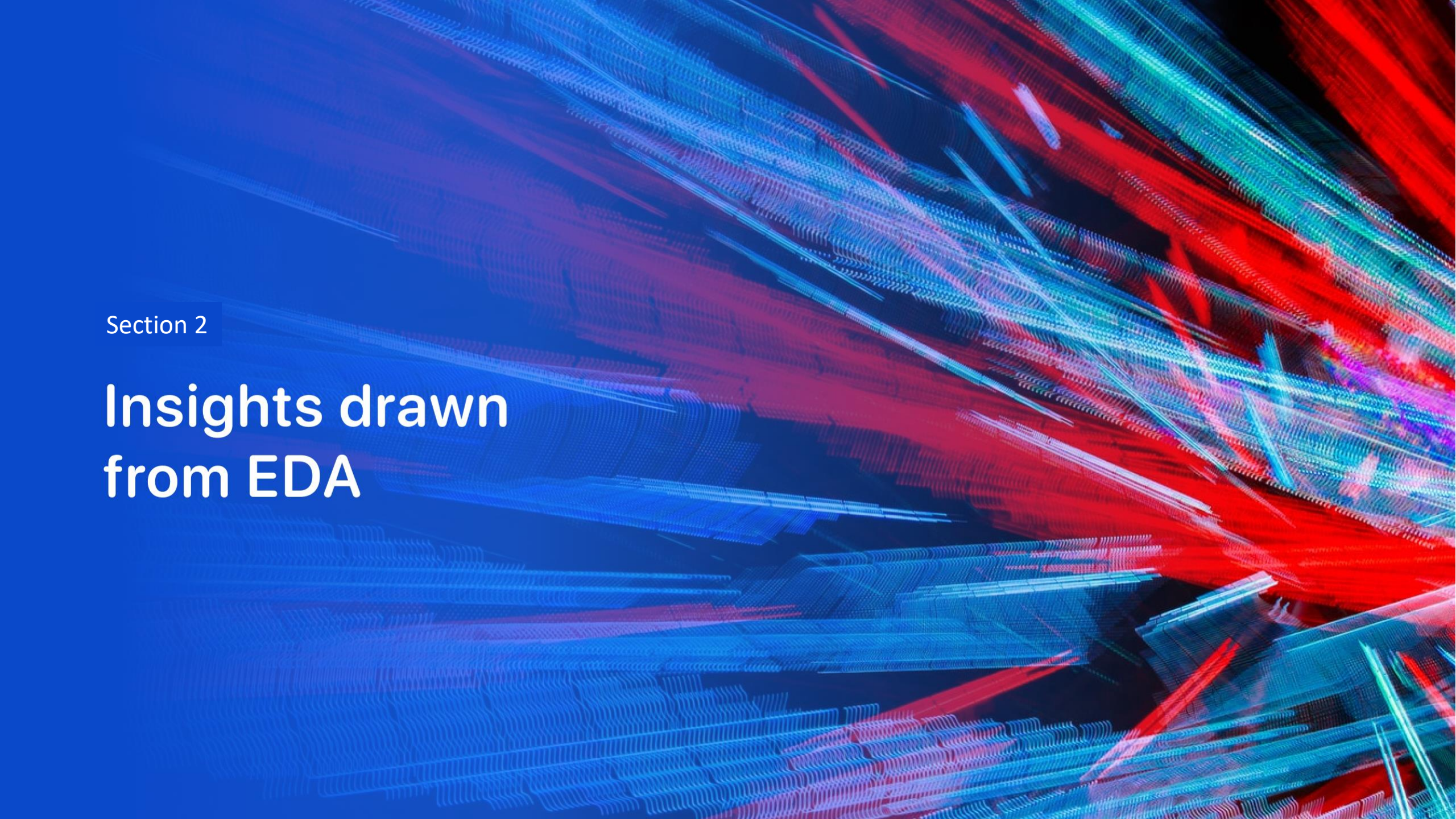
- The table below shows the test data accuracy score for each of the methods comparing them to show which performed best using the test data between SVM, Classification Trees, k nearest neighbors and Logistic Regression. **Practically all these algorithms give the same result**

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

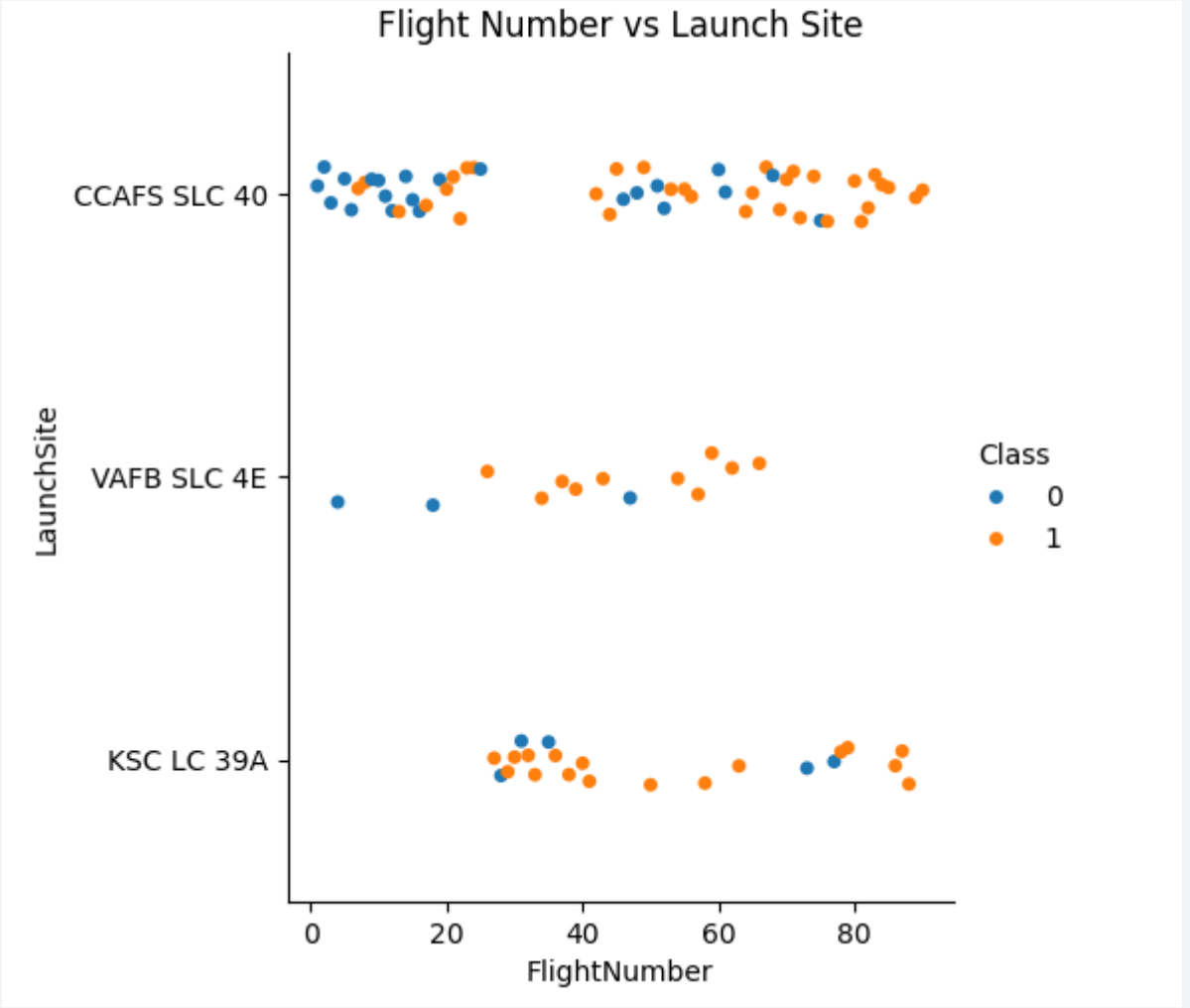
- https://github.com/dimasik99/course_final_project/blob/main/ml.ipynb

Results

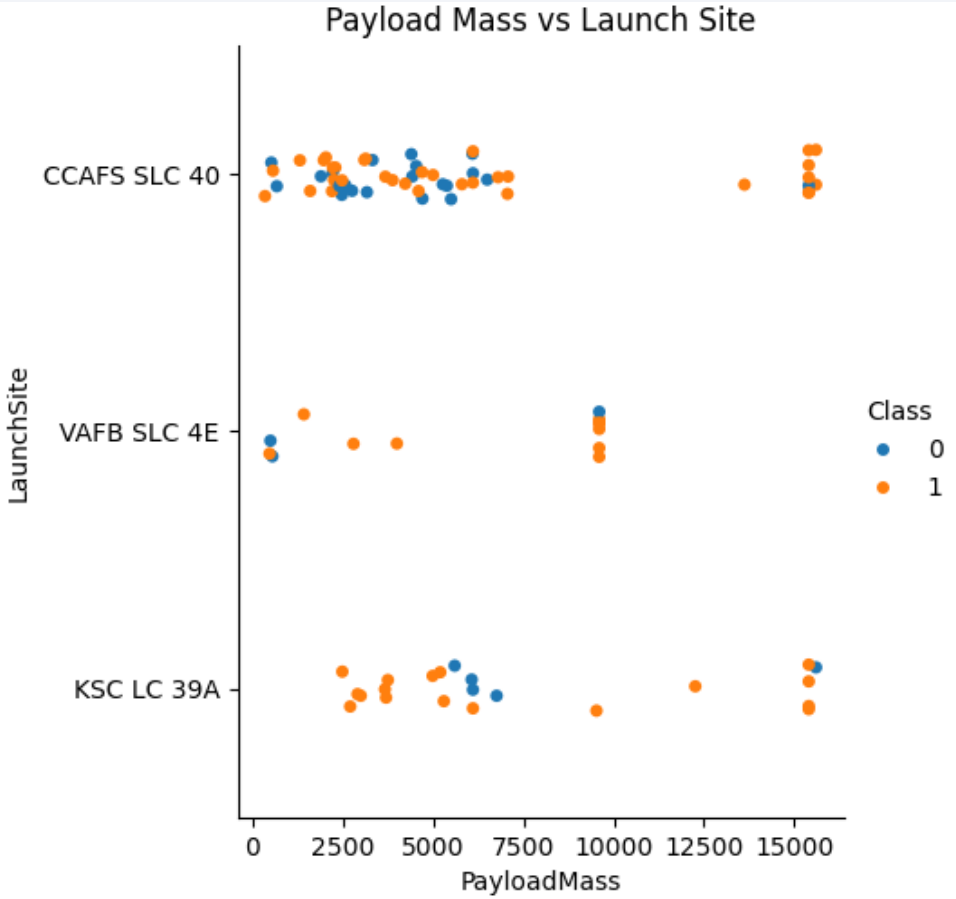
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

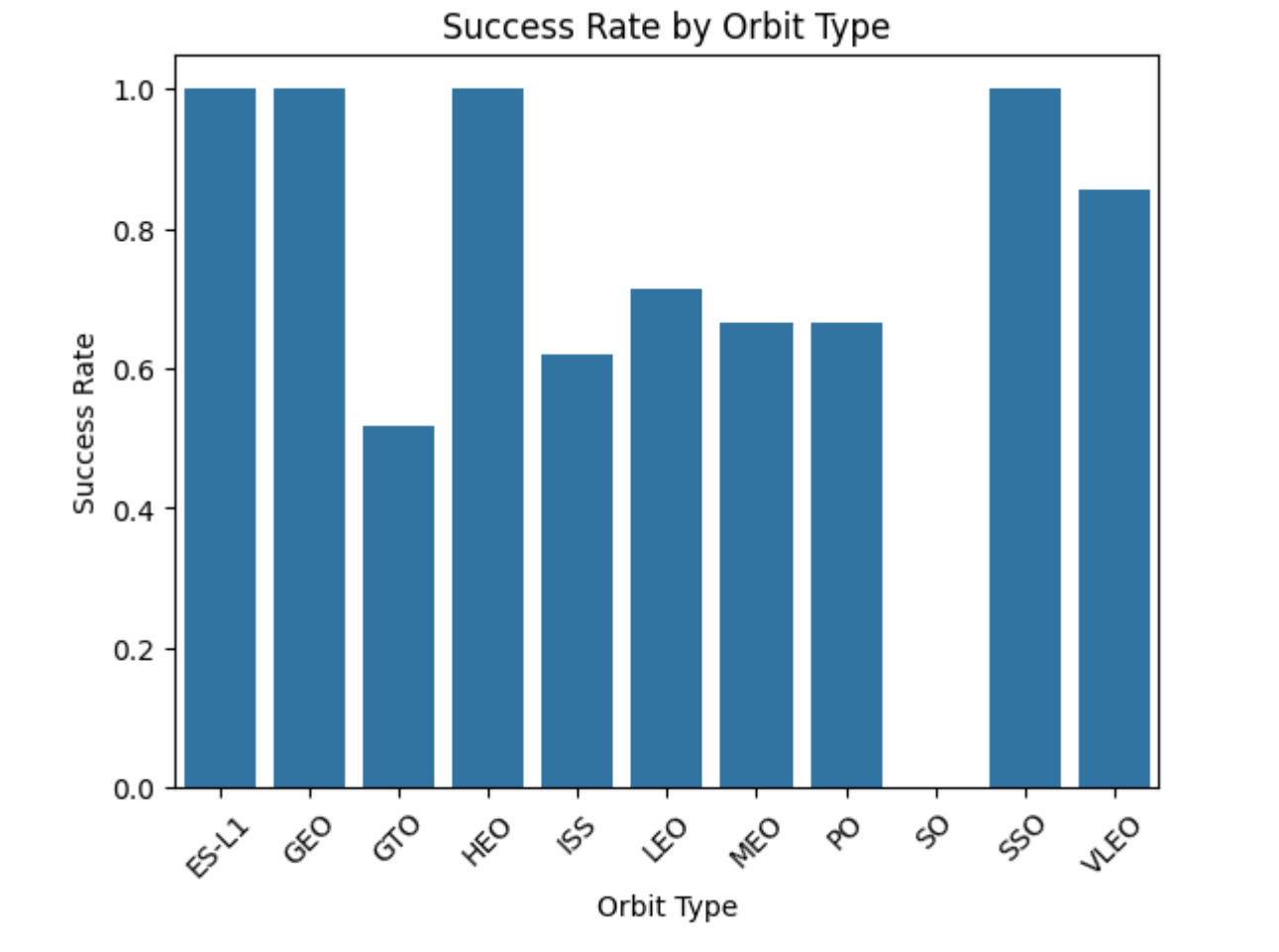


Payload vs. Launch Site

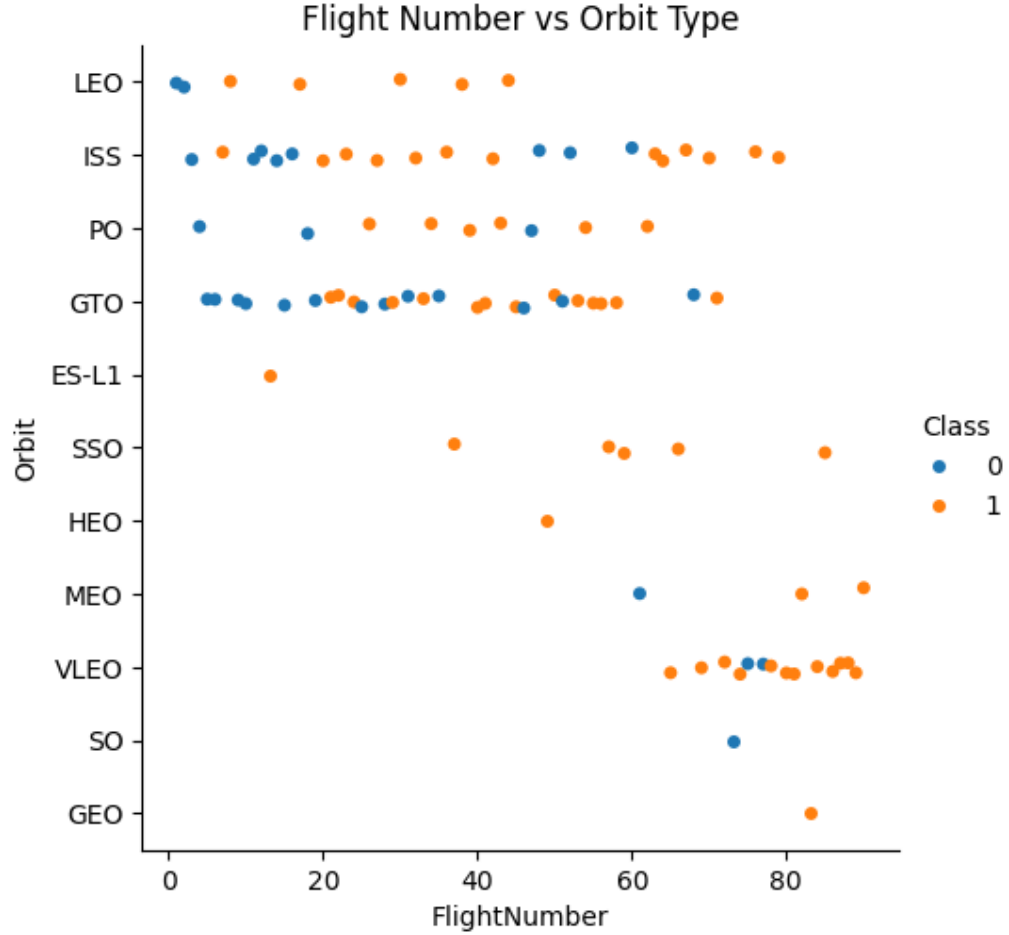


Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

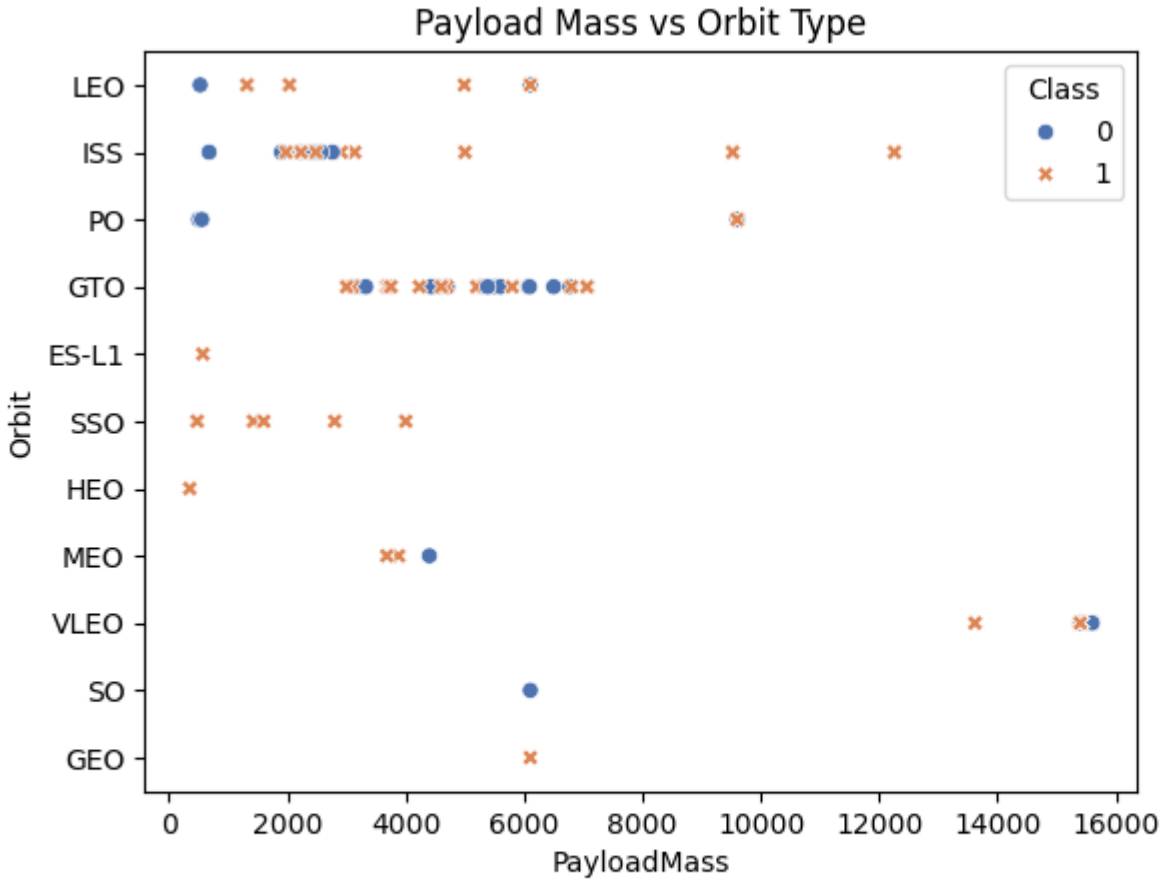


Flight Number vs. Orbit Type



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

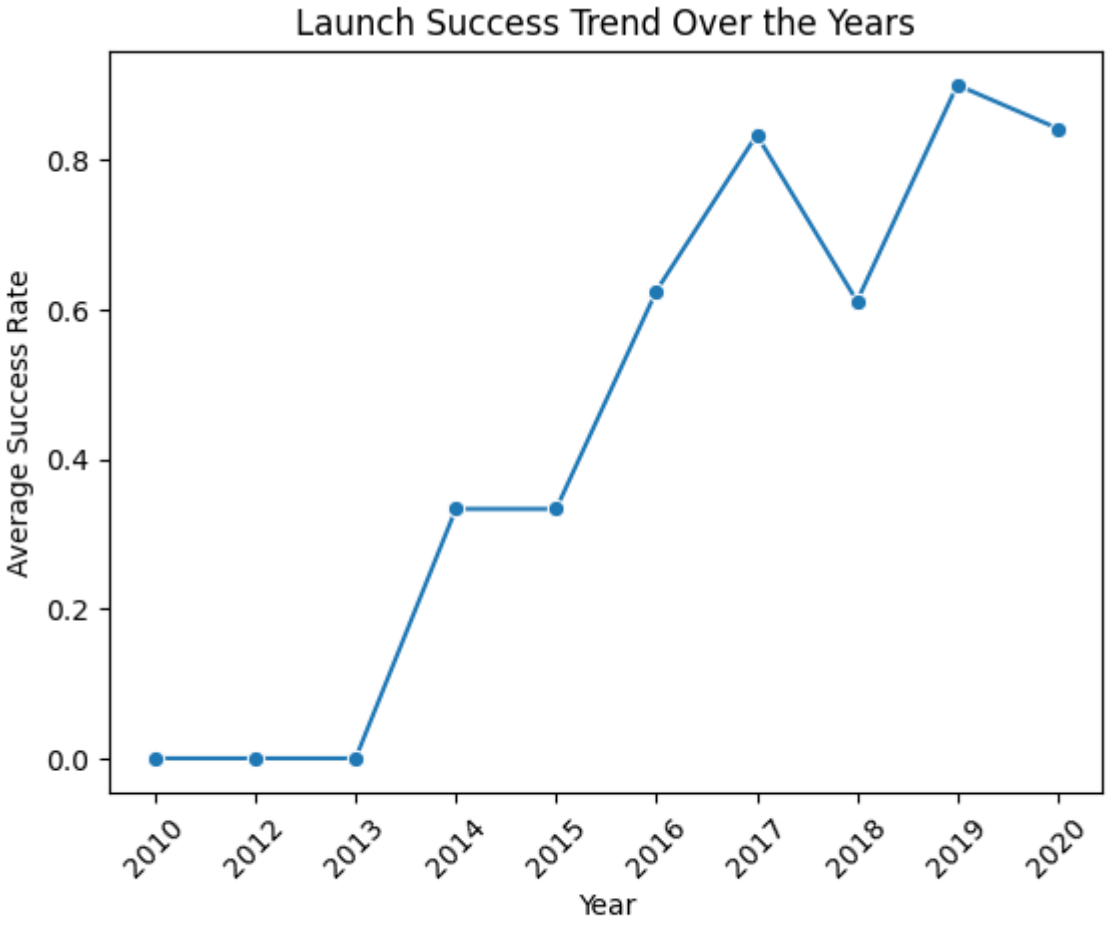
Payload vs. Orbit Type



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

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend



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

All Launch Site Names

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

In [10]:
```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

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

Out[10]:

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

Launch Site Names Begin with 'CCA'

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

In [11]:
```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[11]:

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

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

In [14]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass FROM SPACEXTABLE WHERE Customer LIKE '%NASA (CRS)%';
```

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

Out[14]:

Total_Payload_Mass
48213

Average Payload Mass by F9 v1.1

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

In [17]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';
```

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

Out[17]:

Average_Payload_Mass
2928.4

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) from SPACEXTBL where Landing_Outcome = 'Success (ground pad)'
```

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

Out[12]:

min(DATE)
01/08/2018

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 [22]:
```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND Payload_Mass__kg_ BETWEEN 4
```

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

Out[22]:

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

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [23]:
```
%sql SELECT Mission_Outcome, COUNT(*) as Total_Count FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

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

Out[23]:

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

Boosters Carried Maximum Payload

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

In [28]:
```
%sql SELECT booster_version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_= (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
```

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

Out[28]:

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

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [29]:
```
%sql SELECT substr(Date, 6, 2) AS month, landing_outcome, booster_version, launch_site FROM SPACEXTABLE WHERE substr(Date, 1
```

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

Out[29]:

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

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 outcome_count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GR(
```

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

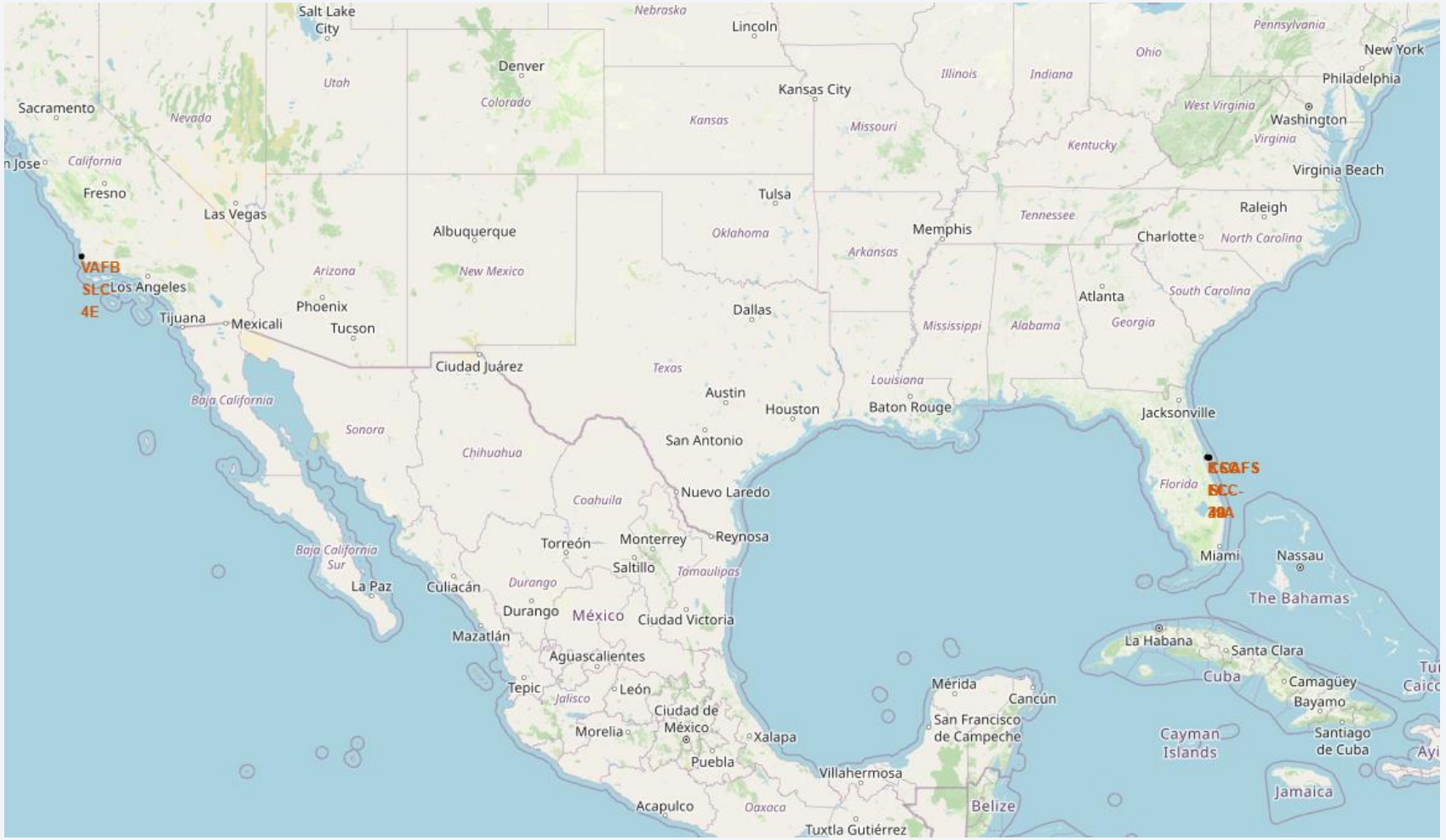
Out[30]:

Landing_Outcome	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

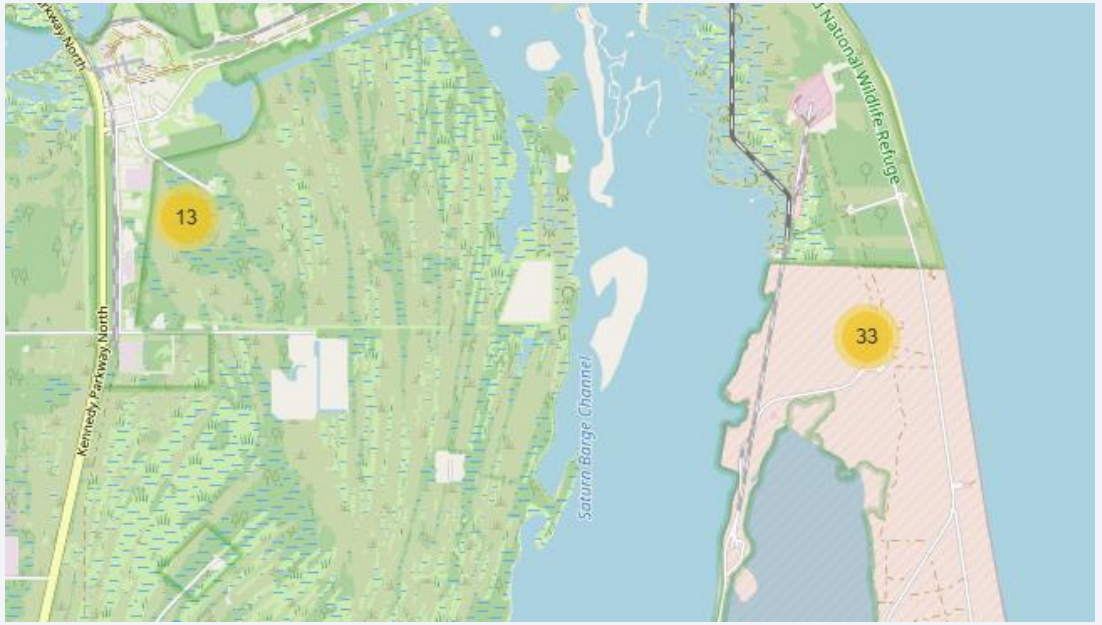
Section 3

Launch Sites Proximities Analysis

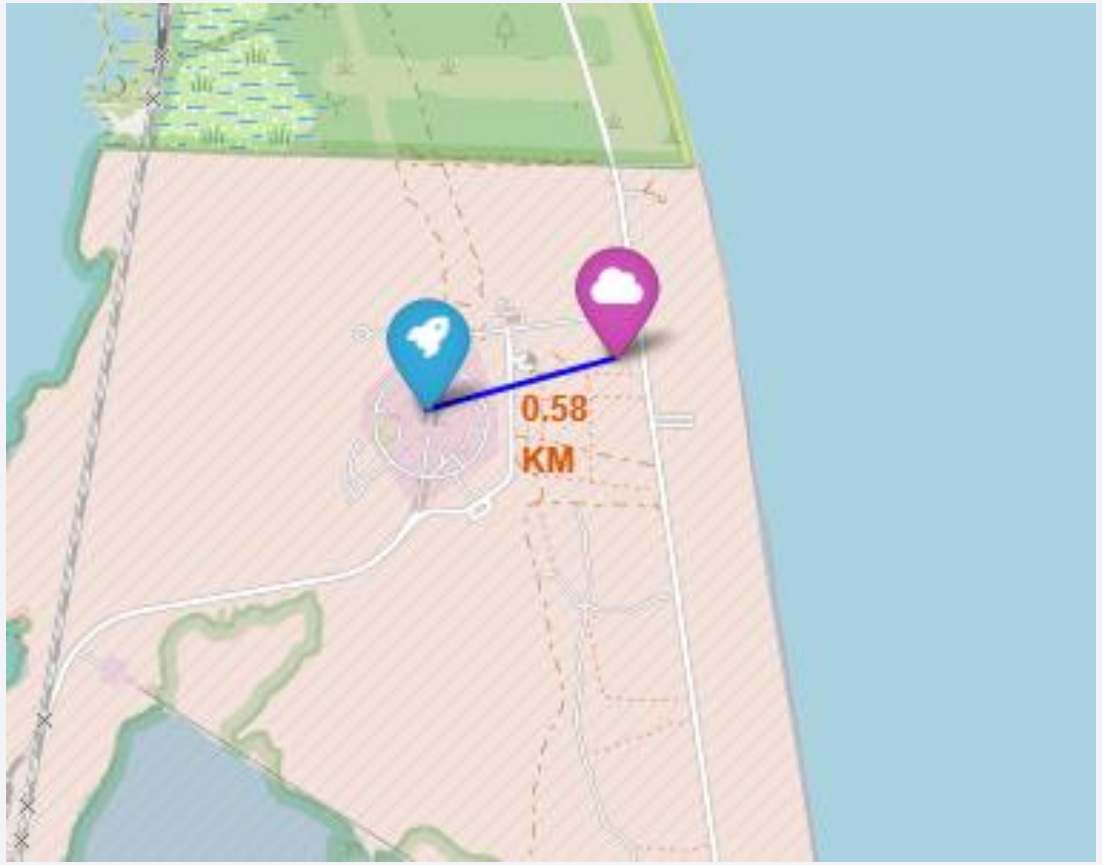
All launch sites on global map

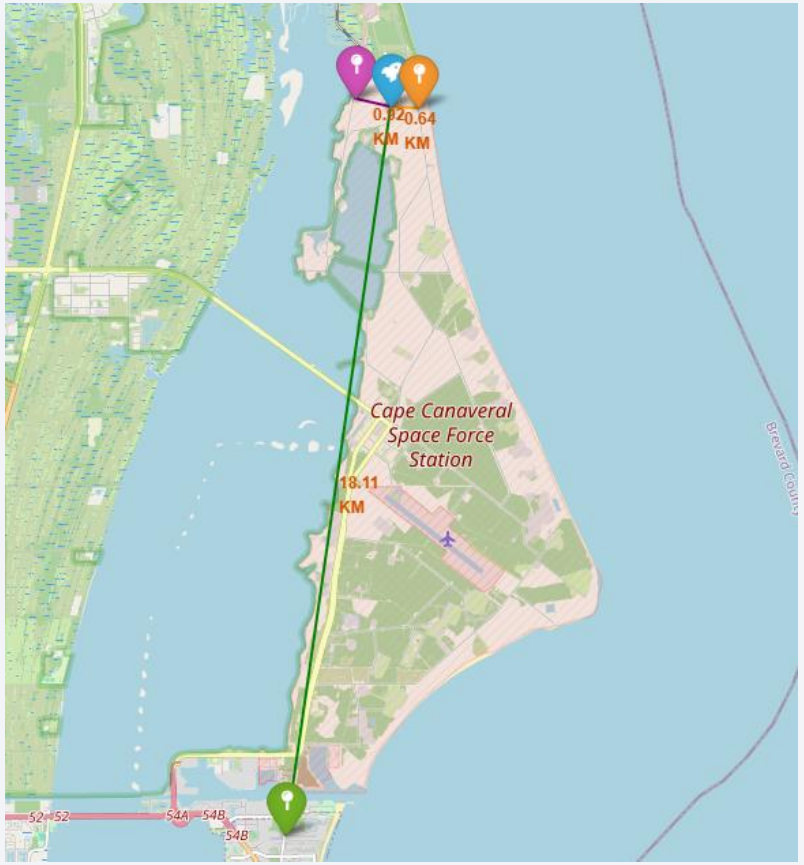


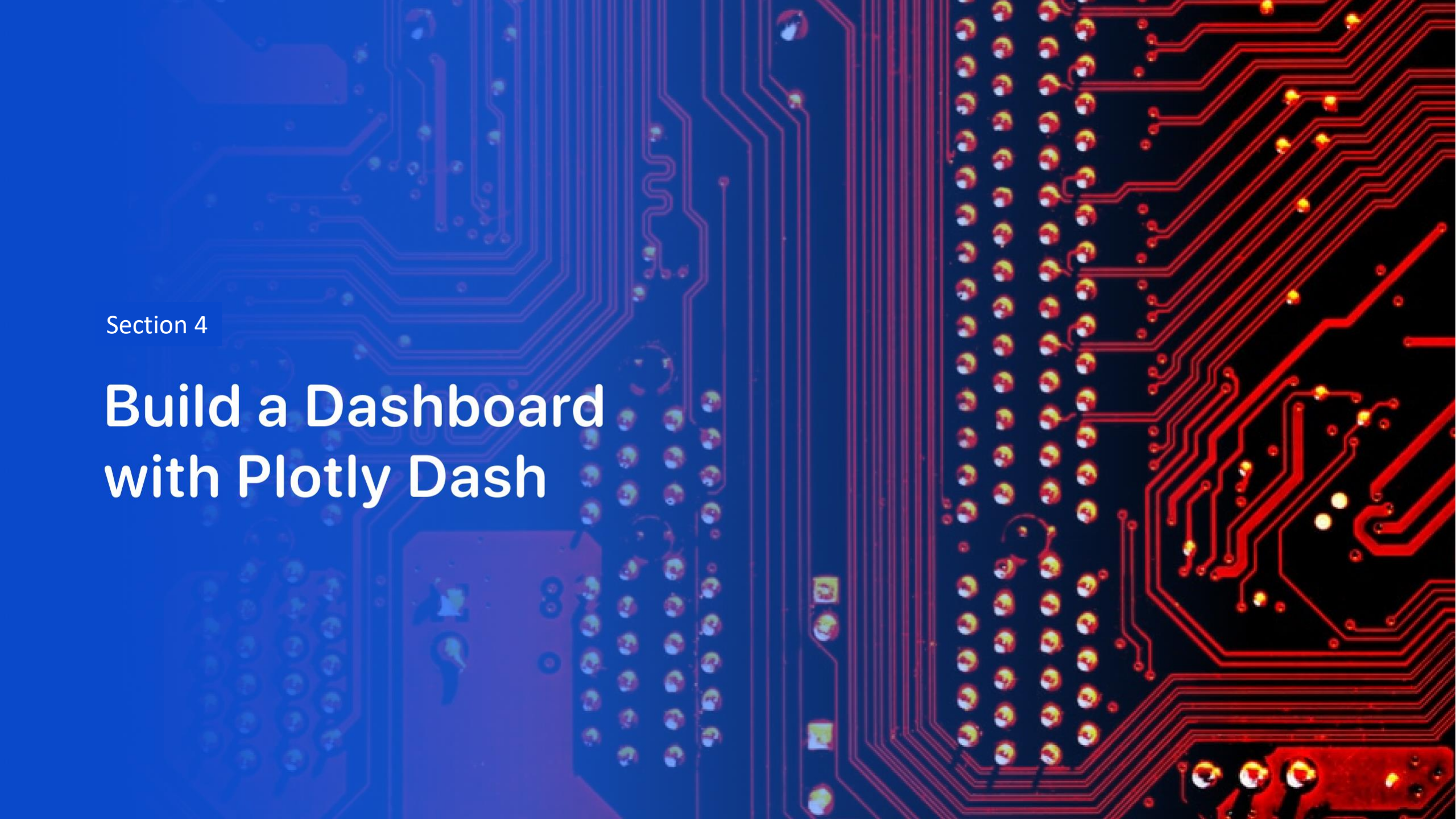
Launch outcomes for each site on the map With Color Markers



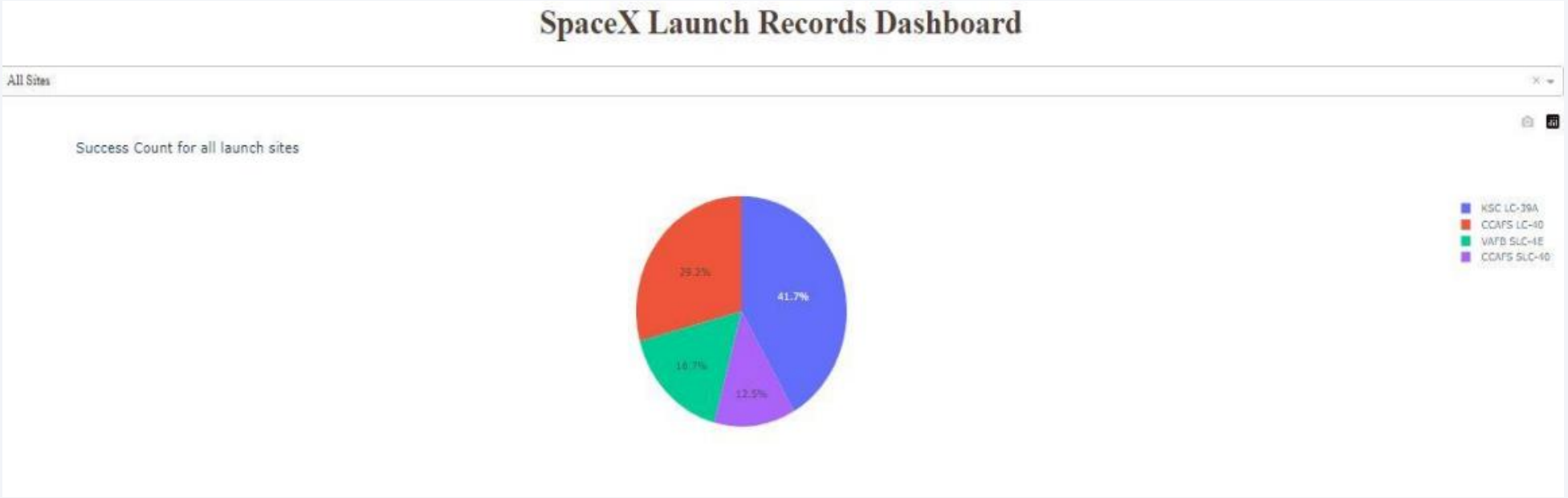
Distances between a launch site to its proximities





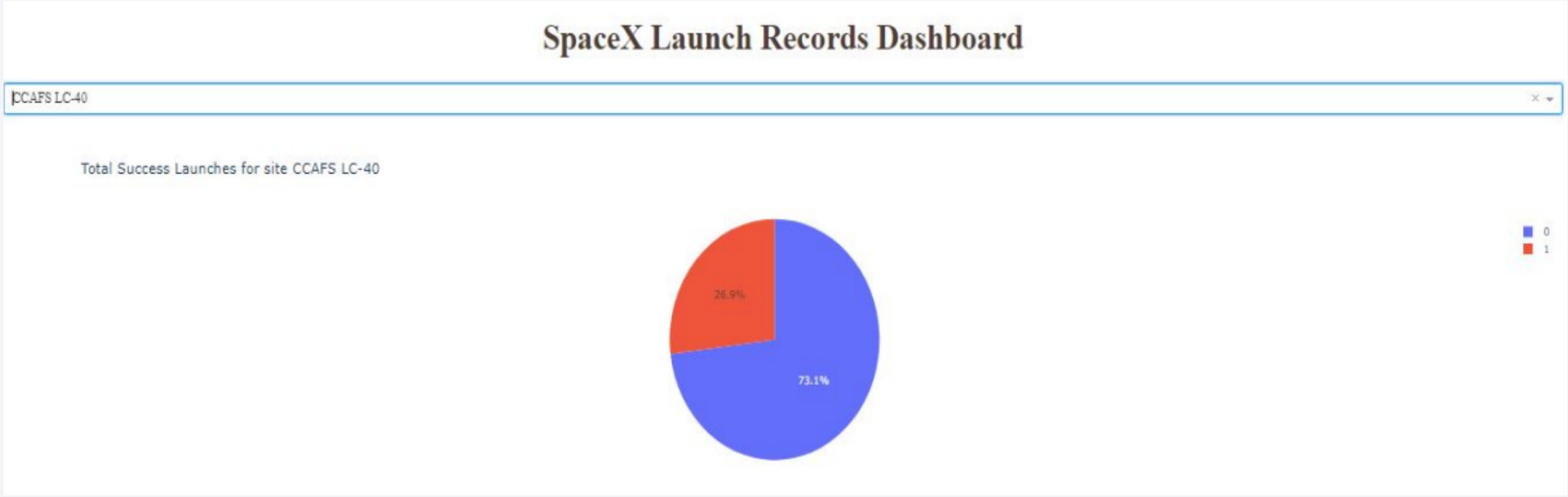

Section 4
Build a Dashboard with Plotly Dash

Pie Chart for launch success count for all sites



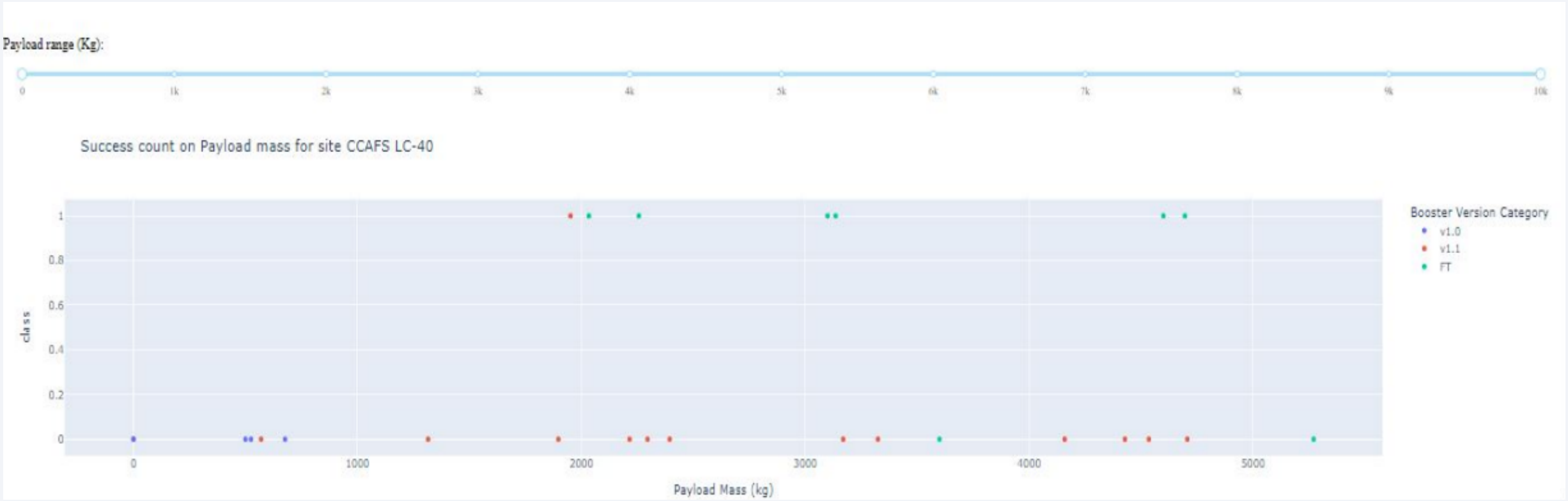
- Launch site KSC LC-39A has the highest launch success rate at 42% followed by CCAFS LC-40 at 29%, VAFB SLC-4E at 17% and lastly launch site CCAFS SLC-40 with a success rate of 13%

Pie chart for the launch site with 2nd highest launch success ratio



- Launch site CCAFS LC-40 had the 2nd highest 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 rate from a payload mass of >2000kg

Section 5

Predictive Analysis (Classification)

Classification Accuracy

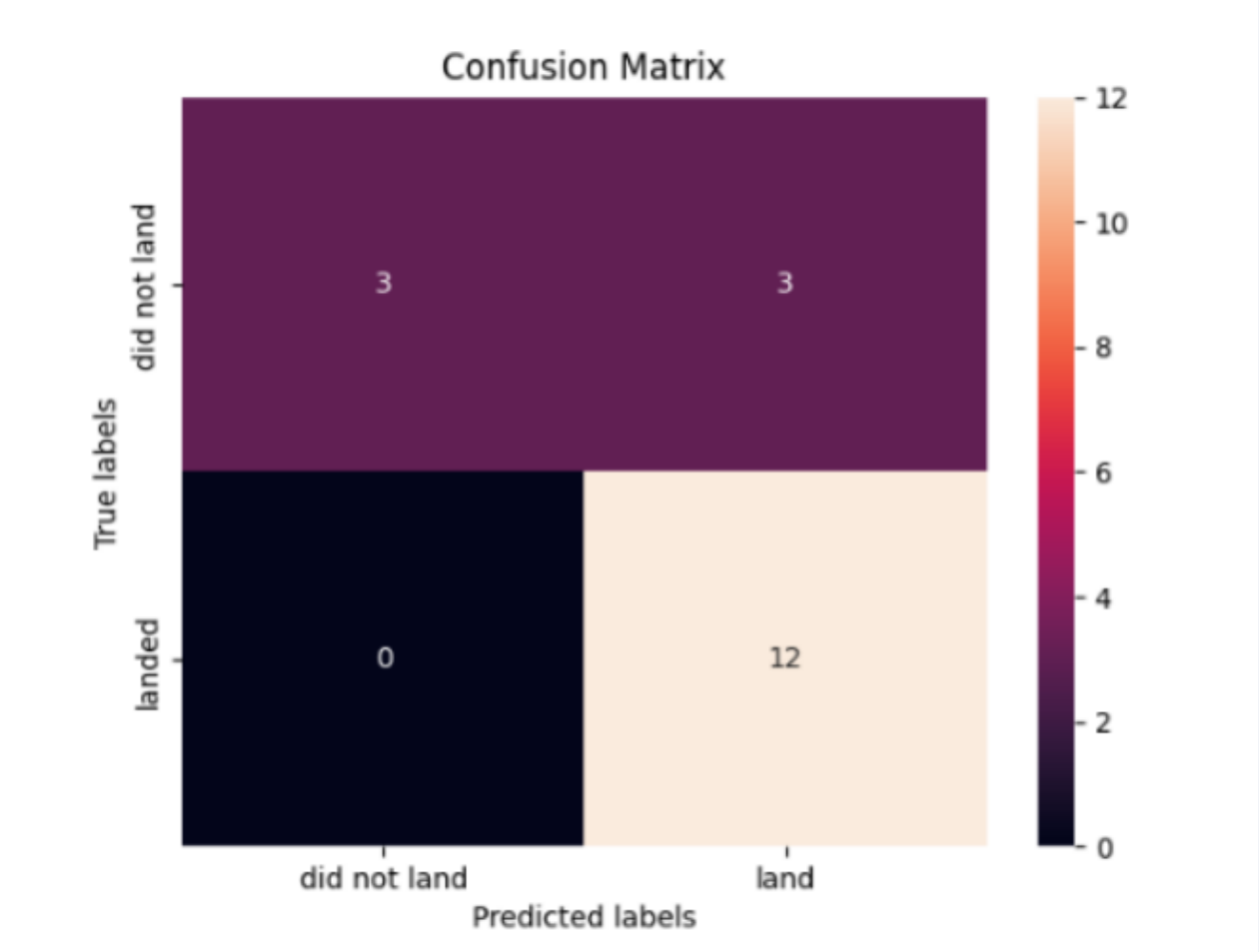
Out[68]:

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

All the methods perform equally on the test data: i.e. They all have the same accuracy of 0.833333 on the test Data

Confusion Matrix

- All the 4 classification model had the same confusion matrixes and were able equally distinguish between the different classes. The major problem is false positives for all the models



Conclusions

- Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
- We can deduce that, as the flight number increases in each of the 3 launcg sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80th flight
- 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).
- Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.
- 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

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!