IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Faith Tobore Edafetanure-Ibeh
31 July 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 Ploty Dash
  - 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 flights for 62 million dollars; other suppliers charge upwards of 165 million dollars for each flight. A large portion of the savings is due to SpaceX's ability to reuse the first stage. In order to calculate the price of a launch, we must first decide if the first stage will land. If another business want to submit a counterbid to SpaceX for a rocket launch, it may do so using this information.

- Problems you want to find answers

  In this capstone, 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:
  - 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

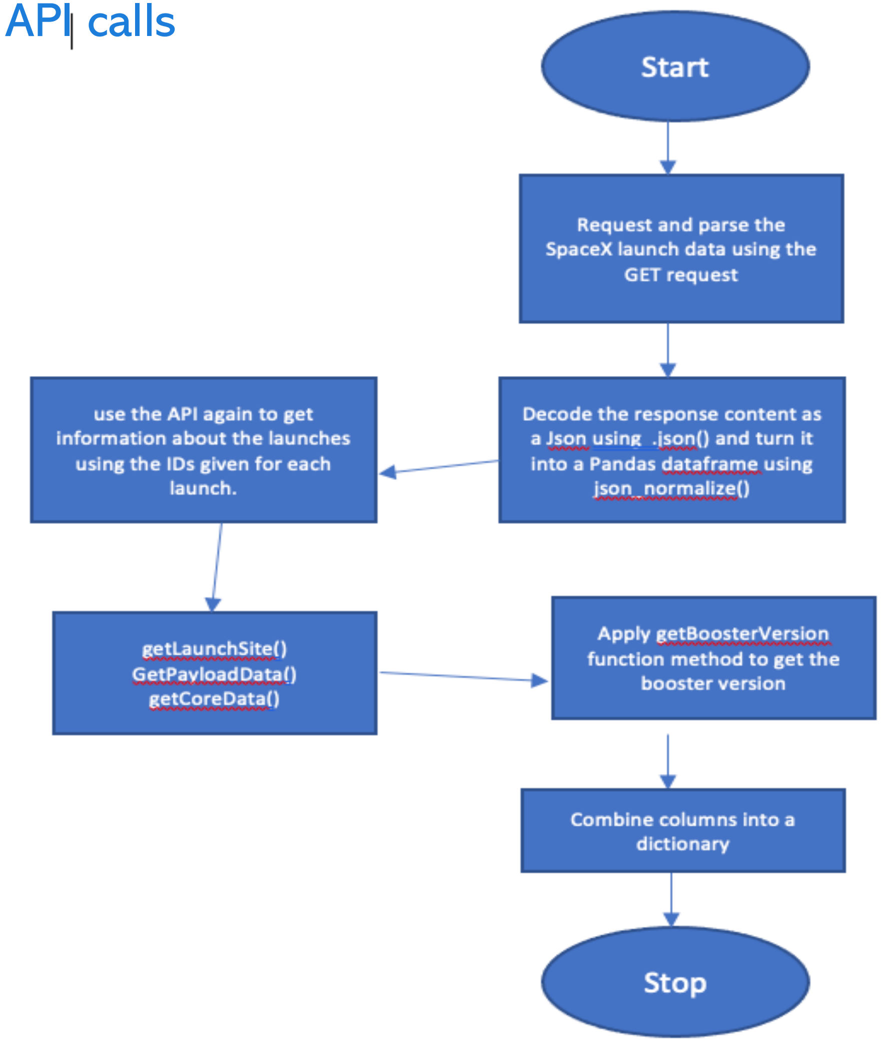
- **Description of how SpaceX Falcon9 data was collected.**
  - Data was initially gathered by sending a get request to the SpaceX API, a RESTful API, using the SpaceX API. In order to accomplish this, a series of helper functions that would aid in the use of the API to extract information utilizing identifying numbers in the launch data were first defined. Following that, the SpaceX API url was requested to obtain rocket launch data.
  - The SpaceX launch data was finally requested and parsed via the GET request, and the return content was then decoded as a JSON result before being turned into a Pandas data frame. This was done to improve the consistency of the requested JSON results.
  - Additionally, web scraping was done to get historical Falcon 9 launch data from a Wikipedia article named "List of Falcon 9 and Falcon Heavy launches," where the data is saved in HTML. I parsed the table and transformed it into a Pandas data frame using BeautifulSoup and request Libraries to extract the Falcon 9 launch HTML table records from the Wikipedia page.

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 combined into a dictionary

- Github link **to SpaceX API calls notebook**

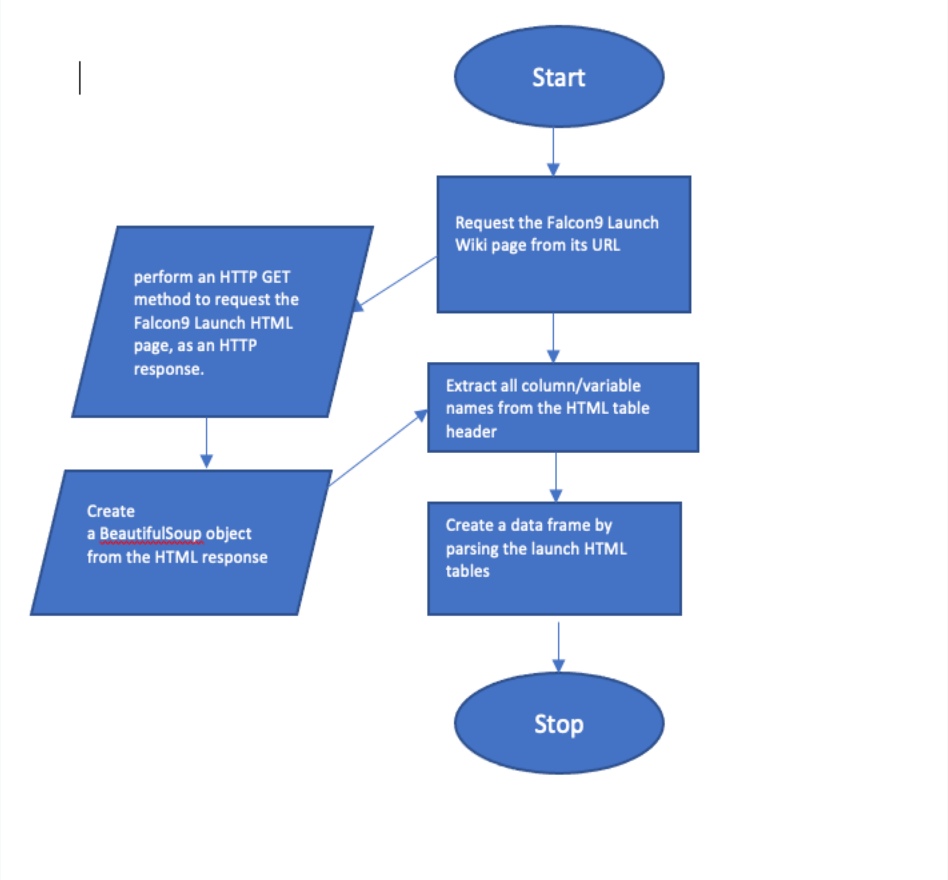
Flowchart of SpaceX API calls



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.
- [Here]() is the GitHub URL of the completed web scraping notebook.

Flowchart of web scraping



Data Wrangling

- Data was filtered using the BoosterVersion column to only keep the Falcon 9 launches after acquiring and generating a Pandas DF from the gathered information. Next, the missing data values in the LandingPad and PayloadMass columns were handled with. The mean value of the column was used to fill in the missing data values for the PayloadMass.
- Additionally, some exploratory data analysis (EDA) was done to identify trends in the data and choose a label for supervised model training.
- Here is the GitHub URL of the completed data wrangling related notebooks.

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`:

```
# 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()
```

```
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

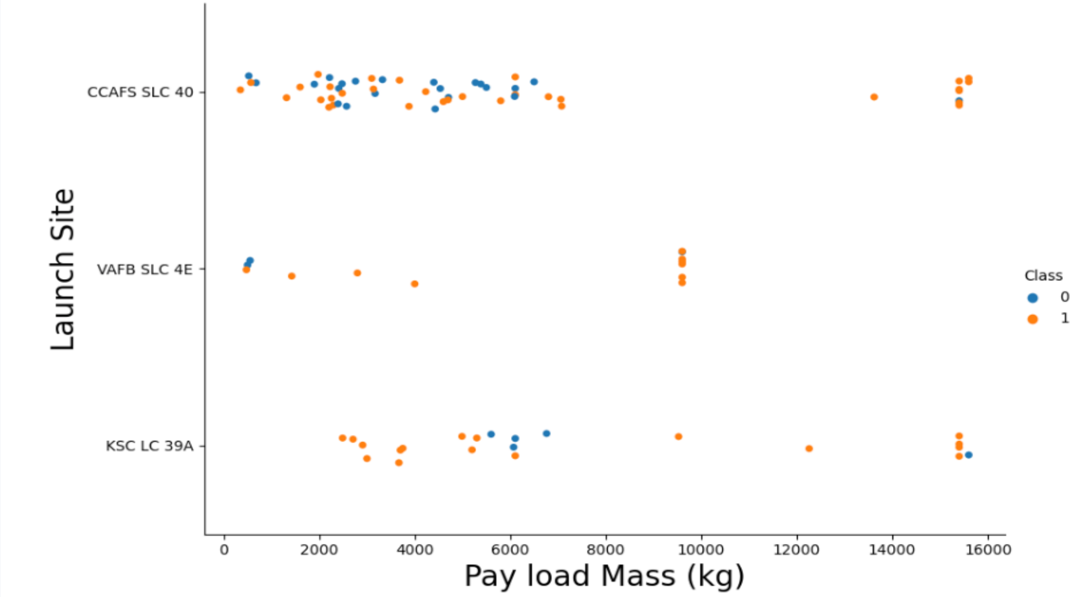
```
landing_class=df['Class']
df[['Class']].head(8)
```

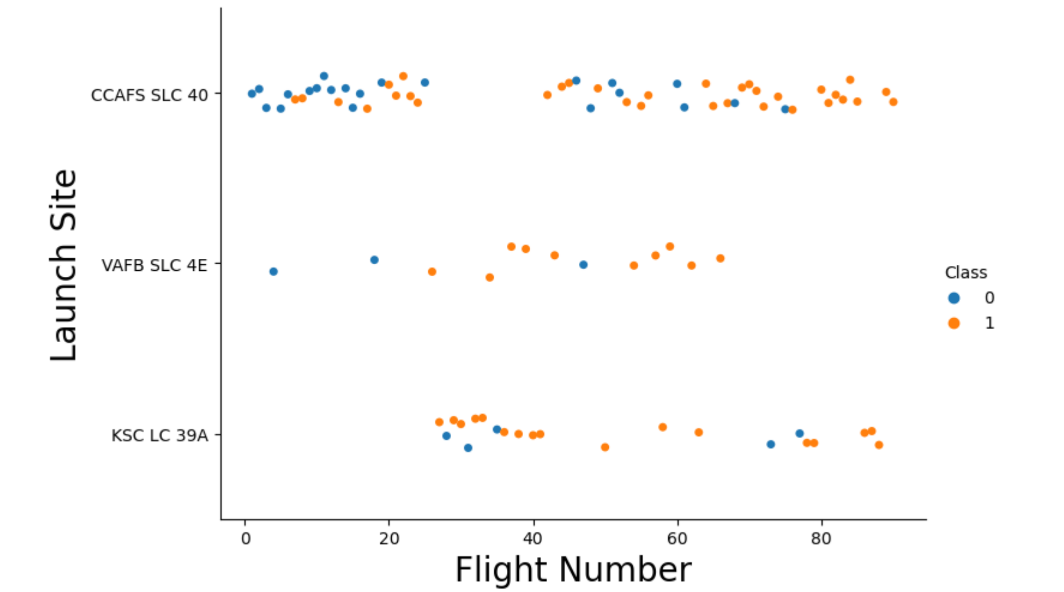
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

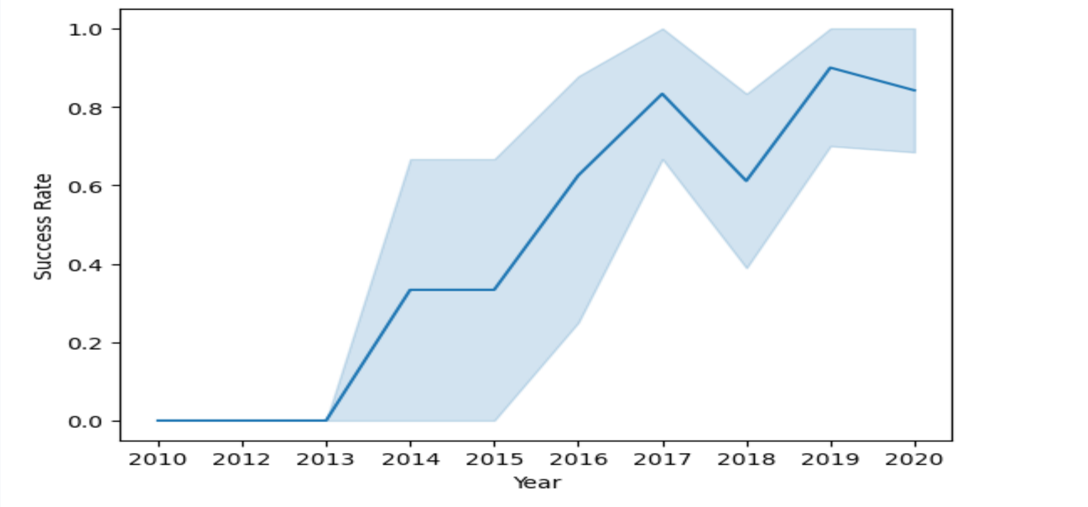
EDA with Data Visualization

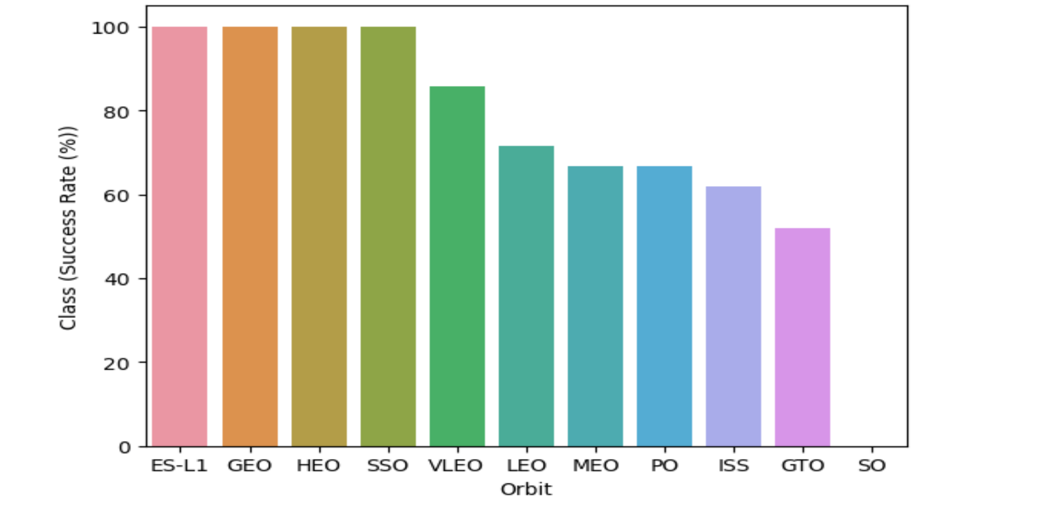
- Performed data Analysis and Feature Engineering using Pandas and Matplotlib. i.e.
  - Exploratory Data Analysis
  - Preparing Data Feature Engineering
- Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, FlightNumber and Orbit type, Payload 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.
- Here is the GitHub URL of your completed EDA with data visualization notebook

EDA with Data Visualization (Plots)









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 as "Launch_Sites" FROM SPACEXTBL;
```

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

```
%sql SELECT * FROM 'SPACEXTBL' 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(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA
```

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

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

EDA with SQL

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

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

- 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 DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)" AND
```

- List the total number of successful and failure mission outcomes

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

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

```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX
```

- 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.

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

- Here is the GitHub URL of the completed EDA with SQL notebook

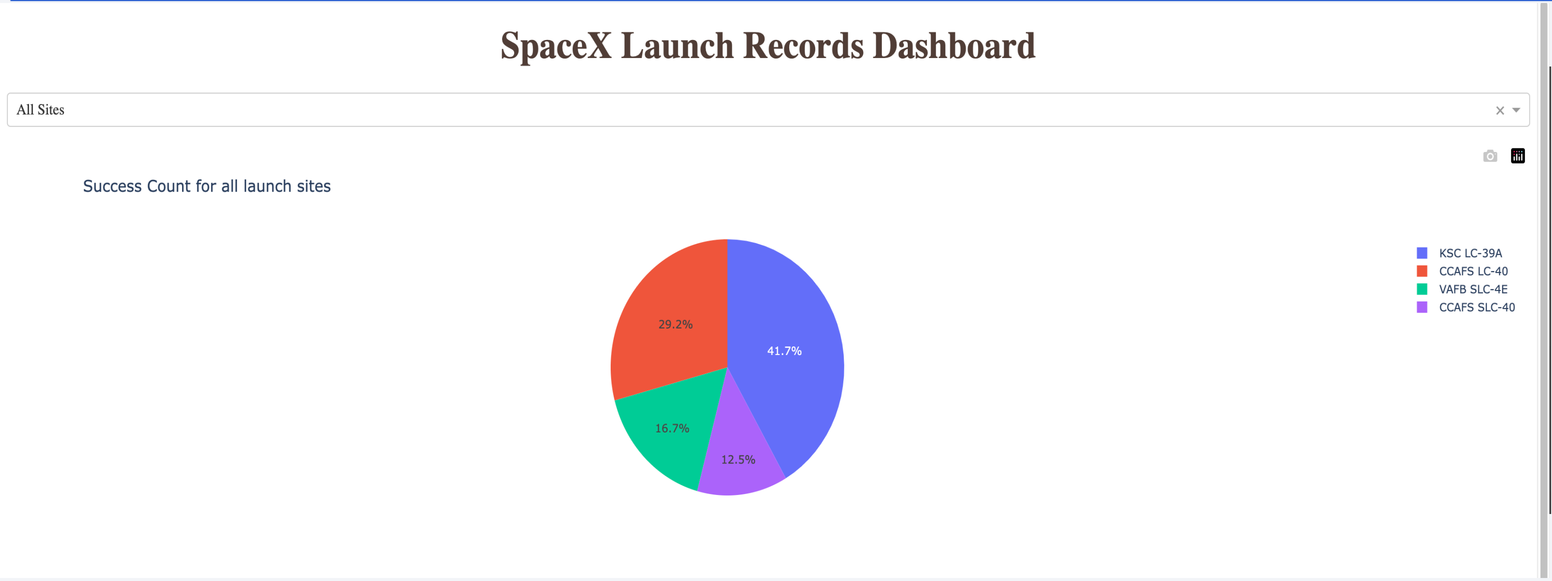
Build an Interactive Map with Folium

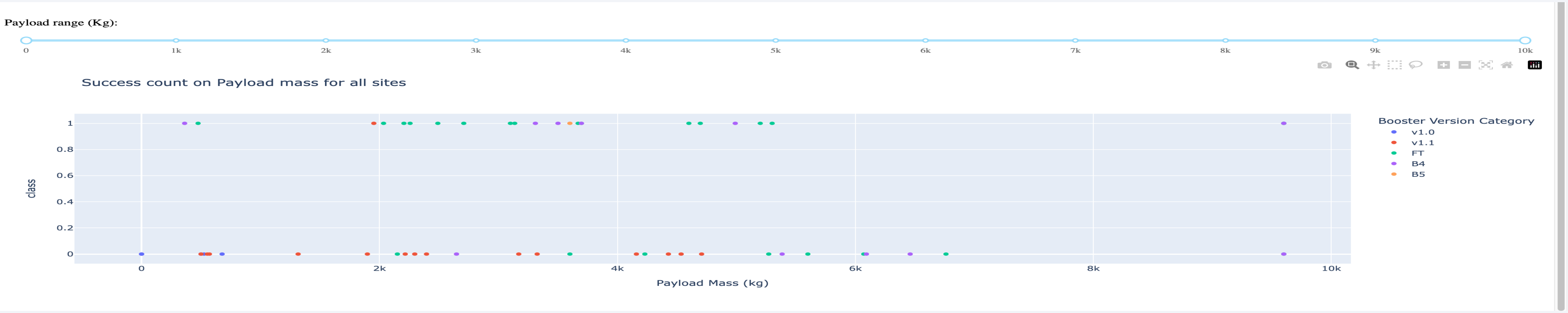
- Developed a folium map to indicate each launch site, and created map items like markers, circles, and lines to show each launch site's success or failure with launches.
- Those objects were added to mark specific locations on the map.
- Created a launch set outcomes (failure=0 or success=1).
- Here is the GitHub URL of the completed interactive map with Folium map

Build a Dashboard with Plotly Dash

- Built an interactive dashboard application with Plotly dash, I:
  - Added a Launch Site Drop-down Input Component
  - Added a callback function to render success-pie-chart based on selected site dropdown
  - Added a Range Slider to Select Payload
  - Added a callback function to render the success-payload-scatter-chart scatter plot
- Here is the GitHub URL of your completed Plotly Dash lab

Space X Dashboard with Plotly Dash





Predictive Analysis (Classification)

Summary of how I built, evaluated, improved, and found the best performing classification model

- I began my exploratory data analysis after loading the data as a Pandas Dataframe. I used the method to_numpy() to create a NumPy array from the column Class in the data, which I then assigned to the variable Y as the outcome variable.
- After that, normalize the feature dataset (x) by preprocessing it.Sklearn's StandardScaler() function.
- Then, using the function train_test_split from sklearn.model_selection and setting the test_size and random_state parameters to 0.2 and 2, the data were divided into training and testing sets.

- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

- In order to compare the performance of SVM, Classification Trees, k closest neighbors, and Logistic Regression using the test data, the best machine learning model or approach should be found.
- Before creating a GridSearchCV object and giving it a set of parameters for each model, each algorithm must first have its own object.
- The GridsearchCV object was made with cv=10 for each of the models being evaluated, and the training data was then fitted into each object to see which hyperparameter was the best.
- We generated GridSearchCV objects for each of the models once they had been fitted to the training set. These objects then reported the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data element best_score_.
- Finally, each model's accuracy on the test data was calculated using the method score, and a confusion matrix was generated for each using the test and predicted results.

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;

All the methods have the same accuracy 0.833333

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

GitHub URL of the completed predictive analysis lab

Results

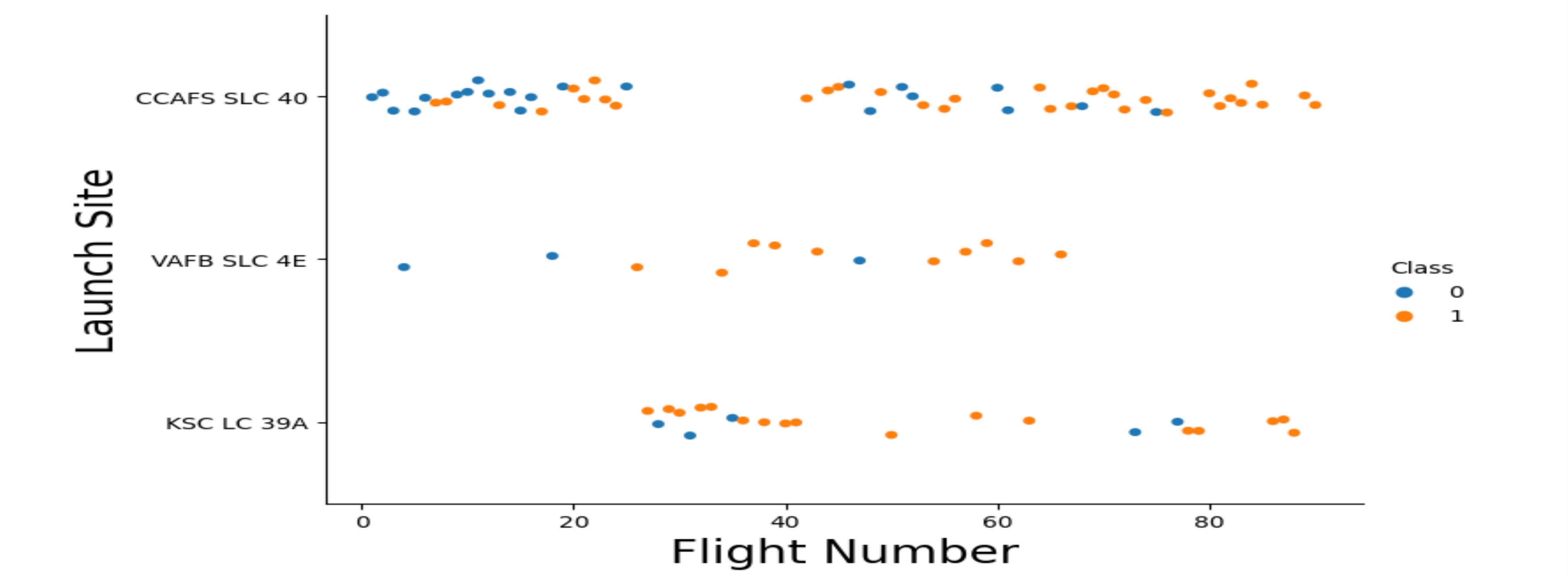
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

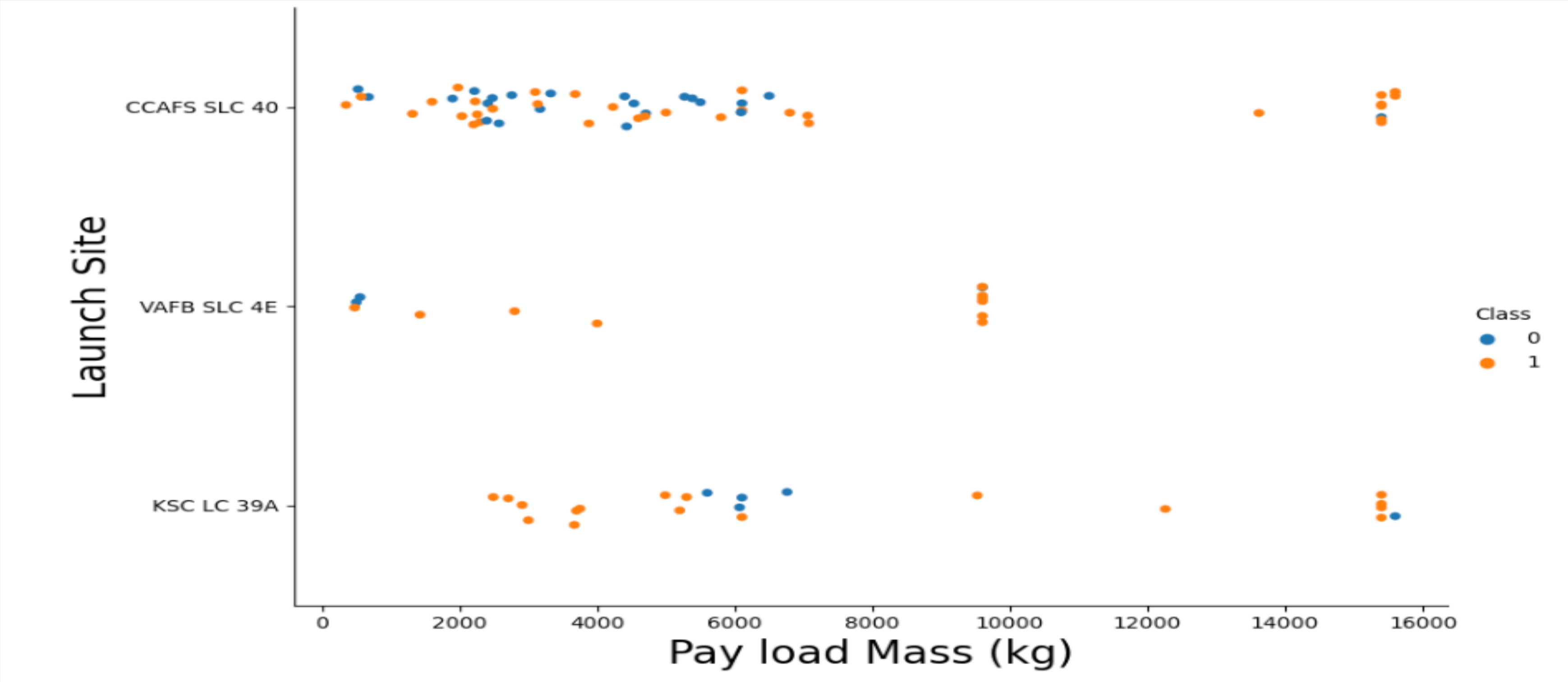
A scatter plot of Flight Number vs. Launch Site



We can infer that the success rate grows when more flights are launched from each of the three launch sites. For instance, the VAFB SLC 4E launch site had a 100% success rate following Flight 50. After the eighty-first flight, both KSC LO 39A and CCAFS SLO 40 had 100% success rates.

Payload vs. Launch Site

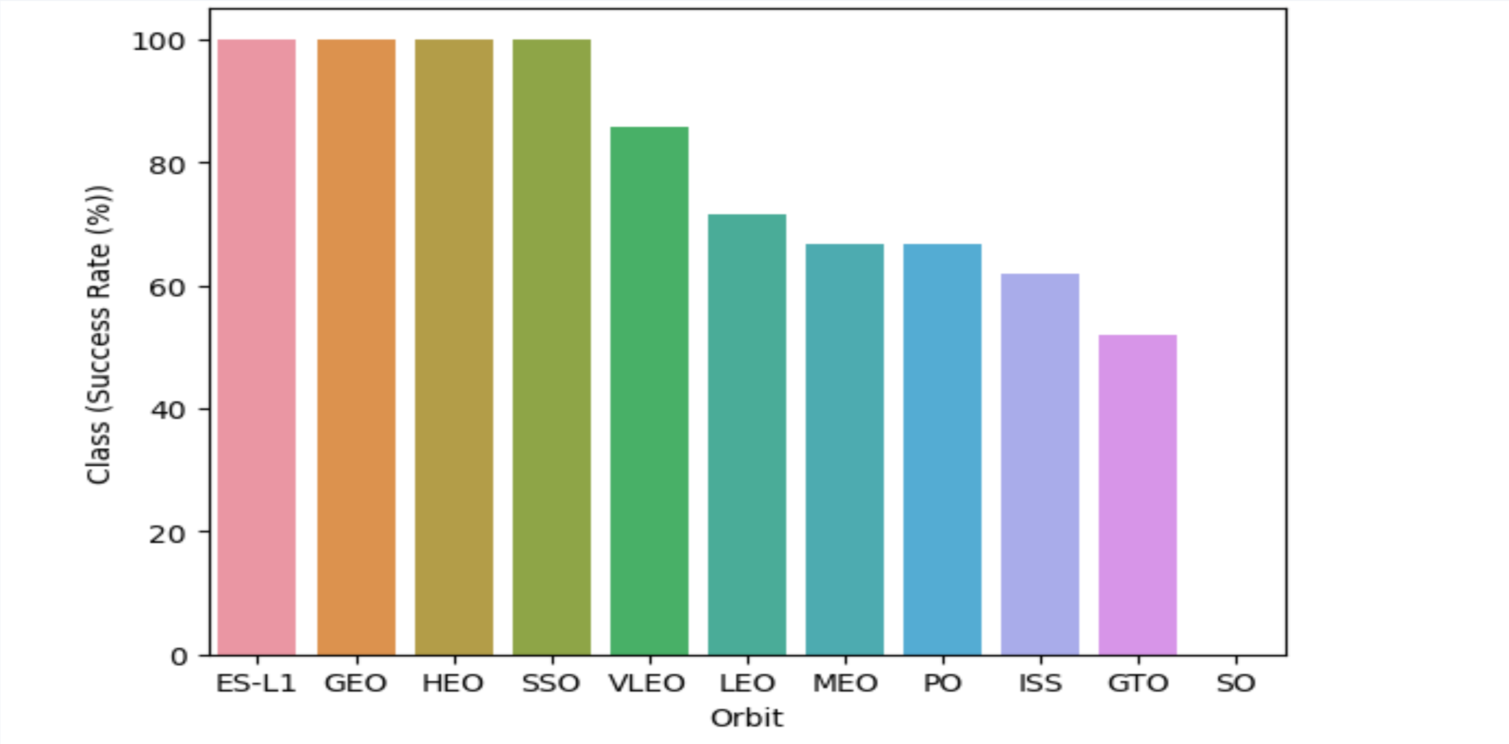
A scatter plot of Payload vs. Launch Site



Currently, no rockets with heavy payload masses (more than 10,000) have been launched from the VAFB-SLC launch site, according to the Payload vs. Launch Site scatter point chart.

Success Rate vs. Orbit Type

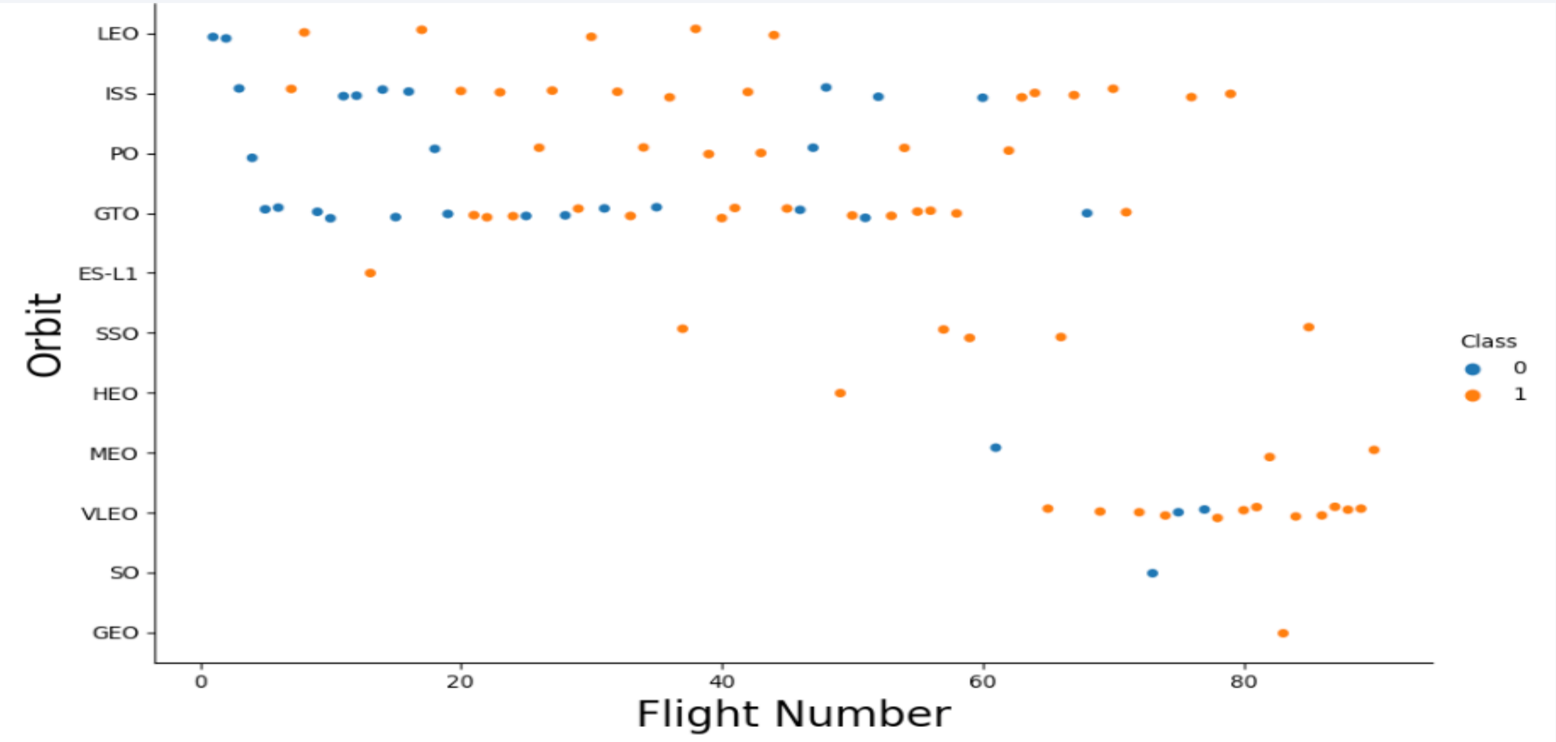
Bar Chart showing Success Rate vs. Orbit Type



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.

Flight Number vs. Orbit Type

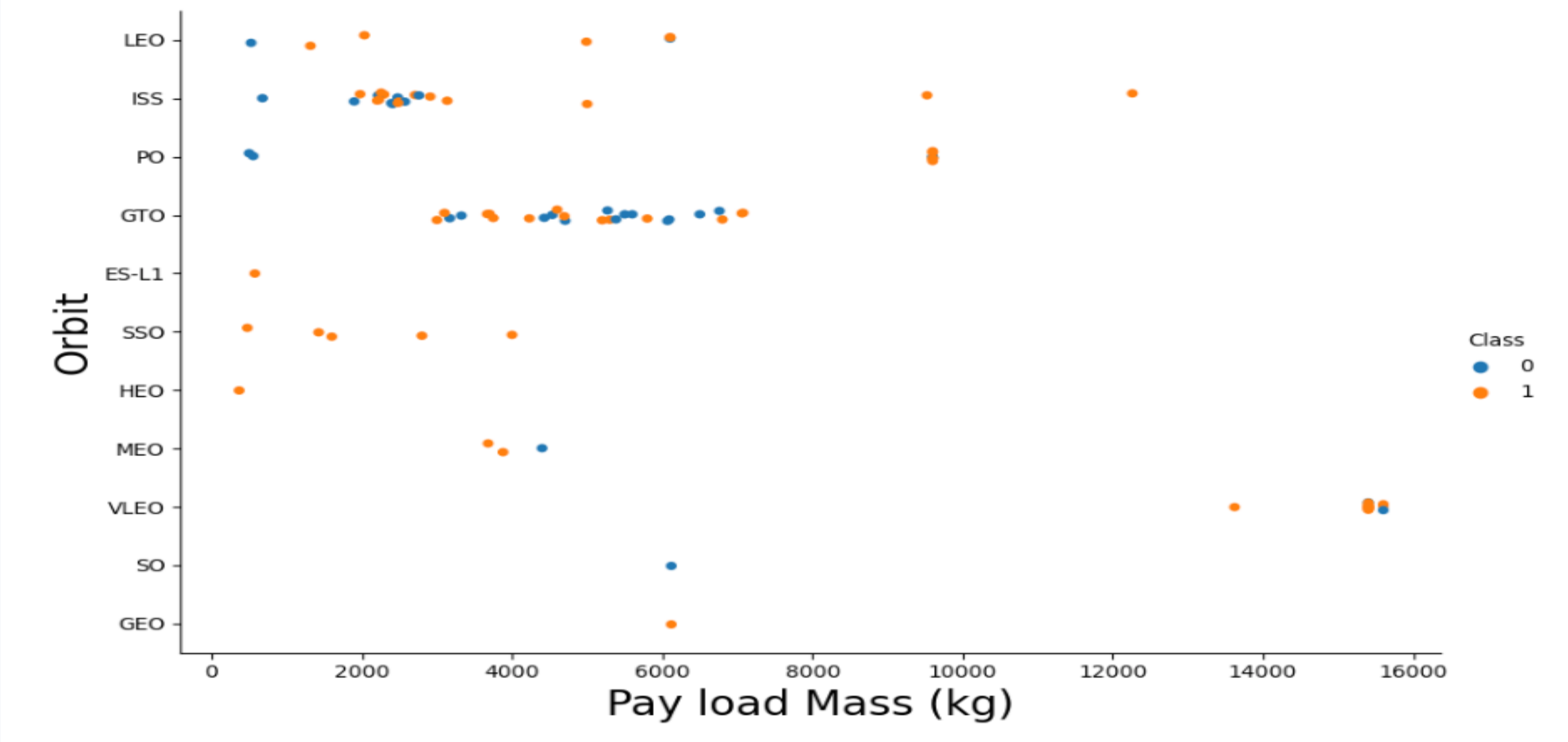
A scatter plot of Flight number vs. Orbit type



You should observe that success seems to be correlated with the number of flights when in LEO orbit, but there doesn't seem to be a correlation when in GTO orbit.

Payload vs. Orbit Type

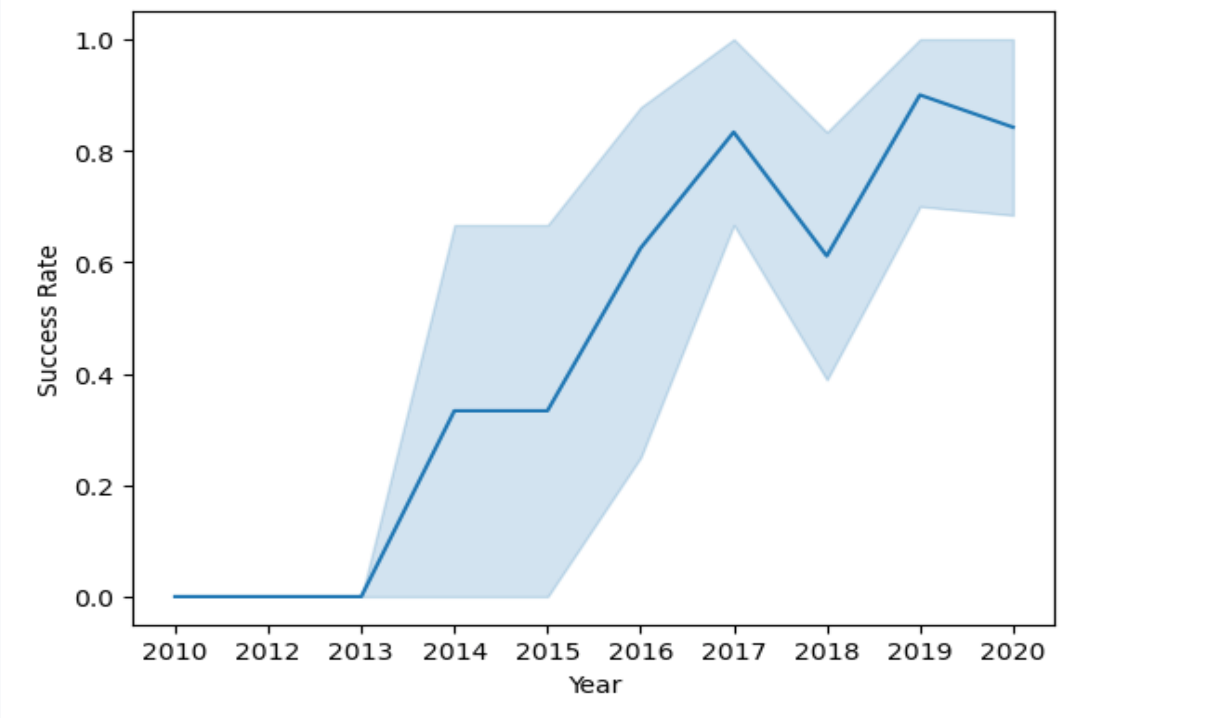
Scatter Plot showing Payload vs. Orbit Type



For polar, LEO, and ISS missions with substantial payloads, the successful landing or positive landing rate is higher.
However, for GTO, it is difficult to make this distinction because both the positive landing rate and the negative landing (missions that fail) are present.

Launch Success Yearly Trend

Yearly Launch Success Rate



You can observe that the success rate since 2013 kept increasing till 2020

All Launch Site Names

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

Task 1

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

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

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

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Used ‘LIKE’ command with ‘%’ wildcard in ‘WHERE’ clause to select and display a table of all records where launch sites begin with the string '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	Lan
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Fai
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Fai
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	

Total Payload Mass

- Calculate and Display the total payload carried by boosters from NASA
- Used the ‘SUM()’ function to return and display the total sum of ‘PAYLOAD_MASS_KG’ column for Customer ‘NASA(CRS)’

Task 3

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

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

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

Total Payload Mass(Kgs)	Customer
45596.0	NASA (CRS)

Average Payload Mass by F9 v1.1

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

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

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

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

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

- Find the dates of the first successful landing outcome on ground pad

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)
01/08/2018

- Used the ‘MIN()’ function to return and display 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

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 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 DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)" AND
```

```
 * 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 between 4000 – 6000 with landing outcome of ‘Success (drone ship)’.

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

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

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

[1]:

Mission_Outcome	Total
None	0
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

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

Task 8

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

```
2]: %sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX
```

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

2]:

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600.0
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600.0
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600.0
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600.0
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600.0
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600.0
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600.0
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600.0
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600.0
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600.0
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600.0
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600.0

- Using a Subquery 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

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

Task 9

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.

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

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

substr(Date,7,4)	substr(Date, 4, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing_Outcon
2015	10	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395.0	Success	Failure (drone shi
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898.0	Success	Failure (drone shi

- 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 matching the filter.

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

Task 10

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.

```
%sql SELECT * FROM SPACEXTBL WHERE "Landing_Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-20
```

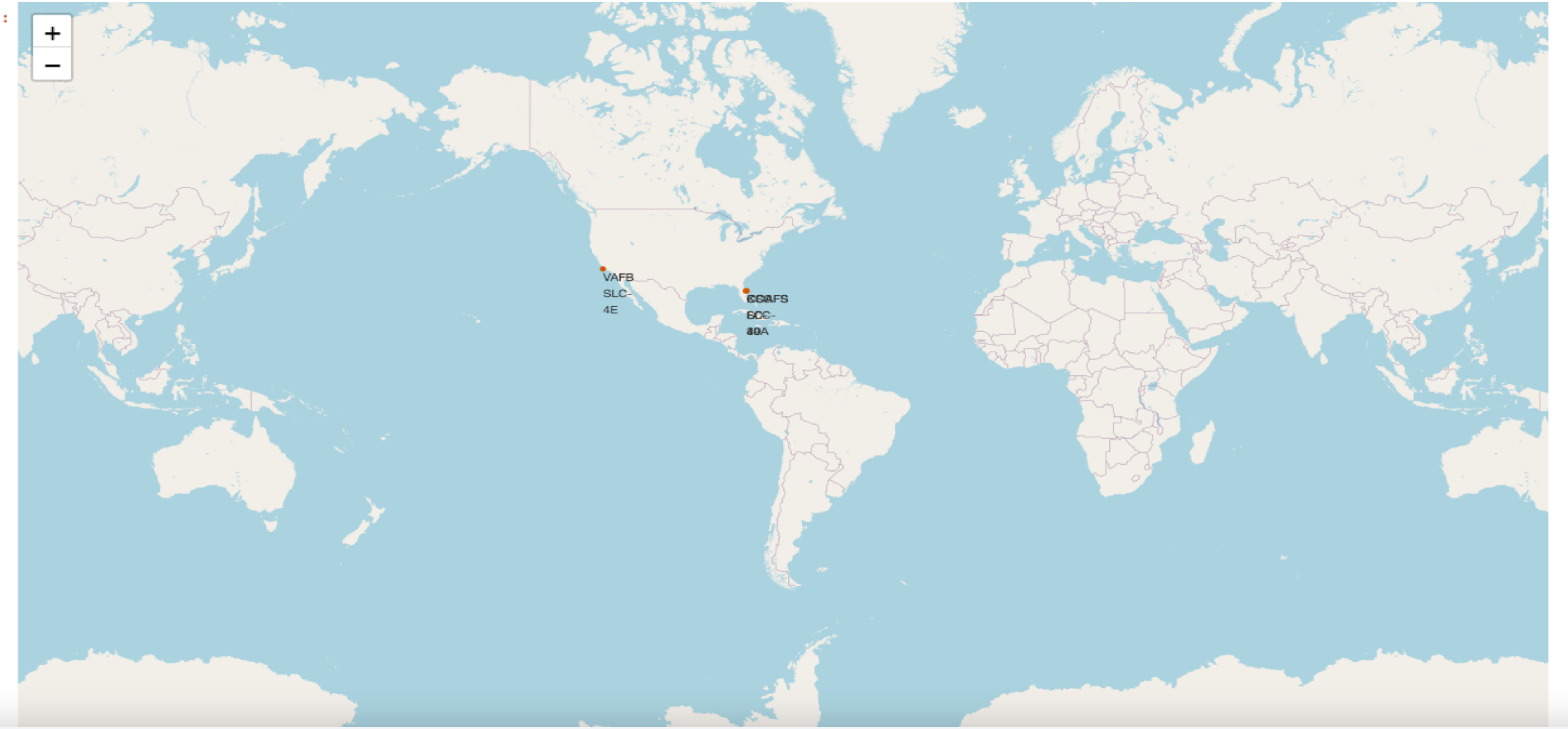
```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outco
19/02/2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490.0	LEO (ISS)	NASA (CRS)	Succ
18/10/2020	12:25:57	F9 B5 B1051.6	KSC LC-39A	Starlink 13 v1.0, Starlink 14 v1.0	15600.0	LEO	SpaceX	Succ
18/08/2020	14:31:00	F9 B5 B1049.6	CCAFS SLC-40	Starlink 10 v1.0, SkySat-19, -20, -21, SAOCOM 1B	15440.0	LEO	SpaceX, Planet Labs, PlanetIQ	Succ
18/07/2016	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257.0	LEO (ISS)	NASA (CRS)	Succ
18/04/2018	22:51:00	F9 B4 B1045.1	CCAFS SLC-40	Transiting Exoplanet Survey Satellite (TESS)	362.0	HEO	NASA (LSP)	Succ

Section 3

Launch Sites Proximities Analysis

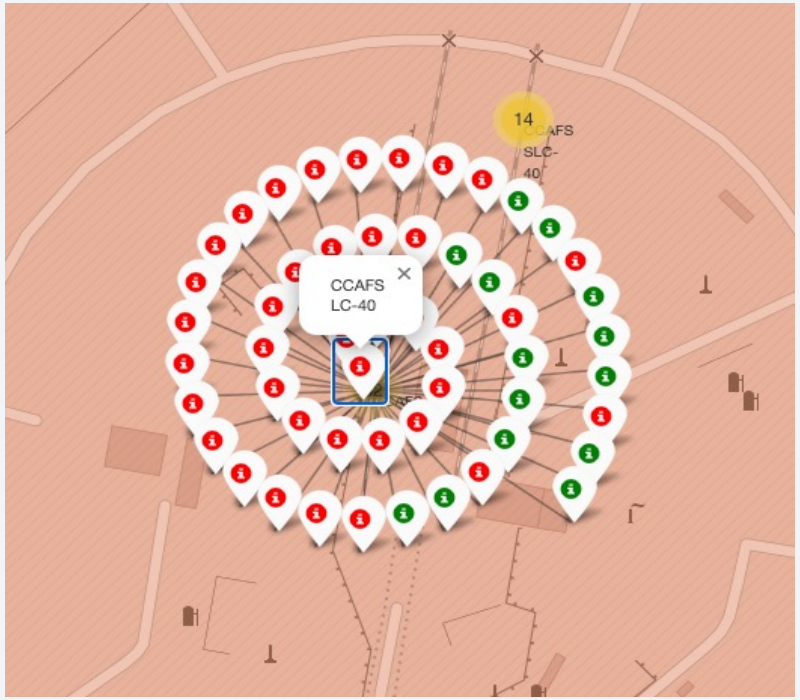
Markers of all launch sites on global map

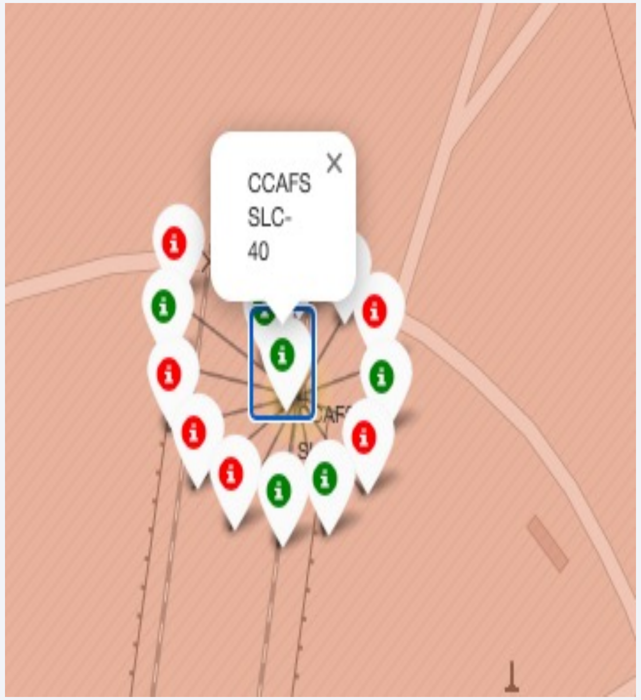


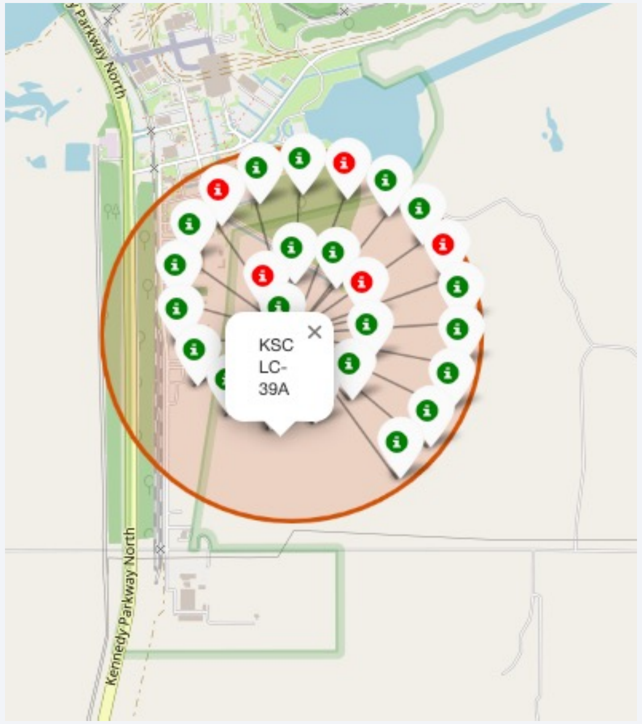
- All launch sites are in proximity to the Equator, (located southwards of the US map). Also all the laumch sites are in very close proximity to the coast.

Launch outcomes for each site on the map With Color Markers

Florida Sites







In the Eastern coast (Florida) Launch site KSC LC-39A has relatively high success rates compared to CCAFS SLC-40 & CCAFS LC-40.

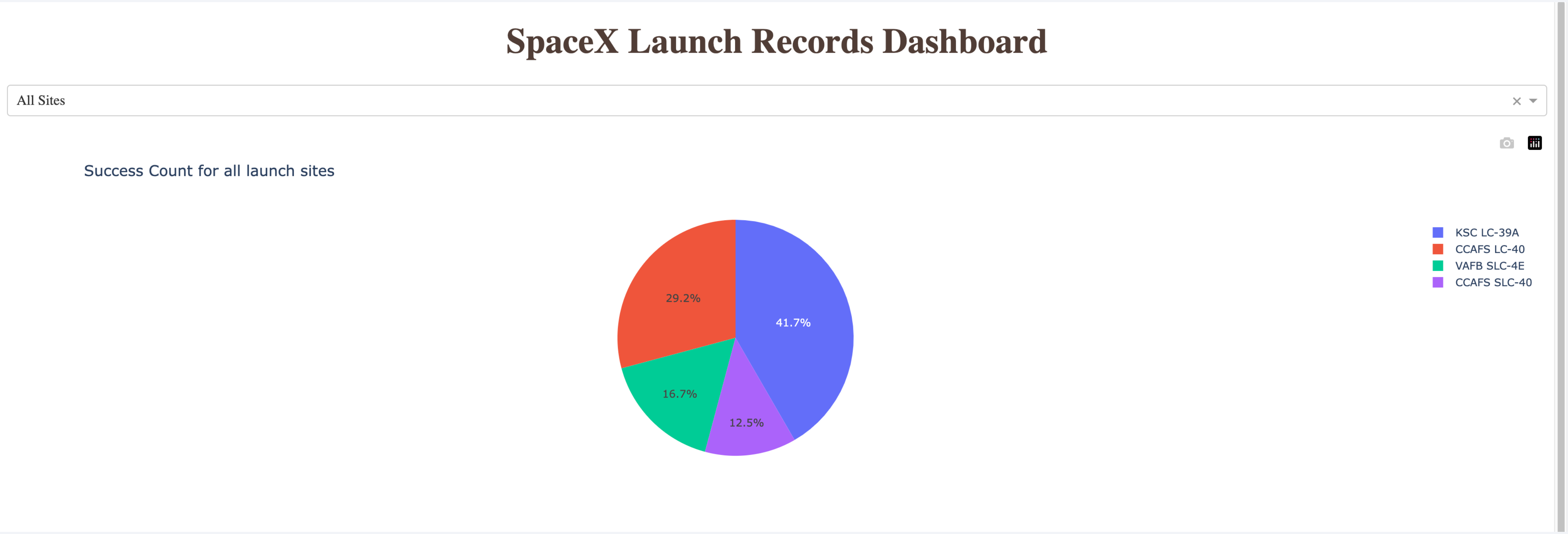
Distances between a launch site to its proximities

Launch site CCAFS SLC-40 proximity to coastline is 0.86km. closest to highway (Centaur Road)

Section 4

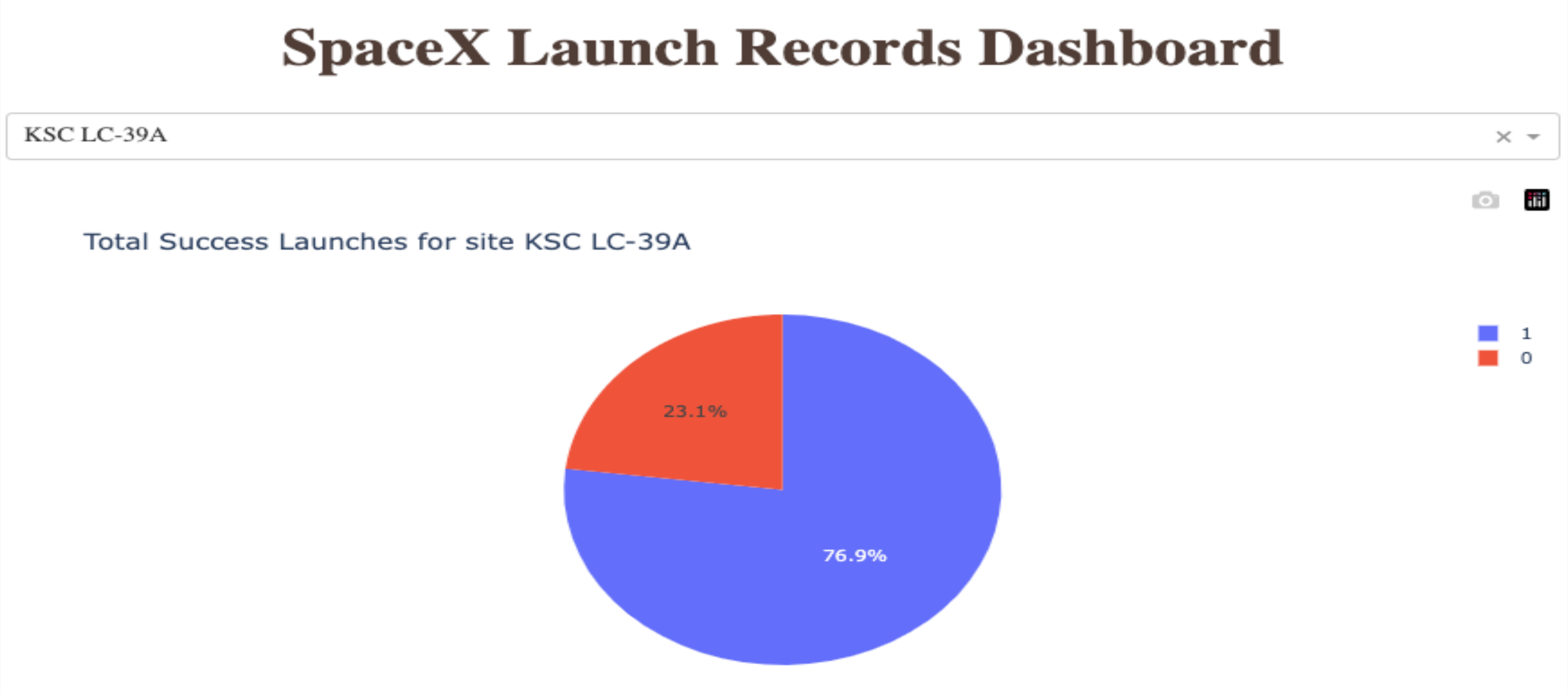
Build a Dashboard with Plotly Dash

Pie-Chart for launch success count for all sites



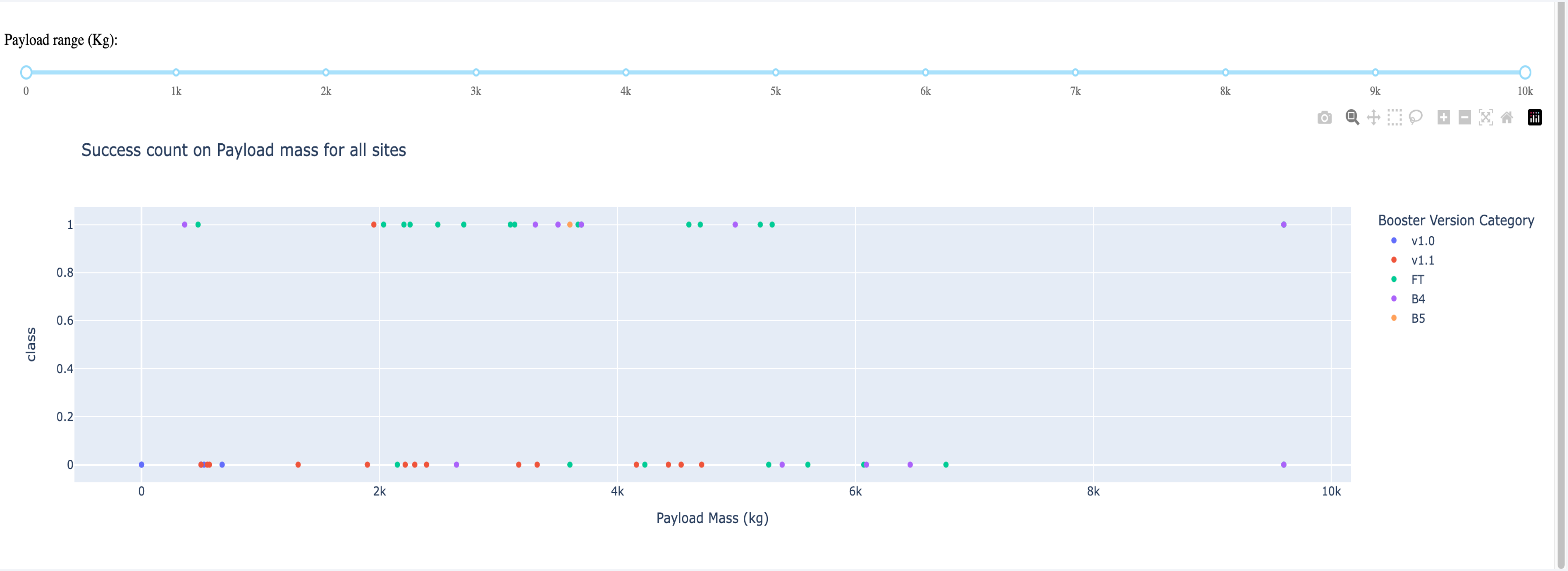
Launch site KSC LC-39A has the highest launch success rate at 42% followed by CCAFS LC-40 at 29%, VAFB SLC-4E at 17% and lastly launch site CCAFS SLC-40 with a success rate of 13%

Pie chart for the launch site with highest launch success ratio



Launch site KSC LC-39A had the highest success ratio of 76.9% success against 23.1% failed launches

Payload vs. Launch Outcome scatter plot for all sites



Section 5

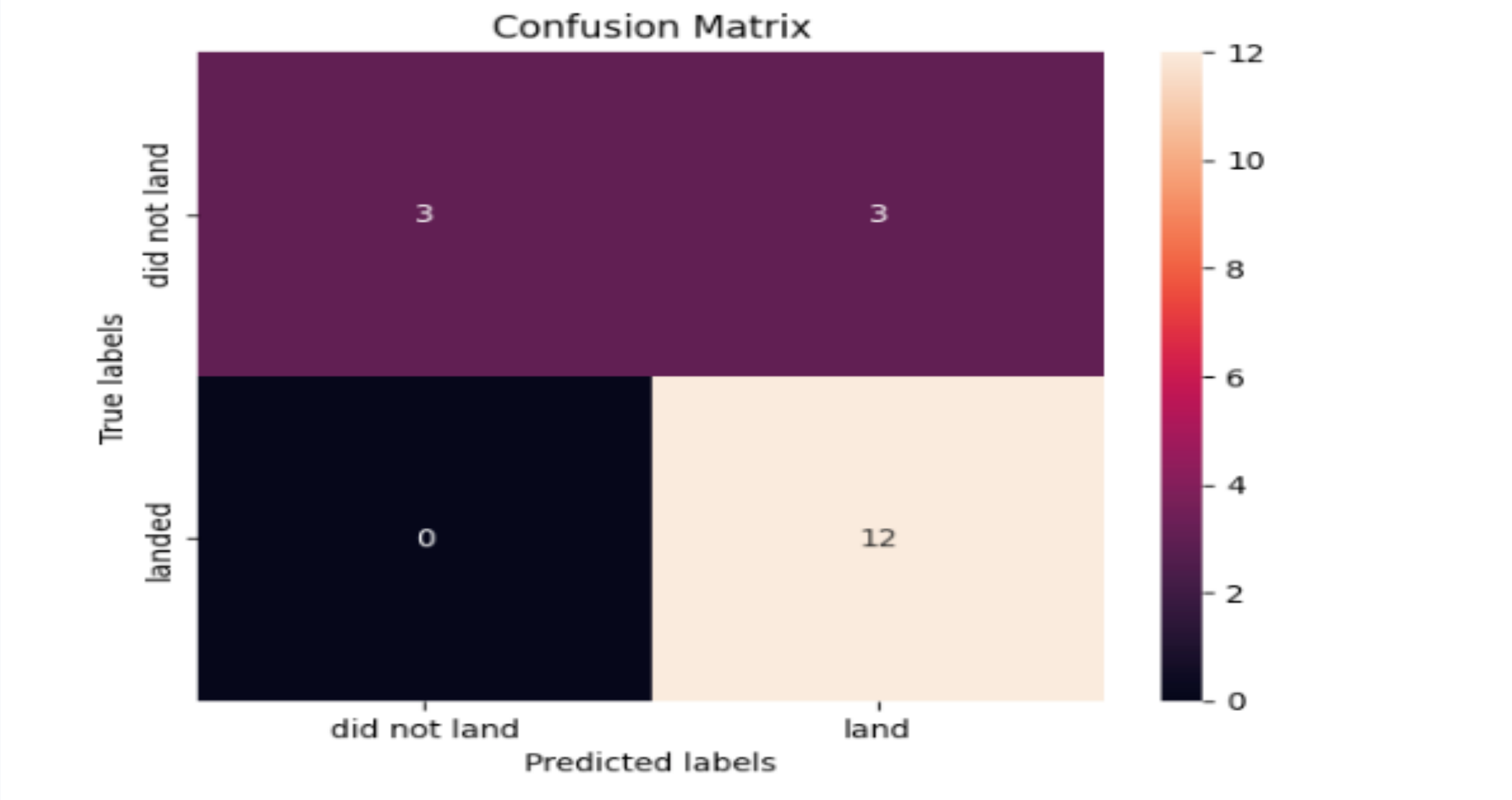
Predictive Analysis (Classification)

Classification Accuracy

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

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

- Success rates vary between launch locations. The success percentage for CCAFS LC-40 is 60%, compared to 77% for KSC LC-39A and VAFB SLC 4E.
- We can infer that the success rate grows when more flights are launched from each of the three launch sites. For instance, the VAFB SLC 4E launch site had a 100% success rate following Flight 50. After the 80th flight, both KSC LC 39A and CCAFS SLC 40 achieved 100% success rates.
- There are no rockets launched for big payload mass (more than 10,000) from the VAFB-SLC launch site, according to the Payload Vs. Launch Site scatter point chart.
- The most successful orbits are ES-L1, GEO, HEO, and SSO, with SO orbit having the least successful orbits at about 50%. Success rates for Orbit SO are zero.
- In GTO orbit, however, there does not appear to be a correlation between the number of flights and success as there is in LEO orbit.

Conclusions Contd.

- For polar, LEO, and ISS missions with substantial payloads, the successful landing or positive landing rate is higher. However, because both positive landing rate and negative landing (missions that fail) are present in GTO, it is difficult for us to differentiate between the two.
- And ultimately, from 2013 until 2020, the success rate grew.

Appendix

Code for creating the app layout

```python
# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                dcc.Dropdown(id='site-dropdown',
                                 options=[
                                        {'label': 'All Sites', 'value': 'ALL'},
                                        {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
                                        {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
                                        {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
                                        {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'}
                                        ],
                                value='ALL',
                                placeholder='Select a Launch Site here',
                                searchable=True
                                ),
                                html.Br(),

                                # TASK 2: Add a pie chart to show the total successful launches count for all sites
                                # If a specific launch site was selected, show the Success vs. Failed counts for the site
                                html.Div(dcc.Graph(id='success-pie-chart')),
                                html.Br(),

                                html.P("Payload range (Kg):"),
                                # TASK 3: Add a slider to select payload range
                                #dcc.RangeSlider(id='payload-slider',...)
                                dcc.RangeSlider(id='payload-slider',
                                                min=0,
                                                max=10000,
                                                step=1000,
                                                value=[min_payload, max_payload]
                                                ),

                                # TASK 4: Add a scatter chart to show the correlation between payload and launch success
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])
```

Appendix Contd.

- **Splitting into training and test data**

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
```

- **Snippet of table containing actual launch records**

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

<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UT
C</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage booster
s">Version,<br/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</
a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#cite_note-Dragon-12">[c]</a></sup
>
```

Thank you!