IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ipine Mercy Abbey
15th June, 2023

Outline

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

Executive Summary

- Summary of methodologies

Data was collected from “https://api.spacexdata.com/v4/launches/past” and analysis was done in the following order:

  - Data collection using SpaceX API
  - Data collection using web scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) using SQL
  - EDA DataViz
  - Launch site analysis with Folium interactive Visual Analyticts and plotly dash
  - Data analysis with Machine Learning

- Summary of all results
  - EDA results
  - Interactive visual analytics and Dashboard

Introduction

- Project background and context

  SpaceX is one of the companies that makes space travel affordable for everyone. It does this by advertising Falcon 9 rocket launches on its website with a cost of $62Million while other providers cost $165Million upwards. One of the savings is that SpaceX can reuse it's first stage, therefore, if we can determine if the first stage will land, then we can determine the cost of a launch.

- Problems you want to find answers

  The problem to be solved is to predict if the first stage of the rocket launch will land successfully.

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

- The data was collected using SpaceX API (REST API) by first importing necessary libraries, that will be useful in extracting the data from the website and into our python file, then using the REQUEST and GET function to call in the data set.
- The file was gotten in a JSON format which is not easily readable. So the data was converted into a Pandas DataFrame and normalization was done on the data set for consistency.

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

[7]: response = requests.get(spacex_url)
```

Check the content of the response

Data Collection – SpaceX API

- Data was collected from the SpaceX website using SpaceX API (REST API). The request function was used and then parsed the data using the GET function. The data was decoded as a Jason file which was then converted into a Pandas DataFrame for easy readability.

- Below is a link to the github file.

[Click Here]

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
```

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

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

```
[10]: 200
```

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

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

Data Collection - Scraping

- Web scraping was done to collect data from a list of Falcon 9 and Falcon heavy launches wikipage using BeautifulSoup. The data was extracted from a HTML table and a pandas dataframe was created by parsing the launch HTML tables.

- Here's a link to the github file

[Click Here]

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.

```
[6]: # 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`

```
[7]: # 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

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

```
[8]: <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

Data Wrangling

- After importing the necessary libraries and defining auxillary functions, analysis was then made on the data such as:
  - Calculating the number of launches on each site.
  - Calculating the number of occurrence of each orbit.
  - Calculating the number of missing outcome per orbit type
  - Creating a landing outcome label from outcome column.

- Below is the link to the github file.

[Click Here]()

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

In [8]:
```
# Apply value_counts on Orbit column
occurence = df['Orbit'].value_counts()
occurence
```

Out[8]:
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

TASK 3: Calculate the number and occurence of mission outcome per orbit type

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes`.Then assign it to a variable landing_outcomes.

EDA with Data Visualization

- A variety of visualizations were used such as:
  - A catplot was used to visualize the relationship between flight number, launch site and payload.
  - A bar graph was used to visualize the relationship between success rate of each orbit type to be able to tell the difference in height,
  - A line chart was used to visualize the yearly trend of launch success

- Below is the link to the github file.

Click Here

EDA with SQL

- The following SQL queries were performed for EDA after connecting to the database.
  - Display the names of the unique launch sites in the space mission

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

  - Display 5 record 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 launch by NASA (CRS)

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

- Below is the link to the github file

[Click Here]

Build an Interactive Map with Folium

- Folium map was created to mark all the launch sites, and map objects such as markers, circles, lines etc. to pin-point exact places on the maps.

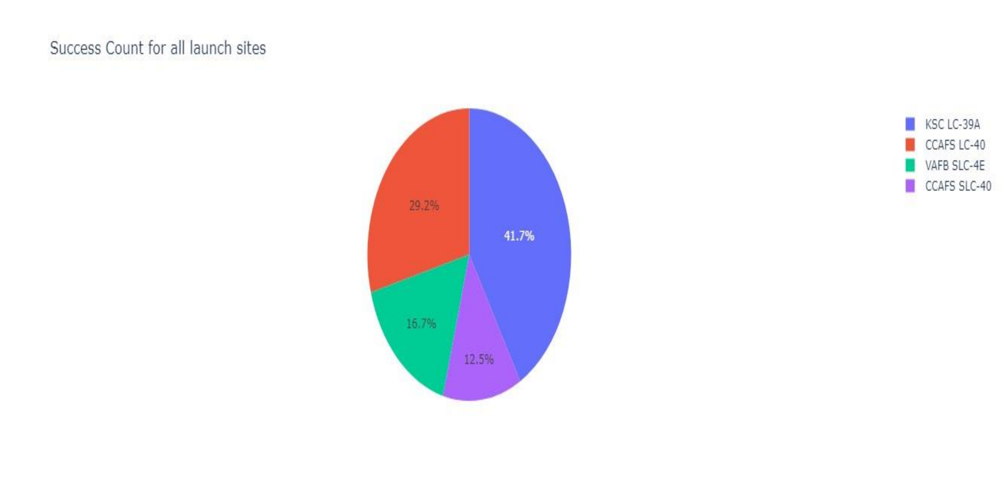
- The objects were created to mark all the launch sites on then map, Mark the success/failed launches for each site on the map and Calculate the distances between a launch site to its proximities.

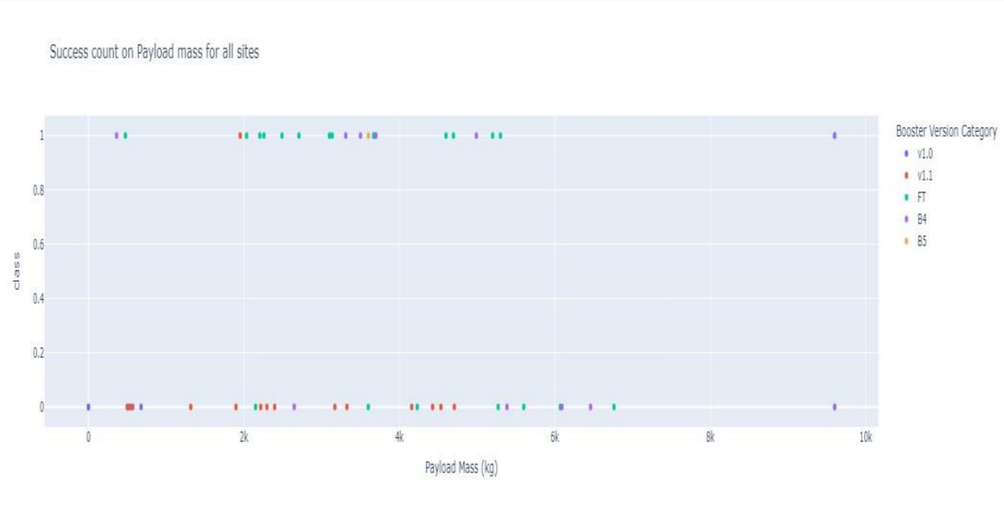
- Below is the link to the github file.
- [Click Here]

Build a Dashboard with Plotly Dash

- An interactive dashboard was built using ploty-dash by:
  - Adding a launch site drop-down input field
  - Adding a callback function to render success-pie-chart based on selected site dropdown
  - Adding a slider bar to select payload
  - Adding a callback function to render the success-payload-scatter-chart scatter plot.

- The plots were added to give different forms of interactions.
- Below is the github link to the file
- [Click Here]





Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- After converting the data to a pandas DataFrame, I performed exploratory data analysis and determined training labels:
  - Creating a numpy array from the column class data by applying the method to numpy() then assigned it to the variable Y as the outcome variable.
  - Then standardized the feature dataset X by transforming it using preprocessing.StandardScalar() function from sklearn
  - After which the data was split into training and testing set using the function train_test_split from sklearn.model_selection with the test_size parameter set to 0 and random_state to 2.
- Below is the link to the github file
- Click Here

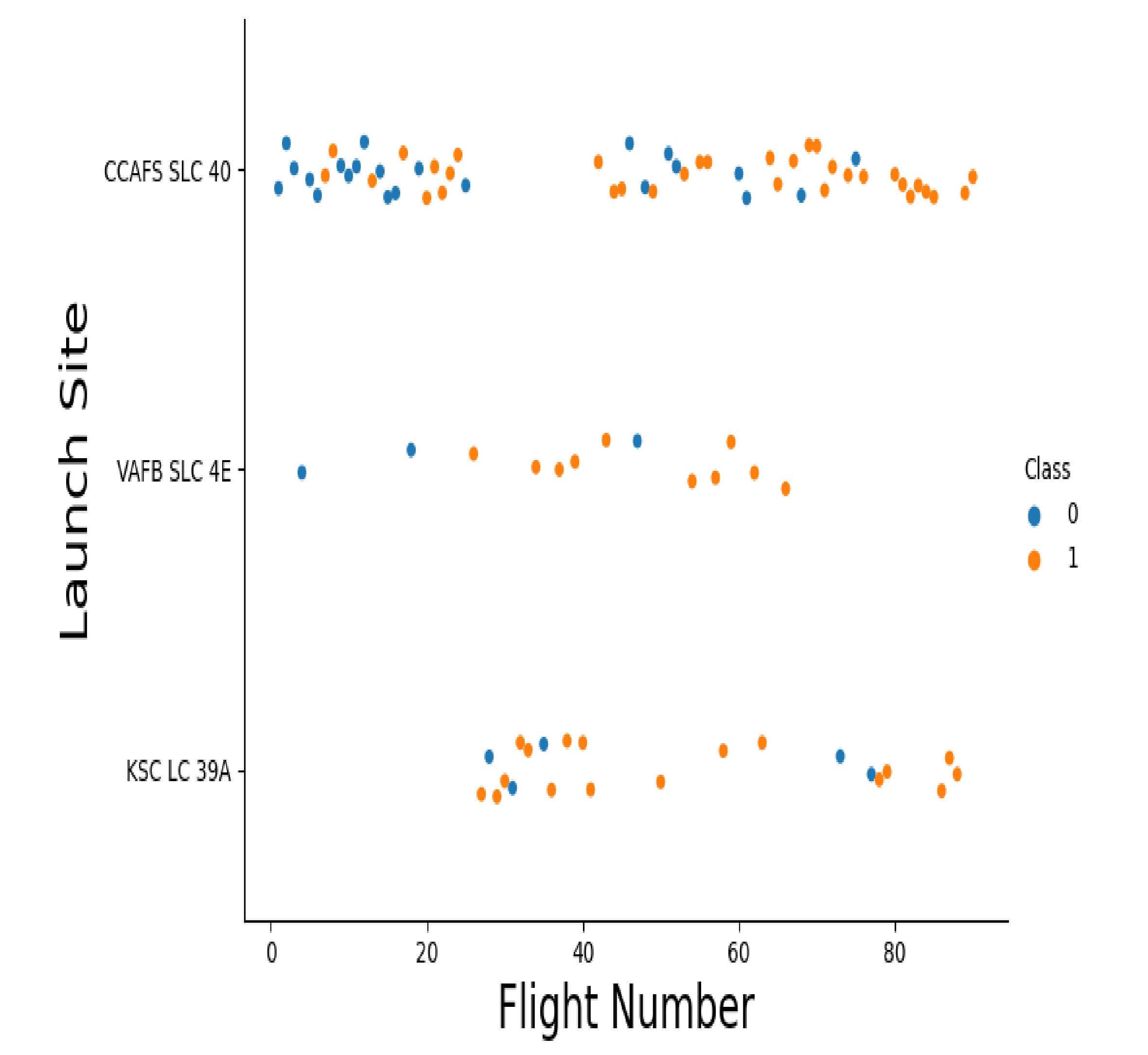
Results

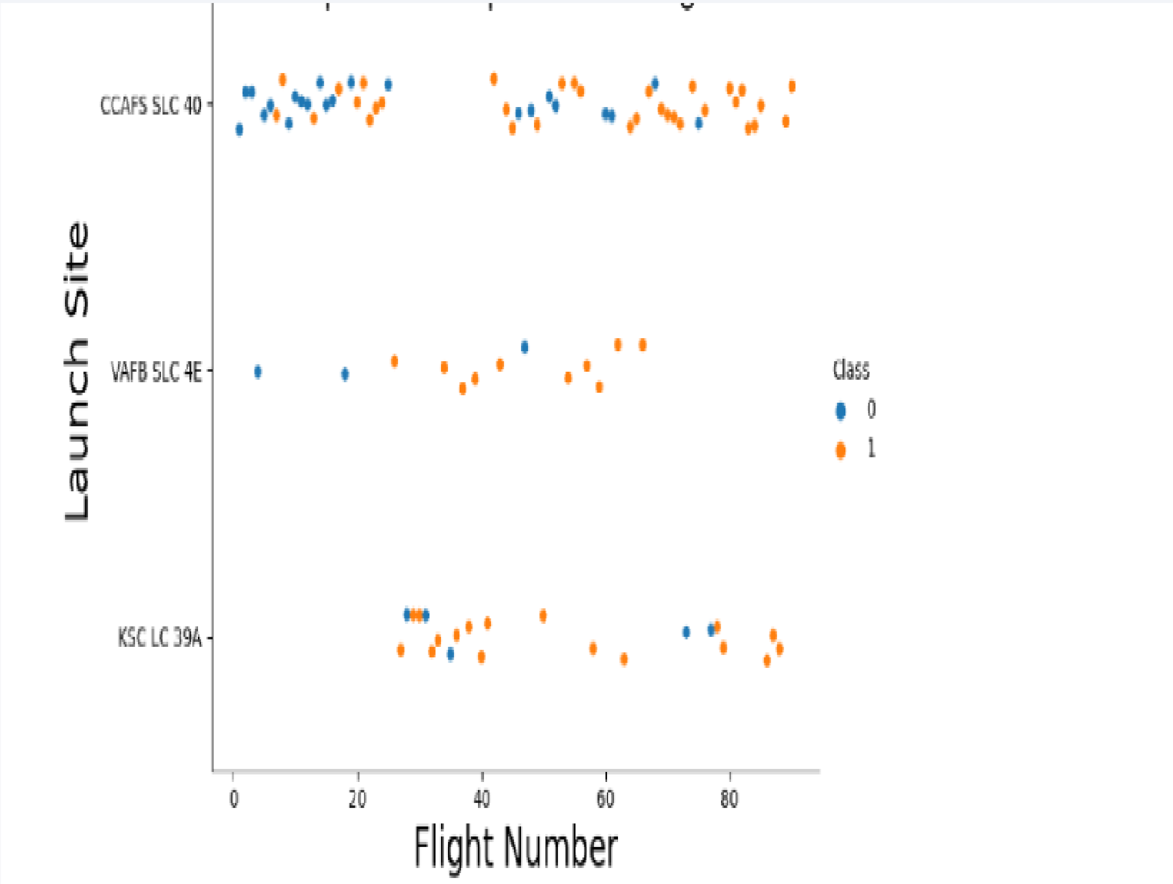
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

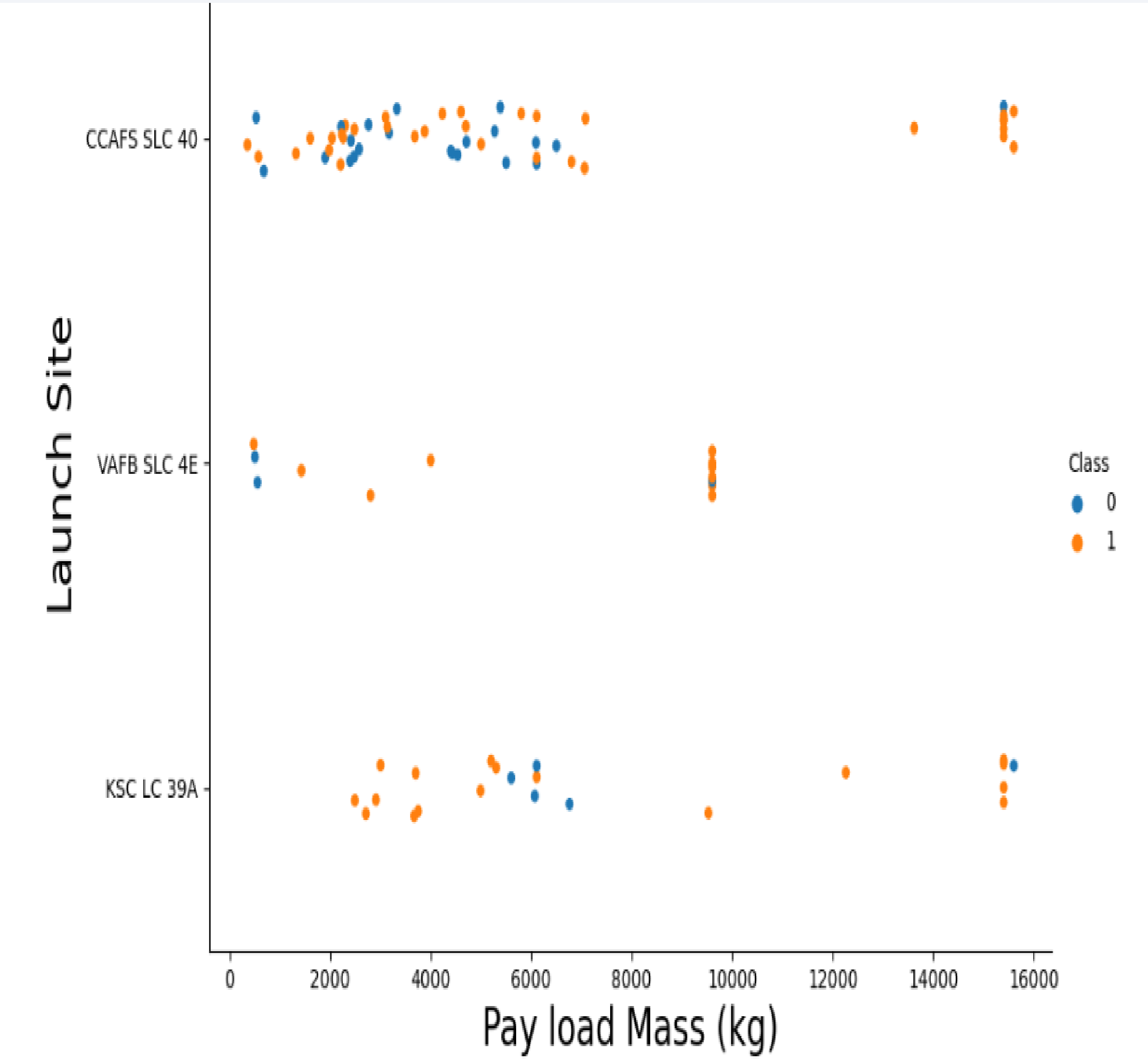


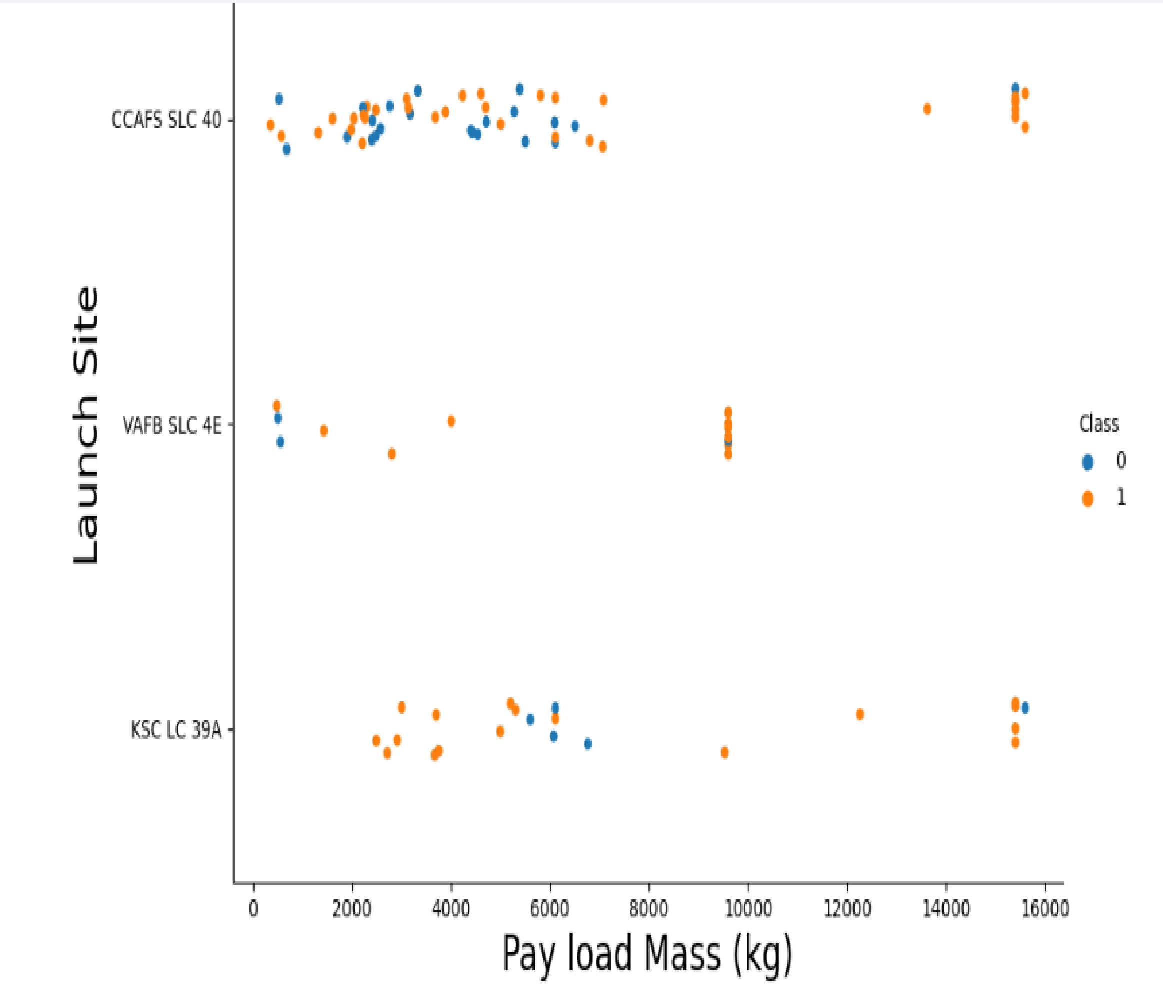


Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

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.

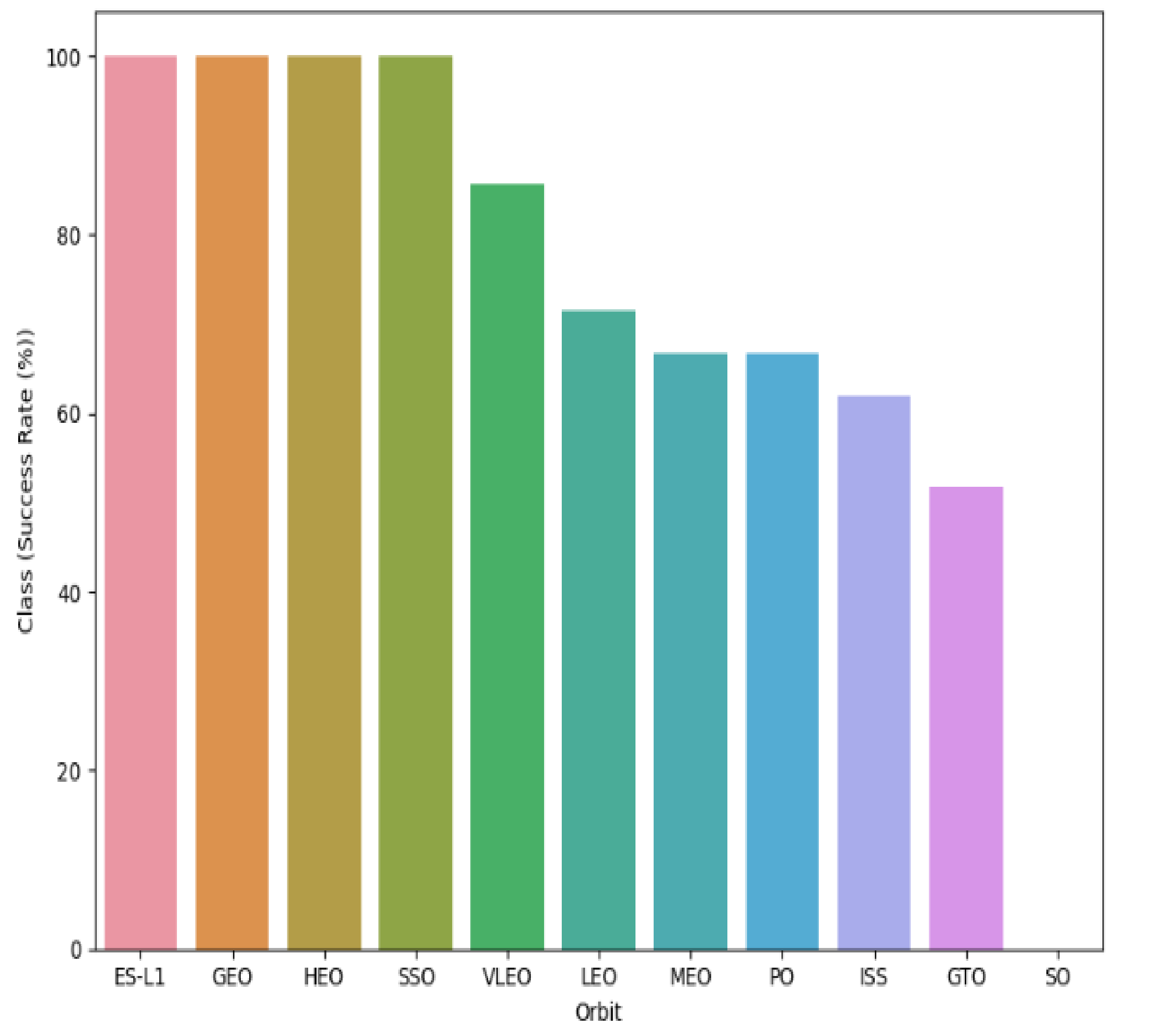
Payload vs. Launch Site

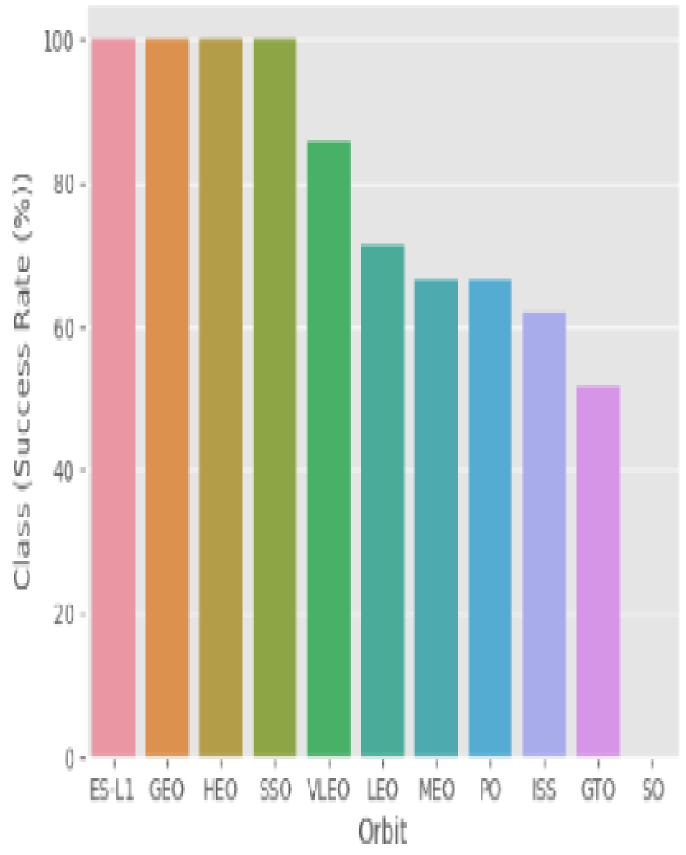




Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

Success Rate vs. Orbit Type

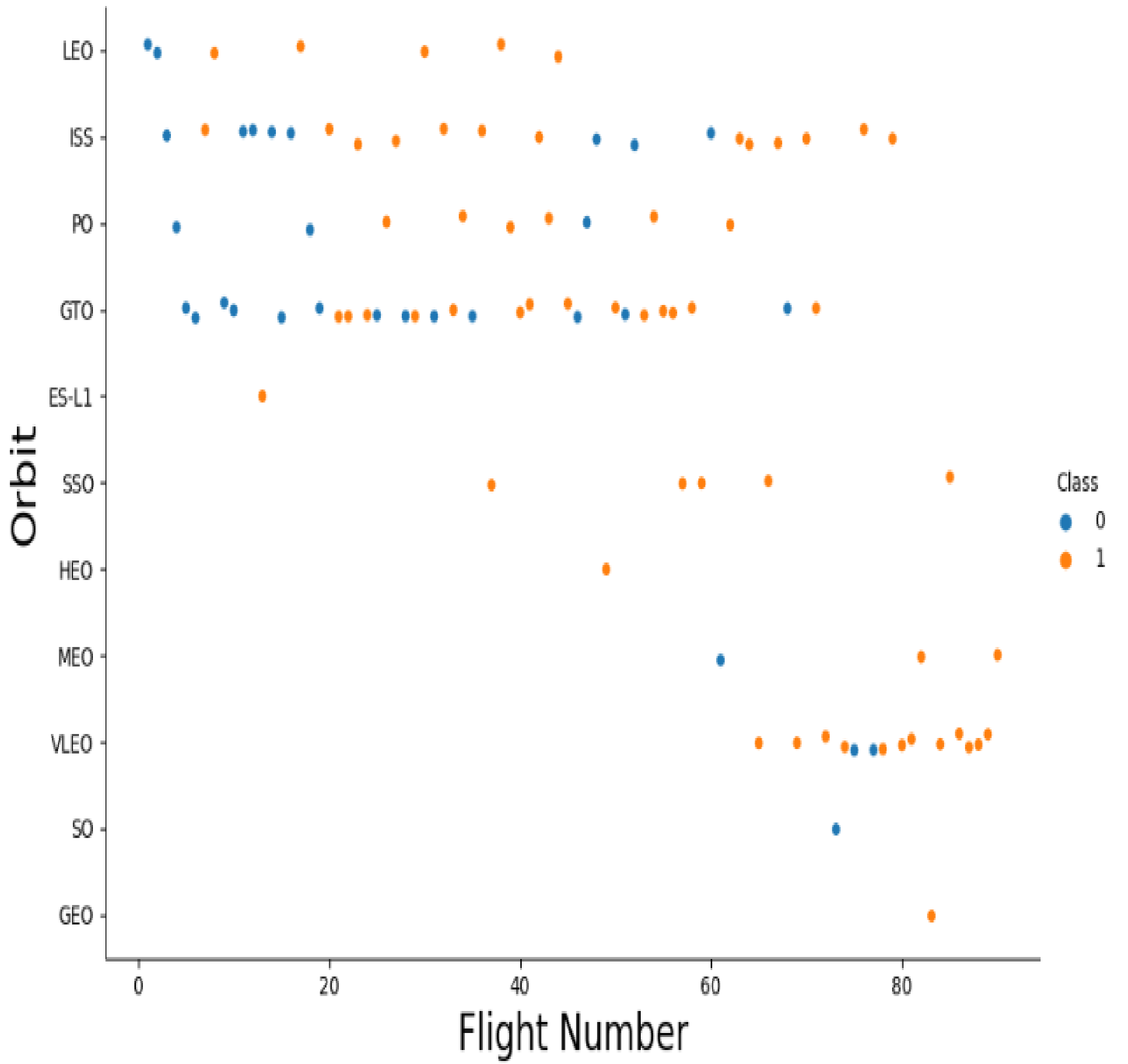


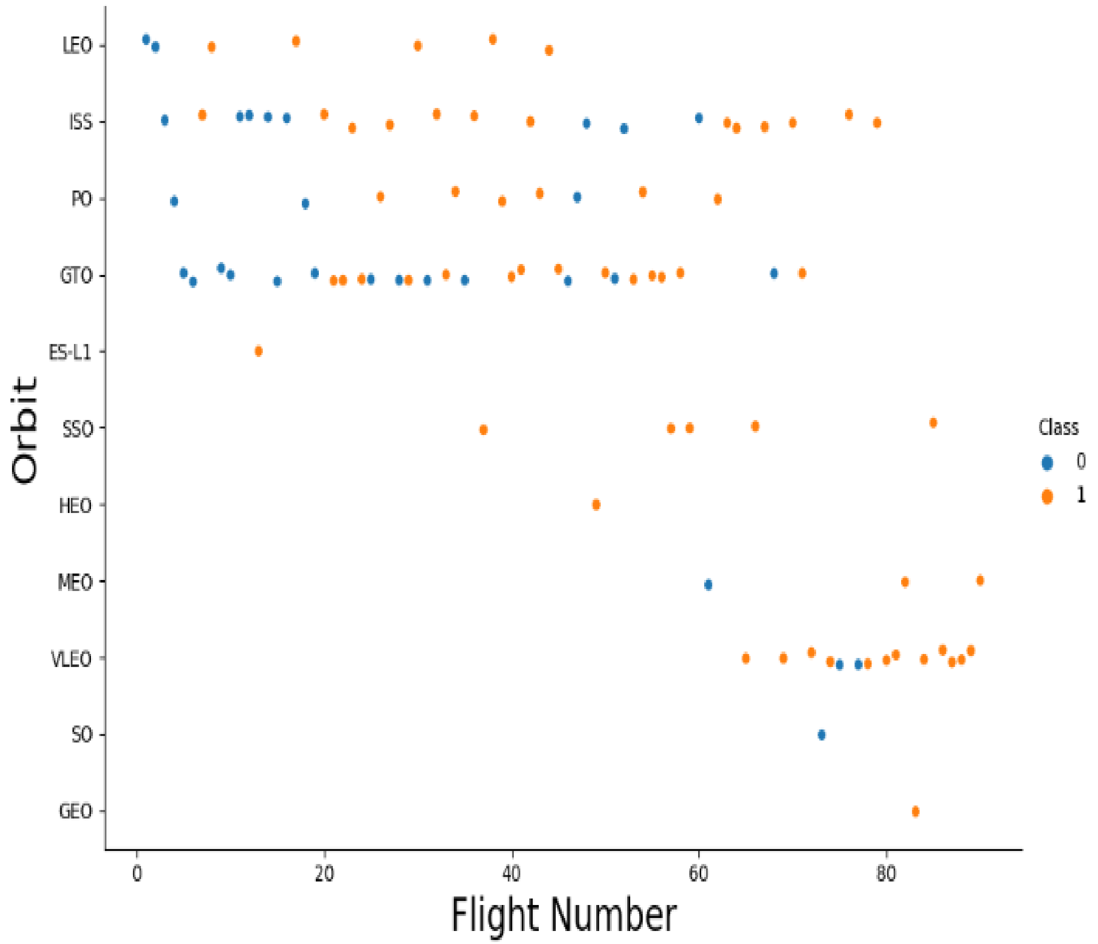


Analyze the ploted bar chart try to find which orbits have high sucess rate.

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.

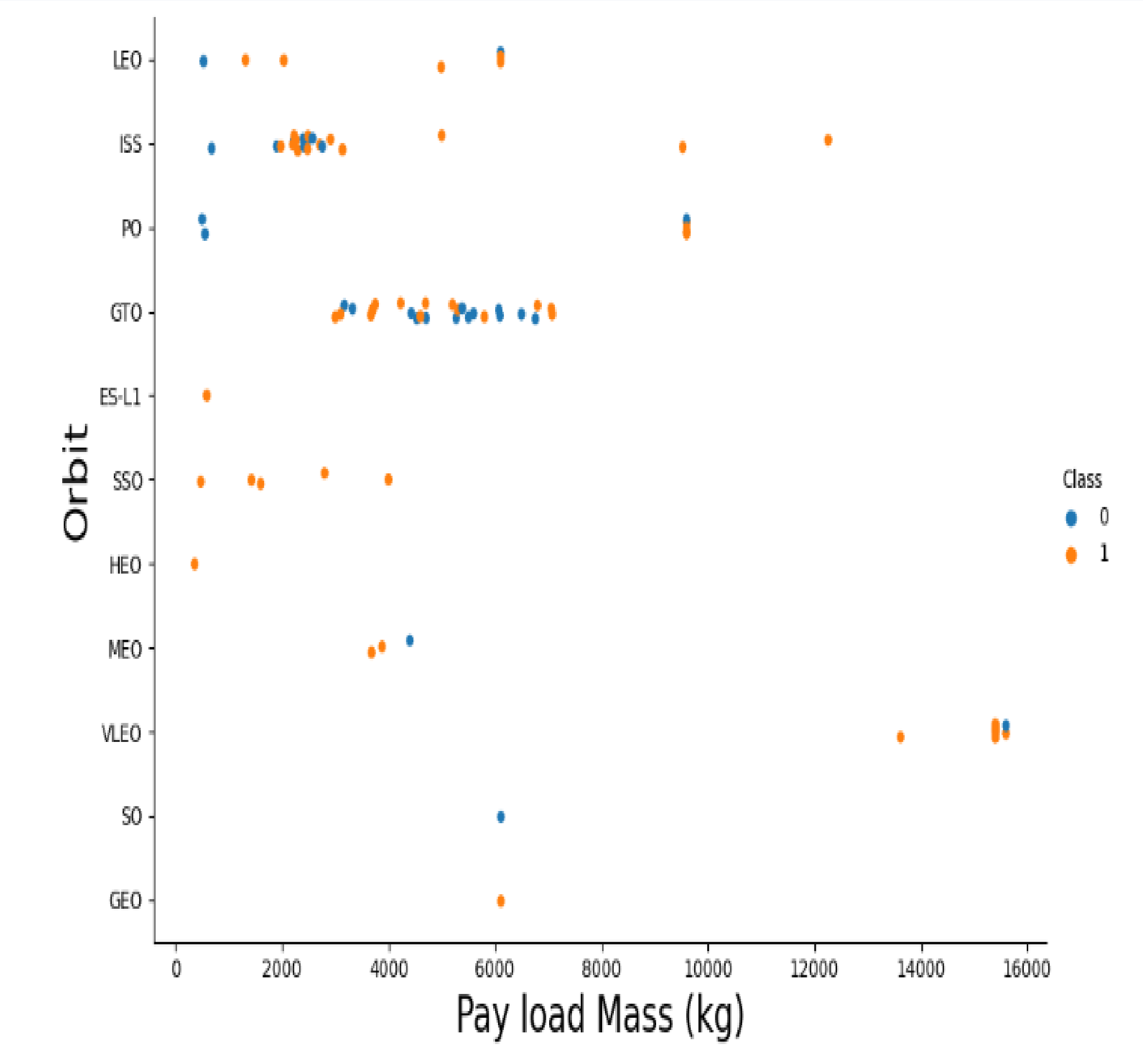
Flight Number vs. Orbit Type

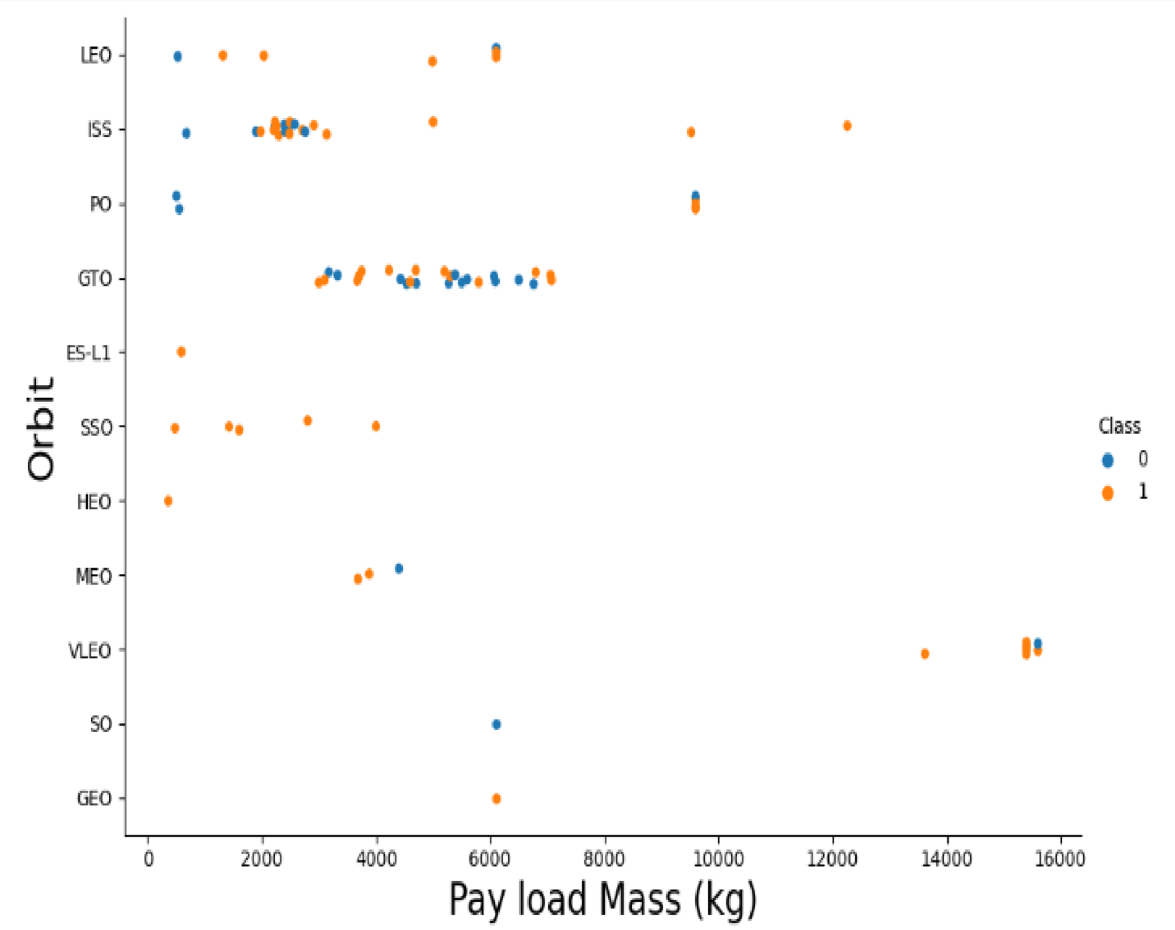




You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type

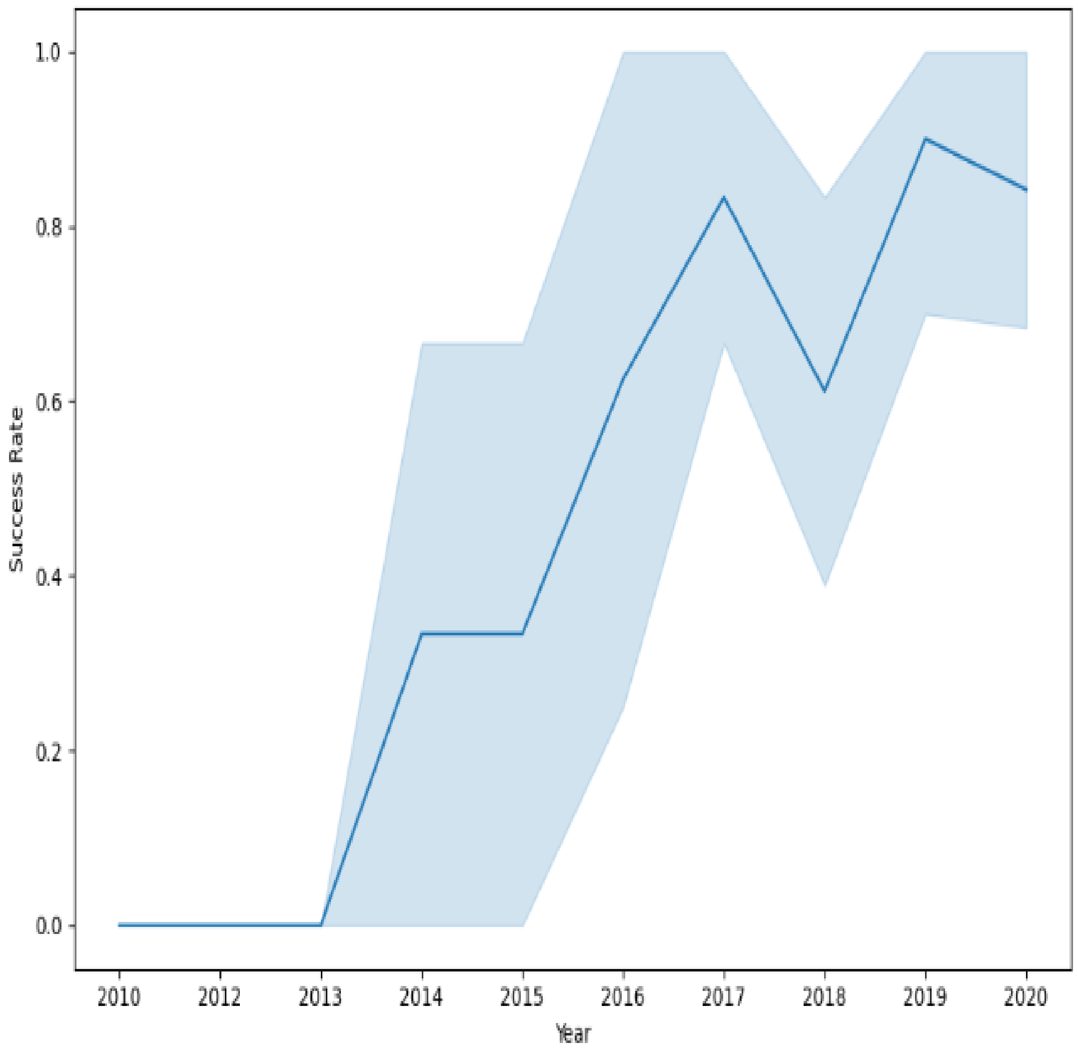


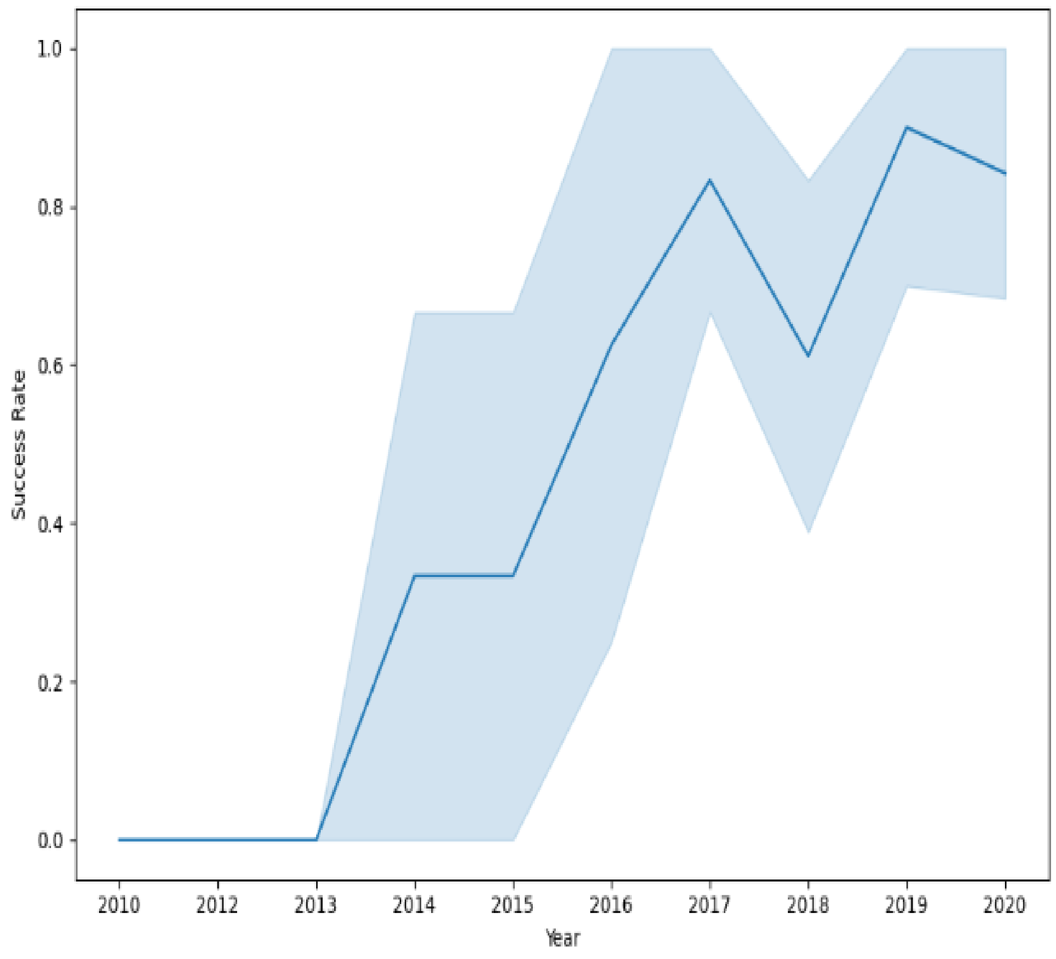


With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

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 [7]:
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

Out[7]:

Launch_Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

- Using the SELCT DISTINCT function in SQL to return the unique names of the columns from the SpaceX table.

Launch Site Names Begin with 'CCA'

Task 2

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

In [8]:
```
%sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[8]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (paracl
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	Failure (paracl
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No att
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No att
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No att

- Using the SELCT * FROM, the command selects all the column from the spaceX table where the launch site name begin with ‘CCA’ and the % reperesents a wildcard.

Total Payload Mass

Task 3

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

In [9]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

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

Out[9]:

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

- The query calculates the total payload mass carried by boosters launched by NASA (CRS)

Average Payload Mass by F9 v1.1

Task 4

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

In [10]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version 
```

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

Out[10]:

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

- The query calculates the payload mass carried out 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

- The above query finds the smallest/earliest date that the first successful launch took place.

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

- The query lists the names of the booster which have success in drone ship and have payload mass greater than 4000 but less than 6000

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
None	0
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- The query lists the total number of successful and failed outcomes from the data in the SpaceX table by using the count and group by function.

Boosters Carried Maximum Payload

Task 8

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

In [17]:
```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_I
```

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

Out[17]:

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

- The query lists the names of the booster version which have carried the maximum payload mass using a subquery, SELECT and MAX function

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)

- The query lists the records which will display the month names, failure landing, outcomes in drone ship booster versions, launch site for the months in year 2015

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)

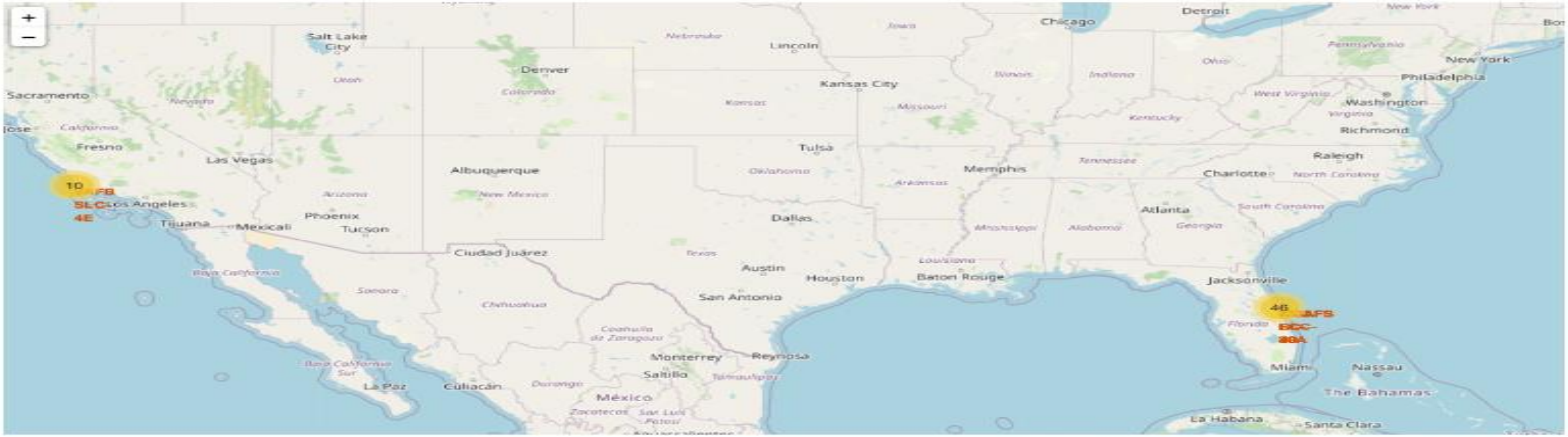
- The query ranks the count of successful landing outcomes between the date 04-06-2010 and 20-03-2-17 in descending order.

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

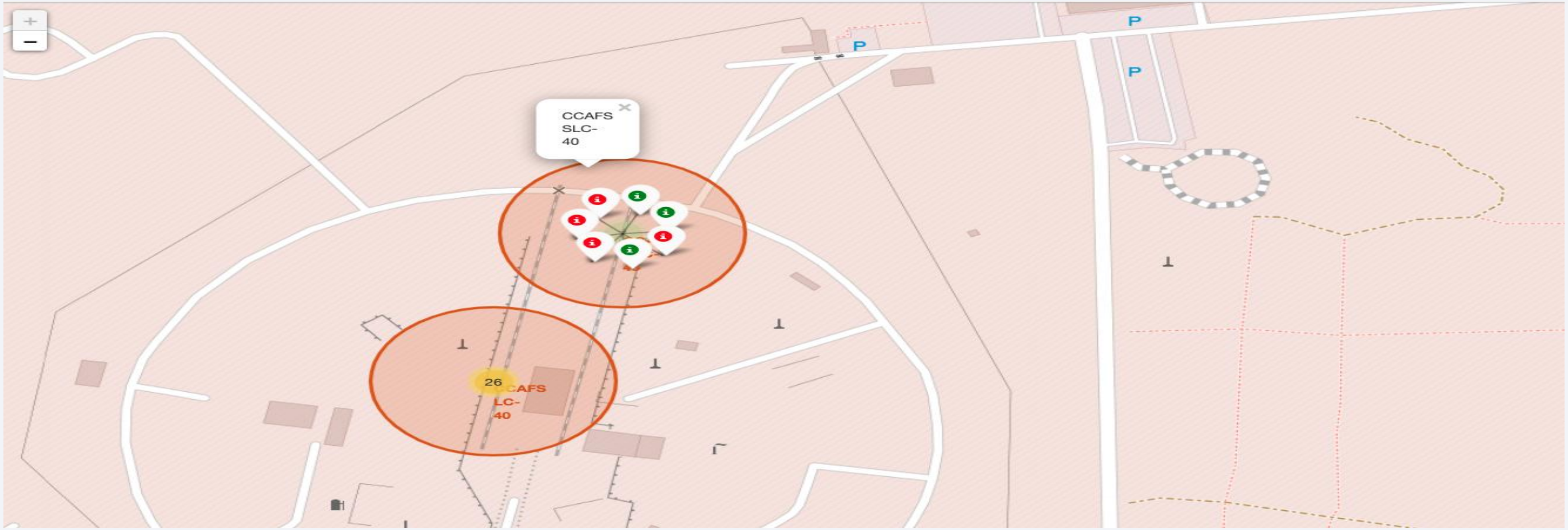
LAUNCH SITE WITH MARKERS



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

<Folium Map Screenshot 2>

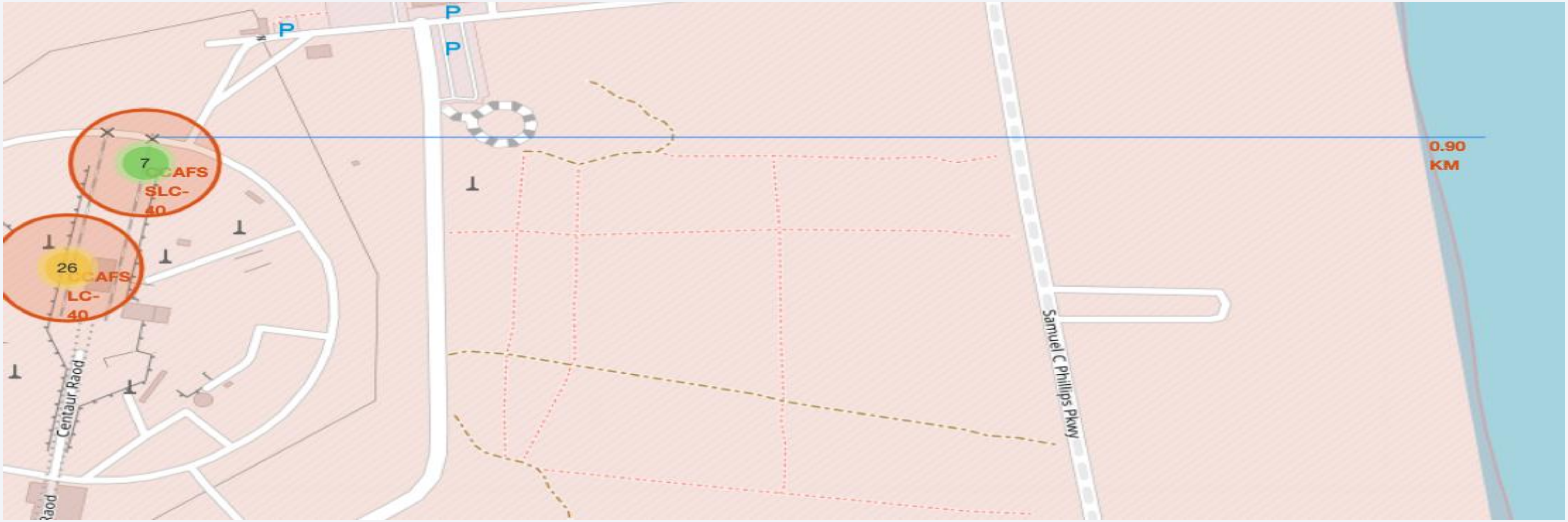
COLOR CODED LAUNCH SITE SUCCESS RATE



From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

<Folium Map Screenshot 3>

MAP WITH DISTANCE MARKER



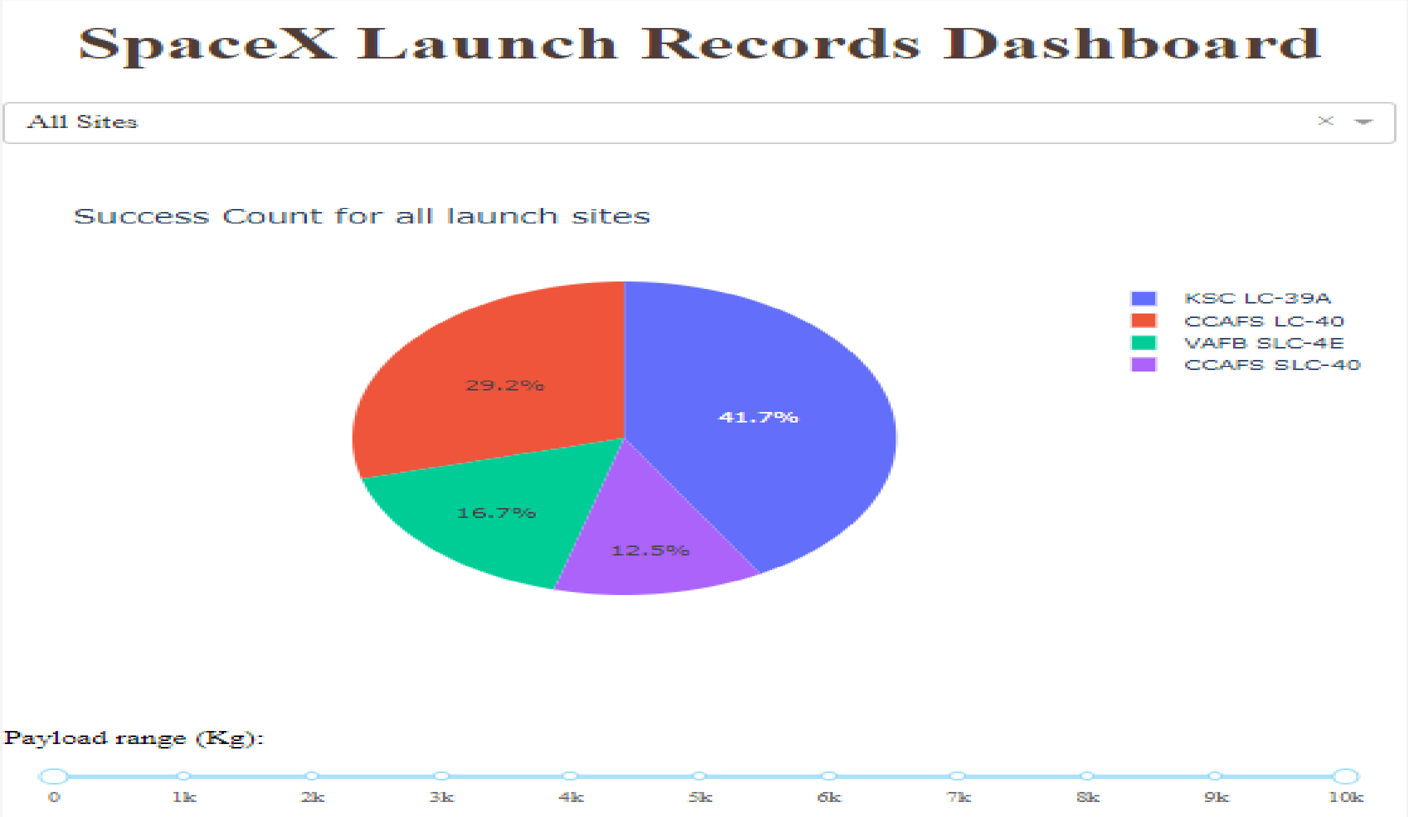
- Distance from a launch site to the closest city, railway, highway etc,

Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

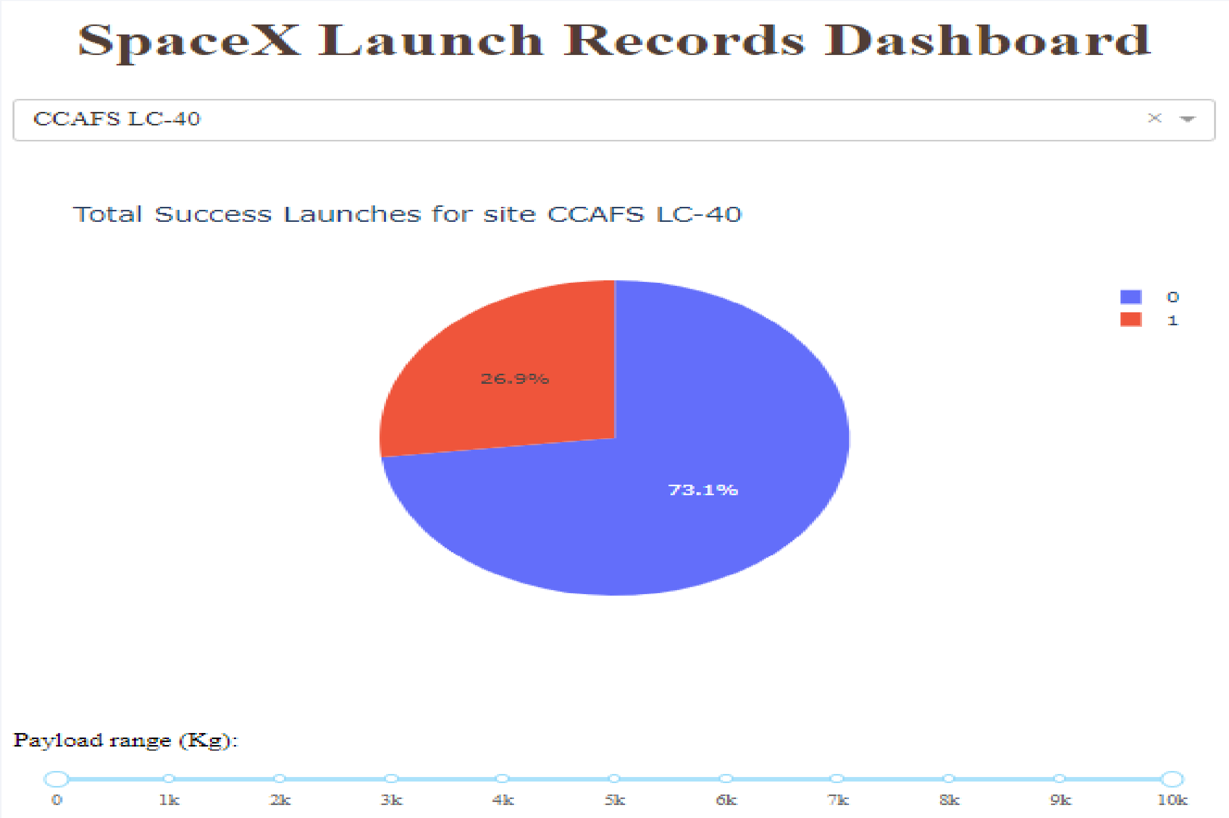
LAUNCH SITE LOCATIONS



- The success rate of each launch site is shown on the pie chart above with the KSC LC-39A launch site having the highest success rate of 41.7%

<Dashboard Screenshot 2>

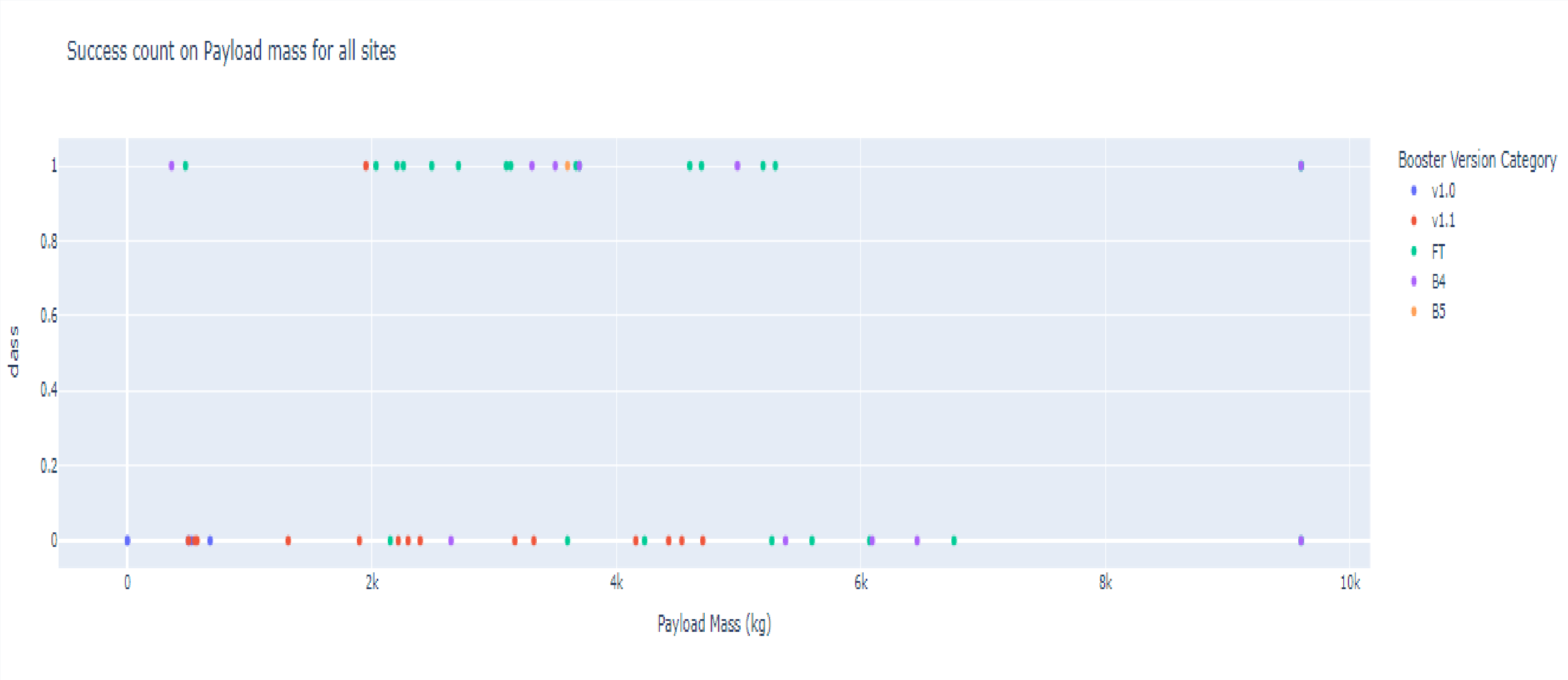
TOP 2 LAUNCH SITE LOCATIONS



- Launch site CCAFS LC-40 has the second highest success rate of 26.9%. Which gives a ratio of approximately 3:8

<Dashboard Screenshot 3>

SUCCESS COUNT ON PAYLOAD MASS FOR ALL SITES



- For launch site CCAFS LC-40, the booster version FT has the largest success rate from a payload mass of >200kg

Section 5

Predictive Analysis (Classification)

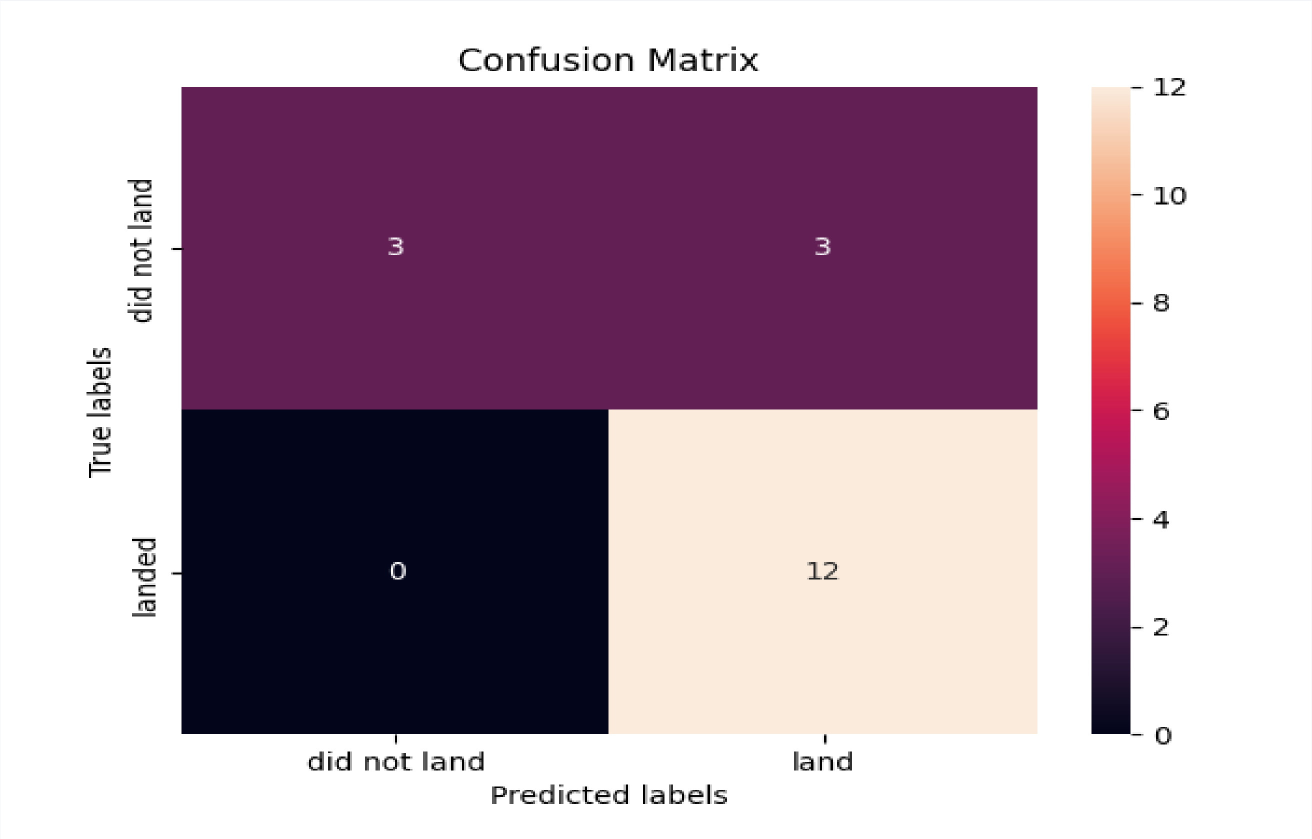
Classification Accuracy

Out[33]:

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

- All the models have the same level of accuracy which is 0.8333 approximately 83.3%

Confusion Matrix



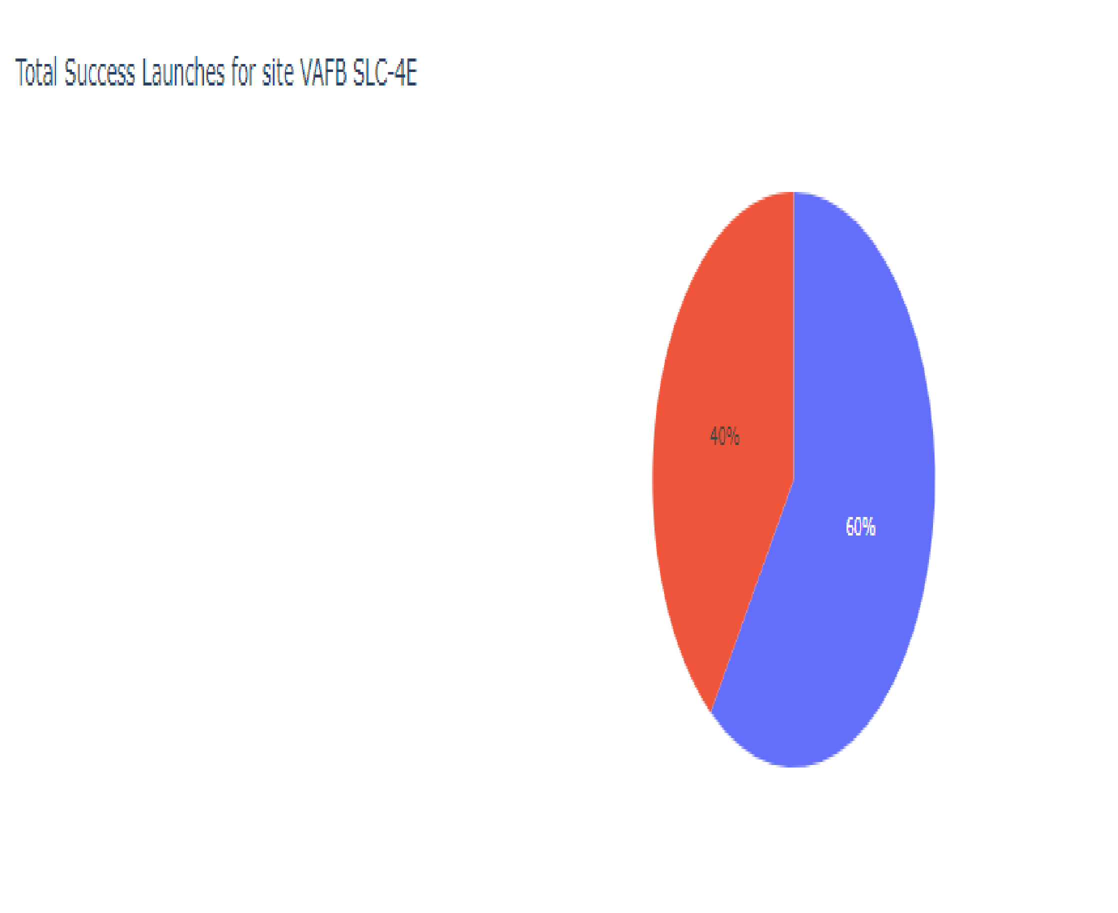
- This is a confusion matrix showing all the models from the data

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 launch 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-LI, 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

- TASK 2:
- # Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
- @app.callback(Output(component_id='success-pie-chart', component_property='figure'),
- Input(component_id='site-dropdown', component_property='value'))
- def get_pie_chart(entered_site):
- filtered_df = spacex_df
- if entered_site == 'ALL':
- fig = px.pie(filtered_df, values='class',
- names='Launch Site',
- title='Success Count for all launch sites')
- return fig
- else:
- # return the outcomes piechart for a selected site
- filtered_df=spacex_df[spacex_df['Launch Site']== entered_site]
- filtered_df=filtered_df.groupby(['Launch Site','class']).size().reset_index(name='class count')
- fig=px.pie(filtered_df,values='class count',names='class',title=f"Total Success Launches for site {entered_site}")
- return fig



Thank you!