IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rohini Patil
01-Feb-2025

Outline

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

General

Executive Summary

- **Summary of methodologies**
  - SpaceX Data Collection using SpaceX API
  - SpaceX Data Collection using Web Scrapping
  - SpaceX Data Wrangling
  - SpaceX Exploratory Data Analysis using SQL
  - SpaceX EDA DataViz using Python Pandas and Matplotlib
  - SpaceX Launch Sites Analysis with Folium – Visual Analytics and Ploty Dash
  - SpaceX Machine Learning Landing Prediction

- **Summary of all results**
  - EDA Results
  - Interactive Visual Analytics And Dashboards
  - Predictive Analysis(Classification)

General

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**

In this capstone, we will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertise on its websites.

General

Section 1

Methodology

General

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

General

Data Collection

- Description of how SpaceX Falcon9 data was collected.

  - **SpaceX API:** Data was first collected using the SpaceX API (a RESTful API) by making a GET request. This involved defining helper functions to extract information using identification numbers in the launch data, then requesting rocket launch data from the SpaceX API URL.

  - **JSON Parsing:** To make the JSON results more consistent, the SpaceX launch data was requested and parsed using the GET request, and the response content was decoded as a JSON result, which was then converted into a Pandas Data Frame.

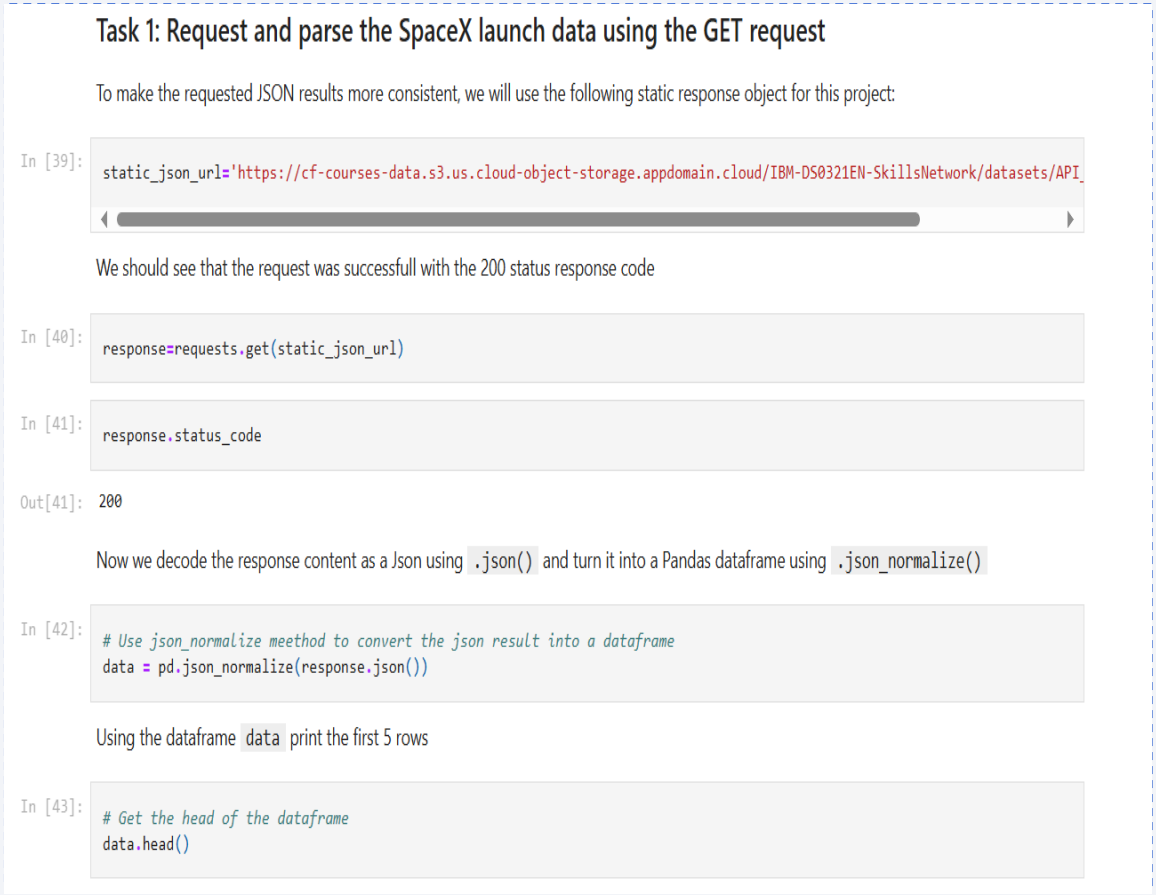
  - **Web Scraping:** Additionally, web scraping was performed to collect Falcon 9 historical launch records from a Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches." The launch records stored in HTML were extracted using Beautiful Soup and request libraries, parsed, and converted into a Pandas Data Frame.

General

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 is then converted into a Pandas Data Frame.

- Here is the GitHub URL of the completed SpaceX API calls notebook (Applied-Data-Science-Capstone/spacex-data-collection-api.ipynb at main · RohiniPatil0/Applied-Data-Science-Capstone), as an external reference and peer-review purpose



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 [39]:
```
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 [40]:
```
response=requests.get(static_json_url)
```

In [41]:
```
response.status_code
```

Out[41]: 200

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

In [42]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

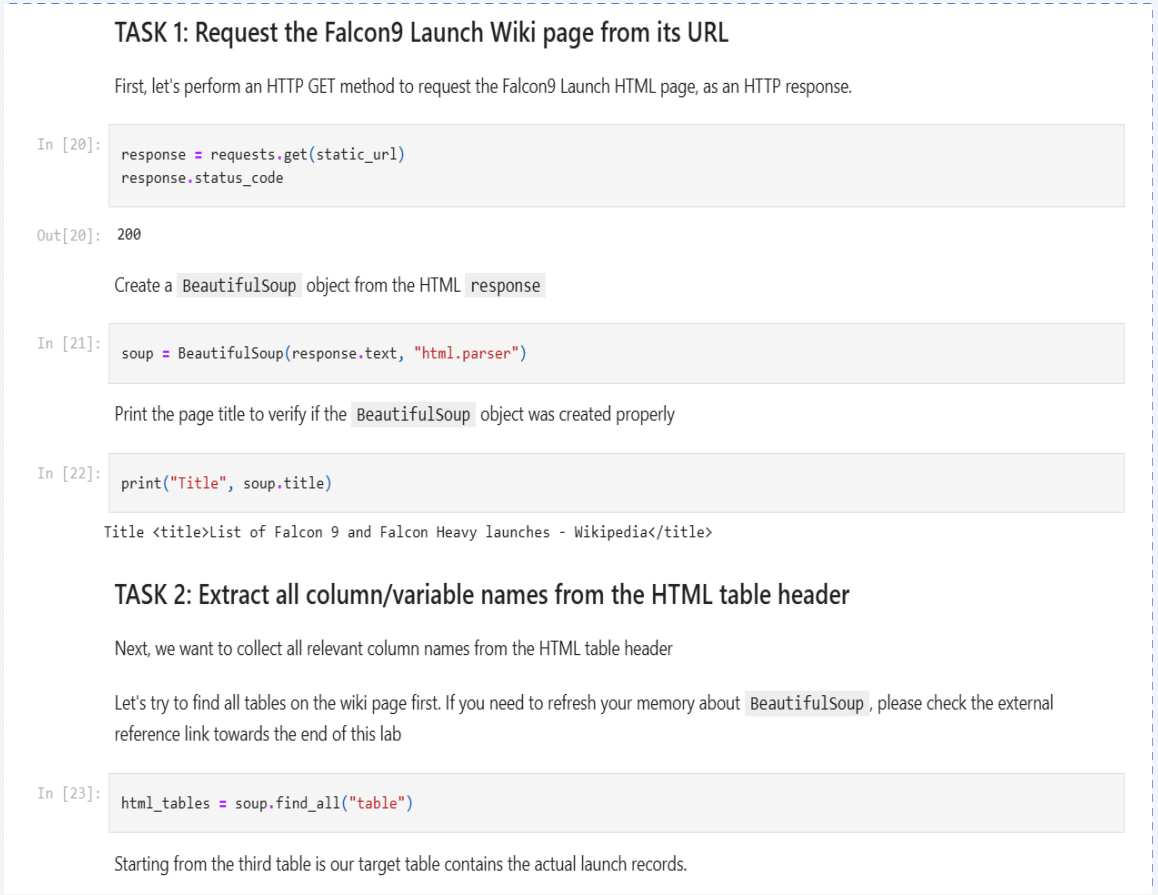
Using the dataframe `data` print the first 5 rows

In [43]:
```
# Get the head of the dataframe
data.head()
```


General

Data Collection - Scraping

- Performed Web scrapping to collect Falcon9 historical launch records from a Wikipedia using Beautiful Soup and request, to extract the Falcon 9 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, as an external reference and peer-review purpose
- [Applied-Data-Science-Capstone/webscraping.ipynb at main · RohiniPatil0/Applied-Data-Science-Capstone]



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 [20]:
```
response = requests.get(static_url)
response.status_code
```

Out[20]: 200

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

In [21]:
```
soup = BeautifulSoup(response.text, "html.parser")
```

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

In [22]:
```
print("Title", soup.title)
```

Title <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 [23]:
```
html_tables = soup.find_all("table")
```

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

General

Data Wrangling

- After obtaining and creating a Pandas DF from the collected data, data was filtered using the Booster Version 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.
- Also performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- Here is the GitHub URL of the completed data wrangling related notebooks.
- [Applied-Data-Science-Capstone/spacex-Data wrangling.ipynb at main · RohiniPatil0/Applied-Data-Science-Capstone]

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 [27]:
```
# 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[27]:
```
Class
1    60
0    30
Name: count, 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 [28]:
```
landing_class=df['Class']
df[['Class']].head(8)
```

Out[28]:

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

General

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
  - Flight Number 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
- Applied-Data-Science-Capstone/edadataviz.ipynb at main · RohiniPatil0/Applied-Data-Science-Capstone

General

EDA with SQL

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

```
In [10]: %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 TotalPayloadMass FROM SPACEXTABLE  WHERE Customer = 'NASA (CRS)'
```

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

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

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

```
%sql select min(Date) as date from SPACEXTABLE 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

```
select Booster_Version from SPACEXTABLE where Landing_Outcome  = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between  4000
```


General

EDA with SQL(cont.....)

- **List the total number of successful and failure mission outcomes**

```
%sql SELECT MISSION_OUTCOME , COUNT(*) as totalCOunt from SPACEXTABLE 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 from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

- **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 CASE WHEN substr(Date, 6, 2) = '01' THEN 'January' WHEN substr(Date, 6, 2) = '02' THEN 'February' WHEN substr(Da
```

- **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 case when landing_outcome = 'Success (ground pad)'  then 'Success (ground pad)' when landing_outcome = 'Failure
```

- Here is the GitHub URL

Applied-Data-Science-Capstone/eda-sql-coursera_sqllite.ipynb at main · RohiniPatil0/Applied-Data-Science-Capstone

Build an Interactive Map with Folium

- Created Folium Map to mark all launch sites and created map objects such as markers, circles, lines to mark the success or failure of launch for each launch site.
- Created a launch set outcomes(failure=0 or success=1)
- Here is the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose
- Applied-Data-Science-Capstone/launch_site_location.ipynb at main · RohiniPatil0/Applied-Data-Science-Capstone

General

Build a Dashboard with Plotly Dash

- **Built an interactive dashboard application with Ploty Dash By**
- 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, as an external reference and peer-review purpose**
- Applied-Data-Science-Capstone/spacex_dash_app.py at main · RohiniPatil0/Applied-Data-Science-Capstone

General

Predictive Analysis (Classification)

- Summary of how I built, evaluated, improved, and found the best performing classification model.
- After loading the data as a Pandas Dataframe, I set out to perform exploratory Data Analysis and determine Training Labels by:
- 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.
- Then standardized the feature dataset (x) by transforming it using preprocessing.StandardScaler() function from Sklearn.
- 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 2.

General

Predictive Analysis (Classification)

- In order to find the best ML model/method that would perform best using the test data between SVM, Classification Trees, K Nearest Neighbors and Logistic Regression:
- First created an object for each of the algorithms then created a GridSearchCV object and assigned them a set of parameters for each model.
- 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.
- 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_.
- Finally using the method score to calculate the accuracy on the test data for each model and plotted a confusion matrix for each using the test and predicted outcomes.

General

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 neighbours and Logistic Regression.

[32]:

	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
- [Applied-Data-Science-Capstone/SpaceX_Machine Learning Prediction_Part_5.ipynb at main · RohiniPatil0/Applied-Data-Science-Capstone]

Results

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

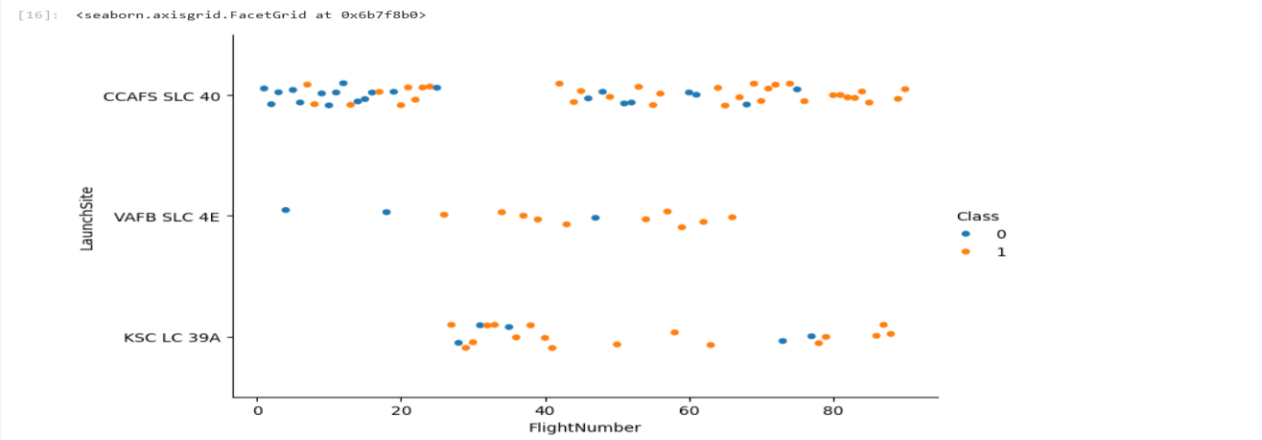
General

Section 2

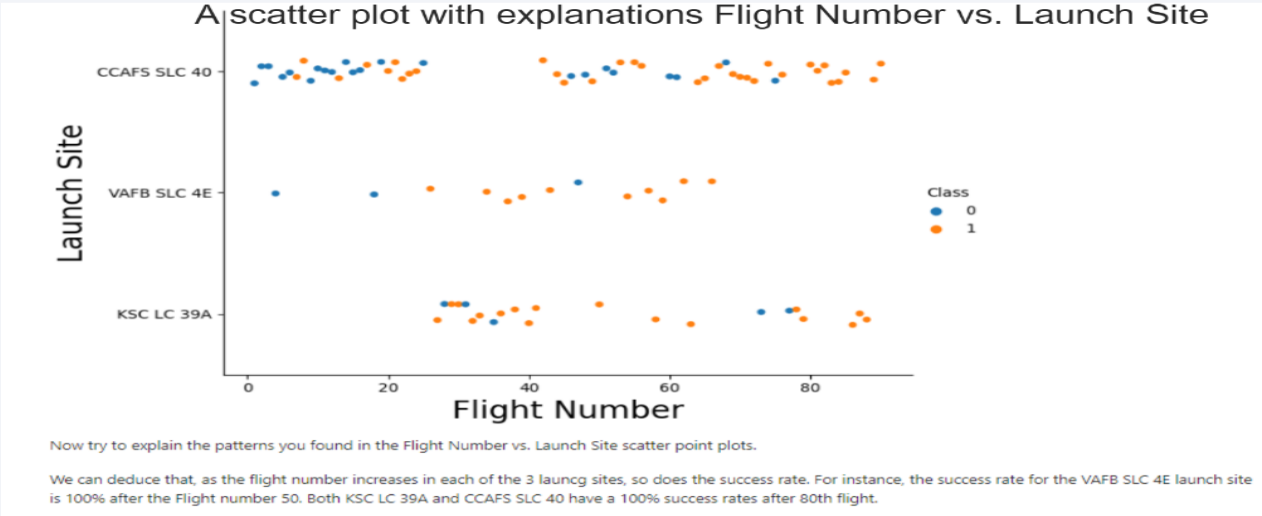
Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site



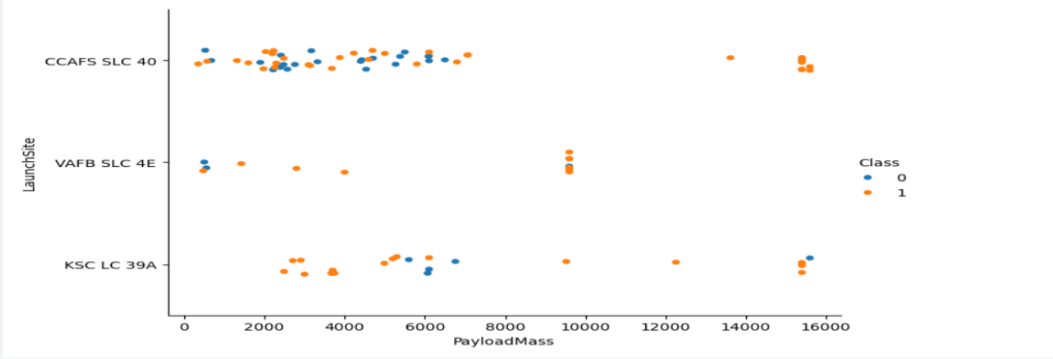
- Show the screenshot of the scatter plot with explanations



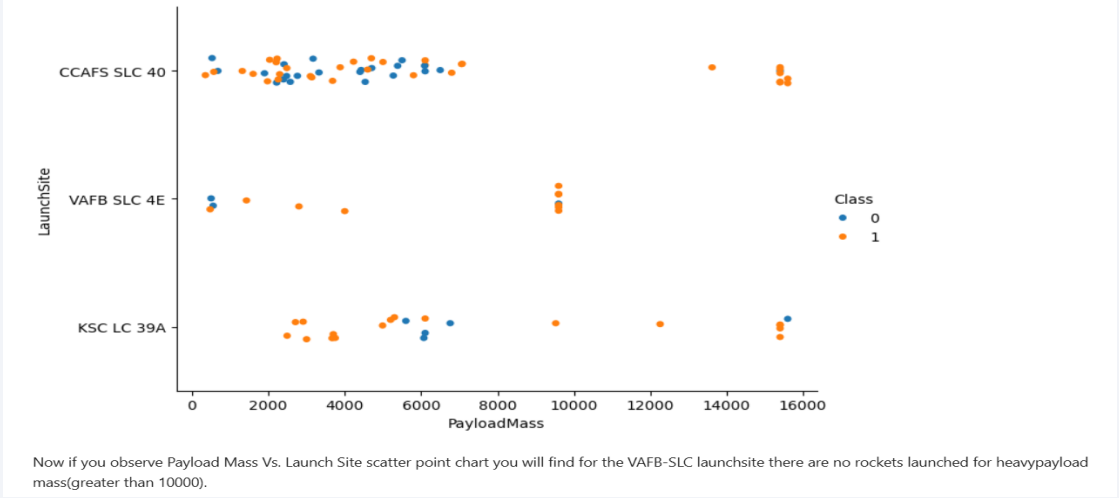
General

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site



- Show the screenshot of the scatter plot with explanations

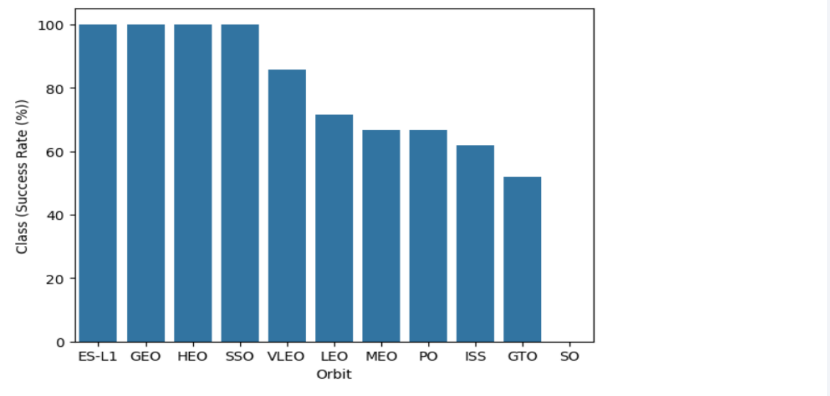


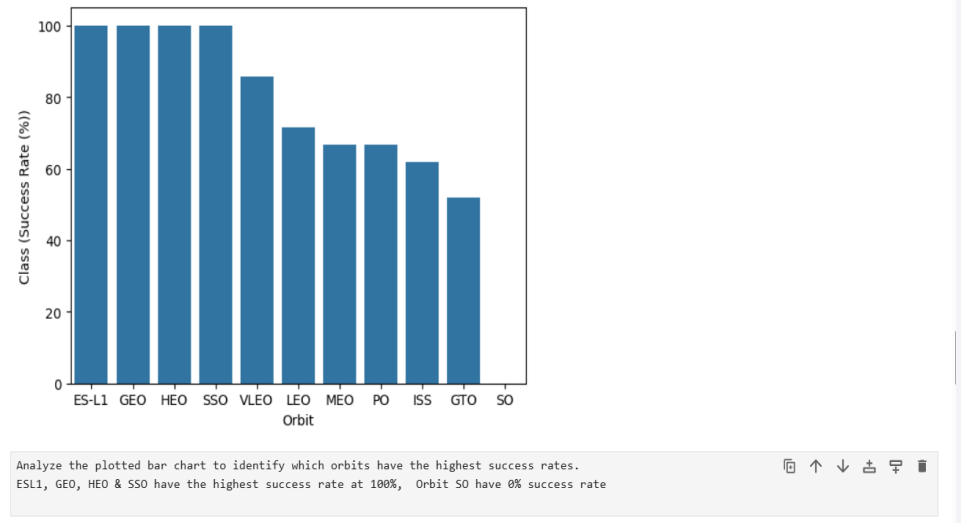
General

Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

- Show the screenshot of the scatter plot with explanations



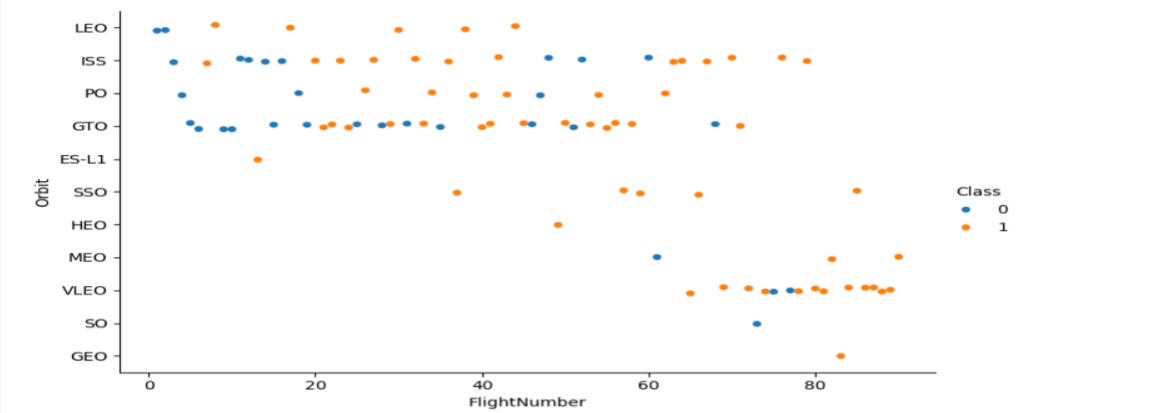


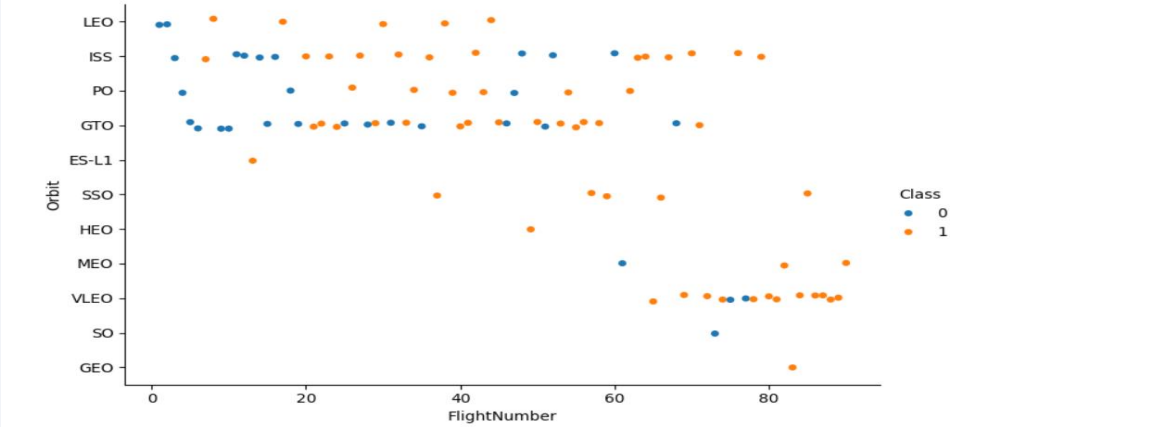
General

Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type

- Show the screenshot of the scatter plot with explanations





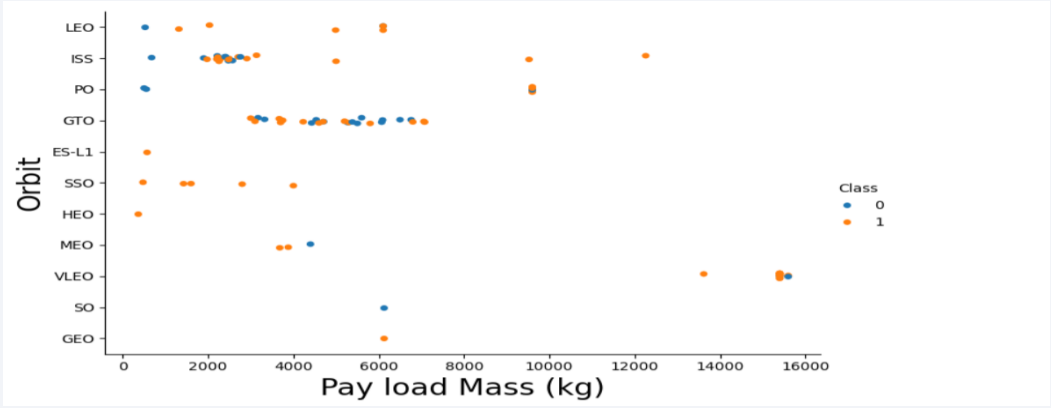
You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

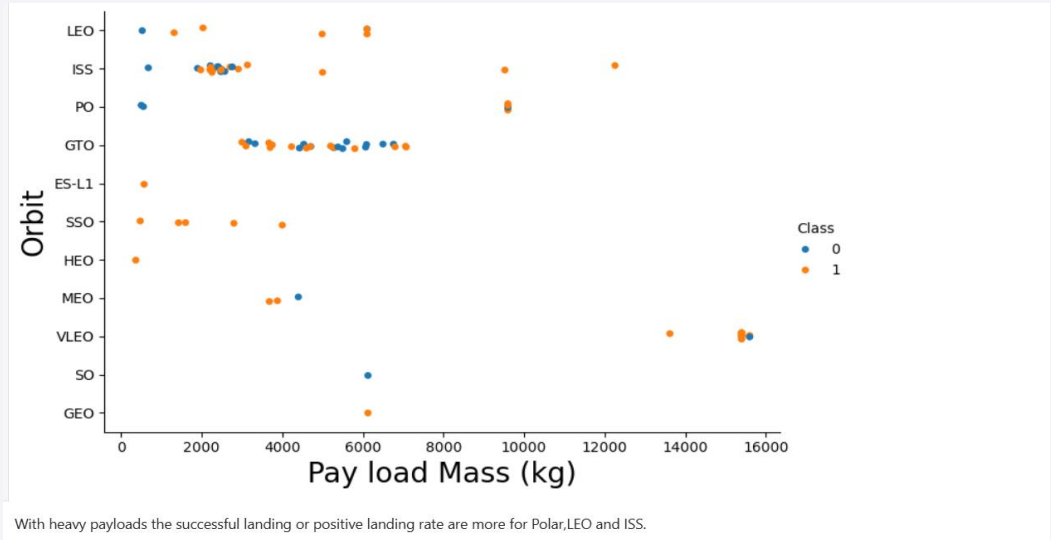
General

Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

- Show the screenshot of the scatter plot with explanations





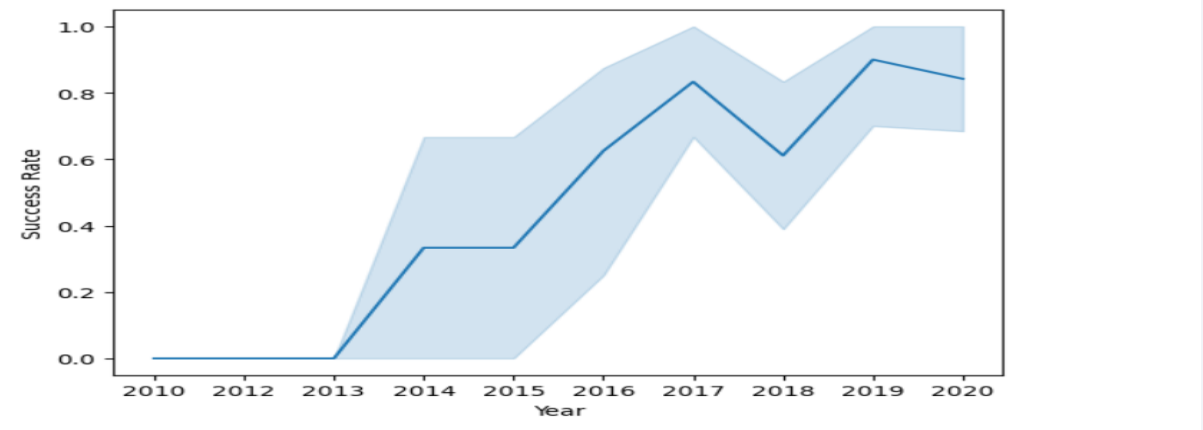
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

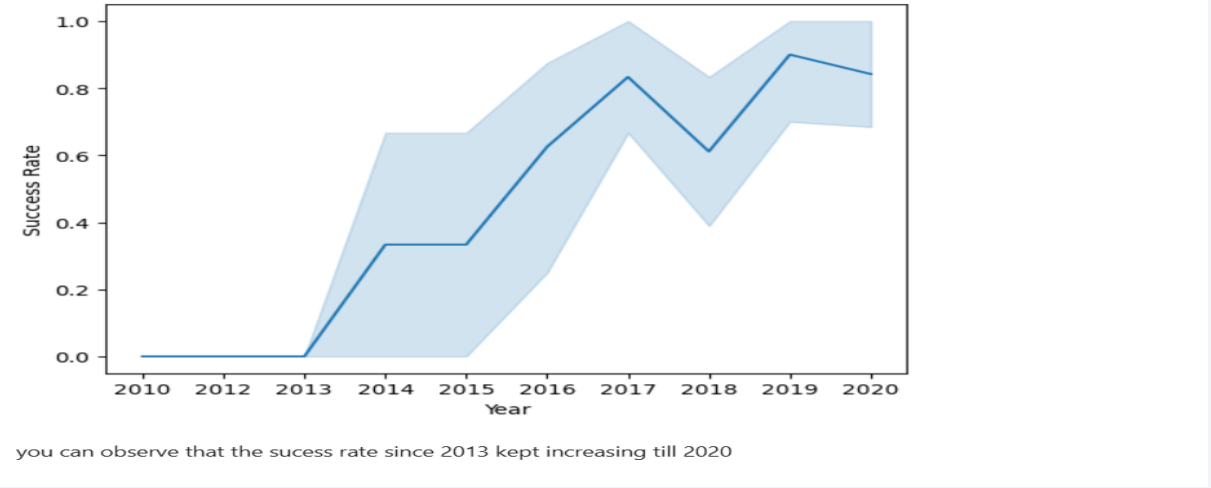
General

Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- Show the screenshot of the scatter plot with explanations





General

All Launch Site Names

- Find the names of the unique launch sites
- Distinct keyword is used to get the unique launch sites from SPACEXTABLE

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

General

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- LIKE keyword is used to get the launch sites begin with ‘CCA’ and LIMIT keyword is used to find only 5 records.

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

General

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- SUM function is used to calculate the total payload carried by boosters from NASA

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

In [12]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass FROM SPACEXTABLE  WHERE Customer = 'NASA (CRS)'
```

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

Out[12]:

TotalPayloadMass
45596

General

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- AVG function is used to calculate the average payload mass

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

In [13]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AvgPayloadMass FROM SPACEXTABLE  WHERE Booster_Version = 'F9 v1.1'
```

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

Out[13]:

AvgPayloadMass
2928.4

General

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Min function is used to find the date of the first successful landing outcome on ground pad

```
In [14]: %sql select min(Date) as date from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)'
 * sqlite:///my_data1.db
Done.
Out[14]:
```

date
2015-12-22

General

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

- Between keyword is used to find the payload mass between 4000 to 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 [15]: ooster_Version from SPACEXTABLE where Landing_Outcome  = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between  4000  AND 6000
 * sqlite:///my_data1.db
Done.
```

Out[15]:

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

General

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Count function is used to find total number of mission outcomes and group by is used to group the successful and failure mission outcomes

In [16]:
```
%sql SELECT MISSION_OUTCOME , COUNT(*) as totalCOunt from SPACEXTABLE GROUP By MISSION_OUTCOME
```

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

Out[16]:

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

General

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Max function is used in the subquery to list the booster names which have carried maximum payload

```
In [17]: %sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
 * sqlite:///my_data1.db
Done.
```

Out[17]:

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

General

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- SELECT CASE WHEN substr(Date, 6, 2) = '01' THEN 'January' WHEN substr(Date, 6, 2) = '02' THEN 'February' WHEN substr(Date, 6, 2) = '03' THEN 'March' WHEN substr(Date, 6, 2) = '04' THEN 'April' WHEN substr(Date, 6, 2) = '05' THEN 'May' WHEN substr(Date, 6, 2) = '06' THEN 'June' WHEN substr(Date, 6, 2) = '07' THEN 'July' WHEN substr(Date, 6, 2) = '08' THEN 'August' WHEN substr(Date, 6, 2) = '09' THEN 'September' WHEN substr(Date, 6, 2) = '10' THEN 'October' WHEN substr(Date, 6, 2) = '11' THEN 'November' WHEN substr(Date, 6, 2) = '12' THEN 'December' END AS month_name, landing_outcome, booster_version, launch_site FROM SPACEXTABLE where Landing_Outcome='Failure (drone ship)' and substr(Date, 0, 5)='2015'
- Substr function is used to get the month and year from Date and CASE is used to display month_name as a string

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 [18]:
```
%sql SELECT CASE WHEN substr(Date, 6, 2) = '01' THEN 'January' WHEN substr(Date, 6, 2) = '02' THEN 'February' WHEN substr(Da
```

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

Out[18]:

month_name	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

General

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
- SELECT case when landing_outcome = 'Success (ground pad)' then 'Success (ground pad)' when landing_outcome = 'Failure (drone ship)' then 'Failure (drone ship)' end as landing_status, COUNT(*) as outcome_count, RANK() OVER (ORDER BY COUNT(*) DESC) AS rank from SPACEXTABLE where date between '2010-06-04' and '2017-03-20' and (landing_status = 'Failure (drone ship)' OR landing_status = 'Success (ground pad)') group by landing_status ORDER BY outcome_count desc
- COUNT is used to get the total count and rank is used to get the rank from SPACEXTable based on count and ORDER by is used to display the records in descending order

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 [19]:
```
%sql SELECT case when landing_outcome = 'Success (ground pad)'  then 'Success (ground pad)' when landing_outcome = 'Failure
```

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

Out[19]:

landing_status	outcome_count	rank
Failure (drone ship)	5	1
Success (ground pad)	3	2

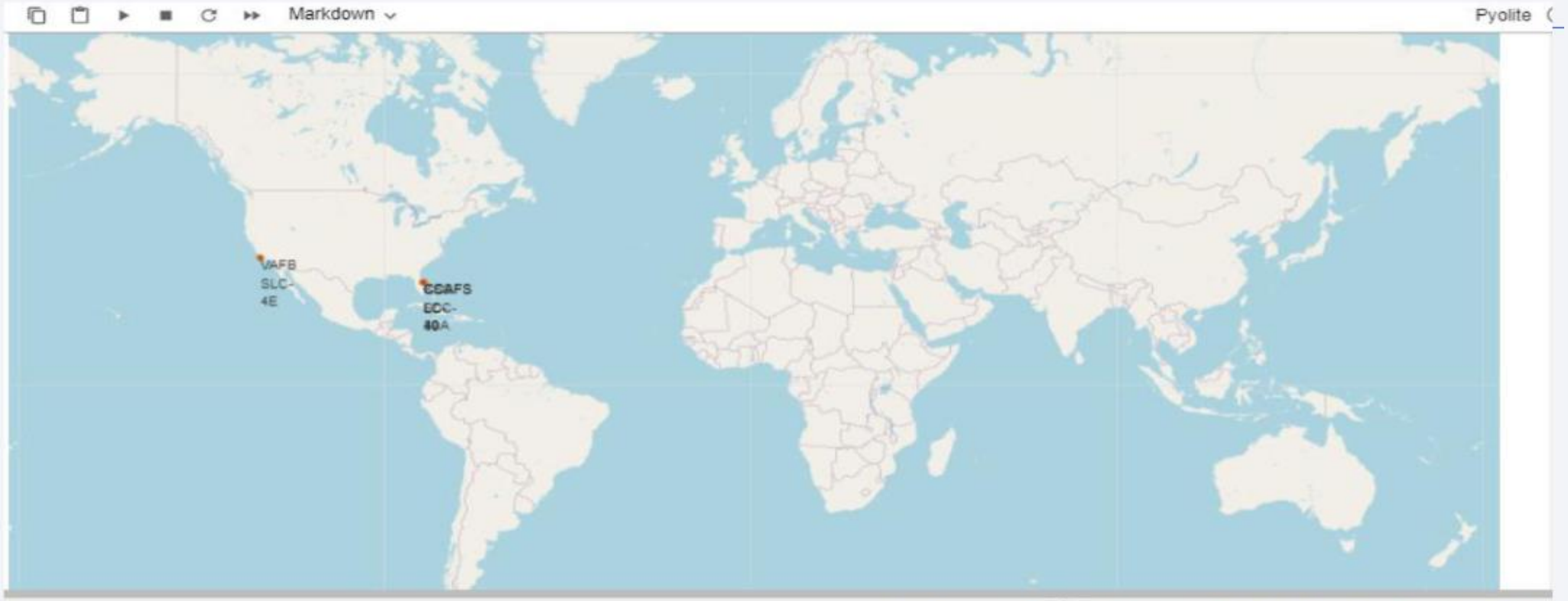
General

Section 3

Launch Sites Proximities Analysis

General

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.

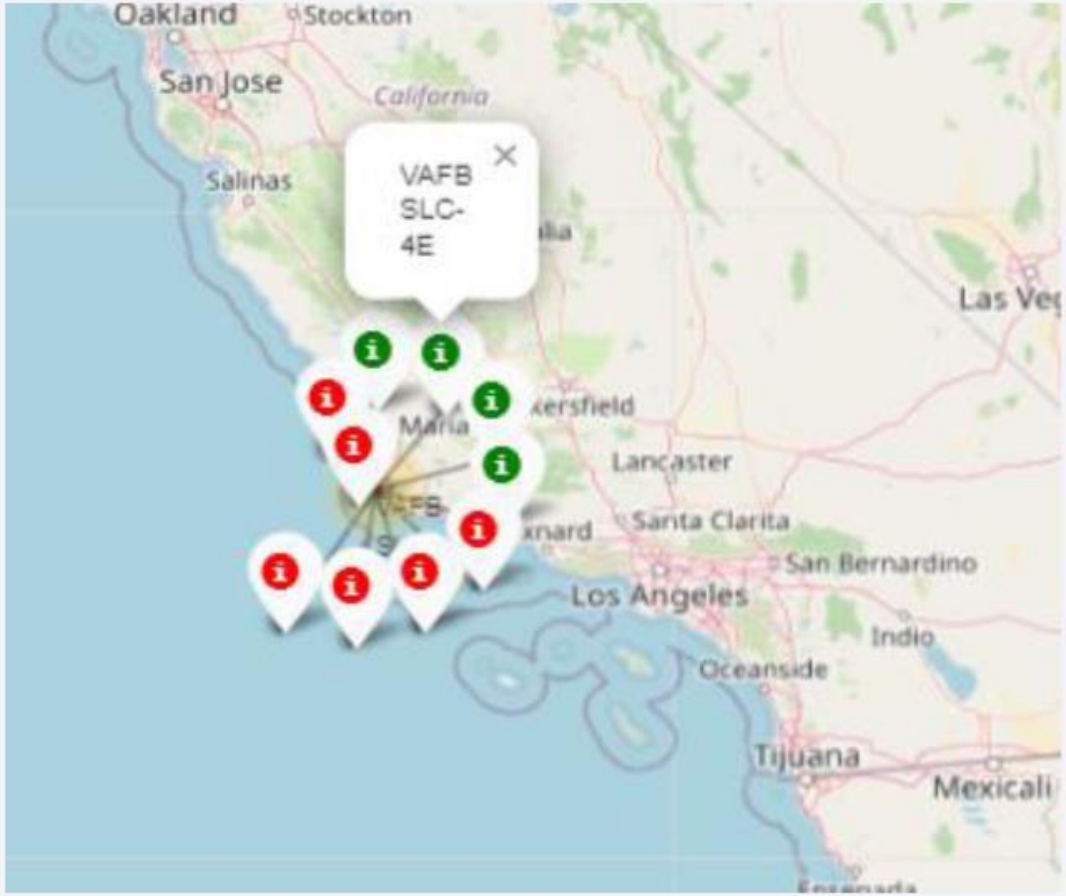
General

Launch Outcomes for each site on the map with color markers

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

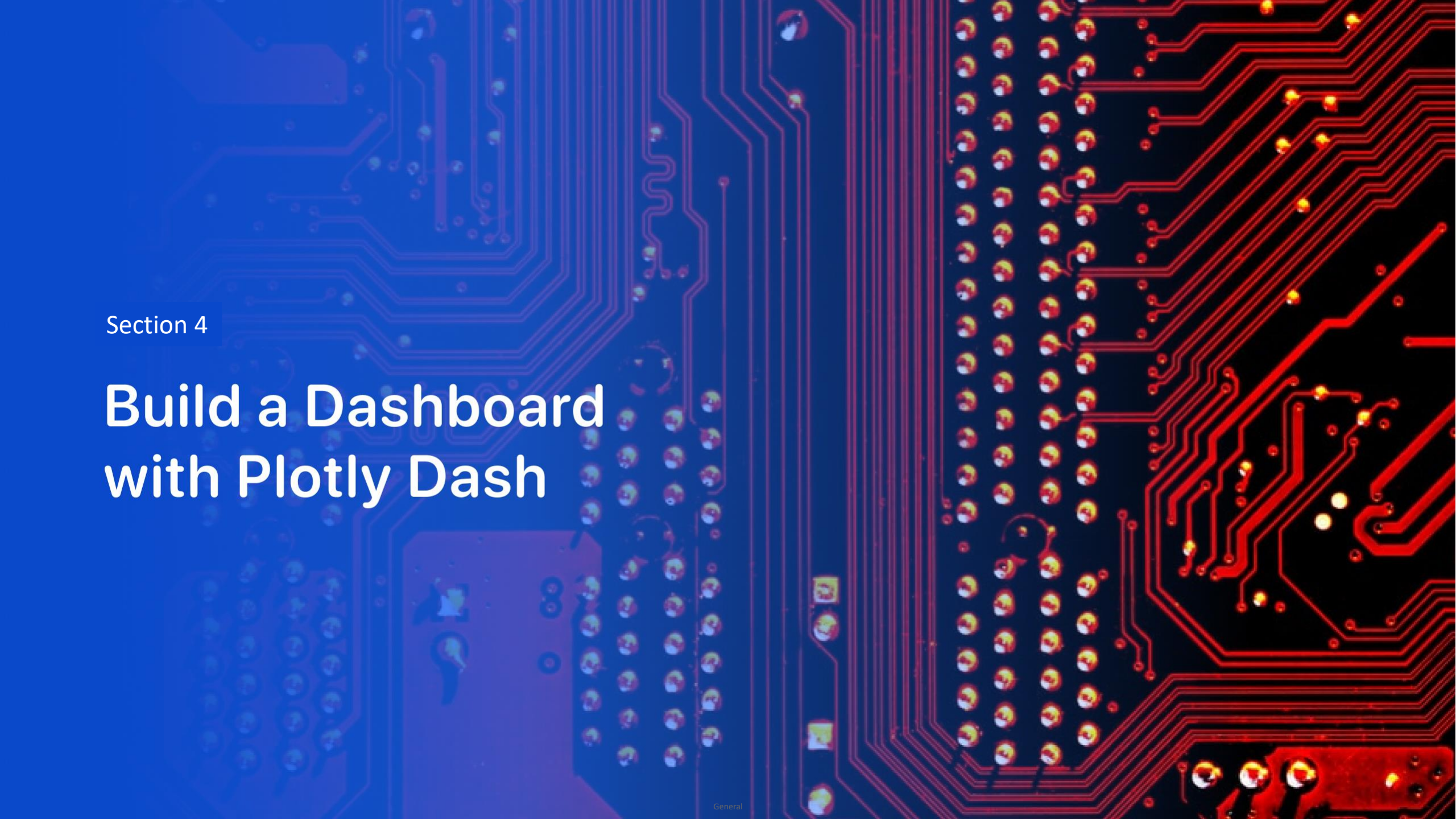
General

Launch Outcomes for Each Site on the Map with Color Markers



- In the West Coast (Californai) Launch site VAFB SLC-4E has relatively lower success rates 4/10 compared to KSC LC-39A launch site in the Eastern Coast of Florida.

General



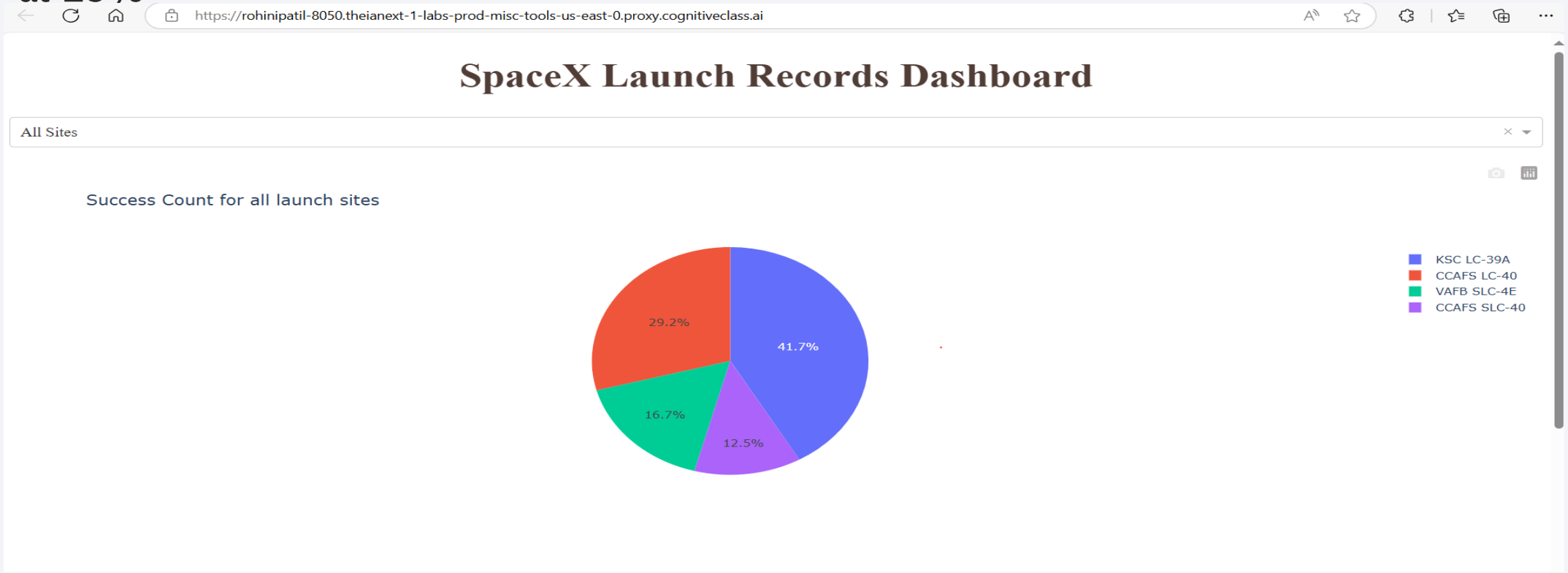
Section 4

Build a Dashboard with Plotly Dash

General

Pie Chart for All Sites Launch Success Count

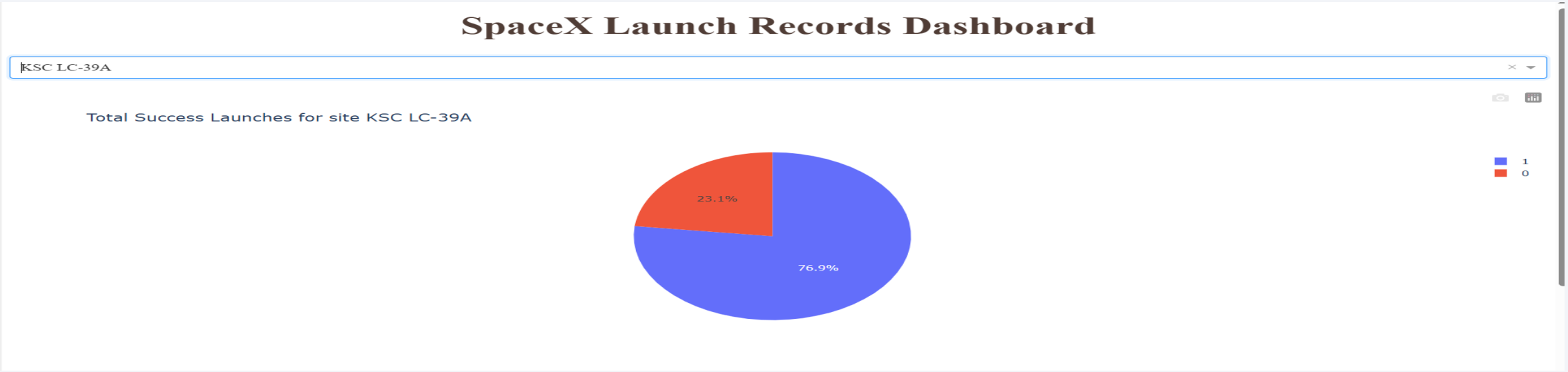
Launch Site KSCLC-39A has the highest Success rate at 42% followed by CCAFS LC 40 at 29%



General

Pie chart for Highest Success rate launch Site

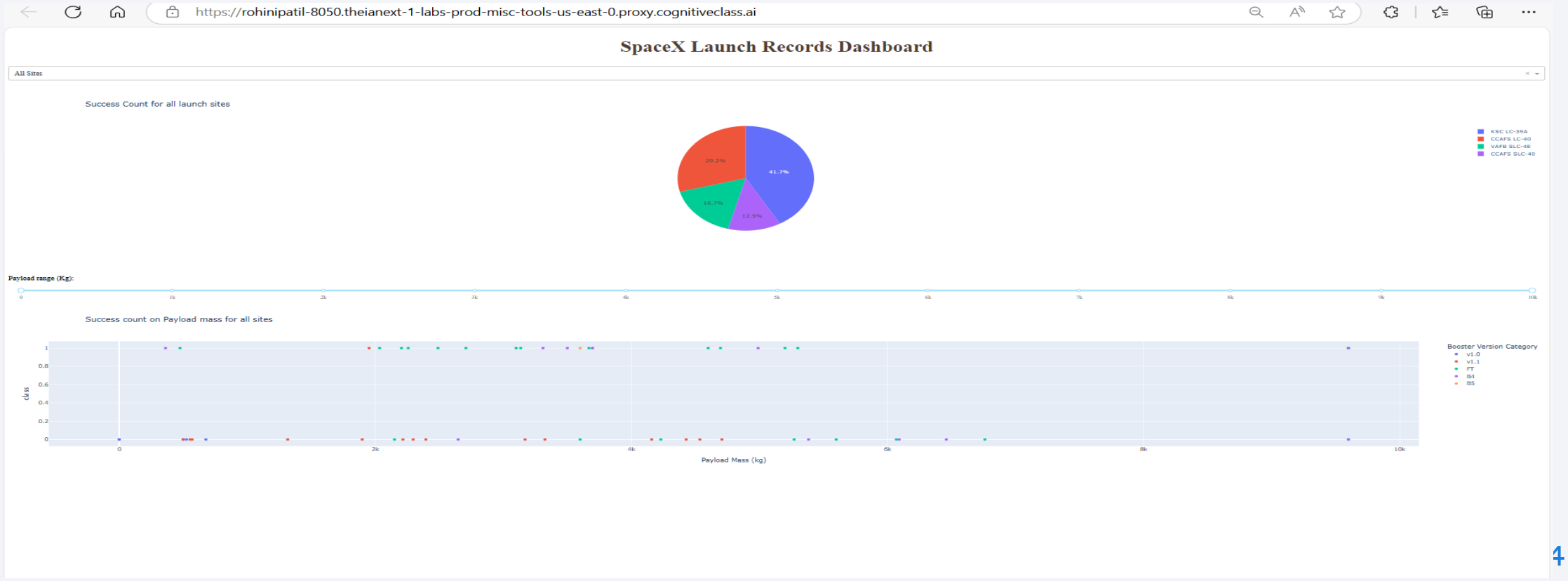
KSC LC-39A has the Highest Success rate at 76.9% against 23.1% failed launches.



General

Payload vs Launch Outcome scatter plot for all sites

- Booster version FT has the largest success rate



General

Section 5

Predictive Analysis (Classification)

General

Classification Accuracy

- All the methods perform equally on the test data. They all have the same accuracy of 0.833333 on the test data

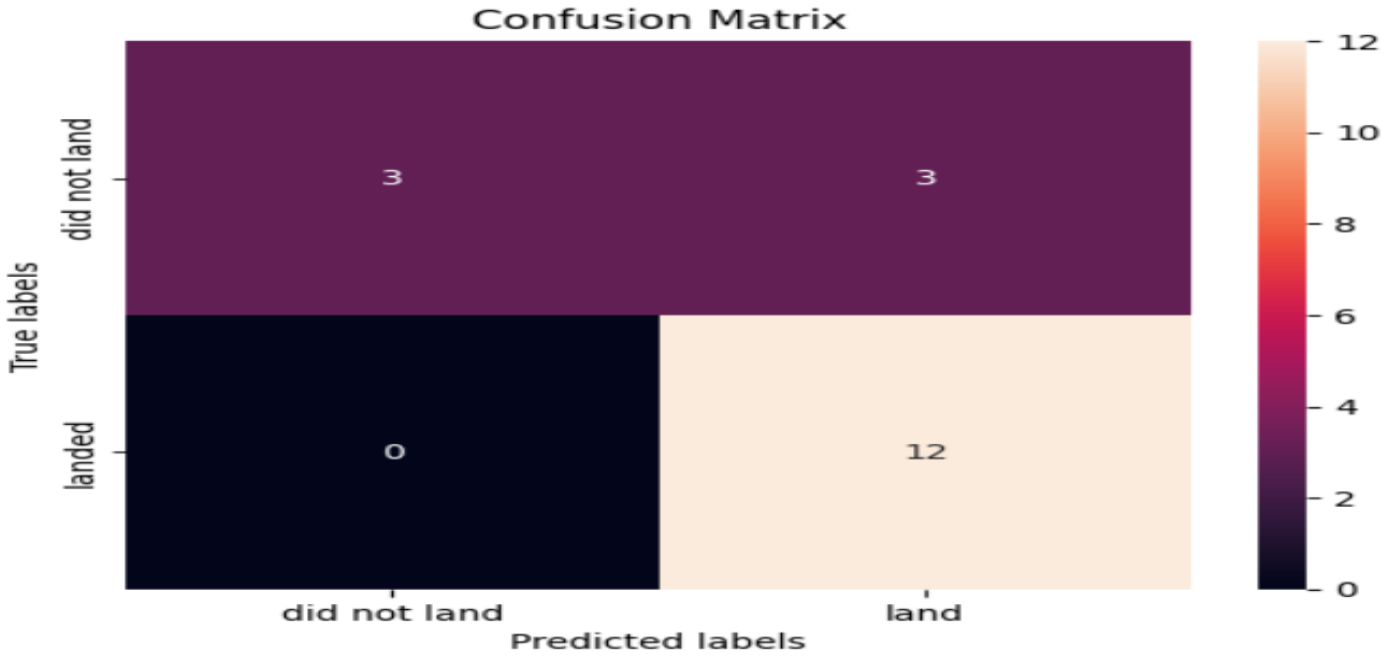
[32]:

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

General

Confusion Matrix

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



General

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 heavy payload 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.
- With heavy payloads, the successful landing or positive landing rate is 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.
- And finally, the success rate since 2013 kept increasing till 2020.

General

Thank you!

General