IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tattikota Praneeth
16th June 2024

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data Collection using SPACEX API
  - Data Collection using Web Scraping
  - Data Wrangling
  - Exploratory Analysis using SQL
  - Exploratory Data Analysis with Visualization
  - Interactive Visual Analytics with Folium lab
  - Interactive Dashboard with Plotly Dash
  - Predictive Analysis with Machine Learning Models
- **Summary of all results**
  - Exploratory Data Analysis results
  - Interactive Visual Analytics and Dashboard
  - Predictive Analysis(Classification)

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 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 Falcon 9 data was collected
  - Data was first collected using SPACEX by Requesting and parsing the SpaceX launch data using the GET request
  - Then, we define some series of helper functions that helps in use of API to extract information using identification numbers in launch data.
  - Request rocket launch from SPACEX API URL.
  - Decode the response content as a Json result which is later converted into a Pandas data frame.
  - Also Performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled **List of Falcon 9 and Falcon Heavy launches** of the launch records are stored in HTML.
  - Extracted the Falcon 9 HTML table records and parsed the table.
  - Converted it into a Pandas data frame.

Data Collection – SpaceX API

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:

```
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

```
response.status_code
```

```
200
```

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

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

- Collected data by making a get request to SPACEX API and parsed it and then decoded it to a Json result.
- Github URL : https://github.com/HteenarpT/Applied-Data-Science-Capstone-Project/blob/main/SPACEX%20-%20Data%20Collection%20API.ipynb

Data Collection – Web Scraping

- Requested the Falcon 9 Launch Wiki page from its URL and extracted column nam es from HTML table header.
- Github URL :

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.

```
# 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`

```
# 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

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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

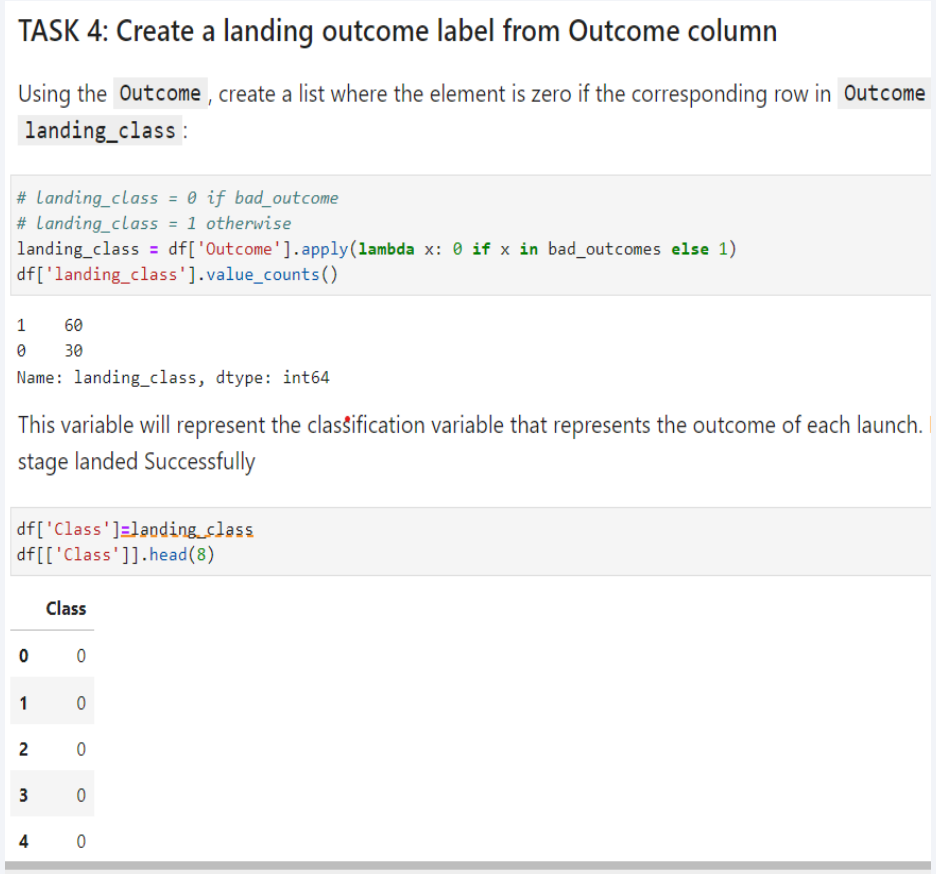
Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`

```
# 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')
```

Data Wrangling

- After obtaining and creating a data frame the collected data is filtered using **BoosterVersion** column to only keep falcon 9 launches in data, for **PayloadMass** column we replace the NULL values with mean value of the column.
- Performed Exploratory Data Analysis to find some patterns in the data and determine the label for training supervised models.
- GitHub
  URL : https://github.com/HteenarpT/AppliedDataScienceCapstoneProject/blob/main/SPACEX%20-%20Data%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`; otherwise it's one. Then assign it to the variable `landing_class`:

```
# 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)
df['landing_class'].value_counts()
```

```
1    60
0    30
Name: landing_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 landed Successfully

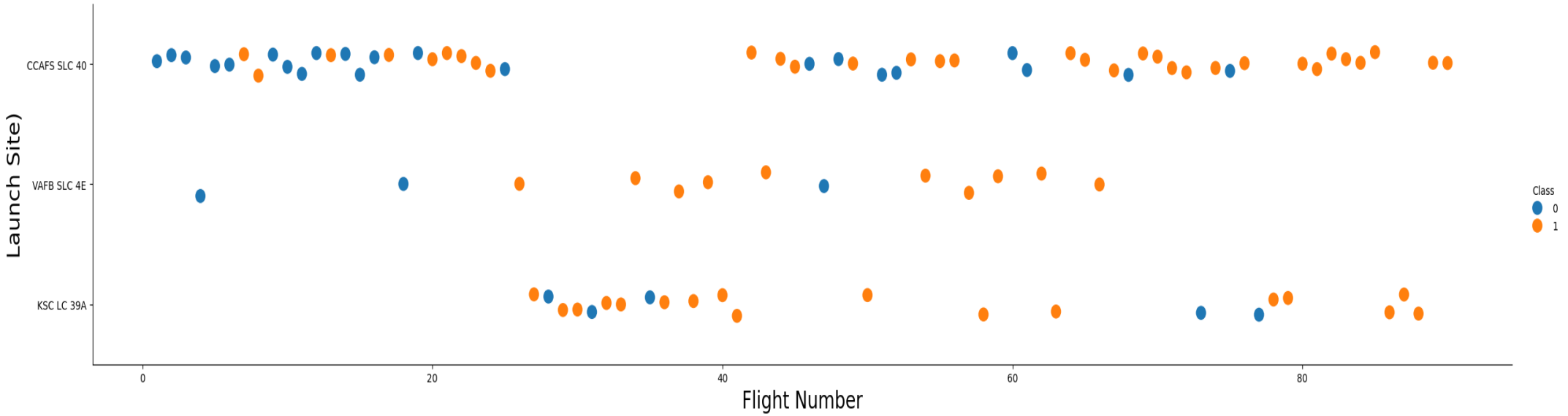
```
df['Class']=landing_class
df[['Class']].head(8)
```

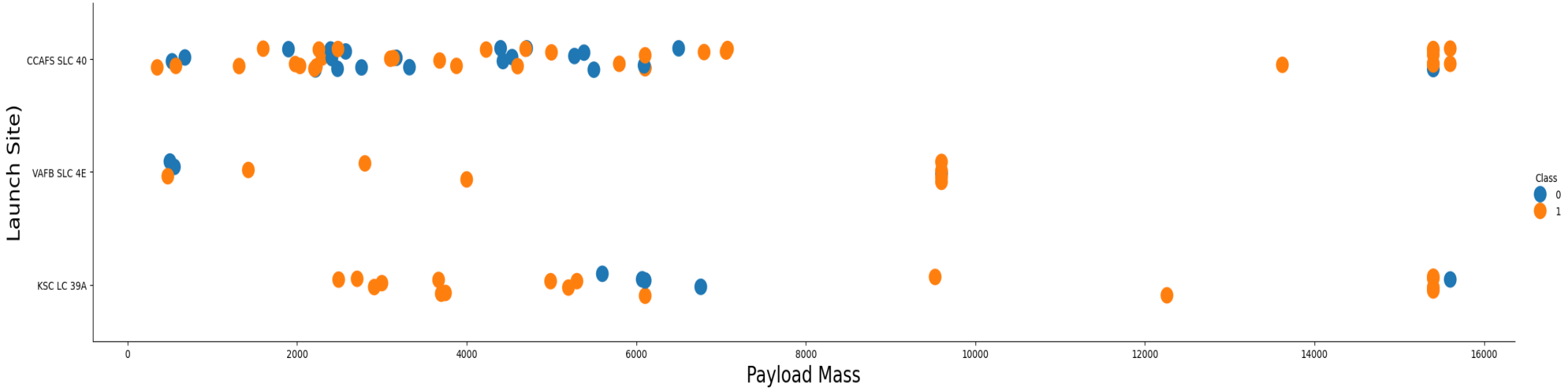
	Class
0	0
1	0
2	0
3	0
4	0

EDA with Data Visualization

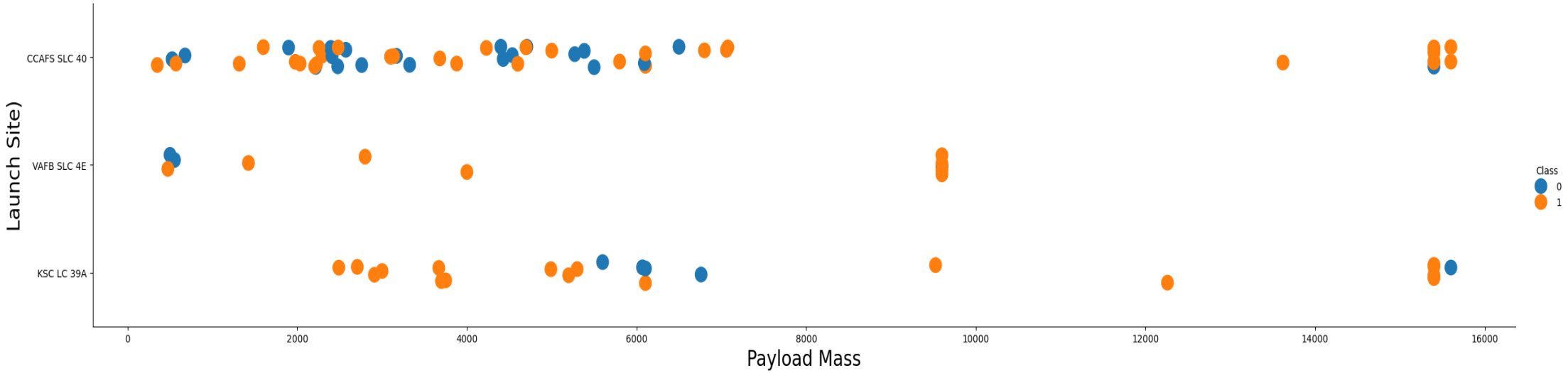
- Performed data Analysis and Feature Engineering using Pandas and Matplotlib.
- Plots used for visualization :
  - Line plot – To Visualize launch success yearly trend
  - Bar Chart – To Compare success rate of each orbit.
  - Scatter plots – To Visualize relationship between :
    - Flight Number and Launch Site
    - Payload and Launch Site
    - Flight Number and Orbit Type
    - Payload and Orbit Type
- GitHub URL : https://github.com/HteenarpT/Applied-Data-Science-Capstone-Project/blob/main/SPACEX%20-%20EDA%20with%20Visualization.ipynb

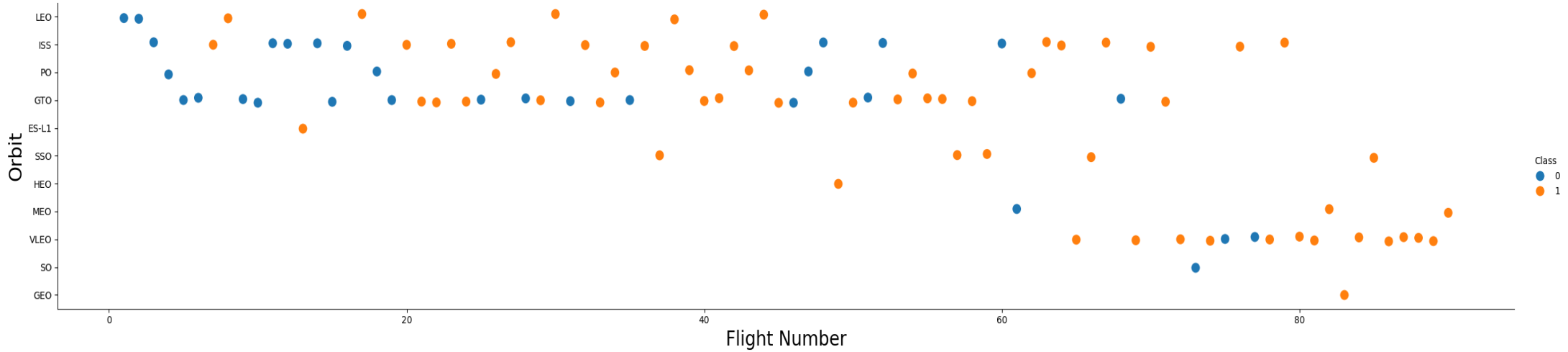
EDA with Data Visualization(Cont'd...)



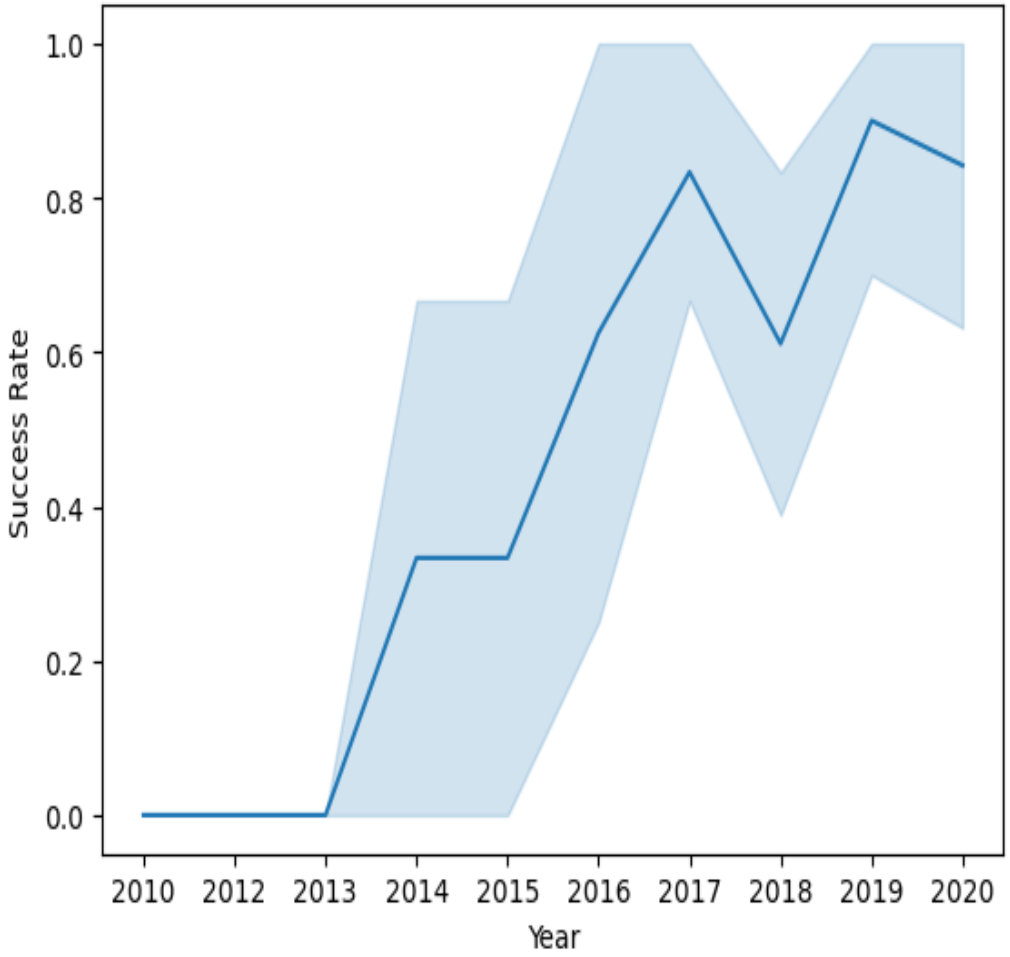


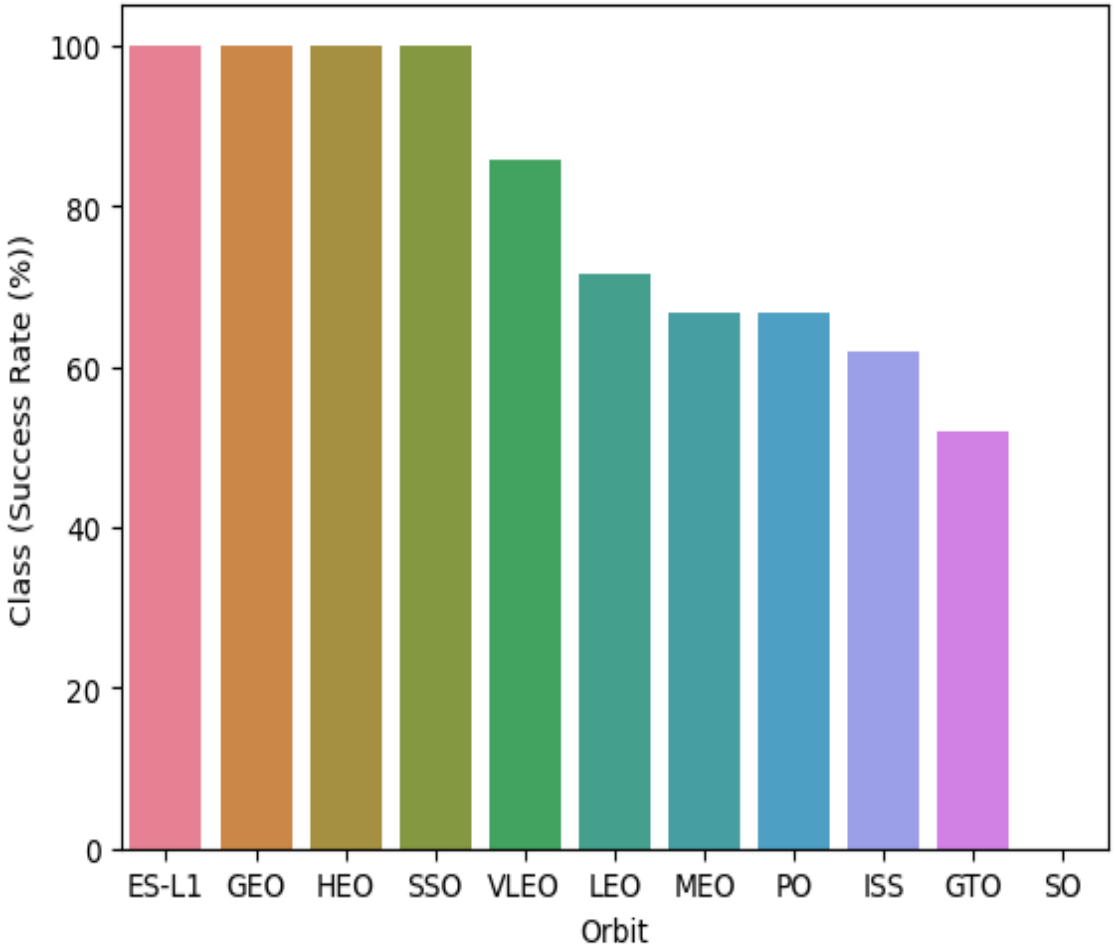
EDA with Data Visualization(Cont'd...)





EDA with Data Visualization(Cont'd...)





EDA with SQL

- SQL queries performed :
  - Displayed the names of the unique launch sites in the space mission.
  - Displayed 5 records where launch sites begin with the string 'CCA'.
  - Displayed the total payload mass carried by boosters launched by NASA (CRS).
  - Displayed average payload mass carried by booster version F9 v1.1
  - Listed the date when the first successful landing outcome in ground pad was achieved.
  - Listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.

EDA with SQL(Cont'd...)

- SQL queries performed :
  - Listed the total number of successful and failure mission outcomes.
  - Listed the names of the booster_versions which have carried the maximum payload mass using a subquery.
  - Listed the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

EDA with SQL(Cont'd...)

- Ranked 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.
- Github URL : https://github.com/HteenarpT/Applied-Data-Science-Capstone-Project/blob/main/SPACEX%20-%20EDA%20with%20SQL.ipynb

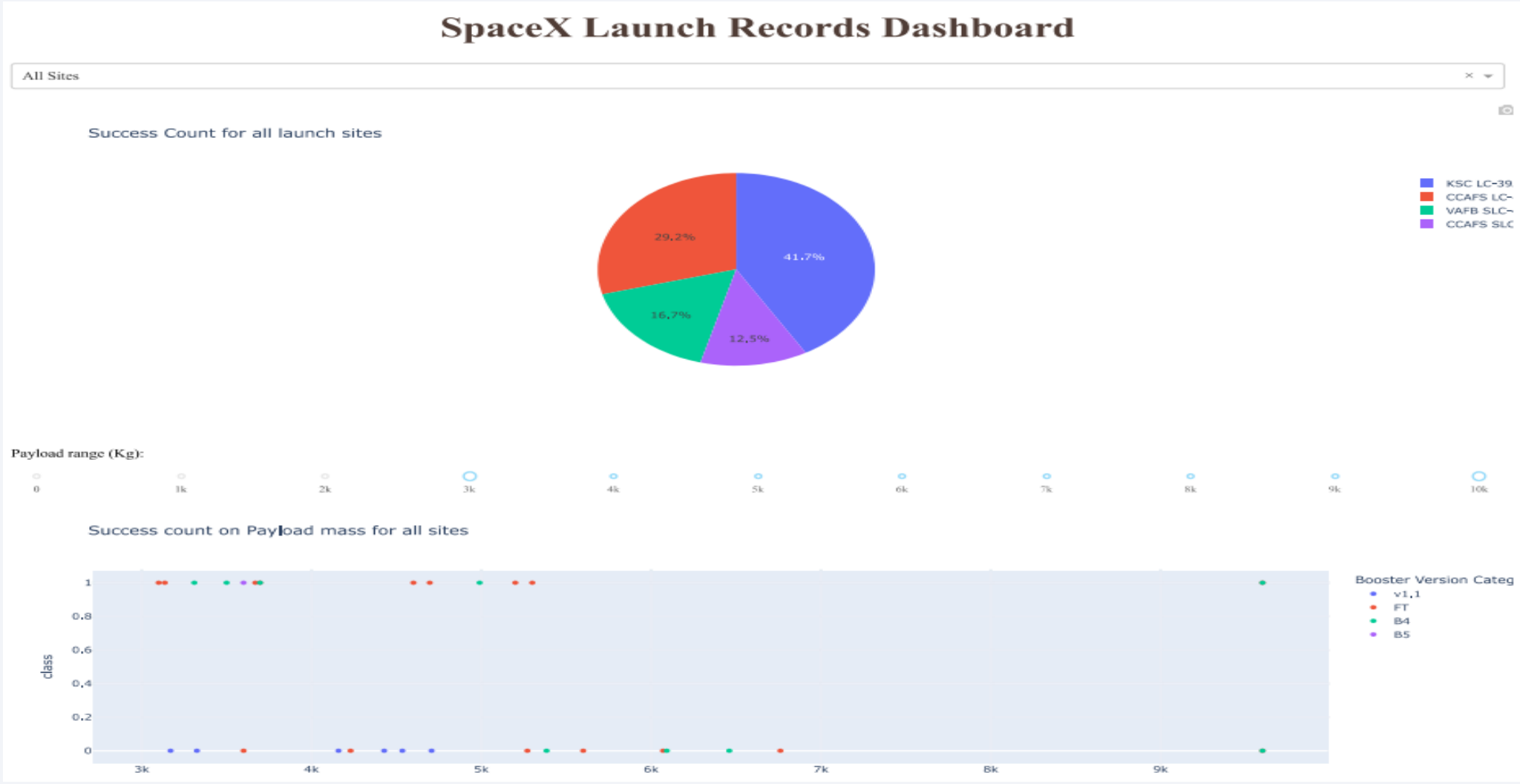
Build an Interactive Map with Folium

- Map objects created and added to Folium map :
  - Used folium.Circle to highlight circle area with a text label on specific coordinate.
  - Created and added folium.Circle and folium.Marker for each launch site on site map.
  - Marked the success/failed launches for each site on the map
  - Created a new column in launch_sites dataframe called marker_color to store the marker colors based on the class value.
  - For each launch result in spacex_df data frame, added a folium.Marker to marker_cluster.
  - Calculate the distances between a launch site to its proximities.
  - Drew a PolyLine between a launch site to the selected coastline point.
  - Mark down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site.
- https://github.com/HteenarpT/AppliedDataScienceCapstoneProject/blob/main/SPACEX%20%20Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- Built a Dashboard with Plotly Dash by :
  - Added a Launch Site dropdown input component.
  - Added a callback function to render success pie chart based on selected site.
  - Added a range slider to select different Payload Mass.
  - Added a callback function to render success Payload Mass scatter plot.
- GitHub
  URL : https://github.com/HteenarpT/AppliedDataScienceCapstoneProject/blob/main/SPACEX%20%20Interactive%20Dashboard%20with%20Ploty%20Dash.py

Dashboard Built using Plotly Dash



Predictive Analysis (Classification)

- Finding the best performing classification model by :
  - Loaded the dataframe into a variable X.
  - Created a Numpy array from the column class in data by applying the method to_numpy() then assign it to a variable Y.
  - Standardized the data in X and reassigned the data back to the same variable.
  - Used the function train_test_split to split the data into training and testing data.
  - Created various objects including logistic regression, SVM, Decision Tree and KNN , used grid search to find the best parameters for each object created.
  - Calculated Accuracy on test data for each object created.
- GitHub
  URL: https://github.com/HteenarpT/AppliedDataScienceCapstoneProject/blob/main/SPACEX%20%20Machine%20Learning%20Model%20Prediction.ipynb

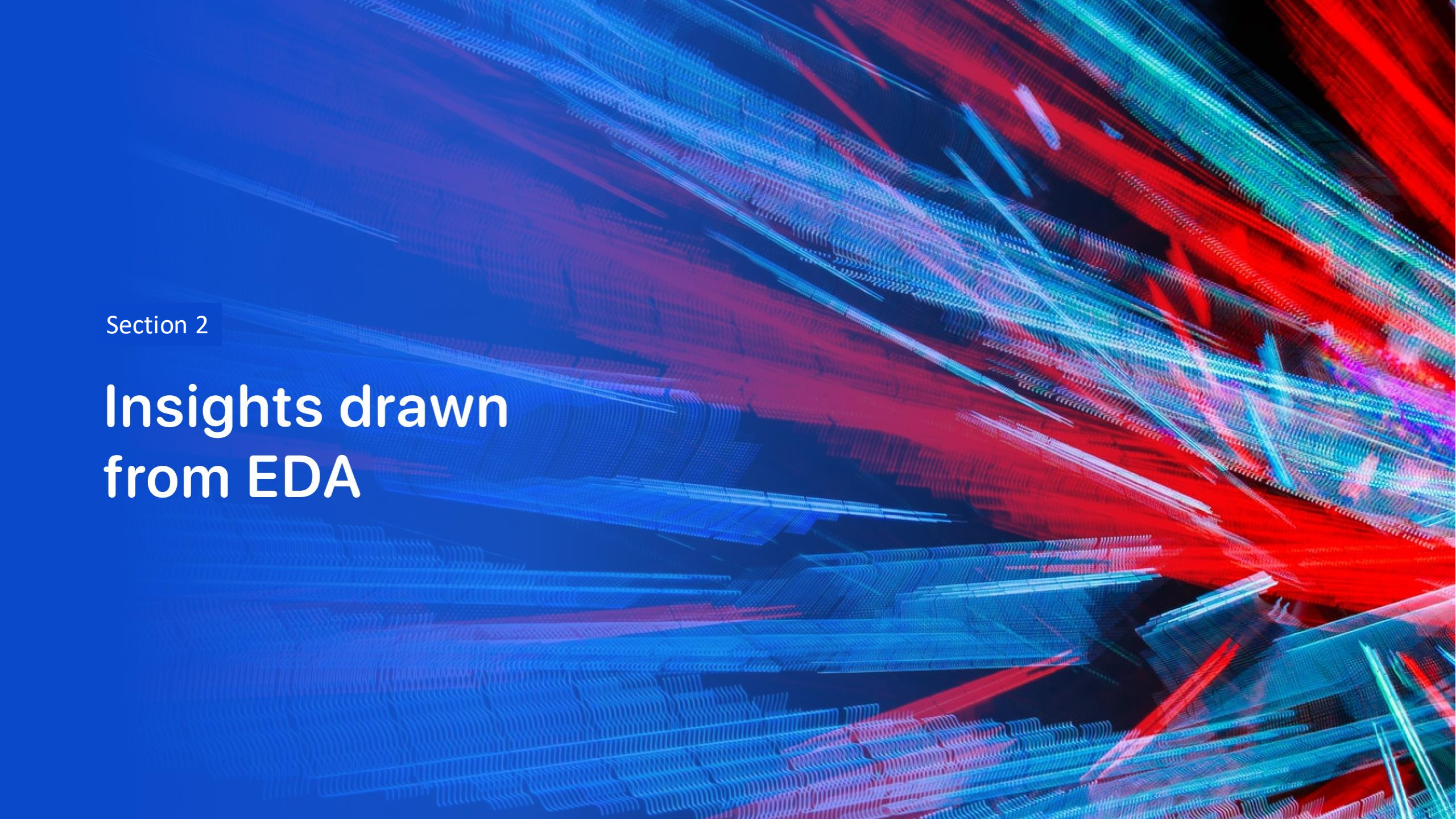
Predictive Analysis (Classification)

- The table below shows the accuracies computed for each of the model used on their test data when used best possible parameters by performing Grid Search method.

	Model	Accuracy
0	Logistic_Regression	0.833333
1	SVM_accuracy	0.833333
2	Decision_tree_accuracy	0.722222
3	knn_accuracy	0.833333

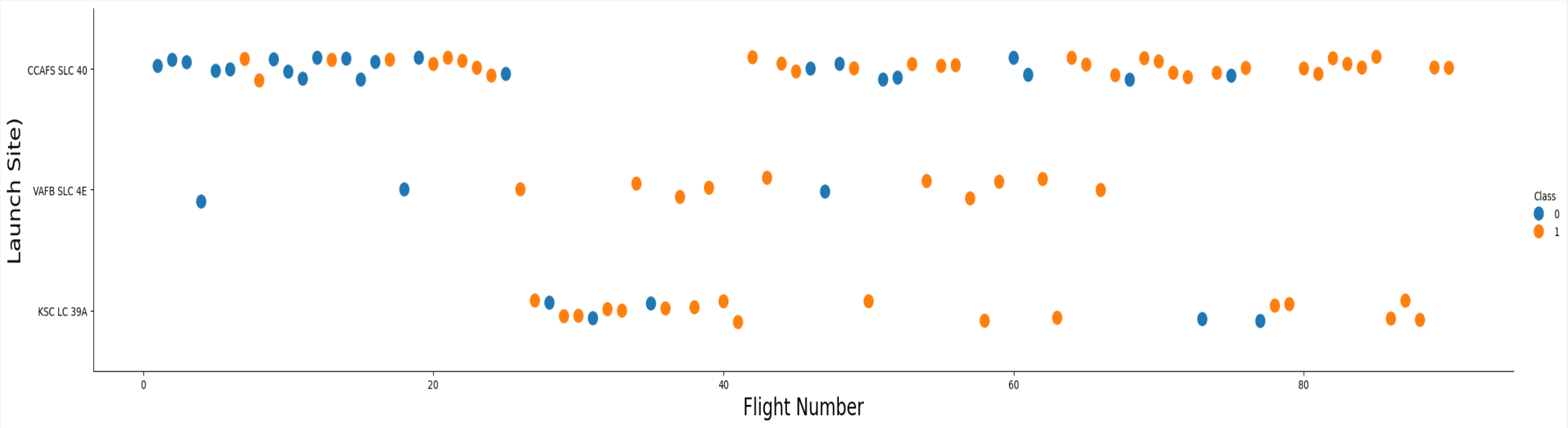
Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

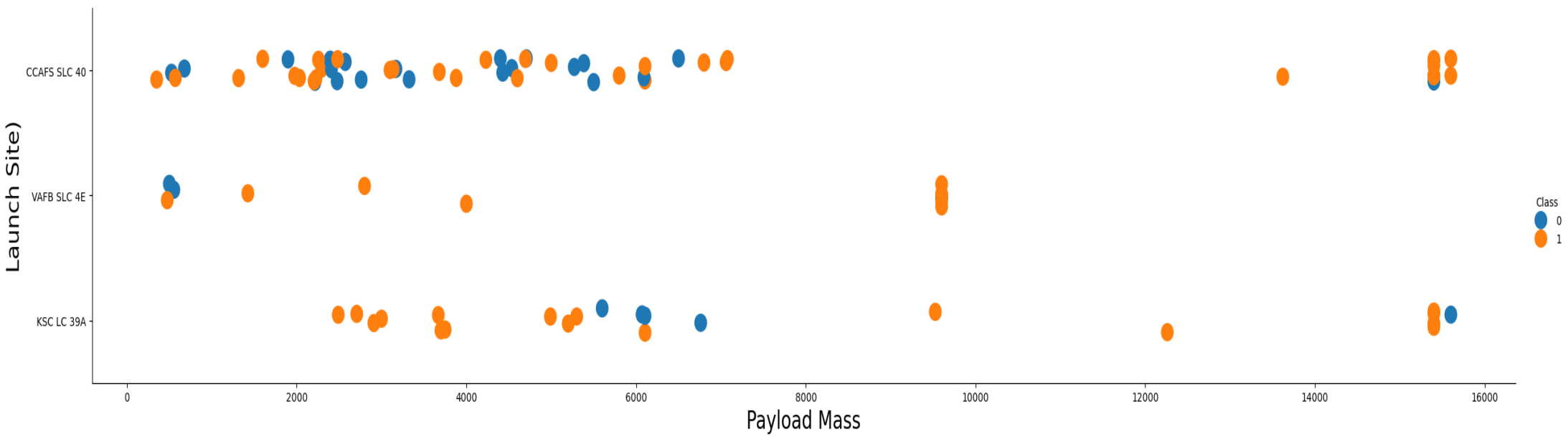
- Scatter plot of Flight Number vs. Launch Site



- As the flight number increases in each of the launch sites , even the success rate does show increase in its rate.

Payload vs. Launch Site

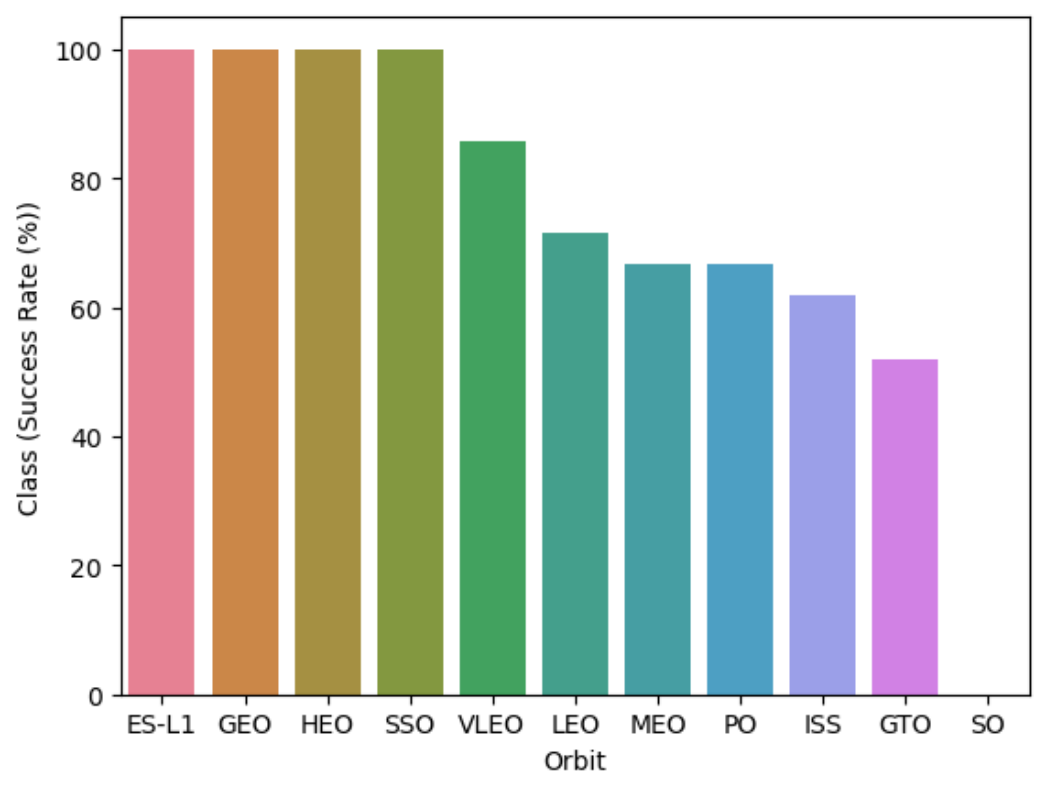
- Scatter plot of Payload vs. Launch Site



- if we observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).

Success Rate vs. Orbit Type

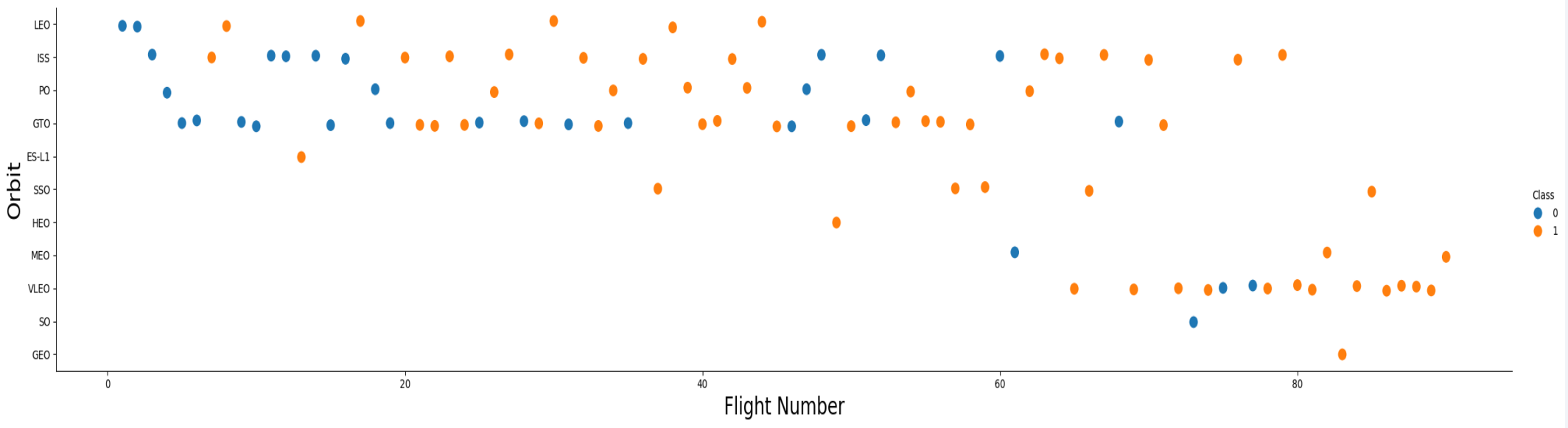
- Bar chart for the success rate of each orbit type



- Interpretation :
  - Orbits with high success rate are ES-L1,GEO,HEO,SSO and the least success rate is for GTO and no success rate for SO type orbit.

Flight Number vs. Orbit Type

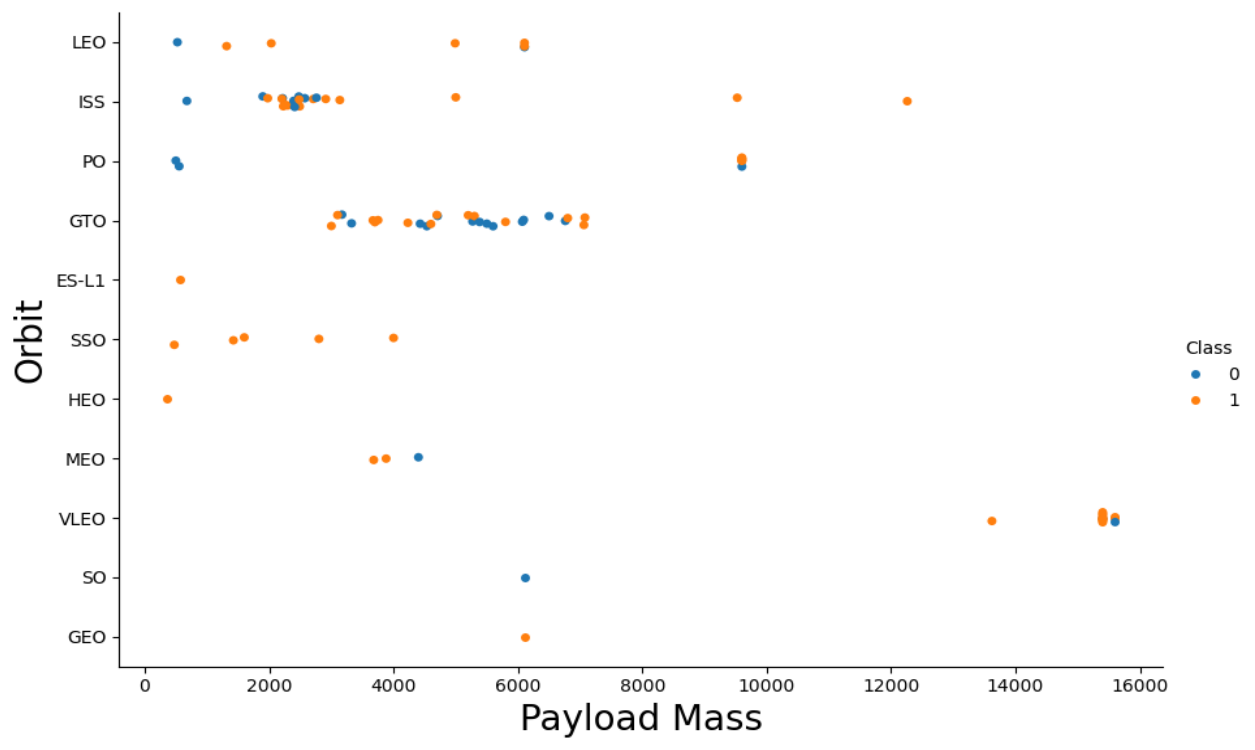
- Scatter point of Flight number vs. Orbit type



- 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

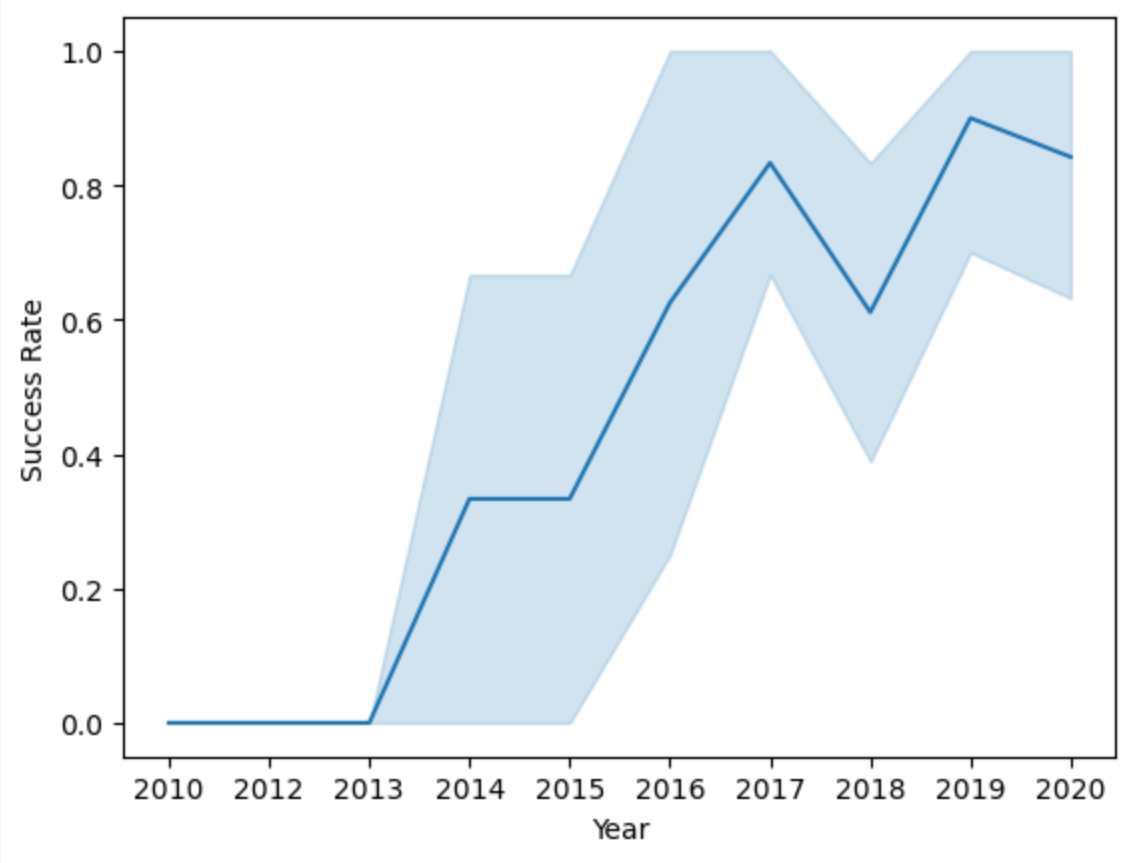
- Scatter point of payload vs. orbit type



- 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

- Line chart of yearly average success rate



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

All Launch Site Names

- Names of unique launch sites :

Task 1

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

```
%sql select distinct Launch_Site from SPACEXTABLE
```

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

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

- Used **SELECT DISTINCT** to select only the unique launch site names from the SPACEXTABLE Dataframe. They are 4 launch sites available.

Launch Site Names Begin with 'CCA'

- 5 records where launch sites begin with `CCA`

```
%sql select * from SPACEXTABLE where Launch_Site  like 'CCA%' limit 5
```

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

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

- Using **LIKE** keyword followed by 'CCA%' only the launch site starting with CCA are selected and limited the number of records by setting **LIMIT** to 5.

Total Payload Mass

- Total payload carried by boosters from NASA.

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

```
%sql select sum(Payload_mass__kg_) from SPACEXTABLE where Customer=='NASA (CRS)'
```

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

sum(Payload_mass__kg_)
45596

- Used the **SUM** function on the column Payload_mass__kg_ to obtain the total payload carried by boosters from NASA.

Average Payload Mass by F9 v1.1

- 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_) from SPACEXTABLE where Booster_Version like 'F9 v1.1%'
```

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

avg(Payload_mass__kg_)
2534.6666666666665

- By using the function **AVG** on the column Payload_mass__kg_ , we obtain the average of payload mass carried by booster version F9 v1.1

First Successful Ground Landing Date

- Date of the first successful landing outcome on ground pad

```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome like 'Success (ground pad)'
```

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

min(Date)
2015-12-22

- We select the minimum date from all the records which have a success on ground pad by using **MIN** function on Date column.

Successful Drone Ship Landing with Payload between 4000 and 6000

- List of names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%sql select Booster_Version,Payload_mass__kg_,Landing_Outcome from SPACEXTABLE where Landing_Outcome like 'Success (drone ship)' and Payload_mass__kg_ between 4000 and 6000
```

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

Booster_Version	PAYLOAD_MASS__KG_	Landing_Outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

- We specify range of values for Payload_mass__kg_ column by using **BETWEEN** keyword between maximum and minimum values needed for the column and also **LIKE** similar to the above queries.

Total Number of Successful and Failure Mission Outcomes

- Total number of successful and failure mission outcomes

```
%sql select distinct Mission_outcome,COUNT(mISSION_OUTCOME) from SPACEXTABLE group by mission_outcome
```

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

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

- We use **GROUP BY** on the column mission_outcome for different classes available in the column and apply COUNT function on the respective groups.

Boosters Carried Maximum Payload

- Names of the booster which have carried the maximum payload mass

```
%sql select Booster_Version,PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_=(SELECT max(Payload_mass__kg_) from SPACEXTABLE)
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

- We use **MAX** function in a subquery to obtain the names of the booster which have carried the maximum payload.

2015 Launch Records

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

```
%sql select substr(Date,6,2) as month,booster_version,launch_site,Landing_Outcome from SPACEXTABLE where substr(Date,0,5)='2015' and Landing_Outcome='Failure (drone ship)'
```

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

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

- Here we use a function **SUBSTR** on the column Date in order to select only year from the date and we select only the records where year is 2015 and outcome is failure in drone ship by using assignment operator =

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

- Ranking the count of landing outcomes (such as Failure (drone ship) or Success

```
%sql select count(Landing_outcome),Landing_outcome from SPACEXTABLE WHERE DATE BETWEEN '2010-06-04' and '2017-03-20' group by Landing_outcome order by count(Landing_outcome) desc
```

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

count(Landing_outcome)	Landing_Outcome
10	No attempt
5	Success (drone ship)
5	Failure (drone ship)
3	Success (ground pad)
3	Controlled (ocean)
2	Uncontrolled (ocean)
2	Failure (parachute)
1	Precluded (drone ship)

- We use **GROUP BY** to classify groups available in the column landing_outcome and use **COUNT** function to count the number of occurances of the respective group and **DESC** keyword for arranging the result in descreasing order.

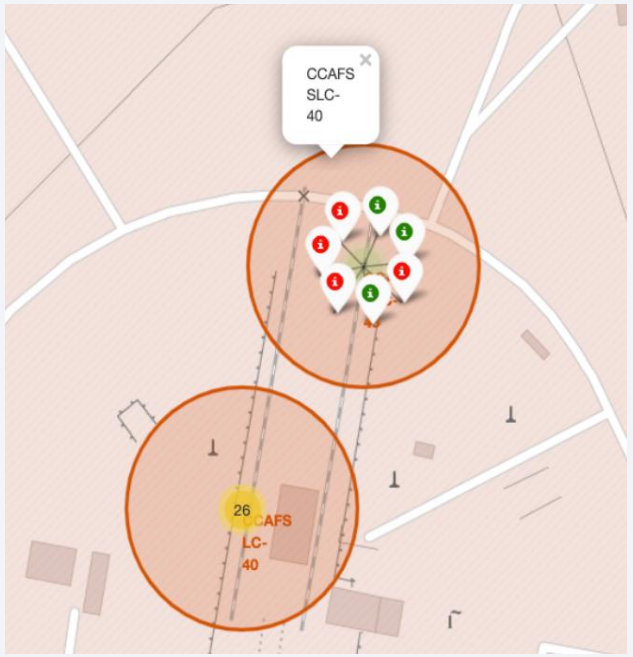
Section 3

Launch Sites Proximities Analysis

Marking all launch sites on a map

- Created and added `folium.Circle` and `folium.Marker` for each launch site on the site map by specifying necessary attributes and their respective values.

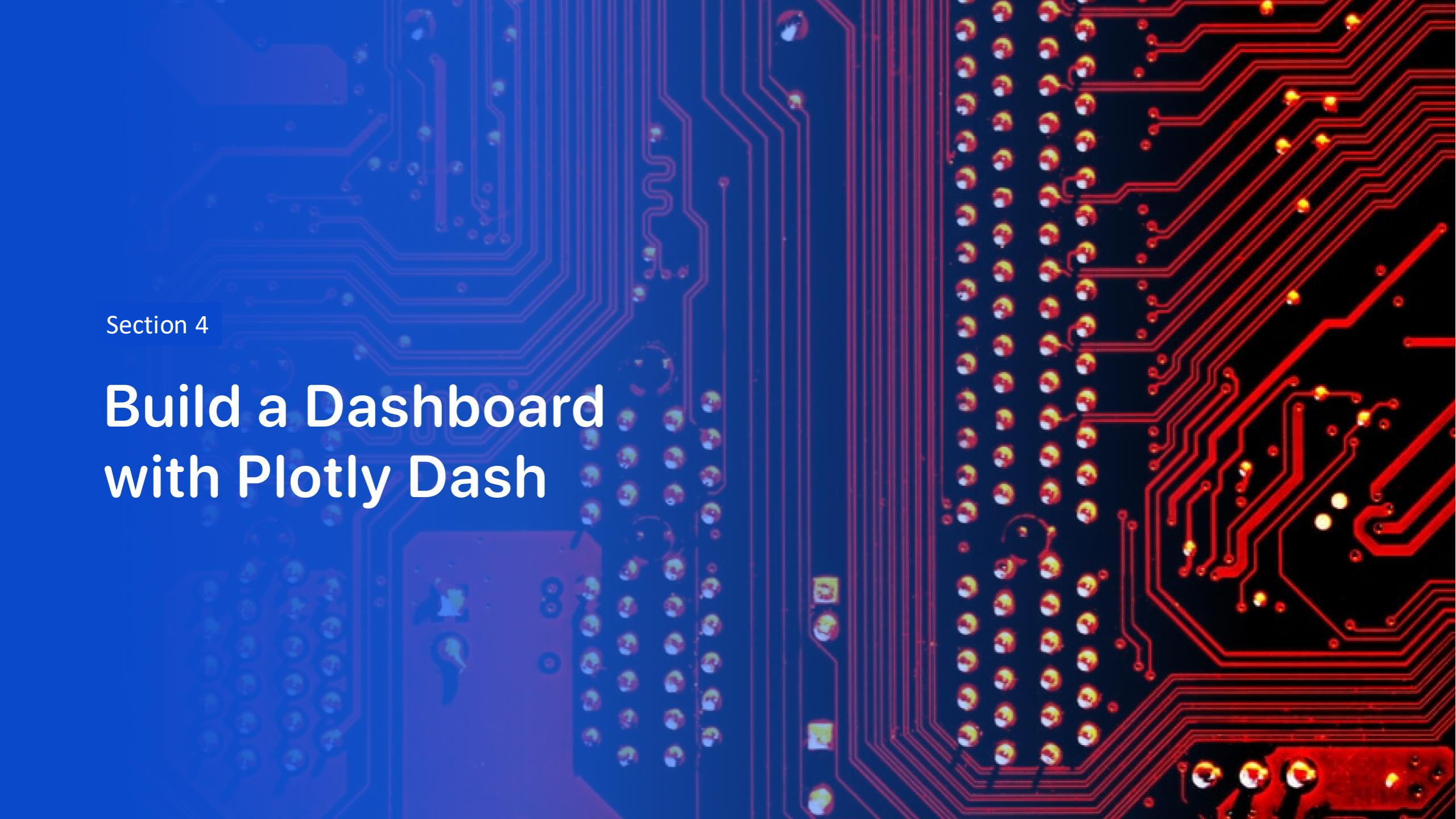
Marking the success/failed launches for each site on the map



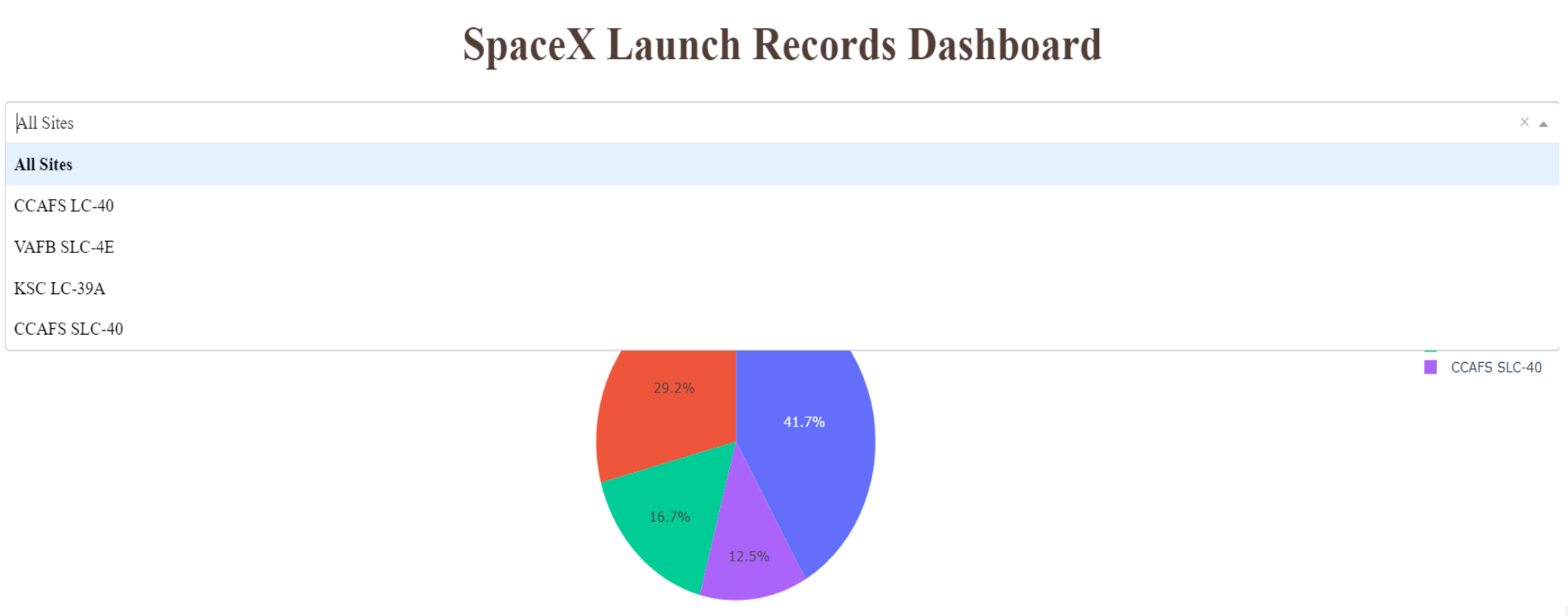
- We create markers for all launch records. If a launch was successful (class=1), then we use a green marker and if a launch was failed, we use a red marker (class=0).
- For each launch result in spacex_df data frame, we add a folium.Marker to marker_cluster.

Calculating the distances between a launch site to its proximities

- Marked down a point on the closest coastline using MousePosition and calculated the distance between the coastline point and the launch site.
- Drew a line between a launch site to its closest city, railway, highway, etc. You need to use `MousePosition` to find their coordinates on the map first.

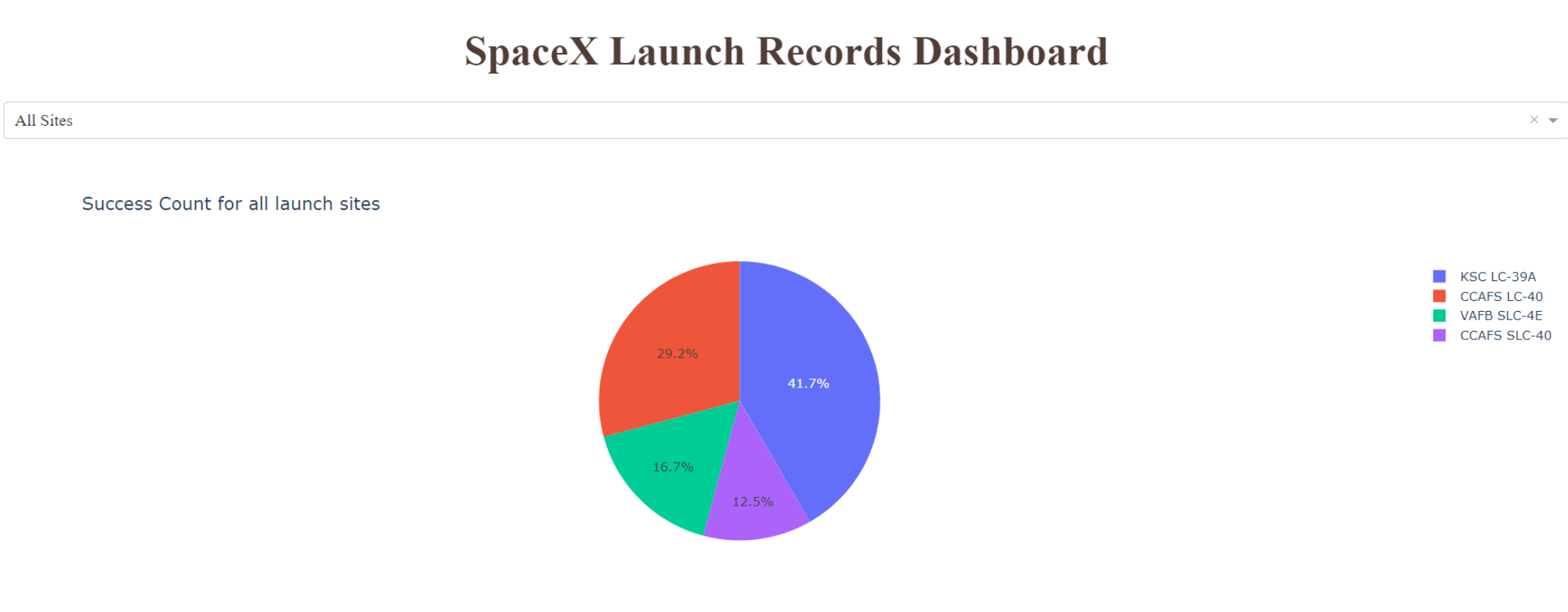

Section 4
Build a Dashboard with Plotly Dash

Adding a dropdown to enable Launch Site selection



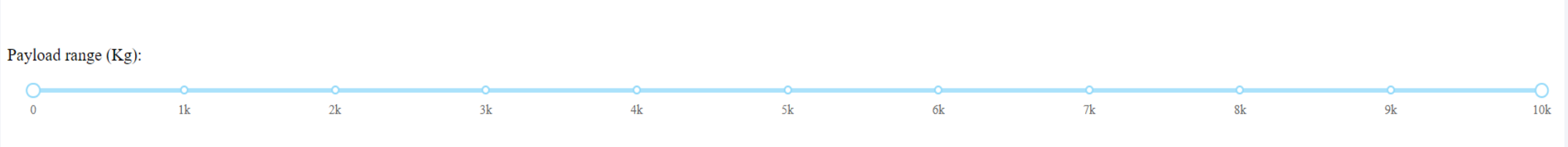
We use **Dropdown** function present in **dash_core_components** by specificifying different labels available in options attribute. This pie chart clearly displays Success count of various available launch sites.

Adding a Pie chart to show the successful launches for different launch sites



We use **Graph** function present in **dash_core_components** by specifying different labels available in options attribute. This pie chart clearly displays Success count of all available launch sites.

Adding a Range Slider to select Payload range



We use **RangeSlider** function in **dash_core_components** by setting min and max values with a step value . This RangeSlider helps us in ranging between minimum and maximum values for Payload attribute.

Adding a scatter chart to show correlation between Payload and Launch success

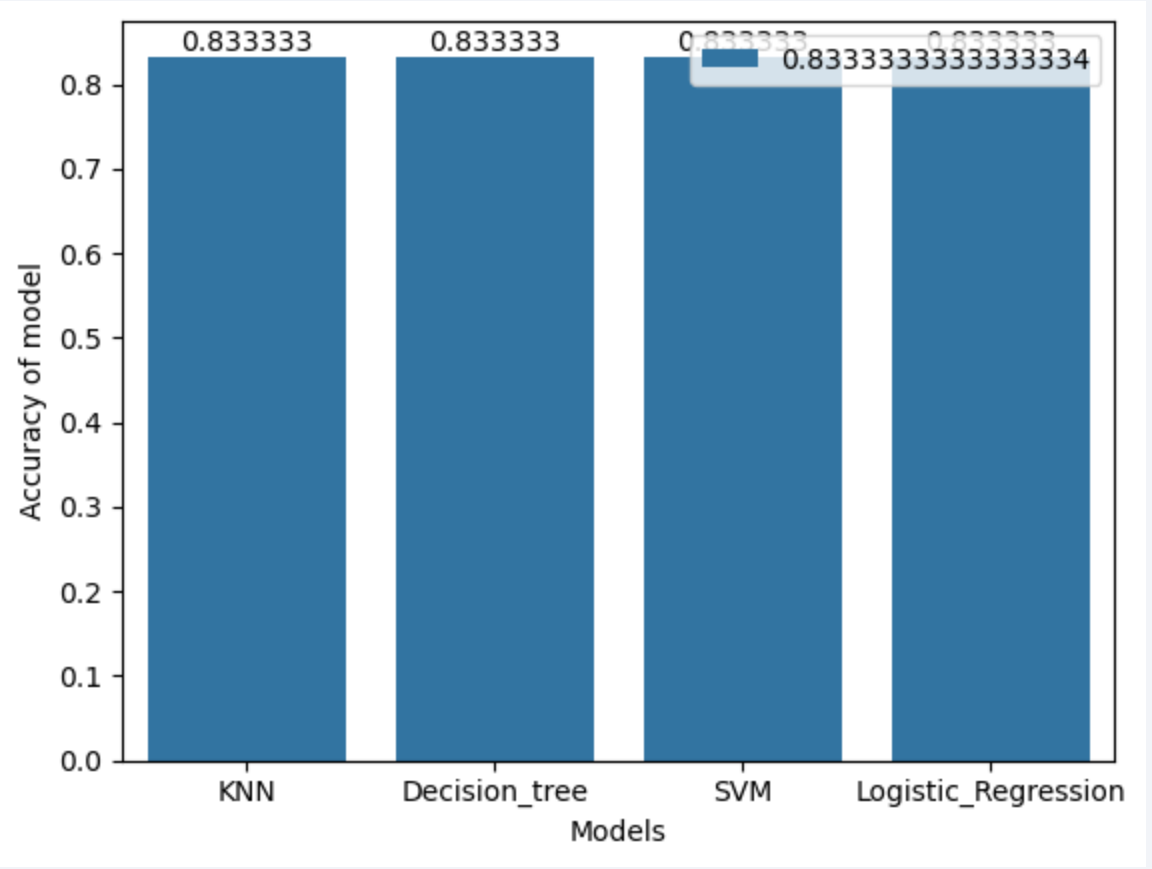
We use **Graph** function in **dash_core_components** where id is set to success-payload-scatter-chart.

Section 5

Predictive Analysis (Classification)

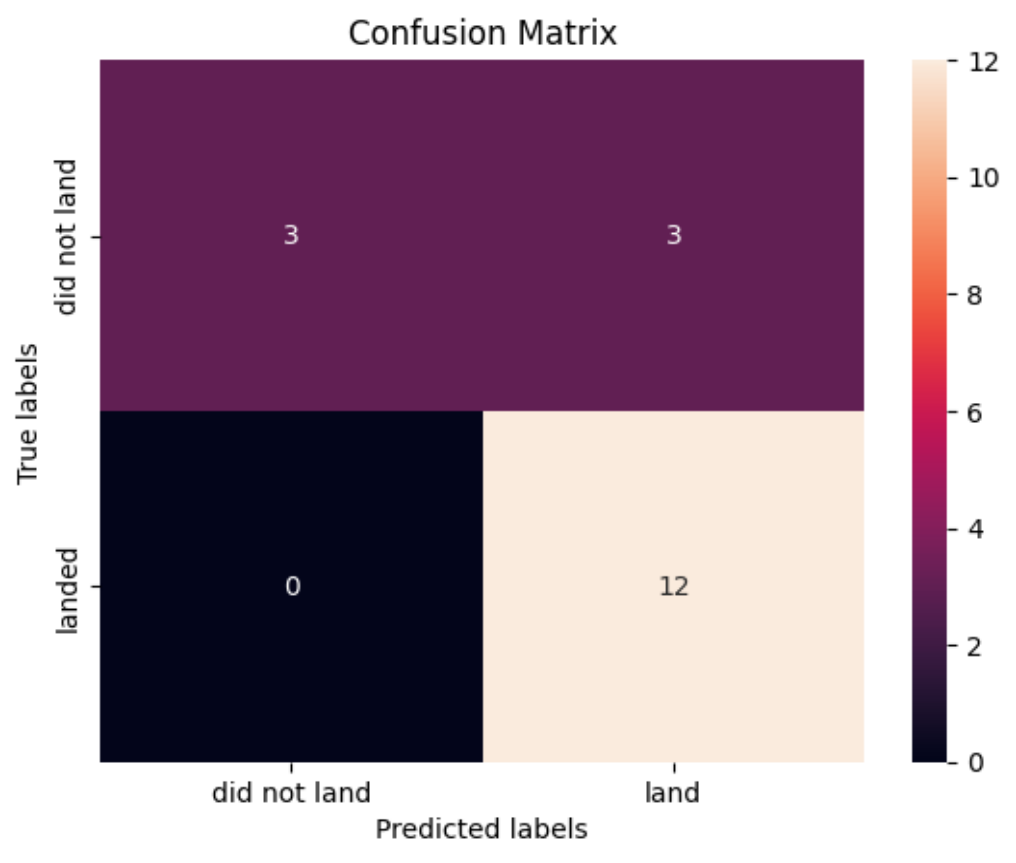
Classification Accuracy

- All the models have the same test accuracy when using their best parameter values computed by Grid Search technique i.e, 0.833333333334. Therefore any model from the four can be best picked.



Confusion Matrix

- Confusion matrix of the best performing model



This Confusion matrix of decision tree states that :

- The model classified 15 data samples correctly and only 3 were classified incorrect, which made all the four models best suited for prediction of first stage of SPACEX.

Conclusions

- Different launch sites have different success rates :
  - **Launch Site** **Success Rate**
  - CCAFS LC - 40 29.2%
  - KSC LC – 39A 41.7%
  - VAFB SLC – 4E 16.7%
  - CCAFS SLC - 40 12.5%
- We can deduce that, as the flight number increases in different launch sites.
- Orbits with high success rate are ES-L1,GEO,HEO,SSO and the least success rate is for GTO and no success rate for SO type orbit
- 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.

Conclusions(cont'd...)

- 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.
- You can observe that the success rate since 2013 kept increasing till 2020.
- All the models have the same test accuracy when using their best parameter values computed by Grid Search technique i.e, 0.833333333334.Therefore any model from the four can be best picked.

Thank you!