IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rutvij Bhatt

May 16th, 2023

Outline

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

Executive Summary

Summary of methodologies on SpaceX data

- Data Collection using SpaceX API and Web Scraping
- Data Wrangling
- Exploratory Data Analysis (EDA) with SQL
- EDA with Data Visualization
- Interactive Visual Analysis and Dashboard using Folium and Plotly Dash
- Training and Prediction

Summary of all results

- Data Exploration Results
- Data Visualization Results
- Prediction (Classification) Result

Introduction

Project background and context

- SpaceX promotes its reusability of the first stage of a Falcon9 rocket launch, results in significant reduction in the cost for a launch as compared to other competitors in the market.
- Thus, we need to determine the possibility of the first stage to be landed successfully, therefore the total cost of the rocket launch can be estimated. It helps for the competitors to bid against.

Problems you want to find answers

- To predict the possibility of the first stage of a Falcon9 rocket to be landed successfully based on the launch data published on the SpaceX site.

Section 1

Methodology

Methodology

- Executive Summary
- Data collection methodology
  - Description of how data was collected
- Perform data wrangling
  - Description of 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 launch data sets were collected:

- Launch records were collected using two sources:

1. SpaceX REST API
2. Using Web Scraping from the Wiki.

    - **SpaceX REST API**: A request to the Falcon9 launch data was made using GET, and a response was collected and parsed in the form of a JSON object, later to be converted in the Pandas dataframe.

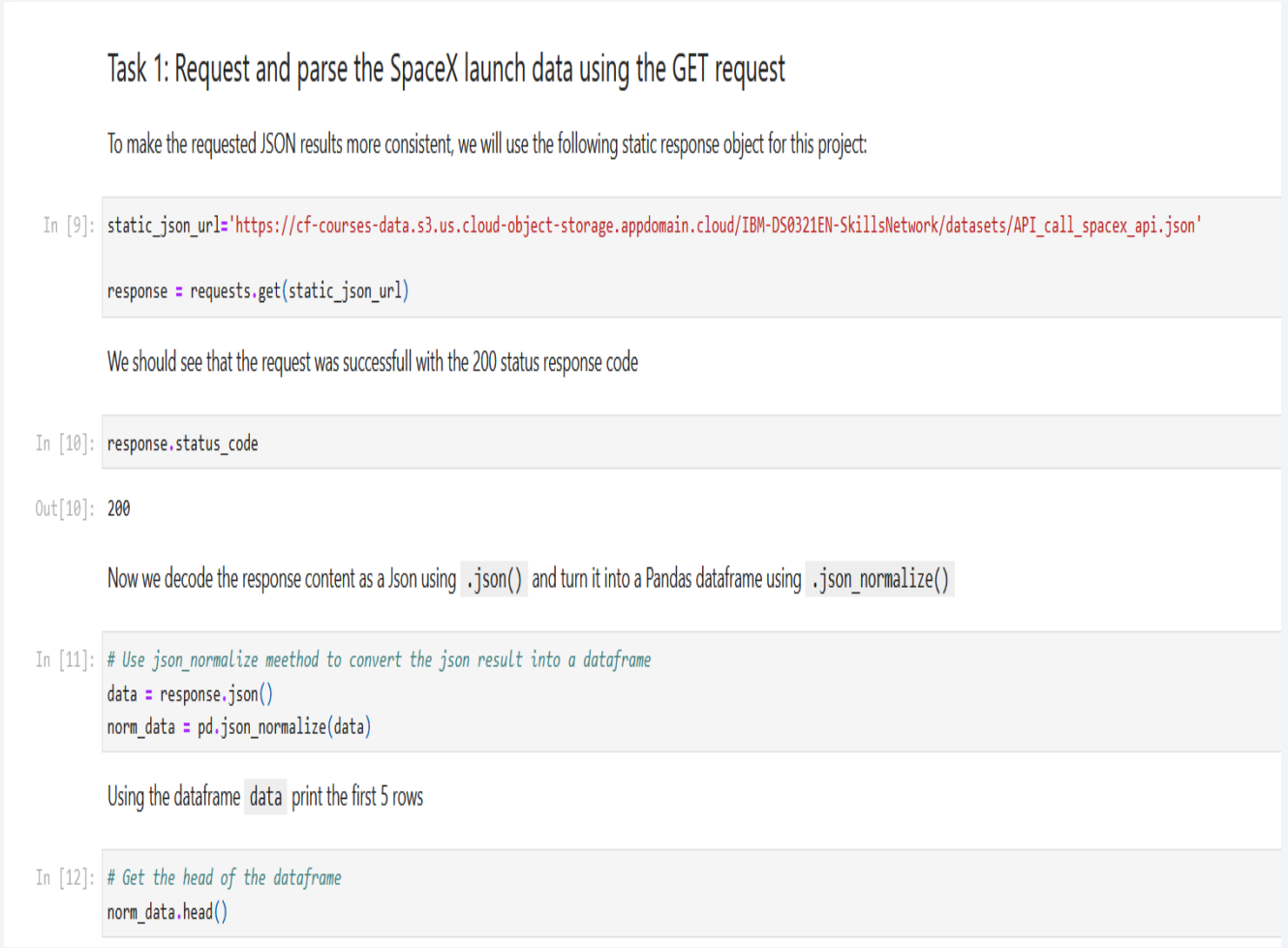
    - **From the Wikipedia page using Web Scraping**: With the help of the BeautifulSoup library, a HTML table comprised of the launch records was extracted from the page 'List of Falcon 9 and Falcon Heavy launches', parsed and converted into the Pandas dataframe.

Data Collection – SpaceX API

Data collection using REST API to the SpaceX Falcon9 launch data:

- A request was made, and a response was received using GET and was parsed into the JSON object later to be converted into a Pandas dataframe.

- GitHub URL of the completed SpaceX API calls notebook.



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

response = requests.get(static_json_url)
```

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

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

Out[10]: 200

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

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

Using the dataframe `data` print the first 5 rows

In [12]:
```
# Get the head of the dataframe
norm_data.head()
```

Data Collection - Scraping

Web Scraping to collect historical data of Falcon9 Launches:

- With the help of the BeautifulSoup library, a HTML table comprised of the launch records was extracted from the page 'List of Falcon 9 and Falcon Heavy launches', parsed and converted into the Pandas dataframe.

- GitHub URL of the completed web scraping notebook.

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

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 [5]:
```
# 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`

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

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

In [7]:
```
# Use soup.title attribute
print(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`, please check the external reference link towards the end of this lab

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

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

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

Data Wrangling

- As the data was collected in a Pandas dataframe, next the data cleaning was done. As the data has all the launches, we had to filter only data related to Falcon9, it was done using ***BoosterVersion*** column.
- This newly filtered dataframe has all the records of the Falcon9 launches, however, some information was missing in the attribute ***LandingPad.***
- Depended variable such as landing outcome was determined, and associate column was added named ***Class*** consist of all the labels for training the model.

- GitHub [URL](URL) of the completed data wrangling related notebook.

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 [12]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for idx, row in df.iterrows():
    if row['Outcome'] in bad_outcomes:
        # print('true')
        landing_class.append(0)
    else:
        landing_class.append(1)
print(landing_class)
```

```
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

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

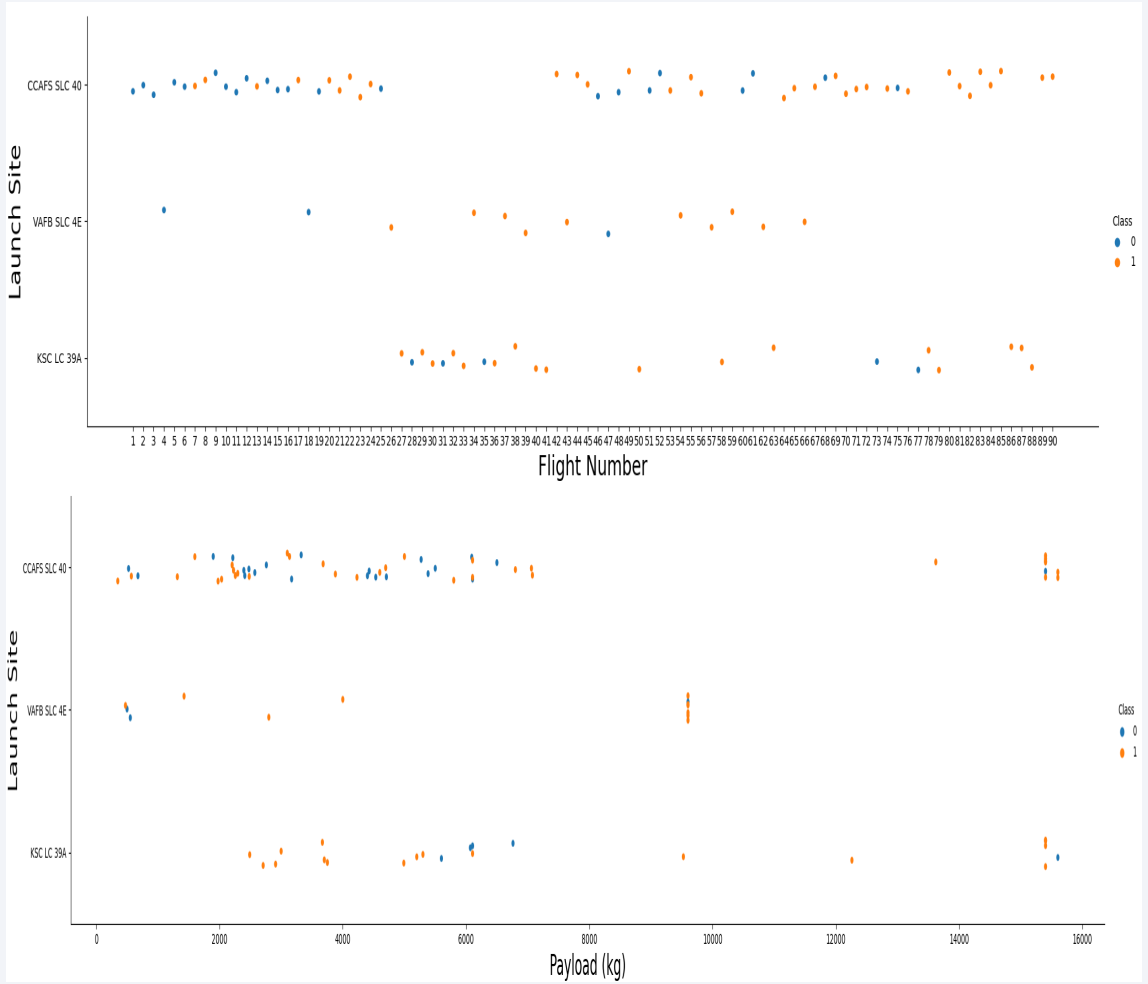
In [13]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

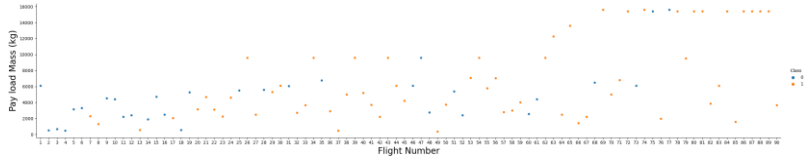
Out[13]:

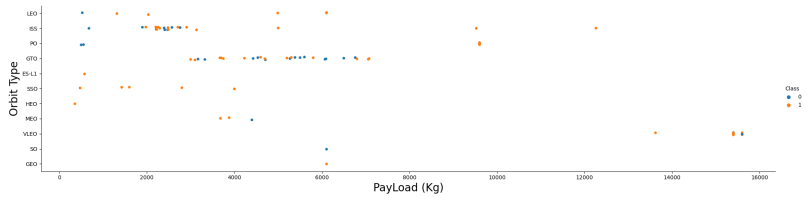
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

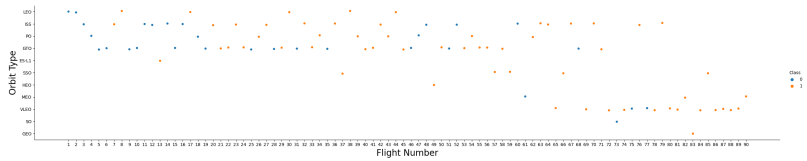
EDA with Data Visualization

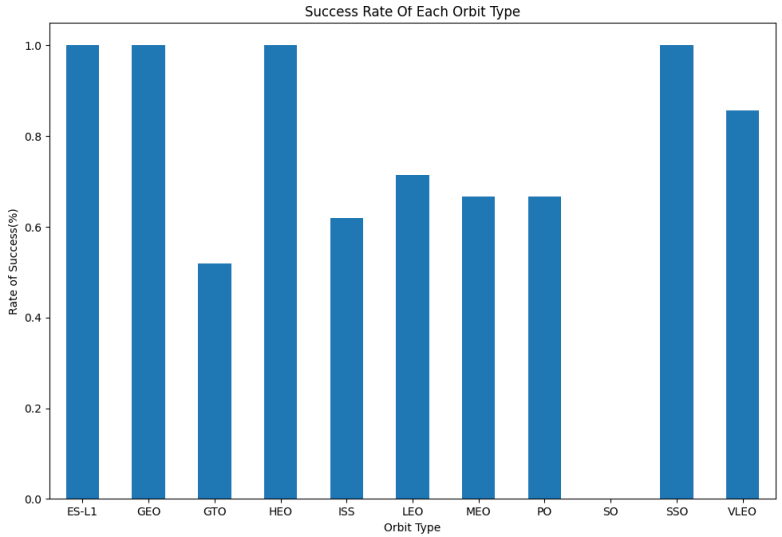
- Data Visualization is done using Matplotlib.
- EDA is performed and insights are displayed using various plots.
  - Scatter Plots:
    - Relationship between Launch Site and Flight Number
    - Relationship between Payload and Launch Site
    - Relationship between Flight Number and Orbit Type
    - Relationship between Payload and Orbit Type
  - Bar Chart:
    - To illustrate the success rate of each orbit type
  - Line Plot:
    - Depicts successful launches per year
- GitHub URL of the completed EDA with data visualization notebook.

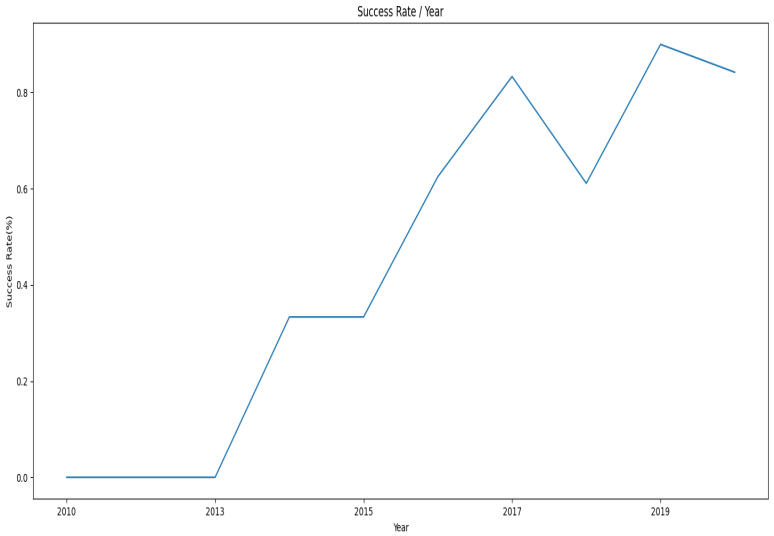












EDA with Data Visualization (Cont.)

EDA with SQL

- Queries performed for Exploratory Data Analysis are as follows:
  - **Display the names of the unique launch sites in the space mission**

```
# cur.execute("SELECT DISTINCT Launch_Site from SPACEXTBL").fetchall()
%sql SELECT DISTINCT Launch_Site from SPACEXTBL
```

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

```
# cur.execute("SELECT * from SPACEXTBL where Launch_Site LIKE 'CCA%' LIMIT 5").fetchall()
%sql SELECT * from SPACEXTBL where Launch_Site LIKE 'CCA%' LIMIT 5
```

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

```
# cur.execute("SELECT SUM(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)'").fetchall()
%sql SELECT SUM(PAYLOAD_MASS__KG_) as Total_Payload_Mass from SPACEXTBL where Customer = 'NASA (CRS)'
```

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

```
# cur.execute("SELECT AVG(PAYLOAD_MASS__KG_) as AVG_Payload_Mass from SPACEXTBL where Booster_Version LIKE 'F9 v1.1%'").fetchall()
%sql SELECT AVG(PAYLOAD_MASS__KG_) as AVG_Payload_Mass from SPACEXTBL where Booster_Version LIKE 'F9 v1.1%'
```

EDA with SQL (Cont.)

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

```
# cur.execute('SELECT MIN(Date) as First_Landing from SPACEXTBL where "Landing _Outcome" = "Success (ground pad)"').fetchall()
%sql SELECT MIN(Date) as First_Landing 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

```
# cur.execute('SELECT "Booster_Version" as Booster_Names from SPACEXTBL where "Landing _Outcome" = "Success (drone ship)" AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000').fetchall()
%sql SELECT "Booster_Version" as Booster_Names from SPACEXTBL where "Landing _Outcome" = "Success (drone ship)" AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000
```

- List the total number of successful and failure mission outcomes

```
# cur.execute('SELECT COUNT("Mission_Outcome") as Count, Mission_Outcome from SPACEXTBL GROUP BY "Mission_Outcome" ').fetchall()
%sql SELECT COUNT("Mission_Outcome") as Count, Mission_Outcome from SPACEXTBL GROUP BY "Mission_Outcome"
```

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

```
# cur.execute('SELECT "Booster_Version" as Booster_Versions from SPACEXTBL where "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") from SPACEXTBL)').fetchall()
%sql SELECT "Booster_Version" as Booster_Versions, "PAYLOAD_MASS__KG_" from SPACEXTBL where "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") from SPACEXTBL)
```

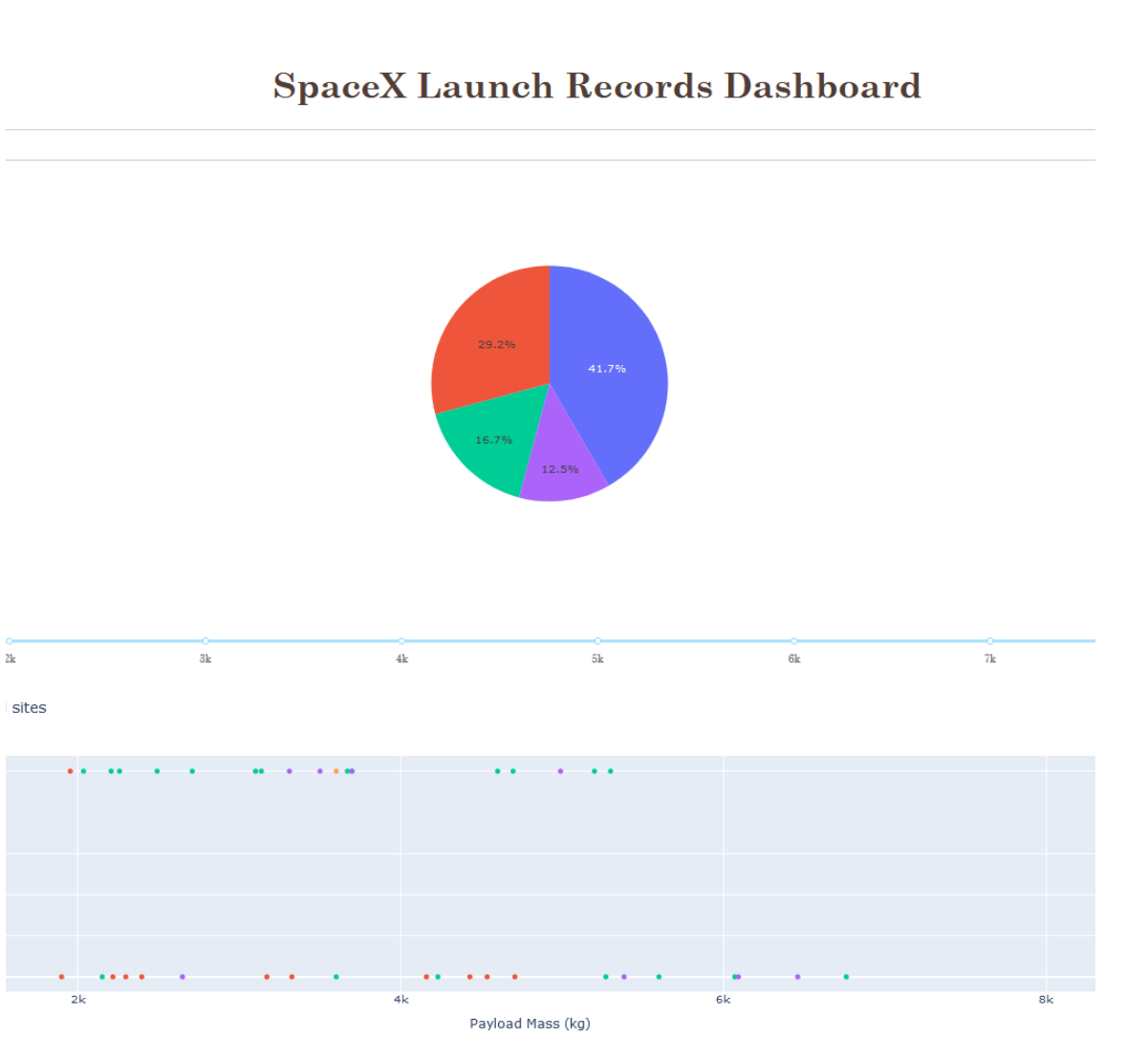
- GitHub URL of the completed EDA with SQL notebook.

Build an Interactive Map with Folium

- Folium map objects such as markers, circles and lines are used to mark all the launch sites and success and failures of each launch performed through such sites.
- Also, it is used to create outcome sets such as (failure = 0 or success = 1).
- GitHub URL of completed interactive map with Folium map.

Build a Dashboard with Plotly Dash

- An interactive dashboard application is created using Plotly dash and following modules were added:
  - A drop-down component for launch site selection.
  - A call back function to display a pie-chart of success for launches from the selected launch site from the drop-down.
  - A range slider to select Payload between certain range.
  - A call back function to display the scatter-plot for launches based on the mentioned attributes such as launch sites, and payload range.
- GitHub URL of Plotly Dash lab.



Predictive Analysis (Classification)

- Summary of the predictive analysis is as follows:
  - Performed Exploratory Data Analysis and determined Training Labels
  - Created a column for the ''class''.
  - Standardized the input data
  - Split into training data and test data
  - Found the best Hyperparameter for SVM, Classification Trees , K nearest neighbors and Logistic Regression
  - Found the method that performed best using test data
- The complete flow is as follows:
- First, a pandas dataframe was created and labels for the training data were determined by creating a Numpy array from the "Class" column of the data using to_numpy() method and it was assigned to a variable Y.
- Then using preprocessing.StandardScaler() function from Sklearn, dataset was standardized.
- Dataset was split into train-test parts using train_test_split from the sklearn.model_selection and test size is set to 0.2 with random_state equal to 2.
- Finally, to find best suited model among SVM, Classification Tree, K Nearest Neighbors, and Logistic Regression, object of each of them was created along with GridSearchCV object.

Predictive Analysis (Classification)(Cont.)

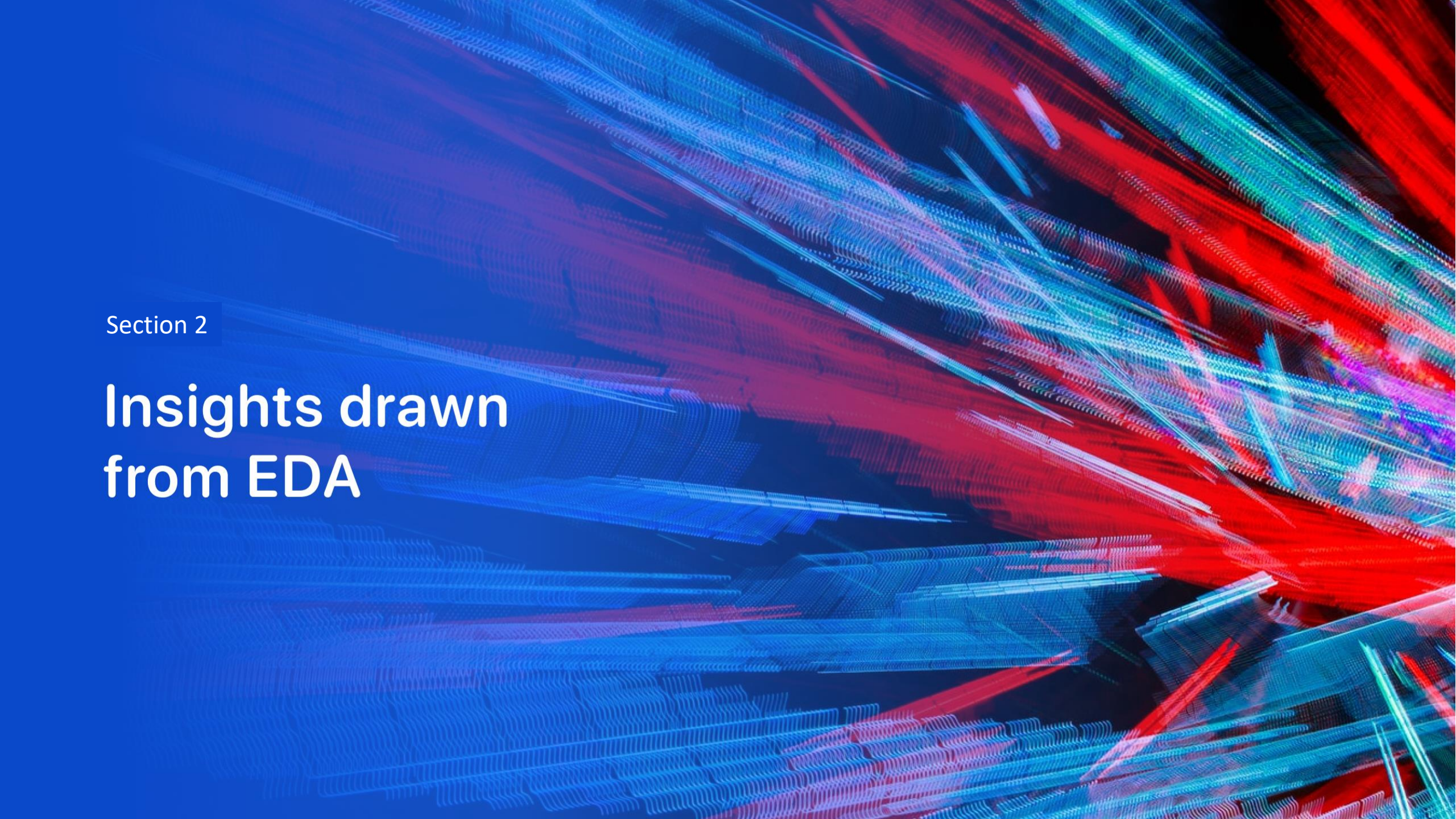
- GridSearchCV object was created and parameter cv was set to 10. Training data was fitted to this object for each of the algorithms under evaluation to obtain best hyperparameters for each of them.
- Result was the GridSearchCV object for each of the models and best parameters was displayed using best_params_ followed by respective accuracy on training set using best_score_.
- At the end, accuracy on the test data was obtained using score method and confusion matrix was created to show actual and predicted results on test data.
- Following is the accuracy score of each model on testing data:

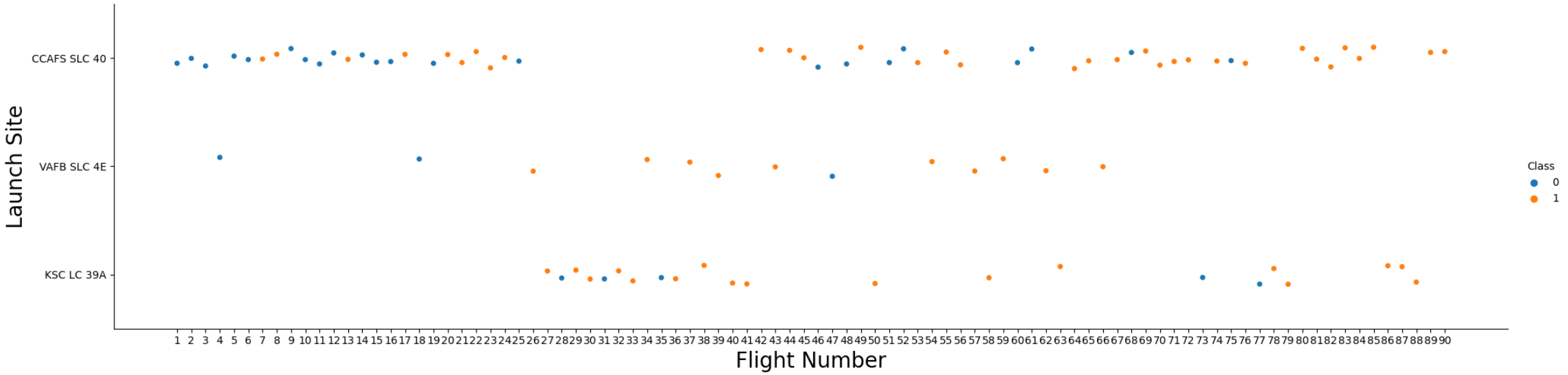
Model	Accuracy on Test data
Logistic Regression	0.833333
SVM	0.833333
Decision Tree Classifier	0.833333
KNN	0.833333

- GitHub URL of completed predictive analysis lab.

Results

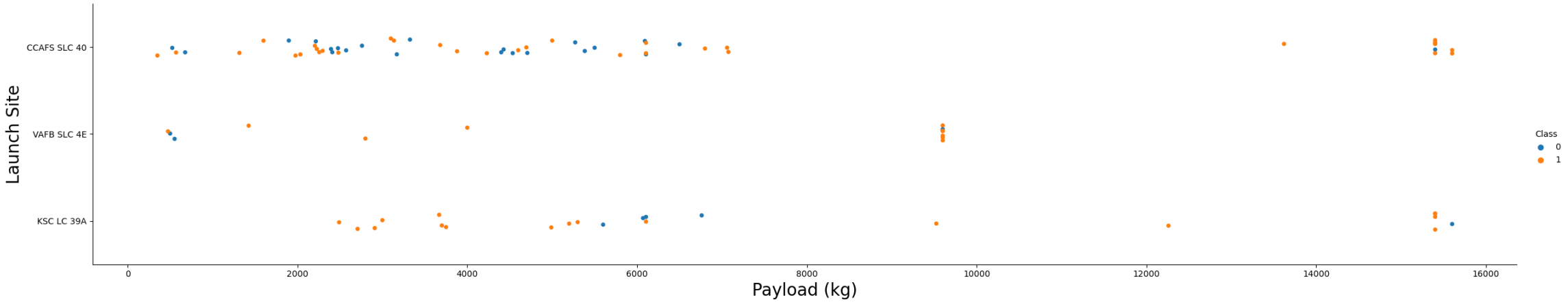
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA



Flight Number vs. Launch Site

- With increasing flight number, failure rate decreases significantly. For launch sites KSC LC 39A and CCFS SLC 40, success rate becomes 100% after flight number 77.
- While, in launch site VAFB SLC 4E, success rate is 100% after flight number 50.

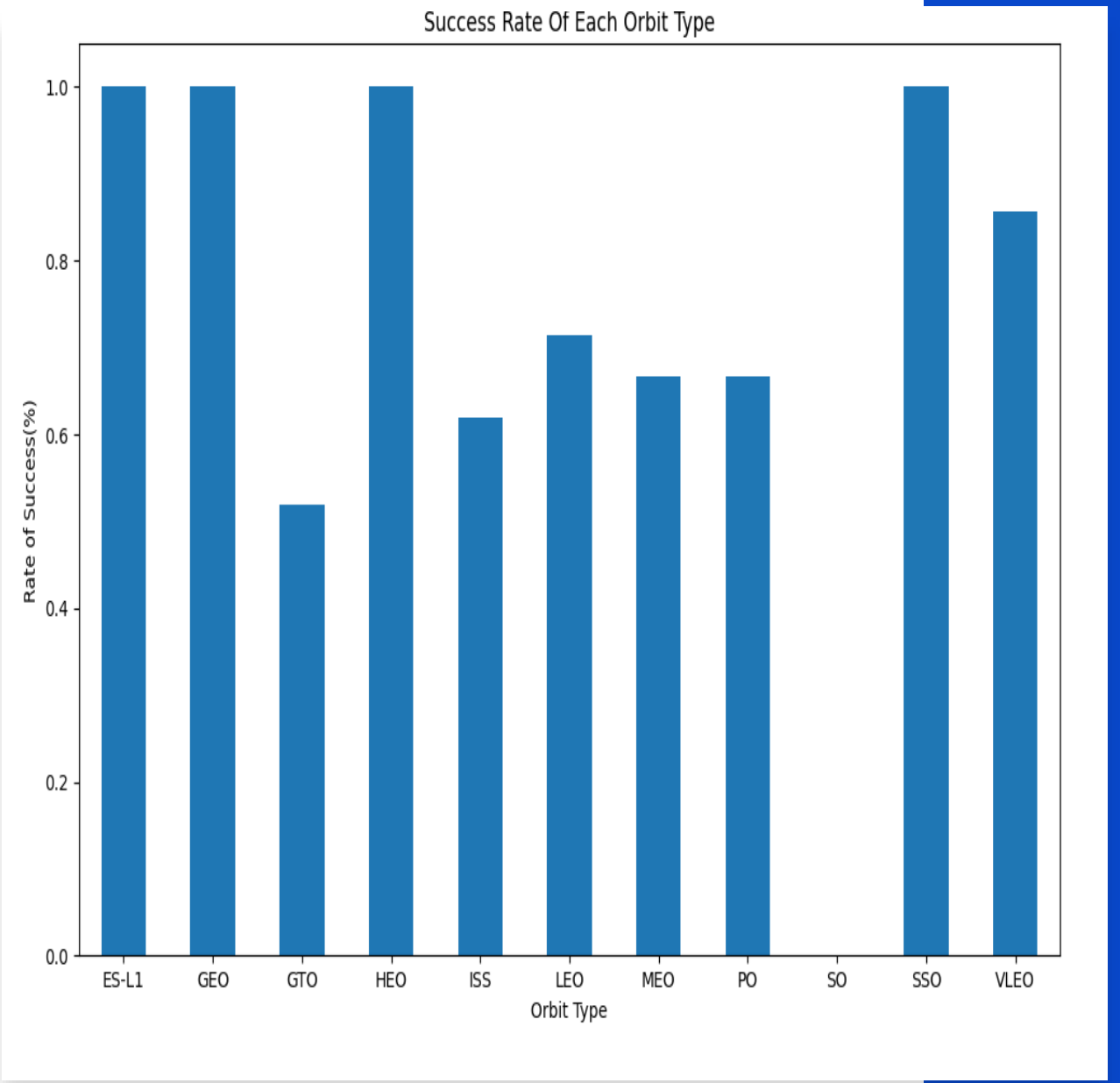


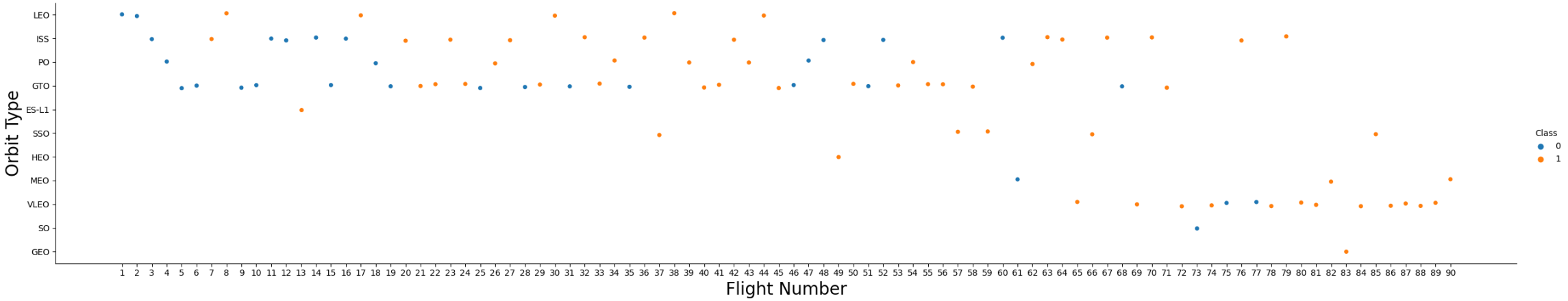
Payload vs. Launch Site

- If you 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

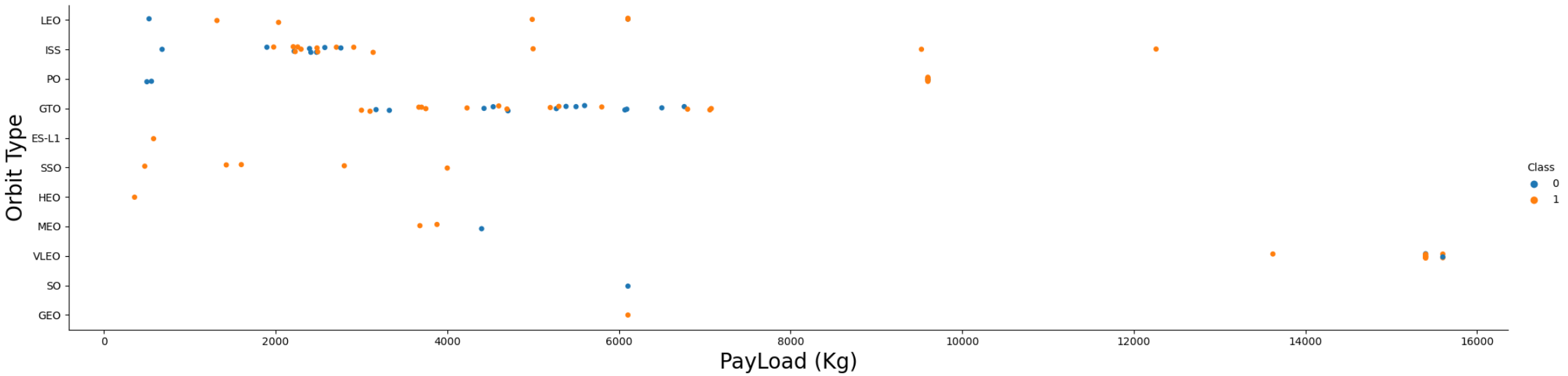
- Orbits ES-L1, GEO, HEO, and SSO have success rate of 100%.
- Orbits GTO, ISS, LEO, MEO, and PO have success rate greater than 50%.
- While SO orbit has the lowest success rate of 0%.





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.

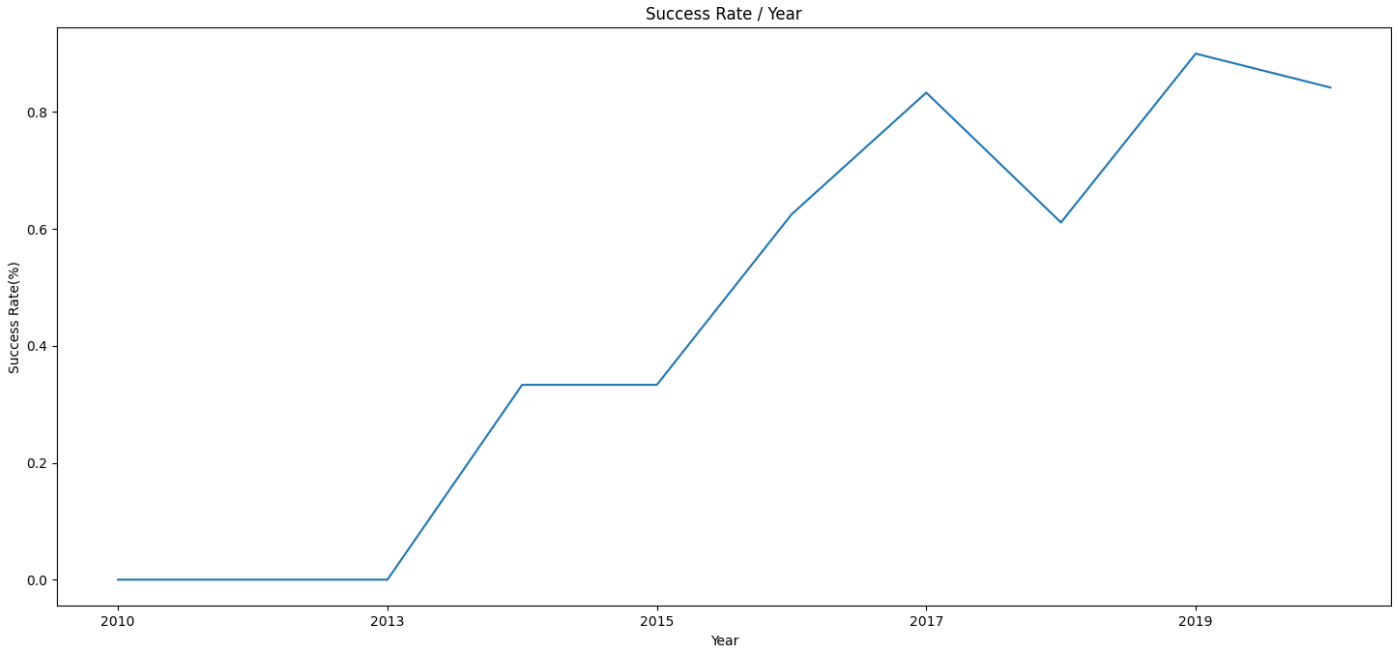


Payload vs. Orbit Type

- 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 success rate since 2013 kept increasing till 2020.



All Launch Site Names

- Find the names of the unique launch sites

```
# cur.execute("SELECT DISTINCT Launch_Site from SPACEXTBL").fetchall()
%sql SELECT DISTINCT Launch_Site from SPACEXTBL
```

- Used DISTINCT to select unique launch sites from the SPACEXTBL.

Task 1

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

```
# cur.execute("SELECT DISTINCT Launch_Site from SPACEXTBL").fetchall()
%sql SELECT DISTINCT Launch_Site from SPACEXTBL
```

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

Launch_Site
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`

```
# cur.execute("SELECT * from SPACEXTBL where Launch_Site LIKE 'CCA%' LIMIT 5").fetchall()
%sql SELECT * from SPACEXTBL where Launch_Site LIKE 'CCA%' LIMIT 5
```

- Used LIKE in where clause with '%' to find 5 records starts with 'CCA' using LIMIT.

Task 2

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

```
# cur.execute("SELECT * from SPACEXTBL where Launch_Site LIKE 'CCA%' LIMIT 5").fetchall()
%sql SELECT * from SPACEXTBL 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
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

```
# cur.execute("SELECT SUM(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)'").fetchall()
%sql SELECT SUM(PAYLOAD_MASS__KG_) as Total_Payload_Mass from SPACEXTBL where Customer = 'NASA (CRS)'
```

- Used SUM to calculate total payload mass for the customer NASA (CRS) from SPACEXTBL

Task 3

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

```
# cur.execute("SELECT SUM(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)'").fetchall()
%sql SELECT SUM(PAYLOAD_MASS__KG_) as Total_Payload_Mass from SPACEXTBL where Customer = 'NASA (CRS)'
```

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

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

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

```
# cur.execute("SELECT AVG(PAYLOAD_MASS__KG_) as AVG_Payload_Mass from SPACEXTBL where Booster_Version LIKE 'F9 v1.1%'").fetchall()
%sql SELECT AVG(PAYLOAD_MASS__KG_) as AVG_Payload_Mass from SPACEXTBL where Booster_Version LIKE 'F9 v1.1%'
```

- Used AVG to find average payload mass for booster version F9 v1.1 from SPACEXTBL

Task 4

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

```
# cur.execute("SELECT AVG(PAYLOAD_MASS__KG_) as AVG_Payload_Mass from SPACEXTBL where Booster_Version LIKE 'F9 v1.1%'").fetchall()
%sql SELECT AVG(PAYLOAD_MASS__KG_) as AVG_Payload_Mass from SPACEXTBL where Booster_Version LIKE 'F9 v1.1%'
```

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

AVG_Payload_Mass
2534.6666666666665

First Successful Ground Landing Date

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

```
# cur.execute('SELECT MIN(Date) as First_Landing from SPACEXTBL where "Landing _Outcome" = "Success (ground pad)"').fetchall()
%sql SELECT MIN(Date) as First_Landing from SPACEXTBL where "Landing _Outcome" = "Success (ground pad)"
```

- Used MIN to find first date on which landing outcome on the ground pad was a success.

Task 5

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

Hint:Use min function

```
# cur.execute('SELECT MIN(Date) as First_Landing from SPACEXTBL where "Landing _Outcome" = "Success (ground pad)"').fetchall()
%sql SELECT MIN(Date) as First_Landing from SPACEXTBL where "Landing _Outcome" = "Success (ground pad)"
```

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

First_Landing
01-05-2017

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

```
# cur.execute('SELECT "Booster_Version" as Booster_Names from SPACEXTBL where "Landing _Outcome" = "Success (drone ship)" AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000').fetchall()
%sql SELECT "Booster_Version" as Booster_Names from SPACEXTBL where "Landing _Outcome" = "Success (drone ship)" AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000
```

- Used AND and BETWEEN operators to find boosters with payload in between 4000 and 6000, and it was landed successfully on a drone ship.

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

```
# cur.execute('SELECT "Booster_Version" as Booster_Names from SPACEXTBL where "Landing _Outcome" = "Success (drone ship)" AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000').fetchall()
%sql SELECT "Booster_Version" as Booster_Names from SPACEXTBL where "Landing _Outcome" = "Success (drone ship)" AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000
```

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

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
# cur.execute('SELECT COUNT("Mission_Outcome") as Count, Mission_Outcome from SPACEXTBL GROUP BY "Mission_Outcome" ').fetchall()
%sql SELECT COUNT("Mission_Outcome") as Count, Mission_Outcome from SPACEXTBL GROUP BY "Mission_Outcome"
```

- Used COUNT and GROUP BY to find each output class and respective number of results from the table SPACEXTBL

Task 7

List the total number of successful and failure mission outcomes

```
# cur.execute('SELECT COUNT("Mission_Outcome") as Count, Mission_Outcome from SPACEXTBL GROUP BY "Mission_Outcome" ').fetchall()
%sql SELECT COUNT("Mission_Outcome") as Count, Mission_Outcome from SPACEXTBL GROUP BY "Mission_Outcome"
```

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

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

Boosters Carried Maximum Payload

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

```
# cur.execute('SELECT "Booster_Version" as Booster_Versions from SPACEXTBL where "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") from SPACEXTBL)').fetchall()
%sql SELECT "Booster_Version" as Booster_Versions, "PAYLOAD_MASS__KG_" from SPACEXTBL where "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") from SPACEXTBL)
```

- Used subquery and MAX operator to obtain the booster versions which have maximum payload mass from the table SPACEXTBL.

Task 8

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

```
# cur.execute('SELECT "Booster_Version" as Booster_Versions from SPACEXTBL where "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") from SPACEXTBL)').fetchall()
%sql SELECT "Booster_Version" as Booster_Versions, "PAYLOAD_MASS__KG_" from SPACEXTBL where "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") from SPACEXTBL)
```

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

Booster_Versions	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

2015 Launch Records

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

- `# cur.execute('SELECT substr(Date, 4, 2) as Month, "Landing _Outcome", "Booster_Version", "Launch_Site" from SPACEXTBL where substr(Date,7,4)="2015" AND "Landing _Outcome" LIKE "Failure (drone ship)"').fetchall()`
- `%sql SELECT substr(Date, 4, 2) as Month, "Landing _Outcome", "Booster_Version", "Launch_Site", substr(Date, 7, 4) as Year from SPACEXTBL where substr(Date,7,4)="2015" AND "Landing _Outcome" LIKE "Failure (drone ship)"`

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.

```
# cur.execute('SELECT substr(Date, 4, 2) as Month, "Landing _Outcome", "Booster_Version", "Launch_Site" from SPACEXTBL where substr(Date,7,4)="2015" AND "Landing _Outcome" LIKE "Failure (drone s
%sql SELECT substr(Date, 4, 2) as Month, "Landing _Outcome", "Booster_Version", "Launch_Site", substr(Date, 7, 4) as Year from SPACEXTBL where substr(Date,7,4)="2015" AND "Landing _Outcome" LIKE
```

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

Month	Landing _Outcome	Booster_Version	Launch_Site	Year
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015

- Used substr for dates, AND and LIKE operators for landing outcome type to be failure on drone ship.

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

- Rank the count of successful landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order
- `%sql SELECT "Landing _Outcome", count(*) AS COUNT_ FROM SPACEXTBL WHERE Date BETWEEN '04-06-2010' AND '20-03-2017' AND "Landing _Outcome" LIKE "%Success%" GROUP BY "Landing _Outcome" ORDER BY COUNT DESC`

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql SELECT "Landing _Outcome", count(*) AS COUNT_ FROM SPACEXTBL WHERE Date BETWEEN '04-06-2010' AND '20-03-2017' AND "Landing _Outcome" LIKE "%Success%" GROUP BY "Landing _Outcome" ORDER BY CO
```

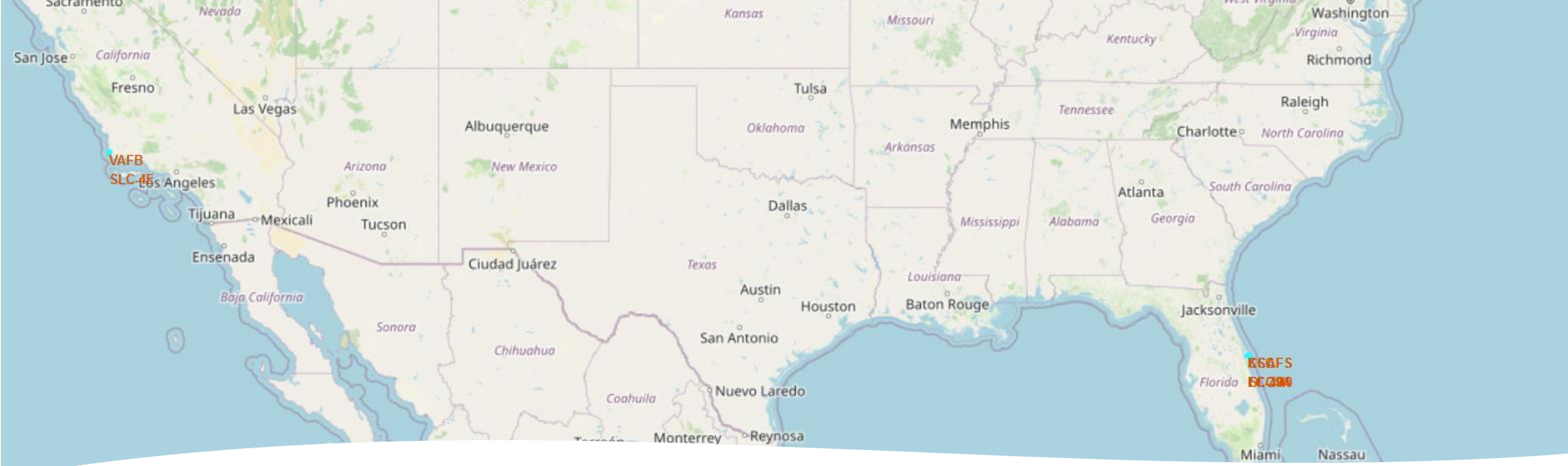
```
 * sqlite:///my_data1.db
Done.
```

Landing _Outcome	COUNT_
Success	20
Success (drone ship)	8
Success (ground pad)	6

- Used GROUP BY and ORDER BY to obtain a descending order list of landing outcome with respective count.

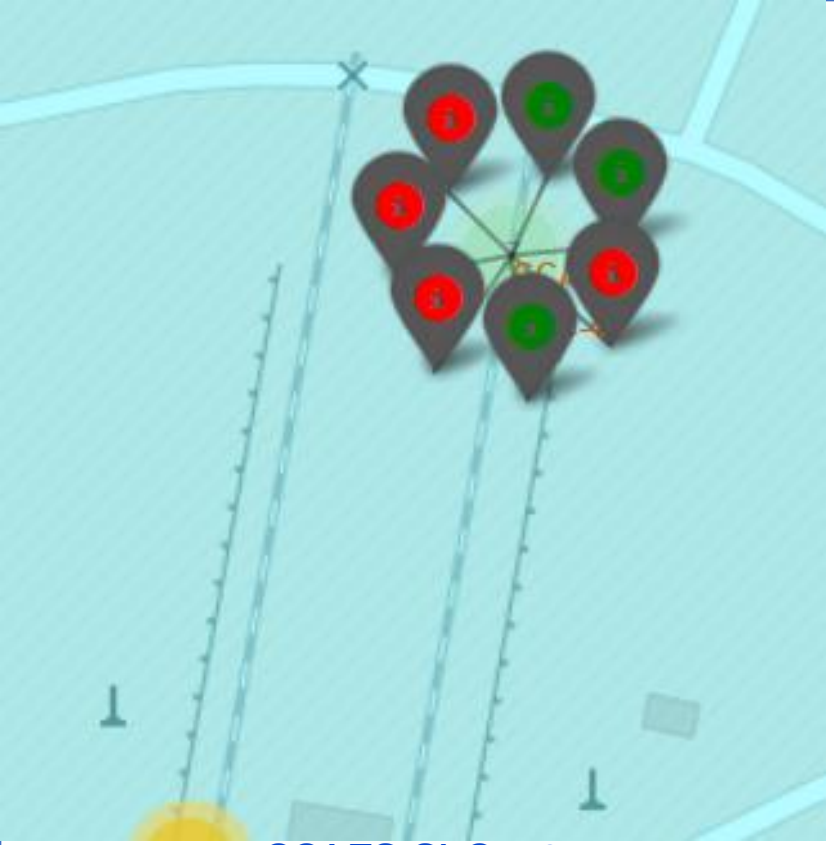
Section 3

Launch Sites Proximities Analysis

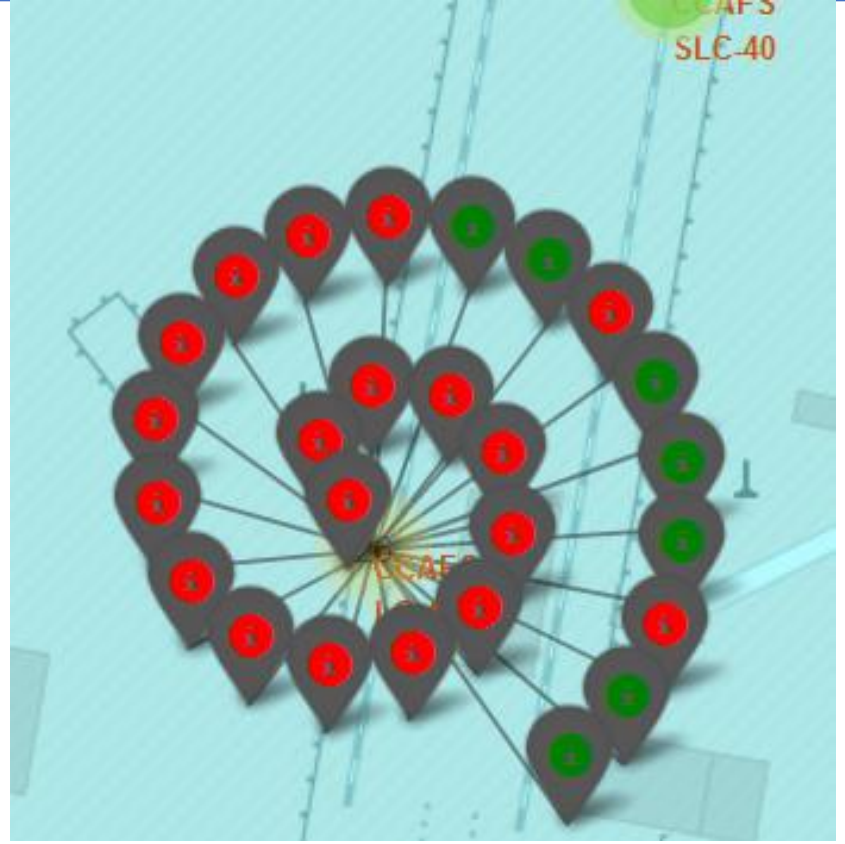


Launch Sites Markers on a Global Map

- All launch sites are near coastline and to the equator.



CCAFS SLC-40



CCAFS LC-40

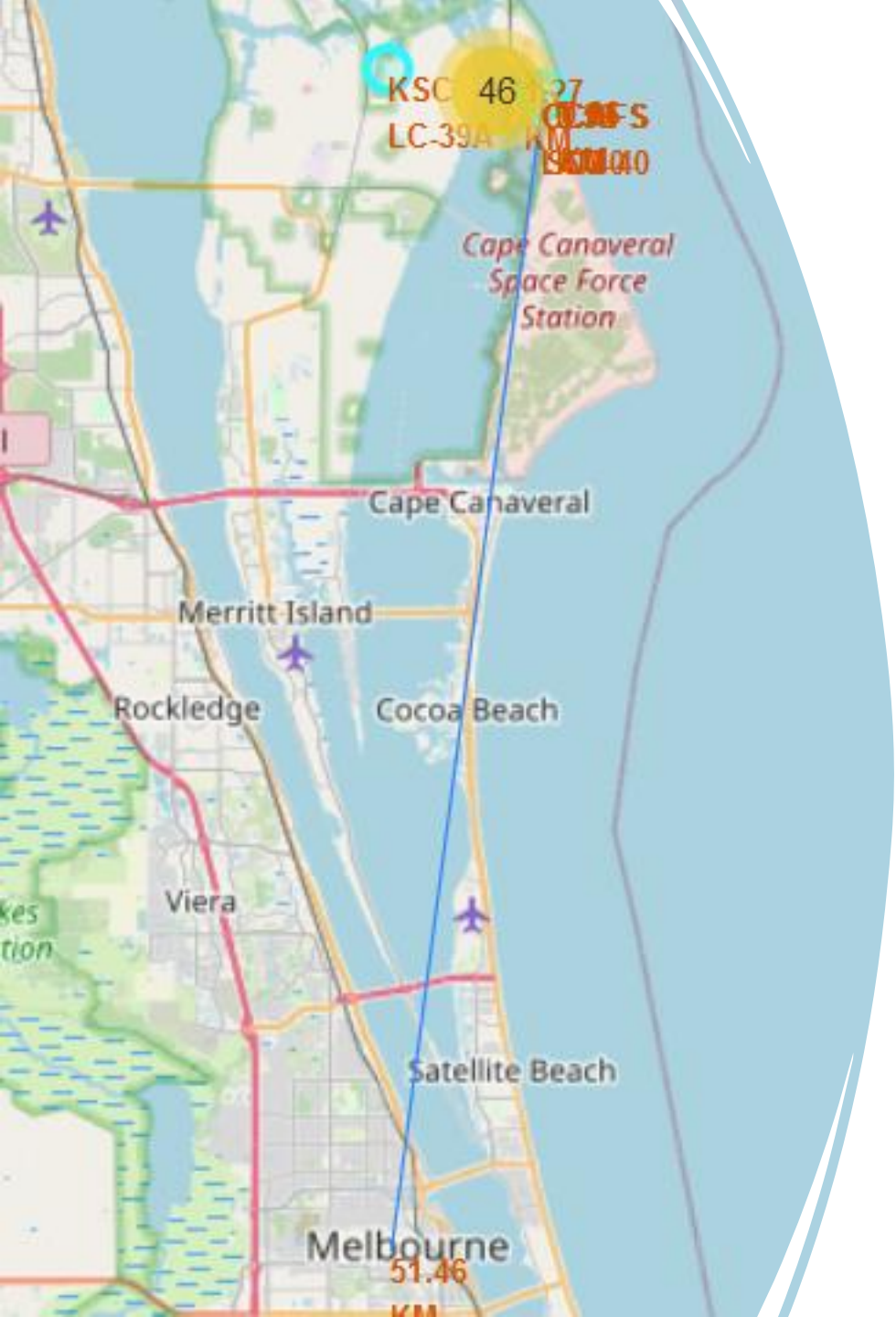
KSC LC-39A

Launch Outcomes for Each Site in Florida

- As per the map, launch site KSC LC-39A has relatively higher success rate as compared to CCAFS SLC-40 and CCAFS LC-40.

Launch Outcomes for VAFB SLC-4E in California

- As compared to KSC LC-39A in Florida, VAFB SLC-4E launch site has relatively lower success rate of nearly 40%.

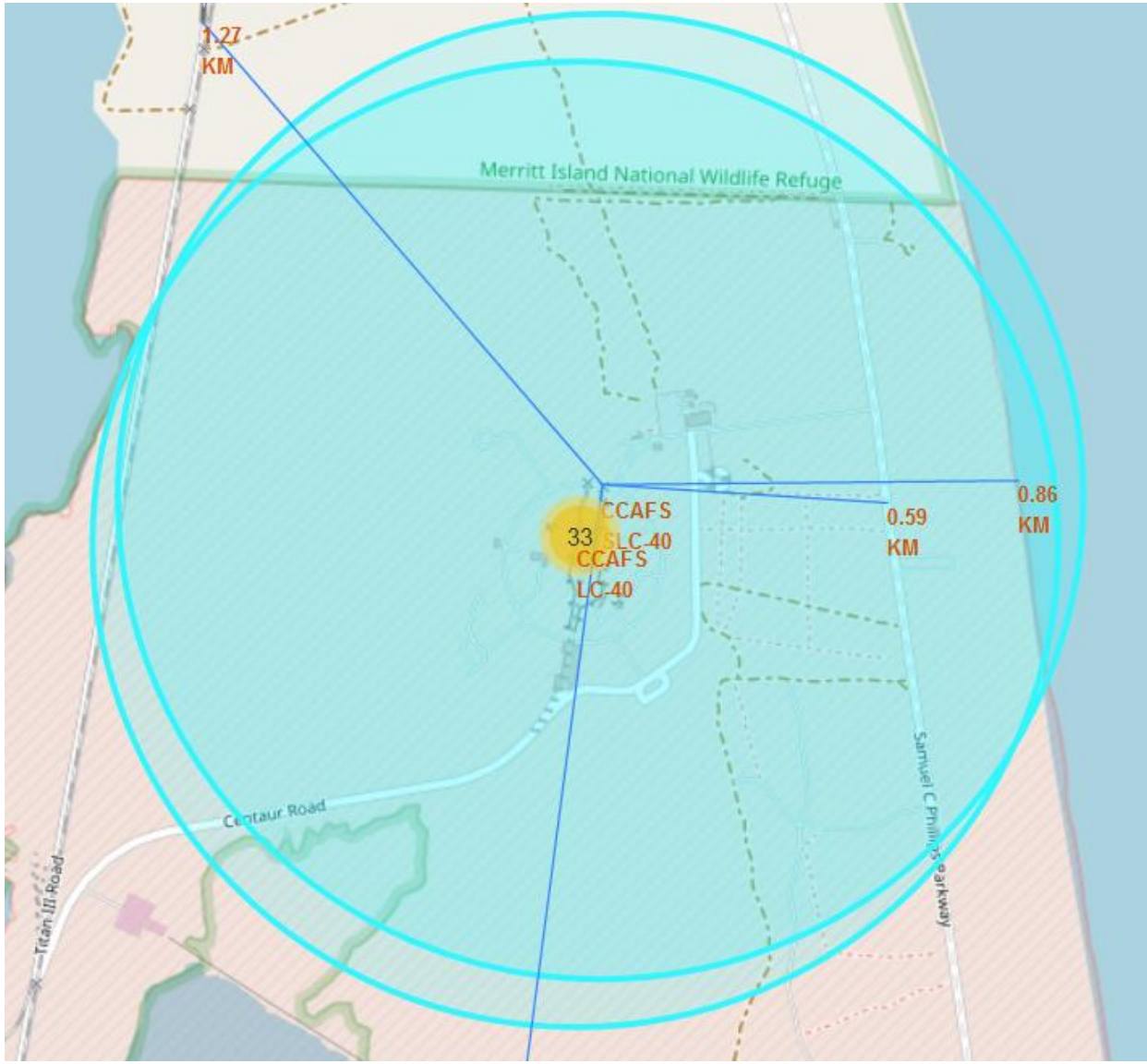


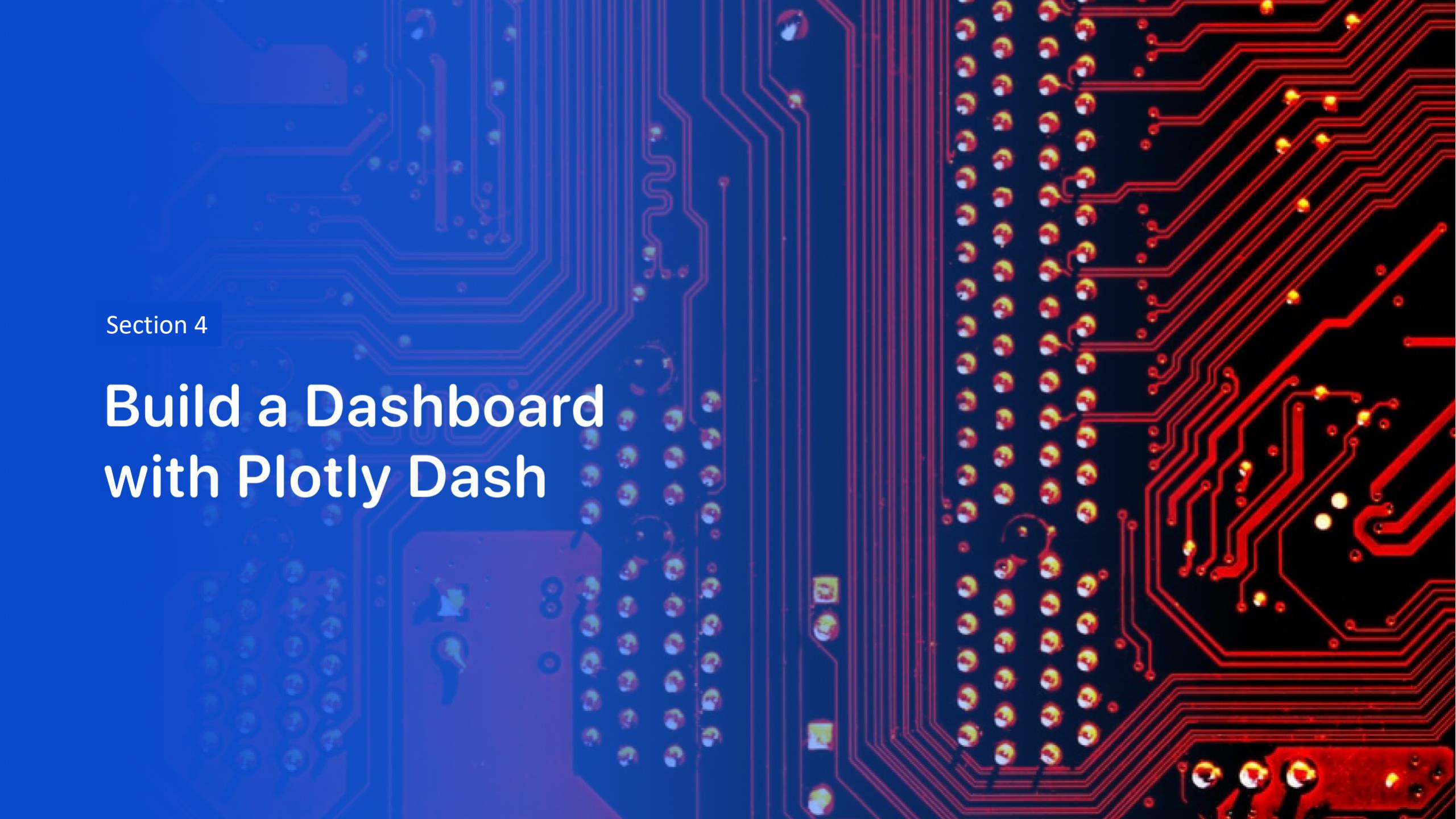
Distance From Closest City

- Closest metro city from the launch site CCAFS SLC-40 is **Melbourne** at a distance of ~ **51 KM**.
- It is so to keep mass from harm's way in case of some accidents and to maintain security of the site by preventing access of unauthorized.

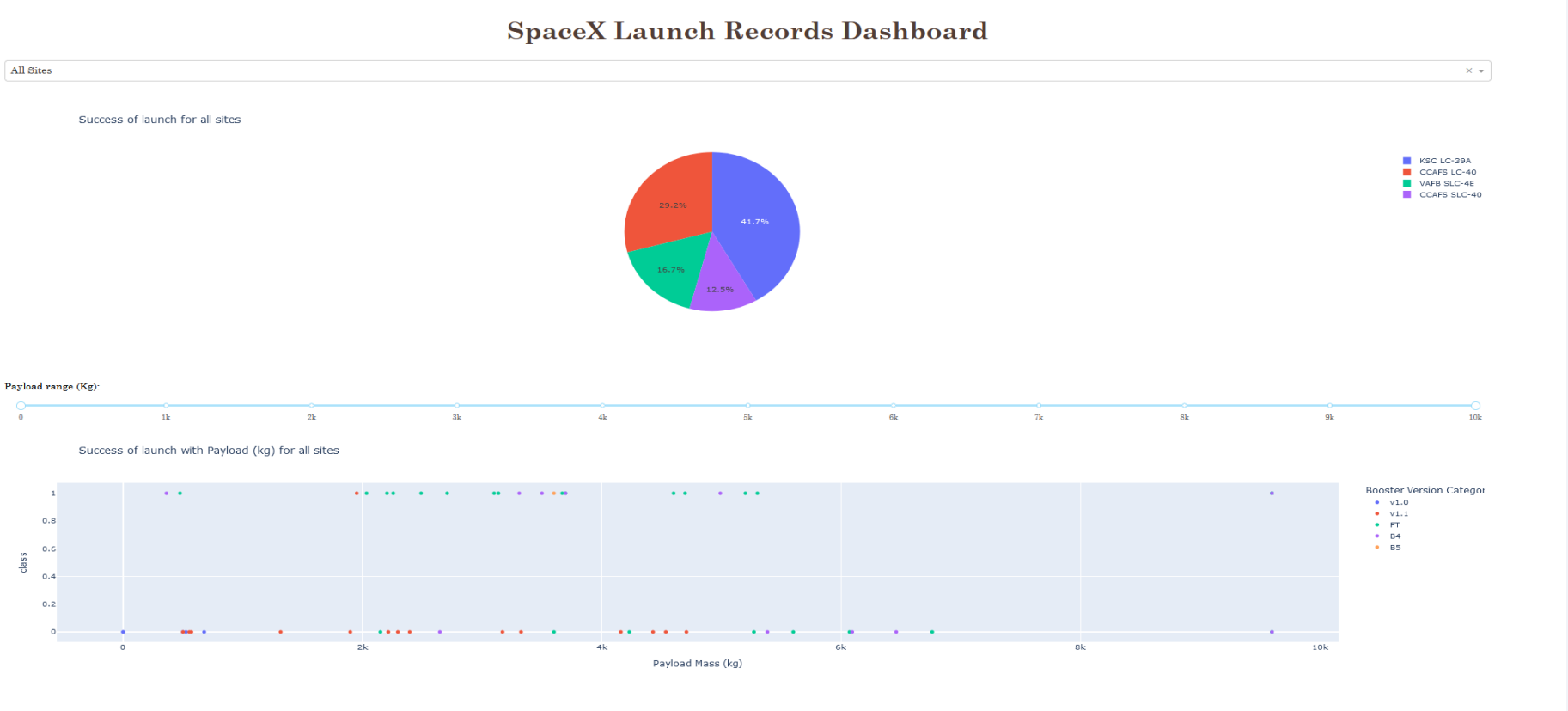
Distance From Launch Site to Other Proximities

- From CCAFS SLC-40:
  - Closest Rail Line: **1.27 KM**
  - Closest Highway: **0.59 KM**
  - Closest Coast Line: **0.86 KM**




Section 4
Build a Dashboard with Plotly Dash

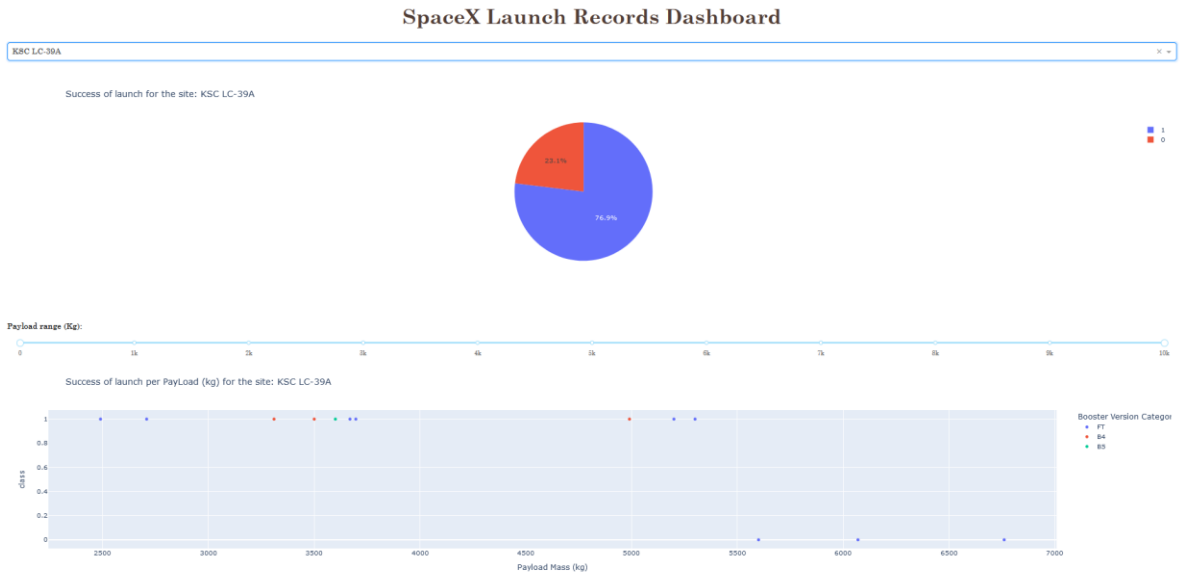
All Launch Sites Success Ratio

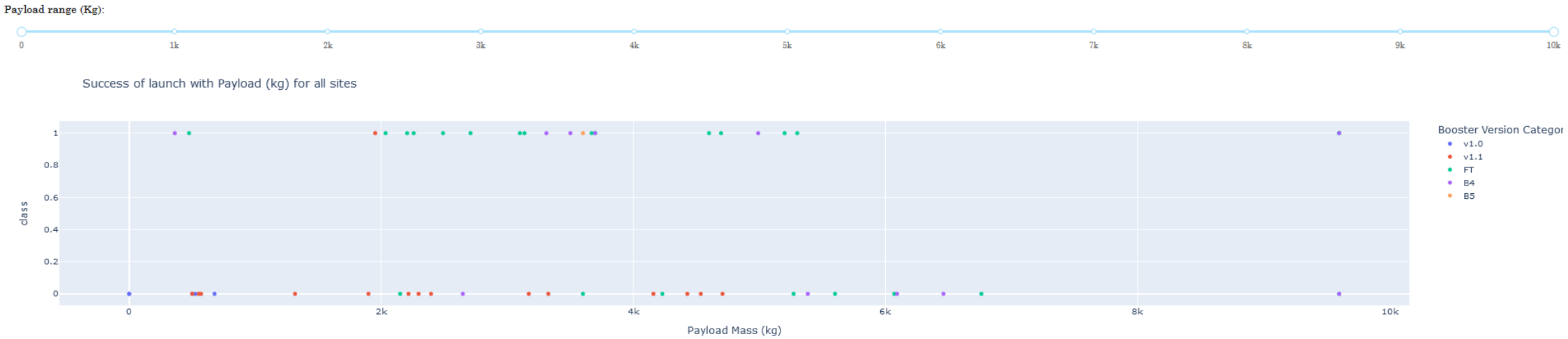


- KSL LC-39A has the highest launch success ratio of 41.7% followed by CCAFS LC-40 with 29.2% success ratio. In contrast, CCAFS SLC-40 has the lowest success ration of 12.5%.

Pie-chart Of The Site With Highest Launch Success Ratio

- The pie-chart depicts the result of the launch site **KSL LC-39A**.
- As per the observation we can deduce the fact that the success ratio is of **76.9%**.
- The scatter plot shows the payload carried by all launches from the site and their result (1 – Success or 0 - Failure)





Scatter Plot of Payload vs Launch Outcome

- **Observation**:
- Booster version **FT** has the highest success ratio among all the versions.
- The range of payload carried and resulted in a success for the booster FT is in between **2000 kg to 6000 kg**.

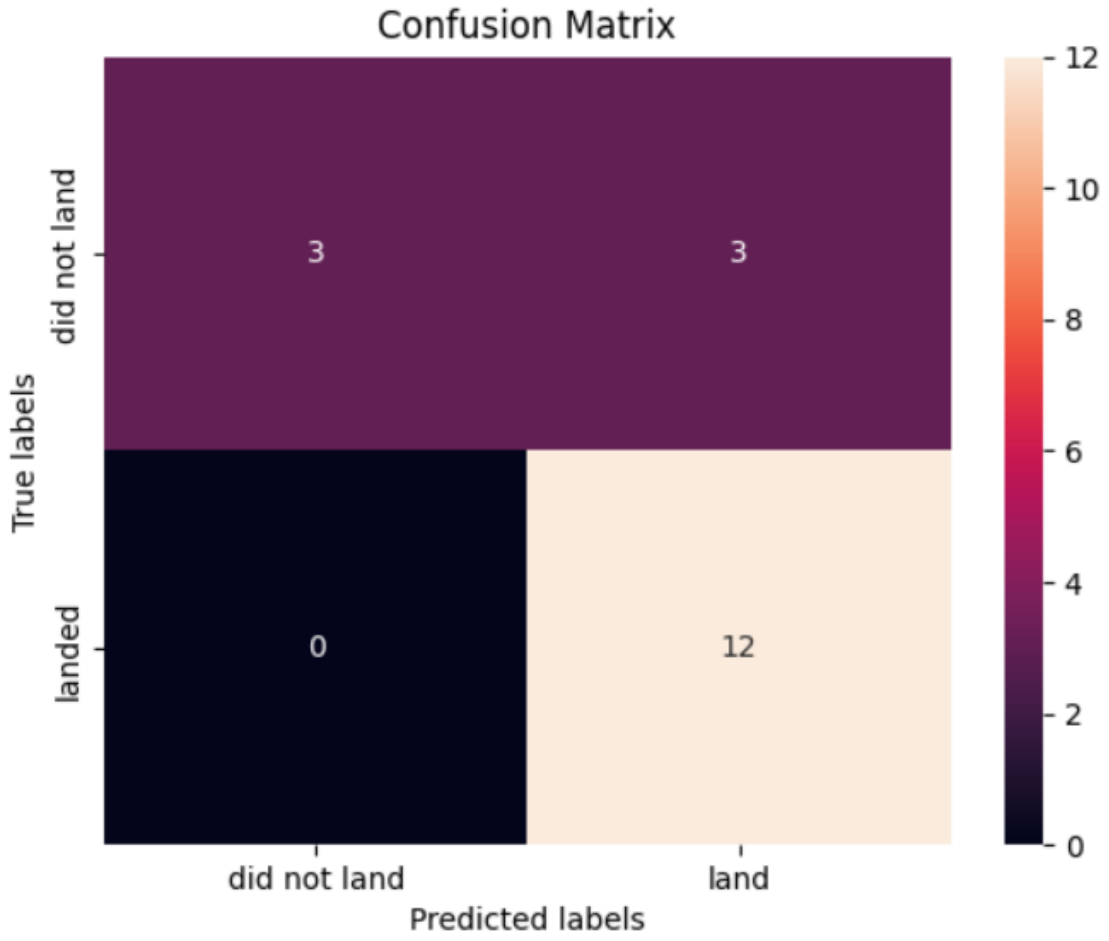
Section 5

Predictive Analysis (Classification)

Model	Accuracy on Test data
Logistic Regression	0.833333
SVM	0.833333
Decision Tree Classifier	0.833333
KNN	0.833333

Classification Accuracy

- All the classification models performed equally well on the testing data.
- Overall accuracy obtained was same across all the models which is **83.34%**.



Confusion Matrix

- Confusion matrix remained the same across all the classification models as they distinguished results equally in each class.
- However, all the models did the same error while predicting outcome when launches were not a success and the results predicted were that of a success. (false positive).

Conclusions

- The success rate of launches at launch sites KSC LC-39A and CCFS SLC-40 significantly improves as the flight number increases. After the 77th flight, the success rate reaches 100% at these sites. Similarly, at launch site VAFB SLC-4E, the success rate reaches 100% after the 50th flight.
- When it comes to heavy payloads, the landing success rate is notably higher for missions targeting polar orbits, low Earth orbits (LEO), and the International Space Station (ISS). However, in the case of Geostationary Transfer Orbits (GTO), it is challenging to differentiate the success of landings accurately. This is because both positive landing rates (successful missions) and negative landing rates (unsuccessful missions) exist for GTO missions.
- Note that the success rate since 2013 kept increasing till 2020.
- According to the data presented on the map, launch site KSC LC-39A exhibits a comparatively higher success rate compared to CCAFS SLC-40, VAFB SLC-4E, and CCAFS LC-40.
- Based on the data provided, KSC LC-39A has the highest launch success ratio, standing at 41.7%. Following closely is CCAFS LC-40 with a success ratio of 29.2%. In contrast, CCAFS SLC-40 has the lowest success ratio of 12.5%.

Conclusions (Cont.)

- While KSC LC-39A may have the highest overall success ratio, it is worth noting that this launch site did not perform well with heavy payloads. Specifically, launches with payloads exceeding 5500 kg resulted in failures at this site.
- Among all the booster versions, the FT version has the highest success ratio. The payload range carried by the FT booster that resulted in a successful mission falls between 2000 kg and 6000 kg.

Thank you!