IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

SUMITH KARUN

JUNE 25 2024

Outline

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

Executive Summary - Methodologies

- Data collection from SpaceX RESTAPI by web scraping using python library-beautiful soup.
- Data wrangling, cleaning, and transformation with Python libraries Numpy and Pandas.
- Uncovering trends through exploratory data analysis with SQLite and SQL queries in Python.
- Interactive charts and maps created with Folium and Plotly.
- Predictive modeling with machine learning techniques to accurately determine landing outcomes

Executive Summary - Results

- Exploratory Data Analysis
- Interactive Visualizations
- Optimal machine learning model for predictive analysis

Introduction

Falcon 9 rocket launches are offered by SpaceX at a substantially lower cost of $62 million each, unlike other providers whose prices exceed $165 million per launch. These savings are due to SpaceX's pioneering practice of reusing the rocket's first stage. This project aims to forecast the landing success of Falcon 9's first stage in future missions. These predictions will not only enhance understanding of landing outcomes but also offer valuable insights into launch costs, aiding companies competing with SpaceX for future rocket launch contracts.

Problem statement

- Space Y would like to compete with SpaceX founded by Billionaire industrialist Allon Musk.
  - Determine the price of each launch by gathering information about Space X and creating dashboards for the team.
  - Determine if SpaceX will reuse the first stage also determine if the first stage will land successfully by developing a machine learning model and use public information to predict if SpaceX will reuse the first stage.

Section 1

Methodology

Methodology-Data Collection

- SpaceX REST API: https://api.spacexdata.com/v4/launches/past
  - SpaceX REST API provides comprehensive data on launches, including details about the rocket, payload information, specifications for launch and landing, and the outcomes of the landing.
- Web Scrapping:https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_He avy_launches&oldid=1027686922
  - Using Python's package - beautiful soup HTML tables from Wikipedia page were retrieved.

Data Collection- SpaceX REST API

```
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
```

Converting JSON>DataFrame

```
# Use json_normalize meethod to convert the json result into a dataframe
file_name="API_call_spacex_api.json"

if response.status_code==200:
    with open(file_name,"wb") as file:
        file.write(response.content)
    print("file downloaded")
else:
    print("File not downloaded,status code: ",response.status_code)
```

```
file downloaded
```

```
data  = pd.read_json(file_name)
# Set pandas display options to show all columns
pd.set_option('display.max_columns', None)
```

Data Collection- SpaceX REST API

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have
# multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Consolidating Dataset

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0

Github url:https://github.com/sumith1980/sumi_code/blob/main/1-jupyter-labs-spacex-data-collection-api.ipynb

Data Collection- Web Scraping

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
3]: 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

```
4]: # use requests.get() method with the provided static_url
    # assign the response to a object
    response = requests.get(static_url)
    # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
    soup = BeautifulSoup(response.content, 'html.parser')
    # Use soup.title attribute
    soup.title
```

Web scraping Data

```
column_names = []

table = first_launch_table.find_all('th')
for row in table:
    name = extract_column_from_header(row)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
```

Data Collection- Web Scraping

```
# Booster landing
# Append the launch_outcome into launch_dict with key `Booster landing`
booster_landing = landing_status(row[8])
launch_dict["Booster landing"].append(booster_landing)
#print(booster_landing)
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

Consolidating Dataset

```
df.head()
```

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	[[SpaceX], \n]	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	[[.mw-parser-output .plainlist ol,.mw-parser-o...	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	[[NASA], (, [COTS],)\n]	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	[[NASA], (, [CRS],)\n]	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	[[NASA], (, [CRS],)\n]	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

Github url:https://github.com/sumith1980/sumi_code/blob/main/2-jupyter-labs-webscraping.ipynb

Data Wrangling-Determine Training Labels

```
# landing_outcomes = values on Outcome column
landing_outcomes=df['Outcome'].value_counts()
```

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

```
bad outcomes=set(landing outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

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 = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
```

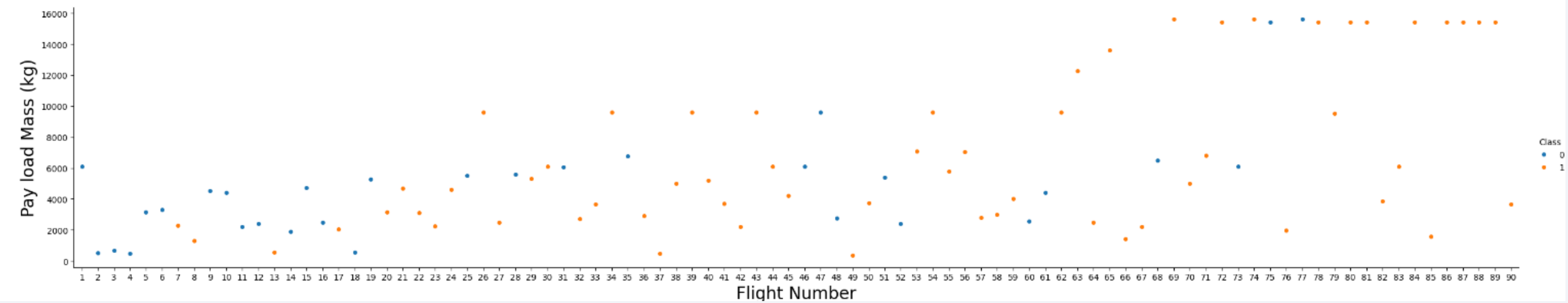
```
df['Class']=landing_class
df[['Class']].head(8)
```

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

Github url:https://github.com/sumith1980/sumi_code/blob/main/3-labs-jupyter-spacex-Data%20wrangling.ipynb

Exploratory Data Analysis

Catplot-FlightNumber Vs PayloadMass

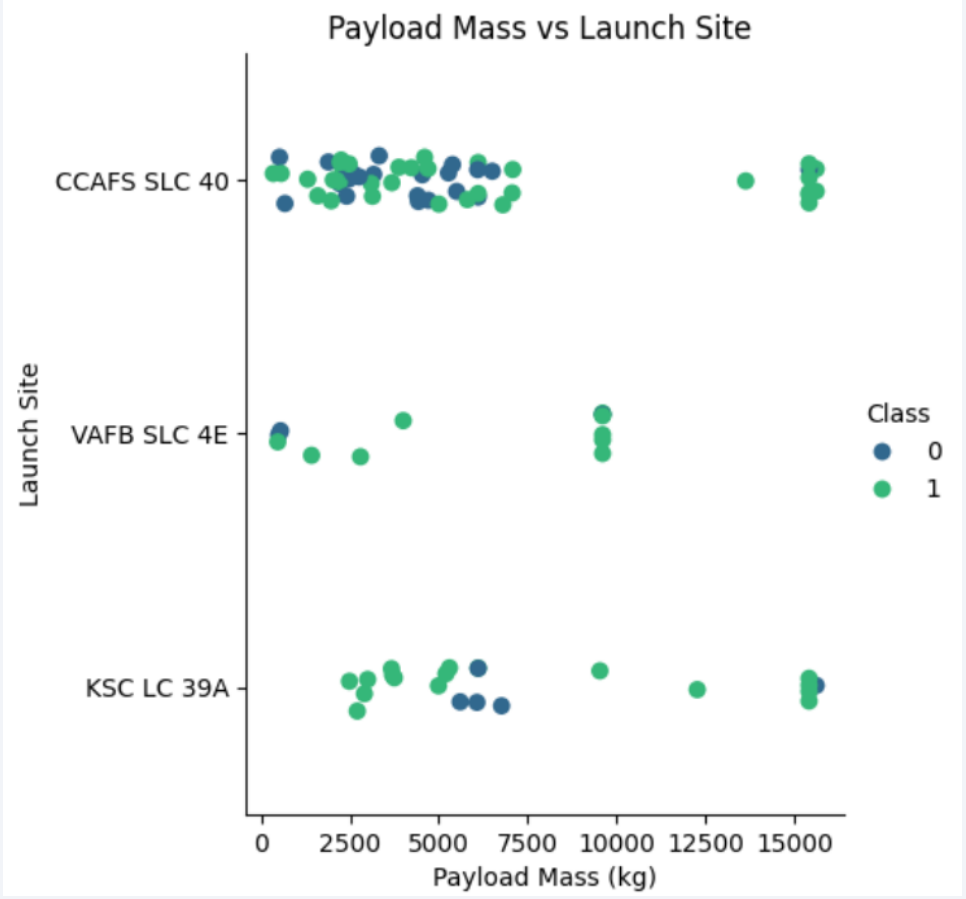


We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

Exploratory Data Analysis

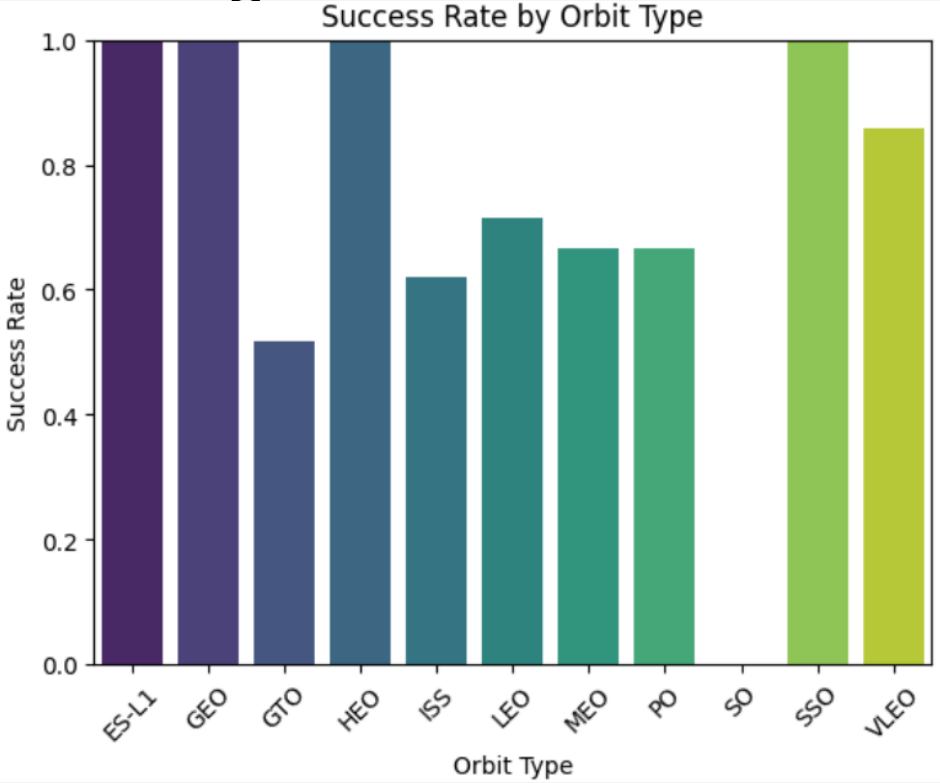
Various scatter plots effectively illustrate relationships between variables

- Launch Site and Payload Mass
- Launch Site and Flight Number
- Flight Number and Orbit Type
- Payload and Orbit Type

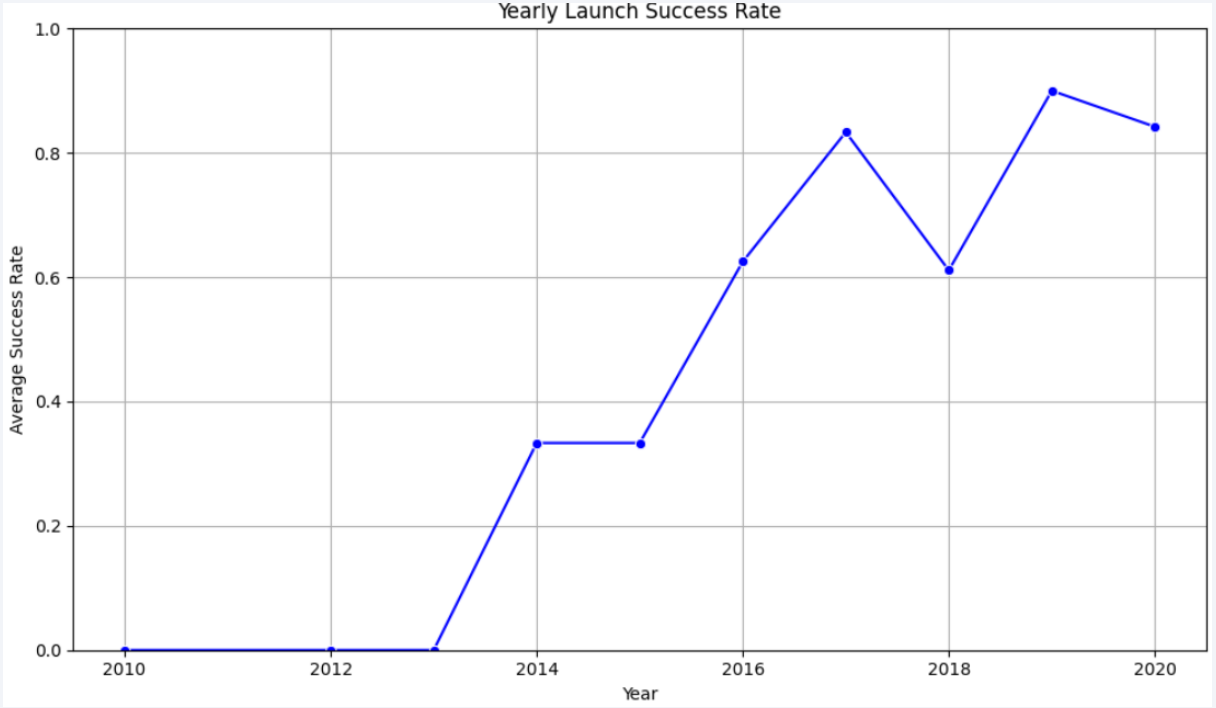


Exploratory Data Analysis

Bar Plot of Success Rates by Orbit. Indicates which orbits have the highest likelihood of successful landings



Line plot of success rates over the years, indicating a gradual increase



Exploratory Data Analysis-SQL

- The SQL queries were executed on a SpaceX Falcon 9 dataset to gain insights into launch outcomes and booster performance.
- The initial query filtered records where the date was not null.
- Subsequent queries identified unique launch sites, displayed specific records with launch sites starting with 'CCA,' calculated the total payload mass for NASA (CRS) launches,
- Determined the average payload mass for booster version F9 v1.1, pinpointed the date of the first successful ground pad landing, listed boosters with successful drone ship landings and specific payload masses, tallied successful and failed mission outcomes, found booster versions with the maximum payload mass.
- Ranked landing outcomes based on counts within a specified date range.

Github url:https://github.com/sumith1980/sumi_code/blob/main/5-jupyter-labs-eda-sql-coursera_sqllite.ipynb

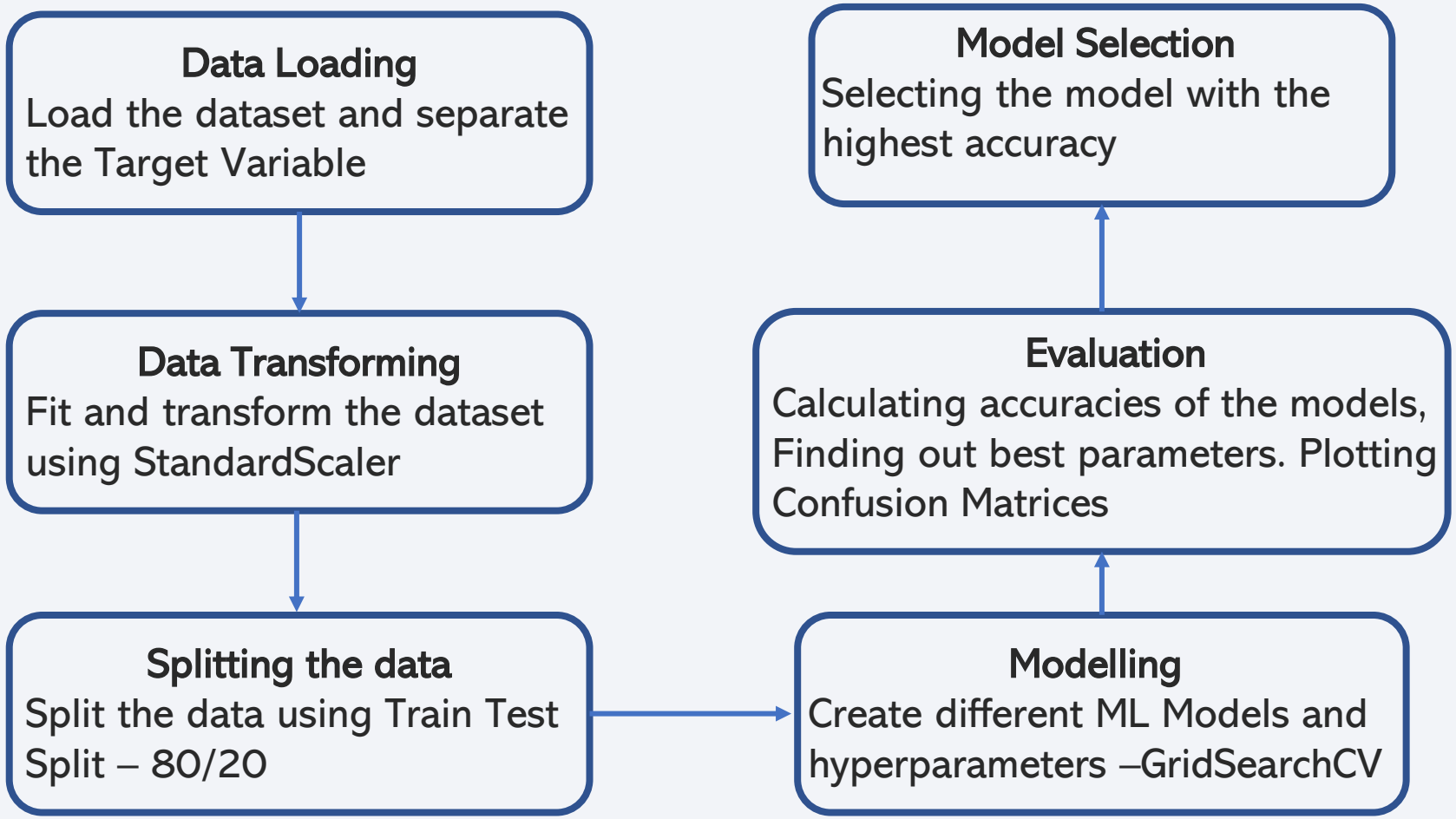
Folium-Interactive Map

- The SQL queries were executed on a SpaceX Falcon 9 dataset to gain insights into launch outcomes and booster performance.
- The initial query filtered records where the date was not null.
- Subsequent queries identified unique launch sites, displayed specific records with launch sites starting with 'CCA,' calculated the total payload mass for NASA (CRS) launches,
- Determined the average payload mass for booster version F9 v1.1, pinpointed the date of the first successful ground pad landing, listed boosters with successful drone ship landings and specific payload masses, tallied successful and failed mission outcomes, found booster versions with the maximum payload mass.
- Ranked landing outcomes based on counts within a specified date range.

Plotly Dash-Dashboard

- The SQL queries were executed on a SpaceX Falcon 9 dataset to gain insights into launch outcomes and booster performance.
- The initial query filtered records where the date was not null.
- Subsequent queries identified unique launch sites, displayed specific records with launch sites starting with 'CCA,' calculated the total payload mass for NASA (CRS) launches,
- Determined the average payload mass for booster version F9 v1.1, pinpointed the date of the first successful ground pad landing, listed boosters with successful drone ship landings and specific payload masses, tallied successful and failed mission outcomes, found booster versions with the maximum payload mass.
- Ranked landing outcomes based on counts within a specified date range.

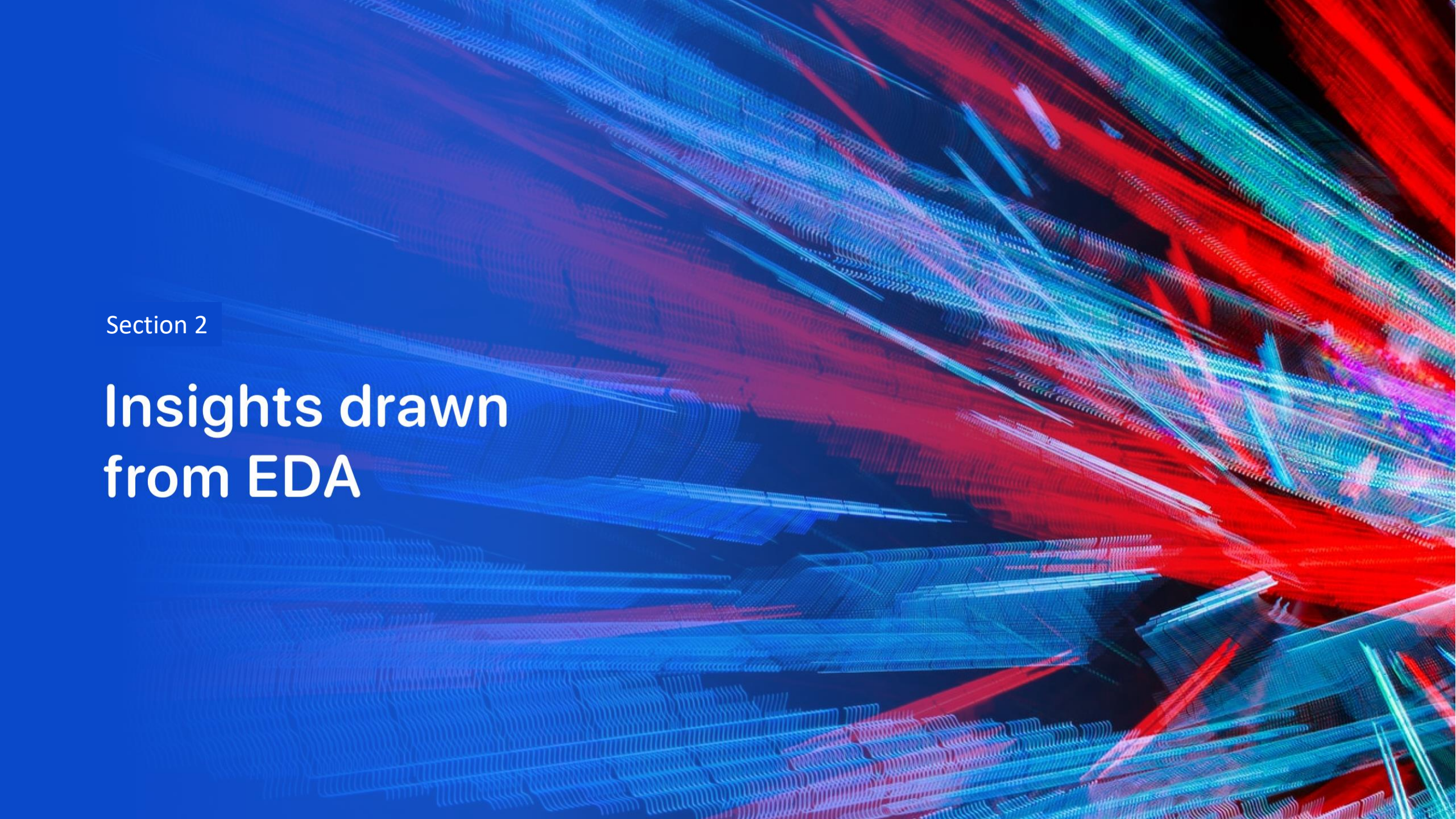
Predictive Analysis (Classification)



Github url: https://github.com/sumith1980/sumi_code/blob/main/8-SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

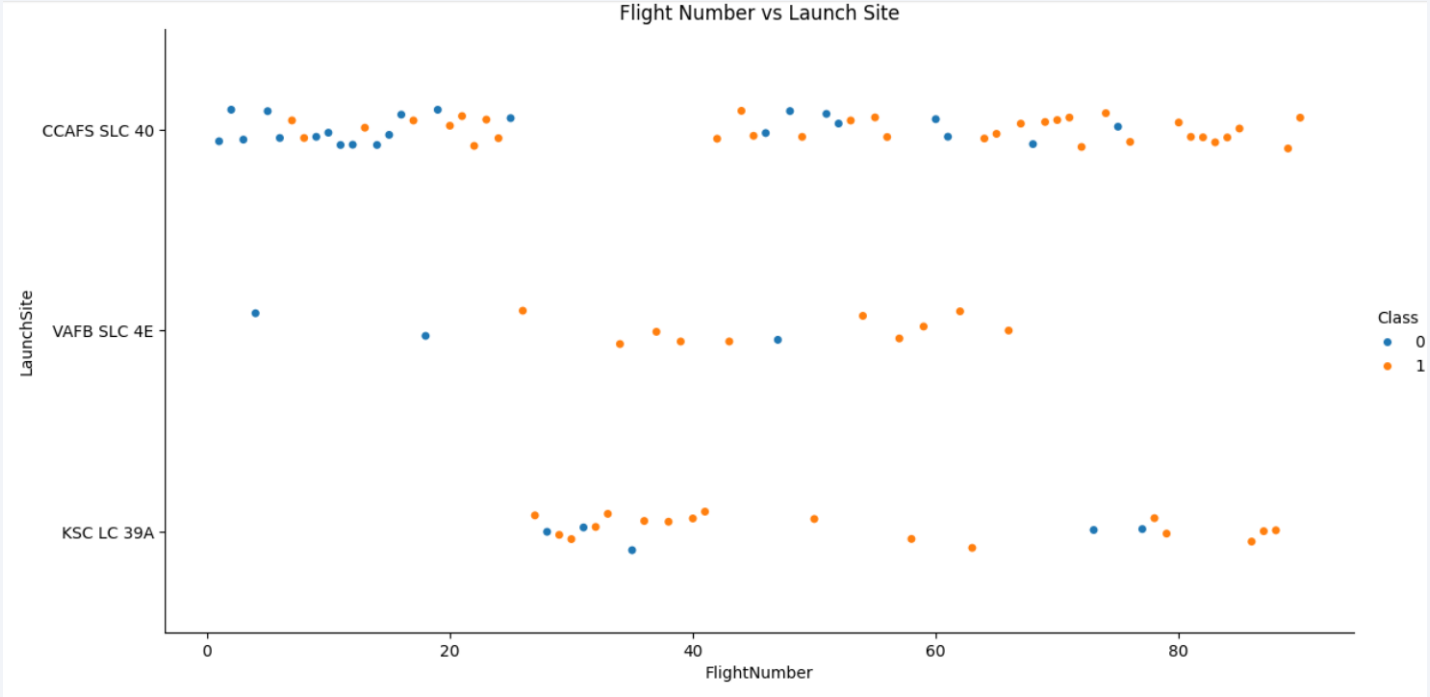
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

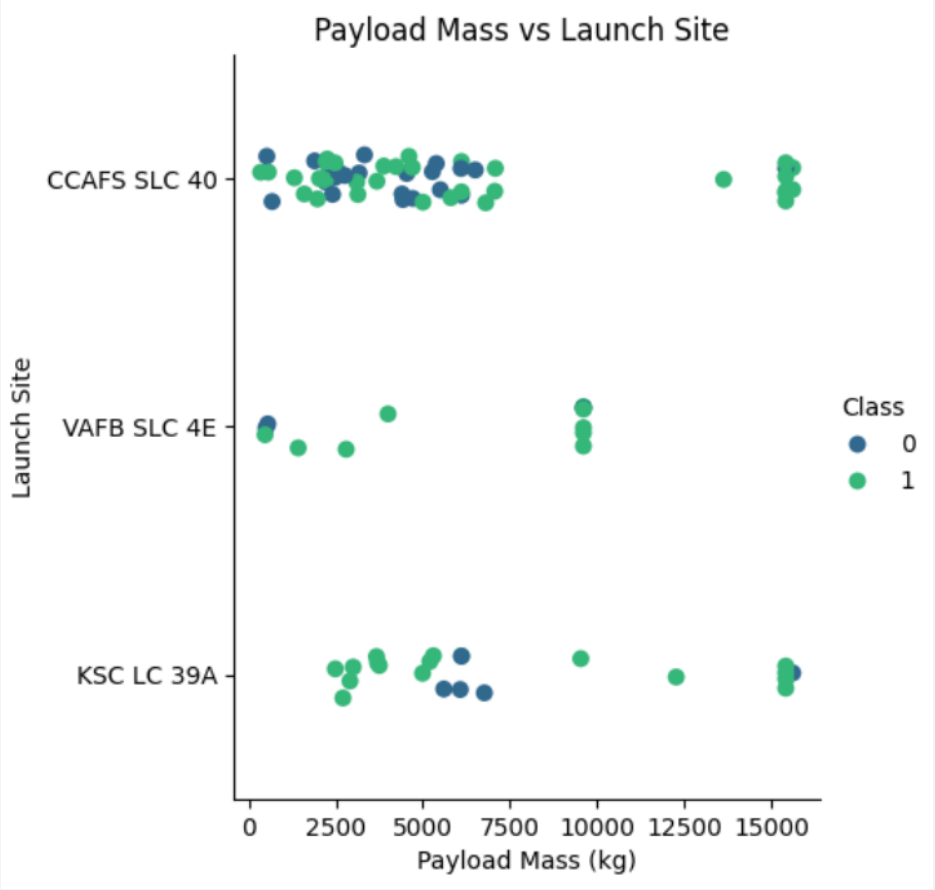
- For every launch site, as the flight number increased, the success rate increased

- From the plot , we can observe that the launch site KSC LC 39A has the highest success rate.



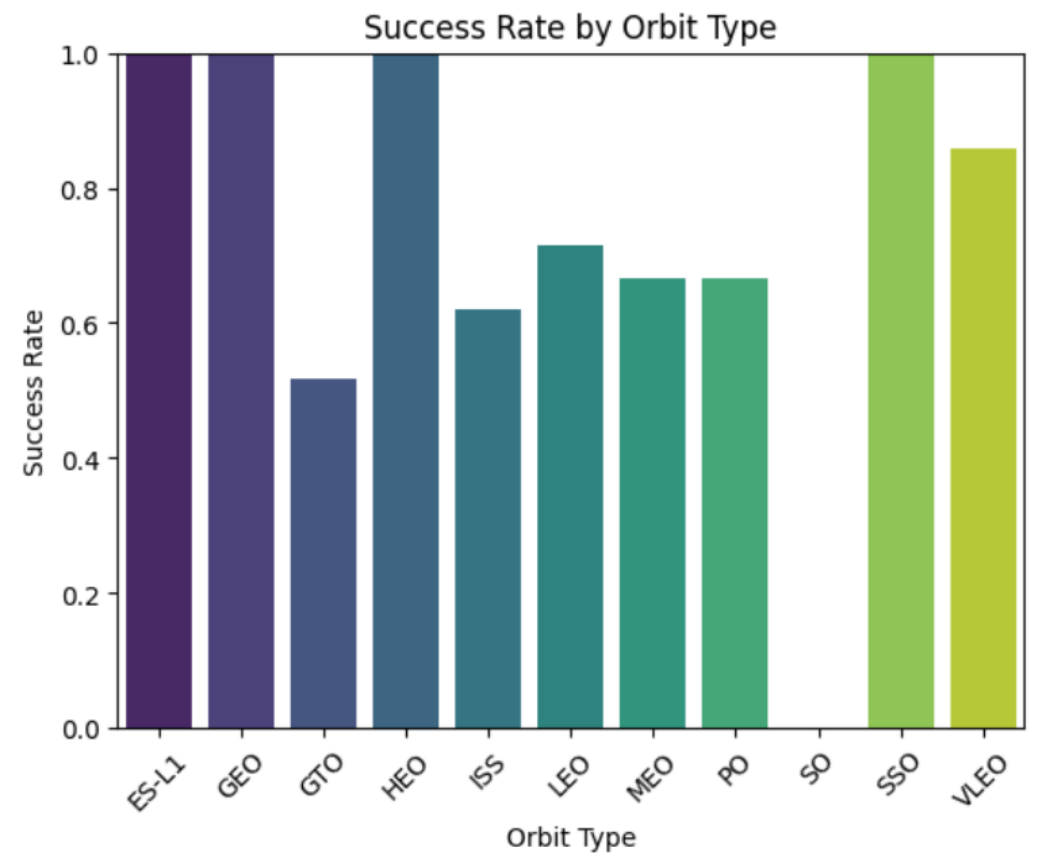
Payload vs. Launch Site

- Payloads over 8000 kg are successful
- Lower payloads have lower success rates
- Spacex opted to launch larger payloads at CCAFS SLC 40 and KSC LC 39A



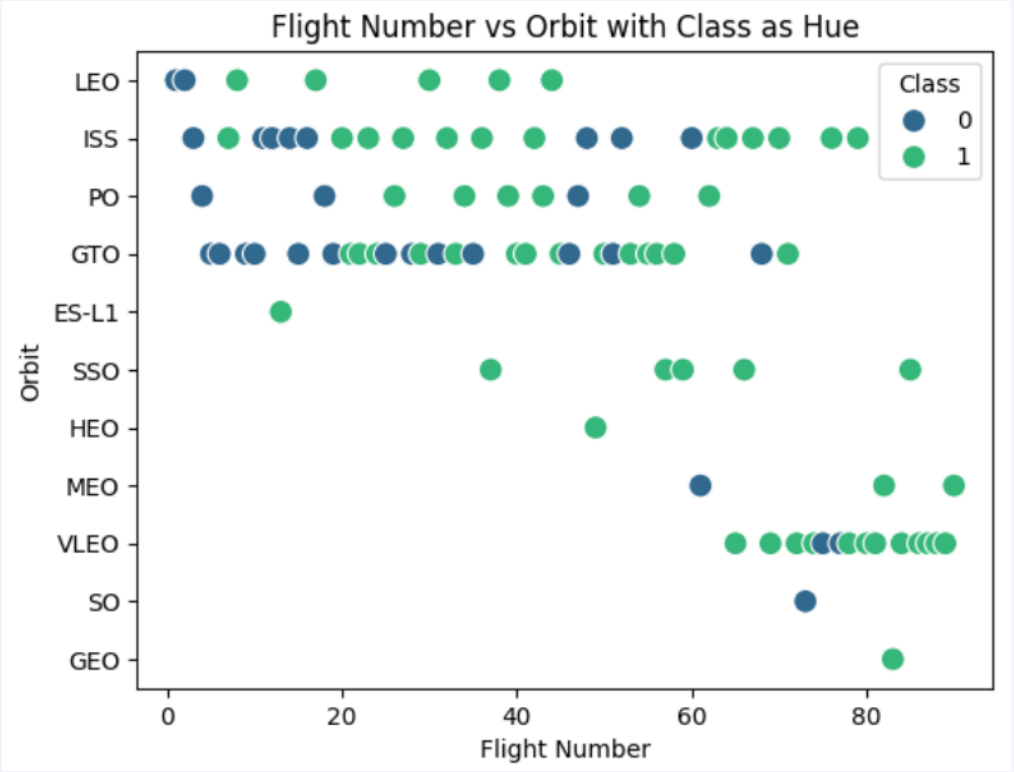
Success Rate vs. Orbit Type

- Orbits ES-L1, GEO, HEO, SSO have the highest success rates.
- Orbit SO did not have a single successful otcome.



Flight Number vs. Orbit Type

- Of the four orbits with the highest success rates identified in the previous slide, only SSO has experienced multiple launches, making it appear to be the most favorable orbit for successful outcomes.

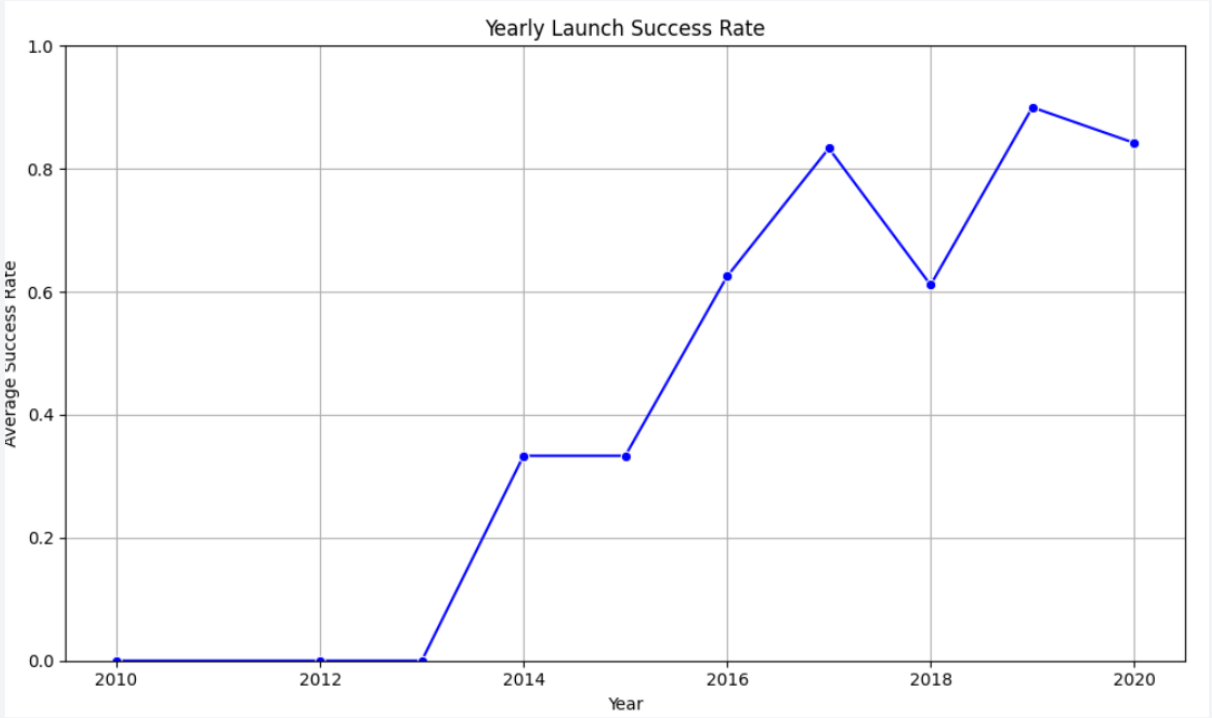


Payload vs. Orbit Type

- Out of ISS, PO, and VLEO, the only orbits with significantly higher payloads, ISS and LEO have demonstrated the highest success rates.
- GTO orbit is tough to analyze at it has positive and negative outcomes .

Launch Success Yearly Trend

- Initial years there was hardly any success.
- We can observe that the success rate since 2013 kept increasing till 2017 (stable in 2014) and after 2015 it started increasing



All Launch Site Names

- %sql SELECT DISTINCT(Launch_Site) FROM SPACEXTBL
- This query is used to retrieve a list of unique launch sites from the SPACEXTBL table. It ensures that each launch site is listed only once in the results, even if it appears multiple times in the table.

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

Launch Site Names Begin with 'CCA'

- %sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
- This SQL query retrieves records from a table named SPACEXTBL where the Launch_Site column values start with 'CCA'. It limits the result to a maximum of 5 rows.

Date	Time (UTC)	Booster_Version	Launch_Site	Payload
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2

Total Payload Mass

- %sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass FROM SPACEXTBL WHERE Customer='NASA (CRS)';
- The provided code snippet is an SQL query designed to calculate the total payload mass for launches where the customer is 'NASA (CRS)'.

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

- %sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1%’
- This SQL query calculates the average payload mass for the booster version 'F9 v1.1' in the SPACEXTABLE dataset.

Average_Payload_Mass
2534.6666666666665

First Successful Ground Landing Date

- %sql SELECT MIN(Date) AS First_Successful_Landing_Date FROM SPACEXTBL WHERE Landing_Outcome= 'Success (ground pad)'
- Chooses the earliest year from the table when a landing was successfully achieved on the ground pad

First_Successful_Landing_Date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- %sql SELECT DISTINCT(Booster_Version) FROM SPACEXTBL WHERE Landing_Outcome ='Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
- Selects the booster version from the table for which a landing was successfully completed on a drone ship with a payload mass between 4000 and 6000

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

Total Number of Successful and Failure Mission Outcomes

- %sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS "SUCCESS", (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS "FAILURE";
- The query returns the counts of mission outcomes for launches, categorized by success and failure

SUCCESS	FAILURE
100	1

Boosters Carried Maximum Payload

- %sql SELECT * FROM (SELECT Booster_Version, MAX(PAYLOAD_MASS__KG_) AS 'MAX_MASS' FROM SPACEXTBL GROUP BY 1 ORDER BY 2 DESC) AS A WHERE MAX_MASS=15600 ;
- A subquery is used to select boosters with max payloads.

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

2015 Launch Records

- %sql SELECT substr(Date, 6, 2) AS Month_Name,Booster_Version,Launch_Site,Landing_Outcome FROM SPACEXTBL WHERE substr(Date, 1, 4) = '2015' AND Landing_Outcome = 'Failure (drone ship)';
- This query returns the month, booster version, launch site, and landing outcome.

Month_Name	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)

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

- %sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTBL WHERE Date >= '2010-06-04' AND Date <= '2017-03-20' GROUP BY Landing_Outcome ORDER BY Outcome_Count DESC;

- Selects distinct landing outcomes and ranks them according to their frequency

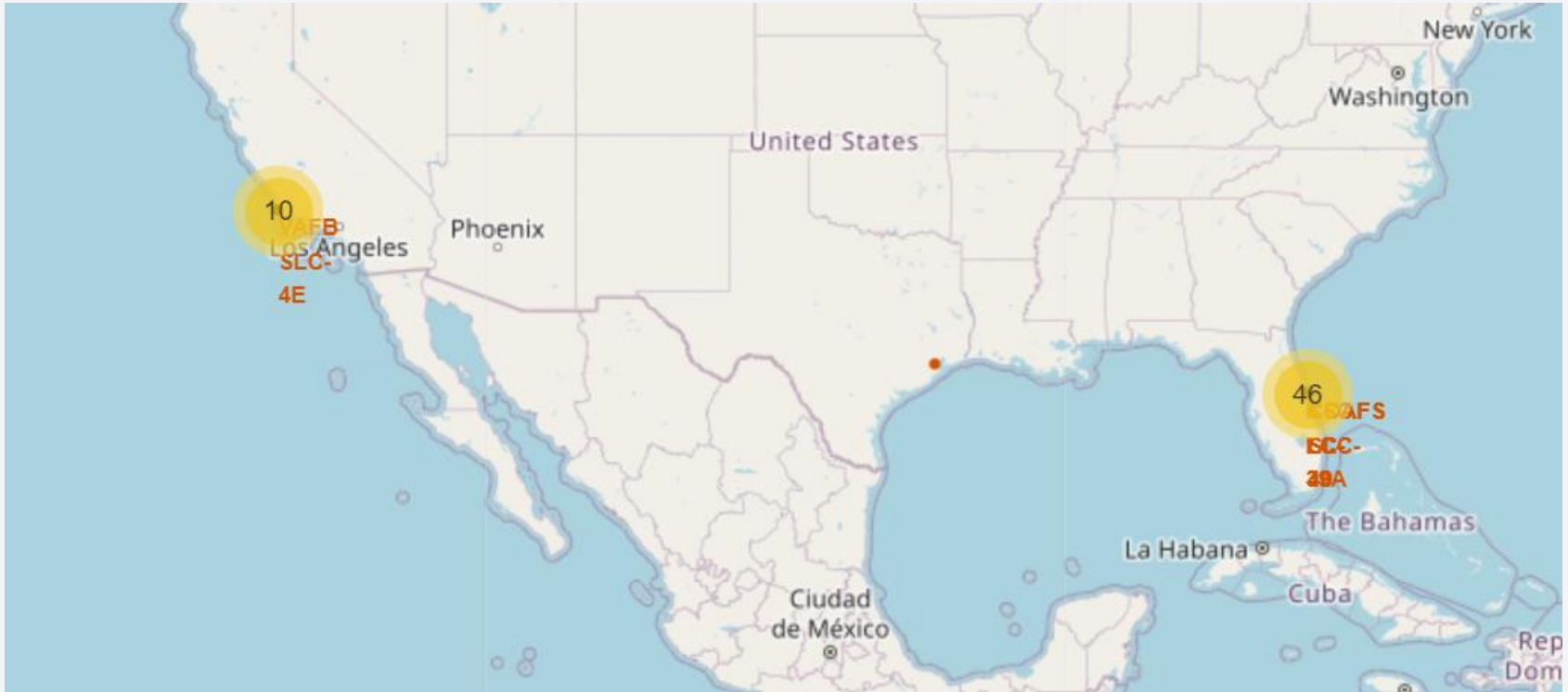
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

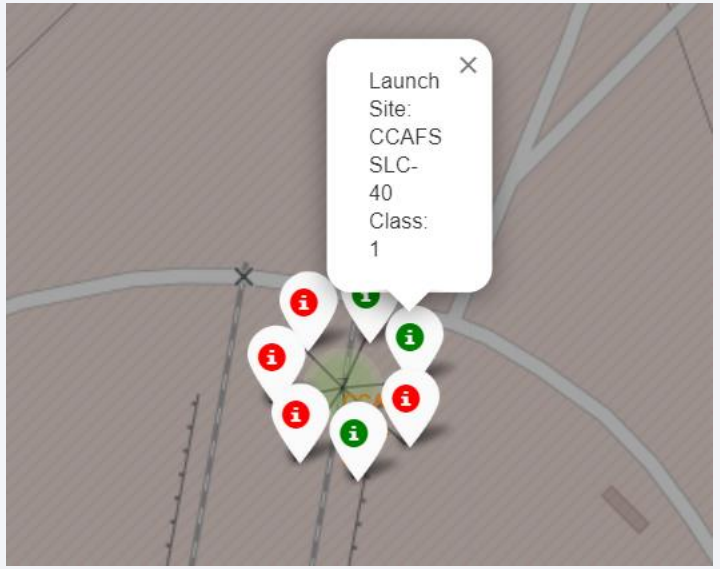
Launch Sites Locations Analysis

- All the launch sites are in USA. Particularly in the states of Florida and California.
- Launch sites are significantly close to the coasts.



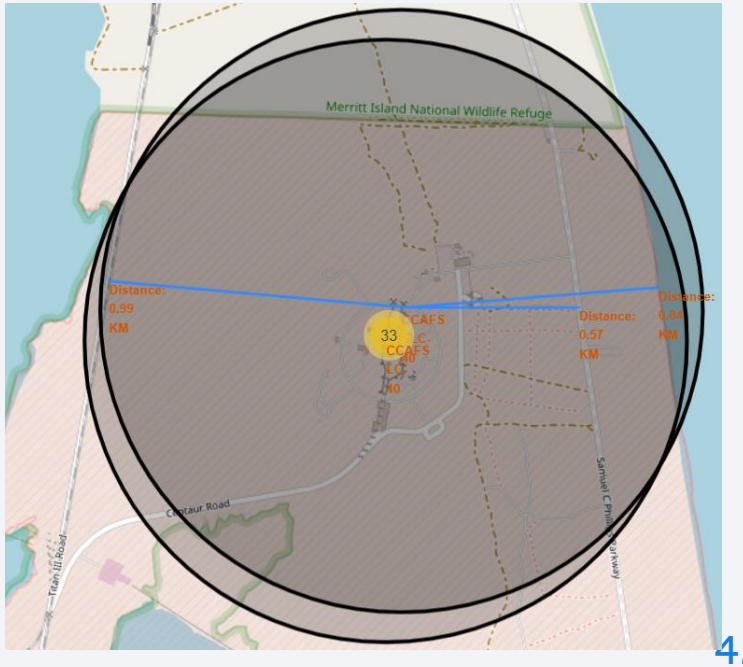
Success and Failed Launches

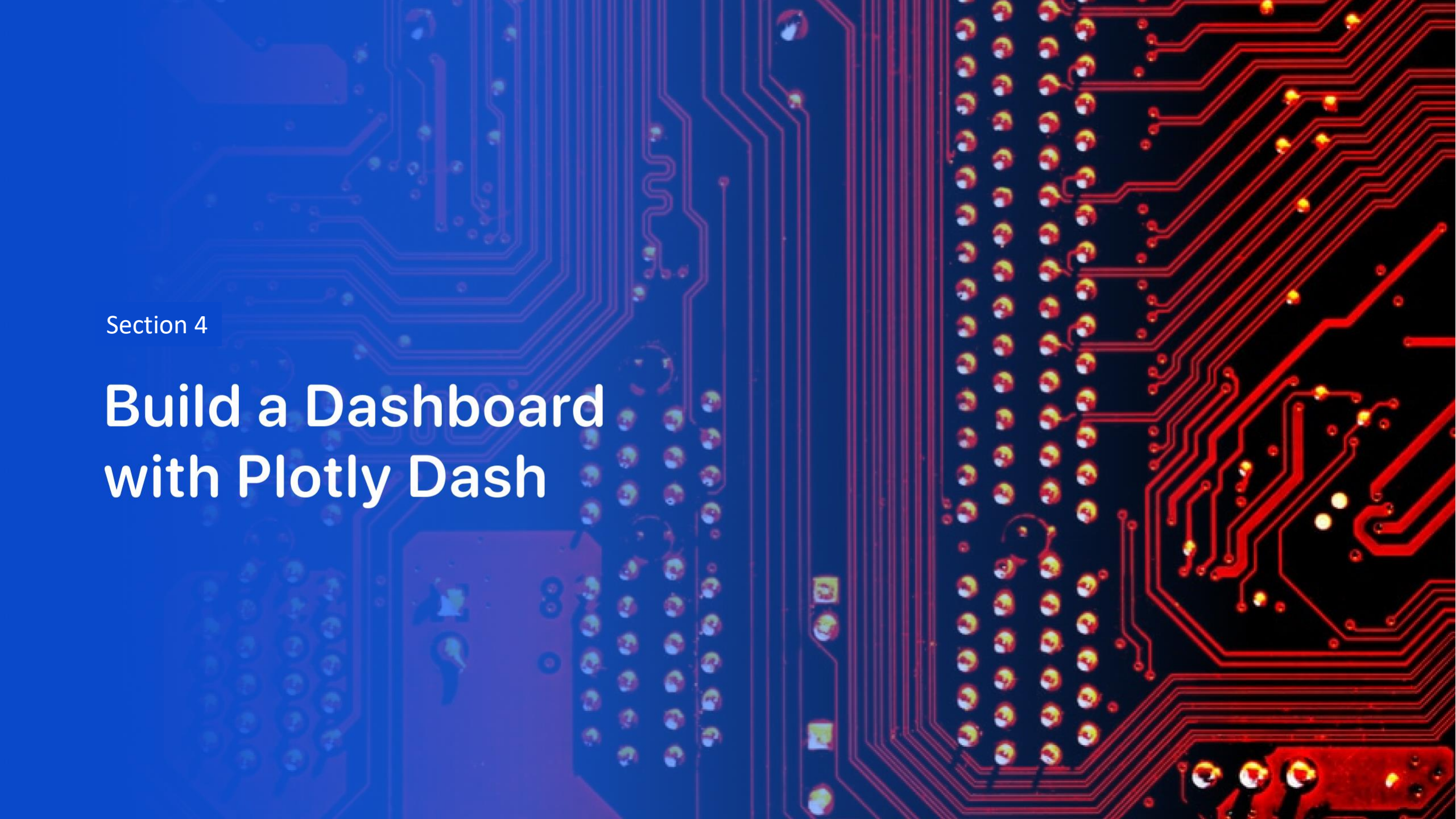
- Green popups represent successful outcomes.
- Red popups represent failed outcomes



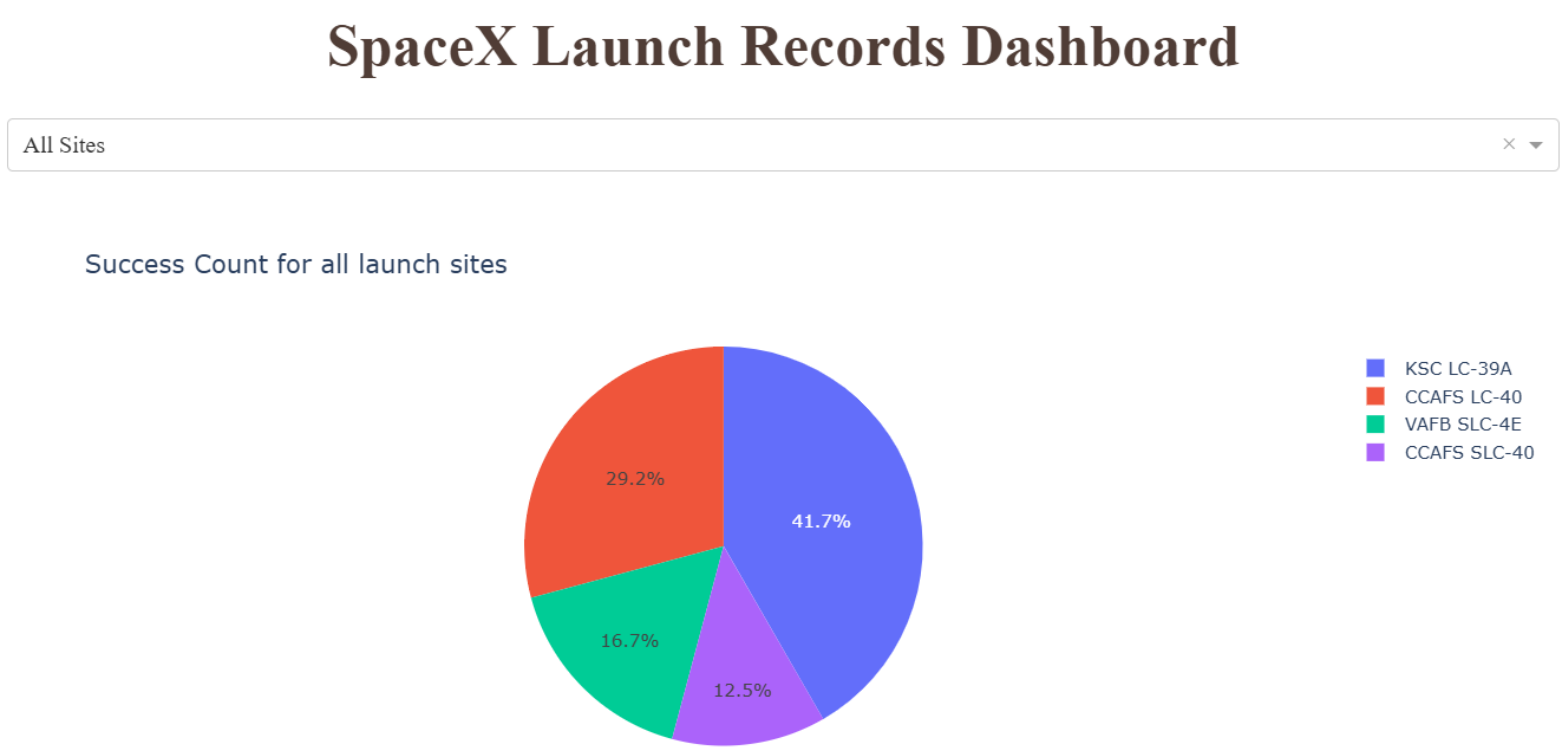
Launch Site Proximity

- Launch sites are close to railroads and coastlines.
- Launch sites aren't necessarily close to highways. (Contrary to what's observed in these images, the launch sites In California are not in close proximity with Highways)
- All Launch sites are considerably away from cities.



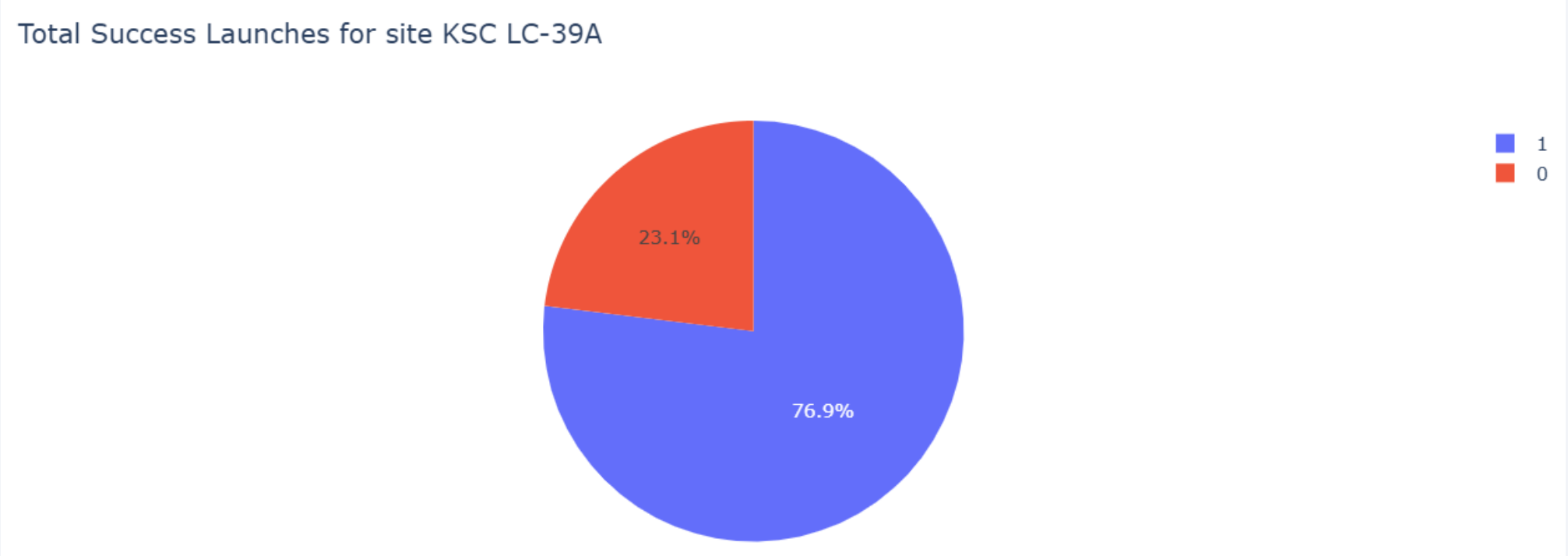

Section 4
Build a Dashboard with Plotly Dash

Total Success Launches for All Sites



Launch sites CCAFS SLC – 40 and KSC LC-39 have the lowest and highest success rates respectively.

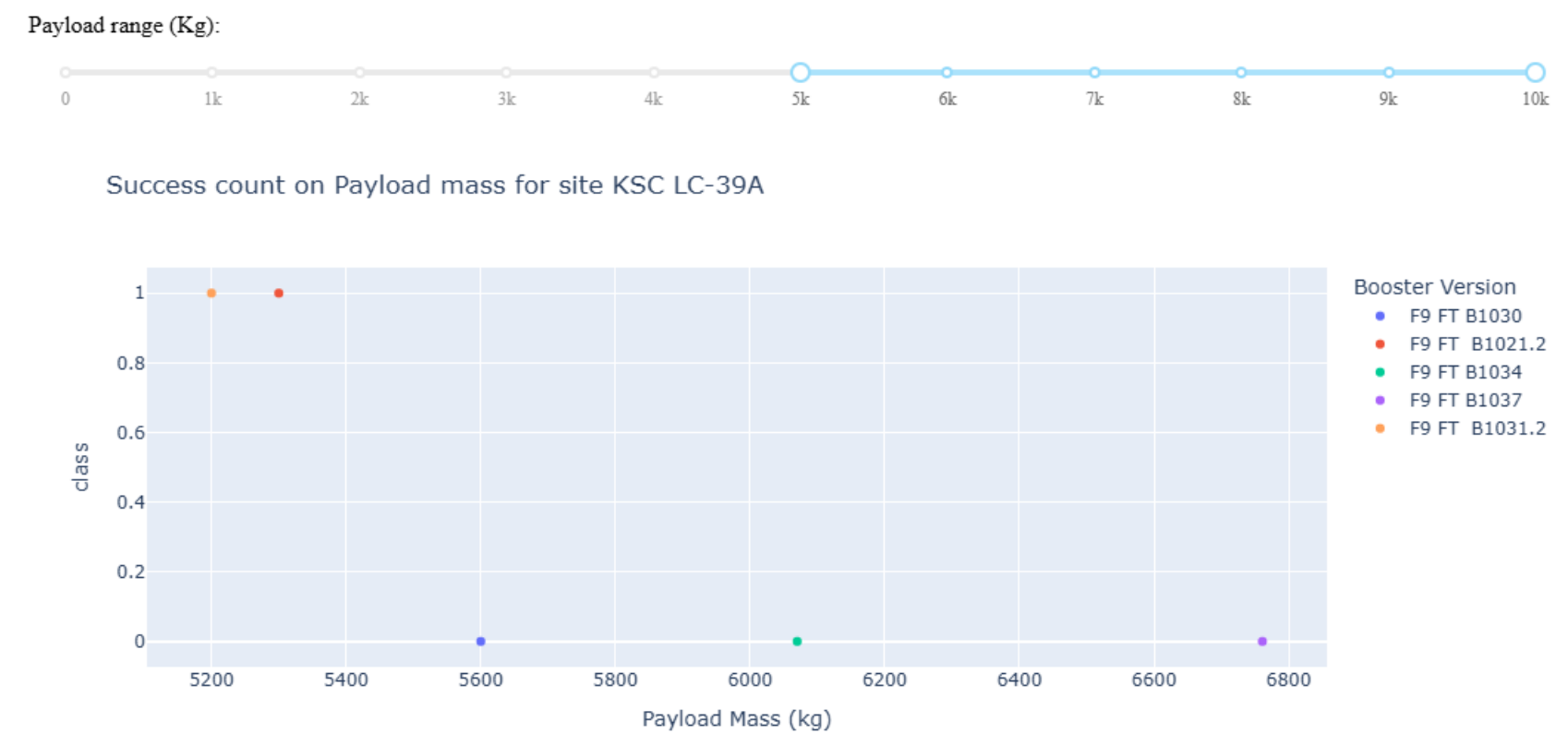
Total Launches for Site KSC LC-39A



This site has a success rate of more than 75%.

Payload Mass Vs Launch Outcome

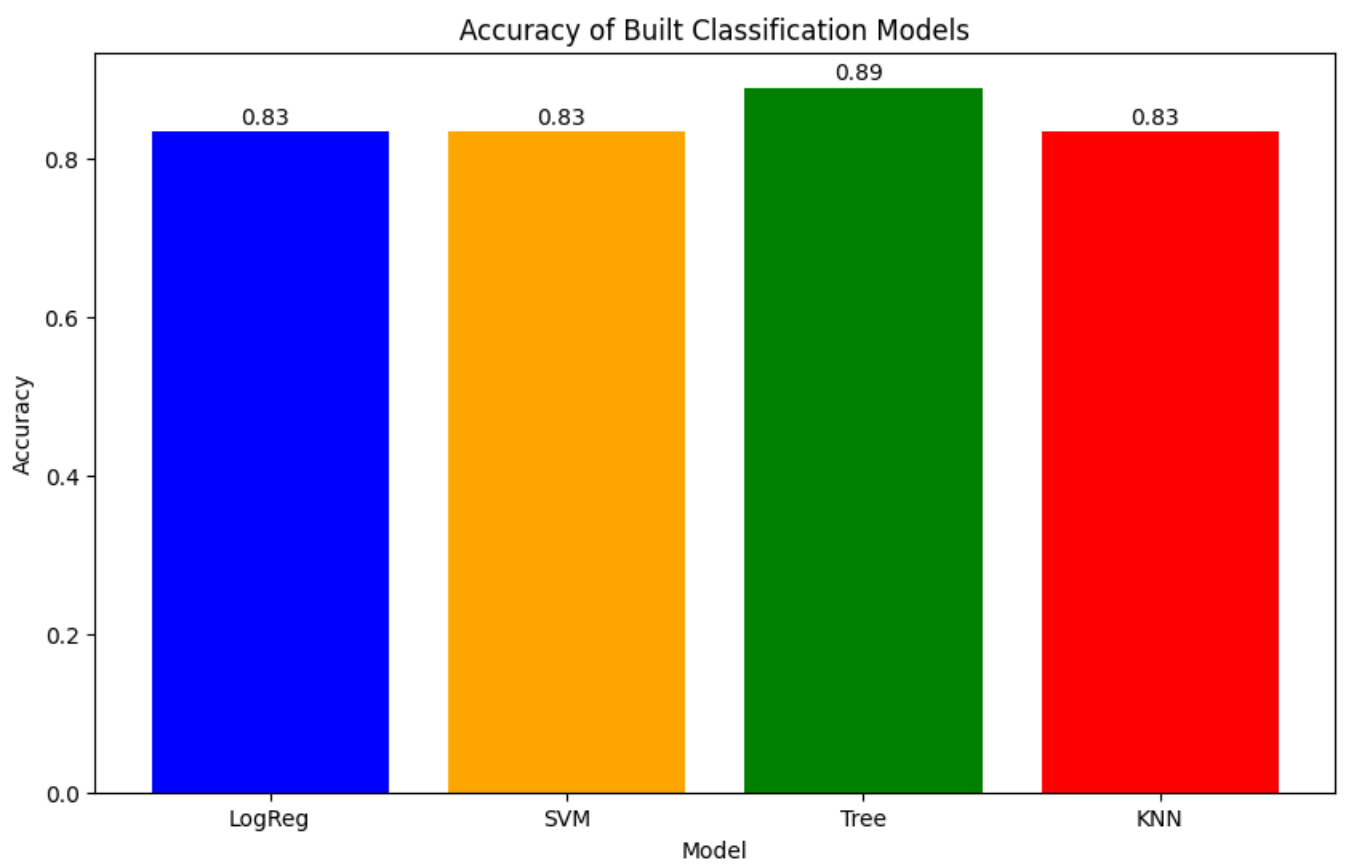
- Success rate is less with heavy payload masses.



Section 5

Predictive Analysis (Classification)

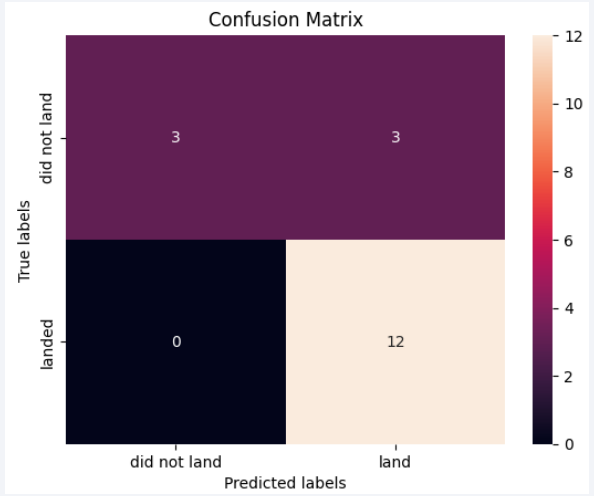
Classification Accuracy



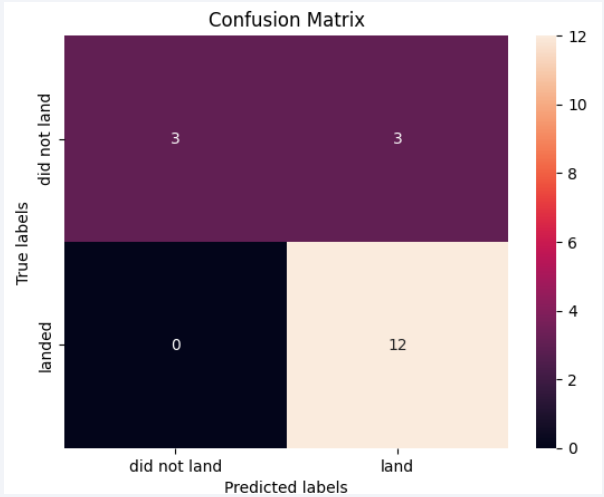
- All Models showed the same level of accuracy for test data.(~83%)
- Decision tree showed highest accuracy on Training data. (~89%)

Confusion Matrix

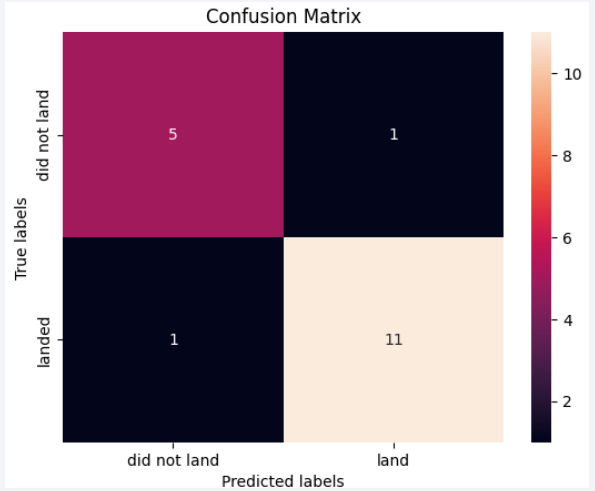
- Confusion Matrix for all Models is the same as they have the same accuracy



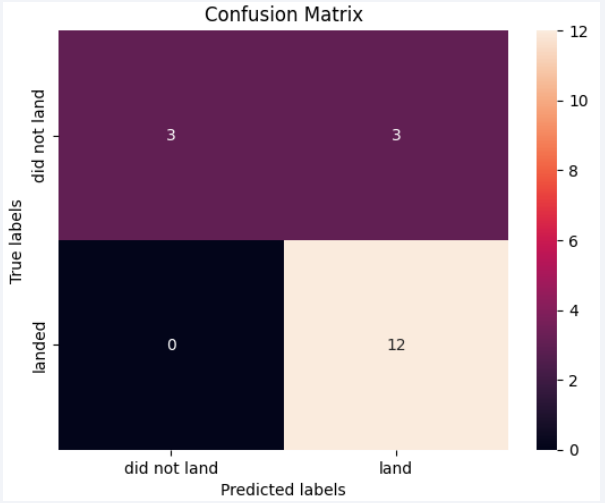
Logistics Reg



Support Vector machine



Decision Tree



K-Nearest Neighbors

Conclusions

- The KSC LC-39A launch site has the highest success rate among all launch sites.
- Launch sites are generally located close to coastlines, railroads, and highways, and are far from cities.
- The success rate of launches increases with the number of flights.
- The SSO orbit is the most favorable for a successful outcome, with all its launches being successful.
- The Decision Tree model performs best for our data in predictive analysis, achieving an accuracy of approximately 89%."

Thank you!