



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Vasudev Karanth
February 11, 2024



Outline

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

Executive Summary

SUMMARY OF METHODOLOGIES:

Step-1 | Data
Collection-
SpaceX Data
collected using:

SpaceX API

Web-scraping
from
Wikipedia

Step -2 | Data
Wrangling-with
focus on:

Exploratory
Data Analysis

Determining
Training
Labels

Step -3 |
Exploratory
Data Analysis-
using:

SQL

Pandas and
Matplotlib for
visualization

Step -4 |
Interactive
visual analytics:

Launch Sites
Locations
Analysis with
Folium

Interactive
Dashboard
with Plotly
Dash

Step -5 |
Machine
Learning
Prediction:

K-nearest
neighbors

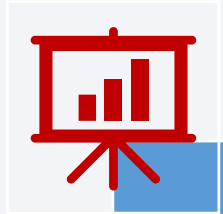
Decision Tree

Logistic
Regression

Support Vector
Machine

Executive Summary

SUMMARY OF ALL RESULTS:



Results of Exploratory Data Analysis:

- SQL
- Padas and Matplotlib



Results of Interactive Analytics using:

- **Folium:**
 - Visualizing launch sites on Map
 - Visualizing the success/failed launches for each launch site on Map
 - Visualizing the distances between a launch site to its proximities
- **Plotly Dash:**
 - Visualizing interactive dashboard showcasing a pie-chart and scatter plot of launch outcomes



Results of Predictive analysis

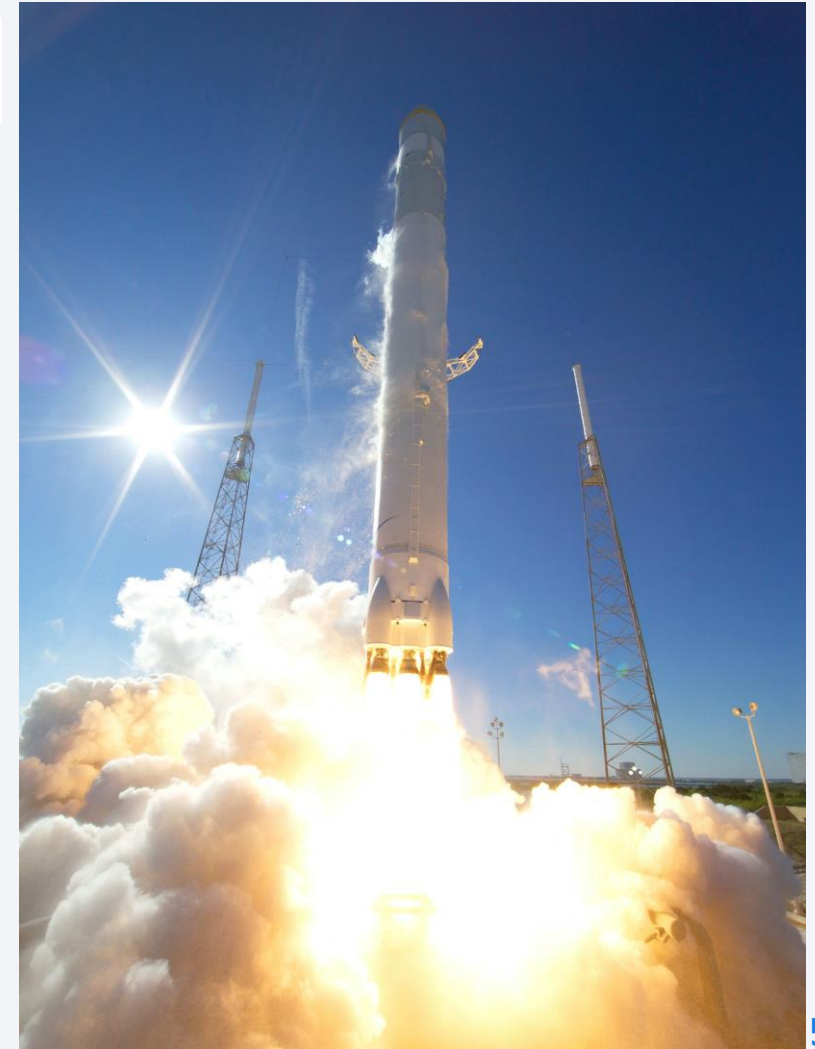
- Best Hyperparameters
- Best model
- Confusion Matrix for the best model

Introduction

Project background and context:

Background: Space X 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 Space X 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 space X for a rocket launch.

Purpose: The Scope of this project is to analyze various sources of data available on Falcon 9 rocket launches and create a Machine Learning pipeline to determine if the First Stage will land successfully or not.



Introduction

Problems you want to find answers:

- ☐ What factors determine the launch outcome?
- ☐ Is there any relation amongst these factors that affect the success of a rocket landing?
- ☐ What are the ideal conditions to ensure a successful landing of rocket?

Section 1

Methodology

Methodology

Executive Summary

- **Data collection methodology:**

- The Data pertaining to the Falcon rocket launches was obtained using the following:
 - Space X website using the SpaceX API (i.e., <https://api.spacexdata.com/v4/>) and
 - Web scraping from Wikipedia page titled 'List of Falcon 9 and Falcon Heavy launches' (i.e., https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches)

- **Perform data wrangling:**

- Data collected from the above mentioned sources were converted into a data frame.
- A new variable named 'landing_class' was created to represent the outcome with value '0' for unsuccessful and '1' for successful landings.
- The function 'get_dummies' was used to apply one-hot encoding to the categorical variables in the data frame.

Methodology

- **Perform exploratory data analysis (EDA) using visualization and SQL**
 - EDA was performed using Matplotlib visualization library and SQL.
 - The purpose of the exercise was to obtain valuable insights into the data for further analysis and model development.
- **Perform interactive visual analytics using Folium and Plotly Dash**

Folium:

Visualizing launch sites on Map.

Visualizing the success/failed launches for each launch site on Map.

Visualizing the distances between a launch site to its proximities.

Plotly Dash:

Visualizing interactive dashboard showcasing a pie-chart and scatter plot of launch outcomes.

Methodology

- **Perform predictive analysis using classification models**
 - The data was standardized using the 'StandardScaler' method of the sklearn pre-processing library.
 - Later, the data was split training and testing data.
 - Different classification models were fit and the best hyper parameter and the best model were evaluated.
 - Confusion matrix for each of the models were plot in the process.

Data Collection

Sources of Data:

Space X website using the SpaceX API (i.e., <https://api.spacexdata.com/v4/>) and

Web scraping from Wikipedia page titled 'List of Falcon 9 and Falcon Heavy launches' (i.e., https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches)

- **Space X API:**

- 'requests.get' was used to call the API.
- Next, the response content was decoded as a Json using 'json()' function call and turn it into a pandas dataframe using 'json_normalize' function.
- From the data frame, the data was filtered keeping only those variables that are essential for the analysis.
- The missing values in the dataset were identified and the mean values for the corresponding variable were calculated.
- The missing values were replaced with the said mean values.

- **Web scraping:**

- Web scraping was further performed using 'Beautiful Soup' to extract Falcon 9 launch records HTML table from Wikipedia.
- The table in the said result page was Parsed and converted it into a Pandas data frame for further analysis.

Data Collection – SpaceX API

- The 'requests' library was imported to make HTTP requests.
- The 'requests.get()' method was used to make the API calls and obtain the required data.
- The response was decoded as a Json using '.json()' method.
- Later the data was converted into a Pandas data frame using 'json_normalize()' function.

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
[71]: spacex_url="https://api.spacexdata.com/v4/launches/past"
[73]: response = requests.get(spacex_url)
```

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

Using the dataframe `data` print the first 5 rows

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

```
[78]: .....
.....
static_fire_date_utc static_fire_date_unix net window rocket succ
```

Data Collection – SpaceX API

- The data was later cleaned to include only the required features, filtered the data to include only the ‘Falcon 9 launches’, performed basic data wrangling to identify and fill missing values

- **Github Link:**

https://github.com/Vasudevkaranth/IBM_Datascience_Course_10_Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

```
[101]: # Hint data['BoosterVersion']!= 'Falcon 1'
data_falcon9 = data[data['BoosterVersion'] != 'Falcon 1']
data_falcon9
```

We can see below that some of the rows are missing values in our dataset.

```
[103]: data_falcon9.isnull().sum()
```

```
[115]: # Calculate the mean value of PayloadMass column
mean_PayloadMass = np.mean(data_falcon9['PayloadMass'])

# Replace the np.nan values with its mean value

data_falcon9['PayloadMass'].replace(np.nan, mean_PayloadMass, inplace=True)
data_falcon9.head(88)
```


Data Collection - Scraping

- Later web scraping was performed to collect Falcon 9 historical launch records from a Wikipedia page titled `List of Falcon 9 and Falcon Heavy launches`.
- The `HTTP get()` method was used to access the data, and a `BeautifulSoup` object was created from the HTML `response`.
- The ` ` elements were iterated on the said response to identify the column names. |
- An empty dictionary was later created, with the column names so identified and later a pandas data frame was created with the column names.

```
[55]: # use requests.get() method with the provided static_url
      # assign the response to a object

      data_new = requests.get(static_url)
```

Create a BeautifulSoup object from the HTML response

```
[56]: # Use BeautifulSoup() to create a BeautifulSoup object from a response
      soup = BeautifulSoup(data_new.text, "html.parser")
      soup
```

```
[56]: <!DOCTYPE html>
```

```
[60]: column_names = []

      element = first_launch_table.find_all('th')
      for row in range(len(element)):
          try:
              name = extract_column_from_header(element[row])
              if (name is not None and len(name) > 0):
                  column_names.append(name)
          except:
              pass
      column_names
```

Data Collection - Scraping

- Later, the html parsed through and the data frame with updated with the values thereon.
- The result was later saved as a 'CSV' file.
- **Github Link:**

https://github.com/Vasudevkaranth/IBM_Datascience_Course_10_Capstone/blob/main/jupyter-labs-webscraping.ipynb

```
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 dictionary
        if flag:
            extracted_row += 1
            # Flight Number value

            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            flight = launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            #print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
```

Data Wrangling

- The data saved earlier in the CSV format was loaded on the jupyter notebooks for the purpose of data wrangling.
- During the process,
 - Number of missing values were identified for each attribute.
 - The data type for each attribute was identified using the '.dtypes' method.
- Number of launches at each launch site were determined

```
df.isnull().sum()/len(df)*100
```

```
FlightNumber      0.000000  
,Date             0.000000  
,BoosterVersion   0.000000  
,PayloadMass      0.000000  
,Orbit            0.000000  
,LaunchSite       0.000000  
,Outcome          0.000000  
,Flights          0.000000  
,GridFins         0.000000  
,Reused           0.000000  
,Legs             0.000000  
,LandingPad       28.888889  
,Block            0.000000  
,ReusedCount      0.000000  
,Serial           0.000000  
,Longitude        0.000000  
,Latitude         0.000000  
,dtype: float64
```

```
[6]: df.dtypes
```

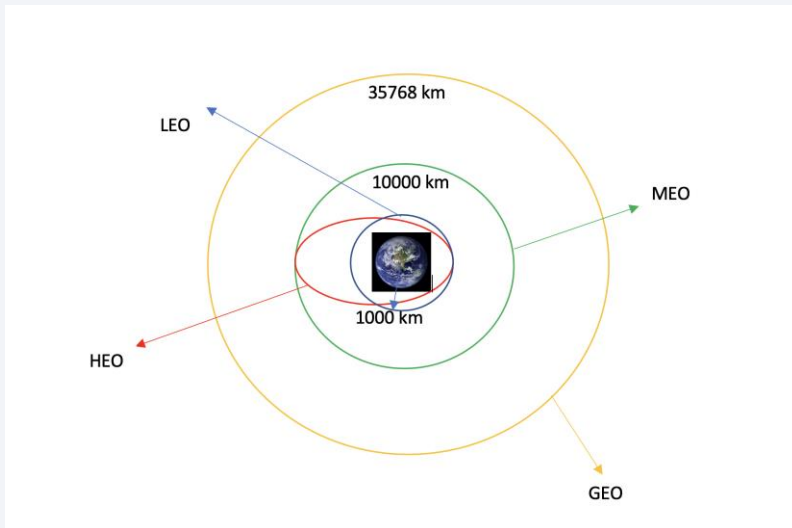
```
[6]: FlightNumber      int64  
,Date              object  
,BoosterVersion     object  
,PayloadMass        float64  
,Orbit              object  
,LaunchSite         object  
,Outcome            object  
,Flights            int64  
,GridFins           bool  
,Reused             bool  
,Legs               bool  
,LandingPad         object  
,Block              float64  
,ReusedCount        int64  
,Serial             object  
,Longitude          float64  
,Latitude           float64  
,dtype: object
```

```
# Apply value_counts() on column LaunchSite  
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40      55  
,KSC LC 39A       22  
,VAFB SLC 4E      13  
,Name: LaunchSite, dtype: int64
```

Data Wrangling

- Number of launches to each orbits and number of occurrences were determined



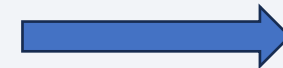
```
[8]: # Apply value_counts on Orbit column  
df['Orbit'].value_counts()
```

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

```
landing_outcomes = df['Outcome'].value_counts()  
landing_outcomes
```

```
True ASDS      41  
,None None     19  
,True RTLS     14  
,False ASDS     6  
,True Ocean     5  
,False Ocean    2  
,None ASDS      2  
,False RTLS     1  
,Name: Outcome, dtype: int64
```

- Subsequently, the data was processed further by creating a variable 'Class' with value 0 representing unsuccessful landing and 1 representing successful landing




Class	
0	0
1	0
2	0
3	0
4	0

Github Link:

https://github.com/Vasudevkaranth/IBM_Datascience_Course_10_Capstone/blob/main/abs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

1. The scope of EDA with visualization was to identify the variables that would affect the outcome.
2. 'Scatter Plot' was used for the purpose of visualization as it was the *most effective visualization tool to visualize the relationship between 2 variables*.
3. Accordingly, the following plots were visualized for analysis:
 - i. Flight Number vs. Payload Mass
 - ii. Flight Number vs. Launch Site
 - iii. Payload Mass vs. Launch Site
 - iv. Flight Number vs. Orbit
 - v. Payload vs. Orbit

EDA with Data Visualization

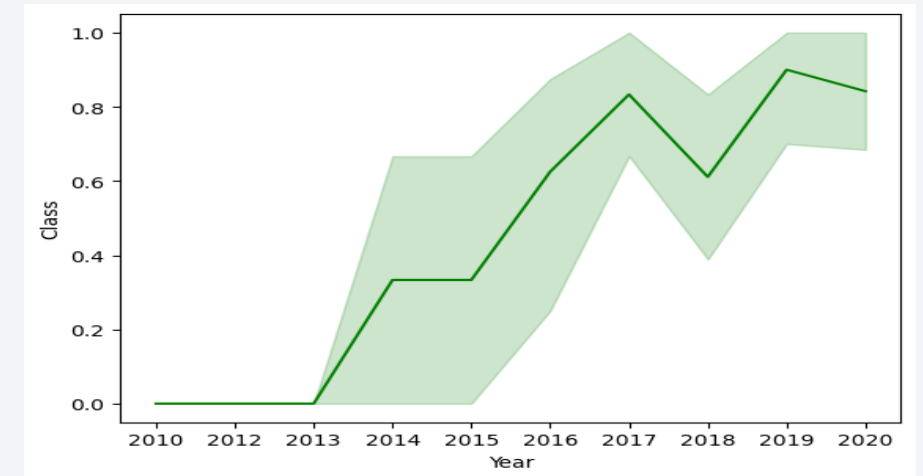
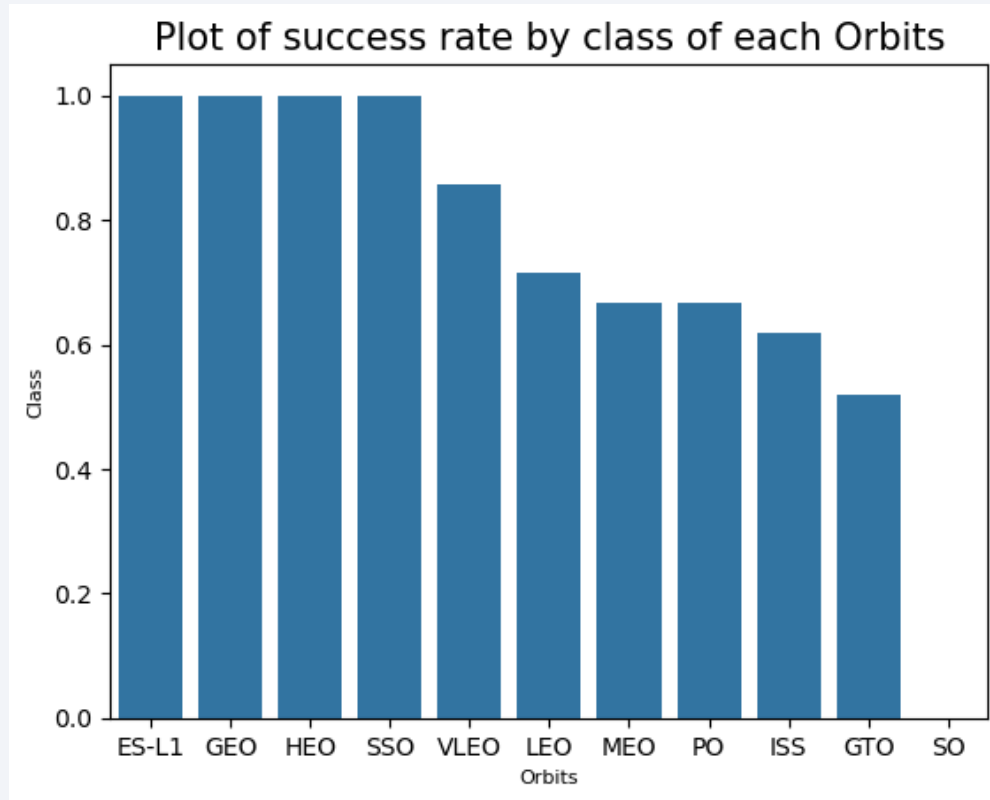
- Based on the analysis of scatter plots, a data frame referred to as ‘features’ was created.
- This data frame contained those attributes that were identified for further analysis and prediction.

```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins',  
              'Reused', 'Legs', 'LandingPad', 'Block', 'ReusedCount', 'Serial']]  
features.head()
```

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0007
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

EDA with Data Visualization

- A Bar chart depicting success rate for each orbits was created.
- Also, a line chart for yearly success rates since 2013 was depicted to visualize yearly trend as follows



- **Github Link:**

https://github.com/Vasudevkaranth/IBM_Data_science_Course_10_Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- To query the SpaceX dataset, 'SQLAlchemy' a SQL toolkit for the Python programming language was used.
- Using the SQL extension, connection to the sqlite database was established as follows:



```
[2]: %load_ext sql

[4]: import csv, sqlite3

    con = sqlite3.connect("my_data1.db")
    cur = con.cursor()

[6]: !pip install -q pandas==1.1.5

[7]: %sql sqlite:///my_data1.db

[7]: 'Connected: @my_data1.db'
```

- Later, a new table 'SPACEXTABLE' was created after filtering the data to remove the rows with blank values using the following SQL query:



```
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null
%sql select * from SPACEXTABLE

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

EDA with SQL

- Later, the following SQL queries were executed for the purpose of analysis of data. The following were found out in the process:
 - Names of unique launch sites in the space mission.
 - Total payload mass carried by boosters launched by NASA (CRS).
 - Average payload mass carried by booster version F9 v1.1.
 - Date when the first successful landing outcome in ground pad was achieved.
 - The names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
 - The total number of successful and failure mission outcomes.
- **Github Link:**
https://github.com/Vasudevkaranth/IBM_Datascience_Course_10_Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqlite.ipynb

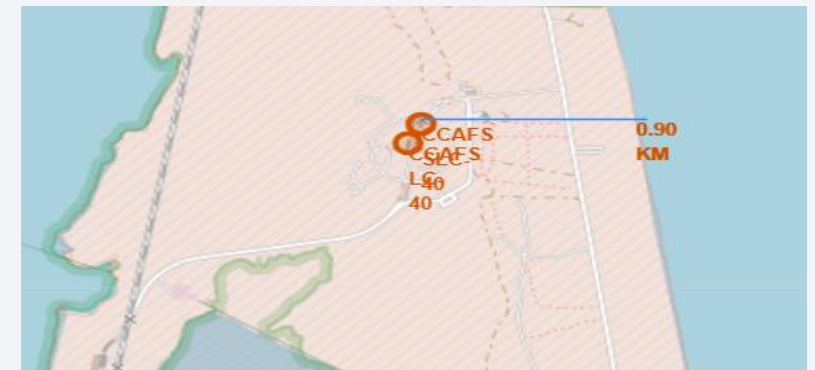
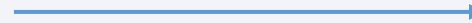
Build an Interactive Map with Folium

- **The objects of building an interactive Map with folium were as follows:**

- Marking the launch Sites on the Map with markers, circles.



- Calculating the distance between a launch site to its proximities such as city, highway, railway, airport using lines



- Marking the success/failed launches for each site on the map with the help of color labelled marker clusters.



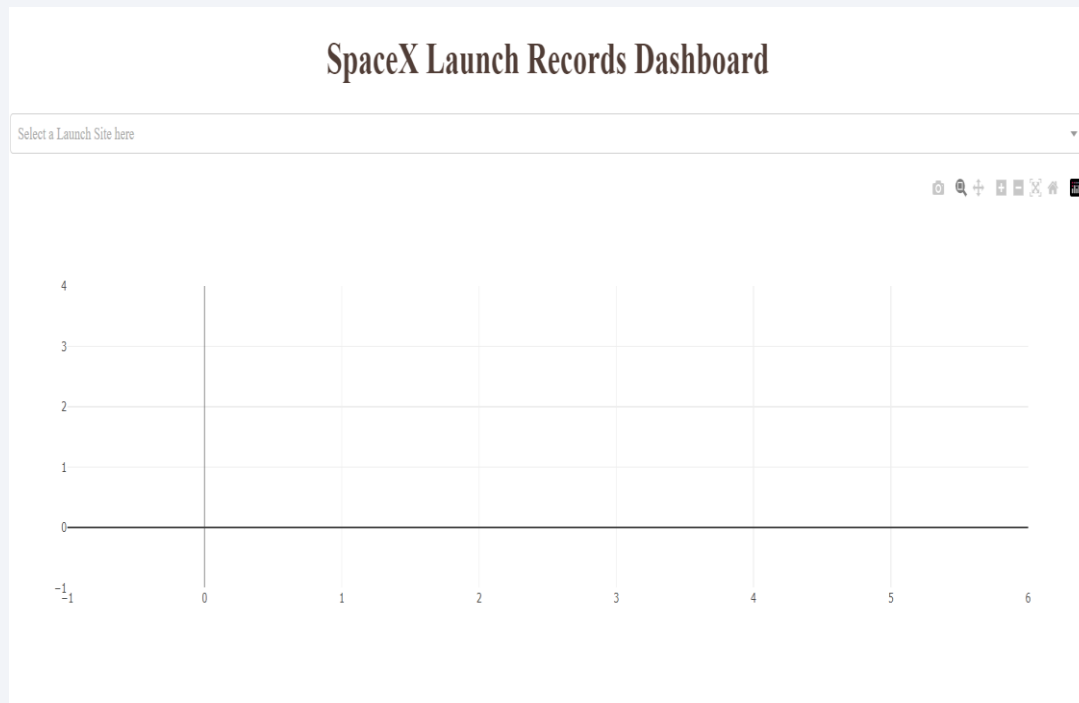
Build an Interactive Map with Folium

- The launch sites were marked on a map to determine its' geographic location.
- Since each launch site had multiple launches, marker cluster (i.e., 'MarkerCluster()') was used to simplify the visualization.
- Using the values in attribute 'Class' as the base, a new attribute 'marker_color' was created and later this was assigned to the marker cluster. Red color was assigned to unsuccessful landings and green color was assigned to the successful landings.
- Later, 'MousePosition' method was used on the map to get the coordinates for a mouse over a point on the map;
- With the help of coordinates, distance to the proximities such as distance to coastline, cities, highways etc., were calculated and relevant questions were answered.
- **Github Link:**

https://github.com/Vasudevkaranth/IBM_Datascience_Course_10_Capstone/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- An interactive Dashboard was created using 'Plotly Dash'.
- Skeleton of the dash application is as follows:



- A drop-down was created to select the launch site for the graphs.
- User has the choice to select all or any of the launch sites from the drop-down



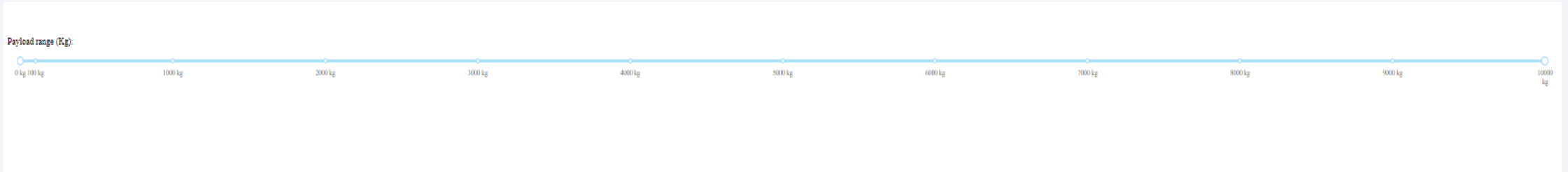
SpaceX

Select a Launch Site here

- All Sites
- CCAFS LC-40
- VAFB SLC-4E
- KSC LC-39A
- CCAFS SLC-40

Build a Dashboard with Plotly Dash

- A slider was further also added to fine tune the results, based on the payload mass with range from 0 to 10,000 Kgs,

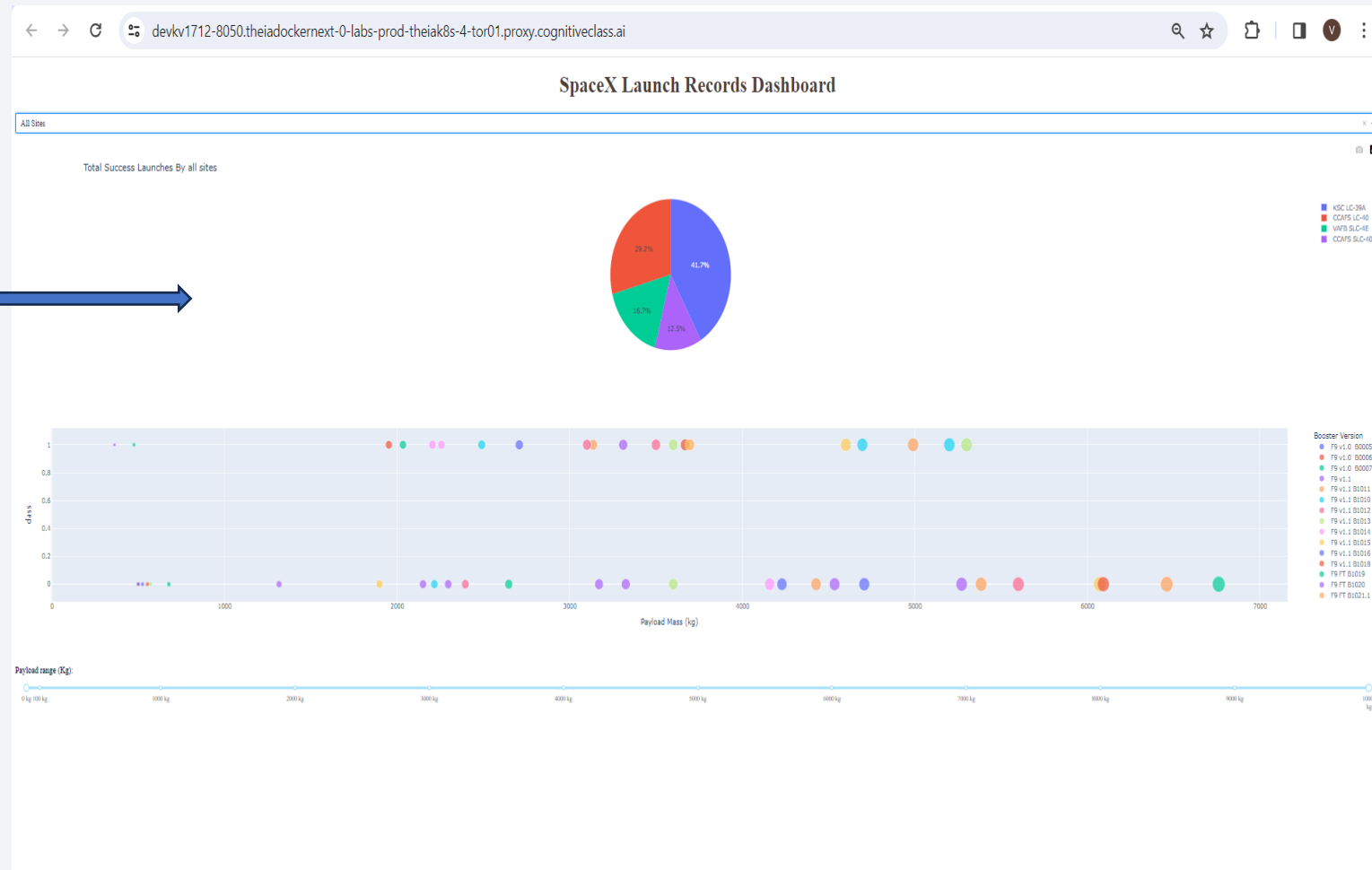


- **Github Link:**

https://github.com/Vasudevkaranth/IBM_Datascience_Course_10_Capstone/blob/main/spacex_dash_app.py

Build a Dashboard with Plotly Dash

- The ‘app.callback’ operator to perform interactive dash call backs, which would take ‘launch site’ from the dropdown and payload mass from the slider as input and output the charts mentioned below:
- A ‘pie-chart’ visualizing the successful launches based on the launch site selected from the dropdown.
- A Scatter Plot visualizing the outcomes, based on the selected payload range.

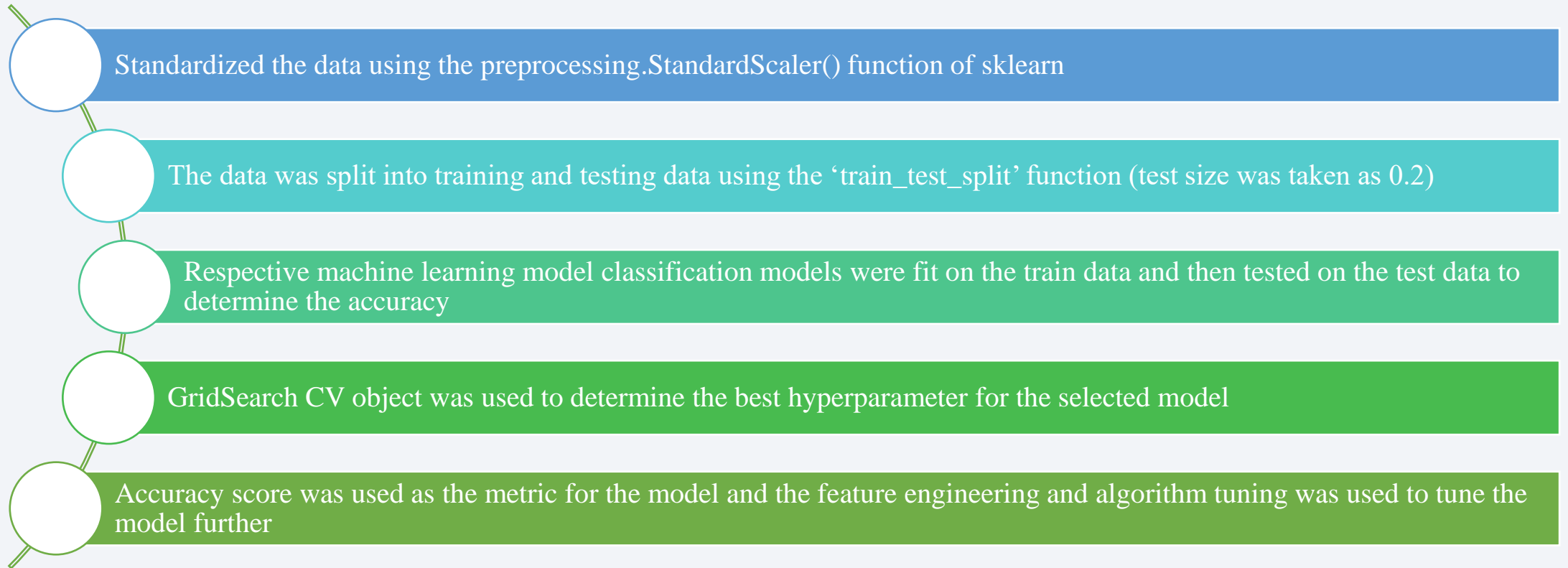


Predictive Analysis (Classification)

- The purpose of the analysis was to determine if a rocket launch by SpaceX would be successful or not depending on factors such as payload mass, number of flights etc.
- As the outcomes of analysis would be to determine if a landing was ‘Successful’ or ‘Unsuccessful’, the classification models were used to perform the analysis.
- The following classification models were used to perform the analysis:
 - Decision Trees
 - SVM
 - Logistic Regression and
 - KNN

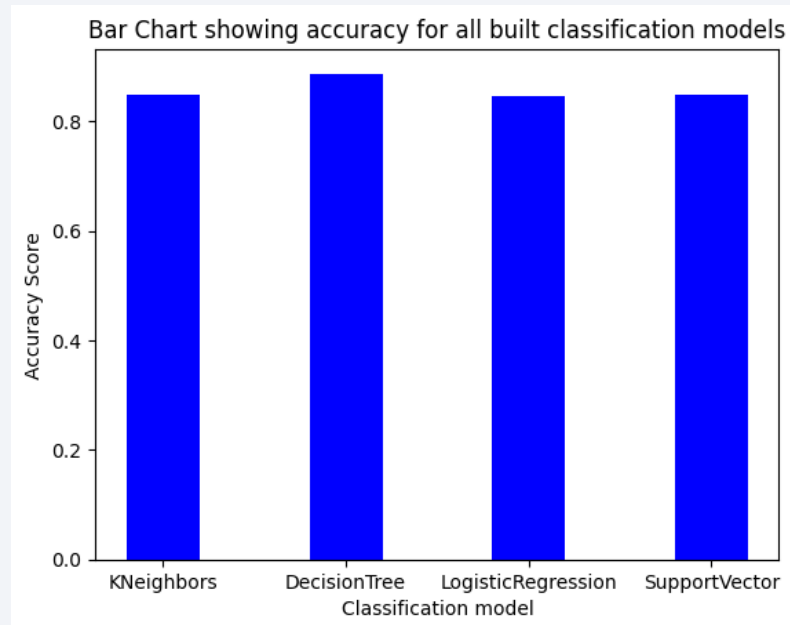
Predictive Analysis (Classification)

- A pipeline was created to predict the landing outcome in the following manner:



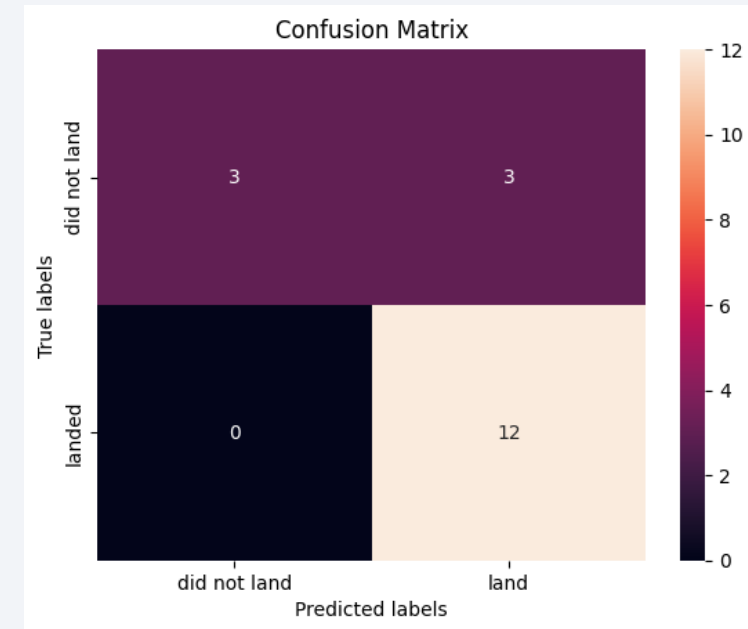
Predictive Analysis (Classification)

- The best model for the launch data was Decision Tree, with a classification score of 0.8875



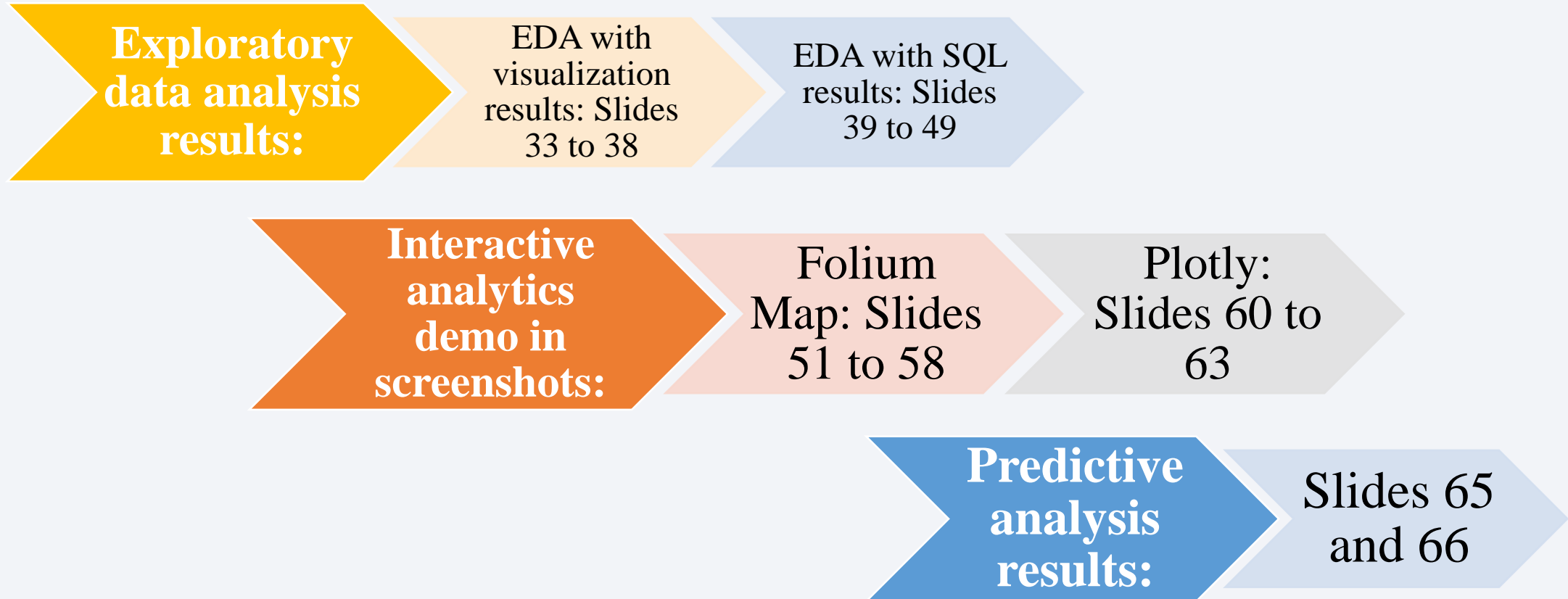
Github Link:

https://github.com/Vasudevkaranth/IBM_Datascience_Course_10_Capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb



- The confusion matrix of the said model is given in the above diagram

Results: Slides need updating

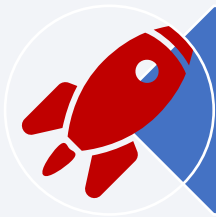
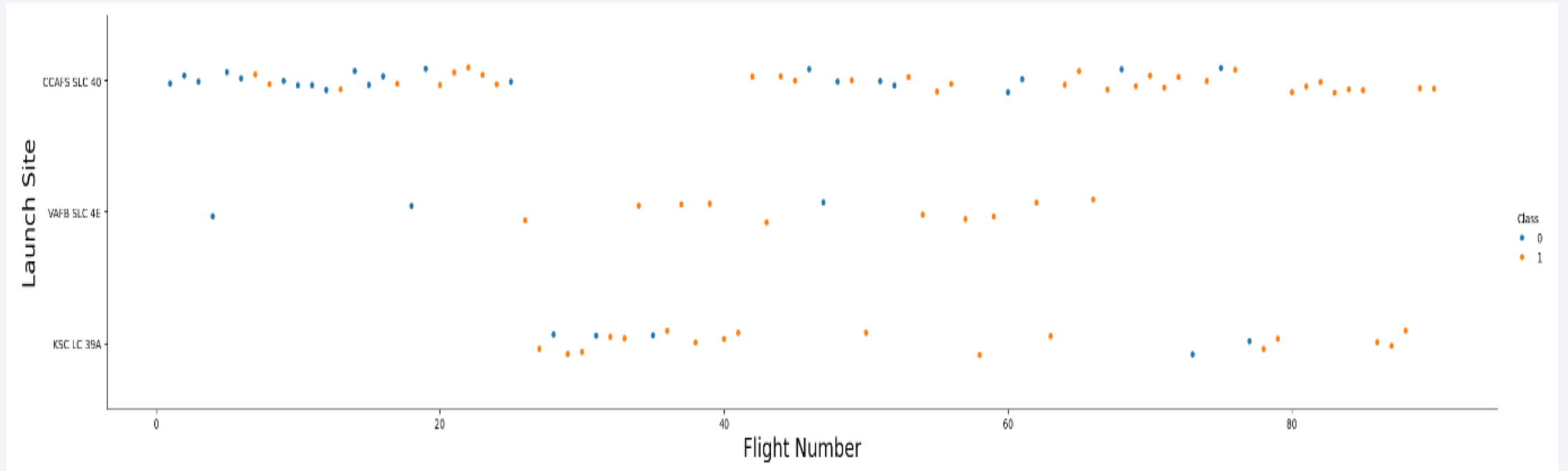


The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower half of the image. The overall effect is dynamic and technological.

Section 2

Insights drawn from EDA

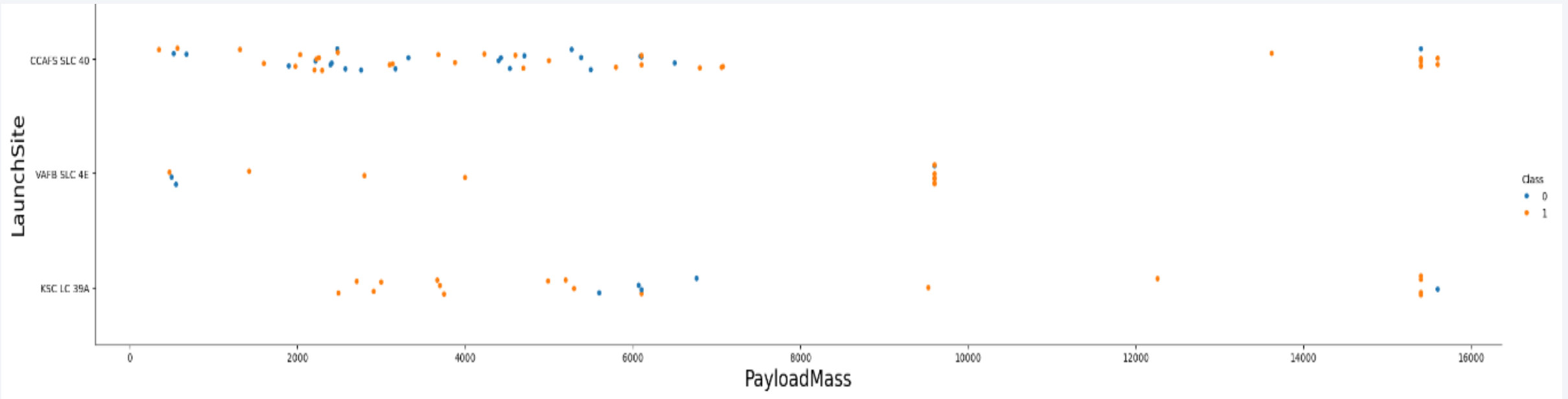
Flight Number vs. Launch Site



Analysis:

- Larger the number of flights at a launch site, greater is the success rate.

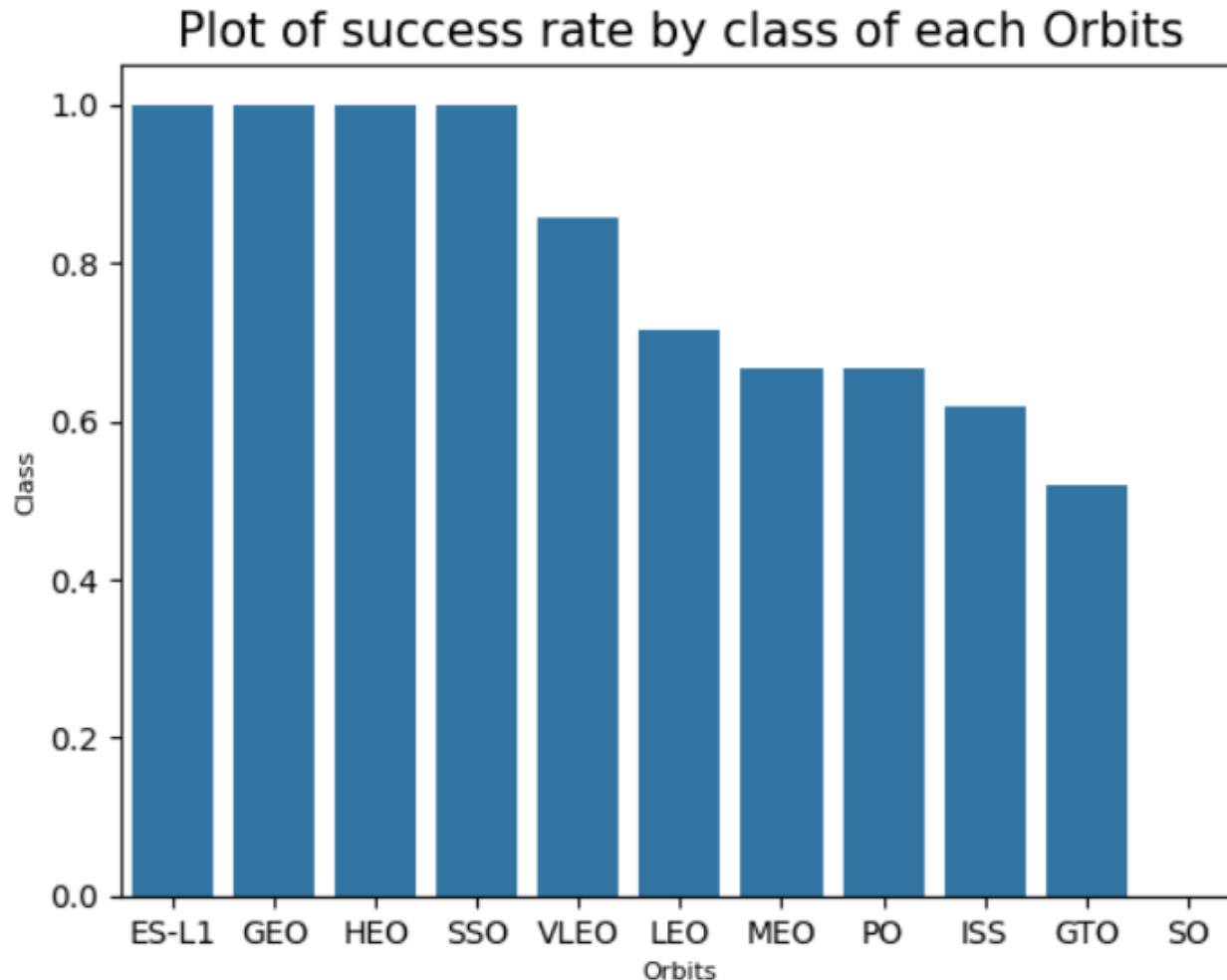
Payload vs. Launch Site



Analysis:

- Larger the payload, greater is the success rate.
- With a few exceptions, most of the launches with payload mass exceeding 10,000 Kg have been successful.

Success Rate vs. Orbit Type



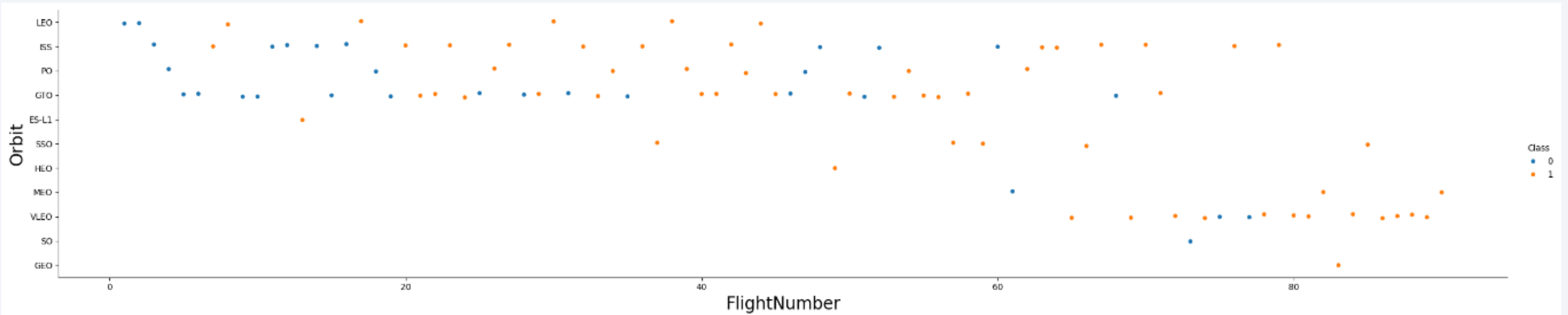
Analysis:

Launches to ES-L1, GEO, HEO and SSO orbits had the most success rates.

Launches to VLEO, LEO, MEO and PO orbits had comparatively lower success rates.

Launches to ISS, GTO and SO had the least success rates.

Flight Number vs. Orbit Type



Analysis:

from the above plot, it can be concluded that **the relationship between the flight number and orbit is ambiguous.**

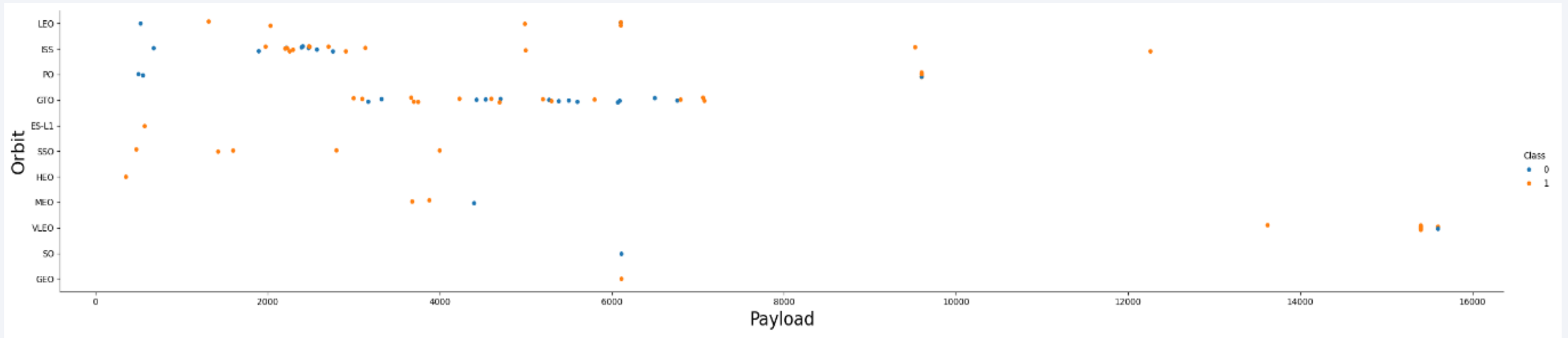


In case of LEO orbit, it appears that the success rate depends on the number of flights. However, the same cannot be said for the other orbits such as the GTO orbit.



Therefore, it is not easy to distinguish the relationship between the flight numbers and orbit for other orbits.

Payload vs. Orbit Type



Analysis:

the LEO, ISS and PO orbits have more successful landings for heavy payloads.

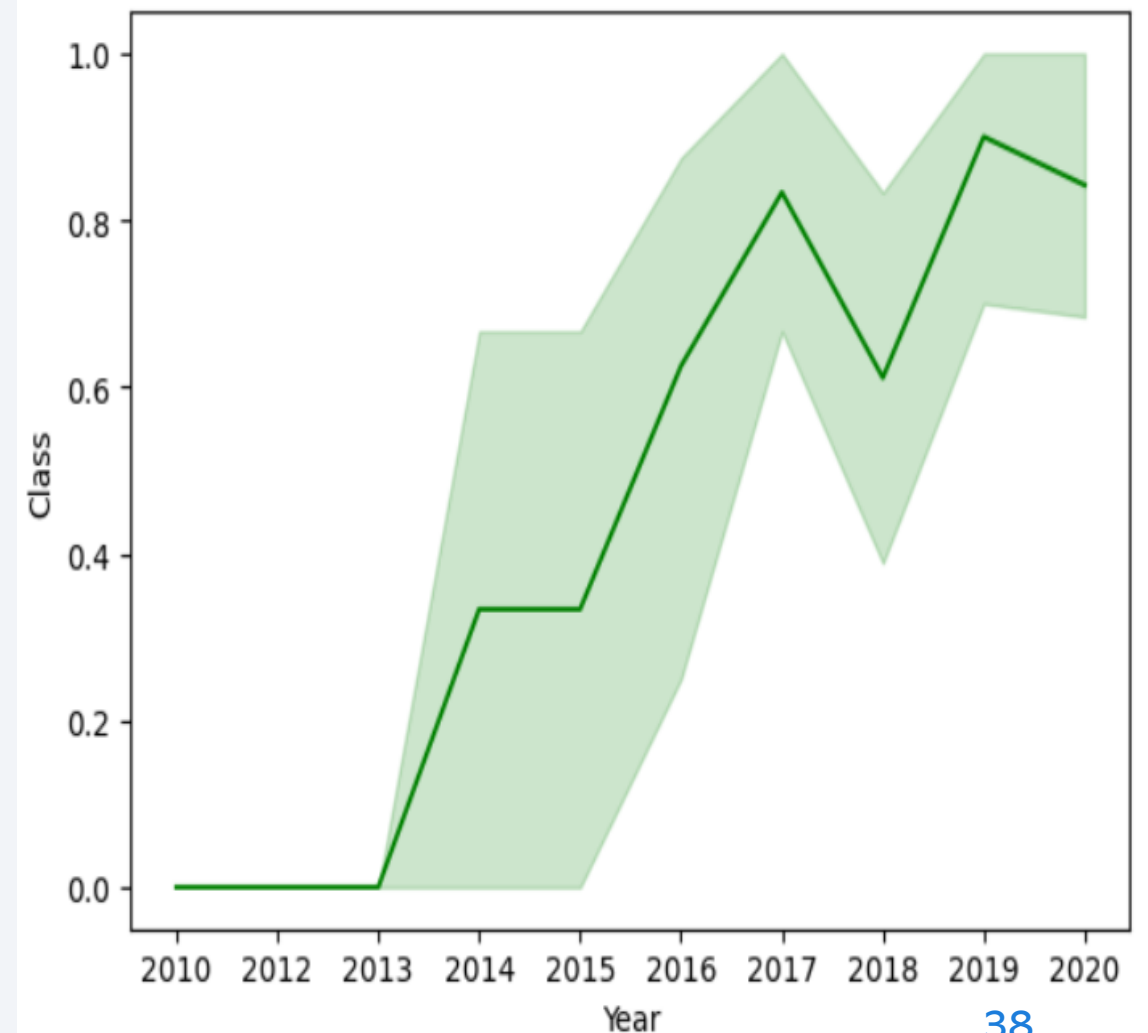
However, similar pattern cannot be determined for other orbits on account of the following reasons:

- There are both observations (i.e., successful and unsuccessful) for each class of payloads in certain orbits such as GTO, VLEO etc.
- Some orbits such as SSO, do not have observations beyond certain payload.

Launch Success Yearly Trend


From the Line plot, an upward trend is observed in the success rates till the year 2020, with a downward movement in the year 2018 (i.e., success rates have been improving since 2013).

Year 2018 may require further analysis to determine the reasons for the downward movement




All Launch Site Names

At the beginning, a SQL query was executed to identify the names of launch sites.



The 'SELECT DISTINCT' statement was used to get the desired result.



The query yielded the following four launch site names:

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

```
[20]: %sql select Distinct "Launch_Site" from SPACEXTBL
      * sqlite:///my_data1.db
```

Done.

```
[20]: .....
```

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

Launch Site Names Begin with 'CCA'

To get the 5 records where launch sites begin with 'CCA', the following SQL query was used.

The 'LIKE' operator was used in the 'WHERE' clause to search for a specified pattern (i.e., launch sites beginning with name 'CCA') to get the desired results.

Display 5 records where launch sites begin with the string '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

Total Payload Mass

To calculate the total payload carried by boosters from NASA, the query mentioned below was used.



The 'SUM()' function was used to calculate the total payload mass.



The 'WHERE' clause specified the specific site (i.e., NASA (CRS)) as criteria as to filter the results.



The query yielded the result of 45596.

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
```

Average Payload Mass by F9 v1.1



To calculate the average payload mass by carried by booster version F9 v1.1, the following query was used.

The 'AVG()' function was used to find the average payload mass.

The 'WHERE' clause specified the specific booster (i.e., F9 v1.1) as criteria to filter the results.

The query yielded the result 2928.4.

The result was saved as 'Average_payload_mass(KG)_Booster F9 v1.1' for easy reference.

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

```
sql select AVG("PAYLOAD_MASS_KG_") as "Average_payload_mass(KG)_Booster F9 v1.1"  
from SPACEXTABLE where "Booster_Version" == "F9 v1.1"
```

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

Done.

.....
Average_payload_mass(KG)_Booster F9 v1.1)

2928.4

First Successful Ground Landing Date

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

Hint: Use min function

```
%sql select min("Date") as "Date_of_First_Successful_Landing_Ground_pad" from SPACEXTABLE
where "Landing_Outcome" == "Success (ground pad)"
```

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

Done.

```
.....
```

Date_of_First_Successful_Landing_Ground_pad

2015-12-22

To calculate the first successful ground launch date, the following query was used.

The MIN() function was used to find the earliest date for the query.

The 'WHERE' clause specified the specific outcome (i.e., 'Success (ground pad)') as criteria as to filter the results

The query yielded the date 2015-12-22 as the result..

The result was saved as 'Date_of_First_Successful_Landing_Ground_Pad' for easy reference

Successful Drone Ship Landing with Payload between 4000 and 6000

To find the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000, following query was used.



The 'SELECT DISTINCT' statement was used on the attribute 'Booster Version' to identify unique booster Versions.



The 'WHERE' clause specified the specific outcome (i.e., 'Success (drone ship)') and payload range between 4000 and 6000 to get the desired result.

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql select distinct "Booster_Version" from SPACEXTABLE  
where "Landing_Outcome" == "Success (drone ship)"  
and "PAYLOAD_MASS_KG_" between 4000 and 6000
```

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

```
Done.
```

```
//////////
```

Booster_Version

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql select "Mission_Outcome", count("Mission_Outcome") from SPACEXTABLE group by "Mission_Outcome"
```

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

Done.

.....

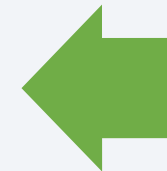
Mission_Outcome	count("Mission_Outcome")
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

To calculate the total number of successful and failure mission outcomes, the following query was used.



The COUNT() function was used on the 'Mission_Outcome' attribute to find number of results in each category.

The 'GROUP BY' function was used to group all the observations based on criteria specified (i.e., mission outcomes).



Boosters Carried Maximum Payload

To find the names of the boosters that have carried the maximum payload mass, the following query was used.

The 'Booster_Version' and 'Payload_Mass_(KG)' attributes were chosen.

As required, a sub-query was used to determine the maximum payload mass in the 'Where' clause.

The 'MAX()' function was applied on the pay load attribute to determine the maximum payload.

The query yielded a result containing the booster versions carrying payload of 15600 Kgs, being the maximum payload.

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

```
%sql select "Booster_Version", "PAYLOAD_MASS_KG_" as "Payload" from SPACEXTABLE
where PAYLOAD_MASS_KG_ == (select max("PAYLOAD_MASS_KG_") from SPACEXTABLE)
```

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

```
Done.
```

```
.....
```

Booster_Version	Payload
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

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

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

```
Done.
```

```
.....
```

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

To find the List of failed landing_outcomes in drone ship, their booster versions, and launch site names for the year 2015, the following query was used.

Since the SQLite did not support the month names. Therefore, 'SUBSTR()' function was used to get the month and date.

The 'WHERE' clause was used as to specify the specific criteria for querying the results

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

To find 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, the query in the subsequent slide was used.



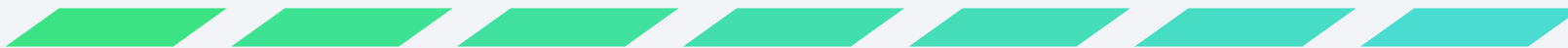
The 'COUNT()' function was used to get the count of distinct outcomes.



The 'WHERE' clause was used as to specify the date range for querying the results.



The 'GROUPBY()' function was used to group the outcomes.



The COUNT() DESC was used to filter the data in the descending order.



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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
|: %sql select "Landing_Outcome", count("Landing_Outcome") as "Count_landing_outcomes" 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.

```
|: .....
```

Landing_Outcome	Count_landing_outcomes
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

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

Launch Sites Proximities Analysis

Locating the Space X launch centers on a Folium Map

Folium map was used to do the following:

- Locate the launch sites on a map using site's latitude and longitude coordinates;
- A circle was added to each launch site using the `folium.Circle` object and marked using the `folium.Marker`;
- Add.child() function was used to add these launch sites on the global map;

```
[6]: # Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

```
[6]:
```

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

```
: # Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition
for index, site in launch_sites_df.iterrows():
    circle = folium.Circle([site['Lat'], site['Long']], color='#d35300', radius=50,
                           fill=True).add_child(folium.Popup(site['Launch Site']))

    marker = folium.Marker(
        [site['Lat'], site['Long']],
        # Create an icon as a text label
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % site['Launch Site'],
        )
    )
    site_map.add_child(circle)
    site_map.add_child(marker)
```

site map

Locating the Space X launch centers on a Folium Map



Locating the Space X launch centers on a Folium Map



Launch
Sites
zoomed in



Analysis:

It is observed from the folium map that each of the launch site are located in the United States of America and in the proximity to the coast.

The CCAFS LC-40, CCAFS SLC-40 and KSC LC-39A launch stations are located near the west coast in Florida.

The VAFB SLC-4E launch station is on the east coast in California.

Color-labeled launch outcomes on the map

The folium map was further enhanced by adding the color-labeled launch outcomes on the map

At first, colors were assigned to identify if a landing was 'successful' or not 'successful' using the function in the slide.

Red color was assigned to unsuccessful and Green color was assigned to successful landing.

Later, MarkerCluster() object was added to the folium map to specify the outcomes to the landing outcomes at a landing site.

The exercise enabled to identify which launch sites have relatively high success rates.

```
def assign_marker_color(launch_outcome):  
    if launch_outcome == 1:  
        return 'green'  
    else:  
        return 'red'  
  
spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)  
spacex_df.tail(10)
```

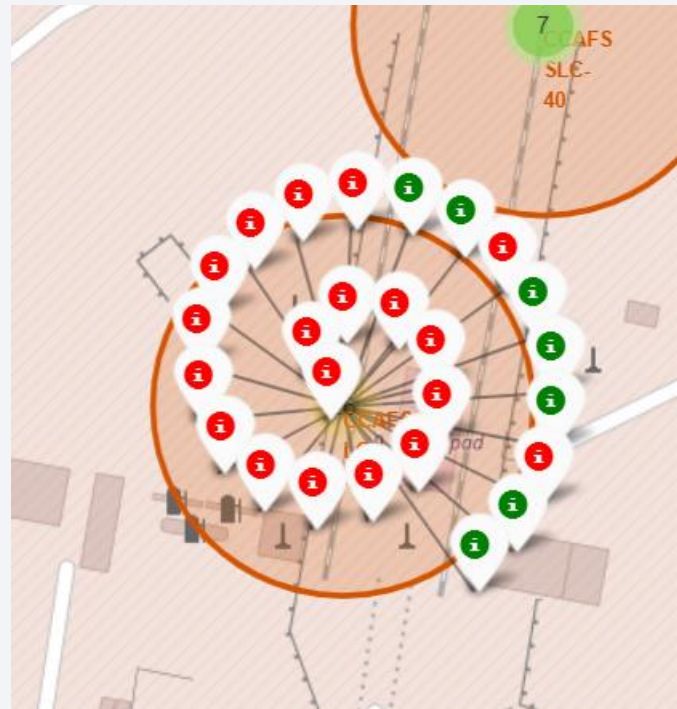
```
]:
```

	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green

Color-labeled launch outcomes on the map

- Results of Launch Sites in Florida

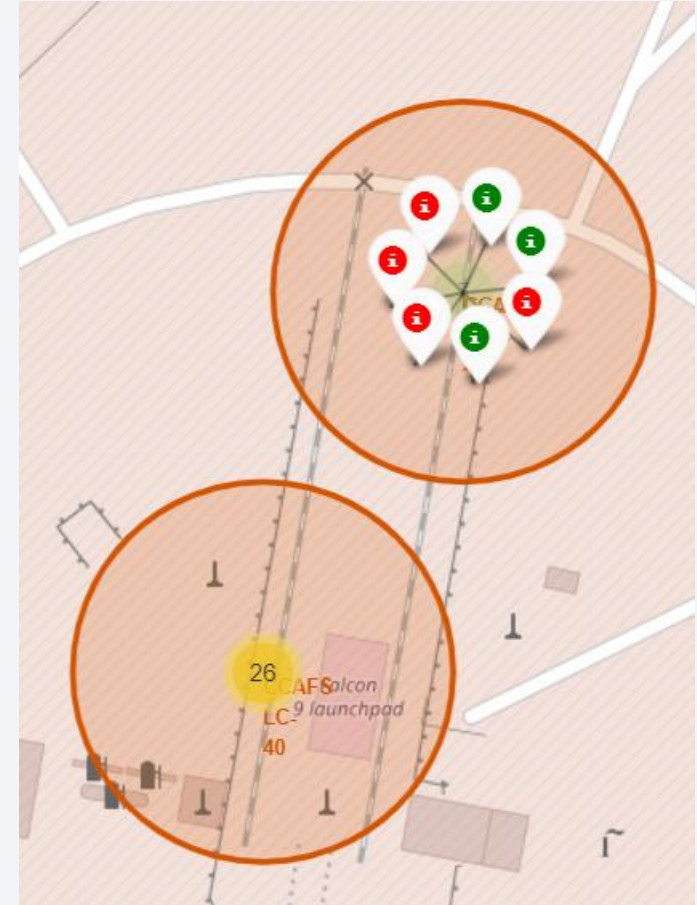
- CCAFS LC-40



- KSC LC-39A

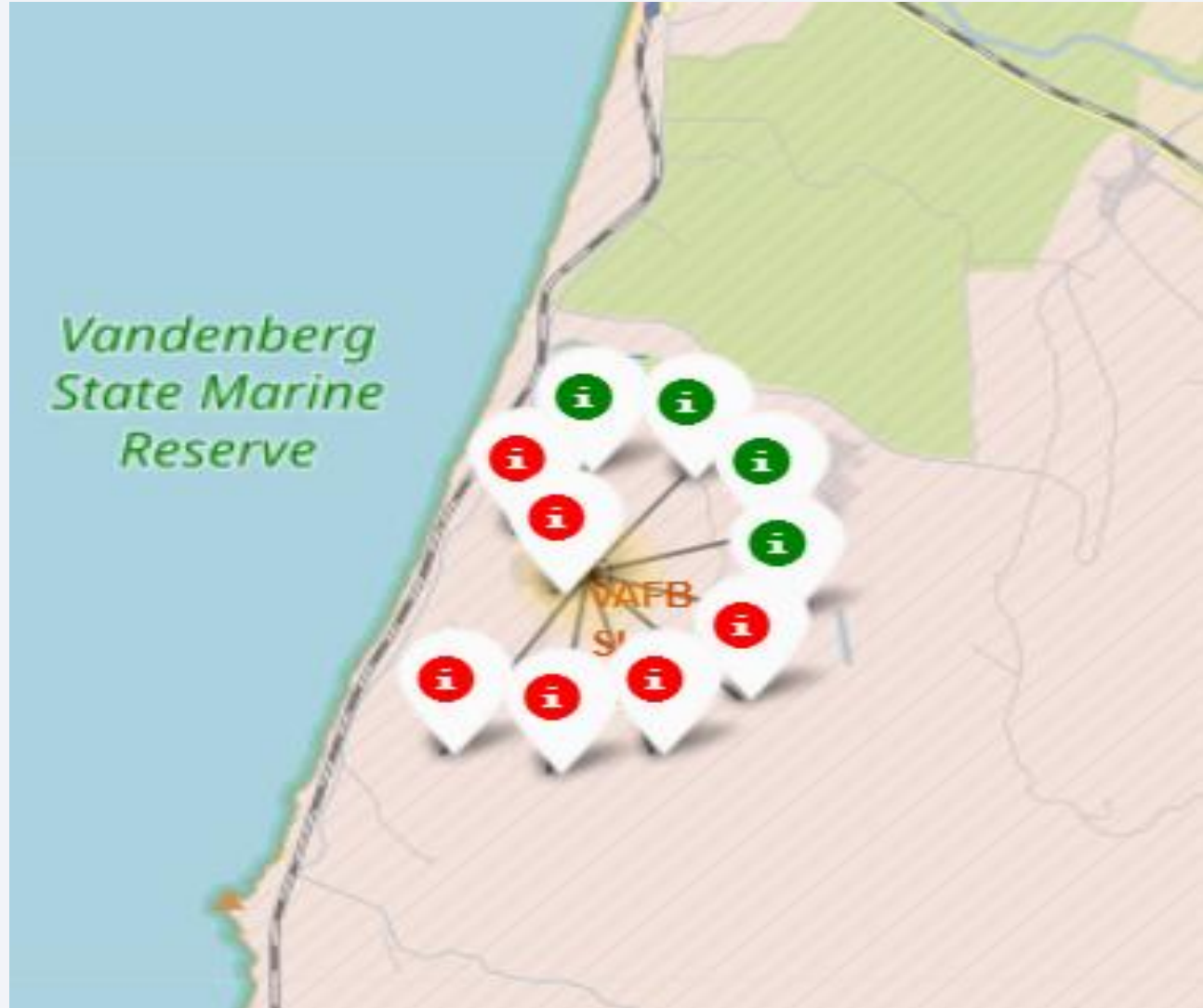


- CCAFS SLC-40



Color-labeled launch outcomes on the map

- Launch Site in California
 - VAFB SLC-4E



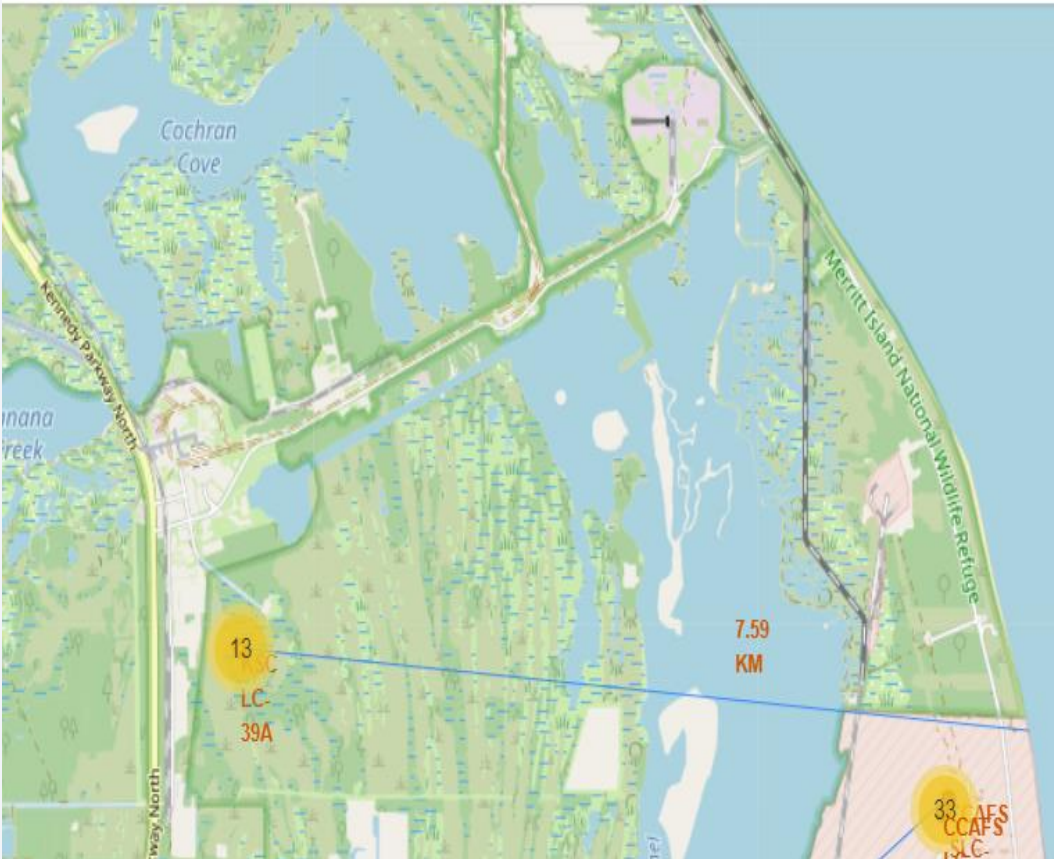
Proximities such as railway, highway, coastline

- Selected Launch Site: KSC LC-39A (**Reason:** Most successful launch site as per the data used).
- Screenshots: enclosed in the subsequent slide.

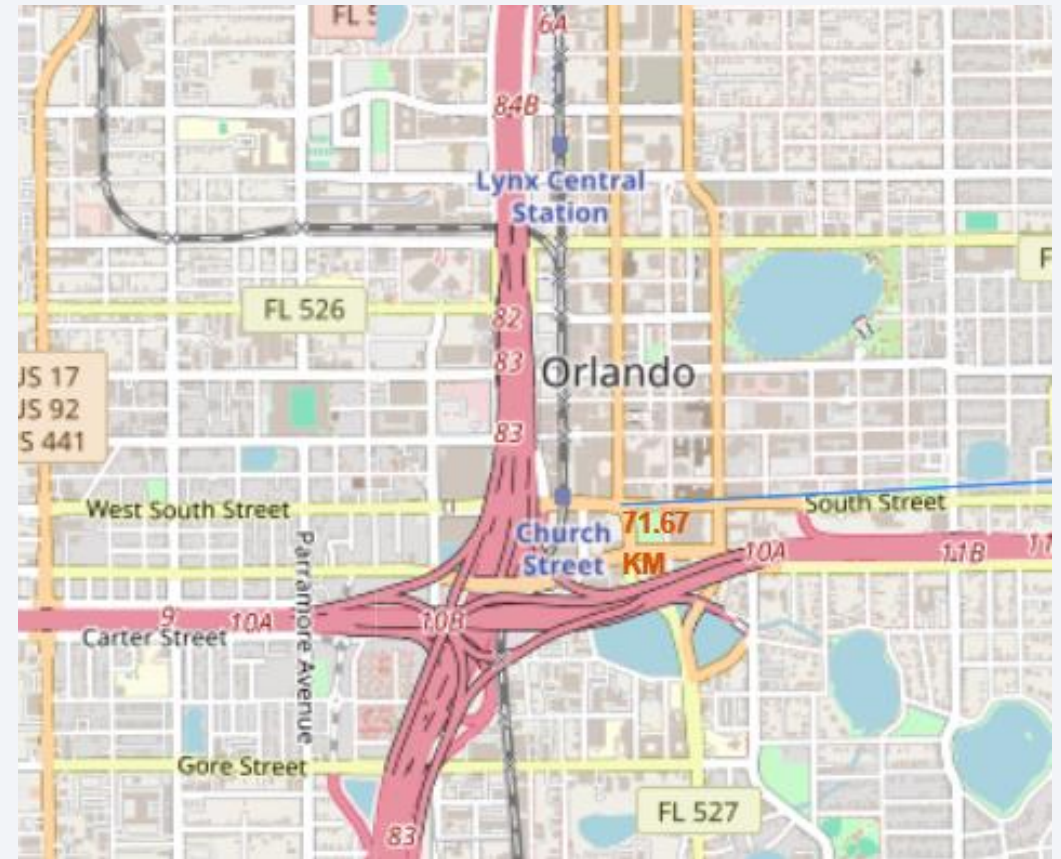
Sl. No.	Particulars	Distance (KM)	Reference taken	Analysis
1	Are launch sites in close proximity to coastline?	7.59	West Coast	Yes, the Launch site is fairly in close proximity to the coastline
2	Do launch sites keep certain distance away from cities?	71.67	Orlando	Yes, the launch sites keep certain distance away from the cities.
3	Are launch sites in close proximity to railways?	21.90	Florida East Coast Railways, Cocoa	No, the launch sites are not in close proximity to the railways.
4	Are launch sites in close proximity to highways?	20.20	Cheney Highway	No, the launch sites are not in close proximity to the highways.
5	Are launch sites in close proximity to Airport?	16.31	Space coast regional airport	No, the launch sites are not in close proximity to the airports.

Proximities such as railway, highway, coastline

Distance: Coast Line



Distance: Orlando city



Proximities such as railway, highway, coastline

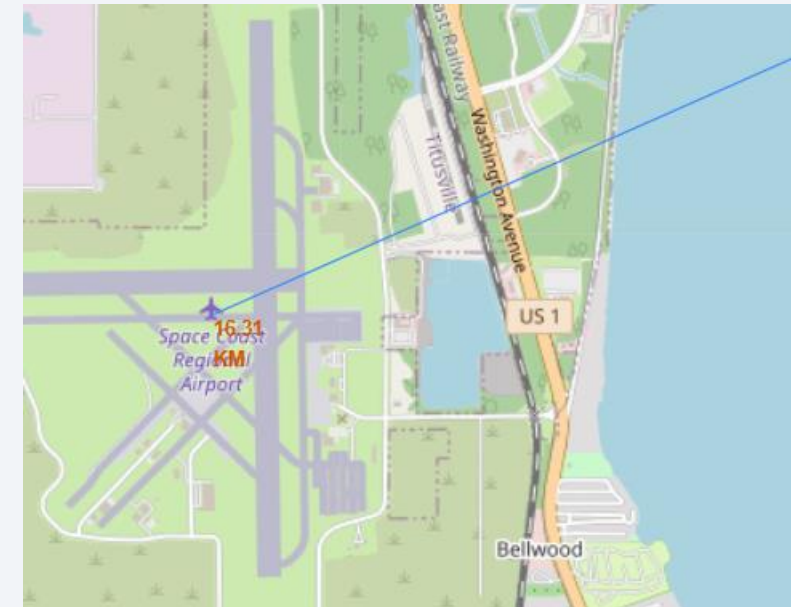
Distance: Cheney Highway



Distance: Florida East Coast Railways, Cocoa



Distance: Space coast regional airport





Section 4

Build a Dashboard with Plotly Dash

Pie-Chart: Launch Success for all sites

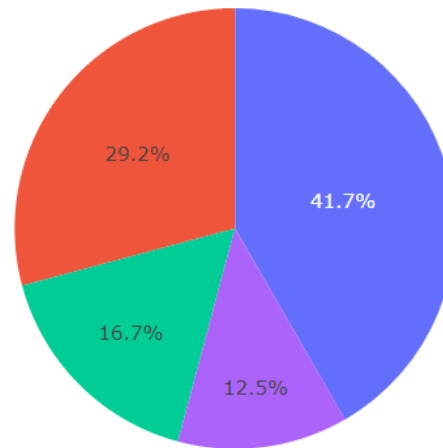
SpaceX Launch Records Dashboard

All Sites



Total Success Launches By all sites

It can be concluded that KSC LC 39A launch site had the most successful launches of all the sites (i.e., 41.7%).



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

KSC LC 39A (success rate: 41.7%) and CCAFS LC-40 (success rate: 29.2%) launch sites contribute to the majority of the successful launches.



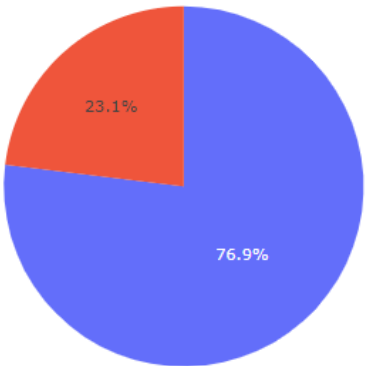
Pie Chart: Launch Site with highest launch success

Total Success Launches for site KSC LC-39A

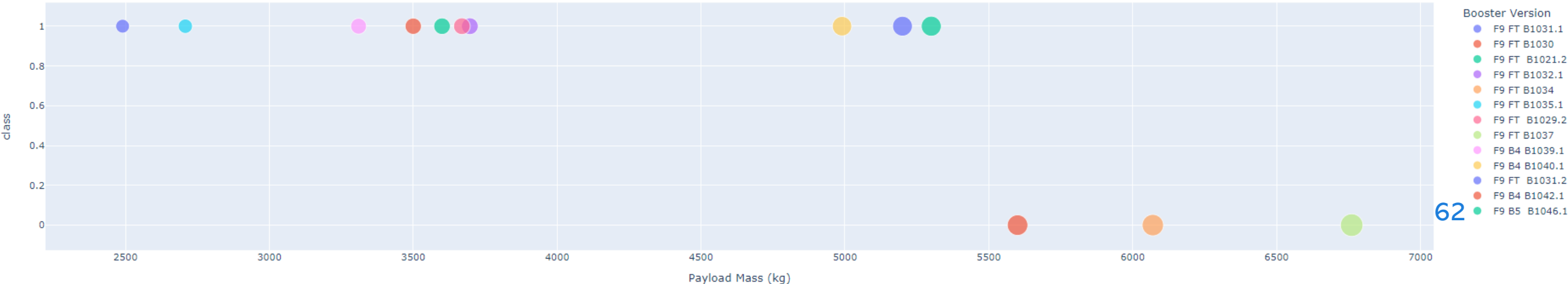
KSC LC 39A launch site had the most successful launches of all the sites.



76.9% of the total launches from this site were successful, while 23.1% were unsuccessful

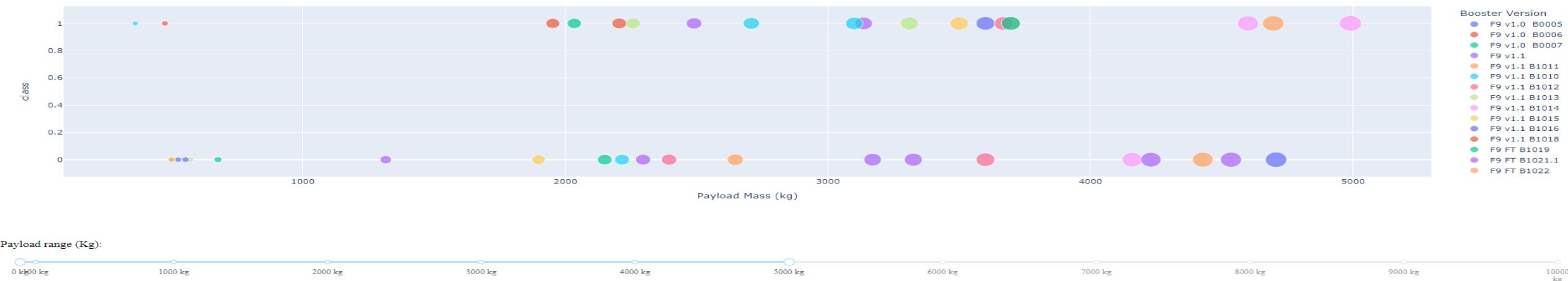


The Scatter plot below shows that most of the success launches carried lighter payloads



Scatter Plot: Payload vs. Launch Outcome scatter plot for all sites

Payload Mass Range(KG): 0 to 5000 KG



Analysis:

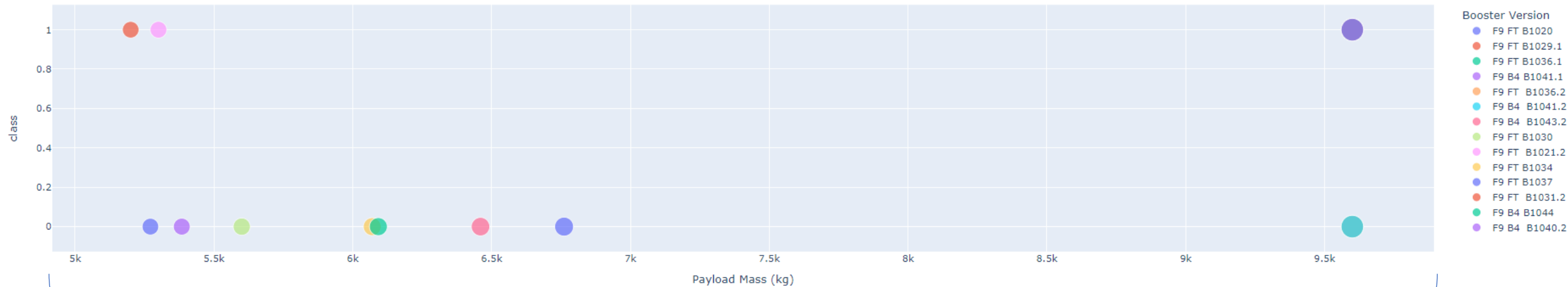
From the above scatter plot, it is observed that for loads up to 5000 Kgs, the success and failure rates are almost similar.

Booster version F9 v1.1 B1014 had the most successful launches for payloads beyond 4000 Kgs.

Booster version F9 FT B1021.1 had the most unsuccessful launches in the range.

Scatter Plot: Payload vs. Launch Outcome scatter plot for all sites

Payload Mass Range(KG): 5000 to 10000 KG



Analysis:

From the above scatter plot, it is observed that for loads exceeding 5000 Kgs, the success rates are lower.

Booster version F9 B4 B1041.1 had the most successful launch for payloads beyond 9000 Kgs.

Booster version F9 B4 B1041.2 had the most unsuccessful launch for payloads beyond 9000 Kgs.

Section 5

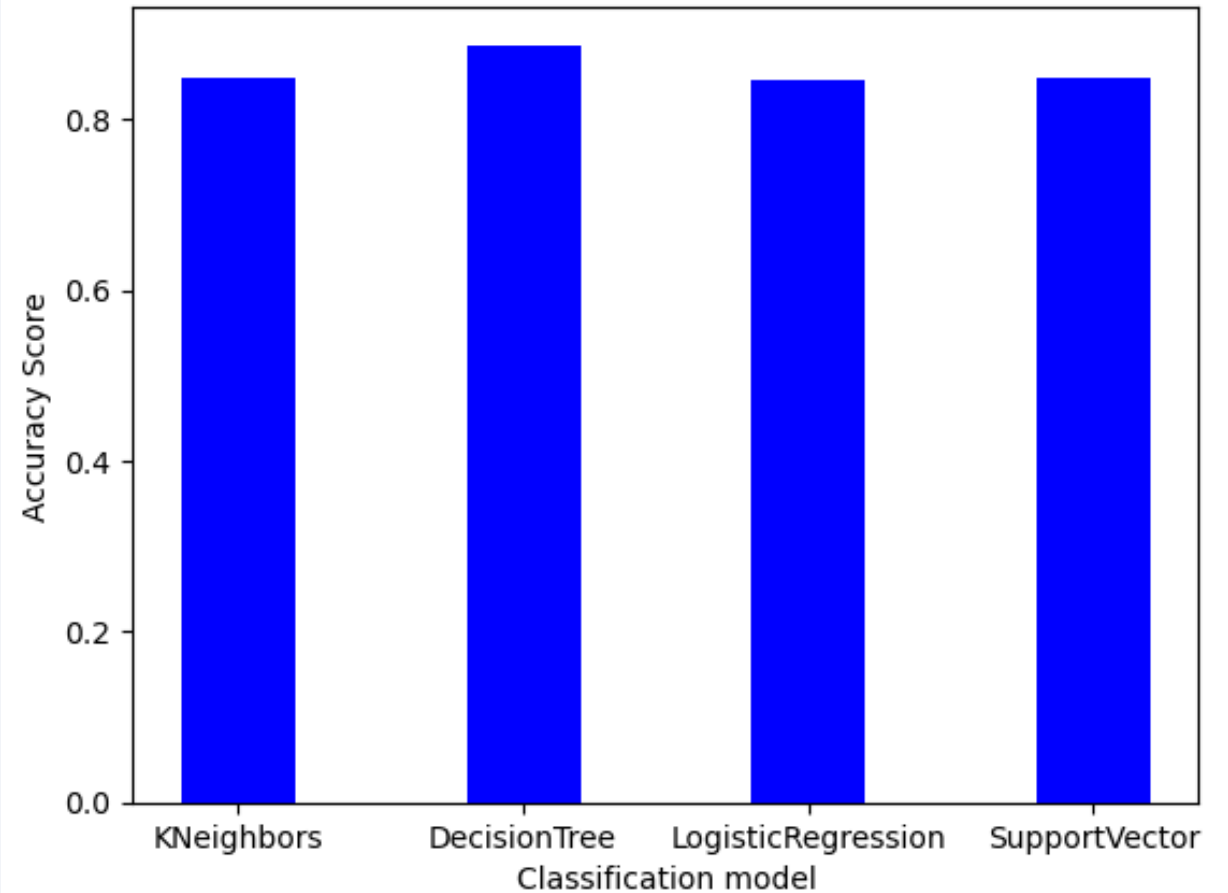
Predictive Analysis (Classification)

Classification Accuracy

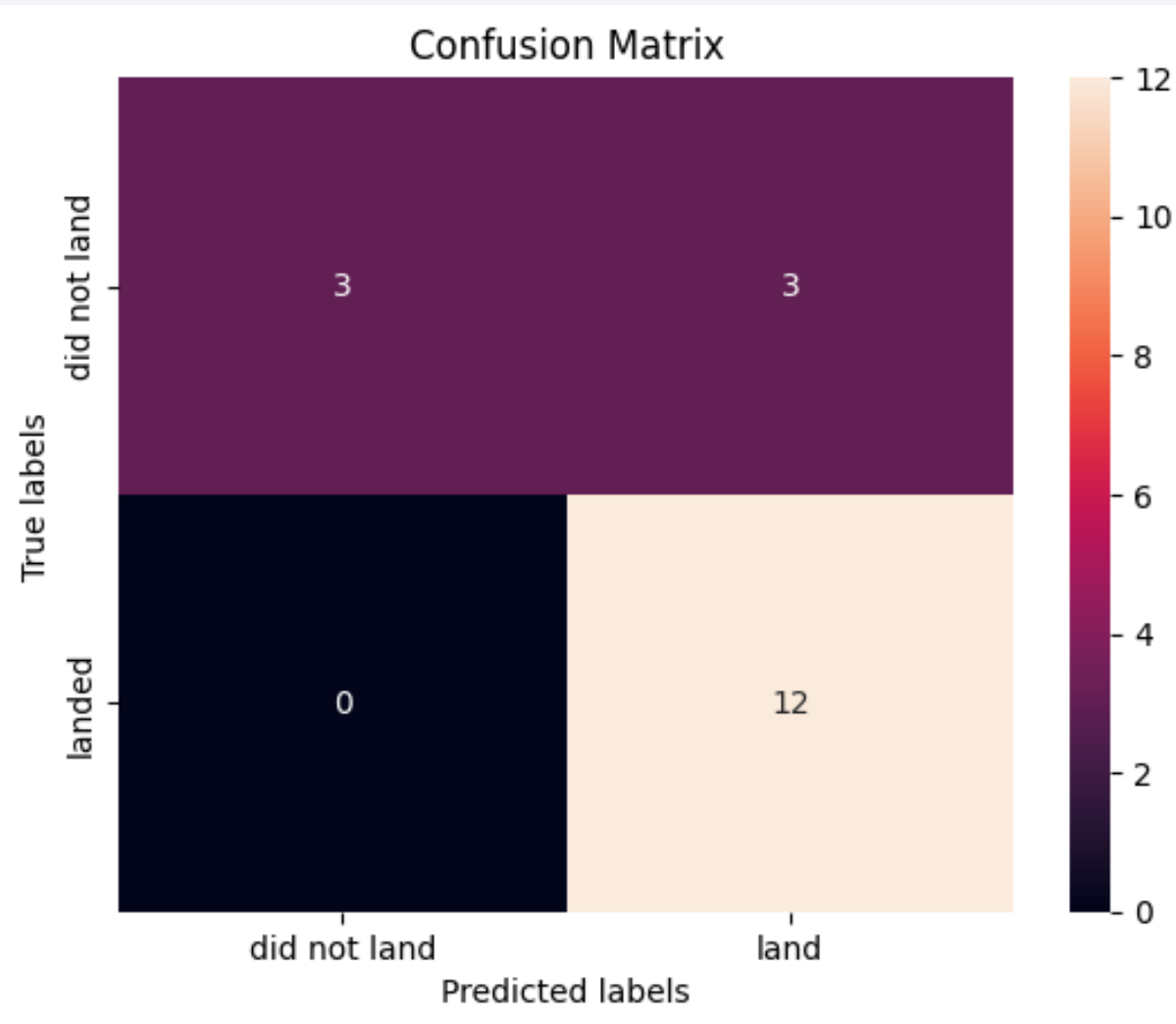
The classification models had an accuracy score of at least 0.8

The best model for the launch data was Decision Tree, with a classification score of 0.8875

Bar Chart showing accuracy for all built classification models



Confusion Matrix



The best model for the launch data was Decision Tree, with a classification score of 0.8875

However, there was an issue with the false positives.

It is observed that majority of unsuccessful landings classified as positive.

Conclusions

Number of Flights:

Larger the number of flights at a launch site, greater was the success rate.

Payload Mass:

Larger the rocket payload, greater is the success rate.

With a few exceptions, most of the launches with payload exceeding 10,000 have been successful.

Orbits:

Launches to ES-L1, GEO, HEO and SSO orbits had the most success rates.

Conclusions

Launch year:

An upward trend is observed in the success rates till the year 2020, with a downward movement in the year 2018

Launch site:

KSC LC 39A launch site had the most successful launches of all the sites

Launch Sites share a few characteristics, such as proximity to the coastal line etc.,

Classification Model:

Upon evaluation of the data with different classification models, decision tree classified turned out to be the best machine learning classification model

Appendix

- **Github Link to the repository:**
https://github.com/Vasudevkaranth/IBM_Datascience_Course_10_Capstone.git
- Links of data used:
 - SpaceX API (i.e., <https://api.spacexdata.com/v4/>) and
 - https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
- Photo used:
 - Slide 5: Photo by SpaceX: <https://www.pexels.com/photo/rocket-launch-space-discovery-23795/>

Thank you!

