Winning Space Race with Data Science

For the completion of the
Applied Data Science Capstone Project

Erwin R. Pasia
January 18, 2023

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data Collections using API and Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis (EDA) using Data Visualizations
  - Exploratory Data Analysis (EDA) using SQL
  - Building Interactive Visual Analytics using Folium
  - Building a Dashboard Web Apps using Plotly Dash
  - Predictive Analysis (using different Machine Learning Algorithms)
- **Summary of all results**
  - Exploratory Data Analysis Results
  - Interactive Analytics
  - Predictive Analysis Tests and Results

Introduction

- **Project background**
  - The commercial space industry has advanced significantly, with companies such as SpaceX making space travel more affordable. Their Falcon 9 rocket launches are particularly cost-effective, with a price of only $62 million, while compared to other providers that can cost upwards of $165 million.
  - This is primarily due to SpaceX's ability to “reuse the first stage of the rocket”. By predicting the success of the Falcon 9 first stage landing, we can estimate the cost of a launch, which can be a useful information for competing companies like SpaceY bidding against SpaceX for a rocket launch contract.
- **Problems you want to find answers**
  - Since SpaceY hired us to predict whether SpaceX can successfully recover Stage1 booster rocket. We need to come up with a Machine Learning model that would accurately predict the said success rate.

Methodology

Methodology

Executive Summary

- Data Collections
  - *Using the combination of SpaceX's public API and SpaceX's Wikipedia page using Web Scraping (Beautiful Soup)*
- Perform data wrangling
  - *Convert results into "training labels" with the booster either successfully or unsuccessful landed*
- Perform exploratory data analysis (EDA) using visualizations and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - *Choose the best Hyperparameters for different ML methods; SVM, Classification Trees, Logistic Regression, and KNN. Also tuned the models with GridSearchCV*

Data Collection

- The data collection process includes a combination of API requests from the SpaceX API and Web Scraping data from tables in the Wikipedia pages of SpaceX, Falcon 9 and Falcon Heavy Launches Records.

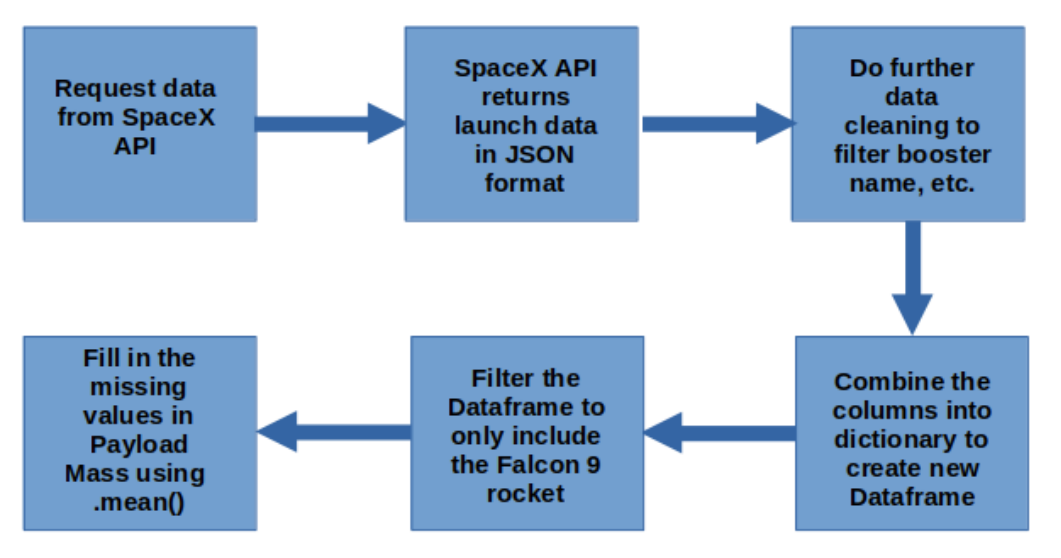
  - **SpaceX API Data Columns:** FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude

  - **Wikipedia Web Scrape Data Columns:** Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time

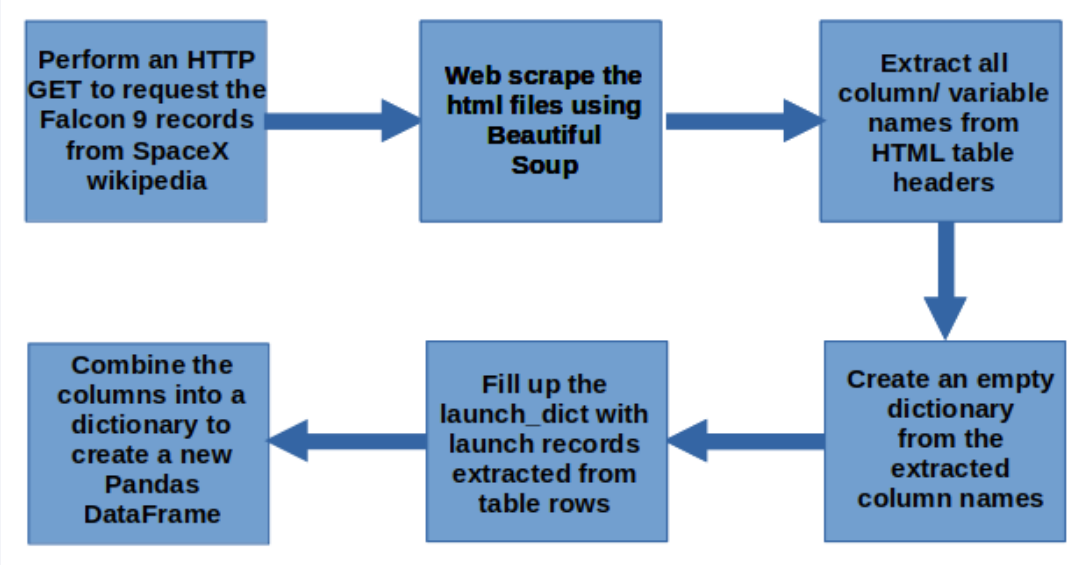
Data Collection

- Flow Charts:

SpaceX API



WebScraping



Note: *The final DataFrames of both Data Collection processes can be exported into CSV files.*

Data Collection – SpaceX API

Github Link

1.Request the rocket launch data from SpaceX API URL.

```
[6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

[7]: response = requests.get(spacex_url)
```

2. Decode the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize().

```
[11]: # Use json_normalize method to convert the json result into a dataframe

      data = pd.json_normalize(response.json())
```

3.Do further cleaning by filtering booster name, mass of the payload, the orbit, launch site, longitude, latitude, etc.

```
[16]: # Call getBoosterVersion
      getBoosterVersion(data)

[18]: # Call getLaunchSite
      getLaunchSite(data)

[19]: # Call getPayloadData
      getPayloadData(data)

[20]: # Call getCoreData
      getCoreData(data)
```

```
[13]: # Lets take a subset of our dataframe keeping only the features we want and the flight n
      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
      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
      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 da
      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)]
```

4. Combining the columns into a dictionary to create a new Pandas dataframe.

```
[23]: launch_dict = {'FlightNumber': list(data['flight_number']),
      'Date': list(data['date']),
      'BoosterVersion':BoosterVersion,
      'PayloadMass':PayloadMass,
      'Orbit':Orbit,
      'LaunchSite':LaunchSite,
      'Outcome':Outcome,
      'Flights':Flights,
      'GridFins':GridFins,
      'Reused':Reused,
      'Legs':Legs,
      'LandingPad':LandingPad,
      'Block':Block,
      'ReusedCount':ReusedCount,
      'Serial':Serial,
      'Longitude': Longitude,
      'Latitude': Latitude}
```

5.Filter the dataframe to only include Falcon 9 launches.

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

      data_falcon9 = data2[data2['BoosterVersion']=='Falcon 9']
      data_falcon9
```

6. Because there are missing values in the PayloadMass column, we use the .mean() and .replace() functions to replace np.nan values.

```
[29]: # Calculate the mean value of PayloadMass column
      pldm_mean = data_falcon9['PayloadMass'].mean()

      # Replace the np.nan values with its mean value
      temp = data_falcon9['PayloadMass'].replace(np.nan,pldm_mean)
      data_falcon9['PayloadMass'] = temp
      data_falcon9
```

Data Collection - WebScraping

Github Link

1. Perform an HTTP GET method to request the Falcon9 Launch HTML page from SpaceX wikipedia.

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

     page = requests.get(static_url)
```

2. Web scrape the html files using BeautifulSoup.

```
[6]: # Use BeautifulSoup() to create a BeautifulSoup object

     soup = BeautifulSoup(page.text, 'html.parser')
```

3. Extract all column/variable names from the HTML table header.

```
[8]: # Use the find_all function in the Be
     # Assign the result to a list called

     html_tables=soup.find_all('table')
```

4. Create an empty dictionary with keys from the extracted column names.

```
[12]: launch_dict= dict.fromkeys(column_names)

      # Remove an irrelvant column
      del launch_dict['Date and time ( )']

      # Let's initial the launch_dict with each value
      launch_dict['Flight No.'] = []
      launch_dict['Launch site'] = []
      launch_dict['Payload'] = []
      launch_dict['Payload mass'] = []
      launch_dict['Orbit'] = []
      launch_dict['Customer'] = []
      launch_dict['Launch outcome'] = []
      # Added some new columns
      launch_dict['Version Booster']=[]
      launch_dict['Booster landing']=[]
      launch_dict['Date']=[]
      launch_dict['Time']=[]
```

5. Fill up the launch_dict with launch records extracted from table rows.

Note: The code snippet is too long for posting. Please check the JupyterLab Notebook instead. Thanks!

6. Combine the columns into a dictionary to create a new Pandas DataFrame.

```
[14]: df=pd.DataFrame(launch_dict)
```

Data Wrangling

Github Link

For the Data Wrangling, the main ***objective*** is to ***convert the different data set*** outcomes into ***Training Labels;***

1 - means the booster successfully landed.
0 - means it was unsuccessful.

The following mission data set outcomes outlines the ffg:

True Ocean - the outcome was successfully landed to a specific region of the ocean.
False Ocean - the outcome was unsuccessfully landed to a specific region of the ocean.

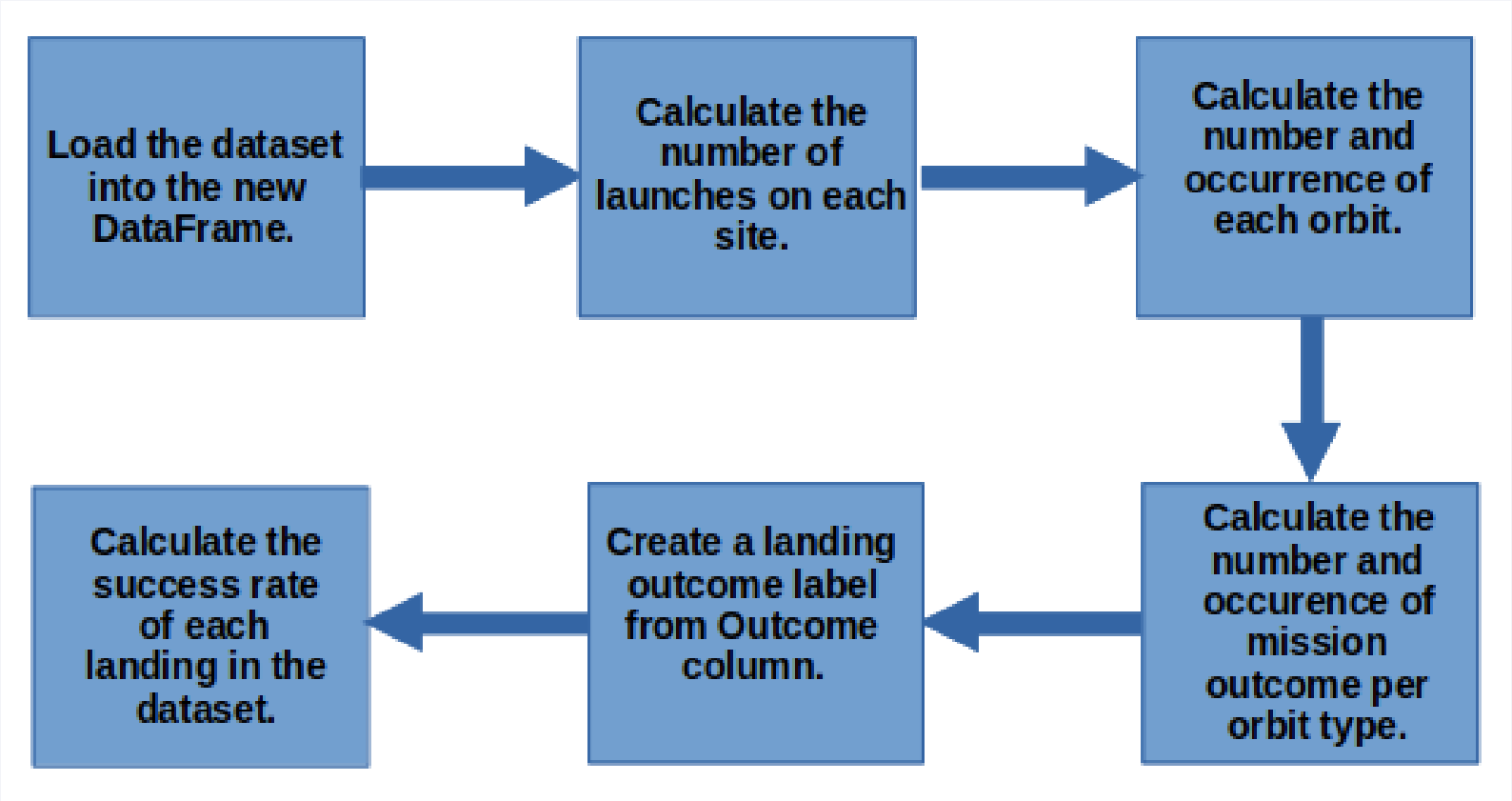
True RTLS - the outcome was successfully landed to a ground pad.
False RTLS - the outcome was unsuccessfully landed to a ground pad.

True ASDS - the outcome was successfully landed on a drone ship.
False ASDS – the outcome was unsuccessfully landed on a drone ship.

Data Wrangling

Github Link

Flow Chart:



Data Wrangling

Github Link

Data Processing:

1. Load the dataset into the new DataFrame.

```
[2]: df=pd.read_csv("https://cf-course
     df.head(10)
```

2. Calculate the number of launches on each site.

```
[5]: # Apply value_counts() on column LaunchSite

     df["LaunchSite"].value_counts()
```

3. Calculate the number and occurrence of each orbit.

```
[6]: # Apply value_counts on Orbit column

     df["Orbit"].value_counts()
```

4. Calculate the number and occurence of mission outcome per orbit type.

```
[8]: # landing_outcomes = values on Outcome column

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

5. Create a landing outcome label from Outcome column.

```
[11]: # landing_class = 0 if bad_outcome
      # landing_class = 1 otherwise

      def onemig(item):
          if item in bad_outcomes:
              return 0
          else:
              return 1
      landing_class = df["Outcome"].apply(onemig)
      landing_class
```

6. Calculate the success rate of each landing in the dataset.

```
[14]: df["Class"].mean()

[14]: 0.6666666666666666
```

EDA with Data Visualization

Github Link

Scatter Point Plots

Payload Mass vs. Flight Number
Launch Site vs. Flight Number
Payload Mass vs. Launch Site
Orbit Type vs. Flight Number
Orbit vs. Payload Mass

- The Scatter Point Plots were used for the following variables to indicate the correlations.

Bar Chart

- *Success Rate vs. Orbit Type*

- The Bar Chart was used here to quickly identify the success rate. Used to show segments of information like numeric percentages that needs to be compared

Line Chart

- *Success Rate vs. Year*

- The Line chart was used to quickly show the trend or projections.

EDA with SQL

Github Link

Load the dataset into the SPACEXTBL table in an IBM Db2 database in the IBM Cloud, then execute SQL queries to answer following requirements;

- Display the names of the unique launch sites in the space mission.
- Display 5 records where launch sites begin with the string 'CCA'.
- Display the total payload mass carried by boosters launched by NASA (CRS).
- Display average payload mass carried by booster version F9 v1.1.
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
- List the total number of successful and failure mission outcomes.
- List the names of the booster_versions which have carried the maximum payload mass.
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names in year 2015.
- Rank the count of landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order

Build an Interactive Map with Folium

IBM Cloud Link

The interactive maps with Folium mark the Launch Sites, successful and unsuccessful landings, and proximity to key locations. Like Railway, Highway, Coast, City, and Airport.

This allow us to understand why launch sites are located where they are. In addition, the Folium maps also visualizes successful landings relative to location.

Build a Dashboard with Plotly Dash Github Link (code only)

The Dashboard made with **Plotly Dash** is a rapid way to create powerful dashboard webapps. In our project, we have included a pie chart and a scatter point plot.

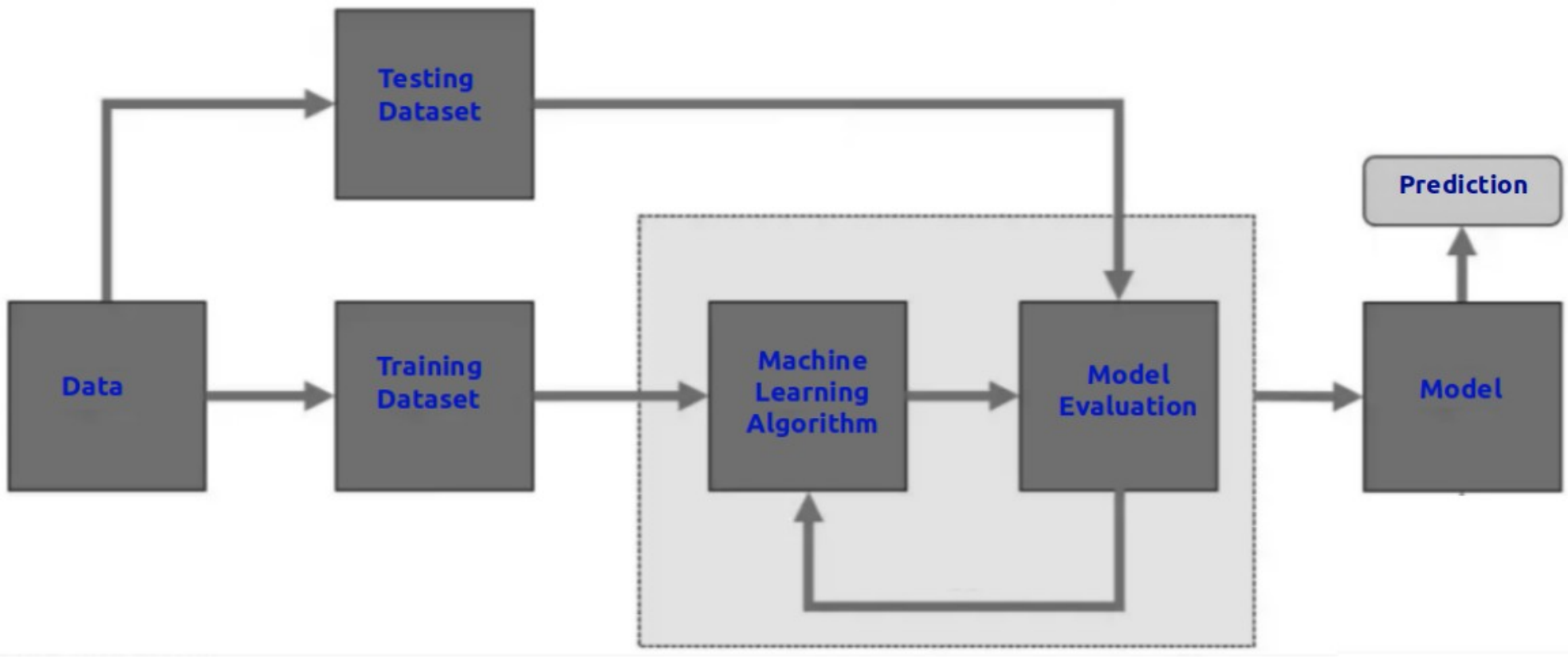
- The **Pie Chart** can be selected to show distribution of successful landings across all launch sites. It was chosen because it is used to visualize “launch site success rate”.

- The **Scatter Point Plot** list inputs: Either “All Sites” or “individual site” and payload mass on a slider between 0 and 10,000 kgs. It was chosen because it can help us see how success varies across launch sites, the payload mass, and booster version category.

Predictive Analysis (Classification)

Github Link

Machine Learning Workflow:



Finding the best ML Method

Predictive Analysis (Classification)

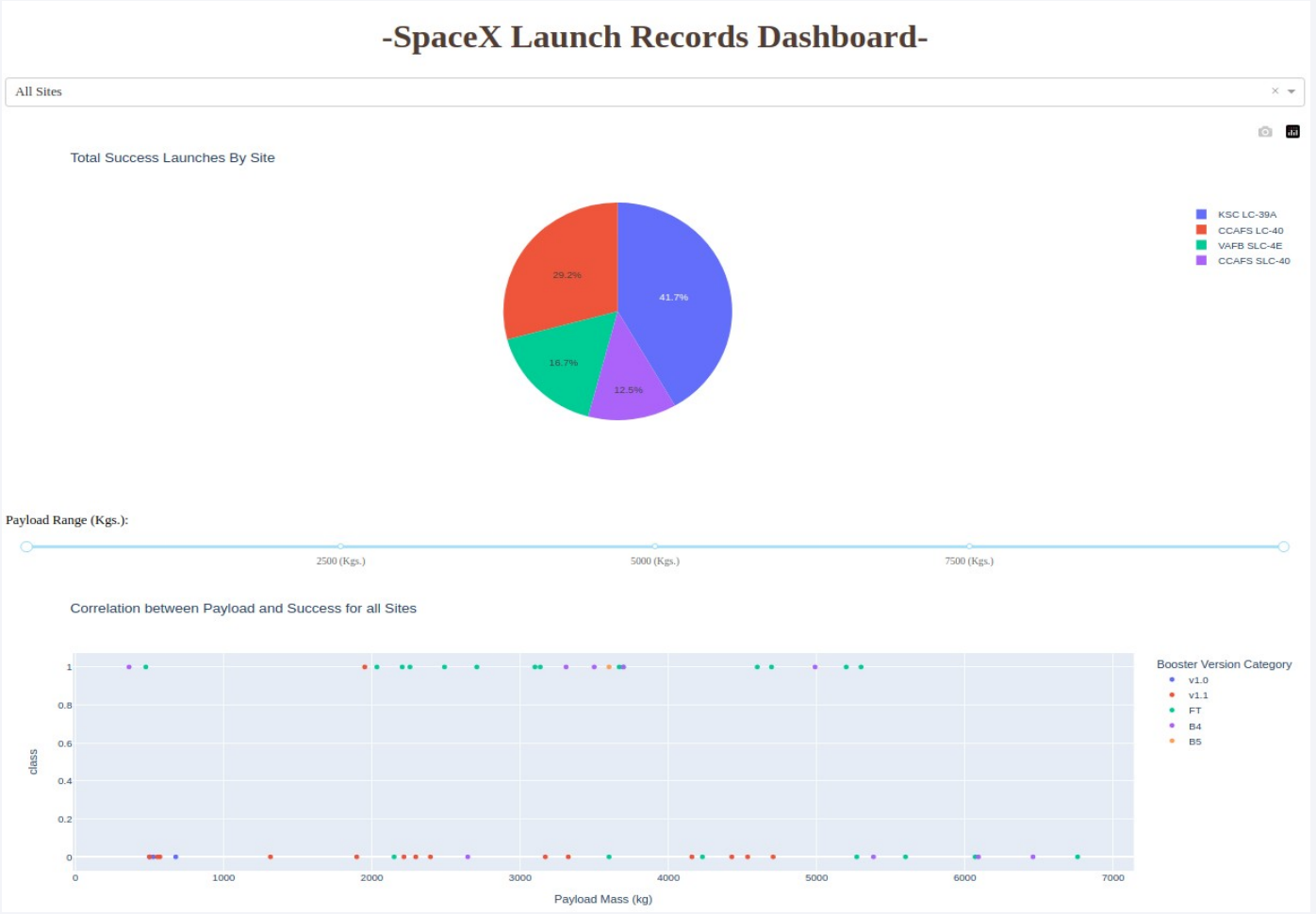
Github Link

Machine Learning Prediction Process

- Perform exploratory Data Analysis and determine Training Labels.
  - Create a column for the class.
  - Standardize the data.
  - Split into training data and test data.
- Find best Hyperparameter for SVM, Classification Trees and Logistic Regression.
  - Find the method performs best using test data.

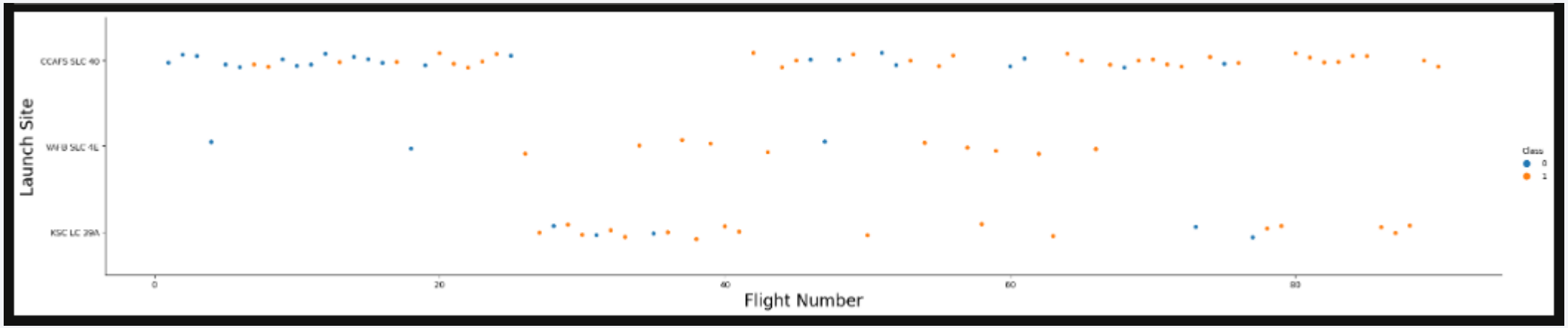
Results

- On the right, is the preview for the Dashboard with Plotly Dash.
- The detailed results of the insights drawn from below will be shown in the succeeding slides;
  - EDA with Visualization
  - EDA with SQL
  - Interactive Map with Folium
  - Interactive Dashboard with Plotly Dash
- By comparing the accuracies of the four ML methods, resulted the same level of about 83% for test data.



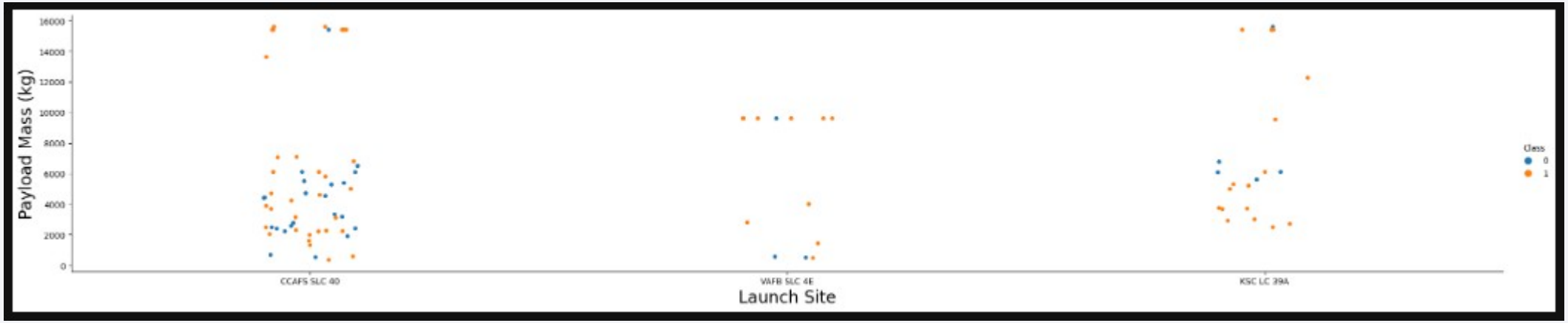
EDA with Visualizations

Flight Number vs. Launch Site



- The “class 1” (orange) represents successful launch while “class 0” (blue) represents unsuccessful launch.
- This scatter point plot shows that the success rate increased as the number of flights increased.
- The scatter point plot also shows that success rate has increased considerably from the 20th flight onwards, it is likely a big breakthrough.
- CCAFS SLC 40 appears to be the launch site that has the most volume.

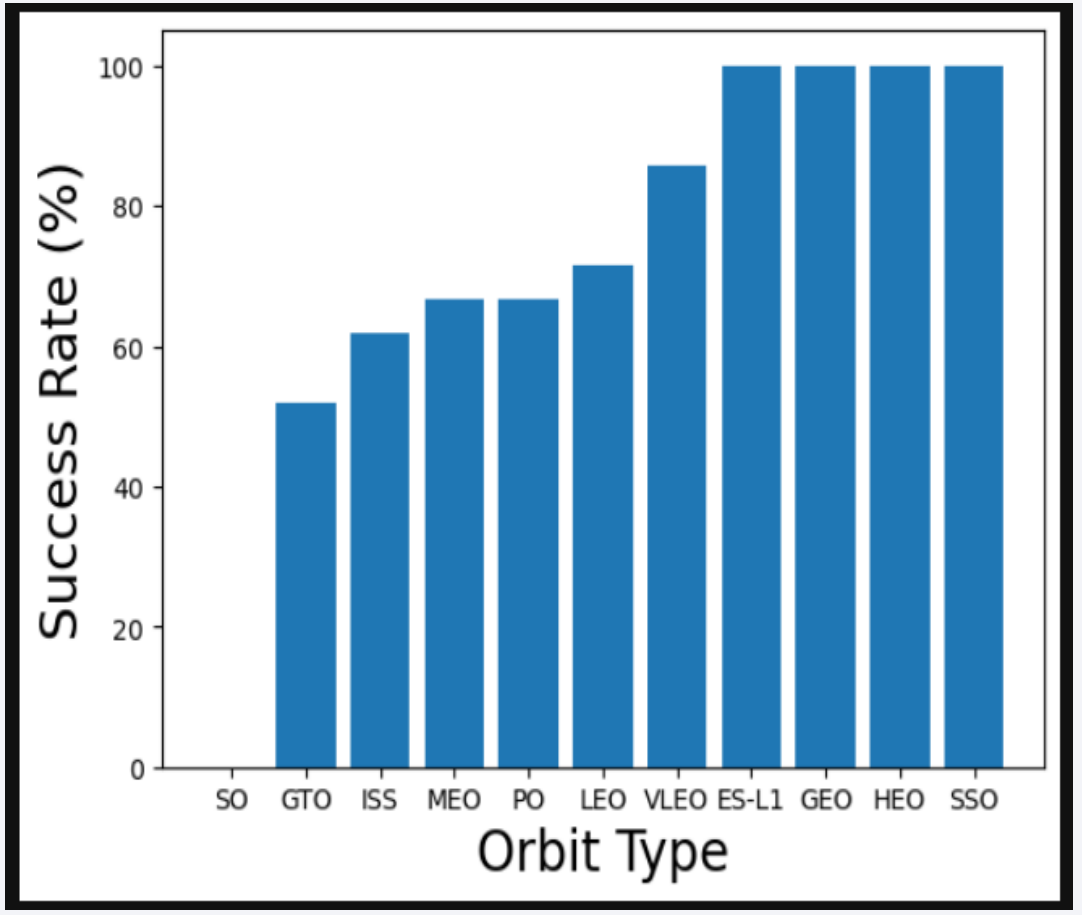
Payload vs. Launch Site



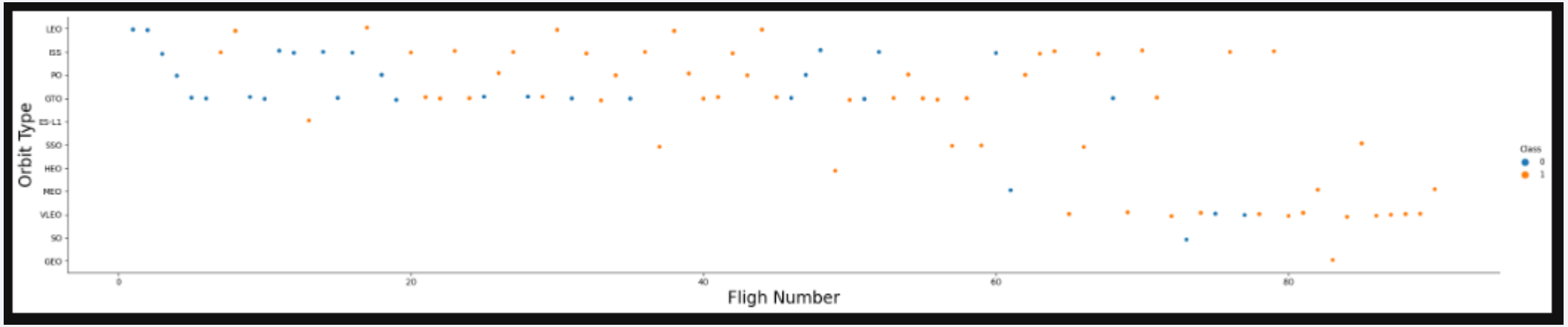
- The “class 1” (orange) represents successful launch while “class 0” (blue) represents unsuccessful launch.
- There are no clear patterns found between successful launch and Payload Mass.
- Payload Mass appears to mostly ranges from 0 to 6000 kgs. Different launch sites also seem to use different payloads.

Success Rate vs. Orbit Type

- The Orbit types SSO, HEO, GEO, and ES-L1 have the highest success rates at 100%.
- However, the success rate of orbit type GTO is only at 50%, which is the lowest except for type SO, which recorded failure in only first attempt.

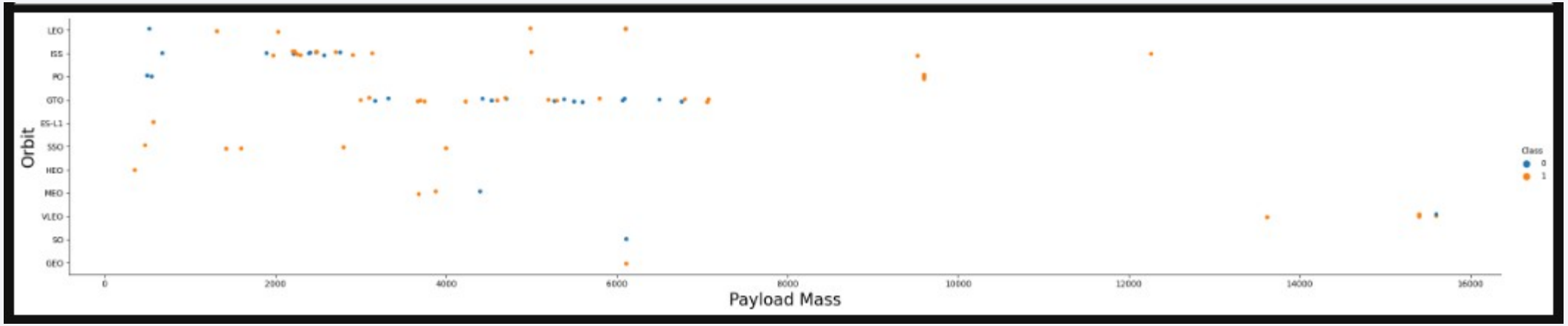


Flight Number vs. Orbit Type



- The “class 1” (orange) represents successful launch while “class 0” (blue) represents unsuccessful launch.
- In most cases, the launch outcome seems to be correlated with the flight number.
- On the other hand, in GTO orbit which seems to have no relationship between flight numbers and success rate.
- The LEO have a moderate success rate, and it seems VLEO, has a high success rate, is used the most in recent launches.

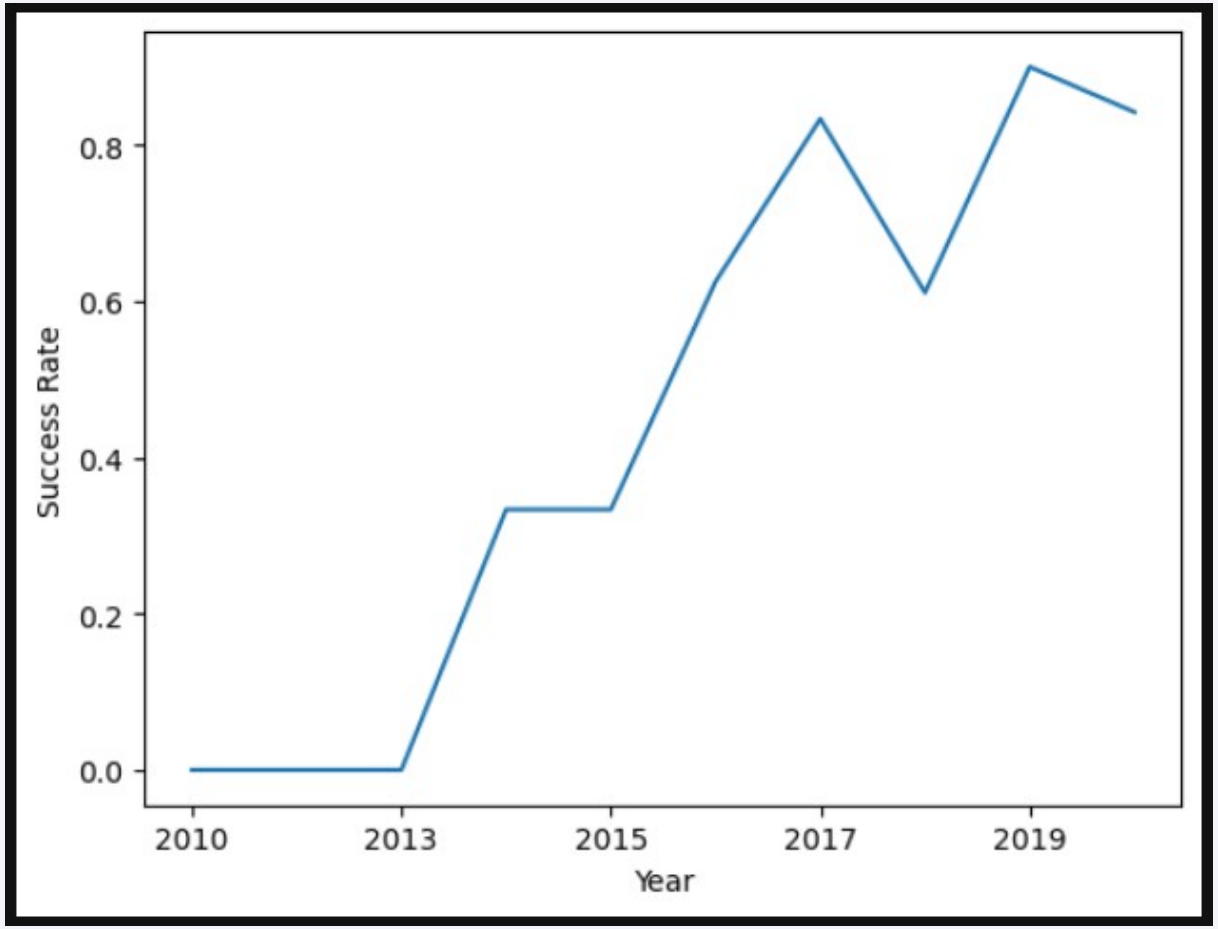
Payload vs. Orbit Type



- The “class 1” (orange) represents successful launch while “class 0” (blue) represents unsuccessful launch.
- With heavy payloads, the successful landing or positive landing rate are mostly for LEO and ISS Orbit Types.
- In the case of GTO, it is quite hard to tell between the positive landing rate and the negative landing because they are all gathered together.

Launch Success Yearly Trend

- Based from the trend, SpaceX success rate started its milestones around year 2013 up until year 2017.
- Around at about year 2018 there was a slight dipping of the success rate.
- Year 2018 onwards success rate reached about 80%.



EDA with SQL

SpaceX Launch Site Names

- SQL Query and Result via Jupyter Notebook:

```
[7]: %sql select DISTINCT LAUNCH_SITE from SPACEXTBL
```

- There are four unique launch sites namely; CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, and VAFB SLC-4E
- When the SQL DISTINCT clause is used in the query, only unique values are displayed in the Launch_Site column from the SpaceX table.

Launch Site Names Begin with 'CCA'

- SQL Query and Result via Jupyter Notebook:

```
[8]: %sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5
```

[8]:

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

- Only 5 records where launch sites begin with `CCA` were displayed using “limit 5” query.
- With the “like” operator and the “%” sign, altogether can query any launch_site that starts with “CAA”.

Total Payload Mass

- SQL Query and Result via Jupyter Notebook:

```
[9]: %sql select sum(payload_mass__kg_) as sum from SPACEXTBL where customer like 'NASA (CRS)'
```

- We can use the sum() function to calculate the sum of column payload_mass__kg_.
- In the “where” clause, we can filter the dataset to perform calculations only if Customer is NASA (CRS).

Average Payload Mass by F9 v1.1

- SQL Query and Result via Jupyter Notebook:

```
[10]: %sql select avg(payload_mass__kg_) as AVERAGE_PAYLOAD_MASS from SPACEXTBL where booster_version like 'F9 v1.1'
```

```
[10]: average_payload_mass
      2928
```

- We can use the avg() function to calculate the average value of column payload_mass__kg_.
- Using the “where” clause, we can filter the dataset to perform calculations only if Booster_version is F9 v1.1.

First Successful Ground Landing Date

- SQL Query and Result via Jupyter Notebook:

```
[21]: %sql SELECT MIN(DATE) AS first_successful_landing_date FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

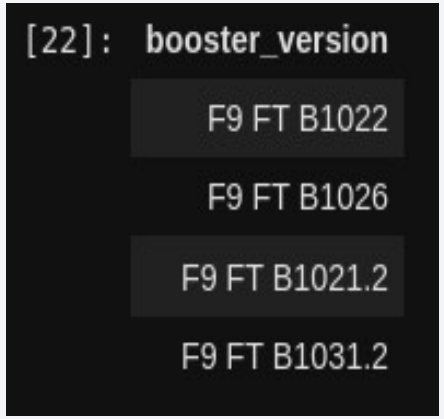
[21]:	first_successful_landing_date
	2015-12-22

- We can use the MIN() function to find out the earliest date in the column DATE.
- In the WHERE clause, we can filter the dataset to perform a query showing only if Landing__outcome is Success (ground pad).

Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL Query and Result via Jupyter Notebook:

```
[22]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)
```



[22]:

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

- In the WHERE clause, we can filter the dataset to perform a search if Landing__outcome is Success (drone ship).
- Using the AND operator we can display a record result if additional condition PAYLOAD_MASS__KG_ is between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes

- SQL Query and Result via Jupyter Notebook:

```
[23]: %sql SELECT MISSION_OUTCOME, COUNT(*) AS total_number FROM SPACEXTBL GROUP BY MISSION_OUTCOME
```

[23]:

mission_outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

- We can use the COUNT() function to calculate the total number of columns.
- We can also incorporate the GROUP BY statement, which groups rows that have the same values into summary rows to find the total number in each “Mission_outcome”.
- Based from the query result, it looks like SpaceX have successfully completed nearly 99% of its launch missions.

Boosters Carried Maximum Payload

- SQL Query and Result via Jupyter Notebook:

```
[29]: %sql SELECT DISTINCT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_)FROM SPACEXTBL)
```

[29]:

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

- We can use a subquery, by first finding the maximum value of the payload with MAX() function. Second to filter the dataset to perform a search if PAYLOAD_MASS__KG_ is the maximum value of the payload.
- Based from the query result, version F9 B5 B10xx.x booster blocks are able to carry the maximum payload.

2015 Launch Records

- SQL Query and Result via Jupyter Notebook:

```
[30]: %sql SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND YEAR(DATE) = '2015'
```

[30]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- In the WHERE clause, we can filter the dataset to perform a search if Landing__outcome is Failure (drone ship).
- We can then use the AND operator to display a record if additional condition YEAR is 2015.
- Based from our query results. In the year 2015, there were two landing failures on the drone ships.

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

- SQL Query and Result via Jupyter Notebook:

```
[31]: %sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS total_number FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING__OUTCOME ORDER BY total_number DESC
```

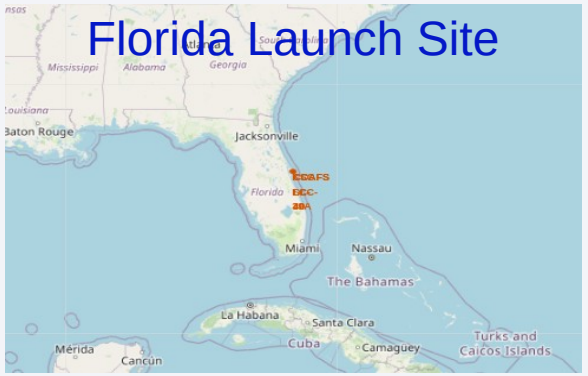
[31]:

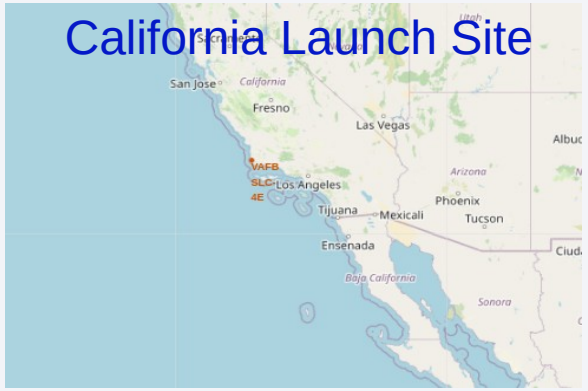
landing__outcome	total_number
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

- In the WHERE clause, we can filter the dataset to perform a search if the date is any between 2010-06-04 and 2017-03-20.
- We can use the ORDER BY keyword to sort the records by total number of landing, and using DESC keyword to sort the records in descending order.
- Based from the query results, the number of successes and failures between 2010-06-04 and 2017-03-20 was similar.

Interactive Map with Folium (Launch Sites Proximities Analysis)

SpaceX Launch Site Locations

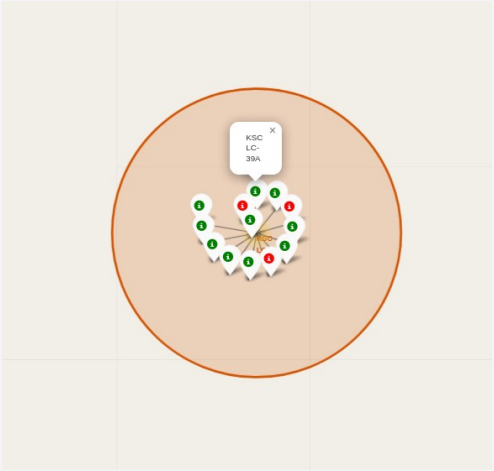


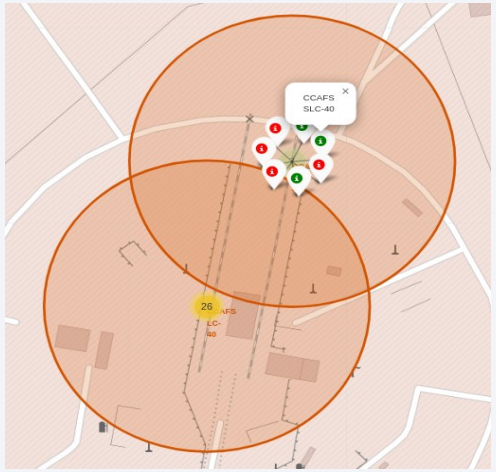


- All the SpaceX Launch Sites can be seen close to the coastlines.

Color-Labeled Launch Outcomes On the Map

SpaceX Launch Sites in Florida

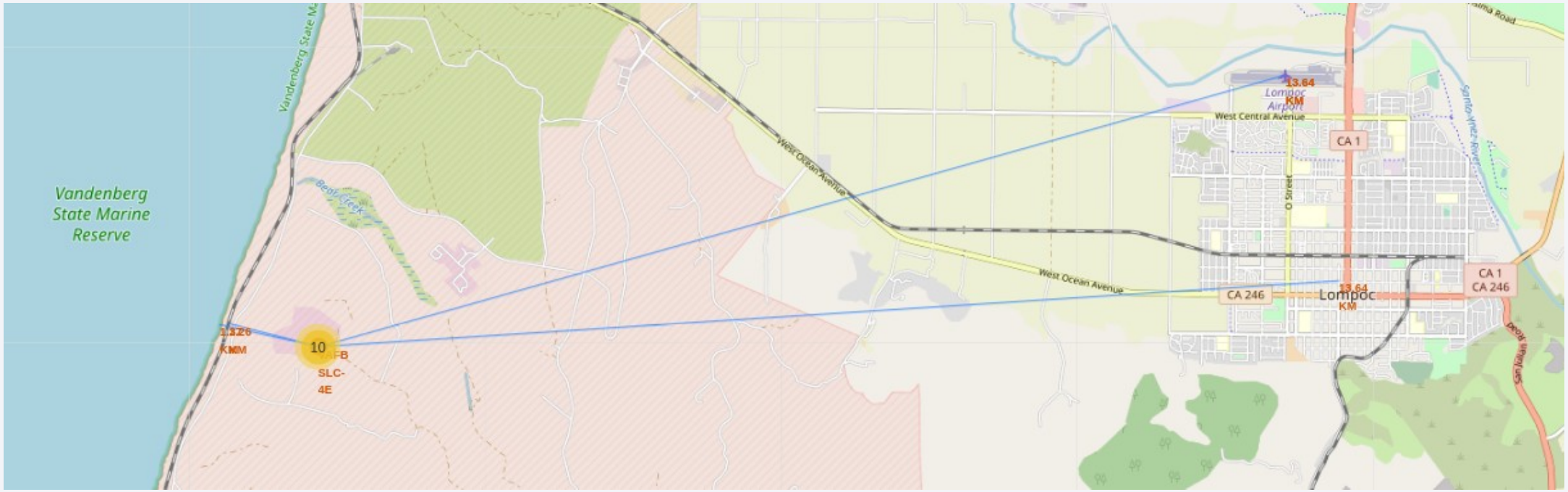




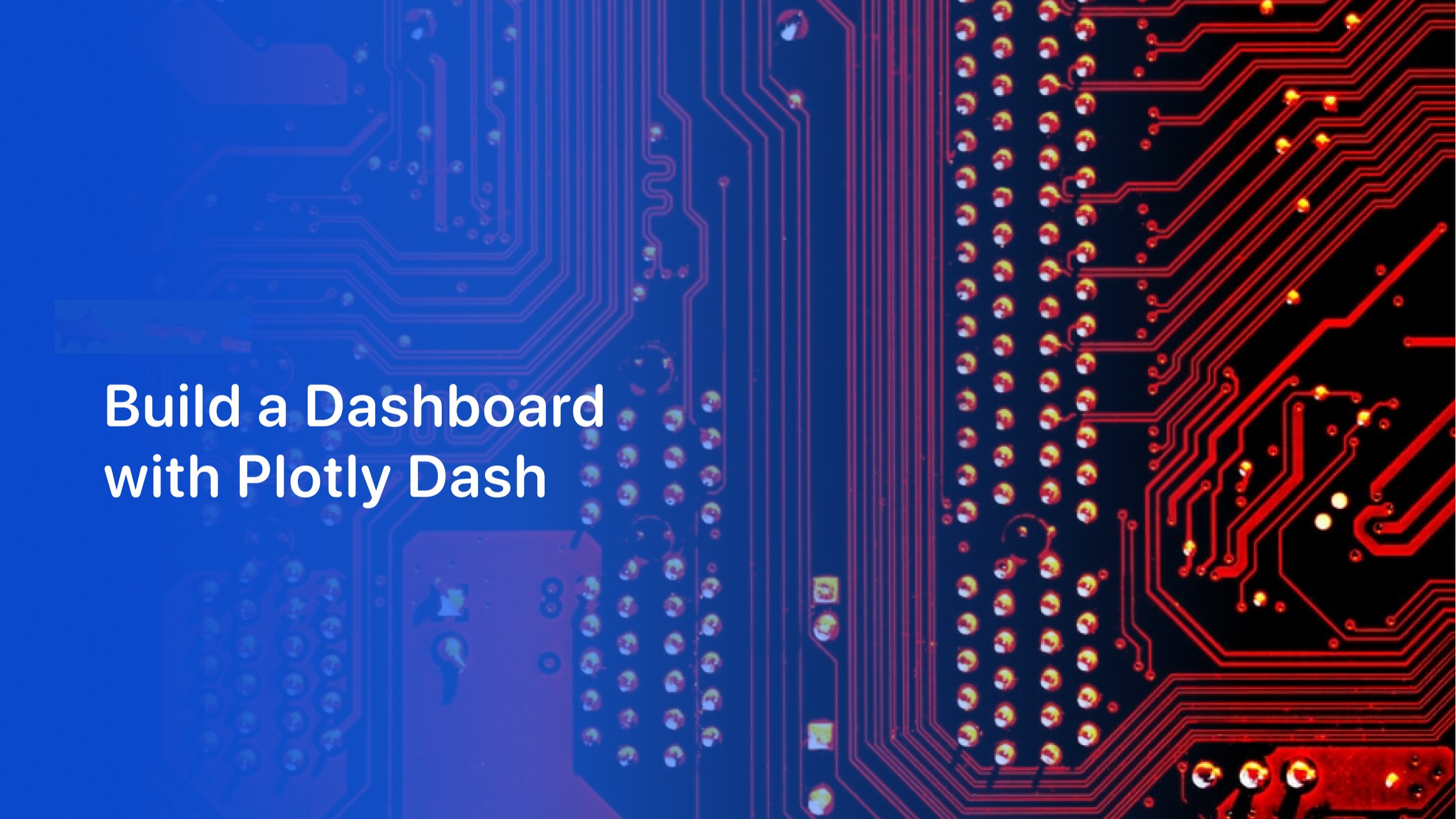
SpaceX Launch Sites in California

- Using the Marker Clusters, clicking each would show whether its successful landing colored “green”. Unsuccessful landing colored “red”.

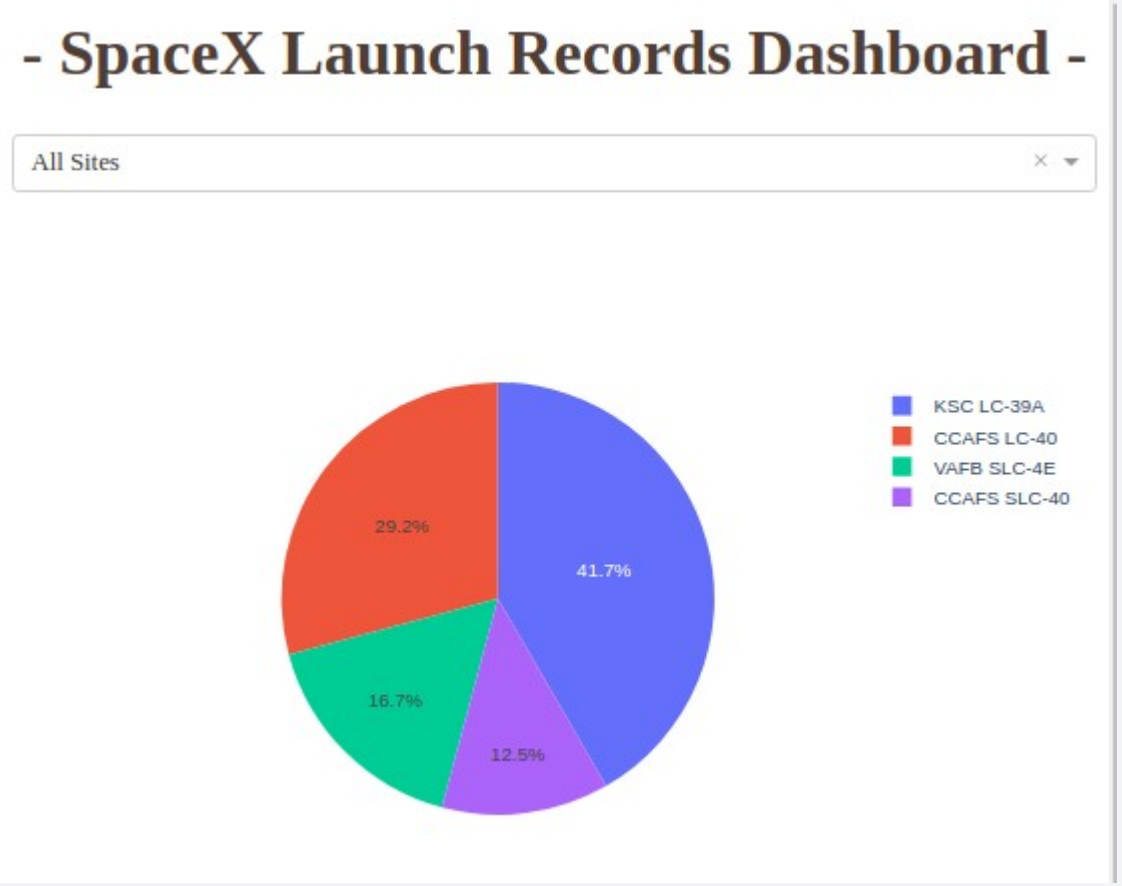
SpaceX Launch Site Proximities



- By calculating the distances of Launch Site Markers, it was discovered that each sites are close to **“railways”** or **“highways”** for transporting equipments and key personnel. It is also close to **“coastlines”**. However, they are far from the **“cities”** and **“airports”** so that every launch failures would not pose a threat to populous areas.

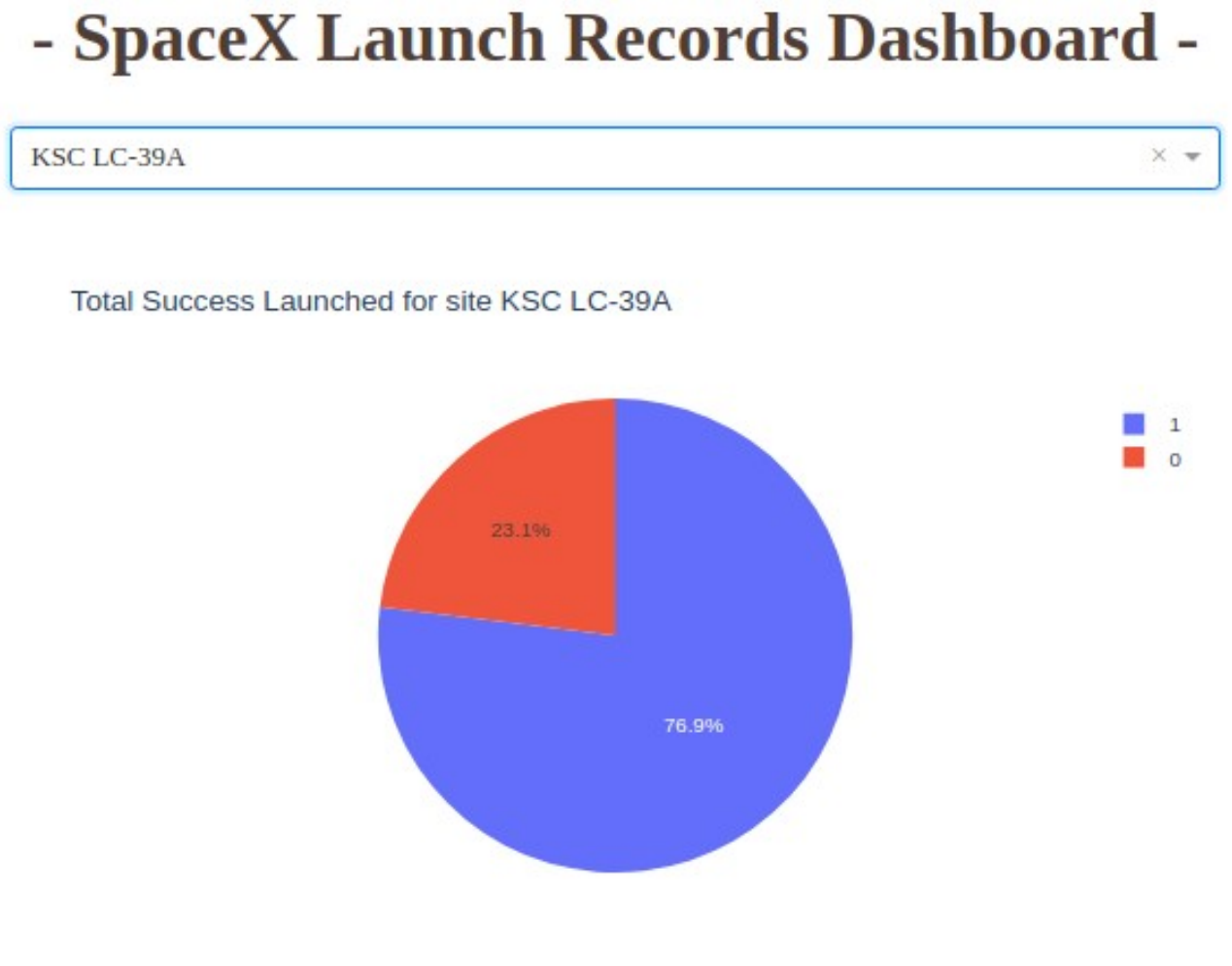

Build a Dashboard
with Plotly Dash

Successful Launches Across Launch Sites



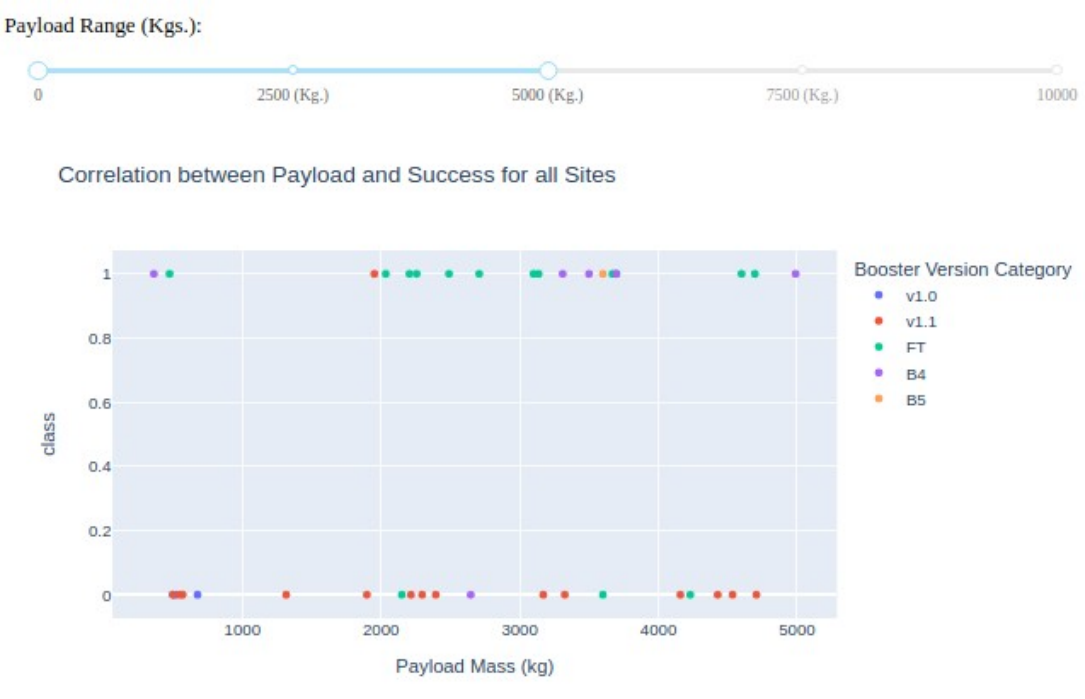
- KSLC-39A launch site records the most launch success among all sites.
- The VAFB SLC-4E has the fewest launch success, possibly due to;
  - Too small data sample.
  - The complex is own by the US Space Force. It could be that some launches are deemed classified national security information.

Launch Site with Highest Launch Success Ratio



- The KSLC-39A SpaceX launch site recorded the highest success rate with 10 landing successes (76.9%) and 3 landing failures (23.1%).

Payload vs. Launch Outcome Scatter Plot for all Sites

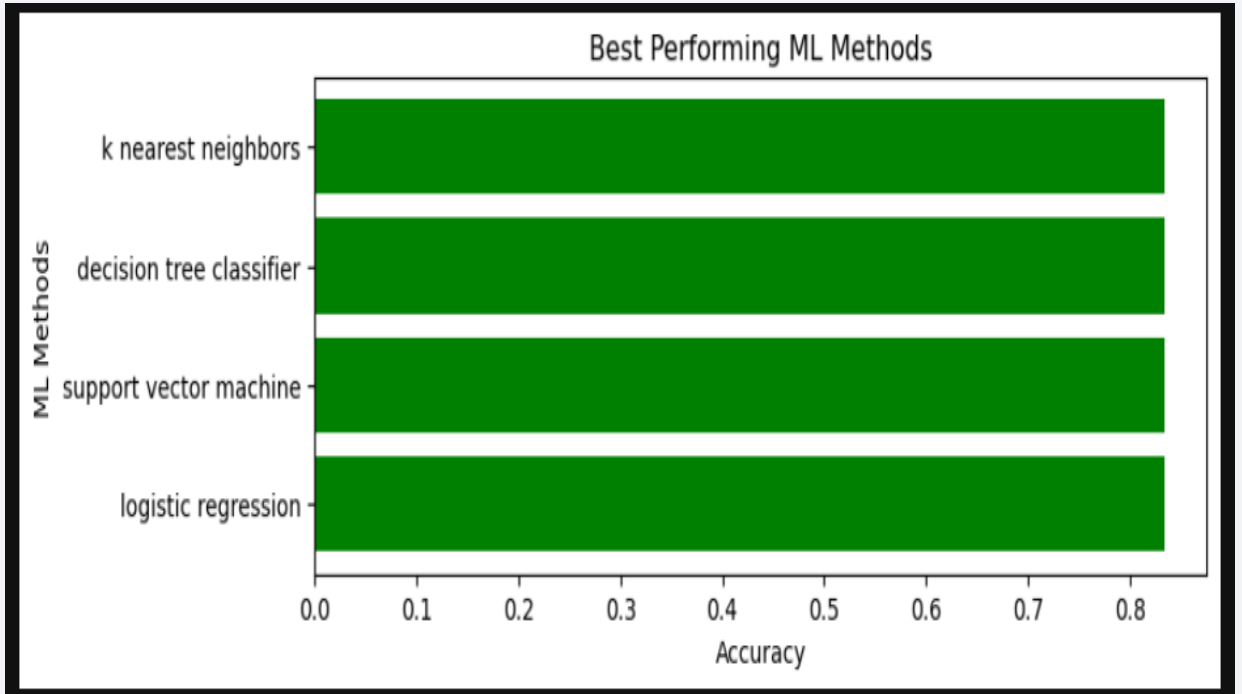


- The scatter point plots showed that the launch success rate (class 1) for “low-weighted” payloads (0kg – 5,000kg range) are higher than “heavy-weighted” payloads (5,000kg - 10,000kg range).

Predictive Analysis (Classification)

Classification Accuracy

- The accuracy of all models was virtually the same at 83.333%.
- It should be noted that the test size was small at 18 test samples.
- More data is needed to determine the optimal model.

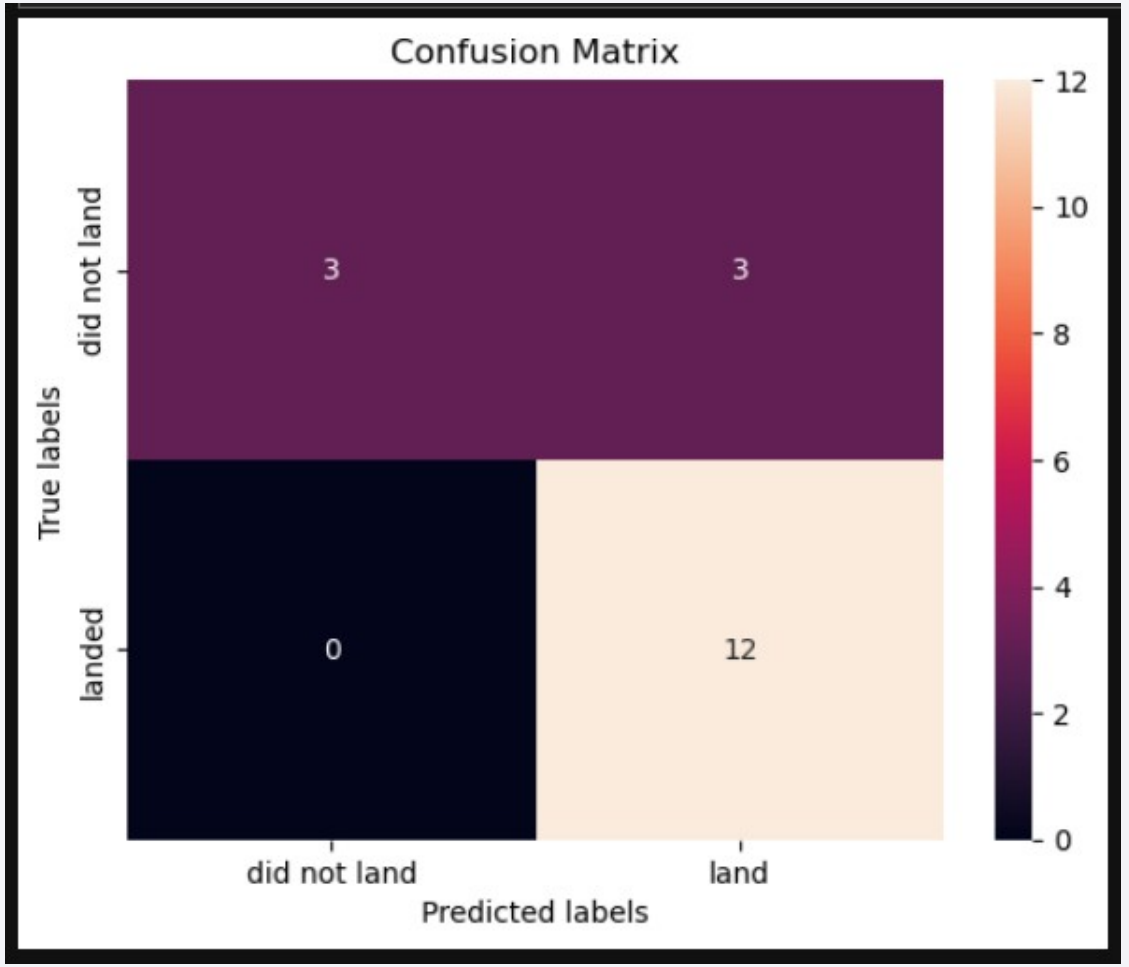



```
print(methods)
print(accuracy)
```

```
['logistic regression', 'support vector machine', 'decision tree classifier', 'k nearest neighbors']
[0.8333333333333334, 0.8333333333333334, 0.8333333333333334, 0.8333333333333334]
```

Confusion Matrix

- The confusion matrix outcomes are the same for all models.
- The models predicted 12 successful landings when the true label was successful and 3 failed landings when the true label was failure. But there were also 3 predictions that said successful landings when the true label was failure (false positive).
- Overall, these models predict successful landings.



Conclusions

- As the number of flights increased, their success rate also increased.
- Launch site KSLC-39A has the highest number of launch success and the highest success rate among all sites.
- At 100% success rates, SSO, HEO, GEO, and ES-L1 have the highest among Orbital Types.
- The launch site is close to railways, highways, and coastline, but far from cities and airports.
- The launch success rate of low-weighted payloads is higher than heavy-weighted ones.
- All the machine learning models tested had the same accuracy of 83.333%. If possible more data should be collected to better determine the best model and improve its accuracy.
- Allon Mask of SpaceY can use our “model” to predict whether a certain launch will have a successful “first stage rocket” return landing, even before take-off to determine whether the bid should be made or not.

Appendix

- Github Link
- Applied Data Science Capstone Course - Coursera
- IBM Data Science Professional Certificate V2 - Coursera

Thank you!