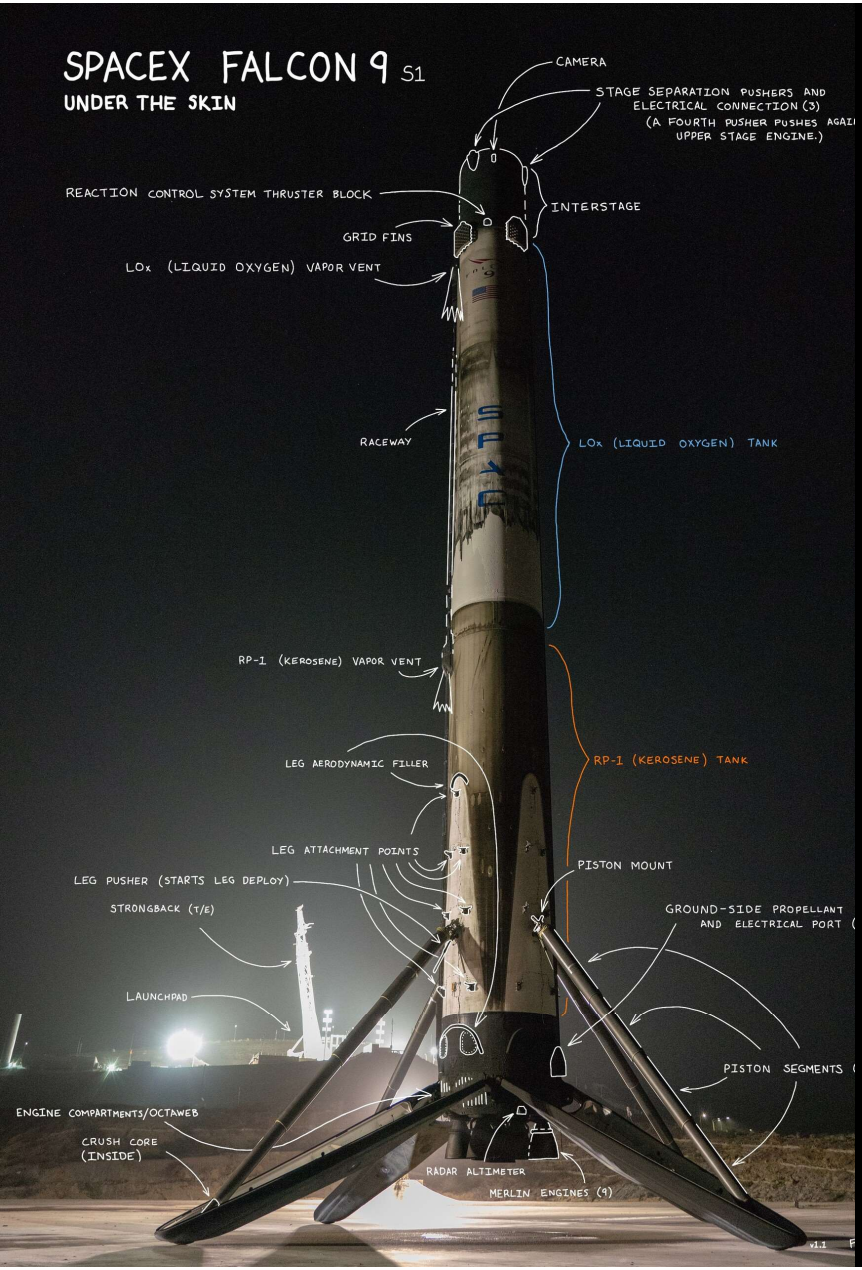
Winning Space Race with Data Science

Rishab Handa
14-08-2022

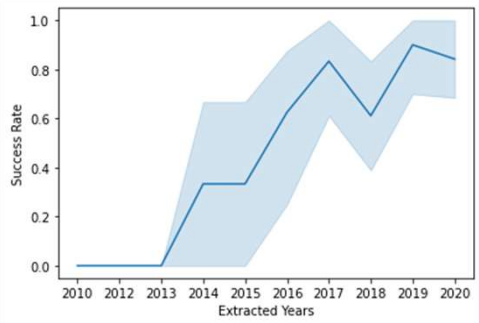


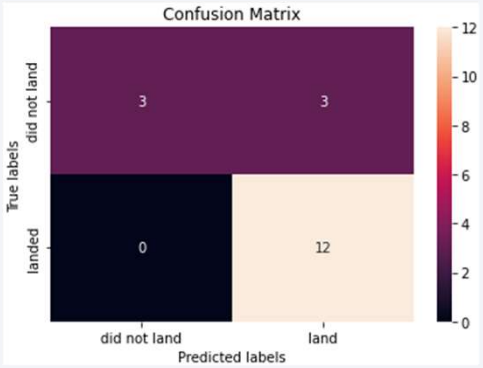
Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via APIs, SQL and Web Scraping
  - Exploratory Data Analysis (EDA) via Data Wrangling, SQL and Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning and Classification Models
- Summary of all results
  - ✓ Exploratory Data Analysis
  - ✓ Interactive Dashboard
  - ✓ Best Model for Predictive Analytics





Introduction

- Project background and context:

  By means of Data Science tools, find out whether Space X Falcon 9 rocket launch will be successful. 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. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems to address:

  1. What factors determine the successful landing of the Falcon 9 rocket?
  2. Correlations between multiple variable to predict possible outcomes.
  3. Conditions to ensure a successful landing of the rocket.

Section 1

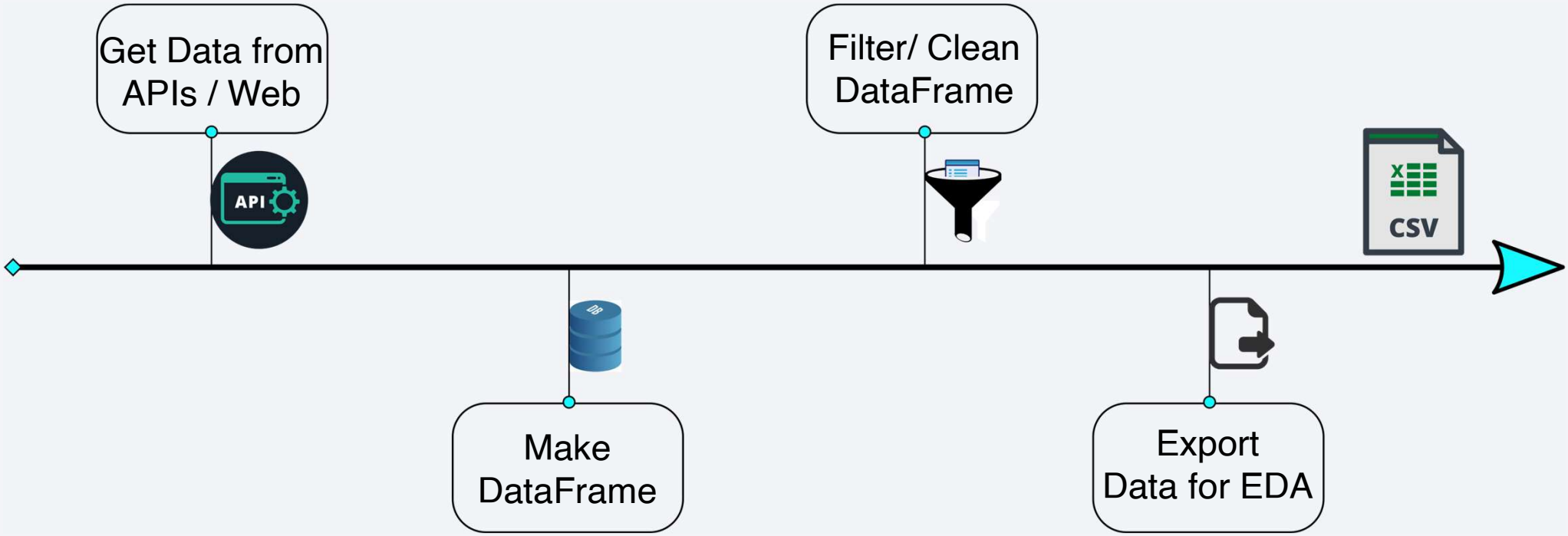
Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST APIs
  - Web Scraping (Wikipedia)
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Pattern recognition via Scatter and Bar plots
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection



Data Collection – SpaceX API

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

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

Requests data for Falcon 9 rocket via API

In [12]:
```
# Use json_normalize meethod to convert the json result in
res = requests.get(static_json_url)
static_json_df = res.json()
```

Using the dataframe `data` print the first 5 rows

In [14]:
```
# Get the head of the dataframe
data = pd.json_normalize(static_json_df)
data.head()
```

JSON to data frame

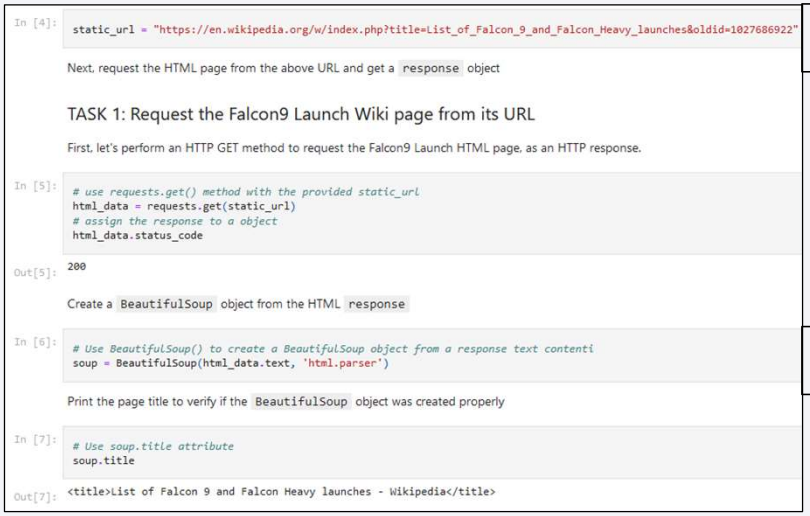
In [15]:
```
# Lets take a subset of our dataframe keeping only the features we want and the flight
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# 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 d
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)]
```

Data cleaning, and fill missing values

github:

https://github.com/Dynamicist-handa/Project_Capstone_RH/blob/master/SpaceX%20APIs%20RH.ipynb

Data Collection - Scraping



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

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

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

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

Out[5]: 200
```

Create a `BeautifulSoup` object from the HTML `response`

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text contenti
        soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

```
In [7]: # Use soup.title attribute
        soup.title

Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Requests the Falcon 9 rocket Wiki page

Create BeautifulSoup via HTML

```
In [17]:
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
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])
```

Extract columns/ variables/ attribute names from HTML header

https://github.com/Dynamicist-handa/Project_Capstone_RH/blob/master/WebScrapping%20Capstone.ipynb

Data Wrangling

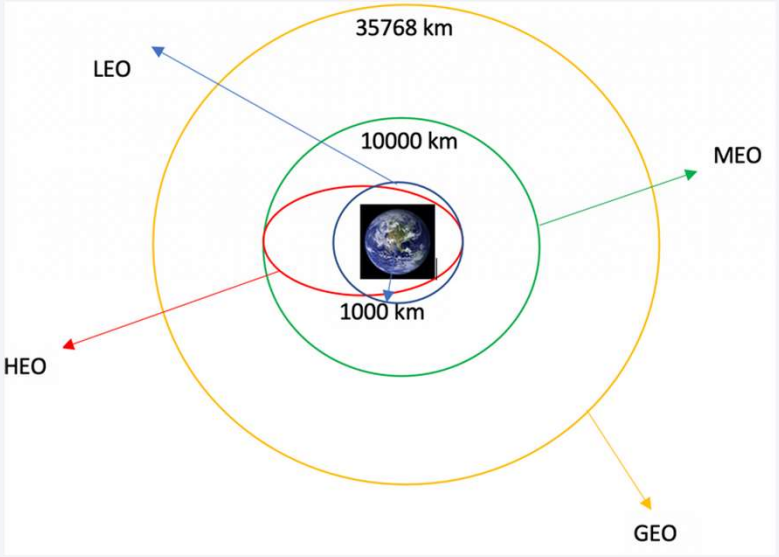
- Data Wrangling: cleaning and simplifying raw and complex data sets to facilitate Exploratory Data Analysis (EDA).

    1. Calculate the number of launches on each site.

    2. Calculate the number and occurrence of mission outcome per orbit type.

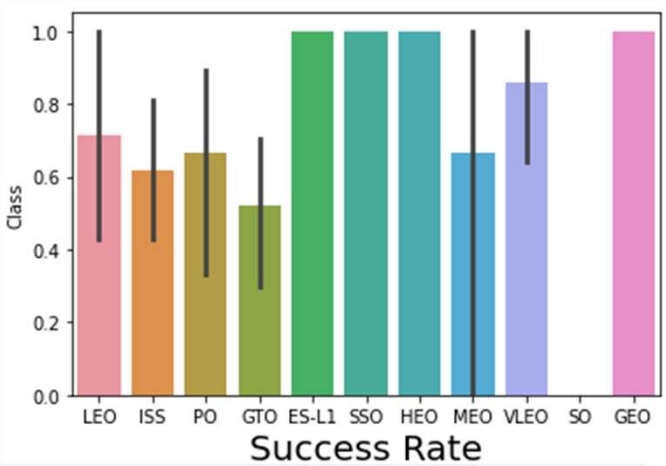
    3. Create a landing outcome label to allow further analysis, visualization, and ML modelling.

    4. Export results to a CSV data file.

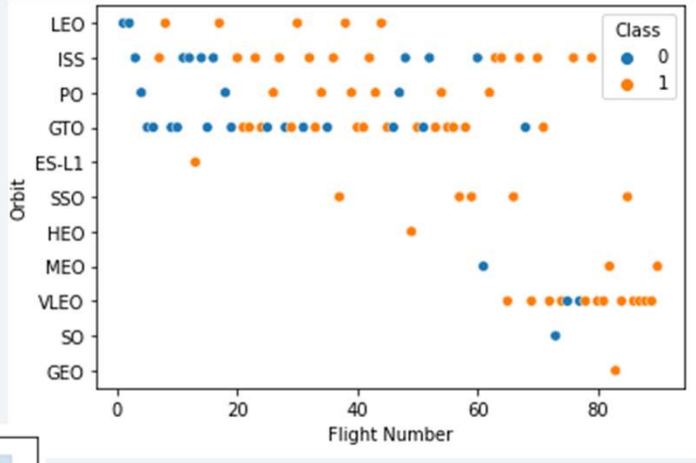


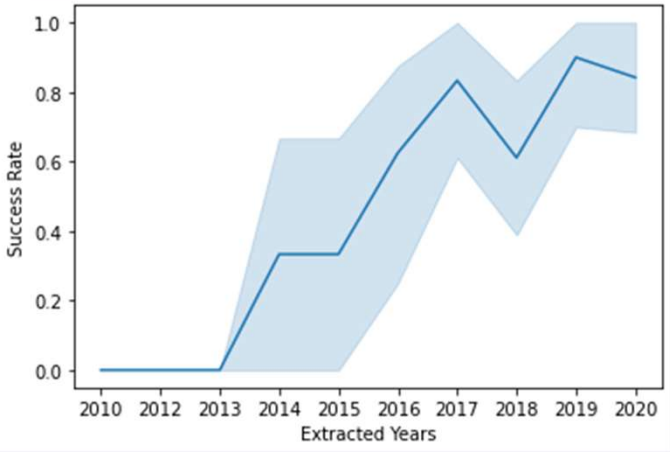
https://github.com/Dynamicist-handa/Project_Capstone_RH/blob/master/EDA_Data_Wrangling.ipynb

EDA with Data Visualization



Scatter, Bar and Line Plots: to find a correlation between variables to optimize the success rate





https://github.com/Dynamicist-handa/Project_Capstone_RH/blob/master/EDA%20Data%20Visualization%20RH.ipynb

EDA with SQL

- Established a DB2 connection and loaded the SpaceX dataset into a PostgreSQL database in the jupyter notebook.
- Applied EDA with SQL to explore and get insight from the data.
- Most relevant queries performed:
  - ✓ The names of unique launch sites in the space mission.
  - ✓ The total payload mass carried by boosters launched by NASA (CRS)
  - ✓ The average payload mass carried by booster version F9 v1.1
  - ✓ The total number of successful and failure mission outcomes.
  - ✓ The failed landing outcomes in drone ship, their booster version and launch site names.

https://github.com/Dynamicist-handa/Project_Capstone_RH/blob/master/jupyter-labs-eda-sql-coursera_sqllite(1).ipynb

Build an Interactive Map with Folium

- Marked all launch sites with markers, circles, lines to highlight the success or failure of launches for each site.
- Assigned the feature “launch outcome”’s failure or success to class 0 and class 1, respectively.
- Used color-labeled marker clusters to identify which launch sites have relatively high success rate.
- Calculated the distances between a launch site to its proximities and answered some important question, for e.g.:
  - Are launch sites near railways, highways and coastlines?
  - Do launch sites keep certain distance away from cities?

Used Haversine’s formula to calculate distances!

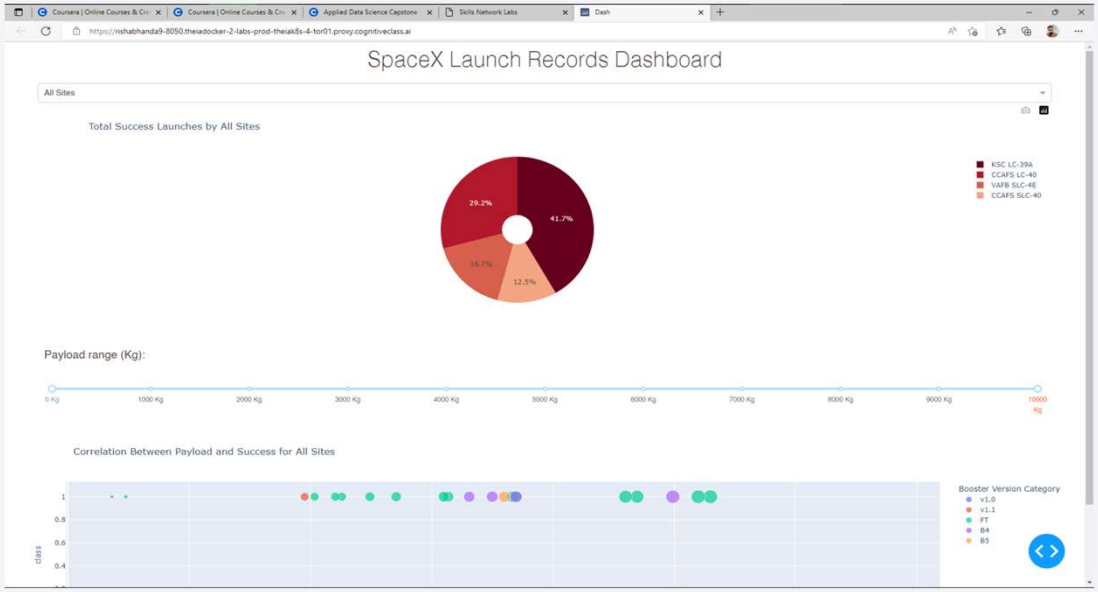
https://github.com/Dynamicist-handa/Project_Capstone_RH/blob/master/LaunchSite%20Location%20Folium%20Lab.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive Dashboard with Plotly Dash with dropdown menus.
  - Interactive Pie Chart for launch sites.
  - Scatter Plot with Outcome and Payload Mass (Kg)

  for the different booster version.

https://github.com/Dynamicist-handa/Project_Capstone_RH/blob/master/spacex_dash_app.py



Predictive Analysis (Classification)

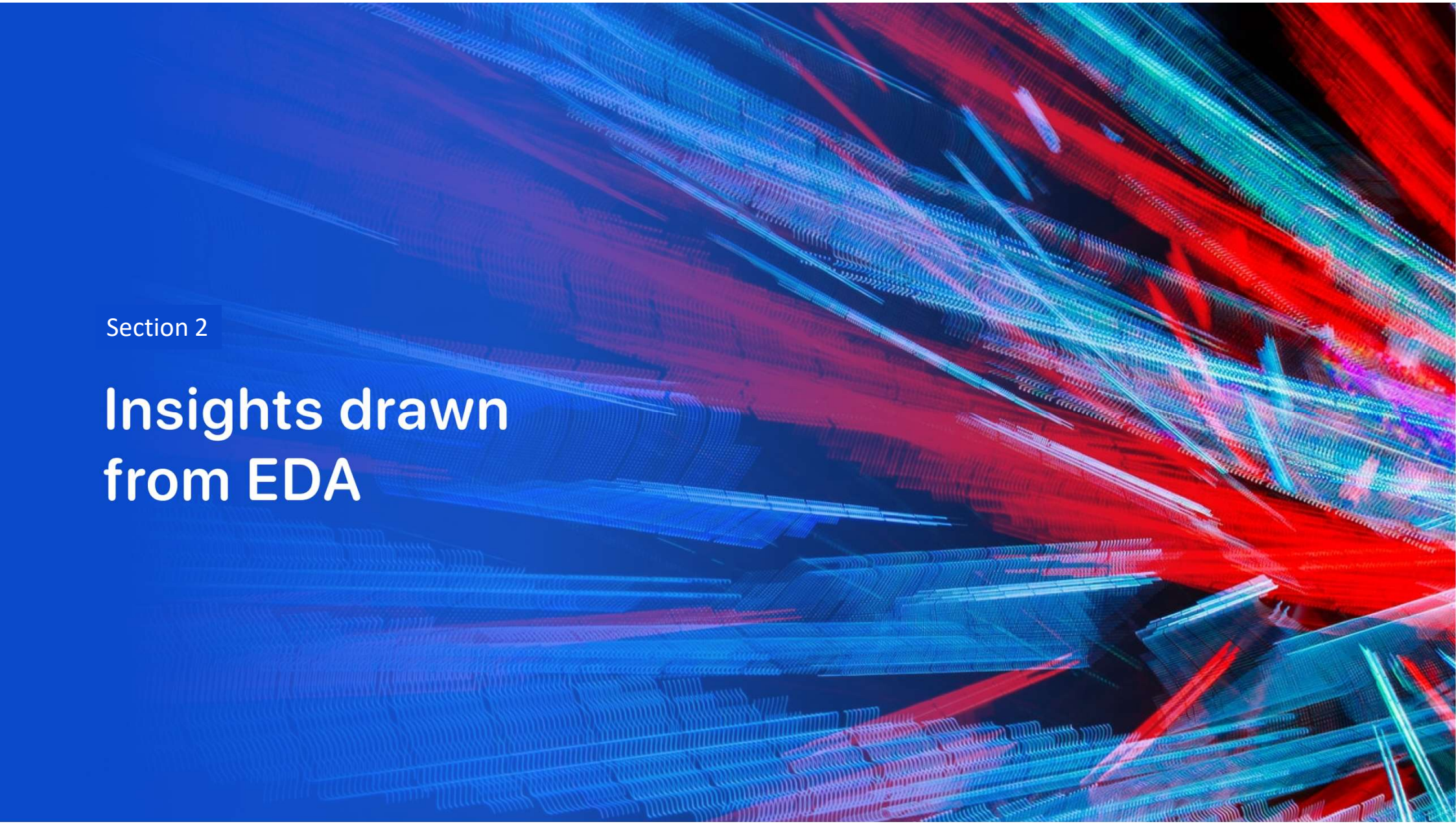
- Transformed data via numpy and pandas, to split into training and testing.
- Built machine learning models and tuned hyperparameters via GridSearchCV.
- Used accuracy as the metric for our models, and further improved the model using feature engineering.
- Thus, found the best performing classification model.

	Model	Accuracy	Prediction score
0	LogisticRegression()	0.8464285714285713	0.8333333333333334
1	SVC()	0.8482142857142856	0.8333333333333334
2	DecisionTreeClassifier()	0.8892857142857142	0.7222222222222222
3	KNeighborsClassifier()	0.8482142857142858	0.8333333333333334

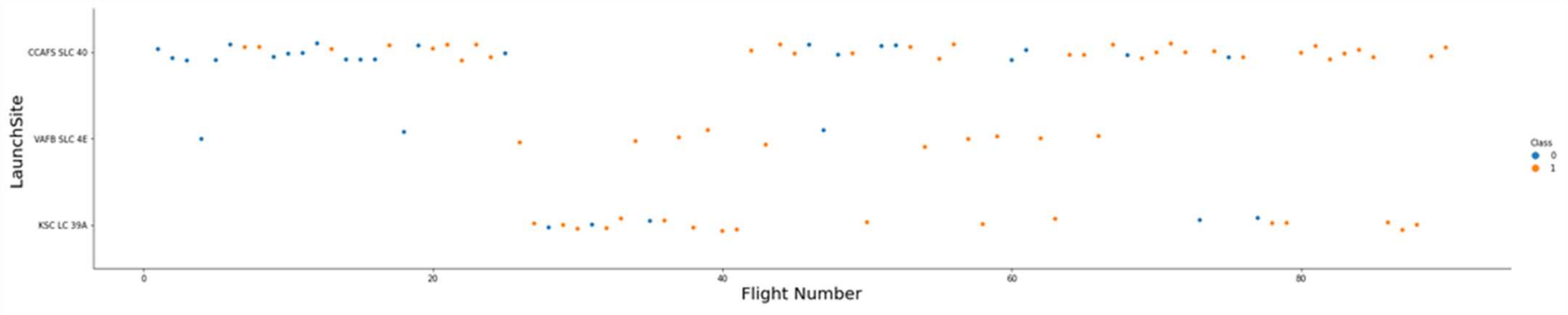
https://github.com/Dynamicist-handa/Project_Capstone_RH/blob/master/SpaceX%20ML%20Prediction%20RH.ipynb

Results

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


Section 2
Insights drawn from EDA

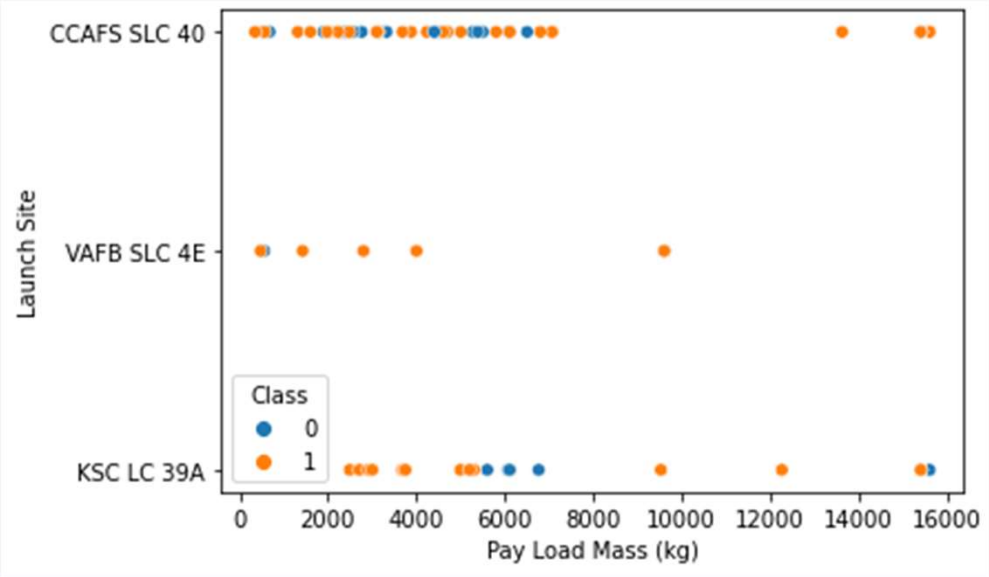
Flight Number vs. Launch Site



Insights:

- Larger the flights amount of the launch site, the greater the the success rate will be.
  - However, site CCAFS SLC40 behaves differently.

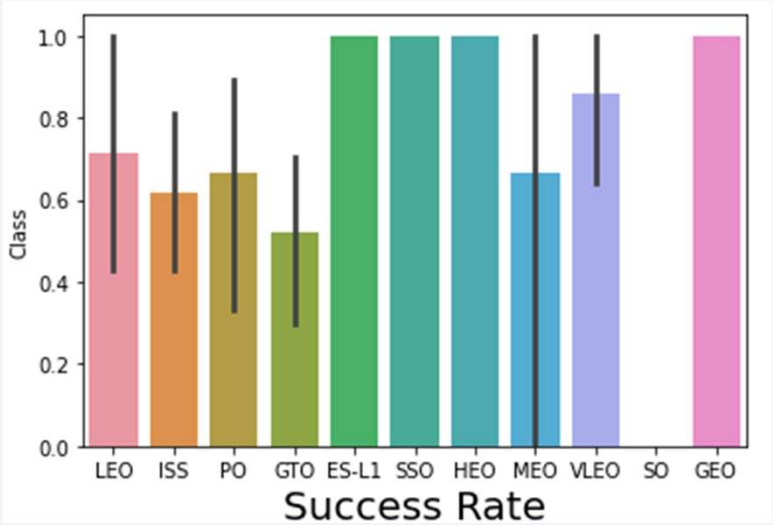
Payload vs. Launch Site



Seemingly, when the pay load mass is greater than 7000kg, the probability of the success rate will most likely increase.

Nevertheless, no clear correlation is found between the launch site and the pay load mass for the success rate.

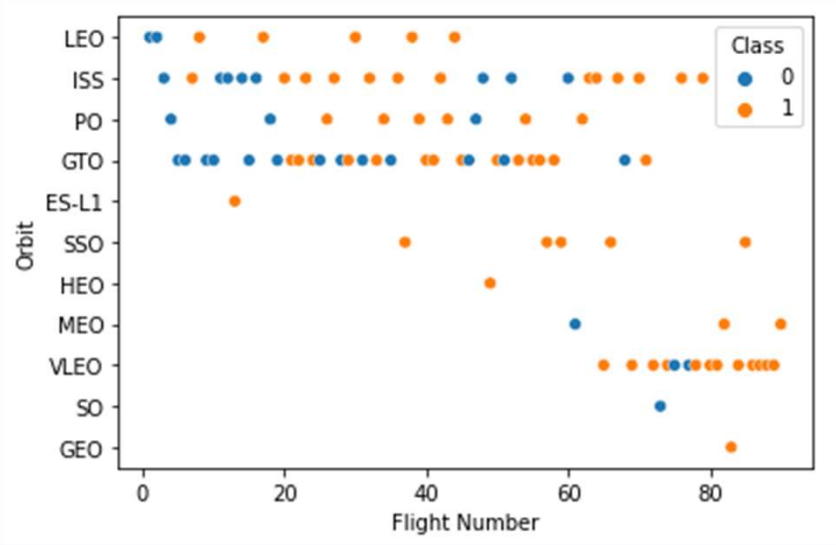
Success Rate vs. Orbit Type



Orbits with 100 % success rate:

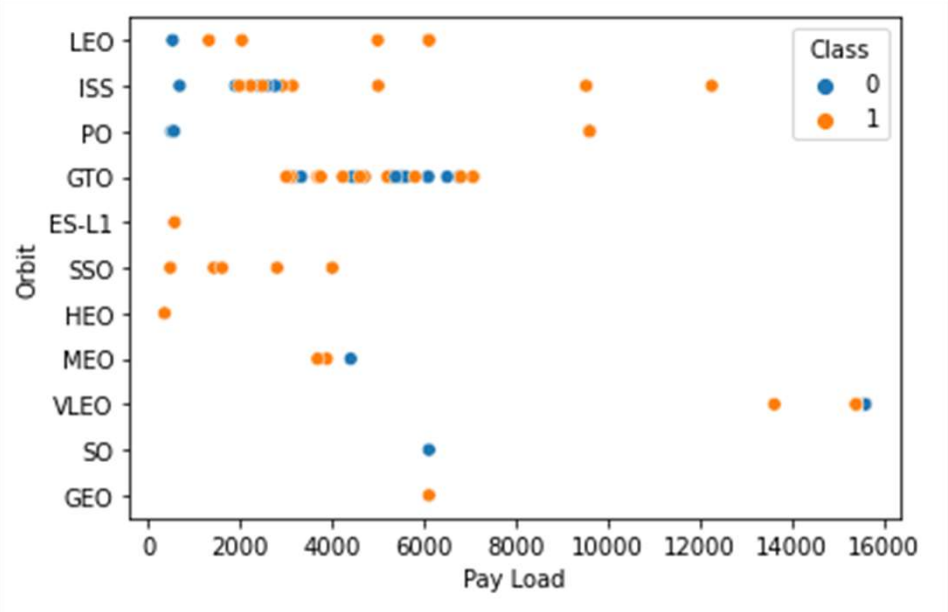
ES-L1, GEO, HEO, SSO, VLEO!!

Flight Number vs. Orbit Type



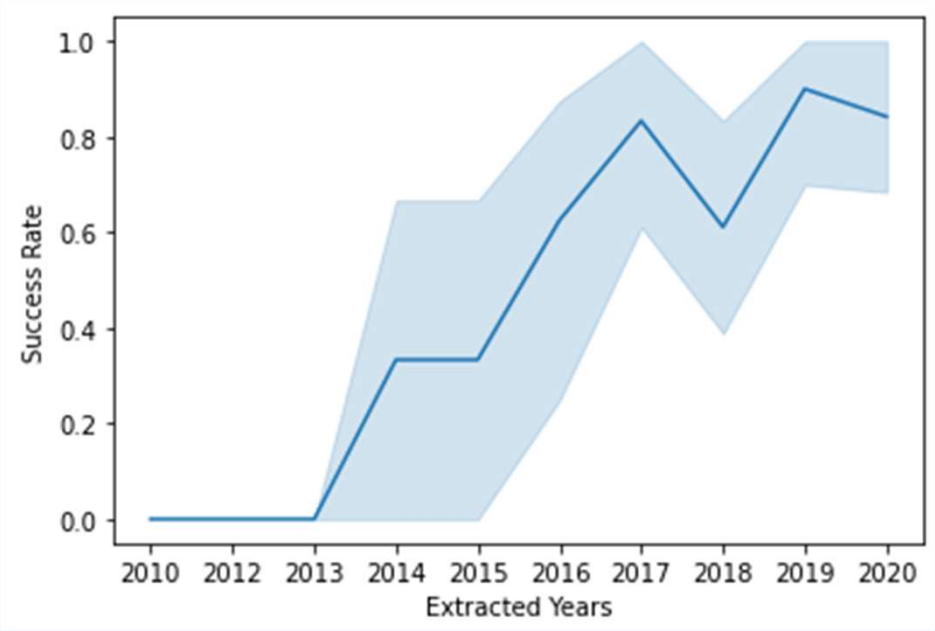
For the LEO orbit, success depends on number of flights, however, in the GTO orbit, such a dependency between flight number and the orbit doesn't exist.

Payload vs. Orbit Type



For heavy payloads, the successful landings are more profound for PO, LEO and ISS orbits.

Launch Success Yearly Trend



Success rate since 2013 has been increasing till 2020

All Launch Site Names

```
In [9]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

Out[9]:

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

Used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

Launch Site Names Begin with 'CCA'

SQL Query to display 5 records where launch sites begin with ‘CCA’

```
In [10]: %sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
 * sqlite:///my_data1.db
Done.
```

Out[10]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

```
In [73]: %sql SELECT SUM(payload_mass__kg_) FROM spacextbl WHERE payload LIKE '%CRS%'
 * sqlite:///my_data1.db
Done.
Out[73]: SUM(payload_mass__kg_)
         111268
```

Calculated the total payload carried by boosters from NASA

Average Payload Mass by F9 v1.1

```
In [74]: %sql SELECT AVG(payload_mass__kg_) FROM spacextbl WHERE booster_version LIKE '%F9 v1.1%'
 * sqlite:///my_data1.db
Done.
Out[74]: AVG(payload_mass__kg_)
         2534.6666666666665
```

Calculated the average payload mass carried by booster version F9 v1.1 as 2534.66 kg

First Successful Ground Landing Date

```
In [75]: %sql SELECT MIN(date) FROM spacextbl WHERE mission_outcome LIKE 'Success'

 * sqlite:///my_data1.db
Done.
Out[75]: MIN(date)
         01-03-2013
```

First successful landing outcome date!

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

Used the **WHERE** clause to filter for boosters and the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

In [86]:
```
%sql SELECT COUNT(*) FROM spacextbl WHERE mission_outcome LIKE '%Success%'
```

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

Out[86]:

COUNT(*)
100

In [87]:
```
%sql SELECT COUNT(*) FROM spacextbl WHERE mission_outcome LIKE '%Failure%'
```

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

Out[87]:

COUNT(*)
1

In [30]:
```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Total Number of Successful and Failure Mission" FROM SPACEXTBL \
WHERE MISSION_OUTCOME LIKE 'Success%' OR MISSION_OUTCOME LIKE 'Failure%';
```

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

Out[30]:

Total Number of Successful and Failure Mission
101

In [32]:
```
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", \
    sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" \
FROM SPACEXTBL;
```

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

Out[32]:

Successful Mission	Failure Mission
100	1

Boosters Carried Maximum Payload

In [89]:
```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

Out[89]:

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

Determined the booster that carried the maximum payload using a subquery in the **WHERE** clause and the **MAX** function.

2015 Launch Records

```
In [103]:
%%sql
SELECT substr(Date, 4, 2) as month, booster_version, LAUNCH_SITE, "Landing _Outcome"
from SPACEXTBL where "Landing _Outcome"
='Failure (drone ship)' and substr(Date,7,4)='2015'

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

Out[103]:

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

Used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

In [99]:
```
%%sql
SELECT "Landing _Outcome",count("Landing _Outcome")as LANDING_OUTCOME_COUNT
from SPACEXTBL where  DATE between '04-06-2010' and '20-03-2017'
group by "Landing _Outcome" order by count("Landing _Outcome") desc
```

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

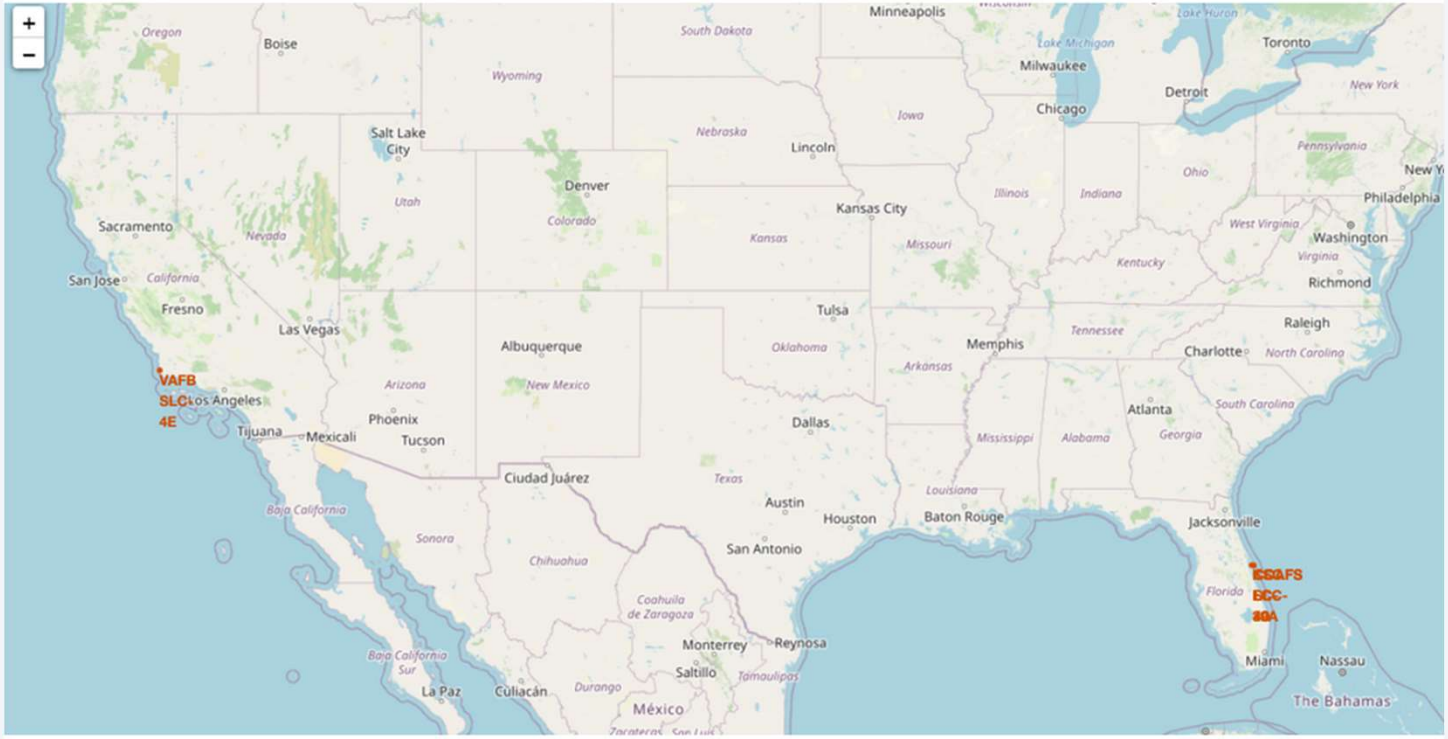
Out[99]:

Landing _Outcome	LANDING_OUTCOME_COUNT
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

Selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20 and Applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

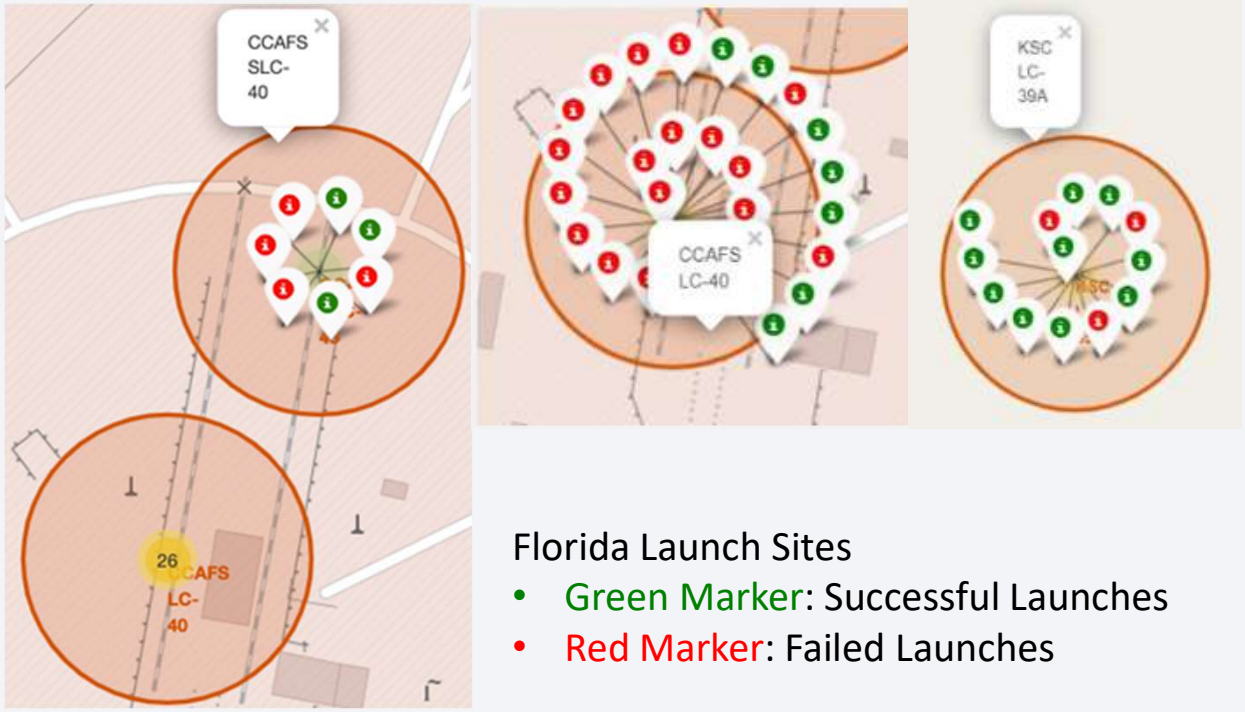

Section 3
Launch Sites
Proximities Analysis

<Folium Map Screenshot 1>



We can see that all the SpaceX launch sites are located inside the United States, in proximity to the Equator Line and very close proximity to the coast!!

<Folium Map Screenshot 2>

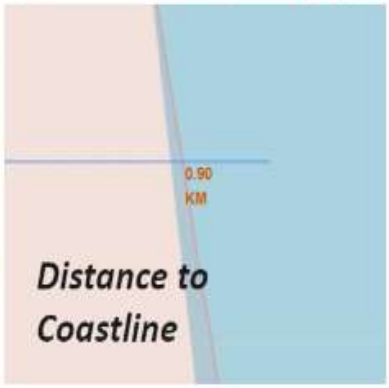


Florida Launch Sites

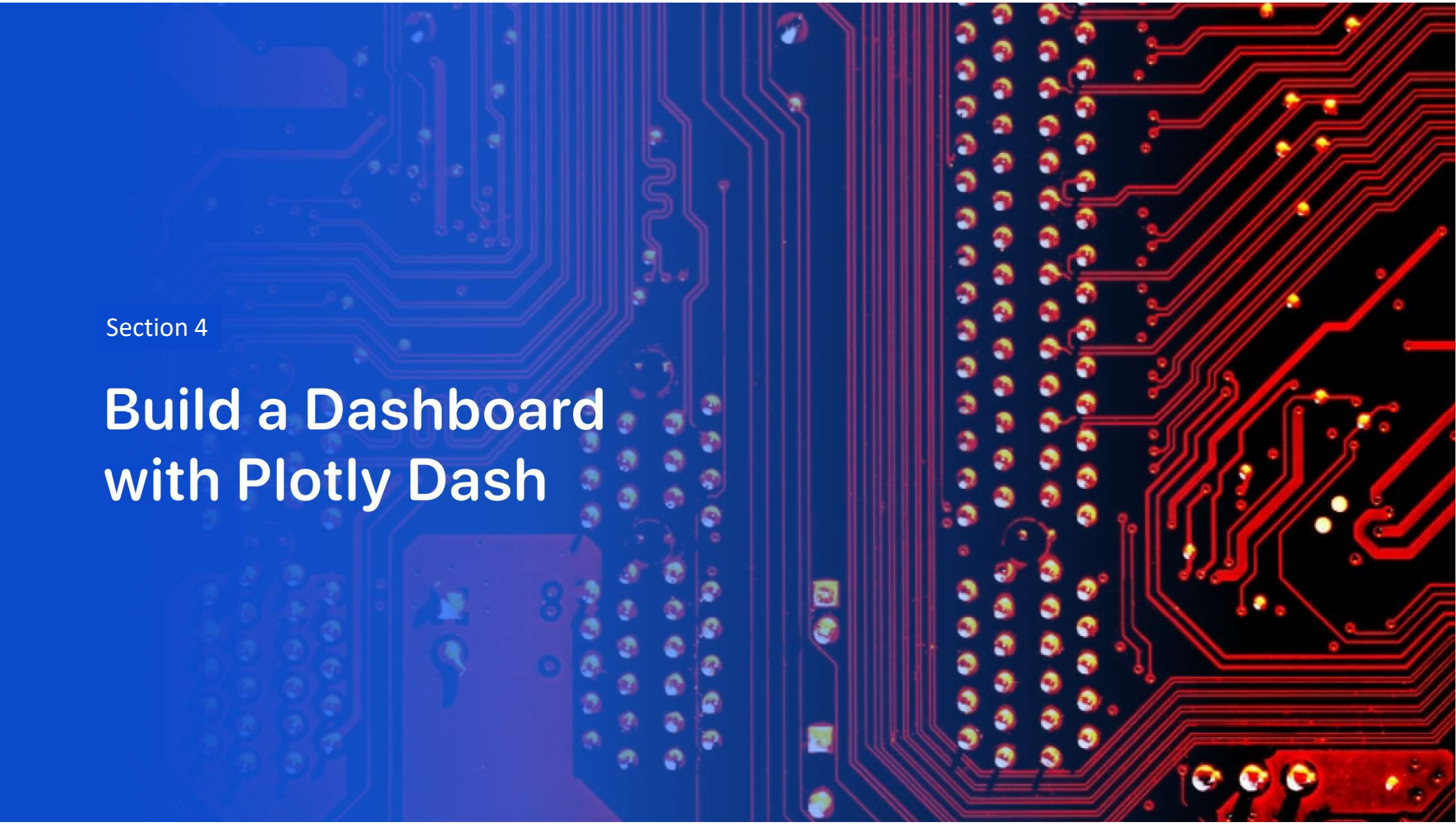
- Green Marker: Successful Launches
- Red Marker: Failed Launches

California Launch Site

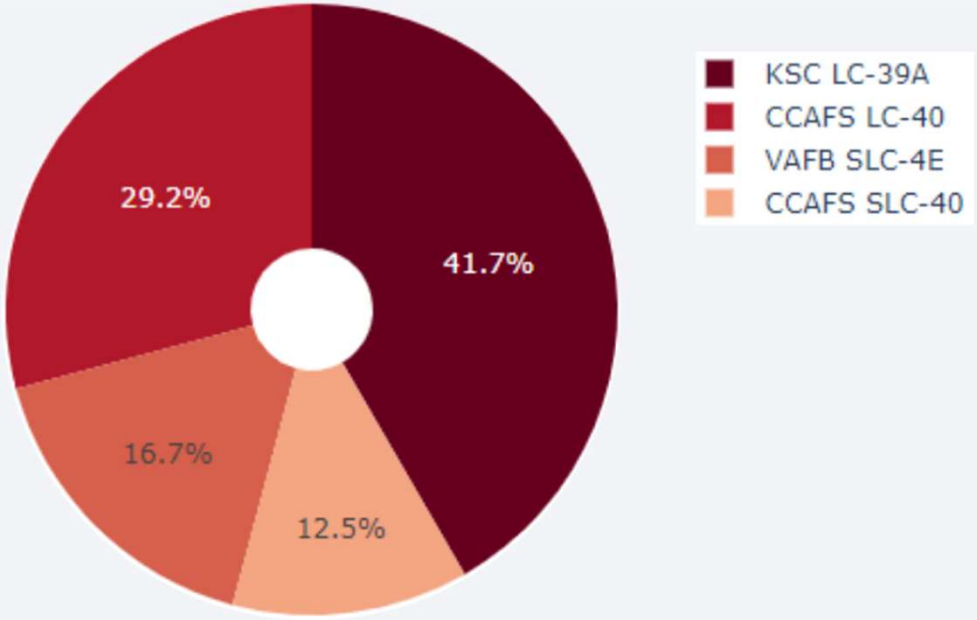
<Folium Map Screenshot 3>



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

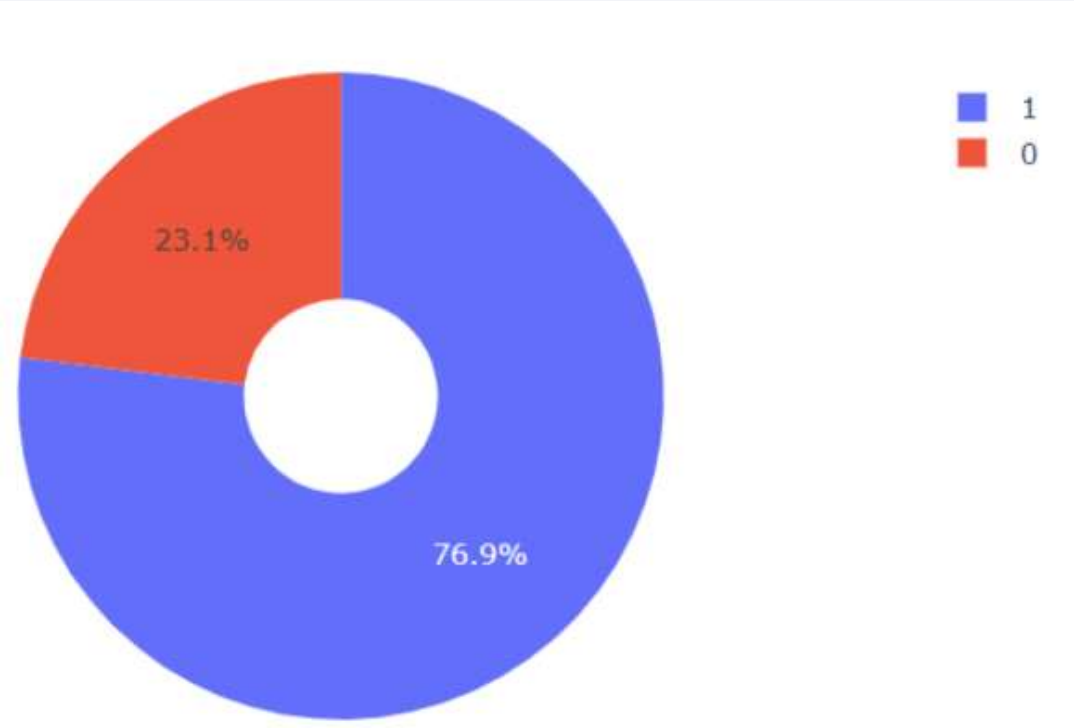

Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



KSC LC-39A had the most successful launches!

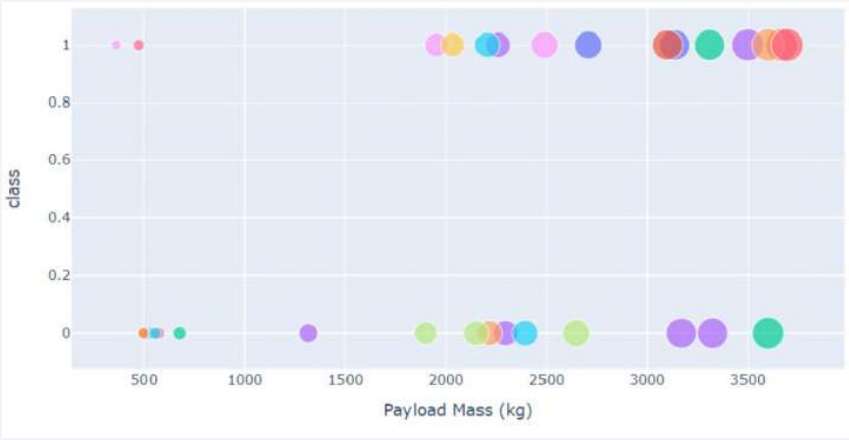
<Dashboard Screenshot 2>



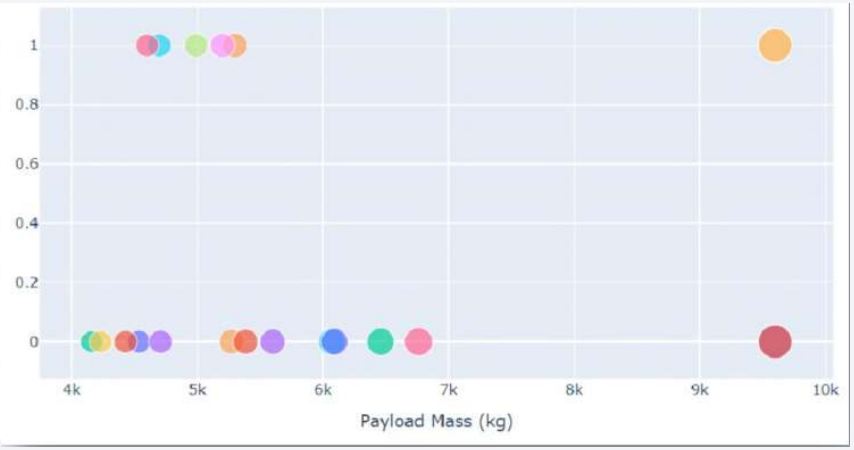
KSC LC-39A is at 76.9 % success rate with a 23.1 % of chances to fail.

<Dashboard Screenshot 3>

Low Weighted Payload 0kg – 4000kg



Heavy Weighted Payload 4000kg – 10000kg



Section 5

Predictive Analysis (Classification)

Classification Accuracy

In [31]:

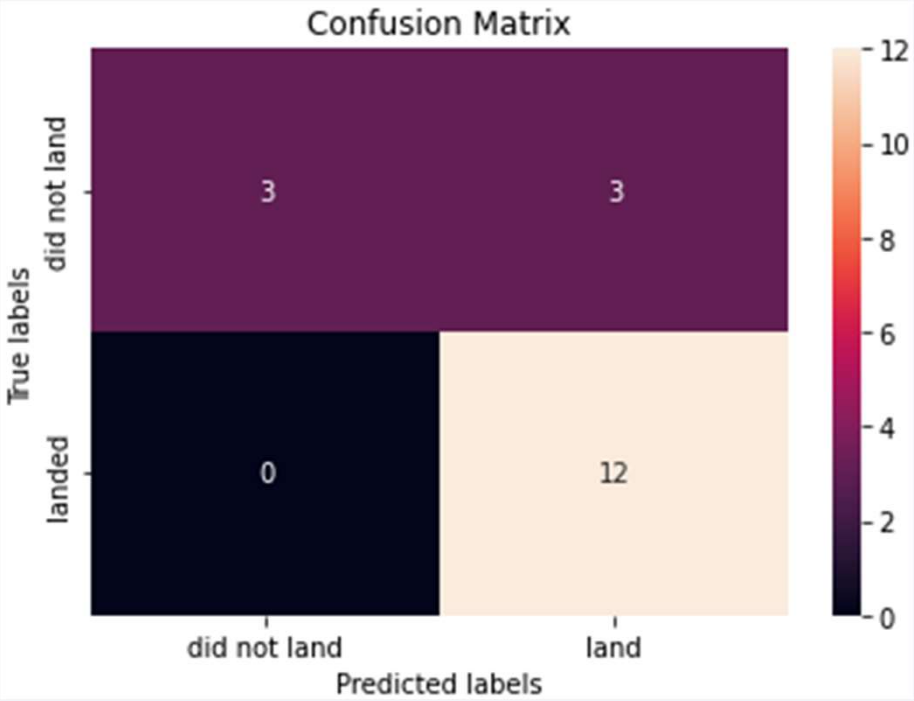
```
machine_learning_models = [logreg_cv, svm_cv, tree_cv, knn_cv]
results = []
for model in machine_learning_models:
    model_dictionary = {'Model': str(model.estimator), 'Accuracy': str(model.best_score_), 'Prediction score': str(model.score(X_test, Y_test))}
    results.append(model_dictionary)
results_df = pd.DataFrame(results)

results_df
```

Out[31]:

	Model	Accuracy	Prediction score
0	LogisticRegression()	0.8464285714285713	0.8333333333333334
1	SVC()	0.8482142857142856	0.8333333333333334
2	DecisionTreeClassifier()	0.8892857142857142	0.7222222222222222
3	KNeighborsClassifier()	0.8482142857142858	0.8333333333333334

The decision tree classifier is the model with the highest classification accuracy!!

Confusion Matrix



The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.

Conclusions

- ✓ The larger the flight amount at a launch site, the greater the success rate at a launch site.
- ✓ Launch success rate began to increase in 2013 till 2020.
- ✓ Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- ✓ KSC LC-39A had the most successful launches in contrast to other available sites.
- ✓ The Decision tree classifier was found to be the best machine learning model for aforesaid predictions.

Thank you!