Winning Space Race with Data Science

Eyob Hailu
May 3, 2023

Outline

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

Executive Summary

Summary of Data Analysis Methodologies

- Data Collection: API and Web Scraping
- Data Wrangling: Ensuring data quality and integrity
- Exploratory Data Analysis: Using SQL and Data Visualization techniques
- Interactive Visual Analytics: Utilizing Folium for in-depth insights
- Machine Learning Prediction: Applying algorithms to predict future trends

Results:

- Exploratory Data Analysis: Identified patterns, trends, and anomalies
- Interactive Analytics: Screenshots provide a more in-depth look at the data
- Predictive Analytics: Used machine learning to predict future trends

Introduction

- Project background and context

Space X offers Falcon 9 rocket launches at a lower cost than other providers because they reuse the first stage. To determine the cost of a launch, it's essential to predict if the first stage will land successfully. Therefore, the goal of this project is to develop a machine learning pipeline that can predict if the first stage will land successfully. This information will help in bidding against Space X for a rocket launch.

- Problems you want to find answers
  - What factors contribute to a successful rocket landing?
  - How do these factors interact to influence the success rate of a landing?
  - What are the necessary operating conditions required for a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected from SpaceX REST API as well as web scraping their Wikipedia page
- Perform data wrangling
  - Data cleaning, and replacement of missing data was done
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

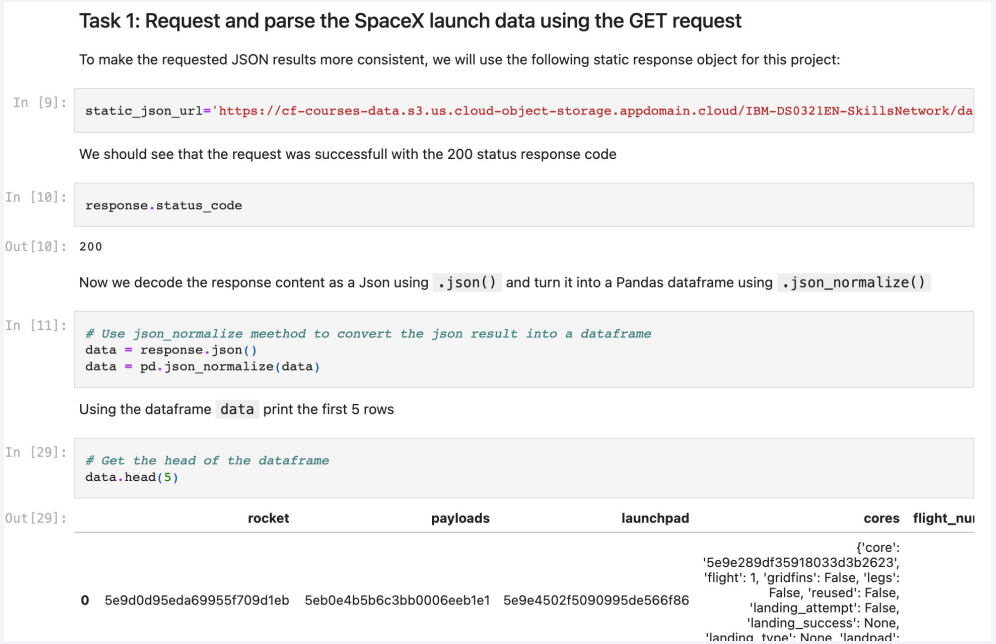
Data Collection

- The data was obtained using a variety of methods. The following steps were taken:
  - To collect the data, a get request was sent to the SpaceX API.
  - The response content was decoded as Json using the .json() function, and then converted to a pandas dataframe using .json_normalize().
  - The data was cleaned, and any missing values were identified and filled in as necessary.
  - Additionally, web scraping was performed on Wikipedia to retrieve Falcon 9 launch records using BeautifulSoup.
  - The goal of this process was to extract the launch records as an HTML table, parse the table, and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- GET request that was made to the API, then parsed the json response, and placed it into a dataframe

- https://github.com/eyobtewe/spacemission/blob/master/spacex%20data%20collection.ipynb



Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

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

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

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

Out[10]: 200

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

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

Using the dataframe `data` print the first 5 rows

In [29]:
```
# Get the head of the dataframe
data.head(5)
```

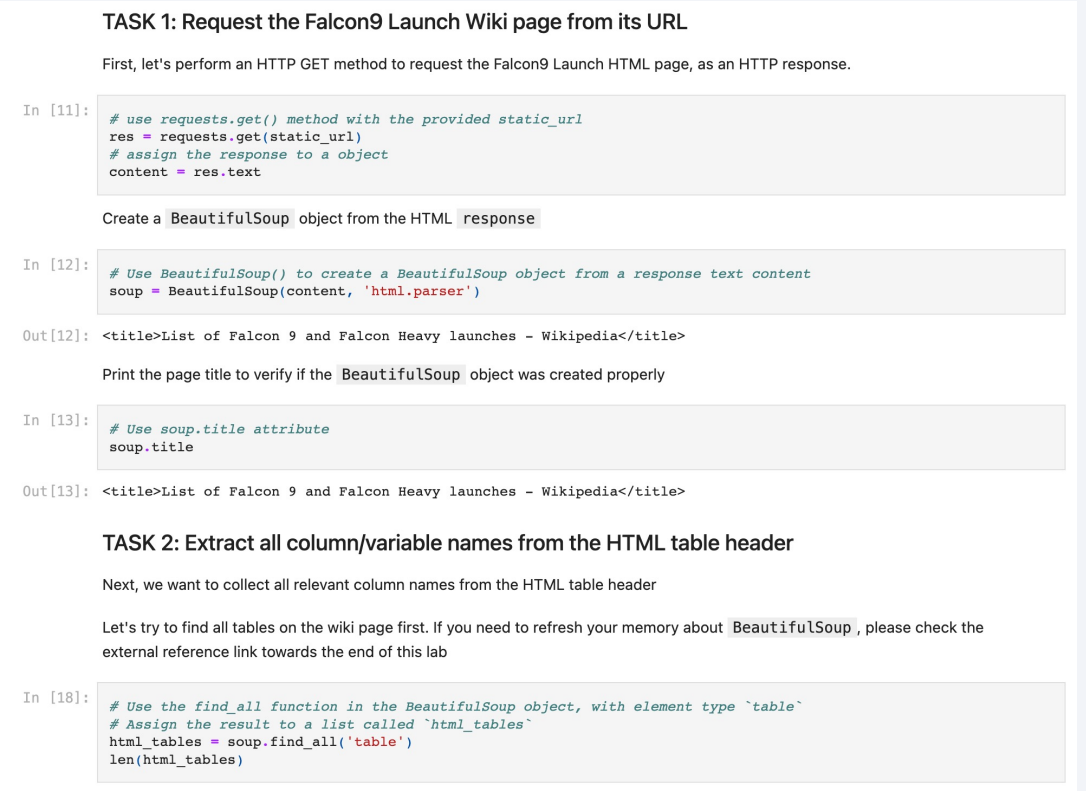
Out[29]:

	rocket	payloads	launchpad	cores	flight_nu
0	5e9d0d95eda69955f709d1eb	5eb0e4b5b6c3bb0006eeb1e1	5e9e4502f5090995de566f86	{'core': '5e9e289df35918033d3b2623', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad':	

Data Collection - Scraping

- Scraped the Wikipedia page using BeautifulSoup, parsed the content, and converted it to pandas dataframe

- https://github.com/eyobtewe/spacemission/blob/master/spacex%20web%20scraping.ipynb



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 [11]:
```
# use requests.get() method with the provided static_url
res = requests.get(static_url)
# assign the response to a object
content = res.text
```

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

In [12]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(content, 'html.parser')
```

Out[12]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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

In [13]:
```
# Use soup.title attribute
soup.title
```

Out[13]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [18]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
len(html_tables)
```

Data Wrangling

1. Counted launches per site
2. Determined frequency of each orbit type
3. Tabulated mission outcomes for each orbit type
4. Generated a landing outcome label based on the outcome column.

- https://github.com/eyobtewe/spacemission/blob/master/Spacex%20Exploratory%20Data%20Analysis.ipynb

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

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

We create a set of outcomes where the second stage did not land successfully:

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

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

TASK 4: Create a landing outcome label from Outcome column

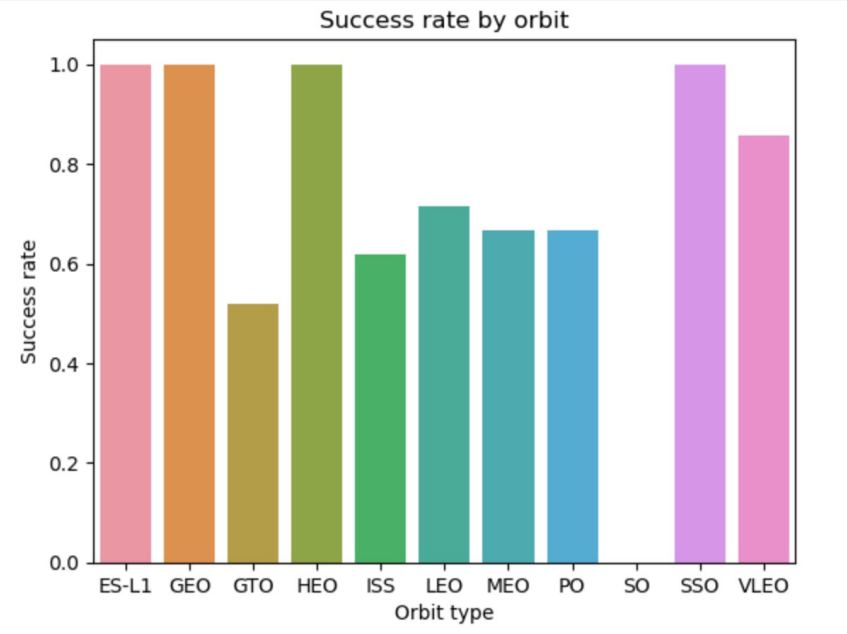
Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

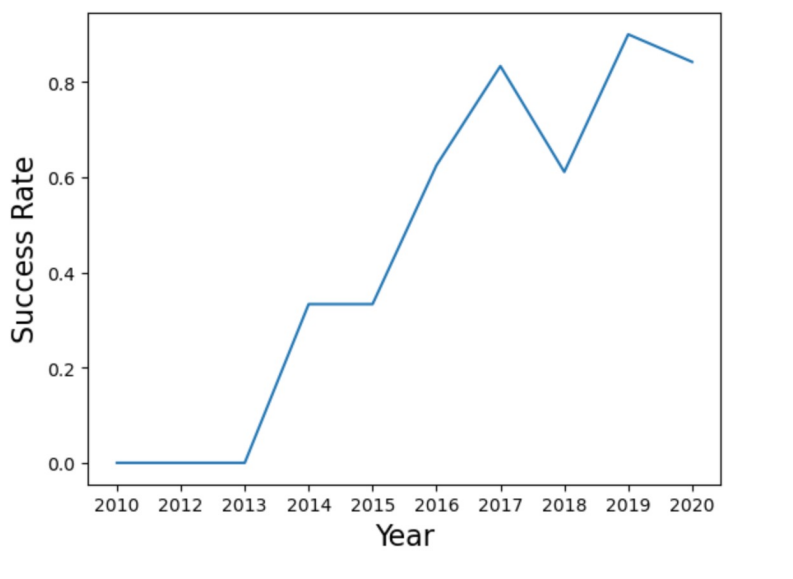
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

landing_class = [0 if o in bad_outcomes else 1 for o in df['Outcome']]
# landing_class
```

EDA with Data Visualization

- As we can see on the right, the success rate since 2013 kept increasing till 2020





- https://github.com/eyobtewe/spacemission/blob/master/Spacex%20Exploratory%20Data%20Analysis%20and%20Visualization.ipynb

EDA with SQL

- Loaded the SpaceX dataset into a sqlite
- Some of the queries that were done
  1. Identify the distinct launch sites involved in the space mission.
  2. Calculate the cumulative payload mass transported by NASA's (CRS) booster launches.
  3. Compute the average payload mass carried by booster version F9 v1.1.
  4. Determine the total count of successful and unsuccessful mission outcomes.
  5. List the instances of unsuccessful landing outcomes on drone ships, including the booster version and launch site names..
- https://github.com/eyobtewe/spacemission/blob/master/Spacex%20Exploratory%20Data%20Analysis%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Marked launch sites and added map objects (markers, circles, lines) to indicate launch success or failure on a folium map.
- Assigned launch outcomes to classes 0 and 1 (0 for failure, 1 for success).
- Used color-labeled marker clusters to identify high success rate launch sites.
- Calculated distances between launch sites and nearby features such as railways, highways, coastlines, and cities.
- Analyzed the data to answer questions such as whether launch sites are located near railways, highways, and coastlines, and whether they keep a certain distance away from cities.
- https://github.com/eyobtewe/spacemission/blob/master/Space_%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

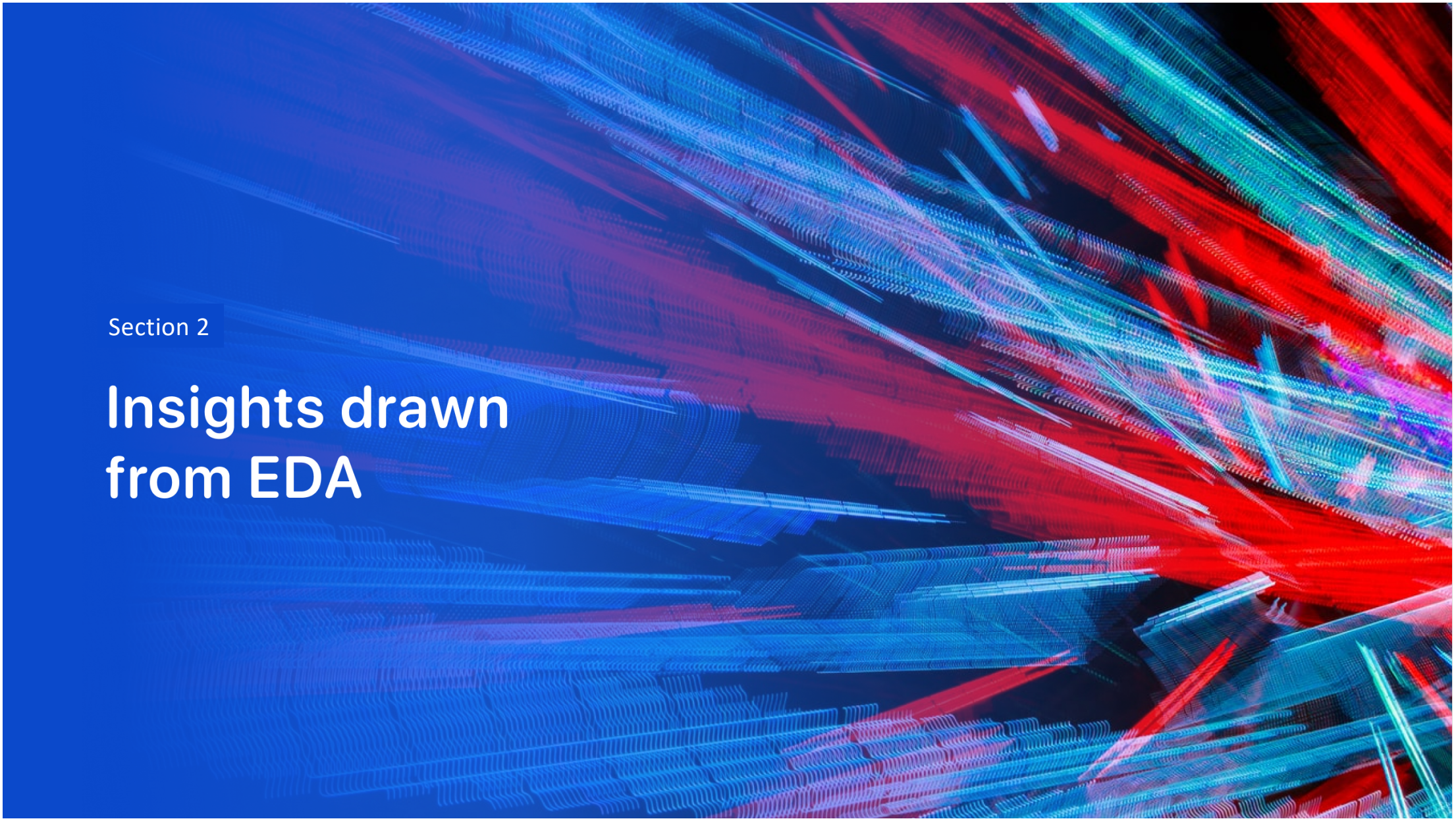
- Developed an interactive dashboard using Plotly Dash
- Generated pie charts to display the total number of launches from specific sites
- Created scatter plots to analyze the relationship between Outcome and Payload Mass (Kg) for different booster versions
- https://github.com/eyobtewe/spacemission/blob/master/Space%20Dashboard%20with%20Plotly.py

Predictive Analysis (Classification)

- Data loading, transformation, and splitting: Numpy and Pandas were used to load the data, which was then transformed and split into training and testing subsets.
- Model building and hyperparameter tuning: Different machine learning models were built and their hyperparameters were tuned using GridSearchCV.
- Model evaluation: The accuracy metric was used to evaluate the performance of the models, and feature engineering and algorithm tuning were applied to improve their performance.
- Best model selection: The decision tree was identified as the best performing classification model after testing multiple models.
- https://github.com/eyobtewe/spacemission/blob/master/Space%20Machine%20Learning.ipynb

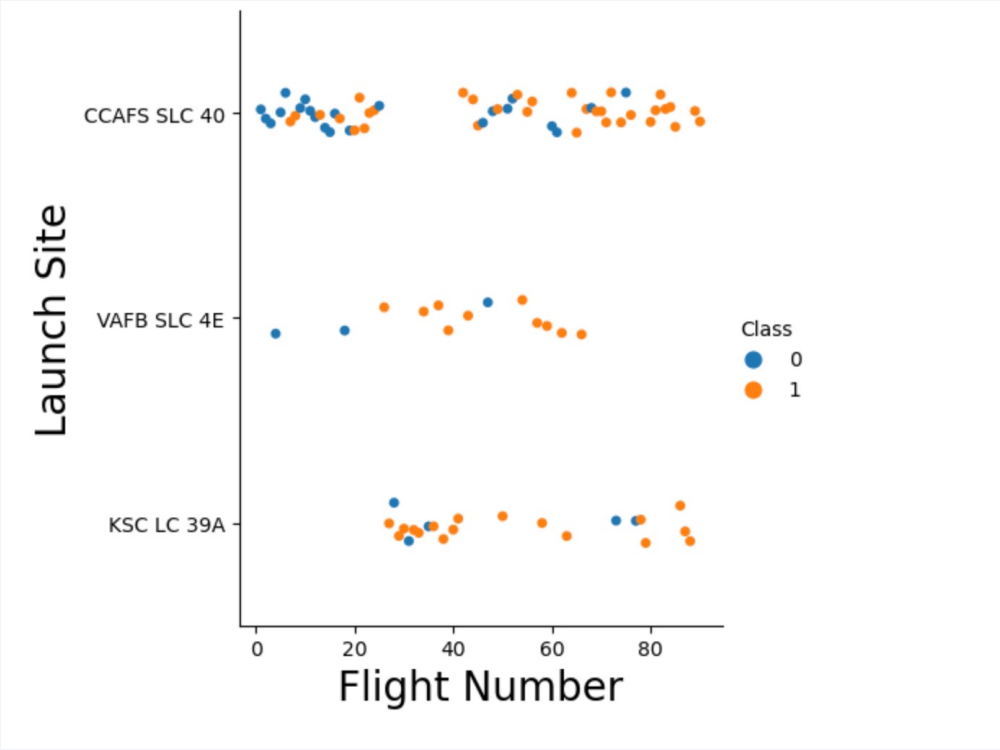
Results

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


Section 2
Insights drawn from EDA

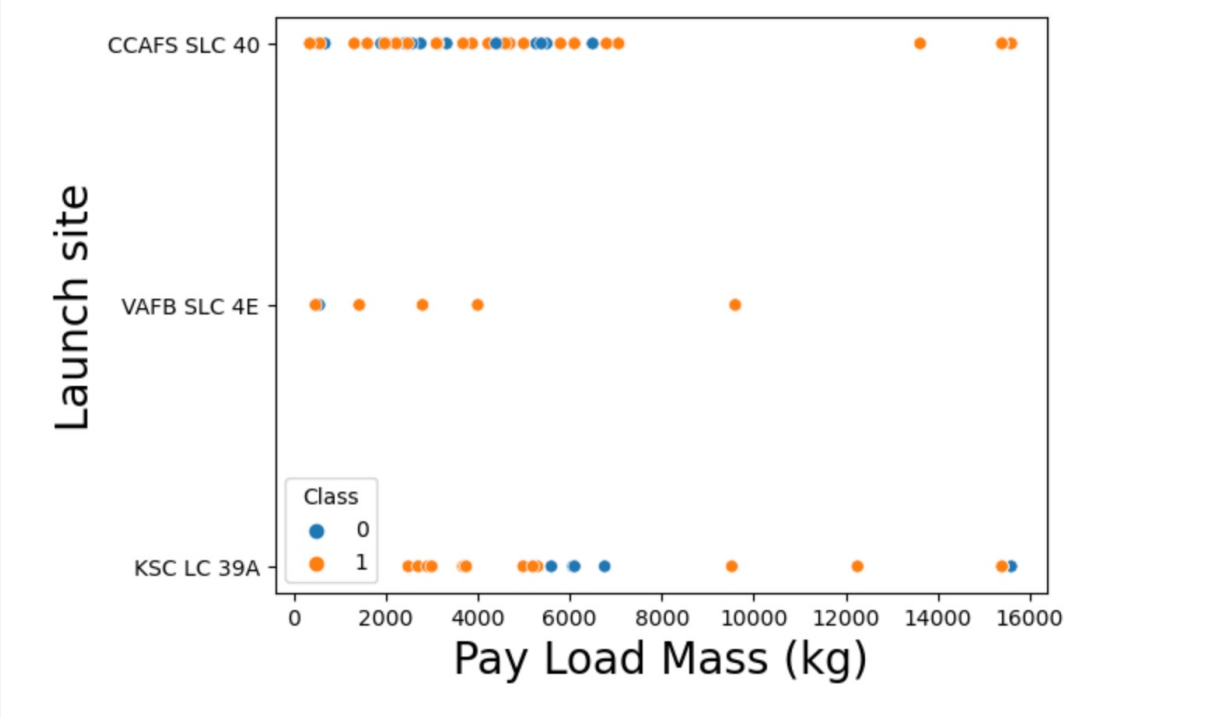
Flight Number vs. Launch Site

- From the graph we can see the larger the flight number the better the success rate



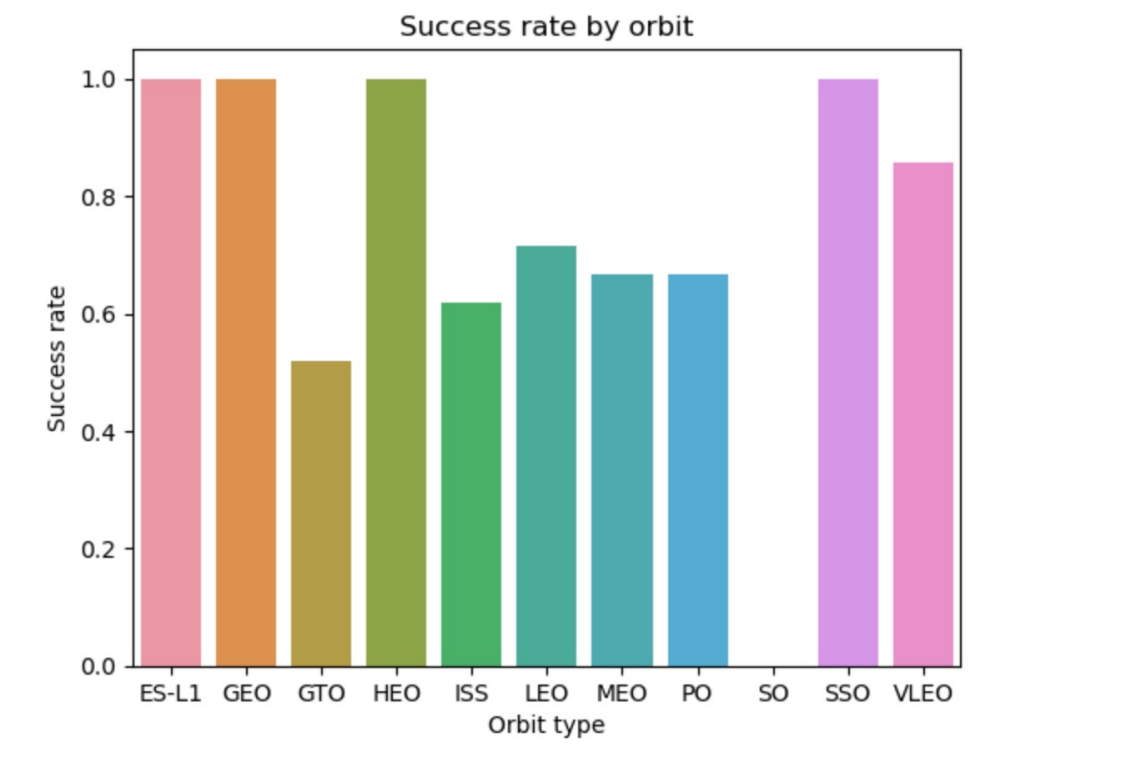
Payload vs. Launch Site

- From the graph we can see the higher the payload the better the success rate



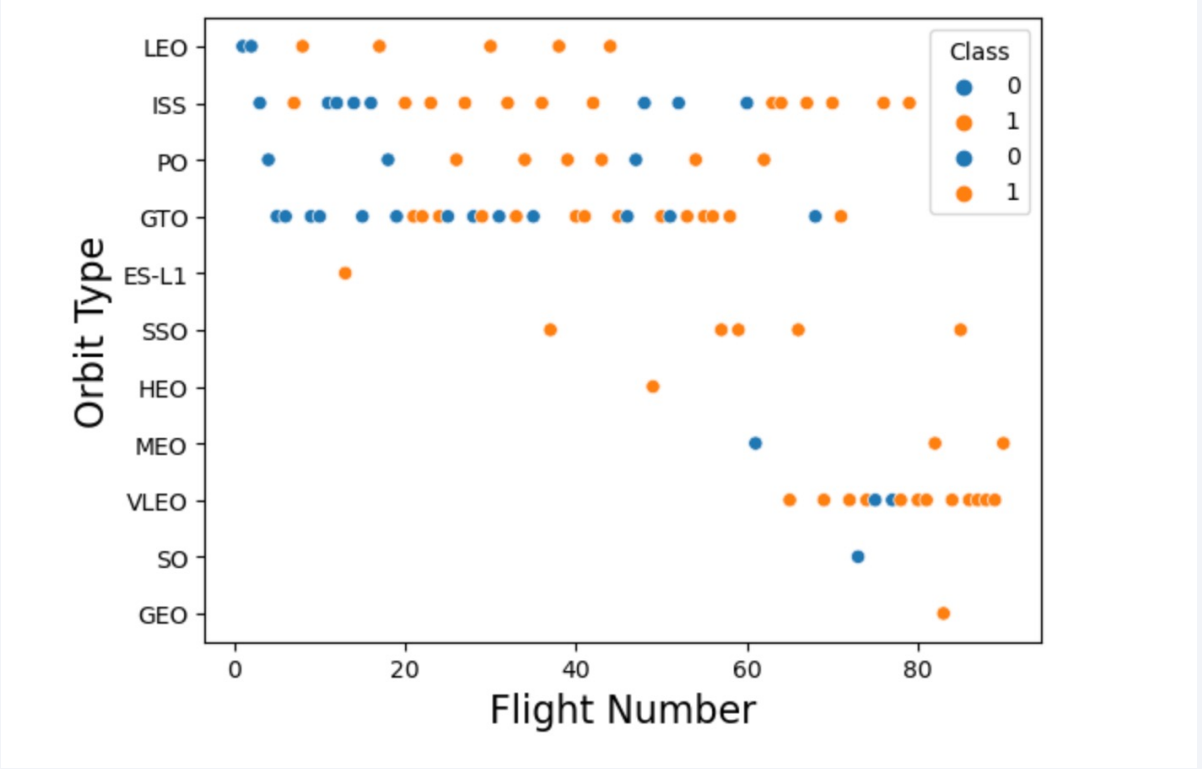
Success Rate vs. Orbit Type

- Orbits HEO, GEO, ES-L1, SSO & VLEO have high success rates
- While GTO, ISS, LEO, MEO, PO have significantly lower success rates



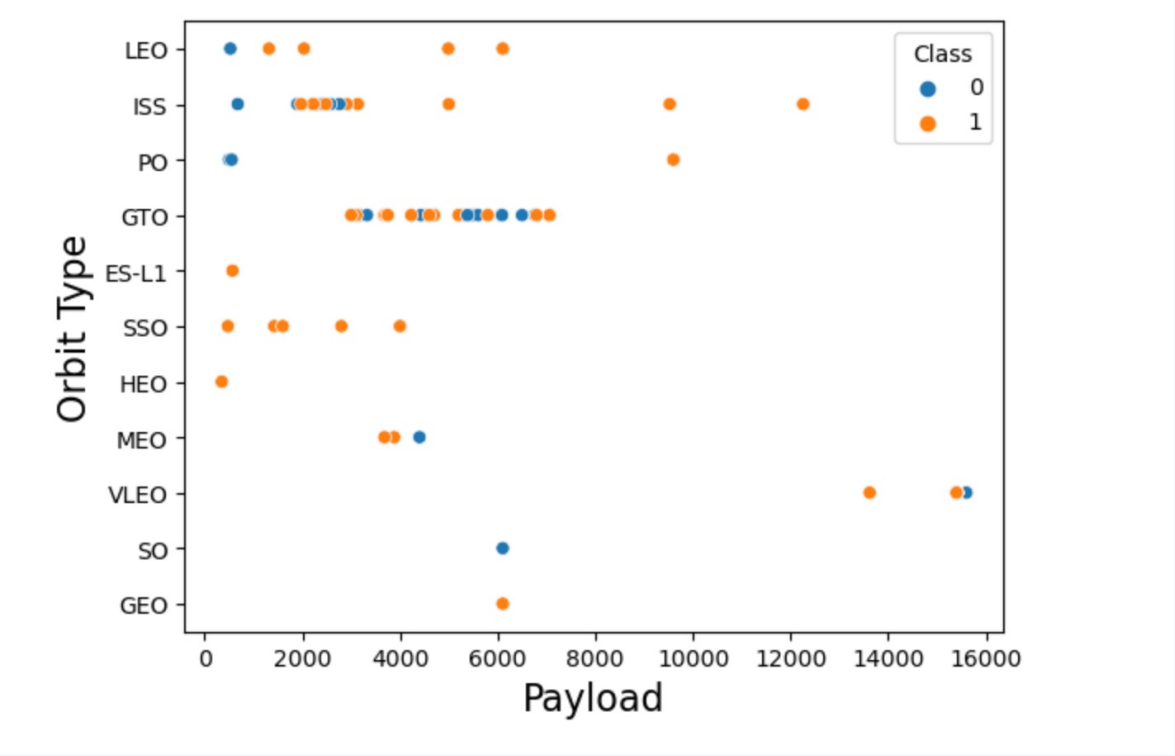
Flight Number vs. Orbit Type

- For LEO success rate is higher in the larger Flight Number range.
- For the rest there is no clear relationship



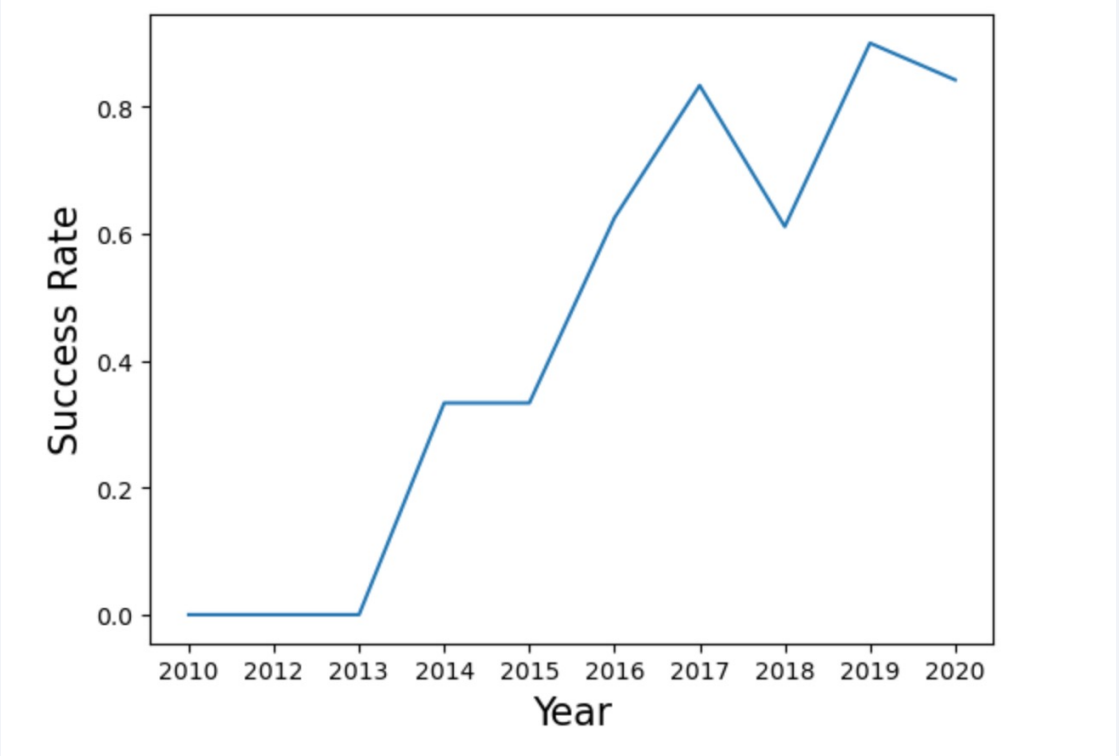
Payload vs. Orbit Type

- For heavier payloads the successful rate is higher for Polar, LEO and ISS.
- However for GTO we cannot distinguish this.



Launch Success Yearly Trend

- We can observe that the success rate since 2013 kept increasing till 2020



All Launch Site Names

- This SQL query retrieves a list of unique launch sites from the "SPACEXTBL" table.

Task 1

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

In [8]:
```
%%sql
SELECT DISTINCT(Launch_Site)
FROM SPACEXTBL
```

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

Out[8]:

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

Launch Site Names Begin with 'CCA'

- This SQL query retrieves the first 5 rows of data from the SPACEXTBL table where the Launch_Site column starts with 'CCA'.

Task 2

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

```
%%sql
SELECT *
FROM SPACEXTBL
WHERE Launch_Site LIKE 'CCA%'
LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landin _Outcom
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failu (parachut
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failu (parachut
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attem
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attem
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attem

Total Payload Mass

- This query sums up the payload mass in kilograms for customers with names starting with "NASA (CRS)" in the SPACEXTBL table.

Task 3

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

```
%%sql
SELECT Customer, SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
GROUP BY Customer
HAVING Customer LIKE 'NASA (CRS)%'
```

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

Customer	SUM(PAYLOAD_MASS__KG_)
NASA (CRS)	45596
NASA (CRS), Kacific 1	2617

Average Payload Mass by F9 v1.1

- This SQL query calculates the average payload mass for booster version 'F9 v1.1' in the SPACEXTBL table.

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

```
%%sql
SELECT Booster_Version, AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
GROUP BY Booster_Version
HAVING Booster_Version LIKE 'F9 v1.1'
```

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

Booster_Version	AVG(PAYLOAD_MASS__KG_)
F9 v1.1	2928.4

First Successful Ground Landing Date

- This SQL query finds the earliest successful landing date from the SpaceX table.

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

Hint:Use min function

```
%%sql
SELECT MIN(DATE)
FROM SPACEXTBL
WHERE `Landing _Outcome` LIKE '%Success%'
```

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

MIN(DATE)
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

- This SQL query selects the booster version from a SpaceX table where the landing outcome was a success on a drone ship and the payload mass was between 4000 and 6000 kg.

List the names of the boosters which have success in drone ship and have payload m

```
%%sql
SELECT Booster_Version
FROM SPACEXTBL
WHERE `Landing _Outcome` LIKE '%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

- This SQL query groups the SpaceX mission outcomes into 'Success' and 'Failure' categories and counts the number of missions in each category.

List the total number of successful and failure mission outcomes

```
%%sql
SELECT
  CASE
    WHEN Mission_Outcome LIKE 'Success%' THEN 'Success'
    ELSE 'Failure'
  END AS `Mission Outcome`,
  COUNT(*) AS Count
FROM
  SPACEXTBL
GROUP BY
  `Mission Outcome`;
```

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

Mission Outcome	Count
Failure	1
Success	100

Boosters Carried Maximum Payload

- This SQL query selects the unique booster versions from the SPACEXTBL table where the payload mass is the highest among all payload masses in the table.

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

```
%%sql
SELECT DISTINCT(Booster_Version)
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Booster_Version
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

2015 Launch Records

- This SQL query extracts the month, landing outcome, booster version, and launch site for all SpaceX launches in 2015 that had a failed landing on a drone ship.

```
%%sql
SELECT substr(Date, 4, 2) AS MONTH, `Landing _Outcome`, Booster_Version, Launch_Site
FROM SPACEXTBL
WHERE `Landing _Outcome` LIKE '%Failure%(drone ship)%' AND
substr(Date,7,4)='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

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

This SQL query retrieves a count of successful landing outcomes from the "SPACEXTBL" table between the dates '04-06-2010' and '20-03-2017', grouped by landing outcome and sorted in descending order by the count of successful outcomes.

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

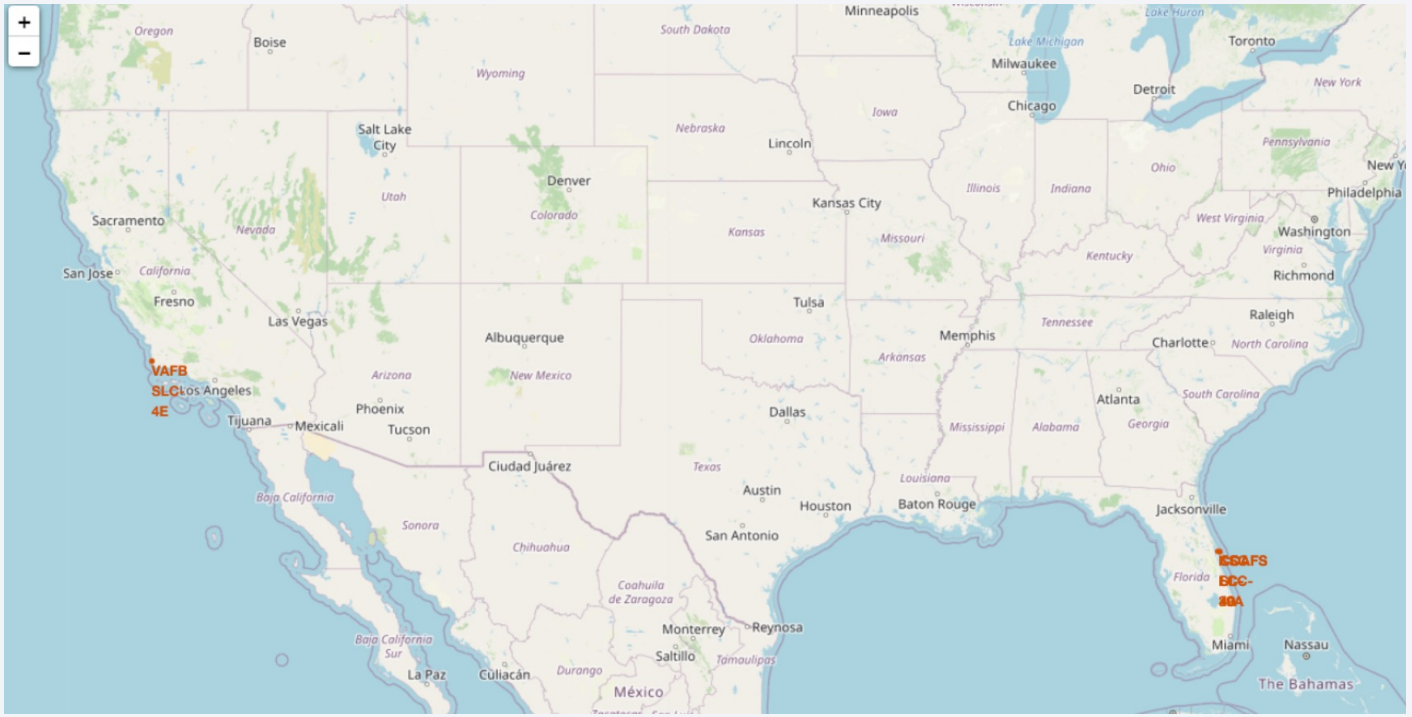
```
* sqlite:///my_data1.db
Done.
```

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


Section 3
Launch Sites
Proximities Analysis

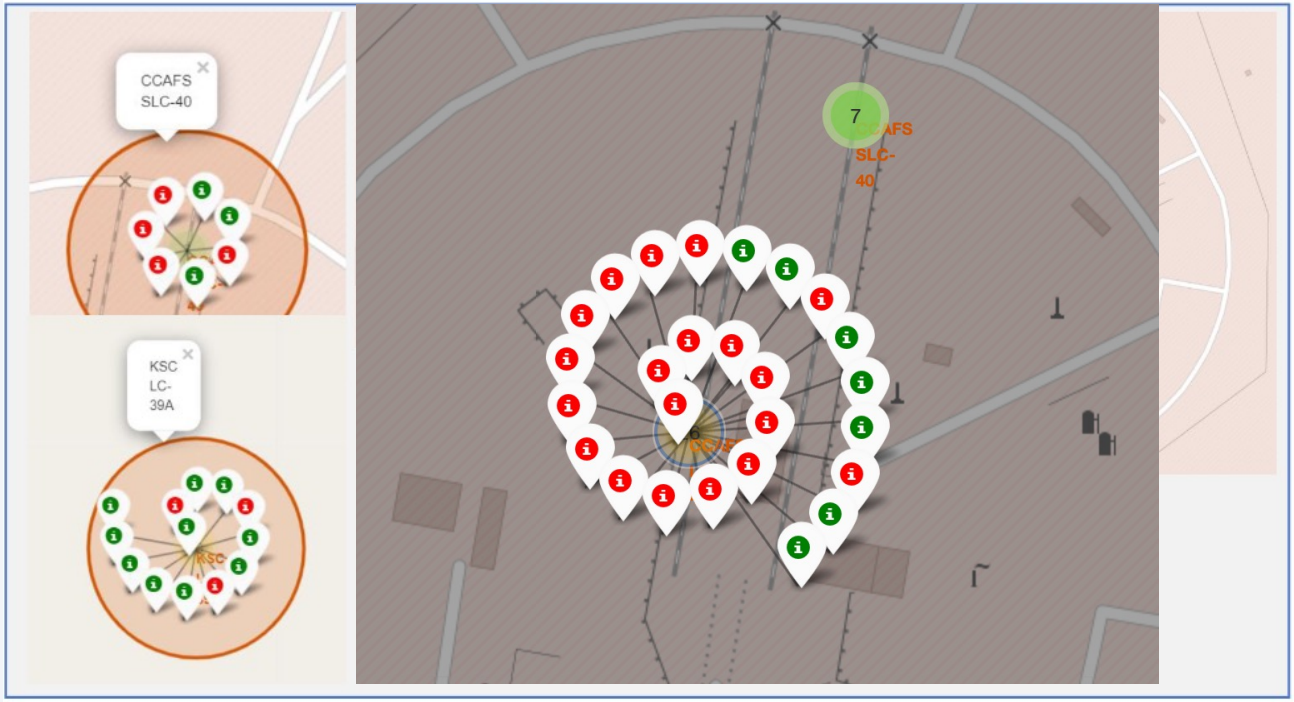
All Launch Sites Marker

- All Launch sites are near coastal lines.
- All except launch sites in California are in close proximity to the equator while still being in the US mainland.



Launch Sites colored by success or failure

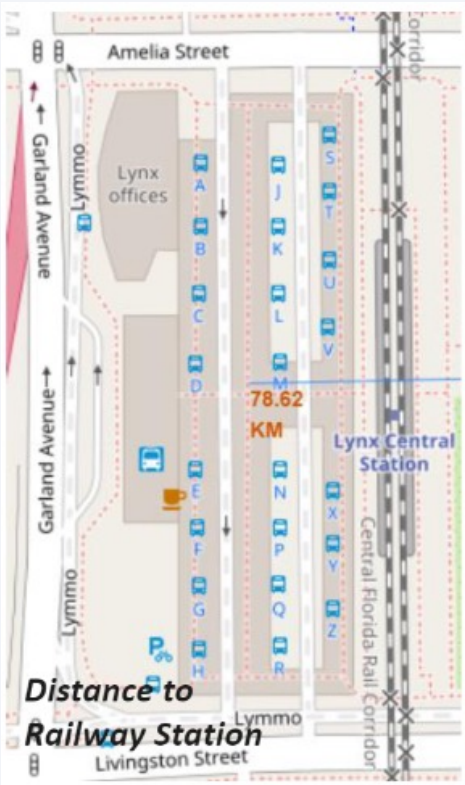
Florida Launches Sites:

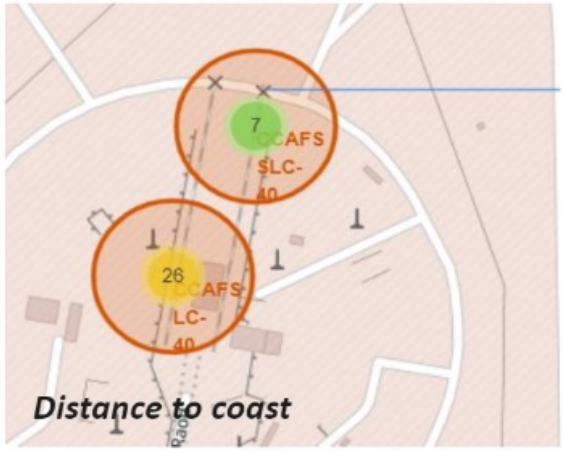


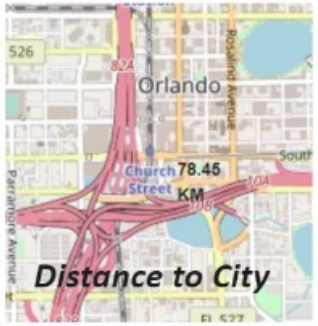
California Launch Sites

Green Markers => Successful Launch, Red Markers => Failure

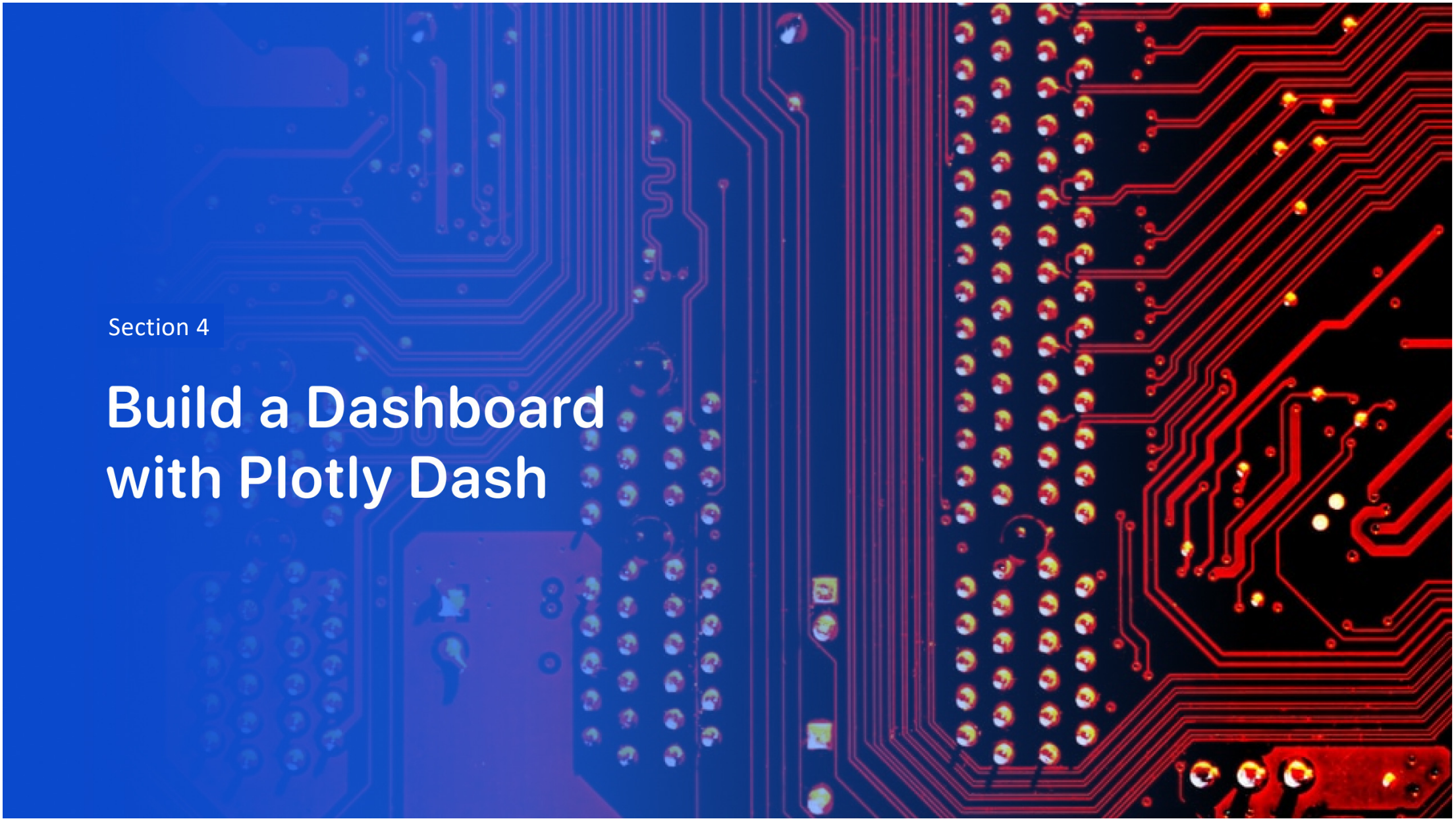
Distance from launch sites areas of interest



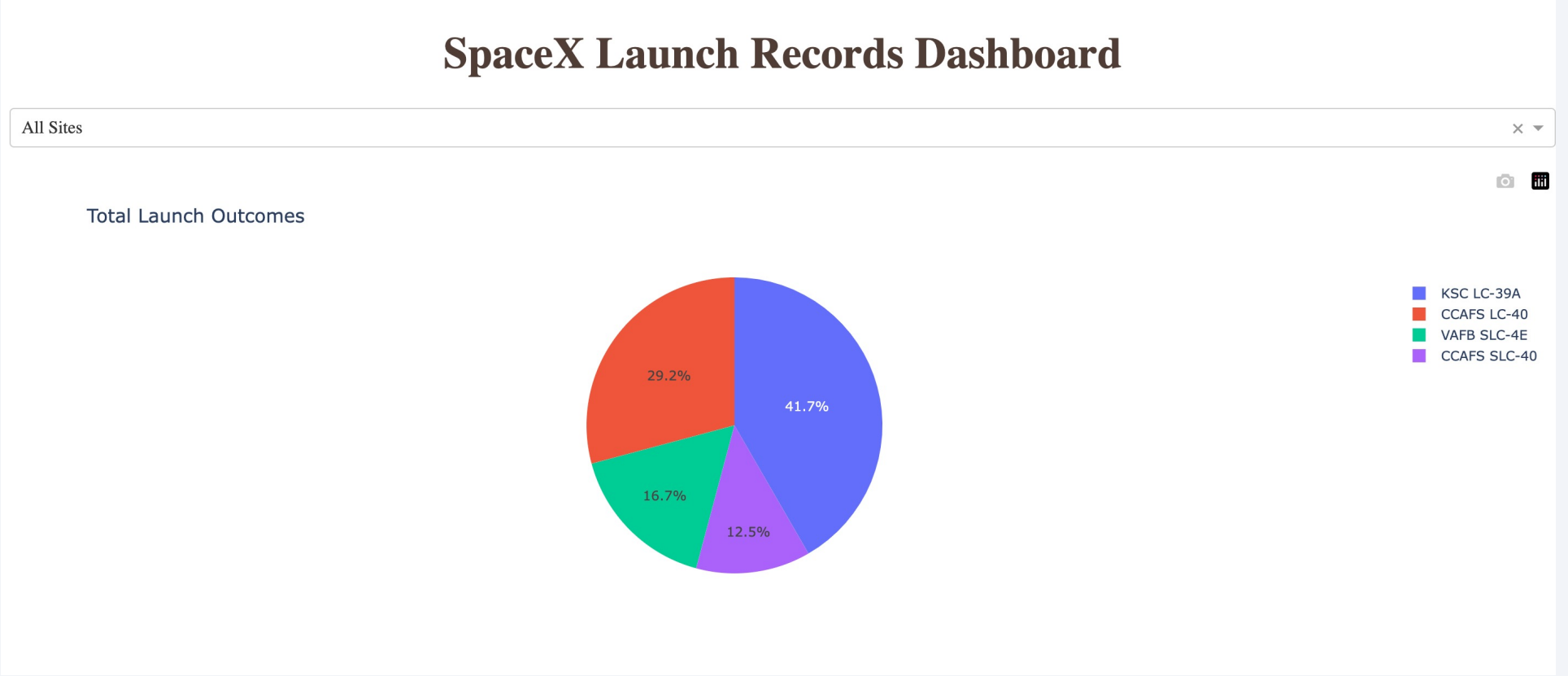




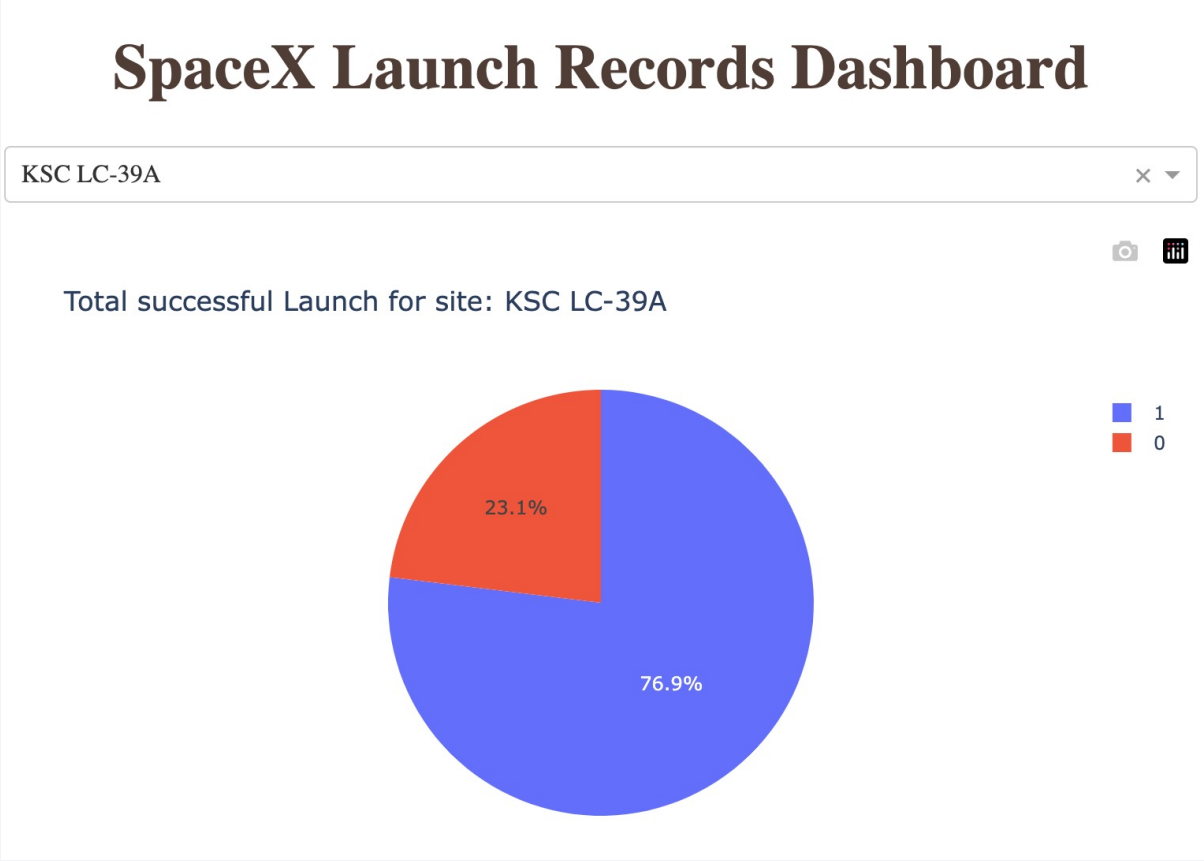
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes


Section 4
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



Pie chart showing the Launch site with the highest launch success ratio



Scatter plot of Payload vs Launch Outcome for all sites

- We can see success rate is higher for low payload weight.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

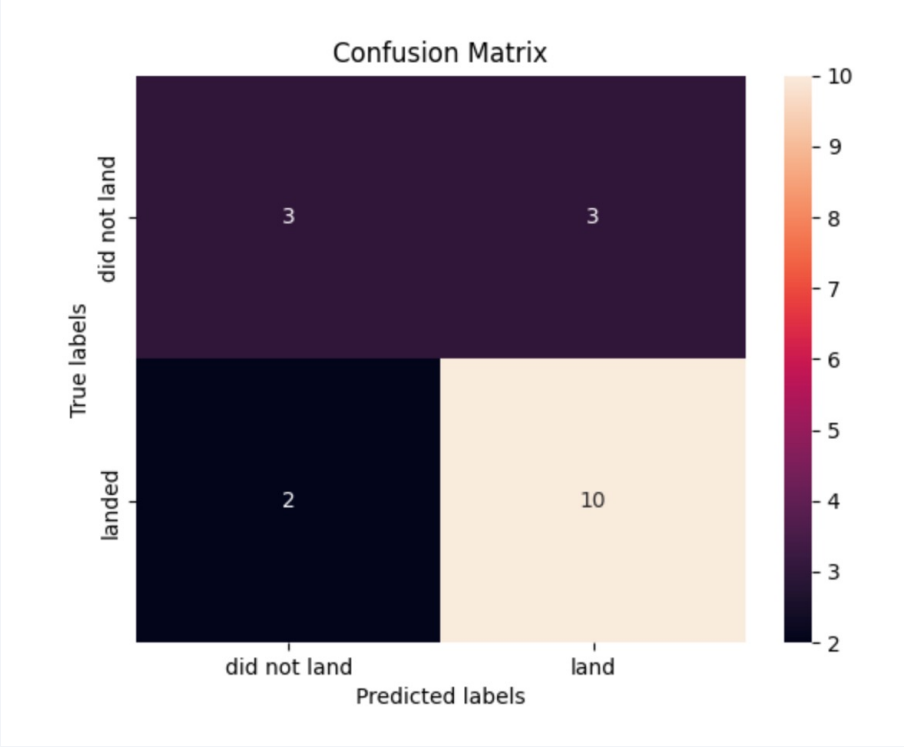
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix of the decision tree classifier indicates that it can distinguish between different classes, but false positives are a major concern.
- This means the classifier may label unsuccessful landings as successful, impacting its overall accuracy.



Conclusions

- A higher number of flights at a launch site is associated with a greater success rate.
- Starting from 2013 and continuing until 2020, the launch success rate showed an upward trend.
- Orbits such as ES-L1, GEO, HEO, SSO, and VLEO had the highest success rates.
- KSC LC-39A was the site with the greatest number of successful launches among all sites.
- The Decision Tree Classifier is considered the most effective machine learning algorithm for this particular task.

Thank you!