IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Abdulla Razick
10/31/2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection with API
  - Data Collection with Web Scraping
  - Data Wrangling
  - EDA with SQL
  - EDA with Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Predictions
- Summary of all results
  - EDA Results
  - Interactive Analytics Screenshot
  - Predictive Analysis Results

Introduction

- Project background and context

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 you want to find answers
  - What factors determine if the rocket will land successfully
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Using SpaceX API and web scraping
- Perform data wrangling
  - One-hot encoding was used on categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- How data sets were collected.
  - We gathered data through a GET request to the SpaceX API.The
  - JSON response was decoded using the .json() method and converted into a pandas DataFrame with json_normalize().
  - The data was then cleaned by checking for and addressing any missing values.
  - Additionally, we used BeautifulSoup to scrape Falcon 9 launch records from Wikipedia.
  - Our goal was to extract the launch records in HTML table format, parse the data, and transform it into a pandas DataFrame for further analysis.

Data Collection – SpaceX API

- We utilized a GET request to the SpaceX API to gather data, then cleaned and performed basic wrangling and formatting on the collected information.
- https://github.com/RazickA/IBM_DataScience_Capstone_SpaceX/blob/main/jupyter-labs-spacex-data-collection-api.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:

```
[29]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
[32]: response.status_code
[32]: 200
```

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

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

Data Collection - Scraping

- We performed web scraping on Falcon 9 launch records using BeautifulSoup.
- The HTML table was parsed and transformed into a pandas DataFrame.
- https://github.com/RazickA/IBM_DataScience_Capstone_SpaceX/blob/main/jupyter-labs-webscraping.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.

```
# use requests.get() method with the provided static_url
# assign the response to a object
data  = requests.get(static_url).text
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html.parser')
```

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

```
# Use soup.title attribute
print(soup.title.string)
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

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

```
# 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')
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

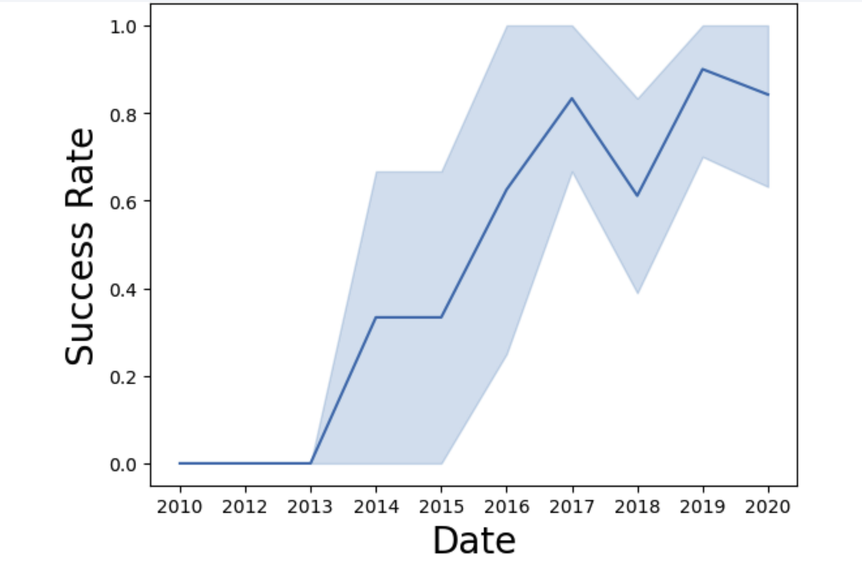
```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
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']=[]
```

Data Wrangling

- We conducted exploratory data analysis to identify the training labels.
- We calculated the number of launches per site and analyzed the frequency and types of orbits.
- A landing outcome label was created from the outcome column, and the results were exported to a CSV file.
- https://github.com/RazickA/IBM_DataScience_Capstone_SpaceX/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We explored the data by visualizing relationships between flight number and launch site, payload and launch site, and flight number and orbit type.
- Additionally, we analyzed the success rate of each orbit type and observed the yearly trend in launch success.
- https://github.com/RazickA/IBM_DataScience_Capstone_SpaceX/blob/main/edadataviz.ipynb

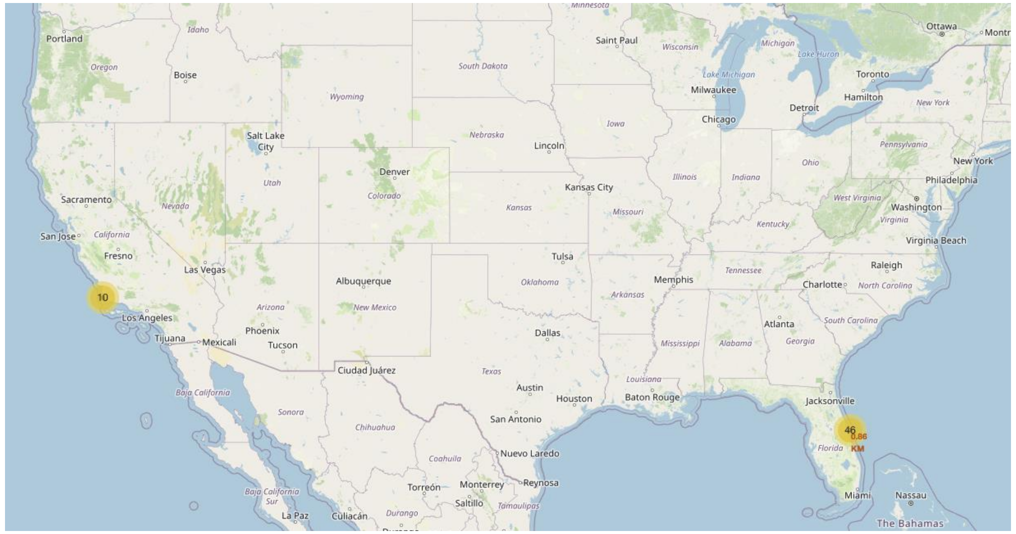


EDA with SQL

- We used SQL for exploratory data analysis (EDA) to gain insights from the data, writing queries to determine:The unique launch site names in the space missions.
  - The total payload mass of boosters launched by NASA (CRS).
  - The average payload mass for booster version F9 v1.1.
  - The total count of successful and failed mission outcomes.
  - Details of failed landing outcomes on drone ships, including booster version and launch site names.
- https://github.com/RazickA/IBM_DataScience_Capstone_SpaceX/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We marked each launch site on a Folium map and added objects like markers, circles, and lines to indicate the success or failure of launches at each location.
- Launch outcomes were classified as 0 for failure and 1 for success.
- Using color-coded marker clusters, we identified launch sites with relatively high success rates.
- We calculated distances between launch sites and nearby features, addressing questions such as:
  - Are launch sites located near railways, highways, and coastlines?
  - Do launch sites maintain a certain distance from cities?
  - https://github.com/RazickA/IBM_DataScience_Capstone_SpaceX/blob/main/lab_jupyter_launch_site_location.ipynb



Build a Dashboard with Plotly Dash

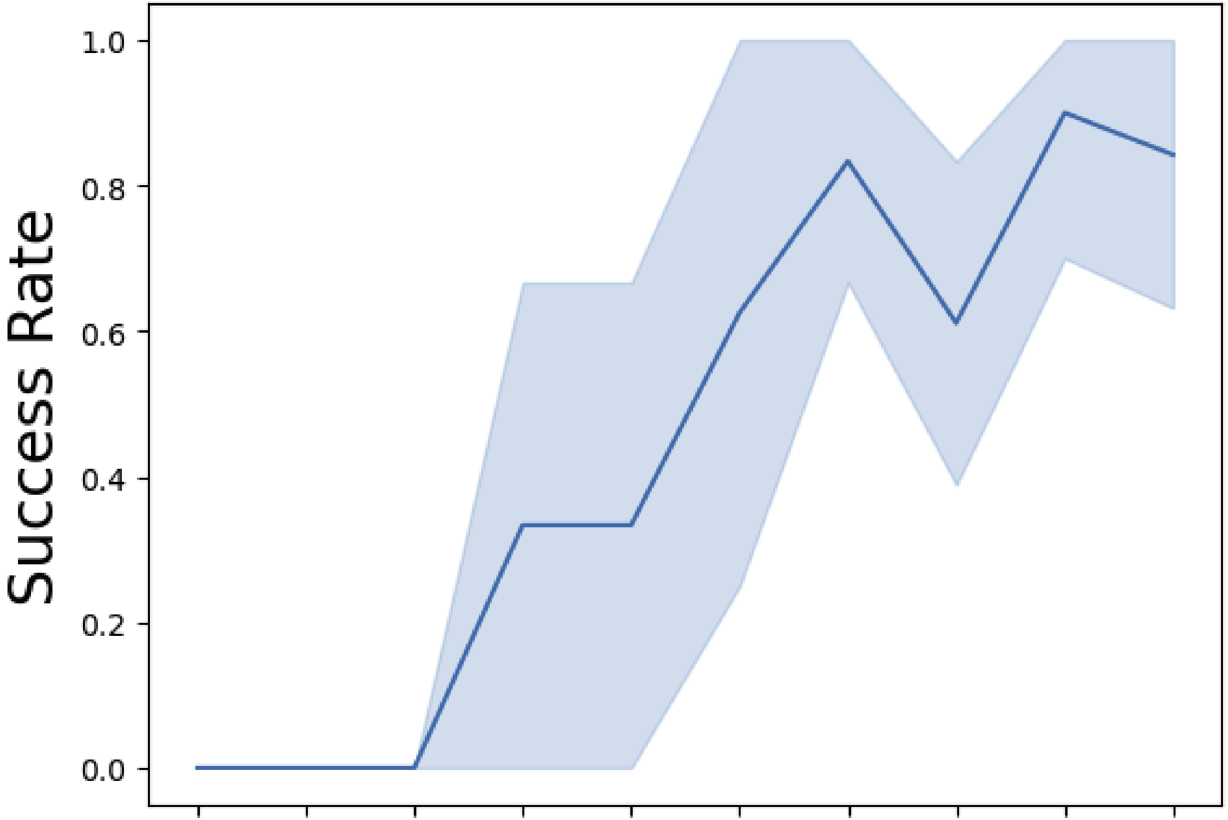
- We created an interactive dashboard using Plotly Dash.
- Pie charts were plotted to display the total launches across various sites.
- A scatter plot was used to illustrate the relationship between launch outcome and payload mass (kg) for different booster versions.
- https://github.com/RazickA/IBM_DataScience_Capstone_SpaceX/blob/main/Spaceex_Dash_app.ipynb

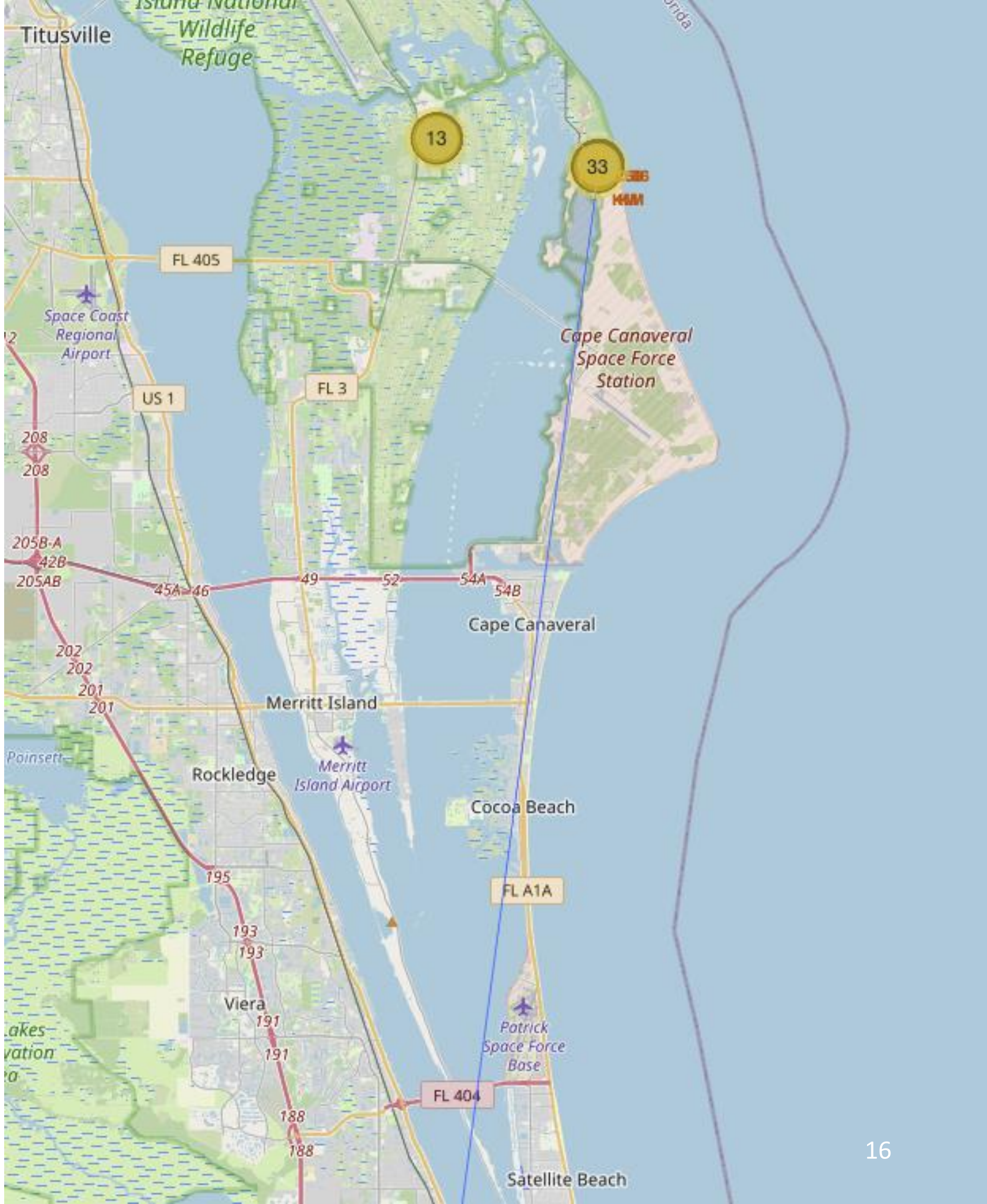
Predictive Analysis (Classification)

- We loaded the data using NumPy and pandas, performed transformations, and split it into training and testing sets.
- Various machine learning models were built, with hyperparameters tuned using GridSearchCV.
- Accuracy was used as the evaluation metric, and the model was enhanced through feature engineering and algorithm tuning. We identified the best-performing classification model.
- https://github.com/RazickA/IBM_DataScience_Capstone_SpaceX/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

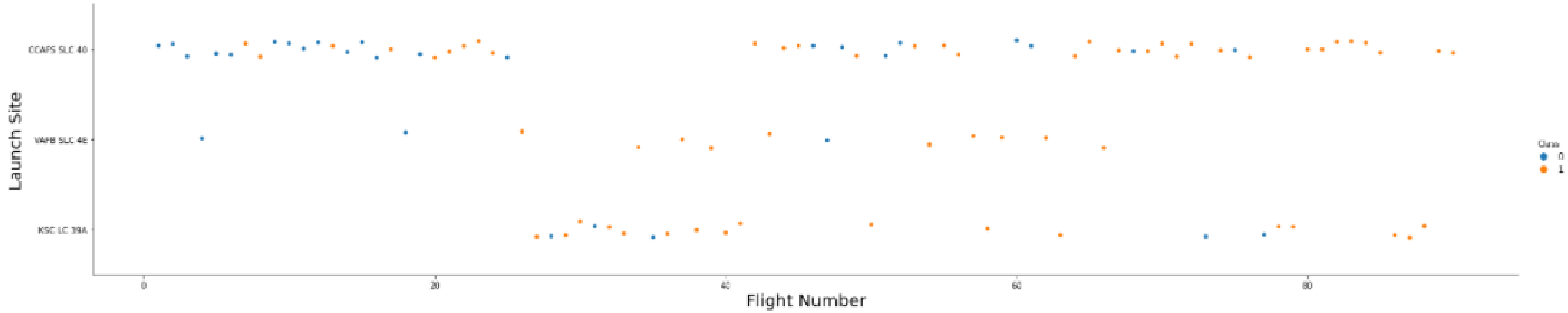
- Success rates continue to rise
- Proximity:
  - distance_highway = 0.58 km
    distance_railroad = 1.28 km
    distance_city = 51.43 km
- Best performing model was decision tree (accuracy: 0.87)





Section 2

Insights drawn from EDA

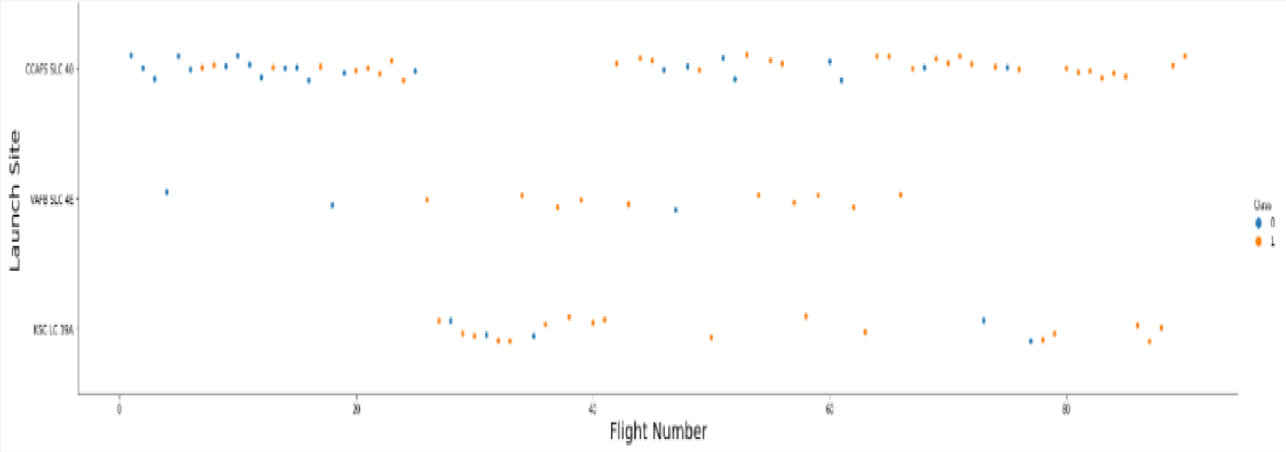


Flight Number vs. Launch Site

- The plot revealed that launch sites with a higher number of flights tend to have a greater success rate

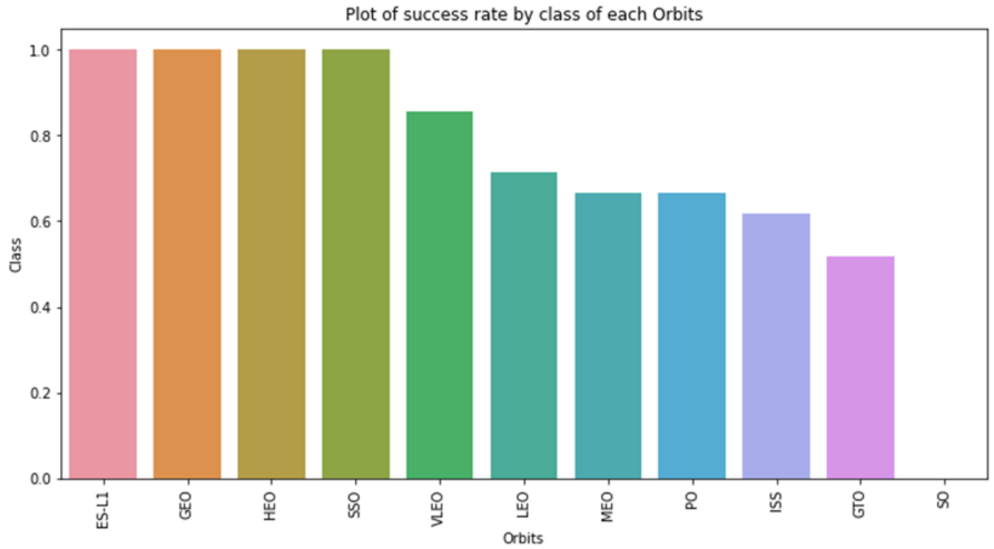
Payload vs. Launch Site

- Higher payload mass at launch site CCAFS SLC 40 is associated with a higher success rate for rockets.



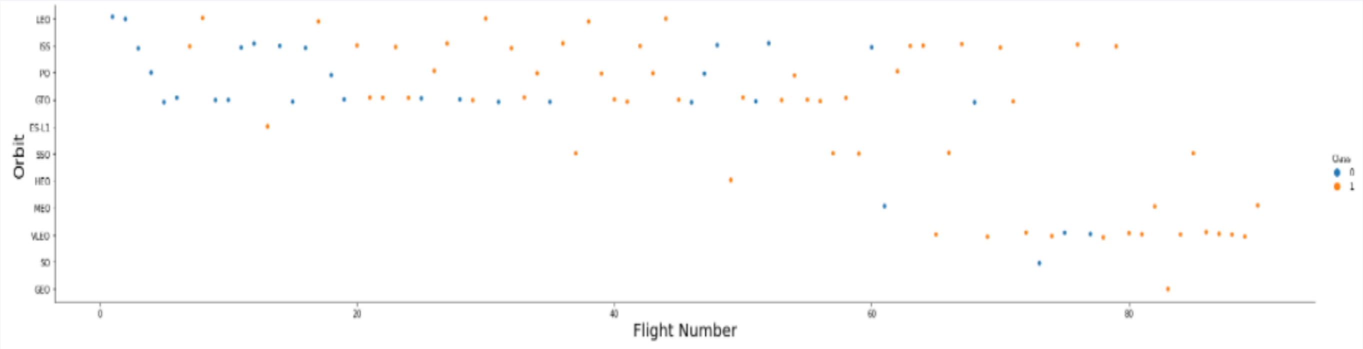
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



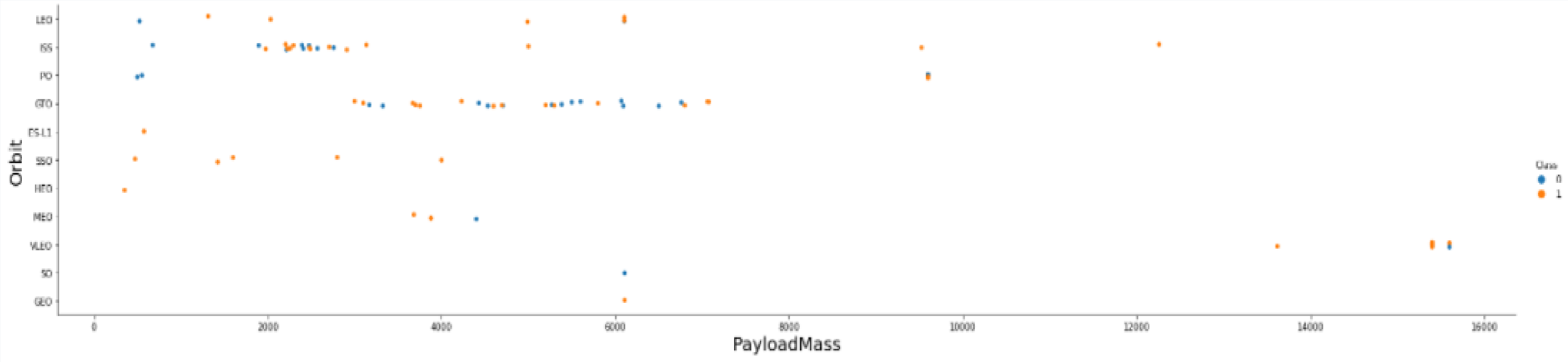
Flight Number vs. Orbit Type

- The plot of Flight Number versus Orbit Type reveals that in Low Earth Orbit (LEO), success is correlated with the number of flights, whereas in Geostationary Transfer Orbit (GTO), there is no evident relationship between flight number and success.



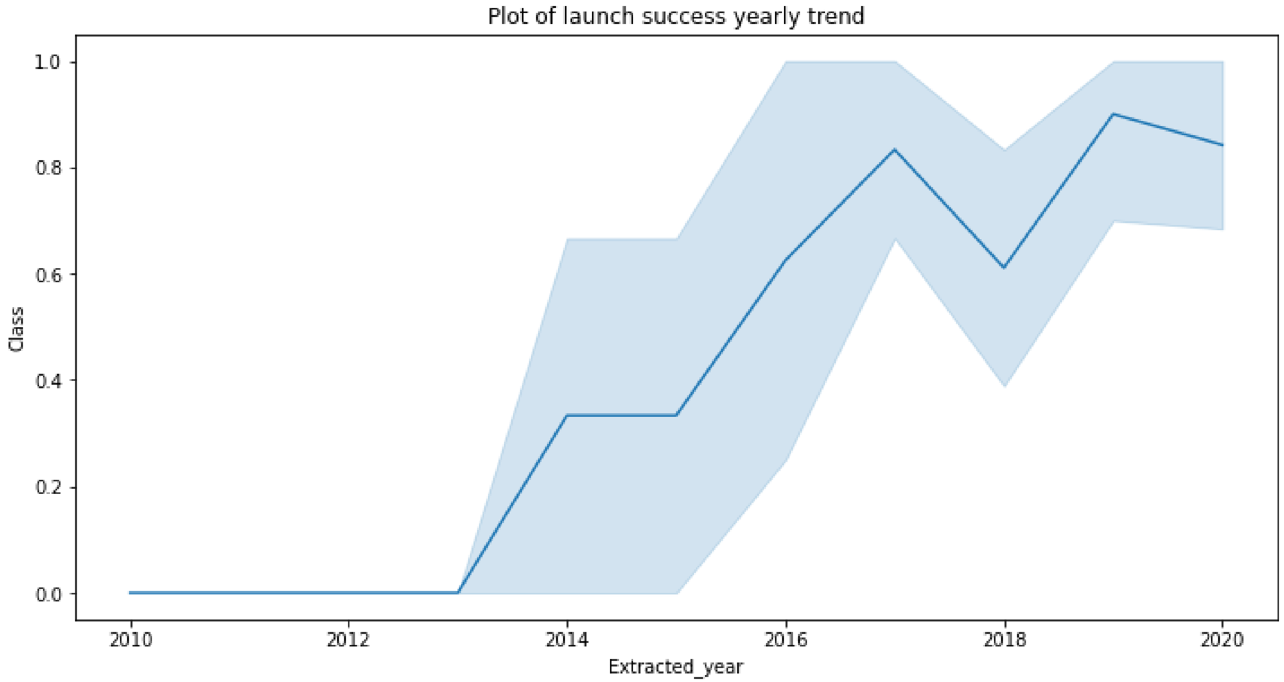
Payload vs. Orbit Type

- We observe that with heavy payloads, successful landings are more common for Polar Orbit (PO), Low Earth Orbit (LEO), and International Space Station (ISS) orbits.



Launch Success Yearly Trend

- Success increased from 2013 to 2020



All Launch Site Names

- Used the keyword DISTINCT to display only unique launch sites from the SpaceX data

```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE
```

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

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

Launch Site Names Begin with 'CCA'

- Used the query above to display 5 records where launch sites begin with `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

- Total payload carried by boosters from NASA were calculated with this query

```
]: %%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Calculated the average payload mass carried by booster version F9 v1.1

```
]: %%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.1';

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

AVG(PAYLOAD_MASS__KG_)
2928.4

```
%sql SELECT MIN(Date) AS first_successful_ground_pad_date FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (ground pad)'
```

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

first_successful_ground_pad_date
2015-12-22

First Successful Ground Landing Date

- The first successful landing outcome on ground pad was 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the WHERE clause to filter for boosters that successfully landed on a drone ship, and applied the AND condition to identify successful landings with a payload mass between 4,000 and 6,000 kg

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 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

- We used % in the WHERE clause to filter records where MissionOutcome indicated either success or failure.

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

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

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

Boosters Carried Maximum Payload

- We identified the booster that carried the maximum payload by using a subquery with the MAX() function in the WHERE clause.

```
%%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

- We combined the WHERE clause with LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes on drone ships, specifying the booster versions and launch site names for the year 2015.

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

0]:
```
%sql SELECT "Landing_Outcome", "Booster_Version", "Launch_Site", substr("Date", 6, 2) AS Month FROM SPACEXTBL WHERE "Landing_Outcome" = 'Failure (drone ship)' AND substr("Date", 1, 4) = '2015'
```

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

0]:

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

```
4]: %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

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

Landing_Outcome	TOTAL_NUMBER
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

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

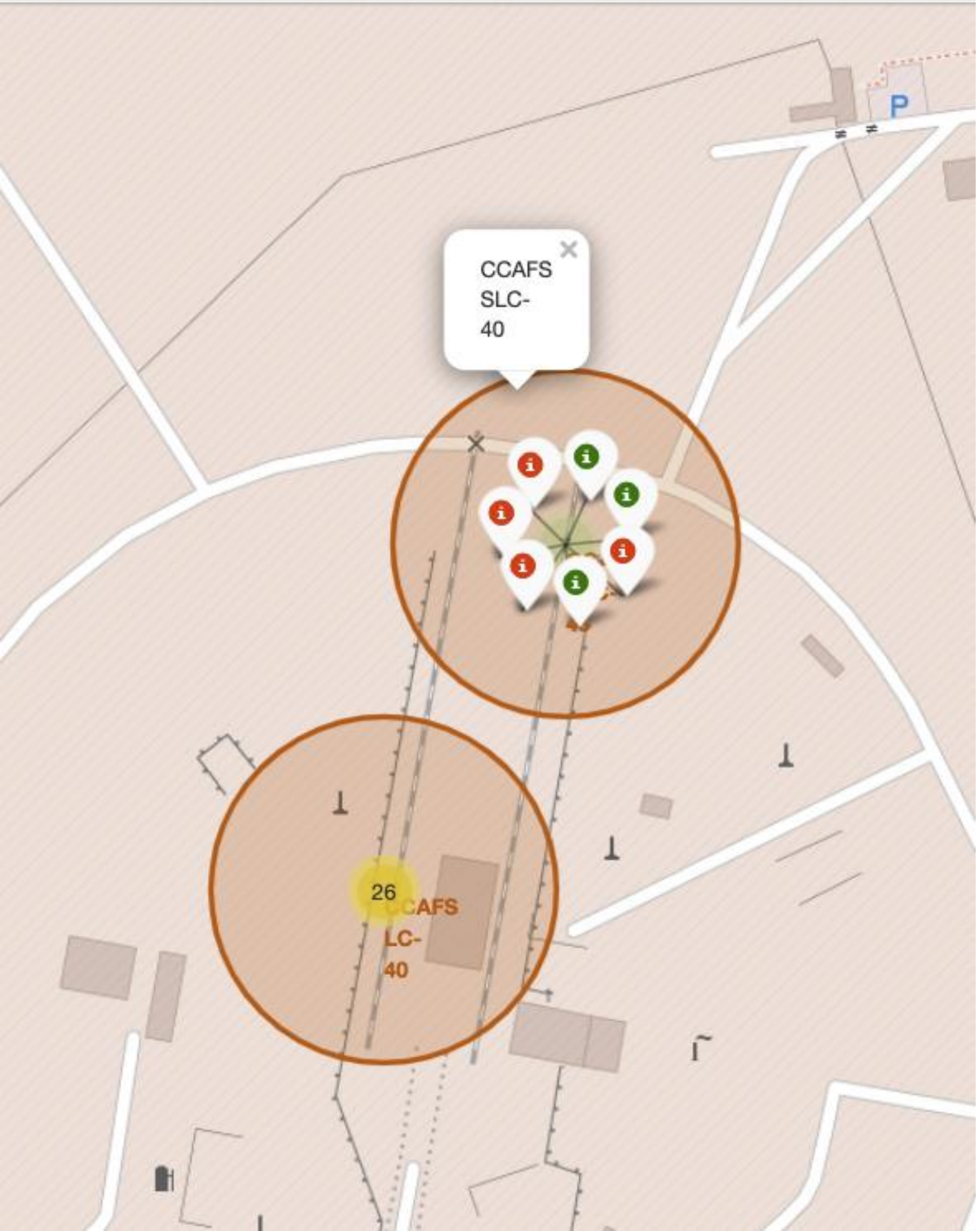
- We selected landing outcomes and their count from the data, using the WHERE clause to filter for outcomes between 2010-06-04 and 2010-03-20.
- The GROUP BY clause was applied to group the landing outcomes, and the ORDER BY clause was used to sort them in descending order.

Section 3

Launch Sites Proximities Analysis

Launch Sites Global Map

- Launch sites are in the United States, specifically on the coasts of Florida and California



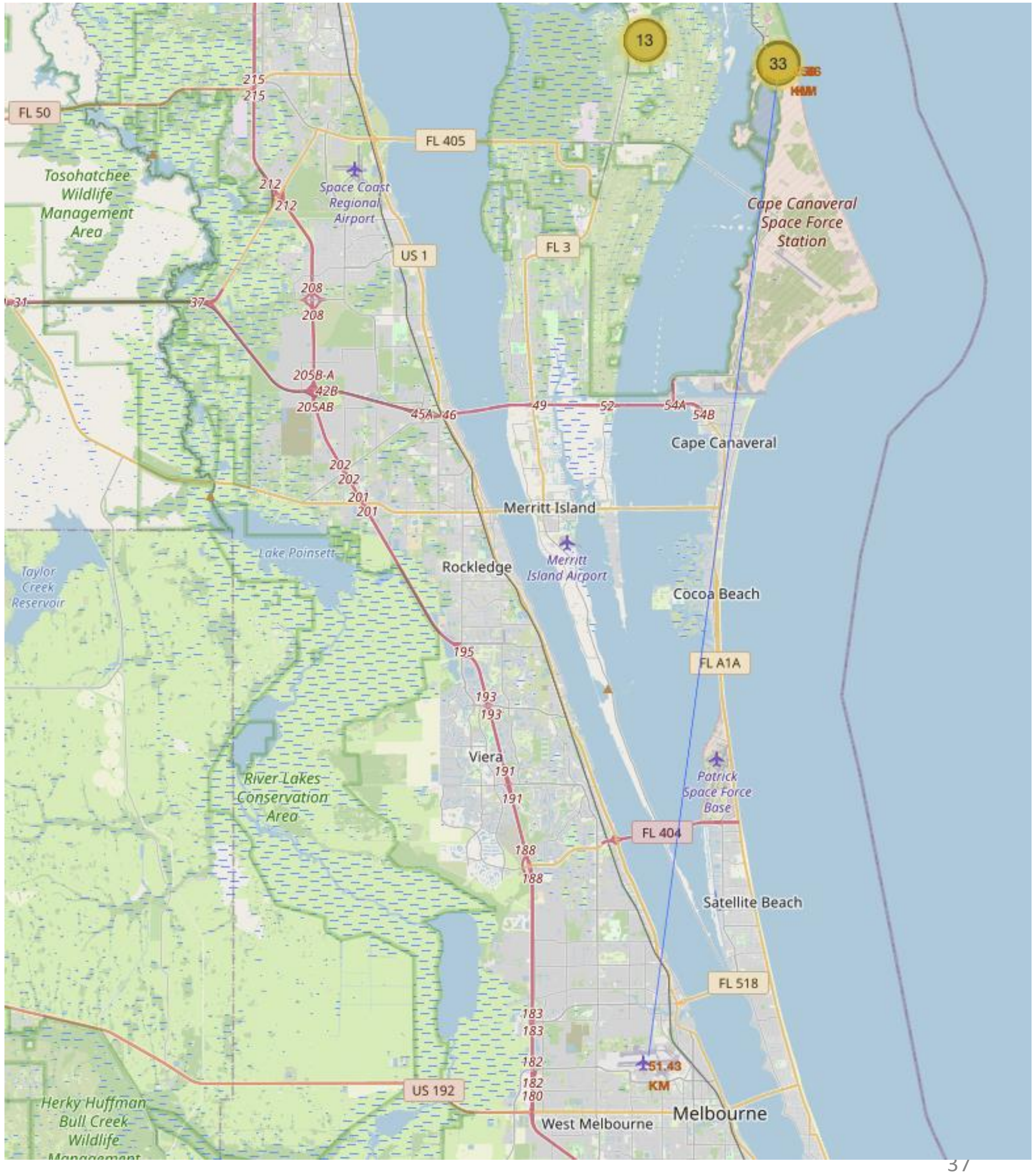
Markers showing launch site success

- Green marker shows successful launches
- Red marker shoes failures

Distance Between Landmarks and Launchsites

Distance between relevant landmarks

```
distance_highway = 0.5834695366934144  km
distance_railroad = 1.2845344718142522  km
distance_city = 51.43416999517233  km
```

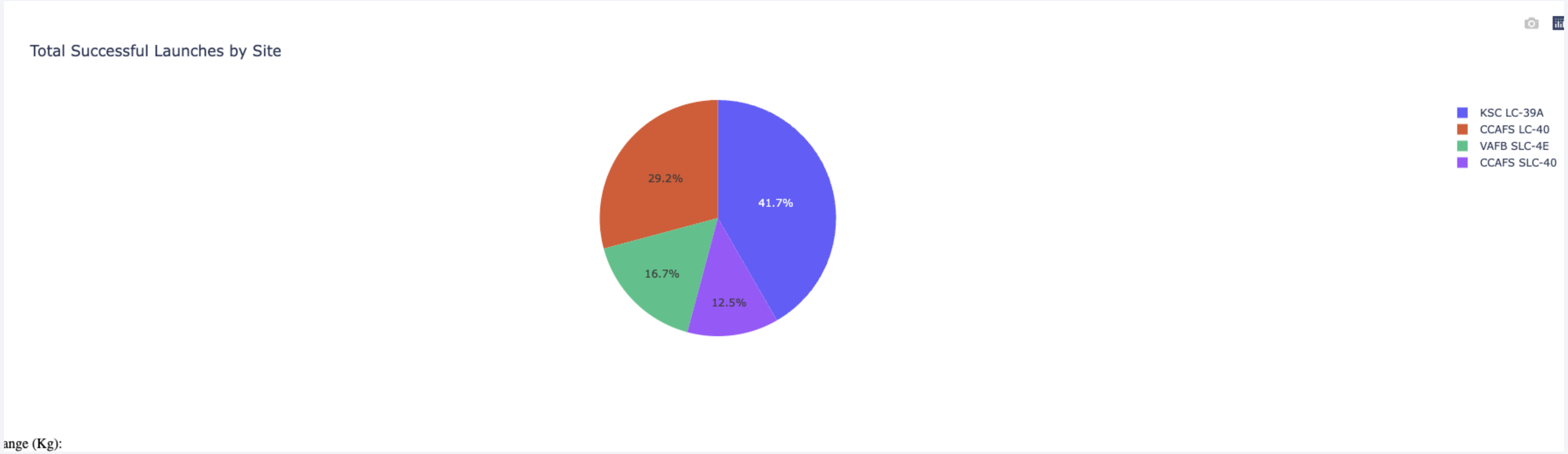


Section 4

Build a Dashboard with Plotly Dash

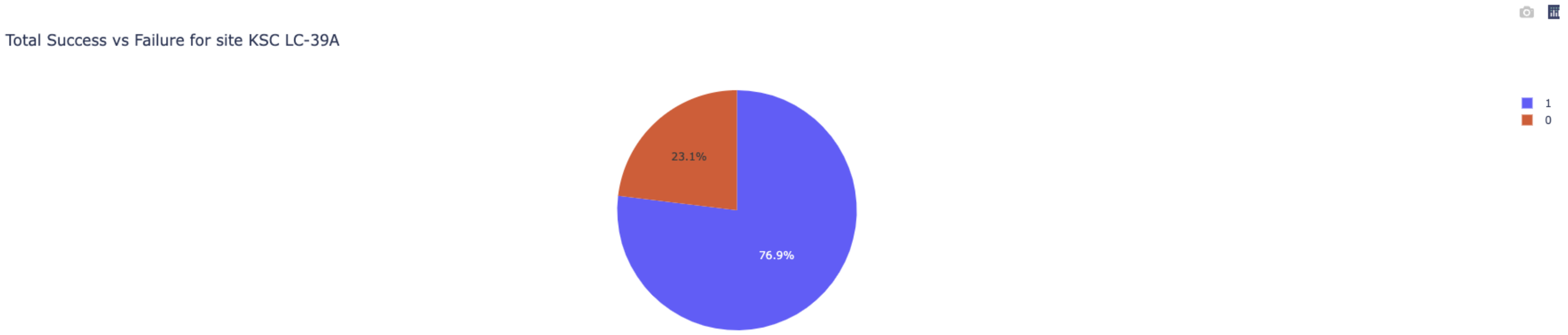
Pie chart – Success achieved by each launch site

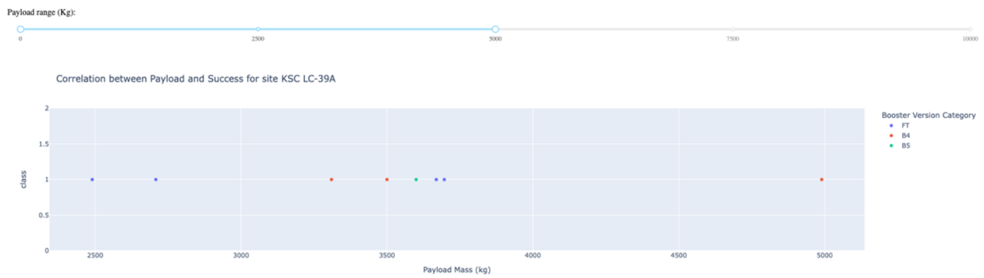
- Using the pie chart, it is seen that KSC LC 39A had the most successful launches

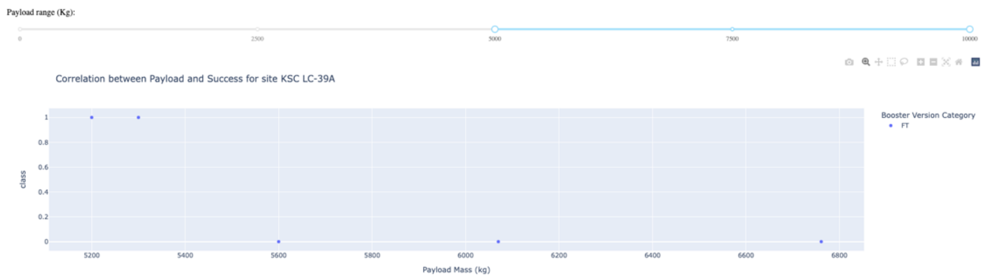


Launch site with highest success ratio

- KSC LC-39A achieved a 76.9% success rate while while getting a 23.1% fail rate







Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

- Based on the scatter plot, low weighted payload have a higher success rate than heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Using GridSearch, the decision tree classifier model yielded the highest accuracy

[104]:
```
logistic_regression_best_score = logreg_cv.best_score_
svm_best_score = svm_cv.best_score_
decision_tree_best_score = tree_cv.best_score_
knn_best_score = knn_cv.best_score_

# Create a dictionary to store and display the scores
model_performance = {
    "Logistic Regression": logistic_regression_best_score,
    "Support Vector Machine": svm_best_score,
    "Decision Tree": decision_tree_best_score,
    "K-Nearest Neighbors": knn_best_score
}

# Find and display the best-performing model
best_model = max(model_performance, key=model_performance.get)
print("Best-performing model:", best_model)
print("Accuracy:", model_performance[best_model])
```

```
Best-performing model: Decision Tree
Accuracy: 0.8714285714285713
```

Confusion Matrix

- The confusion matrix for the decision tree classifier indicates that the model can distinguish between different classes. However, the main issue is the false positives, where unsuccessful landings are incorrectly marked as successful by the classifier.

Conclusions

We can conclude that:

The greater the number of flights at a launch site, the higher the success rate at that site.

The launch success rate increased from 2013 to 2020.

The orbits ES-L1, GEO, HEO, SSO, and VLEO had the highest success rates.

KSC LC-39A recorded the most successful launches of all sites.

The decision tree classifier proved to be the best machine learning algorithm for this task.

Appendix

- Python code for data retrieval, cleaning, and visualization.
- SQL queries for filtering, grouping, and finding specific outcomes.
- Charts (pie, scatter) for launch analysis.
- Folium map for launch site success rates.

Thank you!