IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rodrigo Figueroa
January 16, 2025

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  SpaceX promotes its Falcon 9 rocket launches on its website at a price of $62 million, significantly lower than other providers, which charge upwards of $165 million per launch. This cost advantage largely stems from SpaceX's ability to reuse the first stage of the rocket. By accurately predicting whether the first stage will successfully land, we can better assess the overall cost of a launch. This information could be valuable for competing companies looking to bid against SpaceX for rocket launches. The objective of this project is to develop a machine learning pipeline that can effectively predict the success of the first stage landing

  Problems you want to find answers

  What factors influence the successful landing of a rocket?

  How do the various elements interact to affect the success rate of a landing?

  What operational conditions must be established to ensure a successful landing program?

  .

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
- Perform data wrangling
  - Describe how data was processed
- 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

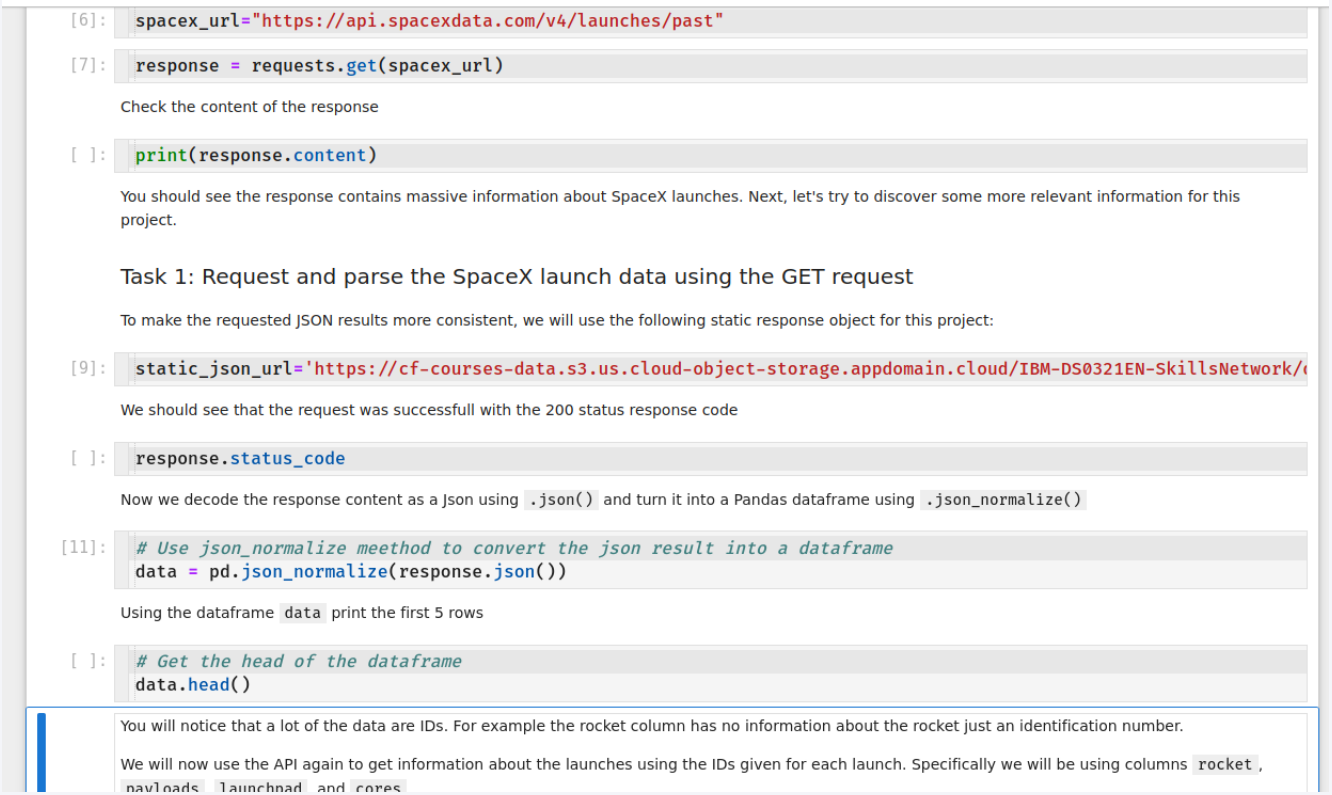
The data collection was conducted using a GET request to the SpaceX API. We then decoded the response content as JSON by utilizing the .json() function and transformed it into a Pandas DataFrame using .json_normalize().

Next, we cleaned the data by checking for missing values and filling in those values where necessary. Additionally, we performed web scraping on Wikipedia to gather Falcon 9 launch records using BeautifulSoup. The goal was to extract the launch records from an HTML table, parse the table, and convert it into a Pandas DataFrame for future analysis.

Data Collection – SpaceX API

- We utilized a GET request to the SpaceX API to collect data, followed by cleaning the retrieved data and performing basic data wrangling and formatting.

- The GitHub URL of the completed SpaceX API notebook: https://github.com/thefigaro/IBM_SkillsBuild_assignments/blob/main/Data Science Capstone/01_SpaceX_Data_Collection_API.ipynb



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

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

Check the content of the response

[ ]:
```
print(response.content)
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

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:

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

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

[ ]:
```
response.status_code
```

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

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

Using the dataframe `data` print the first 5 rows

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

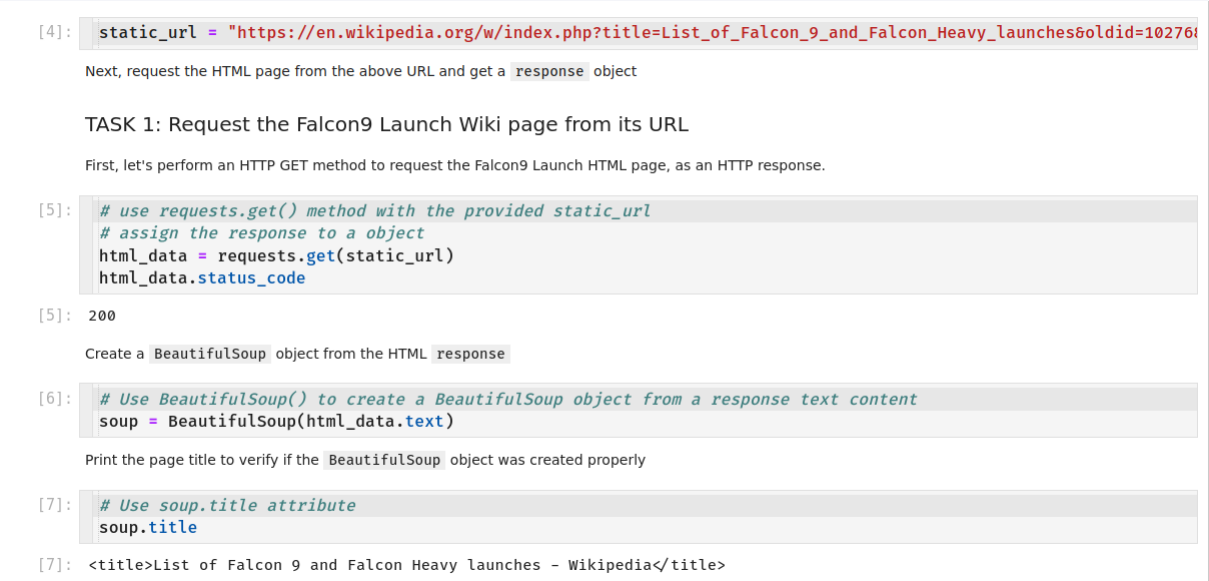
You will notice that a lot of the data are IDs. For example the rocket column has no information about the rocket just an identification number.

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`

Data Collection - Scraping

- We perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response. Then we, Extract all column/variable names from the HTML table header, extract all column/variable names from the HTML table header to pass in a pandas DataFrame. Finally we export all in .csv format.

- We can see all the procedure in GitHub: https://github.com/thefigaro/IBM_SkillsBuild_assignments/blob/main/Data Science Capstone/02_SpaceX_Web_Scraping.ipynb



```
[4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=10276
```

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.

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

```
[5]: 200
```

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

```
[6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
     soup = BeautifulSoup(html_data.text)
```

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

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

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

Data Wrangling

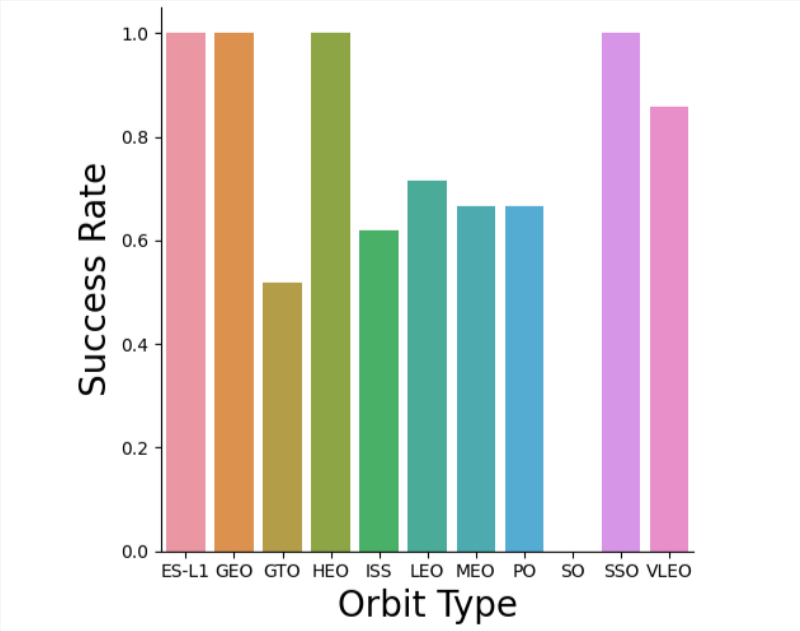
We will conduct an Exploratory Data Analysis (EDA) to identify patterns in the data and determine the appropriate label for training supervised models.

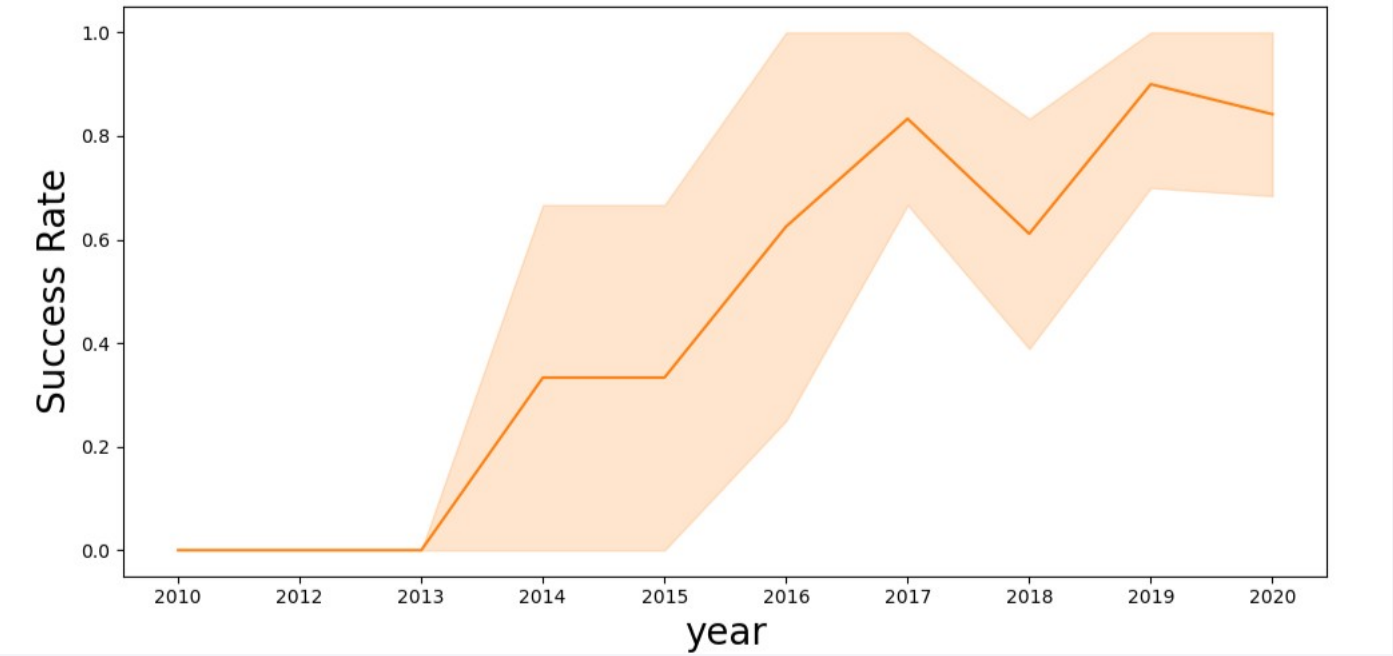
Next, we will calculate the number of launches at each site, as well as the frequency and types of each orbit.

Finally, we will create a landing outcome label based on the outcome column and export the results to a CSV file.

To get a glimpse of the notebook, please check the GitHub URL:
https://github.com/thefigaro/IBM_SkillsBuild_assignments/blob/main/Data Science Capstone/03_SpaceX_Data_Wrangling.ipynb

EDA with Data Visualization





The bar chart on the left displays the landing success rate based on orbit, while the chart on the right illustrates the success rate over time.

For an overview of the results, please see:
https://github.com/thefigaro/IBM_SkillsBuild_assignments/blob/main/Data Science Capstone/05_SpaceX_EDA_Data_Visualization.ipynb

EDA with SQL

We loaded the SpaceX dataset into a SQLite database using Jupyter's magic features. We then applied Exploratory Data Analysis (EDA) using SQL to gain insights from the data. For example, we aimed to find out:

- Displaying the names of the unique launch sites in the space mission
- Total payload mass carried by boosters launched by NASA
- Total average payload mass carried by booster version F9 v1.1

To get a glimpse of the notebook, please check the GitHub URL:

https://github.com/thefigaro/IBM_SkillsBuild_assignments/blob/main/Data Science Capstone/04_SpaceX_EDA_SQL.ipynb

Build an Interactive Map with Folium

We marked all launch sites on the Folium map and added various map objects, including markers, circles, and lines, to indicate the success or failure of launches at each site. We classified launch outcomes as follows: class 0 for failure and class 1 for success.

By utilizing color-coded marker clusters, we identified launch sites with relatively high success rates. Additionally, we calculated the distances from each launch site to nearby features and addressed several questions, such as:

Are launch sites located near railways, highways, and coastlines?

What is the distance from urban areas to the launch sites?

For an overview of the charts, please see:

https://github.com/thefigaro/IBM_SkillsBuild_assignments/blob/main/Data Science Capstone/06_SpaceX_Interactive_Visual_Analytics_Folium.ipynb

Build a Dashboard with Plotly Dash

We created an interactive dashboard using Plotly Dash.

The dashboard features pie charts that display the total number of launches by specific sites, as well as a scatter plot illustrating the relationship between launch outcomes and payload mass (in kilograms) for different booster versions.

To review the code, please see:

https://github.com/thefigaro/IBM_SkillsBuild_assignments/blob/main/Data Science Capstone/07_SpaceX_Interactive_Visual_Analytics_Plotly.py

Predictive Analysis (Classification)

We loaded the data using NumPy and Pandas, transformed it, and split it into training and testing sets.

We developed various machine learning models and tuned their hyperparameters using GridSearchCV. Accuracy was used as the evaluation metric for our models, and we enhanced performance through feature engineering and algorithm tuning.

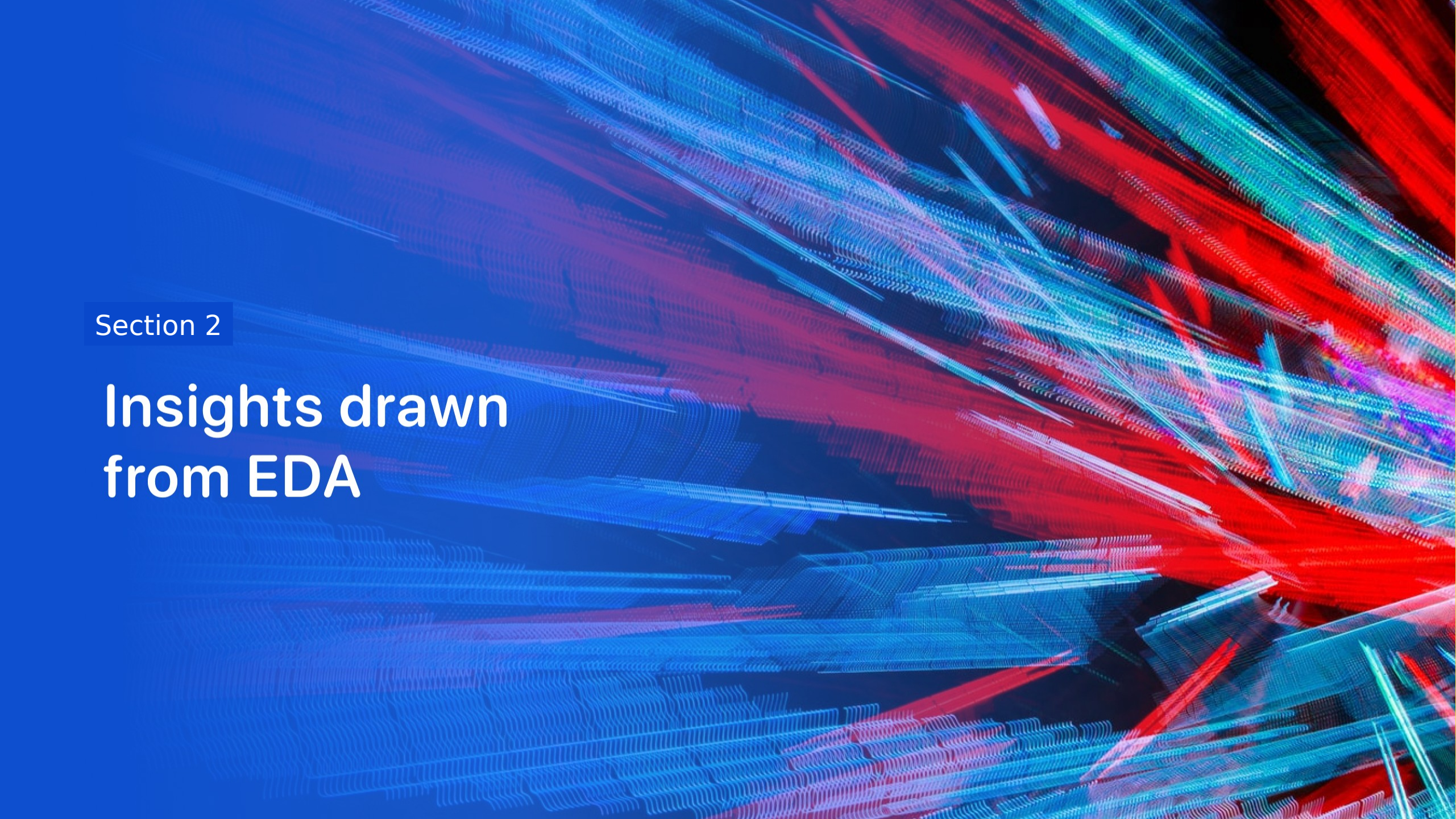
Ultimately, we identified the best-performing classification model.

To review the notebook, please see:

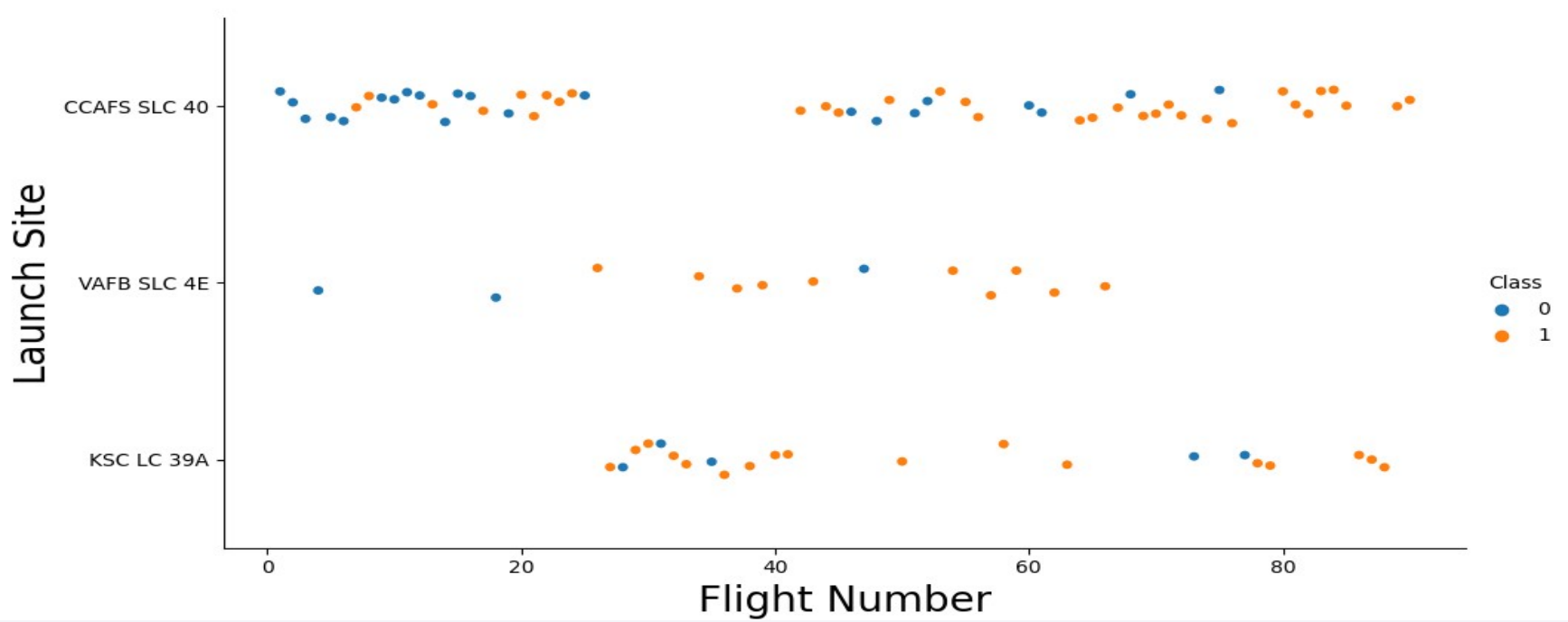
https://github.com/thefigaro/IBM_SkillsBuild_assignments/blob/main/Data Science Capstone/08_SpaceX_Predictive_Analytics.ipynb

Results

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

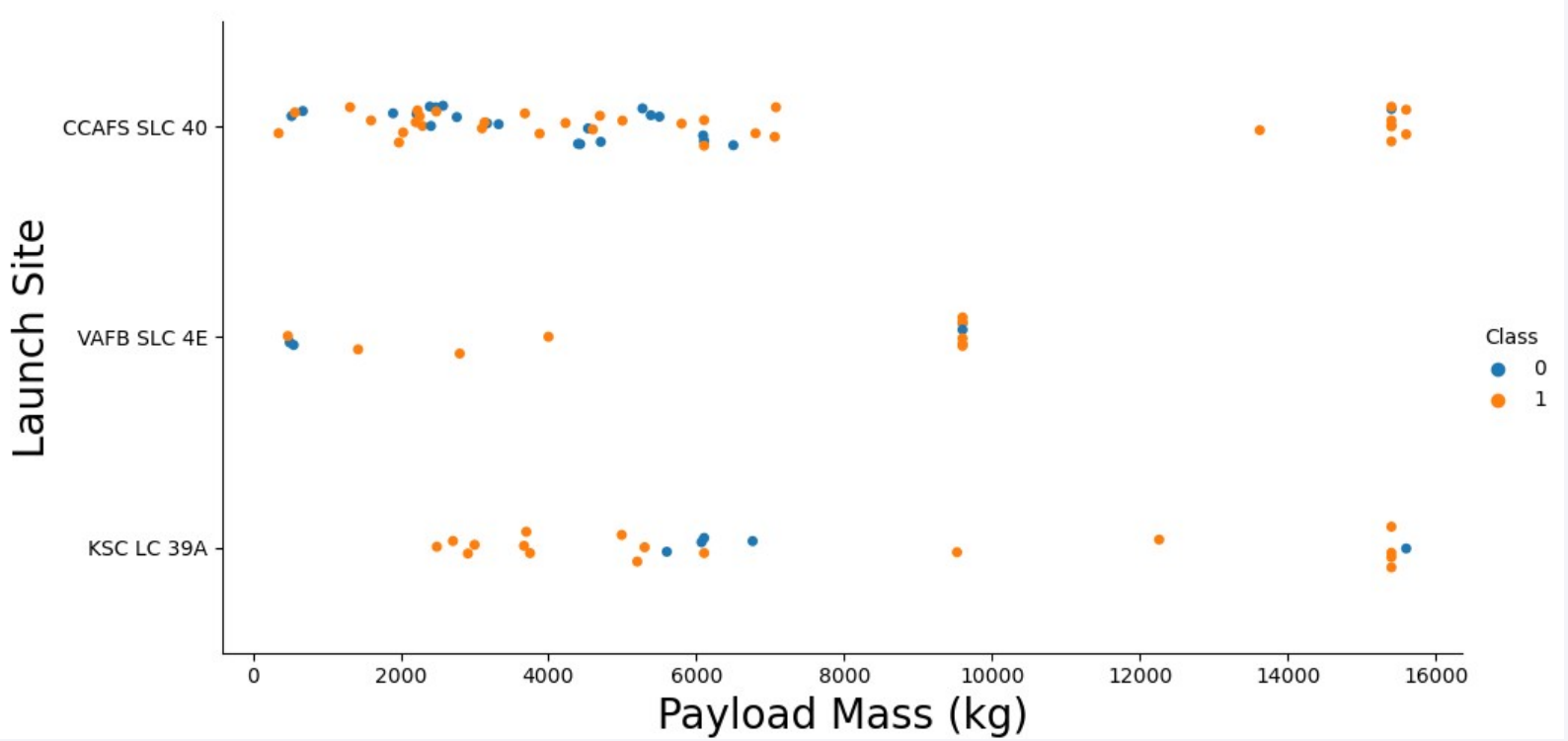

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



- The launch site CCAFS-SLC 40 is where the majority of launches occur, with over 40 flights contributing to its success rate.
- CCAFS SLC-40 has a success rate of 60%, while KSC LC-39A and VAFB SLC 4E boast a higher success rate of 77%

Payload vs. Launch Site



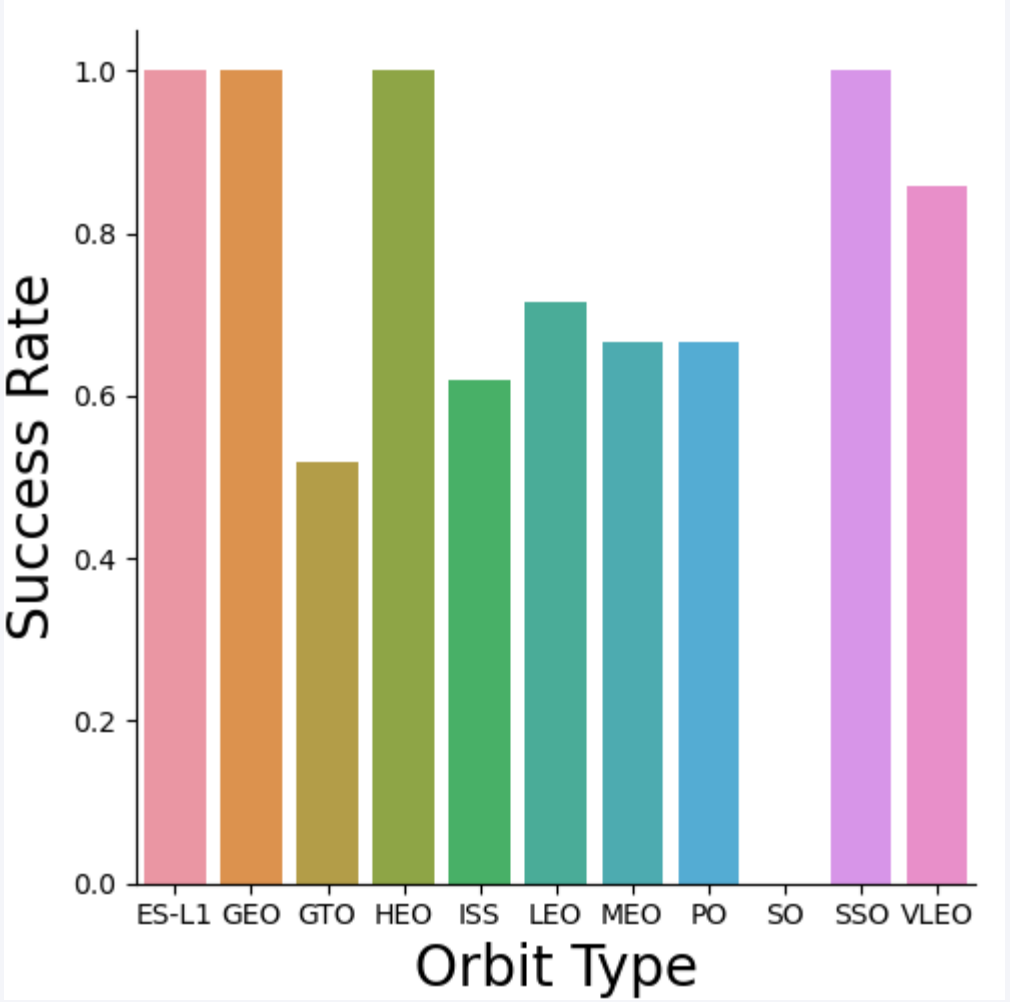
CCAFS SLC 40 has a strong success rate for heavy payload launches.

In contrast, the VAFB SLC launch site has not conducted any launches for payloads exceeding 10,000 kg.

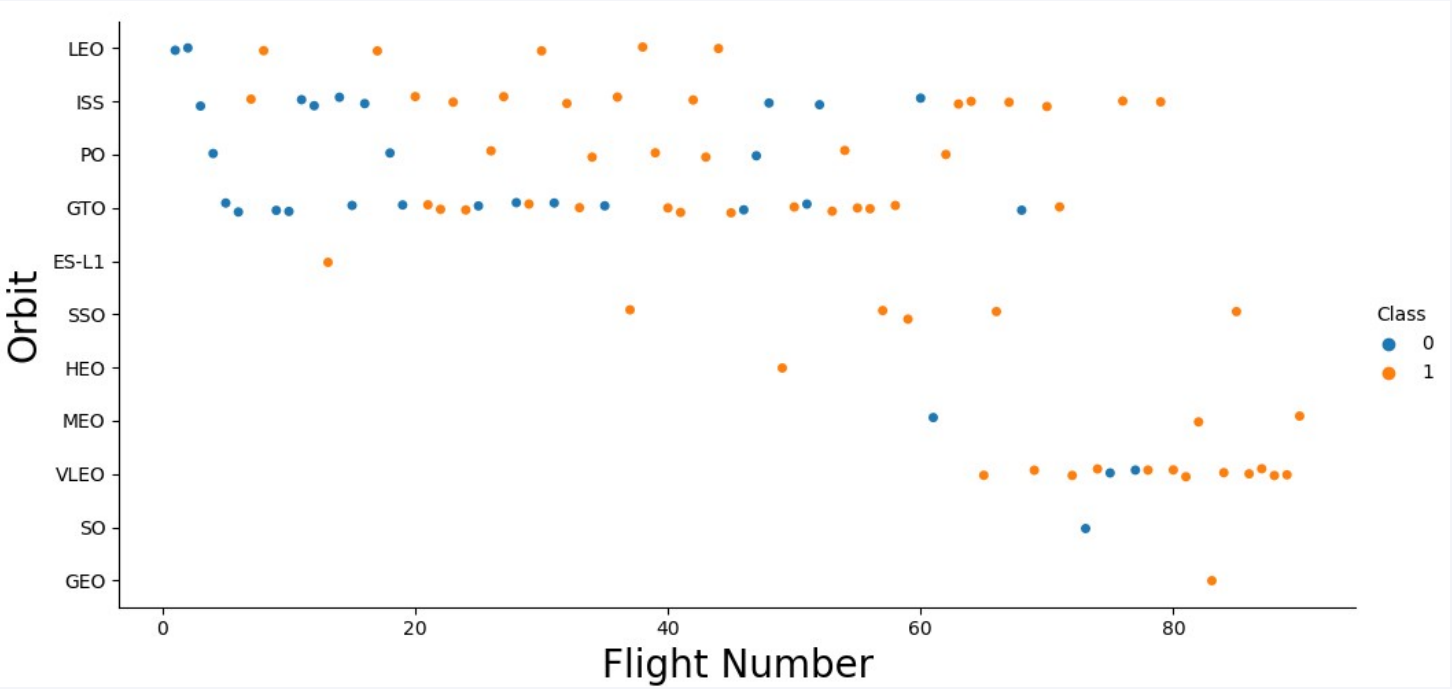
KSC LC 39A generally demonstrates a good success rate for both light and heavy payload launches.

Success Rate vs. Orbit Type

- The orbit types ES-L1, GEO, HEO, and SSO each have a 100% success rate.
- The orbit types GTO, ISS, LEO, MEO, PO, and VLEO have success rates ranging from 50% to 85%.
- In contrast, SO has a 0% success rate in landings.

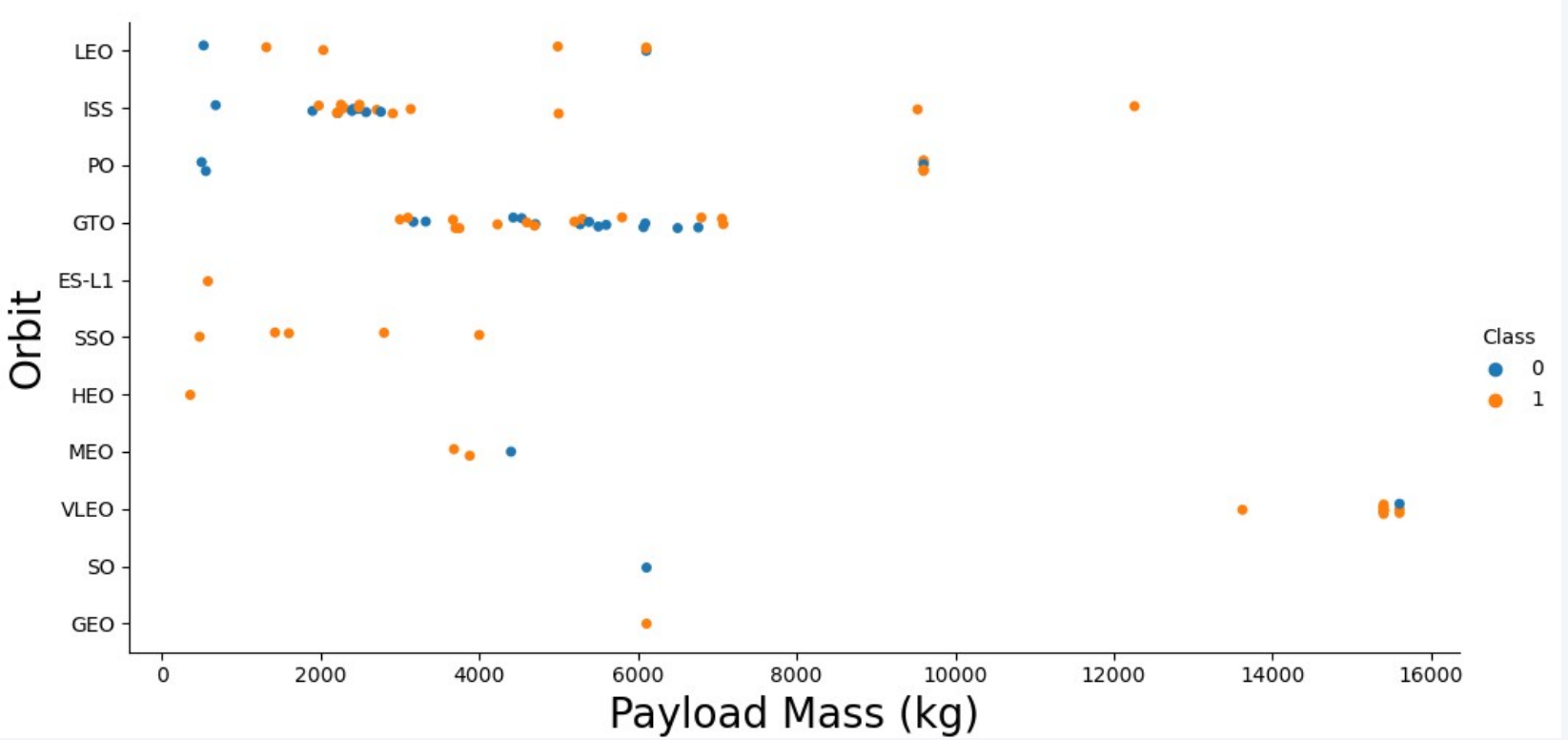


Flight Number vs. Orbit Type



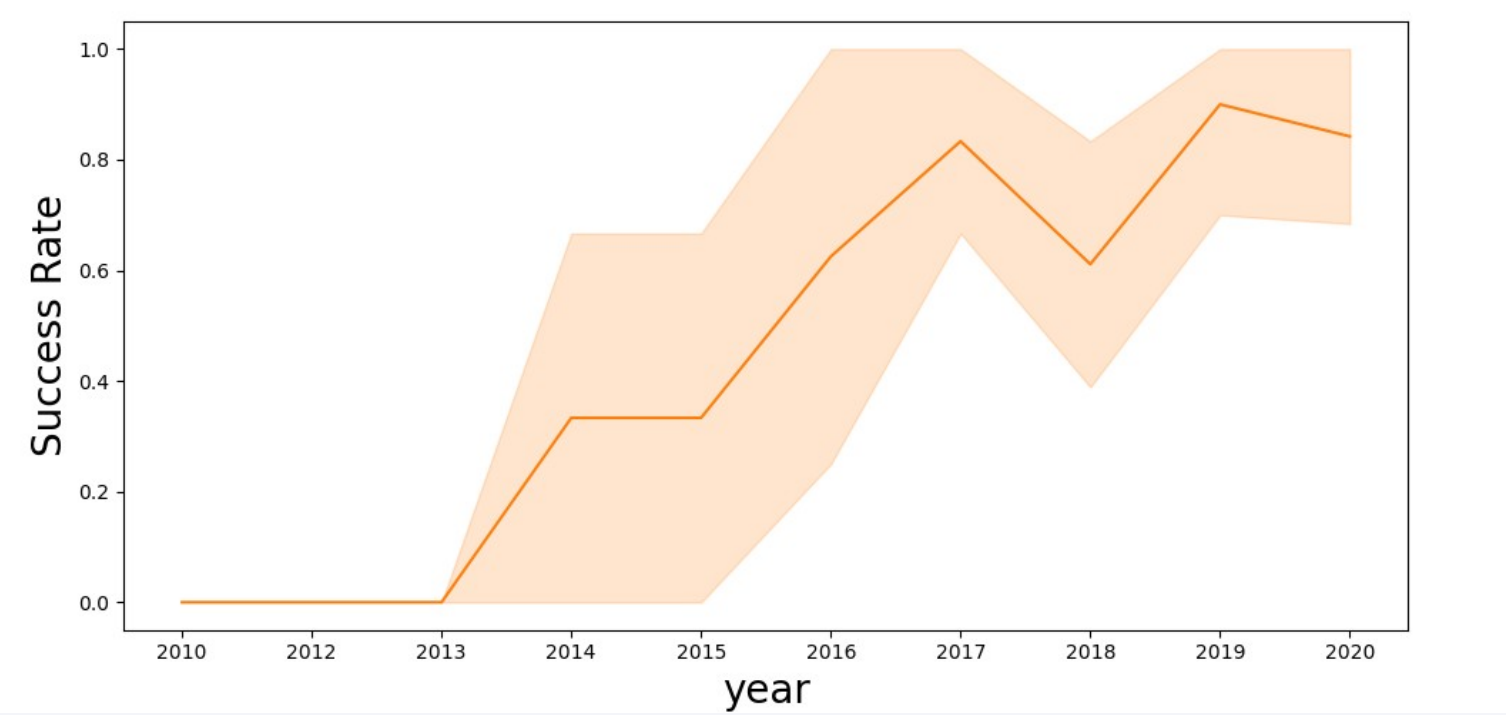
- The VLEO orbit has a strong success rate across over 60 flights. GTO is the most frequently tested orbit.
- ISS shows (maybe) a pattern between the number of flights and success rates.
- In most orbits, an increase in the number of flights is associated with a higher success rate.

Payload vs. Orbit Type



- We can observe that for heavy payloads, successful landings are more frequent in the VLEO and LEO orbits
- GTO does not demonstrate a clear success rate in relation to incremental payload mass
- All other orbit types do not demonstrate a clear relationship between success rates and heavy payload masses.

Launch Success Yearly Trend



- The line chart illustrates a consistent increase in the success rate from 2013 to 2020
- For an overview of the charts, please see:
- https://github.com/thefigaro/IBM_SkillsBuild_assignments/blob/main/Data Science Capstone/05_SpaceX_EDA_Data_Visualization.ipynb

All Launch Site Names

```
[11]: %sql SELECT DISTINCT Launch_site from SPACEXTABLE;
 * sqlite:///my_data1.db
Done.
[11]:
```

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

To retrieve unique launch site names, we execute the query: SELECT DISTINCT Launch_site FROM SPACETABLE

Launch Site Names Begin with 'CCA'

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

[12]:
```
%sql SELECT * from SPACEXTABLE WHERE Launch_site LIKE "CCA%" LIMIT 5;
```

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

[12]:

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

To identify launch site names that begin with 'CCA', we use the LIKE operator with the wildcard % in the expression 'CCA%', which matches any names starting with 'CCA'. Additionally, we limit the results to the first 5 entries from the query.

Total Payload Mass

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

```
[13]: %sql SELECT sum(PAYLOAD_MASS__KG_) AS "Total Payload mass" FROM SPACEXTABLE WHERE Customer="NASA (CRS)";
 * sqlite:///my_data1.db
Done.
```

[13]:

Total Payload mass
45596

- To calculate the total payload mass, we sum all entries in the PAYLOAD_MASS__KG field using the query SELECT SUM(PAYLOAD_MASS__KG).
- This computation is specifically for 'NASA (CRS)'. We also assign an alias to the result as 'Total Payload Mass

Average Payload Mass by F9 v1.1

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

[14]:
```
%%sql
SELECT
    AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass"
FROM
    SPACEXTABLE
WHERE
    Booster_Version LIKE "F9 v1.1%"
    ;
```

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

[14]:

Average Payload Mass
2534.6666666666665

- To calculate the average payload mass for the F9 v.1.1 version, we use the AVG function on the PAYLOAD_MASS__KG field, applying a condition to include only those records where the Booster_Version field matches strings that start with 'F9 v1.1' using the wildcard '%'.
- We also assign an alias to the result as 'Average Payload Mass'

First Successful Ground Landing Date

```
[16]: %%sql
SELECT
    min(Date) AS "first succesful landing outcome in ground pad"
FROM
    SPACEXTABLE
WHERE
    Landing_Outcome = "Success (ground pad)"
;
```

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

[16]:

first succesful landing outcome in ground pad
2015-12-22

To determine the date of the first successful ground landing, we calculate the minimum date where the Landing_Outcome matches 'Success (ground pad)'.

To enhance clarity, we assign an alias to the result using 'AS', labeling it as 'First Successful Landing Outcome in Ground Pad

Successful Drone Ship Landing with Payload between 4000 and 6000

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

[17]:
```
%%sql
SELECT
    Booster_Version
FROM
    SPACEXTABLE
WHERE
    PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000
    and
    Landing_Outcome = "Success (drone ship)"
    ;
```

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

[17]:

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

To identify successful drone ship landings with payloads between 4000 and 6000 kg, we select the Booster_Version from the SPACEXTABLE.

The query filters records where the PAYLOAD_MASS__KG field is BETWEEN 4000 AND 6000, and the Landing_Outcome field matches the string 'Success (drone ship)'

Total Number of Successful and Failure Mission Outcomes

[19]:
```
%%sql
SELECT
    CASE
    WHEN Mission_Outcome LIKE 'Success%' THEN 'success'
    ELSE
    'fail'
    END OUTCOMES , count(*) as TOTAL_OUTCOMES
    FROM SPACEXTABLE
    GROUP BY OUTCOMES
    ;
```

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

[19]:

OUTCOMES	TOTAL_OUTCOMES
fail	1
success	100

To calculate the total number of successful and failed outcomes, we first categorize the cases. We count entries in the Mission_Outcome field that start with 'Success' as 'success', while all other outcomes are counted as 'fail'.

We then group these counts into 'success' and 'fail'. To enhance clarity, we assign aliases to the results, naming the first column 'OUTCOMES' and the second column 'TOTAL_OUTCOMES'.

Boosters Carried Maximum Payload

To identify the boosters that carried the maximum payload, we use the query to SELECT the maximum value in the PAYLOAD_MASS__KG field FROM the SPACEXTABLE, grouping the results by the Booster_Version field in descending order and limiting the output to the top 12 results.

[21]:
```
%%sql
-- alternativa
SELECT
    Booster_Version, max(PAYLOAD_MASS__KG_) as max_payload
FROM
    SPACEXTABLE
GROUP BY
    Booster_Version
ORDER BY
    max_payload desc
LIMIT 12
    ;
```

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

[21]:

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

2015 Launch Records

To identify all launches that failed in 2015, we first extract the month from the Date field by taking the substring starting from the 6th character with a length of 2 characters.

We then filter the results where the Landing_Outcome starts with 'Fail' and contains the string 'drone ship'.

Finally, we ensure that the first 5 characters of the Date field correspond to the year '2015'

In [19]:

```
%%sql
SELECT
    substr(Date,6,2) as Month, Landing_Outcome, Booster_Version, launch_site
FROM
    SPACEXTABLE
WHERE
    Landing_Outcome like "Fail%drone ship%"
    and
    substr(Date,0,5)='2015'
;
```

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

Out[19]:

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

Ranking All Landing Outcomes Between 2010-06-04 and 2017-03-20

- Data Selection: We chose two things from our data: the "Landing outcomes" and the number of those outcomes (COUNT).
- Filtering: We used a filter (the WHERE clause) to only include landing outcomes that occurred between June 4, 2010, and March 20, 2010.)
- Grouping: We grouped the landing outcomes together using the GROUP BY clause. This means we combined similar outcomes to see how many there are of each type.
- Ordering: Finally, we arranged the grouped landing outcomes in descending order (from highest to lowest) using the ORDER BY clause.

```
[25]: %%sql
SELECT
Landing_Outcome, count(*) as N_Outcomes
FROM
    SPACEXTABLE
WHERE
    Date BETWEEN '2010-06-04' and '2017-03-20'
GROUP BY
    Landing_Outcome
ORDER BY
    N_Outcomes desc
;
```

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

[25]:

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

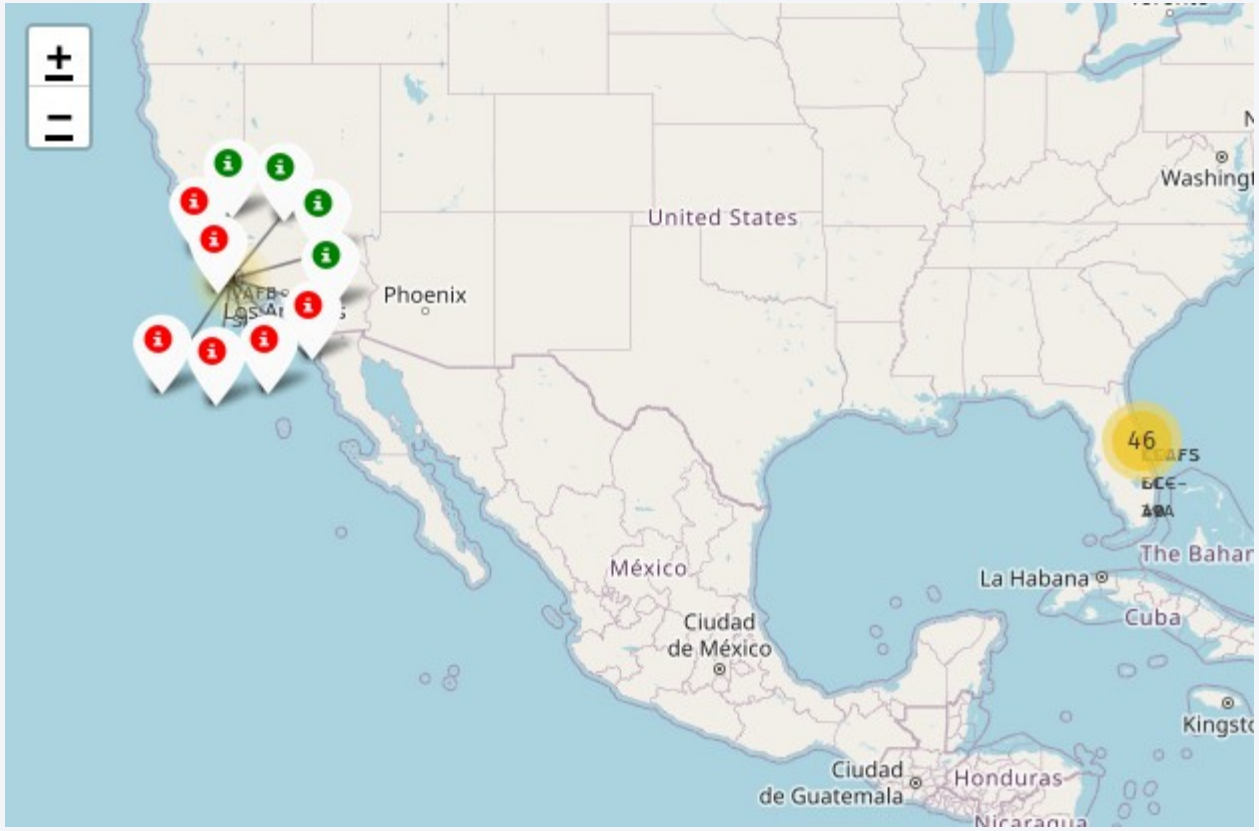
Section 3

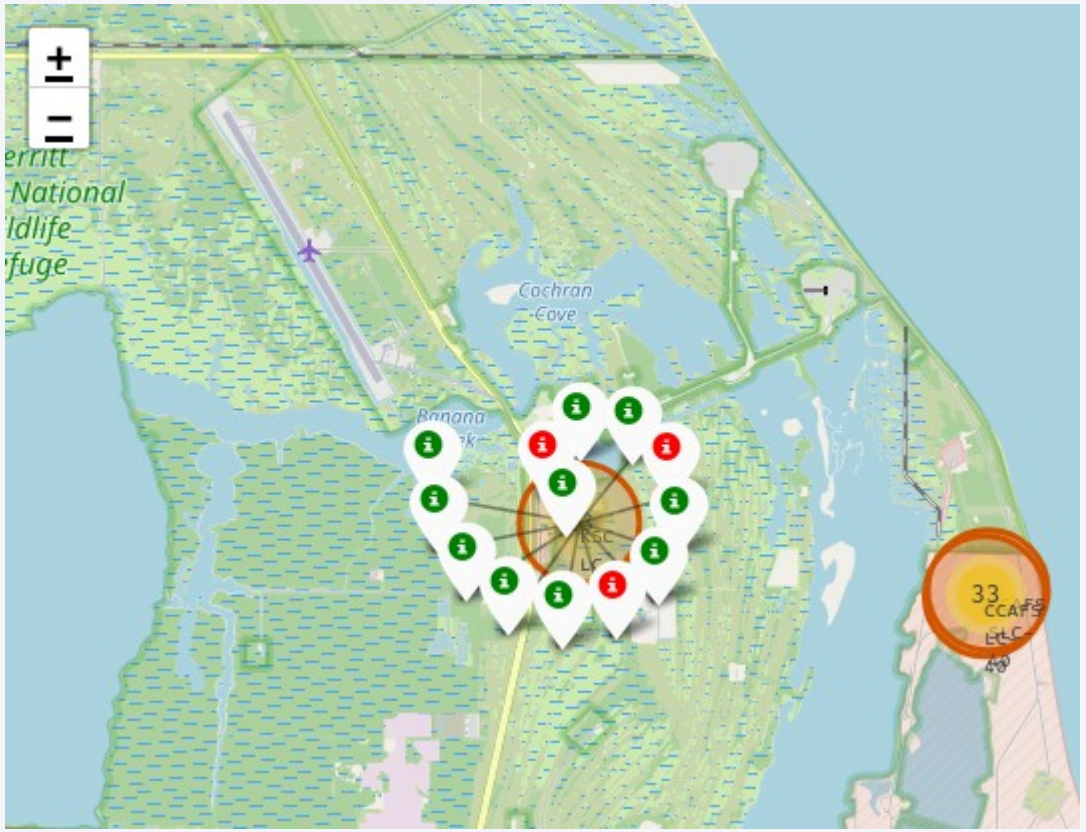
Launch Sites Proximities Analysis

Launch SpaceX sites map markers

The map shows the launch sites are located in the southern coast of the United States.

Success and Failure by Launch Locations

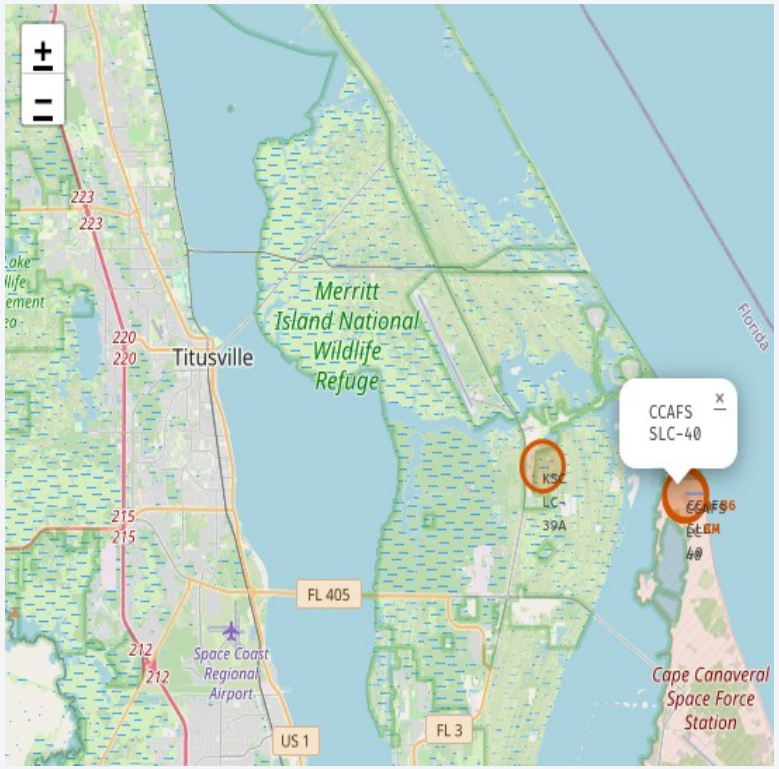




The left side of the map indicates points of success and failure, with **red (i)** representing failure and **green (i)** representing success.

The right side of the map provides a closer view of the launch sites located along the southeastern coast of the United States.

Calculating proximities

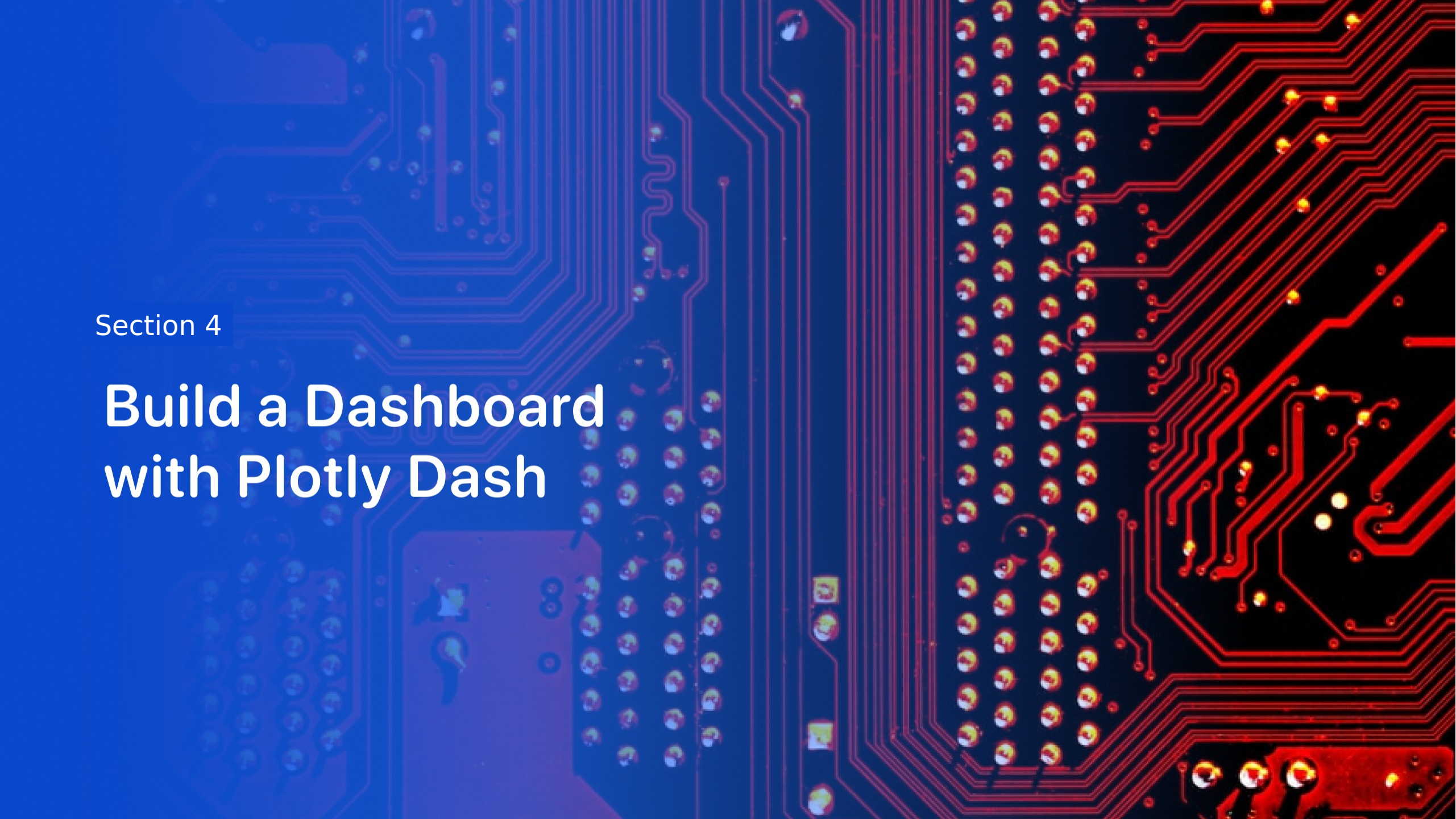


The **blue** line represents a distance of 0.86 km from the launch site CCAFS SLC-40 in Florida to the nearest coast.

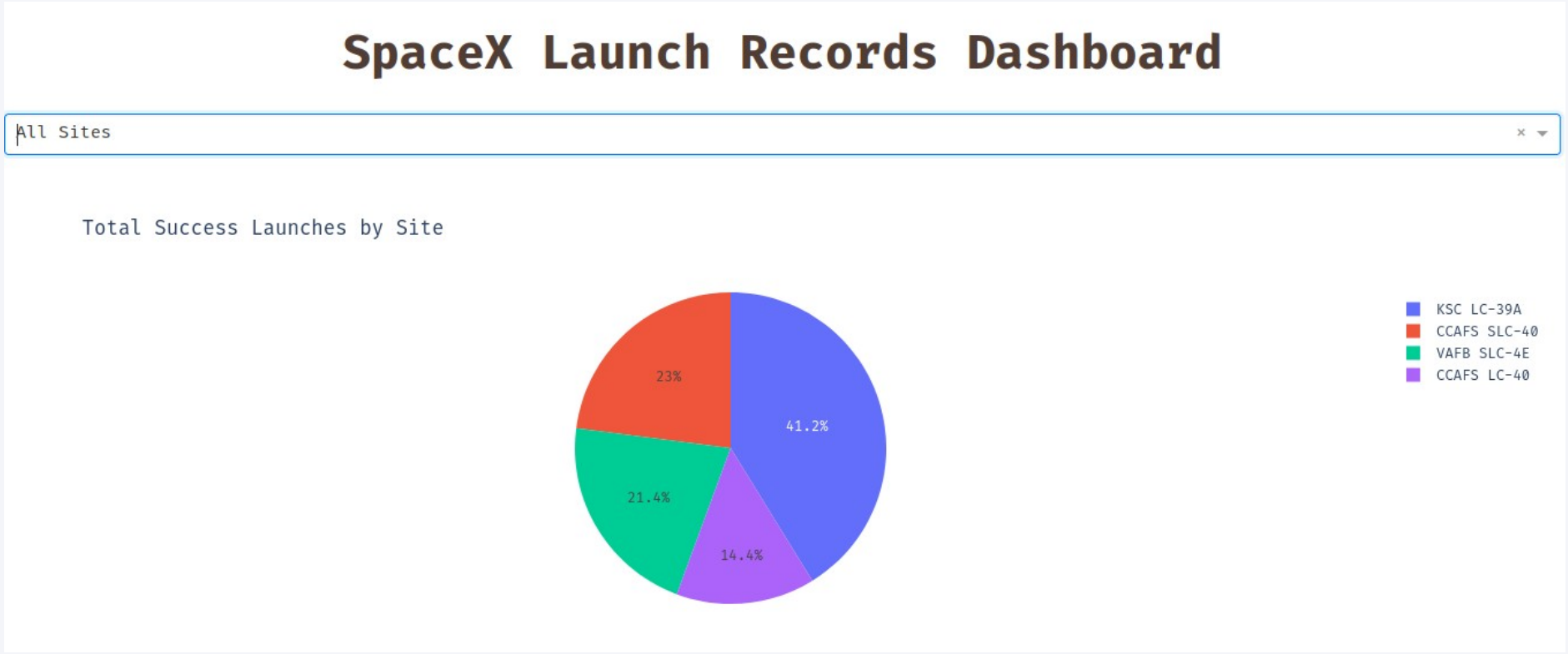
The **red** line indicates a distance of 23.23 km from CCAFS SLC-40 to Titusville, the closest town.

The **yellow** line shows a distance of 21.96 km from CCAFS SLC-40 to the nearest railway.

Lastly, the **green** line illustrates the approximate distance of 26.88 km from CCAFS SLC-40 to the nearest highway.

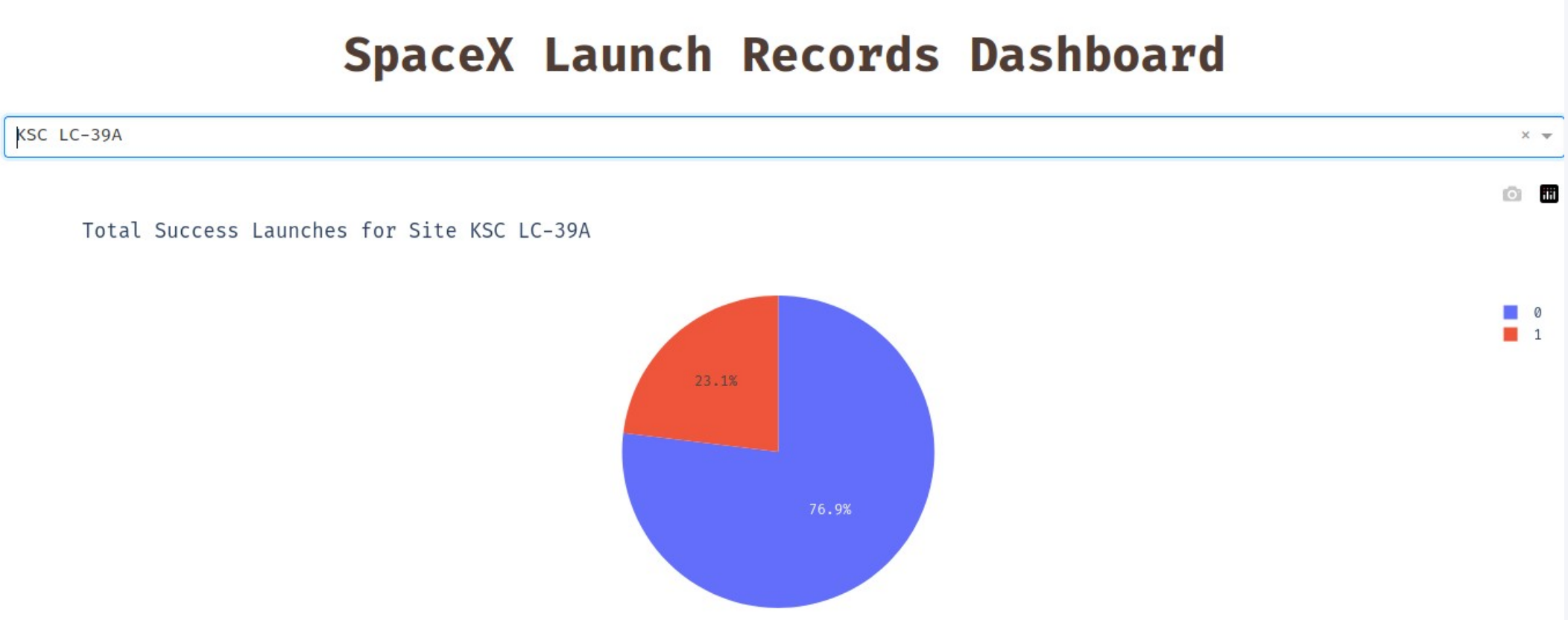

Section 4
Build a Dashboard with Plotly Dash

Success rates for each launch site



- The KSC LC-39A is the launch site with the highest success rate at 41.2%.
- Following closely is CCAFS SLC-40, which has a success rate of 23.0%.
- In third place is VAFB SLC-4E with a success rate of 21.4%.
- Lastly, CCAFS LC-40 ranks fourth with the lowest success rate at 14.4%

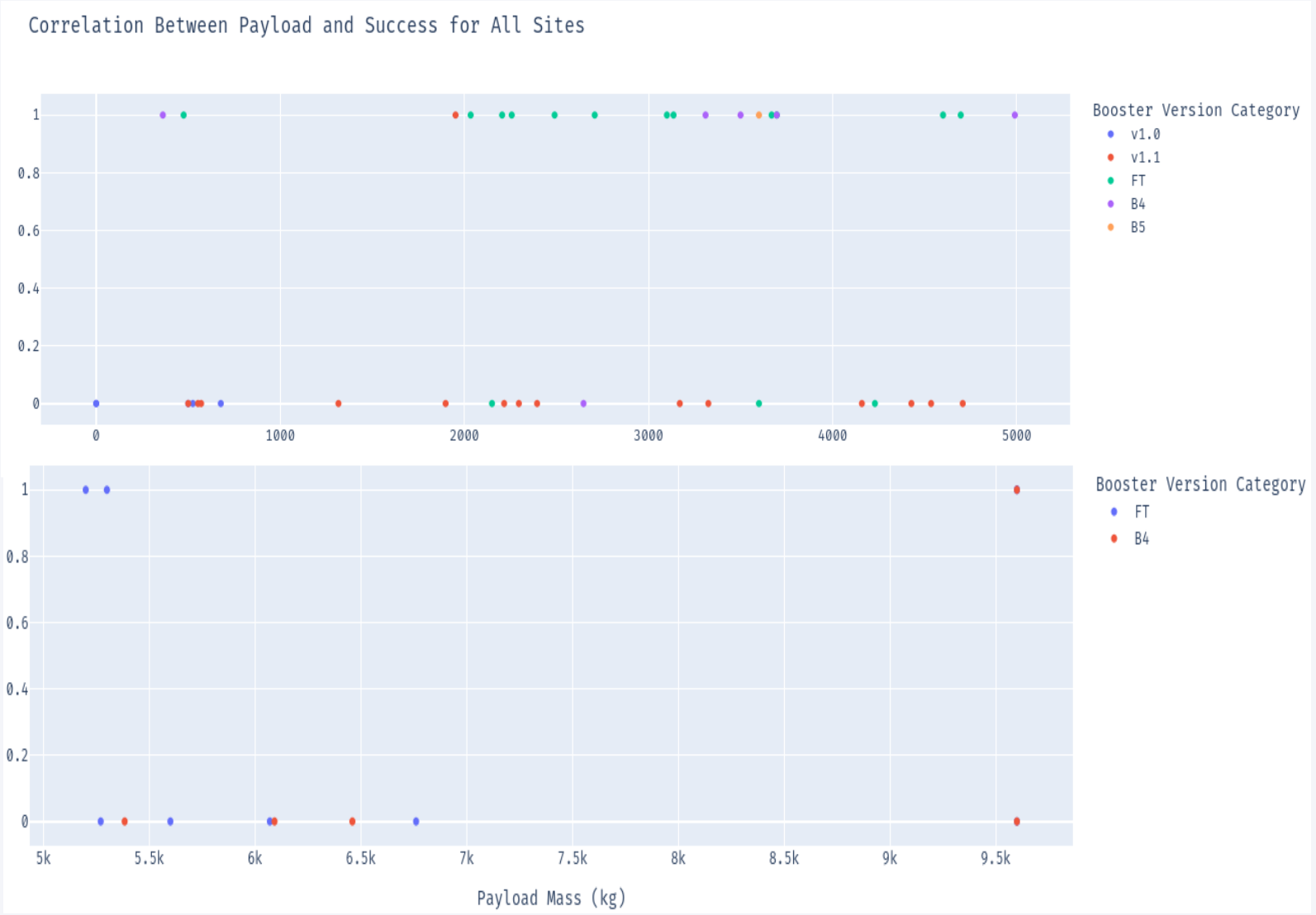
KSC LC-39A is the launch site with the highest success rate



KSC LC-39A has achieved a success rate of 76.9% for its rocket launches, with a failure rate of 23.1%.

1 indicates success, while 0 indicates failure.

Scatter plot of correlations between Payload and Launch Outcome for all sites, with a range slider to select different payloads



In a payload range from 0 to 5000 kg, there is a strong correlation between successful landings and the FT booster version category.

In contrast, other booster version categories do not show a significant correlation with the launch sites.

For payloads exceeding 5000 kg, no correlation is observed among the booster version categories.

In summary, success rates are more closely associated with lower payload masses than with heavier ones

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
[42]: models = {
          'Logistic Regression': logreg_cv,
          'Support Vector Machine': svm_cv,
          'Decision Tree': tree_cv,
          'K-Nearest Neighbors': knn_cv
      }

      best_score_models={i: j.best_score_ for i, j in models.items()}
      df=pd.DataFrame(best_score_models.values(), index=best_score_models.keys(), columns=['Best Score'])
      df.style.highlight_max(color='yellow')
```

[42]:

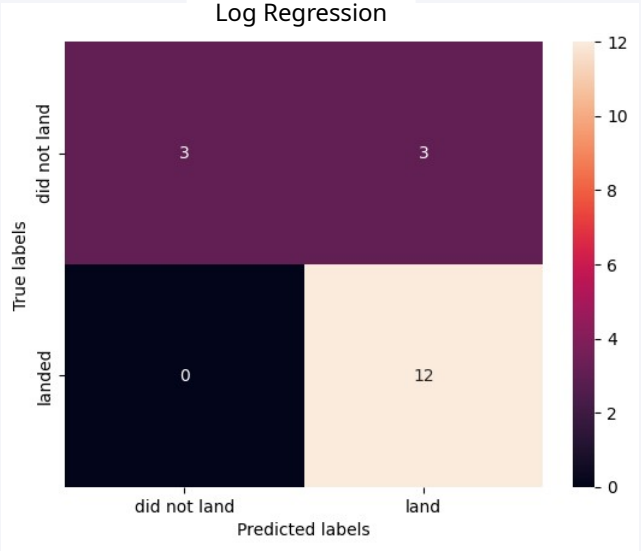
	Best Score
Logistic Regression	0.846429
Support Vector Machine	0.848214
Decision Tree	0.862500
K-Nearest Neighbors	0.848214

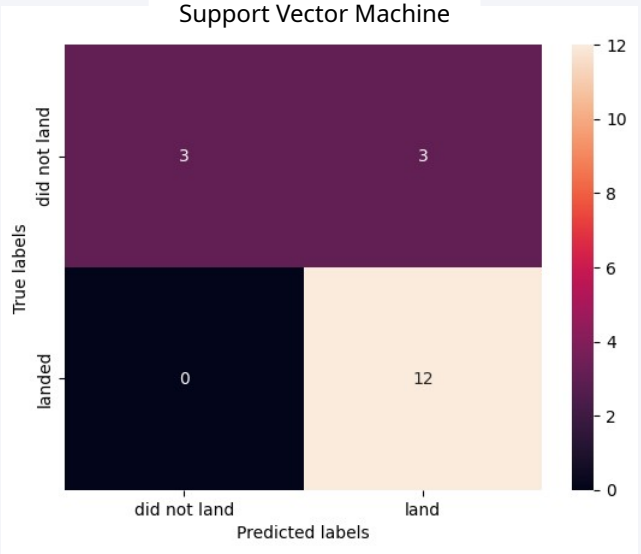
We have developed four predictive models to determine whether the first stage will land, using data from the preceding labs. The models include:

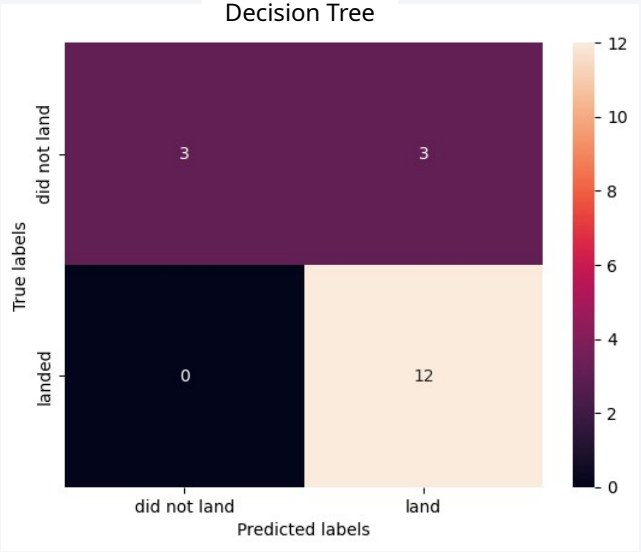
1. **log_cv** (Logistic Regression)
2. **svm_cv** (Support Vector Machine)
3. **tree_cv** (Decision Tree Classifier)
4. **knn_cv** (K-Nearest Neighbors)

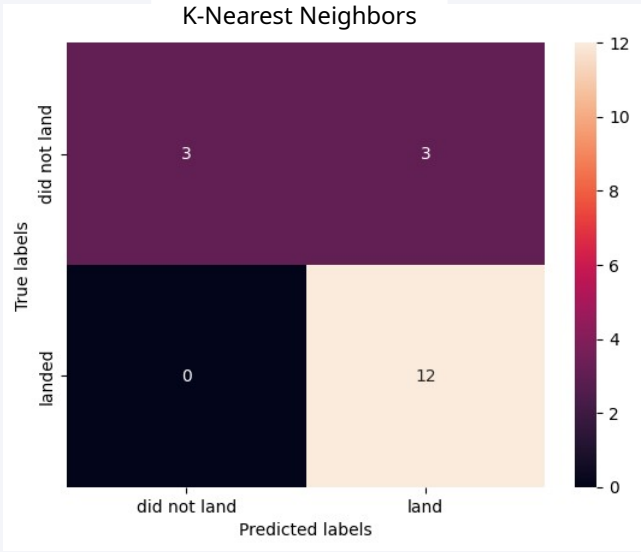
For each of these models, we computed the best tuning parameters. Ultimately, we found that the **tree_cv** (Decision Tree Classifier) achieved the highest performance with 0.8625 score.

Confusion Matrix









- The performance of all 4 classification models are the same
- 12 true, positives (TP)
- 3 true, negatives (TN)
- 3 false, positives (FP)
- 0 false, negatives (FN)
- Precision = TP/(TP + FP)

  12 / 15 = 0.80
- Recall = TP/ (TP + FN)

  12 / 12 = 1.0
- F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

  2 * (0.80 * 1.0) / (0.80 + 1.0) = 0.888889
- Accuracy = (TP + TN)/ (TP + TN + FP + FN)

  (12 + 3) / (12 + 3 + 3 + 0) = 0.8333333333333334

Conclusions

- SpaceX launch sites are located along the southern coast of the United States
- The orbit types ES-L1, GEO, HEO, and SSO each have a 100% success rate
- The KSC LC-39A is the launch site with the highest success rate at 41.2%
- From 2010 to 2020, the success rate began to increase starting in 2013.
- In a payload range from 0 to 5000 kg, there is a strong correlation between successful landings and the FT booster version category.
- The Decision Tree Classifier (tree_cv) achieved the best performance among all the models we evaluated, with a score of 0.8625.

Appendix

Here are some helpful web pages that provide Jupyter Notebooks

- **Official Jupyter Documentation:** Jupyter Installation

  (This page provides comprehensive instructions on how to install Jupyter Notebook using various methods, including Anaconda and pip).

- **Anaconda Distribution:** Anaconda Installation

  (Anaconda is a popular distribution that includes Jupyter Notebook and many other useful packages for data science. This page guides you through the installation process.

- **Using pip:** Installing Jupyter with pip

  This Real Python article explains how to install Jupyter Notebook using pip, along with a brief introduction to using Jupyter.

- **Google Colab:** Google Colab

  While not a local installation, Google Colab allows you to run Jupyter notebooks in the cloud for free. You can create and share notebooks without any installation.

- **Jupyter Notebook Quick Start:** Jupyter Notebook Quick Start

  This DataCamp tutorial provides a quick start guide to installing and using Jupyter Notebook, including installation steps and basic usage.

■ **To access all the Jupyter notebooks used in this project, please click the link below.**

https://github.com/thefigaro/IBM_SkillsBuild_assignments/tree/main/Data Science Capstone

Thank you!