IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ichsan Haikal
14 August 2023

Outline

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

Executive Summary

- Summary of methodologies
  - SpaceX Data Collection using SpaceX API
  - SpaceX Data Collection with Web Scraping
  - SpaceX Data Wrangling
  - SpaceX Exploratory Data Analysis using SQL
  - Space-X EDA DataViz Using Python Pandas and Matplotlib
  - Space-X Launch Sites Analysis with Folium-Interactive Visual Analytics and Ploty Dash
  - SpaceX Machine Learning Landing Prediction
- Summary of all results
  - EDA results
  - Interactive Visual Analytics and Dashboards
  - Predictive Analysis(Classification)

Introduction

- Project background and context

SpaceX promotes Falcon 9 rocket missions on their website at a price of $62 million, whereas other competitors charge upwards of $165 million per launch. Much of these cost savings can be attributed to SpaceX's ability to recover and reuse the initial stage of the rocket. Consequently, assessing whether the first stage will successfully land allows us to ascertain the launch cost. This data becomes valuable for potential rival companies aiming to compete with SpaceX in bidding for rocket launch contracts.

- Problems you want to find answers

In this final project, we aim to make predictions about the successful landing of the Falcon 9 first stage. We will achieve this by analyzing data sourced from Falcon 9 rocket launches that are publicly advertised on the company's website.

Section 1

Methodology

Methodology

Executive Summary

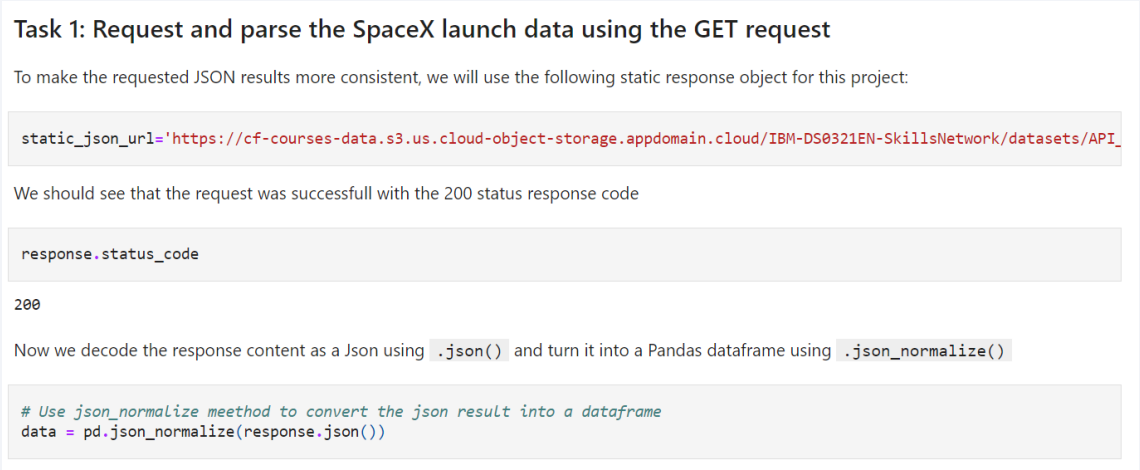
- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
  - The data collection process began by utilizing the SpaceX API, a RESTful API. This involved initiating a GET request to the SpaceX API, facilitated through a set of auxiliary functions. These functions were designed to enable the extraction of specific information from the launch data by utilizing unique identification numbers. Consequently, the rocket launch data was obtained from the designated SpaceX API URL.
  - To enhance the consistency of the acquired JSON results, the SpaceX launch data was retrieved through a GET request. The response content was then decoded as a JSON outcome. This JSON result was subsequently transformed into a Pandas data frame for further analysis.
  - Additionally, web scraping techniques were employed to gather historical Falcon 9 launch records from a Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches." These launch records were embedded within HTML content. By leveraging the BeautifulSoup and request libraries, the Falcon 9 launch table records were extracted from the Wikipedia page. The process involved parsing the table and converting it into a Pandas data frame for subsequent use.

Data Collection – SpaceX API

- The data was gathered utilizing the SpaceX API, a RESTful API, through a GET request made to the SpaceX API. The subsequent step involved acquiring and parsing the SpaceX launch data using this GET request. The response content was decoded as a JSON result, which was further transformed into a Pandas data frame for analysis.
- Here is the GitHub URL for the completed notebook containing SpaceX API calls: https://github.com/IchsanHaikal/coursera/blob/cd7f5e1c2cbd843db41e54584e79adc10c0757f8/IBM%20Applied%20Data%20Science%20Capstone/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:

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

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

```
response.status_code
```

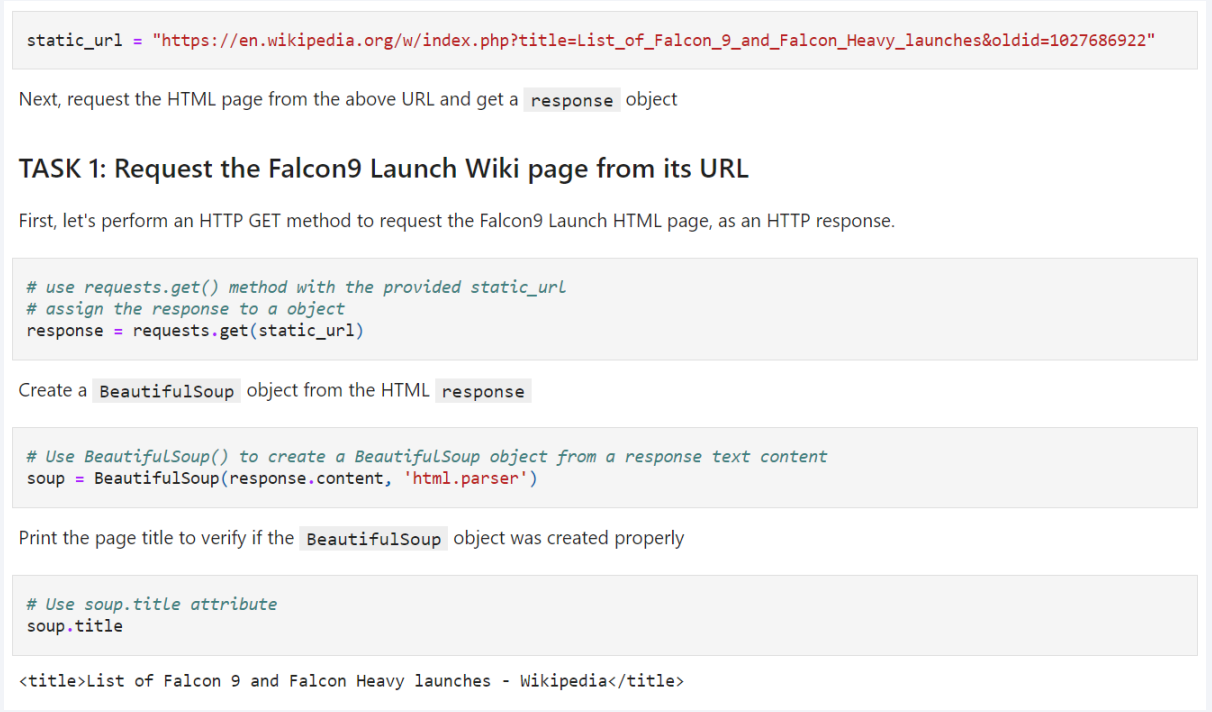
```
200
```

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

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

Data Collection - Scraping

- Conducted web scraping to gather past Falcon 9 launch records from a Wikipedia page using BeautifulSoup and requests libraries. The goal was to extract the Falcon 9 launch data from the HTML table within the Wikipedia page. Following this extraction process, a data frame was generated by parsing the launch-related HTML information.
- Here is the GitHub URL for the web scrapinglab: https://github.com/IchsanHaikal/coursera/blob/cd7f5e1c2cbd843db41e54584e79adc10c0757f8/IBM%20Applied%20Data%20Science%20Capstone/jupyter-labs-webscraping.ipynb



```
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.

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

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

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

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

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data Wrangling

- Once the data was acquired and a Pandas data frame was constructed, a filtering process was executed utilizing the 'BoosterVersion' column. This filtering aimed to retain exclusively the Falcon 9 launches. Subsequently, attention was directed towards addressing the missing data entries in the 'LandingPad' and 'PayloadMass' columns.
- To address the missing values in the 'PayloadMass' column, a strategy was adopted whereby these gaps were filled using the mean value of the column.
- Furthermore, an Exploratory Data Analysis (EDA) was conducted to uncover potential patterns within the dataset. This analysis was instrumental in identifying trends that could serve as the foundation for selecting an appropriate label for training supervised models.
- Here is the GitHub URL for the data wrangling lab: https://github.com/IchsanHaikal/coursera/blob/cd7f5e1c2cbd843db41e54584e79adc10c0757f8/IBM%20Applied%20Data%20Science%20Capstone/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

Data Wrangling

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
df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df['Class'].value_counts()
```

```
1    60
0    30
Name: Class, dtype: int64
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
landing_class=df['Class']
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0

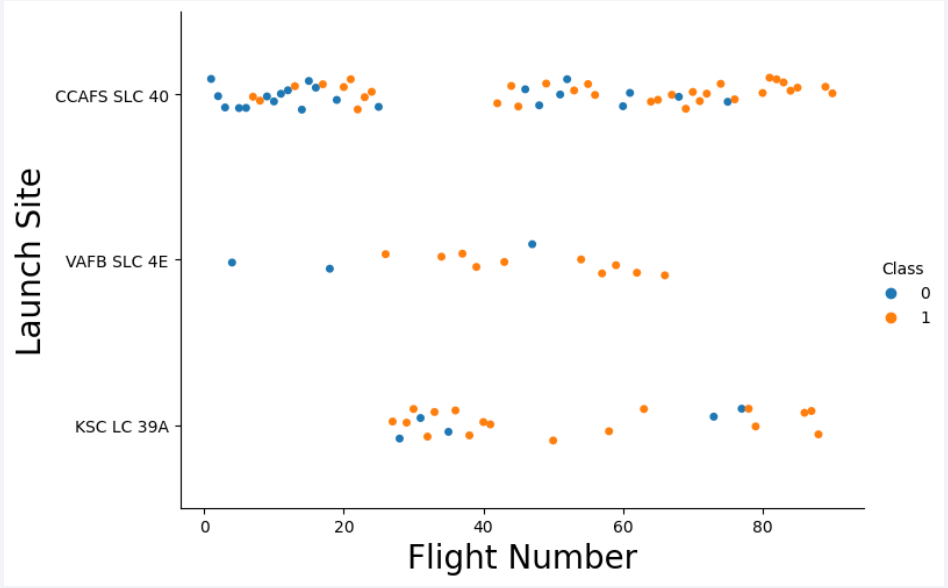
EDA with Data Visualization

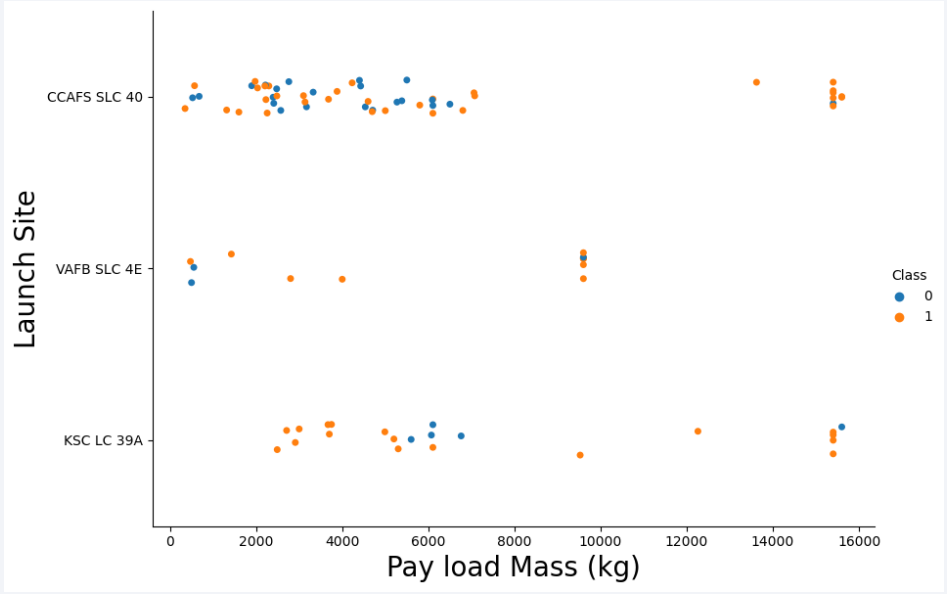
Conducted data analysis and feature engineering using Pandas and Matplotlib, encompassing the following steps:

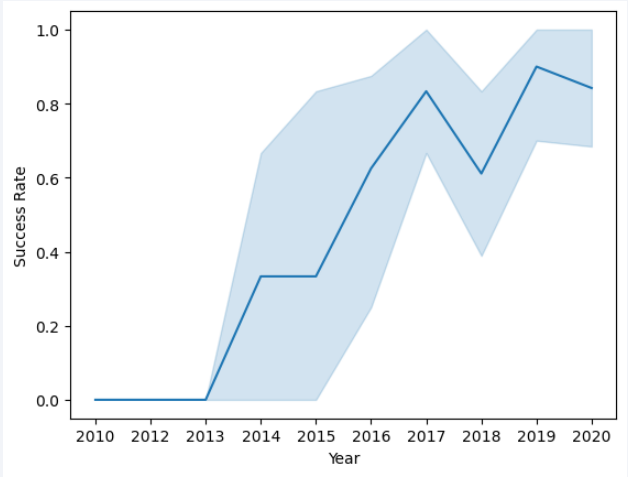
- Carried out Exploratory Data Analysis (EDA)
- Prepared and performed feature engineering on the data
- Employed scatter plots for visualizing correlations between Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit Type, as well as Payload and Orbit Type
- Utilized bar charts to illustrate the connection between the success rate of each orbit type
- Employed a line plot to visualize the annual trend in launch success.

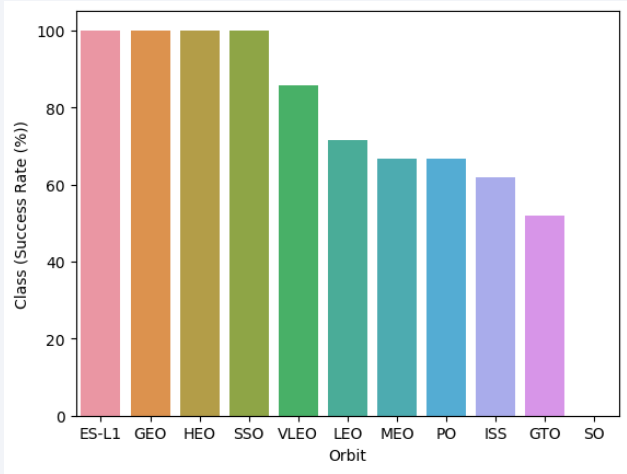
Here is the GitHub URL for the EDA with Data Visualization lab:
https://github.com/IchsanHaikal/coursera/blob/main/IBM%20Applied%20Data%20Science%20Capstone/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with Data Visualization









EDA with SQL

- The subsequent SQL queries were executed as part of the Exploratory Data Analysis (EDA):
  - Retrieve the distinct names of launch sites involved in space missions.
  - Show five entries where launch sites start with the prefix 'CCA.'
  - Present the cumulative payload mass carried by boosters launched under NASA's CRS category.
  - Demonstrate the mean payload mass transported by booster version F9 v1.1.
- Here is the GitHub URL for the EDA with SQL lab: https://github.com/IchsanHaikal/coursera/blob/main/IBM%20Applied%20Data%20Science%20Capstone/jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with SQL

Task 1

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

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

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

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.
```

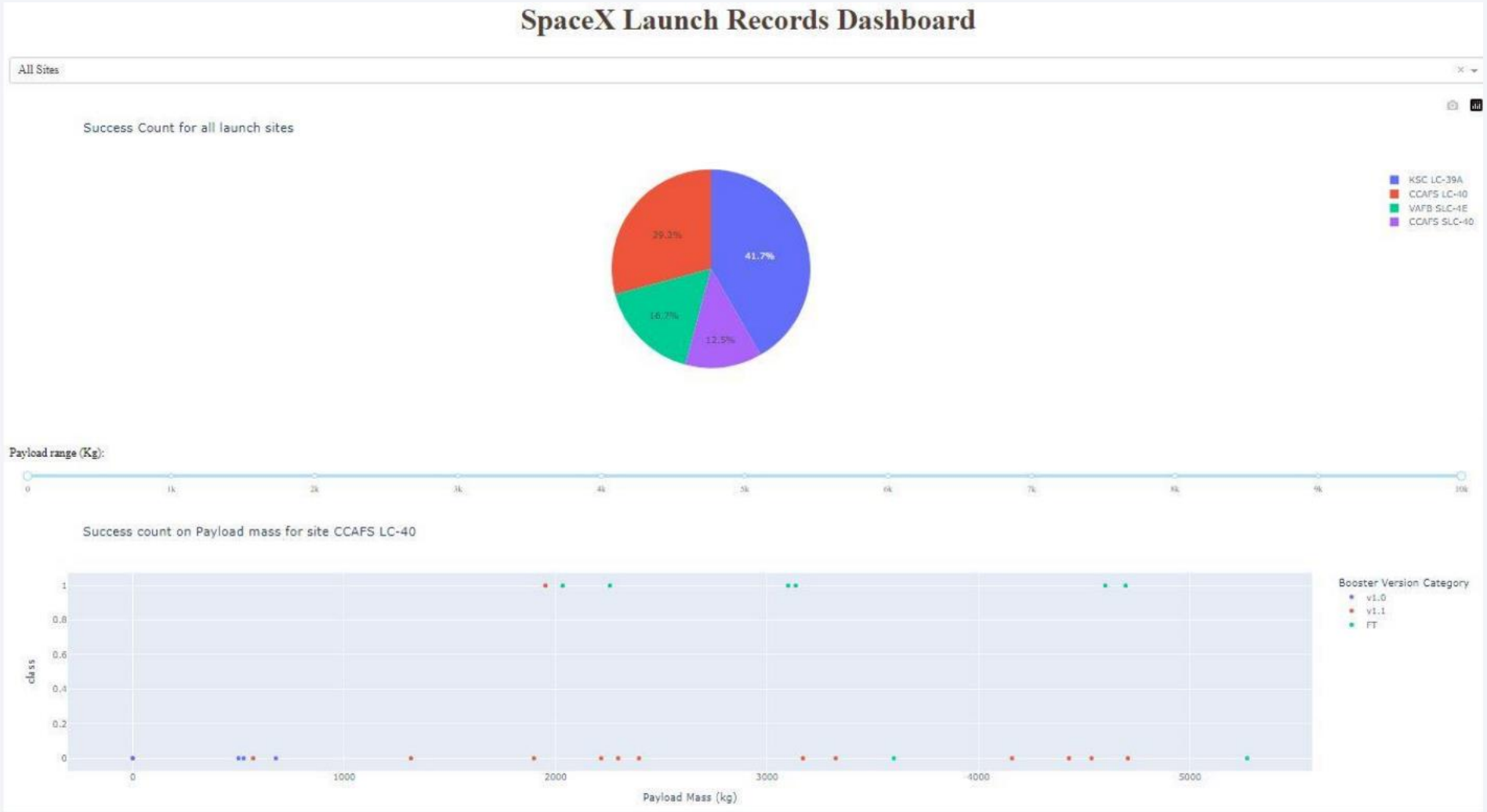
Build an Interactive Map with Folium

- Developed a Folium map to pinpoint all launch sites, generating map components like markers, circles, and lines to indicate the outcomes (success or failure) of launches at each specific launch site.
- Additionally, formulated a classification of launch outcomes, where 'failure' corresponds to a value of 0 and 'success' corresponds to a value of 1.
- Here is the GitHub URL for the Build an Interactive Map with Folium lab: https://github.com/IchsanHaikal/coursera/blob/main/IBM%20Applied%20Data%20Science%20Capstone/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Constructed an interactive dashboard application using Plotly Dash that encompassed the following steps:
  - Incorporated a dropdown input component for selecting launch sites.
  - Established a callback function to visualize a success pie chart contingent on the chosen launch site from the dropdown.
  - Integrated a range slider for payload selection.
  - Implemented a callback function to display a scatter plot depicting the success and payload relationship.
- Here is the GitHub URL for the Build a Dashboard with Plotly Dash lab: https://github.com/IchsanHaikal/coursera/blob/main/IBM%20Applied%20Data%20Science%20Capstone/spacex_dash_app.py

SpaceX Dash App



Predictive Analysis (Classification)

- Here's a summary of how I developed, assessed, enhanced, and identified the most effective classification model:
- Upon importing the data as a Pandas DataFrame, I initiated the process by conducting exploratory data analysis to define the training labels. This involved:
- Creating a NumPy array from the 'Class' column in the dataset using the to_numpy() method, which was then assigned to the variable 'Y' to serve as the target variable.
- Subsequently, I performed standardization on the feature dataset 'X' by applying the preprocessing.StandardScaler() function from the Sklearn library.
- Afterwards, the data underwent division into training and testing sets, facilitated by the train_test_split function from sklearn.model_selection. The test_size parameter was set to 0.2 to allocate 20% of the data for testing purposes, while the random_state was set to 2 to ensure reproducibility.

Predictive Analysis (Classification)

- To identify the most optimal ML model/method from SVM, Classification Trees, k-nearest neighbors, and Logistic Regression, the following approach was taken:
  - Created individual objects for each algorithm and established a GridSearchCV object.
  - Defined a set of parameters for each model within the GridSearchCV setup.
  - For each model being evaluated, a GridSearchCV object was instantiated with cross-validation (cv) set to 10. Subsequently, the training data was fitted to each GridSearch object to determine the best hyperparameters.
  - Following the training set fitting, the GridSearchCV object was outputted for each model. This enabled the extraction of the best parameters using the 'best_params_' data attribute, and the accuracy on the validation data using the 'best_score_' data attribute.
  - Finally, the 'score' method was employed to compute the accuracy on the test data for each model. Additionally, a confusion matrix was plotted for each model using the test outcomes and predicted results.

Predictive Analysis (Classification)

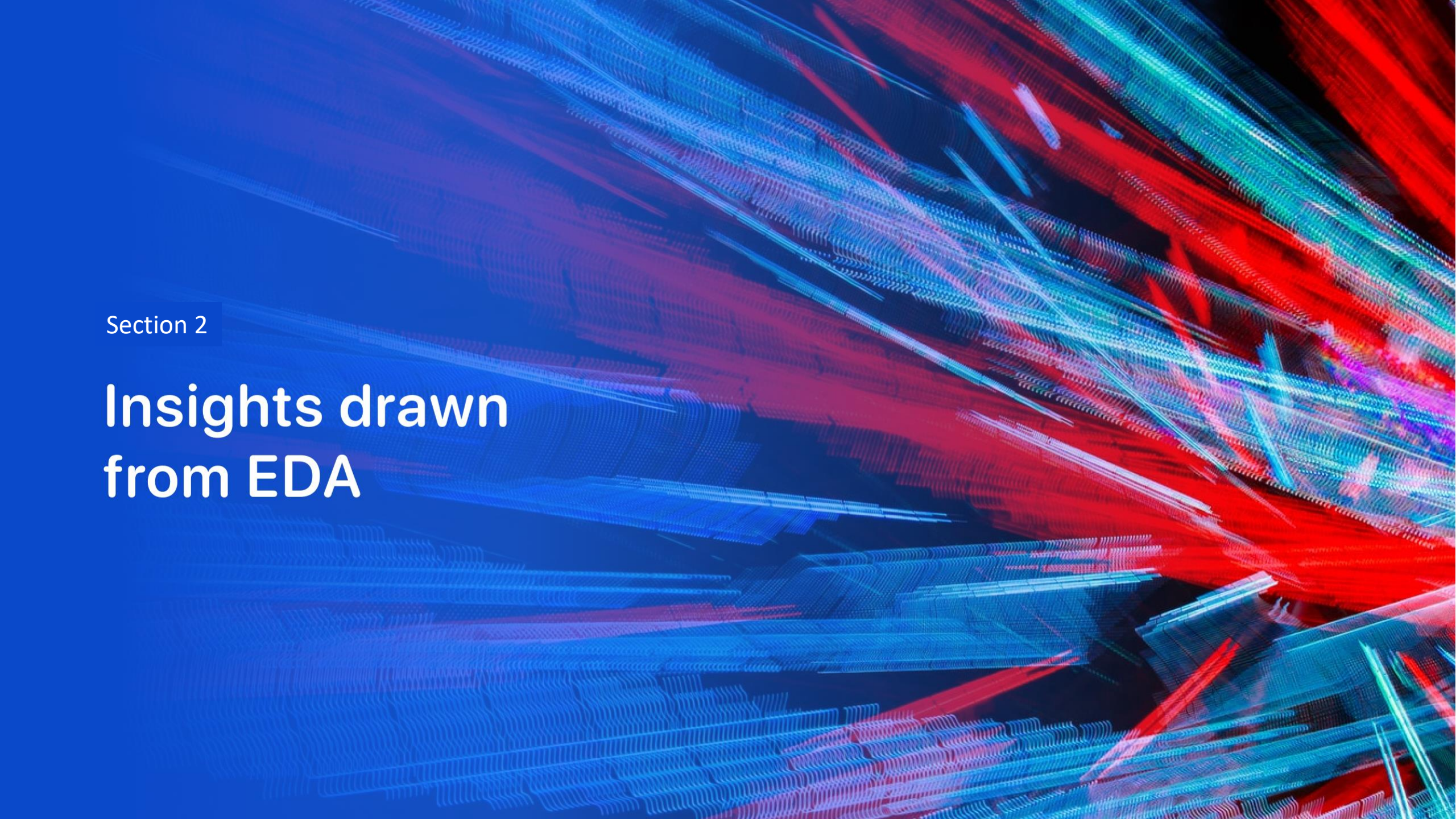
- The table presented below illustrates the accuracy scores on the test data for each of the methods. This comparison is intended to highlight the performance of SVM, Classification Trees, k-nearest neighbors, and Logistic Regression, enabling the determination of the most effective approach.

Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.722222
KNN	0.833333

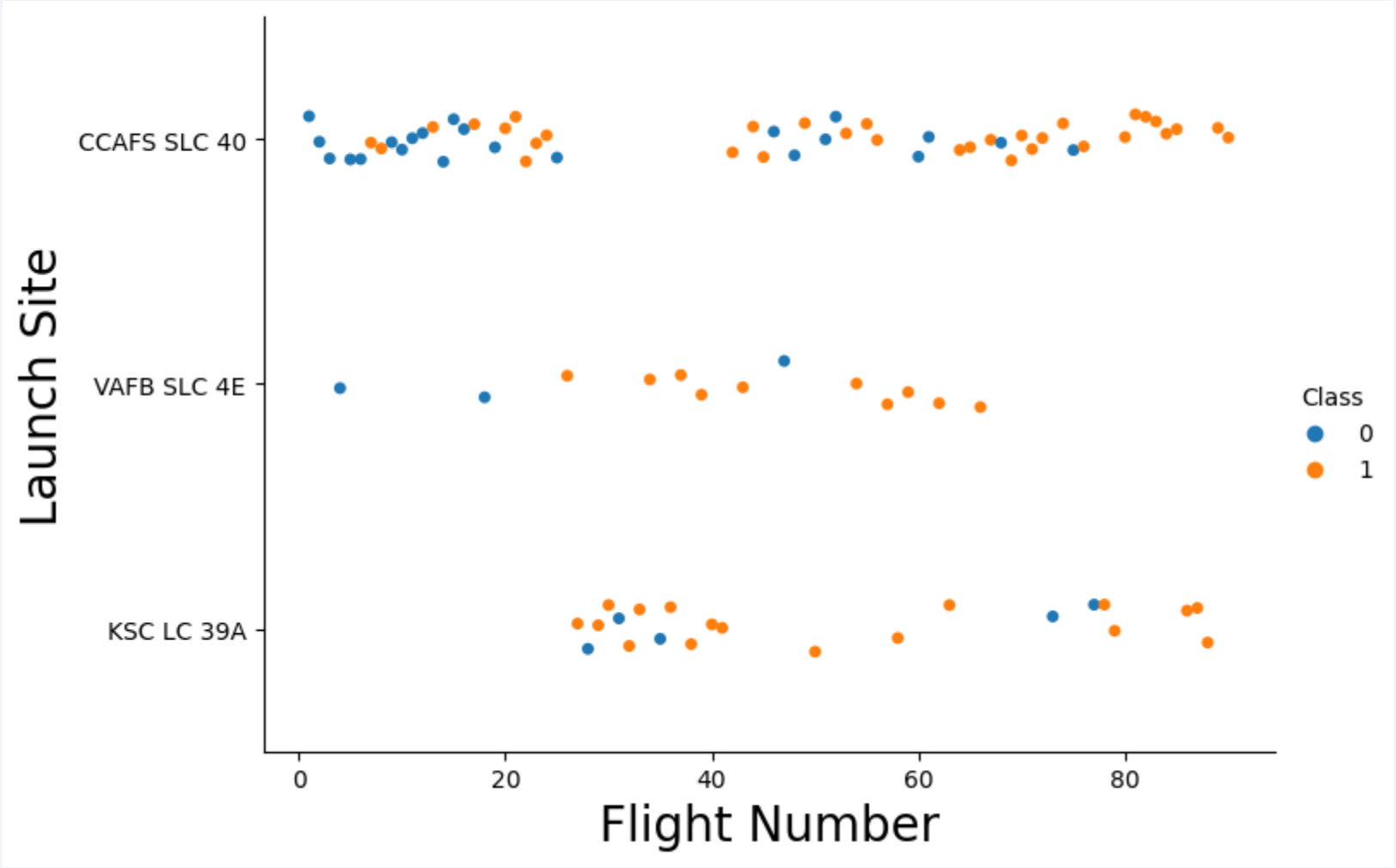
- Here is the GitHub URL for the predictive analysis lab: https://github.com/IchsanHaikal/coursera/blob/main/IBM%20Applied%20Data%20Science%20Capstone/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

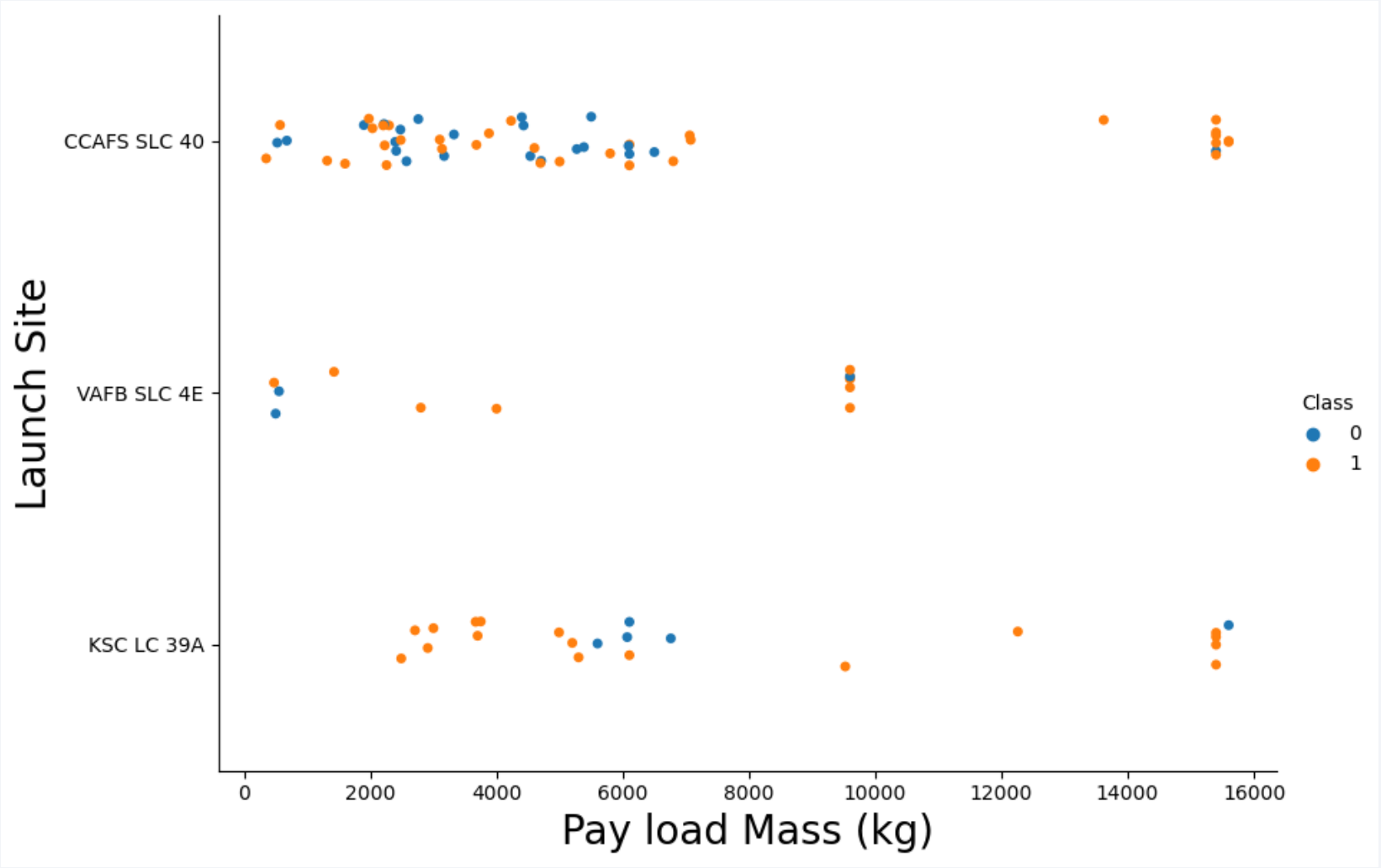
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

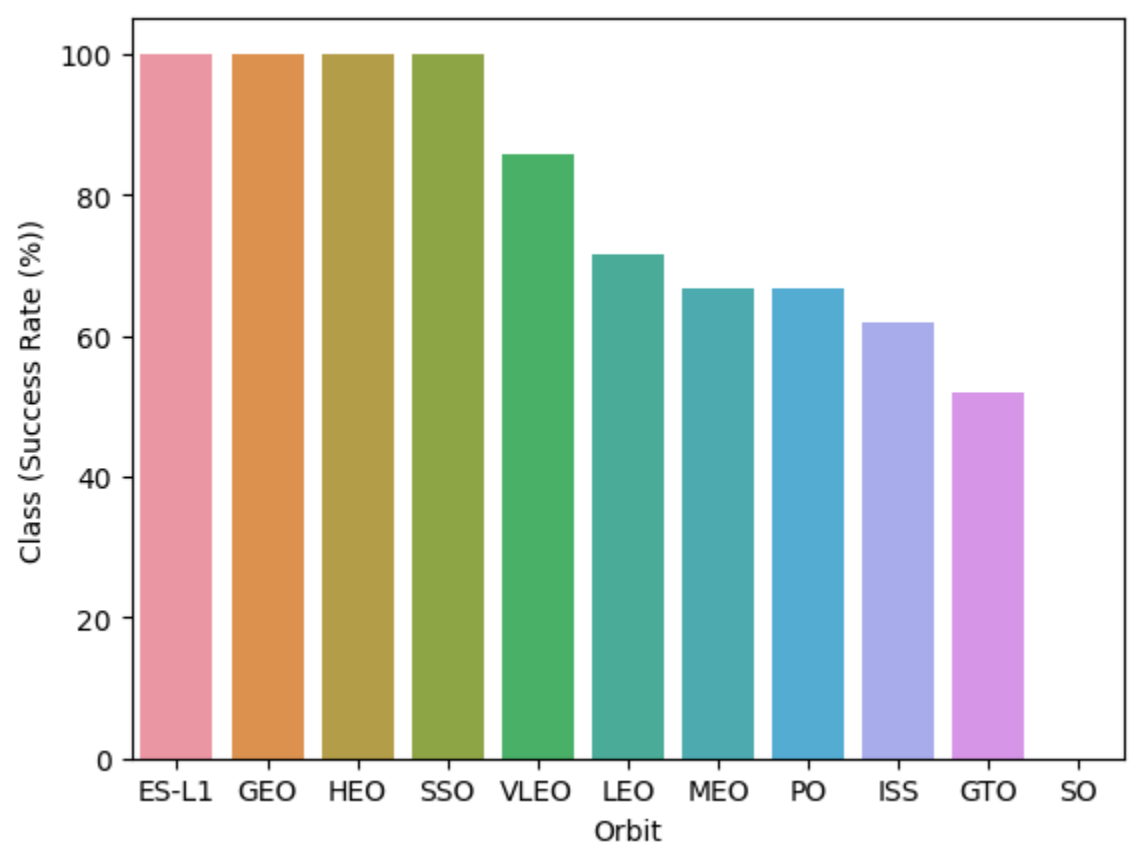
Flight Number vs. Launch Site



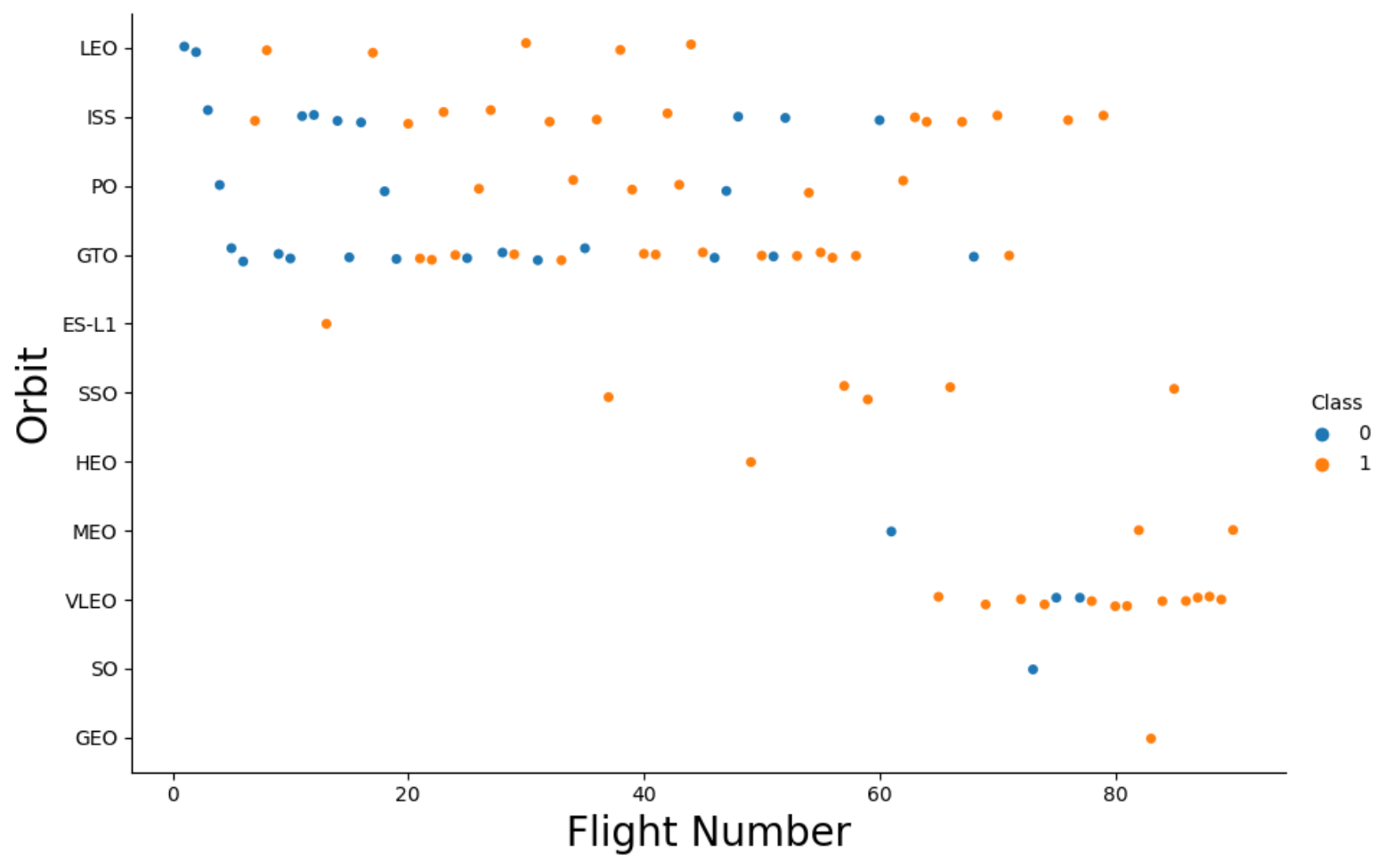
Payload vs. Launch Site



Success Rate vs. Orbit Type

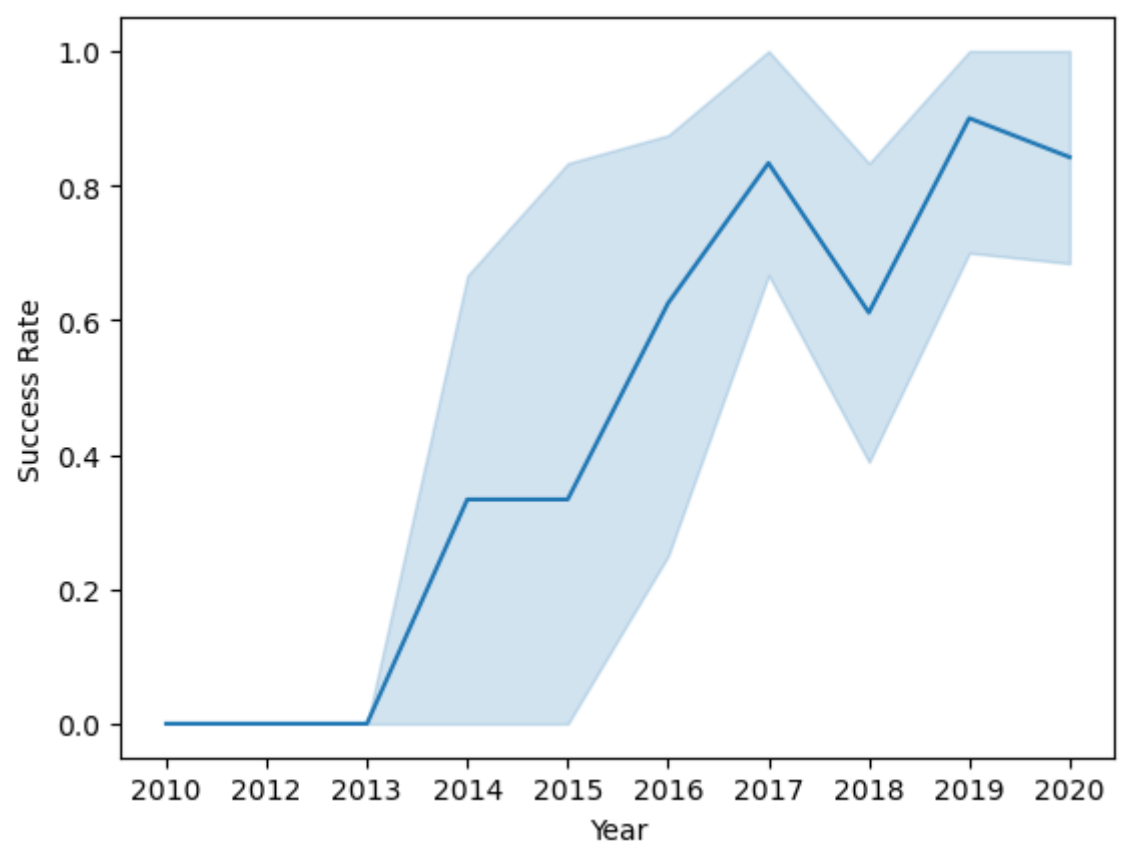


Flight Number vs. Orbit Type



Payload vs. Orbit Type

Launch Success Yearly Trend



All Launch Site Names

Task 1

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

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

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	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Task 3

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

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

Total Payload Mass(Kgs)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

Task 4

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version l
```

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

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

First Successful Ground Landing Date

Task 5

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" = "Success (ground pad)";
```

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

MIN(DATE)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

```
%sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)" AND PAYLOAD_MA
```

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

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

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

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

Boosters Carried Maximum Payload

Task 8

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

```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_M
```

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

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

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, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Ou
```

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

substr(Date,7,4)	substr(Date, 4, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing _Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone ship)

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

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql SELECT * FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORDER
```

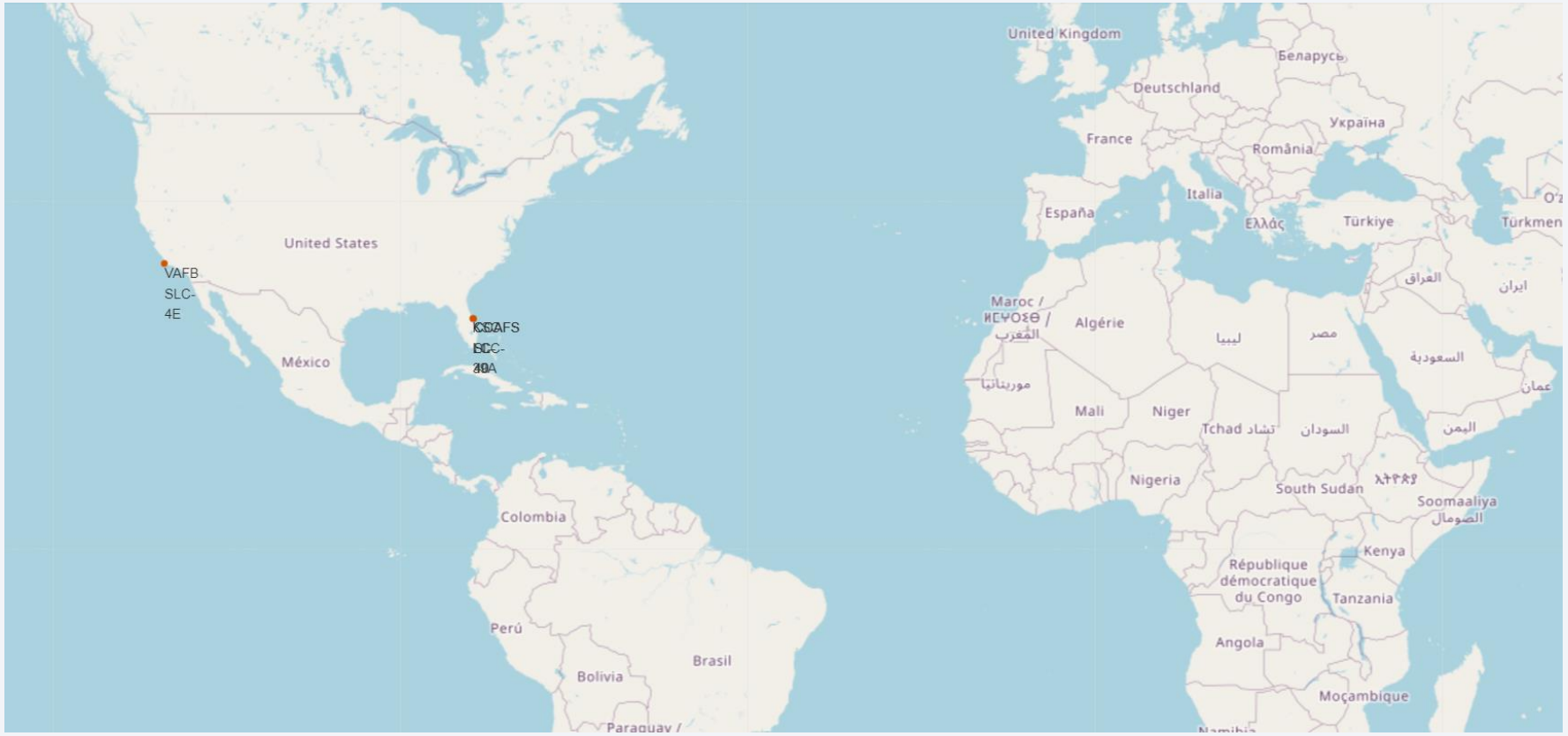
```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-10-2020	12:25:57	F9 B5 B1051.6	KSC LC-39A	Starlink 13 v1.0, Starlink 14 v1.0	15600	LEO	SpaceX	Success	Success
18-08-2020	14:31:00	F9 B5 B1049.6	CCAFS SLC-40	Starlink 10 v1.0, SkySat-19, -20, -21, SAOCOM 1B	15440	LEO	SpaceX, Planet Labs, PlanetIQ	Success	Success
18-07-2016	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-04-2018	22:51:00	F9 B4 B1045.1	CCAFS SLC-40	Transiting Exoplanet Survey Satellite (TESS)	362	HEO	NASA (LSP)	Success	Success (drone ship)

Section 3

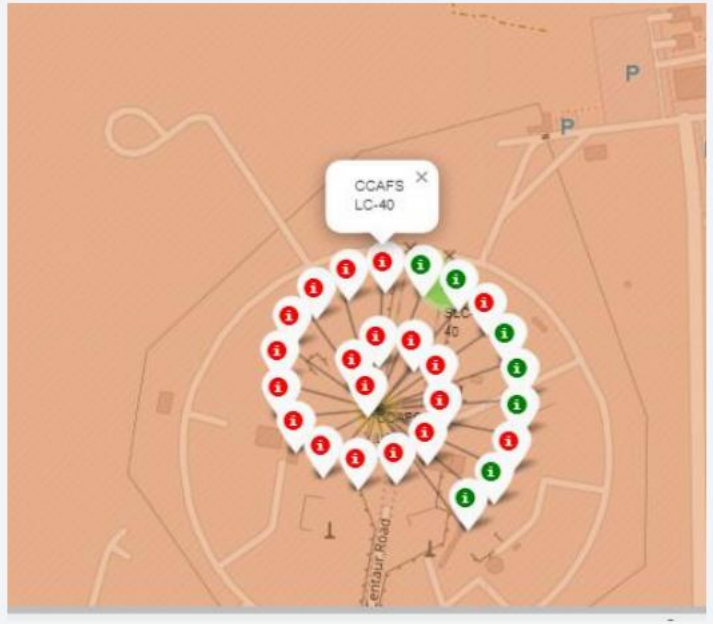
Launch Sites Proximities Analysis

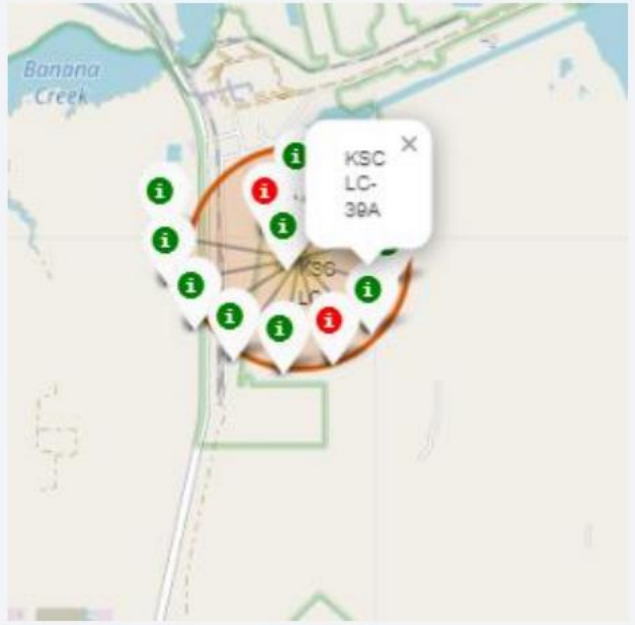
Locations of all launch sites worldwide marked on a global map



All launch facilities are situated near the Equator, positioned to the south on the US map. Additionally, these sites are very near the coastline.

Launch results for each site on the map indicated using colored markers.

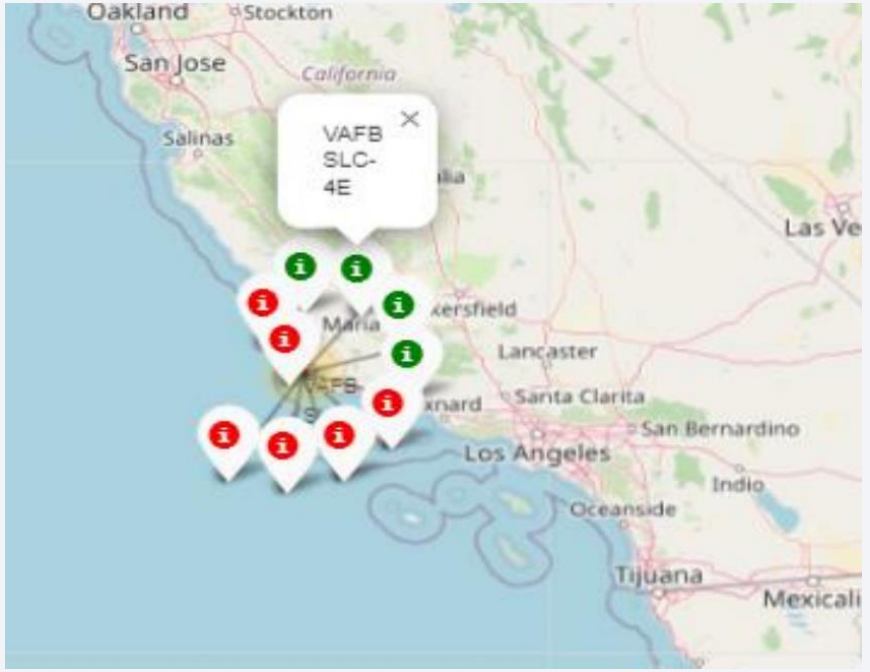




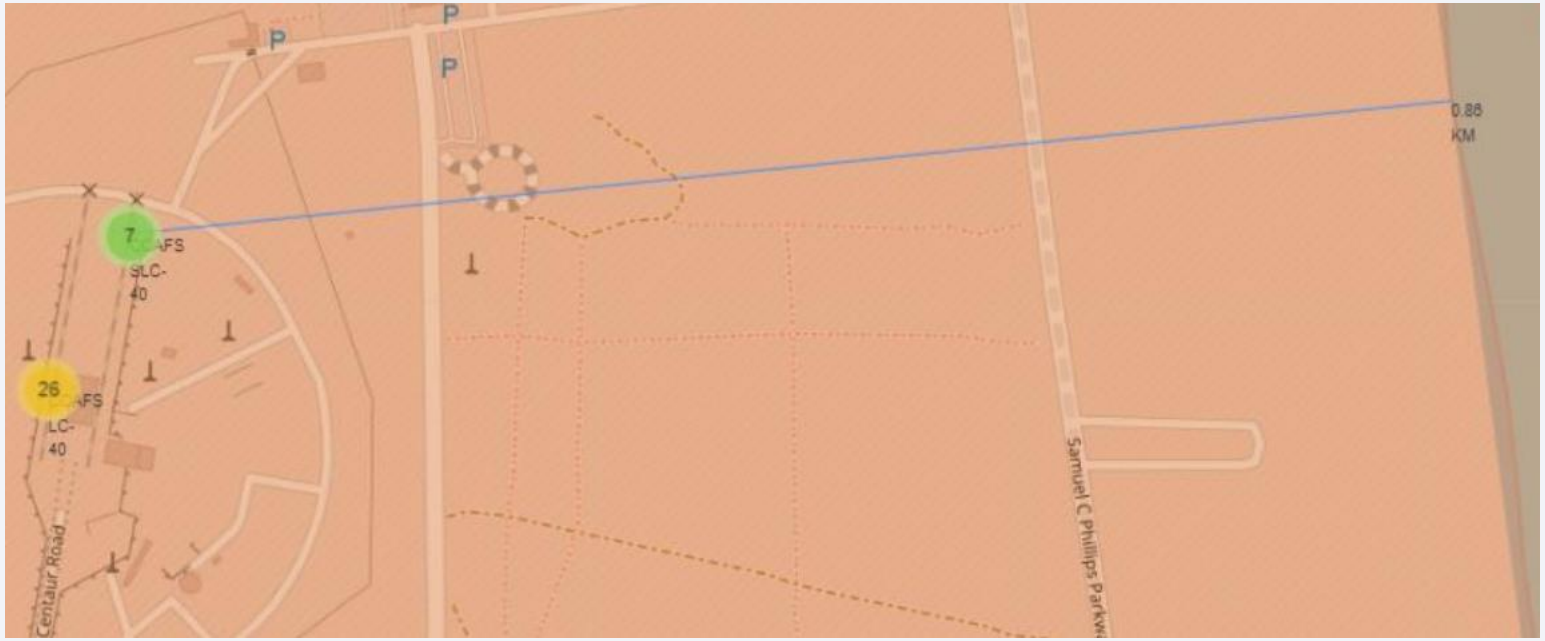
On the eastern coast, specifically in Florida, the launch site KSC LC-39A demonstrates notably higher success rates in contrast to CCAFS SLC-40 and CCAFS LC-40.

Launch results for each site on the map indicated using colored markers.

On the western coast, situated in California, the VAFB SLC-4E launch site exhibits comparatively lower success rates of 4 out of 10 launches when compared to the KSC LC-39A launch site on the eastern coast of Florida.

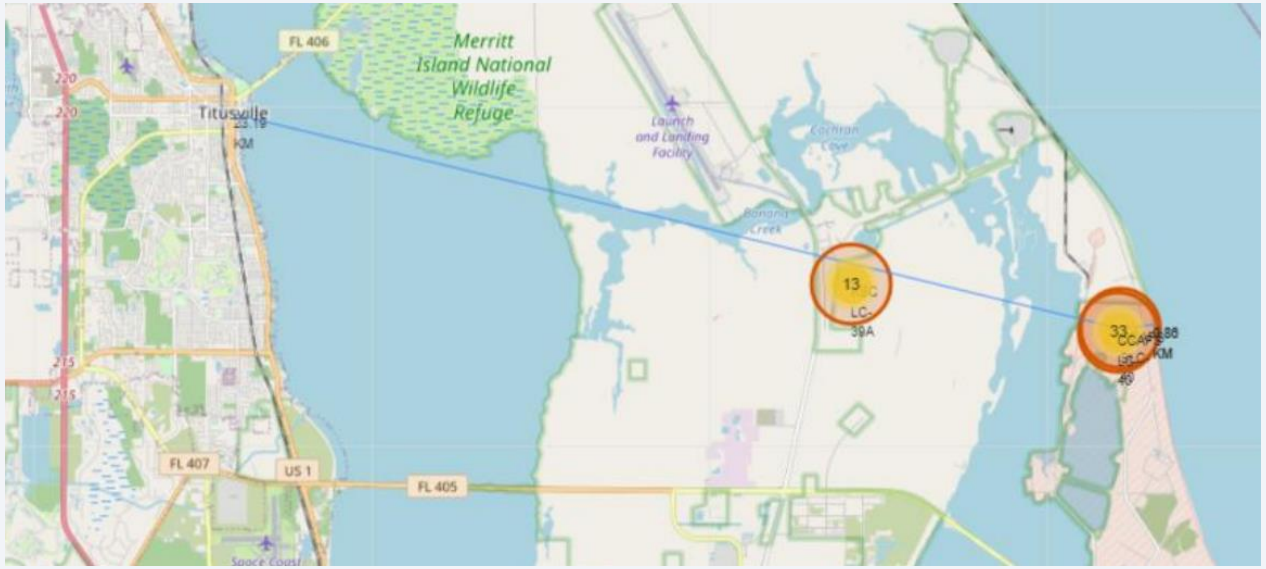


Distances from a launch site to nearby areas indicated.

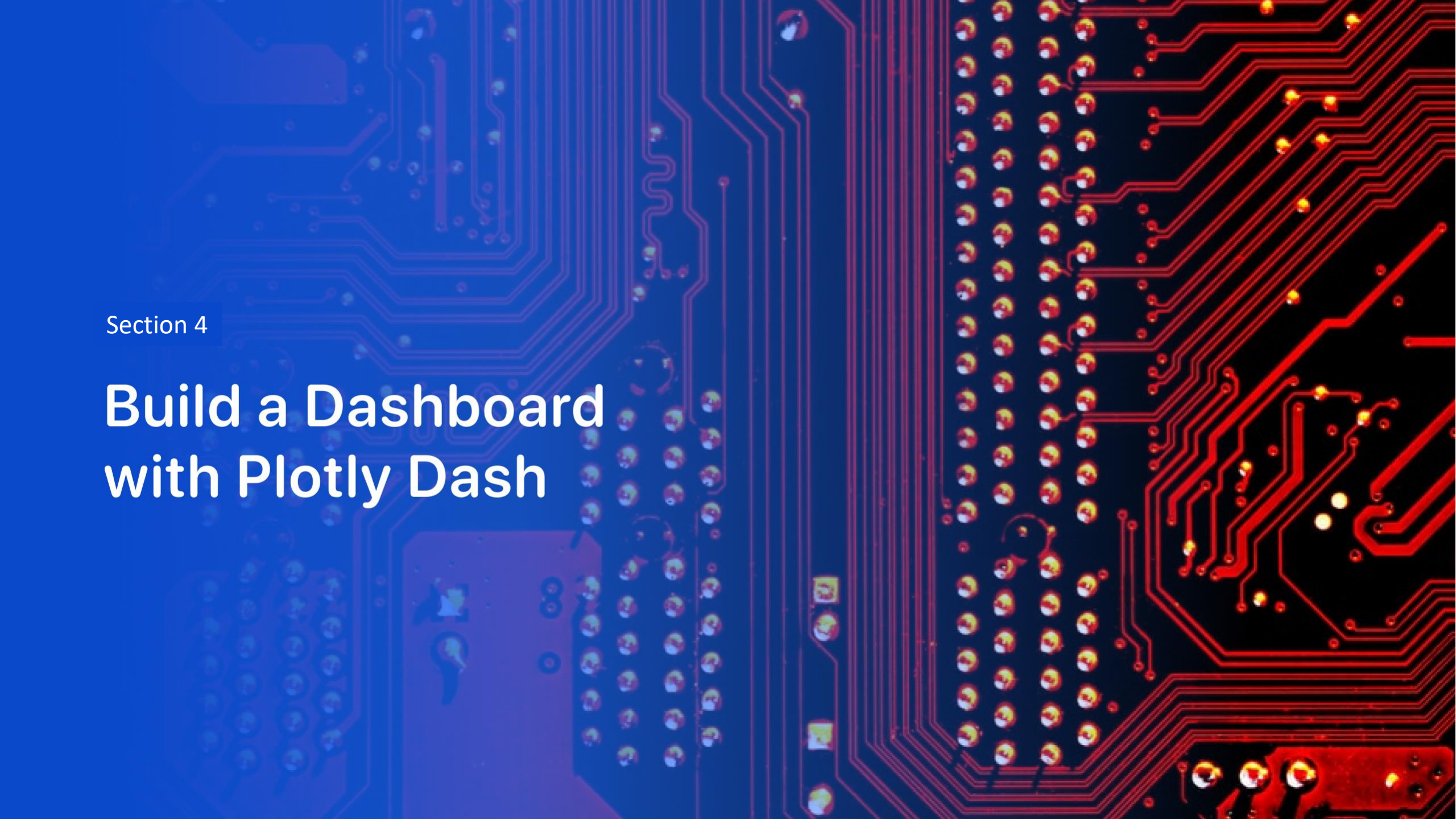


The CCAFS SLC-40 launch site is located at a distance of approximately 0.86 kilometers from the coastline.

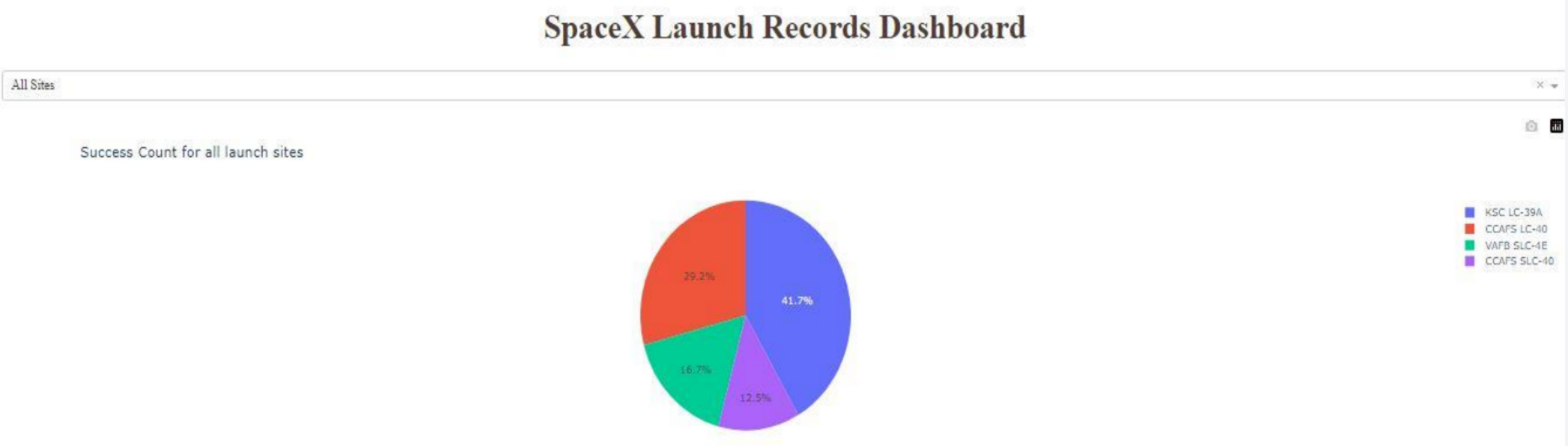
Distances from a launch site to nearby areas indicated.



The launch site CCAFS SLC-40 is nearest to the highway (Washington Avenue) at a distance of approximately 23.19 kilometers.

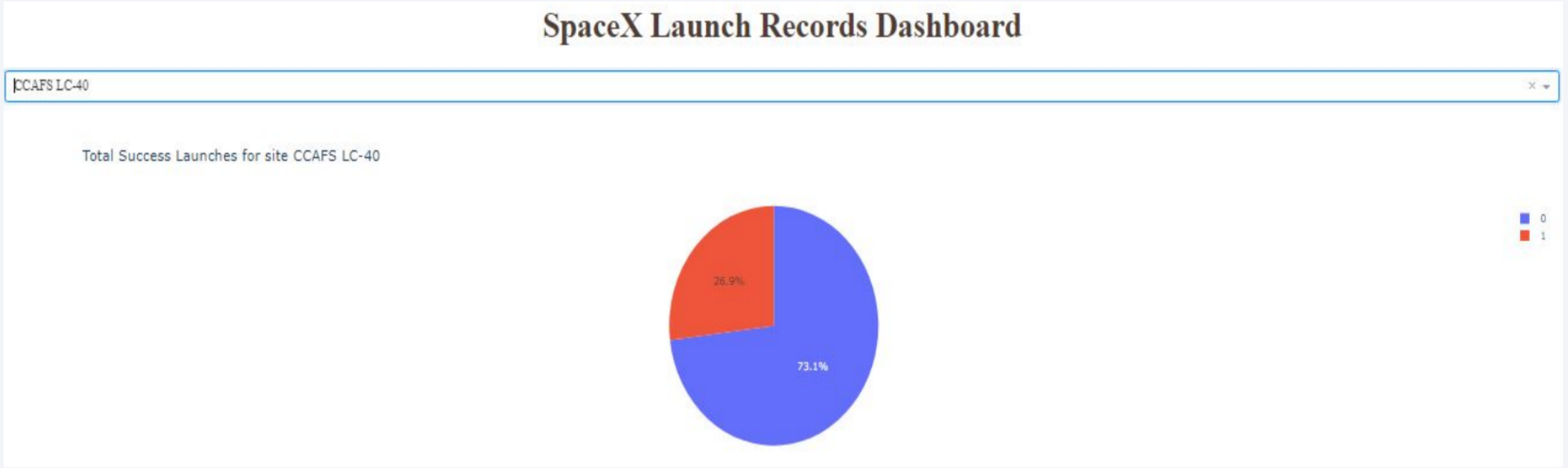

Section 4
Build a Dashboard with Plotly Dash

Pie chart illustrating the count of successful launches for each site.



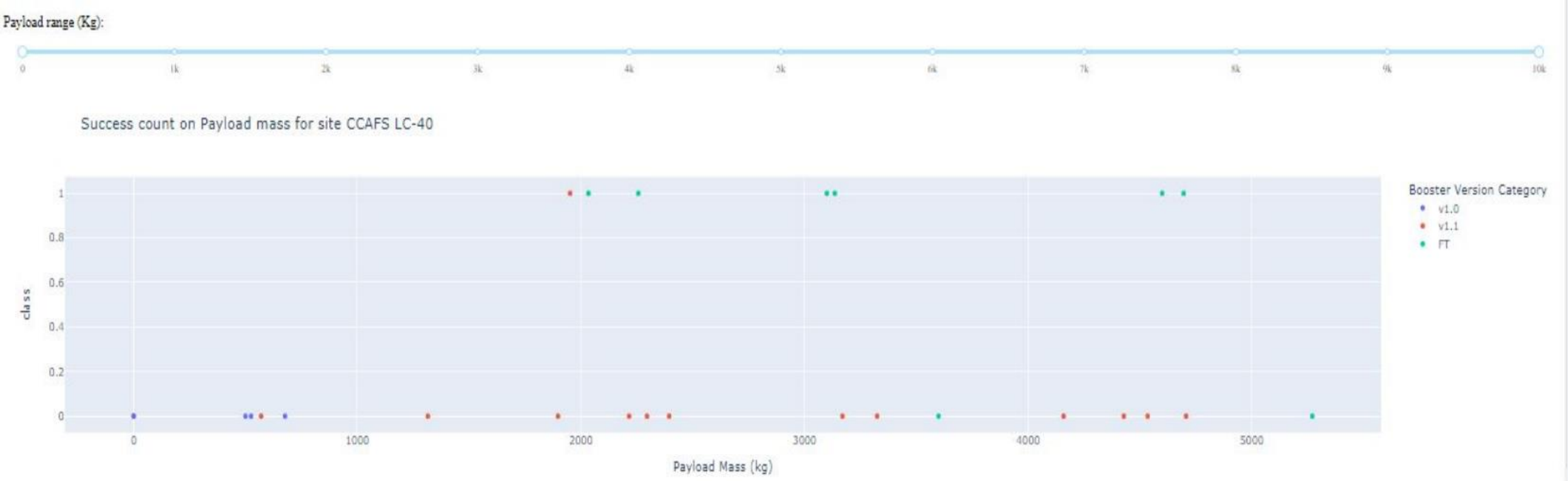
Among the launch sites, KSC LC-39A boasts the highest launch success rate of 42%, succeeded by CCAFS LC-40 at 29%. Following that is VAFB SLC-4E with a rate of 17%, and finally, CCAFS SLC-40 with a success rate of 13%.

Pie chart representing the launch site with the second highest rate of successful launches.



CCAFS LC-40 held the second-highest success ratio with a 73% rate of successful launches, while the remaining 27% ended in failure.

Scatter plot depicting the relationship between Payload and Launch Outcome for all launch sites.



In the case of CCAFS LC-40 launch site, the booster version FT demonstrates the highest success rate when launching payloads weighing over 2000kg.

Section 5

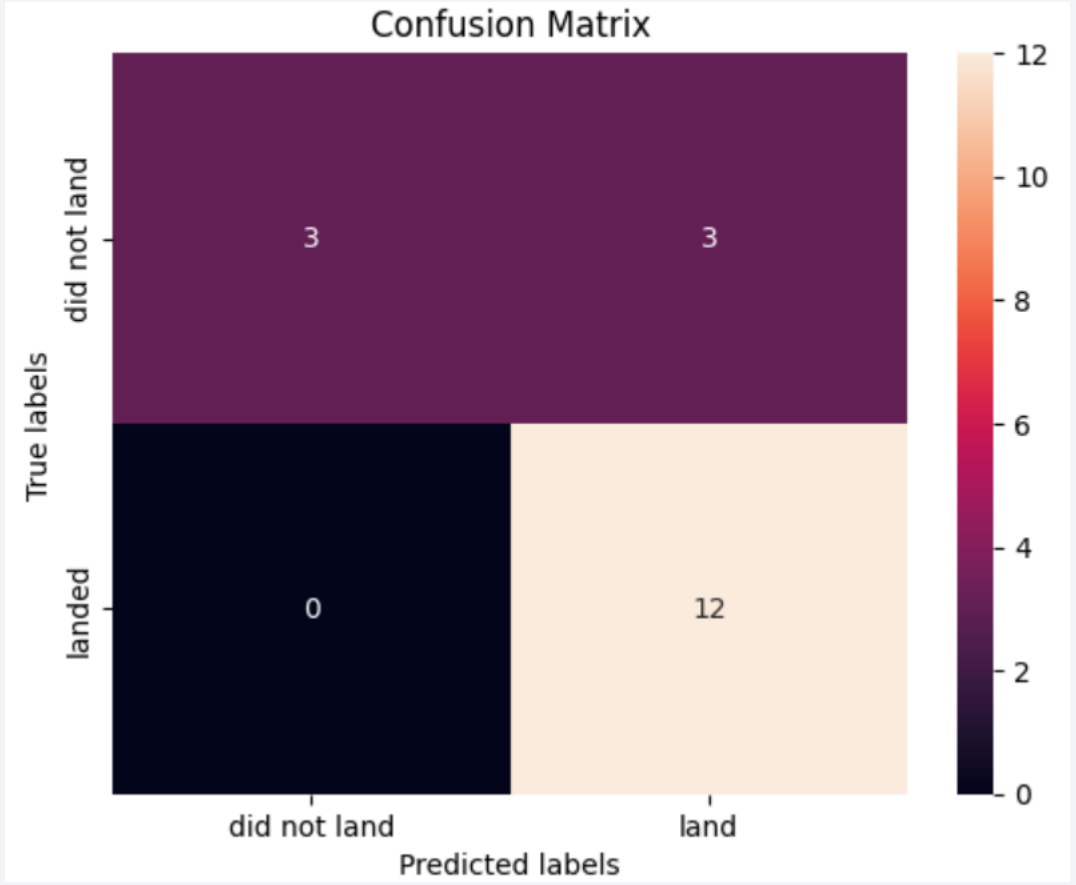
Predictive Analysis (Classification)

Classification Accuracy

Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.722222
KNN	0.833333

Confusion Matrix

The four classification models exhibited identical confusion matrices, demonstrating equal proficiency in distinguishing among various classes. However, a common issue across all models is the occurrence of false positives, which stands out as a significant challenge.



Conclusions

- Distinct launch sites exhibit varying success rates. CCAFS LC-40 maintains a 60% success rate, while both KSC LC-39A and VAFB SLC-4E boast a 77% success rate.
- It can be inferred that with the increase in flight numbers at each of the three launch sites, the success rate also rises. For instance, after the 50th flight, VAFB SLC-4E achieves a 100% success rate. Similarly, both KSC LC-39A and CCAFS SLC-40 attain a 100% success rate after the 80th flight.
- Observing the Payload vs. Launch Site scatter plot, it's evident that the VAFB-SLC launch site lacks rockets launched for heavy payload masses (greater than 10000).
- Among the orbits, ES-L1, GEO, HEO, and SSO register the highest success rates at 100%, whereas SO orbit displays the lowest success rate at approximately 50%. Notably, the SO orbit has a 0% success rate.
- In the LEO orbit, success appears to be correlated with the number of flights. Conversely, in the GTO orbit, there seems to be no discernible relationship between flight number and success rate.

Conclusions

- For heavy payloads, the positive landing rate is notably higher for Polar, LEO, and ISS orbits. However, in the case of GTO, the distinction is less clear since both positive landing rates and negative landings (unsuccessful missions) coexist.
- Lastly, the success rate has shown a consistent increase from 2013 to 2020.

Thank you!