IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Preet Kothari
20th September 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
    - Through API
    - With Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis
    - SQL
    - Data Visualization
  - Interactive Visual Analytics with Folium and Plotly Dash
  - Predictions using different Machine Learning algorithms
- Summary of all results
  - Exploratory Data Analysis
  - Interactive Analytics
  - Predictive Analytics

Introduction

Project Background

Space X advertises Falcon 9 rocket launches on their website with a cost of 62 million dollars; whereas 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. Using this information we as a competitor can 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.

Major Objectives

- 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 program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology
  - Data was collected using the SpaceX API and some important tables were web scraped from Wikipedia.
- Perform data wrangling
  - Applied one-hot encoding on the categorical features collected within the data.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Standardize data, train different classification models and fine tune each model using GridSearchCV and finding the best performing model based on model accuracy.

Data Collection

- The data collection was done using get request to the SpaceX API.
- Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
- Then data was cleaned, checked for missing values and fill in missing values where necessary.
- In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
- The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- Click here for the Spacex-data-collection-api notebook.

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

In [12]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

In [13]:
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

In [30]:
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup.
- We parsed the table and converted it into a pandas dataframe.
- Click here for the Spacex-webscraping-wikipedia notebook.

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

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

In [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
```

Out[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

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

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

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

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

3. Extract all column names from the HTML table header

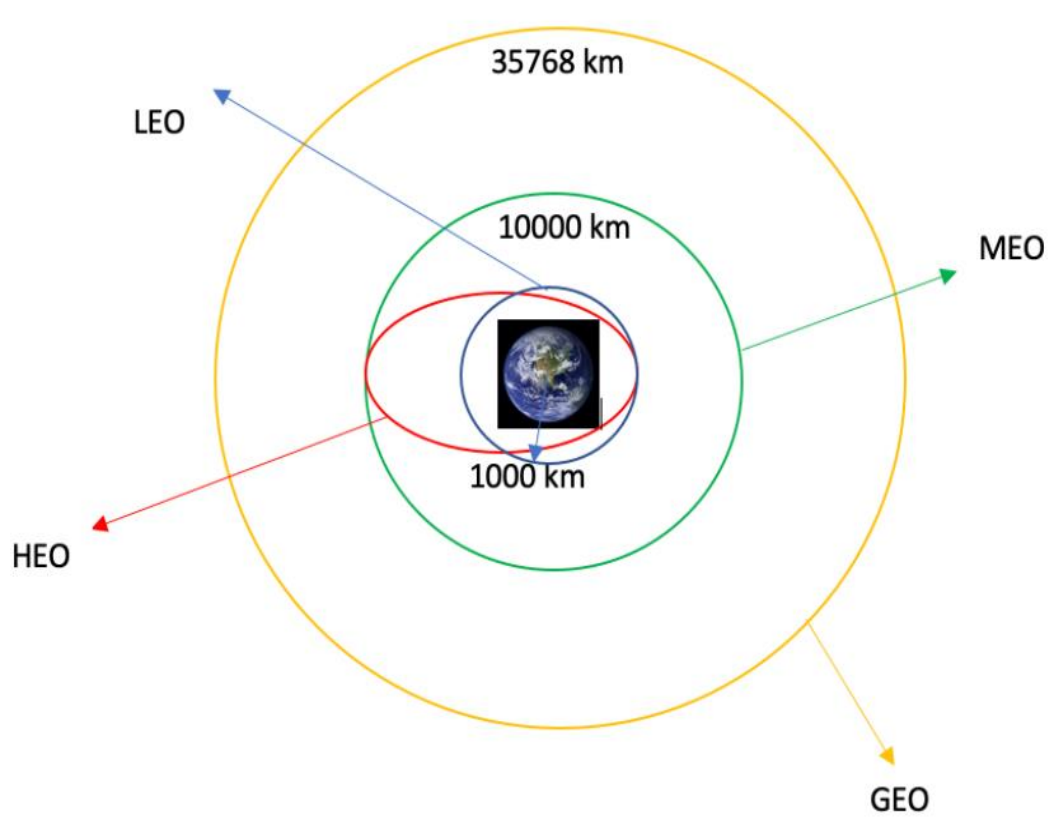
In [10]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

Data Wrangling



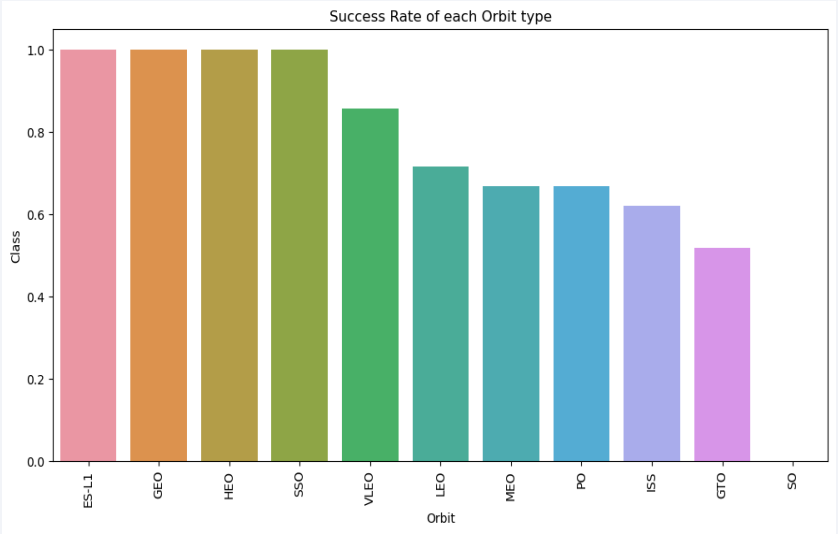
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- Click here for the Spacex-data-wrangling notebook.

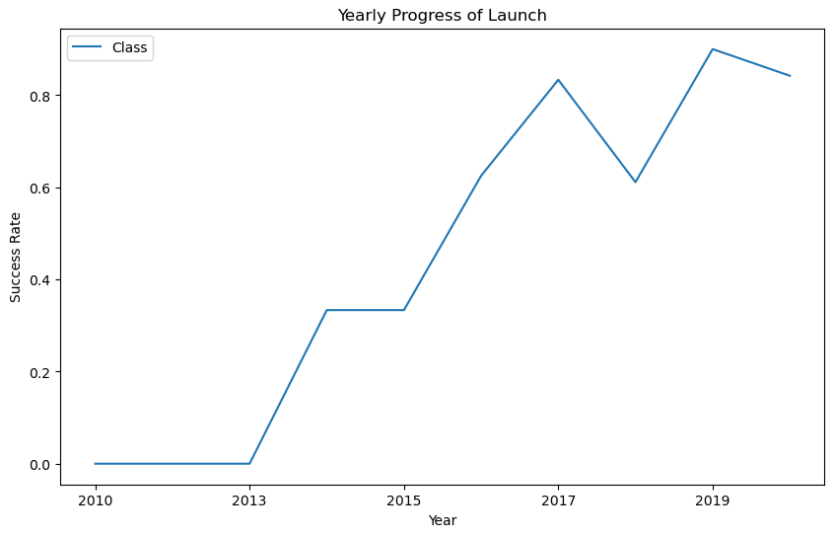
EDA with SQL

- We loaded the SpaceX dataset using SQLAlchemy to access and manage SQL database without leaving the jupyter notebook. Further wrote queries using Sqlite3 library in the notebook.
- We performed EDA with SQL to get various insights from the data. We wrote queries to find:
  - Unique launch sites utilized in the space mission
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful or failure mission outcomes
  - Fetched the failed landing outcomes which occurred in drone ships, with their booster version and launch site names.
- Click here for the EDA-with-SQL notebook.

EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.





- Click here for the EDA-with-Data-Visualization notebook.

Interactive Map with Folium

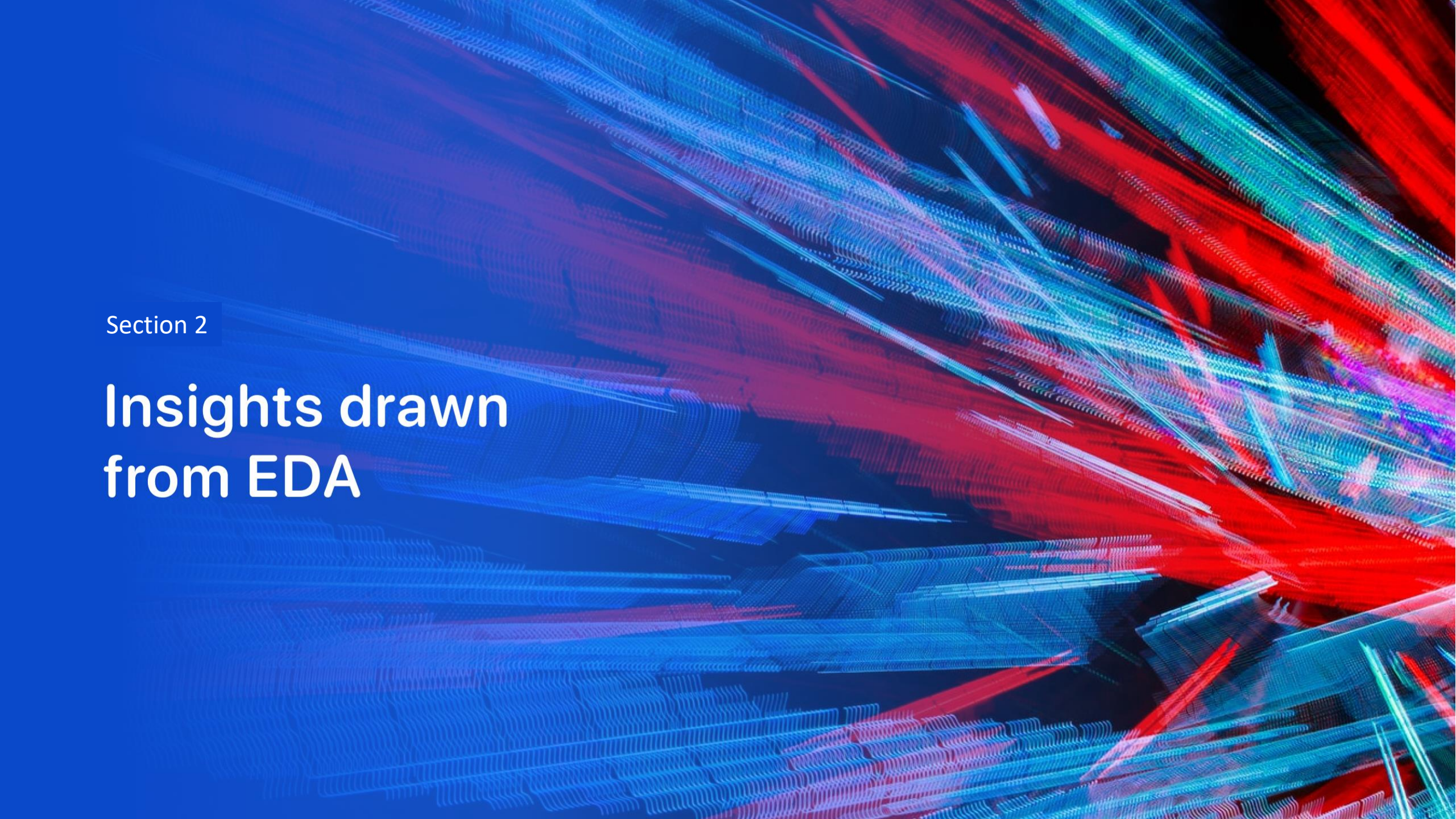
- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines?
  - Do launch sites keep certain distance away from cities?
- Click here for the Interactive_Visual_Launch_Site_Location_Analysis_with_Folium notebook.

Interactive Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash.
- We plotted pie charts showing the percentage of successful launches for different sites along with success to failure percentages for particular launch sites.
- We plotted scatter graph showing the relationship between the Mission Outcome and Payload Mass (Kg) for the different booster version utilized in the launch.
- Click here for the Spacex_dash_app python code.

Predictive Analysis (Classification)

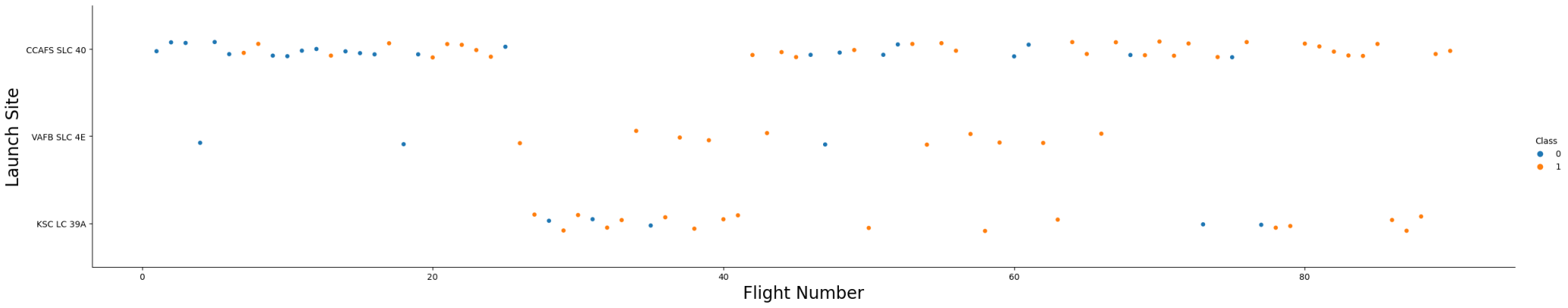
- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and fine-tuned different hyperparameters using GridSearchCV for each model.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- Click here for SpaceX_Machine_Learning_Prediction notebook.


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

Insights:

- As the flight number increases, the likelihood of a successful landing appears to increase. This could be attributed to the improvements SpaceX has made over time, learning from earlier launches.
- The Payload Mass does not seem to have a strong negative impact on the landing success rate. Even with larger payloads, many landings are still successful. This suggests that SpaceX's landing technology is robust, even when dealing with heavier rockets.

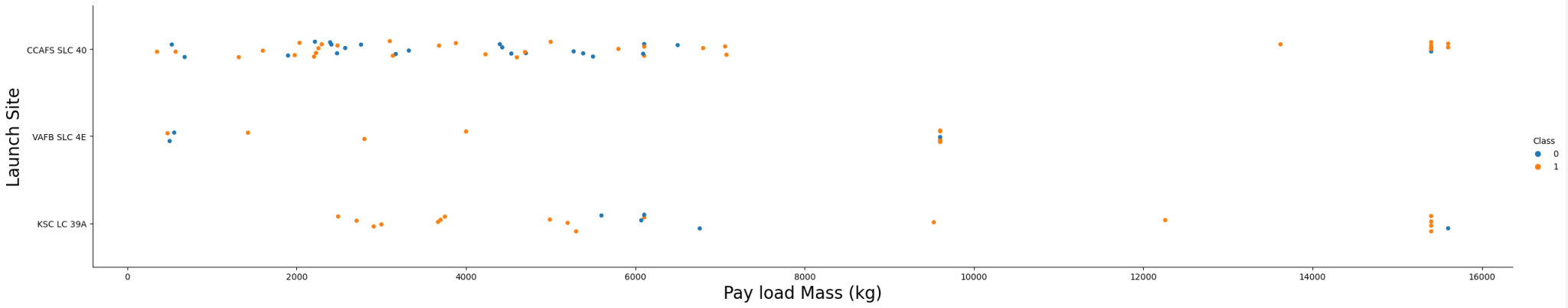


Conclusion: SpaceX has shown increasing landing success over time, and the payload mass does not seem to hinder this progress.

Payload Mass vs. Launch Site

Insights:

- The VAFB SLC 4E launch site does not seem to handle heavy payloads, as there are no launches with payloads over 10,000 kg.
- CCAFS SLC 40 and KSC LC 39A both handle heavier payloads and tend to have a high success rate regardless of the mass.

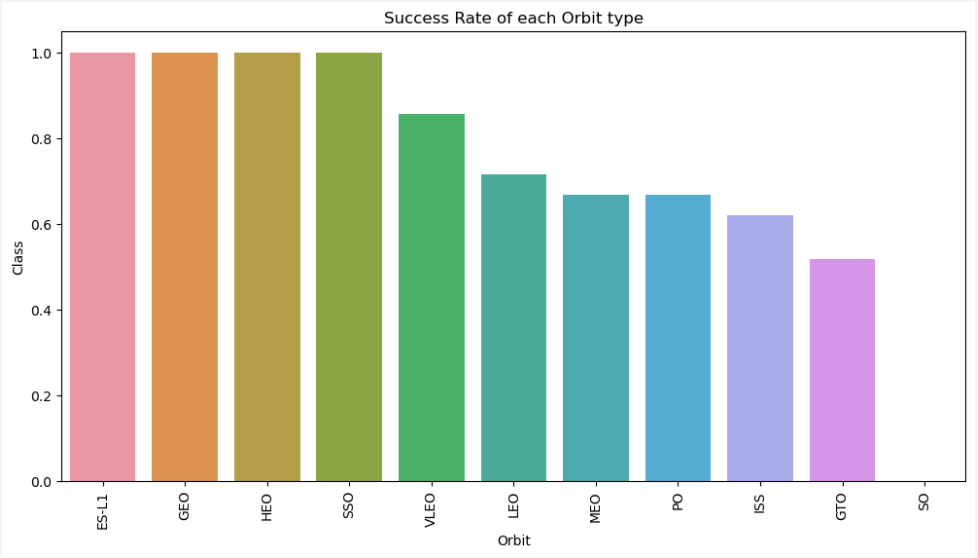


Conclusion: Not all launch sites are suitable for heavy payloads, with VAFB SLC 4E handling lighter loads. CCAFS SLC 40 and KSC LC 39A are more versatile, able to handle a wider range of payloads while maintaining success.

Success Rate vs. Orbit Type

Insights:

- While ES-L1, GEO, HEO & SSO have 100% success rate, orbits like the LEO (Low Earth Orbit), ISS, and Polar Orbit (PO) have higher success rates than other orbits like GTO (Geostationary Transfer Orbit), where success rates are lower.
- This suggests that certain orbit types might be more challenging for successful landings, particularly GTO.

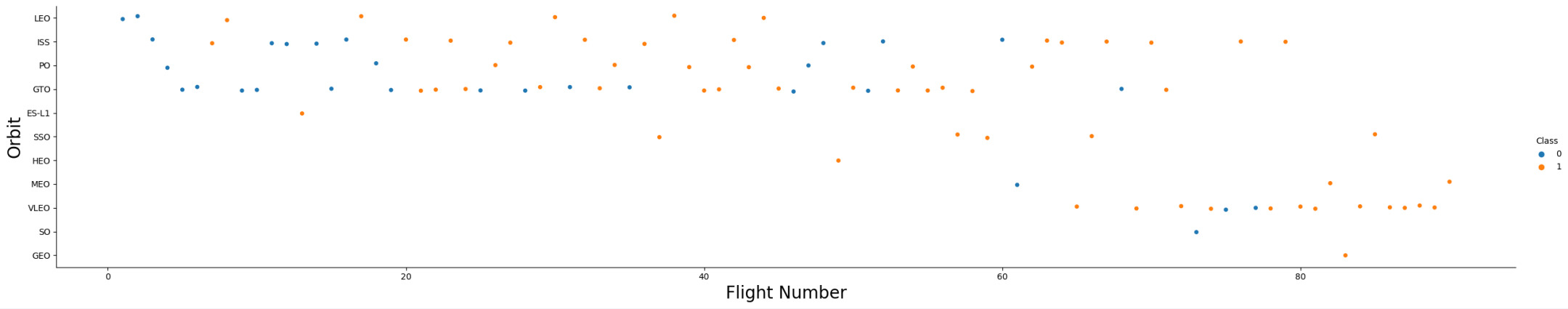


Conclusion: Launches into LEO and ISS orbits are generally more successful, while GTO launches are more challenging. If you're a competitor, focusing on orbits with higher success rates might help you offer more reliable services.

Flight Number vs. Orbit Type

Insights:

- LEO orbit shows a clear trend where the number of flights correlates with a higher success rate over time.
- GTO does not exhibit this trend, indicating that increased launch experience does not necessarily translate to higher success in GTO orbit.

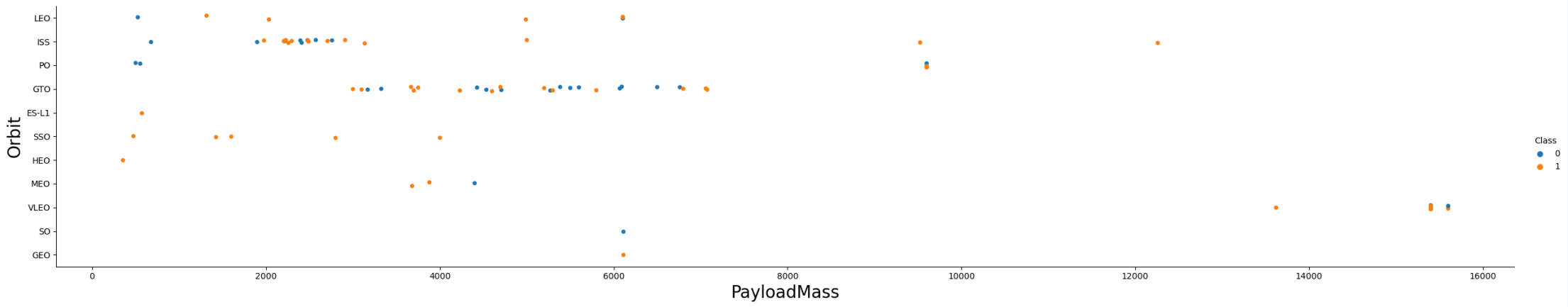


Conclusion: Experience improves success rates for some orbits (e.g., LEO), but for challenging orbits like GTO, other factors beyond experience may be more important.

Payload vs. Orbit Type

Insights:

- Heavier payloads (> 10,000 kg) tend to be launched into Polar, LEO, and ISS orbits, and these orbits generally show high success rates.
- For GTO launches, the payload mass doesn't seem to correlate well with success; both successful and unsuccessful landings occur with similar payloads.

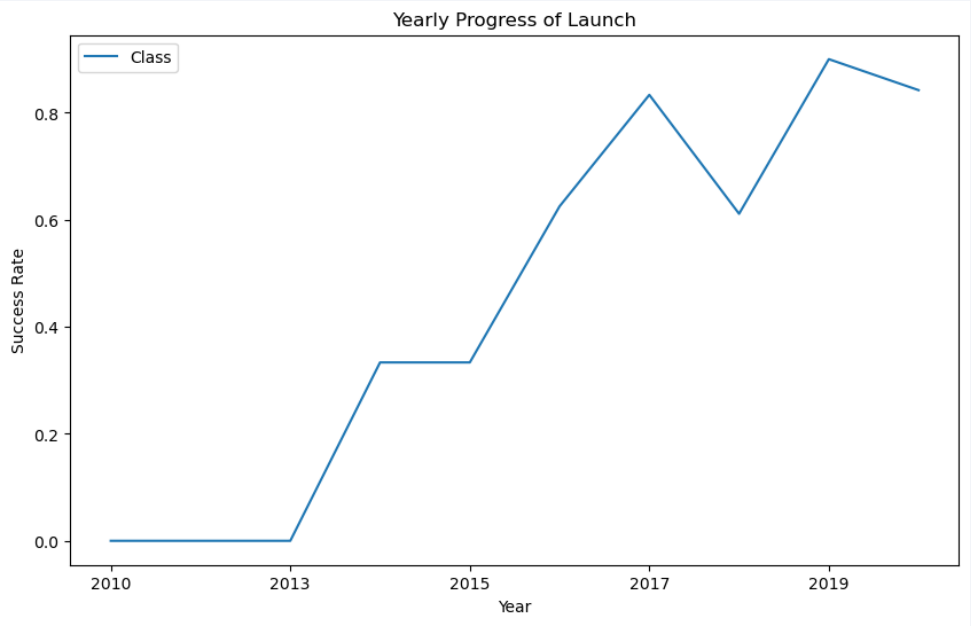


Conclusion: Heavy payloads are more reliably landed in LEO and ISS orbits, whereas GTO remains unpredictable regardless of the payload mass.

Launch Success Yearly Trend

Insights:

- The success rate of Falcon 9 launches has consistently improved since 2013, reaching close to 100% in recent years.
- This trend reflects SpaceX's learning curve and the improvements made to their technology over time.



Conclusion: SpaceX has significantly improved its technology, with the success rate increasing steadily. This makes recent launches far more predictable in terms of success.

All Launch Site Names

- We used the DISTINCT keyword in the SELECT statement to show unique launch sites from the SpaceX data.

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

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
```

```
 * 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'

- We used the LIMIT clause with the WHERE clause specifying the condition that Launch site should begin with ‘CCA’ to display 5 records.

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
2010-06-04	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 for NASA (CRS)

- The total payload mass in kilograms carried by boosters from NASA was calculated as 45,596 Kgs using the SUM function in the SELECT statement.

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

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

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

Total_Payload_Mass_kg
45596

Average Payload Mass by F9 v1.1

- The average payload mass in kilograms carried by F9 v1.1 booster version was calculated as 2,928.4 Kgs using the AVG operation in the SELECT statement.

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_Payload_Mass_kg FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1';
```

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

Avg_Payload_Mass_kg
2928.4

First Successful Ground Landing Date

- We found the first successful landing took place on 22nd December 2015, using the MIN function with WHERE clause specifying that the landing outcome was successful.

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

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

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

First_successful_landing_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the BETWEEN … AND condition to determine successful landing with payload mass greater than 4000 kg but less than 6000 kg.

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 Booster_Version FROM SPACEXTBL WHERE (Landing_Outcome = '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

- We used GROUPBY clause on Mission Outcome column to group by the entries by mission outcomes and then used the COUNT function to count the number of successful and failure outcomes.

List the total number of successful and failure mission outcomes

```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) AS Outcome_Count FROM SPACEXTBL GROUP BY Mission_Outcome;
```

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

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

Boosters Carried Maximum Payload

- We determined the booster version that have carried the maximum payload using a subquery in the WHERE clause and the MAX function.

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

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

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

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

2015 Launch Records

- We used a combinations of the WHERE clause, SUBSTR function, and AND conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015.

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.

```
%sql SELECT SUBSTR(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTBL WHERE (SUBSTR(DATE, 1, 4) = '2015') AND (Landing_Outcome = 'Failure (drone ship)') ORDER BY Month;
```

```
 * 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

- We selected different landing outcomes, their counts using the COUNT function and DENSE RANK function on their counts to rank the outcomes with the WHERE clause and BETWEEN … AND condition to filter for landing outcomes between 2010-06-04 to 2010-03-20. We applied the GROUP BY clause to group the landing outcomes.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) AS Outcome_Count, DENSE_RANK() OVER(ORDER BY COUNT(Landing_Outcome) DESC) AS Ranking
FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome;
```

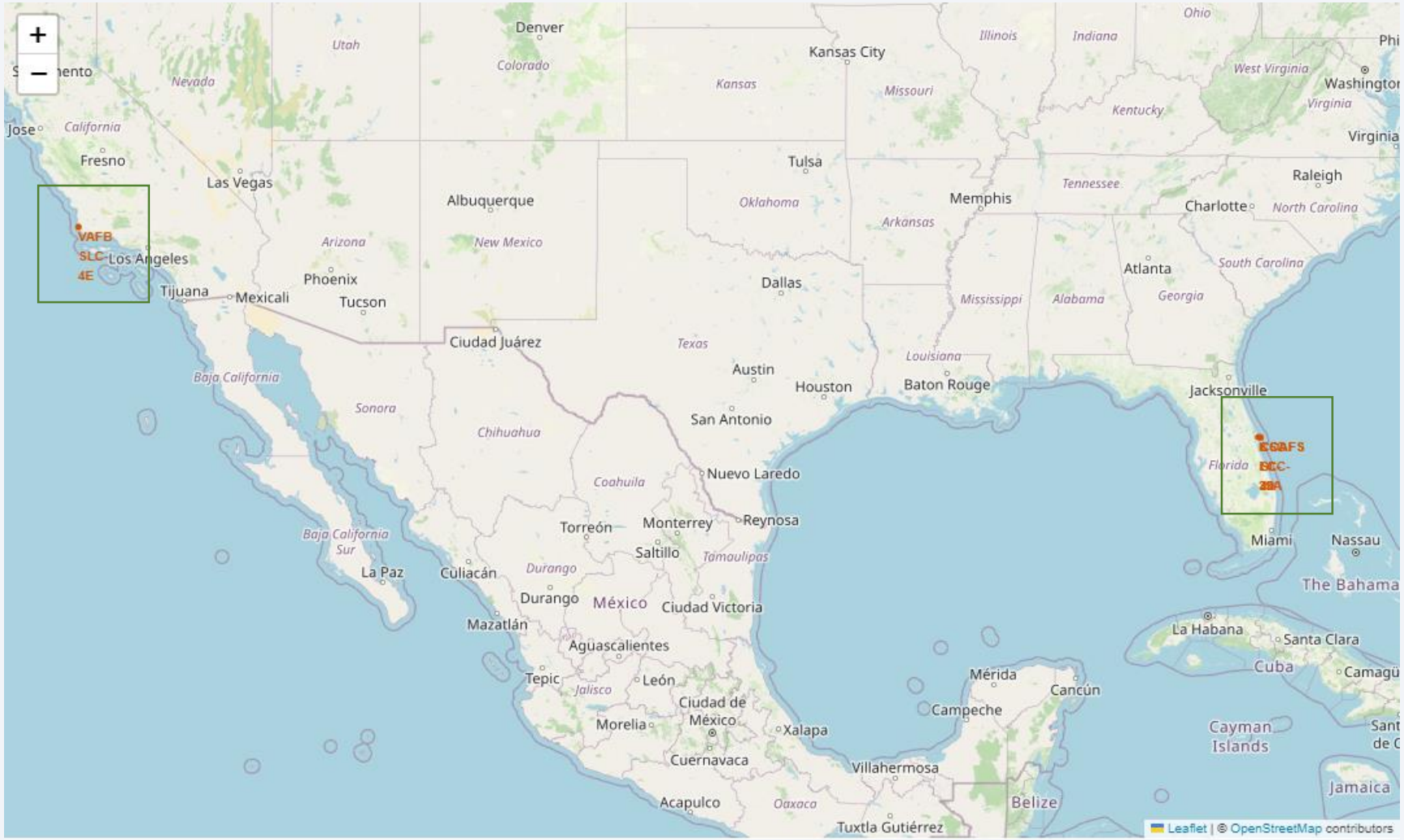
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	Outcome_Count	Ranking
No attempt	10	1
Success (drone ship)	5	2
Failure (drone ship)	5	2
Success (ground pad)	3	3
Controlled (ocean)	3	3
Uncontrolled (ocean)	2	4
Failure (parachute)	2	4
Precluded (drone ship)	1	5

Section 3

Launch Sites Proximities Analysis

Global Map with Launch Sites Markers

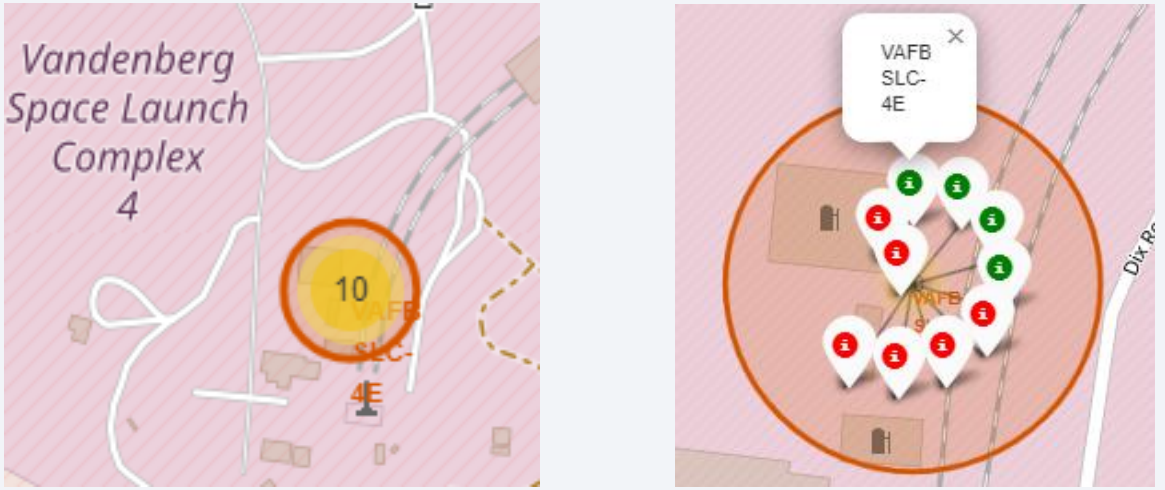


The SpaceX launch sites are located at coasts of the United States of America:

- Florida
- California

Launch Site Markers with Mission Outcome hue labels

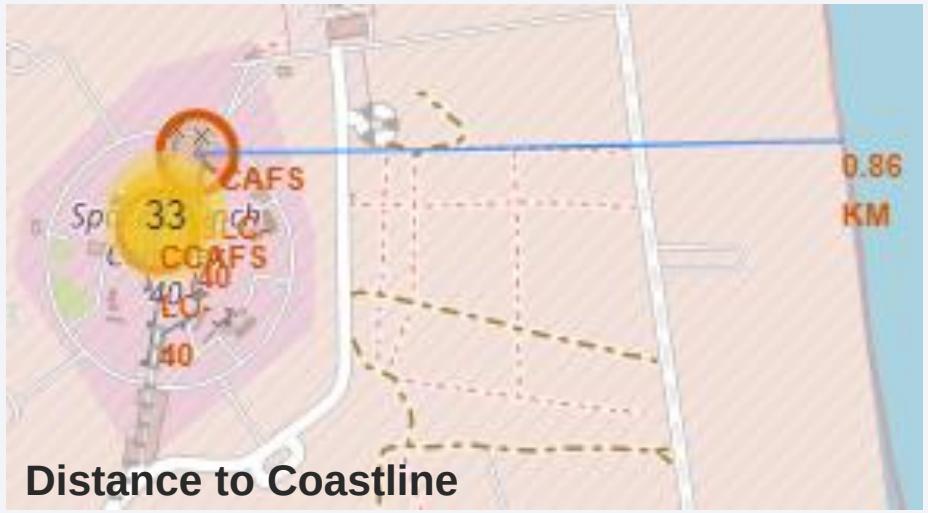
Florida Launch Sites



California Launch Sites

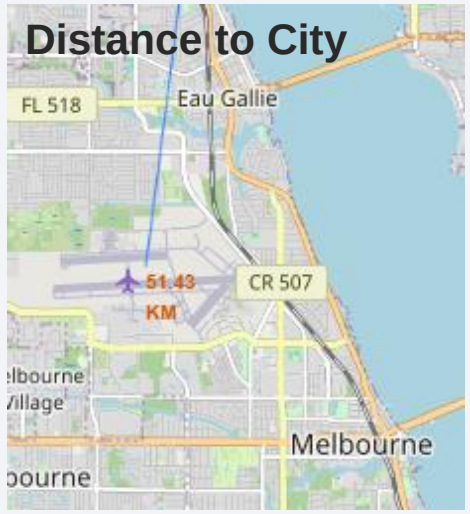
The different markers at the launch location signify the mission outcomes. The *Green Marker* shows that the mission was successful and *Red Marker* shows failure to land.

Launch Site distance to Landmarks



Distance to Coastline

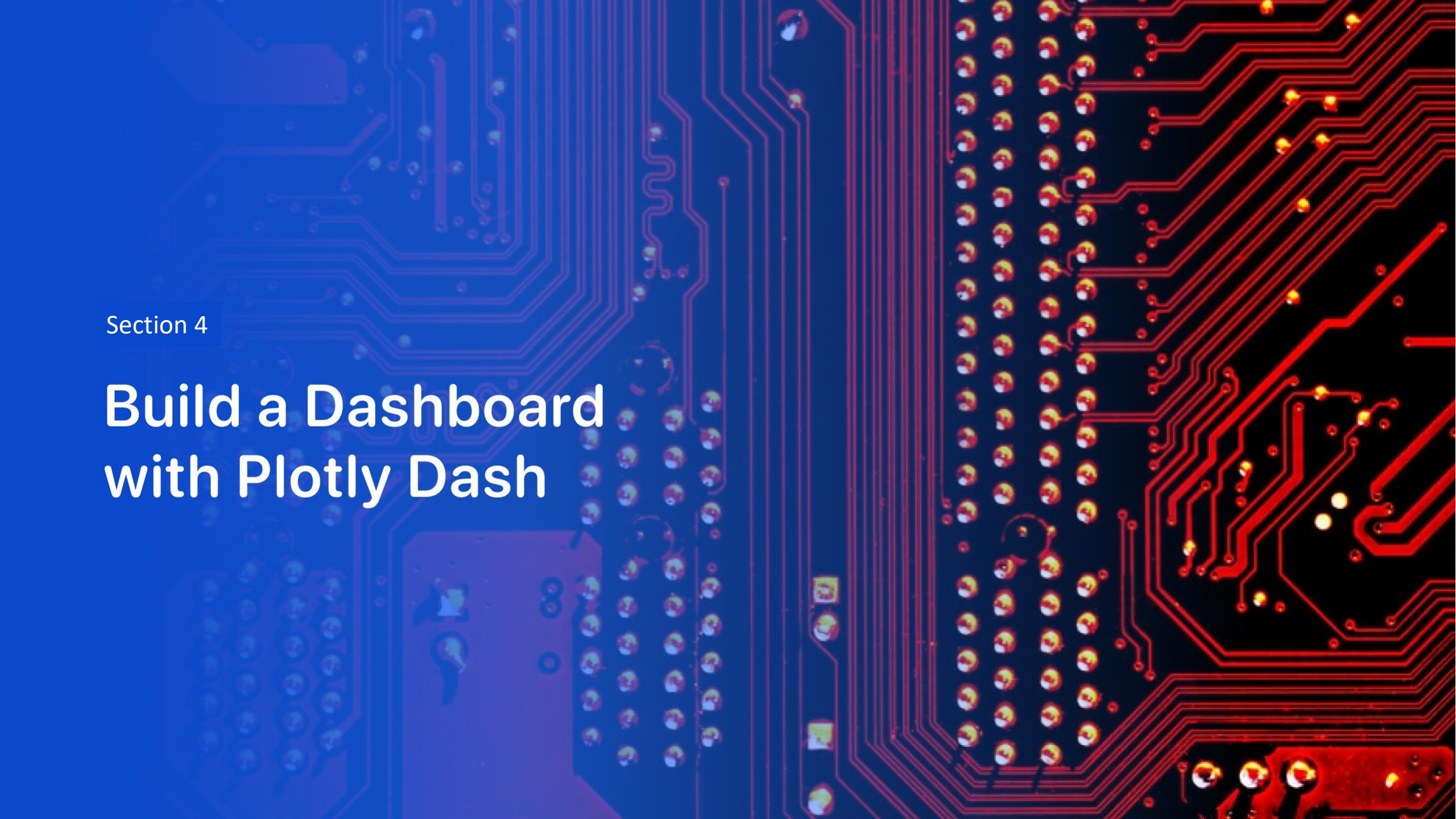
Distance to Highway



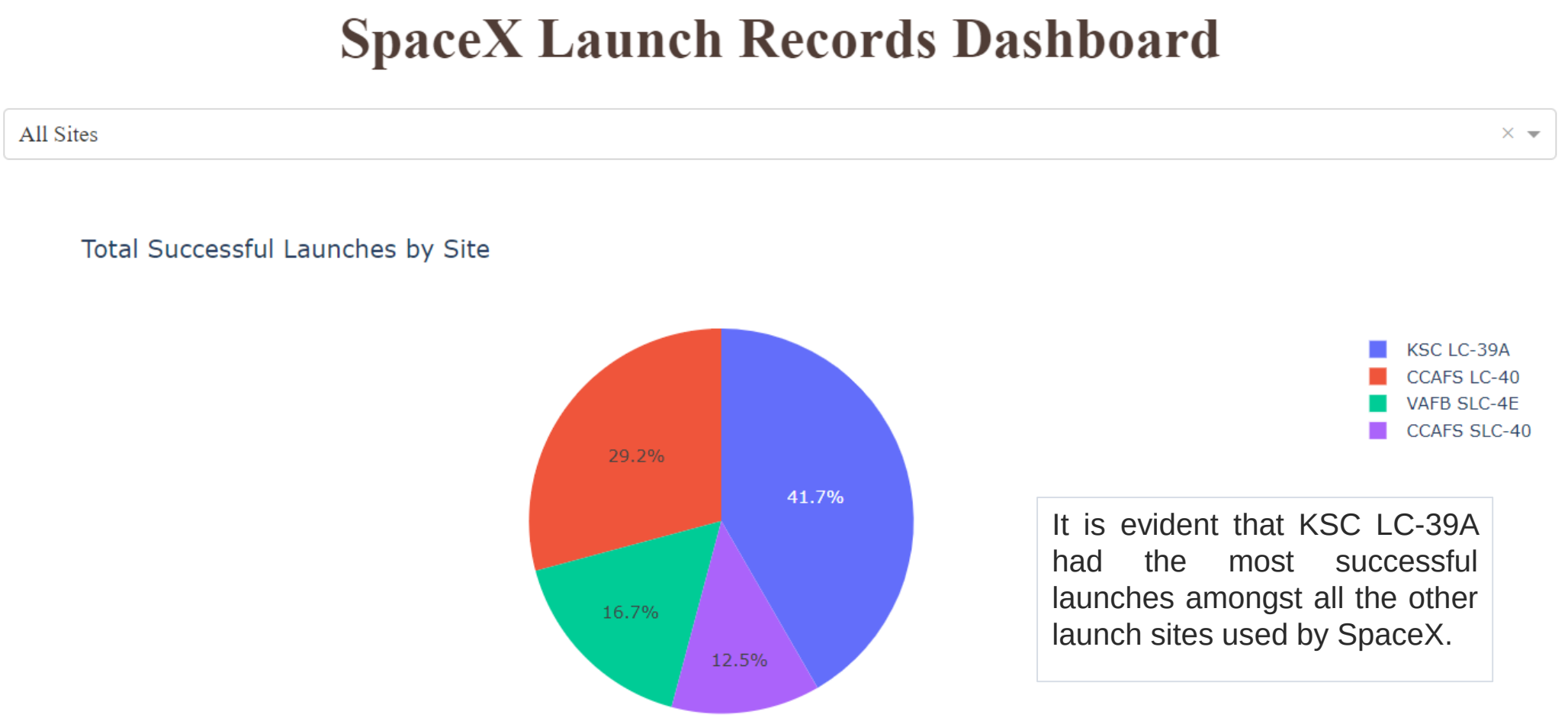
Distance to City

Distance to Railway Line

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

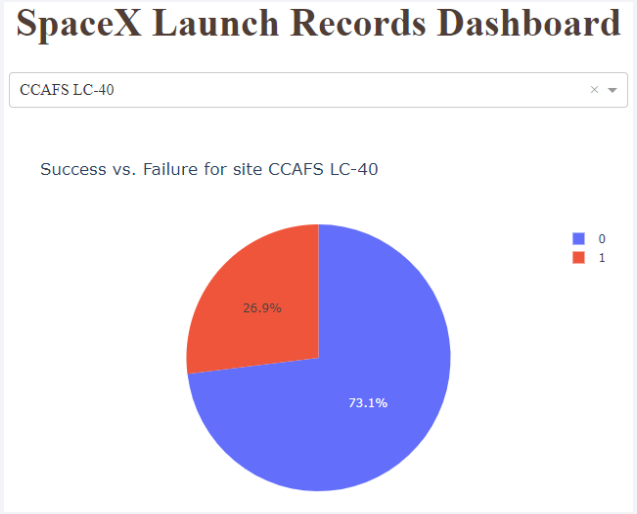

Section 4
Build a Dashboard with Plotly Dash

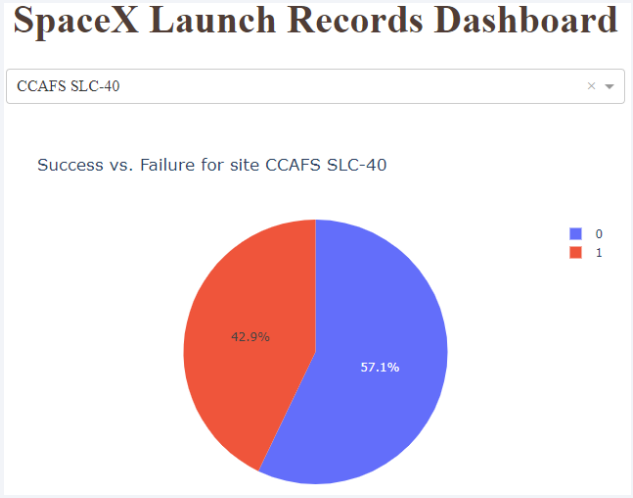
Success Percentage achieved by each Launch Site - Pie Chart

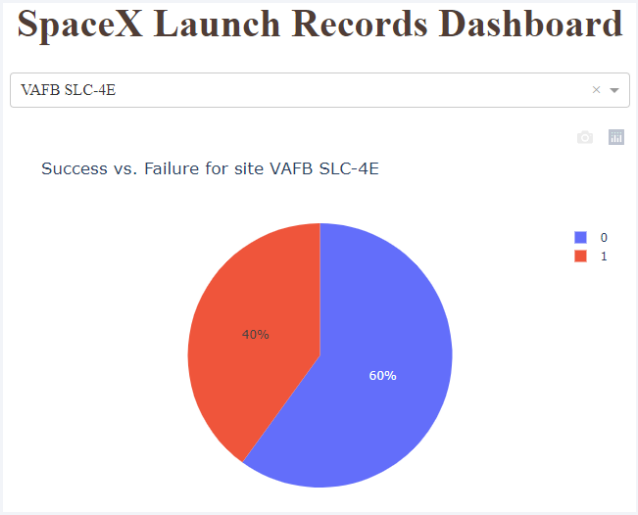


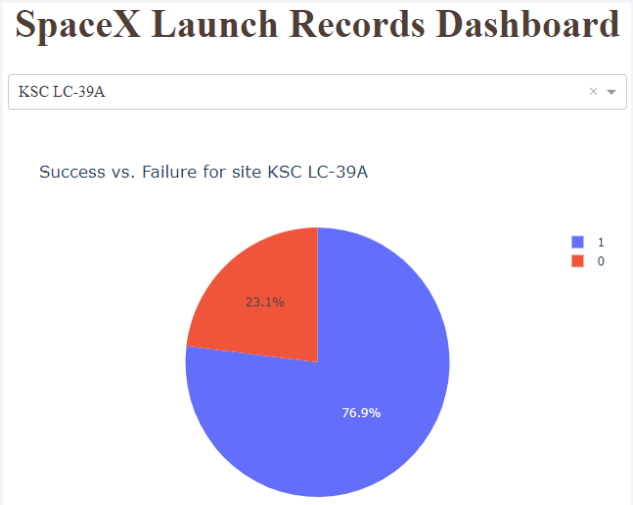
It is evident that KSC LC-39A had the most successful launches amongst all the other launch sites used by SpaceX.

Launch Site Success Ratio – Pie Chart



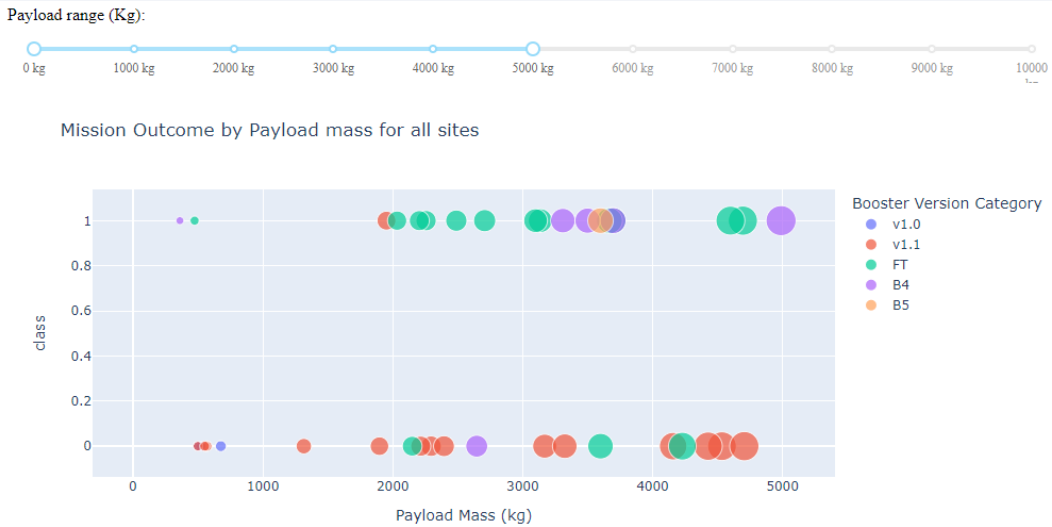






Mission Outcome by Payload Mass – Scatter Plot

Low Weighted Payload Mass: 0 kg – 5000 kg



Low Weighted Payload Mass: 5000 kg – 10000 kg

We can see the success rates for low weighted payloads is higher than the heavy weighted payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Amongst all the different models trained, Decision Tree classifier is the model with the highest classification accuracy.

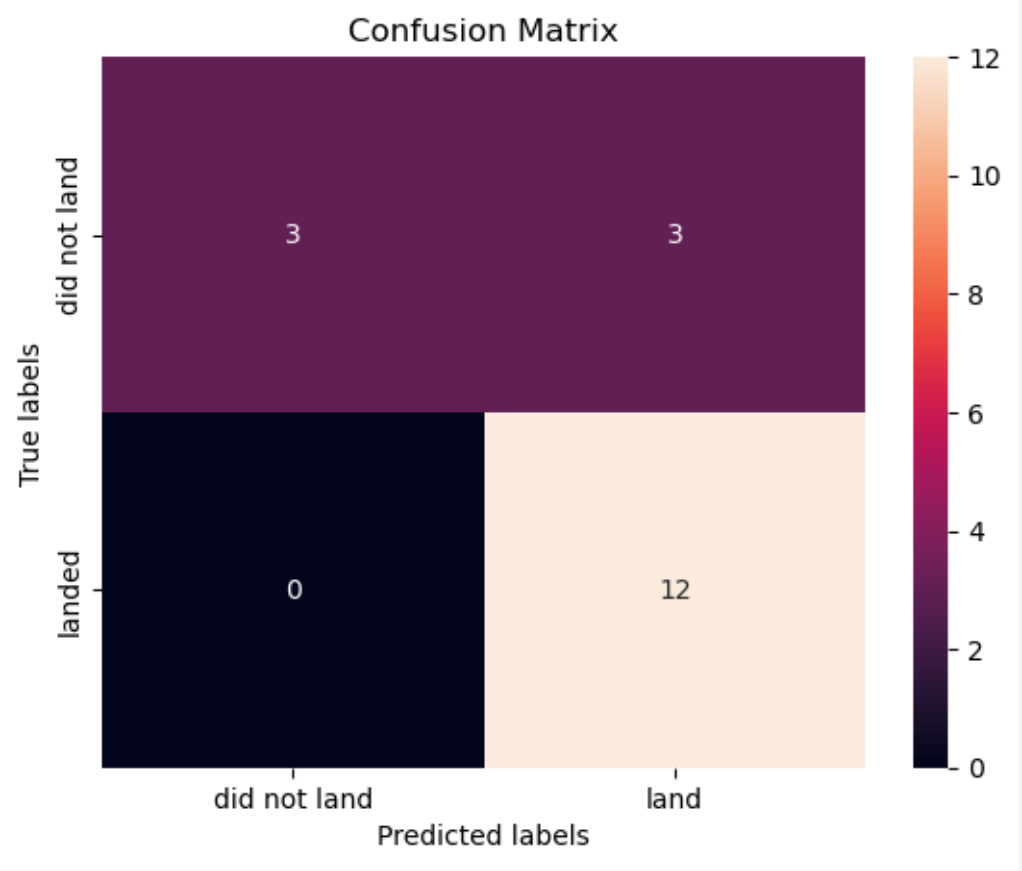
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.8892857142857145
Best params is : {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

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



Conclusions

We can conclude that:

- **Flight Number** and **Payload Mass** are important variables, but the number of launches (experience) is a stronger indicator of success over time.
- **Launch Site** plays a crucial role: certain sites like **CCAFS SLC-40** and **KSC LC-39A** are consistently more reliable.
- **Orbit Type** is a major factor in determining success, orbits **ES-L1**, **GEO**, **HEO**, **SSO**, **VLEO** had the most success rate while **LEO** and **ISS** orbits having some failures still show higher success rates, while **GTO** remains challenging.
- **Payload Mass** influences success primarily in some orbits, but in **GTO**, success is more random, possibly due to other technical factors.
- The success trend has dramatically improved over the years, demonstrating that SpaceX's technology has become far **more reliable**.

If you are competing against SpaceX, these insights suggest focusing on launches into **LEO** or **ISS** orbits with a proven track record, especially from reliable sites like **CCAFS SLC-40**. This could provide more competitive bids for specific missions where high success rates are expected.

For the purpose of our project we can conclude that the **Decision Tree** classifier is the best machine learning algorithm for this task.

Thank you!