IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Serhii Melnichenko
November 23, 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection using API
  - Data collection using web scraping
  - Data processing
  - Exploratory data analysis using SQL
  - Exploratory data analysis using data visualization
  - Interactive visual analytics using Folium
  - Predictive Analysis with Machine Learning

- Summary of all results
  - Exploratory data analysis result
  - Interactive analytics in screenshots
  - Predictive analytics result

Introduction: Background

- **Project background and context**

  In this study, I will attempt to predict whether SpaceX's Falcon 9 first stage will land successfully. SpaceX lists on its website that a Falcon 9 launch costs $62 million, while other providers charge up to $165 million, a savings that comes largely from SpaceX being able to reuse the first stage. So, if I can determine whether the first stage will land, I can determine the cost of the launch. This information can be used if another company wants to bid against SpaceX for a launch.

Introduction: Problems

Problems you want to find answers

In this project I will try to answer the following questions:

- What factors determine the success of the landing of the first stage of the Falcon 9 rocket.

- How the interaction between various characteristics determines the percentage of successful landing.

- What operating conditions must be provided for a successful landing.

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

For this project, I obtained the data as follows:

Retrieved launch data from the public REST API for SpaceX:

- Query and process SpaceX launch data using a GET request
- Filtered the data frames to only get Falcon 9 launches
- Replaced missing payload mass values from classified missions with the average value

Obtained launch data by web scraping the Wikipedia page titled "List of Falcon 9 and Falcon Heavy launches ":

- Requested the Falcon9 Launch page via its Wikipedia URL
- Extracted all column/variable names from the HTML table header
- Parsed the table and converted it to a Pandas dataframe

Methodology

Executive Summary

- Perform data wrangling

  At this stage, the collected data were analyzed, summarized and transformed.

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash

Methodology

Executive Summary

- Perform predictive analysis using classification models

  At this stage I created a machine learning pipeline:

  • Standardized the data

  • Split the data into training and testing

  • Fit the training data to different types of models:

    - Logistic regression
    - Support vector machine
    - Decision tree classifier
    - K nearest neighbors classifier

  • Used cross-validated grid search on different hyperparameters to select the best ones for each model

Data Collection

The data collection was done using different methods.

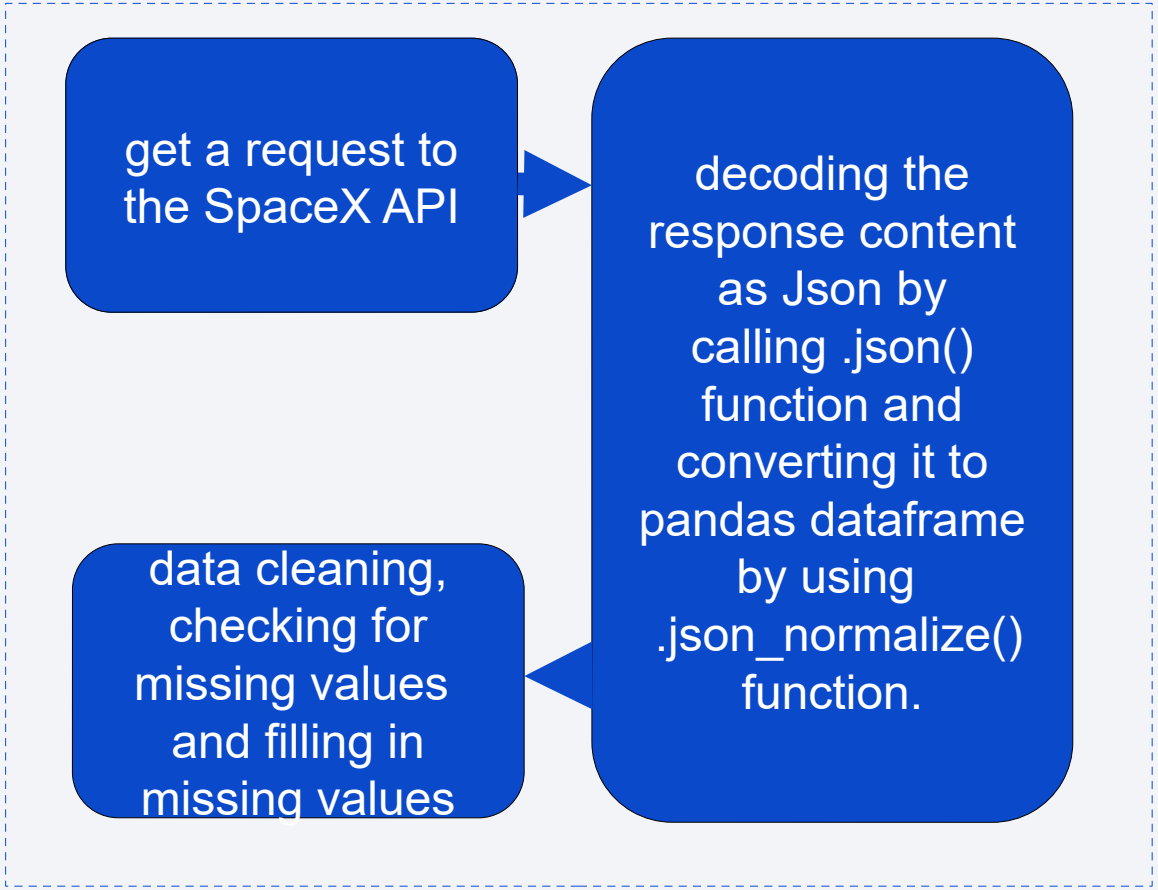
- First, I used a GET request to the SpaceX API to collect the data. Then, I decoded the response content as Json using the .json() function call and converted it into a pandas data frame using the .json_normalize() function. Then, I cleaned the data, checked for missing values, and filled in missing values with averages where necessary.

- In the second step, I web scraped the Wikipedia page about the Falcon 9 launch using BeautifulSoup. I had to extract the launch records as an HTML table, parse the table, and convert it into a pandas data frame for further analysis.

Data Collection – SpaceX API

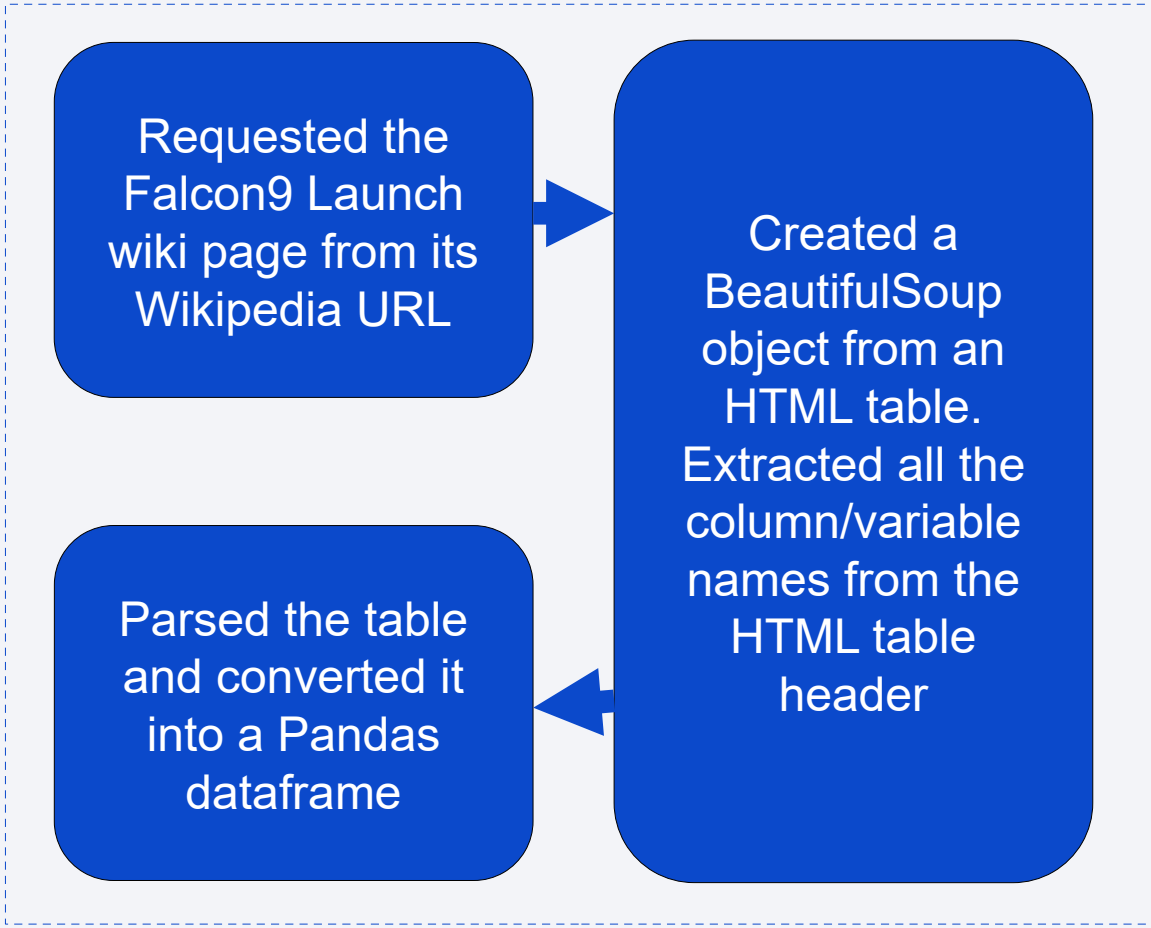
- I used SpaceX public API to collect the data, the API was used as per the flow chart and stored in a data frame.
- Link to the notebook on GitHub:

  https://github.com/Tarkas/DataScience/blob/master/Applied%20Data%20Science%20Capstone/jupyter-labs-spacex-data-collection-api.ipynb



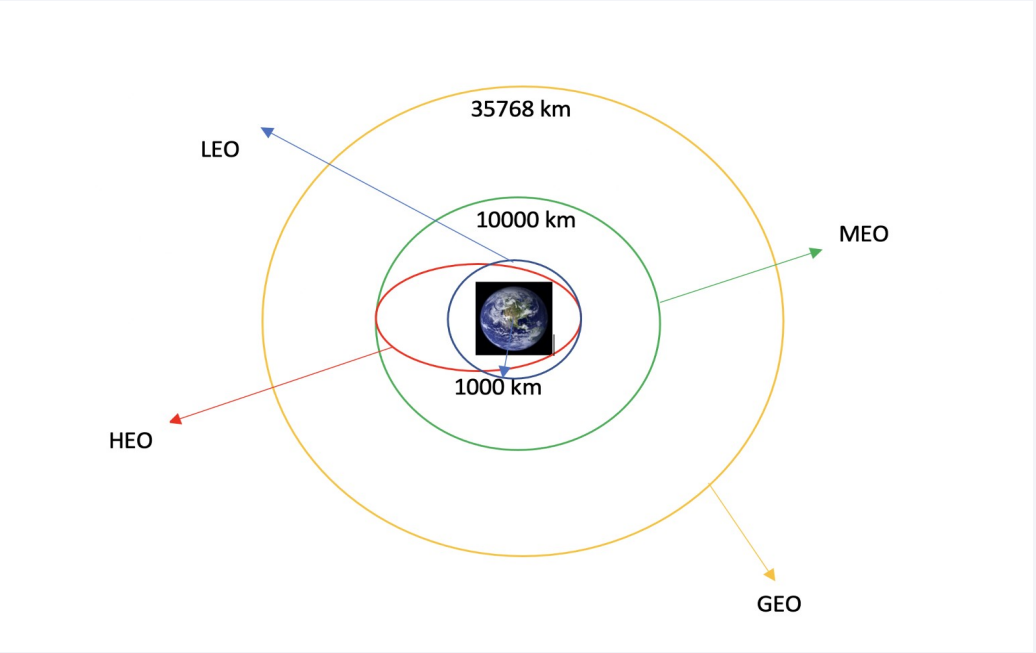
Data Collection - Scraping

- Some data about Falcon 9 launches can also be obtained from Wikipedia using BeautifulSoup. The data is downloaded from Wikipedia according to the flow chart and then saved.
- Link to the notebook on GitHub: https://github.com/Tarkas/DataScience/blob/master/Applied%20Data%20Science%20Capstone/jupyter-labs-webscraping%20(1).ipynb



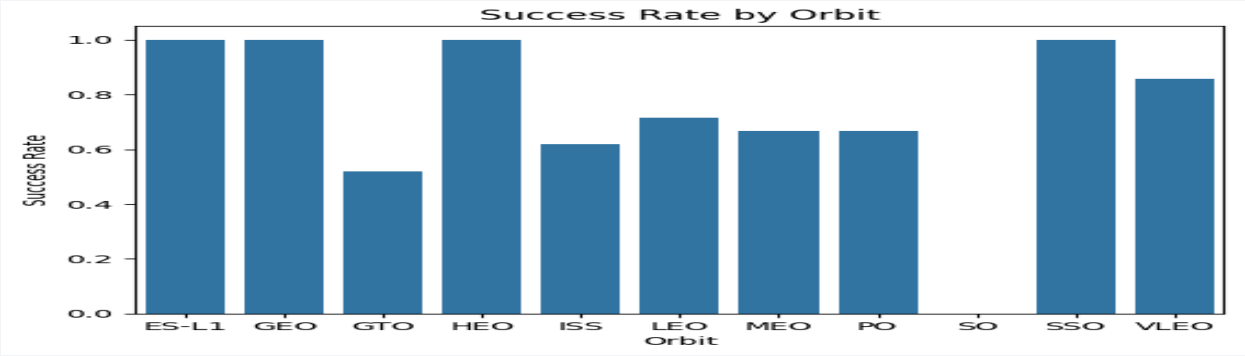
Data Wrangling

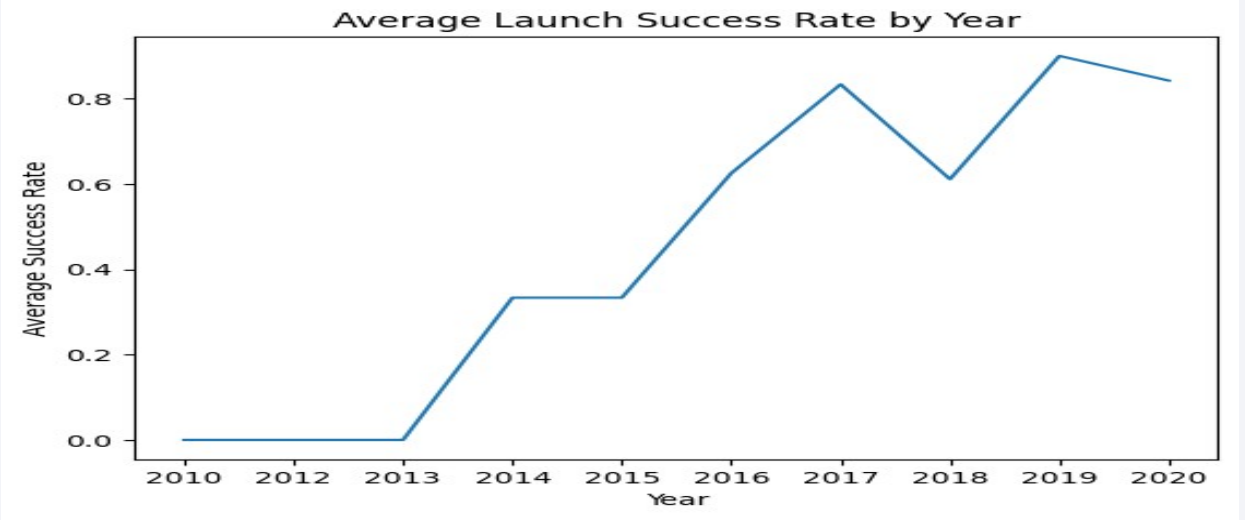
- In this part of the project I performed exploratory data analysis (EDA) to find some patterns in the data and defined labels for training supervisory models.
- Counted the number of launches at each site, as well as the number of launches per orbit and the number of launches per orbit type.
- Created a landing result label from the results column and exported the results to csv format.
- Link to the notebook on GitHub: https://github.com/Tarkas/DataScience/blob/master/Applied%20Data%20Science%20Capstone/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- In this part of the project I read the data into a Pandas data frame.
- Using the Matplotlib and Seaborn visualization libraries to plot the graphs I visualized the relationship between the parameters of the data frame.
- Link to the notebook on GitHub: https://github.com/Tarkas/Data Science/blob/master/Applied%20Data%20Science%20Capstone/edadataviz.ipynb





EDA with SQL

In this part of the project, I used SQL queries to extract and analyze data from the database. The main tasks included:

- Fetching the required data using SELECT, FROM, and WHERE commands to filter the results based on the specified criteria.
- Aggregating the data using GROUP BY, ORDER BY, and aggregate functions such as MIN(), SUM(), AVG() to obtain summary information.
- Link to the notebook on GitHub: https://github.com/Tarkas/DataScience/blob/master/Applied%20Data%20Science%20Capstone/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

	min(Date)
0	2015-12-22

	Landing_Outcome	outcome_count
0	Failure (drone ship)	5
1	Success (ground pad)	3

	Average payload mass (kg)
0	2928.4

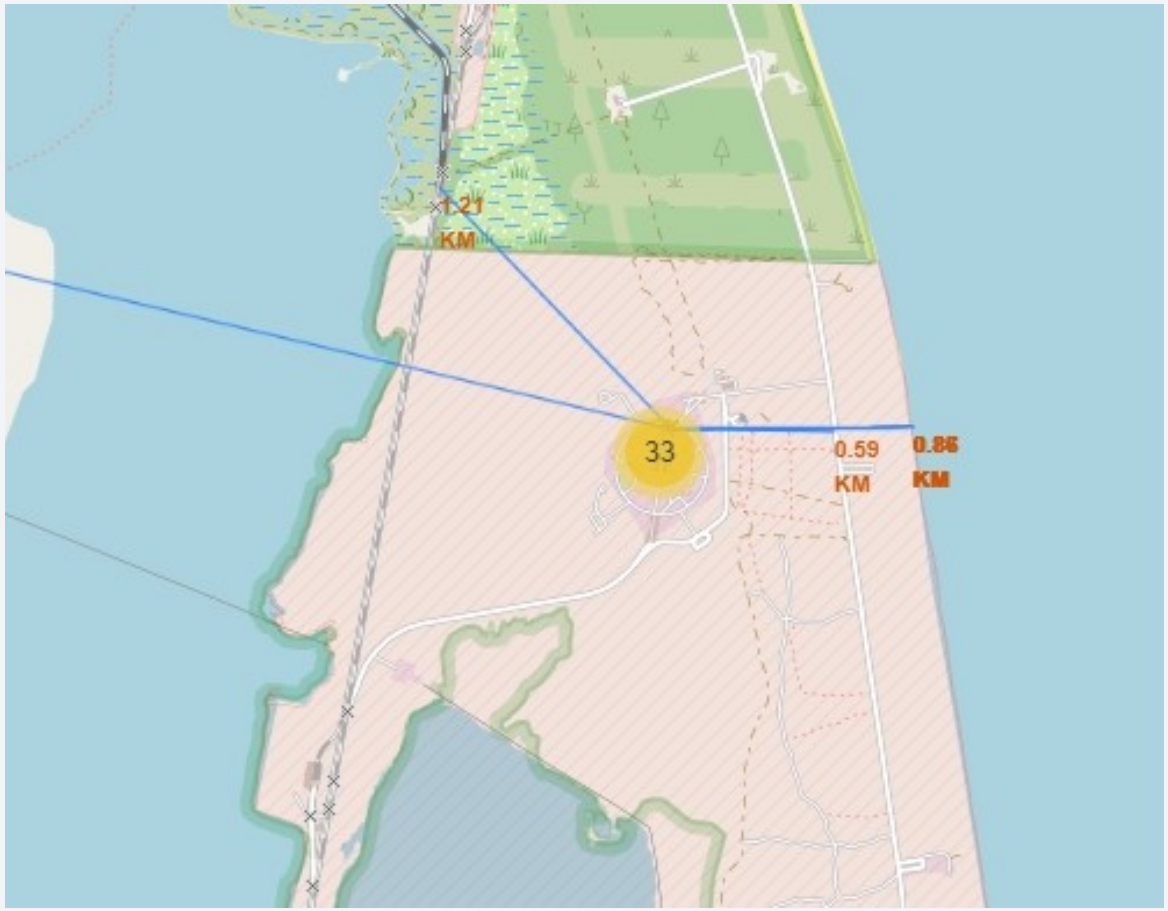
	Total payload mass (kg)
0	45596

Build an Interactive Map with Folium

In this part of the project I used circles, lines, markers and marker clusters on the Folium map. I marked all the launch sites on the map. Also, using marker clusters, I determined which launch sites were more successful. And using lines, I calculated the distances between the launch sites and the closest places to them, such as cities, coastlines, roads and railways.

Link to the notebook on GitHub:

https://github.com/Tarkas/DataScience/blob/master/Applied%20Data%20Science%20Capstone/lab-jupyter-launch-site-location-v2_1.ipynb

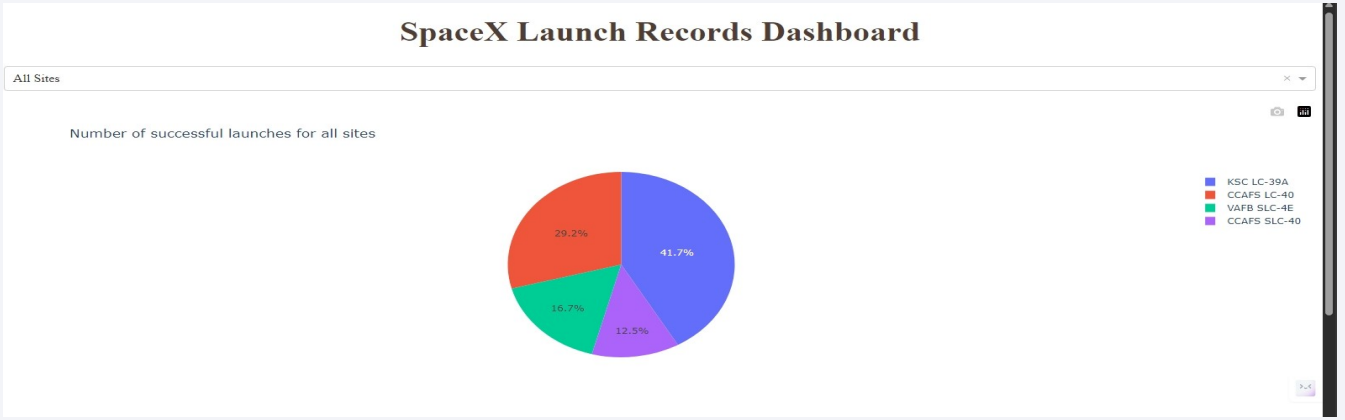


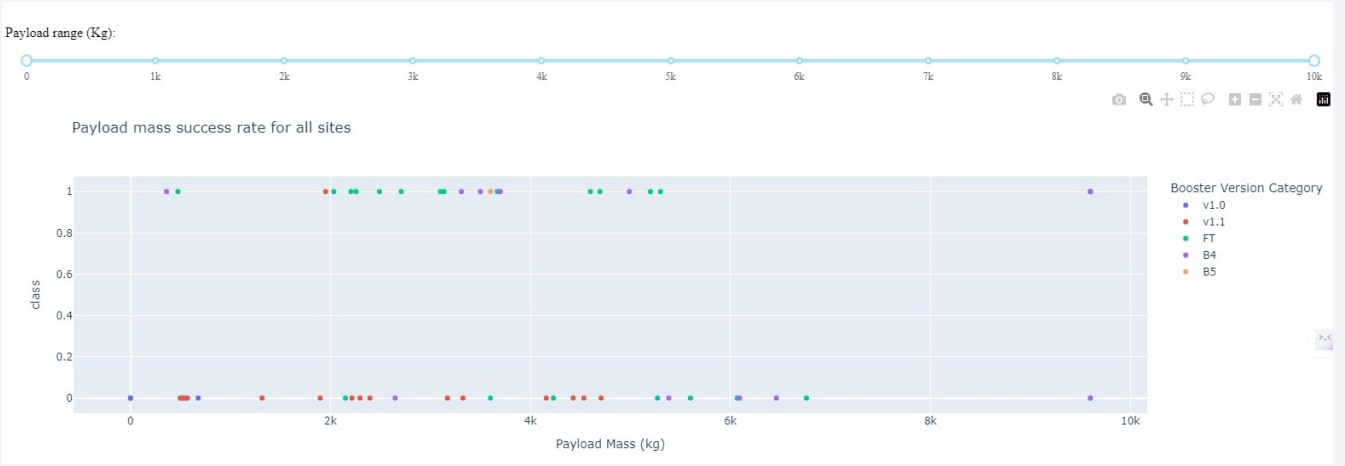
Build a Dashboard with Plotly Dash

In this part of the project, I created a Plotly Dash app that allows for interactive visual analytics of SpaceX launch data in real time.

This dashboard app contains input components such as a dropdown and a range slider to interact with the pie chart and scatter plot.

Link to the app on GitHub: https://github.com/Tarkas/DataScience/blob/master/Applied%20Data%20Science%20Capstone/spacex_dash_app.py





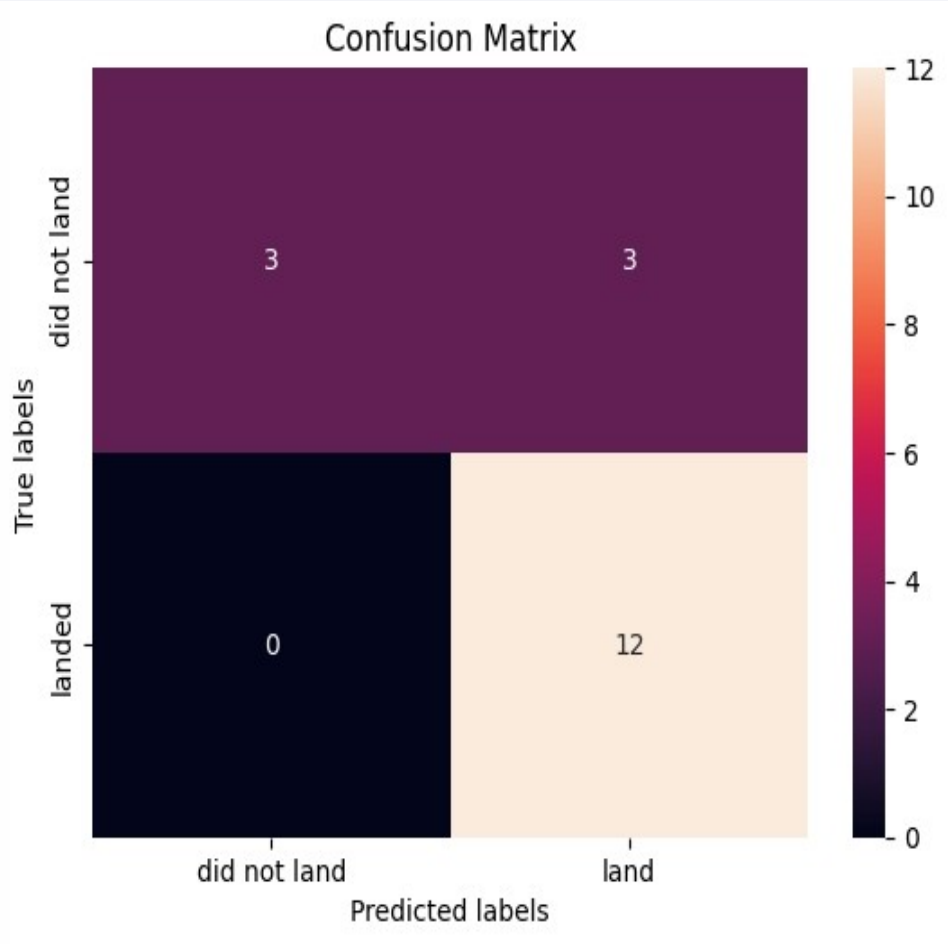
Predictive Analysis (Classification)

This part of the project developed a machine learning pipeline for landing prediction.

- **Data loading :** First, the data frames created during data collection were loaded.
- **Training label creation :** A “Class” column was added to serve as a training label and was created during data processing.
- **Data standardization :** The data was standardized to ensure that it could be used correctly in the models.
- **Data splitting :** The data was split into training and test sets to evaluate the performance of the models.
- **Fitting models :** The training data was used to fit a variety of models including:
  - Logistic regression
  - Support vector machine
  - Tree decision classifier
  - K-room neighbor classifier
- **Hyperparameter optimization :** Cross-validated grid search was used to select the best hyperparameters for each model.

This pipeline allows for efficient first stage landing prediction using state-of-the-art technology.

Link to the app on GitHub: https://github.com/Tarkas/DataScience/blob/master/Applied%20Data%20Science%20Capstone/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb



Results

- Exploratory data analysis results

  Exploratory Data Analysis (EDA) collected data using SpaceX public API and BeautifulSoup to retrieve information from Wikipedia. Performed exploratory data analysis (EDA) to identify patterns. Counted launches at each site, orbit types, and created a landing outcome label.

- Interactive analytics demo in screenshots

  Used Matplotlib and Seaborn libraries to visualize the data. Plotted circles, lines, markers, and clusters on a Folium map to display launch sites and their success rates. Developed Plotly Dash to interactively analyze SpaceX launch data.

- Predictive analysis results

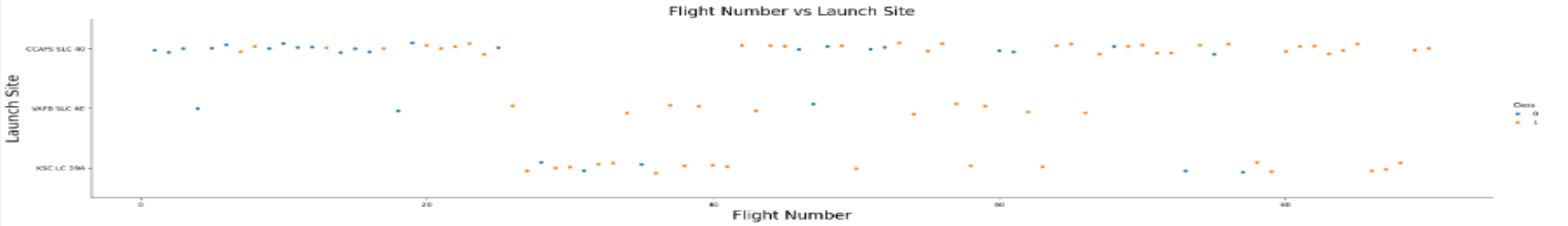
  Created a machine learning pipeline to predict landing outcomes. Standardized the data, split it into training and testing sets, and fitted different models. Optimized hyperparameters using cross-grid search to improve predictions.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

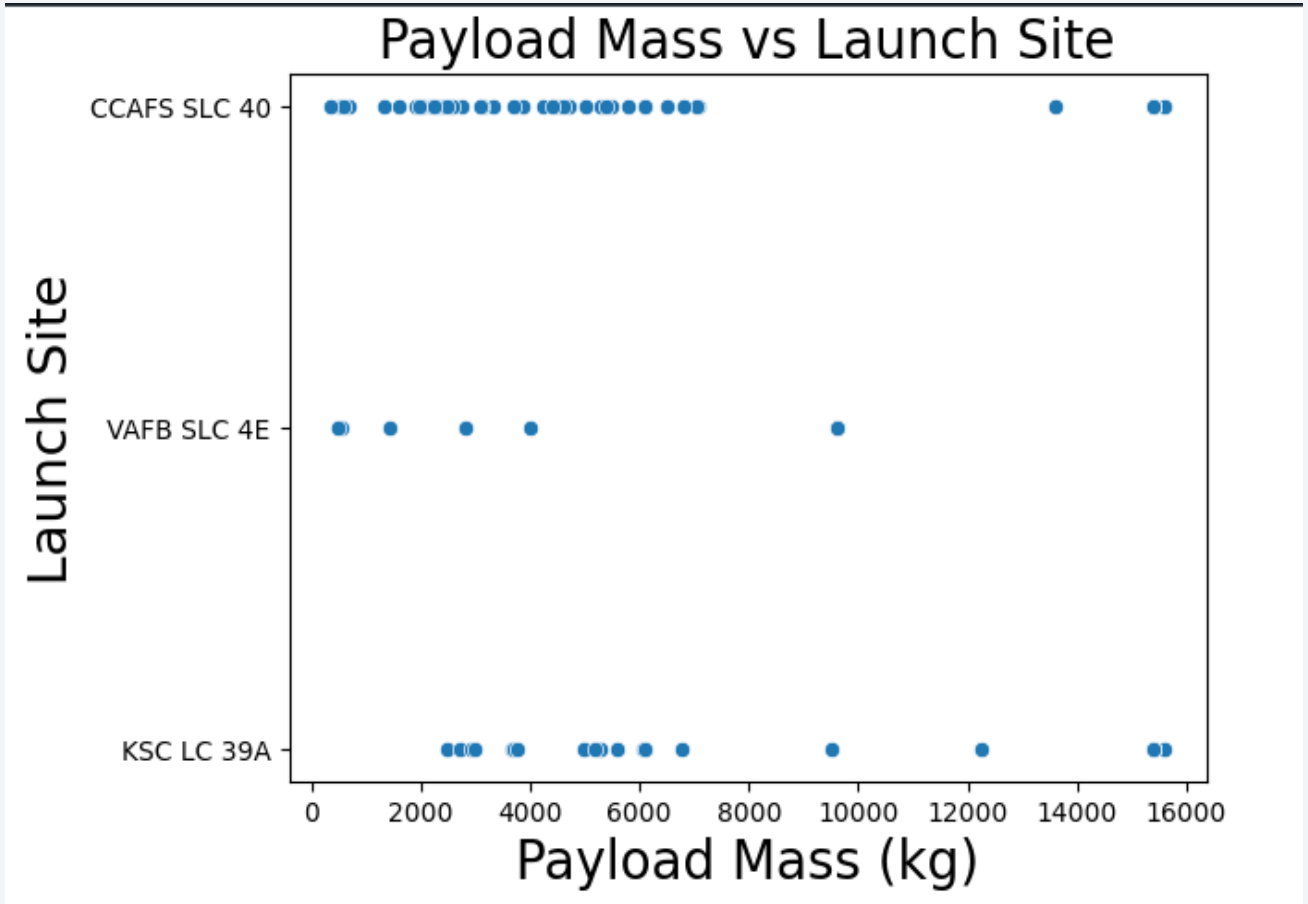
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class va
sns.catplot(x="FlightNumber", y="LaunchSite", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.title('Flight Number vs Launch Site',fontsize=20)
plt.show()
```



The Flight Number vs Launch Site plot shows that the CCAFS SLC 40 pad was where the majority of first stage landing failures occurred. Analysis of the plot also shows that as the number of launches at a launch site increases, the percentage of successful missions increases.

Payload vs. Launch Site

In a plot with PayloadMass on one axis and LaunchSite on the other, certain trends can be seen. Two launch sites, CCAFS SLC 40 and KSC LC 39A, stand out as the most favored for heavier payload launches. This means that these sites are the most likely to launch rockets carrying larger and heavier payloads.

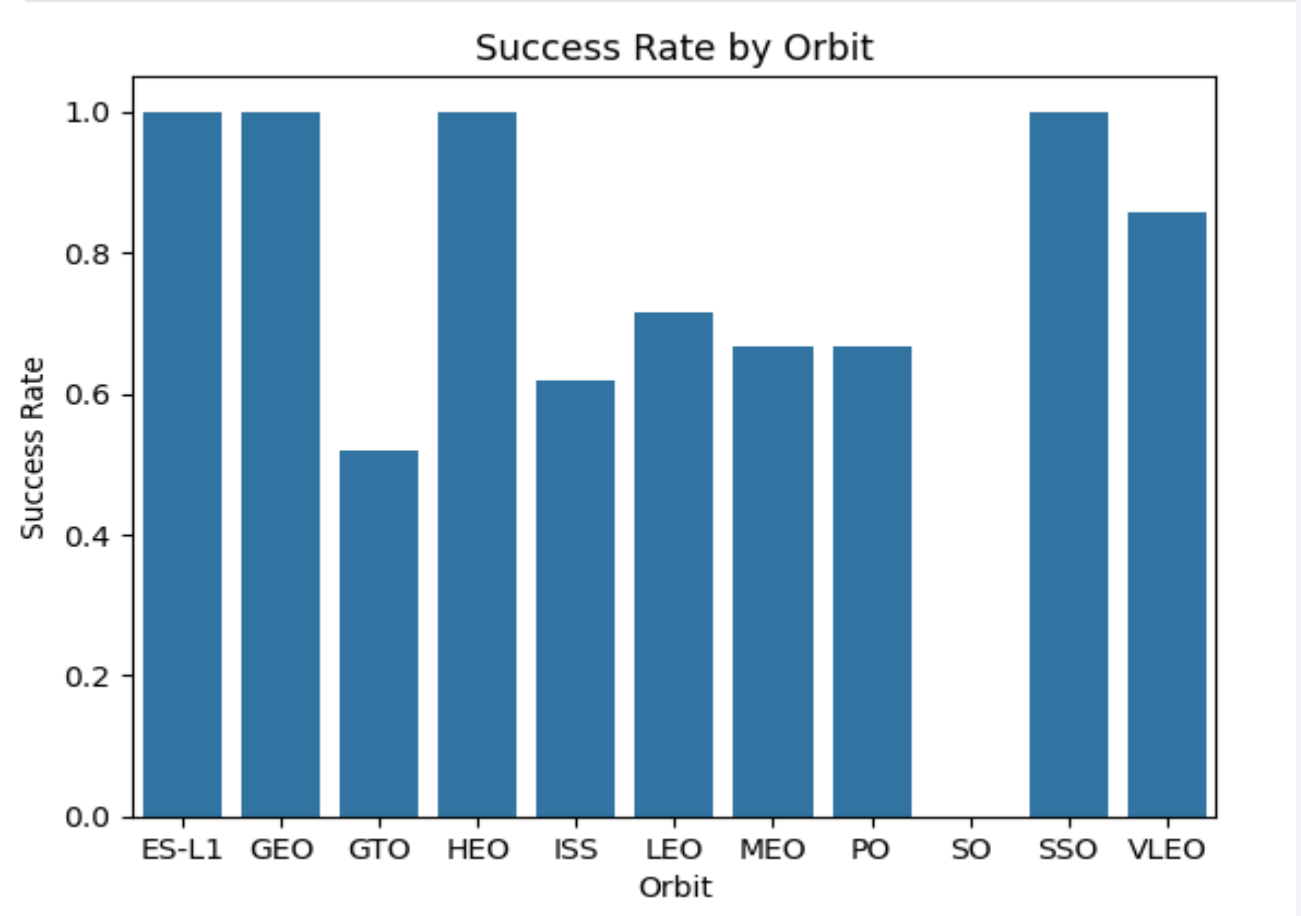


Success Rate vs. Orbit Type

From the Success Rate vs. Orbit Type plot, the following conclusions can be drawn:

- All orbit types except 'SO' had successful first stage landings.
- The orbit types with the highest success rates were ES-L1, GEO, HEO, SSO, and VLEO.

These results indicate that rockets launched into these orbits had the highest success rates, which may indicate that launches into these orbits are stable and reliable.

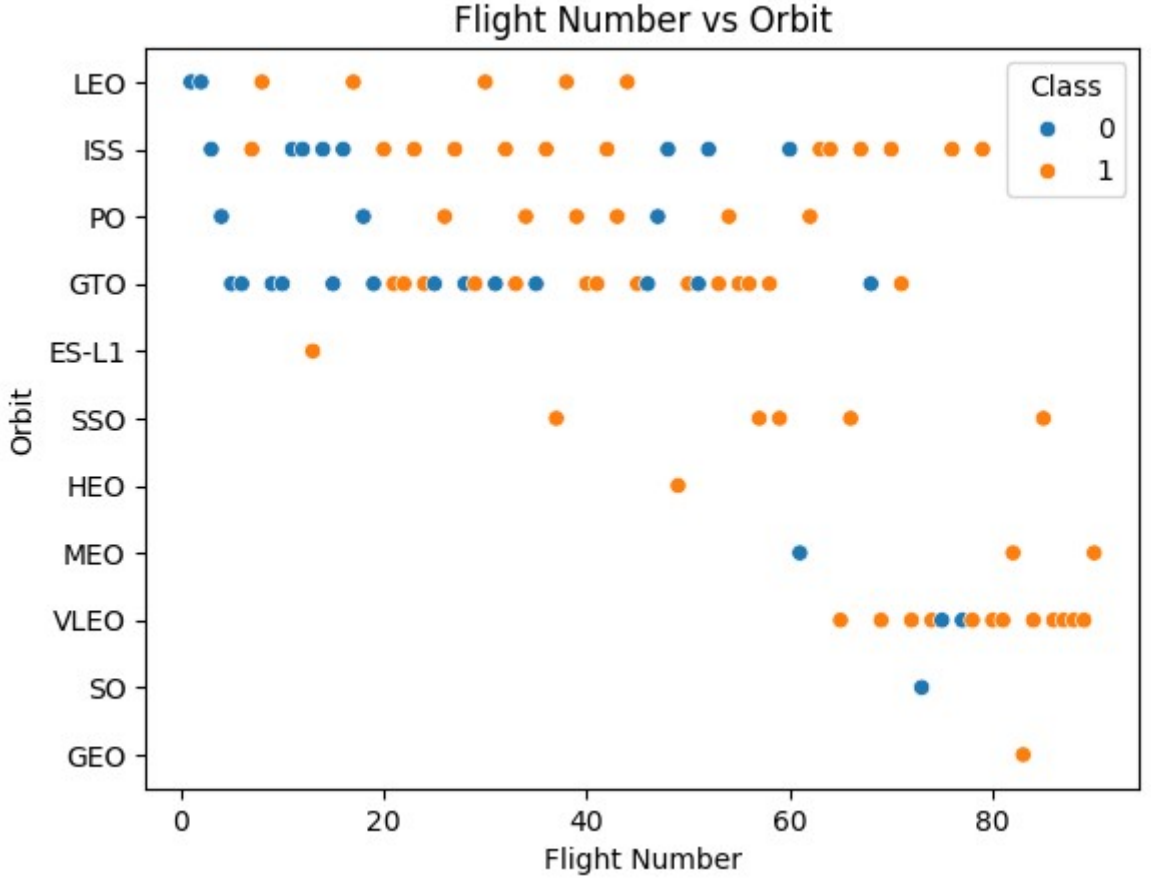


Flight Number vs. Orbit Type

The following conclusions can be drawn from the Flight Number vs Orbit Type plot:

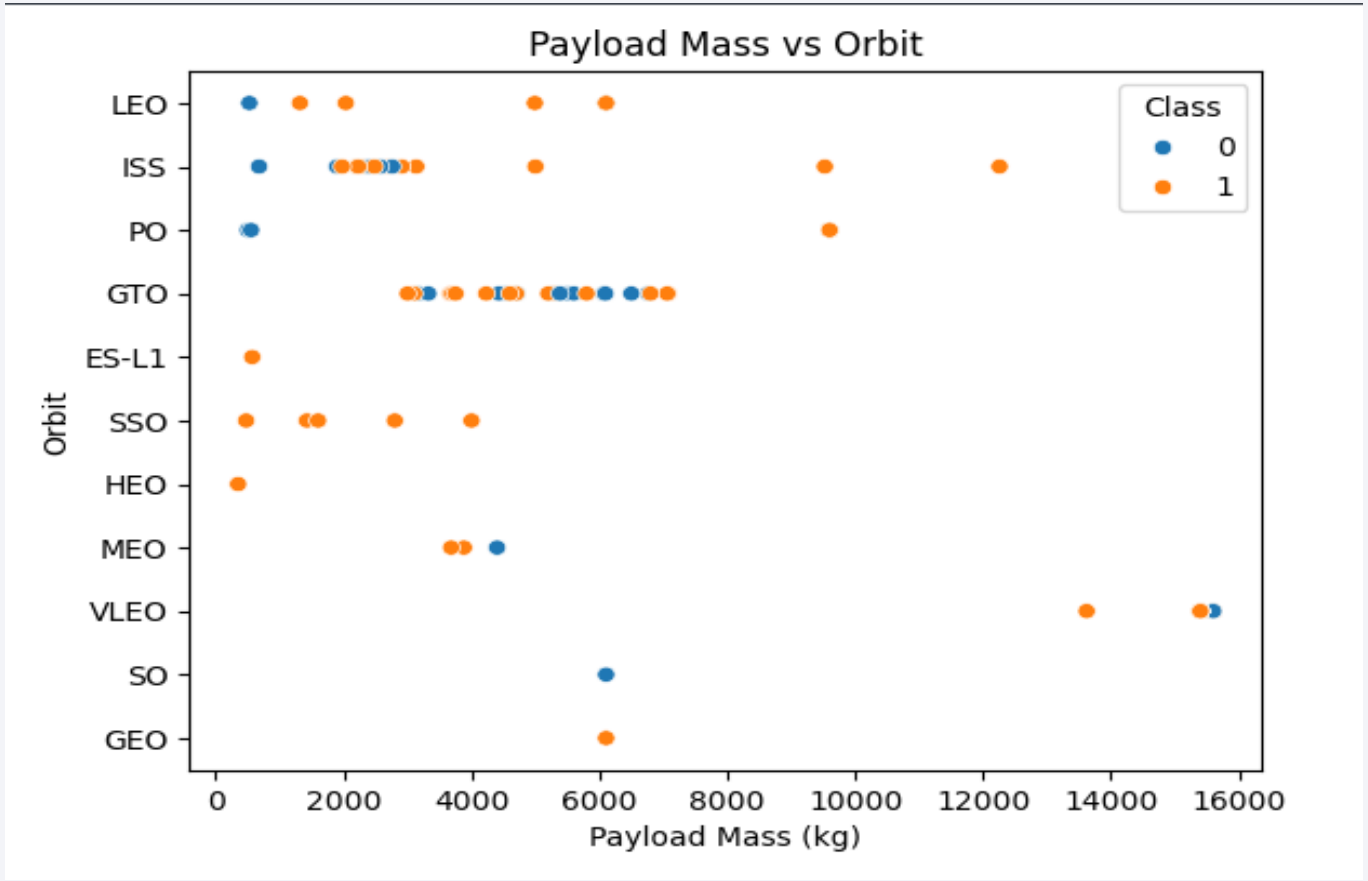
- Flight Number is positively correlated with first stage landing success for all orbit types.
- The plot shows that for LEO orbit, success is related to the number of flights: the more flights, the higher the success rate.
- No such relationship is observed for GTO orbit: the number of flights is not related to the landing success rate for this orbit.

These results indicate that for LEO orbit, accumulated flight experience contributes to improved landing success rates, while for GTO orbit, other factors may be more important.



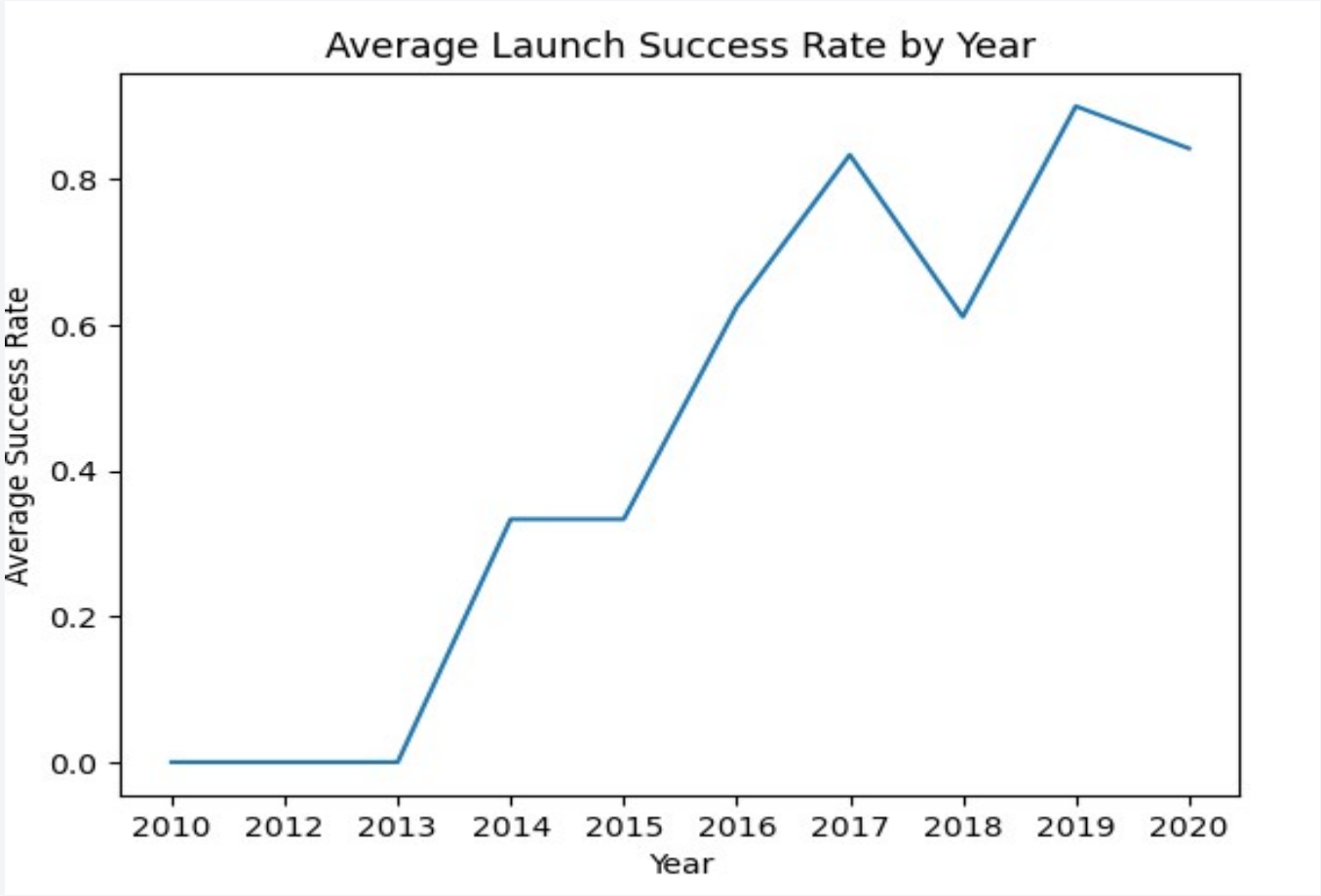
Payload vs. Orbit Type

The "PayloadMass vs. Orbit type" plot shows that heavy payloads have higher success rates in Polar, LEO, and ISS orbits. This means that heavy payloads reduce the probability of a successful landing in a GTO orbit, but increase the probability of a successful landing in an ISS orbit. However, for the GTO orbit, it is difficult to distinguish between successful and unsuccessful landings, since both outcomes are present.



Launch Success Yearly Trend

According to the "Average Launch Success Rate by Year" it is clear that starting in 2013, a positive trend in increasing launch success can be seen, which continues through 2020. This shows continuous improvements and increased launch reliability over time.



All Launch Site Names

This code executes a SQL query to retrieve unique values of the Launch_Site column from the spacextbl table in the database connected to using the con variable. The results of the query are stored in the names variable as a DataFrame using the pd.read_sql function from the Pandas library.

```
names = pd.read_sql('select distinct Launch_Site from spacextbl', con)
names
```

	Launch_Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	CCAFS SLC-40

Launch Site Names Begin with 'CCA'

```
cca = pd.read_sql("select * from spacextbl where Launch_Site like 'CCA%' limit 5", con)
cca
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcor
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachu
1	2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachu
2	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attem
3	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attem
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attem

This code executes a SQL query that retrieves all records from the spacextbl table where the values in the Launch_Site column begin with 'CCA'. The query limits the result to the first five rows.

Total Payload Mass

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

```
payload_mass = pd.read_sql("select sum(PAYLOAD_MASS__KG_) as 'Total payload mass (kg)' from spacextbl where Customer='NASA
payload_mass
```

	Total payload mass (kg)
0	45596

This code executes a SQL query to sum the total payload mass (PAYLOAD_MASS__KG_) from the spacextbl table where the customer is NASA (CRS). The query returns a result with the header Total payload mass (kg).

Average Payload Mass by F9 v1.1

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

```
average_payload = pd.read_sql("select avg(PAYLOAD_MASS__KG_) as 'Average payload mass (kg)' from spacextbl where Booster_Ve
average_payload
```

	Average payload mass (kg)
0	2928.4

This code executes a SQL query to calculate the average payload mass (PAYLOAD_MASS__KG_) from the spacextbl table, where the booster version is F9 v1.1. The query returns a result with the header Average payload mass (kg).

First Successful Ground Landing Date

```
date_first_success = pd.read_sql("select min(Date) from spacextbl where Landing_Outcome ='Success (ground pad)'", con)
date_first_success
```

	min(Date)
0	2015-12-22

This code executes a SQL query to retrieve the earliest date (min(Date)) from the spacextbl table where the landing_outcome is "Success (ground pad)".

Successful Drone Ship Landing with Payload between 4000 and 6000

```
names_boosters = pd.read_sql("select Booster_Version from spacextbl where Landing_Outcome = 'Success (drone ship)' AND Payl
names_boosters
```

	Booster_Version
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

This code executes a SQL query to retrieve the Booster_Version from the spacextbl table where the Landing_Outcome is "Success (drone ship)" and the Payload_Mass__Kg_ is in the range of 4000 to 6000 kg.

Total Number of Successful and Failure Mission Outcomes

```
total_mission = pd.read_sql("""SELECT SUM(Mission_Outcome LIKE '%Success%') AS success_mission, SUM(Mission_Outcome LIKE '%I
total_mission
```

	success_mission	failure_mission
0	100	1

This code executes a SQL query to count the number of successful and failed missions from the spacextbl table, using the LIKE condition to find rows containing the words "Success" and "Failure" in the Mission_Outcome column. The query returns two values: success_mission to count successful missions, and failure_mission to count failed missions.

Boosters Carried Maximum Payload

This code executes a SQL query to retrieve the booster version (booster_version) from the spacextbl table where the payload mass (PAYLOAD_MASS__KG_) is maximum. The query uses a subquery to find the maximum payload mass and selects all records that match this value.

```
list_names_max_mass = pd.read_sql(""" SELECT booster_version FROM spacextbl WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MA
list_names_max_mass
```

	Booster_Version
0	F9 B5 B1048.4
1	F9 B5 B1049.4
2	F9 B5 B1051.3
3	F9 B5 B1056.4
4	F9 B5 B1048.5
5	F9 B5 B1051.4
6	F9 B5 B1049.5
7	F9 B5 B1060.2
8	F9 B5 B1058.3
9	F9 B5 B1051.6
10	F9 B5 B1060.3
11	F9 B5 B1049.7

2015 Launch Records

```
list_failure_drone_ship_2015 = pd.read_sql(""" SELECT substr(Date, 6, 2) AS month, booster_version, launch_site, landing_ou
list_failure_drone_ship_2015
```

	month	Booster_Version	Launch_Site	Landing_Outcome
0	01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

This code executes a SQL query to retrieve data about drone ship landing failures in 2015. The query selects the month (using the substr function to extract a substring from the Date column), booster version (booster_version), launch_site (launch_site), and landing_outcome (landing_outcome). The LIKE '%Failure (drone ship)%' condition is used to filter for drone ship landing failures, and the substr(Date, 0, 5) = '2015' condition is used to select data for 2015.

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

```
list_landing_outcomes = pd.read_sql(""" SELECT landing_outcome, COUNT(*) AS outcome_count FROM spacextbl WHERE DATE BETWEEN
list_landing_outcomes
```

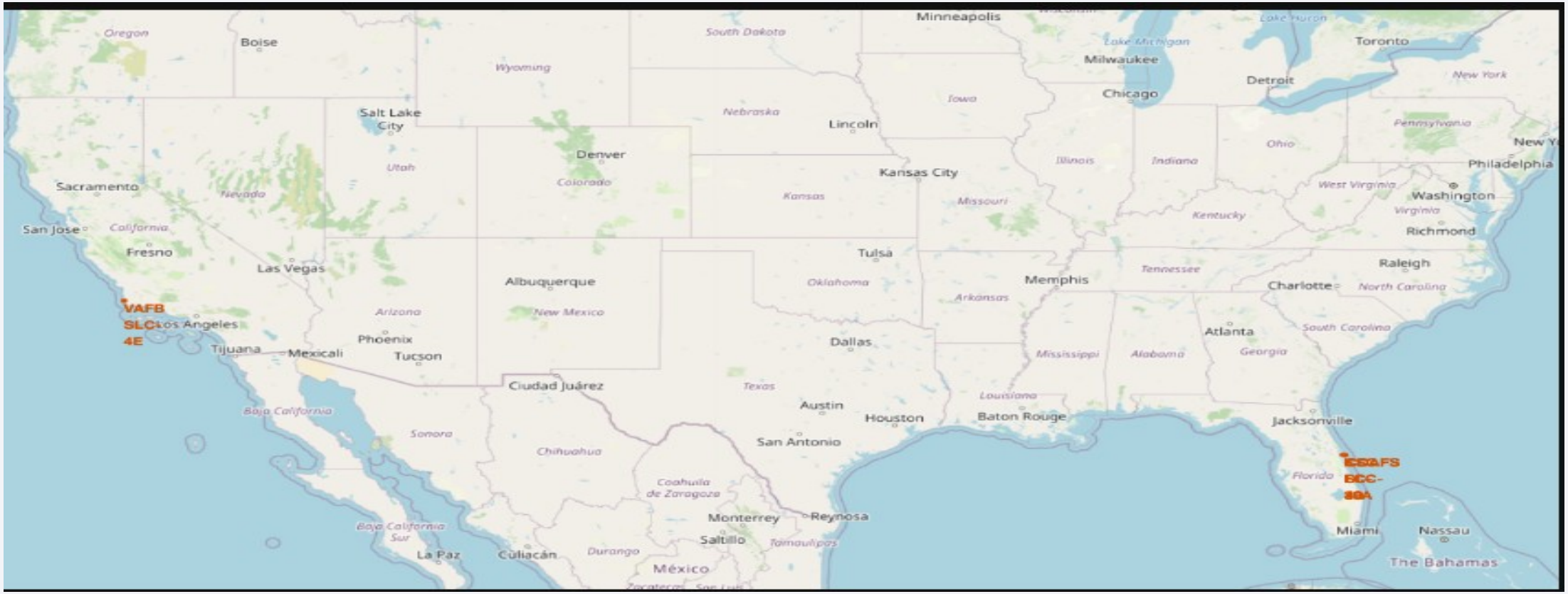
	Landing_Outcome	outcome_count
0	Failure (drone ship)	5
1	Success (ground pad)	3

This code executes a SQL query to count the number of distinct landing_outcomes from the spacextbl table for the period from June 4, 2010 to March 20, 2017. The query filters the data to only consider two types of outcomes: "Failure (drone ship)" and "Success (ground pad)". The results are grouped by outcome type and sorted by outcome_count in descending order.

Section 3

Launch Sites Proximities Analysis

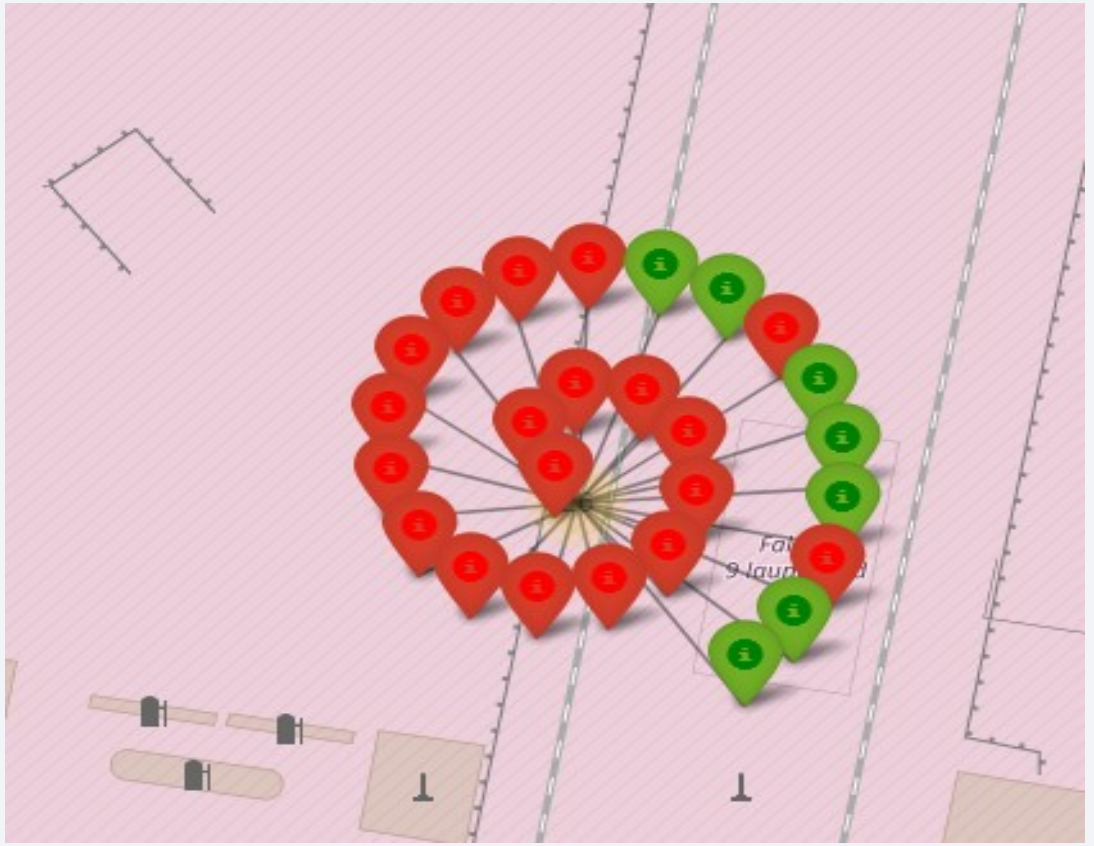
LAUNCH SITE LOCATION



Mapping out these locations underscores the significance of their coastal and equatorial positions. All SpaceX launch sites are situated along the US coastline, specifically in Florida and California.

COLOR-LABELED LAUNCH OUTCOMES

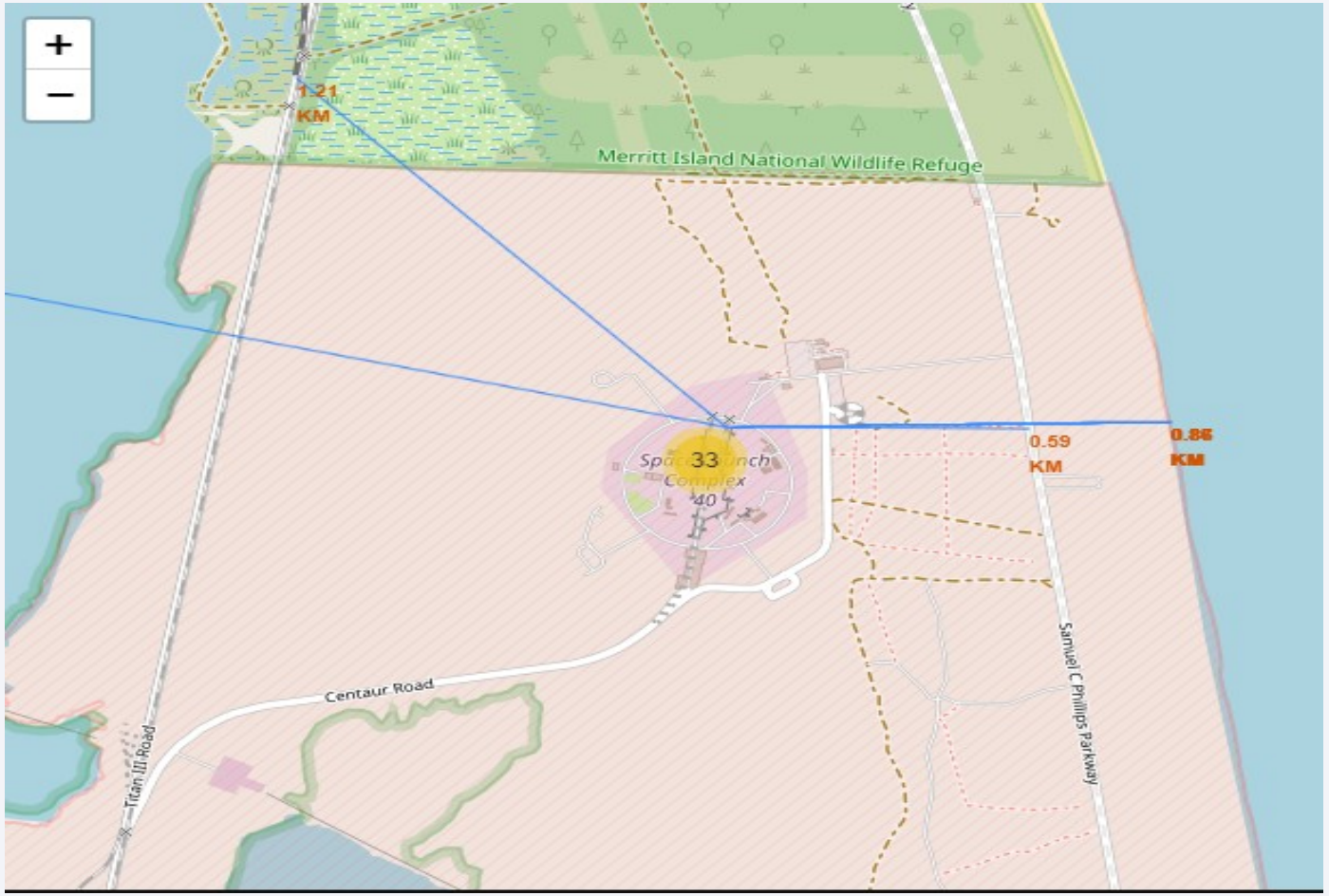
Visualization of launch vehicle landing results on launch pads uses green markers to indicate successful landings and red markers for unsuccessful ones.

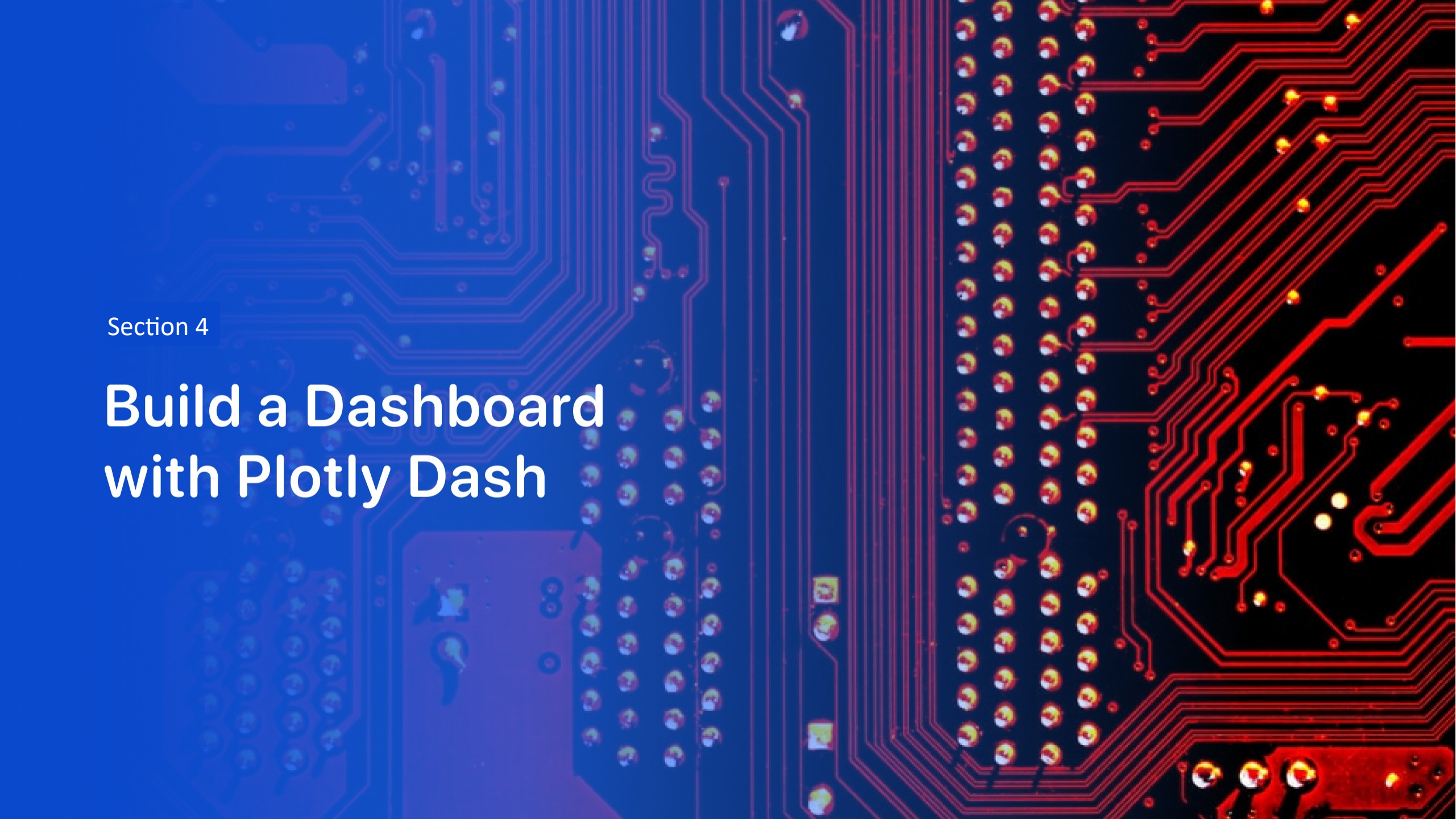


LOCATION OF LAUNCH PADS

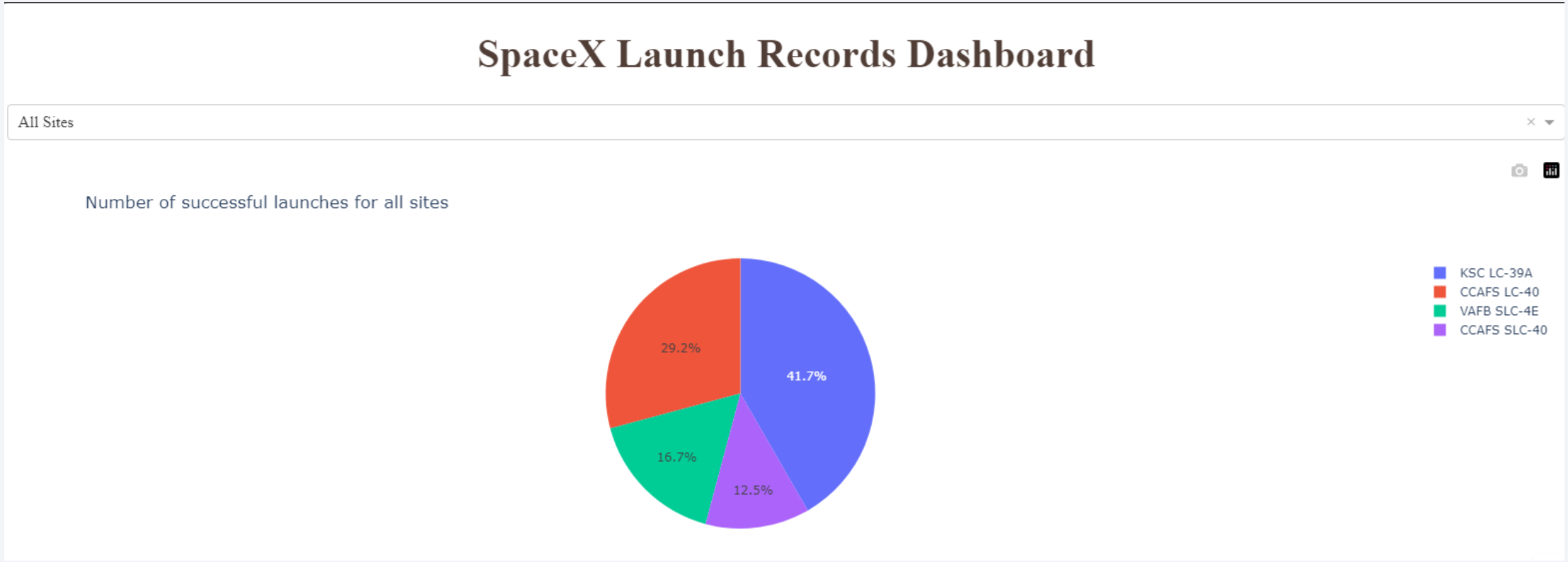
Visualization of the railroad, highway, coastline, and city proximity to each launch site shows how close they are. For example, to CCAFS SLC-40:

- Railroad: 1.21 km
- Road: 0.59 km
- Coastline: 0.88 km
- City: 23.45 km



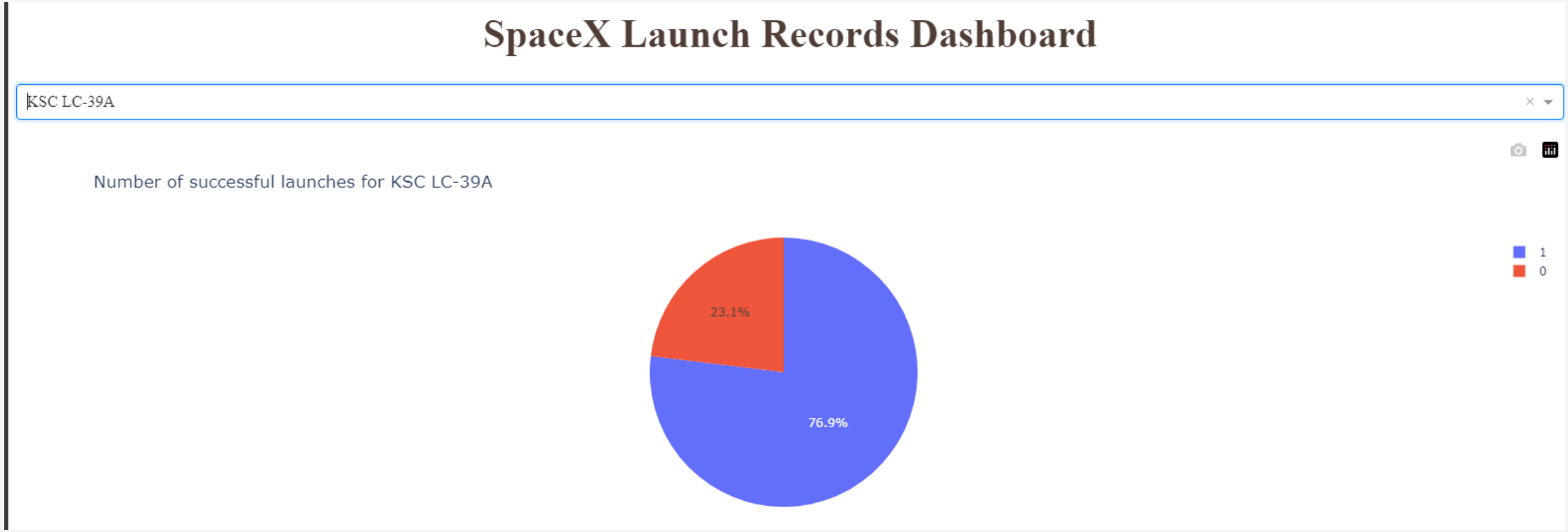

Section 4
Build a Dashboard with Plotly Dash

SUCCESSFUL LAUNCHES AT EACH LAUNCH SITE



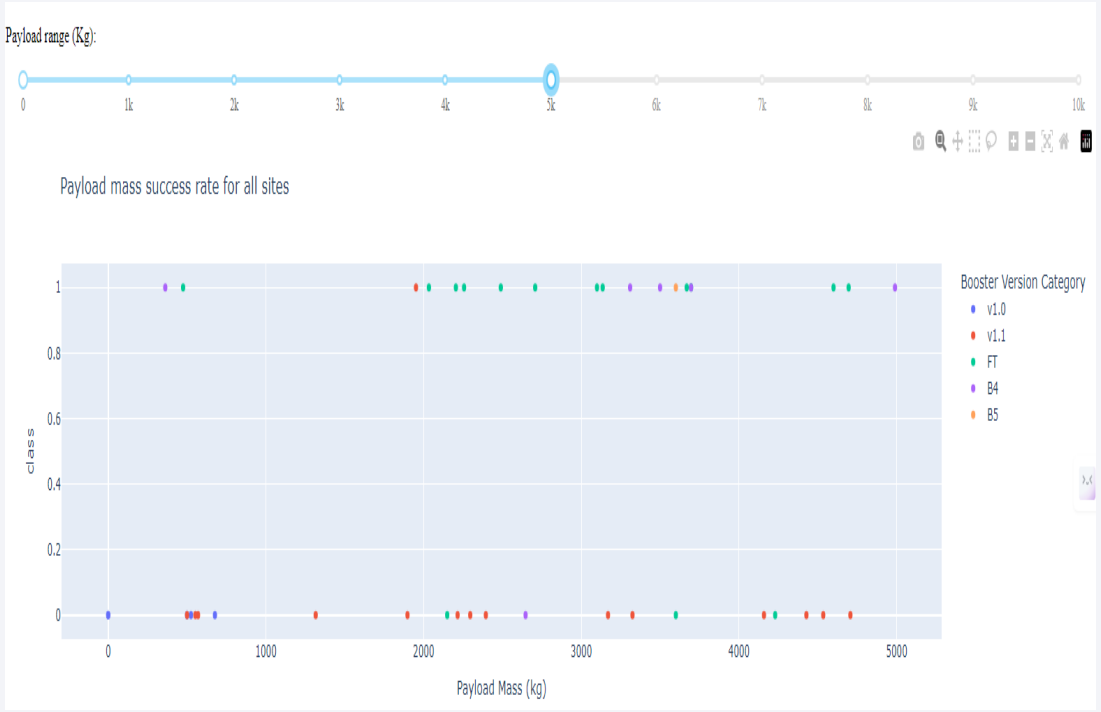
A pie chart showing the percentage of successful launches at each launch site.

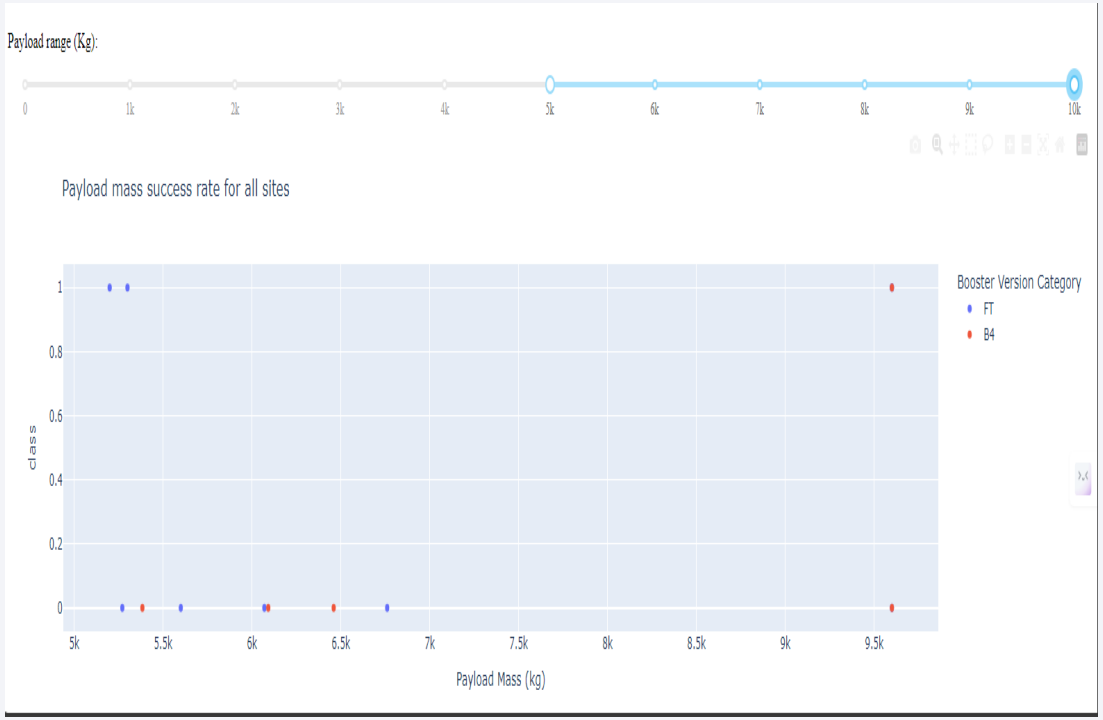
HIGHEST LAUNCH SUCCESS RATE



Screenshot of pie chart for launch site with highest launch success rate KSC LC-39A

LOADS WITH A MASS





Loads with a mass of less than 5000 kg demonstrated the highest success rate of landing the first stage of the rocket. While loads over 5000 kg demonstrated the lowest success rate of landing the first stage of the rocket.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

}

best_algorithm = max(models, key=models.get)
best_score = models[best_algorithm]

print(f'Best model is {best_algorithm} with a score of {best_score}')

best_params = {
        'LogisticRegression': logreg_cv.best_params_,
        'SupportVector': svm_cv.best_params_,
    'DecisionTree': tree_cv.best_params_,
    'KNeighbors': knn_cv.best_params_

}
print(f"Best parameters for {best_algorithm}: {best_params[best_algorithm]}")
```

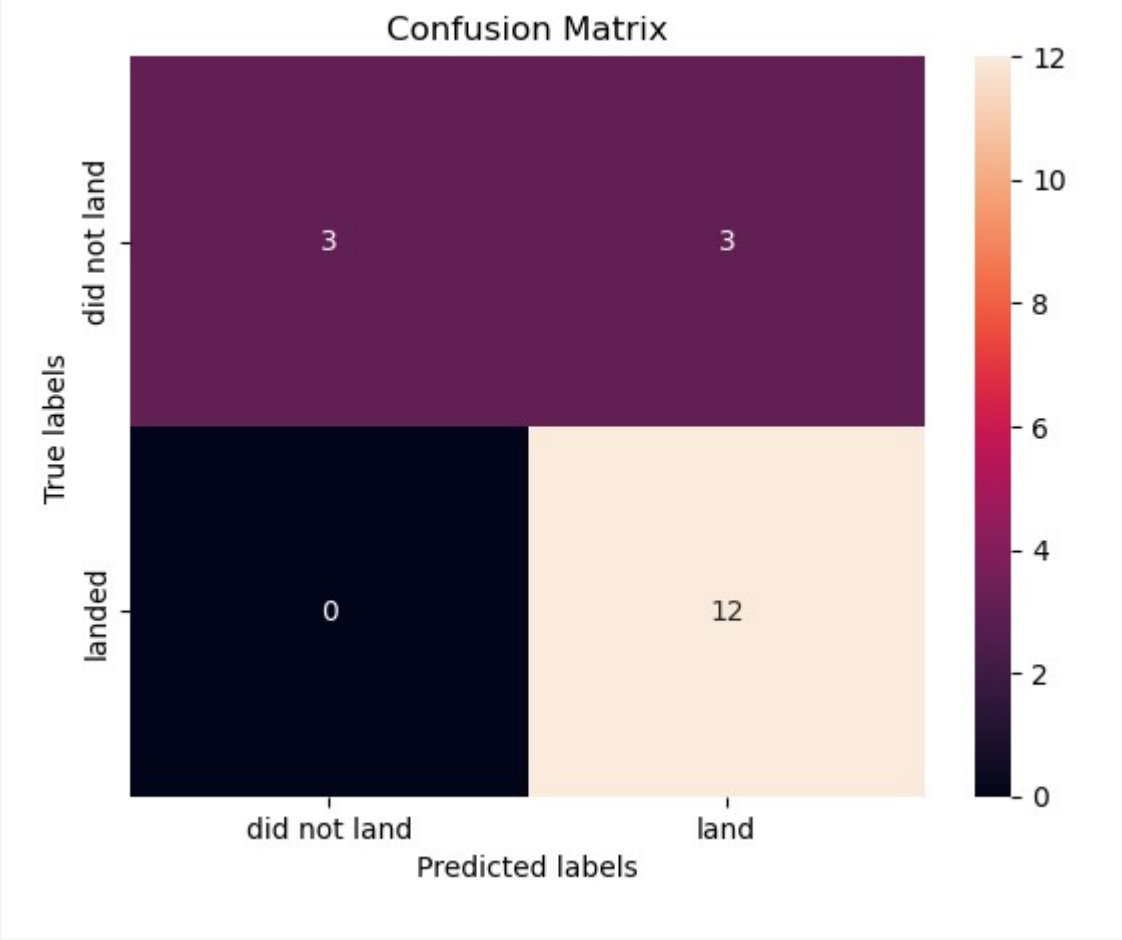
```
Best model is DecisionTree with a score of 0.8875
Best parameters for DecisionTree: {'criterion': 'gini', 'max_depth': 18, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_
samples_split': 10, 'splitter': 'random'}
```

The decision tree classifier demonstrated the highest classification accuracy with a score of 0.8875. The optimal parameters for the model include the use of the gini criterion, a maximum tree depth of 18, the selection of the maximum number of features by square root, a minimum number of samples per leaf of 2, a minimum number of samples for splitting of 10, and a random splitting method.

Confusion Matrix

The confusion matrix of the decision tree classifier demonstrates its ability to discriminate between classes, but the main problem is false positives, where failed landings are incorrectly classified as successful. For the best-performing models (4-way linkage), the confusion matrices remain the same.

The key problem is false positives, with the models incorrectly predicting the landing of the first stage booster in 3 out of 18 cases in the test set.



Conclusions

- Impact of Number of Flights: There is a direct correlation between the number of launches at a launch site and the percentage of successful landings. The more flights are carried out at a particular site, the higher the probability of successful completion of missions.
- Success Rate Growth: The launch success rate has shown a steady increase from 2013 to 2020, which can be attributed to improvements in launch technology and preparation methods.
- Successful Orbits: The highest percentage of successful launches is observed in ES-L1, GEO, HEO, SSO, and VLEO orbits, indicating that these orbits are preferred for successful missions.
- Leader in Launch Success: KSC LC-39A has become the site of the highest number of successful launches, highlighting its importance in the space program.
- Algorithm Performance: The Decision Tree classifier was found to be the best machine learning algorithm for analyzing launch data, confirming its high accuracy in classification.
- Predicting Successful Landings: Using the models presented in the project, SpaceX can predict the successful landing of a SpaceX rocket's first stage with 83.3% accuracy, opening up new possibilities for launch planning and optimization.

Appendix

Data:

REST API for SpaceX - https://github.com/r-spacex/SpaceX-API

Wikipedia page - https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

Notebooks:

https://github.com/Tarkas/DataScience/blob/master/Applied Data Science Capstone/jupyter-labs-spacex-data-collection-api.ipynb

https://github.com/Tarkas/DataScience/blob/master/Applied Data Science Capstone/jupyter-labs-webscraping(1).ipynb

https://github.com/Tarkas/DataScience/blob/master/Applied Data Science Capstone/labs-jupyter-spacex-Data wrangling.ipynb

https://github.com/Tarkas/DataScience/blob/master/Applied Data Science Capstone/edadataviz.ipynb

https://github.com/Tarkas/DataScience/blob/master/Applied Data Science Capstone/jupyter-labs-eda-sql-coursera_sqllite(1).ipynb

https://github.com/Tarkas/DataScience/blob/master/Applied Data Science Capstone/lab-jupyter-launch-site-location-v2_1.ipynb

https://github.com/Tarkas/DataScience/blob/master/Applied Data Science Capstone/spacex_dash_app.py

https://github.com/Tarkas/DataScience/blob/master/Applied Data Science Capstone/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb

Thank you!